Error-correcting codes and cryptology

Ruud Pellikaan \textsuperscript{1},
Xin-Wen Wu \textsuperscript{2},
Stanislav Bulygin \textsuperscript{3} and
Relinde Jurrius \textsuperscript{4}

PRELIMINARY VERSION
23 January 2012

All rights reserved.
To be published by Cambridge University Press.
No part of this manuscript is to be reproduced
without written consent of the authors and the publisher.

\textsuperscript{1}g.r.pellikaan@tue.nl, Department of Mathematics and Computing Science, Eindhoven University of Technology, P.O. Box 513, NL-5600 MB Eindhoven, The Netherlands
\textsuperscript{2}x.wu@griffith.edu.au, School of Information and Communication Technology, Griffith University, Gold Coast, QLD 4222, Australia
\textsuperscript{3}Stanislav.Bulygin@cased.de, Department of Mathematics, Technische Universität Darmstadt, Mornewegstrasse 32, 64293 Darmstadt, Germany
\textsuperscript{4}r.p.m.j.jurrius@tue.nl, Department of Mathematics and Computing Science, Eindhoven University of Technology, P.O. Box 513, NL-5600 MB Eindhoven, The Netherlands
# Contents

1 Introduction  
1.1 Notes ................................. 11

2 Error-correcting codes  
2.1 Block codes .............................. 14  
2.1.1 Repetition, product and Hamming codes .......... 15  
2.1.2 Codes and Hamming distance .................. 18  
2.1.3 Exercises ................................ 21  
2.2 Linear Codes .............................. 21  
2.2.1 Linear codes .......................... 21  
2.2.2 Generator matrix and systematic encoding .......... 22  
2.2.3 Exercises ................................ 25  
2.3 Parity checks and dual code .................... 26  
2.3.1 Parity check matrix ..................... 26  
2.3.2 Hamming and simplex codes ................... 28  
2.3.3 Inner product and dual codes ................. 30  
2.3.4 Exercises ................................ 32  
2.4 Decoding and the error probability ............... 33  
2.4.1 Decoding problem ........................ 34  
2.4.2 Symmetric channel ........................ 35  
2.4.3 Exercises ................................ 38  
2.5 Equivalent codes .......................... 38  
2.5.1 Number of generator matrices and codes ......... 38  
2.5.2 Isometries and equivalent codes ................ 40  
2.5.3 Exercises ................................ 44  
2.6 Notes .................................... 45

3 Code constructions and bounds .................. 47  
3.1 Code constructions .......................... 47  
3.1.1 Constructing shorter and longer codes .......... 47  
3.1.2 Product codes .......................... 52  
3.1.3 Several sum constructions ................... 55  
3.1.4 Concatenated codes ...................... 60  
3.1.5 Exercises ................................ 62  
3.2 Bounds on codes ........................... 63  
3.2.1 Singleton bound and MDS codes ............... 63  
3.2.2 Griesmer bound .......................... 68  
3.2.3 Hamming bound .......................... 69
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2.4</td>
<td>Plotkin bound</td>
<td>71</td>
</tr>
<tr>
<td>3.2.5</td>
<td>Gilbert and Varshamov bounds</td>
<td>72</td>
</tr>
<tr>
<td>3.2.6</td>
<td>Exercises</td>
<td>73</td>
</tr>
<tr>
<td>3.3</td>
<td>Asymptotically good codes</td>
<td>74</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Asymptotic Gilbert-Varshamov bound</td>
<td>74</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Some results for the generic case</td>
<td>77</td>
</tr>
<tr>
<td>3.3.3</td>
<td>Exercises</td>
<td>78</td>
</tr>
<tr>
<td>3.4</td>
<td>Notes</td>
<td>78</td>
</tr>
<tr>
<td>4</td>
<td>Weight enumerator</td>
<td>79</td>
</tr>
<tr>
<td>4.1</td>
<td>Weight enumerator</td>
<td>79</td>
</tr>
<tr>
<td>4.1.1</td>
<td>Weight spectrum</td>
<td>79</td>
</tr>
<tr>
<td>4.1.2</td>
<td>Average weight enumerator</td>
<td>83</td>
</tr>
<tr>
<td>4.1.3</td>
<td>MacWilliams identity</td>
<td>85</td>
</tr>
<tr>
<td>4.1.4</td>
<td>Exercises</td>
<td>88</td>
</tr>
<tr>
<td>4.2</td>
<td>Error probability</td>
<td>88</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Error probability of undetected error</td>
<td>89</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Probability of decoding error</td>
<td>89</td>
</tr>
<tr>
<td>4.2.3</td>
<td>Random coding</td>
<td>90</td>
</tr>
<tr>
<td>4.2.4</td>
<td>Exercises</td>
<td>90</td>
</tr>
<tr>
<td>4.3</td>
<td>Finite geometry and codes</td>
<td>90</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Projective space and projective systems</td>
<td>90</td>
</tr>
<tr>
<td>4.3.2</td>
<td>MDS codes and points in general position</td>
<td>95</td>
</tr>
<tr>
<td>4.3.3</td>
<td>Exercises</td>
<td>97</td>
</tr>
<tr>
<td>4.4</td>
<td>Extended weight enumerator</td>
<td>97</td>
</tr>
<tr>
<td>4.4.1</td>
<td>Arrangements of hyperplanes</td>
<td>97</td>
</tr>
<tr>
<td>4.4.2</td>
<td>Weight distribution of MDS codes</td>
<td>102</td>
</tr>
<tr>
<td>4.4.3</td>
<td>Extended weight enumerator</td>
<td>104</td>
</tr>
<tr>
<td>4.4.4</td>
<td>Puncturing and shortening</td>
<td>107</td>
</tr>
<tr>
<td>4.4.5</td>
<td>Exercises</td>
<td>110</td>
</tr>
<tr>
<td>4.5</td>
<td>Generalized weight enumerator</td>
<td>111</td>
</tr>
<tr>
<td>4.5.1</td>
<td>Generalized Hamming weights</td>
<td>111</td>
</tr>
<tr>
<td>4.5.2</td>
<td>Generalized weight enumerators</td>
<td>113</td>
</tr>
<tr>
<td>4.5.3</td>
<td>Generalized weight enumerators of MDS-codes</td>
<td>115</td>
</tr>
<tr>
<td>4.5.4</td>
<td>Connections</td>
<td>118</td>
</tr>
<tr>
<td>4.5.5</td>
<td>Exercises</td>
<td>120</td>
</tr>
<tr>
<td>4.6</td>
<td>Notes</td>
<td>120</td>
</tr>
<tr>
<td>5</td>
<td>Codes and related structures</td>
<td>123</td>
</tr>
<tr>
<td>5.1</td>
<td>Graphs and codes</td>
<td>123</td>
</tr>
<tr>
<td>5.1.1</td>
<td>Colorings of a graph</td>
<td>123</td>
</tr>
<tr>
<td>5.1.2</td>
<td>Codes on graphs</td>
<td>126</td>
</tr>
<tr>
<td>5.1.3</td>
<td>Exercises</td>
<td>127</td>
</tr>
<tr>
<td>5.2</td>
<td>Matroids and codes</td>
<td>128</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Matroids</td>
<td>128</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Realizable matroids</td>
<td>129</td>
</tr>
<tr>
<td>5.2.3</td>
<td>Graphs and matroids</td>
<td>130</td>
</tr>
<tr>
<td>5.2.4</td>
<td>Tutte and Whitney polynomial of a matroid</td>
<td>131</td>
</tr>
<tr>
<td>5.2.5</td>
<td>Weight enumerator and Tutte polynomial</td>
<td>132</td>
</tr>
<tr>
<td>5.2.6</td>
<td>Deletion and contraction of matroids</td>
<td>133</td>
</tr>
</tbody>
</table>
### CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.2.7</td>
<td>McWilliams type property for duality</td>
<td>134</td>
</tr>
<tr>
<td>5.2.8</td>
<td>Exercises</td>
<td>136</td>
</tr>
<tr>
<td>5.3</td>
<td>Geometric lattices and codes</td>
<td>136</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Posets, the Möbius function and lattices</td>
<td>136</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Geometric lattices</td>
<td>141</td>
</tr>
<tr>
<td>5.3.3</td>
<td>Geometric lattices and matroids</td>
<td>144</td>
</tr>
<tr>
<td>5.3.4</td>
<td>Exercises</td>
<td>145</td>
</tr>
<tr>
<td>5.4</td>
<td>Characteristic polynomial</td>
<td>146</td>
</tr>
<tr>
<td>5.4.1</td>
<td>Characteristic and Möbius polynomial</td>
<td>146</td>
</tr>
<tr>
<td>5.4.2</td>
<td>Characteristic polynomial of an arrangement</td>
<td>148</td>
</tr>
<tr>
<td>5.4.3</td>
<td>Characteristic polynomial of a code</td>
<td>150</td>
</tr>
<tr>
<td>5.4.4</td>
<td>Minimal codewords and subcodes</td>
<td>156</td>
</tr>
<tr>
<td>5.4.5</td>
<td>Two variable zeta function</td>
<td>157</td>
</tr>
<tr>
<td>5.4.6</td>
<td>Overview</td>
<td>157</td>
</tr>
<tr>
<td>5.4.7</td>
<td>Exercises</td>
<td>158</td>
</tr>
<tr>
<td>5.5</td>
<td>Combinatorics and codes</td>
<td>158</td>
</tr>
<tr>
<td>5.5.1</td>
<td>Orthogonal arrays and codes</td>
<td>158</td>
</tr>
<tr>
<td>5.5.2</td>
<td>Designs and codes</td>
<td>161</td>
</tr>
<tr>
<td>5.5.3</td>
<td>Exercises</td>
<td>161</td>
</tr>
<tr>
<td>5.6</td>
<td>Notes</td>
<td>162</td>
</tr>
<tr>
<td>6</td>
<td>Complexity and decoding</td>
<td>165</td>
</tr>
<tr>
<td>6.1</td>
<td>Complexity</td>
<td>165</td>
</tr>
<tr>
<td>6.1.1</td>
<td>Big-Oh notation</td>
<td>165</td>
</tr>
<tr>
<td>6.1.2</td>
<td>Boolean functions</td>
<td>166</td>
</tr>
<tr>
<td>6.1.3</td>
<td>Hard problems</td>
<td>171</td>
</tr>
<tr>
<td>6.1.4</td>
<td>Exercises</td>
<td>172</td>
</tr>
<tr>
<td>6.2</td>
<td>Decoding</td>
<td>173</td>
</tr>
<tr>
<td>6.2.1</td>
<td>Decoding complexity</td>
<td>173</td>
</tr>
<tr>
<td>6.2.2</td>
<td>Decoding erasures</td>
<td>174</td>
</tr>
<tr>
<td>6.2.3</td>
<td>Information and covering set decoding</td>
<td>177</td>
</tr>
<tr>
<td>6.2.4</td>
<td>Nearest neighbor decoding</td>
<td>184</td>
</tr>
<tr>
<td>6.2.5</td>
<td>Exercises</td>
<td>184</td>
</tr>
<tr>
<td>6.3</td>
<td>Difficult problems in coding theory</td>
<td>184</td>
</tr>
<tr>
<td>6.3.1</td>
<td>General decoding and computing minimum distance</td>
<td>184</td>
</tr>
<tr>
<td>6.3.2</td>
<td>Is decoding up to half the minimum distance hard?</td>
<td>187</td>
</tr>
<tr>
<td>6.3.3</td>
<td>Other hard problems</td>
<td>188</td>
</tr>
<tr>
<td>6.4</td>
<td>Notes</td>
<td>188</td>
</tr>
<tr>
<td>7</td>
<td>Cyclic codes</td>
<td>189</td>
</tr>
<tr>
<td>7.1</td>
<td>Cyclic codes</td>
<td>189</td>
</tr>
<tr>
<td>7.1.1</td>
<td>Definition of cyclic codes</td>
<td>189</td>
</tr>
<tr>
<td>7.1.2</td>
<td>Cyclic codes as ideals</td>
<td>191</td>
</tr>
<tr>
<td>7.1.3</td>
<td>Generator polynomial</td>
<td>192</td>
</tr>
<tr>
<td>7.1.4</td>
<td>Encoding cyclic codes</td>
<td>195</td>
</tr>
<tr>
<td>7.1.5</td>
<td>Reversible codes</td>
<td>196</td>
</tr>
<tr>
<td>7.1.6</td>
<td>Parity check polynomial</td>
<td>197</td>
</tr>
<tr>
<td>7.1.7</td>
<td>Exercises</td>
<td>200</td>
</tr>
<tr>
<td>7.2</td>
<td>Defining zeros</td>
<td>201</td>
</tr>
<tr>
<td>7.2.1</td>
<td>Structure of finite fields</td>
<td>201</td>
</tr>
</tbody>
</table>
7.2.2 Minimal polynomials ........................................... 205
7.2.3 Cyclotomic polynomials and cosets ......................... 206
7.2.4 Zeros of the generator polynomial ......................... 211
7.2.5 Exercises ........................................................ 213
7.3 Bounds on the minimum distance ............................... 214
7.3.1 BCH bound ...................................................... 214
7.3.2 Quadratic residue codes ...................................... 217
7.3.3 Hamming, simplex and Golay codes as cyclic codes ...... 217
7.3.4 Exercises ........................................................ 218
7.4 Improvements of the BCH bound ............................... 219
7.4.1 Hartmann-Tzeng bound ....................................... 219
7.4.2 Roos bound ..................................................... 220
7.4.3 AB bound ......................................................... 223
7.4.4 Shift bound ..................................................... 224
7.4.5 Exercises ........................................................ 228
7.5 Locator polynomials and decoding cyclic codes ............. 229
7.5.1 Mattson-Solomon polynomial ................................ 229
7.5.2 Newton identities ............................................. 230
7.5.3 APGZ algorithm ................................................. 232
7.5.4 Closed formulas ................................................ 234
7.5.5 Key equation and Forney's formula ......................... 235
7.5.6 Exercises ........................................................ 238
7.6 Notes ............................................................... 239
8 Polynomial codes .................................................. 241
8.1 RS codes and their generalizations ......................... 241
8.1.1 Reed-Solomon codes .......................................... 241
8.1.2 Extended and generalized RS codes ....................... 243
8.1.3 GRS codes under transformations ......................... 247
8.1.4 Exercises ........................................................ 250
8.2 Subfield and trace codes ...................................... 251
8.2.1 Restriction and extension by scalars .................... 251
8.2.2 Parity check matrix of a restricted code ................ 252
8.2.3 Invariant subspaces .......................................... 254
8.2.4 Cyclic codes as subfield subcodes ....................... 257
8.2.5 Trace codes ..................................................... 258
8.2.6 Exercises ........................................................ 258
8.3 Some families of polynomial codes ........................... 259
8.3.1 Alternant codes ............................................... 259
8.3.2 Goppa codes .................................................... 260
8.3.3 Counting polynomials ........................................ 263
8.3.4 Exercises ........................................................ 265
8.4 Reed-Muller codes .............................................. 266
8.4.1 Punctured Reed-Muller codes as cyclic codes ........... 266
8.4.2 Reed-Muller codes as subfield subcodes and trace codes 267
8.4.3 Exercises ........................................................ 270
8.5 Notes ............................................................... 270
Chapter 1

Introduction

Acknowledgement:

1.1 Notes
The idea of redundant information is a well-known phenomenon in reading a newspaper. Misspellings go usually unnoticed for a casual reader, while the meaning is still grasped. In Semitic languages such as Hebrew, and even older in the hieroglyphics in the tombs of the pharaohs of Egypt, only the consonants are written while the vowels are left out, so that we do not know for sure how to pronounce these words nowadays. The letter “e” is the most frequent occurring symbol in the English language, and leaving out all these letters would still give in almost all cases an understandable text to the expense of greater attention of the reader.

The art and science of deleting redundant information in a clever way such that it can be stored in less memory or space and still can be expanded to the original message, is called data compression or source coding. It is not the topic of this book. So we can compress data but an error made in a compressed text would give a different message that is most of the time completely meaningless.

The idea in error-correcting codes is the converse. One adds redundant information in such a way that it is possible to detect or even correct errors after transmission. In radio contacts between pilots and radars the letters in the alphabet are spoken phonetically as "Alpha, Bravo, Charlie, ..." but "Adams, Boston, Chicago, ..." is more commonly used for spelling in a telephone conversation. The addition of a parity check symbol enables one to detect an error, such as on the former punch cards that were fed to a computer, in the ISBN code for books, the European Article Numbering (EAN) and the Universal Product Code (UPC) for articles. Error-correcting codes are common in numerous situations such as in audio-visual media, fault-tolerant computers and deep space telecommunication.

more examples: QR quick response 2D code.
deeper space, compact disc and DVD, ......
more pictures
CHAPTER 2. ERROR-CORRECTING CODES

2.1 Block codes

Legend goes that Hamming was so frustrated the computer halted every time it detected an error after he handed in a stack of punch cards, he thought about a way the computer would be able not only to detect the error but also to correct it automatically. He came with his nowadays famous code named after him. Whereas the theory of Hamming is about the actual construction, the encoding and decoding of codes and uses tools from combinatorics and algebra, the approach of Shannon leads to information theory and his theorems tell us what is and what is not possible in a probabilistic sense.

According to Shannon we have a message $m$ in a certain alphabet and of a certain length, we encode $m$ to $c$ by expanding the length of the message and adding redundant information. One can define the information rate $R$ that measures the slowing down of the transmission of the data. The encoded message $c$ is sent over a noisy channel such that the symbols are changed, according to certain probabilities that are characteristic of the channel. The received word $r$ is decoded to $m'$. Now given the characteristics of the channel one can define the capacity $C$ of the channel and it has the property that for every $R < C$ it is possible to find an encoding and decoding scheme such that the error probability that $m' \neq m$ is arbitrarily small. For $R > C$ such a scheme is not possible. The capacity is explicitly known as a function of the characteristic probability for quite a number of channels.

The notion of a channel must be taken in a broad sense. Not only the transmission of data via satellite or telephone but also the storage of information on a hard disk of a computer or a compact disc for music and film can be modeled by a channel.

The theorem of Shannon tells us the existence of certain encoding and decoding schemes and one can even say that they exist in abundance and that almost all schemes satisfy the required conditions, but it does not tell us how to construct a specific scheme efficiently. The information theoretic part of error-correcting codes is considered in this book only so far to motivate the construction of coding and decoding algorithms.
The situation for the best codes in terms of the maximal number of errors that one can correct for a given information rate and code length is not so clear. Several existence and nonexistence theorems are known, but the exact bound is in fact still an open problem.

### 2.1 Repetition, product and Hamming codes

Adding a parity check such that the number of ones is even, is a well-known way to detect one error. But this does not correct the error.

**Example 2.1.1** Replacing every symbol by a threefold repetition gives the possibility of correcting one error in every 3-tuple of symbols in a received word by a majority vote. The price one has to pay is that the transmission is three times slower. We see here the two conflicting demands of error-correction: to correct as many errors as possible and to transmit as fast a possible. Notice furthermore that in case two errors are introduced by transmission the majority decoding rule will introduce an **decoding error**.

**Example 2.1.2** An improvement is the following product construction. Suppose we want to transmit a binary message \((m_1, m_2, m_3, m_4)\) of length 4 by adding 5 redundant bits \((r_1, r_2, r_3, r_4, r_5)\). Put these 9 bits in a \(3 \times 3\) array as shown below. The redundant bits are defined by the following conditions. The sum of the number of bits in every row and in every column should be even.

\[
\begin{array}{ccc}
m_1 & m_2 & r_1 \\
m_3 & m_4 & r_2 \\
r_3 & r_4 & r_5 \\
\end{array}
\]

It is clear that \(r_1, r_2, r_3\) and \(r_4\) are well defined by these rules. The condition on the last row and on the last column are equivalent, given the rules for the first two rows and columns. Hence \(r_5\) is also well defined.

If in the transmission of this word of 9 bits, one symbol is flipped from 0 to 1 or vice versa, then the receiver will notice this, and is able to correct it. Since if the error occurred in row \(i\) and column \(j\), then the receiver will detect an odd parity in this row and this column and an even parity in the remaining rows and columns. Suppose that the message is \(m = (1, 1, 0, 1)\). Then the redundant part is \(r = (0, 1, 1, 0, 1)\) and \(c = (1, 1, 0, 1, 0, 1, 1, 0, 1)\) is transmitted. Suppose that \(y = (1, 1, 0, 1, 0, 1, 0, 1, 0, 1)\) is the received word.

\[
\begin{array}{ccc}
1 & 1 & 0 \\
0 & 1 & 0 \\
1 & 0 & 1 \\
\end{array}
\]

Then the receiver detects an error in row 2 and column 3 and will change the corresponding symbol.

So this product code can also correct one error as the repetition code but its information rate is improved from 1/3 to 4/9.

This decoding scheme is **incomplete** in the sense that in some cases it is not decided what to do and the scheme will fail to determine a candidate for the transmitted word. That is called a **decoding failure**. Sometimes two errors can be corrected. If the first error is in row \(i\) and column \(j\), and the second in row \(i'\).
and column \( j' \) with \( i' > i \) and \( j' \neq j \). Then the receiver will detect odd parities in rows \( i \) and \( i' \) and in columns \( j \) and \( j' \). There are two error patterns of two errors with this behavior. That is errors at the positions \((i, j)\) and \((i', j')\) or at the two pairs \((i, j')\) and \((i', j)\). If the receiver decides to change the first two pairs if \( j' > j \) and the second two pairs if \( j' < j \), then it will recover the transmitted word half of the time this pattern of two errors takes place. If for instance the word \( c = (1, 1, 0, 1, 0, 1, 0, 1) \) is transmitted and \( y = (1, 0, 0, 1, 0, 1, 0, 1) \) is received, then the above decoding scheme will change it correctly in \( c \). But \( y' = (1, 1, 0, 1, 1, 0, 1) \) is received, then the scheme will change it in the codeword \( c' = (1, 0, 0, 1, 0, 1, 0, 1) \) and we have a decoding error.

If two errors take place in the same row, then the receiver will see an even parity in all rows and odd parities in the columns \( j \) and \( j' \). We can expand the decoding rule to change the bits at the positions \((1, j)\) and \((1, j')\). Likewise we will change the bits in positions \((i, 1)\) and \((i', 1)\) if the columns give even parity and the rows \( i \) and \( i' \) have an odd parity. This decoding scheme will correct all patterns with 1 error correctly, and sometimes the patterns with 2 errors. But it is still incomplete, since the received word \((1, 1, 0, 1, 0, 0, 0, 1)\) has an odd parity in every row and in every column and the scheme fails to decode. One could extend the decoding rule to get a complete decoding in such a way that every received word is decoded to a nearest codeword. This nearest codeword is not always unique.

In case the transmission is by means of certain electro-magnetic pulses or waves one has to consider modulation and demodulation. The message consists of letters of a finite alphabet, say consisting of zeros and ones, and these are modulated, transmitted as waves, received and demodulated in zeros and ones. In the demodulation part one has to make a hard decision between a zero or a one. But usually there is a probability that the signal represents a zero. The hard decision together with this probability is called a soft decision. One can make use of this information in the decoding algorithm. One considers the list of all nearest codewords, and one chooses the codeword in this list that has the highest probability.

**Example 2.1.3** An improvement of the repetition code of rate \(1/3\) and the product code of rate \(4/9\) is given by Hamming. Suppose we have a message \((m_1, m_2, m_3, m_4)\) of 4 bits. Put them in the middle of the following Venn-diagram of three intersecting circles as given in Figure 2.2. Complete the three empty areas of the circles according to the rule that the number of ones in every circle is even. In this way we get 3 redundant bits \((r_1, r_2, r_3)\) that we add to the message and which we transmit over the channel. In every block of 7 bits the receiver can correct one error. Since the parity in every circle should be even. So if the parity is even we declare the circle correct, if the parity is odd we declare the circle incorrect. The error is in the incorrect circles and in the complement of the correct circles. We see that every pattern of at most one error can be corrected in this way. For instance, if \( m = (1, 1, 0, 1) \) is the message, then \( r = (0, 0, 1) \) is the redundant information.
added and $c = (1, 1, 0, 1, 0, 1, 1, 0, 1)$ the codeword sent. If after transmission one symbol is flipped and $y = (1, 0, 0, 1, 0, 1, 0, 1)$ is the received word as given in Figure 2.3.

Then we conclude that the error is in the left and upper circle, but not in the right one. And we conclude that the error is at $m_2$. But in case of 2 errors and for instance the word $y' = (1, 0, 0, 1, 1, 0, 1)$ is received, then the receiver would assume that the error occurred in the upper circle and not in the two lower circles, and would therefore conclude that the transmitted codeword was $(1, 0, 0, 1, 1, 0, 0)$. Hence the decoding scheme creates an extra error.

The redundant information $r$ can be obtained from the message $m$ by means of three linear equations or parity checks modulo two

\[
\begin{align*}
   r_1 &= m_2 + m_3 + m_4 \\
   r_2 &= m_1 + m_3 + m_4 \\
   r_3 &= m_1 + m_2 + m_4
\end{align*}
\]

Let $c = (m, r)$ be the codeword. Then $c$ is a codeword if and only if $He^T = 0$,
where

$$H = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}.$$ 

The information rate is improved from 1/3 for the repetition code and 4/9 for the product code to 4/7 for the Hamming code.

*** gate diagrams of encoding/decoding scheme ***

### 2.1.2 Codes and Hamming distance

In general the alphabets of the message word and the encoded word might be distinct. Furthermore the length of both the message word and the encoded word might vary such as in a convolutional code. We restrict ourselves to \([n,k]\) block codes that is the message words have a fixed length of \(k\) symbols and the encoded words have a fixed length of \(n\) symbols both from the same alphabet \(Q\).

For the purpose of error control, before transmission, we add redundant symbols to the message in a clever way.

**Definition 2.1.4** Let \(Q\) be a set of \(q\) symbols called the alphabet. Let \(Q^n\) be the set of all \(n\)-tuples \(x = (x_1, \ldots, x_n)\), with entries \(x_i \in Q\). A block code \(C\) of length \(n\) over \(Q\) is a non-empty subset of \(Q^n\). The elements of \(C\) are called codewords. If \(C\) contains \(M\) codewords, then \(M\) is called the size of the code.

For an \((n,M)\) code defined over \(Q\), the value \(n - \log_q(M)\) is called the redundancy. The information rate is defined as \(R = \log_q(M)/n\).

**Example 2.1.5** The repetition code has length 3 and 2 codewords, so its information rate is 1/3. The product code has length 9 and 2\(^4\) codewords, hence its rate is 4/9. The Hamming code has length 7 and 2\(^4\) codewords, therefore its rate is 4/7.

**Example 2.1.6** Let \(C\) be the binary block code of length \(n\) consisting of all words with exactly two ones. This is an \((n,n(n-1)/2)\) code. In this example the number of codewords is not a power of the size of the alphabet.

**Definition 2.1.7** Let \(C\) be an \([n,k]\) block code over \(Q\). An encoder of \(C\) is a one-to-one map 

\[E : Q^k \rightarrow Q^n\]

such that \(C = E(Q^k)\). Let \(c \in C\) be a codeword. Then there exists a unique \(m \in Q^k\) with \(c = E(m)\). This \(m\) is called the message or source word of \(c\).

In order to measure the difference between two distinct words and to evaluate the error-correcting capability of the code, we need to introduce an appropriate metric to \(Q^n\). A natural metric used in Coding Theory is the Hamming distance.

**Definition 2.1.8** For \(x = (x_1, \ldots, x_n), y = (y_1, \ldots, y_n) \in Q^n\), the Hamming distance \(d(x,y)\) is defined as the number of places where they differ:

\[d(x,y) = |\{i \mid x_i \neq y_i\}|.\]
Proposition 2.1.9 The Hamming distance is a metric on $\mathbb{Q}^n$, that means that the following properties hold for all $x, y, z \in \mathbb{Q}^n$:

1. $d(x, y) \geq 0$ and equality holds if and only if $x = y$,
2. $d(x, y) = d(y, x)$ (symmetry),
3. $d(x, z) \leq d(x, y) + d(y, z)$ (triangle inequality),

Proof. Properties (1) and (2) are trivial from the definition. We leave (3) to the reader as an exercise. \hfill \lozenge

Definition 2.1.10 The minimum distance of a code $C$ of length $n$ is defined as

$$d = d(C) = \min \{ d(x, y) \mid x, y \in C, x \neq y \}$$

if $C$ consists of more than one element, and is by definition $n + 1$ if $C$ consists of one word. We denote by $(n, M, d)$ a code $C$ with length $n$, size $M$ and minimum distance $d$.

The main problem of error-correcting codes from “Hamming’s point view” is to construct for a given length and number of codewords a code with the largest possible minimum distance, and to find efficient encoding and decoding algorithms for such a code.

Example 2.1.11 The triple repetition code consists of two codewords: $(0, 0, 0)$ and $(1, 1, 1)$, so its minimum distance is 3. The product and Hamming code both correct one error. So the minimum distance is at least 3, by the triangle inequality. The product code has minimum distance 4 and the Hamming code has minimum distance 3. Notice that all three codes have the property that $x + y$ is again a codeword if $x$ and $y$ are codewords.

Definition 2.1.12 Let $x \in \mathbb{Q}^n$. The ball of radius $r$ around $x$, denoted by $B_r(x)$, is defined as $B_r(x) = \{ y \in \mathbb{Q}^n \mid d(x, y) \leq r \}$. The sphere of radius $r$ around $x$ is denoted by $S_r(x)$ and defined as $S_r(x) = \{ y \in \mathbb{Q}^n \mid d(x, y) = r \}$. 
Figure 2.5: Ball of radius $\sqrt{2}$ in the Euclidean plane

Figure 2.6: Balls of radius 0 and 1 in the Hamming metric

Figure 2.1.2 shows the ball in the Euclidean plane. This is misleading in some respects, but gives an indication what we should have in mind.

Figure 2.1.2 shows $Q^2$, where the alphabet $Q$ consists of 5 elements. The ball $B_0(x)$ consists of the points in the circle, $B_1(x)$ is depicted by the points inside the cross, and $B_2(x)$ consists of all 25 dots.

**Proposition 2.1.13** Let $Q$ be an alphabet of $q$ elements and $x \in Q^n$. Then

$$|S_i(x)| = \binom{n}{i} (q-1)^i \quad \text{and} \quad |B_r(x)| = \sum_{i=0}^{r} \binom{n}{i} (q-1)^i.$$  

**Proof.** Let $y \in S_i(x)$. Let $I$ be the subset of $\{1, \ldots, n\}$ consisting of all positions $j$ such that $y_j \neq x_j$. Then the number of elements of $I$ is equal to $i$. And $(q-1)^i$ is the number of words $y \in S_i(x)$ that have the same fixed $I$. The number of possibilities to choose the subset $I$ with a fixed number of elements $i$ is equal to $\binom{n}{i}$. This shows the formula for the number of elements of $S_i(x)$. Furthermore $B_r(x)$ is the disjoint union of the subsets $S_i(x)$ for $i = 0, \ldots, r$. This proves the statement about the number of elements of $B_r(x)$. \(\diamondsuit\)
2.2. LINEAR CODES

2.1.3 Exercises

2.1.1 Consider the code of length 8 that is obtained by deleting the last entry $r_5$ from the product code of Example 2.1.2. Show that this code corrects one error.

2.1.2 Give a gate diagram of the decoding algorithm for the product code of Example 2.1.2 that corrects always 1 error and sometimes 2 errors.

2.1.3 Give a proof of Proposition 2.1.9 (3), that is the triangle inequality of the Hamming distance.

2.1.4 Let $Q$ be an alphabet of $q$ elements. Let $x, y \in Q^n$ have distance $d$. Show that the number of elements in the intersection $B_r(x) \cap B_s(y)$ is equal to

$$\sum_{i,j,k} \binom{d}{i} \binom{d-i}{j} \binom{n-d}{k} (q-2)^i(q-1)^k,$$

where $i, j$ and $k$ are non-negative integers such that $i + j \leq d$, $k \leq n - d$, $i + j + k \leq r$ and $d - i + k \leq s$.

2.1.5 Write a procedure in GAP that takes $n$ as an input and constructs the code as in Example 2.1.6.

2.2 Linear Codes

Linear codes are introduced in case the alphabet is a finite field. These codes have more structure and are therefore more tangible than arbitrary codes.

2.2.1 Linear codes

If the alphabet $Q$ is a finite field, then $Q^n$ is a vector space. This is for instance the case if $Q = \{0, 1\} = \mathbb{F}_2$. Therefore it is natural to look at codes in $Q^n$ that have more structure, in particular that are linear subspaces.

Definition 2.2.1 A linear code $C$ is a linear subspace of $\mathbb{F}_q^n$, where $\mathbb{F}_q$ stands for the finite field with $q$ elements. The dimension of a linear code is its dimension as a linear space over $\mathbb{F}_q$. We denote a linear code $C$ over $\mathbb{F}_q$ of length $n$ and dimension $k$ by $[n,k]_q$, or simply by $[n,k]$. If furthermore the minimum distance of the code is $d$, then we call by $[n,k,d]_q$ or $[n,k,d]$ the parameters of the code.

It is clear that for a linear $[n,k]$ code over $\mathbb{F}_q$, its size $M = q^k$. The information rate is $R = k/n$ and the redundancy is $n - k$.

Definition 2.2.2 For a word $x \in \mathbb{F}_q^n$, its support, denoted by supp$(x)$, is defined as the set of nonzero coordinate positions, so supp$(x) = \{i \mid x_i \neq 0\}$. The weight of $x$ is defined as the number of elements of its support, which is denoted by wt$(x)$. The minimum weight of a code $C$, denoted by mwt$(C)$, is defined as the minimal value of the weights of the nonzero codewords:

$$\text{mwt}(C) = \min \{ \text{wt}(c) \mid c \in C, \ c \neq 0 \},$$

in case there is a $c \in C$ not equal to 0, and $n + 1$ otherwise.
Proposition 2.2.3 The minimum distance of a linear code $C$ is equal to its minimum weight.

Proof. Since $C$ is a linear code, we have that $0 \in C$ and for any $c_1, c_2 \in C$, $c_1 - c_2 \in C$. Then the conclusion follows from the fact that $\text{wt}(c) = d(0, c)$ and $d(c_1, c_2) = \text{wt}(c_1 - c_2)$.

Definition 2.2.4 Consider the situation of two $\mathbb{F}_q$-linear codes $C$ and $D$ of length $n$. If $D \subseteq C$, then $D$ is called a subcode of $C$, and $C$ a supercode of $D$.

Remark 2.2.5 Suppose $C$ is an $[n, k, d]$ code. Then, for any $r$, $1 \leq r \leq k$, there exist subcodes with dimension $r$. And for any given $r$, there may exist more than one subcode with dimension $r$. The minimum distance of a subcode is always greater than or equal to $d$. So, by taking an appropriate subcode, we can get a new code of the same length which has a larger minimum distance. We will discuss this later in Section 3.1.

Now let us see some examples of linear codes.

Example 2.2.6 The repetition code over $\mathbb{F}_q$ of length $n$ consists of all words $c = (c, c, \ldots, c)$ with $c \in \mathbb{F}_q$. This is a linear code of dimension 1 and minimum distance $n$.

Example 2.2.7 Let $n$ be an integer with $n \geq 2$. The even weight code $C$ of length $n$ over $\mathbb{F}_q$ consists of all words in $\mathbb{F}_q^n$ of even weight. The minimum weight of $C$ is by definition 2, the minimum distance of $C$ is 2 if $q = 2$ and 1 otherwise. The code $C$ linear if and only if $q = 2$.

Example 2.2.8 Let $C$ be a binary linear code. Consider the subset $C_{ev}$ of $C$ consisting of all codewords in $C$ of even weight. Then $C_{ev}$ is a linear subcode and is called the even weight subcode of $C$. If $C \neq C_{ev}$, then there exists a codeword $c$ in $C$ of odd weight and $C$ is the disjunct union of the cosets $c + C_{ev}$ and $C_{ev}$. Hence $\dim(C_{ev}) \geq \dim(C) - 1$.

Example 2.2.9 The Hamming code $C$ of Example 2.1.3 consists of all the words $c \in \mathbb{F}_2^7$ satisfying $Hc^T = 0$, where

$$H = \begin{pmatrix}
0 & 1 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 1 & 0 \\
1 & 1 & 0 & 1 & 0 & 0 & 1
\end{pmatrix}.$$

This code is linear of dimension 4, since it is given by the solutions of three independent homogeneous linear equations. The minimum weight is 3 as shown in Example 2.1.11. So it is a $[7, 4, 3]$ code.

2.2.2 Generator matrix and systematic encoding

Let $C$ be an $[n, k]$ linear code over $\mathbb{F}_q$. Since $C$ is a $k$-dimensional linear subspace of $\mathbb{F}_q^n$, there exists a basis that consists of $k$ linearly independent codewords, say $g_1, \ldots, g_k$. Suppose $g_i = (g_{i1}, \ldots, g_{in})$ for $i = 1, \ldots, k$. Denote

$$G = \begin{pmatrix}
\begin{bmatrix}
g_1 \\
g_2 \\
\vdots \\
g_k
\end{bmatrix} \\
\begin{bmatrix}
g_{11} & g_{12} & \cdots & g_{1n} \\
g_{21} & g_{22} & \cdots & g_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
g_{k1} & g_{k2} & \cdots & g_{kn}
\end{bmatrix}
\end{pmatrix}.$$
Every codeword \( \mathbf{c} \) can be written uniquely as a linear combination of the basis elements, so \( \mathbf{c} = m_1 \mathbf{g}_1 + \cdots + m_k \mathbf{g}_k \) where \( m_1, \ldots, m_k \in \mathbb{F}_q \). Let \( \mathbf{m} = (m_1, \ldots, m_k) \in \mathbb{F}_q^k \). Then \( \mathbf{c} = \mathbf{m} \mathbf{G} \). The \textit{encoding}
\[ \mathcal{E} : \mathbb{F}_q^k \rightarrow \mathbb{F}_q^n, \]
from the message word \( \mathbf{m} \in \mathbb{F}_q^k \) to the codeword \( \mathbf{c} \in \mathbb{F}_q^n \) can be done efficiently by a matrix multiplication.

\[ \mathbf{c} = \mathcal{E}(\mathbf{m}) := \mathbf{m} \mathbf{G}. \]

**Definition 2.2.10** A \( k \times n \) matrix \( \mathbf{G} \) with entries in \( \mathbb{F}_q \) is called a \textit{generator matrix} of an \( \mathbb{F}_q \)-linear code \( \mathcal{C} \) if the rows of \( \mathbf{G} \) are a basis of \( \mathcal{C} \).

A given \([n,k]\) code \( \mathcal{C} \) can have more than one generator matrix, however every generator matrix of \( \mathcal{C} \) is a \( k \times n \) matrix of rank \( k \). Conversely every \( k \times n \) matrix of rank \( k \) is the generator matrix of an \( \mathbb{F}_q \)-linear \([n,k]\) code.

**Example 2.2.11** The linear codes with parameters \([n,0,n+1]\) and \([n,n,1]\) are the trivial codes \{0\} and \( \mathbb{F}_q^n \), and they have the empty matrix and the \( n \times n \) identity matrix \( I_n \) as generator matrix, respectively.

**Example 2.2.12** The repetition code of length \( n \) has generator matrix
\[ \mathbf{G} = \begin{pmatrix} 1 & 1 & \cdots & 1 \end{pmatrix}. \]

**Example 2.2.13** The binary even-weight code of length \( n \) has for instance the following two generator matrices
\[
\frac{1}{2} \begin{pmatrix} 1 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 1 & 1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 & 1 \end{pmatrix}
\text{ and }
\begin{pmatrix} 1 & 0 & \cdots & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & \cdots & 0 & 1 & 0 & 1 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 & 1 & 1 & 1 \end{pmatrix}.
\]

**Example 2.2.14** The Hamming code \( C \) of Example 2.1.3 is a \([7,4]\) code. The message symbols \( m_i \) for \( i = 1, \ldots, 4 \) are free to choose. If we take \( m_1 = 1 \) and the remaining \( m_j = 0 \) for \( j \neq i \) we get the codeword \( \mathbf{g}_i \). In this way we get the basis \( \mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3, \mathbf{g}_4 \) of the code \( \mathcal{C} \), that are the rows of following generator matrix
\[ \mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}. \]

From the example, the generator matrix \( \mathbf{G} \) of the Hamming code has the following form
\[ (I_k \mid P) \]
where \( I_k \) is the \( k \times k \) identity matrix and \( P \) a \( k \times (n-k) \) matrix.
Remark 2.2.15 Let \( G \) be a generator matrix of \( C \). From Linear Algebra, see Section ??, we know that we can transform \( G \) by Gaussian elimination in a row equivalent matrix in row reduced echelon form by a sequence of the three elementary row operations:
1) interchanging two rows,
2) multiplying a row with a nonzero constant,
3) adding one row to another row.
Moreover for a given matrix \( G \), there is exactly one row equivalent matrix that is in row reduced echelon form, denoted by \( \text{rref}(G) \). In the following proposition it is stated that \( \text{rref}(G) \) is also a generator matrix of \( C \).

Proposition 2.2.16 Let \( G \) be a generator matrix of \( C \). Then \( \text{rref}(G) \) is also a generator matrix of \( C \) and \( \text{rref}(G) = MG \), where \( M \) is an invertible \( k \times k \) matrix with entries in \( \mathbb{F}_q \).

Proof. The row reduced echelon form \( \text{rref}(G) \) of \( G \) is obtained from \( G \) by a sequence of elementary operations. The code \( C \) is equal to the row space of \( G \), and the row space does not change under elementary row operations. So \( \text{rref}(G) \) generates the same code \( C \). Furthermore \( \text{rref}(G) = E_1 \cdots E_i G \), where \( E_1, \ldots, E_i \) are the elementary matrices that correspond to the elementary row operations. Let \( M = E_1 \cdots E_i \). Then \( M \) is an invertible matrix, since the \( E_i \) are invertible, and \( \text{rref}(G) = MG \).

Proposition 2.2.17 Let \( G_1 \) and \( G_2 \) be two \( k \times n \) generator matrices generating the codes \( C_1 \) and \( C_2 \) over \( \mathbb{F}_q \). Then the following statements are equivalent:
1) \( C_1 = C_2 \),
2) \( \text{rref}(G_1) = \text{rref}(G_2) \),
3) there is a \( k \times k \) invertible matrix \( M \) with entries in \( \mathbb{F}_q \) such that \( G_2 = MG_1 \).

Proof. 1) implies 2): The row spaces of \( G_1 \) and \( G_2 \) are the same, since \( C_1 = C_2 \). So \( G_1 \) and \( G_2 \) are row equivalent. Hence \( \text{rref}(G_1) = \text{rref}(G_2) \).
2) implies 3): Let \( R_i = \text{rref}(G_i) \). There is a \( k \times k \) invertible matrix \( M_i \) such that \( G_i = M_i R_i \) for \( i = 1, 2 \), by Proposition 2.2.17. Let \( M = M_2 M_1^{-1} \). Then
\[
MG_1 = M_2 M_1^{-1} M_1 R_1 = M_2 R_2 = G_2.
\]
3) implies 1): Suppose \( G_2 = MG_1 \) for some \( k \times k \) invertible matrix \( M \). Then every codeword of \( C_2 \) is a linear combination of the rows of \( G_1 \) that are in \( C_1 \). So \( C_2 \) is a subcode of \( C_1 \). Similarly \( C_1 \subseteq C_2 \), since \( G_1 = M^{-1} G_2 \). Hence \( C_1 = C_2 \).

Remark 2.2.18 Although a generator matrix \( G \) of a code \( C \) is not unique, the row reduced echelon form \( \text{rref}(G) \) is unique. That is to say, if \( G \) is a generator matrix of \( C \), then \( \text{rref}(G) \) is also a generator matrix of \( C \), and furthermore if \( G_1 \) and \( G_2 \) are generator matrices of \( C \), then \( \text{rref}(G_1) = \text{rref}(G_2) \). Therefore the row reduced echelon form \( \text{rref}(C) \) of a code \( C \) is well-defined, being \( \text{rref}(G) \) for a generator matrix \( G \) of \( C \) by Proposition 2.2.17.

Example 2.2.19 The generator matrix \( G_2 \) of Example 2.2.13 is in row-reduced echelon form and a generator matrix of the binary even-weight code \( C \). Hence \( G_2 = \text{rref}(G_1) = \text{rref}(C) \).
2.2. LINEAR CODES

Definition 2.2.20 Let $C$ be an $[n, k]$ code. The code is called systematic at the positions $(j_1, \ldots, j_k)$ if for all $m \in \mathbb{F}_q^k$ there exists a unique codeword $c$ such that $c_{j_i} = m_i$ for all $i = 1, \ldots, k$. In that case, the set $\{j_1, \ldots, j_k\}$ is called an information set. A generator matrix $G$ of $C$ is called systematic at the positions $(j_1, \ldots, j_k)$ if the $k \times k$ submatrix $G'$ consisting of the $k$ columns of $G$ at the positions $(j_1, \ldots, j_k)$ is the identity matrix. For such a matrix $G$ the mapping $m \mapsto mG$ is called systematic encoding.

Remark 2.2.21 If a generator matrix $G$ of $C$ is systematic at the positions $(j_1, \ldots, j_k)$ and $c$ is a codeword, then $c = mG$ for a unique $m \in \mathbb{F}_q^k$ and $c_{j_i} = m_i$ for all $i = 1, \ldots, k$. Hence $C$ is systematic at the positions $(j_1, \ldots, j_k)$.

Now suppose that the $j_i$ with $1 \leq j_1 < \cdots < j_k \leq n$ indicate the positions of the pivots of $\text{rref}(G)$. Then the code $C$ and the generator matrix $\text{rref}(G)$ are systematic at the positions $(j_1, \ldots, j_k)$.

Proposition 2.2.22 Let $C$ be a code with generator matrix $G$. Then $C$ is systematic at the positions $j_1, \ldots, j_k$ if and only if the $k$ columns of $G$ at the positions $j_1, \ldots, j_k$ are linearly independent.

Proof. Let $G$ be a generator matrix of $C$. Let $G'$ be the $k \times k$ submatrix of $G$ consisting of the $k$ columns at the positions $(j_1, \ldots, j_k)$. Suppose $C$ is systematic at the positions $(j_1, \ldots, j_k)$. Then the map given by $x \mapsto xG'$ is injective. Hence the columns of $G'$ are linearly independent.

Conversely, if the columns of $G'$ are linearly independent, then there exists a $k \times k$ invertible matrix $M$ such that $MG'$ is the identity matrix. Hence $MG$ is a generator matrix of $C$ and $C$ is systematic at $(j_1, \ldots, j_k)$. ⊓⊔

Example 2.2.23 Consider a code $C$ with generator matrix

$$G = \begin{pmatrix}
1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\
1 & 1 & 0 & 1 & 0 & 1 & 1 & 1
\end{pmatrix}.$$ 

Then

$$\text{rref}(C) = \text{rref}(G) = \begin{pmatrix}
1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}$$

and the code is systematic at the positions 1, 2, 4 and 8. By the way we notice that the minimum distance of the code is 1.

2.2.3 Exercises

2.2.1 Determine for the product code of Example 2.1.2 the number of codewords, the number of codewords of a given weight, the minimum weight and the minimum distance. Express the redundant bits $r_j$ for $j = 1, \ldots, 5$ as linear equations over $\mathbb{F}_2$ in the message bits $m_i$ for $i = 1, \ldots, 4$. Give a $5 \times 9$ matrix $H$ such that $c = (m, r)$ is a codeword of the product code if and only if $He^T = 0$, where $m$ is the message of 4 bits $m_i$ and $r$ is the vector with the 5 redundant bits $r_j$. 

2.2.2 Let \( x \) and \( y \) be binary words of the same length. Show that

\[
\text{wt}(x + y) = \text{wt}(x) + \text{wt}(y) - 2|\text{supp}(x) \cap \text{supp}(y)|.
\]

2.2.3 Let \( C \) be an \( \mathbb{F}_q \)-linear code with generator matrix \( G \). Let \( q = 2 \). Show that every codeword of \( C \) has even weight if and only if every row of a \( G \) has even weight. Show by means of a counter example that the above statement is not true if \( q \neq 2 \).

2.2.4 Consider the following matrix with entries in \( \mathbb{F}_5 \)

\[
G = \begin{pmatrix}
1 & 1 & 1 & 1 & 0 \\
0 & 1 & 2 & 3 & 4 \\
0 & 1 & 4 & 1 & 1
\end{pmatrix}.
\]

Show that \( G \) is a generator matrix of a \([5, 3, 3]\) code. Give the row reduced echelon form of this code.

2.2.5 Compute the complexity of the encoding of a linear \([n, k]\) code by an arbitrary generator matrix \( G \) and in case \( G \) is systematic, respectively, in terms of the number of additions and multiplications.

2.3 Parity checks and dual code

Linear codes are implicitly defined by parity check equations and the dual of a code is introduced.

2.3.1 Parity check matrix

There are two standard ways to describe a subspace, explicitly by giving a basis, or implicitly by the solution space of a set of homogeneous linear equations. Therefore there are two ways to describe a linear code. That is explicitly as we have seen by a generator matrix, or implicitly by a set of homogeneous linear equations that is by the null space of a matrix.

Let \( C \) be an \( \mathbb{F}_q \)-linear \([n, k]\) code. Suppose that \( H \) is an \( m \times n \) matrix with entries in \( \mathbb{F}_q \). Let \( C \) be the null space of \( H \). So \( C \) is the set of all \( c \in \mathbb{F}_q^n \) such that \( Hc^T = 0 \). These \( m \) homogeneous linear equations are called parity check equations, or simply parity checks. The dimension \( k \) of \( C \) is at least \( n - m \). If there are dependent rows in the matrix \( H \), that is if \( k > n - m \), then we can delete a few rows until we obtain an \((n - k) \times n\) matrix \( H' \) with independent rows and with the same null space as \( H \). So \( H' \) has rank \( n - k \).

**Definition 2.3.1** An \((n - k) \times n\) matrix of rank \( n - k \) is called a parity check matrix of an \([n, k]\) code \( C \) if \( C \) is the null space of this matrix.

**Remark 2.3.2** The parity check matrix of a code can be used for error detection. This is useful in a communication channel where one asks for retransmission in case more than a certain number of errors occurred. Suppose that \( C \) is a linear code of minimum distance \( d \) and \( H' \) is a parity check matrix of \( C \). Suppose that the codeword \( c \) is transmitted and \( r = c + e \) is received. Then \( e \)
is called the error vector and \( \text{wt}(e) \) the number of errors. Now \( Hr^T = 0 \) if there is no error and \( Hr^T \neq 0 \) for all \( e \) such that \( 0 < \text{wt}(e) < d \). Therefore we can detect any pattern of \( t \) errors with \( t < d \). But not more, since if the error vector is equal to a nonzero codeword of minimal weight \( d \), then the receiver would assume that no errors have been made. The vector \( Hr^T \) is called the syndrome of the received word.

We show that every linear code has a parity check matrix and we give a method to obtain such a matrix in case we have a generator matrix \( G \) of the code.

**Proposition 2.3.3** Suppose \( C \) is an \([n,k]\) code. Let \( I_k \) be the \( k \times k \) identity matrix. Let \( P \) be a \( k \times (n-k) \) matrix. Then, \((I_k|P)\) is a generator matrix of \( C \) if and only if \((-P^T|I_{n-k})\) is a parity check matrix of \( C \).

**Proof.** Every codeword \( c \) is of the form \( mG \) with \( m \in \mathbb{F}_q^k \). Suppose that the generator matrix \( G \) is systematic at the first \( k \) positions. So \( c = (m, r) \) with \( r \in \mathbb{F}_q^{n-k} \) and \( r = mp \). Hence for a word of the form \( c = (m, r) \) with \( m \in \mathbb{F}_q^k \) and \( r \in \mathbb{F}_q^{n-k} \) the following statements are equivalent:

\[
\begin{align*}
c \text{ is a codeword,} \\
-mP + r &= 0, \\
-P^Tm + r^T &= 0, \\
(-P^T|I_{n-k})(m, r)^T &= 0, \\
(-P^T|I_{n-k})c^T &= 0.
\end{align*}
\]

Hence \((-P^T|I_{n-k})\) is a parity check matrix of \( C \). The converse is proved similarly.

**Example 2.3.4** The trivial codes \{0\} and \( \mathbb{F}_q^n \) have \( I_n \) and the empty matrix as parity check matrix, respectively.

**Example 2.3.5** As a consequence of Proposition 2.3.3 we see that a parity check matrix of the binary even weight code is equal to the generator matrix \((1 \ 1 \ \cdots \ 1)\) of the repetition code, and the generator matrix \( G_2 \) of the binary even weight code of Example 2.2.13 is a parity check matrix of the repetition code.

**Example 2.3.6** The ISBN code of a book consists of a word \((b_1, \ldots, b_{10})\) of 10 symbols of the alphabet with the 11 elements: 0, 1, 2, \ldots, 9 and \( X \) of the finite field \( \mathbb{F}_{11} \), where \( X \) is the symbol representing 10, that satisfies the parity check equation:

\[
b_1 + 2b_2 + 3b_3 + \cdots + 10b_{10} = 0.
\]

Clearly his code detects one error. This code corrects many patterns of one transposition of two consecutive symbols. Suppose that the symbols \( b_i \) and \( b_{i+1} \) are interchanged and there are no other errors, then the parity check gives as outcome

\[
i b_{i+1} + (i + 1)b_i + \sum_{j \neq i,i+1} j b_j = s.
\]
We know that $\sum_j j b_j = 0$, since $(b_1, \ldots, b_{10})$ is an ISBN codeword. Hence $s = b_i - b_{i+1}$. But this position $i$ is in general not unique.

Consider for instance the following code: 0444815933. Then the checksum gives 4, so it is not a valid ISBN code. Now assume that the code is the result of transposition of two consecutive symbols. Then 4044815933, 0448415933, 0444185933, 044851933 and 0444851933 are the possible ISBN codes. The first and third code do not match with existing books. The second, fourth and fifth code correspond to books with the titles: “The revenge of the dragon lady,” “The theory of error-correcting codes” and “Nagasaki’s symposium on Chernobyl,” respectively.

**Example 2.3.7** The generator matrix $G$ of the Hamming code $C$ in Example 2.2.14 is of the form $(I_4|P)$ and in Example 2.2.9 we see that the parity check matrix is equal to $(P^T|I_3)$.

**Remark 2.3.8** Let $G$ be a generator matrix of an $[n,k]$ code $C$. Then the row reduced echelon form $G_1 = \text{rref}(G)$ is not systematic at the first $k$ positions but at the positions $(j_1, \ldots, j_k)$ with $1 \leq j_1 < \cdots < j_k \leq n$. After a permutation $\pi$ of the $n$ positions with corresponding $n \times n$ permutation matrix, denoted by $\Pi$, we may assume that $G_2 = G_1 \Pi$ is of the form $(I_k|P)$. Now $G_2$ is a generator matrix of the code $C_2$ which is not necessarily equal to $C$. A parity check matrix $H_2$ for $C_2$ is given by $(-P^T|I_{n-k})$ according to Proposition 2.3.3. A parity check matrix $H$ for $C$ is now of the form $(-P^T|I_{n-k})\Pi^T$, since $\Pi^{-1} = \Pi^T$.

This remark motivates the following definition.

**Definition 2.3.9** Let $I = \{i_1, \ldots, i_k\}$ be an information set of the code $C$. Then its complement $\{1, \ldots, n\} \setminus I$ is called a check set.

**Example 2.3.10** Consider the code $C$ of Example 2.2.23 with generator matrix $G$. The row reduced echelon form $G_1 = \text{rref}(G)$ is systematic at the positions 1, 2, 4 and 8. Let $\pi$ be the permutation (348765) with corresponding permutation matrix $\Pi$. Then $G_2 = G_1 \Pi = (I_4|P)$ and $H_2 = (P^T|I_4)$ with

$$G_2 = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}, \quad H_2 = \begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Now $\pi^{-1} = (356784)$ and

$$H = H_2 \Pi^T = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

is a parity check matrix of $C$.

### 2.3.2 Hamming and simplex codes

The following proposition gives a method to determine the minimum distance of a code in terms of the number of dependent columns of the parity check matrix.
2.3. PARITY CHECKS AND DUAL CODE

Proposition 2.3.11 Let $H$ be a parity check matrix of a code $C$. Then the minimum distance $d$ of $C$ is the smallest integer $d$ such that $d$ columns of $H$ are linearly dependent.

Proof. Let $h_1, \ldots, h_n$ be the columns of $H$. Let $c$ be a nonzero codeword of weight $w$. Let $\text{supp}(c) = \{j_1, \ldots, j_w\}$ with $1 \leq j_1 < \cdots < j_w \leq n$. Then $HC^T = 0$, so $c_{j_1}h_{j_1} + \cdots + c_{j_w}h_{j_w} = 0$ with $c_{j_i} \neq 0$ for all $i = 1, \ldots, w$. Therefore the columns $h_{j_1}, \ldots, h_{j_w}$ are dependent. Conversely if $h_{j_1}, \ldots, h_{j_w}$ are dependent, then there exist constants $a_1, \ldots, a_w$, not all zero, such that $a_1h_{j_1} + \cdots + a_wh_{j_w} = 0$. Let $c$ be the word defined by $c_j = 0$ if $j \neq j_i$ for all $i$, and $c_j = a_i$ if $j = j_i$ for some $i$. Then $HC^T = 0$. Hence $c$ is a nonzero codeword of weight at most $w$. \hfill \Box

Remark 2.3.12 Let $H$ be a parity check matrix of a code $C$. As a consequence of Proposition 2.3.11 we have the following special cases. The minimum distance of code is 1 if and only if $H$ has a zero column. An example of this is seen in Example 2.3.10. Now suppose that $H$ has no zero column, then the minimum distance of $C$ is at least 2. The minimum distance is equal to 2 if and only if $H$ has two columns say $h_{j_1}, h_{j_2}$ that are dependent. In the binary case that means $h_{j_1} = h_{j_2}$. In other words the minimum distance of a binary code is at least 3 if and only if $H$ has no zero columns and all columns are mutually distinct. This is the case for the Hamming code of Example 2.2.9. For a given redundancy $r$ the length of a binary linear code $C$ of minimum distance 3 is at most $2^r - 1$, the number of all nonzero binary columns of length $r$. For arbitrary $F_q$, the number of nonzero columns with entries in $F_q$ is $q^r - 1$. Two such columns are dependent if and only if one is a nonzero multiple of the other. Hence the length of an $F_q$-linear code code $C$ with $d(C) \geq 3$ and redundancy $r$ is at most $(q^r - 1)/(q - 1)$.

Definition 2.3.13 Let $n = (q^r - 1)/(q - 1)$. Let $H_r(q)$ be a $r \times n$ matrix over $F_q$ with nonzero columns, such that no two columns are dependent. The code $H_r(q)$ with $H_r(q)$ as parity check matrix is called a $q$-ary Hamming code. The code with $H_r(q)$ as generator matrix is called a $q$-ary simplex code and is denoted by $S_r(q)$.

Proposition 2.3.14 Let $r \geq 2$. Then the $q$-ary Hamming code $H_r(q)$ has parameters $[(q^r - 1)/(q - 1), (q^r - 1)/(q - 1) - r, 3]$.

Proof. The rank of the matrix $H_r(q)$ is $r$, since the $r$ standard basis vectors of weight 1 are among the columns of the matrix. So indeed $H_r(q)$ is a parity check matrix of a code with redundancy $r$. Any 2 columns are independent by construction. And a column of weight 2 is a linear combination of two columns of weight 1, and such a triple of columns exists, since $r \geq 2$. Hence the minimum distance is 3 by Proposition 2.3.11. \hfill \Box

Example 2.3.15 Consider the following ternary Hamming $H_3(3)$ code of redundancy 3 of length 13 with parity check matrix

$$H_3(3) = \begin{pmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
2 & 2 & 2 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
2 & 1 & 0 & 2 & 1 & 0 & 2 & 1 & 0 & 2 & 1 & 0 & 1
\end{pmatrix}.$$
By Proposition 2.3.14 the code $H_3(3)$ has parameters $[13, 10, 3]$. Notice that all rows of $H_3(3)$ have weight 9. In fact every linear combination $xH_3(3)$ with $x \in \mathbb{F}_3^3$ and $x \neq 0$ has weight 9. So all nonzero codewords of the ternary simplex code of dimension 3 have weight 9. Hence $S_3(3)$ is a constant weight code. This is a general fact of simplex codes as is stated in the following proposition.

**Proposition 2.3.16** The ary simplex code $S_r(q)$ is a constant weight code with parameters $[(q^r - 1)/(q - 1), r, q^{r - 1}]$.

**Proof.** We have seen already in Proposition 2.3.14 that $H_r(q)$ has rank $r$, so it is indeed a generator matrix of a code of dimension $r$. Let $c$ be a nonzero codeword of the simplex code. Then $c = mH_r(q)$ for some nonzero $m \in \mathbb{F}_q^r$. Let $h_j^T$ be the $j$-th column of $H_r(q)$. Then $c_j = 0$ if and only if $m \cdot h_j = 0$. Now $m \cdot x = 0$ is a nontrivial homogeneous linear equation. This equation has $q^{r - 1}$ solutions $x \in \mathbb{F}_q^r$, it has $q^{r - 1} - 1$ nonzero solutions. It has $(q^{r - 1} - 1)/(q - 1)$ solutions $x$ such that $x^T$ is a column of $H_r(q)$, since for every nonzero $x \in \mathbb{F}_q^r$ there is exactly one column in $H_r(q)$ that is a nonzero multiple of $x^T$. So the number of zeros of $c$ is $(q^{r - 1} - 1)/(q - 1)$. Hence the weight of $c$ is the number of nonzeros which is $q^{r - 1}$. $\diamond$

### 2.3.3 Inner product and dual codes

**Definition 2.3.17** The inner product on $\mathbb{F}_q^n$ is defined by

$$ x \cdot y = x_1y_1 + \cdots + x_ny_n $$

for $x, y \in \mathbb{F}_q^n$.

This inner product is bilinear, symmetric and nondegenerate, but the notion of “positive definite” makes no sense over a finite field as it does over the real numbers. For instance for a binary word $x \in \mathbb{F}_2^n$ we have that $x \cdot x = 0$ if and only if the weight of $x$ is even.

**Definition 2.3.18** For an $[n, k]$ code $C$ we define the dual or orthogonal code $C^\perp$ as

$$ C^\perp = \{ x \in \mathbb{F}_q^n \mid c \cdot x = 0 \quad \text{for all} \quad c \in C \}. $$

**Proposition 2.3.19** Let $C$ be an $[n, k]$ code with generator matrix $G$. Then $C^\perp$ is an $[n, n - k]$ code with parity check matrix $G$.

**Proof.** From the definition of dual codes, the following statements are equivalent:

$$ x \in C^\perp, $$

$$ c \cdot x = 0 \quad \text{for all} \quad c \in C, $$

$$ mGx^T = 0 \quad \text{for all} \quad m \in \mathbb{F}_q^k, $$

$$ Gx^T = 0. $$

This means that $C^\perp$ is the null space of $G$. Because $G$ is a $k \times n$ matrix of rank $k$, the linear space $C^\perp$ has dimension $n - k$ and $G$ is a parity check matrix of $C^\perp$. $\diamond$
2.3. **PARITY CHECKS AND DUAL CODE**

Example 2.3.20 The trivial codes \{0\} and \(\mathbb{F}_q^n\) are dual codes.

Example 2.3.21 The binary even weight code and the repetition code of the same length are dual codes.

Example 2.3.22 The simplex code \(S_r(q)\) and the Hamming code \(H_r(q)\) are dual codes, since \(H_r(q)\) is a parity check matrix of \(H_r(q)\) and a generator matrix of \(S_r(q)\).

A subspace \(C\) of a real vector space \(\mathbb{R}^n\) has the property that \(C \cap C^\perp = \{0\}\), since the standard inner product is positive definite. Over finite fields this is not always the case.

**Definition 2.3.23** Two codes \(C_1\) and \(C_2\) in \(\mathbb{F}_q^n\) are called **orthogonal** if \(x \cdot y = 0\) for all \(x \in C_1\) and \(y \in C_2\), and they are called **dual** if \(C_2 = C_1^\perp\). If \(C \subseteq C^\perp\), we call \(C\) **weakly self-dual** or **self-orthogonal**. If \(C = C^\perp\), we call \(C\) **self-dual**. The hull of a code \(C\) is defined by \(H(C) = C \cap C^\perp\). A code is called **complementary dual** if \(H(C) = \{0\}\).

Example 2.3.24 The binary repetition code of length \(n\) is self-orthogonal if and only if \(n\) is even. This code is self-dual if and only if \(n = 2\).

**Proposition 2.3.25** Let \(C\) be an \([n,k]\) code. Then:
(1) \((C^\perp)^\perp = C\).
(2) \(C\) is self-dual if and only \(C\) is self-orthogonal and \(n = 2k\).

**Proof.**
(1) Let \(c \in C\). Then \(c \cdot x = 0\) for all \(x \in C^\perp\). So \(C \subseteq (C^\perp)^\perp\). Moreover, applying Proposition 2.3.19 twice, we see that \(C\) and \((C^\perp)^\perp\) have the same finite dimension. Therefore equality holds.
(2) Suppose \(C\) is self-orthogonal, then \(C \subseteq C^\perp\). Now \(C = C^\perp\) if and only if \(k = n - k\), by Proposition 2.3.19. So \(C\) is self-dual if and only if \(n = 2k\). \(\diamond\)

Example 2.3.26 Consider

\[
G = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 1 & 1 & 1 & 0
\end{pmatrix}.
\]

Let \(G\) be the generator matrix of the binary \([8,4]\) code \(C\). Notice that \(GG^T = 0\). So \(x \cdot y = 0\) for all \(x, y \in C\). Hence \(C\) is self-orthogonal. Furthermore \(n = 2k\). Therefore \(C\) is self-dual. Notice that all rows of \(G\) have weight 4, therefore all codewords have weights divisible by 4 by Exercise 2.3.11. Hence \(C\) has parameters \([8,4,4]\).

**Remark 2.3.27** Notice that \(x \cdot x \equiv \text{wt}(x) \mod 2\) if \(x \in \mathbb{F}_2^n\) and \(x \cdot x \equiv \text{wt}(x) \mod 3\) if \(x \in \mathbb{F}_3^n\). Therefore all weights are even for a binary self-orthogonal code and all weights are divisible by 3 for a ternary self-orthogonal code.
Example 2.3.28 Consider the ternary code $C$ with generator matrix $G = (I_6 | A)$ with

$$A = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 2 & 2 & 1 \\ 1 & 1 & 0 & 1 & 2 & 2 \\ 1 & 2 & 1 & 0 & 1 & 2 \\ 1 & 2 & 2 & 1 & 0 & 1 \\ 1 & 1 & 2 & 2 & 1 & 0 \end{pmatrix}.$$  

It is left as an exercise to show that $C$ is self-dual. The linear combination of any two columns of $A$ has weight at least 3, and the linear combination of any two columns of $I_6$ has weight at most 2. So no three columns of $G$ are dependent and $G$ is also a parity check matrix of $C$. Hence the minimum distance of $C$ is at least 4, and therefore it is 6 by Remark 2.3.27. Thus $C$ has parameters $[12, 6, 6]$ and it is called the extended ternary Golay code. By puncturing $C$ we get a $[11, 6, 5]$ code and it is called the ternary Golay code.

Corollary 2.3.29 Let $C$ be a linear code. Then:

1. $G$ is generator matrix of $C$ if and only if $G$ is a parity check matrix of $C^\perp$.
2. $H$ is parity check matrix of $C$ if and only if $H$ is a generator matrix of $C^\perp$.

Proof. The first statement is Proposition 2.3.19 and the second statement is a consequence of the first applied to the code $C^\perp$ using Proposition 2.3.25(1).

Proposition 2.3.30 Let $C$ be an $[n, k]$ code. Let $G$ be a $k \times n$ generator matrix of $C$ and let $H$ be an $(n - k) \times n$ matrix of rank $n - k$. Then $H$ is a parity check matrix of $C$ if and only if $GH^T = 0$, the $k \times (n - k)$ zero matrix.

Proof. Suppose $H$ is a parity check matrix. For any $m \in \mathbb{F}_q^k$, $mG$ is a codeword of $C$. So, $HG^Tm^T = H(mG)^T = 0$. This implies that $mGH^T = 0$. Since $m$ can be any vector in $\mathbb{F}_q^k$, we have $GH^T = 0$.

Conversely, suppose $GH^T = 0$. We assumed that $G$ is a $k \times n$ matrix of rank $k$ and $H$ is an $(n - k) \times n$ matrix of rank $n - k$. So $H$ is the parity check matrix of an $[n, k]$ code $C'$. For any $c \in C$, we have $c = mG$ for some $m \in \mathbb{F}_q^k$. Now $Hc^T = (mGH^T)^T = 0$.

So $c \in C'$. This implies that $C \subseteq C'$. Hence $C' = C$, since both $C$ and $C'$ have dimension $k$. Therefore $H$ is a parity check matrix of $C$.

Remark 2.3.31 A consequence of Proposition 2.3.30 is another proof of Proposition 2.3.3 Because, let $G = (I_k | P)$ be a generator matrix of $C$. Let $H = (-P^T| I_{n-k})$. Then $G$ has rank $k$ and $H$ has rank $n - k$ and $GH^T = 0$. Therefore $H$ is a parity check matrix of $C$.

2.3.4 Exercises

2.3.1 Assume that 3540461335 is obtained from an ISBN code by interchanging two neighboring symbols. What are the possible ISBN codes? Now assume moreover that it is an ISBN code of an existing book. What is the title of this book?
2.3.2 Consider the binary product code $C$ of Example 2.1.2. Give a parity
check matrix and a generator matrix of this code. Determine the parameters of
the dual of $C$.

2.3.3 Give a parity check matrix of the $C$ of Exercise 2.2.4. Show that $C$ is
self-dual.

2.3.4 Consider the binary simplex code $S_3(2)$ with generator matrix $H$ as
given in Example 2.2.9. Show that there are exactly seven triples $(i_1, i_2, i_3)$ with
increasing coordinate positions such that $S_3(2)$ is not systematic at $(i_1, i_2, i_3)$.
Give the seven four-tuples of positions that are not systematic with respect to
the Hamming code $H_3(2)$ with parity check matrix $H$.

2.3.5 Let $C_1$ and $C_2$ be linear codes of the same length. Show the following
statements:
(1) If $C_1 \subseteq C_2$, then $C_2^\perp \subseteq C_1^\perp$.
(2) $C_1$ and $C_2$ are orthogonal if and only if $C_1 \subseteq C_2^\perp$ if and only if $C_2 \subseteq C_1^\perp$.
(3) $(C_1 \cap C_2)^\perp = C_1^\perp + C_2^\perp$.
(4) $(C_1 + C_2)^\perp = C_1^\perp \cap C_2^\perp$.

2.3.6 Show that a linear code $C$ with generator matrix $G$ has a complementary
dual if and only if $\det(GG^T) \neq 0$.

2.3.7 Show that there exists a $[2k, k]$ self-dual code over $\mathbb{F}_q$ if and only if there
is a $k \times k$ matrix $P$ with entries in $\mathbb{F}_q$ such that $PP^T = -I_k$.

2.3.8 Give an example of a ternary $[4,2]$ self-dual code and show that there is
no ternary self-dual code of length 6.

2.3.9 Show that the extended ternary Golay code in Example 2.3.28 is self-
dual.

2.3.10 Show that a binary code is self-orthogonal if the weights of all code-
words are divisible by 4. Hint: use Exercise 2.2.2.

2.3.11 Let $C$ be a binary self-orthogonal code which has a generator matrix
such that all its rows have a weight divisible by 4. Then the weights of all
codewords are divisible by 4.

2.3.12 Write a procedure either in GAP or Magma that determines whether
the given code is self-dual or not. Test correctness of your procedure with
commands IsSelfDualCode and IsSelfDual in GAP and Magma respectively.

2.4 Decoding and the error probability

Intro
2.4.1 Decoding problem

Definition 2.4.1 Let $C$ be a linear code in $\mathbb{F}_q^n$ of minimum distance $d$. If $c$ is a transmitted codeword and $r$ is the received word, then $\{i| r_i \neq c_i\}$ is the set of error positions and the number of error positions is called the number of errors of the received word. Let $e = r - c$. Then $e$ is called the error vector and $r = c + e$. Hence $\text{supp}(e)$ is the set of error positions and $\text{wt}(e)$ the number of errors. The $c_i$’s are called the error values.

Remark 2.4.2 If $r$ is the received word and $t' = d(C, r)$ is the distance of $r$ to the code $C$, then there exists a nearest codeword $c'$ such that $t' = d(c', r)$. So there exists an error vector $e'$ such that $r = c' + e'$ and $\text{wt}(e') = t'$. If the number of errors $t$ is at most $(d - 1)/2$, then we are sure that $c = c'$ and $e = e'$. In other words, the nearest codeword to $r$ is unique when $r$ has distance at most $(d - 1)/2$ to $C$.

Definition 2.4.3 $e(C) = [(d(C) - 1)/2]$ is called the error-correcting capacity decoding radius of the code $C$.

Definition 2.4.4 A decoder $D$ for the code $C$ is a map

$$D : \mathbb{F}_q^n \rightarrow \mathbb{F}_q^n \cup \{*\}$$

such that $D(c) = c$ for all $c \in C$. If $E : \mathbb{F}_q^k \rightarrow \mathbb{F}_q^n$ is an encoder of $C$ and $D : \mathbb{F}_q^n \rightarrow \mathbb{F}_q^n \cup \{*\}$ is a map such that $D(E(m)) = m$ for all $m \in \mathbb{F}_q^k$, then $D$ is called a decoder with respect to the encoder $E$.

Remark 2.4.5 If $E$ is an encoder of $C$ and $D$ is a decoder with respect to $E$, then the composition $E \circ D$ is a decoder of $C$. It is allowed that the decoder gives as outcome the symbol $*$ in case it fails to find a codeword. This is called a decoding failure. If $c$ is the codeword sent and $r$ is the received word and $D(r) = c' \neq c$, then this is called a decoding error. If $D(r) = c$, then $r$ is decoded correctly. Notice that a decoding failure is noted on the receiving end, whereas there is no way that the decoder can detect a decoding error.

Definition 2.4.6 A complete decoder is a decoder that always gives a codeword in $C$ as outcome. A nearest neighbor decoder, also called a minimum distance decoder, is a complete decoder with the property that $D(r)$ is a nearest codeword. A decoder $D$ for a code $C$ is called a $t$-bounded distance decoder or a decoder that corrects $t$ errors if $D(r)$ is a nearest codeword for all received words $r$ with $d(C, r) \leq t$ errors. A decoder for a code $C$ with error-correcting capacity $e(C)$ decodes up to half the minimum distance if it is an $e(C)$-bounded distance decoder, where $e(C) = [(d(C) - 1)/2]$ is the error-correcting capacity of $C$.

Remark 2.4.7 If $D$ is a $t$-bounded distance decoder, then it is not required that $D$ gives a decoding failure as outcome for a received word $r$ if the distance of $r$ to the code is strictly larger than $t$. In other words: $D$ is also a $t'$-bounded distance decoder for all $t' \leq t$.

A nearest neighbor decoder is a $t$-bounded distance decoder for all $t \leq \rho(C)$, where $\rho(C)$ is the covering radius of the code. A $\rho(C)$-bounded distance decoder is a nearest neighbor decoder, since $d(C, r) \leq \rho(C)$ for all received words $r$. 

34

CHAPTER 2. ERROR-CORRECTING CODES
Definition 2.4.8 Let \( r \) be a received word with respect to a code \( C \). A coset leader of \( r + C \) is a choice of an element of minimal weight in the coset \( r + C \). The weight of a coset is the minimal weight of an element in the coset. Let \( \alpha_i \) be the number of cosets of \( C \) that are of weight \( i \). Then \( \alpha_C(X,Y) \), the coset leader weight enumerator of \( C \) is the polynomial defined by

\[
\alpha_C(X,Y) = \sum_{i=0}^{n} \alpha_i X^{n-i}Y^i.
\]

Remark 2.4.9 The choice of a coset leader of the coset \( r + C \) is unique if \( d(C,r) \leq (d-1)/2 \), and \( \alpha_i = \binom{n}{i}(q-1)^i \) for all \( i \leq (d-1)/2 \), where \( d \) is the minimum distance of \( C \). Let \( \rho(C) \) be the covering radius of the code, then there is at least one codeword \( c \) such that \( d(c,r) \leq \rho(C) \). Hence the weight of a coset leader is at most \( \rho(C) \) and \( \alpha_i = 0 \) for \( i > \rho(C) \). Therefore the coset leader weight enumerator of a perfect code \( C \) of minimum distance \( d = 2t+1 \) is given by

\[
\alpha_C(X,Y) = \sum_{i=0}^{t} \binom{n}{i}(q-1)^i X^{n-i}Y^i.
\]

The computation of the coset leader weight enumerator of a code is in general a very hard problem.

Definition 2.4.10 Let \( r \) be a received word. Let \( e \) be the chosen coset leader of the coset \( r + C \). The coset leader decoder gives \( r - e \) as output.

Remark 2.4.11 The coset leader decoder is a nearest neighbor decoder.

Definition 2.4.12 Let \( r \) be a received word with respect to a code \( C \) of dimension \( k \). Choose an \( (n-k) \times n \) parity check matrix \( H \) of the code \( C \). Then \( s = RH^T \in \mathbb{F}_q^{n-k} \) is called the syndrome of \( r \) with respect to \( H \).

Remark 2.4.13 Let \( C \) be a code of dimension \( k \). Let \( r \) be a received word. Then \( r + C \) is called the coset of \( r \). Now the cosets of the received words \( r_1 \) and \( r_2 \) are the same if and only if \( r_1H^T = r_2H^T \). Therefore there is a one to one correspondence between cosets of \( C \) and values of syndromes. Furthermore every element of \( \mathbb{F}_q^{n-k} \) is the syndrome of some received word \( r \), since \( H \) has rank \( n-k \). Hence the number of cosets is \( q^{n-k} \).

A list decoder gives as output the collection of all nearest codewords.

Knowing the existence of a decoder is nice to know from a theoretical point of view, in practice the problem is to find an efficient algorithm that computes the outcome of the decoder. To compute of a given vector in Euclidean \( n \)-space the closest vector to a given linear subspace can be done efficiently by an orthogonal projection to the subspace. The corresponding problem for linear codes is in general not such an easy task. This is treated in Section 6.2.1.

### 2.4.2 Symmetric channel
Definition 2.4.14 The \( q \)-ary symmetric channel (\( q \)SC) is a channel where \( q \)-ary words are sent with independent errors with the same cross-over probability \( p \) at each coordinate, with \( 0 \leq p \leq \frac{1}{2} \), such that all the \( q - 1 \) wrong symbols occur with the same probability \( p/(q-1) \). So a symbol is transmitted correctly with probability \( 1 - p \). The special case \( q = 2 \) is called the binary symmetric channel (BSC).

Remark 2.4.15 Let \( P(x) \) be the probability that the codeword \( x \) is sent. Then this probability is assumed to be the same for all codewords. Hence \( P(c) = \frac{1}{|C|} \) for all \( c \in C \). Let \( P(r|c) \) be the probability that \( r \) is received given that \( c \) is sent. Then

\[
P(r|c) = \left( \frac{p}{q-1} \right)^{d(c,r)} (1-p)^{n-d(c,r)}
\]

for a \( q \)-ary symmetric channel.

Definition 2.4.16 For every decoding scheme and channel one defines three probabilities \( P_{cd}(p) \), \( P_{de}(p) \) and \( P_{df}(p) \), that is the probability of correct decoding, decoding error and decoding failure, respectively. Then

\[
P_{cd}(p) + P_{de}(p) + P_{df}(p) = 1 \quad \text{for all} \quad 0 \leq p \leq \frac{1}{2}.
\]

So it suffices to find formulas for two of these three probabilities. The error probability, also called the error rate is defined by \( P_{err}(p) = 1 - P_{cd}(p) \). Hence

\[
P_{err}(p) = P_{de}(p) + P_{df}(p).
\]

Proposition 2.4.17 The probability of correct decoding of a decoder that corrects up to \( t \) errors with \( 2t + 1 \leq d \) of a code \( C \) of minimum distance \( d \) on a \( q \)-ary symmetric channel with cross-over probability \( p \) is given by

\[
P_{cd}(p) = \sum_{w=0}^{t} \binom{n}{w} p^w (1-p)^{n-w}.
\]

Proof. Every codeword has the same probability of transmission. So

\[
P_{cd}(p) = \sum_{c \in C} P(c) \sum_{d(c,r) \leq t} P(y|r) = \frac{1}{|C|} \sum_{c \in C} \sum_{d(c,r) \leq t} P(r|c),
\]

Now \( P(r|c) \) depends only on the distance between \( r \) and \( c \) by Remark 2.4.15. So without loss of generality we may assume that \( 0 \) is the codeword sent. Hence

\[
P_{cd}(p) = \sum_{d(0,r) \leq t} P(r|0) = \sum_{w=0}^{t} \binom{n}{w} (q-1)^w \left( \frac{p}{q-1} \right)^w (1-p)^{n-w}
\]

by Proposition 2.1.13. Clearing the factor \((q-1)^w\) in the numerator and the denominator gives the desired result. \( \Box \)

In Proposition 4.2.6 a formula will be derived for the probability of decoding error for a decoding algorithm that corrects errors up to half the minimum distance.
Example 2.4.18 Consider the binary triple repetition code. Assume that 
\((0,0,0)\) is transmitted. In case the received word has weight 0 or 1, then it 
is correctly decoded to \((0,0,0)\). If the received word has weight 2 or 3, then it 
is decoded to \((1,1,1)\) which is a decoding error. Hence there are no decoding 
failures and 
\[ P_{cd}(p) = (1 - p)^3 + 3p(1 - p)^2 = 1 - 3p^2 + 2p^3 \] and 
\[ P_{err}(p) = P_{de}(p) = 3p^2 - 2p^3. \]
If the Hamming code is used, then there are no decoding failures and 
\[ P_{cd}(p) = (1 - p)^7 + 7p(1 - p)^6 \] and 
\[ P_{err}(p) = P_{de}(p) = 21p^2 - 70p^3 + 105p^4 - 84p^5 + 35p^6 - 6p^7. \]
This shows that the error probabilities of the repetition code is smaller than 
the one for the Hamming code. This comparison is not fair, since only one bit 
of information is transmitted with the repetition code and four bits with the 
Hamming code. One could transmit 4 bits of information by using the repetition 
code four times. This would give the error probability 
\[ 1 - (1 - 3p^2 + 2p^3)^4 = 12p^2 - 8p^3 - 54p^4 + 72p^5 + 84p^6 - 216p^7 + \cdots \]

Example 2.4.19 Consider the binary \(n\)-fold repetition code. Let \(t = (n-1)/2\). 
Use the decoding algorithm correcting all patterns of \(t\) errors. Then 
\[ P_{err}(p) = \sum_{i=t+1}^{n} \binom{n}{i} p^i (1 - p)^{n-i}. \]
Hence the error probability becomes arbitrarily small for increasing \(n\). The price 
one has to pay is that the information rate \(R = 1/n\) tends to 0. The remarkable 
result of Shannon states that for a fixed rate \(R < C(p)\), where 
\[ C(p) = 1 + p \log_2(p) + (1 - p) \log_2(1 - p) \] is the capacity of the binary symmetric channel, one can devise encoding and 
decoding schemes such that \(P_{err}(p)\) becomes arbitrarily small. This will be 
treated in Theorem 4.2.9.

The main problem of error-correcting codes from “Shannon’s point view” is to 
construct efficient encoding and decoding algorithms of codes with the smallest 
error probability for a given information rate and cross-over probability.

Proposition 2.4.20 The probability of correct decoding of the coset leader 
decoder on a \(q\)-ary symmetric channel with cross-over probability \(p\) is given by 
\[ P_{cd}(p) = \alpha C \left( 1 - p, \frac{p}{q-1} \right). \]
CHAPTER 2. ERROR-CORRECTING CODES

Proof. This is left as an exercise.

Example 2.4.21

2.4.3 Exercises

2.4.1 Consider the binary repetition code of length \( n \). Compute the probabilities of correct decoding, decoding error and decoding failure in case of incomplete decoding \( t = \lfloor (n - 1)/2 \rfloor \) errors and complete decoding by choosing one nearest neighbor.

2.4.2 Consider the product code of Example 2.1.2. Compute the probabilities of correct decoding, decoding error and decoding failure in case the decoding algorithm corrects all error patterns of at most \( t \) errors for \( t = 1, t = 2 \) and \( t = 3 \), respectively.

2.4.3 Give a proof of Proposition 2.4.20.

2.4.4 ***Give the probability of correct decoding for the code .... for a coset leader decoder. ***

2.4.5 ***Product code has error probability at most \( P_1(P_2(p)) \).***

2.5 Equivalent codes

Notice that a Hamming code over \( \mathbb{F}_q \) of a given redundancy \( r \) is defined up to the order of the columns of the parity check matrix and up to multiplying a column with a nonzero constant. A permutation of the columns and multiplying the columns with nonzero constants gives another code with the same parameters and is in a certain sense equivalent.

2.5.1 Number of generator matrices and codes

The set of all invertible \( n \times n \) matrices over the finite field \( \mathbb{F}_q \) is denoted by \( \text{Gl}(n,q) \). Now \( \text{Gl}(n,q) \) is a finite group with respect to matrix multiplication and it is called the general linear group.

Proposition 2.5.1 The number of elements of \( \text{Gl}(n,q) \) is

\[(q^n - 1)(q^n - q) \cdots (q^n - q^{n-1}).\]

Proof. Let \( M \) be an \( n \times n \) matrix with rows \( m_1, \ldots, m_n \). Then \( M \) is invertible if and only if \( m_1, \ldots, m_n \) are independent and that is if and only if \( m_1 \neq 0 \) and \( m_i \) is not in the linear subspace generated by \( m_1, \ldots, m_{i-1} \) for all \( i = 2, \ldots, n \). Hence for an invertible matrix \( M \) we are free to choose a nonzero vector for the first row. There are \( q^n - 1 \) possibilities for the first row. The second row should not be a multiple of the first row, so we have \( q^n - q \) possibilities for the second row for every nonzero choice of the first row. The subspace generated by \( m_1, \ldots, m_{i-1} \) has dimension \( i - 1 \) and \( q^{i-1} \) elements. The \( i \)-th row is not in this subspace if \( M \) is invertible. So we have \( q^n - q^{i-1} \) possible choices for the \( i \)-th row for every legitimate choice of the first \( i - 1 \) rows. This proves the claim.
Proposition 2.5.2
1) The number of \( k \times n \) generator matrices over \( \mathbb{F}_q \) is
\[
(q^n - 1)(q^n - q) \cdots (q^n - q^{k-1}).
\]
2) The number of \([n, k]\) codes over \( \mathbb{F}_q \) is equal to the Gaussian binomial
\[
\left[ \begin{array}{c} n \\ k \end{array} \right]_q := \frac{(q^n - 1)(q^n - q) \cdots (q^n - q^{k-1})}{(q^k - 1)(q^k - q) \cdots (q^k - q^{k-1})}.
\]

Proof.
1) A \( k \times n \) generator matrix consists of \( k \) independent rows of length \( n \) over \( \mathbb{F}_q \).

The counting of the number of these matrices is done similarly as in the proof of Proposition 2.5.1.

2) The second statement is a consequence of Propositions 2.5.1 and 2.2.17, and the fact the \( MG = G \) if and only if \( M = I_k \) for every \( M \in \text{Gl}(k, q) \) and \( k \times n \) generator matrix \( G \), since \( G \) has rank \( k \).

It is a consequence of Proposition 2.5.2 that the Gaussian binomials are integers for every choice of \( n, k \) and \( q \). In fact more is true.

Proposition 2.5.3
The number of \([n, k]\) codes over \( \mathbb{F}_q \) is a polynomial in \( q \) of degree \( k(n - k) \) with non-negative integers as coefficients.

Proof. There is another way to count the number of \([n, k]\) codes over \( \mathbb{F}_q \), since the row reduced echelon form \( \text{ref}(C) \) of a generator matrix of \( C \) is unique by Proposition 2.2.17. Now suppose that \( \text{ref}(C) \) has pivots at \( j = (j_1, \ldots, j_k) \) with \( 1 \leq j_1 < \cdots < j_k \leq n \), then the remaining entries are free to choose as long as the row reduced echelon form at the given pivots \( (j_1, \ldots, j_k) \) is respected. Let the number of these free entries be \( e(j) \). Then the number of \([n, k]\) codes over \( \mathbb{F}_q \) is equal to
\[
\sum_{1 \leq j_1 < \cdots < j_k \leq n} q^{e(j)}.
\]
Furthermore \( e(j) \) is maximal and equal to \( k(n - k) \) for \( j = (1, 2, \ldots, k) \) This is left as Exercise 2.5.2 to the reader.

Example 2.5.4 Let us compute the number of \([3, 2]\) codes over \( \mathbb{F}_q \). According to Proposition 2.5.2 it is equal to
\[
\left[ \begin{array}{c} 3 \\ 2 \end{array} \right]_q = \frac{(q^3 - 1)(q^3 - q)}{(q^2 - 1)(q^2 - q)} = q^2 + q + 1.
\]

which is a polynomial of degree \( 2 \cdot (3 - 2) = 2 \) with non-negative integers as coefficients. This is in agreement with Proposition 2.5.3. If we follow the proof of this proposition then the possible row echelon forms are
\[
\begin{pmatrix} 1 & 0 & * \\ 0 & 1 & * \end{pmatrix}, \quad \begin{pmatrix} 1 & * & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix},
\]
where the *’s denote the entries that are free to choose. So \( e(1, 2) = 2, e(1, 3) = 1 \) and \( e(2, 3) = 0 \). Hence the number of \([3, 2]\) codes is equal to \( q^2 + q + 1 \), as we have seen before.
2.5.2 Isometries and equivalent codes

Definition 2.5.5 Let $M \in \text{Gl}(n,q)$. Then the map

$$M : \mathbb{F}_q^n \rightarrow \mathbb{F}_q^n,$$

defined by $M(x) = xM$ is a one-to-one linear map. Notice that the map and the matrix are both denoted by $M$. Let $S$ be a subset of $\mathbb{F}_q^n$. The operation $xM$, where $x \in S$ and $M \in \text{Gl}(n,q)$, is called an action of the group $\text{Gl}(n,q)$ on $S$. For a given $M \in \text{Gl}(n,q)$, the set $SM = \{xM \mid x \in S\}$, also denoted by $M(S)$, is called the image of $S$ under $M$.

Definition 2.5.6 The group of permutations of $\{1, \ldots, n\}$ is called the symmetric group on $n$ letters and is denoted by $S_n$. Let $\pi \in S_n$. Define the corresponding permutation matrix $\Pi$ with entries $p_{ij}$ by $p_{ij} = 1$ if $i = \pi(j)$ and $p_{ij} = 0$ otherwise.

Remark 2.5.7 $S_n$ is indeed a group and has $n!$ elements. Let $\Pi$ be the permutation matrix of a permutation $\pi$ in $S_n$. Then $\Pi$ is invertible and orthogonal, that means $\Pi^\top = \Pi^{-1}$. The corresponding map $\Pi : \mathbb{F}_q^n \rightarrow \mathbb{F}_q^n$ is given by $\Pi(x) = y$ with $y_i = x_{\pi(i)}$ for all $i$. Now $\Pi$ is an invertible linear map. Let $e_i$ be the $i$-th standard basis row vector. Then $\Pi^{-1}(e_i) = e_{\pi(i)}$ by the above conventions. The set of $n \times n$ permutation matrices is a subgroup of $\text{Gl}(n,q)$ with $n!$ elements.

Definition 2.5.8 Let $v \in \mathbb{F}_q^n$. Then $\text{diag}(v)$ is the $n \times n$ diagonal matrix with $v$ on its diagonal and zeros outside the diagonal. An $n \times n$ matrix with entries in $\mathbb{F}_q$ is called monomial if every row has exactly one non-zero entry and every column has exactly one non-zero entry. Let $\text{Mono}(n,q)$ be the set of all $n \times n$ monomial matrices with entries in $\mathbb{F}_q$.

Remark 2.5.9 The matrix $\text{diag}(v)$ is invertible if and only if every entry of $v$ is not zero. Hence the set of $n \times n$ invertible diagonal matrices is a subgroup of $\text{Gl}(n,q)$ with $(q-1)^n$ elements.

Let $M$ be an element of $\text{Mono}(n,q)$. Define the vector $v \in \mathbb{F}_q^n$ with nonzero entries and the map $\pi$ from $\{1, \ldots, n\}$ to itself by $\pi(j) = i$ if $v_i$ is the unique nonzero entry of $M$ in the $i$-th row and the $j$-th column. Now $\pi$ is a permutation by the definition of a monomial matrix. So $M$ has entries $m_{ij}$ with $m_{ij} = v_i$ if $i = \pi(j)$ and $m_{ij} = 0$ otherwise. Hence $M = \text{diag}(v)\Pi$. Therefore a matrix is monomial if and only if it is the product of a diagonal and a permutation matrix. The corresponding monomial map $M : \mathbb{F}_q^n \rightarrow \mathbb{F}_q^n$ of the monomial matrix $M$ is given by $M(x) = y$ with $y_i = v_{i,\pi(i)}$. The set of $\text{Mono}(n,q)$ is a subgroup of $\text{Gl}(n,q)$ with $(q-1)^n$ elements.

Definition 2.5.10 A map $\varphi : \mathbb{F}_q^n \rightarrow \mathbb{F}_q^n$ is called an isometry if it leaves the Hamming metric invariant, that means that

$$d(\varphi(x), \varphi(y)) = d(x, y)$$

for all $x, y \in \mathbb{F}_q^n$. Let $\text{Isom}(n,q)$ be the set of all isometries of $\mathbb{F}_q^n$.

Proposition 2.5.11 $\text{Isom}(n,q)$ is a group under the composition of maps.
2.5. EQUIVALENT CODES

Proof. The identity map is an isometry. Let \( \varphi \) and \( \psi \) be isometries of \( \mathbb{F}_q^n \). Let \( x, y \in \mathbb{F}_q^n \). Then

\[
d((\varphi \circ \psi)(x), (\varphi \circ \psi)(y)) = d(\varphi(\psi(x)), \varphi(\psi(y))) = d(\psi(x), \psi(y)) = d(x, y).
\]

Hence \( \varphi \circ \psi \) is an isometry.

Let \( \varphi \) be an isometry of \( \mathbb{F}_q^n \). Suppose that \( x, y \in \mathbb{F}_q^n \) and \( \varphi(x) = \varphi(y) \). Then

\[
0 = d(\varphi(x), \varphi(y)) = d(x, y).
\]

So \( x = y \). Hence \( \varphi \) is bijective. Therefore it has an inverse map \( \varphi^{-1} \).

Let \( x, y \in \mathbb{F}_q^n \). Then

\[
d(x, y) = d(\varphi^{-1}(x), \varphi^{-1}(y)) = d(\varphi^{-1}(x), \varphi^{-1}(y)),
\]

since \( \varphi \) is an isometry. Therefore \( \varphi^{-1} \) is an isometry.

So \( \text{Isom}(n, q) \) is not-empty and closed under taking the composition of maps and taking the inverse. Therefore \( \text{Isom}(n, q) \) is a group. \( \diamond \)

Remark 2.5.12 Permutation matrices define isometries. Translations and invertible diagonal matrices and more generally the coordinatewise permutation of the elements of \( \mathbb{F}_q \) define also isometries. Conversely, every isometry is the composition of the before mentioned isometries. This fact we leave as Exercise 2.5.4. The following proposition characterizes linear isometries.

Proposition 2.5.13 Let \( M \in \text{Gl}(n, q) \). Then the following statements are equivalent:

(1) \( M \) is an isometry,
(2) \( \text{wt}(M(x)) = \text{wt}(x) \) for all \( x \in \mathbb{F}_q^n \), so \( M \) leaves the weight invariant,
(3) \( M \) is a monomial matrix.

Proof. Statements (1) and (2) are equivalent, since \( M(x - y) = M(x) - M(y) \) and \( d(x, y) = \text{wt}(x - y) \).

Statement (3) implies (1), since permutation matrices and invertible diagonal matrices leave the weight of a vector invariant, and a monomial matrix is a product of such matrices by Remark 2.5.9.

Statement (2) implies (3): Let \( e_i \) be the \( i \)-th standard basis vector of \( \mathbb{F}_q^n \). Then \( e_i \) has weight 1. So \( M(e_i) \) has also weight 1. Hence \( M(e_i) = v_i e_{\pi(i)} \), where \( v_i \) is a nonzero element of \( \mathbb{F}_q \), and \( \pi \) is a map from \( \{1, \ldots, n\} \) to itself. Now \( \pi \) is a bijection, since \( M \) is invertible. So \( \pi \) is a permutation and \( M = \text{diag}(v)\Pi^{-1} \).

Therefore \( M \) is a monomial matrix. \( \diamond \)

Corollary 2.5.14 An isometry is linear if and only if it is a map coming from a monomial matrix, that is

\[
\text{Gl}(n, q) \cap \text{Isom}(n, q) = \text{Mono}(n, q).
\]

Proof. This follows directly from the definitions and Proposition 2.5.13. \( \diamond \)

Definition 2.5.15 Let \( C \) and \( D \) be codes in \( \mathbb{F}_q^n \) that are not necessarily linear. Then \( C \) is called equivalent to \( D \) if there exists an isometry \( \varphi \) of \( \mathbb{F}_q^n \) such that \( \varphi(C) = D \). If moreover \( C = D \), then \( \varphi \) is called an automorphism of \( C \). The
automorphism group of $C$ is the set of all isometries $\varphi$ such that $\varphi(C) = C$ and is denoted by $\text{Aut}(C)$. 

$C$ is called permutation equivalent to $D$, and is denoted by $D \equiv C$ if there exists a permutation matrix $\Pi$ such that $\Pi(C) = D$. If moreover $C = D$, then $\Pi$ is called a permutation automorphism of $C$. The permutation automorphism group of $C$ is the set of all permutation automorphism of $C$ and is denoted by $\text{PAut}(C)$. 

$C$ is called generalized equivalent or monomial equivalent to $D$, denoted by $D \cong C$ if there exists a monomial matrix $M$ such that $M(C) = D$. If moreover $C = D$, then $M$ is called a monomial automorphism of $C$. The monomial automorphism group of $C$ is the set of all monomial automorphism of $C$ and is denoted by $\text{MAut}(C)$.

**Proposition 2.5.16** Let $C$ and $D$ be two $F_q$-linear codes of the same length. Then:

1. If $C \equiv D$, then $C^\perp \equiv D^\perp$.
2. If $C \cong D$, then $C^\perp \cong D^\perp$.
3. If $C \equiv D$, then $C \cong D$.
4. If $C \cong D$, then $C$ and $D$ have the same parameters.

**Proof.** We leave the proof to the reader as an exercise. $\diamond$

**Remark 2.5.17** Every $[n,k]$ code is equivalent to a code which is systematic at the first $k$ positions, that is with a generator matrix of the form $(I_k|P)$ according to Remark 2.3.8.

Notice that in the binary case $C \equiv D$ if and only if $C \cong D$.

**Example 2.5.18** Let $C$ be a binary $[7,4,3]$ code with parity check matrix $H$. Then $H$ is a $3 \times 7$ matrix such that all columns are nonzero and mutually distinct by Proposition 2.3.11, since $C$ has minimum distance 3. There are exactly 7 binary nonzero column vectors with 3 entries. Hence $H$ is a permutation of the columns of a parity check matrix of the $[7,4,3]$ Hamming code. Therefore: every binary $[7,4,3]$ code is permutation equivalent with the Hamming code.

**Proposition 2.5.19**

1. Every $F_q$-linear code with parameters $[(q^r-1)/(q-1), (q^r-1)/(q-1)-r, 3]$ is generalized equivalent with the Hamming code $H_r(q)$.
2. Every $F_q$-linear code with parameters $[(q^r-1)/(q-1), r, q^{r-1}]$ is generalized equivalent with the simplex code $S_r(q)$.

**Proof.** (1) Let $n = (q^r-1)/(q-1)$. Then $n$ is the number of lines in $F_q^n$ through the origin. Let $H$ be a parity check matrix of an $F_q$-linear code with parameters $[n, n-r, 3]$. Then there are no zero columns in $H$ and every two columns are independent by Proposition 2.3.11. Every column of $H$ generates a unique line in $F_q^n$ through the origin, and every such line is obtained in this way. Let $H'$ be the parity check matrix of a code $C'$ with the same parameters $[n, n-r, 3]$. Then for every column $h_i'$ of $H'$ there is a unique column $h_i$ of $H$ such that $h_i'$ is nonzero multiple of $h_i$. Hence $H' \equiv HM$ for some monomial matrix $M$. Hence $C$ and $C'$ are generalized equivalent.

(2) The second statement follows form the first one, since the simplex code is the dual of the Hamming code. $\diamond$
2.5. EQUIVALENT CODES

Remark 2.5.20 A code of length \( n \) is called cyclic if the cyclic permutation of coordinates \( \sigma(i) = i - 1 \) modulo \( n \) leaves the code invariant. A cyclic code of length \( n \) has an element of order \( n \) in its automorphism group. Cyclic codes are extensively treated in Chapter 7.1.

Remark 2.5.21 Let \( C \) be an \( \mathbb{F}_q \)-linear code of length \( n \). Then \( \text{PAut}(C) \) is a subgroup of \( S_n \) and \( \text{MAut}(C) \) is a subgroup of \( \text{Mono}(n, q) \). If \( C \) is a trivial code, then \( \text{PAut}(C) = S_n \) and \( \text{MAut}(C) = \text{Mono}(n, q) \). The matrices \( \lambda I_n \in \text{MAut}(C) \) for all nonzero \( \lambda \in \mathbb{F}_q \). So \( \text{MAut}(C) \) always contains \( \mathbb{F}_q^* \) as a subgroup. Furthermore \( \text{Mono}(n, q) = S_n \) and \( \text{MAut}(C) = \text{PAut}(C) \) if \( q = 2 \).

Example 2.5.22 Let \( C \) be the \( n \)-fold repetition code. Then \( \text{PAut}(C) = S_n \) and \( \text{MAut}(C) \) isomorphic with \( \mathbb{F}_q^* \times S_n \).

Proposition 2.5.23 Let \( G \) be a generator matrix of an \( \mathbb{F}_q \)-linear code \( C \) of length \( n \). Let \( \Pi \) be an \( n \times n \) permutation matrix. Let \( M \in \text{Mono}(n, q) \). Then:

1. \( \Pi \in \text{PAut}(C) \) if and only if \( \text{rref}(G) = \text{rref}(G \Pi) \),
2. \( M \in \text{MAut}(C) \) if and only if \( \text{rref}(G) = \text{rref}(GM) \).

Proof. (1) Let \( \Pi \) be a \( n \times n \) permutation matrix. Then \( G \Pi \) is a generator matrix of \( \Pi(C) \). Moreover \( \Pi(C) = C \) if and only if \( \text{rref}(G) = \text{rref}(G \Pi) \) by Proposition 2.2.17.

(2) The second statement is proved similarly. ∗

Example 2.5.24 Let \( C \) be the code with generator matrix \( G \) and let \( M \) be the monomial matrix given by

\[
G = \begin{pmatrix}
1 & 0 & a_1 \\
0 & 1 & a_2
\end{pmatrix}
\text{ and } M = \begin{pmatrix}
0 & x_2 & 0 \\
x_1 & 0 & 0 \\
0 & 0 & x_3
\end{pmatrix},
\]

where the \( a_i \) and \( x_j \) are nonzero elements of \( \mathbb{F}_q \). Now \( G \) is already in reduced row echelon form. One verifies that

\[
\text{rref}(GM) = \begin{pmatrix}
1 & 0 & a_2x_3/x_1 \\
0 & 1 & a_1x_3/x_2
\end{pmatrix}.
\]

Hence \( M \) is monomial automorphism of \( C \) if and only if \( a_1x_1 = a_2x_3 \) and \( a_2x_2 = a_1x_3 \).

Definition 2.5.25 A map \( f \) from the set of all (linear) codes to another set is called an invariant of a (linear) code if \( f(C) = f(\varphi(C)) \) for every code \( C \) in \( \mathbb{F}_q^n \) and every isometry \( \varphi \) of \( \mathbb{F}_q^n \). The map \( f \) is called a permutation invariant if \( f(C) = f(\Pi(C)) \) for every code \( C \) in \( \mathbb{F}_q^n \) and every \( n \times n \) permutation matrix \( \Pi \). The map \( f \) is called a monomial invariant if \( f(C) = f(M(C)) \) for every code \( C \) in \( \mathbb{F}_q^n \) and every \( M \in \text{Mono}(n, q) \).

Remark 2.5.26 The length, the number of elements and the minimum distance are clearly invariants of a code. The dimension is a permutation and a monomial invariant of a linear code. The isomorphy class of the group of automorphisms of a code is an invariant of a code. The isomorphy classes of \( \text{PAut}(C) \) and \( \text{MAut}(C) \) are permutation and monomial invariants, respectively of a linear code.
2.5.3 Exercises

2.5.1 Determine the number of \([5, 3]\) codes over \(\mathbb{F}_q\) by Proposition 2.5.2 and show by division that it is a polynomial in \(q\). Determine the exponent \(e(j)\) and the number of codes such that \(\text{rref}(C)\) is systematic at a given 3-tuple \((j_1, j_2, j_3)\) for all 3-tuples with \(1 \leq j_1 < j_2 < j_3 \leq 5\), as in Proposition 2.5.3, and verify that they sum up to the total number of \([5, 3]\) codes.

2.5.2 Show that \(e(j) = \sum_{t=1}^{k} t(j_t+1-j_t-1)\) for every \(k\)-tuple \((j_1, \ldots, j_k)\) with \(1 \leq j_1 < \cdots < j_k \leq n\) and \(j_k+1 = n+1\) in the proof of Proposition 2.5.3. Show that the maximum of \(e(j)\) is equal to \(k(n-k)\) and that this maximum is attained by exactly one \(k\)-tuple that is by \((1, 2, \ldots, k)\).

2.5.3 Let \(p\) be a prime. Let \(q = p^m\). Consider the map \(\varphi : \mathbb{F}_q^n \rightarrow \mathbb{F}_q^n\) defined by \(\varphi(x_1, \ldots, x_n) = (x_1^p, \ldots, x_n^p)\). Show that \(\varphi\) is an isometry that permutes the elements of the alphabet \(\mathbb{F}_q\) coordinatewise. Prove that \(\varphi\) is a linear map if and only if \(m = 1\). Show that \(\varphi(C)\) is a linear code if \(C\) is a linear code.

2.5.4 Show that permutation matrices and the coordinatewise permutation of the elements of \(\mathbb{F}_q\) define isometries. Show that every element of \(\text{Isom}(n, q)\) is the composition of a permutation matrix and the coordinatewise permutation of the elements of \(\mathbb{F}_q\). Moreover, such a composition is unique. Show that the number of elements of \(\text{Isom}(n, q)\) is equal to \(n!(q!)^n\).

2.5.5 Give a proof of Proposition 2.5.16.

2.5.6 Show that every binary \((7, 16, 3)\) code is isometric with the Hamming code.

2.5.7 Let \(C\) be a linear code of length \(n\). Assume that \(n\) is a power of a prime. Show that if there exists an element in \(\text{PAut}(C)\) of order \(n\), then \(C\) is equivalent with a cyclic code. Show that the assumption on \(n\) being a prime power is necessary by means of a counterexample.

2.5.8 A code \(C\) is called \(\text{quasi self-dual}\) if it is monomial equivalent with its dual. Consider the \([2k, k]\) code over \(\mathbb{F}_q\) with generator matrix \((I_k | I_k)\). Show that this code quasi self-dual for all \(q\) and self-dual if \(q\) is even.

2.5.9 Let \(C\) be an \(\mathbb{F}_q\)-linear code of length \(n\) with hull \(H(C) = C \cap C^\perp\). Let \(\Pi\) be an \(n \times n\) permutation matrix. Let \(D\) be an invertible \(n \times n\) diagonal matrix. Let \(M \in \text{Mono}(n, q)\).
(1) Show that \((\Pi(C))^\perp = \Pi(C^\perp)\).
(2) Show that \(H(\Pi(C)) = \Pi(H(C))\).
(3) Show that \((D(C))^\perp = D^{-1}(C^\perp)\).
(4) Show that \(H(M(C)) = M(H(C))\) if \(q = 2\) or \(q = 3\).
(5) Show by means of a counter example that the dimension of the hull of a linear code over \(\mathbb{F}_q\) is not a monomial invariant for \(q > 3\).

2.5.10 Show that every linear code over \(\mathbb{F}_q\) is monomial equivalent to a code with a complementary dual if \(q > 3\).
2.5.11 Let $C$ be the code of Example 2.5.24. Show that this code has $6(q - 1)$ monomial automorphisms. Compute $\text{Aut}(C)$ for all possible choices of the $a_i$.

2.5.12 Show that $\text{PAut}(C^\perp)$ and $\text{MAut}(C^\perp)$ are isomorphic as a groups with $\text{PAut}(C)$ and $\text{MAut}(C)$, respectively.

2.5.13 Determine the automorphism group of the ternary code with generator matrix

$$
\begin{pmatrix}
1 & 0 & 1 & 1 \\
0 & 1 & 1 & 2
\end{pmatrix}.
$$

2.5.14 Show that in Example 12.5.5 the permutation automorphism groups obtained for Hamming codes in GAP- and Magma- programs are different. This implies that these codes are not the same. Find out what is the permutation equivalence between these codes.

2.6 Notes

One considers the seminal papers of Shannon [107] and Hamming [61] as the starting point of information theory and coding theory. Many papers that appeared in the early days of coding theory and information theory are published in Bell System Technical Journal, IEEE Transaction on Information Theory and Problemy Peredachi Informatsii. They were collected as key papers in [21, 10, 111].

We mention the following classical textbooks in coding theory [3, 11, 19, 62, 75, 76, 78, 84, 93] and several more recent ones [20, 67, 77]. The Handbook on coding theory [95] gives a wealth on information.

Audio-visual media, compact disc and DVD [76, 105]
fault-tolerant computers ...

deep space telecommunication [86, 134]

***Elias, sequence of codes with with $R > 0$ and error probability going to zero.***

***Forney, concatenated codes, sequence of codes with with $R$ near capacity and error probability going to zero and efficient decoding algorithm.***

***Elias Wozencraft, list decoding***.
Chapter 3

Code constructions and bounds

Ruud Pellikaan and Xin-Wen Wu

This chapter treats the existence and nonexistence of codes. Several constructions show that the existence of one particular code gives rise to a cascade of derived codes. Upper bounds in terms of the parameters exclude codes and lower bounds show the existence of codes.

3.1 Code constructions

In this section, we discuss some classical methods of constructing new codes using known codes.

3.1.1 Constructing shorter and longer codes

The most obvious way to make a shorter code out of a given code is to delete several coordinates. This is called puncturing.

Definition 3.1.1 Let $C$ be an $[n,k,d]$ code. For any codeword, the process of deleting one or more fixed coordinates is called puncturing. Let $P$ be a subset of $\{1,\ldots,n\}$ consisting of $p$ integers such that its complement is the set $\{i_1,\ldots,i_{n-p}\}$ with $1 \leq i_1 < \cdots < i_{n-p} \leq n$. Let $x \in \mathbb{F}_q^n$. Define $x_P = (x_{i_1},\ldots,x_{i_{n-p}}) \in \mathbb{F}_q^{n-p}$. Let $C_P$ be the set of all punctured codewords of $C$, where the puncturing takes place at all the positions of $P$:

$$C_P = \{ c_P \mid c \in C \}.$$

We will also use the notation w.r.t non-punctured positions.

Definition 3.1.2 Let $R$ be a subset of $\{1,\ldots,n\}$ consisting of $r$ integers $\{i_1,\ldots,i_r\}$ with $1 \leq i_1 < \cdots < i_r \leq n$. Let $x \in \mathbb{F}_q^n$. Define $x_{(R)}(x_{i_1},\ldots,x_{i_r}) \in \mathbb{F}_q^r$. Let $C_{(R)}$ be the set of all codewords of $C$ restricted to the positions of $R$:

$$C_{(R)} = \{ c_{(R)} \mid c \in C \}.$$
Remark 3.1.3 So, $C_P$ is a linear code of length $n - p$, where $p$ is the number of elements of $P$. Furthermore $C_P$ is linear, since $C$ is linear. In fact, suppose $G$ is a generator matrix of $C$. Then $C_P$ is a linear code generated by the rows of $G_P$, where $G_P$ is the $k \times (n - p)$ matrix consisting of the $n - p$ columns at the positions $i_1, \ldots, i_{n-p}$ of $G$. If we consider the restricted code $C_{(R)}$, then its generator matrix $G_{(R)}$ is the $k \times r$ submatrix of $G$ composed of the columns indexed by $j_1, \ldots, j_r$, where $R = \{j_1, \ldots, j_r\}$.

Proposition 3.1.4 Let $C$ be an $[n, k, d]$ code. Suppose $P$ consists of $p$ elements. Then the punctured code $C_P$ is an $[n - p, k_P, d_P]$ code with
\[
d - p \leq d_P \leq d \quad \text{and} \quad k - p \leq k_P \leq k.
\]
If moreover $p < d$, then $k_P = k$.

Proof. The given upper bounds are clear. Let $c \in C$. Then at most $p$ nonzero positions are deleted from $c$ to obtain $c_P$. Hence $\text{wt}(c_P) \geq \text{wt}(c) - p$. Hence $d_P \geq d - p$.

The column rank of $G$, which is equal to the row rank, is $k$. The column rank of $G_P$ must be greater than or equal to $k - p$, since $p$ columns are deleted. This implies that the row rank of $G_P$ is at least $k - p$. So $k_P \geq k - p$.

Suppose $p < d$. If $c$ and $c'$ are two distinct codewords in $C$, then $d(c_P, c'_P) \geq d - p > 0$ so $c_P$ and $c'_P$ are distinct. Therefore $C$ and $C_P$ have the same number of codewords. Hence $k = k_P$.

Example 3.1.5 It is worth pointing out that the dimension of $C_P$ can be smaller than $k$. From the definition of puncturing, $C_P$ seemingly has the same number of codewords as $C$. However, it is possible that $C$ contains some distinct codewords that have the same coordinates outside the positions of $P$. In this case, after deleting the coordinates in the complement of $P$, the number of codewords of $C_P$ is less than that of $C$. Look at the following simple example. Let $C$ be the binary code with generator matrix
\[
G = \begin{pmatrix}
1 & 1 & 0 & 0 \\
1 & 1 & 1 & 0 \\
0 & 0 & 1 & 1
\end{pmatrix}.
\]

This is a $[4, 3, 1]$ code. Let $P = \{4\}$. Then, the rows of $G_P$ are $(1, 1, 0), (1, 1, 1)$ and $(0, 0, 1)$. It is clear that the second row is the sum of the first and second ones. So, $G_P$ has row rank 2, and $C_P$ has dimension 2.

In this example we have $d = 1 = p$.

We now introduce an inverse process to puncturing the code $C$, which is called extending the code.

Definition 3.1.6 Let $C$ be a linear code of length $n$. Let $v \in \mathbb{F}_q^n$. The extended code $C^e(v)$ of length $n + 1$ is defined as follows. For every codeword $c = (c_1, \ldots, c_n) \in C$, construct the word $c^e(v)$ by adding the symbol $c_{n+1}(v) \in \mathbb{F}_q$ at the end of $c$ such that the following parity check holds
\[
v_1c_1 + v_2c_2 + \cdots + v_nc_n + c_{n+1} = 0.
\]

Now $C^e(v)$ consists of all the codewords $c^e(v)$, where $c$ is a codeword of $C$. In case $v$ is the all-ones vector, then $C^e(v)$ is denoted by $C^e$. 

Remark 3.1.7 Let $C$ be an $[n,k]$ code. Then it is clear that $C^e(v)$ is a linear subspace of $\mathbb{F}_{q}^{n+1}$, and has dimension $k$. So, $C^e(v)$ is an $[n+1,k]$ code. Suppose $G$ and $H$ are generator and parity check matrices of $C$, respectively. Then, $C^e(v)$ has a generator matrix $G^e(v)$ and a parity check matrix $H^e(v)$, which are given by

$$G^e(v) = \begin{pmatrix} g_{1n+1} \\ g_{2n+1} \\ \vdots \\ g_{kn+1} \end{pmatrix}$$

and

$$H^e(v) = \begin{pmatrix} v_1 & v_2 & \cdots & v_n & 1 \\ 0 & 0 & \vdots & 0 & 0 \end{pmatrix},$$

where the last column of $G^e(v)$ has entries $g_{in+1} = -\sum_{j=1}^{n} g_{ij} v_j$.

Example 3.1.8 The extension of the $[7,4,3]$ binary Hamming code with the generator matrix given in Example 2.2.14 is equal to the $[8,4,4]$ code with the generator matrix given in Example 2.3.26. The increase of the minimum distance by one in the extension of a code of odd minimum distance is a general phenomenon for binary codes.

Proposition 3.1.9 Let $C$ be a binary $[n,k,d]$ code. Then $C^e$ has parameters $[n+1,k,d^e]$ with $d^e = d$ if $d$ is even and $d^e = d + 1$ if $d$ is odd.

Proof. Let $C$ be a binary $[n,k,d]$ code. Then $C^e$ is an $[n+1,k]$ code by Remark 3.1.7. The minimum distance $d^e$ of the extended code satisfies $d \leq d^e \leq d + 1$, since $\text{wt}(c) \leq \text{wt}(c^e) \leq \text{wt}(c) + 1$ for all $c \in C$. Suppose moreover that $C$ is a binary code. Assume that $d$ is even. Then there is a codeword $c$ of weight $d$ and $c^e$ is obtained from $c$ by extending with a zero. So $c^e$ has also weight $d$. If $d$ is odd, then the claim follows, since all the codewords of the extended code $C^e$ have even weight by the parity check $c_1 + \cdots + c_{n+1} = 0$. \hfill \diamond

Example 3.1.10 The binary $[2^r - 1, 2^r - r - 1, 3]$ Hamming code $H_r(2)$ has the extension $H_r(2)^e$ with parameters $[2^r, 2^r - 1, 4]$. The binary $[2^r - 1, r, 2^{r-1}]$ Simplex code $S_r(2)$ has the extension $S_r(2)^e$ with parameters $[2^r, r, 2^{r-1}]$. These claims are a direct consequence of Propositions 2.3.14 and 2.3.16, Remark 3.1.7 and Proposition 3.1.9.

The operations extending and puncturing at the last position are inverse to each other.

Proposition 3.1.11 Let $C$ be a linear code of length $n$. Let $v \in \mathbb{F}_q^n$. Let $P = \{n+1\}$ and $Q = \{n\}$. Then $(C^e(v))_P = C$. If the all-ones vector is a parity check of $C$, then $(C_Q)^e = C$.

Proof. The first statement is a consequence of the fact that $(c^e(v))_P = c$ for all words. The last statement is left as an exercise. \hfill \diamond

Example 3.1.12 The puncturing of the extended binary Hamming code $H_r(2)^e$ gives the original Hamming code back.

By taking subcodes appropriately, we can get some new codes. The following technique of constructing a new code involves a process of taking a subcode and puncturing.
Definition 3.1.13 Let $C$ be an $[n,k,d]$ code. Let $S$ be a subset of $\{1, \ldots, n\}$. Let $C(S)$ be the subcode of $C$ consisting of all $c \in C$ such that $c_i = 0$ for all $i \in S$. The shortened code $C^S$ is defined by $C^S = (C(S))_S$. It is obtained by puncturing the subcode $C(S)$ at $S$, so by deleting the coordinates that are not in $S$.

Remark 3.1.14 Let $S$ consist of $s$ elements. Let $x \in \mathbb{F}_q^{n-s}$. Let $x^S \in \mathbb{F}_q^n$ be the unique word of length $n$ such that $x = (x^S)_S$ and the entries of $x^S$ at the positions of $S$ are zero, by extending $x$ with zeros appropriately. Then

\[ x \in C^S \text{ if and only if } x^S \in C. \]

Furthermore

\[ x^S \cdot y = x \cdot y^S \text{ for all } x \in \mathbb{F}_q^{n-s} \text{ and } y \in \mathbb{F}_q^n. \]

Proposition 3.1.15 Let $C$ be an $[n,k,d]$ code. Suppose $S$ consists of $s$ elements. Then the shortened code $C^S$ is an $[n-s,k^S,d^S]$ code with

\[ k - s \leq k^S \leq k \quad \text{and} \quad d \leq d^S. \]

Proof. The dimension of $C^S$ is equal to the dimension of the subcode $C(S)$ of $C$, and $C(S)$ is defined by $s$ homogeneous linear equations of the form $c_i = 0$. This proves the statement about the dimension. The minimum distance of $C^S$ is the same as the minimum distance of $C(S)$, and $C(S)$ is a subcode of $C$. Hence $d \leq d^S$. \hfill \Box

Example 3.1.16 Consider the binary $[8,4,4]$ code of Example 2.3.26. In the following diagram we show what happens with the generator matrix by shortening at the first position in the left column of the diagram, by puncturing at the first position in the right column, and by taking the dual in the upper and lower row of the diagram.

\[
\begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 1 & 0
\end{pmatrix}
\begin{array}{c}
\text{dual} \\
\downarrow \text{shorten at first position} \\
\downarrow \text{puncture at first position}
\end{array}
\begin{pmatrix}
0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\
1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}
\begin{array}{c}
\text{dual} \\
\end{array}
\]

Notice that the diagram commutes. This is a general fact as stated in the following proposition.
Remark 3.1.14. Hence let $x$.

Proof. Proposition 3.1.17 Let $P$ and $S$ be subsets of $\{1, \ldots, n\}$. Then

$$(C_P^+) = (C^+) \quad \text{and} \quad (CS) = (C_S^+),$$

$$\dim C_P + \dim(C^+) = |P| \quad \text{and} \quad \dim C_S + \dim(C^+) = |S|.$$  

Proof. Let $x \in (C_P^+)$. Let $z \in C$. Then $z_P \in C_P$. So $x_P \cdot z = x \cdot z_P = 0$, by Remark 3.1.14. Hence $x_P \subseteq (C^+)$. Therefore $C_P^+ \subseteq (C^+)$. Conversely, let $x \in (C^+)$. Then $x_P \in C_P$. Let $y \in C_P$. Then $y = z_P$ for some $z \in C$. So $x \cdot y = x \cdot z_P = x_P \cdot z = 0$. Hence $x \in (C_P^+)$. Therefore $(C^+) \subseteq (C_P^+)$, and if fact equality holds, since the converse inclusion was already shown.

The statement on the dimensions is a direct consequence of the corresponding equality of the codes.

The claim about the shortening of $C$ with $S$ is a consequence on the equality on the puncturing with $S = P$ applied to the dual $C$.

If we want to increase the size of the code without changing the code length. We can augment the code by adding a word which is not in the code.

Definition 3.1.18 Let $C$ be an $F_q$-linear code of length $n$. Let $v$ in $F_q^n$. The augmented code, denoted by $C^a(v)$, is defined by

$$C^a(v) = \{ a v + c \mid a \in F_q, \ c \in C \}.$$  

If $v$ is the all-ones vector, then we denote $C^a(v)$ by $C^a$.

Remark 3.1.19 The augmented code $C^a(v)$ is a linear code. Suppose that $G$ is a generator matrix of $C$. Then the $(k+1) \times n$ matrix $G^a(v)$, which is obtained by adding the row $v$ to $G$, is a generator matrix of $C^a(v)$ if $v$ is not an element of $C$.

Proposition 3.1.20 Let $C$ be a code of minimum distance $d$. Suppose that the vector $v$ is not in $C$ and has weight $w$. Then

$$\min\{d - w, w\} \leq d(C^a(v)) \leq \min\{d, w\}.$$  

In particular $d(C^a(v)) = w$ if $w \leq d/2$.

Proof. $C$ is a subcode and $v$ is an element of the augmented code. This implies the upper bound.

The lower bound is trivially satisfied if $d \leq w$. Suppose $w < d$. Let $x$ be a nonzero element of $C^a(v)$. Then $x = \alpha v + c$ for some $\alpha \in F_q$ and $c \in C$. If $\alpha = 0$, then $\text{wt}(x) = \text{wt}(c) \geq d > w$. If $c = 0$, then $\text{wt}(x) = \text{wt}(v) = w$. If $\alpha \neq 0$ and $c \neq 0$, then $c = \alpha v - x$. So $d \leq \text{wt}(c) \leq w + \text{wt}(x)$. Hence $d - w \leq \text{wt}(x)$.

If $w \leq d/2$, then the upper and lower bound are both equal to $w$.

Suppose $C$ is a binary $[n, k, d]$ code. We get a new code by deleting the codewords of odd weight. In other words, the new code $C_{ev}$ consists of all the codewords in $C$ which have even weight. It is called the even weight subcode in Example 2.2.8. This process is also called expurgating the code $C$. 

3.1. CODE CONSTRUCTIONS
Definition 3.1.21 Let $C$ be an $F_q$-linear code of length $n$. Let $v \in F_q^n$. The expurgated code of $C$ is denoted by $C_e(v)$ and is defined by

$$C_e(v) = \{ c : c \in C \text{ and } c \cdot v = 0 \}.$$ 

If $v = 1$, then $C_e(1)$ is denoted by $C_e$.

Proposition 3.1.22 Let $C$ be an $[n,k,d]$ code. Then

$$(C^a(v))^\perp = (C^\perp)(v).$$

Proof. If $v \in C$, then $C^a(v) = C$ and $v$ is a parity check of $C$, so $(C^\perp)_e(v) = C^\perp$. Suppose $v$ is not an element of $C$. Let $G$ be a generator matrix of $C$. Then $G$ is a parity check matrix of $C^\perp$, by Proposition 2.3.29. Now $G^a(v)$ is a generator matrix of $C^a(v)$ by definition. Hence $G^a(v)$ is a parity check matrix of $(C^a(v))^\perp$. Furthermore $G^a(v)$ is also a parity check matrix of $(C^\perp)_e(v)$ by definition. Hence $(C^a(v))^\perp = (C^\perp)_e(v)$.

Lengthening a code is a technique which combines augmenting and extending.

Definition 3.1.23 Let $C$ be an $[n,k]$ code. Let $v \in F_q^n$. The lengthened code $C^l(v)$ is obtained by first augmenting $C$ by $v$, and then extending it: $C^l(v) = (C^a(v))^e$. If $v = 1$, then $C^l(v)$ is denoted by $C^l$.

Remark 3.1.24 The lengthening of an $[n,k]$ code is linear code. If $v$ is not element of $C$, then $C^l(v)$ is an $[n+1,k+1]$ code.

3.1.2 Product codes

We describe a method for combining two codes to get a new code. In Example 2.1.2 the $[9,4,4]$ product code is introduced. This construction will be generalized in this section.

Consider the identification of the space of all $n_1 \times n_2$ matrices with entries in $F_q$ and the space $F_q^{n_1 \times n_2}$, where the matrix $X = (x_{ij})_{1 \leq i \leq n_1, 1 \leq j \leq n_2}$ is mapped to the vector $x$ with entries $x_{(i-1)n_2+j} = x_{ij}$. In other words, the rows of $X$ are put in linear order behind each other:

$$x = (x_{11}, x_{12}, \ldots, x_{1n_2}, x_{21}, \ldots, x_{2n_2}, x_{31}, \ldots, x_{n_1n_2}).$$

For $\alpha \in F_q$ and $n_1 \times n_2$ matrices $(x_{ij})$ and $(y_{ij})$ with entries in $F_q$, the scalar multiplication and addition are defined by:

$$\alpha (x_{ij}) = (\alpha x_{ij}), \quad (x_{ij}) + (y_{ij}) = (x_{ij} + y_{ij}).$$

These operations on matrices give the corresponding operations of the vectors under the identification. Hence the identification of the space of $n_1 \times n_2$ matrices and the space $F_q^{n_1 \times n_2}$ is an isomorphism of vector spaces. In the following these two spaces are identified.

Definition 3.1.25 Let $C_1$ and $C_2$ be respectively $[n_1,k_1,d_1]$ and $[n_2,k_2,d_2]$ codes. Let $n = n_1n_2$. The product code, denoted by $C_1 \otimes C_2$ is defined by

$$C_1 \otimes C_2 = \left\{ (c_{ij})_{1 \leq i \leq n_1, 1 \leq j \leq n_2} : (c_{ij})_{1 \leq i \leq n_1} \in C_1, \text{ for all } j \right\} \cup \left\{ (c_{ij})_{1 \leq j \leq n_2} : (c_{ij})_{1 \leq i \leq n_1} \in C_2, \text{ for all } i \right\}.$$
From the definition, the product code $C_1 \otimes C_2$ is exactly the set of all $n_1 \times n_2$ arrays whose columns belong to $C_1$ and rows to $C_2$. In the literature, the product code is called direct product, or Kronecker product, or tensor product code.

**Example 3.1.26** Let $C_1 = C_2$ be the $[3,2,2]$ binary even weight code. So it consists of the following codewords:

$$(0,0,0), \ (1,1,0), \ (1,0,1), \ (0,1,1).$$

This is the set of all words $(m_1,m_2,m_1 + m_2)$ where $m_1$ and $m_2$ are arbitrary bits. By the definition, the following 16 arrays are the codewords of the product code $C_1 \otimes C_2$:

$$
\begin{pmatrix}
  m_1 & m_2 & m_1 + m_2 \\
  m_3 & m_4 & m_3 + m_4 \\
  m_1 + m_3 & m_2 + m_4 & m_1 + m_2 + m_3 + m_4
\end{pmatrix},
$$

where the $m_i$ are free to choose. So indeed this is the product code of Example 2.1.2. The sum of two arrays $(c_{ij})$ and $(c'_{ij})$ is the array $(c_{ij} + c'_{ij})$. Therefore, $C_1 \otimes C_2$ is a linear codes of length $9 = 3 \times 3$ and dimension $4 = 2 \times 2$. And it is clear that the minimum distance of $C_1 \otimes C_2$ is $4 = 2 \times 2$.

This is a general fact, but before we state this result we need some preparations.

**Definition 3.1.27** For two vectors $x = (x_1,\ldots,x_{n_1})$ and $y = (y_1,\ldots,y_{n_2})$, we define the tensor product of them, denoted by $x \otimes y$, as the $n_1 \times n_2$ array whose $(i,j)$-entry is $x_i y_j$.

**Remark 3.1.28** It is clear that $C_1 \otimes C_2$ is a linear code if $C_1$ and $C_2$ are both linear.

Remark that $x \otimes y \in C_1 \otimes C_2$ if $x \in C_1$ and $y \in C_2$, since the $i$-th row of $x \otimes y$ is $x_i y \in C_2$ and the $j$-th column is $y_j x \in C_1$. But the set of all $x \otimes y \in C_1 \otimes C_2$ with $x \in C_1$ and $y \in C_2$ is not equal to $C_1 \otimes C_2$. In the previous example

$$
\begin{pmatrix}
  0 & 1 & 1 \\
  1 & 0 & 1 \\
  1 & 1 & 0
\end{pmatrix}
$$

is in the product code, but it is not of the form $x \otimes y \in C_1 \otimes C_2$ with $x \in C_1$, since otherwise it would have at least one zero row and at least one zero column.

In general, the number of elements of the form $x \otimes y \in C_1 \otimes C_2$ with $x \in C_1$ and $y \in C_2$ is equal to $q^{k_1 + k_2}$, but $x \otimes y = 0$ if $x = 0$ or $y = 0$. Moreover $\lambda(x \otimes y) = (\lambda x) \otimes y = x \otimes (\lambda y)$ for all $\lambda \in \mathbb{F}_q$. Hence the we get at most $(q^{k_1} - 1)(q^{k_2} - 1)/(q - 1) + 1$ of such elements. If $k_1 > 1$ and $k_2 > 1$ then this is smaller than $q^{k_1 k_2}$, the number of elements of $C_1 \otimes C_2$ according to the following proposition.

**Proposition 3.1.29** Let $x_1,\ldots,x_k \in \mathbb{F}_q^{n_1}$ and $y_1,\ldots,y_k \in \mathbb{F}_q^{n_2}$. If $y_1,\ldots,y_k$ are independent and $x_1 \otimes y_1 + \cdots + x_k \otimes y_k = 0$, then $x_i = 0$ for all $i$.

**Proof.** Suppose that $y_1,\ldots,y_k$ are independent and $x_1 \otimes y_1 + \cdots + x_k \otimes y_k = 0$. Let $x_{is}$ be the $s$-the entry of $x_i$. Then the $s$-th row of $\sum_j x_{ij} \otimes y_j$ is equal to $\sum_j x_{js} y_j$, which is equal to 0 by assumption. Hence $x_{js} = 0$ for all $j$ and $s$. Hence $x_j = 0$ for all $j$. \hfill \Diamond
Corollary 3.1.30 Let \( x_1, \ldots, x_{k_1} \in \mathbb{F}_q^{n_1} \) and \( y_1, \ldots, y_{k_2} \in \mathbb{F}_q^{n_2} \). If \( x_1, \ldots, x_{k_1} \) and \( y_1, \ldots, y_{k_2} \) are both independent, then \( \{ x_i \otimes y_j \mid 1 \leq i \leq k_1, 1 \leq j \leq k_2 \} \) is an independent set of matrices.

Proof. Suppose that \( \sum_{i,j} \lambda_{ij} x_i \otimes y_j = 0 \) for certain scalars \( \lambda_{ij} \in \mathbb{F}_q \). Then \( \sum_j (\sum_i \lambda_{ij} x_i) \otimes y_j = 0 \) and \( y_1, \ldots, y_{k_2} \in \mathbb{F}_q^{n_2} \) are independent. So \( \sum_i \lambda_{ij} x_i = 0 \) for all \( j \) by Proposition 3.1.29. Hence \( \lambda_{ij} = 0 \) for all \( i, j \), since \( x_1, \ldots, x_{k_1} \) are independent.

Proposition 3.1.31 Let \( x_1, \ldots, x_{k_1} \) in \( \mathbb{F}_q^{n_1} \) be a basis of \( C_1 \) and \( y_1, \ldots, y_{k_2} \) in \( \mathbb{F}_q^{n_2} \) a basis of \( C_2 \). Then

\[
\{ x_i \otimes y_j \mid 1 \leq i \leq k_1, 1 \leq j \leq k_2 \}
\]

is a basis of \( C_1 \otimes C_2 \).

Proof. The given set is an independent set by Corollary 3.1.30. This set is a subset of \( C_1 \otimes C_2 \). So the dimension of \( C_1 \otimes C_2 \) is at least \( k_1 k_2 \). Now we will show that they form in fact a basis for \( C_1 \otimes C_2 \). Without loss of generality we may assume that \( C_1 \) is systematic at the first \( k_1 \) coordinates with generator matrix \( (I_{k_1}) A \) and \( C_2 \) is systematic at the first \( k_2 \) coordinates with generator matrix \( (I_{k_2}) B \). Then \( U \) is an \( l \times n_2 \) matrix, with rows in \( C_2 \) if and only if \( U = (M \otimes MB) \), where \( M \) is an \( l \times k_2 \) matrix. And \( V \) is an \( n_1 \times m \) matrix, with columns in \( C_1 \) if and only if \( V^T = (N \otimes NA) \), where \( N \) is an \( m \times k_1 \) matrix. Now let \( M \) be an \( k_1 \times k_2 \) matrix. Then \( (M \otimes MB) \) is a \( k_1 \times n_2 \) matrix with rows in \( C_2 \), and

\[
\begin{pmatrix}
    M \\
    A^T M
\end{pmatrix}
\]

is an \( n_1 \times k_2 \) matrix with columns in \( C_1 \). Therefore

\[
\begin{pmatrix}
    M & MB \\
    A^T M & A^T MB
\end{pmatrix}
\]

is an \( n_1 \times n_2 \) matrix with columns in \( C_1 \) and rows in \( C_2 \) for every \( k_1 \times k_2 \) matrix \( M \). And conversely every codeword of \( C_1 \otimes C_2 \) is of this form. Hence the dimension of \( C_1 \otimes C_2 \) is equal to \( k_1 k_2 \) and the given set is a basis of \( C_1 \otimes C_2 \).

Theorem 3.1.32 Let \( C_1 \) and \( C_2 \) be respectively \( [n_1,k_1,d_1] \) and \( [n_2,k_2,d_2] \). Then the product code \( C_1 \otimes C_2 \) is an \( [n_1n_2,k_1k_2,d_1d_2] \) code.

Proof. By definition \( n = n_1 n_2 \) is the length of the product code. It was already mentioned that \( C_1 \otimes C_2 \) is a linear subspace of \( \mathbb{F}_q^{n_1 n_2} \). The dimension of the product code is \( k_1 k_2 \) by Proposition 3.1.31.

Next, we prove that the minimum distance of \( C_1 \otimes C_2 \) is \( d_1 d_2 \). For any codeword of \( C_1 \otimes C_2 \), which is a \( n_1 \times n_2 \) array, every nonzero column has weight \( \geq d_1 \), and every nonzero row has weight \( \geq d_2 \). So, the weight of a nonzero codeword of the product code is at least \( d_1 d_2 \). This implies that the minimum distance of \( C_1 \otimes C_2 \) is at least \( d_1 d_2 \). Now suppose \( x \in C_1 \) has weight \( d_1 \), and \( y \in C_2 \) has weight \( d_2 \). Then, \( x \otimes y \) is a codeword of \( C_1 \otimes C_2 \) and has weight \( d_1 d_2 \).
3.1. CODE CONSTRUCTIONS

Definition 3.1.33 Let \( A = (a_{ij}) \) be a \( k_1 \times n_1 \) matrix and \( B = (b_{ij}) \) a \( k_2 \times n_2 \) matrix. The Kronecker product or tensor product \( A \otimes B \) of \( A \) and \( B \) is the \( k_1 k_2 \times n_1 n_2 \) matrix obtained from \( A \) by replacing every entry \( a_{ij} \) by \( a_{ij} B \).

Remark 3.1.34 The tensor product \( x \otimes y \) of the two row vectors \( x \) and \( y \) of length \( n_1 \) and \( n_2 \), respectively, as defined in Definition 3.1.27 is the same as the Kronecker product of \( x^T \) and \( y \), now considered as \( n_1 \times 1 \) and \( 1 \times n_2 \) matrices, respectively, as in Definition 3.1.33.

Proposition 3.1.35 Let \( G_1 \) be a generator matrix of \( C_1 \), and \( G_2 \) a generator matrix of \( C_2 \). Then \( G_1 \otimes G_2 \) is a generator matrix of \( C_1 \otimes C_2 \).

Proof. In this proposition the codewords are considered as elements of \( \mathbb{F}_q^n \) and no longer as matrices. Let \( x_i \) the \( i \)-th row of \( G_1 \), and denote by \( y_j \) the \( j \)-th row of \( G_2 \). So \( x_1, \ldots, x_{k_1}, y_1, \ldots, y_{k_2} \in \mathbb{F}_q^{n_1} \) is a basis of \( C_1 \) and \( y_1, \ldots, y_{k_2} \in \mathbb{F}_q^{n_2} \) is a basis of \( C_2 \). Hence the set \( \{x_1 \otimes y_j \mid 1 \leq i \leq k_1, 1 \leq j \leq k_2 \} \) is a basis of \( C_1 \otimes C_2 \) by Proposition 3.1.31. Furthermore, if \( l = (i - 1)k_2 + j \), then \( x_i \otimes y_j \) is the \( l \)-th row of \( G_1 \otimes G_2 \). Hence the matrix \( G_1 \otimes G_2 \) is a generator matrix of \( C_1 \otimes C_2 \).

Example 3.1.36 Consider the ternary codes \( C_1 \) and \( C_2 \) with generator matrices

\[
G_1 = \begin{pmatrix}
1 & 1 & 1 \\
0 & 1 & 2 \\
\end{pmatrix}
\quad \text{and} \quad
G_2 = \begin{pmatrix}
1 & 1 & 1 & 0 \\
0 & 1 & 2 & 0 \\
0 & 0 & 0 & 1 \\
0 & 2 & 1 & 0 \\
\end{pmatrix},
\]

respectively. Then

\[
G_1 \otimes G_2 = \begin{pmatrix}
1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\
0 & 1 & 2 & 0 & 0 & 1 & 2 & 0 \\
0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 1 & 1 & 1 & 0 & 2 & 2 & 2 \\
0 & 0 & 0 & 0 & 1 & 2 & 0 & 0 & 2 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 2 & 2 & 2 \\
\end{pmatrix}.
\]

The second row of \( G_1 \) is \( x_2 = (0,1,2) \) and \( y_2 = (0,1,2,0) \) is the second row of \( G_2 \). Then \( x_2 \otimes y_2 \) is equal to

\[
\begin{pmatrix}
0 & 0 & 0 & 0 \\
0 & 1 & 2 & 0 \\
0 & 2 & 1 & 0 \\
\end{pmatrix},
\]

considered as a matrix, and equal to \((0,0,0,0,0,1,2,0,0,2,1,0)\) written as a vector, which is indeed equal to the \((2 - 1)3 + 2 = 5\)-th row of \( G_1 \otimes G_2 \).

3.1.3 Several sum constructions

We have seen that given an \([n_1, k_1]\) code \( C_1 \) and an \([n_2, k_2]\) code \( C_2 \), by the product construction, we get an \([n_1 n_2, k_1 k_2]\) code. The product code has information rate \((k_1 k_2)/(n_1 n_2) = R_1 R_2\), where \( R_1 \) and \( R_2 \) are the rates of \( C_1 \) and \( C_2 \), respectively. In this subsection, we introduce some simple constructions by which we can get new codes with greater rate from two given codes.
Definition 3.1.37 Given an \([n_1,k_1]\) code \(C_1\) and an \([n_2,k_2]\) code. Their *direct sum* \(C_1 \oplus C_2\), also called \((u|v)\) construction is defined by
\[
C_1 \oplus C_2 = \{ (u|v) \mid u \in C_1, v \in C_2 \},
\]
where \((u|v)\) denotes the word \((u_1,\ldots,u_{n_1},v_1,\ldots,v_{n_2})\) if \(u = (u_1,\ldots,u_{n_1})\) and \(v = (v_1,\ldots,v_{n_2})\).

Proposition 3.1.38 Let \(C_i\) be an \([n_i,k_i,d_i]\) code with generator matrix \(G_i\) for \(i = 1,2\). Let \(d = \min\{d_1,d_2\}\). Then \(C_1 \oplus C_2\) is an \([n_1 + n_2,k_1 + k_2,d]\) code with generator matrix
\[
G = \begin{pmatrix} G_1 & 0 \\ 0 & G_2 \end{pmatrix},
\]

Proof. Let \(x_1,\ldots,x_{k_1}\) and \(y_1,\ldots,y_{k_2}\) be bases of \(C_1\) and \(C_2\), respectively. Then \((x_1|0),\ldots,(x_{k_1}|0),(0|y_1),\ldots,(0|y_{k_2})\) is a basis of the direct sum code. Therefore, the direct sum is an \([n_1 + n_2,k_1 + k_2]\) with the generator matrix \(G\). The minimum distance of the direct sum is \(\min\{d_1,d_2\}\). The direct sum or \((u|v)\) construction is defined by the juxtaposition of arbitrary codewords \(u \in C_1\) and \(v \in C_2\). In the following definition only a restricted set pairs of codewords are put behind each other. This definition depends on the choice of the generator matrices of the codes \(C_1\) and \(C_2\).

Definition 3.1.39 Let \(C_i\) be an \([n_i,k_i,d_i]\) code and \(C_j\) an \([n_j,k_j,d_j]\) code with generator matrices \(G_i\) and \(G_j\), respectively. The *juxtaposition* of the codes \(C_i\) and \(C_j\) is the code with generator matrix \((G_i|G_j)\).

Proposition 3.1.40 Let \(C_i\) be an \([n_i,k_i,d_i]\) code for \(i = 1,2\). Then the juxtaposition of the codes \(C_1\) and \(C_2\) is an \([n_1 + n_2,k,d]\) with \(d \geq d_1 + d_2\).

Proof. The length and the dimension are clear from the definition. A nonzero codeword \(c\) is of the form \(mG = (mG_1,mG_2)\) for a nonzero element \(m\) in \(\mathbb{Z}_q^k\). So \(mG_i\) is a nonzero codeword of \(C_i\). Hence the weight of \(c\) is at least \(d_1 + d_2\).

Theorem 3.1.42 Let \(C_i\) be an \([n_i,k_i,d_i]\) code with generator matrix \(G_i\) for \(i = 1,2\). Then the \((u|u+v)\) construction of \(C_1\) and \(C_2\) is an \([2n,k_1 + k_2,d]\) code with minimum distance \(d = \min\{2d_1,d_2\}\) and generator matrix
\[
G = \begin{pmatrix} G_1 & G_1 \\ 0 & G_2 \end{pmatrix}.
\]
Remark 3.1.44 For two vectors \(\mathbf{x}_1, \ldots, \mathbf{x}_{k_1}\) and \(\mathbf{y}_1, \ldots, \mathbf{y}_{k_2}\) are bases of \(C_1\) and \(C_2\), respectively. Then, it is easy to see that \((\mathbf{x}_1, \mathbf{x}_2), (\mathbf{y}_1), \ldots, (\mathbf{y}_{k_2})\) is a basis of the \((\mathbf{u} + \mathbf{v})\) construction. So, it is an \([2n, k_1 + k_2]\) with generator matrix \(G\) as given.

Consider the minimum distance \(d\) of the \((\mathbf{u} + \mathbf{v})\) construction. For any codeword \((\mathbf{x}|\mathbf{x} + \mathbf{y})\), we have \(\text{wt}(\mathbf{x}|\mathbf{x} + \mathbf{y}) = \text{wt}(\mathbf{x}) + \text{wt}(\mathbf{x} + \mathbf{y})\). If \(\mathbf{y} = \mathbf{0}\), then \(\text{wt}(\mathbf{x}|\mathbf{x} + \mathbf{y}) = 2\text{wt}(\mathbf{x}) \geq 2d_1\). If \(\mathbf{y} \neq \mathbf{0}\), then

\[
\text{wt}(\mathbf{x}|\mathbf{x} + \mathbf{y}) = \text{wt}(\mathbf{x}) + \text{wt}(\mathbf{x} + \mathbf{y}) \geq \text{wt}(\mathbf{x}) + \text{wt}(\mathbf{y}) - \text{wt}(\mathbf{x}) = \text{wt}(\mathbf{y}) \geq d_2.
\]

Hence, \(d \geq \min\{2d_1, d_2\}\). Let \(\mathbf{x}_0\) be a codeword of \(C_1\) with weight \(d_1\), and \(\mathbf{y}_0\) be a codeword of \(C_2\) with weight \(d_2\). Then, either \((\mathbf{x}_0|\mathbf{x}_0)\) or \((\mathbf{0}|\mathbf{y}_0)\) has weight \(\min\{2d_1, d_2\}\).

Example 3.1.43 The \((\mathbf{u} + \mathbf{v})\) construction of the binary even weight \([4,3,2]\) code and the 4-tuple repetition \([4,1,4]\) code gives a \([8,4,4]\) code with generator matrix

\[
\begin{pmatrix}
1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1
\end{pmatrix},
\]

which is equivalent with the extended Hamming code of Example 2.3.26.

Remark 3.1.44 For two vectors \(\mathbf{u}\) of length \(n_1\) and \(\mathbf{v}\) of length \(n_2\), we can still define the sum \(\mathbf{u} + \mathbf{v}\) as a vector of length \(\max\{n_1, n_2\}\), by adding enough zeros at the end of the shorter vector. From this definition of sum, the \((\mathbf{u} + \mathbf{v})\) construction still works for codes \(C_1\) and \(C_2\) of different lengths.

Proposition 3.1.45 If \(C_1\) is an \([n_1, k_1, d_1]\) code, and \(C_2\) is an \([n_2, k_2, d_2]\) code, then the \((\mathbf{u} + \mathbf{v})\) construction is an \([n_1 + \max\{n_1, n_2\}, k_1 + k_2, \min\{2d_1, d_2\}]\) linear code.

Proof. The proof is similar to the proof of Theorem 3.1.42.

Definition 3.1.46 The \((\mathbf{u} + \mathbf{v})|\mathbf{u} - \mathbf{v}\) construction is a slightly modified construction, which is defined as the following code

\[
\left\{ (\mathbf{u} + \mathbf{v}|\mathbf{u} - \mathbf{v}) \mid \mathbf{u} \in C_1, \mathbf{v} \in C_2 \right\}.
\]

When we consider this construction, we restrict ourselves to the case \(q\) odd. Since \(\mathbf{u} + \mathbf{v} = \mathbf{u} - \mathbf{v}\) if \(q\) is even.

Proposition 3.1.47 Let \(C_i\) be an \([n_i, k_i, d_i]\) code with generator matrix \(G_i\) for \(i = 1, 2\). Assume that \(q\) is odd. Then, the \((\mathbf{u} + \mathbf{v})|\mathbf{u} - \mathbf{v}\) construction of \(C_1\) and \(C_2\) is an \([2n, k_1 + k_2, d]\) code with \(d \geq \min\{2d_1, 2d_2, \max\{d_1, d_2\}\}\) and generator matrix

\[
G = \begin{pmatrix}
G_1 & G_1 \\
G_2 & -G_2
\end{pmatrix}.
\]
CHAPTER 3. CODE CONSTRUCTIONS AND BOUNDS

Proof. The proof of the proposition is similar to that of Theorem 3.1.42. In fact, suppose \( x_1, \ldots, x_{k_1} \) and \( y_1, \ldots, y_{k_2} \) are bases of \( C_1 \) and \( C_2 \), respectively, then every codeword is of the form \((u \oplus v)u - v \). With \( u \in C_1 \) and \( v \in C_2 \). So \((u \oplus v)u + (v - v)\) is a linear combination of \((x_1 | x_2), \ldots, (x_{k_1} | x_{k_2})\), and \((v - v)\) is a linear combination of \((y_1 - y_1), \ldots, (y_{k_2} - y_{k_2})\). Using the assumption that \( q \) is odd, we can prove that this set of vectors \( (x_i | x_i) \), \( (y_j - y_j) \) is linearly independent. Suppose that

\[
\sum_i \lambda_i(x_i | x_i) + \sum_j \mu_j(y_j - y_j) = 0,
\]

Then

\[
\begin{cases}
\sum_i \lambda_i x_i + \sum_j \mu_j y_j = 0, \\
\sum_i \lambda_i x_i - \sum_j \mu_j y_j = 0.
\end{cases}
\]

Adding the two equations and dividing by 2 gives \( \sum_i \lambda_i x_i = 0 \). So \( \lambda_i = 0 \) for all \( i \), since the \( x_i \) are independent. Similarly, the subtraction of the equations gives that \( \mu_j = 0 \) for all \( j \).

So the \( (x_i | x_i), (y_j - y_j) \) are independent and generate the code. Hence they form a basis and this shows that the given \( G \) is a generator matrix of this construction.

Let \((u \oplus v)u - v\) be a nonzero codeword. The weight of this word is at least \( 2d_1 \) if \( v = 0 \), and at least \( 2d_2 \) if \( u = 0 \). Now suppose \( u \neq 0 \) and \( v \neq 0 \). Then the weight of \( u \oplus v \) is at least \( \text{wt}(u) - w \), where \( w \) is the number of positions \( i \) such that \( u_i = v_i \neq 0 \). If \( u_i = v_i \neq 0 \), then \( u_i + v_i \neq 0 \), since \( q \) is odd. Hence \( \text{wt}(u \oplus v) \geq w \), and \( (u \oplus v)u - v \geq w + (\text{wt}(u) - w) = \text{wt}(u) \geq d_1 \). In the same way \( \text{wt}(u \oplus v)u - v \geq d_2 \). Hence \( \text{wt}(u \oplus v)u - v \geq \max\{d_1, d_2\} \). This proves the estimate on the minimum distance. \[ \diamond \]

Example 3.1.48 Consider the following ternary codes

\[ C_1 = \{000, 110, 220\}, \quad C_2 = \{000, 011, 022\}. \]

They are \([3, 1, 2]\) codes. The \((u \oplus v)u - v\) construction of these codes is a \([6, 2, d]\) code with \( d \geq 2 \) by Proposition 3.1.47. It consists of the following nine codewords:

\[
\begin{align*}
(0, 0, 0, 0, 0, 0), & \quad (0, 1, 1, 0, 2, 2), & \quad (0, 2, 2, 0, 1, 1), \\
(1, 1, 0, 1, 1, 0), & \quad (1, 2, 1, 1, 0, 2), & \quad (1, 0, 2, 1, 2, 1), \\
(2, 2, 0, 2, 2, 0), & \quad (2, 0, 1, 2, 1, 2), & \quad (2, 1, 2, 2, 0, 1).
\end{align*}
\]

Hence \( d = 4 \). On the other hand, by the \((u \oplus u)u + v\) construction, we get a \([6, 2, 2]\) code, which has a smaller minimum distance than the \((u \oplus v)u - v\) construction.

Now a more complicated construction is given.

Definition 3.1.49 Let \( C_1 \) and \( C_2 \) be \([n, k_1]\) and \([n, k_2]\) codes, respectively. The \((a + x)|b + x|a + b - x)\) construction of \( C_1 \) and \( C_2 \) is the following code

\[
\{ (a + x)|b + x|a + b - x) \mid a, b \in C_1, x \in C_2 \}
\]
Proposition 3.1.50 Let $C_1$ and $C_2$ be $[n,k_1]$ and $[n,k_2]$ codes over $\mathbb{F}_q$, respectively. Suppose $q$ is not a power of 3. Then, the $(a + x| b + x| a + b - x)$ construction of $C_1$ and $C_2$ is an $[3n, 2k_1 + k_2]$ code with generator matrix

$$G = \begin{pmatrix} G_1 & 0 & G_1 \\ 0 & G_1 & G_1 \\ G_2 & G_2 & -G_2 \end{pmatrix}.$$ 

Proof. Let $x_1, \ldots, x_{k_1}$ and $y_1, \ldots, y_{k_2}$ be bases of $C_1$ and $C_2$, respectively. Consider the following $2k_1 + k_2$ vectors

$$(x_1|0|x_1), \ldots, (x_{k_1}|0|x_{k_1}),$$

$$(0|x_1|x_1), \ldots, (0|x_{k_1}|x_{k_1}),$$

$$(y_1|y_1 - y_1), \ldots, (y_{k_2}|y_{k_2} - y_{k_2}).$$

It is left as an exercise to check that they form a basis of this construction in case $q$ is not a power of 3. This shows that the given $G$ is a generator matrix of the code and that it dimension is $2k_1 + k_2$.

For binary codes, some simple inequalities, for example, Exercise 3.1.9, can be used to estimate the minimum distance of the last construction. In general we have the following estimate for the minimum distance.

Proposition 3.1.51 Let $C_1$ and $C_2$ be $[n,k_1,d_1]$ and $[n,k_2,d_2]$ codes over $\mathbb{F}_q$, respectively. Suppose $q$ is not a power of 3. Let $d_0$ and $d_3$ be the minimum distance of $C_1 \cap C_2$ and $C_1 + C_2$, respectively. Then, the minimum distance $d$ of the $(a + x| b + x| a + b - x)$ construction of $C_1$ and $C_2$ is at least $\min\{d_0, 2d_1, 3d_3\}$.

Proof. This is left as an exercise.

The choice of the minus sign in the $(a + x| b + x| a + b - x)$ construction becomes apparent in the construction of self-dual codes over $\mathbb{F}_q$ for arbitrary $q$ not divisible by 3.

Proposition 3.1.52 Let $C_1$ and $C_2$ be self-dual $[2k,k]$ codes. The the codes obtained from $C_1$ and $C_2$ by the direct sum, the $(u|u + v)$ if $C_1 = C_2$, and the $(u + v|u - v)$ constructions and the $(a + x| b + x| a + b - x)$ construction in case $q$ is not divisible by 3 are also self-dual.

Proof. The generator matrix $G_i$ of $C_i$ has size $k \times 2k$ and satisfies $G_iG_i^T = 0$ for $i = 1, 2$. In all the constructions the generator matrix $G$ has size $2k \times 4k$ or $3k \times 6k$ as given in Theorem 3.1.42 and Propositions 3.1.38, 3.1.48 and 3.1.50 satisfies also $GG^T = 0$. For instance in the case of the $(a + x| b + x| a + b - x)$ construction we have

$$GG^T = \begin{pmatrix} G_1 & 0 & G_1 \\ 0 & G_1 & G_1 \\ G_2 & G_2 & -G_2 \end{pmatrix} \begin{pmatrix} G_1^T & 0 & G_1^T \\ 0 & G_1^T & G_1^T \\ G_2^T & G_2^T & -G_2^T \end{pmatrix}.$$ 

All the entries in this product are the sum of terms of the form $G_iG_i^T$ or $G_1G_2^T - G_1G_2$ which are all zero. Hence $GG^T = 0$. 

$\diamondsuit$
Example 3.1.53 Let $C_1$ be the binary [8, 4, 4] self-dual code with the generator matrix $G_1$ of the form $(I_4|A_1)$ as given in Example 2.3.26. Let $C_2$ be the code with generator matrix $G_2 = (I_4|A_2)$ where $A_2$ is obtained from $A_1$ by a cyclic shift of the columns.

\[
A_1 = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}, \quad A_2 = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix}.
\]

The codes $C_1$ and $C_2$ are both [8, 4, 4] self-dual codes and $C_1 \cap C_2 = \{0, 1\}$ and $C_1 + C_2$ is the even weight code. Let $C$ be the $(a + x)|b + x|a + b + x)$ construction applied to $C_1$ and $C_2$. Then $C$ is a binary self-dual [24, 12, 8] code. The claim on the minimum distance is the only remaining statement to verify, by Proposition 3.1.50. The weights of the rows of $G$ are all divisible by 4. Hence the weights of all codewords are divisible by 4 by Exercise ???. Let $c = (a + x)|b + x|a + b + x)$ be a nonzero codeword with $a, b \in C_1$ and $x \in C_2$. If $a + x = 0$, then $a = x \in C_1 \cap C_2$. So $a = x = 0$ and $c = (0)|b + x)$ or $a = x = 1$ and $c = (b + 1)|b)$, and in both cases the weight of $c$ is at least 8, since the weight of $b$ is at least 4 and the weight of 1 is 8. Similarly it is argued that the weight of $c$ is at least 8 if $b + x = 0$ or $a + b + x = 0$. So we may assume that neither of $a + x$, $b + x$, nor $a + b + x$ is zero. Hence all three are nonzero even weight codewords and $\text{wt}(c) \geq 6$. But the weight is divisible by 4. Hence the minimum distance is at least 8. Let $a$ be a codeword of $C_1$ of weight 4, then $c = (a, 0, a)$ is a codeword of weight 8. In this way we have constructed a binary self-dual [24, 12, 8] code. It is called the extended binary Golay code. The binary Golay code is the [23, 12, 7] code obtained by puncturing one coordinate.

### 3.1.4 Concatenated codes

For this section we need some theory of finite fields. See Section 7.2.1. Let $q$ be a prime power and $k$ a positive integer. The finite field $\mathbb{F}_q$ with $q^k$ elements contains $\mathbb{F}_q$ as a subfield. Now $\mathbb{F}_q^k$ is a $k$-dimensional vector over $\mathbb{F}_q$. Let $\xi_1, \ldots, \xi_k$ be a basis of $\mathbb{F}_q^k$ over $\mathbb{F}_q$.

Consider the map

\[
\varphi : \mathbb{F}_q^k \longrightarrow \mathbb{F}_q^k,
\]

defined by $\varphi(a) = a_1\xi_1 + \cdots + a_k\xi_k$. Then $\varphi$ is an isomorphism of vector spaces with inverse map $\varphi^{-1}$.

The vector space $\mathbb{F}_q^{K \times k}$ of $K \times k$ matrices over $\mathbb{F}_q$ form a vector space of dimension $Kk$ over $\mathbb{F}_q$ and it is linear isometric with $\mathbb{F}_q^{Kk}$ by taking some ordering of the $Kk$ entries of such matrices. Let $M$ be a $K \times k$ matrix over $\mathbb{F}_q$ with $i$-th row $m_i$. The map

\[
\varphi_K : \mathbb{F}_q^{K \times k} \longrightarrow \mathbb{F}_q^{Kk}
\]

is defined by $\varphi_K(M) = (\varphi(m_1), \ldots, \varphi(m_K))$. The inverse map

\[
\varphi_N^{-1} : \mathbb{F}_q^{N \times k} \longrightarrow \mathbb{F}_q^{N \times k}
\]

is given by $\varphi_N^{-1}(a_1, \ldots, a_n) = P$, where $P$ is the $N \times k$ matrix with $i$-th row $p_i = \varphi^{-1}(a_i)$. 

Let $A$ be an $[N,K]$ code over $\mathbb{F}_{q^k}$, and $B$ an $[n,k]$ code over $\mathbb{F}_q$. Let $G_A$ and $G_B$ be generator matrices of $A$ and $B$, respectively. The $N$-fold direct sum $G^{(N)}_B = G_B \oplus \cdots \oplus G_B : \mathbb{F}_q^{Nk} \rightarrow \mathbb{F}_q^{Nn}$ is defined by $G^{(N)}_B(P)$ which is the $N \times n$ matrix $Q$ with $i$-th row $q_i = p_i G_B$ for a given $N \times k$ matrix $P$ with $i$-th row $p_i$ in $\mathbb{F}_q^k$.

By the following concatenation procedure a message of length $Kk$ over $\mathbb{F}_q$ is encoded to a codeword of length $Nn$ over $\mathbb{F}_q$.

**Step 1:** The $K \times k$ matrix $M$ is mapped to $m = \varphi_K(M)$.

**Step 2:** $m$ in $\mathbb{F}_q^K$ is mapped to $a = m G_A$ in $\mathbb{F}_q^N$.

**Step 3:** $a$ in $\mathbb{F}_q^N$ is mapped to $P = \varphi_N^{-1}(a)$.

**Step 4:** The $N \times k$ matrix $P$ with $i$-th row $p_i$ is mapped to the $N \times n$ matrix $Q$ with $i$-th row $q_i = p_i G_B$.

The encoding map $\mathcal{E} : \mathbb{F}_q^{K \times k} \rightarrow \mathbb{F}_q^{N \times n}$ is the composition of the four maps explained above:

$$\mathcal{E} = G_B^{(N)} \circ \varphi_N^{-1} \circ G_A \circ \varphi_K.$$ 

Let

$$C = \{ \mathcal{E}(M) \mid M \in \mathbb{F}_q^{K \times k} \}.$$

We call $C$ the concatenated code with outer code $A$ and inner code $B$.

**Theorem 3.1.54** Let $A$ be an $[N,K,D]$ code over $\mathbb{F}_{q^k}$, and $B$ an $[n,k,d]$ code over $\mathbb{F}_q$. Let $C$ be the concatenated code with outer code $A$ and inner code $B$. Then $C$ is an $\mathbb{F}_q$-linear $[Nn,Kk]$ code and its minimum distance is at least $Dd$.

**Proof.** The encoding map $\mathcal{E}$ is an $\mathbb{F}_q$-linear map, since it is a composition of four $\mathbb{F}_q$-linear maps. The first and third map are isomorphisms, and the second and last map are injective, since they are given by generator matrices of full rank. Hence $\mathcal{E}$ is injective. Hence the concatenated code $C$ is an $\mathbb{F}_q$-linear code of length $Nn$ and dimension $Kk$.

Next consider the minimum distance of $C$. Since $A$ is an $[N,K,D]$ code, every nonzero codeword $a$ obtained in Step 2 has weight at least $D$. As a result, the $N \times k$ matrix $P$ obtained from Step 3 has at least $D$ nonzero rows $p_i$. Now, because $B$ is a $[n,k,d]$ code, every $p_i G_B$ has weight $d$, if $p_i$ is not zero. Therefore, the minimum distance of $C$ is at least $Dd$.

**Example 3.1.55** The definition of the concatenated code depends on the choice of the map $\varphi$ that is on the choice of the basis $\xi_1, \ldots, \xi_n$. In fact the minimum distance of the concatenated code can be strictly larger than $Dd$ as the following example shows.

The field $\mathbb{F}_3$ contains the ternary field $\mathbb{F}_3$ as a subfield and an element $\xi$ such that $\xi^2 = 1 + \xi$, since the polynomial $X^2 - X - 1$ is irreducible in $\mathbb{F}_3[X]$. Now take $\xi_1 = 1$ and $\xi_2 = \xi$ as a basis of $\mathbb{F}_3$ over $\mathbb{F}_3$. Let $A$ be the $[2,1,2]$ outer code
over \( \mathbb{F}_3 \) with generator matrix \( G_A = [1, \xi^2] \). Let \( B \) be the trivial \([2,2,1]\) code over \( \mathbb{F}_3 \) with generator matrix \( G_B = I_2 \). Let \( M = (m_1, m_2) \in \mathbb{F}_3^{1 \times 2} \). Then \( m = \varphi_1(M) = m_1 + m_2 \xi \in \mathbb{F}_3 \). So \( a = mG_A = (m_1 + m_2 \xi, (m_1 + m_2) + (m_1 - m_2) \xi) \), since \( \xi^3 = 1 - \xi \). Hence

\[
Q = P = \varphi_2^{-1}(a) = \left( \begin{array}{cc} m_1 & m_2 \\ m_1 + m_2 & m_1 - m_2 \end{array} \right).
\]

Therefore the concatenated code has minimum distance \( 3 > Dd \).

Suppose we would have taken \( \xi_1 = 1 \) and \( \xi_2 = \xi^3 \) as a basis instead. Take \( M = (1,0) \). Then \( m = \varphi_1(M) = 1 \in \mathbb{F}_3 \). So \( a = mG_A = (1, \xi^2) \). Hence \( Q = P = \varphi_2^{-1}(a) = I_2 \) is a codeword in the concatenated code that has weight 2.

Thus, the definition and the parameters of a concatenated code dependent on the specific choice of the map \( \varphi \).

### 3.1.5 Exercises

3.1.1 Prove Proposition 3.1.11.

3.1.2 Let \( C \) be the binary \([9,4,4]\) product code of Example 2.1.2. Show that puncturing \( C \) at the position \( i \) gives a \([8,4,3]\) code for every choice of \( i = 1, \ldots, 9 \). Is it possible to obtain the binary \([7,4,3]\) Hamming code by puncturing \( C \)? Show that shortening \( C \) at the position \( i \) gives a \([8,3,4]\) code for every choice of \( i \). Is it possible to obtain the binary \([7,3,4]\) simplex code by a combination of puncturing and shortening the product code?

3.1.3 Suppose that there exists an \([n', k', d']_q\) code and an \([n, k, d]_q\) code with \( n' < d' \) as a subcode. Use a generalization of the construction for \( C^e(v) \) to show that there exists an \([n', k, d + d']_q\) code.

3.1.4 Let \( C' \) be a binary code with minimum distance \( d' \). Let \( d' \) be the largest weight of any codeword of \( C \). Suppose that the all-ones vector is not in \( C \). Then, the augmented code \( C^a \) has minimum distance \( \min\{d, n - d'\} \).

3.1.5 Let \( C \) be an \( \mathbb{F}_q \)-linear code of length \( n \). Let \( v \in \mathbb{F}_q^n \) and \( S = \{n + 1\} \). Suppose that the all-ones vector is a parity check of \( C \) but not of \( v \). Show that \( (C^e(v))^S = C \).

3.1.6 Show that the shortened binary \([7,3,4]\) code is a product code of codes of length 2 and 3.

3.1.7 Let \( C \) be a nontrivial linear code of length \( n \). Then \( C \) is the direct sum of two codes of lengths strictly smaller than \( n \) if and only if \( C = v \oplus C \) for some \( v \in \mathbb{F}_q^n \) with nonzero entries that are not all the same.

3.1.8 Show that the punctured binary \([7,3,4]\) is equal to the \((u|u + v)\) construction of a \([3,2,2]\) code and a \([3,1,3]\) code.

3.1.9 For binary vectors \( a, b \) and \( x \),

\[
\text{wt}(a + x|b + x(a + b + x)) \geq 2\text{wt}(a + b + a \ast b) - \text{wt}(x),
\]

with equality if and only if \( a_i = 1 \) or \( b_i = 1 \) or \( x_i = 0 \) for all \( i \), where \( a \ast b = (a_1b_1, \ldots, a_nb_n) \).
3.2. BOUNDS ON CODES

3.1.10 Give a parity check matrix for the direct sum, the \((u|u + v)\), the \((u + v|u - v)\) and the \((a + x|b + x|a + b - x)\) construction in terms of the parity check matrices \(H_1\) and \(H_2\) of the codes \(C_1\) and \(C_2\), respectively.

3.1.11 Give proofs of Propositions 3.1.50 and 3.1.51.

3.1.12 Let \(C_i\) be an \([n,k_i,d_i]\) code over \(F_q\) for \(i = 1,2\), where \(q\) is a power of 3. Let \(k_0\) be the dimension of \(C_1 \cap C_2\) and \(d_3\) the minimum distance of \(C_1 + C_2\). Show that the \((a + x|b + x|a + b - x)\) construction with \(C_1\) and \(C_2\) gives a \([3n,2k_1 + k_2 - k_0,d]\) code with \(d \geq \min\{2d_1,3d_3\}\).

3.1.13 Show that \(C_1 \cap C_2 = \{0,1\}\) and \(C_1 + C_2\) is the even weight code, for the codes \(C_1\) and \(C_2\) of Example 3.1.53.

3.1.14 Show the existence of a binary \([45,15,16]\) code.

3.1.15 Show the existence of a binary self-dual \([72,36,12]\) code.

3.1.16 [CAS] Construct a binary random \([100,50]\) code and make sure that identities from Proposition 3.1.17 take place for different position sets: the last position, the last five, the random five.

3.1.17 [CAS] Write procedures that take generator matrices \(G_1\) and \(G_2\) of the codes \(C_1\) and \(C_2\) and return a matrix \(G\) that is the generator matrix of the code \(C\), which is the result of the

- \((u + v|u - v)\)-construction of Proposition 3.1.47;
- \((a + x|b + x|a + b - x)\)-construction of Proposition 3.1.50.

3.1.18 [CAS] Using the previous exercise construct the extended Golay code as in Example 3.1.53. Compare this code with the one returned by \(\text{ExtendedBinaryGolayCode}()\) (in GAP) and \(\text{GolayCode}(GF(2),\text{true})\) (in Magma).

3.1.19 Show by means of an example that the concatenation of an \([3,2,2]\) outer and \([2,2,1]\) inner code gives a \([6,4]\) code of minimum distance 2 or 3 depending on the choice of the basis of the extended field.

3.2 Bounds on codes

We have introduced some parameters of a linear code in the previous sections. In coding theory one of the most basic problems is to find the best value of a parameter when other parameters have been given. In this section, we discuss some bounds on the code parameters.

3.2.1 Singleton bound and MDS codes

The following bound gives us the maximal minimum distance of a code with a given length and dimension. This bound is called the Singleton bound.

**Theorem 3.2.1 (The Singleton Bound)** If \(C\) is an \([n,k,d]\) code, then

\[ d \leq n - k + 1. \]
Proof. Let $H$ be a parity check matrix of $C$. This is an $(n - k) \times n$ matrix of row rank $n - k$. The minimum distance of $C$ is the smallest integer $d$ such that $H$ has $d$ linearly dependent columns, by Proposition 2.3.11. This means that every $d - 1$ columns of $H$ are linearly independent. Hence, the column rank of $H$ is at least $d - 1$. By the fact that the column rank of a matrix is equal to the row rank, we have $n - k \geq d - 1$. This implies the Singleton bound.

\[ \Box \]

Definition 3.2.2 Let $C$ be an $[n, k, d]$ code. If $d = n - k + 1$, then $C$ is called a maximum distance separable code or an MDS code, for short.

Remark 3.2.3 From the Singleton bound, a maximum distance separable code achieves the maximum possible value for the minimum distance given the code length and dimension.

Example 3.2.4 The minimum distance of the the zero code of length $n$ is $n + 1$, by definition. Hence the zero code has parameters $[n, 0, n + 1]$ and is MDS. Its dual is the whole space $\mathbb{F}_q^n$ with parameters $[n, n, 1]$ and is also MDS. The $n$-fold repetition code has parameters $[n, 1, n]$ and its dual is an $[n, n - 1, 2]$ code and both are MDS.

Proposition 3.2.5 Let $C$ be an $[n, k, d]$ code over $\mathbb{F}_q$. Let $G$ be a generator matrix and $H$ a parity check matrix of $C$. Then the following statements are equivalent:

(1) $C$ is an MDS code,

(2) every $(n - k)$-tuple of columns of a parity check matrix $H$ are linearly independent,

(3) every $k$-tuple of columns of a generator matrix $G$ are linearly independent.

Proof. As the minimum distance of $C$ is $d$ any $d - 1$ columns of $H$ are linearly independent, by Proposition 2.3.11. Now $d \leq n - k + 1$ by the Singleton bound. So $d = n - k + 1$ if and only if every $n - k$ columns of $H$ are independent. Hence (1) and (2) are equivalent.

Now let us assume (3). Let $c$ be an element of $C$ which is zero at $k$ given coordinates. Let $c = xG$ for some $x \in \mathbb{F}_q^k$. Let $G'$ be the square matrix consisting of the $k$ columns of $G$ corresponding to the $k$ given zero coordinates of $c$. Then $xG' = 0$. Hence $x = 0$, since the $k$ columns of $G'$ are independent by assumption. So $c = 0$. This implies that the minimum distance of $C$ is at least $n - (k - 1) = n - k + 1$. Therefore $C$ is an $[n, k, n - k + 1]$ MDS code, by the Singleton bound.

Assume that $C$ is MDS. Let $G$ be a generator matrix of $C$. Let $G'$ be the square matrix consisting of $k$ chosen columns of $G$. Let $x \in \mathbb{F}_q^k$ such that $xG' = 0$. Then $c = xG$ is codeword and its weight is at most $n - k$. So $c = 0$, since the minimum distance is $n - k + 1$. Hence $x = 0$, since the rank of $G$ is $k$. Therefore the $k$ columns are independent.

\[ \Box \]

Example 3.2.6 Consider the code $C$ over $\mathbb{F}_5$ of length 5 and dimension 2 with generator matrix

\[
G = \begin{pmatrix}
1 & 1 & 1 & 1 & 1 \\
0 & 1 & 2 & 3 & 4 \\
\end{pmatrix}
\]
Note that while the first row of the generator matrix is the all 1’s vector, the entries of the second row are distinct. Since every codeword of $C$ is a linear combination of the first and second row, the minimum distance of $C$ is at least 5. On the other hand, the second row is a word of weight 4. Hence $C$ is a $[5, 2, 4]$ MDS code. The matrix $G$ is a parity check matrix for the dual code $C^\perp$. All columns of $G$ are nonzero, and every two columns are independent since
\[
\det \begin{pmatrix} 1 & 1 \\ i & j \end{pmatrix} = j - i \neq 0
\]
for all $0 \leq i < j \leq 4$. Therefore, $C^\perp$ is also an MDS code.

In fact, we have the following general result.

**Corollary 3.2.7** The dual of an $[n, k, n-k+1]$ MDS code is an $[n, n-k, k+1]$ MDS code.

**Proof.** The trivial codes are MDS and are dual of each other by Example 3.2.4. Assume $0 < k < n$. Let $H$ be a parity check matrix of an $[n, k, n-k+1]$ MDS code $C$. Then any $n-k$ columns of $H$ are linearly independent, by (2) of Proposition 3.2.5. Now $H$ is a generator matrix of the dual code. Therefore $C^\perp$ is an $[n, n-k, k+1]$ MDS code, since (3) of Proposition 3.2.5 holds.

**Definition 3.2.8** Let $a$ be a vector of $\mathbb{F}_q^k$. Then $V(a)$ is the Vandermonde matrix with entries $a_i^{j-1}$.

**Lemma 3.2.9** Let $a$ be a vector of $\mathbb{F}_q^k$. Then
\[
\det V(a) = \prod_{1 \leq r < s \leq k} (a_s - a_r).
\]

**Proof.** This is left as an exercise.

**Proposition 3.2.10** Let $n \leq q$. Let $a = (a_1, \ldots, a_n)$ be an $n$-tuple of mutually distinct elements of $\mathbb{F}_q$. Let $k$ be an integer such that $0 \leq k \leq n$. Define the matrices $G_k(a)$ and $G'_k(a)$ by
\[
G_k(a) = \begin{pmatrix} 1 & \cdots & 1 \\ a_1 & \cdots & a_n \\ \vdots & \ddots & \vdots \\ a_1^{k-1} & \cdots & a_n^{k-1} \end{pmatrix} \quad \text{and} \quad G'_k(a) = \begin{pmatrix} 1 & \cdots & 1 \\ a_1 & \cdots & a_n \\ \vdots & \ddots & \vdots \\ a_1^{k-1} & \cdots & a_n^{k-1} \end{pmatrix}.
\]

The codes with generator matrix $G_k(a)$ and $G'_k(a)$ are MDS codes.

**Proof.** Consider a $k \times k$ submatrix of $G(a)$. Then this is a Vandermonde matrix and its determinant is not zero by Lemma 3.2.9, since the $a_i$ are mutually distinct. So any system of $k$ columns of $G_k(a)$ is independent. Hence $G_k(a)$ is the generator matrix of an MDS code by Proposition 3.2.5. The proof for $G'_k(a)$ is similar and is left as an exercise.
Remark 3.2.11 The codes defined in Proposition 3.2.10 are called generalized Reed-Solomon codes and are the prime examples of MDS codes. These codes will be treated in Section 8.1. The notion of an MDS code has a nice interpretation of \( n \) points in general position in projective space as we will see in Section 4.3.1. The following proposition shows the existence of MDS codes over \( \mathbb{F}_q \) with parameters \([n, k, n-k+1]\) for all possible values of \( k \) and \( n \) such that \( 0 \leq k \leq n \leq q + 1 \).

Example 3.2.12 Let \( q \) be a power of 2. Let \( n = q + 2 \) and \( a_1, a_2, \ldots, a_q \) be an enumeration of the elements of \( \mathbb{F}_q \). Consider the code \( C \) with generator matrix

\[
\begin{pmatrix}
1 & 1 & \ldots & 1 & 0 & 0 \\
1 & a_2 & \ldots & a_q & 0 & 1 \\
a_1^2 & a_2^2 & \ldots & a_q^2 & 1 & 0
\end{pmatrix}
\]

Then any 3 columns of this matrix are independent, since by the Proposition 3.2.10, the only remaining nontrivial case to check is

\[
\begin{vmatrix}
1 & 1 & 0 \\
a_i & a_j & 1 \\
a_i^2 & a_j^2 & 0
\end{vmatrix} = -(a_j^2 - a_i^2) = (a_i - a_j)^2 \neq 0, \text{ in characteristic 2}
\]

for all \( 1 \leq i < j \leq q - 1 \). Hence \( C \) is a \([q + 2, 3, q]\) code.

Remark 3.2.13 From (3) of Proposition 3.2.5 and Proposition 2.2.22 we see that any \( k \) symbols of the codewords of an MDS code of dimension \( k \) may be taken as message symbols. This is another reason for the name of maximum distance separable codes.

Corollary 3.2.14 Let \( C \) be an \([n, k, d]\) code. Then \( C \) is MDS if and only if for any given \( d \) coordinate positions \( i_1, i_2, \ldots, i_d \), there is a minimum weight codeword with the set of these positions as support. Furthermore two codewords of an MDS code of minimum weight with the same support are a nonzero multiple of each other.

Proof. Let \( G \) be a generator matrix of \( C \). Suppose \( d < n-k+1 \). There exist \( k \) positions \( j_1, j_2, \ldots, j_k \) such that the columns of \( G \) at these positions are independent. The complement of these \( k \) positions consists of \( n-k \) elements and \( d \leq n-k \). Choose a subset \( \{i_1, i_2, \ldots, i_d\} \) of \( d \) elements in this complement. Let \( c \) be a codeword with support that is contained in \( \{i_1, i_2, \ldots, i_d\} \). Then \( c \) is zero at the positions \( j_1, j_2, \ldots, j_k \). Hence \( c \) is 0 and the support of \( c \) is empty. If \( C \) is MDS, then \( d = n-k+1 \). Let \( \{i_1, i_2, \ldots, i_d\} \) be a set of \( d \) coordinate positions. Then the complement of this set consists of \( k-1 \) elements \( j_1, j_2, \ldots, j_{k-1} \). Let \( j_k = i_1 \). Then \( j_1, j_2, \ldots, j_k \) are \( k \) elements that can be used for systematic encoding by Remark 3.2.13. So there is a unique codeword \( c \) such that \( c_j = 0 \) for all \( j = j_1, j_2, \ldots, j_{k-1} \) and \( c_{j_k} = 1 \). Hence \( c \) is a nonzero codeword of weight at most \( d \) and support contained in \( \{i_1, i_2, \ldots, i_d\} \). Therefore \( c \) is a codeword of weight \( d \) and support equal to \( \{i_1, i_2, \ldots, i_d\} \), since \( d \) is the minimum weight of the code.

Furthermore, let \( c' \) be another codeword of weight \( d \) and support equal to
Then \( c'_j = 0 \) for all \( j = j_1, j_2, \ldots, j_{k-1} \) and \( c'_{j_k} \neq 0 \). Then \( c' \) and \( c'_{j_k}c \) are two codewords that coincide at \( j_1, j_2, \ldots, j_k \). Hence \( c' = c'_{j_k}c. \) \( \diamond \)

**Remark 3.2.15** It follows from Corollary 3.2.14 that the number of nonzero codewords of an \([n, k] \) MDS code of minimum weight \( n - k + 1 \) is equal to

\[
(q - 1) \binom{n}{n - k + 1}.
\]

In Section 4.1, we will introduce the weight distribution of a linear code. Using the above result the weight distribution of an MDS code can be completely determined. This will be determined in Proposition 4.4.22.

**Remark 3.2.16** Let \( C \) be an \([n, k, n - k + 1] \) code. Then it is systematic at the first \( k \) positions. Hence \( C \) has a generator matrix of the form \((I_k | A)\). It is left as an exercise to show that every square submatrix of \( A \) is nonsingular. The converse is also true.

**Definition 3.2.17** Let \( n \leq q \). Let \( a, b, r, s \) be vectors of \( \mathbb{F}_q^k \) such that \( a_i \neq b_j \) for all \( i, j \). Then \( C(a, b) \) is the \( k \times k \) Cauchy matrix with entries \( 1/(a_i - b_j) \), and \( C(a, b, r, s) \) is the \( k \times k \) generalized Cauchy matrix with entries \( r_is_j/(a_i - b_j) \). Let \( k \) be an integer such that \( 0 \leq k \leq n \). Let \( A(a) \) be the \( k \times (n - k) \) matrix with entries \( 1/(a_j + k - a_i) \) for \( 1 \leq i \leq k, 1 \leq j \leq n - k \). Then the Cauchy code \( C_k(a) \) is the code with generator matrix \((I_k | A(a))\). If \( r_i \) is not zero for all \( i \), then \( A(a, r) \) is the \( k \times (n - k) \) matrix with entries

\[
\frac{r_{i+k}r_i^{-1}}{a_{j+k} - a_i} \quad \text{for } 1 \leq i \leq k, \ 1 \leq j \leq n - k.
\]

The generalized Cauchy code \( C_k(a, r) \) is the code with generator matrix \((I_k | A(a, r))\).

**Lemma 3.2.18** Let \( a, b, r, s \) be vectors of \( \mathbb{F}_q^k \) such that \( a_i \neq b_j \) for all \( i, j \). Then

\[
\det C(a, b; r, s) = \prod_{i=1}^n r_i \prod_{j=1}^n s_j \prod_{i<j} (a_i - a_j)(b_j - b_i) / \prod_{i, j=1}^n (a_i - b_i).
\]

**Proof.** This is left as an exercise. \( \diamond \)

**Proposition 3.2.19** Let \( n \leq q \). Let \( a \) be an \( n \)-tuple of mutually distinct elements of \( \mathbb{F}_q \), and \( r \) an \( n \)-tuple of nonzero elements of \( \mathbb{F}_q \). Let \( k \) be an integer such that \( 0 \leq k \leq n \). Then the generalized Cauchy code \( C_k(a, r) \) is an \([n, k, n-k+1] \) code.

**Proof.** Every square \( t \times t \) submatrix of \( A \) is Cauchy matrix of the form \( C((a_{i_1}, \ldots, a_{i_t}),(a_{k+j_1}, \ldots, a_{k+j_t}),(b_{i_1}^{-1}, \ldots, b_{i_t}^{-1}),(b_{k+j_1}, \ldots, b_{k+j_t})) \). The determinant of this matrix is not zero by Lemma 3.2.18, since the entries of \( a \) are mutually distinct and the entries of \( r \) are not zero. Hence \((I_k | A(a, r))\) is the generator matrix of an MDS code by Remark 3.2.16. \( \diamond \)

In Section 8.1 it will be shown that generalized Reed-Solomon codes and Cauchy codes are the same.
3.2.2 Griesmer bound

Clearly, the Singleton bound can be viewed as a lower bound on the code length $n$ with given dimension $k$ and minimum distance $d$, that is $n \geq d + k - 1$. In this subsection, we will give another lower bound on the length.

**Theorem 3.2.20 (The Griesmer Bound)** If $C$ is an $[n, k, d]$ code with $k > 0$, then

$$n \geq k - 1 \sum_{i=0}^{k-1} \left\lceil \frac{d}{q^i} \right\rceil.$$ 

Note that the Griesmer bound implies the Singleton bound. In fact, we have $\left\lceil \frac{d}{q^0} \right\rceil = d$ and $\left\lceil \frac{d}{q^i} \right\rceil \geq 1$ for $i = 1, \ldots, k - 1$, which follow the Singleton bound. In the previous Section 3.1 we introduced some methods to construct new codes from a given code. In the following, we give another construction of a new code, which will be used to prove Theorem 3.2.20.

Let $C$ be an $[n, k, d]$ code, and $c$ be a codeword with $w = \text{wt}(c)$. Let $I = \text{supp}(c)$ (see the definition in Subsection 2.1.2). The residual code of $C$ with respect to $c$, denoted by $\text{Res}(C, c)$, is the code of length $n - w$ punctured on all the coordinates of $I$.

**Proposition 3.2.21** Suppose $C$ is an $[n, k, d]$ code over $\mathbb{F}_q$ and $c$ is a codeword of weight $w < \left\lfloor \frac{qd}{q - 1} \right\rfloor$. Then $\text{Res}(C, c)$ is an $[n - w, k - 1, d']$ code with

$$d' \geq d - w + \left\lceil \frac{w}{q} \right\rceil.$$ 

**Proof.** By replacing $C$ by an equivalent code we may assume without loss of the generality that $c = (1, 1, \ldots, 1, 0, \ldots, 0)$ where the first $w$ components are equal to 1 and other components are 0. Clearly, the dimension of $\text{Res}(C, c)$ is less than or equal to $k - 1$. If the dimension is strictly less than $k - 1$, then there must be a nonzero codeword in $C$ of the form $x = (x_1, \ldots, x_w, 0, \ldots, 0)$, where not all the $x_i$ are the same. There exists $\alpha \in \mathbb{F}_q$ such that at least $w/q$ coordinates of $(x_1, \ldots, x_w)$ equal to $\alpha$. Thus,

$$d \leq \text{wt}(x - \alpha c) \leq w - w/q = w(q - 1)/q,$$

which contradicts the assumption on $w$. Hence $\dim \text{Res}(C, c) = k - 1$.

Next, consider the minimum distance. Let $(x_{w+1}, \ldots, x_n)$ be any nonzero codeword in $\text{Res}(C, c)$, and $x = (x_1, \ldots, x_w, x_{w+1}, \ldots, x_n)$ be a corresponding codeword in $C$. There exists $\alpha \in \mathbb{F}_q$ such that at least $w/q$ coordinates of $(x_1, \ldots, x_w)$ equal $\alpha$. Therefore,

$$d \leq \text{wt}(x - \alpha c) \leq w - w/q + \text{wt}(x_{w+1}, \ldots, x_n)).$$

Thus every nonzero codeword of $\text{Res}(C, c)$ has weight at least $d - w + \lfloor w/q \rfloor$. 

**Proof of Theorem 3.2.20.** We will prove the theorem by mathematical induction on $k$. If $k = 1$, the inequality that we want to prove is $n \geq d$, which
is obviously true. Now suppose $k > 1$. Let $c$ be a codeword of weight $d$. Using Proposition 3.2.21, $\text{Res}(C, c)$ is an $[n - d, k - 1, d']$ code with $d' \geq \lfloor d/q \rfloor$. Applying the inductive assumption to $\text{Res}(C, c)$, we have

$$n - d \geq \sum_{i=0}^{k-2} \left\lfloor \frac{d'}{q^i} \right\rfloor \geq \sum_{i=0}^{k-2} \left\lfloor \frac{d}{q^{i+1}} \right\rfloor.$$ 

The Griesmer bound follows.

### 3.2.3 Hamming bound

In practical applications, given the length and the minimum distance, the codes which have more codewords (in other words, codes of larger size) are often preferred. A natural question is, what is the maximal possible size of a code given the length and minimum distance. Denote by $A_q(n, d)$ the maximum number of codewords in any code over $\mathbb{F}_q$ (which can be linear or nonlinear) of length $n$ and minimum distance $d$. The maximum when restricted to linear codes is denoted by $B_q(n, d)$. Clearly $B_q(n, d) \leq A_q(n, d)$. The following is a well-known upper bound for $A_q(n, d)$.

**Remark 3.2.22** Denote the number of vectors in $B_t(x)$ the ball of radius $t$ around a given vector $x \in \mathbb{F}_q^n$ as defined in 2.1.12 by $V_q(n, t)$. Then

$$V_q(n, t) = \sum_{i=0}^{t} \binom{n}{i} (q - 1)^i$$

by Proposition 2.1.13.

**Theorem 3.2.23** (Hamming or sphere-packing bound)

$$B_q(n, d) \leq A_q(n, d) \leq \frac{q^n}{V_q(n, t)},$$

where $t = \lfloor (d - 1)/2 \rfloor$.

**Proof.** Let $C$ be any code over $\mathbb{F}_q$ (which can be linear or nonlinear) of length $n$ and minimum distance $d$. Denote by $M$ the number of codewords of $C$. Since the distance between any two codewords is greater than or equal to $d \geq 2t + 1$, the balls of radius $t$ around the codewords must be disjoint. From Proposition 2.1.13, each of these $M$ balls contain $\sum_{i=0}^{t} (q - 1)^i \binom{n}{i}$ vectors. The total number of vectors in the space $\mathbb{F}_q^n$ is $q^n$. Thus, we have

$$M \cdot V_q(n, t) \leq q^n.$$ 

As $C$ is any code with length $n$ and minimum distance $d$, we have established the theorem.

**Definition 3.2.24** The covering radius $\rho(C)$ of a code $C$ of length $n$ over $\mathbb{F}_q$ is defined to be the smallest integer $s$ such that

$$\bigcup_{c \in C} B_t(c) = \mathbb{F}_q^n,$$

where $t = \lfloor (d - 1)/2 \rfloor$. 

\[ \diamond \]
that is every vector $F_q^n$ is in the union of the balls of radius $t$ around the codewords. A code is of covering radius $\rho$ is called perfect if the balls $B_\rho(c), c \in C$ are mutually disjoint.

**Theorem 3.2.25 (Sphere-covering bound)** Let $C$ be a code of length $n$ with $M$ codewords and covering radius $\rho$. Then

$$M \cdot V_q(n, \rho) \geq q^n.$$  

**Proof.** By definition

$$\bigcup_{c \in C} B_\rho(c) = F_q^n,$$  

now $|B_\rho(c)| = V_q(n, \rho)$ for all $c$ in $C$ by Proposition 2.1.13. So $M \cdot V_q(n, \rho) \geq q^n$.

**Example 3.2.26** If $C = F_q^n$, then the balls $B_0(c) = \{c\}, c \in C$ cover $F_q^n$ and are mutually disjoint. So $F_q^n$ is perfect and has covering radius 0.

If $C = \{0\}$, then the ball $B_n(0)$ covers $F_q^n$ and there is only one codeword. Hence $C$ is perfect and has covering radius $n$.

Therefore the trivial codes are perfect.

**Remark 3.2.27** It is easy to see that

$$\rho(C) = \max_{x \in F_q^n} \min_{c \in C} d(x, c).$$

Let $e(C) = \lfloor (d(C) - 1)/2 \rfloor$. Then obviously $e(C) \leq \rho(C)$. Let $C$ be code of length $n$ and minimum distance $d$ with more than one codeword. Then $C$ is a perfect code if and only if $\rho(C) = e(C)$.

**Proposition 3.2.28** The following codes are perfect:

1. the trivial codes,
2. $(2e + 1)$-fold binary repetition code,
3. the Hamming code,
4. the binary and ternary Golay code.

**Proof.** (1) The trivial codes are perfect as shown in Example 3.2.26.

(2) The $(2e + 1)$-fold binary repetition code consists of two codewords, has minimum distance $d = 2e + 1$ and error-correcting capacity $e$. Now

$$2^{2e+1} = \sum_{i=0}^{2e+1} \binom{2e+1}{i} = \sum_{i=0}^{e} \binom{2e+1}{i} + \sum_{i=0}^{e} \binom{2e+1}{e+1+i}$$

and $\binom{2e+1}{e+1+i} = \binom{2e+1}{i}$. So $2\sum_{i=0}^{e} \binom{2e+1}{i} = 2^{2e+1}$. Therefore the covering radius is $e$ and the code is perfect.

(3) From Definition 2.3.13 and Proposition 2.3.14, the $q$-ary Hamming code $H_r(q)$ is an $[n, k, d]$ code with

$$n = \frac{q^r - 1}{q - 1}, \quad k = n - r, \quad \text{and} \quad d = 3.$$
For this code, \( t = 1, n = k + r \), and the number of codewords is \( M = q^k \). Thus,

\[
M \left( 1 + (q - 1) \binom{n}{1} \right) = M(1 + (q - 1)n) = Mq^r = q^{k+r} = q^n.
\]

Therefore, \( \mathcal{H}_r(q) \) is a perfect code.

(4) It is left to the reader to show that the binary and ternary Golay codes are perfect.

### 3.2.4 Plotkin bound

The Plotkin bound is an upper bound on \( A_q(n,d) \) which is valid when \( d \) is large enough comparing with \( n \).

**Theorem 3.2.29 (Plotkin bound)** Let \( C \) be an \((n, M, d)\) code over \( \mathbb{F}_q \) such that \( qd > (q - 1)n \). Then

\[
M \leq \left\lfloor \frac{qd}{qd - (q - 1)n} \right\rfloor.
\]

**Proof.** We calculate the following sum

\[
S = \sum_{x \in C} \sum_{y \in C} d(x, y)
\]

in two ways. First, since for any \( x, y \in C \) and \( x \neq y \), the distance \( d(x, y) \geq d \), we have

\[
S \geq M(M - 1)d.
\]

On the other hand, let \( M \) be the \( M \times n \) matrix consisting of the codewords of \( C \). For \( i = 1, \ldots, n \), let \( n_{i,\alpha} \) be the number of times \( \alpha \in \mathbb{F}_q \) occurs in column \( i \) of \( M \). Clearly, \( \sum_{\alpha \in \mathbb{F}_q} n_{i,\alpha} = M \) for any \( i \). Now, we have

\[
S = \sum_{i=1}^{n} \sum_{\alpha \in \mathbb{F}_q} n_{i,\alpha}(M - n_{i,\alpha}) = nM^2 - \sum_{i=1}^{n} \sum_{\alpha \in \mathbb{F}_q} n_{i,\alpha}^2.
\]

Using the Cauchy-Schwartz inequality,

\[
\sum_{\alpha \in \mathbb{F}_q} n_{i,\alpha}^2 \geq \frac{1}{q} \left( \sum_{\alpha \in \mathbb{F}_q} n_{i,\alpha} \right)^2.
\]

Thus,

\[
S \leq nM^2 - \sum_{i=1}^{n} \frac{1}{q} \left( \sum_{\alpha \in \mathbb{F}_q} n_{i,\alpha} \right)^2 = n(1 - 1/q)M^2.
\]

Combining the above two inequalities on \( S \), we prove the theorem.
CHAPTER 3. CODE CONSTRUCTIONS AND BOUNDS

Example 3.2.30 Consider the simplex code $S_3(3)$, that is the dual code of the Hamming code $H_3(3)$ over $\mathbb{F}_3$ of Example 2.3.15. This is an $[13, 3, 9]$ code which has $M = 3^3 = 27$ codewords. Every nonzero codeword in this code has Hamming weight 9, and $d(x, y) = 9$ for any distinct codewords $x$ and $y$. Thus, $qd = 27 > 26 = (q - 1)n$. Since

$$\left\lfloor \frac{qd}{qd - (q - 1)n} \right\rfloor = 27 = M,$$

this code achieves the Plotkin bound.

Remark 3.2.31 For a code, if all the nonzero codewords have the same weight, we call it a constant weight code; if the distances between any two distinct codewords are same, we call it an equidistant code. For a linear code, it is a constant weight code if and only if it is an equidistant code. From the proof of Theorem 3.2.29, only constant weight and equidistant codes can achieve the Plotkin bound. So the simplex code $S_r(q)$ achieves the Plotkin bound by Proposition 2.3.16.

Remark 3.2.32 ***Improved Plotkin Bound in the binary case.***

3.2.5 Gilbert and Varshamov bounds

The Hamming and Plotkin bounds give upper bounds for $A_q(n, d)$ and $B_q(n, d)$. In this subsection, we discuss lower bounds for these numbers. Since $B_q(n, d) \leq A_q(n, d)$, each lower bound for $B_q(n, d)$ is also a lower bound for $A_q(n, d)$.

Theorem 3.2.33 (Gilbert bound)

$$\log_q(A_q(n, d)) \geq n - \log_q(V_q(n, d - 1)).$$

**Proof.** Let $C$ be a code over $\mathbb{F}_q$, not necessarily linear of length $n$ and minimum distance $d$, which has $M = A_q(n, d)$ codewords. If

$$M \cdot V_q(n, d - 1) < q^n,$$

then the union of the balls of radius $d - 1$ of all codewords in $C$ is not equal to $\mathbb{F}_q^n$ by Proposition 2.1.13. Take $x \in \mathbb{F}_q^n$ outside this union. Then $d(x, c) \geq d$ for all $c \in C$. So $C \cup \{x\}$ is a code of length $n$ with $M + 1$ codewords and minimum distance $d$. This contradicts the maximality of $A_q(n, d)$. Hence

$$A_q(n, d) \cdot V_q(n, d - 1) \geq q^n.$$

⋄

In the following the greedy algorithm one can construct a linear code of length $n$, minimum distance $\geq d$, and dimension $k$ and therefore the number of codewords as large as possible.

Theorem 3.2.34 Let $n$ and $d$ be integers satisfying $2 \leq d \leq n$. If

$$k \leq n - \log_q(1 + V_q(n - 1, d - 2)),$$  \hspace{1cm} (3.1)

then there exists an $[n, k]$ code over $\mathbb{F}_q$ with minimum distance at least $d$. 

3.2. BOUNDS ON CODES

Proof. Suppose \( k \) is an integer satisfying the inequality (3.1), which is equivalent to

\[
V_q(n - 1, d - 2) < q^{n-k}.
\]

(3.2)

We construct by induction the columns \( h_1, \ldots, h_n \in \mathbb{F}_q^{n-k} \) of an \((n-k) \times n\) matrix \( H \) over \( \mathbb{F}_q \) such that every \( d-1 \) columns of \( H \) are linearly independent. Choose for \( h_1 \) any nonzero vector. Suppose that \( j < n \) and \( h_1, \ldots, h_j \) are chosen such that any \( d-1 \) of them are linearly independent. Choose \( h_{j+1} \) such that \( h_{j+1} \) is not a linear combination of any \( d-2 \) or fewer of the vectors \( h_1, \ldots, h_j \).

The above procedure is a greedy algorithm. We now prove the correctness of the algorithm, by induction on \( j \). When \( j = 1 \), it is trivial that there exists a nonzero vector \( h_1 \). Suppose that \( j < n \) and any \( d-1 \) of \( h_1, \ldots, h_j \) are linearly independent. The number of different linear combinations of \( d-2 \) or fewer of the \( h_1, \ldots, h_j \) is

\[
\sum_{i=0}^{d-2} \binom{j}{i} (q-1)^i \leq \sum_{i=0}^{d-2} \binom{n-1}{i} (q-1)^i = V_q(n-1, d-2).
\]

Hence under the condition (3.2), there always exists a vector \( h_{j+1} \) which is not a linear combination of \( d-2 \) or fewer of \( h_1, \ldots, h_j \).

By induction, we find \( h_1, \ldots, h_n \) such that \( h_j \) is not a linear combination of any \( d-2 \) or fewer of the vectors \( h_1, \ldots, h_{j-1} \). Hence, every \( d-1 \) of \( h_1, \ldots, h_n \) are linearly independent.

The null space of \( H \) is a code \( C \) of dimension at least \( k \) and minimum distance at least \( d \) by Proposition 2.3.11. Let \( C' \) be be a subcode of \( C \) of dimension \( k \). Then the minimum distance of \( C' \) is at least \( d \). \( \diamond \)

Corollary 3.2.35 (Varshamov bound)

\[
\log_q B_q(n, d) \geq n - \lceil \log_q(1 + V_q(n-1, d-2)) \rceil.
\]

Proof. The largest integer \( k \) satisfying (3.1) of Theorem 3.2.34 is given by the right hand side of the inequality. \( \diamond \)

In the next subsection, we will see that the Gilbert bound and the Varshamov bound are the same asymptotically. In the literature, sometimes any of them is called the Gilbert-Varshamov bound. The resulting asymptotic bound is called the asymptotic Gilbert-Varshamov bound.

3.2.6 Exercises

3.2.1 Show that for an arbitrary code, possibly nonlinear, of length \( n \) over an alphabet with \( q \) elements with \( M \) codewords and minim distance \( d \) the following form of the Singleton bounds holds: \( M \leq q^{n+1-d} \).

3.2.2 Let \( C \) be an \([n, k]\) code. Let \( d^\perp \) be the minimum distance of \( C^\perp \). Show that \( d^\perp \leq k + 1 \), and that equality holds if and only if \( C \) is MDS.
3.2.3 Give a proof of the formula in Lemma 3.2.9 of the determinant of a Vandermonde matrix.

3.2.4 Prove that the code $G'(a)$ in Proposition 3.2.10 is MDS.

3.2.5 Let $C$ be an $[n, k, d]$ code over $\mathbb{F}_q$. Prove that the number of codewords of minimum weight $d$ is divisible by $q - 1$ and is at most equal to $(q - 1)(\binom{n}{d})$. Show that $C$ is MDS in case equality holds.

3.2.6 Give a proof of Remark 3.2.16.

3.2.7 Give a proof of the formula in Lemma 3.2.18 of the determinant of a Cauchy matrix.

3.2.8 Let $C$ be a binary MDS code. If $C$ is not trivial, then it is a repetition code or an even weight code.

3.2.9 [20] ***Show that the code $C_1$ in Proposition 3.2.10 is self-orthogonal if $n = q$ and $k \leq n/2$. Self-dual ***

3.2.10 [CAS] Take $q = 256$ in Proposition 3.2.10 and construct the matrices $G_{10}(a)$ and $G_{10}(a')$. Construct the corresponding codes with these matrices as generator matrices. Show that these codes are MDS by using commands `IsMDSCode` in GAP and `IsMDS` in Magma.

3.2.11 Give a proof of the statements made in Remark 3.2.27.

3.2.12 Show that the binary and ternary Golay codes are perfect.

3.2.13 Let $C$ be the binary $[7, 4, 3]$ Hamming code. Let $D$ be the $\mathbb{F}_4$ linear code with the same generator matrix as $C$. Show that $\rho(C) = 2$ and $\rho(D) = 3$.

3.2.14 Let $C$ be an $[n, k]$ code. let $H$ be a parity check matrix of $C$. Show that $\rho(C)$ is the minimal number $\rho$ such that $x^T$ is a linear combination of at most $\rho$ columns of $H$ for every $x \in \mathbb{F}_q^{n-k}$. Show that the redundancy bound: $\rho(C) \leq n - k$.

3.2.15 Give an estimate of the complexity of finding a code satisfying (3.1) of Theorem 3.2.34 by the greedy algorithm.

3.3 Asymptotically good codes

***

3.3.1 Asymptotic Gilbert-Varshamov bound

In practical applications, sometimes long codes are preferred. For an infinite family of codes, a measure of the goodness of the family of codes is whether the family contains so-called asymptotically good codes.
3.3. ASYMPTOTICALLY GOOD CODES

Definition 3.3.1 An infinite sequence $\mathcal{C} = \{C_i\}_{i=1}^{\infty}$ of codes $C_i$ with parameters $[n_i, k_i, d_i]$ is called asymptotically good, if $\lim_{i \to \infty} n_i = \infty$, and

$$R(\mathcal{C}) = \liminf_{i \to \infty} \frac{k_i}{n_i} > 0 \quad \text{and} \quad \delta(\mathcal{C}) = \liminf_{i \to \infty} \frac{d_i}{n_i} > 0.$$ 

Using the bounds that we introduced in the previous subsection, we will prove the existence of asymptotically good codes.

Definition 3.3.2 Define the $q$-ary entropy function $H_q$ on $[0, (q - 1)/q]$ by

$$H_q(x) = \begin{cases} \frac{1}{n} \log_q (\theta n) & = \theta \\ 0, & \text{if } x = 0. \end{cases}$$

The function $H_q(x)$ is increasing on $[0, (q - 1)/q]$. The function $H_2(x)$ is the entropy function.

Lemma 3.3.3 Let $q \geq 2$ and $0 \leq \theta \leq (q - 1)/q$. Then

$$\lim_{n \to \infty} \frac{1}{n} \log_q (\theta n) = H_q(\theta).$$

Proof. Since $\theta n - 1 < [\theta n] \leq \theta n$, we have

$$\lim_{n \to \infty} \frac{1}{n} \theta n = \theta \quad \text{and} \quad \lim_{n \to \infty} \frac{1}{n} \log_q 1 = 0. \quad (3.3)$$

Now we are going to prove the following equality

$$\lim_{n \to \infty} \frac{1}{n} \log_q \left( \frac{n}{[\theta n]} \right) = -\theta \log_q \theta - (1 - \theta) \log_q (1 - \theta). \quad (3.4)$$

To this end we introduce the little-$o$ notation and use the Stirling Formula.

$$\log n! = \left( n + \frac{1}{2} \right) \log n - n + \frac{1}{2} \log(2n) + o(1), \quad (n \to \infty).$$

For two functions $f(n)$ and $g(n)$, $f(n) = o(g(n))$ means for all $c > 0$ there exists some $k > 0$ such that $0 \leq f(n) < cg(n)$ for all $n \geq k$. The value of $k$ must not depend on $n$, but may depend on $c$. Thus, $o(1)$ is a function of $n$, which tends to 0 when $n \to \infty$. By the Stirling Formula, we have

$$\frac{1}{n} \log_q \left( \frac{n}{[\theta n]} \right) = \frac{1}{n} (\log_q n! - \log_q [\theta n]! - \log_q (n! - [\theta n]!) =

= \log_q n - \theta \log_q [\theta n] - (1 - \theta) \log_q (n! - [\theta n]) + o(1) =

= -\theta \log_q \theta - (1 - \theta) \log_q (1 - \theta) + o(1).$$

Thus (3.4) follows.

From the definition we have

$$\left( \frac{n}{[\theta n]} \right) (q - 1)^{\theta n} \leq V_q(n, [\theta n]) \leq (1 + [\theta n]) \left( \frac{n}{[\theta n]} \right) (q - 1)^{\theta n}. \quad (3.5)$$

From the right-hand part of (3.5) we have

$$\log_q V_q(n, [\theta n]) \leq \log_q (1 + [\theta n]) + \log_q \left( \frac{n}{[\theta n]} \right) + [\theta n] \log_q (q - 1).$$
By (3.3) and (3.4), we have
\[
\lim_{n \to \infty} \frac{1}{n} \log_q V_q(n, \lceil \theta n \rceil) \leq \theta \log_q (q - 1) - \theta \log_q \theta - (1 - \theta) \log_q (1 - \theta).
\]  
(3.6)

The right hand side is equal to \( H_q(\theta) \) by definition. Similarly, using the left-hand part of (3.5) we prove
\[
\lim_{n \to \infty} \frac{1}{n} \log_q V_q(n, \lceil \theta n \rceil) \geq H_q(\theta).
\]  
(3.7)

Combining (3.6) and (3.7), we obtain the result.

Now we are ready to prove the existence of asymptotically good codes. Specifically, we have the following stronger result.

**Theorem 3.3.4** Let \( 0 < \theta < (q - 1)/q \). Then there exists an asymptotically good sequence \( C \) of codes such that \( \delta(C) = \theta \) and \( R(C) = 1 - H_q(\theta) \).

**Proof.** Let \( 0 < \theta < (q - 1)/q \). Let \( \{n_i\}_{i=1}^{\infty} \) be a sequence of positive integers with \( \lim_{i \to \infty} n_i = \infty \), for example, we can take \( n_i = i \). Let \( d_i = \lceil \theta n_i \rceil \) and
\[
k_i = n_i - \left\lfloor \log_q (1 + V_q(n_i - 1, d_i - 2)) \right\rfloor.
\]
By Theorem 3.2.34 and the Varshamov bound, there exists a sequence \( C = \{C_i\}_{i=1}^{\infty} \) of \([n_i, k_i, d_i]\) codes \( C_i \) over \( F_q \).

Clearly \( \delta(C) = \theta > 0 \) for this sequence of \( q \)-ary codes. We now prove \( R(C) = 1 - H_q(\theta) \). To this end, we first use Lemma 3.3.3 to prove the following equation:
\[
\lim_{i \to \infty} \frac{1}{n_i} \log_q (1 + V_q(n_i - 1, d_i - 2)) = H_q(\theta).
\]  
(3.8)

First, we have
\[
1 + V_q(n_i - 1, d_i - 2) \leq V_q(n_i, d_i).
\]
By Lemma 3.3.3, we have
\[
\limsup_{i \to \infty} \frac{1}{n_i} \log_q (1 + V_q(n_i - 1, d_i - 2)) \leq \lim_{i \to \infty} \frac{1}{n_i} \log_q V_q(n_i, d_i) = H_q(\theta).
\]  
(3.9)

Let \( \delta = \max\{1, \lceil 3/\theta \rceil\} \), \( m_i = n_i - \delta \) and \( e_i = \lceil \theta m_i \rceil \). Then,
\[
d_i - 2 = \lceil \theta n_i \rceil - 2 > \theta n_i - 3 \\
\geq \theta (n_i - \delta) = \theta m_i \\
\geq e_i
\]
and \( n_i - 1 \geq n_i - \delta = m_i \). Therefore
\[
\frac{1}{n_i} \log_q V_q(n_i - 1, d_i - 2) \geq \frac{1}{m_i + \delta} \log_q V_q(e_i, m_i) = \frac{1}{m_i} \log_q V_q(e_i, m_i) \cdot \frac{m_i}{m_i + \delta}
\]
3.3. ASYMPTOTICALLY GOOD CODES

Since $\delta$ is a constant and $m_i \to \infty$, we have $\lim_{i \to \infty} m_i/(m_i + \delta) = 1$. Again by Lemma 3.3.3, we have that the right hand side of the above inequality tends to $H_q(\theta)$. It follows that

$$\liminf_{i \to \infty} \frac{1}{n_i} \log_q (1 + V_q(n_i - 1, d_i - 2)) \geq H_q(\theta). \quad (3.10)$$

By inequalities (3.9) and (3.10), we obtain (3.8).

Now by (3.8), we have

$$R(\mathcal{C}) = \lim_{i \to \infty} \frac{k_i}{n_i} = 1 - \lim_{i \to \infty} \frac{1}{n_i} \left[ \log_q (1 + V_q(n_i - 1, d_i - 2)) \right] = 1 - H_q(\theta),$$

and $1 - H_q(\theta) > 0$, since $\theta < (q - 1)/q$.

So the sequence $\mathcal{C}$ of codes satisfying Theorem 3.3.4 is asymptotically good. However, asymptotically good codes are not necessarily codes satisfying the conditions in Theorem 3.3.4.

The number of codewords increases exponentially with the code length. So for large $n$, instead of $A_q(n, d)$ the following parameter is used

$$\alpha(\theta) = \limsup_{n \to \infty} \frac{\log_q A_q(n, \theta n)}{n}.$$  

Since $A_q(n, \theta n) \geq B_q(n, \theta n)$ and for a linear code $C$ the dimension $k = \log_q |C|$, a straightforward consequence of Theorem 3.3.4 is the following asymptotic bound.

**Corollary 3.3.5** (Asymptotically Gilbert-Varshamov bound) Let $0 \leq \theta \leq (q - 1)/q$. Then

$$\alpha(\theta) \geq 1 - H_q(\theta).$$

Not that both the Gilbert and Varshamov bound that we introduced in the previous subsection imply the asymptotically Gilbert-Varshamov bound.

***Manin $\alpha_q(\delta)$ is a decreasing continuous function. picture ***

3.3.2 Some results for the generic case

In this section we investigate the parameters of "generic" codes. It turns out that almost all codes have the same minimum distance and covering radius with the length $n$ and dimension $k = nR, 0 < R < 1$ fixed. By "almost all" we mean that as $n$ tends to infinity, the fraction of $[n, nR]$ codes that do not have "generic" minimum distance and covering radius tends to 0.

**Theorem 3.3.6** Let $0 < R < 1$, then almost all $[n, nR]$ codes over $\mathbb{F}_q$ have

- minimum distance $d_0 := nH_q^{-1}(1 - R) + o(n)$
- covering radius $d_0(1 + o(1))$

Here $H_q$ is a $q$-ary entropy function.

**Theorem 3.3.7** ***it gives a number of codewords that project on a given $k$-set. Handbook of Coding theory, p.691. ***
3.3.3 Exercises

3.4 Notes

Puncturing and shortening at arbitrary sets of positions and the duality theorem is from Simonis [?].

Golay code, Turyn [?] construction, Pless handbook [?].

MacWillimas

In 1973 by J. H. van Lint and A. Tietavainen theorem in regards to perfect codes:

- puncturing gives the binary [23,12,7] Golay code, which is cyclic.
- automorphism group of (extended) Golay code.
- (extended) ternary Golay code.
- designs and Golay codes.
- lattices and Golay codes.

repeated decoding of product code (Hoeholdt-Justesen).

Singleton defect $s(C) = n + 1 - k - d$

$s(C) \geq 0$ and equality holds if and only if $C$ is MDS.

$s(C) = 0$ if and only if $s(C^\perp) = 0$.

Example where $s(C) = 1$ and $s(C^\perp) > 1$.

Almost MDS and near MDS.

Genus $g = \max \{s(C), s(C^\perp)\}$ in 4.1. If $k \geq 2$, then $d \leq q(s+1)$. If $k \geq 3$ and $d = q(s+1)$, then $s+1 \leq q$.

Faldum-Willems, de Boer, Dodunekov-Langev, relation with Griesmer bound.
Chapter 4

Weight enumerator

Relinde Jurrius, Ruud Pellikaan and Xin-Wen Wu

The weight enumerator of a code is introduced and a random coding argument gives a proof of Shannon’s theorem.

4.1 Weight enumerator

Apart from the minimum Hamming weight, a code has other important invariants. In this section, we will introduce the weight spectrum and the generalized weight spectrum of a code.

4.1.1 Weight spectrum

The weight spectrum of a code is an important invariant, which provides useful information for both the code structure and practical applications of the code.

Definition 4.1.1 Let $C$ be a code of length $n$. The weight spectrum, also called the weight distribution is the following set

$$\{(w, A_w) \mid w = 0, 1, \ldots, n\}$$

where $A_w$ denotes the number of codewords in $C$ of weight $w$.

The so-called weight enumerator is a convenient representation of the weight spectrum.

Definition 4.1.2 The weight enumerator of $C$ is defined as the following polynomial

$$W_C(Z) = \sum_{w=0}^{n} A_w Z^w.$$ 

The homogeneous weight enumerator of $C$ is defined as

$$W_C(X, Y) = \sum_{w=0}^{n} A_w X^{n-w} Y^w.$$
Remark 4.1.3 Note that \( W_C(Z) \) and \( W_C(X, Y) \) are equivalent in representing the weight spectrum. They determine each other uniquely by the following equations

\[
W_C(Z) = W_C(1, Z)
\]

and

\[
W_C(X, Y) = X^n W_C(X^{-1} Y).
\]

Given the weight enumerator or the homogeneous weight enumerator, the weight spectrum is determined completely by the coefficients.

Clearly, the weight enumerator and homogeneous weight enumerator can be written in another form, that is

\[
W_C(Z) = \sum_{c \in C} Z^{wt(c)} \tag{4.1}
\]

and

\[
W_C(X, Y) = \sum_{c \in C} X^{n-wt(c)} Y^{wt(c)}. \tag{4.2}
\]

Example 4.1.4 The zero code has one codeword, and its weight is zero. Hence the homogeneous weight enumerator of this code is \( W_{\{0\}}(X, Y) = X^n \). The number of words of weight \( w \) in the trivial code \( F_n^q \) is \( A_w = \binom{n}{w} (q - 1)^w \). So

\[
W_{F_n^q}(X, Y) = \sum_{w=0}^{n} \binom{n}{w} (q - 1)^w X^{n-w} Y^w = (X + (q - 1)Y)^n.
\]

Example 4.1.5 The \( n \)-fold repetition code \( C \) has homogeneous weight enumerator

\[
W_C(X, Y) = X^n + (q - 1)Y^n.
\]

In the binary case its dual is the even weight code. Hence it has homogeneous weight enumerator

\[
W_C(X, Y) = \sum_{t=0}^{\lfloor n/2 \rfloor} \binom{n}{2t} X^{n-2t} Y^{2t} = \frac{1}{2} ((X + Y)^n + (X - Y)^n).
\]

Example 4.1.6 The nonzero entries of the weight distribution of the \([7,4,3]\) binary Hamming code are given by \( A_0 = 1, A_3 = 7, A_4 = 7, A_7 = 1 \), as is seen by inspecting the weights of all 16 codewords. Hence its homogeneous weight enumerator is

\[
X^7 + 7X^4 Y^3 + 7X^2 Y^4 + Y^7.
\]

Example 4.1.7 The simplex code \( S_r(q) \) is a constant weight code by Proposition 2.3.16 with parameters \([[(q^r - 1)/(q - 1), r, q^{r-1}]\). Hence its homogeneous weight enumerator is

\[
W_{S_r(q)}(X, Y) = X^n + (q^r - 1)X^{n-q^{r-1}} Y^{q^{r-1}}.
\]
4.1. WEIGHT ENUMERATOR

Remark 4.1.8 Let $C$ be a linear code. Then $A_0 = 1$ and the minimum distance $d(C)$ which is equal to the minimum weight, is determined by the weight enumerator as follows:

$$d(C) = \min \{ i \mid A_i \neq 0, i > 0 \}.$$  

It also determines the dimension $k(C)$, since

$$W_C(1, 1) = \sum_{w=0}^{n} A_w = q^{k(C)}.$$  

Example 4.1.9 The Hamming code over $\mathbb{F}_q$ of length $n = (q^r - 1)/(q - 1)$ has parameters $[n, n - r, 3]$ and is perfect with covering radius 1 by Proposition 3.2.28. The following recurrence relation holds for the weight distribution $(A_0, A_1, \ldots, A_n)$ of these codes:

$$\binom{n}{w} (q - 1)^w = A_{w-1}(n - w + 1)(q - 1) + A_w(1 + w(q - 2)) + A_{w+1}(w + 1)$$  

for all $w$. This is seen as follows. Every word $y$ of weight $w$ is at distance at most 1 to a unique codeword $c$, and such a codeword has possible weights $w - 1$, $w$ or $w + 1$. Let $c$ be a codeword of weight $w - 1$, then there are $n - w + 1$ possible positions $j$ in the complement of the support of $c$ where $c_j = 0$ could be changed into a nonzero element in order to get the word $y$ of weight $w$. Similarly, let $c$ be a codeword of weight $w$, then either $y = c$ or there are $w$ possible positions $j$ in the support of $c$ where $c_j$ could be changed into another nonzero element to get $y$. Finally, let $c$ be a codeword of weight $w + 1$, then there are $w + 1$ possible positions $j$ in the support of $c$ where $c_j$ could be changed into zero to get $y$. Multiply the recurrence relation with $Z^w$ and sum over $w$. Let $W(Z) = \sum_w A_w Z^w$. Then

$$(1 + (q - 1)Z)^n = (q - 1)nZW(Z) - (q - 1)Z^2W'(Z) + W(Z) + (q - 2)ZW'(Z) + W'(Z),$$  

since

$$\sum_w \binom{n}{w} (q - 1)^w Z^w = (1 + (q - 1)Z)^n,$$

$$\sum_w (w + 1)A_{w+1} Z^w = W'(Z),$$

$$\sum_w wA_w Z^w = ZW'(Z),$$

$$\sum_w (w - 1)A_{w-1} Z^w = Z^2W'(Z).$$  

Therefore $W(Z)$ satisfies the following ordinary first order differential equation:

$$((q - 1)Z^2 - (q - 2)Z - 1)W'(Z) - (1 + (q - 1)nZ)W(Z) + (1 + (q - 1)Z)^n = 0.$$  

The corresponding homogeneous differential equation is separable:

$$\frac{W'(Z)}{W(Z)} = \frac{1 + (q - 1)nZ}{(q - 1)Z^2 - (q - 2)Z - 1}$$  

and has general solution:

$$W_h(Z) = C(Z - 1)^{q - 1}((q - 1)Z + 1)^{n - q - 1},$$
where \( C \) is some constant. A particular solution is given by:

\[
P(Z) = \frac{1}{q^r}(1 + (q - 1)Z)^n.
\]

Therefore the solution that satisfies \( W(0) = 1 \) is equal to

\[
W(Z) = \frac{1}{q^r}(1 + (q - 1)Z)^n + \frac{q^r - 1}{q^r}(Z - 1)^{q^r - 1}((q - 1)Z + 1)^{n - q^r - 1}.
\]

To prove that the weight enumerator of a perfect code is completely determined by its parameters we need the following lemma.

**Lemma 4.1.10** The number \( N_q(n, v, w, s) \) of words in \( F_q^n \) of weight \( w \) that are at distance \( s \) from a given word of weight \( v \) does not depend on the chosen word and is equal to

\[
N_q(n, v, w, s) = \sum_{i+j+k=s, v+i-j=w} \binom{n-v}{k} \binom{v-i}{j} \binom{(q-2)q-1}{k}.
\]

**Proof.** Consider a given word \( x \) of weight \( v \). Let \( y \) be a word of weight \( w \) and distance \( s \) to \( x \). Suppose that \( y \) has \( k \) nonzero coordinates in the complement of the support of \( x \), \( j \) zero coordinates in the support of \( x \), and \( i \) nonzero coordinates in the support of \( x \) that are distinct from the coordinates of \( x \). Then \( s = d(x, y) = i + j + k \) and \( wt(y) = w = v + k - j \).

There are \( \binom{n}{k} \) possible subsets of \( k \) elements in the complement of the support of \( x \) and there are \( (q-1)^k \) possible choices for the nonzero symbols at the corresponding \( k \) coordinates.

There are \( \binom{i}{j} \) possible subsets of \( i \) elements in the support of \( x \) and there are \( (q-2)^i \) possible choices of the symbols at those \( i \) positions that are distinct from the coordinates of \( x \).

There are \( \binom{j}{j} \) possible subsets of \( j \) elements in the support of \( x \) that are zero at those positions. Therefore

\[
N_q(n, v, w, s) = \sum_{i+j+k=s, v+i-j=w} \binom{n-v}{k} \binom{v-i}{j} \binom{(q-2)q-1}{k}.
\]

**Remark 4.1.11** Let us consider special values of \( N_q(n, v, w, s) \). If \( s = 0 \), then \( N_q(n, v, w, 0) = 1 \) if \( v = w \) and \( N_q(n, v, w, 0) = 1 \) otherwise. If \( s = 1 \), then

\[
N_q(v, w, 1) = \begin{cases} 
(n - w + 1)(q - 1) & \text{if } v = w - 1, \\
w(q - 2) & \text{if } v = w, \\
w + 1 & \text{if } v = w + 1, \\
0 & \text{otherwise.}
\end{cases}
\]

**Proposition 4.1.12** Let \( C \) be a perfect code of length \( n \) and covering radius \( \rho \) and weight distribution \( (A_0, A_1, \ldots, A_n) \). Then

\[
\binom{n}{w}(q - 1)^w = \sum_{v=w-\rho}^{w+\rho} A_v \sum_{s=|v-w|}^{\rho} N_q(n, v, w, s) \text{ for all } w.
\]
4.1. WEIGHT ENUMERATOR

Proof. Define the set

\[ \mathcal{N}(w, \rho) = \{ (y, c) \mid y \in \mathbb{F}_q^n, \ wt(y) = w, \ c \in C, \ d(y, c) \leq \rho \}. \]

(1) For every \( y \) in \( \mathbb{F}_q^n \) of weight \( w \) there is a unique codeword \( c \) in \( C \) that has distance at most \( \rho \) to \( y \), since \( C \) is perfect with covering radius \( \rho \). Hence

\[ |\mathcal{N}(w, \rho)| = \binom{n}{w} (q - 1)^w. \]

(2) On the other hand consider the fibre of the projection on the second factor:

\[ \mathcal{N}(c, w, \rho) = \{ y \in \mathbb{F}_q^n \mid wt(y) = w, \ d(y, c) \leq \rho \}. \]

for a given codeword \( c \) in \( C \). If \( c \) has weight \( v \), then

\[ |\mathcal{N}(c, w, \rho)| = \sum_{s=0}^{\rho} N_q(n, v, w, s). \]

Hence

\[ |\mathcal{N}(w, \rho)| = \sum_{v=0}^{n} A_v \sum_{s=0}^{\rho} N_q(n, v, w, s) \]

Notice that \( |wt(x) - wt(y)| \leq d(x, y) \). Hence \( N_q(n, v, w, s) = 0 \) if \( |v - w| > s \). Combining (1) and (2) gives the desired result.

\[ \diamond \]

Example 4.1.13 The ternary Golay code has parameters [11, 6, 5] and is perfect with covering radius 2 by Proposition 3.2.28. We leave it as an exercise to show by means of the recursive relations of Proposition 4.1.12 that the weight enumerator of this code is given by

\[ 1 + 132Z^5 + 132Z^6 + 330Z^8 + 110Z^9 + 24Z^{11}. \]

Example 4.1.14 The binary Golay code has parameters [23, 12, 7] and is perfect with covering radius 3 by Proposition 3.2.28. We leave it as an exercise to show by means of the recursive relations of Proposition 4.1.12 that the weight enumerator of this code is given by

\[ 1 + 253Z^7 + 506Z^8 + 1288Z^{11} + 1288Z^{12} + 506Z^{15} + 203Z^{16} + Z^{23}. \]

4.1.2 Average weight enumerator

Remark 4.1.15 The computation of the weight enumerator of a given code is most of the time hard. For the perfect codes such as the Hamming codes and the binary and ternary Golay codes this is left as exercises to the reader and can be done by using Proposition 4.1.12. In Proposition 4.4.22 the weight distribution of MDS codes is treated. The weight enumerator of only a few infinite families of codes is known. On the other hand the average weight enumerator of a class of codes is very often easy to determine.
Definition 4.1.16 Let $C$ be a nonempty class of codes over $\mathbb{F}_q$ of the same length. The 
**average weight enumerator** of $C$ is defined as the average of all $W_C$ with $C \in C$:

$$W_C(Z) = \frac{1}{|C|} \sum_{C \in C} W_C(Z),$$

and similarly for the homogeneous average weight enumerator $W_C(X,Y)$ of this class.

Definition 4.1.17 A class $C$ of $[n,k]$ codes over $\mathbb{F}_q$ is called **balanced** if there is a number $N(C)$ such that

$$N(C) = |\{ C \in C \mid y \in C \}|$$

for every nonzero word $y$ in $\mathbb{F}_n^q$.

Example 4.1.18 The prime example of a class of balanced codes is the set $C[n,k]$ of all $[n,k]$ codes over $\mathbb{F}_q$. Other examples are:

Lemma 4.1.19 Let $C$ be a balanced class of $[n,k]$ codes over $\mathbb{F}_q$. Then

$$N(C) = |C| \frac{q^k - 1}{q^n - 1}.$$  

Proof. Compute the number of elements of the set of pairs

$$\{ (y, C) \mid y \neq 0, y \in C \in C \}$$

in two ways. In the first place by keeping a nonzero $y$ in $\mathbb{F}_n^q$ fixed, and letting $C$ vary in $C$ such that $y \in C$. This gives the number $(q^n - 1)N(C)$, since $C$ is balanced. Secondly by keeping $C$ in $C$ fixed, and letting the nonzero $y$ in $C$ vary. This gives the number $|C|(q^k - 1)$. This gives the desired result, since both numbers are the same.

Proposition 4.1.20 Let $f$ be a function on $\mathbb{F}_n^q$ with values in a complex vector space. Let $C$ be a balanced class of $[n,k]$ codes over $\mathbb{F}_q$. Then

$$\frac{1}{|C|} \sum_{C \in C} \sum_{c \in C^*} f(c) = \frac{q^k - 1}{q^n - 1} \sum_{v \in (\mathbb{F}_q^n)^*} f(v),$$

where $C^*$ denotes the set of all nonzero elements of $C$.

Proof. By interchanging the order of summation we get

$$\sum_{C \in C} \sum_{v \in C^*} f(v) = \sum_{v \in (\mathbb{F}_q^n)^*} f(v) \sum_{v \in C \in C} 1.$$  

The last summand is constant and equals $N(C)$, by assumption. Now the result follows by the computation of $N(C)$ in Lemma 4.1.20.

Corollary 4.1.21 Let $C$ be a balanced class of $[n,k]$ codes over $\mathbb{F}_q$. Then

$$W_C(Z) = 1 + \frac{q^k - 1}{q^n - 1} \sum_{w=1}^{n} \binom{n}{w} (q - 1)^w Z^w.$$  

Proof. Apply Proposition 4.1.20 to the function $f(v) = Z^{\text{wt}(v)}$, and use (4.1) of Remark 4.1.3.

***GV bound for a collection of balanced codes, Loeliger***
4.1. WEIGHT ENUMERATOR

4.1.3 MacWilliams identity

Although there is no apparent relation between the minimum distances of a code and its dual, the weight enumerators satisfy the MacWilliams identity.

Theorem 4.1.22 Let $C$ be an $[n, k]$ code over $\mathbb{F}_q$. Then
\[ W_{C^\perp}(X, Y) = q^{-k}W_C(X + (q - 1)Y, X - Y). \]

The following simple result is useful to the proof of the MacWilliams identity.

Lemma 4.1.23 Let $C$ be an $[n, k]$ linear code over $\mathbb{F}_q$. Let $v$ be an element of $\mathbb{F}_n^q$, but not in $C^\perp$. Then, for every $\alpha \in \mathbb{F}_q$, there exist exactly $q^k - 1$ codewords $c$ such that $c \cdot v = \alpha$.

Proof. Consider the map $\varphi : C \to \mathbb{F}_q$ defined by $\varphi(c) = c \cdot v$. This is a linear map. The map is not constant zero, since $v$ is not in $C^\perp$. Hence every fibre $\varphi^{-1}(\alpha)$ consists of the same number of elements $q^k - 1$ for all $\alpha \in \mathbb{F}_q$. ⋄

To prove Theorem 4.1.22, we introduce the characters of Abelian groups and prove some lemmas.

Definition 4.1.24 Let $(G, +)$ be an abelian group with respect to the addition $+$. Let $(\mathbb{S}, \cdot)$ be the multiplicative group of the complex numbers of modulus one. A character $\chi$ of $G$ is a homomorphism from $G$ to $\mathbb{S}$. So, $\chi$ is a mapping satisfying
\[ \chi(g_1 + g_2) = \chi(g_1) \cdot \chi(g_2), \quad \text{for all } g_1, g_2 \in G. \]

If $\chi(g) = 1$ for all elements $g \in G$, we call $\chi$ the principal character.

Remark 4.1.25 For any character $\chi$ we have $\chi(0) = 1$, since $\chi(0)$ is not zero and $\chi(0) = \chi(0 + 0) = \chi(0)^2$.

If $G$ is a finite group of order $N$ and $\chi$ is a character of $G$, then $\chi(g)$ is an $N$-th root of unity for all $g \in G$, since
\[ 1 = \chi(0) = \chi(Ng) = \chi(g)^N. \]

Lemma 4.1.26 Let $\chi$ be a character of a finite group $G$. Then
\[ \sum_{g \in G} \chi(g) = \begin{cases} |G| & \text{when } \chi \text{ is a principal character}, \\ 0 & \text{otherwise}. \end{cases} \]

Proof. The result is trivial when $\chi$ is principal. Now suppose $\chi$ is not principal. Let $h \in G$ such that $\chi(h) \neq 1$. We have
\[ \chi(h) \sum_{g \in G} \chi(g) = \sum_{g \in G} \chi(h + g) = \sum_{g \in G} \chi(g), \]

since the map $g \mapsto h + g$ is a permutation of $G$. Hence, $(\chi(h) - 1) \sum_{g \in G} \chi(g) = 0$, which implies $\sum_{g \in G} \chi(g) = 0$. ⋄
Definition 4.1.27 Let $V$ be a complex vector space. Let $f : \mathbb{F}_{q^n} \to V$ be a mapping on $\mathbb{F}_{q^n}$ with values in $V$. Let $\chi$ be a character of $\mathbb{F}_{q}$. The Hadamard transform $\hat{f}$ of $f$ is defined as

$$\hat{f}(u) = \sum_{v \in \mathbb{F}_{q^n}} \chi(u \cdot v) f(v).$$

Lemma 4.1.28 Let $f : \mathbb{F}_{q^n} \to V$ be a mapping on $\mathbb{F}_{q^n}$ with values in the complex vector space $V$. Let $\chi$ be a non-principal character of $\mathbb{F}_{q}$. Then,

$$\sum_{c \in C} \hat{f}(c) = |C| \sum_{v \in C^\perp} f(v).$$

Proof. By definition, we have

$$\sum_{c \in C} \hat{f}(c) =$$

$$\sum_{c \in C} \sum_{v \in \mathbb{F}_{q^n}} \chi(c \cdot v) f(v) =$$

$$\sum_{v \in \mathbb{F}_{q^n}} f(v) \sum_{c \in C} \chi(c \cdot v) =$$

$$\sum_{v \in C^\perp} f(v) \sum_{c \in C} \chi(c \cdot v) + \sum_{v \in \mathbb{F}_{q^n} \setminus C^\perp} f(v) \sum_{c \in C} \chi(c \cdot v) =$$

$$|C| \sum_{v \in C^\perp} f(v) + \sum_{v \in \mathbb{F}_{q^n} \setminus C^\perp} f(v) \sum_{c \in C} \chi(c \cdot v).$$

The result follows, since

$$\sum_{c \in C} \chi(c \cdot v) = q^{k-1} \sum_{\alpha \in \mathbb{F}_{q}} \chi(\alpha) = 0$$

for any $v \in \mathbb{F}_{q^n} \setminus C^\perp$ and $\chi$ not principal, by Lemmas 4.1.23 and 4.1.26.

Proof of Theorem 4.1.22. Let $\chi$ be a non-principal character of $\mathbb{F}_{q}$. Consider the following mapping

$$f(v) = X^{n - \text{wt}(v)} Y^{\text{wt}(v)}$$

from $\mathbb{F}_{q^n}$ to the vector space of polynomials in the variables $X$ and $Y$ with complex coefficients. Then

$$\sum_{v \in C^\perp} f(v) = \sum_{v \in C^\perp} X^{n - \text{wt}(v)} Y^{\text{wt}(v)} = W_{C^\perp}(X,Y),$$

by applying (4.2) of Remark 4.1.3 to $C^\perp$. Let $c = (c_1, \ldots, c_n)$ and $v = (v_1, \ldots, v_n)$. Define $\text{wt}(0) = 0$ and $\text{wt}(\alpha) = 1$ for all nonzero $\alpha \in \mathbb{F}_{q}$. Then $\text{wt}(v) = \text{wt}(v_1) + \cdots + \text{wt}(v_n)$. The Hadamard transform $\hat{f}(c)$ is equal to

$$\sum_{v \in \mathbb{F}_{q^n}} \chi(c \cdot v) X^{n - \text{wt}(v)} Y^{\text{wt}(v)} =$$
4.1. WEIGHT ENUMERATOR

\[
\sum_{\mathbf{v} \in \mathbb{F}_q^n} X^{n - \text{wt}(\mathbf{v})} \cdots \text{wt}(\mathbf{v}_n) Y^{\text{wt}(\mathbf{v}_1) + \cdots + \text{wt}(\mathbf{v}_n)} X (c_1 v_1 + \cdots + c_n v_n) = \\
X^n \sum_{\mathbf{v} \in \mathbb{F}_q^n} \prod_{i=1}^n \left( \frac{Y}{X} \right)^{\text{wt}(\mathbf{v})} \chi(c_i v_i) = \\
X^n \prod_{i=1}^n \sum_{\mathbf{v} \in \mathbb{F}_q^n} \left( \frac{Y}{X} \right)^{\text{wt}(\mathbf{v})} \chi(c_i v_i).
\]

If \( c_i \neq 0 \), then

\[
\sum_{\mathbf{v} \in \mathbb{F}_q^n} \left( \frac{Y}{X} \right)^{\text{wt}(\mathbf{v})} \chi(c_i v_i) = 1 + \frac{Y}{X} \sum_{\alpha \in \mathbb{F}_q^*} \chi(\alpha) = 1 - \frac{Y}{X},
\]

by Lemma 4.1.26. Hence

\[
\sum_{\mathbf{v} \in \mathbb{F}_q^n} \left( \frac{Y}{X} \right)^{\text{wt}(\mathbf{v})} \chi(c_i v_i) = \begin{cases} 
1 + (q-1) \frac{Y}{X} & \text{if } c_i = 0, \\
1 - \frac{Y}{X} & \text{if } c_i \neq 0.
\end{cases}
\]

Therefore \( \hat{f}(\mathbf{c}) \) is equal to

\[
X^n \left( 1 - \frac{Y}{X} \right)^{\text{wt}(\mathbf{c})} \left( 1 + (q-1) \frac{Y}{X} \right)^{n - \text{wt}(\mathbf{c})} = \\
(X - Y)^{\text{wt}(\mathbf{c})} (X + (q-1)Y)^{n - \text{wt}(\mathbf{c})}.
\]

Hence

\[
\sum_{\mathbf{c} \in C} \hat{f}(\mathbf{c}) = \sum_{\mathbf{c} \in C} U^{n - \text{wt}(\mathbf{c})} V^{\text{wt}(\mathbf{c})} = W_C(U, V),
\]

by (4.2) of Remark 4.1.3 with the substitution \( U = X + (q-1)Y \) and \( V = X - Y \). It is shown that on the one hand

\[
\sum_{\mathbf{v} \in C^\perp} f(\mathbf{v}) = W_{C^\perp}(X, Y),
\]

and on the other hand

\[
\sum_{\mathbf{c} \in C} \hat{f}(\mathbf{c}) = W_C(X + (q-1)Y, X - Y),
\]

The results follows by Lemma 4.1.28 on the Hadmard transform.

\[\Box\]

Example 4.1.29 The zero code \( C \) has homogeneous weight enumerator \( X^n \) and its dual \( \mathbb{F}_q^n \) has homogeneous weight enumerator \( (X + (q-1)Y)^n \), by Example 4.1.4, which is indeed equal to \( q^n W_C(X + (q-1)Y, X - Y) \) and confirms MacWilliams identity.

Example 4.1.30 The \( n \)-fold repetition code \( C \) has homogeneous weight enumerator \( X^n + (q - 1)Y^n \) and the homogeneous weight enumerator of its dual code in the binary case is \( \frac{1}{2} ((X + Y)^n + (X - Y)^n) \), by Example 4.1.5, which is
equal to $2^{-1}W_C(X + Y, X - Y)$, confirming the MacWilliams identity for $q = 2$.

For arbitrary $q$ we have

$$W_{C^\perp}(X, Y) = q^{-1}W_C(X + (q - 1)Y, X - Y) = q^{-1}((X + (q - 1)Y)^n + (q - 1)(X - Y)^n) = \sum_{w=0}^{n} \binom{n}{w} \frac{(q - 1)^w + (q - 1)(-1)^w}{q} X^{n-w} Y^w.$$

Example 4.1.31 ***dual of a balanced class of codes, $C^\perp$ balanced?***

Definition 4.1.32 An $[n,k]$ code $C$ over $\mathbb{F}_q$ is called formally self-dual if $C$ and $C^\perp$ have the same weight enumerator.

Remark 4.1.33 ***A quasi self-dual code is formally self-dual, existence of an asymp. good family of codes***

4.1.4 Exercises

4.1.1 Compute the weight spectrum of the dual of the $q$-ary $n$-fold repetition code directly, that is without using MacWilliams identity. Compare this result with Example 4.1.30.


4.1.3 Compute the weight enumerator of the Hamming code $H_r(q)$ by solving the given differential equation as given in Example 4.1.9.

4.1.4 Compute the weight enumerator of the ternary Golay code as given in Example 4.1.13.

4.1.5 Compute the weight enumerator of the binary Golay code as given in Example 4.1.14.

4.1.6 Consider the quasi self-dual code with generator matrix $(I_k | I_k)$ of Exercise 2.5.8. Show that its weight enumerator is equal $(X^2 + (q - 1)Y^2)^k$. Verify that this code is formally self-dual.

4.1.7 Let $C$ be the code over $\mathbb{F}_q$, with $q$ even, with generator matrix $H$ of Example 2.2.9. For which $q$ does this code contain a word of weight 7?

4.2 Error probability

*** Some introductory results on the error probability of correct decoding up to half the minimum distance were given in Section ??.” ***
4.2. ERROR PROBABILITY

4.2.1 Error probability of undetected error

***

**Definition 4.2.1** Consider the $q$-ary symmetric channel where the receiver checks whether the received word $r$ is a codeword or not, for instance by computing whether $Hr^T$ is zero or not for a chosen parity check matrix $H$, and asks for retransmission in case $r$ is not a codeword. See Remark 2.3.2. Now it may occur that $r$ is again a codeword but not equal to the codeword that was sent. This is called an undetected error.

**Proposition 4.2.2** Let $W_C(X,Y)$ be the weight enumerator of the code $C$. Then the probability of undetected error on a $q$-ary symmetric channel with cross-over probability $p$ is given by

$$P_{ue}(p) = W_C \left( 1 - p, \frac{p}{q-1} \right) - (1-p)^n.$$  

**Proof.** Every codeword has the same probability of transmission and the code is linear. So without loss of generality we may assume that the zero word is sent. Hence

$$P_{ue}(p) = \frac{1}{|C|} \sum_{x \in C} \sum_{y \neq C} P(y|x) = \sum_{0 \neq y \in C} P(y|0).$$

If the received codeword $y$ has weight $w$, then $w$ symbols are changed and the remaining $n-w$ symbols remained the same. So $P(y|0) = (1-p)^{n-w} \left( \frac{p}{q-1} \right)^w$ by Remark 2.4.15. Hence

$$P_{ue}(p) = \sum_{w=1}^{n} A_w (1-p)^{n-w} \left( \frac{p}{q-1} \right)^w.$$  

Substituting $X = 1-p$ and $Y = p/(q-1)$ in $W_C(X,Y)$ gives the desired result, since $A_0 = 1$.  

**Remark 4.2.3** Now $P_{retr}(p) = 1 - P_{ue}(p)$ is the probability of retransmission.

**Example 4.2.4** Let $C$ be the binary triple repetition code. Then $P_{ue}(p) = p^3$, since $W_C(X,Y) = X^3 + Y^3$ by Example 4.1.5.

**Example 4.2.5** Let $C$ be the $[7, 4, 3]$ Hamming code. Then

$$P_{ue}(p) = 7p^3 - 21p^4 + 21p^5 - 7p^6 + p^7$$

by Example 4.1.6.

4.2.2 Probability of decoding error

Remember that in Lemma 4.1.10 a formula was derived for $N_q(n,v,w,s)$, the number of words in $\mathbb{F}_q^n$ of weight $w$ that are at distance $s$ from a given word of weight $v$. 
Proposition 4.2.6 The probability of decoding error of a decoder that corrects up to \( t \) errors with \( 2t + 1 \leq d \) of a code \( C \) of minimum distance \( d \) on a \( q \)-ary symmetric channel with cross-over probability \( p \) is given by

\[
P_{de}(p) = \sum_{w=0}^{n} \left( \frac{p}{q-1} \right)^{w} (1-p)^{n-w} \sum_{s=0}^{t} \sum_{v=1}^{n} A_{v} N_{q}(n, v, w, s).
\]

Proof. This is left as an exercise. \( \diamond \)

Example 4.2.7 ............

4.2.3 Random coding

***ML (maximum likelihood) decoding = MD (minimum distance or nearest neighbor) decoding for the BSC.***

Proposition 4.2.8 ***...***

\[
P_{err}(p) = W_{C}(\gamma) - 1, \text{ where } \gamma = 2\sqrt{p(1-p)}.
\]

Proof. .... \( \diamond \)

Theorem 4.2.9 ***Shannon’s theorem for random codes***

Proof. ***...*** \( \diamond \)

4.2.4 Exercises

4.2.1 ***Give the probability of undetected error for the code ....***

4.2.2 Give a proof of Proposition 4.2.6.

4.2.3 ***Give the probability of decoding error and decoding failure for the code .... for a decoder correcting up to ... errors.***

4.3 Finite geometry and codes

***Intro***

4.3.1 Projective space and projective systems

The notion of a linear code has a geometric equivalent in the concept of a projective system which is a set of points in projective space.

Remark 4.3.1 The affine line \( A \) over a field \( F \) is nothing else than the field \( F \). The projective line \( P \) is an extension of the affine line by one point at infinity.

\[
\ldots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cup \{\infty\}
\]
4.3. FINITE GEOMETRY AND CODES

The elements are fractions \((x_0 : x_1)\) with \(x_0, x_1\) elements of a field \(\mathbb{F}\) not both zero, and the fraction \((x_0 : x_1)\) is equal to \((y_0 : y_1)\) if and only if \((x_0, x_1) = \lambda(y_0 : y_1)\) for some \(\lambda \in \mathbb{F}^\ast\). The point \((x_0 : x_1)\) with \(x_0 \neq 0\) is equal to \((1 : x_1/x_0)\) and corresponds to the point \(x_1/x_0 \in \mathbb{A}\). The point \((x_0 : x_1)\) with \(x_0 = 0\) is equal to \((0 : 1)\) and is the unique point at infinity. The notation \(\mathbb{P}(\mathbb{F})\) and \(\mathbb{A}(\mathbb{F})\) is used to emphasize that the elements are in the field \(\mathbb{F}\).

The affine plane \(\mathbb{A}^2\) over a field \(\mathbb{F}\) consists of points and lines. The points are in \(\mathbb{F}^2\) and the lines are the subsets of the form \(\{ a + \lambda v \mid \lambda \in \mathbb{F} \}\) with \(v \neq 0\), in a parametric explicit description. A line is alternatively given by an implicit description by means of an equation \(aX + bY + c = 0\), with \(a, b, c \in \mathbb{F}\) not all zero. Every two distinct points are contained in exactly one line. Two lines are either parallel, that is they coincide or do not intersect, or they intersect in exactly one point. If \(\mathbb{F}\) is equal to the finite field \(\mathbb{F}_q\), then there are \(q^2\) points and \(q^2 + q\) lines, and every line consists of \(q\) points, and the number of lines though a given point is \(q + 1\).

Being parallel defines an equivalence relation on the set of lines in the affine plane, and every equivalence or parallel class of a line \(l\) defines a unique point at infinity \(P_l\). So \(P_l = P_m\) if and only if \(l\) and \(m\) are parallel. In this way the affine plane is extended to the projective plane \(\mathbb{P}^2\) by adding the points at infinity \(P_l\). A line in the projective plane is a line \(l\) in the affine plane extended with its point at infinity \(P_l\) or the line at infinity, consisting of all the points at infinity. Every two distinct points in \(\mathbb{P}^2\) are contained in exactly one line, and two distinct lines intersect in exactly one point. If \(\mathbb{F}\) is equal to the finite field \(\mathbb{F}_q\), then there are \(q^2 + q + 1\) points and the same number of lines, and every line consists of \(q + 1\) points, and the number of lines though a given point is \(q + 1\).

***picture***

Another model of the projective plane can be obtained as follows. Consider the points of the affine plane as the plane in three space \(\mathbb{F}^3\) with coordinates \((x, y, z)\) given by the equation \(Z = 1\). Every point \((x, y, 1)\) in the affine plane corresponds with a unique line in \(\mathbb{F}^3\) through the origin parameterized by \(\lambda(x, y, 1), \lambda \in \mathbb{F}\). Conversely, a line in \(\mathbb{F}^3\) through the origin parameterized by \(\lambda(x, y, z), \lambda \in \mathbb{F}\), intersects the affine plane in the unique point \((x/z, y/z, 1)\) if \(z \neq 0\), and corresponds to the unique parallel class \(P_l\) of the line \(l\) in the affine plane with equation \(xY = yX\) if \(z = 0\). Furthermore every line in the affine plane corresponds through the origin in \(\mathbb{F}^3\), and conversely every line through the origin in \(\mathbb{F}^3\) with equation \(aX + bY + cZ = 0\) intersects the affine plane in the unique line with equation \(aX + bY + c = 0\) if not both \(a = 0\) and \(b = 0\), or corresponds to the line at infinity if \(a = b = 0\).

***picture***

An \(\mathbb{F}\)-rational point of the projective plane is a line through the origin in \(\mathbb{F}^3\). Such a point is determined by a three-tuple \((x, y, z)\) in \(\mathbb{F}^3\), not all of them being zero. A scalar multiple determines the same point in the projective plane. This defines an equivalence relation \(\equiv\) by \((x, y, z) \equiv (x', y', z')\) if and only if there exists a nonzero \(\lambda \in \mathbb{F}\) such that \((x, y, z) = \lambda(x', y', z')\). The equivalence class with representative \((x, y, z)\) is denoted by \((x : y : z)\), and \(x, y\) and \(z\) are called homogeneous coordinates of the point. The set of all projective points \((x : y : z)\),
with $x, y, z \in F$ not all zero, is called the projective plane over $F$. The set of $F$-rational projective points is denoted by $P^2(F)$. A line in the projective plane that is defined over $F$ is a plane through the origin in $F^3$. Such a line has a homogeneous equation $aX + bY + cZ = 0$ with $a, b, c \in F$ not all zero.

The affine plane is embedded in the projective plane by the map $(x, y) \mapsto (x : y : 1)$. The image is the subset of all projective points $(x : y : z)$ such that $z \neq 0$.

The line at infinity is the line with equation $Z = 0$. A point at infinity of the affine plane is a point on the line at infinity in the projective plane. Every line in the affine plane intersects the line at infinity in a unique point and all lines in the affine plane which are parallel, that is to say which do not intersect in the affine plane, intersect in the same point at infinity. The above embedding of the affine plane in the projective plane is standard, but the mappings $(x, z) \mapsto (x : 1 : z)$ and $(y, z) \mapsto (1 : y : z)$ give two alternative embeddings of the affine plane. The images are the complement of the line $Y = 0$ and $X = 0$, respectively. Thus the projective plane is covered with three copies of the affine plane.

**Definition 4.3.2** An affine subspace of $F^r$ of dimension $s$ is a subset of the form

$$\{ a + \lambda_1 v_1 + \cdots + \lambda_s v_s \mid \lambda_i \in F, i = 1, \ldots, s \},$$

where $a \in F^r$, and $v_1, \ldots, v_s$ is a linearly independent set of vectors in $F^r$, and $r - s$ is called the codimension of the subspace. The affine space of dimension $r$ over a field $F$, denoted by $A^r(F)$, consists of all affine subsets of $F^r$. The elements of $F^r$ are called points of the affine space. Lines and planes are the linear subspaces of dimension one and two, respectively. A hyperplane is an affine subspace of codimension 1.

**Definition 4.3.3** A point of the projective space over a field $F$ of dimension $r$ is a line through the origin in $F^{r+1}$. A line in $P^r(F)$ is a plane through the origin in $F^{r+1}$. More generally a projective subspace of dimension $s$ in $P^r(F)$ is a linear subspace of dimension $s + 1$ of the vector space $F^{r+1}$, and $r - s$ is called the codimension of the subspace. The projective space of dimension $r$ over a field $F$, denoted by $P^r(F)$, consists of all its projective subspaces. A point of a projective space is incident with or an element of a projective subspace if the line corresponding to the point is contained in the linear subspace that corresponds with the projective subspace. A hyperplane in $P^r(F)$ is a projective subspace of codimension 1.

**Definition 4.3.4** A point in $P^r(F)$ is denoted by its homogeneous coordinates $(x_0 : x_1 : \cdots : x_r)$ with $x_0, x_1, \ldots, x_r \in F$ and not all zero, where $\lambda(x_0, x_1, \ldots, x_r)$, $\lambda \in F$, is a parametrization of the corresponding line in $P^{r+1}$. Let $(x_0, x_1, \ldots, x_r)$ and $(y_0, y_1, \ldots, y_r)$ be two nonzero vectors in $F^{r+1}$. Then $(x_0 : x_1 : \cdots : x_r)$ and $(y_0 : y_1 : \cdots : y_r)$ represent the same point in $P^r(F)$ if and only if $(x_0, x_1, \ldots, x_r) = \lambda(y_0, y_1, \ldots, y_r)$ for some $\lambda \in F^*$. The standard homogeneous coordinates of a point in $P^r(F)$ are given by $(x_0 : x_1 : \cdots : x_r)$ such that there exists a $j$ with $x_j = 1$ and $x_i = 0$ for all $i < j$.

The standard embedding of $A^r(F)$ in $P^r(F)$ is given by

$$(x_1, \ldots, x_r) \mapsto (1 : x_1 : \cdots : x_r).$$

**Remark 4.3.5** Every hyperplane in $P^r(F)$ is defined by an equation

$$a_0X_0 + a_1X_1 + \cdots + a_rX_r = 0,$$
where $a_0, a_1, \ldots, a_r$ are $r$ elements of $\mathbb{F}$, not all zero. Furthermore

$$a_0'X_0 + a_1'X_1 + \cdots + a_r'X_r = 0,$$

defines the same hyperplane if and only if there exists a nonzero $\lambda$ in $\mathbb{F}$ such that $a_i' = \lambda a_i$ for all $i = 0, 1, \ldots, r$. Hence there is a duality between points and hyperplanes in $\mathbb{P}^r(\mathbb{F})$, where a $(a_0 : a_1 : \ldots : a_r)$ is send to the hyperplane with equation $a_0X_0 + a_1X_1 + \cdots + a_rX_r = 0$.

**Example 4.3.6** The columns of a generator matrix of a simplex code $S_r(q)$ represent all the points of $\mathbb{P}^{r-1}(\mathbb{F}_q)$.

**Proposition 4.3.7** Let $r$ and $s$ be non-negative integers such that $s \leq r$. The number of $s$ dimensional projective subspaces of $\mathbb{P}^r(\mathbb{F}_q)$ is equal to the Gaussian binomial

$$\begin{bmatrix} r + 1 \\ s + 1 \end{bmatrix}_q = \frac{(q^{r+1} - 1)(q^{r+1} - q) \cdots (q^{r+1} - q^s)}{(q^{r+1} - 1)(q^{r+1} - q) \cdots (q^{r+1} - q^s)}.$$

In particular, the number of points of $\mathbb{P}^r(\mathbb{F}_q)$ is equal to

$$\begin{bmatrix} r + 1 \\ 1 \end{bmatrix}_q = \frac{q^{r+1} - 1}{q - 1} = q^r + q^{r-1} + \cdots + q + 1.$$

**Proof.** An $s$ dimensional projective subspace of $\mathbb{P}^r(\mathbb{F}_q)$ is an $s + 1$ dimensional subspace of $\mathbb{F}_q^{r+1}$, which is an $[r+1, s+1]$ code over $\mathbb{F}_q$. The number of the latter objects is equal to the stated Gaussian binomial, by Proposition 2.5.2. ∘

**Definition 4.3.8** Let $\mathcal{P} = (P_1, \ldots, P_n)$ be an $n$-tuple of points in $\mathbb{P}^r(\mathbb{F}_q)$. Then $\mathcal{P}$ is called a projective system in $\mathbb{P}^r(\mathbb{F}_q)$ if not all these points lie in a hyperplane. This system is called simple if the $n$ points are mutually distinct.

**Definition 4.3.9** A code $C$ is called degenerate if there is a coordinate $i$ such that $c_i = 0$ for all $c \in C$.

**Remark 4.3.10** A code $C$ is nondegenerate if and only if there is no zero column in a generator matrix of the code if and only if $d(C^\perp) \geq 2$.

**Example 4.3.11** Let $G$ be a generator matrix of a nondegenerate code $C$ of dimension $k$. So $G$ has no zero columns. Take the columns of $G$ as homogeneous coordinates of points in $\mathbb{P}^{k-1}(\mathbb{F}_q)$. This gives the projective system $\mathcal{P}_G$ of $G$. Conversely, let $(P_1, \ldots, P_n)$ be an enumeration of the points of a projective system $\mathcal{P}$ in $\mathbb{P}^r(\mathbb{F}_q)$. Let $(p_{0j} : p_{1j} : \cdots : p_{rj})$ be homogeneous coordinates of $P_j$. Let $G_{\mathcal{P}}$ be the $(r+1) \times n$ matrix with $(p_{0j}, p_{1j}, \ldots, p_{rj})^T$ as $j$-th column. Then $G_{\mathcal{P}}$ is the generator matrix of a nondegenerate code of length $n$ and dimension $r + 1$, since not all points lie in a hyperplane.

**Proposition 4.3.12** Let $C$ be a nondegenerate code of length $n$ with generator matrix $G$. Let $\mathcal{P}_G$ be the projective system of $G$. The code has generalized Hamming weight $d_r$ if and only if $n - d_r$ is the maximal number of points of $\mathcal{P}_G$ in a linear subspace of codimension $r$. 
**Proof.** Let \( G = (g_{ij}) \) and \( P_j = (g_{1j} : \ldots : g_{kj}) \). Then \( P = (P_1, \ldots, P_n) \). Let \( D \) be a subspace of \( C \) of dimension \( r \) of minimal weight \( d_r \). Let \( c_1, \ldots, c_r \) be a basis of \( D \). Then \( c_i = (c_{i1}, \ldots, c_{in}) = h_i G \) for some \( h_i \). Let \( h_i = (h_{i1}, \ldots, h_{ik}) \in \mathbb{F}_q^k \).

Let \( H_i \) be the hyperplane in \( \mathbb{P}^{k-1}(\mathbb{F}_q) \) with equation \( h_{i1}X_1 + \ldots + h_{ik}X_k = 0 \). Then \( c_{ij} = 0 \) if and only if \( P_j \in H_i \) for all \( 1 \leq i \leq r \) and \( 1 \leq j \leq n \). Let \( H \) be the intersection of \( H_1, \ldots, H_r \). Then \( H \) is a linear subspace of codimension \( r \), since the \( c_1, \ldots, c_r \) are linearly independent. Furthermore \( P_j \in H \) if and only if \( c_{ij} = 0 \) for all \( 1 \leq i \leq r \) and only if \( j \not\in \text{supp}(D) \). Hence \( n - d_r \) points lie in a linear subspace of codimension \( r \).

The proof of the converse is left to the reader. \( \diamond \)

**Definition 4.3.13** A code \( C \) is called **projective** if \( d(C^\perp) \geq 3 \).

**Remark 4.3.14** A code of length \( n \) is projective if and only if \( G \) has no zero column and a column is not a scalar multiple of another column of \( G \) if and only if the projective system \( \mathcal{P}_G \) is simple for every generator matrix \( G \) of the code.

**Definition 4.3.15** A map \( \varphi : \mathbb{P}^r(\mathbb{F}) \to \mathbb{P}^r(\mathbb{F}) \) is called a **projective transformation** if \( \varphi \) is given by \( \varphi(x_0 : x_1 : \cdots : x_r) = (y_0 : y_1 : \cdots : y_r) \), where \( y_i = \sum_{j=0}^r a_{ij}x_j \) for all \( i = 0, \ldots, r \), for a given invertible matrix \((a_{ij})\) of size \( r + 1 \) with entries in \( \mathbb{F}_q \).

**Remark 4.3.16** The map \( \varphi \) is well defined by \( \varphi(x) = y \) with \( y_i = \sum_{j=0}^r a_{ij}x_j \). Since the equations for the \( y_i \) are homogeneous in the \( x_j \), the diagonal matrices \( \lambda I_{r+1} \) induce the identity map on \( \mathbb{P}^r(\mathbb{F}) \) for all \( \lambda \in \mathbb{F}_q^* \).

**Definition 4.3.17** Let \( P = (P_1, \ldots, P_n) \) and \( Q = (Q_1, \ldots, Q_n) \) be two projective systems in \( \mathbb{P}^r(\mathbb{F}) \). They are called **equivalent** if there exists a projective transformation \( \varphi \) of \( \mathbb{P}^r(\mathbb{F}) \) and a permutation \( \sigma \) of \( \{1, \ldots, n\} \) such that \( Q = (\varphi(P_{\sigma(1)}), \ldots, \varphi(P_{\sigma(n)}) \).

**Proposition 4.3.18** There is a one-to-one correspondence between generalized equivalence classes of non-degenerate \([n, k, d]\) codes over \( \mathbb{F}_q \) and equivalence classes of projective systems of \( n \) points in \( \mathbb{P}^{k-1}(\mathbb{F}_q) \).

**Proof.** The correspondence between codes and projective systems is given in Example 4.3.11.

Let \( C \) be a nondegenerate code over \( \mathbb{F}_q \) with parameters \([n, k, d] \). Let \( G \) be a generator matrix of \( C \). Take the columns of \( G \) as homogeneous coordinates of points in \( \mathbb{P}^{k-1}(\mathbb{F}_q) \). This gives the projective system \( \mathcal{P}_G \) of \( G \). If \( G' \) is another generator matrix of \( C \), then \( G' = AG \) for some invertible \( k \times k \) matrix \( A \) with entries in \( \mathbb{F}_q \). Furthermore \( A \) induces a projective transformation \( \varphi \) of \( \mathbb{P}^{k-1}(\mathbb{F}_q) \) such that \( \mathcal{P}_G = \varphi(\mathcal{P}_G) \). So \( \mathcal{P}_G \) and \( \mathcal{P}_G' \) are equivalent.

Conversely, let \( P = (P_1, \ldots, P_n) \) be a projective system in \( \mathbb{P}^{k-1}(\mathbb{F}_q) \). This gives the \( k \times n \) generator matrix \( G_P \) of a nondegenerate code. Another enumeration of the points of \( P \) and another choice of the homogeneous coordinates of the \( P_j \) gives a permutation of the columns of \( G_P \) and a nonzero scalar multiple of the columns and therefore a generalized equivalent code. \( \diamond \)

**Proposition 4.3.19** Every \( r \)-tuple of points in \( \mathbb{P}^r(\mathbb{F}_q) \) lie in a hyperplane.
4.3. FINITE GEOMETRY AND CODES

Proof. Let $P_1, \ldots, P_r$ be $r$ points in $\mathbb{P}^r(F_q)$. Let $(p_{0j} : p_{1j} : \cdots : p_{rj})$ be the standard homogeneous coordinates of $P_j$. The $r$ homogeneous equations

$$Y_0 p_{0j} + Y_1 p_{1j} + \cdots + Y_r p_{rj} = 0, \quad j = 1, \ldots, r,$$

in the $r + 1$ variables $Y_0, \ldots, Y_r$ have a nonzero solution $(h_0, \ldots, h_r)$. Let $H$ be the hyperplane with equation $h_0 X_0 + \cdots + h_r X_r = 0$. Then $P_1, \ldots, P_r$ lie in $H$.

4.3.2 MDS codes and points in general position

A second geometric proof of the Singleton bound is given by means of projective systems.

Corollary 4.3.20 (Singleton bound)
The minimum distance $d$ of a code of length $n$ and dimension $k$ is at most $n - k + 1$.

Proof. The zero code has parameters $[n, 0, n + 1]$ by definition, and indeed this code satisfies the Singleton bound. If $C$ is not the zero code, we may assume without loss of generality that the code is not degenerate, by deleting the coordinates where all the codewords are zero. Let $\mathcal{P}$ be the projective system in $\mathbb{P}^{k-1}(F_q)$ of a generator matrix of the code. Then $k - 1$ points of the system lie in a hyperplane by Proposition 4.3.19. Hence $n - d \geq k - 1$, by Proposition 4.3.12.

The notion for projective systems that corresponds to MDS codes is the concept of general position.

Definition 4.3.21 A projective system of $n$ points in $\mathbb{P}^r(F_q)$ is called in general position or an $n$-arc if no $r + 1$ points lie in a hyperplane.

Example 4.3.22 Let $n = q + 1$ and $a_1, a_2, \ldots, a_{q-1}$ be an enumeration of the nonzero elements of $F_q$. Consider the code $C$ with generator matrix

$$G = \begin{pmatrix}
a_1 & a_2 & \cdots & a_{q-1} & 0 & 0 \\
a_1^2 & a_2^2 & \cdots & a_{q-1}^2 & 0 & 1 \\
1 & 1 & \cdots & 1 & 1 & 0
\end{pmatrix}$$

Then $C$ is a $[q + 1, 3, q - 1]$ code by Proposition 3.2.10. Let $P_j = (x_j : x_j^2 : 1)$ for $1 < j \leq q - 1$ and $P_q = (0 : 0 : 1)$, $P_{q+1} = (0 : 1 : 0)$. Let $\mathcal{P} = (P_1, \ldots, P_n)$. Then $\mathcal{P} = \mathcal{P}_C$ and $\mathcal{P}$ is a projective system in the projective plane in general position. Remark that $\mathcal{P}$ is the set all points in the projective plane with coordinates $(x : y : z)$ in $F_q$ that lie on the conic with equation $X^2 = YZ$.

Remark 4.3.23 If $q$ is large enough with respect to $n$, then almost every projective system of $n$ points in $\mathbb{P}^r(F_q)$ is in general position, or equivalently a random code over $F_q$ of length $n$ is MDS. The following proposition and corollary show that every $F_q$-linear code with parameters $[n, k, d]$ is contained in an $F_q^n$-linear MDS code with parameters $[n, n - d + 1, d]$ if $m$ is large enough.
Proposition 4.3.24 Let $B$ be a $q$-ary code. If $q^m > \max\{ \binom{n}{i} | 0 \leq i \leq t \}$ and $d(B^t) > t$, then there exists a sequence $\{B_r | 0 \leq r \leq t\}$ of $q^m$-ary codes such that $B_{r-1} \subseteq B_r$ and $B_r$ is an $[n, r, n-r+1]$ code and contained in the $\mathbb{F}_q$-linear code generated by $B$ for all $0 \leq r \leq t$.

Proof. The minimum distances of $B^t$ and $(B \otimes \mathbb{F}_q)^t$ are the same. Induction on $t$ is used. In case $t = 0$, there is nothing to prove, we can take $B_0 = 0$. Suppose the statement is proved for $t$. Let $B$ be a code such that $d(B^t) > t + 1$ and suppose $q^m > \max\{ \binom{n}{i} | 0 \leq i \leq t + 1 \}$. By induction we may assume that there is a sequence $\{B_r | 0 \leq r \leq t\}$ of $q^m$-ary codes such that $B_{r-1} \subseteq B_r \subseteq B \otimes \mathbb{F}_q$ and $B_r$ is an $[n, r, n-r+1]$ code for all $r$, $0 \leq r \leq t$. So $B \otimes \mathbb{F}_q$ has a generator matrix $G$ with entries $g_{ij}$ for $1 \leq i \leq k$ and $1 \leq j \leq n$, such that the first $r$ rows of $G$ give a generator matrix $G_r$ of $B_r$. In particular the determinants of all $(t \times t)$-sub matrices of $G_t$ are nonzero, by Proposition 3.2.5. Let $\Delta(j_1, \ldots, j_t)$ be the determinant of $G_t(j_1, \ldots, j_t)$, which is the matrix obtained from $G_t$ by taking the columns numbered by $j_1, \ldots, j_t$, where $1 \leq j_1 < \ldots < j_t \leq n$. For $t < i \leq n$ and $1 \leq j_1 < \ldots < j_{t+1} \leq n$ we define $\Delta(i; j_1, \ldots, j_{t+1})$ to be the determinant of the $(t+1) \times (t+1)$ sub matrix of $G$ formed by taking the columns numbered by $j_1, \ldots, j_{t+1}$ and the rows numbered by $1, \ldots, i, t$. Now consider for every $(t+1)$-tuple $J = (j_1, \ldots, j_{t+1})$ such that $1 \leq j_1 < \ldots < j_{t+1} \leq n$, the linear equation in the variables $X_{t+1}, \ldots, X_n$ given by:

$$\sum_{s=1}^{t+1} (-1)^s \Delta(j_1, \ldots, \hat{j}_s, \ldots, j_{t+1}) \left( \sum_{i>t} g_{ij} X_i \right) = 0,$$

where $(j_1, \ldots, j_s, \ldots, j_{t+1})$ is the $t$-tuple obtained from $j$ by deleting the $s$-th element. Rewrite this equation by interchanging the order of summation as follows:

$$\sum_{i>t} \Delta(i; j) X_i = 0.$$

If for a given $j$ the coefficients $\Delta(i; j)$ are zero for all $i > t$, then all the rows of the matrix $G(j)$, which is the sub matrix of $G$ consisting of the columns numbered by $j_1, \ldots, j_{t+1}$, are dependent on the first $t$ rows of $G(j)$. Thus $\text{rank}(G(j)) \leq t$, so $G$ has $t+1$ columns which are dependent. But $G$ is a parity check matrix for $(B \otimes \mathbb{F}_q)^t$, therefore $d((B \otimes \mathbb{F}_q)^t) \leq t + 1$, which contradicts the assumption in the induction hypothesis. We have therefore proved that for a given $(t+1)$-tuple, at least one of the coefficients $\Delta(i; j)$ is nonzero. Therefore the above equation defines a hyperplane $H(j)$ in a vector space over $\mathbb{F}_q$ of dimension $n - t$. We assumed $q^m > \binom{n}{t+1}$, so

$$(q^m)^{n-t} > \binom{n}{t+1} (q^m)^{n-t-1}.$$

Therefore $(\mathbb{F}_q)^{n-t}$ has more elements than the union of all $(\binom{n}{t+1})$ hyperplanes of the form $H(j)$. Thus there exists an element $(x_{t+1}, \ldots, x_n) \in (\mathbb{F}_q)^{n-t}$ which does not lie in this union. Now consider the code $B_{t+1}$, defined by the generator matrix $G_{t+1}$ with entries $(g'_{ij} | 1 \leq l \leq t+1, 1 \leq j \leq n)$, where

$$g'_{ij} = \begin{cases} g_{ij} & \text{if } 1 \leq l \leq t \\ \sum_{i>t} g_{ij} x_i & \text{if } l = t+1 \end{cases}$$
Then \( B_{t+1} \) is a subcode of \( B \otimes F_{q^m} \), and for every \((t+1)\)-tuple \( j \), the determinant of the corresponding \((t+1) \times (t+1)\) sub matrix of \( G_{t+1} \) is equal to \( \sum_{i \geq t} \Delta(i;j)x_i \), which is not zero, since \( x \) is not an element of \( H(j) \). Thus \( B_{t+1} \) is an \([n, t+1, n-t]\) code.

**Corollary 4.3.25** Suppose \( q^m > \max\{\binom{n}{i} | 1 \leq i \leq d-1\} \). Let \( C \) be a \( q \)-ary code of minimum distance \( d \), then \( C \) is contained in a \( q^m \)-ary MDS code of the same minimum distance as \( C \).

**Proof.** The Corollary follows from Proposition 4.3.24 by taking \( B = C^\perp \) and \( t = d-1 \). Indeed, we have \( B_0 \subseteq B_1 \subseteq \cdots \subseteq B_{d-1} \subseteq C \otimes F_{q^m} \), for some \( F_{q^m} \)-linear codes \( B_r, r = 0, \ldots, d-1 \) with parameters \([n, r, n-r+1]\). Applying Exercise 2.3.5 (1) we obtain \( C \otimes F_{q^m} \subseteq B_{d-1}^\perp \), so also \( C \subseteq B_{d-1}^\perp \) holds. Now \( B_{d-1} \) is an \( F_{q^m} \)-linear MDS code, thus \( B_{d-1}^\perp \) also is and has parameters \([n, n-d+1, d]\) by Corollary 3.2.14.

**4.3.3 Exercises**

4.3.1 Give a proof of Remarks 4.3.10 and 4.3.14.

4.3.2 Let \( C \) be the binary \([7,3,4]\) Simplex code. Give a parity check matrix of an \([7,4,4]\) MDS code over \( D \) over \( F_4 \) that contains \( C \) as a subfield subcode.

4.3.3 ....

**4.4 Extended weight enumerator**

***Intro***

**4.4.1 Arrangements of hyperplanes**

***affine/projective arrangements***

The weight spectrum can be computed by counting points in certain configurations of a set of hyperplanes.

**Definition 4.4.1** Let \( \mathbb{F} \) be a field. A *hyperplane* in \( \mathbb{F}^k \) is the set of solutions in \( \mathbb{F}^k \) of a given linear equation

\[
a_1 X_1 + \cdots + a_k X_k = b,
\]

where \( a_1, \ldots, a_k \) and \( b \) are elements of \( \mathbb{F} \) such that not all the \( a_i \) are zero. The hyperplane is called *homogeneous* if the equation is homogeneous, that is \( b = 0 \).

**Remark 4.4.2** The equations \( a_1 X_1 + \cdots + a_k X_k = b \) and \( a'_1 X_1 + \cdots + a'_k X_k = b' \) define the same hyperplane if and only if \( (a'_1, \ldots, a'_k, b') = \lambda(a_1, \ldots, a_k, b) \) for some nonzero \( \lambda \in \mathbb{F} \).
Definition 4.4.3 An \( n \)-tuple \( (H_1, \ldots, H_n) \) of hyperplanes in \( \mathbb{F}^k \) is called an arrangement in \( \mathbb{F}^k \). The arrangement is called simple if all the \( n \) hyperplanes are mutually distinct. The arrangement is called central if all the hyperplanes are linear subspaces. A central arrangement is called essential if the intersection of all its hyperplanes is equal to \( \{0\} \).

Remark 4.4.4 A central arrangement of hyperplanes in \( \mathbb{F}^{r+1} \) gives rise to an arrangement of hyperplanes in \( \mathbb{P}^r(\mathbb{F}) \), since the defining equations are homogeneous. The arrangement is essential if the intersection of all its hyperplanes is empty in \( \mathbb{P}^r(\mathbb{F}) \). The dual notion of an arrangement in projective space is a projective system.

Definition 4.4.5 Let \( G = (g_{ij}) \) be a generator matrix of a nondegenerate code \( C \) of dimension \( k \). So \( G \) has no zero columns. Let \( H_j \) be the linear hyperplane in \( \mathbb{F}^k_q \) with equation

\[
g_{1j}X_1 + \cdots + g_{kj}X_k = 0
\]

The arrangement \( (H_1, \ldots, H_n) \) associated with \( G \) will be denoted by \( A_G \).

Remark 4.4.6 Let \( G \) be a generator matrix of a code \( C \). Then the rank of \( G \) is equal to the number of rows of \( G \). Hence the arrangement \( A_G \) is essential. A code \( C \) is projective if and only if \( d(C^\perp) \geq 3 \) if and only if \( A_G \) is simple. Similarly as in Definition 4.3.17 on equivalent projective systems one defines the equivalence of the dual notion, that is of essential arrangements of hyperplanes in \( \mathbb{P}^r(\mathbb{F}) \). Then there is a one-to-one correspondence between generalized equivalence classes of non-degenerate \( [n, k, d] \) codes over \( \mathbb{F}_q \) and equivalence classes of essential arrangements of \( n \) hyperplanes in \( \mathbb{P}^{k-1}(\mathbb{F}_q) \) as in Proposition 4.3.18.

Example 4.4.7 Consider the matrix \( G \) given by

\[
G = \begin{pmatrix}
1 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 1 & 1 & 1 & 0 & 1
\end{pmatrix}
\]

Let \( C \) be the code over \( \mathbb{F}_q \) with generator matrix \( G \). For \( q = 2 \), this is the simplex code \( S_2(2) \). The columns of \( G \) represent also the coefficients of the lines of \( A_G \). The projective picture of \( A_G \) is given in Figure 4.1.

Proposition 4.4.8 Let \( C \) be a nondegenerate code with generator matrix \( G \). Let \( \mathbf{c} \) be a codeword \( \mathbf{c} = \mathbf{x}G \) for some \( \mathbf{x} \in \mathbb{F}^k \). Then

\[
\text{wt}(\mathbf{c}) = n - \text{number of hyperplanes in } A_G \text{ through } \mathbf{x}.
\]

Proof. Now \( \mathbf{c} = \mathbf{x}G \). So \( c_j = g_{1j}x_1 + \cdots + g_{kj}x_k \). Hence \( c_j = 0 \) if and only if \( \mathbf{x} \) lies on the hyperplane \( H_j \). The results follows, since the weight of \( \mathbf{c} \) is equal to \( n \) minus the number of positions \( j \) such that \( c_j = 0 \).

Remark 4.4.9 The number \( A_w \) of codewords of weight \( w \) equals the number of points that are on exactly \( n - w \) of the hyperplanes in \( A_G \), by Proposition 4.4.8. In particular \( A_n \) is equal to the number of points that is in the complement of
4.4. EXTENDED WEIGHT ENUMERATOR

Figure 4.1: Arrangement of $G$ for $q$ odd and $q$ even

the union of these hyperplanes in $\mathbb{F}_q^k$. This number can be computed by the principle of inclusion/exclusion:

$$A_n = q^k - |H_1 \cup \cdots \cup H_n| = q^k + \sum_{w=1}^{n} (-1)^w \sum_{i_1 < \cdots < i_w} |H_{i_1} \cap \cdots \cap H_{i_w}|.$$

The following notations are introduced to find a formalism as above for the computation of the weight enumerator.

**Definition 4.4.10** For a subset $J$ of $\{1,2,\ldots,n\}$ define

$$C(J) = \{\mathbf{c} \in C \mid c_j = 0 \text{ for all } j \in J\}.$$  
$$l(J) = \dim C(J)$$  
$$B_J = q^{l(J)} - 1 \text{ and } B_t = \sum_{|J|=t} B_J.$$  

**Remark 4.4.11** The encoding map $\mathbf{x} \mapsto \mathbf{x}G = \mathbf{c}$ from vectors $\mathbf{x} \in \mathbb{F}_q^k$ to codewords gives the following isomorphism of vector spaces

$$\bigcap_{j \in J} H_j \cong C(J)$$  

by Proposition 4.4.8. Furthermore $B_J$ is equal to the number of nonzero codewords $\mathbf{c}$ that are zero at all $j$ in $J$, and this is equal to the number of nonzero elements of the intersection $\bigcap_{j \in J} H_j$.

The following two lemmas about the determination of $l(J)$ will become useful later.

**Lemma 4.4.12** Let $C$ be a linear code with generator matrix $G$. Let $J \subseteq \{1,\ldots,n\}$ and $|J| = t$. Let $G_J$ be the $k \times t$ submatrix of $G$ consisting of the columns of $G$ indexed by $J$, and let $r(J)$ be the rank of $G_J$. Then the dimension $l(J)$ is equal to $k - r(J)$. 
Proof. The code $C_J$ is defined in 3.1.2 by restricting the codewords of $C$ to $J$. Then $G_J$ is a generator matrix of $C_J$ by Remark 3.1.3. Consider the the projection map $\pi_J : C \to \mathbb{F}_q^s$ given by $\pi_J(c) = c|_J$. Then $\pi_J$ is a linear map. The image of $C$ under $\pi_J$ is $C_J$ and the kernel of $\pi_J$ is $C(J)$ by definition. It follows that $\dim C_J + \dim C(J) = \dim C$. So $l(J) = k - r(J)$.

Lemma 4.4.13 Let $k$ be the dimension of $C$. Let $d$ and $d^\perp$ be the minimum distance the code $C$ and its dual code, respectively. Then

$$l(J) = \begin{cases} k - t & \text{for all } t < d^\perp, \\ 0 & \text{for all } t > n - d. \end{cases}$$

Furthermore

$$k - t \leq l(J) \leq \begin{cases} k - d^\perp + 1 & \text{for all } t \geq d^\perp, \\ n - d - t + 1 & \text{for all } t \leq n - d. \end{cases}$$

Proof. (1) Let $t > n - d$ and let $J$ be a subset of $\{1, \ldots, n\}$ of size $t$ and $c$ a codeword such that $c \in C(J)$. Then $J$ is contained in the complement of the support of $c$. Hence $t \leq n - \text{wt}(c)$. Hence $\text{wt}(c) \leq n - t < d$. So $c = 0$. Therefore $C(J) = 0$ and $l(J) = 0$.

(2) Let $J$ be $t$-subset of $\{1, \ldots, n\}$. Then $C(J)$ is defined by $t$ homogenous linear equations on the vector space $C$ of dimension $t$. So $l(J) \geq k - t$.

(3) The matrix $G$ is a parity check matrix for the dual code, by (2) of Corollary 2.3.29. Now suppose that $t < d^\perp$. Then any $t$ columns of $G$ are independent, by Proposition 2.3.11. So $l(J) = k - t$ for all $t$-subsets $J$ of $\{1, \ldots, n\}$ by Lemma 4.4.12.

(4) Assume that $t \leq n - d$. Let $J$ be a $t$-subset. Let $t' = n - d + 1$. Choose a $t'$-subset $J'$ such that $J \subseteq J'$. Then

$$C(J') = \{ c \in C(J) \mid c_j = 0 \text{ for all } j \in J' \setminus J \}.$$ 

Now $l(J') = 0$ by (1). Hence $C(J') = 0$ and $C(J')$ is obtained from $C(J)$ by imposing $|J' \setminus J| = n - d - t + 1$ linear homogeneous equations. Hence $l(J) = \dim C(J) \leq n - d - t + 1$.

(5) Assume that $d^\perp \leq t$. Let $J$ be a $t$-subset. Let $t' = d^\perp - 1$. Choose a $t'$-subset $J'$ such that $J' \subseteq J$. Then $l(J') = k - d^\perp + 1$ by (3) and $l(J) \leq l(J')$, since $J' \subseteq J$. Hence $l(J) \leq k - d^\perp + 1$.

Remark 4.4.14 Notice that $d^\perp \leq n - (n - k) + 1$ and $n - d \leq k - 1$ by the Singleton bound. So for $t = k$ both cases of Lemma 4.4.13 apply and both give $l(J) = 0$.

Proposition 4.4.15 Let $k$ be the dimension of $C$. Let $d$ and $d^\perp$ be the minimum distance the code $C$ and its dual code, respectively. Then

$$B_t = \begin{cases} \binom{n}{t}(q^{k-t} - 1) & \text{for all } t < d^\perp, \\ 0 & \text{for all } t > n - d. \end{cases}$$

Furthermore

$$\binom{n}{t}(q^{k-t} - 1) \leq B_t \leq \binom{n}{t}(q^{\min\{n - d - t + 1, k - d^\perp + 1\}} - 1)$$

for all $d^\perp \leq t \leq n - d$. 

Proof.
Proof. This is a direct consequence of Lemma 4.4.13 and the definition of $B_t$.

Proposition 4.4.16 The following formula holds

$$B_t = \sum_{w=d}^{n-t} \binom{n-w}{t} A_w.$$  

Proof. This is shown by computing the number of elements of the set of pairs

$B_t = \{(J, c) \mid J \subseteq \{1, 2, \ldots, n\}, |J| = t, c \in C(J), c \neq 0\}$

in two different ways, as in Lemma 4.1.19.

For fixed $J$, the number of these pairs is equal to $B_J$, by definition.

If we fix the weight $w$ of a nonzero codeword $c$ in $C$, then the number of zero entries of $c$ is $n - w$ and if $c \in C(J)$, then $J$ is contained in the complement of the support of $c$, and there are $\binom{n-w}{t}$ possible choices for such a $J$. In this way we get the right hand side of the formula.

Theorem 4.4.17 The homogeneous weight enumerator of $C$ can be expressed in terms of the $B_t$ as follows.

$$W_C(X,Y) = X^n + \sum_{t=0}^{n} B_t (X - Y)^t Y^{n-t}.$$  

Proof. Now

$$X^n + \sum_{t=0}^{n} B_t (X - Y)^t Y^{n-t} = X^n + \sum_{t=0}^{n-d} B_t (X - Y)^t Y^{n-t},$$  

since $B_t = 0$ for all $t > n - d$ by Proposition 4.4.15. Substituting the formula for $B_t$ of Proposition 4.4.16, interchanging the order of summation in the double sum and applying the binomial expansion of $((X - Y) + Y)^{n-w}$ gives that the above formula is equal to

$$X^n + \sum_{t=0}^{n-d} \sum_{w=d}^{n-t} \binom{n-w}{t} A_w (X - Y)^t Y^{n-t} =$$

$$X^n + \sum_{w=d}^{n} A_w \sum_{t=0}^{n-w} \binom{n-w}{t} (X - Y)^t Y^{n-w-t}) Y^w =$$

$$X^n + \sum_{w=d}^{n} A_w X^{n-w} Y^w = W_C(X,Y).$$

Proposition 4.4.18 Let $A_0, \ldots, A_n$ be the weight spectrum of a code of minimum distance $d$. Then $A_0 = 1$, $A_w = 0$ if $0 < w < d$ and

$$A_w = \sum_{t=n-w}^{n-d} (-1)^{n+w+t} \binom{t}{n-w} B_t \text{ if } d \leq w \leq n.$$
Proof. This identity is proved by inverting the argument of the proof of the formula of Theorem 4.4.17 and using the binomial expansion of \((X - Y)^t\). This is left as an exercise. An alternative proof is given by the principle of inclusion/exclusion. A third proof can be obtained by using Proposition 4.4.16. A fourth proof is obtained by showing that the transformation of the \(B_t\)'s in the \(A_w\)'s and vice versa are given by the linear maps given in Propositions 4.4.16 and 4.4.18 that are each others inverse. See Exercise 4.4.5.

Example 4.4.19 Consider the \([7, 4, 3]\) Hamming code as in Examples 2.2.14 and ?? . Then its dual is the \([7, 3, 4]\) Simplex code. Hence \(d = 3\) and \(d^\perp = 4\). So \(B_t = \binom{7}{t}(2^{4-t} - 1)\) for all \(t < 4\) and \(B_t = 0\) for all \(t > 4\) by Proposition 4.4.15. Of the 35 subsets \(J\) of size 4 there are exactly 7 of them such that \(l(J) = 1\) and \(l(J) = 0\) for the 28 remaining subsets, by Exercise 2.3.4. Therefore \(B_4 = 7(2^{3} - 1) = 7\). To find the the \(A_w\) we apply Proposition 4.4.18.

\[
\begin{align*}
B_0 &= 15  & A_3 &= B_4 &= 7 \\
B_1 &= 49  & A_4 &= B_3 - 4B_4 &= 7 \\
B_2 &= 63  & A_5 &= B_2 - 3B_3 + 6B_4 &= 0 \\
B_3 &= 35  & A_6 &= B_1 - 2B_2 + 3B_3 - 4B_4 &= 0 \\
B_4 &= 7   & A_7 &= B_0 - B_1 + B_2 - B_3 + B_4 &= 1
\end{align*}
\]

This is in agreement with Example 4.1.6

### 4.4.2 Weight distribution of MDS codes

Definition 4.4.20 Let \(C\) be a code of length \(n\), minimum distance \(d\) and dual minimum distance \(d^\perp\). The genus of \(C\) is defined by \(g(C) = \max\{n + 1 - k, k + 1 - d^\perp\}\).

Remark 4.4.21 The \(B_t\) are known as functions of the parameters \([n, k]_q\) of the code for all \(t < d^\perp\) and for all \(t > n - d\). So the \(B_t\) is unknown for the \(n - d - d^\perp + 1\) values of \(t\) such that \(d^\perp \leq t \leq n - d\). In particular the weight enumerator of an MDS code is completely determined by the parameters \([n, k]_q\) of the code.

Proposition 4.4.22 The weight distribution of an MDS code of length \(n\) and dimension \(k\) is given by

\[
A_w = \binom{n}{w} \sum_{j=0}^{w-d} (-1)^j \binom{w}{j} \left( q^{w-d+1-j} - 1 \right).
\]

for \(w \geq d = n - k + 1\).

Proof. Let \(C\) be an \([n, k, n-k+1]\) MDS code. Then its dual is also an MDS code with parameters \([n, n-k, k+1]\) by Proposition 3.2.7. Then \(B_t = \binom{n}{t} \left(q^{k-t} - 1\right)\)
4.4. EXTENDED WEIGHT ENUMERATOR

for all \( t < d+1 = k + 1 \) and \( B_t = 0 \) for all \( t > n - d = k - 1 \) by Proposition 4.4.15. Hence

\[
A_w = \sum_{t=n-w}^{n-d} (-1)^{n+w+t} \binom{n}{t} \binom{t}{n} \left( q^{k-t} - 1 \right)
\]

by Proposition 4.4.18. Make the substitution \( j = t - n + w \). Then the summation is from \( j = 0 \) to \( j = w - d \). Furthermore

\[
\binom{t}{n-w} \binom{n}{t} = \binom{n}{w} \binom{w}{j}.
\]

This gives the formula for \( A_w \).

\[\Box\]

**Remark 4.4.23** Let \( C \) be an \([n,k,n-k+1]\) MDS code. Then the number of nonzero codewords of minimal weight is

\[
A_d = \binom{n}{d} (q-1)
\]

according to Proposition 4.4.22. This is in agreement with Remark 3.2.15.

**Remark 4.4.24** The trivial codes with parameters \([n,n,1]\) and \([n,0,n+1]\), and the repetition code and its dual with parameters \([n,1,n]\) and \([n,n-1,2]\) are MDS codes of arbitrary length. But the length is bounded if \( 2 \leq k \) according to the following proposition.

**Proposition 4.4.25** Let \( C \) be an MDS code over \( \mathbb{F}_q \) of length \( n \) and dimension \( k \). If \( k \geq 2 \), then \( n \leq q + k - 1 \).

**Proof.** Let \( C \) be an \([n,k,n-k+1]\) code such that \( 2 \leq k \). Then \( d+1 = n-k+2 \leq n \) and

\[
A_{d+1} = \binom{n}{d+1} ((q^2-1) - (d+1)(q-1)) = \binom{n}{d+1} (q-1)(q-d)
\]

by Proposition 4.4.22. This implies that \( d \leq q \), since \( A_{d+1} \geq 0 \). Now \( n = \frac{n}{d+1} \leq q + k - 1 \).

\[\Box\]

**Remark 4.4.26** Proposition 4.4.25 also holds for nonlinear codes. That is: if there exists an \((n,q^k,n-k+1)\) code such that \( k \geq 2 \), then \( d = n-k+1 \leq q \). This is proved by means of orthogonal arrays by Bush as we will see in Section 5.5.1.

**Corollary 4.4.27** (Bush bound) Let \( C \) be an MDS code over \( \mathbb{F}_q \) of length \( n \) and dimension \( k \). If \( k \geq q \), then \( n \leq k + 1 \).

**Proof.** If \( n > k + 1 \), then \( C^\perp \) is an MDS code of dimension \( n - k \geq 2 \). Hence \( n \leq q + (n-k) - 1 \) by Proposition 4.4.25. Therefore \( k < q \).

\[\Box\]

**Remark 4.4.28** The length of the repetition code is arbitrary long. The length \( n \) of a \( q \)-ary MDS code of dimension \( k \) is at most \( q+k-1 \) if \( 2 \leq k \), by Proposition 4.4.25. In particular the maximal length of an MDS code is a function of \( k \) and \( q \), if \( k \geq 2 \).
Definition 4.4.29 Let $k \geq 2$. Let $m(k, q)$ be the maximal length of an MDS code over $F_q$ of dimension $k$.

Remark 4.4.30 So $m(k, q) \leq k + q - 1$ if $2 \leq k$, and $m(k, q) \leq k + 1$ if $k \geq q$ by the Bush bound.

We have seen that $m(k, q)$ is at least $q + 1$ for all $k$ and $q$ in Proposition 3.2.10. Let $C$ be an $[n, 2, n - 1]$ code. Then $C$ is systematic at the first two positions, so we may assume that its generator matrix $G$ is of the form

$$G = \begin{pmatrix} 1 & 0 & x_3 & x_4 & \cdots & x_n \\ 0 & 1 & y_3 & y_4 & \cdots & y_n \end{pmatrix}.$$

The weight of all codewords is at least $n - 1$. Hence $x_j \neq 0$ and $y_j \neq 0$ for all $3 \leq j \leq n$. The code is generalized equivalent with a code with $x_j = 1$, after dividing the $j$-th coordinate by $x_j$ for $j \geq 3$. Let $g_i$ be the $i$-th row of $G$. If $3 \leq j < l$ and $y_j = y_l$, then $g_2 - y_j g_1$ is a codeword of weight at most $n - 2$, which is a contradiction. So $n - 2 \leq q - 1$. Therefore $m(2, q) = q + 1$. Dually we get $m(q - 1, q) = q + 1$.

If case $q$ is even, then $m(3, q)$ is least $q + 2$ by the following Example 3.2.12 and dually $m(q - 1, q) \geq q + 2$.

Later it will be shown in Proposition 13.5.1 that these values are in fact optimal.

Remark 4.4.31 The MDS conjecture states that for a nontrivial $[n, k, n - k + 1]$ MDS code over $F_q$ we have that $n \leq q + 2$ if $q$ is even and $k = 3$ or $k = q - 1$; and $n \leq q + 1$ in all other cases. So it is conjectured that

$$m(k, q) = \begin{cases} q + 1 & \text{if } 2 \leq k \leq q, \\ k + 1 & \text{if } q < k, \end{cases}$$

except for $q$ is even and $k = 3$ or $k = q - 1$, then

$$m(3, q) = m(q - 1, q) = q + 2.$$

4.4.3 Extended weight enumerator

Definition 4.4.32 Let $F_{q^m}$ be the extension field of $F_q$ of degree $m$. Let $C$ be a $F_q$-linear code of length $n$. The extension by scalars of $C$ to $F_{q^m}$ is the $F_{q^m}$-linear subspace in $F_{q^m}^n$ generated by $C$ and will be denoted by $C \otimes F_{q^m}$.

Remark 4.4.33 Let $G$ be a generator matrix of the code $C$ of length $n$ over $F_q$. Then $G$ is also a generator matrix of $C \otimes F_{q^m}$ over $F_{q^m}$-linear code with. The dimension $l(J)$ is equal to $k - r(J)$ by Lemma 4.4.12, where $r(J)$ is the rank of the $k \times t$ submatrix $G_J$ of $G$ consisting of the $t$ columns indexed by $J$. This rank is equal to the number of pivots of $G_J$, so this rank does not change by an extension of $F_q$ to $F_{q^m}$. So

$$\dim_{F_{q^m}} C \otimes F_{q^m}(J) = \dim_{F_q} C(J).$$

Hence the numbers $B_J(q^m)$ and $B_t(q^m)$ of the code $C \otimes F_{q^m}$ are equal to

$$B_J(q^m) = q^{m-l(J)} - 1 \quad \text{and} \quad B_t(q^m) = \sum_{|J|=t} B_J(q^m).$$

This motivates to consider $q^m$ as a variable in the following definitions.
Definition 4.4.34 Let $C$ be an $F_q$-linear code of length $n$.

$$B_J(T) = T^{|J|} - 1 \quad \text{and} \quad B_t(T) = \sum_{|J|=t} B_J(T).$$

The extended weight enumerator is defined by

$$W_C(X,Y,T) = X^n + \sum_{t=0}^{n-d} B_t(T)(X-Y)^t Y^{n-t}.$$  

Proposition 4.4.35 Let $d$ and $d^\perp$ be the minimum distance of code and the dual code, respectively. Then

$$B_t(T) = \begin{cases} \binom{n}{t}(T^{k-t} - 1) & \text{for all} \quad t < d^\perp, \\ 0 & \text{for all} \quad t > n - d. \end{cases}$$

Proof. This is a direct consequence of Lemma 4.4.13 and the definition of $B_t$. 

Theorem 4.4.36 The extended weight enumerator of a linear code of length $n$ and minimum distance $d$ can be expressed as a homogeneous polynomial in $X$ and $Y$ of degree $n$ with coefficients $A_w(T)$ that are integral polynomials in $T$.

$$W_C(X,Y,T) = \sum_{w=0}^{n} A_w(T)X^{n-w}Y^w,$$

where $A_0(T) = 1$, and $A_w(T) = 0$ if $0 < w < d$, and

$$A_w(T) = \sum_{t=n-w}^{n-d} (-1)^{n+w+t} \binom{t}{n-w} B_t(T) \quad \text{if} \quad d \leq w \leq n.$$  

Proof. The proof is similar to the proof of Proposition 4.4.18 and is left as an exercise. 

Remark 4.4.37 The definition of $A_w(T)$ is consistent with the fact that $A_w(q^m)$ is the number of codewords of weight $w$ in $C \otimes F_{q^m}$ and

$$W_C(X,Y,q^m) = \sum_{w=0}^{n} A_w(q^m)X^{n-w}Y^w = W_{C \otimes F_{q^m}}(X,Y)$$

by Proposition 4.4.18 and Theorem 4.4.36.

Proposition 4.4.38 The following formula holds

$$B_t(T) = \sum_{w=d}^{n-t} \binom{n-w}{t} A_w(T).$$

Proof. This is left as an exercise.
Remark 4.4.39 Using Theorem 4.4.36 it is immediate to find the weight distribution of a code over any extension $\mathbb{F}_q^m$ if one knows the $l(J)$ over the ground field $\mathbb{F}_q$ for all subsets $J$ of $\{1, \ldots, n\}$. Computing the $C(J)$ and $l(J)$ for a fixed $J$ is just linear algebra. The large complexity for the computation of the weight enumerator and the minimum distance in this way stems from the exponential growth of the number of all possible subsets of $\{1, \ldots, n\}$.

Example 4.4.40 Consider the $[7, 4, 3]$ Hamming code as in Example 4.4.19 but now over all extensions of the binary field. Then $B_t(T) = \binom{n}{t}(T^{4-t} - 1)$ for all $t < 4$ and $B_t = 0$ for all $t > 4$ by Proposition 4.4.35 and $B_4(T) = 7(T - 1) = 7$.

To find the the $A_w(T)$ we apply Theorem 4.4.36.

<table>
<thead>
<tr>
<th>$A_3(T)$</th>
<th>$B_4(T)$</th>
<th>$7(T-1)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_4(T)$</td>
<td>$B_3(T) - 4B_4(T)$</td>
<td>$7(T-1)$</td>
</tr>
<tr>
<td>$A_5(T)$</td>
<td>$B_2(T) - 3B_3(T) + 6B_4(T)$</td>
<td>$21(T-1)(T-2)$</td>
</tr>
<tr>
<td>$A_6(T)$</td>
<td>$B_1(T) - 2B_2(T) + 3B_3(T) - 4B_4(T)$</td>
<td>$7(T-1)(T-2)(T-3)$</td>
</tr>
</tbody>
</table>

Hence

$$A_7(T) = B_0(T) - B_1(T) + B_2(T) - B_3 + B_4(T) = T^4 - 7T^3 + 21T^2 - 28T + 13$$

***factorize, example 4.1.8***

The following description of the extended weight enumerator of a code will be useful.

Proposition 4.4.41 The extended weight enumerator of a code of length $n$ can be written as

$$W_C(X, Y, T) = \sum_{t=0}^{n} \sum_{|J|=t} T^{l(J)}(X - Y)^t Y^{n-t}.$$ 

Proof. By rewriting $((X - Y) + Y)^n$, we get

$$\sum_{t=0}^{n} \sum_{|J|=t} T^{l(J)}(X - Y)^t Y^{n-t} = \sum_{t=0}^{n} (X - Y)^t Y^{n-t} \sum_{|J|=t} ((T^{l(J)} - 1) + 1)$$

$$= \sum_{t=0}^{n} (X - Y)^t Y^{n-t} \left( \binom{n}{t} + \sum_{|J|=t} (T^{l(J)} - 1) \right)$$

$$= \sum_{t=0}^{n} \binom{n}{t} (X - Y)^t Y^{n-t} + \sum_{t=0}^{n} B_t(X - Y)^t Y^{n-t}$$

$$= X^n + \sum_{t=0}^{n} B_t(X - Y)^t Y^{n-t}$$

$$= W_C(X, Y, T).$$

***Examples, repetition code, Hamming, simplex, Golay, MDS code***

***MacWilliams identity***
4.4. EXTENDED WEIGHT ENUMERATOR

4.4.4 Puncturing and shortening

There are several ways to get new codes from existing ones. In this section, we will focus on puncturing and shortening of codes and show how they are used in an alternative algorithm for finding the extended weight enumerator. The algorithm is based on the Tutte-Grothendieck decomposition of matrices introduced by Brylawski [31]. Greene [59] used this decomposition for the determination of the weight enumerator.

Let $C$ be a linear $[n,k]$ code and let $J \subseteq \{1, \ldots, n\}$. Then the code $C$ punctured by $J$ is obtained by deleting all the coordinates indexed by $J$ from the codewords of $C$. The length of this punctured code is $n - |J|$ and the dimension is at most $k$. Let $C$ be a linear $[n,k]$ code and let $J \subseteq \{1, \ldots, n\}$. If we puncture the code $C(J)$ by $J$, we get the code $C$ shortened by $J$. The length of this shortened code is $n - |J|$ and the dimension is $l(J)$.

The operations of puncturing and shortening a code are each others dual: puncturing a code $C$ by $J$ and then taking the dual, gives the same code as shortening $C^\perp$ by $J$.

We have seen that we can determine the extended weight enumerator of a $[n,k]$ code $C$ with the use of a $k \times n$ generator matrix of $C$. This concept can be generalized for arbitrarily matrices, not necessarily of full rank.

**Definition 4.4.42** Let $F$ be a field. Let $G$ be a $k \times n$ matrix over $F$, possibly of rank smaller than $k$ and with zero columns. Then for each $J \subseteq \{1, \ldots, n\}$ we define

$$l(J) = l(J,G) = k - r(G_J).$$

as in Lemma 7.4.37. Define the extended weight enumerator $W_G(X,Y,T)$ as in Definition 4.4.34.

We can now make the following remarks about $W_G(X,Y,T)$.

**Proposition 4.4.43** Let $G$ be a $k \times n$ matrix over $F$ and $W_G(X,Y,T)$ the associated extended weight enumerator. Then the following statements hold:

(i) $W_G(X,Y,T)$ is invariant under row-equivalence of matrices.

(ii) Let $G'$ be a $l \times n$ matrix with the same row-space as $G$, then we have $W_{G'}(X,Y,T) = T^{k-l}W_G(X,Y,T)$. In particular, if $G$ is a generator matrix of a $[n,k]$ code $C$, we have $W_G(X,Y,T) = W_C(X,Y,T)$.

(iii) $W_G(X,Y,T)$ is invariant under permutation of the columns of $G$.

(iv) $W_G(X,Y,T)$ is invariant under multiplying a column of $G$ with an element of $F^\ast$.

(v) If $G$ is the direct sum of $G_1$ and $G_2$, i.e. of the form

$$\begin{pmatrix} G_1 & 0 \\ 0 & G_2 \end{pmatrix},$$

then $W_G(X,Y,T) = W_{G_1}(X,Y,T) \cdot W_{G_2}(X,Y,T)$. 


CHAPTER 4. WEIGHT ENUMERATOR

Proof. (i) If we multiply $G$ from the left with an invertible $k \times k$ matrix, the $r(J)$ do not change, and therefore (i) holds.

For (ii), we may assume without loss of generality that $k \geq l$. Because $G$ and $G'$ have the same row-space, the ranks $r(G_J)$ and $r(G'_J)$ are the same. So $l(J,G) = k - l + l(J,G')$. Using Proposition 4.4.41 we have for $G$

$$W_G(X,Y,T) = \sum_{t=0}^{n} \sum_{|J|=t} T^{l(J,G)}(X - Y)^t Y^{n-t}$$

$$= \sum_{t=0}^{n} \sum_{|J|=t} T^{k-l+(J,G')}(X - Y)^t Y^{n-t}$$

$$= T^{k-l} \sum_{t=0}^{n} \sum_{|J|=t} T^{l(J,G')}(X - Y)^t Y^{n-t}$$

$$= T^{k-l} W_{G'}(X,Y,T).$$

The last part of (ii) and (iii)–(v) follow directly from the definitions.

With the use of the extended weight enumerator for general matrices, we can derive a recursive algorithm to determine the extended weight enumerator of a code. Let $G$ be a $k \times n$ matrix with entries in $F$. Suppose that the $j$-th column is not the zero vector. Then there exists a matrix row-equivalent to $G$ such that the $j$-th column is of the form $(1,0,...,0)^T$. Such a matrix is called reduced at the $j$-th column. In general, this reduction is not unique.

Let $G$ be a matrix that is reduced at the $j$-th column $a$. The matrix $G \setminus a$ is the $k \times (n-1)$ matrix $G$ with the column $a$ removed, and $G/a$ is the $(k-1) \times (n-1)$ matrix $G$ with the column $a$ and the first row removed. We can view $G \setminus a$ as $G$ punctured by $a$, and $G/a$ as $G$ shortened by $a$.

For the extended weight enumerators of these matrices, we have the following connection (we omit the $(X,Y,T)$ part for clarity):

**Proposition 4.4.44** Let $G$ be a $k \times n$ matrix that is reduced at the $j$-th column $a$. For the extended weight enumerator of a reduced matrix $G$ holds

$$W_G = (X - Y)W_{G/a} + YW_{G \setminus a}.$$

Proof. We distinguish between two cases here. First, assume that $G \setminus a$ and $G/a$ have the same rank. Then we can choose a $G$ with all zeros in the first row, except for the 1 in the column $a$. So $G$ is the direct sum of 1 and $G/a$. By Proposition 4.4.43 parts (v) and (ii) we have

$$W_G = (X + (T - 1)Y)W_{G/a} \quad \text{and} \quad W_{G \setminus a} = TW_{G/a}.$$

Combining the two gives

$$W_G = (X + (T - 1)Y)W_{G/a}$$

$$= (X - Y)W_{G/a} + YTW_{G/a}$$

$$= (X - Y)W_{G/a} + YW_{G \setminus a}.$$
For the second case, assume that \(G \setminus a\) and \(G/a\) do not have the same rank. So 
\(r(G \setminus a) = r(G/a) + 1\). This implies \(G\) and \(G \setminus a\) do have the same rank. We have that
\[
W_G(X, Y, T) = \sum_{t=0}^{n} \sum_{|J|=t} T^{l(J,G)}(X - Y)^t Y^{n-t},
\]
by Proposition 4.4.41. This double sum splits into the sum of two parts by distinguishing between the cases \(j \in J\) and \(j \not\in J\).
Let \(j \in J\), \(t = |J|\), \(J' = J \setminus \{j\}\) and \(t' = |J'| = t - 1\). Then
\[
l(J', G/a) = k - 1 - r((G/a)_{J'}) = k - r(G) = l(J, G).
\]
So the first part is equal to
\[
\sum_{t=0}^{n} \sum_{|J|=t\atop j \in J} T^{l(J,G)}(X - Y)^t Y^{n-t} = \sum_{t'=0}^{n-1} \sum_{|J'|=t'} T^{l(J',G/a)}(X - Y)^{t'+1} Y^{n-1-t'}
\]
which is equal to \((X - Y)W_{G/a}\).
Let \(j \not\in J\). Then \((G \setminus a)_J = G_J\). So \(l(J, G \setminus a) = l(J, G)\). Hence the second part is equal to
\[
\sum_{t=0}^{n} \sum_{|J|=t\atop j \not\in J} T^{l(J,G)}(X - Y)^t Y^{n-t} = Y \sum_{t'=0}^{n-1} \sum_{|J'|=t'} T^{l(J,G/a)}(X - Y)^{t'} Y^{n-1-t'}
\]
which is equal to \(YW_{G/a}\). \(\diamondsuit\)

**Theorem 4.4.45** Let \(G\) be a \(k \times n\) matrix over \(\mathbb{F}\) with \(n > k\) of the form \(G = (I_k|P)\), where \(P\) is a \(k \times (n-k)\) matrix over \(\mathbb{F}\). Let \(A \subseteq [k]\) and write \(P_A\) for the matrix formed by the rows of \(P\) indexed by \(A\). Let \(W_A(X, Y, T) = W_{P_A}(X, Y, T)\). Then the following holds:
\[
W_G(X, Y, T) = \sum_{i=0}^{k} \sum_{|A|=i} Y^i(X - Y)^{k-i} W_A(X, Y, T).
\]

**Proof.** We use the formula of the last proposition recursively. We denote the construction of \(G \setminus a\) by \(G_1\) and the construction of \(G/a\) by \(G_2\). Repeating this procedure, we get the matrices \(G_{11}\), \(G_{12}\), \(G_{21}\) and \(G_{22}\). So we get for the weight enumerator
\[
W_G = Y^2 W_{G_{11}} + Y(X - Y) W_{G_{12}} + Y(X - Y) W_{G_{21}} + (X - Y)^2 W_{G_{22}}.
\]
Repeating this procedure \(k\) times, we get \(2^k\) matrices with \(n-k\) columns and \(0, \ldots, k\) rows, which form exactly the \(P_A\). In the diagram are the sizes of the matrices of the first two steps: note that only the \(k \times n\) matrix on top has to be of full rank. The number of matrices of size \((k-i) \times (n-j)\) is given by the
binomial coefficient \( \binom{j}{i} \).

On the last line we have \( W_0(X,Y,T) = X^{n-k} \). This proves the formula.

**Example 4.4.46** Let \( C \) be the even weight code of length \( n = 6 \) over \( \mathbb{F}_2 \). Then a generator matrix of \( C \) is the \( 5 \times 6 \) matrix \( G = (I_5|P) \) with \( P = (1,1,1,1,1)^T \). So the matrices \( P_A \) are \( l \times 1 \) matrices with all ones. We have \( W_0(X,Y,T) = X \) and \( W_l(X,Y,T) = T^{l-1}(X + (T - 1)Y) \) by part (ii) of Proposition 4.4.43. Therefore the weight enumerator of \( C \) is equal to

\[
W_C(X,Y,T) = W_G(X,Y,T) = X^6 + 15(T - 1)X^4Y^2 + 20(T^2 - 3T + 2)X^3Y^3 + 15(T^3 - 4T^2 + 6T - 3)X^2Y^4 + 6(T^4 - 5T^3 + 10T^2 - 10T + 4)XY^5 + (T^5 - 6T^4 + 15T^3 - 20T^2 + 15T - 5)Y^6.
\]

For \( T = 2 \) we get \( W_C(X,Y,2) = X^6 + 15X^4Y^2 + 15X^2Y^4 + Y^6 \), which we indeed recognize as the weight enumerator of the even weight code that we found in Example 4.1.5.

**4.4.5 Exercises**

4.4.1 Compute the extended weight enumerator of the binary simplex code \( S_3(2) \).

4.4.2 Compute the extended weight enumerators of the \( n \)-fold repetition code and its dual.

4.4.3 Compute the extended weight enumerator of the binary Golay code.

4.4.4 Compute the extended weight enumerator of the ternary Golay code.

4.4.5 Consider the square matrices \( A \) and \( B \) of size \( n + 1 \) with entries \( a_{ij} \) and \( b_{ij} \), respectively given by

\[
a_{ij} = (-1)^{i+j} \binom{i}{j}, \quad b_{ij} = \binom{i}{j} \quad \text{for } 0 \leq i, j \leq n.
\]

Show that \( A \) and \( B \) are inverses of each other.
4.4.6 Give a proof of Theorem 4.4.36.

4.4.7 Give a proof of Proposition 4.4.38.

4.4.8 Compare the complexity of the methods "exhaustive search" and "arrangements of hyperplanes" to compute the weight enumerator as a function of $q$ and the parameters $[n, k, d]$ and $d^\perp$.

4.5 Generalized weight enumerator

### Intro

4.5.1 Generalized Hamming weights

We recall that for a linear code $C$, the minimum Hamming weight is the minimal one among all Hamming weights $\text{wt}(c)$ for nonzero codewords $c \neq 0$. In this subsection, we generalize this parameter to get a sequence of values, the so-called generalized Hamming weights, which are useful in the study of the complexity of the trellis decoding and other properties of the code $C$.

Let $D$ be a subcode of $C$. Generalizing Definition 2.2.2, we define the support of $D$, denoted by $\text{supp}(D)$, as the set of positions where at least one codeword in $D$ is not zero, i.e.,

$$\text{supp}(D) = \{i \mid \text{there exists } x \in D, \text{ such that } x_i \neq 0\}.$$  

The weight of $D$, $\text{wt}(D)$, is defined as the size of $\text{supp}(D)$.

Suppose $C$ is an $[n, k]$ code. For any $1 \leq r \leq k$, the $r$-th generalized Hamming weight (GHW) of $C$ is defined as

$$d_r(C) = \min \{\text{wt}(D) \mid D \text{ is a } k-r \text{-dimensional subcode of } C\}.$$  

The set of GHWs $\{d_1(C), \ldots, d_k(C)\}$ is called the weight hierarchy of $C$. Note that since any $1$-dimensional subcode has a nonzero codeword as its basis, the first generalized Hamming weight $d_1(C)$ is exactly equal to the minimum weight of $C$.

We now state several properties of generalized Hamming weights.

**Proposition 4.5.1 (Monotonicity)** For an $[n, k]$ code $C$, the generalized Hamming weights satisfy

$$1 \leq d_1(C) < d_2(C) < \ldots < d_k(C) \leq n.$$  

**Proof.** For any $1 \leq r \leq k-1$, it is trivial to verify $1 \leq d_r(C) \leq d_{r+1}(C) \leq n$.

Let $D$ be a subcode of dimension $r+1$, such that $\text{wt}(D) = d_{r+1}(C)$. We choose any index $i \in \text{supp}(D)$. Consider

$$E = \{x \mid x \in D, \text{ and } x_i = 0\}.$$  

We now state several properties of generalized Hamming weights.
CHAPTER 4. WEIGHT ENUMERATOR

By Definition 3.1.13 and Proposition 3.1.15, $E$ is a shortened code of $D$, and $r \leq \dim(E) \leq r + 1$. However, by the choice of $i$, there exists a codeword $c \in D$ with $c_i \neq 0$. Thus, $c$ cannot be a codeword of $E$. This implies that $E$ is a proper subcode of $D$, that is $\dim(E) = r$. Now, by the definition of the GHWs, we have

$$d_r(C) \leq \wt(E) \leq \wt(D) - 1 = d_{r+1}(C) - 1.$$  

This proves that $d_r(C) < d_{r+1}(C)$.

Proposition 4.5.2 (Generalized Singleton Bound) For an $[n, k]$ code $C$, we have

$$d_r(C) \leq n - k + r.$$  

This bound on $d_r(C)$ is a straightforward consequence of the Proposition 4.5.1. When $r = 1$, we get the Singleton bound (see Theorem 3.2.1).

Let $H$ be a parity check matrix of the $[n, k]$ code $C$, which is a $(n-k) \times n$ matrix of rank $n-k$. From Proposition 2.3.11, we know that the minimum distance of $C$ is the smallest integer $d$ such that $d$ columns of $H$ are linearly dependent. We now present a generalization of this property. Let $H_i$, $1 \leq i \leq n$, be the column vectors of $H$. For any subset $I$ of $\{1, 2, \ldots, n\}$, let $\langle H_i \mid i \in I \rangle$ be the subset of $\mathbb{F}_q^n$ generated by the vectors $H_i$, $i \in I$, which, for simplicity, is denoted by $V_I$.

Lemma 4.5.3 The $r$-th generalized Hamming weight of $C$ is

$$d_r(C) = \min\{|I| \mid \dim(\langle H_i \mid i \in I \rangle) \leq |I| - r\}.$$  

Proof. We denote $V_I^\perp = \{x \mid x_i = 0 \text{ for } i \notin I, \text{ and } \sum_{i \in I} x_i H_i = 0\}$. Then it is easy to see that $\dim(V_I) + \dim(V_I^\perp) = |I|$. Also, from the definition, for any $I$, $V_I^\perp$ is a subcode of $C$.

Let $D$ be a subcode of $C$ with $\dim(D) = r$ and $|\supp(D)| = d_r(C)$. Let $I = \supp(D)$. Then $D \subseteq V_I^\perp$. This implies that $\dim(V_I) = |I| - \dim(V_I^\perp) \leq |I| - \dim(D) = |I| - r$. Therefore, $d_r(C) = |\supp(D)| = |I| \geq \min\{|I| \mid \dim(V_I) \leq |I| - r\}$. We now prove the inverse inequality. Denote $d = \min\{|I| \mid \dim(V_I) \leq |I| - r\}$. Let $I$ be a subset of $\{1, 2, \ldots, n\}$ such that $\dim(V_I) \leq |I| - r$ and $|I| = d$. Then $\dim(V_I^\perp) \geq r$. Therefore, $d_r(C) \leq |\supp(V_I^\perp)| \leq |I| = d$.

Proposition 4.5.4 (Duality) Let $C$ be an $[n, k]$ code. Then the weight hierarchy of its dual code $C^\perp$ is completely determined by the weight hierarchy of $C$, precisely

$$\{d_r(C^\perp) \mid 1 \leq r \leq n-k\} = \{1, 2, \ldots, n\} \setminus \{n + 1 - d_s(C) \mid 1 \leq s \leq k\}.$$  

Proof. Look at the two sets $\{d_r(C^\perp) \mid 1 \leq r \leq n-k\}$ and $\{n + 1 - d_s(C) \mid 1 \leq s \leq k\}$. Both are subsets of $\{1, 2, \ldots, n\}$. And by the Monotonicity, the first one has size $n-k$, the second one has size $k$. Thus, it is sufficient to prove that these two sets are distinct.

We now prove an equivalent fact that for any $1 \leq r \leq k$, the value $n + 1 - d_r(C)$ is not a generalized Hamming weight of $C^\perp$. Let $t = n - k + r - d_r(C)$.
It is sufficient to prove that \( d_t(C^+) < n + 1 - d_r(C) \) and for any \( \delta \geq 1 \), \( d_{t+\delta}(C^+) \neq n + 1 - d_r(C) \). Let \( D \) be a subcode of \( C \) with \( \dim(D) = r \) and \( |\text{supp}(D)| = d_r(C) \). There exists a parity check matrix \( G \) for \( C^+ \) (which is a generator matrix for \( C \)), where the first \( r \) rows are words in \( D \), and the last \( k - r \) rows are not. The column vectors \( \{ G_i \mid i \notin \text{supp}(D) \} \) have their first \( r \) coordinates zero. Thus, \( \dim((G_i \mid i \notin \text{supp}(D)) = \text{column rank of the matrix (G_i \mid i \notin \text{supp}(D))} \leq \text{row rank of the matrix (R_i \mid r + 1 \leq i \leq k) = k} \). By Lemma 4.5.3, \( \dim(I) = n - d_r(C) \). And \( \dim((G_i \mid i \in I)) \leq k - r = |I| - t \). Thus, by Lemma 4.5.3, we have \( d_t(C^+) \leq |I| = n - d_r(C) < n - d_r(C) + 1 \).

Next, we show \( d_{t+\delta}(C^+) \neq n + 1 - d_r(C) \). Otherwise, \( d_{t+\delta}(C^+) = n + 1 - d_r(C) \) holds for some \( \delta \). Then by the definition of generalized Hamming weight, there exists a generator matrix \( H \) for \( C^+ \) (which is a parity check matrix for \( C \)) and \( d_r(C) - 1 \) positions \( 1 \leq i_1, \ldots, i_{d_r(C)-1} \leq n \), such that the coordinates of the first \( t + \delta \) rows of \( H \) are all zero at these \( d_r(C) - 1 \) positions. Without loss of generality, we assume these positions are exactly the last \( d_r(C) - 1 \) positions \( n - d_r(C) + 2, \ldots, n \). And let \( I = \{ n - d_r(C) + 2, \ldots, n \} \). Clearly, the last \( |I| \) column vectors span a space of dimension \( \leq n - k - t - \delta = d_r(C) - r - \delta \). By Lemma 4.5.3, \( d_s(I) \leq d_r(C) - 1 \), where \( s = |I| - (d_r(C) - r - \delta) = r + \delta - 1 \geq r \). This contradicts the Monotonicity.

### 4.5.2 Generalized weight enumerators

The weight distribution is generalized in the following way. Instead of looking at words of \( C \), we consider all the subcodes of \( C \) of a certain dimension \( r \).

**Definition 4.5.5** Let \( C \) be a linear code of length \( n \). The number of subcodes with a given weight \( w \) and dimension \( r \) is denoted by \( A_w^{(r)} \), that is

\[
A_w^{(r)} = |\{ D \subseteq C \mid \dim(D) = r, \text{wt}(D) = w \}|.
\]

Together they form the \( r \)-th *generalized weight distribution* of the code. The \( r \)-th *generalized weight enumerator* \( W_C^{(r)}(X,Y) \) of \( C \) is the polynomial with the weight distribution as coefficients, that is

\[
W_C^{(r)}(X,Y) = \sum_{w=0}^{n} A_w^{(r)} X^{n-w} Y^w.
\]

**Remark 4.5.6** From this definition it follows that \( A_0^{(0)} = 1 \) and \( A_w^{(r)} = 0 \) for all \( 0 < w \leq n \). Furthermore, every 1-dimensional subspace of \( C \) contains \( q-1 \) non-zero codewords, so \( (q-1)A_w^{(1)} = A_w \) for \( 0 < w \leq n \). This means we can find back the original weight enumerator by using

\[
W_C(X,Y) = W_C^{(0)}(X,Y) + (q-1)W_C^{(1)}(X,Y).
\]

**Definition 4.5.7** We introduce the following notations:

\[
[m,r]_q = \prod_{i=0}^{r-1} (q^m - q^i)
\]
\[ \langle r \rangle_q = [r, r]_q \]
\[ \binom{k}{r}_q = \frac{[k, r]_q}{\langle r \rangle_q} \]

Remark 4.5.8 In Proposition 2.5.2 it is shown that the first number is equal to the number of \(m \times r\) matrices of rank \(r\) over \(F_q\). and the third number is the Gaussian binomial, and it represents the number of \(r\)-dimensional subspaces of \(F^k_q\). Hence the second number is the number of bases of \(F^r_q\).

Definition 4.5.9 For \(J \subseteq \{1, \ldots, n\}\) and \(r \geq 0\) an integer we define:
\[ B^{(r)}_J = |\{ D \subseteq C(J) | D \text{ subspace of dimension } r \}| \]
\[ B^{(r)}_t = \sum_{|J|=t} B^{(r)}_J \]

Remark 4.5.10 Note that \(B^{(r)}_J = \binom{l(J)}{r}_q\). For \(r = 0\) this gives \(B^{(0)}_t = \binom{n}{t}\). So we see that in general \(l(J) = 0\) does not imply \(B^{(r)}_J = 0\), because \(\binom{0}{r}_q = 1\). But if \(r \neq 0\), we do have that \(l(J) = 0\) implies \(B^{(r)}_J = 0\) and \(B^{(r)}_t = 0\).

Proposition 4.5.11 Let \(d_r\) be the \(r\)-th generalized Hamming weight of \(C\), and \(d^\perp\) the minimum distance of the dual code \(C^\perp\). Then we have
\[ B^{(r)}_t = \binom{n}{t} \binom{k-t}{r}_q \text{ for all } t < d^\perp \]
\[ 0 \text{ for all } t > n - d_r \]

Proof. The first case is a direct corollary of Lemma 4.4.13, since there are \(\binom{n}{t}\) subsets \(J \subseteq \{1, \ldots, n\}\) with \(|J| = t\). The proof of the second case goes analogous to the proof of the same lemma: let \(|J| = t, t > n - d_r\) and suppose there is a subspace \(D \subseteq C(J)\) of dimension \(r\). Then \(J\) is contained in the complement of \(\text{supp}(D)\), so \(t \leq n - \text{wt}(D)\). It follows that \(\text{wt}(D) \leq n - t < d_r\), which is impossible, so such a \(D\) does not exist. So \(B^{(r)}_J = 0\) for all \(J\) with \(|J| = t\) and \(t > n - d_r\), and therefore \(B^{(r)}_t = 0\) for \(t > n - d_r\).

We can check that the formula is well-defined: if \(t < d^\perp\) then \(l(J) = k - t\). If also \(t > n - d_r\), we have \(t > n - d_r \geq k - r\) by the generalized Singleton bound. This implies \(r > k - t = l(J)\), so \(\binom{k-t}{r}_q = 0\).

The relation between \(B^{(r)}_t\) and \(A^{(r)}_w\) becomes clear in the next proposition.

Proposition 4.5.12 The following formula holds:
\[ B^{(r)}_t = \sum_{w=0}^{n} \binom{n-w}{t} A^{(r)}_w. \]

Proof. We will count the elements of the set
\[ B^{(r)}_t = \{(D, J) | J \subseteq \{1, \ldots, n\}, |J| = t, D \subseteq C(J) \text{ subspace of dimension } r\} \]
in two different ways. For each \( J \) with \(|J| = t\) there are \( B_{J}^{(r)} \) pairs \((D,J)\) in \( B_{J}^{(r)} \), so the total number of elements in this set is \( \sum_{|J|=t} B_{J}^{(r)} = B_{t}^{(r)} \). On the other hand, let \( D \) be an \( r \)-dimensional subcode of \( C \) with \( \text{wt}(D) = w \). There are \( A_{w}^{(r)} \) possibilities for such a \( D \). If we want to find a \( J \) such that \( D \subseteq C(J) \), we have to pick \( t \) coordinates from the \( n-w \) all-zero coordinates of \( D \). Summation over all \( w \) proves the given formula.

Note that because \( A_{w}^{(r)} = 0 \) for all \( w < d_{r} \), we can start summation at \( w = d_{r} \). We can end summation at \( w = n-t \) because for \( t > n-w \) we have \( \binom{n-w}{t} = 0 \). So the formula can be rewritten as

\[
B_{t}^{(r)} = \sum_{w=d_{r}}^{n-t} \binom{n-w}{t} A_{w}^{(r)}.
\]

In practice, we will often prefer the summation given in the proposition.

**Theorem 4.5.13** The generalized weight enumerator is given by the following formula:

\[
W_{C}^{(r)}(X,Y) = \sum_{t=0}^{n} B_{t}^{(r)}(X - Y)^t Y^{n-t}.
\]

**Proof.** The proof is similar to the one given for Theorem 4.4.17 and is left as an exercise.

It is possible to determine the \( A_{w}^{(r)} \) directly from the \( B_{t}^{(r)} \), by using the next proposition.

**Proposition 4.5.14** The following formula holds:

\[
A_{w}^{(r)} = \sum_{t=n-w}^{n} (-1)^{n+w+t} \binom{t}{n-w} B_{t}^{(r)}.
\]

**Proof.** The proof is similar to the one given for Proposition 4.4.18 and is left as an exercise.

### 4.5.3 Generalized weight enumerators of MDS-codes

We can use the theory in Sections 4.5.2 and 4.4.3 to calculate the weight distribution, generalized weight distribution, and extended weight distribution of a linear \([n,k]\) code \( C \). This is done by determining the values \( l(J) \) for each \( J \subseteq \{1, \ldots, n\} \). In general, we have to look at the \( 2^n \) different subcodes of \( C \) to find the \( l(J) \), but for the special case of MDS codes we can find the weight distributions much faster.

**Proposition 4.5.15** Let \( C \) be a linear \([n,k]\) MDS code, and let \( J \subseteq \{1, \ldots, n\} \). Then we have

\[
l(J) = \begin{cases} 
0 & \text{for } t > k \\
 k - t & \text{for } t \leq k
\end{cases}
\]

so for a given \( t \) the value of \( l(J) \) is independent of the choice of \( J \).
Proof. We know that the dual of an MDS code is also MDS, so \( d^\perp = k + 1 \). Now use \( d = n - k + 1 \) in Lemma 7.4.39.

Now that we know all the \( l(J) \) for an MDS code, it is easy to find the weight distribution.

**Theorem 4.5.16** Let \( C \) be an MDS code with parameters \([n,k]\). Then the generalized weight distribution is given by

\[
A_w^{(r)} = \binom{n}{w} \sum_{j=0}^{w-d} (-1)^j \binom{w}{j} \left[ \binom{w - d + 1 - j}{r} \right]_q.
\]

The coefficients of the extended weight enumerator are given by

\[
A_w(T) = \binom{n}{w} \sum_{j=0}^{w-d} (-1)^j \binom{w}{j} \left( T^{w-d+1-j} - 1 \right).
\]

Proof. We will give the construction for the generalized weight enumerator here: the case of the extended weight enumerator goes similar and is left as an exercise. We know from Proposition 4.5.15 that for an MDS code, \( B_t^{(r)} \) depends only on the size of \( J \), so \( B_t^{(r)} = \binom{n}{t} \left[ \frac{k-t}{r} \right]_q \). Using this in the formula for \( A_w^{(r)} \) and substituting \( j = t - n + w \), we have

\[
A_w^{(r)} = \sum_{t=n-w}^{n-d} (-1)^{n+w+t} \binom{t}{n-w} B_t^{(r)}
\]

\[
= \sum_{t=n-w}^{n-d} (-1)^{n+w+t} \binom{n}{t} \binom{k-t}{r} \left[ \binom{w}{j} \right]_q
\]

\[
= \sum_{j=0}^{w-d} (-1)^j \binom{n}{w} \binom{w-j}{r} \left[ \binom{k+w-n-j}{r} \right]_q
\]

\[
= \binom{n}{w} \sum_{j=0}^{w-d} (-1)^j \binom{w-j}{r} \left[ \binom{w-d+1-j}{r} \right]_q.
\]

In the second step, we are using the binomial equivalence

\[
\binom{n}{t} \binom{t}{n-w} = \binom{n}{n-w} \binom{n-(n-w)}{t-(n-w)} = \binom{n}{w} \binom{w}{n-t}.
\]

So, for all MDS-codes with given parameters \([n,k]\) the extended and generalized weight distributions are the same. But not all such codes are equivalent. We can conclude from this, that the generalized and extended weight enumerators are not enough to distinguish between codes with the same parameters. We illustrate the non-equivalence of two MDS codes by an example.
Example 4.5.17 Let $C$ be a linear $[n, 3]$ MDS code over $\mathbb{F}_q$. It is possible to write the generator matrix $G$ of $C$ in the following form:

$$
\begin{pmatrix}
1 & 1 & \ldots & 1 \\
x_1 & x_2 & \ldots & x_n \\
y_1 & y_2 & \ldots & y_n
\end{pmatrix}.
$$

Because $C$ is MDS we have $d = n - 2$. We now view the $n$ columns of $G$ as points in the projective plane $\mathbb{P}^2(\mathbb{F}_q)$, say $P_1, \ldots , P_n$. The MDS property that every $k$ columns of $G$ are independent is now equivalent with saying that no three points are on a line. To see that these $n$ points do not always determine an equivalent code, consider the following construction. Through the $n$ points there are $\binom{n}{3} = N$ lines, the set $\mathcal{N}$. These lines determine (the generator matrix of) a $[N, 3]$ code $\hat{C}$. The minimum distance of the code $\hat{C}$ is equal to the total number of lines minus the maximum number of lines from $\mathcal{N}$ through an arbitrarily point $P \in \mathbb{P}^2(\mathbb{F}_q)$ by Proposition 4.4.8. If $P \notin \{P_1, \ldots , P_n\}$ then the maximum number of lines from $\mathcal{N}$ through $P$ is at most $\frac{1}{2}n$, since no three points of $\mathcal{N}$ lie on a line. If $P = P_i$ for some $i \in \{1, \ldots , n\}$ then $P$ lies on exactly $n - 1$ lines of $\mathcal{N}$, namely the lines $P_iP_j$ for $j \neq i$. Therefore the minimum distance of $\hat{C}$ is $d = N - n + 1$.

We now have constructed a $[N, 3, N - n + 1]$ code $\hat{C}$ from the original code $C$. Notice that two codes $\hat{C}_1$ and $\hat{C}_2$ are generalized equivalent if $C_1$ and $C_2$ are generalized equivalent. The generalized and extended weight enumerators of an MDS code of length $n$ and dimension $k$ are completely determined by the pair $(n, k)$, but this is not generally true for the weight enumerator of $\hat{C}$.

Take for example $n = 6$ and $q = 9$, so $\hat{C}$ is a $[15, 3, 10]$ code. Look at the codes $C_1$ and $C_2$ generated by the following matrices respectively, where $\alpha \in \mathbb{F}_9$ is a primitive element:

$$
\begin{pmatrix}
1 & 1 & 1 & 1 & 1 & 1 \\
0 & 1 & 0 & 1 & \alpha^5 & \alpha^6 \\
0 & 0 & 1 & \alpha^3 & \alpha & \alpha^3
\end{pmatrix}, \quad
\begin{pmatrix}
1 & 1 & 1 & 1 & 1 & 1 \\
0 & 1 & 0 & \alpha^7 & \alpha^4 & \alpha^6 \\
0 & 0 & 1 & \alpha^5 & \alpha & 1
\end{pmatrix}
$$

Being both MDS codes, the weight distribution is $(1, 0, 0, 120, 240, 368)$. If we now apply the above construction, we get $\hat{C}_1$ and $\hat{C}_2$ generated by

$$
\begin{pmatrix}
1 & 0 & 0 & 1 & 1 & \alpha^4 & \alpha^6 & \alpha^3 & \alpha^7 & \alpha & 1 & \alpha^2 & 1 & \alpha^7 & 1 \\
0 & 1 & 0 & \alpha^7 & 1 & 0 & \alpha^4 & 1 & 1 & 0 & \alpha^6 & \alpha & 1 & \alpha^3 \\
0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1
\end{pmatrix}, \quad
\begin{pmatrix}
1 & 0 & 0 & \alpha^7 & \alpha^2 & \alpha^3 & \alpha & 0 & \alpha^7 & \alpha^4 & \alpha^7 & \alpha & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & \alpha^3 & 0 & \alpha^6 & \alpha^4 & \alpha & \alpha^6 & \alpha^3 & \alpha \\
0 & 0 & 1 & \alpha^5 & \alpha^5 & \alpha^6 & \alpha^3 & \alpha^7 & \alpha^4 & \alpha^5 & \alpha^2 & \alpha^4 & \alpha & \alpha^5
\end{pmatrix}
$$

The weight distribution of $\hat{C}_1$ and $\hat{C}_2$ are, respectively,

$$(1, 0, 0, 0, 0, 0, 0, 0, 48, 0, 16, 312, 288, 64) \text{ and } (1, 0, 0, 0, 0, 0, 0, 0, 48, 0, 32, 264, 336, 48).$$

So the latter two codes are not generalized equivalent, and therefore not all $[6, 3, 4]$ MDS codes over $\mathbb{F}_9$ are generalized equivalent.
Another example was given in [110, 29] showing that two [6, 3, 4] MDS codes could have distinct covering radii.

4.5.4 Connections

There is a connection between the extended weight enumerator and the generalized weight enumerators. We first proof the next proposition.

**Proposition 4.5.18** Let \( C \) be a linear \([n, k]\) code over \( \mathbb{F}_q \), and let \( C^m \) be the linear subspace consisting of the \( m \times n \) matrices over \( \mathbb{F}_q \) whose rows are in \( C \). Then there is an isomorphism of \( \mathbb{F}_q \)-vector spaces between \( C \otimes \mathbb{F}_q^m \) and \( C^m \).

**Proof.** Let \( \alpha \) be a primitive \( m \)-th root of unity in \( \mathbb{F}_q^m \). Then we can write an element of \( \mathbb{F}_q^m \) in an unique way on the basis \((1, \alpha, \alpha^2, \ldots, \alpha^{m-1})\) with coefficients in \( \mathbb{F}_q \). If we do this for all the coordinates of a word in \( C \otimes \mathbb{F}_q^m \), we get an \( m \times n \) matrix over \( \mathbb{F}_q \). The rows of this matrix are words of \( C \), because \( C \) and \( C \otimes \mathbb{F}_q^m \) have the same generator matrix. This map is clearly injective. There are \( (q^m)^k = q^{km} \) words in \( C \otimes \mathbb{F}_q^m \), and the number of elements of \( C^m \) is \( (q^k)^m = q^{km} \), so our map is a bijection. It is given by

\[
\left( \sum_{i=0}^{m-1} c_{1i} \alpha^i, \sum_{i=0}^{m-1} c_{2i} \alpha^i, \ldots, \sum_{i=0}^{m-1} c_{ni} \alpha^i \right) \mapsto \\
\begin{pmatrix}
  c_{01} & c_{02} & c_{03} & \cdots & c_{0n} \\
  c_{11} & c_{12} & c_{13} & \cdots & c_{1n} \\
  \vdots & \vdots & \vdots & \ddots & \vdots \\
  c_{(m-1)1} & c_{(m-1)2} & c_{(m-1)3} & \cdots & c_{(m-1)n}
\end{pmatrix}
\]

We see that the map is \( \mathbb{F}_q \)-linear, so it gives an isomorphism \( C \otimes \mathbb{F}_q^m \to C^m \). \( \diamond \)

Note that this isomorphism depends on the choice of a primitive element \( \alpha \). We also need the next subresult.

**Lemma 4.5.19** Let \( \mathbf{c} \in C \otimes \mathbb{F}_q^m \) and \( M \in C^m \) the corresponding \( m \times n \) matrix under a given isomorphism. Let \( D \subseteq C \) be the subcode generated by the rows of \( M \). Then \( \text{wt}(\mathbf{c}) = \text{wt}(D) \).

**Proof.** If the \( j \)-th coordinate \( c_j \) of \( \mathbf{c} \) is zero, then the \( j \)-th column of \( M \) consists of only zero’s, because the representation of \( c_j \) on the basis \((1, \alpha, \alpha^2, \ldots, \alpha^{m-1})\) is unique. On the other hand, if the \( j \)-th column of \( M \) consists of all zeros, then \( c_j \) is also zero. Therefore \( \text{wt}(\mathbf{c}) = \text{wt}(D) \). \( \diamond \)

**Proposition 4.5.20** Let \( C \) be a linear code over \( \mathbb{F}_q \). Then the weight numerator of an extension code and the generalized weight enumerators are connected via

\[
A_w(q^m) = \sum_{r=0}^{m} [m, r]_q A_w^{(r)}.
\]

**Proof.** We count the number of words in \( C \otimes \mathbb{F}_q^m \) of weight \( w \) in two ways, using the bijection of Proposition 4.5.18. The first way is just by substituting \( T = q^m \) in \( A_w(T) \): this gives the left side of the equation. For the second way,
note that every $M \in C^m$ generates a subcode of $C$ whose weight is equal to the weight of the corresponding word in $C \otimes \mathbb{F}_q^m$. Fix this weight $w$ and a dimension $r$: there are $A_w^{(r)}$ subcodes of $C$ of dimension $r$ and weight $w$. Every such subcode is generated by an $m \times r$ matrix whose rows are words of $C$. Left multiplication by an $m \times r$ matrix of rank $r$ gives an element of $C^m$ which generates the same subcode of $C$, and all such elements of $C^m$ are obtained this way. The number of $m \times r$ matrices of rank $r$ is $\left\langle m, r \right\rangle_q$, so summation over all dimensions $r$ gives

$$A_w(q^m) = \sum_{r=0}^{k} \left\langle m, r \right\rangle_q A_w^{(r)}.$$ 

We can let the summation run to $m$, because $A_w^{(r)} = 0$ for $r > k$ and $\left\langle m, r \right\rangle_q = 0$ for $r > m$. This proves the given formula.

In general, we have the following theorem.

**Theorem 4.5.21** Let $C$ be a linear code over $\mathbb{F}_q$. Then the extended weight numerator is determined by the generalized weight enumerators:

$$W_C(X,Y,T) = \sum_{r=0}^{k} \left( \prod_{j=0}^{r-1} (T - q^j) \right) W_C^{(r)}(X,Y).$$

**Proof.** If we know $A_w^{(r)}$ for all $r$, we can determine $A_w(q^m)$ for every $m$. If we have $k + 1$ values of $m$ for which $A_w(q^m)$ is known, we can use Lagrange interpolation to find $A_w(T)$, for this is a polynomial in $T$ of degree at most $k$. In fact, we have

$$A_w(T) = \sum_{r=0}^{k} \left( \prod_{j=0}^{r-1} (T - q^j) \right) A_w^{(r)}.$$ 

This formula has the right degree and is correct for $T = q^m$ for all integer values $m \geq 0$, so we know it must be the correct polynomial. Therefore the theorem follows.

The converse of the theorem is also true: we can write the generalized weight enumerator in terms of the extended weight enumerator. For this end the following lemma is needed.

**Lemma 4.5.22**

$$\prod_{j=0}^{r-1} (Z - q^j) = \sum_{j=0}^{r} \left[ \begin{array}{c} r \\ j \end{array} \right] q^{-j r} (\bar{z})^j Z^j.$$ 

**Proof.** This identity can be proven by induction and is left as an exercise.

**Theorem 4.5.23** Let $C$ be a linear code over $\mathbb{F}_q$. Then the $r$-th generalized weight enumerator is determined by the extended weight enumerator:

$$W_C^{(r)}(X,Y) = \frac{1}{(r)_q} \sum_{j=0}^{r} \left[ \begin{array}{c} r \\ j \end{array} \right] q^{-j r} (\bar{z})^j W_C(X,Y,q^j).$$
**Proof.** We consider the generalized weight enumerator in terms of Theorem 4.5.13. Using Remark ?? and rewriting gives the following:

\[ W_C^{(r)}(X,Y) = \sum_{t=0}^{n} B_t^{(r)} (X - Y)^t Y^{n-t} \]

\[ = \sum_{t=0}^{n} \sum_{|J|=t} \binom{|J|}{r} q^{|J|} (X - Y)^t Y^{n-t} \]

\[ = \sum_{t=0}^{n} \prod_{j=0}^{r-1} \left( \frac{q^{l(J)} - q^j}{q^r - q^j} \right) (X - Y)^t Y^{n-t} \]

\[ = \frac{1}{\prod_{v=0}^{r-1} (q^r - q^v)} \sum_{t=0}^{n} \sum_{|J|=t} \left( \prod_{j=0}^{r-1} (q^{l(J)} - q^j) \right) (X - Y)^t Y^{n-t} \]

\[ = \frac{1}{\langle r \rangle q} \sum_{t=0}^{r} \sum_{|J|=t} \left( \prod_{j=0}^{r} \left( -1 \right)^{r-j} q^{r-j} \sum_{t=0}^{n} (q^j)^{l(J)} (X - Y)^t Y^{n-t} \right) \]

\[ = \frac{1}{\langle r \rangle q} \sum_{t=0}^{r} \sum_{|J|=t} \left( -1 \right)^{r-j} q^{r-j} W_C(X,Y,q^j) \]

In the fourth step Lemma 4.5.22 is used.

\[ \Box \]

### 4.5.5 Exercises

**4.5.1** Give a proof of Proposition 4.5.16.

**4.5.2** Compute the generalized weight enumerator of the binary Golay code.

**4.5.3** Compute the generalized weight enumerator of the ternary Golay code.

**4.5.4** Give a proof of Lemma 4.5.22.

### 4.6 Notes

Puncturing and shortening at arbitrary sets of positions and the duality theorem is from Simonis [?].

Golay code, Turyn [?] construction, Pless handbook [?].

MacWillimas

***–puncturing gives the binary [23,12,7] Golay code, which is cyclic.
–automorphism group of (extended) Golay code.
– (extended) ternary Golay code.**
- designs and Golay codes.
- lattices and Golay codes.

***repeated decoding of product code (Hoeholdt-Justesen).

***Singleton defect $s(C) = n + 1 - k - d$

$s(C) \geq 0$ and equality holds if and only if $C$ is MDS.
$s(C) = 0$ if and only if $s(C^\perp) = 0$.
Example where $s(C) = 1$ and $s(C^\perp) > 1$.
Almost MDS and near MDS.
Genus $g = \max\{s(C), s(C^\perp)\}$ in 4.1. If $k \geq 2$, then $d \leq q(s + 1)$. If $k \geq 3$ and $d = q(s + 1)$, then $s + 1 \leq q$.
Faldum-Willems, de Boer, Dodunekov-Langev, relation with Griesmer bound.

***Incidence structures and geometric codes
Chapter 5

Codes and related structures

Relinde Jurrius and Ruud Pellikaan

***In this chapter seemingly unrelated topics are discussed.***

5.1 Graphs and codes

5.1.1 Colorings of a graph

Graph theory is regarded to start with the paper of Euler [57] with his solution of the problem of the Königberg bridges. For an introduction to the theory of graphs we refer to [14, 136].

**Definition 5.1.1** A graph \( \Gamma \) is a pair \((V, E)\) where \( V \) is a non-empty set and \( E \) is a set disjoint from \( V \). The elements of \( V \) are called vertices, and members of \( E \) are called edges. Edges are incident to one or two vertices, which are called the ends of the edge. If an edge is incident with exactly one vertex, then it is called a loop. If \( u \) and \( v \) are vertices that are incident with an edge, then they are called neighbors or adjacent. Two edges are called parallel if they are incident with the same vertices. The graph is called simple if it has no loops and no parallel edges.

![A planar graph](image.png)

Figure 5.1: A planar graph
**Definition 5.1.2** A graph is called **planar** if there is an injective map \( f : V \rightarrow \mathbb{R}^2 \) from the set of vertices \( V \) to the real plane such that for every edge \( e \) with ends \( u \) and \( v \) there is a simple curve in the plane connecting the ends of the edge such that mutually distinct simple curves do not intersect except at the endpoints. More formally: for every edge \( e \) with ends \( u \) and \( v \) there is an injective continuous map \( g_e : [0, 1] \rightarrow \mathbb{R}^2 \) from the unit interval to the plane such that \( \{ f(u), f(v) \} = \{ g_e(0), g_e(1) \} \), and \( g_e(0, 1) \cap g_{e'}(0, 1) = \emptyset \) for all edges \( e, e' \) with \( e \neq e' \).

**Example 5.1.3** Consider the next riddle:

Three new-build houses have to be connected to the three nearest terminals for gas, water and electricity. For security reasons, the connections are not allowed to cross. How can this be done?

The answer is “not”, because the corresponding graph (see Figure 5.3) is not planar. This riddle is very suitable to occupy kids who like puzzles, but make sure to have an easy explainable proof of the improbability. We leave it to the reader to find one.

**Definition 5.1.4** Let \( \Gamma_1 = (V_1, E_1) \) and \( \Gamma_2 = (V_2, E_2) \) be graphs. A map \( \phi : V_1 \rightarrow V_2 \) is called a **morphism of graphs** if \( \phi(v) \) and \( \phi(w) \) are connected in \( \Gamma_2 \) for all \( v, w \in V_1 \) that are connected in \( \Gamma_1 \). The map is called an **isomorphism of graphs** if it is a morphism of graphs and there exists a map \( \psi : V_2 \rightarrow V_1 \) such that it is a morphism of graphs and it is the inverse of \( \phi \). The graphs are called **isomorphic** if there is an isomorphism of graphs between them.

**Definition 5.1.5** An edge of a graph is called an **isthmus** if the number of components of the graph increases by deleting the edge. If the graph is connected, then deleting an isthmus gives a graph that is no longer connected. Therefore an isthmus is also called a **bridge**. An edge is an isthmus if and only if it is in no cycle. Therefore an edge that is an isthmus is also called an **acyclic** edge.

**Remark 5.1.6** By deleting loops and parallel edges from a graph \( \Gamma \) one gets a simple graph. There is a choice in the process of deleting parallel edges, but the resulting graphs are all isomorphic. We call this simple graph the **simplification** of the graph and it is denoted by \( \overline{\Gamma} \).

**Definition 5.1.7** Let \( \Gamma = (V, E) \) be a graph. Let \( K \) be a finite set and \( k = |K| \). The elements of \( K \) are called **colors**. A **\( k \)-coloring** of \( \Gamma \) is a map \( \gamma : V \rightarrow K \) such that \( \gamma(u) \neq \gamma(v) \) for all distinct adjacent vertices \( u \) and \( v \) in \( V \). So vertex \( u \) has color \( \gamma(u) \) and all other adjacent vertices have a color distinct from \( \gamma(u) \). Let \( P_\Gamma(k) \) be the number of \( k \)-colorings of \( \Gamma \). Then \( P_\Gamma \) is called the **chromatic polynomial** of \( \Gamma \).

**Remark 5.1.8** If the graph \( \Gamma \) has no edges, then \( P_\Gamma(k) = k^v \) where \( |V| = v \) and \( |K| = k \), since it is equal to the number of all maps from \( V \) to \( K \). In particular there is no map from \( V \) to an empty set in case \( V \) is nonempty. So the number of 0-colorings is zero for every graph.

The number of colorings of graphs was studied by Birkhoff [16], Whitney [130, 129] and Tutte [121, 124, 125, 126, 127]. Much research on the chromatic polynomial was motivated by the four-color problem of planar graphs.
Example 5.1.9 Let $K_n$ be the complete graph on $n$ vertices in which every pair of two distinct vertices is connected by exactly one edge. Then there is no $k$ coloring if $k < n$. Now let $k \geq n$. Take an enumeration of the vertices. Then there are $k$ possible choices of a color of the first vertex and $k-1$ choices for the second vertex, since the first and second vertex are connected. Now suppose by induction that we have a coloring of the first $i$ vertices, then there are $k-i$ possibilities to color the next vertex, since the $(i+1)$-th vertex is connected to the first $i$ vertices. Hence

$$P_{K_n}(k) = k(k-1) \cdots (k-n+1)$$

So $P_{K_n}(k)$ is a polynomial in $k$ of degree $n$.

**Proposition 5.1.10** Let $\Gamma = (V,E)$ be a graph. Then $P_\Gamma(k)$ is a polynomial in $k$.

**Proof.** See[16]. Let $\gamma : V \to K$ be a $k$-coloring of $\Gamma$ with exactly $i$ colors. Let $\sigma$ be a permutation of $K$. Then the composition of maps $\sigma \circ \gamma$ is also $k$-coloring of $\Gamma$ with exactly $i$ colors. Two such colorings are called equivalent. Then $k(k-1) \cdots (k-i+1)$ is the number of colorings in the equivalence class of a given $k$-coloring of $\Gamma$ with exactly $i$ colors. Let $m_i$ be the number of equivalence classes of colorings with exactly $i$ colors of the set $K$. Let $v = |V|$. Then $P_\Gamma(k)$ is equal to

$$m_1k + m_2k(k-1) + \ldots + m_{i}k(k-1) \cdots (k-i+1) + \ldots + m_vk(k-1) \cdots (k-v+1).$$

Therefore $P_\Gamma(k)$ is a polynomial in $k$. \hfill \Box

**Definition 5.1.11** A graph $\Gamma = (V,E)$ is called bipartite if $V$ is the disjoint union of two nonempty sets $M$ and $N$ such that the ends of an edge are in $M$ and in $N$. Hence no two points in $M$ are adjacent and no two points in $N$ are adjacent. Let $m$ and $n$ be integers such that $1 \leq m \leq n$. The complete bipartite graph $K_{m,n}$ is the graph on a set of vertices $V$ that is the disjoint union of two sets $M$ and $N$ with $|M| = m$ and $|N| = n$, and such that every vertex in $M$ is connected with every vertex in $N$ by a unique edge.

Another tool to show that $P_\Gamma(k)$ is a polynomial this by deletion-contraction of graphs, a process which is similar to the puncturing and shortening of codes from Section ??.
Definition 5.1.12 Let $\Gamma = (V, E)$ be a graph. Let $e$ be an edge that is incident to the vertices $u$ and $v$. Then the deletion $\Gamma \setminus e$ is the graph with vertices $V$ and edges $E \setminus \{e\}$. The contraction $\Gamma/e$ is the graph obtained by identifying $u$ and $v$ and deleting $e$. Formally this is defined as follows. Let $\tilde{u} = \tilde{v} = \{u,v\}$, and $\tilde{w} = \{w\}$ if $w \neq u$ and $w \neq v$. Let $\tilde{V} = \{\tilde{w} : w \in V\}$. Then $\Gamma/e$ is the graph $(\tilde{V}, \tilde{E} \setminus \{e\})$, where an edge $f \neq e$ is incident with $\tilde{w}$ in $\Gamma/e$ if $f$ is incident with $w$ in $\Gamma$.

Remark 5.1.13 Notice that the number of $k$-colorings of $\Gamma$ does not change by deleting loops and a parallel edge. Hence the chromatic polynomial $\Gamma$ and its simplification $\tilde{\Gamma}$ are the same.

The following proposition is due to Foster. See the concluding note in [129].

Proposition 5.1.14 Let $\Gamma = (V, E)$ be a simple graph. Let $e$ be an edge of $\Gamma$.

Then the following deletion-contraction formula holds:

$$P_\Gamma(k) = P_{\Gamma \setminus e}(k) - P_{\Gamma/e}(k)$$

for all positive integers $k$.

Proof. Let $u$ and $v$ be the vertices of $e$. Then $u \neq v$, since the graph is simple. Let $\gamma$ be a $k$-coloring of $\Gamma \setminus e$. Then $\gamma$ is also a coloring of $\Gamma$ if and only if $\gamma(u) \neq \gamma(v)$. If $\gamma(u) = \gamma(v)$, then consider the induced map $\tilde{\gamma}$ on $\tilde{V}$ defined by $\tilde{\gamma}(\tilde{u}) = \gamma(u)$ and $\tilde{\gamma}(\tilde{w}) = \gamma(w)$ if $w \neq u$ and $w \neq v$. The map $\tilde{\gamma}$ gives a $k$-coloring of $\Gamma/e$. Conversely, every $k$-coloring of $\Gamma/e$ gives a $k$-coloring $\gamma$ of $\Gamma \setminus e$ such that $\gamma(v) = \gamma(w)$. Therefore

$$P_{\Gamma \setminus e}(k) = P_{\Gamma}(k) + P_{\Gamma/e}(k).$$

This follows also from a more general deletion-contraction formula for matroids that will be treated in Section 5.2.6 and Proposition ??.

5.1.2 Codes on graphs

Definition 5.1.15 Let $\Gamma = (V, E)$ be a graph. Suppose that $V' \subseteq V$ and $E' \subseteq E$ and all the endpoints of $e'$ in $E'$ are in $V'$. Then $\Gamma' = (V', E')$ is a graph and it is called a subgraph of $\Gamma$.

Definition 5.1.16 Two vertices $u$ to $v$ are connected by a path from $u$ to $v$ if there is a $t$-tuple of mutually distinct vertices $(v_1, \ldots, v_t)$ with $u = v_1$ and $v = v_t$, and a $(t-1)$-tuple of mutually distinct edges $(e_1, \ldots, e_{t-1})$ such that $e_i$ is incident with $v_i$ and $v_{i+1}$ for all $1 \leq i < t$. If moreover $e_i$ is an edge that is incident with $u$ and $v$ and distinct from $e_i$ for all $i < t$, then $(e_1, \ldots, e_{t-1}, e_t)$ is called a cycle. The length of the smallest cycle is called the girth of the graph and is denoted by $\gamma(\Gamma)$. 
5.1. GRAPHS AND CODES

Definition 5.1.17 The graph is called connected if every two vertices are connected by a path. A maximal connected subgraph of $\Gamma$ is called a connected component of $\Gamma$. The vertex set $V$ of $\Gamma$ is a disjoint union of subsets $V_i$ and set of edges $E$ is a disjoint union of subsets $E_i$ such that $\Gamma_i = (V_i, E_i)$ is a connected component of $\Gamma$. The number of connected components of $\Gamma$ is denoted by $c(\Gamma)$.

Definition 5.1.18 Let $\Gamma = (V, E)$ be a finite graph. Suppose that $V$ consists of $m$ elements enumerated by $v_1, \ldots, v_m$. Suppose that $E$ consists of $n$ elements enumerated by $e_1, \ldots, e_n$. The incidence matrix $I(\Gamma)$ is a $m \times n$ matrix with entries $a_{ij}$ defined by

$$a_{ij} = \begin{cases} 1 & \text{if } e_j \text{ is incident with } v_i \text{ and } v_k \text{ for some } i < k, \\ -1 & \text{if } e_j \text{ is incident with } v_i \text{ and } v_k \text{ for some } i > k, \\ 0 & \text{otherwise.} \end{cases}$$

Suppose moreover that $\Gamma$ is simple. Then $A_\Gamma$ is the arrangement $(H_1, \ldots, H_n)$ of hyperplanes where $H_j = X_i - X_k$ if $e_j$ is incident with $v_i$ and $v_k$ with $i < k$. An arrangement $A$ is called graphic if $A$ is isomorphic with $A_\Gamma$ for some graph $\Gamma$.

***characteristic polynomial $\det(A - \lambda I)$, Matrix tree theorem

Definition 5.1.19 The graph code of $\Gamma$ over $\mathbb{F}_q$ is the $\mathbb{F}_q$-linear code that is generated by the rows of the incidence matrix $I(\Gamma)$. The cycle code $C_\Gamma$ of $\Gamma$ is the dual of the graph code of $\Gamma$.

Remark 5.1.20 Let $\Gamma$ be a finite graph without loops. Then the arrangement $A_\Gamma$ is isomorphic with $A_{C_\Gamma}$.

Proposition 5.1.21 Let $\Gamma$ be a finite graph. Then $C_\Gamma$ is a code with parameters $[n, k, d]$, where $n = |E|$, $k = |E| - |V| + c(\Gamma)$ and $d = \gamma(\Gamma)$.

Proof. See [14, Prop. 4.3] ⋄

***Sparse graph codes, Gallager or Low-density parity check codes and Tanner graph codes play an important role in the research of coding theory at this moment. See [77, 99].

5.1.3 Exercises

5.1.1 Determine the chromatic polynomial of the bipartite graph $K_{3,2}$.

5.1.2 Determine the parameters of the cycle code of the complete graph $K_m$. Show that the code $C_{K_4}$ over $\mathbb{F}_2$ is equivalent to the punctured binary $[7, 3, 4]$ simplex code.

5.1.3 Determine the parameters of the cycle code of the bipartite graph $K_{m,n}$. Let $C(m)$ be the dual of the $m$-fold repetition code. Show that $C_{K_{m,n}}$ is equivalent to the product code $C(m) \otimes C(n)$. 

5.2 Matroids and codes

Matroids were introduced by Whitney [130, 131] in axiomatizing and generalizing the concepts of independence in linear algebra and circuit in graph theory. In the theory of arrangements one uses the notion of a geometric lattice. In graph and coding theory one refers more to matroids.

5.2.1 Matroids

Definition 5.2.1 A matroid $M$ is a pair $(E, \mathcal{I})$ consisting of a finite set $E$ and a collection $\mathcal{I}$ of subsets of $E$ such that the following three conditions hold.

(I.1) $\emptyset \in \mathcal{I}$.

(I.2) If $J \subseteq I$ and $I \in \mathcal{I}$, then $J \in \mathcal{I}$.

(I.3) If $I, J \in \mathcal{I}$ and $|I| < |J|$, then there exists a $j \in (J \setminus I)$ such that $I \cup \{j\} \in \mathcal{I}$.

A subset $I$ of $E$ is called independent if $I \in \mathcal{I}$, otherwise it is called dependent. Condition (I.2) is called the independence augmentation axiom.

Remark 5.2.2 If $J$ is a subset of $E$, then $J$ has a maximal independent subset, that is there exists an $J \in \mathcal{I}$ such that $I \subseteq J$ and $I$ is maximal with respect to this property and the inclusion. If $I_1$ and $I_2$ are maximal independent subsets of $J$, then $|I_1| = |I_2|$ by condition (I.3). The rank or dimension of a subset $J$ of $E$ is the number of elements of a maximal independent subset of $J$. An independent set of rank $r(M)$ is called a basis of $M$. The collection of all bases of $M$ is denoted by $\mathcal{B}$.

Example 5.2.3 Let $n$ and $k$ be non-negative integers such that $k \leq n$. Let $U_{n,k}$ be a set consisting of $n$ elements and $\mathcal{I}_{n,k} = \{I \subseteq U_{n,k} | |I| \leq k\}$. Then $(U_{n,k}, \mathcal{I}_{n,k})$ is a matroid and called the uniform matroid of rank $k$ on $n$ elements. A subset $B$ of $U_{n,k}$ is a basis if and only if $|B| = k$. The matroid $U_{n,n}$ has no dependent sets and is called free.

Definition 5.2.4 Let $(E, \mathcal{I})$ be a matroid. An element $x$ in $E$ is called a loop if $\{x\}$ is a dependent set. Let $x$ and $y$ in $E$ be two distinct elements that are not loops. Then $x$ and $y$ are called parallel if $r(\{x, y\}) = 1$. The matroid is called simple if it has no loops and no parallel elements. Now $U_{n,r}$ is the only simple matroid of rank $r$ if $r \leq 2$.

Remark 5.2.5 Let $G$ be a $k \times n$ matrix with entries in a field $\mathbb{F}$. Let $E$ be the set $[n]$ indexing the columns of $G$ and $\mathcal{I}_G$ be the collection of all subsets $I$ of $E$ such that the submatrix $G_I$ consisting of the columns of $G$ at the positions of $I$ are independent. Then $M_G = (E, \mathcal{I}_G)$ is a matroid. Suppose that $\mathbb{F}$ is a finite field and $G_1$ and $G_2$ are generator matrices of a code $C$, then $(E, \mathcal{I}_{G_1}) = (E, \mathcal{I}_{G_2})$. So the matroid $M_C = (E, \mathcal{I}_C)$ of a code $C$ is well defined by $(E, \mathcal{I}_G)$ for some generator matrix $G$ of $C$. If $C$ is degenerate, then there is a position $i$ such that $c_i = 0$ for every codeword $c \in C$ and all such positions correspond one-to-one with loops of $M_C$. Let $C$ be nondegenerate. Then $M_C$ has no loops, and the positions $i$ and $j$ with $i \neq j$ are parallel in $M_C$ if and only if the $i$-th column of $G$ is a scalar multiple of the $j$-th column. The code $C$ is projective if and only if the arrangement $\mathcal{A}_G$ is simple if and only if the matroid $M_C$ is simple. A $[n, k]$ code $C$ is MDS if and only if the matroid $M_C$ is the uniform matroid $U_{n,k}$.
5.2. MATROIDS AND CODES

Definition 5.2.6 Let $M = (E, \mathcal{I})$ be a matroid. Let $\mathcal{B}$ be the collection of all bases of $M$. Define $B^\perp = (E \setminus B)$ for $B \in \mathcal{B}$, and $\mathcal{B}^\perp = \{ B^\perp : B \in \mathcal{B} \}$. Define $\mathcal{I}^\perp = \{ I \subseteq E : I \subseteq B \text{ for some } B \in \mathcal{B}^\perp \}$. Then $(E, \mathcal{I}^\perp)$ is called the dual matroid of $M$ and is denoted by $M^\perp$.

Remark 5.2.7 The dual matroid is indeed a matroid. Let $C$ be a code over a finite field. Then $(M_C)^\perp$ is isomorphic with $M_C^\perp$ as matroids.

Let $e$ be a loop of the matroid $M$. Then $e$ is not a member of any basis of $M$. Hence $e$ is in every basis of $M^\perp$. An element of $M$ is called an isthmus if it is an element of every basis of $M$. Hence $e$ is an isthmus of $M$ if and only if $e$ is a loop of $M^\perp$.

Proposition 5.2.8 Let $(E, \mathcal{I})$ be a matroid with rank function $r$. Then the dual matroid has rank function $r^\perp$ given by

$$r^\perp(J) = |J| - r(E) + r(E \setminus J).$$

Proof. The proof is based on the observation that $r(J) = \max_{B \in \mathcal{B}} |B \cap J|$ and $B \setminus J = B \cap (E \setminus J)$.

$$r^\perp(J) = \max_{B \in \mathcal{B}^\perp} |B \cap J|$$
$$= \max_{B \in \mathcal{B}} (|E \setminus B| \cap J|)$$
$$= \max_{B \in \mathcal{B}} |J \setminus B|$$
$$= |J| - \min_{B \in \mathcal{B}} |J \cap B|$$
$$= |J| - (|B| - \max_{B \in \mathcal{B}} |B \setminus J|)$$
$$= |J| - r(E) + \max_{B \in \mathcal{B}} |B \cap (E \setminus J)|$$
$$= |J| - r(E) + r(E \setminus J).$$

5.2.2 Realizable matroids

Definition 5.2.9 Let $M_1 = (E_1, \mathcal{I}_1)$ and $M_2 = (E_2, \mathcal{I}_2)$ be matroids. A map $\varphi : E_1 \to E_2$ is called a morphism of matroids if $\varphi(I) \in \mathcal{I}_2$ for all $I \in \mathcal{I}_1$. The map is called an isomorphism of matroids if it is a morphism of matroids and there exists a map $\psi : E_2 \to E_1$ such that it is a morphism of matroids and it is the inverse of $\varphi$. The matroids are called isomorphic if there is an isomorphism of matroids between them.

Remark 5.2.10 A matroid $M$ is called realizable or representable over the field $\mathbb{F}$ if there exists a matrix $G$ with entries in $\mathbb{F}$ such that $M$ is isomorphic with $M_G$.

**six points in a plane is realizable over every field?**

**The Fano plane is realizable over $\mathbb{F}$ if and only if $\mathbb{F}$ has characteristic two.**

**Pappos, Desargues configuration.**
For more on representable matroids we refer to Tutte [123] and Whittle [132, 133]. Let \( g_n \) be the number of simple matroids on \( n \) points. The values of \( g_n \) are determined for \( n \leq 8 \) by [18] and are given in the following table:

<table>
<thead>
<tr>
<th>( n )</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>( g_n )</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>9</td>
<td>26</td>
<td>101</td>
<td>950</td>
</tr>
</tbody>
</table>

Extended tables can be found in [51]. Clearly \( g_n \leq 2^{2^n} \). Asymptotically the number \( g_n \) is given in [73] and is as follows:

\[
\begin{align*}
  g_n &\leq n - \log_2 n + O(\log_2 \log_2 n), \\
  g_n &\geq n - \frac{3}{2} \log_2 n + O(\log_2 \log_2 n).
\end{align*}
\]

A crude upper bound on the number of \( k \times n \) matrices with \( k \leq n \) and entries in \( \mathbb{F}_q \) is given by \((n+1)q^{n^2}\). Hence the vast majority of all matroids on \( n \) elements is not representable over a given finite field for \( n \to \infty \).

### 5.2.3 Graphs and matroids

**Definition 5.2.11** Let \( M = (E, I) \) be a matroid. A subset \( C \) of \( E \) is called a **circuit** if it is dependent and all its proper subsets are independent. A circuit of the dual matroid of \( M \) is called a **cocircuit** of \( M \).

**Proposition 5.2.12** Let \( C \) be the collection of circuits of a matroid. Then

- \((C.0)\) \( \emptyset \not\in C \).
- \((C.1)\) If \( C_1, C_2 \in C \) and \( C_1 \subseteq C_2 \), then \( C_1 = C_2 \).
- \((C.2)\) If \( C_1, C_2 \in C \) and \( C_1 \neq C_2 \) and \( x \in C_1 \cap C_2 \), then there exists a \( C_3 \in C \) such that \( C_3 \subseteq (C_1 \cup C_2) \setminus \{x\} \).

**Proof.** See [?, Lemma 1.1.3].

Condition \((C.2)\) is called the **circuit elimination axiom**. The converse of Proposition 5.2.12 holds.

**Proposition 5.2.13** Let \( C \) be a collection of subsets of a finite set \( E \) that satisfies the conditions \((C.1)\), \((C.2)\) and \((C.3)\). Let \( I \) be the collection of all subsets of \( E \) that contain no member of \( C \). Then \((E, I)\) is a matroid with \( C \) as its collection of circuits.

**Proof.** See [?, Theorem 1.1.4].

**Proposition 5.2.14** Let \( \Gamma = (V, E) \) be a finite graph. Let \( C \) the collection of all subsets \( \{e_1, \ldots, e_t\} \) such that \( (e_1, \ldots, e_t) \) is a cycle in \( \Gamma \). Then \( C \) is the collection of circuits of a matroid \( M_\Gamma \) on \( E \). This matroid is called the cycle matroid of \( \Gamma \).

**Proof.** See [?, Proposition 1.1.7].
5.2. Matroids and Codes

Remark 5.2.15 Loops in \( \Gamma \) correspond one-to-one to loops in \( M_\Gamma \). Two edges that are no loops, are parallel in \( \Gamma \) if and only if they are parallel in \( M_\Gamma \). So \( \Gamma \) is simple if and only if \( M_\Gamma \) is simple. Let \( e \) in \( E \). Then \( e \) is an isthmus in the graph \( \Gamma \) if and only is \( e \) is an isthmus in the matroid \( M_\Gamma \).

Remark 5.2.16 A matroid \( M \) is called graphic if \( M \) is isomorphic with \( M_\Gamma \) for some graph \( \Gamma \), and it is called cographic if \( M^\perp \) is graphic. If \( \Gamma \) is a planar graph, then the matroid \( M_\Gamma \) is graphic by definition but it is also cographic.

Let \( \Gamma \) be a finite graph with incidence matrix \( I(\Gamma) \). This is a generator matrix for \( C_\Gamma \) over a field \( F \). Suppose that \( F \) is the binary field. Look at all the columns indexed by the edges of a cycle of \( \Gamma \). Since every vertex in a cycle is incident with exactly two edges, the sum of these columns is zero and therefore they are dependent. Removing a column gives an independent set of vectors. Hence the cycles in the matroid \( M_{C_\Gamma} \) coincide with the cycles in \( \Gamma \). Therefore \( M_\Gamma \) is isomorphic with \( M_{C_\Gamma} \). One can generalize this argument for any field. Hence graphic matroids are representable over any field.

The matroids of the binary Hamming \([7,4,3]\) code is not graphic and not cographic. Clearly the matroids \( M_{K_5} \) and \( M_{K_3,3} \) are graphic by definition, but they are not cographic. Tutte [122] found a classification for graphic matroids.

5.2.4 Tutte and Whitney polynomial of a matroid

See [7, 8, 25, 26, 28, 34, 59, 68] for references of this section.

Definition 5.2.17 Let \( M = (E, I) \) be a matroid. Then the Whitney rank generating function \( R_M(X, Y) \) is defined by

\[
R_M(X, Y) = \sum_{J \subseteq E} X^{r(E) - r(J)} Y^{|J| - r(J)}
\]

and the Tutte-Whitney or Tutte polynomial by

\[
t_M(X, Y) = \sum_{J \subseteq E} (X - 1)^{r(E) - r(J)} (Y - 1)^{|J| - r(J)}.
\]

In other words,

\[
t_M(X, Y) = R_M(X - 1, Y - 1).
\]

Remark 5.2.18 Whitney [129] defined the coefficients \( m_{ij} \) of the polynomial \( R_M(X, Y) \) such that

\[
R_M(X, Y) = \sum_{i=0}^{r(M)} \sum_{j=0}^{|M|-r(M)} m_{ij} X^i Y^j,
\]

but he did not define the polynomial \( R_M(X, Y) \) as such. It is clear that these coefficients are nonnegative, since they count the number of elements of certain sets. The coefficients of the Tutte polynomial are also nonnegative, but this is not a trivial fact, it follows from the counting of certain internal and external bases of a matroid. See [56].
5.2.5 Weight enumerator and Tutte polynomial

As we have seen, we can interpret a linear \([n,k]\) code \(C\) over \(\mathbb{F}_q\) as a matroid via the columns of a generator matrix \(G\).

Proposition 5.2.19 Let \(C\) be a \([n,k]\) code over \(\mathbb{F}_q\). Then the Tutte polynomial \(t_C\) associated with the matroid \(M_C\) of the code \(C\) is

\[
t_C(X,Y) = \sum_{t=0}^{n} \sum_{|J|=t} (X - 1)^{l(J)}(Y - 1)^{l(J) - (k-t)} .
\]

Proof. This follows from \(l(J) = k - r(J)\) by Lemma 4.4.12 and \(r(M) = k\). \(\diamondsuit\)

This formula and Proposition 4.4.41 suggest the next connection between the weight enumerator and the Tutte polynomial. Greene [59] was the first to notice this connection.

Theorem 5.2.20 Let \(C\) be a \([n,k]\) code over \(\mathbb{F}_q\) with generator matrix \(G\). Then the following holds for the Tutte polynomial and the extended weight enumerator:

\[
W_C(X,Y,T) = (X - Y)^k Y^{n-k} t_C\left(\frac{X + (T - 1)Y}{X - Y}, \frac{X}{Y}\right) .
\]

Proof. By using Proposition 5.2.19 about the Tutte polynomial, rewriting, and Proposition 4.4.41 we get

\[
\begin{align*}
(X - Y)^k Y^{n-k} t_C\left(\frac{X + (T - 1)Y}{X - Y}, \frac{X}{Y}\right) &= (X - Y)^k Y^{n-k} \sum_{t=0}^{n} \sum_{|J|=t} \left(\frac{TY}{X-Y}\right)^{l(J)} \left(\frac{X - Y}{Y}\right)^{l(J) - (k-t)} \\
&= (X - Y)^k Y^{n-k} \sum_{t=0}^{n} \sum_{|J|=t} T^{l(J)} Y^{k-t} (X - Y)^{- (k-t)} \\
&= \sum_{t=0}^{n} \sum_{|J|=t} T^{l(J)} (X - Y)^t Y^{n-t} \\
&= W_C(X, Y, T) .
\end{align*}
\]

We use the extended weight enumerator here, because extending a code does not change the generator matrix and therefore not the matroid \(G\). The converse of this theorem is also true: the Tutte polynomial is completely defined by the extended weight enumerator.

Theorem 5.2.21 Let \(C\) be a \([n,k]\) code over \(\mathbb{F}_q\). Then the following holds for the extended weight enumerator and the Tutte polynomial:

\[
t_C(X,Y) = Y^n (Y - 1)^{-k} W_C(1, Y^{-1}, (X - 1)(Y - 1)) .
\]
Proof. The proof of this theorem goes analogous to the proof of the previous theorem.

\[
Y^n (Y - 1)^{-k} W_C (1, Y^{-1}, (X - 1)(Y - 1)) = Y^n (Y - 1)^{-k} \sum_{t=0}^{n} \sum_{|J|=t} ((X - 1)(Y - 1))^{|J|} (1 - Y^{-1})^t Y^{-(n-t)} = \sum_{t=0}^{n} \sum_{|J|=t} (X - 1)^{|J|} (Y - 1)^{|J|} Y^{-t}(Y - 1)^{t} Y^{-(n-k)} Y^n (Y - 1)^{-k} = \sum_{t=0}^{n} \sum_{|J|=t} (X - 1)^{|J|} (Y - 1)^{|J|} (k-t) = t_C (X,Y)
\]

We see that the Tutte polynomial depends on two variables, while the extended weight enumerator depends on three variables. This is no problem, because the weight enumerator is given in its homogeneous form here: we can view the extended weight enumerator as a polynomial in two variables via \(W_C (Z,T) = W_C (1, Z, T)\).

Greene [59] already showed that the Tutte polynomial determines the weight enumerator, but not the other way round. By using the extended weight enumerator, we get a two-way equivalence and the proof reduces to rewriting.

We can also give expressions for the generalized weight enumerator in terms of the Tutte polynomial, and the other way round. The first formula was found by Britz [28] and independently by Jurrius [68].

**Theorem 5.2.22** For the generalized weight enumerator of a \([n,k]\) code \(C\) and the associated Tutte polynomial we have that \(W_C^{(r)} (X,Y)\) is equal to

\[
\frac{1}{(r)!} \sum_{j=0}^{r} r_j (-1)^{r-j} q^j (X - Y)^k Y^{n-k} \ t_C \left( \frac{X + (q^j - 1)Y}{X - Y} \right) \left( \frac{X}{Y} \right).
\]

And, conversely,

\[
t_C (X,Y) = Y^n (Y - 1)^{-k} \sum_{r=0}^{k} \left( \prod_{j=0}^{r-1} ((X - 1)(Y - 1) - q^j) \right) W_C^{(r)} (1, Y^{-1})
\]

Proof. For the first formula, use Theorems 4.5.23 and 5.2.20. Use Theorems 4.5.21 and 5.2.21 for the second formula.

\[\diamond\]

### 5.2.6 Deletion and contraction of matroids

**Definition 5.2.23** Let \(M = (E, I)\) be a matroid of rank \(k\). Let \(e\) be an element of \(E\). Then the deletion \(M \setminus e\) is the matroid on the set \(E \setminus \{e\}\) with independent sets of the form \(I \setminus \{e\}\) where \(I\) is independent in \(M\). The contraction \(M/e\) is the matroid on the set \(E \setminus \{e\}\) with independent sets of the form \(I \setminus \{e\}\) where \(I\) is independent in \(M\) and \(e \in I\).
Remark 5.2.24 Let $M$ be a graphic matroid. So $M = M_\Gamma$ for some finite graph $\Gamma$. Let $e$ be an edge of $\Gamma$, then $M \setminus e = M_{\Gamma \setminus e}$ and $M/e = M_{\Gamma/e}$.

Remark 5.2.25 Let $C$ be a code with reduced generator matrix $G$ at position $e$. So $a = (1,0,\ldots,0)^T$ is the column of $G$ at position $e$. Then $M \setminus e = M_{G \setminus a}$ and $M/e = M_{G/a}$. A puncturing-shortening formula for the extended weight enumerator is given in Proposition 4.4.44. By virtue of the fact that the extended weight enumerator and the Tutte polynomial of a code determine each other by the Theorems 5.2.20 and 5.2.21, one expects that an analogous generalization for the Tutte polynomial of matroids holds.

Proposition 5.2.26 Let $M = (E,I)$ be a matroid. Let $e \in E$ that is not a loop and not an isthmus. Then the following deletion-contraction formula holds:

\[
t_M(X,Y) = t_{M \setminus e}(X,Y) + t_{M/e}(X,Y).
\]

Proof. See [119, 120, 125, 31].

5.2.7 McWilliams type property for duality

For both codes and matroids we defined the dual structure. These objects obviously completely define there dual. But how about the various polynomials associated to a code and a matroid? We know from Example 4.5.17 that the weight enumerator is a less strong invariant for a code then the code itself: this means there are non-equivalent codes with the same weight enumerator. So it is a priori not clear that the weight enumerator of a code completely defines the weight enumerator of its dual code. We already saw that there is in fact such a relation, namely the MacWilliams identity in Theorem 4.1.22. We will give a proof of this relation by considering the more general question for the extended weight enumerator. We will prove the MacWilliams identities using the Tutte polynomial. We do this because of the following simple and very useful relation between the Tutte polynomial of a matroid and its dual.

Theorem 5.2.27 Let $t_M(X,Y)$ be the Tutte polynomial of a matroid $M$, and let $M^\perp$ be the dual matroid. Then

\[
t_M(X,Y) = t_{M^\perp}(Y,X).
\]

Proof. Let $M$ be a matroid on the set $E$. Then $M^\perp$ is a matroid on the same set. In Proposition 5.2.8 we proved $r^\perp(J) = |J| - r(E) + r(E \setminus J)$. In particular, we have $r^\perp(E) + r(E) = |E|$. Substituting this relation into the definition of the Tutte polynomial for the dual code, gives

\[
t_{M^\perp}(X,Y) = \sum_{J \subseteq E} (X-1)^{r^\perp(E)-r^\perp(J)}(Y-1)^{|J|-r^\perp(J)}
\]

\[
= \sum_{J \subseteq E} (X-1)^{r^\perp(E)-|J|+r(E\setminus J)+r(E)}(Y-1)^{r(E)-r(E\setminus J)}
\]

\[
= \sum_{J \subseteq E} (X-1)^{|E\setminus J|-r(E\setminus J)}(Y-1)^{r(E)-r(E\setminus J)}
\]

\[
= t_M(Y,X)
\]
In the last step, we use that the summation over all $J \subseteq E$ is the same as a summation over all $E \setminus J \subseteq E$. This proves the theorem.

If we consider a code as a matroid, then the dual matroid is the dual code. Therefore we can use the above theorem to prove the MacWilliams relations. Greene[59] was the first to use this idea, see also Brylawsky and Oxley[33].

**Theorem 5.2.28 (MacWilliams)** Let $C$ be a code and let $C^\perp$ be its dual. Then the extended weight enumerator of $C$ completely determines the extended weight enumerator of $C^\perp$ and vice versa, via the following formula:

$$W_{C^\perp}(X,Y,T) = T^{-k}W_C(X + (T - 1)Y, X - Y, T).$$

**Proof.** Let $G$ be the matroid associated to the code. Using the previous theorem and the relation between the weight enumerator and the Tutte polynomial, we find

$$
\begin{align*}
T^{-k}W_C(X + (T - 1)Y, X - Y, T) &= T^{-k}(TY)^k(X - Y)^{n-k} t_C \left( \frac{X}{Y}, \frac{X + (T - 1)Y}{X - Y} \right) \\
&= Y^k(X - Y)^{n-k} t_{C^\perp} \left( \frac{X + (T - 1)Y}{X - Y}, \frac{X}{Y} \right) \\
&= W_{C^\perp}(X, Y, T).
\end{align*}
$$

Notice in the last step that $\dim C^\perp = n - k$, and $n - (n - k) = k$.

We can use the relations in Theorems 4.5.21 and 4.5.23 to prove the MacWilliams identities for the generalized weight enumerator.

**Theorem 5.2.29** Let $C$ be a code and let $C^\perp$ be its dual. Then the generalized weight enumerators of $C$ completely determine the generalized weight enumerators of $C^\perp$ and vice versa, via the following formula:

$$W^{(r)}_{C^\perp}(X,Y) = \sum_{j=0}^{r} \sum_{l=0}^{j} (-1)^{j-l} q^{r-j-l} \left( \frac{T^{r-j}}{q} \right) \frac{W^{(l)}_C(X + (q^j - 1)Y, X - Y)}{r-j}\binom{r-j}{l}_q.$$ 

**Proof.** We write the generalized weight enumerator in terms of the extended weight enumerator, use the MacWilliams identities for the extended weight enum-
merator, and convert back to the generalized weight enumerator.

\[
W_C^{(1)}(X,Y) = \frac{1}{(r)!} \sum_{j=0}^{r} \binom{r}{j} q^{\binom{r-j}{2}} W_{C^\perp}(X,Y,q^j)
\]

\[
= \sum_{j=0}^{r} \frac{(-1)^{r-j} q^{\binom{r-j}{2}}}{\binom{r-j}{2}} \frac{q^{-j} \sum_{i=0}^{r-j} \binom{r-j}{i} q^{-i} \binom{r-j-i}{2}}{q^{r-j} \binom{r-j}{2}} W_{C}(X+(q^j-1)Y,Y,q^j)
\]

\[
= \sum_{j=0}^{r} \frac{(-1)^{r-j} q^{\binom{r-j}{2}}}{\binom{r-j}{2}} \frac{q^{-j} \sum_{i=0}^{r-j} \binom{r-j}{i} q^{-i} \binom{r-j-i}{2}}{q^{r-j} \binom{r-j}{2}} \times W_{C}(X+(q^j-1)Y,X-Y)
\]

This theorem was proved by Kløve\[72\], although the proof uses only half of the relations between the generalized weight enumerator and the extended weight enumerator. Using both makes the proof much shorter.

5.2.8 Exercises

5.2.1 Give a proof of the statements in Remark 5.2.2.

5.2.2 Give a proof of the statements in Remark 5.2.7.

5.2.3 Show that all matroids on at most 3 elements are graphic. Give an example of a matroid that is not graphic.

5.3 Geometric lattices and codes

5.3.1 Posets, the Möbius function and lattices

**Definition 5.3.1** Let \( L \) be a set and \( \leq \) a relation on \( L \) such that:

- (PO.1) \( x \leq x \), for all \( x \) in \( L \) (reflexive),
- (PO.2) if \( x \leq y \) and \( y \leq x \), then \( x = y \), for all \( x, y \) in \( L \) (anti-symmetric),
- (PO.3) if \( x \leq y \) and \( y \leq z \), then \( x \leq z \), for all \( x, y \) and \( z \) in \( L \) (transitive).

Then the pair \( (L, \leq) \) or just \( L \) is called a poset with partial order \( \leq \) on the set \( L \). Define \( x < y \) if \( x \leq y \) and \( x \neq y \). The elements \( x \) and \( y \) in \( L \) are comparable if \( x \leq y \) or \( y \leq x \). A poset \( L \) is called a linear order if every two elements are comparable. Define \( L_x = \{ y \in L | x \leq y \} \) and \( L^y = \{ y \in L | y \leq x \} \) and the the interval between \( x \) and \( y \) by \([x,y] = \{ z \in L | x \leq z \leq y \}\). Notice that \([x,y] = L_x \cap L^y\).
**Definition 5.3.2** Let \((L, \leq)\) be a poset. A *chain of length* \(r\) *from* \(x\) *to* \(y\) *in* \(L\) *is a sequence of elements* \(x_0, x_1, \ldots, x_r\) *in* \(L\) *such that*

\[
x = x_0 < x_1 < \cdots < x_r = y.
\]

Let \(r\) be a number. Let \(x, y\) in \(L\). Then \(c_r(x, y)\) denotes the number of chains of length \(r\) from \(x\) to \(y\). Now \(c_r(x, y)\) is finite if \(L\) is finite. The poset is called *locally finite* if \(c_r(x, y)\) is finite for all \(x, y \in L\) and every number \(r\).

**Proposition 5.3.3** Let \(L\) be a locally finite poset. Let \(x \leq y\) in \(L\). Then

(C.0) \(c_0(x, y) = 0\) if \(x\) and \(y\) are not comparable.

(C.1) \(c_0(x, x) = 1\), \(c_r(x, x) = 0\) for all \(r > 0\) and \(c_0(x, y) = 0\) if \(x < y\).

(C.2) \(c_{r+1}(x, y) = \sum_{x \leq z < y} c_r(x, z) = \sum_{x < z \leq y} c_r(z, y)\).

**Proof.** Statements (C.0) and (C.1) are trivial. Let \(z < y\) and \(x = x_0 < x_1 < \cdots < x_r = z\) a chain of length \(r\) from \(x\) to \(z\), then \(x = x_0 < x_1 < \cdots < x_r < x_{r+1} = y\) is a chain of length \(r+1\) from \(x\) to \(y\), and every chain of length \(r+1\) from \(x\) to \(y\) is obtained uniquely in this way. Hence \(c_{r+1}(x, y) = \sum_{x \leq z < y} c_r(x, z)\). The last equality is proved similarly.

**Definition 5.3.4** The *Möbius function* of \(L\), denoted by \(\mu_L\) or \(\mu\) is defined by

\[
\mu(x, y) = \sum_{r=0}^{\infty} (-1)^r c_r(x, y).
\]

**Proposition 5.3.5** Let \(L\) be a locally finite poset. Then for all \(x, y\) in \(L\):

(M.0) \(\mu(x, y) = 0\) if \(x\) and \(y\) are not comparable.

(M.1) \(\mu(x, x) = 1\).

(M.2) If \(x < y\), then \(\sum_{x \leq z \leq y} \mu(x, z) = \sum_{x \leq z \leq y} \mu(z, y) = 0\).

(M.3) If \(x < y\), then \(\mu(x, y) = -\sum_{x \leq z < y} \mu(x, z) = -\sum_{x < z \leq y} \mu(z, y)\).

**Proof.** (M.0) and (M.1) follow from (C.0) and (C.1), respectively of Proposition 5.3.3. (M.2) is clearly equivalent with (M.3). (M.3) If \(x < y\), then \(c_0(x, y) = 0\). So

\[
\mu(x, y) = \sum_{r=1}^{\infty} (-1)^r c_r(x, y) = \sum_{r=0}^{\infty} (-1)^{r+1} c_{r+1}(x, y) =
\]

\[- \sum_{r=0}^{\infty} (-1)^r \sum_{x \leq z < y} c_r(x, z) = - \sum_{x \leq z < y} \sum_{r=0}^{\infty} (-1)^r c_r(x, z) = - \sum_{x \leq z < y} \mu(x, z).\]

The first and last equality uses the definition of \(\mu\). The second equality starts counting at \(r = 0\) instead of \(r = 1\), the third uses (C.2) of Proposition 5.3.3 and in the fourth the order of summation is interchanged.
Remark 5.3.6 (M.1) and (M.3) of Proposition 5.3.5 can be used as an alternative way to compute $\mu(x,y)$ by induction.

Definition 5.3.7 Let $L$ be a poset. If $L$ has an element $0_L$ such that $0_L$ is the unique minimal element of $L$, then $0_L$ is called the minimum of $L$. Similarly $1_L$ is called the maximum of $L$ if $1_L$ is the unique maximal element of $L$. If $x, y$ in $L$ and $x \leq y$, then the interval $[x,y]$ has $x$ as minimum and $y$ as maximum. Suppose that $L$ has $0_L$ and $1_L$ as minimum and maximum also denoted by 0 and 1, respectively. Then $0 \leq x \leq 1$ for all $x \in L$. Define $\mu(x) = \mu(0,x)$ and $\mu(L) = \mu(0,1)$ if $L$ is finite.

Definition 5.3.8 Let $L$ be a locally finite poset with a minimum element. Let $A$ be an abelian group and $f : L \to A$ a map from $L$ to $A$. The sum function $\hat{f}$ of $f$ is defined by $\hat{f}(x) = \sum_{y \leq x} f(y)$. Define similarly the sum functions $\check{f}$ of $f$ by $\check{f}(x) = \sum_{x \leq y} f(y)$ if $L$ is a locally finite poset with a maximum element.

Remark 5.3.9 A poset $L$ is locally finite if and only if $[x,y]$ is finite for all $x \leq y$ in $L$. So $[0,x]$ is finite if $L$ is a locally finite poset with minimum element 0. Hence the sum function $\hat{f}(x)$ is well-defined, since it is a finite sum of $f(y)$ in $A$ with $y$ in $[0,x]$. In the same way $\check{f}(x)$ is well-defined, since $[x,1]$ is finite.

Theorem 5.3.10 (Möbius inversion formula) Let $L$ be a locally finite poset with a minimum element. Then

$$f(x) = \sum_{y \leq x} \mu(y,x) \hat{f}(y).$$

Similarly $f(x) = \sum_{x \leq y} \mu(x,y) \check{f}(y)$ if $L$ is a locally finite poset with a maximum element.

Proof. Let $x$ be an element of $L$. Then

$$\sum_{y \leq x} \mu(y,x) \hat{f}(y) = \sum_{y \leq x} \sum_{z \leq y} \mu(y,x) f(z) = \sum_{z \leq x} f(z) \sum_{z \leq y \leq x} \mu(y,x) = f(x) \mu(x,x) + \sum_{z < x} f(z) \sum_{z \leq y \leq x} \mu(y,x) = f(x)$$

The first equality uses the definition of $\hat{f}(y)$. In the second equality the order of summation is interchanged. In the third equality the first summation is split in the parts $z = x$ and $z < x$, respectively. Finally $\mu(x,x) = 1$ and the second summation is zero for all $z < x$, by Proposition 5.3.5.

The proof of the second equality is similar. \hfill \diamond

Example 5.3.11 Let $f(x) = 1$ if $x = 0$ and $f(x) = 0$ otherwise. Then the sum function $\hat{f}(x) = \sum_{y \leq x} f(y)$ is constant 1 for all $x$. The Möbius inversion formula gives that

$$\sum_{y \leq x} \mu(x) = \begin{cases} 1 & \text{if } x = 0, \\
0 & \text{if } x > 0,
\end{cases}$$

which is a special case of Proposition 5.3.5.
5.3. GEOMETRIC LATTICES AND CODES

Remark 5.3.12 Let \((L, \leq)\) be a poset. Let \(\leq_R\) be the reverse relation on \(L\) defined by \(x \leq_R y\) if and only if \(y \leq x\). Then \((L, \leq_R)\) is a poset. Suppose that \((L, \leq)\) is locally finite with M"obius function \(\mu\). Then the number of chains of length \(r\) from \(x\) to \(y\) in \((L, \leq_R)\) is the same as the number of chains of length \(r\) from \(y\) to \(x\) in \((L, \leq)\). Hence \((L, \leq_R)\) is locally finite with M"obius function \(\mu_R\) such that \(\mu_R(x, y) = \mu(y, x)\). If \((L, \leq)\) has minimum \(0_L\) or maximum \(1_L\), then \((L, \leq_R)\) has minimum \(0_L\) or maximum \(1_L\), respectively.

Definition 5.3.13 Let \(L\) be a poset. Let \(x, y \in L\). Then \(y\) is called a cover of \(x\) if \(x < y\), and there is no \(z\) such that \(x < z < y\). The Hasse diagram of \(L\) is a directed graph that has the elements of \(L\) as vertices, and there is a directed edge from \(y\) to \(x\) if and only if \(y\) is a cover of \(x\).

Example 5.3.14 Let \(L = \mathbb{Z}\) be the set of integers with the usual linear order. Let \(x, y \in L\) and \(x \leq y\). Then \(c_0(x, x) = 1\), \(c_0(x, y) = 0\) if \(x < y\), and \(c_r(x, y) = (y - x - 1)^r\) for all \(r \geq 1\). So \(L\) infinite and locally finite. Furthermore \(\mu(x, x) = 1\), \(\mu(x, x + 1) = -1\) and \(\mu(x, y) = 0\) if \(y > x + 1\).

Definition 5.3.15 Let \(L\) be a poset. Let \(x, y \in L\). Then \(x\) and \(y\) have a least upper bound if there is a \(z \in L\) such that \(x \leq z\) and \(y \leq z\), and if \(x \leq w\) and \(y \leq w\), then \(z \leq w\) for all \(w \in L\). If \(x\) and \(y\) have a least upper bound, then such an element is unique and it is called the join of \(x\) and \(y\) and denoted by \(x \lor y\). Similarly the greatest lower bound of \(x\) and \(y\) is defined. If it exists, then it is unique and it is called the meet of \(x\) and \(y\) and denoted by \(x \land y\). A poset \(L\) is called a lattice if \(x \lor y\) and \(x \land y\) exist for all \(x, y \in L\).

Remark 5.3.16 Let \((L, \leq)\) be a finite poset with maximum \(1\) such that \(x \land y\) exists for all \(x, y \in L\). The collection \(\{z \mid x \leq z, y \leq z\}\) is finite and not empty, since it contains \(1\). The meet of all the elements in this collection is well defined and is given by

\[
x \lor y = \bigwedge \{z \mid x \leq z, y \leq z\}.
\]

Hence \(L\) is a lattice. Similarly \(L\) is a lattice if \(L\) is a finite poset with minimum \(0\) such that \(x \lor y\) exists for all \(x, y \in L\), since \(x \lor y = \bigvee \{z \mid z \leq x, z \leq y\}\).

Example 5.3.17 Let \(L\) be the collection of all finite subsets of a given set \(X\). Let \(\leq\) be defined by the inclusion, that means \(I \leq J\) if and only if \(I \subseteq J\). Then \(0_L = \emptyset\), and \(L\) has a maximum if and only if \(X\) is finite in which case \(1_L = X\). Let \(I, J \in L\) and \(I \leq J\). Then \(|I| \leq |J| < \infty\). Let \(m = |J| - |I|\). Then

\[
c_r(I, J) = \sum_{m_1 < m_2 < \cdots < m_{r-1} < m} \binom{m}{m_1} \binom{m_1}{m_2} \cdots \binom{m_{r-1}}{m_r}.
\]

Hence \(L\) is locally finite. \(L\) is finite if and only if \(X\) is finite. Furthermore \(I \lor J = I \cup J\) and \(I \land J = I \cap J\). So \(L\) is a lattice. Using Remark 5.3.6 we see that \(\mu(I, J) = (-1)^{|I| - |J|}\) if \(I \leq J\). This is much easier than computing \(\mu(I, J)\) by means of Definition 5.3.4.
Example 5.3.18 Now suppose that $X = \{1, \ldots, n\}$. Let $L$ be the poset of subsets of $X$. Let $A_1, \ldots, A_n$ be a collection of subsets of a finite set $A$. Define for a subset $J$ of $X$

$$A_J = \bigcap_{j \in J} A_j$$

and

$$f(J) = |A_J \setminus \left( \bigcup_{I \in J} A_I \right)|.$$

Then $A_J$ is the disjoint union of the subsets $A_I \setminus \left( \bigcup_{K < I} A_K \right)$ for all $I \leq J$. Hence the sum function

$$\hat{f}(J) = \sum_{I \leq J} f(I) = \sum_{I \leq J} |A_I \setminus \left( \bigcup_{K < I} A_K \right)| = |A_J|.$$

Möbius inversion gives that

$$|A_J \setminus \left( \bigcup_{I \leq J} A_I \right)| = \sum_{I \leq J} (-1)^{|J|-|I|} |A_I|$$

which is called the principle of inclusion/exclusion.

Example 5.3.19 A variant of the principle of inclusion/exclusion is given as follows. Let $H_1, \ldots, H_n$ be a collection of subsets of a finite set $H$. Let $L$ be the poset of all intersections of the $H_j$ with the reverse inclusion as partial order. Then $H$ is the minimum of $L$ and $H_1 \cap \cdots \cap H_n$ is the maximum of $L$. Let $x \in L$. Define

$$f(x) = |x \setminus \left( \bigcup_{y < x} y \right)|.$$

Then

$$\hat{f}(x) = \sum_{x \leq y} f(y) = \sum_{x \leq y} |y \setminus \left( \bigcup_{z < y} z \right)| = |x|.$$

Hence

$$|x \setminus \left( \bigcup_{x \leq y} y \right)| = \sum_{x \leq y} \mu(x, y)|y|.$$

Example 5.3.20 Let $L = \mathbb{N}$ be the set of positive integers with the divisibility relation as partial order. Then $0_L = 1$ is the minimum of $L$, it is locally finite and it has no maximum. Now $m \lor n = \text{lcm}(m, n)$ and $m \land n = \text{gcd}(m, n)$. Hence $L$ is a lattice. By Remark 5.3.6 we see that

$$\mu(n) = \begin{cases} 1 & \text{if } n = 1, \\ (-1)^r & \text{if } n \text{ is the product of } r \text{ mutually distinct primes}, \\ 0 & \text{if } n \text{ is divisible by the square of a prime}. \end{cases}$$

Hence $\mu(n)$ is the classical Möbius function. Furthermore $\mu(d, n) = \mu(d) \mu(n/d)$ if $d|n$. Let

$$\varphi(n) = |\{i \in \mathbb{N} | \gcd(i, n) = 1\}|$$

be Euler’s $\varphi$ function. Define

$$V_d = \{i \in \{1, \ldots, n\} | \gcd(i, n) = d\}$$
for $d|n$. Then
\[
\{ i \in \{1, \ldots, d\} \mid \gcd(i,d) = 1 \} \cdot \frac{n}{d} = V_d.
\]
so $|V_d| = \varphi(d)$. Now $\{1, \ldots, n\}$ is the disjoint union of the subsets $V_d$ with $d|n$.

Hence the sum function of $\varphi(n)$ is given by
\[
\varphi(n) = \sum_{d|n} \varphi(d) = n.
\]

Therefore
\[
\varphi(n) = \sum_{d|n} \mu(d) \frac{n}{d},
\]
by Möbius inversion.

**Definition 5.3.21** Let $(L_1, \leq_1)$ and $(L_2, \leq_2)$ be posets. A map $\varphi : L_1 \to L_2$ is called **monotone** if $\varphi(x) \leq_2 \varphi(y)$ for all $x \leq_1 y$ in $L_1$. The map $\varphi$ is called **strictly monotone** if $\varphi(x) \prec_2 \varphi(y)$ for all $x <_1 y$ in $L_1$. The map is called an **isomorphism of posets** if it is strictly monotone and there exists a strictly monotone map $\psi : L_2 \to L_1$ that is the inverse of $\varphi$. The posets are called **isomorphic** if there is an isomorphism of posets between them.

**Remark 5.3.22** If $\varphi : L_1 \to L_2$ is an isomorphism between locally finite posets with a minimum, then $\mu_2(\varphi(x), \varphi(y)) = \mu_1(x, y)$ for all $x, y$ in $L_1$.

If $(L_1, \leq_1)$ and $(L_2, \leq_2)$ are isomorphic posets and $L_1$ is a lattice, then $L_2$ is also a lattice.

**Example 5.3.23** Let $n$ be a positive integer that is the product of $r$ mutually distinct primes $p_1, \ldots, p_r$. Let $L_1$ be the set of all positive integers that divide $n$ with divisibility as partial order $\leq_1$ as in Example 5.3.20. Let $L_2$ be the collection of all subsets of $\{1, \ldots, r\}$ with the inclusion as partial order $\leq_2$ as in Example 5.3.17. Define the maps $\varphi : L_1 \to L_2$ and $\psi : L_2 \to L_1$ by $\varphi(d) = \{i \mid p_i \text{ divides } n\}$ and $\psi(x) = \prod_{i \in x} p_i$. Then $\varphi$ and $\psi$ are strictly monotone and they are inverses of each other. Hence $L_1$ and $L_2$ are isomorphic lattices.

### 5.3.2 Geometric lattices

**Remark 5.3.24** Let $(L, \leq)$ be a lattice without infinite chains. Then $L$ has a minimum and a maximum.

**Definition 5.3.25** Let $L$ be a lattice with minimum 0. An **atom** is an element $a \in L$ that is a cover of 0. A lattice is called **atomic** if for every $x > 0$ in $L$ there exist atoms $a_1, \ldots, a_r$ such that $x = a_1 \lor \cdots \lor a_r$, and the minimum possible $r$ is called the **rank** of $x$ and is denoted by $r_L(x)$ or $r(x)$ for short. A lattice is called **semimodular** if for all mutually distinct $x, y$ in $L$, $x \lor y$ covers $x$ and $y$ if there exists a $z$ such that $x$ and $y$ cover $z$. A lattice is called **modular** if $x \lor (y \land z) = (x \lor y) \land z$ for all $x, y$ and $z$ in $L$ such that $x \leq z$. A lattice $L$ is called a **geometric lattice** if it is atomic and semimodular and has no infinite chains. If $L$ is a geometric lattice $L$, then it has a minimum and a maximum and $r(1)$ is called the rank of $L$ and is denoted by $r(L)$. 
Example 5.3.26 Let $L$ be the collection of all finite subsets of a given set $X$ as in Example 5.3.17. The atoms are the singleton sets, that is subsets consisting of exactly one element of $X$. Every $x \in L$ is the finite union of its singleton subsets. So $L$ is atomic and $r(x) = |x|$. Now $y$ covers $x$ if and only if there is an element $Q$ not in $x$ such that $y = x \cup \{Q\}$. If $x \neq y$ and $x$ and $y$ both cover $z$, then there is an element $P$ not in $z$ such that $x = z \cup \{P\}$, and there is an element $Q$ not in $z$ such that $y = z \cup \{Q\}$. Now $P \neq Q$, since $x \neq y$. Hence $x \lor y = z \cup \{P, Q\}$ covers $x$ and $y$. Hence $L$ is semimodular. In fact $L$ is modular. $L$ is a geometric lattice if and only if $X$ is finite.

Example 5.3.27 Let $L$ be the set of positive integers with the divisibility relation as in Example 5.3.20. The atoms of $L$ are the primes. But $L$ is not atomic, since a square is not the join of finitely many elements. $L$ is semimodular. The interval $[1, n]$ in $L$ is a geometric lattice if and only if $n$ is square free. If $n$ is square free and $m \leq n$, then $r(m) = r$ if and only if $m$ is the product of $r$ mutually distinct primes.

Proposition 5.3.28 Let $L$ be a geometric lattice. Then for all $x, y \in L$:

(GL.1) If $x < y$, then $r(x) < r(y)$ (strictly monotone)

(GL.2) $r(x \lor y) + r(x \land y) \leq r(x) + r(y)$ (semimodular inequality)

(GL.3) All maximal chains from 0 to $x$ have the same length $r(x)$.

Proof. See [113, Prop. 3.3.2] and [114, Prop. 3.7].

Remark 5.3.29 Let $L$ be an atomic lattice. Then $L$ is semimodular if and only if the semimodular inequality (GL.2) holds for all $x, y \in L$. And $L$ is modular if and only if the modular equality:

$$r(x \lor y) + r(x \land y) = r(x) + r(y)$$

for all $x, y \in L$.

Remark 5.3.30 Let $L$ be a geometric lattice. Let $x, y \in L$ and $x \leq y$. The chain $x = y_0 < y_1 < \cdots < y_s = y$ from $x$ to $y$ is called an extension of the chain $x = x_0 < x_1 < \cdots < x_r = y$ if $\{x_0, x_1, \ldots, x_r\}$ is a subset of $\{y_0, y_1, \ldots, y_s\}$. A chain from $x$ to $y$ is called maximal if there is no extension to a longer chain from $x$ to $y$. Every chain from $x$ to $y$ can be extended to a maximal chain with the same end points, and all such maximal chains have the same length $r(y) - r(x)$. This is called the Jordan-Hölder property.

Remark 5.3.31 Let $L$ be a geometric lattice. Let $L_j = \{x \in L | r(x) = j\}$. Then $L_j$ is called the level of $L$. Then the Hasse diagram of $L$ is a graph that has the elements of $L$ as vertices. If $x, y \in L$, $x < y$ and $r(y) = r(x) + 1$, then $x$ and $y$ are connected by an edge. So only elements between two consecutive levels $L_j$ and $L_{j+1}$ are connected by an edge. The Hasse diagram of $L$ considered as a poset as in Definition 5.3.13 is the directed graph with an arrow from $y$ to $x$ if $x, y \in L$, $x < y$ and $r(y) = r(x) + 1$.  

***picture***
Remark 5.3.32 Let $L$ be a geometric lattice. Then $L_x$ is a geometric lattice with $x$ as minimum element and of rank $r_L(1) - r_L(x)$, and $\mu_{L_x}(y) = \mu(x, y)$ and $r_{L_x}(y) = r_L(y) - r_L(x)$ for all $x \in L$ and $y \in L_x$. Similar remarks hold for $L^x$ and $[x, y]$.

Example 5.3.33 Let $L$ be the collection of all linear subspaces of a given finite dimensional vector space $V$ over a field $\mathbb{F}$ with the inclusion as partial order. Then $0_L = \{0\}$ is the minimum and $1_L = V$ is the maximum of $L$. The partial order $L$ is locally finite if and only if $L$ is finite if and only if the field $\mathbb{F}$ is finite. Let $x$ and $y$ be linear subspaces of $V$. Then $x \cap y$ the intersection of $x$ and $y$ is the largest linear subspace that is contained in $x$ and $y$. So $x \cap y = x \cap y$. The sum $x + y$ of of $x$ and $y$ is by definition the set of elements $a + b$ with $a$ in $x$ and $b$ in $y$. Then $x + y$ is the smallest linear subspace containing both $x$ and $y$. Hence $x \cup y = x + y$. So $L$ is a lattice. The atoms are the one dimensional linear subspaces. Let $x$ be a subspace of dimension $r$ over $\mathbb{F}$. So $x$ is generated by a basis $g_1, \ldots, g_r$. Let $a_i$ be the one dimensional subspace generated by $g_i$. Then $x = a_1 \cup \cdots \cup a_r$. Hence $L$ is atomic and $r(x) = \dim(x)$. Moreover $L$ is modular, since

$$\dim(x \cap y) + \dim(x + y) = \dim(x) + \dim(y)$$

for all $x, y \in L$. Furthermore $L$ has no infinite chains, since $V$ is finite dimensional. Therefore $L$ is a modular geometric lattice.

Example 5.3.34 Let $\mathbb{F}$ be a field. Let $V = (v_1, \ldots, v_n)$ be an $n$-tuple of nonzero vectors in $\mathbb{F}^k$. Let $L = L(V)$ be the collection of all linear subspaces of $\mathbb{F}^k$ that are generated by subsets of $V$ with inclusion as partial order. So $L$ is finite and a fortiori locally finite. By definition $\{0\}$ is the linear subspace space generated by the empty set. Then $0_L = \{0\}$ and $1_L$ is the subspace generated by all $v_1, \ldots, v_n$. Furthermore $L$ is a lattice with $x \cap y = x + y$ and

$$x \cup y = \bigvee \{ z \mid z \leq x, z \leq y \}$$

by Remark 5.3.16. Let $a_j$ be the linear subspace generated by $v_j$. Then $a_1, \ldots, a_n$ are the atoms of $L$. Let $x$ be the subspace generated by $\{v_j \mid j \in J\}$. Then $x = \bigvee_{j \in J} a_j$. If $x$ has dimension $r$, then there exists a subset $I$ of $J$ such that $|I| = r$ and $x = \bigvee_{i \in I} a_i$. Hence $L$ is atomic and $r(x) = \dim(x)$. Now $x \cap y \subseteq x \cap y$, so

$$r(x \cup y) + r(x \cap y) \leq \dim(x + y) + \dim(x \cap y) = r(x) + r(y).$$

Hence the semimodular inequality holds and $L$ is a geometric lattice. In most cases $L$ is not modular.

Example 5.3.35 Let $\mathbb{F}$ be a field. Let $\mathcal{A} = (H_1, \ldots, H_n)$ be an arrangement over $\mathbb{F}$ of hyperplanes in the vector space $V = \mathbb{F}^k$. Let $L = L(\mathcal{A})$ be the collection of all nonempty intersections of elements of $\mathcal{A}$. By definition $\mathbb{F}^k$ is the empty intersection. Define the partial order $\leq$ by

$$x \leq y \text{ if and only if } y \subseteq x.$$
Then $V$ is the minimum element and $\{0\}$ is the maximum element. Furthermore

$$x \lor y = x \cap y \text{ if } x \cap y \neq \emptyset \text{ and } x \land y = \bigcap \{ z \mid x \lor y \subseteq z \}.$$ 

Suppose that $A$ is a central arrangement. Then $x \cap y$ is nonempty for all $x, y$ in $L$. So $x \lor y$ and $x \land y$ exist for all $x, y$ in $L$, and $L$ is a lattice. Let $\mathbf{v}_j = (v_{1j}, \ldots, v_{kj})$ be a nonzero vector such that $\sum_{i=1}^{k} v_{ij} x_i = 0$ is a homogeneous equation of $H_j$. Let $V = (v_1, \ldots, v_n)$. Consider the map $\varphi : L(V) \to L(A)$ defined by

$$\varphi(x) = \bigcap_{j \in J} H_j \text{ if } x \text{ is the subspace generated by } \{v_j \mid j \in J\}.$$ 

Now $x \subseteq y$ if and only if $\varphi(y) \subseteq \varphi(x)$ for all $x, y \in L(V)$. So $\varphi$ is a strictly monotone map. Furthermore $\varphi$ is a bijection and its inverse map is also strictly monotone. Hence $L(V)$ and $L(A)$ are isomorphic lattices. Therefore $L(A)$ is also a geometric lattice.

### 5.3.3 Geometric lattices and matroids

The notion of a geometric lattice is "cryptomorphic" that is almost equivalent to a matroid. See [34, 38, 44, ?, 114].

**Proposition 5.3.36** Let $L$ be a finite geometric lattice. Let $M(L)$ be the set of all atoms of $L$. Let $I(L)$ be the collection of all subsets $I$ of $M(L)$ such that $r(a_1 \lor \cdots \lor a_r) = r$ if $I = \{a_1, \ldots, a_r\}$ is a collection of $r$ atoms of $L$. Then $(M(L), I(L))$ is a matroid.

**Proof.** The proof is left as an exercise. \hfill \(\diamondsuit\)

**Proposition 5.3.37 (Rota’s Crosscut Theorem)** Let $L$ be a finite geometric lattice. Let $M(L)$ be the matroid associated with $L$. Then

$$\chi_L(T) = \sum_{I \subseteq M(L)} (-1)^{|I|} \tau(L) - r(I).$$

**Proof.** See [101] and [24, Theorem 3.1]. \hfill \(\diamondsuit\)

**Definition 5.3.38** Let $(M, I)$ be a matroid. An element $x$ in $M$ is called a loop if $\{x\}$ is a dependent set. Let $x$ and $y$ in $M$ be two distinct elements that are not loops. Then $x$ and $y$ are called parallel if $r(\{x, y\}) = 1$. The matroid is called simple if it has no loops and no parallel elements.

**Remark 5.3.39** Let $G$ be a $k \times n$ matrix with entries in a field $F$. Let $M_G$ be the set $\{1, \ldots, n\}$ indexing the columns of $G$ and $I_G$ be the collection of all subsets $I$ of $M_G$ such that the submatrix $G_I$ consisting of the columns of $G$ at the positions of $I$ are independent. Then $(M_G, I_G)$ is a matroid. Suppose that $F$ is a finite field and $G_1$ and $G_2$ are generator matrices of a code $C$, then $(M_{G_1}, I_{G_1}) = (M_{G_2}, I_{G_2})$. So the matroid $(M_C, I_C)$ of a code $C$ is well defined by $(M_G, I_G)$ for some generator matrix $G$ of $C$. If $C$ is degenerate, then there is a position $i$ such that $c_i = 0$ for every codeword $c \in C$ and all such positions correspond one-to-one with loops of $M_C$. Let $C$ be nondegenerate. Then $M_C$
has no loops, and the positions \(i\) and \(j\) with \(i \neq j\) are parallel in \(M_C\) if and only if the \(i\)-th column of \(G\) is a scalar multiple of the \(j\)-th column. The code \(C\) is projective if and only if the arrangement \(\mathcal{A}_G\) is simple if and only if the matroid \(M_C\) is simple. An \([n, k]\) code \(C\) is MDS if and only if the matroid \(M_C\) is the uniform matroid \(U_{n,k}\).

**Remark 5.3.40** Let \(C\) be a projective code with generator matrix \(G\). Then \(\mathcal{A}_G\) is an essential simple arrangement with geometric lattice \(L(\mathcal{A}_G)\). Furthermore the matroids \(M(L(\mathcal{A}_G))\) and \(M_C\) are isomorphic.

**Definition 5.3.41** Let \((M, \mathcal{I})\) be a matroid. A \(k\)-flat of \(M\) is a maximal subset of \(M\) of rank \(k\). Let \(L(M)\) be the collection of all flats of \(M\), it is called the lattice of flats of \(M\). Let \(J\) be a subset of \(M\). Then the closure \(\bar{J}\) is by definition the intersection of all flats that contain \(J\).

**Remark 5.3.42** \(M\) is a \(k\)-flat with \(k = r(M)\). If \(F_1\) and \(F_2\) are flats, then \(F_1 \cap F_2\) is also a flat. Consider \(L(M)\) with the inclusion as partial order. Then \(M\) is the maximum of \(L(M)\). And \(F_1 \cap F_2 = F_1 \wedge F_2\) for all \(F_1\) and \(F_2\) in \(L(M)\).

**Proposition 5.3.46** Let \((M, \mathcal{I})\) be a matroid. Then \(L(M)\) with the inclusion as partial order is a geometric lattice and \(L(M)\) is isomorphic with \(L(\bar{M})\).

**Proof.** See [114, Theorem 3.8].

**5.3.4 Exercises**

5.3.1 Give a proof of Remark 5.3.9.

5.3.2 Give a proof of Remark 5.3.16.

5.3.3 Give a proof of the formulas for \(c_r(x, y)\) and \(\mu(x, y)\) in Example 5.3.17.

5.3.4 Give a proof of the formula for \(\mu(x)\) in Example 5.3.20.
5.3.5 Give a proof of the statements in Example 5.3.27.

5.3.6 Give an example of an atomic finite lattice with minimum 0 and maximum 1 that is not semimodular.

5.3.7 Give a proof of the statements in Remark 5.3.29.

5.3.8 Let \( L \) be a finite geometric lattice. Show that \((M(L), I(L))\) is a matroid as stated in Proposition 5.3.36. Show moreover that this matroid is simple.

5.3.9 Give a proof of the statements in Remark 5.3.39.

5.3.10 Give a proof of the statements in Remark 5.3.42.

5.3.11 Give a proof of Proposition 5.3.46.

5.3.12 Let \( L \) be a geometric lattice. Let \( a \) be an atom of \( L \) and \( x \in L \). Show that \( r(x \vee a) \leq r(x) + 1 \) and \( r(x \vee a) = r(x) \) if and only if \( a \leq x \).

5.3.13 Let \( L \) be a geometric lattice. Show that \( r(y) - r(x) \) is the length of every maximal chain from \( x \) to \( y \) for all \( x \leq y \) in \( L \).

5.3.14 Give a proof of Remark 5.3.32.

5.3.15 Give an example of a central arrangement \( A \) such that the lattice \( L(A) \) is not modular.

5.4 Characteristic polynomial

5.4.1 Characteristic and Möbius polynomial

Definition 5.4.1 Let \( L \) be a finite geometric lattice. The characteristic polynomial \( \chi_L(T) \) and the Poincaré polynomial \( \pi_L(T) \) of \( L \) are defined by:

\[
\chi_L(T) = \sum_{x \in L} \mu_L(x) T^{r(L) - r(x)}, \quad \text{and} \quad \pi_L(T) = \sum_{x \in L} \mu_L(x) (-T)^{r(x)}.
\]

The two variable Möbius polynomial \( \mu_L(S, T) \) in \( S \) and \( T \) is defined by

\[
\mu_L(S, T) = \sum_{x \in L} \sum_{x \leq y \in L} \mu(x, y) S^{r(x)} T^{r(L) - r(y)}.
\]

The two variable characteristic polynomial or coboundary polynomial is defined by

\[
\chi_L(S, T) = \sum_{x \in L} \sum_{x \leq y \in L} \mu(x, y) S^{a(x)} T^{r(L) - r(y)},
\]

where \( a(x) \) is the number of atoms \( a \) in \( L \) such that \( a \leq x \).

Remark 5.4.2 Now \( \mu(L) = \chi_L(0), \) and \( \chi_L(1) = 0 \) if and only if \( L \) consists of one element. Furthermore \( \chi_L(T) = T^{r(L)} \pi_L(-T^{-1}), \) and \( \mu_L(0, T) = \chi_L(0, T) = \chi_L(T). \)
5.4. CHARACTERISTIC POLYNOMIAL

Remark 5.4.3 Let \( r \) be the rank of \( L \). Then the following relation holds for the Möbius polynomial in terms of characteristic polynomials

\[
\mu_L(S, T) = \sum_{i=0}^{r} S^i \mu_i(T) \quad \text{with} \quad \mu_i(T) = \sum_{x \in L, i} \chi_{L_x}(T),
\]

where \( L_i = \{ x \in L \mid r(x) = i \} \) and \( n = L_1 \) the number of atoms in \( L \). Then similarly

\[
\chi_L(S, T) = \sum_{i=0}^{n} S^i \chi_i(T) \quad \text{with} \quad \chi_i(T) = \sum_{x \in L, a(x) = i} \chi_{L_x}(T).
\]

Remark 5.4.4 Let \( L \) be a geometric lattice. Then

\[
\sum_{i=0}^{r} \mu_i(T) = \mu_L(1, T) = \sum_{y \in L} \mu(x, y) T^{r(L) - r(y)} = T^{r(L)},
\]

since \( \sum_{0 \leq x \leq y} \mu(x, y) = 0 \) for all \( 0 < y \) in \( L \) by Proposition 5.3.5. Similarly \( \sum_{i=0}^{n} \chi_i(T) = \chi_L(1, T) = T^{r(L)} \). Also \( \sum_{i=0}^{n} A_i(T) = T^k \) for the extended weights of a code of dimension \( k \) by Proposition 4.4.38 for \( t = 0 \).

Example 5.4.5 Let \( L \) be the lattice of all subsets of a given finite set of \( r \) elements as in Example 5.3.17. Then \( r(x) = a(x) \) and \( \mu(x, y) = (-1)^{a(y) - a(x)} \) if \( x \leq y \). Hence

\[
\chi_L(T) = \sum_{j=0}^{r} \binom{r}{j} (-1)^j T^{r-j} = (T-1)^r
\]

and \( \mu_i(T) = \binom{r}{i} (T-1)^{r-i} \).

Therefore \( \mu_L(S, T) = (S + T - 1)^r \).

Example 5.4.6 Let \( L \) be the lattice of all linear subspaces of a given vector space of dimension \( r \) over the finite field \( \mathbb{F}_q \) as in Example 5.3.33. Then \( r(x) \) is the dimension of \( x \) over \( \mathbb{F}_q \). The number of subspaces of dimension \( i \) is counted in Proposition 4.3.7. It is left as an exercise to show that

\[
\mu(x, y) = (-1)^i q^{(j-i)(j-i-1)/2}
\]

if \( r(x) = i, r(y) = j \) and \( x \leq y \), and

\[
\chi_L(T) = \sum_{i=0}^{r} \binom{r}{i} (-1)^i q^{(i)(j-i)} T^{r-i} = (T-1)(T-q) \cdots (T-q^{r-1})
\]

and

\[
\mu_i(T) = \binom{r}{i} q^{(i)(j-i)} (T-1)(T-q) \cdots (T-q^{r-i-1}).
\]

See [71].

Remark 5.4.7 Every polynomial in one variable with coefficients in a field \( \mathbb{F} \) factorizes in linear factors over the algebraic closure \( \bar{\mathbb{F}} \) of \( \mathbb{F} \). In Examples 5.4.5 and 5.4.6 we see that \( \chi_L(T) \) factorizes in linear factors over \( \mathbb{Z} \). This is always the case for so called super solvable geometric lattices and lattices from free central arrangements. See [92].
CHAPTER 5. CODES AND RELATED STRUCTURES

Definition 5.4.8 Let $L$ be a finite geometric lattice. The Whitney numbers $w_i$ and $W_i$ of the first and second kind, respectively are defined by

$$w_i = \sum_{x \in L_i} \mu(x) \quad \text{and} \quad W_i = |L_i|.$$ 

The doubly indexed Whitney numbers $w_{ij}$ and $W_{ij}$ of the first and second kind, respectively are defined by

$$w_{ij} = \sum_{x \in L_i} \sum_{y \in L_j} \mu(x, y) \quad \text{and} \quad W_{ij} = |\{(x, y) | x \in L_i, y \in L_j, x \leq y\}|.$$ 

See [60], [34, §6.6.D], [?], Chapter 14] and [113, §3.11].

Remark 5.4.9 We have that

$$\chi_L(T) = \sum_{i=0}^{r(L)} w_i T^{r(L) - i} \quad \text{and} \quad \mu_L(S, T) = \sum_{i=0}^{r(L)} \sum_{j=0}^{r(L)} w_{ij} S^i T^{r(L) - j}.$$ 

Hence the (doubly indexed) Whitney numbers of he first kind are determined by $\mu_L(S, T)$. The leading coefficient of

$$\mu_i(T) = \sum_{x \in L_i} \sum_{x \leq y} \mu(x, y) T^{r(L_i) - r_L(y)}$$

is equal to $\sum_{x \in L_i} \mu(x, x) = |L_i| = W_i$. Hence the Whitney numbers of the second kind $W_i$ are determined by $\mu_L(S, T)$. We will see in Example 5.4.32 that the Whitney numbers are not determined by $\chi_L(S, T)$. Finally, let $r = r(L)$. Then

$$\mu_{r-1}(T) = W_{r-1}(T - 1)$$

5.4.2 Characteristic polynomial of an arrangement

A central arrangement $A$ gives rise to a geometric lattice $L(A)$ and characteristic polynomial $\chi_{L(A)}$ that will be denoted by $\chi_A$. Similarly $\pi_A$ denotes the Poincaré polynomial of $A$. If $A$ is an arrangement over the real numbers, then $\pi_A(1)$ counts the number of connected components of the complement of the arrangement. See [139]. Something similar can be said about arrangements over finite fields.

Proposition 5.4.10 Let $q$ be a prime power, and let $A = (H_1, \ldots, H_n)$ be a simple and central arrangement in $\mathbb{F}_q^k$. Then

$$\chi_A(q^m) = |\mathbb{F}_{q^m}^k \setminus (H_1 \cup \cdots \cup H_n)|.$$ 

Proof. See [7, Theorem 2.2], [17, Proposition 3.2], [44, Sect. 16] [92, Theorem 2.69].

Let $A = \mathbb{F}_{q^m}^k$ and $A_j = H_j(\mathbb{F}_{q^m})$. Let $L$ be the poset of all intersections of the $A_j$. The principle of inclusion/exclusion as formulated in Example 5.3.19 gives that

$$|\mathbb{F}_{q^m}^k \setminus (H_1 \cup \cdots \cup H_n)| = \sum_{x \in L} \mu(x)|x| = \sum_{x \in L} \mu(x)q^{m \dim(x)}.$$
5.4. CHARACTERISTIC POLYNOMIAL

The expression on the right hand side is equal to $\chi_A(q^m)$, since $L$ is isomorphic with the reverse of the geometric lattice $L(A)$ of the arrangement $A = (H_1, \ldots, H_n)$, so $\dim(x) = \mu_{L(A)} - \mu_{L(A)}(x)$ and $\mu_L(x) = \mu_{L(A)}(x)$ by Remark 5.3.12.

Definition 5.4.11 Let $A = (H_1, \ldots, H_n)$ be an arrangement in $\mathbb{F}^k$ over the field $\mathbb{F}$. Let $H = H_i$. Then the deletion $A \setminus H$ is the arrangement in $\mathbb{F}^k$ obtained from $(H_1, \ldots, H_n)$ by deleting all the $H_j$ such that $H_j = H$. Let $x = \cap_{i \in I} H_i$ be an intersection of hyperplanes of $A$. Let $l$ be the dimension of $x$. The restriction $A_x$ is the arrangement in $\mathbb{F}^l$ of all hyperplanes $x \cap H_j$ in $x$ such that $x \cap H_j \neq \emptyset$ and $x \cap H_j \neq x$, for a chosen isomorphism of $x$ with $\mathbb{F}^l$.

Proposition 5.4.12 Deletion-restriction formula Let $A = (H_1, \ldots, H_n)$ be a simple and central arrangement in $\mathbb{F}^k$ over the field $\mathbb{F}$. Let $H = H_i$. Then

$$\chi_A(T) = \chi_{A \setminus H}(T) - \chi_{A_H}(T).$$

Proof. A proof for an arbitrary field can be found in [92, Theorem 2.56]. Here the special case of a central arrangement over the finite field $\mathbb{F}_q$ will be treated. Without loss of generality we may assume that $H = H_1$. Denote $H_j(\mathbb{F}_q^m)$ by $H_j$ and $\mathbb{F}^k$ by $V$. Then the following set is written as the disjoint union of two others.

$$V \setminus (H_2 \cup \cdots \cup H_n) = (V \setminus (H_1 \cup H_2 \cup \cdots \cup H_n)) \cup (H_1 \setminus (H_2 \cup \cdots \cup H_n)).$$

The number of elements of the left hand side is equal to $\chi_{A_1}(q^m)$, and the number of elements of the two sets on the right hand side are equal to $\chi_{A_1}(q^m)$ and $\chi_{A_H}(q^m)$, respectively by Proposition 5.4.10. Hence

$$\chi_{A_H}(q^m) = \chi_A(q^m) + \chi_{A_H}(q^m)$$

for all positive integers $m$, since the union is disjoint. Therefore the identity of the polynomial holds.

Definition 5.4.13 Let $A = (H_1, \ldots, H_n)$ be a central simple arrangement over the field $\mathbb{F}$ in $\mathbb{F}^k$. Let $J \subseteq \{1, \ldots, n\}$. Define $H_J = \cap_{j \in J} H_j$. Consider the decreasing sequence

$$N_k \subset N_{k-1} \subset \cdots \subset N_1 \subset N_0,$$

of algebraic subsets of the affine space $\mathbb{A}^k$, defined by

$$N_i = \bigcup_{J \subseteq \{1, \ldots, n\}, r(H_J) = i} H_J.$$

Define $M_i = (N_i \setminus N_{i+1})$.

Remark 5.4.14 $N_0 = \mathbb{A}^k$, $N_1 = \cup_{j=1}^n H_j$, $N_k = \{0\}$ and $N_{k+1} = \emptyset$. Furthermore $N_i$ is a union of linear subspaces of $\mathbb{A}^k$ all of dimension $k - i$. Notice that $H_J$ is isomorphic with $C(J)$ in case $A$ is the arrangement of the generator matrix $G$ of the code $C$ as remarked in the proof of Proposition 4.4.8.
Proposition 5.4.15 Let $A = (H_1, \ldots, H_n)$ be a central simple arrangement over the field $F$ in $F^k$. Let $z(x) = \{ j \in \{1, \ldots, n\} \mid x \in H_j \}$ and $r(x) = r(H_z(x))$ the rank of $x$ for $x \in A^k$. Then

$$N_i = \{ x \in A^k \mid r(x) \geq i \} \text{ and } M_i = \{ x \in A^k \mid r(x) = i \}.$$  

Proof. Let $x \in A^k$ and $c = xG$. Let $x \in N_i$. Then there exists a $J \subseteq \{1, \ldots, n\}$ such that $r(H_J) = i$ and $x \in H_J$. So $c_j = 0$ for all $j \notin J$. So $J \subseteq z(x)$. Hence $H_z(x) \subseteq H_J$. Therefore $r(x) = r(H_z(x)) \geq r(H_J) = i$. The converse implication is proved similarly. The statement about $M_i$ is a direct consequence of the one about $N_i$.  

Proposition 5.4.16 Let $A$ be a central simple arrangement over $F_q$. Let $L = L(A)$ be the geometric lattice of $A$. Then

$$\mu_i(q^m) = |M_i(F_{q^m})|.$$  

Proof. See also [7, Theorem 6.3]. Remember that $\mu_i(T) = \sum_{r(x) = i} \chi_{L,r}(T)$ as defined in Remark 5.4.3. Let $L = L(A)$ and $x \in L$. Then $L(A_x) = L_x$. Let $\cup A_x$ be the union of the hyperplanes of $A_x$. Then $|(x \setminus (\cup A_x))(F_{q^m})| = \chi_{L_x}(q^m)$ by Proposition 5.4.10. Now $M_i$ is the disjoint union of complements of the arrangements of $A_x$ for all $x \in L$ such that $r(x) = i$ by Proposition 5.4.15. Hence

$$|M_i(F_{q^m})| = \sum_{x \in L, r(x) = i} |(x \setminus (\cup A_x))(F_{q^m})| = \sum_{x \in L, r(x) = i} \chi_{L_x}(q^m).$$  

5.4.3 Characteristic polynomial of a code

Proposition 5.4.17 Let $C$ be a nondegenerate $F_q$-linear code. Then

$$A_n(T) = \chi_C(T).$$  

Proof. The elements in $F_{q^m}^n \setminus (H_1 \cup \cdots \cup H_n)$ correspond one-to-one to codewords of weight $n$ in $C \otimes F_{q^m}$ by Proposition 4.4.8. So $A_n(q^m) = \chi_C(q^m)$ for all positive integers $m$ by Proposition 5.4.10. Hence $A_n(T) = \chi_C(T)$.

Definition 5.4.18 Let $G$ be a generator matrix of an $[n, k]$ code $C$ over $F_q$. Define

$$\mathcal{Y}_i = \{ x \in A^k \mid \text{wt}(xG) \leq n - i \} \text{ and } \mathcal{X}_i = \{ x \in A^k \mid \text{wt}(xG) = n - i \}.$$  

Remark 5.4.19 The $\mathcal{Y}_i$ form a decreasing sequence

$$\mathcal{Y}_n \subseteq \mathcal{Y}_{n-1} \subseteq \cdots \subseteq \mathcal{Y}_1 \subseteq \mathcal{Y}_0,$$

of algebraic subsets of $A^k$ and $\mathcal{X}_i = (\mathcal{Y}_i \setminus \mathcal{Y}_{i+1})$.

Proposition 5.4.20 Let $C$ be a projective code of length $n$. Then

$$\chi_i(q^m) = |\mathcal{X}_i(F_{q^m})| = A_{n-i}(q^m).$$
5.4. Characteristic Polynomial

Proof. Every \( x \in \mathbb{F}_q^m \) corresponds one-to-one to codeword in \( C \otimes \mathbb{F}_q^m \) via the map \( x \mapsto xG \). So \( |X_i(\mathbb{F}_q^m)| = A_{n-i}(q^m) \). And \( A_{n-i}(q^m) = \chi_i(q^m) \) for all \( i \), by Remark ??.

Corollary 5.4.21 Let \( C \) be a projective code of length \( n \). Then \( \chi_i(T) = A_{n-i}(T) \) for all \( i \).

Remark 5.4.22 Another way to define \( X_i \) is the collection of all points \( P \in \mathbb{A}^k \) such that \( P \) is on exactly \( i \) distinct hyperplanes of the arrangement \( \mathcal{A}_G \). Denote the arrangement of hyperplanes in \( \mathbb{P}^{k-1} \) also by \( \mathcal{A}_G \), and let \( P \) be the point in \( \mathbb{P}^{k-1} \) corresponding to \( P \in \mathbb{A}^k \). Define

\[ X_i = \{ P \in \mathbb{P}^{k-1} | P \text{ is on exactly } i \text{ hyperplanes of } \mathcal{A}_G \}. \]

For all \( i < n \) the polynomial \( \chi_i(T) \) is divisible by \( T - 1 \). Define \( \bar{\chi}_i(T) = \chi_i(T)/(T - 1) \). Then \( \bar{\chi}_i(q^m) = |X_i(\mathbb{F}_q^m)| \) for all \( i < n \) by Proposition 5.4.20.

Theorem 5.4.23 Let \( G \) be a generator matrix of a nondegenerate code \( C \). Let \( \mathcal{A}_G \) be the associated central arrangement. Let \( d^\perp = d(C^\perp) \). Then \( N_i \subseteq \gamma_i \) for all \( i \), equality holds for all \( i < d^\perp \) and \( \mathcal{M}_i = X_i^* \) for all \( i < d^\perp - 1 \). If furthermore \( C \) is projective, then

\[ \mu_i(T) = \bar{\chi}_i(T) = A_{n-i}(T) \text{ for all } i < d^\perp - 1. \]

Proof. Let \( x \in N_i \). Then \( x \in H_J \) for some \( J \subseteq \{1, \ldots, n\} \) such that \( r(H_J) = i \). So \( |J| \geq i \) and \( wt(xG) \leq n - i \) by Proposition 4.4.8. Hence \( x \in \gamma_i \). Therefore \( N_i \subseteq \gamma_i \).

Let \( i < d^\perp \) and \( x \in \gamma_i \). Then \( wt(xG) \leq n - i \). Let \( J = \supp(xG) \). Then \( |J| \geq i \). Take a subset \( I \) of \( J \) such that \( |I| = i \). Then \( x \in H_I \) and \( r(I) = |I| = i \) by Lemma 7.4.39, since \( i < d^\perp \). Hence \( x \in N_i \). Therefore \( \gamma_i \subseteq N_i \). So \( \gamma_i = N_i \) for all \( i < d^\perp \), and \( \mathcal{M}_i = X_i^* \) for all \( i < d^\perp - 1 \).

The code is nondegenerate. So \( d(C^\perp) \geq 2 \). Suppose furthermore that \( C \) is projective. Then \( \mu_i(T) = \bar{\chi}_i(T) = A_{n-i}(T) \) for all \( i < d^\perp - 1 \), by Remark ?? and Propositions 5.4.20 and 5.4.16.

The extended and generalized weight enumerators are determined by the pair \((n, k)\) for an \([n, k]\) MDS code by Remark ?? If \( C \) is an \([n, k]\) code, then \( d(C^\perp) \) is at most \( k + 1 \). Furthermore \( d(C^\perp) = k + 1 \) if and only if \( C \) is MDS if and only if \( C^\perp \) is MDS. An \([n, k, d]\) code is called almost MDS if \( d = n - k \). So \( d(C^\perp) = k \) if and only if \( C^\perp \) is almost MDS. If \( C \) is almost MDS, then \( C^\perp \) is not necessarily almost MDS. The code \( C \) is called near MDS if both \( C \) and \( C^\perp \) are almost MDS. See [?].

Proposition 5.4.24 Let \( C \) be an \([n, k, d]\) code such that \( C^\perp \) is MDS or almost MDS and \( k \geq 3 \). Then both \( \chi_C(S, T) \) as \( W_C(X, Y, T) \) determine \( \mu_C(S, T) \). In particular

\[ \mu_i(T) = \chi_i(T) = A_{n-i}(T) \text{ for all } i < k - 1, \]

\[ \mu_{k-1}(T) = \sum_{i=k-1}^{n-1} \chi_i(T) = \sum_{i=k-1}^{n-1} A_{n-i}(T), \]

and \( \mu_k(T) = 1 \).
Proof. Let $C$ be a code such that $d(C^⊥) \geq k \geq 3$. Then $C$ is projective and $A_{n-1} = \chi_i$ for all $i < k - 1$ by Remark ??.

If $i < k - 1$, then the expression for $\mu_i(T)$ is given by Theorem 5.4.23. Furthermore $\mu_k(T) = \chi_n(T) = A_0(T) = 1$. Finally let $L = L(C)$. Then $\sum_{i=0}^{k} \mu_i(T) = T^k$, $\sum_{i=0}^{n} \chi_i(T) = T^k$ and $\sum_{i=0}^{n} A_i(T) = T^k$ by Remark 5.4.4. Hence the formula for $\mu_{k-1}(T)$ holds. Therefore $\mu_C(S, T)$ is determined both by $W_C(X, Y, T)$ and $\chi_C(S, T)$.

Projective codes of dimension 3 are examples of codes $C$ such that $C^⊥$ is almost MDS. In the following we will give explicit formulas for $\mu_C(S, T)$ for such codes.

Let $C$ be a projective code of length $n$ and dimension 3 over $\mathbb{F}_q$ with generator matrix $G$. The arrangement $A_G = (H_1, \ldots, H_n)$ of planes in $\mathbb{P}^3_q$ is simple and essential, and the corresponding arrangement of lines in $\mathbb{P}^2(\mathbb{F}_q)$ is also denoted by $A_G$. We defined

$$\mathcal{X}_i(\mathbb{F}_q) = \{ P \in \mathbb{P}^2(\mathbb{F}_q) \mid \text{ is correct on i lines of } A_G \}$$

and $\chi_i(q^m) = |\mathcal{X}_i(\mathbb{F}_q^m)|$ in Remark 5.4.22 for all $i < n$.

Remark 5.4.25 Notice that for projective codes of dimension three $\mathcal{X}_i(\mathbb{F}_q^m) = \mathcal{X}_i(\mathbb{F}_q)$ for all positive integers $m$ and $2 \leq i < n$. Abbreviate in this case $\chi_i(q^m) = \chi_i$ for $2 \leq i < n$.

Proposition 5.4.26 Let $C$ be a projective code of length $n$ and dimension 3 over $\mathbb{F}_q$. Then

$$\begin{align*}
\mu_0(T) &= (T - 1) \left( T^2 - (n - 1)T + \sum_{i=2}^{n-1} (i - 1)\chi_i - n + 1 \right), \\
\mu_1(T) &= (T - 1) \left( nT + n - \sum_{i=2}^{n-1} i\chi_i \right), \\
\mu_2(T) &= (T - 1) \left( \sum_{i=2}^{n-1} i\chi_i \right).
\end{align*}$$

Proof. A more general statement and proof is possible for $[n, k]$ codes $C$ such that $d(C^⊥) \geq k$, using Proposition 5.4.24, the fact that $B_t(T) = T^k-t - 1$ for all $t < d(C^⊥)$ by Lemma 7.4.39 and the expression of $B_t(T)$ in terms of $A_w(T)$ by Proposition ??.. We will give a second geometric proof for the special case of projective codes of dimension 3.

It is enough to show this proposition with $T = q^m$ for all $m$ by Lagrange interpolation. Notice that $\mu_i(q^m)$ is the number of elements of $\mathcal{M}_i(\mathbb{F}_q^m)$ by Proposition 5.4.16. Let $P$ be the corresponding point in $\mathbb{P}^2(\mathbb{F}_q^m)$ for $P \in \mathbb{P}^3_q$ and $P \neq 0$. Abbreviate $\mathcal{M}_i(\mathbb{F}_q^m)$ by $\mathcal{M}_i$. Define $\mathcal{M}_i = \{ P \mid P \in \mathcal{M}_i \}$. Then $|\mathcal{M}_i| = (q^m - 1)\sum_{i=2}^{n-1} \chi_i$ for all $i < 3$.

(1) If $P \in \mathcal{M}_2$, then $P \in H_j \cap H_k$ for some $j \neq k$. Hence $P \in \mathcal{X}_i(\mathbb{F}_q)$ for some $i \geq 2$, since the code is projective. So $\mathcal{M}_2$ is the disjoint union of the $\mathcal{X}_i(\mathbb{F}_q)$, $2 \leq i < n$. Therefore $|\mathcal{M}_2| = \sum_{i=2}^{n-1} \chi_i$.

(2) $P \in \mathcal{M}_1$ if and only if $P$ is on exactly one line $H_j$. There are $n$ lines, and every line has $q^m + 1$ points that are defined over $\mathbb{F}_q$. If $i \geq 2$, then every $P \in \mathcal{X}_i(\mathbb{F}_q)$ is on $i$ lines $H_j$. Hence $|\mathcal{M}_1| = n(q^m + 1) - \sum_{i=2}^{n-1} i\chi_i$.

(3) $\mathbb{P}^2$ is the disjoint union of $\mathcal{M}_1, \mathcal{M}_2$ and $\mathcal{M}_0$. The numbers $|\mathcal{M}_2|$ and $|\mathcal{M}_1|$ are computed in (1) and (2), and $|\mathbb{P}^2(\mathbb{F}_q)| = q^6 + q^2 + 1$. From this we derive the number of elements of $\mathcal{M}_0$. 

\(\diamondsuit\)
Example 5.4.27 Consider the matrices $G$ and $P$ given by

$$
G = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix} \quad \text{and} \quad P = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & -1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & -1 & 1 & 1 & 1 \\ 0 & 0 & 1 & -1 & -1 & 0 & 1 & 1 & -1 \end{pmatrix}.
$$

Let $C$ be the code over $F_q$ with generator matrix $G$. The columns of $G$ represent also the coefficients of the lines of $A_G$. The $j$-th column of $P$ represents the homogenous coordinates of the points $P_j$ in the projective plane that occur as intersections of two lines of $A_G$. In case $q$ is even, the points $P_7, P_8$ and $P_9$ coincide.

***two pictures: q odd and q even***

If $q$ is even, then $\bar{\chi}_2 = 0$ and $\bar{\chi}_3 = 7$. If $q$ is odd, then $\bar{\chi}_2 = 3$ and $\bar{\chi}_3 = 6$.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q$ even</td>
<td>$\chi_\bar{i}$</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>$\bar{\mu}_{\bar{i}}$</td>
<td>7</td>
<td>7T - 14</td>
<td>$T^2 - 6T + 8$</td>
<td>7T - 14</td>
<td>$T^2 - 6T + 8$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q$ odd</td>
<td>$\chi_\bar{i}$</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>$\bar{\mu}_{\bar{i}}$</td>
<td>9</td>
<td>7T - 17</td>
<td>$T^2 - 6T + 9$</td>
<td>7T - 17</td>
<td>$T^2 - 6T + 9$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Notice that there is a codeword of weight 7 in case $q$ is even and $q > 4$ or $q$ is odd and $q > 3$, since $A_7(T) = (T - 2)(T - 4)$ or $A_7(T) = (T - 3)^2$, respectively.

Example 5.4.28 Let $G$ be a $3 \times n$ generator matrix of an MDS code. The lines of the arrangement $A_G$ are in general position. That means that every two distinct lines meet in one point, and every three mutually distinct lines have an empty intersection. So $\bar{\chi}_2 = \binom{n}{2}$ and $\bar{\chi}_3 = 0$ for all $i > 2$. Hence $\bar{\mu}_{n-2}(T) = \bar{\mu}_{n-3}(T) = \binom{n}{2}$ and $A_{n-1}(T) = \bar{\mu}_1(T) = nT + 2n - n^2$ and $A_n(T) = \bar{\mu}_0(T) = T^2 - (n - 1)T + \binom{n-1}{2}$, by Proposition 5.4.16 and Theorem ?? which is in agreement with Proposition 4.4.22.

Example 5.4.29 Let $a$ and $b$ be positive integers such that $2 < a < b$. Let $n = a + b$. Let $G$ be a $3 \times n$ generator matrix of a nondegenerate code. Suppose that there are two points $P$ and $Q$ in the projective plane over $F_q$ such that the $a + b$ lines of the projective arrangement of $A_G$ consists of $a$ distinct lines incident with $P$, and $b$ distinct lines incident with $Q$ and there is no line incident with $P$ and $Q$. Then $A_{n-2} = \bar{\chi}_2 = ab, \bar{\mu}_a = \bar{\chi}_a = 1$ and $A_b = \bar{\chi}_b = 1$. Hence $\bar{\mu}_2(T) = ab + 2$. Furthermore

$$
A_{n-1}(T) = \bar{\mu}_1(T) = (a + b)T - 2ab,
$$

$$
A_a(T) = \bar{\mu}_a(T) = T^2 - (a + b - 1)T + ab - 1
$$

and $\bar{\mu}_i(T) = 0$ for all $i \neq a, b, n - 2, n - 1, n$.  

5.4. CHARACTERISTIC POLYNOMIAL

153
Example 5.4.30 Let \( a, b \) and \( c \) be positive integers such that \( 2 < a < b < c \). Let \( n = a + b + c \). Let \( G \) be a \( 3 \times n \) generator matrix of a nondegenerate code \( C(a, b, c) \). Suppose that there are three points \( P, Q \) and \( R \) in the projective plane over \( \mathbb{F}_q \) such that the lines of the projective arrangement of \( A_G \) consist of a distinct line incident with \( P \) and not with \( Q \) and \( R \), \( b \) distinct lines incident with \( Q \) and not with \( P \) and \( R \), and \( c \) distinct lines incident with \( R \) and not with \( P \) and \( Q \). If \( q \) is large enough, then such a configurations exists. The \( a \) lines through \( P \) intersect the \( b \) lines through \( Q \) in \( ab \) points. Similarly statements hold for the lines through \( P \) and \( R \) intersecting in \( ac \) points, and the lines through \( Q \) and \( R \) intersecting in \( bc \) points. All these intersection points are on exactly two lines of the arrangement and there are no other. Hence \( \chi_2 = ab + bc + ca \). Now \( P \) is the unique point on exactly \( a \) lines of the arrangement. So \( \chi_a = 1 \). Similarly \( \chi_b = \chi_c = 1 \). Finally \( \chi_i = 0 \) for all \( 2 \leq i < n \) and \( i \notin \{2, a, b, c\} \). Now \( \mu_i(T) \) is divisible by \( T - 1 \) for all \( 0 \leq i < k \). Define \( \bar{\mu}_1(T) = \mu_1(T)/(T - 1) \). Define similarly \( A_w(T) = A_w(T)/(T - 1) \) for all \( 0 < w \leq n \). Propositions 5.4.24 and 5.4.26 imply that \( \bar{A}_{n-a} = \bar{A}_{n-b} = \bar{A}_{n-c} = 1 \) and \( \bar{A}_{n-2} = ab + bc + ca \) and \( \bar{\mu}_2(T) = ab + bc + ca + 3 \). Furthermore

\[
\bar{A}_{n-1}(T) = \bar{\mu}_1(T) = nT - 2(ab + bc + ca),
\]

\[
\bar{A}_n(T) = \bar{\mu}_0(T) = T^2 - (n - 1)T + ab + bc + ca - 2
\]

and \( \bar{A}_i(T) = 0 \) for all \( i \notin \{0, n - a, n - b, n - c, n - 2, n - 1, n\} \).

Therefore \( W_{C(a, b, c)}(X, Y, T) = W_{C(a', b', c')}(X, Y, T) \) if and only if \( (a, b, c) = (a', b', c') \), and \( \mu_{C(a, b, c)}(S, T) = \mu_{C(a', b', c')}(S, T) \) if and only if \( a + b + c = a' + b' + c' \) and \( ab + bc + ca = a'b' + b'c' + c'a' \). In particular let \( C_1 = C(3, 9, 14) \) and \( C_2 = C(5, 6, 15) \). Then \( C_1 \) and \( C_2 \) are two projective codes with the same Möbius polynomials \( \mu_{C}(S, T) \) but distinct extended weight enumerators and coboundary polynomials \( \chi_C(S, T) \).

Example 5.4.31 Consider the codes \( C_3 \) and \( C_4 \) over \( \mathbb{F}_q \) with \( q > 2 \) with generator matrices \( G_3 \) and \( G_4 \) given by

\[
G_3 = \begin{pmatrix}
1 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 1 & 0 \\
-1 & 0 & 1 & 1 & 0 & 0 & 1
\end{pmatrix}
\quad \text{and} \quad
G_4 = \begin{pmatrix}
1 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 1 & 0 \\
0 & 1 & 1 & a & 0 & 1 & 0
\end{pmatrix},
\]

where \( a \in \mathbb{F}_q \setminus \{0, 1\} \). It was shown in [34, Exercise 6.96] that the duals of these codes have the same Tutte polynomial. So the codes \( C_3 \) and \( C_4 \) have the same Tutte polynomial

\[
t_C(X, Y) = 2X + 2Y + 3X^2 + 5XY + 4Y^2 + X^3 + X^2Y + 2XY^2 + 3Y^3 + Y^4.
\]

Hence \( C_3 \) and \( C_4 \) have the extended weight enumerator given by

\[
X^7 + (2T^2 - 2)X^4Y^3 + (3T^3 - 3)X^3Y^4 + (T^2 + T - 2)X^2Y^5 + (5T^2 - 15T + 10)XY^6 + (3T^3 - 6T^2 + 11T - 6)Y^7.
\]

The codes \( C_3 \) and \( C_4 \) are not projective and their reductions \( \bar{C}_3 \) and \( \bar{C}_4 \), respectively have generator matrices given by

\[
\bar{G}_3 = \begin{pmatrix}
1 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 & 0 & 1 \\
-1 & 0 & 1 & 0 & 0 & 1 & 0
\end{pmatrix}
\quad \text{and} \quad
\bar{G}_4 = \begin{pmatrix}
1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & a & 0 & 1 & 0
\end{pmatrix},
\]
From the arrangement $\mathcal{A}(C_3)$ and $\mathcal{A}(C_4)$ we deduce the $\tilde{\chi}_i(T)$ that are given in the following table.

<table>
<thead>
<tr>
<th>code \ $i$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_3$</td>
<td>$T^2 - 5T + 6$</td>
<td>$6T - 12$</td>
<td>3</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$C_4$</td>
<td>$T^2 - 5T + 6$</td>
<td>$6T - 13$</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Therefore $t_{C_3}(X,Y) = t_{C_4}(X,Y)$ but $\chi_{C_3}(S,T) \neq \chi_{C_4}(S,T)$ and $t_{\bar{C}_3}(X,Y) \neq t_{\bar{C}_4}(X,Y)$.

**Example 5.4.32** Let $C_5 = C_3^4$ and $C_6 = C_4^4$. Then $C_5$ and $C_6$ have the same Tutte polynomial $t_{C_{\perp}}(X,Y) = t_C(Y,X)$ as given by by Example 5.4.31:

$$2X + 2Y + 4X^2 + 5XY + 3Y^2 + 2X^2Y + YY^2 + Y^3 + 3X^4.$$  

Hence $C_5$ and $C_6$ have the same extended weight enumerator given by

$$X^7 + (T-1)X^5Y^2 + (67-6)X^4Y^3 + (2T^2 - T - 1)X^3Y^4 + (15T^2 - 43T + 28)X^2Y^5 +$$

$$+(7T^3 - 36T^2 + 60T - 31)XY^6 + (T^4 - 7T^3 + 19T^2 - 23T + 10)Y^7.$$  

The geometric lattice $L(C_5)$ has atoms $a,b,c,d,e,f,g$ corresponding to the first, second, etc. column of $G_3$. The second level of $L(C_5)$ consists of the following 17 elements:

$$abe, ac, ad, af, ag, bc, bd, bf, bg, cd, ce, cf, cg, de, df, dg, efg.$$  

The third level consists of the following 12 elements:

$$abc, abe, abf, acdg, acf, adf, bcdf, bcg, bdg, cde, cefg, defg.$$  

Similarly, the geometric lattice $L(C_6)$ has atoms $a,b,c,d,e,f,g$ corresponding to the first, second, etc. column of $G_4$. The second level of $L(C_6)$ consists of the following 17 elements:

$$abe, ac, ad, af, ag, bc, bd, bf, bg, cd, ce, cf, cg, de, dfg, ef, eg.$$  

The third level consists of the following 13 elements:

$$abc, abd, abf, abcg, acdg, acf, adf, bcdfg, cde, cef, ceg, defg.$$  

Proposition 5.4.24 implies that $\mu_0(T)$ and $\mu_1(T)$ are the same for both codes and equal to

$$\mu_0(T) = \chi_0(T) = A_7(T) = (T-1)(T-2)(T^2 - 4T + 5)$$

$$\mu_1(T) = \chi_1(T) = A_6(T) = (T-1)(7T^2 - 29T + 31).$$  

The polynomials $\mu_3(T)$ and $\mu_2(T)$ are given in the following table using Remarks 5.4.9 and 5.4.4.

<table>
<thead>
<tr>
<th></th>
<th>$C_5$</th>
<th>$C_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu_2(T)$</td>
<td>$17T^2 - 49T + 32$</td>
<td>$17T^2 - 50T + 33$</td>
</tr>
<tr>
<td>$\mu_3(T)$</td>
<td>$12T - 12$</td>
<td>$13T - 13$</td>
</tr>
</tbody>
</table>

This example shows that the Möbius polynomial $\mu_C(S,T)$ is not determined by coboundary polynomials $\chi_C(S,T)$.  

5.4.4 Minimal codewords and subcodes

Definition 5.4.33 A minimal codeword of a code $C$ is a codeword whose support does not properly contain the support of another codeword.

Remark 5.4.34 The zero word is a minimal codeword. Notice that the nonzero scalar multiple of a minimal codeword is again a minimal codeword. Nonzero minimal codewords play a role in minimum distance decoding. Minimal codewords play a role in minimum distance decoding algorithms [6, 8, 9] and secret sharing schemes and access structures [80, 117]. We can generalize this notion to subcodes instead of words.

Definition 5.4.35 A minimal subcode of dimension $r$ of a code $C$ is an $r$-dimensional subcode whose support is not properly contained in the support of another $r$-dimensional subcode.

Remark 5.4.36 A minimal codeword generates a minimal subcode of dimension one, and all the elements of a minimal subcode of dimension one are minimal codewords. A codeword of minimal weight is a nonzero minimal codeword, but the converse is not always the case.

In the Example 5.4.32 it is shown that the codes $C_5$ and $C_6$ have the same Tutte polynomial whereas the number of minimal codewords of the code $C_5$ is 12 and of $C_6$ is 13. Hence the number of minimal codewords and subcodes is not determined by the Tutte polynomial. However the number of minimal codewords and the number of minimal subcodes of a given dimension are given by the Möbius polynomial.

Theorem 5.4.37 Let $C$ be a code of dimension $k$. Let $0 \leq r \leq k$. Then the number of minimal subcodes of dimension $r$ is equal to $W_{k-r}$, the $(r-k)$-th Whitney number of the second kind and it is determined by the Möbius polynomial.

Proof. Let $D$ be a subcode of $C$ of dimension $r$. Let $J$ be the complement in $[n]$ of the support of $D$. If $d \in D$ and $d_j \neq 0$, then $j \in \text{supp}(D)$ and $j \notin J$. Hence $D \subseteq C(J)$. Now suppose moreover that $D$ is a minimal subcode of $C$. Without loss of generality we may assume that $D$ is systematic at the first $r$ positions. So $D$ has a generator matrix of the form $(I_r|A)$. Let $d_j$ be the $j$-th row of this matrix. Let $c \in C(J)$. If $c = \sum_{j=1}^{r} c_j d_j$ is not the zero word, then the subcode $D'$ of $C$ generated by $c, d_1, \ldots, d_r$ has dimension $r$ and its support is contained in $\text{supp}(D) \setminus \{1\}$ and $1 \in \text{supp}(D)$. This contradicts the minimality of $D$. Hence $c = \sum_{j=1}^{r} c_j d_j = 0$ and $c \in D$. Therefore $D = C(J)$. To find a minimal subcode of dimension $r$, we fix $l(J) = r$ and minimalize the support of $C(J)$ with respect to inclusion. Because $J$ is contained in the complement in $[n]$ of the support of $C(J)$, this is equivalent to maximize $J$ with respect to inclusion. In matroid terms this means we are maximizing $J$ for $r(J) = k - l(J) = k - r$. This means $J = \bar{J}$ is a flat of rank $k - r$ by Remark 5.3.45. The flats of a matroid are the elements in the geometric lattice $L = L(M)$. The number of $(k-r)$-dimensional elements in $L(M)$ is equal to $|L_{k-r}|$, which is equal to the Whitney number of the second kind $W_{k-r}$ and thus equal to the leading coefficient of $\mu_{k-r}(T)$ by Remark 5.4.9. Hence the Möbius polynomial determines all the numbers of minimal subcodes of dimension $r$ for $0 \leq r \leq k$. \hfill \diamondsuit
Remark 5.4.38 Note that the flats of dimension \( k - r \) in a matroid are exactly the hyperplanes in the \((r - 1)\)-th truncated matroid \( T^{r-1}(M) \). This gives another proof of the result of Britz [28, Theorem 3] that the minimal supports of dimension \( r \) are the cocircuits of the \((r - 1)\)-th truncated matroid. For \( r = 1 \), this gives the well-known equivalence between nonzero minimal codewords and cocircuits See [? , Theorem 9.2.4] and [123, 1.21].

5.4.5 Two variable zeta function

Generally the counting of rational points over field extensions \( F_q^n \) is computed by the zeta function.

Definition 5.4.39 Let \( X \) be an affine variety in \( \mathbb{A}^k \) defined over \( F_q \), that is the zeroset of a collection of polynomials in \( F_q[X_1, \ldots, X_k] \). Then \( X(F_q^m) \) is the set of all points \( X \) with coordinates in \( F_q^m \), also called the the set of \( F_q^n\)-rational points of \( X \). The zeta function \( Z_X(T) \) of \( X \) is the formal power series in \( T \) defined by

\[
Z_X(T) = \exp \left( \sum_{m=1}^{\infty} \frac{|X(F_q^m)|}{T^r} \right).
\]

Theorem 5.4.40 Let \( A \) be a central simple arrangement in \( \mathbb{A}^k \). Let \( \chi_A(T) = \sum_{j=0}^{k} c_j T^j \) be the characteristic polynomial of \( A \). Let \( M = \mathbb{A}^k \setminus (H_1 \cup \cdots \cup H_n) \) be the complement of the arrangement. Then the zeta function of \( M \) is given by:

\[
Z_M(T) = \prod_{j=0}^{k} (1 - q^j T)^{-c_j}.
\]

Proof. See [17, Theorem 3.6].

Two variable zeta function of Duursma

5.4.6 Overview

We have established relations between the generalized weight enumerators for \( 0 \leq r \leq k \), the extended weight enumerator and the Tutte polynomial. We summarize this in the following diagram:

\[
\begin{align*}
W_C(X,Y) & \quad \xrightarrow{4.5.21} \quad W_C(X,Y,T) \\
\{W_C^{(r)}(X,Y)\}_{r=0}^{k} & \quad \xrightarrow{5.2.22} \quad t_C(X,Y) \\
\{W_C^{(r)}(X,Y,T)\}_{r=0}^{k} & \quad \xrightarrow{5.2.22} \quad W_C(X,Y,T)
\end{align*}
\]
We see that the Tutte polynomial, the extended weight enumerator and the collection of generalized weight enumerators all contain the same amount of information about a code, because they completely define each other. The original weight enumerator $W_C(X,Y)$ contains less information and therefore does not determine $W_C(X,Y,T)$ or $\{W_C^{(r)}(X,Y)\}_{r=0}^k$. See Simonis [109].

One may wonder if the method of generalizing and extending the weight enumerator can be continued, creating the generalized extended weight enumerator, in order to get a stronger invariant. The answer is no: the generalized extended weight enumerator can be defined, but does not contain more information then the three underlying polynomials.

It was shown by Gray [29] that the matroid of a code is a stronger invariant than its Tutte polynomial.

5.4.7 Exercises

5.4.1 Give a proof of the formulas in Example 5.4.6.

5.4.2 Give a proof of Remark 5.4.25.

5.4.3 Compute the two variable Möbius and coboundary polynomial of the simplex code $S_3(q)$.

5.5 Combinatorics and codes

5.5.1 Orthogonal arrays and codes

Definition 5.5.1 Let $q$ be a positive integer, not necessarily a power of a prime. A Latin square of order $q$ is a $q \times q$ array with entries from a set $Q$ of $q$ elements, such that every column and every row is a permutation of the symbols $Q$.

Example 5.5.2 An example of a Latin square of order 4 with $Q = \{a, b, c, d\}$ is given by

\[
\begin{array}{cccc}
  a & d & c & b \\
  d & a & b & c \\
  c & b & a & d \\
  b & c & d & a \\
\end{array}
\]

Remark 5.5.3 An alternative way to represent a Latin square is by a map $L : R \times C \rightarrow Q$, where $R$, $C$ and $Q$ are the sets of rows, columns and values, respectively, with all three of size $q$. Then $L$ represents a Latin square if and only if $L(x,j) = k$ has a unique solution $x \in R$, for all $j \in C$ and $k \in Q$, and $L(i,y) = k$ has a unique solution $y \in C$, for all $i \in R$ and $k \in Q$.

Any permutation of the rows, that is of the set $R$, gives another Latin square, and similarly permutations of the columns $C$ and the entries $Q$ give again Latin squares.

Example 5.5.4 Let $(G, \cdot)$ be a group where $\cdot$ is the multiplication on $G$. Let $R, C$ and $Q$ all three be equal to $G$. Let $L(x,y) = x \cdot y$. Then $L$ defines a Latin square of order $|G|$.
Remark 5.5.5 A pair of Greek-Latin squares.
Euler’s problem of 36 officers and the non-existence of two mutual orthogonal Latin squares of the order 6.

Definition 5.5.6 Two Latin squares $L_1$ and $L_2$ are called mutually orthogonal if $Q^2$ is equal to the set of all pairs $(L_1(x, y), L_2(x, y))$ with $x, y \in Q$. A collection \{\(L_i : i \in J\)\} of Latin squares $L_i$ of order $q$ with entries from a set $Q$, is called a set of mutually orthogonal Latin squares (MOLS) if $L_i$ and $L_j$ are mutual orthogonal for all $i, j \in J$ with $i \neq j$.

Example 5.5.7 Consider $Q = \mathbb{F}_q$ where $+$ is the addition. Let $L_a(x, y) = x + ay$. Then $L_a$ defines a Latin square of order $q$ for all $a \in \mathbb{F}_q^*$. Furthermore \{\(L_a : a \in \mathbb{F}_q^*\)\} form a collection of $q - 1$ MOLS of order $q$.

Example 5.5.8 In GAP one can constructs lists of MOLS. For example for $q = 7$ we can construct 6 MOLS:
\begin{verbatim}
> M:=MOLS(7,6);;
> M[1];
[ [ 0, 1, 2, 3, 4, 5, 6 ], [ 1, 2, 3, 4, 5, 6, 0 ], \]
[ 2, 3, 4, 5, 6, 0, 1 ], [ 3, 4, 5, 6, 0, 1, 2 ], \]
[ 4, 5, 6, 0, 1, 2, 3 ], [ 5, 6, 0, 1, 2, 3, 4 ], \]
[ 6, 0, 1, 2, 3, 4, 5 ]
\end{verbatim}

Definition 5.5.9 Let $n \geq 2$. An orthogonal array $OA(q,n)$ of order $q$ and depth $n$ is a $q^2 \times n$ array whose entries are from a set $Q$ of $q$ elements, such that for every two columns all $q^2$ pairs of symbols from $Q$ appear in exactly one row.

Remark 5.5.10 Let $J = \{1, 2, \ldots, j\}$. Let \{\(L_i : i \in J\)\} be a collection of $j$ MOLS of order $q$. Let $n = j + 2$. We can construct a $q^2 \times n$ orthogonal array as follows. Identify $R$ and $C$ with $Q$ by means of bijections. So we may assume that they are equal. In the first two columns all $q^2$ pairs of $Q^2$ are tabulated. If $(x, y)$ is in the row of the first two columns, then $L_i(x, y)$ is in the column $i + 2$ of the same row.
Conversely an $OA(q,n)$ gives rise to $n - 2$ MOLS of order $q$ if $n \geq 3$. In particular an $OA(q,3)$ is a Latin square and an $OA(q,4)$ corresponds to two mutual orthogonal Latin squares.

Example 5.5.11 Let $q$ be a power of a prime. Then a collection of $q - 1$ MOLS of order $q$ is constructed in Example 5.5.7. Therefore there exists an $OA(q,q + 1)$.

Remark 5.5.12 Let \{\(L_i : i \in J\)\} be a collection of $n - 2$ MOLS of order $q$ with an array $A$ the corresponding $OA(q,n)$. A permutation $\sigma$ of the rows $R$ gives a collection \{\(L'_i : i \in J\)\} of Latin squares which are again mutually orthogonal with a corresponding array $A_1$. Then $A_1$ is obtained from $A$ by permuting the symbols in the first column under $\sigma$ and leaving the remaining columns unchanged. Similarly, a permutation of the columns $C$ gives an array $A_2$ that is obtained from $A$ by permuting the symbols in the second column. A permutation of the entries from $Q$ of $L_i$ gives an array $A_{i+2}$ that is obtained from $A$ by permuting the symbols in the $(i + 2)$-th column.
CHAPTER 5. CODES AND RELATED STRUCTURES

Remark 5.5.13 Let $A$ be an $OA(q,n)$ with entries in $\mathbb{Q}$. Then two rows of $A$ are distinct and coincide in at most one position. Let $C$ be the subset of $\mathbb{Q}^n$ consisting of the rows of $A$. Then $C$ is a nonlinear code of length $n$ with $q^2$ codewords and minimum distance $n - 1$. So $C$ attains the Singleton bound of Exercise 3.2.1. Conversely any nonlinear $(n, q^2, n - 1)$ code yields an $OA(q, n)$.

The following proposition is a generalization of Proposition 4.4.25 in case $k = 2$, that is $n \leq q + 1$ if there exists an $[n, 2, n - 1]$ code over $\mathbb{F}_q$.

Proposition 5.5.14 Suppose there exists an orthogonal array $OA(q,n)$. Then $n \leq q + 1$.

Proof. Let $A$ be the array of an $OA(q,n)$. Choose an element in $\mathbb{Q}$ and denote it by 0. If the symbols in the the $i$-th column of $A$ are permuted, where the other columns remain unchanged, the new array is again $OA(q,n)$ by Remark 5.5.12. Therefore we may assume without loss of generality that the first row of $A$ consists of zeros. The distance between two rows is at least $n - 1$ by Remark 5.5.13. Hence apart form the first row, no other row contains two zeros. Next, it can be easily observed that each element from $\mathbb{Q}$ occurs in every column of $A$ exactly $q$ times. We leave this as an exercise for the reader.

Count the number of rows that contain one zero. This number is $n(q - 1)$. Indeed, zero should appear $n$ times in each column, but zero in the first column has already been counted. In addition, since the $i$-th row with $i > 1$, cannot have more than one zero, we see that all these zeros lie in different rows. So $1 + n(q - 1)$ is the number of rows that contain a zero, and this is at most $q^2$, the total number of rows. Therefore $n \leq q + 1$.

Remark 5.5.15 The bound of Proposition 5.5.14 is tight if $q$ is a power of a prime by Example 5.5.11.

Consider the following generalization of an orthogonal array.

Definition 5.5.16 An orthogonal array $OA(q,n,\lambda)$ is a $\lambda q^2 \times n$ array whose entries are from a set $\mathbb{Q}$ of $q$ elements, such that for every two columns any of $q^2$ pairs of symbols from $\mathbb{Q}$ occurs in exactly $\lambda$ rows. In particular $OA(q,n) = OA(q,n,1)$.

The next result we present here without a proof. It provides a lower bound on the value of $\lambda$ in terms of $q$ and $n$.

Theorem 5.5.17 If there exists an orthogonal array $OA(q,n,\lambda)$, then

$$\lambda \geq \frac{n(q - 1) + 1}{q^2}.$$

Proof. Reference: ***...**

Definition 5.5.18 An orthogonal array $OA_\lambda(t,n,q)$ is an $M \times n$ array, where $M = \lambda q^t$, whose entries are from a set $\mathbb{Q}$ of $q \geq 2$ elements, such that for every $M \times t$ subarray all $q^t$ possible $t$-tuples occur exactly $\lambda$ times as a row. The parameters $\lambda$, $t$, $n$, $q$ and $M$ are called the index, strength, constraints, levels and size, respectively. The orthogonal array is called linear if $\mathbb{Q} = \mathbb{F}_q$ and the rows of the array form an $\mathbb{F}_q$-linear subspace of $\mathbb{F}_q^n$.
Remark 5.5.19 An $OA(q,n,\lambda)$ is an orthogonal array of strength 2, that is $OA(q,n,\lambda) = OA_2(n,q)$. Notice that the order of $n$ and $q$ is interchanged according to the literature!!! should we adopt this convention too???

Theorem 5.5.20 The following objects correspond to each other:
1) An $F_q$-linear $[n,k,d]$ code,
2) A linear orthogonal array $OA_q(d-1,n,q)$, where $s = n - k + 1 - d$ is the Singleton defect of $C$.

Proof. Let $C$ be an $F_q$-linear $[n,k,d]$ code with Singleton defect $s = s(C) = n - k + 1 - d$. Consider the $q^{n-k} \times n$ matrix $A$ having as rows the codewords of $C^\perp$. Then $A$ is a linear $OA_q(d-1,n,q)$. *** ....*** ○

Remark 5.5.21 An $OA_1(n-k,n,q)$ is a nonlinear generalization of an $F_q$-linear MDS code of length $n$ and dimension $k$.

Consider the following a generalization of Corollary 4.4.27 on MDS codes.

Theorem 5.5.22 (Bush bound) Let $A$ be an $OA_1(k,n,q)$. If $q \leq k$, then $n \leq k + 1$.

Proof. ***... *** ○

5.5.2 Designs and codes

5.5.3 Exercises

5.5.1 Proof that Example 5.5.7 gives a set of $q - 1$ mutually orthogonal Latin squares of order $q$.

5.5.2 Let $q$ be positive integer. Show that $q - 1$ is the maximal number of MOLS of order $q$.

5.5.3 Show that there exist $t$ MOLS of order $qr$ if there exist $t$ MOLS of orders $q$ and $r$, respectively.

5.5.4 Let $n \geq 3$. Give a proof of the correspondence between an $OA(q,n)$ and $n - 2$ MOLS of order $q$ of Remark 5.5.10.

5.5.5 Let $A$ be the array of an $OA(q,n,\lambda)$ with entries from $Q$. Show that every symbol of $Q$ occurs in every column of $A$ exactly $\lambda q$ times.

5.5.6 Let $A$ be the array of an $OA_3(t,q,n)$ with entries from $Q$. Let $A'$ be obtained from $A$ by permuting the symbols in a given column and leaving the remaining columns unchanged. Show that $A'$ is the array of an $OA_3(t,q,n)$.

5.5.7 [CAS] Write two procedures:
- first takes as an input a $q \times q$ table and checks if the table is a Latin square. Check your procedure with IsLatinSquare in GAP;
- second given a list of $q \times q$ tables checks if they are MOLS. Use AreMOLS from GAP to test your procedure.
5.6 Notes

Section 4.1.6: MDS Conjecture is confirmed for all \( q \) such that \( 2 \leq q \leq 11 \), Blokhuis-Bruen-Thas, Hirschfeld-Storme.

Section 4.2:

Theory of arrangements of hyperplanes [92].

The use of the isomorphism in Proposition 4.5.18 for the proof of Theorem 4.5.21 was suggested in [109] by Simonis.

Proposition 4.5.20 first appears in [63, Theorem 3.2], although the term “generalized weight enumerator” was yet to be invented.

The identity of Lemma 4.5.22 one can find in [5, 27, 71, 128, 113].

Section 4.3:

Applications of GHW’s
***dimension/length profile, Forney***
***Wire-tap channel of type II***
***trellis complexity***

***\( p \)-th rank MDS, Kloeve,Simonis,Wei***

***Question: two var. wt enumerator determines the generalized wit enumerator?***

***\( C \) AMDS \( C^\perp \) AMDS iff \( d_2 = d_1 + 2.***

***If \( d > qs(C) \), then ..***

*** wt enumerator of AMDS code***

Section: 4.4:

Theory of lattices [38, ?].

The polynomial \( \mu_L(S,T) \) is defined by Zaslavsky in [139, Section 1]. In [140, Section 2], and [?, Section 6] it is called the Whitney polynomial. The polynomial \( \chi_L(S,T) \) is called the coboundary polynomial by Crapo in [42, p. 605] and [43]. See also [30, 32].

Blocking sets and codes meeting the Griesmer bound

mini-hypers, blocking sets and codes meeting the Griesmer bound

Belov, Hamada-Helleseth, Storme

Section 4.4.2: Corollary 4.3.25 was proved first Oberst and Dür [?], with the weaker assumption \( q^m > \binom{n-1}{d-1} - \binom{n-k-1}{d-1} \), where \( C \) is an \([n,k,d]\) code. Proposition 4.3.24 was shown by Pellikaan [?] with a stronger conclusion.
(Complete) \( n \)-arcs, ovals, Segre: an oval is \((q + 1)\)-arc if \( q \) is odd, ***B. Segre, conic, odd curve in char 2, nucleus***

Conjectures of Segre, Hirschfeld-Thas, Hirschfeld-Kochmaros-Torres pp. 599.

Section: 4.5:

Section 4.6:

Literature on (mutual orthogonal) Latin squares, orthogonal arrays, codes and designs:


P. Cameron and J.H. van Lint *Designs, graphs, codes and their links*. Pages: 14, 93, 170, 209.

Links between coding theory and statistical objects:


The construction of OA of max length and the Bush bound.


The notion of a \( OA_\lambda(t,q,n) \) as a generalization of MOLS is from:


***Bose-Bush,Bierbrauer, Stinson***

***t-resilient functions,

***The design of statistical experiments.

***Lattices and codes.***
Chapter 6

Complexity and decoding

Stanislav Bulygin, Ruud Pellikaan and Xin-Wen Wu

6.1 Complexity

In this section we briefly explain the theory of complexity and introduce some hard problems which are related to the theme of this book and will be useful in the following chapters.

6.1.1 Big-Oh notation

The following definitions and notations are essential in the evaluation of the complexity of an algorithm.

**Definition 6.1.1** Let $f(n)$ and $g(n)$ be functions mapping non-negative integers to real numbers. We define

1. $f(n) = O(g(n))$ for $n \to \infty$, if there exists a real constant $c > 0$ and an integer constant $n_0 > 0$ such that $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0$.

2. $f(n) = \Omega(g(n))$ for $n \to \infty$, if there exists a real constant $c > 0$ and an integer constant $n_0 > 0$ such that $0 \leq cg(n) \leq f(n)$ for all $n \geq n_0$.

3. $f(n) = \Theta(g(n))$ for $n \to \infty$, if there exist real constants $c_1 > 0$ and $c_2 > 0$, and an integer constant $n_0 > 0$ such that $c_1g(n) \leq f(n) \leq c_2g(n)$ for all $n \geq n_0$.

4. $f(n) \approx g(n)$ for $n \to \infty$, if $\lim_{n \to \infty} f(n)/g(n) = 1$.

5. $f(n) = o(g(n))$ for $n \to \infty$, if for every real constant $\epsilon > 0$ there exists an integer constant $n_0 > 0$ such that $0 \leq f(n) < \epsilon g(n)$ for all $n \geq n_0$.

**Remark 6.1.2** The notations $f(n) = O(g(n))$ and $f(n) = o(g(n))$ of Landau are often referred to as the “big-Oh” and “little-oh” notations. Furthermore $f(n) = \Omega(g(n))$ is expressed as “$f(n)$ is of the order $g(n)$”. Intuitively, this means that $f(n)$ grows no faster asymptotically than $g(n)$ up to a constant. And
$f(n) \approx g(n)$ is expressed as “$f(n)$ is approximately equal to $g(n)$”. Similarly, in the literature $f(n) = \Omega(g(n))$ and $f(n) = \Theta(g(n))$, are referred to as the “big-Omega”, “big-Theta”, notations, respectively.

**Example 6.1.3** It is easy to see that for every positive constant $a$, we have $a = \mathcal{O}(1)$ and $a/n = \mathcal{O}(1/n)$. Let $f(n) = a_k n^k + a_{k-1} n^{k-1} + \cdots + a_0$, where $k$ is an integer constant and $a_k, a_{k-1}, \ldots, a_0$ are real constants with $a_k > 0$. For this polynomial in $n$, we have $f(n) = \mathcal{O}(n^k)$, $f(n) = \Theta(n^k)$, $f(n) = a_k n^k$ and $f(n) = o(n^{k+1})$ for $n \to \infty$.

We have $2 \log n + 3 \log \log n = \mathcal{O}(\log n)$, $2 \log n + 3 \log \log n = \Theta(\log n)$ and $2 \log n + 3 \log \log n \approx 2 \log n$ for $n \to \infty$, since $2 \log n \leq 2 \log n + 3 \log \log n \leq 5 \log n$ when $n \geq 2$ and $\lim_{n \to \infty} \log n / \log n = 0$.

### 6.1.2 Boolean functions

An **algorithm** is a well-defined computational procedure such that every execution takes a variable input and halts with an output.

The complexity of an algorithm or a computational problem includes time complexity and storage space complexity.

**Definition 6.1.4** A (binary) **elementary (arithmetic) operation** is an addition, a comparison or a multiplication of two elements $x, y \in \{0, 1\} = \mathbb{F}_2$. Let $\mathcal{A}$ be an algorithm that has as input a binary word. Then the **time or work complexity** $C_T(\mathcal{A}, n)$ is the number of elementary operations in the algorithm $\mathcal{A}$ to get the output as a function of the length $n$ of the input, that is the number of bits of the input. The **space or memory complexity** $C_S(\mathcal{A}, n)$ is the maximum number of bits needed for memory during the execution of the algorithm with an input of $n$ bits. The complexity $C(\mathcal{A}, n)$ is the maximum of $C_T(\mathcal{A}, n)$ and $C_S(\mathcal{A}, n)$.

**Example 6.1.5** Let $C$ be a binary $[n, k]$ code given the generator matrix $G$. Then the encoding procedure

$$(a_1, \ldots, a_k) \mapsto (a_1, \ldots, a_k)G$$

is an algorithm. For every execution of the encoding algorithm, the input is a vector of length $k$ which represents a message block; the output is a codeword of length $n$. To compute one entry of a codeword one has to perform $k$ multiplications and $k - 1$ additions. The work complexity of this encoding is therefore $n(2k - 1)$. The memory complexity is $nk + k + n$: the number of bits needed to store the input vector, the matrix $G$ and the output codeword. Thus the complexity is dominated by work complexity and thus is $n(2k - 1)$.

**Example 6.1.6** In coding theory the code length is usually taken as a measure of an input size. In case of binary codes this coincides with the above complexity measures. For $q$-ary codes an element of $\mathbb{F}_q$ has a minimal binary representation by $\lceil \log(q) \rceil$ bits. A decoding algorithm with a received word of length $n$ as input can be represented by a binary word of length $N = n \lceil \log(q) \rceil$. In case the finite field is fixed there is no danger of confusion, but in case the efficiency of algorithms for distinct finite fields are compared, everything should be expressed in
terms of the number of binary elementary operations as a function of the length of the input as a binary string.

Let us see how this works out for solving a system of linear equations over a finite field. Whereas the addition and multiplication is counted for 1 unit in the binary case, this is no longer the case in the $q$-ary case. An addition in $\mathbb{F}_q$ is equal to $\lceil \log(q) \rceil$ binary elementary operations and multiplication needs $O(m^2 \log^2(p) + m \log^3(p)) = O((\log^3(q))$ elementary operations, where $q = p^m$ and $p$ is the characteristic of the finite field, see ???. The Gauss-Jordan algorithm to solve a system of $n$ linear equations in $n$ unknowns over a finite field $\mathbb{F}_q$ needs $O(n^3)$ additions and multiplications in $\mathbb{F}_q$. That means the binary complexity is $O(n^3 \log^3(q)) = O(N^3)$, where $N = n[\log(q)]$ is the length of the binary input. The known decoding algorithms that have polynomial complexity and that will be treated in the sequel reduce all to linear algebra computations, so they have complexity $O(n^3)$ elementary operations in $\mathbb{F}_q$ or $O(N^3)$ bit operations. So we will take the code length $n$ as a measure of the input size, and state the complexity as a function of $n$. These polynomial decoding algorithms apply to restricted classes of linear codes.

To study the theory of complexity, two different computational models which both are widely used in the literature are the Turing machine (TM) model and Boolean circuit model. Between these two models the Boolean circuit model has an especially simple definition and is viewed more amenable to combinatorial analysis. A Boolean circuit represents a Boolean function in a natural way. And Boolean functions have a lot of applications in the theory of coding. In this book we choose Boolean circuits as the computational model.

*** One of two paragraphs on Boolean Circuits vs. Turing Machines (c.f. R.B. Boppana & M. Sipser, ”The Complexity of Finite Function”) ***

The basic elements of a Boolean circuit are Boolean gates, namely, AND, OR, NOT, and XOR, which are defined by the following truth tables.

The truth table of AND (denoted by $\land$):

<table>
<thead>
<tr>
<th></th>
<th>F</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

The truth of OR (denoted by $\lor$):

<table>
<thead>
<tr>
<th></th>
<th>F</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

The truth table of NOT (denoted by $\neg$):

<table>
<thead>
<tr>
<th></th>
<th>F</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
</tbody>
</table>

The truth table of XOR:
It is easy to check that XOR gate can be represented by AND, OR and NOT as the following
\[ x \text{ XOR } y = (x \land (\neg y)) \lor ((\neg x) \land y). \]

The NAND operation is an AND operation followed by a NOT operation. The NOR operation is an OR operation followed by a NOT operation. In the following definition of Boolean circuits, we restrict to operations AND, OR and NOT.

Substituting F = 0 and T = 1, the Boolean gates above are actually operations on bits (called logical operations on bits). We have

\[ \wedge \text{ operation:} \]
\[
\begin{align*}
0 \wedge 0 &= 0 \\
0 \wedge 1 &= 0 \\
1 \wedge 0 &= 0 \\
1 \wedge 1 &= 1 \\
\end{align*}
\]

\[ \lor \text{ operation:} \]
\[
\begin{align*}
0 \lor 0 &= 0 \\
0 \lor 1 &= 1 \\
1 \lor 0 &= 1 \\
1 \lor 1 &= 1 \\
\end{align*}
\]

\[ \neg \text{ operation:} \]
\[
\begin{align*}
\neg 0 &= 1 \\
\neg 1 &= 0 \\
\end{align*}
\]

Consider the binary elementary arithmetic operations + and ·. It is easy to verify that
\[ x \cdot y = x \land y, \quad \text{and} \quad x + y = x \text{ XOR } y = (x \land (\neg y)) \lor ((\neg x) \land y). \]

**Definition 6.1.7** Given positive integers \( n \) and \( m \), a Boolean function is a function \( b : \{0, 1\}^n \to \{0, 1\}^m \). It is also called an \( n \)-input, \( m \)-output Boolean function and the set of all such functions is denoted by \( B(n, m) \). Denote \( B(n, 1) \) by \( B(n) \).

**Remark 6.1.8** The number of elements of \( B(n, m) \) is \( (2^n)^{2^m} = 2^{n \cdot 2^m} \). Identify \( \{0, 1\} \) with the binary field \( \mathbb{F}_2 \). Let \( b_1 \) and \( b_2 \) be elements of \( B(n, m) \). Then the sum \( b_1 + b_2 \) is defined by \( (b_1 + b_2)(x) = b_1(x) + b_2(x) \) for \( x \in \mathbb{F}_2^n \). In this way the set of Boolean functions \( B(n, m) \) is a vector space over \( \mathbb{F}_2 \) of dimension \( n \cdot 2^m \). Let \( b_1 \) and \( b_2 \) be elements of \( B(n) \). Then the product \( b_1 b_2 \) is defined by \( (b_1 b_2)(x) = b_1(x)b_2(x) \) for \( x \in \mathbb{F}_2^n \). In this way \( B(n) \) is an \( \mathbb{F}_2 \)-algebra with the property \( b^2 = b \) for all \( b \) in \( B(n) \).
6.1. COMPLEXITY

Every polynomial \( f(X) \) in \( \mathbb{F}_2[X_1, \ldots, X_n] \) yields a Boolean function \( \tilde{f} : \mathbb{F}_2^n \to \mathbb{F}_2 \) by evaluation: \( \tilde{f}(\mathbf{x}) = f(\mathbf{x}) \) for \( \mathbf{x} \in \mathbb{F}_2^n \). Consider the map
\[
ev : \mathbb{F}_2[X_1, \ldots, X_n] \longrightarrow B(n),
\]
defined by \( ev(f) = \tilde{f} \). Then \( ev \) is an algebra homomorphism. Now \( \tilde{X}_i^2 = \tilde{X}_i \) for all \( i \). Hence the ideal \( \langle X_1^2 + X_1, \ldots, X_n^2 + X_n \rangle \) is contained in the kernel of \( ev \). The factor ring and \( \mathbb{F}_2[X_1, \ldots, X_n]/(X_1^2 + X_1, \ldots, X_n^2 + X_n) \) and \( B(n) \) are both \( \mathbb{F}_2 \)-algebras of the same dimension \( n \). Hence \( ev \) induces an isomorphism
\[
ev : \mathbb{F}_2[X_1, \ldots, X_n]/(X_1^2 + X_1, \ldots, X_n^2 + X_n) \longrightarrow B(n).
\]

**Example 6.1.9** Let \( \text{sym}_k(x) \) be the Boolean function defined by the following polynomial in \( k^2 \) variables \( x_{ij}, 1 \leq i, j \leq k \),
\[
\text{sym}_k(x) = \prod_{i=1}^{k} \sum_{j=1}^{k} x_{ij}.
\]
Hence this description needs \( k(k-1) \) additions and \( k-1 \) multiplications. Therefore \( k^2 - 1 \) elementary operations are needed in total. If we would have written \( \text{sym}_k \) in normal form by expanding the products, the description is of the form
\[
\text{sym}_k(x) = \sum_{\sigma \in K^K} \prod_{i=1}^{k} x_{i\sigma(i)},
\]
where \( K^K \) is the set of all functions \( \sigma : \{1, \ldots, k\} \to \{1, \ldots, k\} \). This expression has \( k^k \) terms of products of \( k \) factors. So this needs \( (k-1)k^k \) multiplications and \( k^k - 1 \) additions. Therefore \( k^{k+1} - 1 \) elementary operations are needed in total. Hence this last description has exponential complexity.

**Example 6.1.10** Computing the binary determinant. Let \( \text{det}_k(x) \) be the Boolean function of \( k^2 \) variables \( x_{ij}, 1 \leq i, j \leq k \), that computes the determinant over \( \mathbb{F}_2 \) of the \( k \times k \) matrix \( x = (x_{ij}) \). Hence
\[
\text{det}_k(x) = \sum_{\sigma \in S_k} \prod_{i=1}^{k} x_{i\sigma(i)},
\]
where \( S_k \) is the symmetric group of \( k \) elements. This expression has \( k! \) terms of products of \( k \) factors. Therefore \( k(k!) - 1 \) elementary operations are needed in total.

Let \( \hat{x}_{ij} \) be the the square matrix of size \( k-1 \) obtained by deleting the \( i \)-th row and the \( j \)-th column from \( x \). Using the cofactor expansion
\[
\text{det}_k(x) = \sum_{j=1}^{k} x_{ij} \text{det}(\hat{x}_{ij}),
\]
we see that the complexity of this computation is of the order \( \mathcal{O}(k!) \). This complexity is still exponential. But \( \text{det}_k \) has complexity \( \mathcal{O}(k^3) \) by Gaussian elimination. This translates in a description of \( \text{det}_k \) as a Boolean function with \( \mathcal{O}(k^3) \) elementary operations.

***explicit description and worked out in and example in \( \text{det}_3 \).***
Example 6.1.11 A Boolean function computing whether an integer is prime or not. Let \( \text{prime}_m(x) \) be the Boolean function that is defined by

\[
\text{prime}_m(x_1, \ldots, x_m) = \begin{cases} 
1 & \text{if } x_1 + x_2 2 + \cdots + x_m 2^{m-1} \text{ is a prime}, \\
0 & \text{otherwise}.
\end{cases}
\]

So \( \text{prime}_2(x_1, x_2) = x_2 \) and \( \text{prime}_3(x_1, x_2, x_3) = x_2 + x_1 x_3 + x_2 x_3 \).

Only very recently it was proved that the decision problem whether an integer is prime or not, has polynomial complexity, see ??.


Remark 6.1.13 From these examples we see that the complexity of a Boolean function depends on the way we write it as a combination of elementary operations.

We can formally define the complexity of a Boolean function \( f \) in terms of the size of a circuit that represents the Boolean function.

Definition 6.1.14 A Boolean circuit is a directed graph containing no cycles (that is, if there is a route from any node to another node then there is no way back), which has the following structure:

(i) Any node (also called vertex) \( v \) has in-degree (that is, the number of edges entering \( v \)) equal to 0, 1 or 2, and the out-degree (that is, the number of edges leaving \( v \)) equal to 0 or 1.

(ii) Each node is labeled by one of AND, OR, NOT, 0, 1, or a variable \( x_i \).

(iii) If a node has in-degree 0, then it is called an input and is labeled by 0, 1, or a variable \( x_i \).

(iv) If a node has in-degree 1 and out-degree 1, then it is labeled by NOT.

(v) If a node has in-degree 2 and out-degree 1, then it is labeled by AND or OR.

In a Boolean circuit, any node with in-degree greater than 0 is called a gate. Any node with out-degree 0 is called an output.

Remark 6.1.15 By the definition, we observe that:

(1) A Boolean circuit can have more than one input and more than one output.

(2) Suppose a Boolean circuit has \( n \) variables \( x_1, x_2, \ldots, x_n \), and has \( m \) outputs, then it represents a Boolean function \( f : \{0,1\}^n \to \{0,1\}^m \) in a natural way.

(3) Any Boolean function \( f : \{0,1\}^n \to \{0,1\}^m \) can be represented by a Boolean circuit.
Definition 6.1.16 The size of a Boolean circuit is the number of gates that it contains. The depth of a Boolean circuit is the length of the longest path from an input to an output. For a Boolean function $f$, the time complexity of $f$, denoted by $C_T(f)$, is the smallest value of the sizes of the Boolean circuits representing $f$. The space complexity (also called depth complexity), denoted by $C_S(f)$ is the smallest value of the depths of the Boolean circuits representing $f$.

Theorem 6.1.17 (Shannon) Existence of a family of Boolean function of exponential complexity.

Proof. Let us first give a upper bound on the number of circuits with $n$ variables and size $s$; and then compare it with the number of Boolean functions of $n$ variables.

In a circuit of size $s$, each gate is assigned an AND or OR operator that on two previous nodes. Each previous node can either be a previous gate with at most $s$ choices, a literal (that is, a variable or its negation) with $2n$ choices, or a constant with 2 choices. Therefore, each gate has at most $2(s+2n+2)^2$ choices, which implies that the number of circuits with $n$ variables and size $s$ is at most $2^s(s + 2n + 2)^{2s}$. Now, setting $s = 2^n/(10n)$, the upper bound $2^s(s + 2n + 2)^{2s}$ is approximately $2^{2n/5} \ll 2^n$. On the other hand, the number of Boolean functions of $n$ variables and one output is $2^{2n}$. This implies that almost every Boolean function requires circuits of size larger than $2^n/(10n)$.

6.1.3 Hard problems

We now look at the classification of algorithms through the complexity.

Definition 6.1.18 Let $L_n(\alpha, a) = \mathcal{O}(\exp(a n^\alpha (\ln n)^{1-\alpha}))$, where $a$ and $\alpha$ are constants with $0 \leq a$ and $0 \leq \alpha \leq 1$. In particular $L_n(1, a) = \mathcal{O}(\exp(a n))$, and $L_n(0, a) = \mathcal{O}(\exp(a \ln n)) = \mathcal{O}(n^a)$. Let $A$ denote an algorithm with input size $n$. Then $A$ is an $L(\alpha)$-algorithm if the complexity of this algorithm has an estimate of the form $L_n(\alpha, a)$ for some $a$. An $L(0)$-algorithm is called a polynomial algorithm and an $L(1)$-algorithm is called an exponential algorithm. An $L(\alpha)$-algorithm is called a subexponential algorithm if $\alpha < 1$.

A problem that has either YES or NO as an answer is called a decision problem. All the computational problems that will be encountered here can be phrased as decision problems in such a way that an efficient algorithm for the decision problem yields an efficient algorithm for the computational problem, and vice versa. In the following complexity classes, we restrict our attention to decision problems.

Definition 6.1.19 The complexity class $\mathbf{P}$ is the set of all decision problems that are solvable in polynomial complexity.
**Definition 6.1.20** The complexity class \( \text{NP} \) is the set of all decision problems for which a YES answer can be verified in polynomial time given some extra information, called a certificate. The complexity class \( \text{co-NP} \) is the set of all decision problems for which a NO answer can be verified in polynomial time given an appropriate certificate.

**Example 6.1.21** Consider the decision problem that has as input a generator matrix of a code \( C \) and a positive integer \( w \), with question “\( d(C) \leq w \)?” In case the answer is yes, there exists a codeword \( c \) of minimum weight \( d(C) \). Then \( c \) is a certificate and the verification \( \text{wt}(c) \leq w \) has complexity \( n \).

**Definition 6.1.22** Let \( D_1 \) and \( D_2 \) be two computational problems. Then \( D_1 \) is said to *polytime reducible* to \( D_2 \), denoted as \( D_1 \leq_P D_2 \), provided that there exists an algorithm \( A_1 \) that solves \( D_1 \) which uses an algorithm \( A_2 \) that solves \( D_2 \), and \( A_1 \) runs in polynomial time if \( A_2 \) does. Informally, if \( D_1 \leq_P D_2 \), we say \( D_1 \) is no harder than \( D_2 \). If \( D_1 \leq_P D_2 \) and \( D_2 \leq_P D_1 \), then \( D_1 \) and \( D_2 \) are said to be *computationally equivalent*.

**Definition 6.1.23** A decision problem \( D \) is said to be \( \text{NP} \)-complete if

- \( D \in \text{NP} \), and
- \( E \leq_P D \) for every \( E \in \text{NP} \).

The class of all \( \text{NP} \)-complete problems is denoted by \( \text{NPC} \).

**Definition 6.1.24** A computational problem (not necessarily a decision problem) is \( \text{NP} \)-hard if there exists some \( \text{NP} \)-complete problem that polytime reduces to it.

Observe that every \( \text{NP} \)-complete problem is \( \text{NP} \)-hard. So the set of all \( \text{NP} \)-hard problems contains \( \text{NPC} \) as a subset. Some other relationships among the complexity classes above are illustrated as follows.

*****A Figure*****

It is natural to ask the following questions

1. Is \( P = \text{NP} \) ?
2. Is \( \text{NP} = \text{co-NP} \) ?
3. Is \( P = \text{NP} \cap \text{co-NP} \) ?

Most experts are of the opinion that the answer to each of these questions is NO. However no mathematical proofs are available, and to answer these questions is an interesting and hard problem in theoretical computer science.

**6.1.4 Exercises**

**6.1.1** Give an explicit expression of \( \text{det}_3(x) \) as a Boolean function.

**6.1.2** Give an explicit expression of \( \text{prime}_4(x) \) as a Boolean function.

**6.1.3** Give an explicit expression of \( \exp_a(x) \) as a Boolean function, where ....
6.2 Decoding

6.2.1 Decoding complexity

The known decoding algorithms that work for all linear codes have exponential complexity. Now we consider some of them.

**Remark 6.2.1** The brute force method compares the distance of a received word with all possible codewords, and chooses a codeword of minimum distance. The time complexity of the brute force method is at most \( nq^k \).

**Definition 6.2.2** Let \( r \) be a received word with respect to a code \( C \) of dimension \( k \). Choose an \((n-k) \times n\) parity check matrix \( H \) of the code \( C \). Then \( s = rH^T \in \mathbb{F}_q^{n-k} \) is called the syndrome of \( r \).

**Remark 6.2.3** Let \( C \) be a code of dimension \( k \). Let \( r \) be a received word. Then \( r + C \) is called the coset of \( r \). Now the cosets of the received words \( r_1 \) and \( r_2 \) are the same if and only if \( r_1H^T = r_2H^T \). Therefore there is a one to one correspondence between cosets of \( C \) and values of syndromes. Furthermore every element of \( \mathbb{F}_q^{n-k} \) is the syndrome of some received word \( r \), since \( H \) has rank \( n-k \). Hence the number of cosets is \( q^{n-k} \).

**Remark 6.2.4** In Definition 2.4.10 of coset leader decoding no mention is given of how this method is implemented. Coset leader decoding can be done in two ways. Let \( H \) be a parity check matrix and \( G \) a generator matrix of \( C \).

1) Preprocess a look-up table and store it in memory with a list of pairs \((s, e)\), where \( e \) is a coset leader of the coset with syndrome \( s \in \mathbb{F}_q^{n-k} \). Suppose a received word \( r \) is the input, compute \( s = rH^T \); look at the unique pair \((s, e)\) in the list with \( s \) as its first entry; give \( r - e \) as output.

2) For a received word \( r \), compute \( s = rH^T \); compute a solution \( e \) of minimal weight of the equation \( eH^T = s \); give \( r - e \) as output.

Now consider the complexity of the two methods for coset leader decoding:

1) The space complexity is clearly \( q^{n-k} \) the number of elements in the table. The time complexity is \( O(k^2(n-k)) \) for finding the solution \( c \). The preprocessing of the table has time complexity \( q^{n-k} \) by going through all possible error patterns \( e \) of non-decreasing weight and compute \( s = eH^T \). Put \((s, e)\) in the list if \( s \) is not already a first entry of a pair in the list.

2) Go through all possible error patterns \( e \) of non-decreasing weight and compute \( s = eH^T \) and compare it with \( rH^T \), where \( r \) is the received word. The first instance where \( eH^T = rH^T \) gives a closest codeword \( c = r - e \). The complexity is at most \( |B_\rho|n^2 \) for finding a coset leader, where \( \rho \) is the covering radius, by Remark 2.4.9.

**Example 6.2.5** **[7,4,3]** Hamming codes and other perfect codes, some small non perfect codes.

In order to compare their complexities we introduce the following definitions.

***work factor, memory factor***
**Definition 6.2.6** Let the complexity of an algorithm be exponential $O(q^n)$ for $n \to \infty$. Then $e$ is called the **complexity exponent** of the algorithm.

**Example 6.2.7** The complexity exponent of the brute force method is $R$ and of coset leader decoding is $1 - R$, where $R$ is the information rate.

6.2.2 Decoding erasures

After receiving a word there is a stage at the beginning of the decoding process where a decision has to be made about which symbol has been received. In some applications it is desirable to postpone a decision and to put a question mark "?" as a new symbol at that position, as if the symbol was erased. This is called an erasure. So a word over the alphabet $\mathbb{F}_q$ with erasures can be viewed as a word in the alphabet $\mathbb{F}_q \cup \{?\}$, that is an element of $(\mathbb{F}_q \cup \{?\})^n$. If only erasures occur and the number of erasures is at most $d - 1$, then we are sure that there is a unique codeword that agrees with the received word at all positions that are not an erasure.

**Proposition 6.2.8** Let $d$ be the minimum distance of a code. Then for every received word with $t$ errors and $s$ erasures such that $2t + s < d$ there is a unique nearest codeword. Conversely, if $d \leq 2t + s$ then there is a received word with at most $t$ errors and $s$ erasures with respect to more than one codeword.

**Proof.** This is left as an exercise to the reader. ⋄

Suppose that we have received a word with $s$ erasures and no errors. Then the brute force method would fill in all the possible $q^s$ words at the erasure positions and check whether the obtained word is a codeword. This method has complexity $O(n^2q^s)$, which is exponential in the number of erasures. In this section it is shown that correcting erasures only by solving a system of linear equations. This can be achieved by using the generator matrix or the parity check matrix. The most efficient choice depends on the rate and the minimum distance of the code.

**Proposition 6.2.9** Let $C$ be a code in $\mathbb{F}_q^n$ with parity check matrix $H$ and minimum distance $d$. Suppose that the codeword $c$ is transmitted and the word $r$ is received with no errors and at most $d - 1$ erasures. Let $J$ be the set of erasure positions of $r$. Let $y \in \mathbb{F}_q^n$ be defined by $y_j = r_j$ if $j \notin J$ and $y_j = 0$ otherwise. Let $s = yH^T$ be the syndrome of $y$. Let $e = y - c$. Then $w(e) < d$ and $e$ is the unique solution of the following system of linear equations in $x$:

$$xH^T = s \quad \text{and} \quad x_j = 0 \quad \text{for all} \quad j \notin J.$$  

**Proof.** By the definitions we have that

$$s = yH^T = cH^T + eH^T = 0 + eH^T = eH^T.$$
6.2. DECODING

The support of $e$ is contained in $J$. Hence $e_j = 0$ for all $j \not\in J$. Therefore $e$ is a solution of the system of linear equations.

If $x$ is another solution, then $(x - e)H^T = 0$. Therefore $x - e$ is an element of $C$, and moreover it is supported at $J$. So its weight is at most $d(C) - 1$. Hence it must be zero. Therefore $x = e$.

The above method of correcting the erasures only by means of a parity check matrix is called syndrome decoding up to the minimum distance.

**Definition 6.2.10** Let the complexity of an algorithm be $f(n)$ with $f(n) \approx cn^e$ for $n \to \infty$. Then the algorithm is called polynomial of degree $e$ with complexity coefficient $c$.

**Corollary 6.2.11** The complexity of correcting erasure only by means of syndrome decoding up to the minimum distance is polynomial of degree 3 and complexity coefficient $\frac{1}{2}(1 - R)^2\delta$ for a code of length $n \to \infty$, where $R$ is the information rate and $\delta$ the relative minimum distance.

**Proof.** This is consequence of Proposition 6.2.9 which amounts to solving a system of $n - k$ linear equations in at most $d - 1$ unknowns, in order to get the error vector $e$. Then $c = y - e$ is the codeword sent. We may assume that the encoding is done systematically at $k$ positions, so the message $m$ is immediately read off from these $k$ positions. The complexity is asymptotically of the order: $\frac{1}{3}(n - k)^2d = \frac{1}{3}(1 - R)^2\delta n$ for $n \to \infty$. See Appendix ??.

**Example 6.2.12** Let $C$ be the binary [7, 4, 3] Hamming code with parity check matrix given in Example 2.2.9. Suppose that $r = (1, 0, ?, ?, 0, 1, 0)$ is a received word with two erasures. Replace the erasures by zeros by $y = (1, 0, 0, 0, 0, 1, 0)$. The syndrome of $y$ is equal to $yH^T = (0, 0, 1)$. Now we want to solve the system of linear equations $xH^T = (1, 1, 0)$ and $x_i = 0$ for all $i \neq 3, 4$. Hence $x_3 = 1$ and $x_4 = 1$ and $c = (1, 0, 1, 1, 0, 1, 0)$ is the transmitted codeword.

**Example 6.2.13** Consider the MDS code $C_1$ over $F_{11}$ of length 11 and dimension 4 with generator matrix $G_1$ as given in Proposition 3.2.10 with $x_i = i \in F_{11}$ for $i = 1, \ldots, 11$. Let $C$ be the dual code of $C_1$. Then $C$ is a [11, 7, 5] code by Corollary 3.2.7, and $H = G_1$ is a parity check matrix for $C$ by Proposition 2.3.19. Suppose that we receive the following word with 4 erasures and no errors.

$$r = (1, 0, ?, 2, ?, 0, 0, 3, ?, ?, 0).$$

What is the sent codeword? Replacing the erasures by 0 gives the word

$$y = (1, 0, 0, 2, 0, 0, 0, 3, 0, 0, 0).$$

So $yH^T = (6, 0, 5, 4)$. Consider the linear system of equations given by the $4 \times 4$ submatrix of $H$ consisting of the columns corresponding to the erasure positions 3, 5, 9 and 10 together with the column $Hy^T$. 

$$
\begin{bmatrix}
1 & 1 & 1 & 1 & 6 \\
3 & 5 & 9 & 10 & 0 \\
9 & 3 & 4 & 1 & 5 \\
5 & 4 & 3 & 10 & 4
\end{bmatrix}
$$
After Gaussian elimination we see that \((0, 8, 9, 0)^T\) is the unique solution of this system of linear equations. Hence
\[
c = (1, 0, 0, 2, 3, 0, 0, 3, 2, 0, 0)
\]
is the codeword sent.

**Remark 6.2.14** Erasures only correction by means of syndrome decoding is efficient in case the information rate \(R\) is close to 1 and the relative minimum distance \(\delta\) is small, but cumbersome if \(R\) is small and \(\delta\) is close to 1. Take for instance the \([n, 1, n]\) binary repetition code. Any received word with \(n - 1\) erasures is readily corrected by looking at the remaining unerased position, if it is 0, then the all zero word was sent, and if it is 1, then the all one word was sent. With syndrome decoding one should solve a system of \(n - 1\) linear equations in \(n - 1\) unknowns.

The following method to correct erasures only uses a generator matrix of a code.

**Proposition 6.2.15** Let \(G\) be a generator matrix of an \([n, k, d]\) code \(C\) over \(\mathbb{F}_q\). Let \(m \in \mathbb{F}_q^k\) be the transmitted message. Let \(s\) be an integer such that \(s < d\). Let \(r\) be the received word with no errors and at most \(s\) erasures. Let \(I = \{j_1, \ldots, j_{n-s}\}\) be the subset of size \(n - s\) that is in the complement of the erasure positions. Let \(y \in \mathbb{F}_q^{n-s}\) be defined by \(y_j = r_{j_i}\) for \(i = 1, \ldots, n - s\). Let \(G'\) be the \(k \times (n-s)\) submatrix of \(G\) consisting of the \(n-s\) columns of \(G\) corresponding to the set \(I\). Then \(xG' = y\) has a unique solution \(m\), and \(mG\) is the codeword sent.

**Proof.** The Singleton bound 3.2.1 states that \(k \leq n - d + 1\). So \(k \leq n - s\). Now \(mG = c\) is the codeword sent and \(y_i = r_{j_i} = c_{j_i}\) for \(i = 1, \ldots, n-s\). Hence \(mG' = y\) and \(m\) is a solution. Now suppose that \(x \in \mathbb{F}_q^k\) satisfies \(xG' = y\), then \((m - x)G\) is a codeword that has a zero at \(n-s\) positions, so its weight is at most \(s < d\). So \((m-x)G\) is the zero codeword and \(xG' = mG'\). Hence \(m-x = 0\), since \(G\) has rank \(k\).

The above method is called correcting erasures only up to the minimum distance by means of the generator matrix.

**Corollary 6.2.16** The complexity of correcting erasures only up to the minimum distance by means of the generator matrix is polynomial of degree 3 and complexity coefficient \(R^2(1 - \delta - \frac{2}{3}R)\) for a code of length \(n \to \infty\), where \(R\) is the information rate and \(\delta\) the relative minimum distance.

**Proof.** This is consequence of Proposition 6.2.15. The complexity is that of solving a system of \(k\) linear equations in at most \(n - d + 1\) unknowns, which is asymptotically of the order: \((n - d - \frac{2}{3}k)k^2 = R^2(1 - \delta - \frac{2}{3}R)n^3\) for \(n \to \infty\). See Appendix ??.

***picture, comparison of G and H method**

**Example 6.2.17** Let \(C\) be the \([7, 2, 6]\) extended Reed-Solomon code over \(\mathbb{F}_7\) with generator matrix
\[
G = \begin{pmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 1 & 2 & 3 & 4 & 5 & 6
\end{pmatrix}
\]
6.2. DECODING

Suppose that \((?, 3, ?, ?, 4, ?)\) is a received word with no errors and 5 erasures. By means of the generator matrix we have to solve the following linear system of equations:

\[
\begin{align*}
    x_1 + x_2 &= 3 \\
    x_1 + 5x_2 &= 4
\end{align*}
\]

which has \((x_1, x_2) = (1, 2)\) as solution. Hence \((1, 2)G = (1, 3, 5, 0, 2, 4, 6)\) was the transmitted codeword. With syndrome decoding a system of 5 linear equations in 5 unknowns must be solved.

**Remark 6.2.18** For MDS codes we have asymptotically \(R \approx 1 - \delta\) and correcting erasures only by syndrome decoding and by a generator matrix has complexity coefficients \(\frac{1}{3}(1 - R)^3\) and \(\frac{1}{3}R^3\), respectively. Therefore syndrome decoding is preferred for \(R > 0.5\) and by a generator matrix if \(R < 0.5\).

6.2.3 Information and covering set decoding

The idea of this section is to decode by finding error-free positions in a received word, thus localizing errors. Let \(r\) be a received word written as \(r = c + e\), where \(c\) is a codeword from an \([n, k, d]\) code \(C\) and \(e\) is an error vector with the support \(\text{supp}(e)\). Note that if \(I\) is some information set (Definition 2.2.20) such that \(\text{supp}(e) \cap I = \emptyset\), then we are actually able to decode. Indeed, as \(\text{supp}(e) \cap I = \emptyset\), we have that \(r(I) = c(I)\) (Definition 3.1.2). Now if we denote by \(G\) the generator matrix of \(C\), then the submatrix \(G(I)\) can be transformed to the identity matrix \(Id_k\). Let \(G' = MG\), where \(M = G(I)^{-1}\), so that \(G'(I) = Id_k\), see Proposition 2.2.22. Thus a unique solution \(m \in \mathbb{F}_q^k\) of \(mG = c\), can be found as \(m = r(I)M\), because \(mG = r(I)MG = r(I)G'\) and the latter restricted to the positions of \(I\) yields \(r(I) = c(I)\). Now the algorithm, called information set decoding exploiting this idea is presented in Algorithm 6.1.

**Algorithm 6.1** Information set decoding

**Input:**
- Generator matrix \(G\) of an \([n, k]\) code \(C\)
- Received word \(r\)
- \(\mathcal{I}(C)\) a collection of all the information sets of a given code \(C\)

**Output:** A codeword \(c \in C\), such that \(d(r, c) = d(r, C)\)

**Begin**

\(c := 0;\)

**for** \(I \in \mathcal{I}(C)\) **do**

\(G' := G(I)^{-1}G\)

\(c' := r(I)G'\)

**if** \(d(c', r) < d(c, r)\) **then**

\(c = c'\)

**end if**

**end for**

**return** \(c\)

**End**

**Theorem 6.2.19** The information set decoding algorithm performs minimum distance decoding.
Proof. Let \( r = c + e \), where \( \text{wt}(e) = d(r, C) \). Let \( rH = eH = s \). Then \( e \) is a coset leader with the support \( E = \text{supp}(e) \) in the coset with the syndrome \( s \). It is enough to prove that there exists some information set disjoint with \( E \), or, equivalently, some check set (Definition 2.3.9) that contains \( E \). Consider an \( (n - k) \times |E| \) submatrix \( H(E) \) of the parity check matrix \( H \). As \( e \) is a coset leader, we have that for no other vector \( v \) in the same coset is \( \text{supp}(v) \subset E \).

Thus the subsystem of the parity check system defined by positions from \( E \) has a unique solution \( e(E) \). Otherwise it would be possible to find a solution with support a proper subset of \( E \). The above implies that \( \text{rank}(H(E)) = |E| \leq n - k \). Thus \( E \) can be expanded to a check set.

For a practical application it is convenient to choose the sets \( I \) randomly. Namely, we choose some \( k \)-subsets randomly in hope that after some reasonable amount of trials we encounter the one that is an information set and error-free.

Algorithm 6.2 Probabilistic information set decoding

\begin{algorithm}
Input:
- Generator matrix \( G \) of an \([n, k]\) code \( C \)
- Received word \( r \)
- Number of trials \( N_{\text{trials}}(n, k) \)

Output: A codeword \( c \in C \)

Begin
\begin{verbatim}
c := 0;
N_{\text{tr}} := 0;
repeat
  N_{\text{tr}} := N_{\text{tr}} + 1;
  Choose uniformly at random a subset \( I \) of \( \{1, \ldots, n\} \) of cardinality \( k \).
  if \( G_{(I)} \) is invertible then
    \( G' := G_{(I)}^{-1}G \)
    \( c' := r_{(I)}G' \)
    if \( d(c', r) < d(c, r) \) then
      \( c := c' \)
  end if
end if
until \( N_{\text{tr}} < N_{\text{trials}}(n, k) \)

return \( c \)
End
\end{verbatim}

We would like to estimate the complexity of the probabilistic information set decoding for the generic codes. Parameters of generic codes are computed in Theorem 3.3.6. We now use this result and notations to formulate the following result on complexity.

\textbf{Theorem 6.2.20} Let \( C \) be a generic \([n, k, d]\) \( q \)-ary code, with the dimension \( k = Rn, 0 < R < 1 \) and the minimum distance \( d = d_0 \), so that the covering radius is \( d_0(1 + o(1)) \). If \( N_{\text{trials}}(n, k) \) is at least

\[ \sigma \cdot n \cdot \frac{n}{d_0} \left( \frac{n - k}{d_0} \right), \]
6.2. DECODING

*** sigma is 1/pr(max matrix is invertible) add to Theorem 3.3.7*** then for large enough n the probabilistic information set decoding algorithm for the generic code C performs minimum distance decoding with negligibly small decoding error. Moreover the algorithm is exponential with complexity exponent

\[ CC_q(R) = (\log_2 2) \left( H_2(\delta_0) - (1 - R) H_2(\frac{\delta_0}{1 - R}) \right), \]  

(6.1)

where \( H_2 \) is the binary entropy function.

**Proof.** In order to succeed in the algorithm, we need that the set \( I \) chosen at a certain iteration is error-free and that the corresponding submatrix of \( G \) is invertible. The probability \( P(n,k,d_0) \) of this event is

\[ \frac{\binom{n - d_0}{k}}{\binom{n}{k}} \sigma_q(n) = \frac{\binom{n - k}{d_0}}{\binom{n}{d_0}} \sigma_q(n). \]

Therefore the probability that \( I \) fails to satisfy these properties is

\[ 1 - \frac{\binom{n - k}{d_0}}{\binom{n}{d_0}} \sigma_q(n). \]

Considering the assumption on \( N_{\text{trials}}(n,k) \) we have that probability of not-finding an error-free information set after \( N_{\text{trials}}(n,k) \) trials is

\[ (1 - P(n,k,d_0))^{n/P(n,k,d_0)} = O(e^{-n}), \]

which is negligible.

Next, due to the fact that determining whether \( G(l) \) is invertible and performing operations in the if-part have polynomial time complexity, we have that \( N_{\text{trials}}(n,k) \) dominates time complexity. Our task now is to give an asymptotic estimate of the latter. First, \( d_0 = \delta_0 n \), where \( \delta_0 = H_q^{-1}(1 - R) \), see Theorem 3.3.6. Then, using Stirling’s approximation \( \log_2 n! = n \log_2 n - n + o(n) \), we have

\[ n^{-1} \log_2 \left( \binom{n}{d_0} \right) = n^{-1} \left( n \log_2 n - d_0 \log_2 d_0 - (n - d_0) \log_2 (n - d_0) + o(n) \right) = \]
\[ = \log_2 n - \delta_0 \log_2(\delta_0 n) - (1 - \delta_0) \log_2((1 - \delta_0)n) + o(1) = H_2(\delta_0) + o(1). \]

Thus

\[ \log_q \left( \frac{n}{d_0} \right) = (nH_2(\delta_0) + o(n)) \log_2 q. \]

Analogously

\[ \log_q \left( \frac{n - k}{d_0} \right) = (n(1 - R)H_2\left( \frac{\delta_0}{1 - R} \right) + o(n)) \log_2 q, \]

where \( n - k = (1 - R)n \). Now

\[ \log_q N_{\text{trials}}(n,k) = \log_q n + \log_q \sigma + \log_q \left( \frac{n}{d_0} \right) + \log_q \left( \frac{n - k}{d_0} \right). \]

Considering that the first two summands are dominated by the last two, the claim on the complexity exponent follows. \( \diamond \)
If we depict complexity coefficient of the exhaustive search, syndrome decoding, and the probabilistic information set decoding, we see that the information set decoding is strongly superior to the former two, see Figure 6.1.

We may think of the above algorithms in a dual way using check sets instead of information sets and parity check matrices instead of generator matrices. The set of all check sets is closely related with the so-called covering systems, which we will consider a bit later in this section and which give the name for the algorithm.

\begin{algorithm}
\textbf{Algorithm 6.3} Covering set decoding
\begin{itemize}
\item Parity check matrix \( H \) of an \( [n,k] \) code \( C \)
\item Received word \( r \)
\item \( J(C) \) a collection of all the check sets of a given code \( C \)
\end{itemize}
\textbf{Input:}
A codeword \( c \in C \), such that \( d(r,c) = d(r,C) \)
\textbf{Begin}
\begin{itemize}
\item \( c := 0 \);
\item \( s := rH^T \);
\end{itemize}
\textbf{for} \( J \in J(C) \) \textbf{do}
\begin{itemize}
\item \( e' := s \cdot (H^{-1}(J))^T \);
\item Compute \( e \) such that \( e_{(J)} = e' \) and \( e_j = 0 \) for \( j \) not in \( J \);
\item \( c' := r - e \);
\item \textbf{if} \( d(c',r) < d(c,r) \) \textbf{then}
\item \( c = c' \)
\item \textbf{end if}
\end{itemize}
\textbf{end for}
\textbf{return} \( c \)
\textbf{End}
\end{algorithm}

\textbf{Theorem 6.2.21} The covering set decoding algorithm performs minimum dis-
6.2. DECODING

tance decoding.

Proof. Let \( r = c + e \) as in the proof of Theorem 6.2.19. From that proof we know that there exists a check set \( J \) such that \( \text{supp}(e) \subset J \). Now we have \( Hr^T = He^T = H(J)e(J) \). Since for the check set \( J \), the matrix \( H(J) \) is invertible, we may find \( e(J) \) and thus \( e \).

Similarly to Algorithm 6.1 one may define the probabilistic version of the covering set decoding. As we have already mentioned the covering set decoding algorithm is closely related to the notion of a covering system. The overview of this notion follows next.

Definition 6.2.22 Let \( n, l, t \) be integers such that \( 0 < t \leq l \leq n \). An \((n, l, t)\) covering system is a collection \( J \) of subsets \( J \) of \( \{1, \ldots, n\} \), such that every \( J \in J \) has \( l \) elements and every subset of \( \{1, \ldots, n\} \) of size \( t \) is contained in at least one \( J \in J \). The elements \( J \) of a covering system \( J \) are also called blocks. If a subset \( T \) of size \( t \) is contained in a \( J \in J \), then we say that \( T \) is covered or trapped by \( J \).

Remark 6.2.23 From the proofs of Theorem 6.2.21 for almost all codes it is enough to find a a collection \( J \) of subsets \( J \) of \( \{1, \ldots, n\} \), such that all \( J \in J \) have \( n - k \) elements and every subset of \( \{1, \ldots, n\} \) of size \( \rho = d_0 + o(1) \) is contained in at least one \( J \in J \), thus obtaining \((n, n - k, d_0)\) covering system.

Example 6.2.24 The collection of all subsets of \( \{1, \ldots, n\} \) of size \( l \) is an \((n, l, t)\) covering system for all \( 0 < t \leq l \). This collection consists of \( \binom{n}{l} \) blocks.

Example 6.2.25 Consider \( \mathbb{F}_q^2 \), the affine plane over \( \mathbb{F}_q \). Let \( n = q^2 \) be the number of its points. Then every line consists of \( q \) points, and every collection of two points is covered by exactly one line. Hence there exists a \((q^2, q, 2)\) covering system. Every line that is not parallel to the \( x \)-axis is given by a unique equation \( y = mx + c \). There are \( q^2 \) of such lines. And there are \( q \) lines parallel to the \( y \)-axis. So the total number of lines is \( q^2 + q \).

Example 6.2.26 Consider the projective plane over \( \mathbb{F}_q \) as treated in Section 4.3.1. Let \( n = q^2 + q + 1 \) be the number of its points. Then every line consists of \( q + 1 \) points, and every collection of two points is covered by exactly one line. There are \( q + 1 \) lines. Hence there exists a \((q^2 + q + 1, q + 1, 2)\) covering system consisting of \( q + 1 \) blocks.

Remark 6.2.27 The number of blocks of an \((n, l, t)\) covering system is considerably smaller than the number of all possible \( t \)-sets. It is still at least \( \binom{n}{t} / \binom{l}{t} \). But also this number grows exponentially in \( n \) if \( \lambda = \lim_{n \to \infty} t/n > 0 \) and \( \tau = \lim_{n \to \infty} t/n > 0 \).

Definition 6.2.28 The covering coefficient \( b(n, l, t) \) is the smallest integer \( b \) such that there is an \((n, l, t)\) covering system consisting of \( b \) blocks.

Although the exact value of the covering coefficient \( b(n, l, t) \) is an open problem we do know its asymptotic logarithmic behavior.
Proposition 6.2.29  Let \( \lambda \) and \( \tau \) be constants such that \( 0 < \tau < \lambda < 1 \). Then
\[
\lim_{n \to \infty} \frac{1}{n} \log b(n, \lfloor \lambda n \rfloor, \lfloor \tau n \rfloor) = H_2(\tau) - \lambda H_2(\tau/\lambda),
\]

Proof. *** I suggest to skip the proof ***
In order to establish this asymptotical result we prove lower and upper bounds, which are asymptotically identical. First the lower bound on \( b(n, l, t) \). Note that every \( l \)-tuple traps \( \binom{l}{t} \) \( t \)-tuples. Therefore, one needs at least
\[
\frac{n \binom{l}{t}}{\binom{n}{l}} \leq \frac{n}{n-l} \cdot \frac{n}{n-t} \cdot \cdots \cdot \frac{n}{n-t/2}.
\]
Now we use the relation from [?]:
\[
2^{n H_2(\tau) - o_+(n)} \leq \binom{n}{\lfloor \lambda n \rfloor} \leq 2^{n H_2(\tau)} \quad \text{for } 0 < \theta < 1,
\]
where \( o_+(n) \) is a non-negative function, such that \( o_+(n) = o(n) \). Applying this lower bound for \( l = \lfloor \lambda n \rfloor \) and \( t = \lfloor \tau n \rfloor \) we have
\[
\binom{n}{\lfloor \lambda n \rfloor} \geq \frac{n}{\lfloor \lambda n \rfloor} \cdot \frac{n}{\lfloor \tau n \rfloor} \geq 2^{n \lambda H_2(\tau/\lambda)}.
\]
For a similar lower bound see Exercise ??.
Now the upper bound. Consider a set \( S \) with \( f(n, l, t) \) independently and uniformly randomly chosen \( l \)-tuples, such that
\[
f(n, l, t) = \frac{\binom{n}{l} \cdot c n}{\binom{n}{l}}.
\]
where \( c > \ln 2 \). The probability that a \( t \)-tuple is not trapped by any tuple from \( S \) is
\[
\left( 1 - \frac{n-t}{n-l} \cdot \frac{n}{l} \right)^{f(n, l, t)}.
\]
Indeed, the number of all \( l \)-tuples is \( \binom{n}{l} \), the probability of trapping a given \( t \)-tuple \( T_1 \) by an \( l \)-tuple \( T_2 \) is the same as probability of trapping the complement of \( T_2 \) by the complement of \( T_1 \) and is equal to \( \binom{n-t}{l-t} / \binom{n-t}{l} \). The expected number of non-trapped \( t \)-tuples is then
\[
\frac{n}{l} \left( 1 - \frac{n-t}{n-l} \cdot \frac{n}{l} \right)^{f(n, l, t)}.
\]
Using the relation \( \lim_{x \to \infty} (1 - 1/x)^x = e^{-1} \) and the expression for \( f(n, l, t) \) we have that the expected number above tends to
\[
T = 2^{n H_2(t/n) + o(n) - c n \log e}
\]
From the condition on \( c \) we have that \( T < 1 \). This implies that among all the sets with \( f(n, l, t) \) independently and uniformly randomly chosen \( l \)-tuples, there exists one that traps all the \( t \)-tuples. Thus \( b(n, l, t) \leq f(n, l, t) \). By the well-known combinatorial identities
\[
\binom{n}{l} \binom{n-t}{n-l} = \binom{n}{l} \binom{n-t}{n-l} = \binom{n}{l} \binom{n}{l}.
\]
we have that for \( t = \lceil \tau n \rceil \) and \( l = \lceil \lambda n \rceil \) holds
\[
b(n, l, t) \leq 2^n \left( H_2(\tau) - \lambda H_2(\tau/\lambda) \right) + o(n),
\]
which asymptotically coincides with the lower bound proven above.

Let us now turn to the case of bounded distance decoding. So here we are aiming at correcting some \( t \) errors, where \( t < \rho \). The complexity result for almost all codes is obtained by substituting \( t/n \) instead of \( \delta_0 \) in (6.1). In particular, for decoding up to half the minimum distance for almost all codes we have the following result.

**Corollary 6.2.30** If \( N_{trials}(n, k) \) is at least
\[
n \cdot \binom{n}{d_0/2} / \binom{n-k}{d_0/2},
\]
then covering set decoding algorithm for almost all codes performs decoding up to half the minimum distance with negligibly small decoding error. Moreover the algorithm is exponential with complexity coefficient
\[
CSB_q(R) = (\log_2 2) \left( H_2(\delta_0/2) - (1 - R)H_2 \left( \frac{\delta_0}{2(1 - R)} \right) \right).
\]

We are interested now in bounded decoding up to \( t \leq d - 1 \). For almost all (long) codes the case \( t = d - 1 \) coincides with the minimum distance decoding, see... . From Proposition 6.2.9 it is enough to find a collection \( J \) of subsets \( J \) of \( \{1, \ldots, n\} \), such that all \( J \in J \) have \( d - 1 \) elements and every subset of \( \{1, \ldots, n\} \) of size \( t \) is contained in at least one \( J \in J \). Thus we need an \((n, d-1, t)\) covering system. Let us call this the *erasure set decoding*.

**Example 6.2.31** Consider a code of length 13, dimension 9 and minimum distance 5. The number of all 2-sets of \( \{1, \ldots, 13\} \) is equal to \( \binom{13}{2} = 78 \). In order to correct two errors one has to compute the linear combinations of two columns of a parity check matrix \( H \), for all the 78 choices of two columns, and see whether it is equal to \( rH^T \) for the received word \( r \).

An improvement can be obtained by a covering set. Consider the projective plane over \( F_3 \) as in Example 6.2.26. Hence we have a \((13, 4, 2)\) covering system. Using this covering system there are 13 subsets of 4 elements for which one has to find \( Hr^T \) as a linear combination of the corresponding columns of the parity check matrix. So we have to consider 13 times a system of 4 linear equations in 4 variables instead of 78 times a system of 4 linear equations in 2 variables.

From Proposition 6.2.29 and Remark ?? we have the complexity result for erasure set decoding.

**Proposition 6.2.32** Erasure set decoding performs bounded distance decoding for every \( t = \alpha \delta_0 n, 0 < \alpha \leq 1 \). The algorithm is exponential with complexity coefficient
\[
ES_q(R) = (\log_q 2)(H_2(\alpha \delta_0) - \delta_0 H_2(\alpha)).
\]

**Proof.** The proof is left to the reader as an exercise.

It can be shown, see Exercise 6.2.7, that erasure set decoding is interior to covering set for all \( \alpha \).

***Permutation decoding, Huffman-Pless 10.2, ex Golay \( q=3 \), exer \( q=2 \)***
6.2.4 Nearest neighbor decoding

decoding using minimal codewords.

6.2.5 Exercises

6.2.1 Count an erasure as half an error. Use this idea to define an extension of the Hamming distance on \((F_q \cup \{\cdot\})^n\) and show that it is a metric.

6.2.2 Give a proof of Proposition 6.2.8.

6.2.3 Consider the code \(C\) over \(F_{11}\) with parameters \([11, 7, 5]\) of Example 6.2.13. Suppose that we receive the word \((7, 6, 5, 4, 3, 2, 1, \cdot, \cdot, \cdot)\) with 4 erasures and no errors. Which codeword is sent?

6.2.4 Consider the code \(C_1\) over \(F_{11}\) with parameters \([11, 4, 8]\) of Example 6.2.13. Suppose that we receive the word \((4, 3, 2, 1, \cdot, \cdot, \cdot, \cdot, \cdot)\) with 7 erasures and no errors. Find the codeword sent.

6.2.5 Consider the covering systems of lines in the affine space \(F_q^m\) of dimension \(m\) over \(F_q\), and the projective space of dimension \(m\) over \(F_q\), respectively. Show the existence of a \((q, n, 2)\) and a \(((q^m + 1)/q - 1, q + 1, 2)\) covering system as in Examples 6.2.25 and 6.2.26 in the case \(m = 2\). Compute the number of lines in both cases.

6.2.6 Prove the following lower bound on \(b(n, l, t)\):

\[
b(n, l, t) \geq \left\lfloor \frac{n}{l} \left\lfloor \frac{n-1}{l-1} \right\rfloor \cdots \left\lfloor \frac{n-t+1}{l-t+1} \right\rfloor \right\rfloor.
\]

Hint: By double counting argument prove first that \(l \cdot b(n, l, t) \geq n \cdot b(n-1, l-1, t-1)\) and then use \(b(n, 1, 1) = \lceil n/l \rceil\).

6.2.7 By using the properties of binary entropy function prove that for all \(0 < R < 1\) and \(0 < \alpha < 1\) holds

\[
(1 - R)H_2\left(\frac{\alpha H_q^{-1}(1 - R)}{1 - R}\right) > H_q^{-1}(1 - R) \cdot H_2(\alpha).
\]

Conclude that covering set decoding is superior to erasure set.

6.3 Difficult problems in coding theory

6.3.1 General decoding and computing minimum distance

We have formulated the decoding problem in Section 6.2. As we have seen that the minimum (Hamming) distance of a linear code is an important parameter which can be used to estimate the decoding performance. However, a larger minimum distance does not guarantee the existence of an efficient decoding algorithm. It is natural to ask the following computational questions: For general linear codes, whether there exists a decoding algorithm with polynomial-time complexity? Whether or not there exists a polynomial-time algorithm which finds the minimum distance for any linear code? It has been proved that these
6.3. DIFFICULT PROBLEMS IN CODING THEORY

computational problems are both intractable.

Let $C$ be an $[n, k]$ binary linear code. Suppose $r$ is the received word. According to the maximum-likelihood decoding principle, we wish to find a codeword such that the Hamming distance between $r$ and the codeword is minimal. As we have seen in previous sections that using the brute force search, correct decoding requires $2^k$ comparisons in the worst case, and thus has exponential-time complexity.

Consider the syndrome of the received word. Let $H$ be a parity-check matrix of $C$, which is an $m \times n$ matrix, where $m = n - k$. The syndrome of $r$ is $s = rH^T$.

The following two computational problems are equivalent, letting $c = r - e$:

1. (Maximum-likelihood decoding problem) Finding a codeword $c$ such that $d(r, c)$ is minimal.

2. Finding a minimum-weight solution $e$ to the equation $xH^T = s$.

Clearly, an algorithm which solves the following computational problem (3) also solves the above Problem (2).

3. For any non-negative integer $w$, find a vector $x$ of Hamming weight $\leq w$ such that $xH^T = s$.

Conversely, an algorithm which solves Problem (2) must solve Problem (3). In fact, suppose $e$ is a minimum-weight solution $e$ to the equation $xH^T = s$. Then, for $w < \text{wt}(e)$, the algorithm will return “no solution”; for $w \geq \text{wt}(e)$, the algorithm returns $e$. Thus, the maximum-likelihood decoding problem is equivalent to the above problem (3).

The decision problem of the maximum-likelihood decoding problem is as

Decision Problem of Decoding Linear Codes

INSTANCE: An $m \times n$ binary matrix $H$, a binary vector $s$ of length $m$, and a non-negative integer $w$.

QUESTION: Is there a binary vector $x \in \mathbb{F}_2^n$ of Hamming weight $\leq w$ such that $xH^T = s$?

Proposition 6.3.1 The decision problem of decoding linear codes is an NP-complete problem.

We will prove this proposition by reducing the three-dimensional matching problem to the decision problem of decoding linear codes. The three-dimensional matching problem is a well-known NP-complete problem. For the completeness, we recall this problem as follows.

Three-Dimensional Matching Problem

INSTANCE: A set $T \subseteq S_1 \times S_2 \times S_3$, where $S_1$, $S_2$, and $S_3$ are disjoint finite sets having same number of elements, $a = |S_1| = |S_2| = |S_3|$.
QUESTION: Does $T$ contain a matching, that is, a subset $U \subseteq T$ such that $|U| = a$ and no two elements of $U$ agree in any coordinate?

We now construct a matrix $M$ which is called the incidence matrix of $T$ as follows. Fix an ordering of the triples of $T$. Let $t_i = (t_{i1}, t_{i2}, t_{i3})$ denote the $i$-th triple of $T$ for $i = 1, \ldots, |T|$. The matrix $M$ has $|T|$ rows and $3a$ columns. Each row $m_i$ of $M$ is a binary vector of length $3a$ and Hamming weight $3$, which is constituted of three blocks $b_{i1}$, $b_{i2}$ and $b_{i3}$ of the same length $a$, i.e., $m_i = (b_{i1}, b_{i2}, b_{i3})$. For $u = 1, 2, 3$, if $t_{iu}$ is the $v$ element of $S_u$, then the $v$-th coordinate of $b_{iu}$ is 1, all the other coordinates of this block is 0.

Clearly, the existence of a matching of the Three-Dimensional Matching Problem is equivalent to the existence of $a$ rows of $M$ such that their mod 2 sum is $(1, 1, \ldots, 1)$, that is, there exist a binary vector $x \in F_2^{|T|}$ of weight $a$ such that $xM = (1, 1, \ldots, 1) \in F_2^a$. Now we are ready to prove Proposition 6.3.1.

Proof of Proposition 6.3.1. Suppose we have a polynomial-time algorithm solving the Decision Problem of Decoding Linear Codes. Given an input $T \subseteq S_1 \times S_2 \times S_3$ for the Three-Dimensional Matching Problem, set $H = M^T$, where $M$ is the incidence matrix of $T$, $s = (1, 1, \ldots, 1)$ and $w = a$. Then running the algorithm for the Decision Problem of Decoding Linear Codes, we will discover whether or not there exist the desired matching. Thus, a polynomial-time algorithm for the Decision Problem of Decoding Linear Codes implies a polynomial-time algorithm for the Three-Dimensional Matching Problem. This proves the Decision Problem of Decoding Linear Codes is $NP$-complete.

Next, let us consider the problem of computing the minimum distance of an $[n,k]$ binary linear code $C$ with a parity-check matrix $H$. For any linear code, the minimum distance is equal to the minimum weight, we use these two terms interchangeably. Consider the following decision problem.

Decision Problem of Computing Minimum Distance

INSTANCE: An $m \times n$ binary matrix $H$ and a non-negative integer $w$.

QUESTION: Is there a nonzero binary vector $x$ of Hamming weight $w$ such that $xH^T = 0$?

If we have an algorithm which solves the above problem, then we can run the algorithm with $w = 1, 2, \ldots$, and the first integer $d$ with affirmative answer is the minimum weight of $C$. On the other hand, if we have an algorithm which finds the minimum weight $d$ of $C$, then we can solve the above problem by comparing $w$ with $d$. Therefore, we call this problem the Decision Problem of Computing Minimum Distance, and the $NP$-completeness of this problem implies the $NP$-hardness of the problem of computing the minimum distance.


6.3. DIFFICULT PROBLEMS IN CODING THEORY

***Computing the minimum distance:
- brute force, complexity \( (q^k - 1)/(q - 1) \), \( \mathcal{O}(q^k) \)
- minimal number of parity checks: \( \mathcal{O}(\binom{n}{k}^3) \)***

***Brouwer’s algorithm and variations, Zimmerman-Canteau-Chabeaud, Sala***

*** Vardy’s result: computing the min. dist. is NP hard***

6.3.2 Is decoding up to half the minimum distance hard?

Finding the minimum distance and decoding up to half the minimum distance are closely related problems.

**Algorithm 6.3.2** Suppose that \( A \) is an algorithm that computes the minimum distance of an \( \mathbb{F}_q \)-linear code \( C \) that is given by a parity check matrix \( H \). We define an algorithm \( D \) with input \( y \in \mathbb{F}_q^n \). Let \( s = Hy^T \) be the syndrome of \( y \) with respect to \( H \). Let \( \tilde{H} = [H|s] \) be the parity check matrix of the code \( \tilde{C} \) of length \( n + 1 \). Let \( \tilde{C}_i \) be the code that is obtained by puncturing \( \tilde{C} \) at the \( i \)-th position. Use algorithm \( A \) to compute \( d(C) \) and \( d(C_i) \) for \( i \leq n \). Let \( t = \min\{d(C_i)|i \leq n\} \). Let \( I = \{i|t = d(C_i), i \leq n\} \). Assume \( |I| = t \) and \( t < d(C) \). Assume furthermore that erasure decoding at the positions \( I \) finds a unique codeword \( c \) in \( C \) such that \( c_i = y_i \) for all \( i \) not in \( I \). Output \( c \) in case the above assumptions are met, and output * otherwise.

**Proposition 6.3.3** Let \( A \) be an algorithm that computes the minimum distance. Let \( D \) be the algorithm that is defined in 6.3.2. Let \( y \in \mathbb{F}_q^n \) be an input. Then \( D \) is a decoder that gives as output \( c \) in case \( d(C, y) < d(C) \) and \( y \) has \( c \) as unique nearest codeword. In particular \( D \) is a decoder of \( C \) that corrects up to half the minimum distance.

**Proof.** Let \( y \) be a word with \( t = d(C, y) < d(C) \) and suppose that \( c \) is a unique nearest codeword. Then \( y = c + e \) with \( e \in C \) and \( t = wt(e) \). Note that \( (e, -1) \in \tilde{C} \), since \( s = H y^T = He^T \). So \( d(C) \leq t + 1 \). Let \( \tilde{z} \) be in \( \tilde{C} \). If \( \tilde{z}_{n+1} = 0 \), then \( \tilde{z} = (\tilde{z}, 0) \) with \( z \in C \). Hence \( wt(\tilde{z}) \geq d(C) \geq t + 1 \). If \( \tilde{z}_{n+1} \neq 0 \), then without loss of generality we may assume that \( \tilde{z} = (z, -1) \). So \( \tilde{H} \tilde{z}^T = 0 \). Hence \( H\tilde{z}^T = s \). So \( c' = y - z \in C \). If \( wt(\tilde{z}) \leq t + 1 \), then \( wt(z) \leq t \). So \( d(y, c') \leq t \). Hence \( c' = c \), since \( c \) is the unique nearest codeword by assumption. Therefore \( z = e \) and \( wt(z) = t \). Hence \( d(C) = t + 1 \), since \( t + 1 \leq d(C) \).

Let \( \tilde{C}_i \) be the code that is obtained by puncturing \( \tilde{C} \) at the \( i \)-th position. Use the algorithm \( A \) to compute \( d(C_i) \) for all \( i \leq n \). An argument similar to above shows that \( d(C_i) = t \) if \( i \) is in the support of \( e \), and \( d(C_i) = t + 1 \) if \( i \) is not in the support of \( e \). So \( t = \min\{d(C_i)|i \leq n\} \) and \( I = \{i|t = d(C_i), i \leq n\} \) is the support of \( e \) and has size \( t \). So the error positions are known. Computing the error values is matter of linear algebra as shown in Proposition 6.2.11. In this way \( e \) and \( c \) are found. 

**Proposition 6.3.4** Let \( MD \) be the problem of computing the minimum distance of a code given by a parity check matrix. Let \( DHMD \) be the problem of decoding up to half the minimum distance. Then

\[
DHMD \leq_P MD.
\]
Proof. Let $\mathcal{A}$ be an algorithm that computes the minimum distance of an $\mathbb{F}_q$-linear code $C$ that is given by a parity check matrix $H$. Let $\mathcal{D}$ be the algorithm given in 6.3.2. Then $\mathcal{A}$ is used $(n+1)$-times in $\mathcal{D}$. Suppose that the complexity of $\mathcal{A}$ is polynomial of degree $e$. We may assume that $e \geq 2$. Computing the error values can be done with complexity $O(n^3)$ by Proposition 6.2.11. Then the complexity of $\mathcal{D}$ is polynomial of degree $e + 1$. ⋄

***Sendrier and Finasz***

***Decoding with preprocessing, Bruck-Naor***

### 6.3.3 Other hard problems

***worse case versus average case, the simplex method for linear programming is an example of an algorithm that runs almost always fast, that is polynomially in its input, but for which is known to be exponentially in the worst case. Ellipsoid method, Khachian’s method***

***approximate solutions of NP-hard problems***

### 6.4 Notes

In 1978, Berlekamp, McEliece and van Tilborg proved that the maximum-likelihood decoding problem is NP-hard for general binary codes. Vardy showed in 1997 that the problem of computing the minimum distance of a binary linear code is NP-hard.
Chapter 7

Cyclic codes

Ruud Pellikaan

Cyclic codes have been in the center of interest in the theory of error-correcting codes since their introduction. Cyclic codes of relatively small length have good parameters. In the list of 62 binary cyclic codes of length 63 there are 51 codes that have the largest known minimum distance for a given dimension among all linear codes of length 63. Binary cyclic codes are better than the Gilbert-Varshamov bound for lengths up to 1023. Although some negative results are known indicating that cyclic codes are asymptotically bad, this still is an open problem. Rich combinatorics is involved in the determination of the parameters of cyclic codes in terms of patterns of the defining set.

7.1 Cyclic codes

7.1.1 Definition of cyclic codes

Definition 7.1.1 The cyclic shift $\sigma(c)$ of a word $c = (c_0, c_1, \ldots, c_{n-1}) \in \mathbb{F}_q^n$ is defined by

$$\sigma(c) := (c_{n-1}, c_0, c_1, \ldots, c_{n-2}).$$

An $\mathbb{F}_q$-linear code $C$ of length $n$ is called cyclic if

$$\sigma(e) \in C \text{ for all } e \in C.$$

The subspaces $\{0\}$ and $\mathbb{F}_q^n$ are clearly cyclic and are called the trivial cyclic codes.

Remark 7.1.2 In the context of cyclic codes it is convenient to consider the index $i$ of a word modulo $n$ and the convention is that the numbering of elements $(c_0, c_1, \ldots, c_{n-1})$ starts with 0 instead of 1. The cyclic shift defines a linear map $\sigma : \mathbb{F}_q^n \rightarrow \mathbb{F}_q^n$. The $i$-fold composition $\sigma^i = \sigma \circ \cdots \circ \sigma$ is the $i$-fold forward shift. Now $\sigma^n$ is the identity map and $\sigma^{n-1}$ is the backward shift. A cyclic code is invariant under $\sigma^i$ for all $i$. 

189
Proposition 7.1.3 Let $G$ be a generator matrix of a linear code $C$. Then $C$ is cyclic if and only if the cyclic shift of every row of $G$ is in $C$.

Proof. If $C$ is cyclic, then the cyclic shift of every row of $G$ is in $C$, since all the rows of $G$ are codewords. Conversely, suppose that the cyclic shift of every row of $G$ is in $C$. Let $g_1, \ldots, g_k$ be the rows of $G$. Let $c \in C$. Then $c = \sum_{i=1}^{k} x_i g_i$ for some $x_1, \ldots, x_k \in \mathbb{F}_q$. Now $\sigma$ is a linear transformation of $\mathbb{F}_q^n$. So $\sigma(c) = \sum_{i=1}^{k} x_i \sigma(g_i) \in C$, since $C$ is linear and $\sigma(g_i) \in C$ for all $i$ by assumption. Hence $C$ is cyclic. □

Example 7.1.4 Consider the $[6,3]$ code over $\mathbb{F}_7$ with generator matrix $G$ defined by

$$G = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 3 & 2 & 6 & 4 & 5 \\ 1 & 2 & 4 & 1 & 2 & 4 \end{pmatrix}.$$ 

Then $\sigma(g_1) = g_1$, $\sigma(g_2) = 5g_2$ and $\sigma(g_3) = 4g_3$. Hence the code is cyclic.

Example 7.1.5 Consider the $[7,4,3]$ Hamming code $C$, with generator matrix $G$ as given in Example 2.2.14. Then $(0, 0, 0, 1, 0, 1, 1)$, the cyclic shift of the third row is not a codeword. Hence this code is not cyclic. After a permutation of the columns and rows of $G$ we get the generator matrix $G'$ of the code $C'$, where

$$G' = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}.$$ 

Let $g_i'$ be the $i$-th row of $G'$. Then $\sigma(g_1') = g_2'$, $\sigma(g_2') = g_3'$, $\sigma(g_3') = g_4'$ and $\sigma(g_4') = g_1'$. Hence $C'$ is cyclic by Proposition 7.1.3. Therefore $C$ is not cyclic, but equivalent to a cyclic code $C'$.

Proposition 7.1.6 The dual of a cyclic code is again cyclic.

Proof. Let $C$ be a cyclic code. Then $\sigma(c) \in C$ for all $c \in C$. So

$$\sigma^{n-1}(c) = (c_1, \ldots, c_{n-1}, c_0) \in C$$ for all $c \in C$.

Let $x \in C^\perp$. Then

$$\sigma(x) \cdot c = x_{n-1}c_0 + x_0c_1 + \cdots + x_{n-2}c_{n-1} = x \cdot \sigma^{n-1}(c) = 0$$

for all $c \in C$. Hence $C^\perp$ is cyclic. □
7.1. CYCLIC CODES

7.1.2 Cyclic codes as ideals

The set of all polynomials in the variable $X$ with coefficients in $\mathbb{F}_q$ is denoted by $\mathbb{F}_q[X]$. Two polynomials can be added and multiplied and in this way $\mathbb{F}_q[X]$ is a ring. One has division with rest this means that very polynomial $f(X)$ has after division with another nonzero polynomial $g(X)$ a quotient $q(X)$ with rest $r(X)$ that is zero or of degree strictly smaller than $\deg g(X)$. In other words

$$f(X) = q(X)g(X) + r(X) \text{ and } r(X) = 0 \text{ or } \deg r(X) < \deg g(X).$$

In this way $\mathbb{F}_q[X]$ with its degree is a Euclidean domain. Using division with rest repeatedly we find the greatest common divisor $\gcd(f(X), g(X))$ of two polynomials $f(X)$ and $g(X)$ by the algorithm of Euclid.

Remark 7.1.8

Every nonempty subset of a ring that is closed under addition and multiplication by an arbitrary element of the the ring is called an ideal. Let $g_1, \ldots, g_m$ be given elements of a ring. The set of all $a_1g_1 + \cdots + a_mg_m$ with $a_1, \ldots, a_m$ in the ring, forms an ideal and is denoted by $\langle g_1, \ldots, g_m \rangle$ and is called the ideal generated by $g_1, \ldots, g_m$. As a consequence of division with rest every ideal in $\mathbb{F}_q[X]$ is either $\{0\}$ or generated by a one unique monic polynomial. Furthermore

$$\langle f(X), g(X) \rangle = \langle \gcd(f(X), g(X)) \rangle.$$

We refer for these notions and properties to Appendix ??.

Definition 7.1.7

Let $R$ be a ring and $I$ an ideal in $R$. Then $R/I$ is the factor ring of $R$ modulo $I$. If $R = \mathbb{F}_q[X]$ and $I = \langle X^n-1 \rangle$ is the ideal generated by $X^n-1$, then $\mathbb{C}_{q,n}$ is the factor ring

$$\mathbb{C}_{q,n} = \mathbb{F}_q[X]/\langle X^n - 1 \rangle.$$

Remark 7.1.8

The factor ring $\mathbb{C}_{q,n}$ has an easy description. Every polynomial $f(X)$ has after division by $X^n-1$ a rest $r(X)$ of degree at most $n-1$, that is there exist polynomials $q(X)$ and $r(X)$ such that

$$f(X) = q(X)(X^n-1) + r(X) \text{ and } \deg r(X) < n \text{ or } r(X) = 0.$$

The coset of the polynomial $f(X)$ modulo $X^n-1$ is denoted by $f(x)$. Hence $f(X)$ and $r(X)$ have the same coset and represent the same element in $\mathbb{C}_{q,n}$. Now $x^i$ denotes the coset of $X^i$ modulo $(X^n-1)$. Hence the cosets $1, x, \ldots, x^{n-1}$ form a basis of $\mathbb{C}_{q,n}$ over $\mathbb{F}_q$. The multiplication of the basis elements $x^i$ and $x^j$ in $\mathbb{C}_{q,n}$ with $0 \leq i, j < n$ is given by

$$x^i x^j = \begin{cases} x^{i+j} & \text{if } i + j < n, \\ x^{i+j-n} & \text{if } i + j \geq n. \end{cases}$$

Definition 7.1.9

Consider the map $\varphi$ between $\mathbb{F}_q^n$ and $\mathbb{C}_{q,n}$

$$\varphi(c) = c_0 + c_1 x + \cdots + c_{n-1} x^{n-1}.$$

Then $\varphi(c)$ is also denoted by $c(x)$. 

Proposition 7.1.10 The map \( \varphi \) is an isomorphism of vector spaces. Ideals in the ring \( \mathbb{C}_{q,n} \) correspond one-to-one to cyclic codes in \( \mathbb{F}_q^n \).

Proof. The map \( \varphi \) is clearly linear and it maps the \( i \)-th standard basis vector of \( \mathbb{F}_q^n \) to the coset \( x^{i-1} \) in \( \mathbb{C}_{q,n} \) for \( i = 1, \ldots, n \). Hence \( \varphi \) is an isomorphism of vector spaces. Let \( \psi \) be the inverse map of \( \varphi \).

Let \( I \) be an ideal in \( \mathbb{C}_{q,n} \). Let \( C := \psi(I) \). Then \( C \) is a linear code, since \( \psi \) is a linear map. Let \( c \in C \). Then \( c(x) = \varphi(c) \in I \) and \( I \) is an ideal. So \( xc(x) \in I \).

But
\[
xc(x) = c_0x + c_1x^2 + \cdots + c_{n-2}x^{2n-1} + c_{n-1}x^n = c_{n-1} + c_0x + c_1x^2 + \cdots + c_{n-2}x^{n-1},
\]
since \( x^n = 1 \). So \( \varphi(xc(x)) = (c_{n-1}, c_0, \ldots, c_{n-2}) \in C \). Hence \( C \) is cyclic.

Conversely, let \( C \) be a cyclic code in \( \mathbb{F}_q^n \) and let \( I := \varphi(C) \). Then \( I \) is closed under addition of its elements, since \( C \) is a linear code and \( \varphi \) is a linear map. If \( a \in \mathbb{F}_q^n \) and \( c \in C \), then
\[
a(x)c(x) = \varphi(a_0c + a_1\sigma(c) + \cdots + a_{n-1}\sigma^{n-1}(c)) \in I.
\]
Hence \( I \) is an ideal in \( \mathbb{C}_{q,n} \).

In the following we will not distinguish between words and the corresponding polynomials under \( \varphi \); we will talk about words \( c(x) \) when in fact we mean the vector \( c \) and vice versa.

Example 7.1.11 Consider the rows of the generator matrix \( G' \) of the \([7, 4, 3]\) Hamming code of Example 7.1.5. They correspond to \( g'_1(x) = 1 + x^4 + x^5 \), \( g'_2(x) = x + x^5 + x^6 \), \( g'_3(x) = x^2 + x^4 + x^5 + x^6 \) and \( g'_4(x) = x^3 + x^4 + x^6 \), respectively. Furthermore \( x \cdot x^6 = 1 \), so \( x \) is invertible in the ring \( \mathbb{F}_2[X]/(X^7 - 1) \).

Now
\[
\langle 1 + x^4 + x^5 \rangle = \langle x + x^5 + x^6 \rangle = \langle x^6 + x^{10} + x^{11} \rangle = \langle x^3 + x^4 + x^6 \rangle.
\]
Hence the ideals generated by \( g'_i(x) \) are the same for \( i = 1, 2, 4 \) and there is no unique generating element. The third row generates the ideal
\[
\langle x^2 + x^4 + x^5 + x^6 \rangle = \langle x^2(1 + x^2 + x^3 + x^4) \rangle = \langle 1 + x^2 + x^3 + x^4 \rangle = \langle (1 + x)(1 + x + x^2) \rangle,
\]
which gives a cyclic code that is a proper subcode of dimension 3. Therefore all except the third element generate the same ideal.

7.1.3 Generator polynomial

Remark 7.1.12 The ring \( \mathbb{F}_q[X] \) with its degree function is an Euclidean ring.

Hence \( \mathbb{F}_q[X] \) is a principal ideal domain, that means that all ideals are generated by one element. If an ideal of \( \mathbb{F}_q[X] \) is not zero, then a generating element is unique up to a nonzero scalar multiple of \( \mathbb{F}_q \). So there is a unique monic polynomial generating the ideal. Now \( \mathbb{C}_{q,n} \) is a factor ring of \( \mathbb{F}_q[X] \), therefore it is also a principal ideal domain. A cyclic code \( C \) considered as an ideal in \( \mathbb{C}_{q,n} \) is generated by one element, but this element is not unique, as we have seen in Example 7.1.11. The inverse image of \( C \) under the map \( \mathbb{F}_q[X] \to \mathbb{C}_{q,n} \) is denoted.
7.1. CYCLIC CODES

by \( I \). Then \( I \) is a nonzero ideal in \( \mathbb{F}_q[X] \) containing \( X^n - 1 \). Therefore \( I \) has a
unique monic polynomial \( g(X) \) as generator. So \( g(X) \) is the monic polynomial
in \( I \) of minimal degree. Hence \( g(X) \) is the monic polynomial of minimal degree
such that \( g(x) \in C \).

Definition 7.1.13 Let \( C \) be a cyclic code. Let \( g(X) \) be the monic polynomial
of minimal degree such that \( g(x) \in C \). Then \( g(X) \) is called the generator
polynomial of \( C \).

Example 7.1.14 The generator polynomial of the trivial code \( \mathbb{F}_q^n \) is 1, and of
the zero code of length \( n \) is \( X^n - 1 \). The repetition code and its dual have as
generator polynomials \( X^{n-1} + \cdots + X + 1 \) and \( X - 1 \), respectively.

Proposition 7.1.15 Let \( g(X) \) be a polynomial in \( \mathbb{F}_q[X] \). Then \( g(X) \) is a generator polynomial of a cyclic code over \( \mathbb{F}_q \) of length \( n \) if and only if \( g(X) \) is
monic and divides \( X^n - 1 \).

Proof. Suppose \( g(X) \) is the generator polynomial of a cyclic code. Then \( g(X) \)
is monic and a generator of an ideal in \( \mathbb{F}_q[X] \) that contains \( X^n - 1 \). Hence \( g(X) \)
divides \( X^n - 1 \).
Conversely, suppose that \( g(X) \) is monic and divides \( X^n - 1 \). So \( b(X)g(X) = X^n - 1 \) for some \( b(X) \). Now \( \langle g(x) \rangle \) is an ideal in \( \mathbb{C}_{q,n} \) and defines a cyclic code
\( C \). Let \( c(X) \) be a monic polynomial such that \( c(x) \in C \). Then \( c(x) = a(x)g(x) \).
Hence there exists an \( h(X) \) such that
\[
c(X) = a(X)g(X) + h(X)(X^n - 1) = (a(X) + b(X)h(X))g(X)
\]
Hence \( \deg g(X) \leq \deg c(X) \). Therefore \( g(X) \) is the monic polynomial of minimal
degree such that \( g(x) \in C \). Hence \( g(X) \) is the generator polynomial of \( C \). ☐

Example 7.1.16 The polynomial \( X^3 + X + 1 \) divides \( X^8 - 1 \) in \( \mathbb{F}_3[X] \), since
\[
(X^3 + X + 1)(X^5 - X^3 - X^2 + X - 1) = X^8 - 1.
\]
Hence \( 1 + X + X^3 \) is a generator polynomial of a ternary cyclic code of length
8.

Remark 7.1.17 Let \( g(X) \) be the generator polynomial of \( C \). Then \( g(X) \) is a
monic polynomial and \( g(x) \) generates \( C \). Let \( c(X) \) be another polynomial such
that \( c(x) \) generates \( C \). Let \( d(X) \) be the greatest common divisor of \( c(X) \) and
\( X^n - 1 \). Then \( d(X) \) is the monic polynomial such that
\[
\langle d(X) \rangle = \langle c(X), X^n - 1 \rangle = I.
\]
But also \( g(X) \) is the unique monic polynomial such that \( \langle g(X) \rangle = I \). Hence
\( g(X) = \gcd(c(X), X^n - 1) \).

Example 7.1.18 Consider the binary cyclic code of length 7 and generated by
\( 1 + x^2 \). Then \( 1 + X^2 = (1 + X)^2 \) and \( 1 + X^7 \) is divisible by \( 1 + X \) in \( \mathbb{F}_2[X] \). So
\( 1 + X \) is the the greatest common divisor of \( 1 + X^7 \) and \( 1 + X^2 \). Hence \( 1 + X \) is the generator polynomial of \( C \).
Example 7.1.19 Let \( C \) be the Hamming code of Examples 7.1.5 and 7.1.11. Then \( 1 + x^4 + x^5 \) generates \( C \). In order to get the greatest common divisor of \( 1 + X^7 \) and \( 1 + X^4 + X^5 \) we apply the Euclidean algorithm:

\[
1 + X^7 = (1 + X + X^2)(1 + X^4 + X^5) + (X + X^2 + X^4),
\]
\[
1 + X^4 + X^5 = (1 + X)(X + X^2 + X^4) + (1 + X + X^3),
\]
\[
X + X^2 + X^4 = X(1 + X + X^3).
\]

Hence \( 1 + X + X^3 \) is the greatest common divisor, and therefore \( 1 + X + X^3 \) is the generator polynomial of \( C \).

Remark 7.1.20 Let \( g(X) \) be a generator polynomial of a cyclic code of length \( n \), then \( g(X) \) divides \( X^n - 1 \) by Proposition 7.1.15. So \( g(X)h(X) = X^n - 1 \) for some \( h(X) \). Hence \( g(0)h(0) = -1 \). Therefore the constant term of the generator polynomial of a cyclic code is not zero.

Proposition 7.1.21 Let \( g(X) = g_0 + g_1X + \cdots + g_lX^l \) be a polynomial of degree \( l \). Let \( n \) be an integer such that \( l \leq n \). Let \( k = n - l \). Let \( G \) be the \( k \times n \) matrix defined by

\[
G = \begin{pmatrix}
g_0 & g_1 & \cdots & g_l & 0 & \cdots & 0 \\
0 & g_0 & g_1 & \cdots & g_l & \ddots & \vdots \\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\
0 & \cdots & 0 & g_0 & g_1 & \cdots & g_l
\end{pmatrix}.
\]

1. If \( g(X) \) is the generator polynomial of a cyclic code \( C \), then the dimension of \( C \) is equal to \( k \) and a generator matrix of \( C \) is \( G \).

2. If \( g_1 = 1 \) and \( G \) is the generator matrix of a code \( C \) such that

\[
(g_l, 0, \cdots, 0, g_0, g_1, \cdots, g_{n-1}) \in C,
\]

then \( C \) is cyclic with generator polynomial \( g(X) \).

Proof.

1) Suppose \( g(X) \) is the generator polynomial of a cyclic code \( C \). Then the element \( g(x) \) generates \( C \) and the elements \( g(x), xg(x), \ldots, x^{k-1}g(x) \) correspond to the rows of the above matrix.

The generator polynomial is monic, so \( g_l = 1 \) and the \( k \times k \) submatrix of \( G \) consisting of the last \( k \) columns is a lower diagonal matrix with ones on the diagonal, so the rows of \( G \) are independent. Every codeword \( c(x) \in C \) is equal to \( a(x)g(x) \) for some \( a(X) \). Division with remainder of \( X^n - 1 \) by \( a(X)g(X) \) gives that there exist \( e(X) \) and \( f(X) \) such that

\[
a(X)g(X) = e(X)(X^n - 1) + f(X) \quad \text{and} \quad \deg f(X) < n \quad \text{or} \quad f(X) = 0.
\]

But \( X^n - 1 \) is divisible by \( g(X) \) by Proposition 7.1.15. So \( f(X) \) is divisible by \( g(X) \). Hence \( f(X) = b(X)g(X) \) and \( \deg b(X) < n - l = k \) or \( b(X) = 0 \) for some polynomial \( b(X) \). Therefore \( c(x) = a(x)g(x) = b(x)g(x) \) and \( \deg b(X) < k \) or \( b(X) = 0 \). So every codeword is a linear combination of \( g(x), xg(x), \ldots, x^{k-1}g(x) \).
7.1. CYCLIC CODES

Hence \( k \) is the dimension of \( C \) and \( G \) is a generator matrix of \( C \).

2) Suppose \( G \) is the generator matrix of a code \( C \) such that \( g_l = 1 \) and

\[
(g_l, 0, \ldots, 0, g_0, g_1, \ldots, g_{l-1}) \in C.
\]

Then the cyclic shift of the \( i \)-th row of \( G \) is the \((i + 1)\)-th row of \( G \) for all \( i < k \), and the cyclic shift of the \( k \)-th row of \( G \) is \((g_l, 0, \ldots, 0, g_0, g_1, \ldots, g_{l-1})\) which is also an element of \( C \) by assumption. Hence \( C \) is cyclic by Proposition 7.1.3.

Now \( g_l = 1 \) and the upper right corner of \( G \) consists of zeros, so \( G \) has rank \( k \) and the dimension of \( C \) is \( k \). Now \( g(x) \) is monic, has degree \( l = n - k \) and \( g(x) \in C \). The generator polynomial of \( C \) has the same degree \( l \) by (1). Hence \( g(X) \) is the generator polynomial of \( C \).

\[\diamondsuit\]

Example 7.1.22 The ternary cyclic code of length 8 with generator polynomial

\[1 + X + X^3\]

of Example 7.1.16 has dimension 5.

Remark 7.1.23 A cyclic \([n, k]\) code is systematic at the first \( k \) positions, since it has a generator matrix as given in Proposition 7.1.21 which is upper diagonal with nonzero entries on the diagonal at the first \( k \) positions, since \( g_0 \neq 0 \) by Remark 7.1.20. So the row reduced echelon form of a generator matrix of the code has the \( k \times k \) identity matrix at the first \( k \) columns. The last row of this rref matrix is up to the constant \( g_0 \) equal to \((0, \ldots, 0, g_0, g_1, \ldots, g_l)\) giving the coefficients of the generator polynomial. This methods of obtaining the generator polynomial out of a given generator matrix \( G \) is more efficient than taking the greatest common divisor of \( g_1(X), \ldots, g_k(X), X^n - 1 \), where \( g_1, \ldots, g_k \) are the rows of \( G \).

Example 7.1.24 Consider generator matrix \( G \) of the \([6,3]\) cyclic code over \( \mathbb{F}_7 \) of Example 7.1.4. The row reduced echelon form of \( G \) is equal to

\[
\begin{pmatrix}
1 & 0 & 0 & 6 & 1 & 3 \\
0 & 1 & 0 & 3 & 3 & 6 \\
0 & 0 & 1 & 6 & 4 & 6
\end{pmatrix}.
\]

The last row represents

\[x^2 + 6x^3 + 4x^4 + 6x^5 = x^2(1 + 6x + 4x^2 + 6x^3)\]

Hence \(1 + 6x + 4x^2 + 6x^3\) is a codeword. The corresponding monic polynomial \( 6 + X + 3X^2 + X^3 \) has degree 3. Hence this is the generator polynomial.

7.1.4 Encoding cyclic codes

Consider a cyclic code of length \( n \) with generator polynomial \( g(X) \) and the corresponding generator matrix \( G \) as in Proposition 7.1.21. Let the message \( m = (m_0, \ldots, m_{k-1}) \in \mathbb{F}_q^k \) be mapped to the codeword \( c = mG \). In terms of polynomials that means that

\[c(x) = m(x)g(x), \text{ where } m(x) = m_0 + \cdots + m_{k-1}x^{k-1}.\]

In this way we get an encoding of message words into codewords.
The \( k \times k \) submatrix of \( G \) consisting of the last \( k \) columns of \( G \) is a lower
triangular matrix with ones on its diagonal, so it is invertible. That means that we can perform row operations on this matrix until we get another matrix $G_2$ such that its last $k$ columns form the $k \times k$ identity matrix. The matrix $G_2$ is another generator matrix of the same code. The encoding

$$m \mapsto c_2 = mG_2$$

by means of $G_2$ is systematic in the last $k$ positions, that means that there exist $r_0, \ldots, r_{n-k-1} \in \mathbb{F}_q$ such that

$$c_2 = (r_0, \ldots, r_{n-k-1}, m_0, \ldots, m_{k-1}).$$

In other words the encoding has the nice property, that one can read off the sent message directly from the encoded word by looking at the last $k$ positions, in case no errors appeared during the transmission at these positions.

Now how does one translate this systematic encoding in terms of polynomials? Let $m(X)$ be a polynomial of degree at most $k-1$. Let $-r(X)$ be the rest after dividing $m(X)X^{n-k}$ by $g(X)$. Now $\deg(g(X)) = n-k$. So there is a polynomial $q(X)$ such that

$$m(X)X^{n-k} = q(X)g(X) - r(X)$$

and $\deg(r(X)) < n-k$ or $r(X) = 0$.

Hence $r(x) + m(x)x^{n-k} = q(x)g(x)$ is a codeword of the form

$$r_0 + r_1x + \cdots + r_{n-k-1}x^{n-k-1} + m_0x^{n-k} + \cdots + m_{k-1}x^{n-1}.$$

**Example 7.1.25** Consider the cyclic $[7,4,3]$ Hamming code of Example 7.1.19 with generator polynomial $g(X) = 1 + X + X^3$. Let $m$ be a message with polynomial $m(X) = 1 + X^2 + X^3$. Then division of $m(X)X^3$ by $g(X)$ gives as quotient $q(X) = 1 + X + X^2 + X^3$ with rest $r(X) = 1$. The corresponding codeword by systematic encoding is

$$c_2(x) = r(x) + m(x)x^3 = 1 + x^3 + x^5 + x^6.$$

**Example 7.1.26** Consider the ternary cyclic code of length 8 with generator polynomial $1 + X + X^3$ of Example 7.1.16. Let $m$ be a message with polynomial $m(X) = 1 + X^2 + X^3$. Then division of $m(X)X^3$ by $g(X)$ gives as quotient $q(X) = -1 - X + X^2 + X^3$ with rest $-r(X) = 1 - X$. The corresponding codeword by systematic encoding is

$$c_2(x) = r(x) + m(x)x^3 = -1 + x + x^3 + x^5 + x^6.$$

### 7.1.5 Reversible codes

**Definition 7.1.27** Define the reversed word $\rho(x)$ of $x \in \mathbb{F}_q^n$ by

$$\rho(x_0, x_1, \ldots, x_{n-2}, x_{n-1}) = (x_{n-1}, x_{n-2}, \ldots, x_1, x_0).$$

Let $C$ be a code in $\mathbb{F}_q^n$, then its reversed code $\rho(C)$ is defined by

$$\rho(C) = \{ \rho(c) \mid c \in C \}.$$  

A code is called reversible if $C = \rho(C)$.  

Remark 7.1.28 The dimensions of $C$ and $\rho(C)$ are the same, since $\rho$ is an automorphism of $\mathbb{F}_q^n$. If a code is reversible, then $\rho \in \text{Aut}(C)$.

Definition 7.1.29 Let $g(X)$ be a polynomial of degree $l$ given by

$$g_0 + g_1 X + \cdots + g_{l-1} X^{l-1} + g_l X^l.$$ 

Then

$$X^l g(X^{-1}) = g_l + g_{l-1} X + \cdots + g_1 X^{l-1} + g_0 X^l.$$ 

is called the reciprocal of $g(X)$. If moreover $g(0) \neq 0$, then $X^l g(X^{-1})/g(0)$ is called the monic reciprocal of $g(X)$. The polynomial $g(X)$ is called reversible if $g(0) \neq 0$ and it is equal to its monic reciprocal.

Remark 7.1.30 If $g = (g_0, g_1, \ldots, g_{l-1}, g_l)$ are the coefficients of the polynomial $g(X)$, then the reversed word $\rho(g)$ give the coefficients of the reciprocal of $g(X)$.

Remark 7.1.31 If $\alpha$ is a zero of $g(X)$ and $\alpha \neq 0$, then the reciprocal $\alpha^{-1}$ is a zero of the reciprocal of $g(X)$.

Proposition 7.1.32 Let $g(X)$ be the generator polynomial of a cyclic code $C$. Then $\rho(C)$ is cyclic with the monic reciprocal of $g(X)$ as generator polynomial, and $C$ is reversible if and only if $g(X)$ is reversible.

Proof. A cyclic code is invariant under the forward shift $\sigma$ and the backward shift $\sigma^{n-1}$. Now $\sigma(\rho(c)) = \rho(\sigma^{n-1}(c))$ for all $c \in C$. Hence $\rho(C)$ is cyclic. Now $g(0) \neq 0$ by Remark 7.1.20. Hence the monic reciprocal of $g(X)$ is well defined and its corresponding word is an element of $\rho(C)$ by Remark 7.1.30. The degree of $g(X)$ and its monic reciprocal are the same, and the dimensions of $C$ and $\rho(C)$ are the same. Hence this monic reciprocal is the generator polynomial of $\rho(C)$.

Therefore $C$ is reversible if and only if $g(X)$ is reversible, by the definition of a reversible polynomial. ⋄

Remark 7.1.33 If $C$ is a reversible cyclic code, then the group generated by $\sigma$ and $\rho$ is the dihedral group of order $2n$ and is contained in $\text{Aut}(C)$.

7.1.6 Parity check polynomial

Definition 7.1.34 Let $g(X)$ be the generator polynomial of a cyclic code $C$ of length $n$. Then $g(X)$ divides $X^n - 1$ by Proposition 7.1.15 and

$$h(X) = \frac{X^n - 1}{g(X)}$$

is called the parity check polynomial of $C$.

Proposition 7.1.35 Let $h(X)$ be the parity check polynomial of a cyclic code $C$. Then $c(x) \in C$ if and only if $c(x) h(x) = 0$. 
Proof. Let \( c(x) \in C \). Then \( c(x) = a(x)g(x) \), for some \( a(x) \). We have that
\[
g(X)h(X) = X^n - 1. \]
Hence \( g(x)h(x) = 0 \). So \( c(x)h(x) = a(x)g(x)h(x) = 0 \).
Conversely, suppose that \( c(x)h(x) = 0 \). There exist polynomials \( a(X) \) and \( b(X) \) such that
\[
c(X) = a(X)g(X) + b(X) \text{ and } b(X) = 0 \text{ or } \deg b(X) < \deg g(X).
\]
Hence
\[
c(x)h(x) = a(x)g(x)h(x) + b(x)h(x) = b(x)h(x).
\]
Notice that \( b(x)h(x) \neq 0 \) if \( b(x) \) is a nonzero polynomial, since \( \deg b(X)h(X) \)
is at most \( n - 1 \). Hence \( b(X) = 0 \) and \( c(x) = a(x)g(x) \in C \).

Remark 7.1.36 If \( H \) is a parity check matrix for a code \( C \), then \( H \) is a gen-
erator matrix for the dual of \( C \). One might expect that if \( h(X) \) is the parity
check polynomial for a cyclic code \( C \), then \( h(X) \) is the generator polynomial of
the dual of \( C \). This is not the case but something of this nature is true as the
following shows.

Proposition 7.1.37 Let \( h(X) \) be the parity check polynomial of a cyclic code
\( C \). Then the monic reciprocal of \( h(X) \) is the generator polynomial of \( C^\perp \).

Proof. Let \( C \) be a cyclic code of length \( n \) and dimension \( k \) with generator
polynomial \( g(X) \) and parity check polynomial \( h(X) \).
If \( k = 0 \), then \( g(X) = X^n - 1 \) and \( h(X) = 1 \) and similarly if \( k = n \), then
\( g(X) = 1 \) and \( h(X) = X^n - 1 \). Hence the proposition is true in these cases.
Now suppose that \( 0 < k < n \). Then \( h(X) = h_0 + h_1X + \cdots + h_kX^k \). Hence
\[
X^k h(X^{-1}) = h_k + h_{k-1}X + \cdots + h_0 X^k.
\]
The \( i \)-th position of \( x^k h(x^{-1}) \) is \( h_{n-i} \). Let \( g(X) \) be the generator polynomial of
\( C \). Let \( t = n - k \). Then \( g(X) = g_0 + g_1X + \cdots + g_tX^t \) and \( g_t = 1 \). The elements
\( x^i g(x) \) generate \( C \). The \( i \)-th position of \( x^i g(x) \) is equal to \( g_{n-t} \). Hence the inner
product of the words \( x^i g(x) \) and \( x^k h(x^{-1}) \) is
\[
\sum_{i=0}^{k} g_{t+i} h_{n-i},
\]
which is the coefficient of the term \( X^{k+t} \) in \( X^t g(X)h(X) \). But \( X^t g(X)h(X) \)
is equal to \( X^{n+t} - X^t \) and \( 0 < k < n \), hence this coefficient is zero. So
\( \sum_{i=1}^{n} g_{t+i} h_{n-i} = 0 \) for all \( t \). So \( x^k h(x^{-1}) \) is an element of the dual of \( C \).
Now \( g(X)h(X) = X^n - 1 \). So \( g(0)h(0) = -1 \). Hence the monic reciprocal of
\( h(X) \) is well defined, is monic, represents an element of \( C^\perp \), has degree \( k \) and
the dimension of \( C^\perp \) is \( n - k \). Hence \( X^k h(X^{-1}) / h(0) \) is the generator polynomial
of \( C^\perp \) by Proposition 7.1.21.

Example 7.1.38 Consider the \([6,3]\) cyclic code over \( \mathbb{F}_7 \) of Example 7.1.24 which
has generator polynomial \( X^3 + 4X^2 + X + 1 \). Hence
\[
h(X) = \frac{X^6 - 1}{X^3 + 4X^2 + X + 1} = X^3 + 4X^2 + X + 1
\]
7.1. CYCLIC CODES

is the parity check polynomial of the code. The generator polynomial of the
dual code is
\[ g^\perp(X) = X^6 h(X^{-1}) = 1 + 4X + X^2 + X^3 \]
by Proposition 7.1.37, since \( h(0) = 1 \).

**Example 7.1.39** Consider in \( \mathbb{F}_2[X] \) the polynomial
\[ g(X) = 1 + X^4 + X^6 + X^7 + X^8. \]
Then \( g(X) \) divides \( X^{15} - 1 \) with quotient
\[ h(X) = \frac{X^{15} - 1}{g(X)} = 1 + X^4 + X^6 + X^7. \]
Hence \( g(X) \) is the generator polynomial of a binary cyclic code of length 15
with parity check polynomial \( h(X) \). The generator polynomial of the dual code
is
\[ g^\perp(X) = X^7 h(X^{-1}) = 1 + X + X^3 + X^7 \]
by Proposition 7.1.37, since \( h(0) = 1 \).

**Example 7.1.40** The generator polynomial \( 1 + X + X^3 \) of the ternary code of
length 8 of Example 7.1.16 has parity check polynomial
\[ h(X) = \frac{X^8 - 1}{g(X)} = X^5 - X^3 - X^2 + X - 1. \]
The generator polynomial of the dual code is
\[ g^\perp(X) = X^8 h(X^{-1})/h(0) = X^5 - X^4 + X^3 + X^2 - 1 \]
by Proposition 7.1.37.

**Example 7.1.41** Let us now take a look at how cyclic codes are constructed
via generator and check polynomials in GAP.

```gap
> x:=Indeterminate(GF(2));;
> f:=x^17-1;;
> F:=Factors(PolynomialRing(GF(2)),f);
[ x_1+Z(2)^0, x_1^8+x_1^5+x_1^4+x_1^3+Z(2)^0, x_1^8+x_1^7+x_1^6+x_1^4+x_1^2+x_1+Z(2)^0 ]
> g:=F[2];;
> C:=GeneratorPolCode(g,17,"code from Example 6.1.41",GF(2));;
> MinimumDistance(C);;
> h:=F[3];;
> C2:=CheckPolCode(h,17,GF(2));;
> MinimumDistance(C2);;
> C;
```
a cyclic \([17,9,5]3..4\) code from Example 6.1.41 over \( \text{GF}(2) \)

```gap
> h:=F[3];;
> C2:=CheckPolCode(h,17,GF(2));;
> MinimumDistance(C2);;
> C2;
```
a cyclic \([17,8,6]3..7\) code defined by check polynomial over \( \text{GF}(2) \)

So here \( x \) is a variable with which the polynomials are built. Note that one can
also define it via \( x:=X(\text{GF}(2)) \), since \( X \) is a synonym of \textit{Indeterminate}. For
this same reason we could not use \( x \) as a variable.
7.1.7 Exercises

7.1.1 Let $C$ be the $\mathbb{F}_q$-linear code with generator matrix

$$G = \begin{pmatrix}
    1 & 1 & 1 & 1 & 0 & 0 & 0 \\
    0 & 1 & 1 & 1 & 1 & 0 & 0 \\
    0 & 0 & 1 & 1 & 1 & 1 & 0 \\
    0 & 0 & 0 & 1 & 1 & 1 & 1 
\end{pmatrix}.$$ 

Show that $C$ is not cyclic for every finite field $\mathbb{F}_q$.

7.1.2 Let $C$ be a cyclic code over $\mathbb{F}_q$ of length 7 such that $(1, 1, 1, 0, 0, 0, 0)$ is an element of $C$. Show that $C$ is a trivial code if $q$ is not a power of 3.

7.1.3 Find the generator polynomial of the binary cyclic code of length 7 generated by $1 + x + x^5$.

7.1.4 Show that $2 + X^2 + X^3$ is the generator polynomial of a ternary cyclic code of length 13.

7.1.5 Let $\alpha$ be an element in $\mathbb{F}_8$ such that $\alpha^3 = \alpha + 1$. Let $C$ be the $\mathbb{F}_8$-linear code with generator matrix $G$, where

$$G = \begin{pmatrix}
    1 & 1 & 1 & 1 & 1 & 1 & 1 \\
    1 & \alpha & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \\
    1 & \alpha^2 & \alpha^4 & \alpha^6 & \alpha & \alpha^3 & \alpha^5 
\end{pmatrix}.$$ 

1) Show that the code $C$ is cyclic.
2) Determine the coefficients of the generator polynomial of this code.

7.1.6 Consider the binary polynomial $g(X) = 1 + X^2 + X^5$.
1) Show that $g(X)$ is the generator polynomial of a binary cyclic code $C$ of length 31 and dimension 26.
2) Give the encoding with respect to the code $C$ of the message $m$ with $m(X) = 1 + X^{10} + X^{20}$ as message polynomial, that is systematic at the last 26 positions.
3) Find the parity check polynomial of $C$.
4) Give the coefficients of the generator polynomial of $C^\perp$.

7.1.7 Give a description of the systematic encoding of an $[n, k]$ cyclic code in the first $k$ positions in terms of division by the generator polynomial with rest.

7.1.8 Estimate the number of additions and the number of multiplications in $\mathbb{F}_q$ needed to encode an $[n, k]$ cyclic code using multiplication with the generator polynomial and compare these with the numbers for systematic encoding in the last $k$ positions by dividing $m(X)X^{n-k}$ by $g(X)$ with rest.

7.1.9 [CAS] Implement the encoding procedure from Section 7.1.4.

7.1.10 [CAS] Having a generator polynomial $g$, code length $n$, and field size $q$ construct a cyclic code dual to the one generated by $g$. Use the function $\text{ReciprocalPolynomial}$ (both in GAP and Magma).
7.2 Defining zeros

*** ***

7.2.1 Structure of finite fields

The finite fields we encountered up to now were always of the form $\mathbb{F}_p = \mathbb{Z}/\langle p \rangle$ with $p$ a prime. For the notion of defining zeros of a cyclic codes this does not suffice and extensions of prime fields are needed. In this section we state basic facts on the structure of finite fields. For proofs we refer to the existing literature.

**Definition 7.2.1** The smallest subfield of a field $\mathbb{F}$ is unique and is called the *prime field* of $\mathbb{F}$. The only prime fields are the rational numbers $\mathbb{Q}$ and the finite field $\mathbb{F}_p$ with $p$ a prime and the characteristic of the field is zero and $p$, respectively.

**Remark 7.2.2** Let $\mathbb{F}$ be a field of characteristic $p$ a prime. Then

$$(x + y)^p = x^p + y^p$$

for all $x, y \in \mathbb{F}$ by Newton’s binomial formula, since $\binom{p}{i}$ is divisible by $p$ for all $i = 1, \ldots, p - 1$.

**Proposition 7.2.3** If $\mathbb{F}$ is a finite field, then the number of elements of $\mathbb{F}$ is a power of a prime number.

**Proof.** The characteristic of a finite field is prime, and such a field is a vector space over the prime field of finite dimension. So the number of elements of a finite field is a power of a prime number. \qed

**Remark 7.2.4** The factor ring over the field of polynomials in one variable with coefficients in a field $\mathbb{F}$ modulo an irreducible polynomial gives a way to construct a field extension of $\mathbb{F}$. In particular, if $f(X) \in \mathbb{F}_p[X]$ is irreducible, and $\langle f(X) \rangle$ is the ideal generated by all the multiples of $f(X)$, then the factor ring $\mathbb{F}_p[X]/\langle f(X) \rangle$ is a field with $p^e$ elements, where $e = \deg f(X)$. The coset of $X$ modulo $\langle f(X) \rangle$ is denoted by $x$, and the monomials $1, x, \ldots, x^{e-1}$ form a basis over $\mathbb{F}_p$. Hence every element in this field is uniquely represented by a polynomial $g(X) \in \mathbb{F}_p[X]$ of degree at most $e - 1$ and its coset is denoted by $g(x)$. This is called the *principal representation*. The sum of two representatives is again a representative. For the product one has to divide by $f(X)$ and take the rest as a representative.

**Example 7.2.5** The irreducible polynomials of degree one in $\mathbb{F}_2[X]$ are $X$ and $1 + X$. And $1 + X + X^2$ is the only irreducible polynomial of degree two in $\mathbb{F}_2[X]$. There are exactly two irreducible polynomials of degree three in $\mathbb{F}_2[X]$. These are $1 + X + X^3$ and $1 + X^2 + X^3$.

Consider the field $\mathbb{F} = \mathbb{F}_2[X]/(1 + X + X^3)$ with 8 elements. Then $1, x, x^2$ is a basis of $\mathbb{F}$ over $\mathbb{F}_2$. Now

$$(1 + X)(1 + X + X^2) = 1 + X^3 \equiv X \mod 1 + X + X^3.$$
Hence \((1 + x)(1 + x + x^2) = x\) in \(F\). In the following table the powers \(x^i\) are written by their principal representatives.

\[
\begin{align*}
x^3 &= 1 + x \\
x^4 &= x + x^2 \\
x^5 &= 1 + x + x^2 \\
x^6 &= 1 + x^2 \\
x^7 &= 1
\end{align*}
\]

Therefore the nonzero elements form a cyclic group of order 7 with \(x\) as generator.

**Definition 7.2.6** Let \(F\) be a field. Let \(f(X) = \sum_{i=0}^{n} a_i X^i\) in \(F[X]\). Then \(f'(X)\) is the formal derivative of \(f(X)\) and is defined by

\[
f'(X) = \sum_{i=1}^{n} ia_i X^{i-1}.
\]

**Remark 7.2.7** The product or Leibniz rule holds for the derivative

\[
(f(X)g(X))' = f'(X)g(X) + f(X)g'(X).
\]

The following criterion gives a way to decide whether the zeros of a polynomial are simple.

**Lemma 7.2.8** Let \(F\) be a field. Let \(f(X) \in F[X]\). Then every zero of \(f(X)\) has multiplicity one if and only if \(\gcd(f(X), f'(X)) = 1\).

**Proof.** Suppose \(\gcd(f(X), f'(X)) = 1\). Let \(\alpha\) be a zero of \(f(X)\) of multiplicity \(m\). Then there exists a polynomial \(a(X)\) such that \(f(X) = (X - \alpha)^m a(X)\). Differentiating this equality gives

\[
f'(X) = m(X - \alpha)^{m-1} a(X) + (X - \alpha)^m a'(X).
\]

If \(m > 1\), then \(X - \alpha\) divides \(f(X)\) and \(f'(X)\). This contradicts the assumption that \(\gcd(f(X), f'(X)) = 1\). Hence every zero of \(f(X)\) has multiplicity one.

Conversely, if \(\gcd(f(X), f'(X)) \neq 1\), then \(f(X)\) and \(f'(X)\) have a common zero \(\alpha\), possibly in an extension of \(F\). Conclude that \((X - \alpha)^2\) divides \(f(X)\), using the product rule again.

**Remark 7.2.9** Let \(p\) be a prime and \(q = p^e\). The formal derivative of \(X^q - X\) is \(-1\) in \(F_p\). Hence all zeros of \(X^q - X\) in an extension of \(F_p\) are simple by Lemma 7.2.8.

For every field \(F\) and polynomial \(f(X)\) in one variable \(X\) there exists a field \(G\) that contains \(F\) as a subfield such that \(f(X)\) splits in linear factors in \(G[X]\). The smallest field with these properties is unique up to an isomorphism of fields and is called the splitting field of \(f(X)\) over \(F\).

A field \(F\) is called algebraically closed if every polynomial in one variable has a zero in \(F\). So every polynomial in one variable over an algebraically closed field splits in linear factors over this field. Every field \(F\) has an extension \(G\)
7.2. DEFINING ZEROS

that is algebraically closed such that \( \mathbb{G} \) does not have a proper subfield that is algebraically closed. Such an extension is unique up to isomorphism and is called the algebraic closure of \( \mathbb{F} \) and is denoted by \( \overline{\mathbb{F}} \). The field \( \mathbb{C} \) of complex numbers is the algebraic closure of the field \( \mathbb{R} \) of real numbers.

**Remark 7.2.10** If \( \mathbb{F} \) is a field with \( q \) elements, then \( \mathbb{F}^* = \mathbb{F} \setminus \{0\} \) is a multiplicative group of order \( q - 1 \). So \( x^{q-1} = 1 \) for all \( x \in \mathbb{F}^* \). Hence \( x^q = x \) for all \( x \in \mathbb{F} \). Therefore the zeros of \( X^q - X \) are precisely the elements of \( \mathbb{F} \).

**Theorem 7.2.11** Let \( p \) be a prime and \( q = p^e \). There exists a field of \( q \) elements and any field with \( q \) elements is isomorphic to the splitting field of \( X^q - X \) over \( \mathbb{F}_p \) is denoted by \( \mathbb{F}_q \) or \( GF(q) \), the Galois field of \( q \) elements.

**Proof.** The splitting field of \( X^q - X \) over \( \mathbb{F}_p \) contains the zeros of \( X^q - X \). Let \( Z \) be the set of zeros of \( X^q - X \) in the splitting field. Then \( |Z| = q \), since \( X^q - X \) splits in linear factors and all zeros are simple by Remark 7.2.9. Now 0 and 1 are elements of \( Z \) and \( Z \) is closed under addition, subtraction, multiplication and division by nonzero elements. Hence \( Z \) is a field. Furthermore \( Z \) contains \( \mathbb{F}_p \) since \( q = p^e \). Hence \( Z \) is equal to the splitting field of \( X^q - X \) over \( \mathbb{F}_p \). Hence the splitting field has \( q \) elements.

If \( \mathbb{F} \) is a field with \( q \) elements, then all elements of \( \mathbb{F} \) are zeros of \( X^q - X \) by Remark 7.2.10. Hence \( \mathbb{F} \) is contained in an isomorphic copy of the splitting field of \( X^q - X \) over \( \mathbb{F}_p \). Therefore they are equal, since both have \( q \) elements. \( \lozenge \)

The set of invertible elements of the finite field \( \mathbb{F}_q \) is an abelian group of order \( q - 1 \). But a stronger statement is true.

**Proposition 7.2.12** The multiplicative group \( \mathbb{F}_q^* \) is cyclic.

**Proof.** The order of an element of \( \mathbb{F}_q^* \) divides \( q - 1 \), since \( \mathbb{F}_q^* \) is a group of order \( q - 1 \). Let \( d \) be the maximal order of an element of \( \mathbb{F}_q^* \). Then \( d \) divides \( q - 1 \). Let \( x \) be an element of order \( d \). If \( y \in \mathbb{F}_q^* \), then the order \( n \) of \( y \) divides \( d \). Otherwise there is a prime \( l \) dividing \( n \) and \( l \) not dividing \( d \). So \( z = y^{n/l} \) has order \( l \). Hence \( xz \) has order \( dl \), contradicting the maximality of \( d \). Therefore the order of an element of \( \mathbb{F}_q^* \) divides \( d \). So the elements of \( \mathbb{F}_q^* \) are zeros of \( X^d - 1 \). Hence \( q - 1 \leq d \) and \( d \) divides \( q - 1 \). We conclude that \( d = q - 1 \), \( x \) is an element of order \( q - 1 \) and \( \mathbb{F}_q^* \) is cyclic generated by \( x \). \( \lozenge \)

**Definition 7.2.13** A generator of \( \mathbb{F}_q^* \) is called a primitive element. An irreducible polynomial \( f(X) \in \mathbb{F}_p[X] \) is called primitive if \( x \) is a primitive element in \( \mathbb{F}_p[X]/(f(X)) \), where \( x \) is the coset of \( X \) modulo \( f(X) \).

**Definition 7.2.14** Choose a primitive element \( \alpha \) of \( \mathbb{F}_q \). Define \( \alpha^* = 0 \). Then for every element \( \beta \in \mathbb{F}_q \) there is a unique \( i \in \{*, 0, 1, \ldots, q - 2\} \) such that \( \beta = \alpha^i \), and this \( i \) is called the logarithm of \( \beta \) with respect to \( \alpha \), and \( \alpha^i \) the exponential representation of \( \beta \). For every \( i \in \{*, 0, 1, \ldots, q - 2\} \) there is a unique \( j \in \{*, 0, 1, \ldots, q - 2\} \) such that

\[
1 + \alpha^i = \alpha^j
\]

and this \( j \) is called the Zech logarithm of \( i \) and is denoted by \( Zech(i) = j \).
Remark 7.2.15 Let $p$ be a prime and $q = p^e$. In a principal representation of $\mathbb{F}_q$, every element is given by a polynomial of degree at most $e - 1$ with coefficients in $\mathbb{F}_p$ and addition in $\mathbb{F}_q$ is easy and done coefficient wise in $\mathbb{F}_p$. But for the multiplication we need to multiply two polynomials and compute a division with rest.

Define the addition $i + j$ for $i, j \in \{\ast, 0, 1, \ldots, q - 2\}$, where $i + j$ is taken modulo $q - 1$ if $i$ and $j$ are both not equal to $\ast$, and $i + j = *$ if $i = *$ or $j = *$. Then multiplication in $\mathbb{F}_q$ is easy in the exponential representation with respect to a primitive element, since $\alpha^i \alpha^j = \alpha^{i + j}$ for $i, j \in \{\ast, 0, 1, \ldots, q - 2\}$. In the exponential representation the addition can be expressed in terms of the Zech logarithm.

$$\alpha^i + \alpha^j = \alpha^{i + \text{Zech}(j-i)}.$$  

Example 7.2.16 Consider the finite field $\mathbb{F}_8$ as given in Example 7.2.5 by the irreducible polynomial $1 + X + X^3$. In the following table the elements are represented as powers in $x$, as polynomials $a_0 + a_1 x + a_2 x^2$ and the Zech logarithm is given.

<table>
<thead>
<tr>
<th>$i$</th>
<th>$x^i$</th>
<th>$(a_0, a_1, a_2)$</th>
<th>Zech($i$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$*$</td>
<td>$x^*$</td>
<td>$(0, 0, 0)$</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>$x^0$</td>
<td>$(1, 0, 0)$</td>
<td>$*$</td>
</tr>
<tr>
<td>1</td>
<td>$x^1$</td>
<td>$(0, 1, 0)$</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>$x^2$</td>
<td>$(0, 0, 1)$</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>$x^3$</td>
<td>$(1, 1, 0)$</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>$x^4$</td>
<td>$(0, 1, 1)$</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>$x^5$</td>
<td>$(1, 1, 1)$</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>$x^6$</td>
<td>$(1, 0, 1)$</td>
<td>2</td>
</tr>
</tbody>
</table>

In the principal representation we immediately see that $x^3 + x^5 = x^2$, since $x^3 = 1 + x$ and $x^5 = 1 + x + x^2$. The exponential representation by means of the Zech logarithm gives

$$x^3 + x^5 = x^{3 + \text{Zech}(2)} = x^2.$$  

***Applications: quasi random generators, discrete logarithm.***

Definition 7.2.17 Let $\text{Irr}_q(n)$ be the number of irreducible monic polynomials over $\mathbb{F}_q$ of degree $n$.

Proposition 7.2.18 Let $q$ be a power of a prime number. Then

$$q^n = \sum_{d|n} d \cdot \text{Irr}_q(d).$$

Proof. ***...***

Proposition 7.2.19

$$\text{Irr}_q(n) = \frac{1}{n} \sum_{d|n} \mu \left( \frac{n}{d} \right) q^d.$$
7.2. DEFINING ZEROS

Proof. Consider the poset $\mathbb{N}$ of Example 5.3.20 with the divisibility as partial order. Define $f(d) = d \cdot \operatorname{Irr}_q(d)$. Then the sum function $\tilde{f}(n) = \sum_{d|n} f(d)$ is equal to $q^n$, by Proposition 7.2.18. The Möbius inversion formula 5.3.10 implies that $n \cdot \operatorname{Irr}_q(n) = \sum_{d|n} \mu \left( \frac{n}{d} \right) q^d$ which gives the desired result.

Remark 7.2.20 Proposition 7.2.19 implies

$$\operatorname{Irr}_q(d) \geq \frac{1}{n} \left( q^n - q^{n-1} - \cdots - q \right) = \frac{1}{n} \left( q^n - \frac{q^n - q}{q - 1} \right) > 0,$$

since $\mu(1) = 1$ and $\mu(d) \geq -1$ for all $d$. By counting the number of irreducible polynomials over $\mathbb{F}_q$ we see that there exists an irreducible polynomial in $\mathbb{F}_q[X]$ of every degree $d$.

Let $q = p^d$ and $p$ a prime. Now $\mathbb{Z}_p$ is a field with $p$ elements. There exists an irreducible polynomial $f(T)$ in $\mathbb{Z}_p[T]$ of degree $d$, and $\mathbb{Z}_p[T]/\langle f(T) \rangle$ is a field with $p^d = q$ elements. This is another way to show the existence of a finite field with $q$ elements, where $q$ is a prime power.

7.2.2 Minimal polynomials

***

Remark 7.2.21 From now on we assume that $n$ and $q$ are relatively prime. This assumption is not necessary but it would complicate matters otherwise. Hence $q$ has an inverse modulo $n$. So $q^e \equiv 1 \pmod{n}$ for some positive integer $e$. Hence $n$ divides $q^e - 1$. Let $\mathbb{F}_{q^e}$ be the extension of $\mathbb{F}_q$ of degree $e$. So $n$ divides the order of $\mathbb{F}_{q^e}^*$, the cyclic group of units. Hence there exists an element $\alpha \in \mathbb{F}_{q^e}^*$ of order $n$. From now on we choose such an element $\alpha$ of order $n$.

Example 7.2.22 The order of the cyclic group $\mathbb{F}_{3^e}^*$ is 2, 8, 26, 80 and 242 for $e = 1, 2, 3, 4$ and 5, respectively. Hence $\mathbb{F}_{3^5}$ is the smallest field extension of $\mathbb{F}_3$ that has an element of order 11.

Remark 7.2.23 The multiplicity of every zero of $X^n - 1$ is one by Lemma 7.2.8, since $\gcd(X^n - 1, nX^{n-1}) = 1$ in $\mathbb{F}_q$ by the assumption that $\gcd(n, q) = 1$. Let $\alpha$ be an element in some extension of $\mathbb{F}_q$ of order $n$. Then $1, \alpha, \alpha^2, \ldots, \alpha^{n-1}$ are $n$ mutually distinct zeros of $X^n - 1$. Hence

$$X^n - 1 = \prod_{i=0}^{n-1} (X - \alpha^i).$$

Definition 7.2.24 Let $\alpha$ be a primitive $n$-th root of unity in the extension field $\mathbb{F}_{q^e}$. For this choice of an element of order $n$ we define $m_\alpha(X)$ as the minimal polynomial of $\alpha^i$, that is the monic polynomial in $\mathbb{F}_q[X]$ of smallest degree such that $m_\alpha(\alpha^i) = 0$.

Example 7.2.25 In particular $m_0(X) = X - 1$.

Proposition 7.2.26 The minimal polynomial $m_\alpha(X)$ is irreducible in $\mathbb{F}_q[X]$.
7.2.30 Let $m_i(X) = f(X)g(X)$ with $f(X), g(X) \in \mathbb{F}_q[X]$. Then $f(\alpha^i)g(\alpha^i) = m_i(\alpha^i) = 0$. So $f(\alpha^i) = 0$ or $g(\alpha^i) = 0$. Hence $\deg(f(X)) \geq \deg(m_i(X))$ or $\deg(g(X)) \geq \deg(m_i(X))$ by the minimality of the degree of $m_i(X)$. Hence $m_i(X)$ is irreducible.

Example 7.2.27 Choose $\alpha = 3$ as the primitive element in $\mathbb{F}_7$ of order 6. Then $X^6 - 1$ is the product of linear factors in $\mathbb{F}_7[X]$. Furthermore $m_1(X) = X - 3$, $m_2(X) = X - 2$, $m_3(X) = X - 6$ and so on. But 5 is also an element of order 6 in $\mathbb{F}_7^*$. The choice $\alpha = 5$ would give $m_1(X) = X - 5$, $m_2(X) = X - 4$ and so on.

Example 7.2.28 There are exactly two irreducible polynomials of degree 3 in $\mathbb{F}_2[X]$. They are factors of $1 + X^3$:

$$1 + X^3 = (1 + X)(1 + X + X^3)(1 + X^2 + X^3).$$

Let $\alpha \in \mathbb{F}_8$ be a zero of $1 + X + X^3$. Then $\alpha$ is a primitive element of $\mathbb{F}_8$ and $\alpha^2$ and $\alpha^4$ are the remaining zeros of $1 + X + X^3$. The reciprocal of $1 + X + X^3$ is

$$X^3(1 + X^{-1} + X^{-3}) = 1 + X^2 + X^3$$

and has $\alpha^{-1} = \alpha^6$, $\alpha^{-2} = \alpha^5$ and $\alpha^{-4} = \alpha^3$ as zeros. So $m_1(X) = 1 + X + X^3$ and $m_3(X) = 1 + X^2 + X^3$.

Proposition 7.2.29 The monic reciprocal of $m_i(X)$ is equal to $m_{-i}(X)$.

Proof. The element $\alpha^i$ is a zero of $m_i(X)$. So $\alpha^{-i}$ is a zero of the monic reciprocal of $m_i(X)$ by Remark 7.1.30. Hence the degree of the monic reciprocal of $m_i(X)$ is at least $\deg(m_{-i}(X))$. So $\deg(m_i(X)) \geq \deg(m_{-i}(X))$. Similarly $\deg(m_i(X)) \leq \deg(m_{-i}(X))$. So $\deg(m_i(X)) = \deg(m_{-i}(X))$ is the degree of the monic reciprocal of $m_i(X)$. Hence the monic reciprocal of $m_i(X)$ is a monic polynomial of minimal degree having $\alpha^{-i}$ as a zero, therefore it is equal to $m_{-i}(X)$.

7.2.3 Cyclotomic polynomials and cosets

Definition 7.2.30 Let $n$ be a nonnegative integer. Then Euler’s function $\varphi$ is given by

$$\varphi(n) = |\{i : \gcd(i, n) = 1, 0 \leq i < n\}|.$$

Lemma 7.2.31 The following properties of Euler’s function hold:

1) $\varphi(mn) = \varphi(m)\varphi(n)$ if $\gcd(m, n) = 1$.
2) $\varphi(1) = 1$.
3) $\varphi(p) = p - 1$ if $p$ is a prime number.
4) $\varphi(p^r) = p^{r-1}(p - 1)$ if $p$ is a prime number.
Chinese remainder theorem gives that \( F \) is prime to the characteristic of cyclotomic polynomial of order \( n \). Hence \( \varphi(n) = |\mathbb{Z}_n^*| \). The \textbf{Definition 7.2.33} Let \( \mathbb{Z} \) be a field. Let \( n \) be a positive integer that is relatively prime to the characteristic of \( \mathbb{F} \). Let \( \alpha \) be an element of order \( n \) in \( \mathbb{F}^* \). The \textbf{cyclotomic polynomial of order} \( n \) is defined by \( \Phi_n(X) = \prod_{\text{gcd}(i,n) = 1, 0 < i < n} (X - \alpha^i) \).

\textbf{Remark 7.2.34} The degree of \( \Phi_n(X) \) is equal to \( \varphi(n) \).

\textbf{Remark 7.2.35} If \( x \) is a primitive element, then \( y \) is a primitive element if and only if \( y = x^i \) for some \( i \) such that \( 1 < i < q - 1 \) and \( \text{gcd}(i, q - 1) = 1 \). Hence the number of primitive elements in \( \mathbb{F}_q^* \) is equal to \( \varphi(q - 1) \), where \( \varphi \) is Euler’s function.

\textbf{Theorem 7.2.36} Let \( \mathbb{F} \) be a field. Let \( n \) be a positive integer that is relatively prime to the characteristic of \( \mathbb{F} \). The polynomial \( \Phi_n(X) \) is in \( \mathbb{F}[X] \), has as zeros all elements in \( \mathbb{F}^* \) of order \( n \) and has degree \( \varphi(n) \), where \( \varphi \) is Euler’s function. Furthermore \( X^n - 1 = \prod_{d|n} \Phi_d(X) \).

\textbf{Proof.} The degree of \( \Phi_n(X) \) is equal to the number \( i \) such that \( 0 < i < n \) and \( \text{gcd}(i, n) = 1 \) which is by definition equal to \( \varphi(n) \). The power \( \alpha^i \) has order \( n \) if and only if \( \text{gcd}(i, n) = 1 \). Conversely if \( \beta \) is an element of order \( n \) in \( \mathbb{F}^* \), then \( \beta \) is a zero of \( X^n - 1 \) and \( X^n - 1 = \prod_{0 < i < n} (X - \alpha^i) \). So \( \beta = \alpha^i \) for some \( i \) with \( 0 < i < n \) and \( \text{gcd}(i, n) = 1 \). Hence \( \Phi_n(X) \) has as zeros all elements in \( \mathbb{F}^* \) of order \( n \). Therefore \( X^n - 1 = \prod_{0 < i < n} (X - \alpha^i) = \prod_{d|n \text{ gcd}(i,d)=1} (X - \alpha^i) = \prod_{d|n} \Phi_d(X) \).

The fact that \( \Phi_n(X) \) has coefficients in \( \mathbb{F} \) is shown by induction on \( n \). Now \( \Phi_1(X) = X - 1 \) is in \( \mathbb{F}[X] \). Suppose that \( \Phi_m(X) \) is in \( \mathbb{F}[X] \) for all \( m < n \). Then \( f(X) = \prod_{n \neq d|n} \Phi_d(X) \) is in \( \mathbb{F}[X] \), and \( X^n - 1 = f(X) \Phi_n(X) \). So \( X^n - 1 \) is divisible by \( f(X) \) in \( \mathbb{F}[X] \), and \( X^n - 1 \) and \( f(X) \) are in \( \mathbb{F}[X] \). Hence \( \Phi_n(X) \) is in \( \mathbb{F}[X] \).
Remark 7.2.37 The factorization of $X^n - 1$ in cyclotomic polynomials gives a way to compute the $\Phi_n(X)$ recursively.

Remark 7.2.38 The cyclotomic polynomial $\Phi_n(X)$ depends on the field $\mathbb{F}$ in Definition 7.2.33. But $\Phi_n(X)$ is universal in the sense that in characteristic zero it has integer coefficients and they do not depend on the field $\mathbb{F}$. By reducing the coefficients of this polynomial modulo a prime $p$ one gets the cyclotomic polynomial over any field of characteristic $p$. In characteristic zero $\Phi_n(X)$ is irreducible in $\mathbb{Q}[X]$ for all $n$. But $\Phi_n(X)$ is sometimes reducible in $\mathbb{F}_p[X]$.

Example 7.2.39 The polynomials $\Phi_1(X) = X - 1$ and $\Phi_2(X) = X + 1$ are irreducible in any characteristic, and $X^2 - 1 = \Phi_1(X)\Phi_2(X)$. Now

$$X^3 - 1 = \Phi_1(X)\Phi_3(X).$$

Hence $\Phi_3(X) = X^2 + X + 1$, and this polynomial is irreducible in $\mathbb{F}_p[X]$ if and only if $\mathbb{F}_p^2$ has no element of order $3$ if and only if $p \equiv 2 \mod 3$.

$$X^4 - 1 = \Phi_1(X)\Phi_2(X)\Phi_4(X).$$

So $\Phi_4(X) = X^2 + 1$, and this polynomial is irreducible in $\mathbb{F}_p[X]$ if and only if $p \equiv 3 \mod 4$.

Proposition 7.2.40 Let $f(X)$ be a polynomial with coefficients in $\mathbb{F}_q$. If $\beta$ is a zero of $f(X)$, then $\beta^q$ is also a zero of $f(X)$.

Proof. Let $f(X) = f_0 + f_1X + \cdots + f_mX^m \in \mathbb{F}_q[X]$. Then $f_i^q = f_i$ for all $i$. If $\beta$ is a zero of $f(X)$, then $f(\beta) = 0$. So

$$0 = f(\beta^q) = (f_0 + f_1\beta + \cdots + f_m\beta^m)^q = f_0^q + f_1^q\beta^q + \cdots + f_m^q\beta^{qm} = f_0 + f_1\beta^q + \cdots + f_m\beta^{qm} = f(\beta^q).$$

Hence $\beta^q$ is a zero of $f(X)$. ⊓⊔

Remark 7.2.41 In particular we have that $m_1(X) = m_q(X)$.

Let $g(X)$ be a generator polynomial of a cyclic code over $\mathbb{F}_q$. If $\alpha^i$ is a zero of $g(X)$, then $\alpha^{qi}$ is also a zero of $g(X)$.

Definition 7.2.42 The cyclotomic coset $C_q(I)$ of the subset $I$ of $\mathbb{Z}_n$ with respect to $q$ is the subset of $\mathbb{Z}_n$ defined by

$$C_q(I) = \{ iq^j \mid i \in I, \ j \in \mathbb{N}_0\}$$

If $I = \{ i \}$, then $C_q(I)$ is denoted by $C_q(i)$.

Proposition 7.2.43 The cyclotomic cosets $C_q(i)$ give a partitioning of $\mathbb{Z}_n$ for a given $q$ such that $\gcd(q,n) = 1$.

Proof. Every $i \in \mathbb{Z}_n$ is in the cyclotomic coset $C_q(i)$.

Suppose that $C_q(i)$ and $C_q(j)$ have an element in common. Then $iq^k = jq^l$ for some $k, l \in \mathbb{N}_0$. We may assume that $k \leq l$, then $i = jq^{l-k}$ and $l - k \in \mathbb{N}_0$. So $iq^m = jq^{l-k+m}$ for all $m \in \mathbb{N}_0$. Hence $C_q(i)$ is contained in $C_q(j)$. Now $n$ and $q$ are relatively prime, so $q$ is invertible in $\mathbb{Z}_n$ and $q^e \equiv 1 \pmod{n}$ for some positive integer $e$. So $jq^{m} = iq^{(e-1)(l-k)+m}$ for all $m \in \mathbb{N}_0$. Hence $C_q(j)$ is contained in $C_q(i)$. Therefore we have shown that two cyclotomic cosets are equal or disjoint. ⊓⊔
7.2. DEFINING ZEROS

Proposition 7.2.44

\[ m_i(X) = \prod_{j \in C_q(i)} (X - \alpha^j) \]

Proof. If \( j \in C_q(i) \), then \( m_i(\alpha^j) = 0 \) by Proposition 7.2.40. Hence the product \( \prod_{j \in C_q(i)} (X - \alpha^j) \) divides \( m_i(X) \). Now raising to the \( q \)-th power gives a permutation of the zeros \( \alpha^j \) with \( j \in C_q(i) \). The coefficients of the product of the linear factors \( X - \alpha^j \) are symmetric functions in the \( \alpha^j \) and therefore invariant under raising to the \( q \)-th power. Hence these coefficients are elements of \( \mathbb{F}_q \) and this product is an element of \( \mathbb{F}_q[X] \) that has \( \alpha^j \) as a zero. Therefore equality holds by the minimality of \( m_i(X) \).

\[ \diamond \]

Proposition 7.2.45 Let \( n \) be a positive integer such that \( \gcd(n, q) = 1 \). Then the number of choices of an element of order \( n \) in an extension of \( \mathbb{F}_q \) is equal to \( \varphi(n) \). The possible choices of the minimal polynomial \( m_1(X) \) corresponds to monic irreducible factors of \( \Phi_n(X) \) and the number of these choices is \( \varphi(n)/d \), where \( d = |C_q(1)| \).

Proof. The number of choices of an element of order \( n \) in an extension of \( \mathbb{F}_q \) is \( \varphi(n) \) by Theorem 7.2.36. Let \( i \in \mathbb{Z}_n \) and \( \gcd(i, n) = 1 \). Consider the map \( C_q(1) \rightarrow C_q(i) \) defined by \( j \mapsto ij \). Then this map is well defined and has an inverse, since \( i \) is invertible in \( \mathbb{Z}_n \). So \( |C_q(1)| = |C_q(i)| \) and the set of elements in \( \mathbb{Z}_n \) such that \( \gcd(i, n) = 1 \) is partitioned in cyclotomic cosets of the same size \( d \) by Proposition 7.2.43, and every choice of such a coset corresponds to a choice of \( m_1(X) \) and is an irreducible monic factor of \( \Phi_n(X) \). Hence the number of possible minimal polynomials \( m_1(X) \) is \( \varphi(n)/d \).

\[ \diamond \]

Example 7.2.46 Let \( n = 11 \) and \( q = 3 \). Then \( \varphi(11) = 10 \). Consider the sequence starting with 1 and obtained by multiplying repeatedly with 3 modulo 11:

\[ 1, 3, 9, 27 \equiv 5, 15 \equiv 4, 12 \equiv 1. \]

So \( C_3(1) = \{1, 3, 4, 5, 9\} \) consists of 5 elements. Hence \( \Phi_{11}(X) \) has two irreducible factors in \( \mathbb{F}_3[X] \) given by:

\[ \Phi_{11}(X) = \frac{X^{11} - 1}{X - 1} = (-1 + X^2 - X^2 + X^4 + X^5)(-1 - X + X^2 - X^3 + X^5). \]

Example 7.2.47 Let \( n = 23 \) and \( q = 2 \). Then \( \varphi(23) = 22 \). Consider the sequence starting with 1 and obtained by multiplying repeatedly with 2 modulo 23:

\[ 1, 2, 4, 8, 16, 32 \equiv 9, 18, 36 \equiv 13, 26 \equiv 3, 6, 12, 24 \equiv 1. \]

So \( C_2(1) = \{1, 2, 3, 4, 6, 8, 9, 12, 13, 16, 26\} \) consists of 11 elements. Hence \( \Phi_{23}(X) = (X^{23} - 1)/(X - 1) \) is the product two irreducible factors in \( \mathbb{F}_2[X] \) given by:

\[ (1 + X^2 + X^4 + X^5 + X^6 + X^{10} + X^{11})(1 + X + X^5 + X^6 + X^7 + X^9 + X^{11}). \]

Proposition 7.2.48 Let \( i \) and \( j \) be integers such that \( 0 < i, j < n \). Suppose \( ij \equiv 1 \mod n \). Then

\[ m_i(X) = \gcd(m_1(X^j), X^n - 1). \]
Proof. Let \( \beta \) be a zero of \( \gcd(m_1(X^j), X^n - 1) \). Then \( \beta \) is a zero of \( m_1(X^j) \) and \( X^n - 1 \). So \( \beta = \alpha^i \) for some \( i \) and \( m_1(\alpha^i) = 0 \). Hence \( j l \in C_q(1) \) by Proposition 7.2.44. So \( j l \equiv q^m \mod n \) for some \( m \). Hence \( l \equiv ij \equiv iq^m \mod n \). Therefore \( l \in C_q(i) \) and \( \beta \) is a zero of \( m_i(X) \).

Similarly, if \( \beta \) is a zero of \( m_i(X) \), then \( \beta \) is a zero of \( \gcd(m_1(X^j), X^n - 1) \). Both polynomials are monic and have the same zeros and all zeros are simple by Remark 7.2.23. Therefore the polynomials are equal.

\[ \square \]

**Proposition 7.2.49** Let \( \gcd(i,n) = d \) and \( j = n/d \). Let \( \alpha \) be an element of order \( n \) in \( \mathbb{F}_q^* \) and \( \beta = \alpha^d \). Let \( m_i(X) \) be the minimal polynomial of \( \alpha^i \) and \( n_j(X) \) the minimal polynomial of \( \beta^j \). Then \( \beta \) is an element of order \( n/d \) in \( \mathbb{F}_q^* \) and \( m_i(X) = n_j(X) \).

**Proof.** The map \( jq^m \mapsto jdq^m \) gives a well defined one-to-one correspondence between elements of \( D \), the cyclotomic coset of \( j \) modulo \( n/d \) and the elements of \( C \), the cyclotomic coset of \( i \) modulo \( n \). Hence

\[
m_i(X) = \prod_{k \in C}(X - \alpha^k) = \prod_{l \in D}(X - \beta^l) = n_j(X)
\]

by Proposition 7.2.44.

\[ \square \]

**Example 7.2.50** Let \( \alpha \) be an element of order 8 in an extension of \( \mathbb{F}_3 \). Let \( m_3(X) \) be the minimal polynomial of \( \alpha \) in \( \mathbb{F}_3[X] \). Then \( m_3(X) \) divides \( X^8 - 1 \). But \( X^8 - 1 = (X^4 - 1)(X^4 + 1) \) and the zeros of \( X^4 - 1 \) have order at most 4. The factorization of \( X^4 - 1 \) is given by

\[
X^4 - 1 = (X - 1)(X + 1)(X^2 + 1)
\]

with \( m_6(X) = X - 1 \) and \( m_4(X) = X + 1 \), since \( \alpha^4 = -1 \). The cyclotomic coset of 2 is \( C_3(2) = \{2, 6\} \) and \( \alpha^2 \) and \( \alpha^6 \) are the elements of order 4 in \( \mathbb{F}_3 \). So

\[
m_4(X) = m_6(X) = \Phi_4(X) = X^2 + 1.
\]

This confirms Proposition 7.2.49 with \( i = d = 2 \) and \( j = 1 \).

Now \( C_8(1) = \{1, 3\} \) and \( C_8(5) = \{5, 7\} \). So \( m_1(X) = m_3(X) \) and \( m_5(X) = m_7(X) \). Notice that \( -1 \equiv 7 \mod 8 \) and \( m_7(X) \) is the monic reciprocal of \( m_1(X) \) by Proposition 7.2.29. The degree of \( m_1(X) \) is 2. Suppose

\[
m_1(X) = a_0 + a_1X + X^2.
\]

Then \( m_7(X) = a_0^{-1} + a_1^{-1}X + X^2 \). The polynomials \( m_1(X) \) and \( m_7(X) \) divide \( X^4 + 1 \). Hence

\[
\Phi_8(X) = X^4 + 1 = m_1(X)m_7(X) = (a_0 + a_1X + X^2)(a_0^{-1} + a_1^{-1}X + X^2).
\]

Expanding the right hand side and comparing coefficients gives that \( a_0 = -1 \) and \( a_1 = 1 \) or \( a_1 = -1 \). Hence there are two possible choices for \( m_1(X) \). Choose \( m_1(X) = X^2 + X - 1 \). So \( X^2 - X - 1 \) is the alternative choice for \( m_1(X) \). Furthermore \( \alpha^2 = -\alpha + 1 \) and \( m_5(X) = m_7(X) = (X - \alpha^2)(X - \alpha^7) \) by Proposition 7.2.44. An application of Proposition 7.2.48 with \( i = j = 5 \), gives a third way to compute \( m_5(X) \) since \( 5 \cdot 5 \equiv 1 \mod 8 \), and \( m_1(X^5) = X^{10} + X^5 - 1 \) and

\[
\gcd(X^{10} + X^5 - 1, X^8 - 1) = X^2 - X - 1.
\]
7.2. DEFINING ZEROS

7.2.4 Zeros of the generator polynomial

We have seen in Proposition 7.1.15 that the generator polynomial divides $X^n - 1$, so its zeros are $n$-th roots of unity if $n$ is not divisible by the characteristic of $\mathbb{F}_q$. Instead of describing a cyclic code by its generator polynomial $g(X)$, one can describe the code alternatively by the set of zeros of $g(X)$ in an extension of $\mathbb{F}_q$.

**Definition 7.2.51** Fix an element $\alpha$ of order $n$ in an extension $\mathbb{F}_{q^n}$ of $\mathbb{F}_q$. A subset $I$ of $\mathbb{Z}_n$ is called a defining set of a cyclic code $C$ if

$$C = \{ c(x) \in \mathbb{C}_{q,n} : c(\alpha^i) = 0 \text{ for all } i \in I \}.$$  

The root set, the set of zeros or the complete defining set $Z(C)$ of $C$ is defined as

$$Z(C) = \{ i \in \mathbb{Z}_n : c(\alpha^i) = 0 \text{ for all } c(x) \in C \}.$$

**Proposition 7.2.52** The relation between the generator polynomial $g(X)$ of a cyclic code $C$ and the set of zeros $Z(C)$ is given by

$$g(X) = \prod_{i \in Z(C)} (X - \alpha^i).$$

The dimension of $C$ is equal to $n - |Z(C)|$.

**Proof.** The generator polynomial $g(X)$ divides $X^n - 1$ by Proposition 7.1.15. The polynomial $X^n - 1$ has no multiple zeros, by Remark 7.2.23 since $n$ and $q$ are relatively prime. So every zero of $g(X)$ is of the form $\alpha^i$ for some $i \in \mathbb{Z}_n$ and has multiplicity one. Let $Z(g) = \{ i \in \mathbb{Z}_n \mid g(\alpha^i) = 0 \}$. Then $g(X) = \prod_{i \in Z(g)} (X - \alpha^i)$. Let $c(x) \in C$. Then $c(x) = a(x)g(x)$, so $c(\alpha^i) = 0$ for all $i \in Z(g)$. So $Z(g) \subseteq Z(C)$. Conversely, $g(x) \in C$, so $g(\alpha^i) = 0$ for all $i \in Z(C)$. Hence $Z(C) \subseteq Z(g)$. Therefore $Z(g) = Z(C)$ and $g(X)$ is a product of the linear factors as claimed. Furthermore the degree of $g(X)$ is equal to $|Z(C)|$. Hence the dimension of $C$ is equal to $n - |Z(C)|$ by Proposition 7.1.21.

**Proposition 7.2.53** The complete defining set of a cyclic code is the disjoint union of cyclotomic cosets.

**Proof.** Let $g(X)$ be the generator polynomial of a cyclic code $C$. Then $g(\alpha^i) = 0$ if and only if $i \in Z(C)$ by Proposition 7.2.52. If $\alpha^i$ is a zero of $g(X)$, then $\alpha^{iq}$ is a zero of $g(X)$ by Remark 7.2.41. So $C_q(i)$ is contained in $Z(C)$ if $i \in Z(C)$. Therefore $Z(C)$ is the union of cyclotomic cosets. This union is a disjoint union by Proposition 7.2.43.

**Example 7.2.54** Consider the binary cyclic code $C$ of length 7 with defining set $\{1\}$. Then $Z(C) = \{1, 2, 4\}$ and $m_1(X) = 1 + X + X^3$ is the generator polynomial of $C$. Hence $C$ is the cyclic Hamming code of Example 7.1.19. The cyclic code with defining set $\{3\}$ has generator polynomial $m_3(X) = 1 + X^2 + X^3$ and complete defining set $\{3, 5, 6\}$.
Remark 7.2.55 If a cyclic code is given by its zero set, then this definition depends on the choice of an element of order \( n \). Consider Example 7.2.54. If we would have taken \( \alpha^3 \) as element of order 7, then the generator polynomial of the binary cyclic code with defining set \{1\} would have been \( 1 + X^2 + X^3 \) instead of \( 1 + X + X^3 \).

Example 7.2.56 Consider the [6,3] cyclic code over \( \mathbb{F}_7 \) of Example 7.1.24 with the generator polynomial \( g(X) = 6 + X + 3X^2 + X^3 \). Then
\[
X^3 + 3X^2 + X + 6 = (X - 2)(X - 3)(X - 6).
\]
So 2, 3 and 6 are the zeros of the generator polynomial. Choose \( \alpha = 3 \) as the primitive element in \( \mathbb{F}_7 \) of order 6 as in Example 7.2.27. Then \( \alpha, \alpha^2 \) and \( \alpha^3 \) are the zeros of \( g(X) \).

Example 7.2.57 Let \( \alpha \) be an element of \( \mathbb{F}_9 \) such that \( \alpha^2 = -\alpha + 1 \) as in Example 7.2.50. Then \( 1, \alpha \) and \( \alpha^3 \) are the zeros of the ternary cyclic code of length 8 with generator polynomial \( 1 + X + X^3 \) of Example 7.1.16, since
\[
X^3 + X + 1 = (X^2 + X - 1)(X - 1) = m_1(X)m_0(X).
\]

Proposition 7.2.58 Let \( C \) be a cyclic code of length \( n \). Then
\[
Z(C^⊥) = \mathbb{Z}_n \setminus \{ -i \mid i \in Z(C) \}.
\]

Proof. The power \( \alpha^i \) is a zero of \( g(X) \) if and only if \( i \in Z(C) \) by Proposition 7.2.52. And \( h(\alpha^i) = 0 \) if and only if \( g(\alpha^i) \neq 0 \), since \( h(X) = (X^n - 1)/g(X) \) and all zeros of \( X^n - 1 \) are simple by Remark 7.2.23. Furthermore \( g^i(X) \) is the monic reciprocal of \( h(X) \) by Proposition 7.1.37. Finally \( g^i(\alpha^{-i}) = 0 \) if and only if \( h(\alpha^i) = 0 \) by Remark 7.1.30.

Example 7.2.59 Consider the [6,3] cyclic code \( C \) over \( \mathbb{F}_7 \) of Example 7.2.56. Then \( \alpha, \alpha^2 \) and \( \alpha^3 \) are the zeros of \( g(X) \). Hence \( Z(C) = \{ 1, 2, 3 \} \) and
\[
Z(C^⊥) = \mathbb{Z}_6 \setminus \{ -1, -2, -3 \} = \{ 0, 1, 2 \},
\]
by Proposition 7.2.58. Therefore
\[
g^i(X) = (X - 1)(X - \alpha)(X - \alpha^2) = (X - 1)(X - 3)(X - 2) = X^3 + X^2 + 2X + 1.
\]
This is in agreement with Example 7.1.38.

Example 7.2.60 Let \( C \) be the ternary cyclic code of length 8 with the generator polynomial \( g(X) = 1 + X + X^3 \) of Example 7.1.16. Then \( g(X) = m_0(X)m_1(X) \) and \( Z(C) = \{ 0, 1, 3 \} \) by Example 7.2.57. Hence
\[
Z(C^⊥) = \mathbb{Z}_8 \setminus \{ 0, -1, -3 \} = \{ 1, 2, 3, 4, 6 \}.
\]

Proposition 7.2.61 The number of cyclic codes of length \( n \) over \( \mathbb{F}_q \) is equal to \( 2^N \), where \( N \) is the number of cyclotomic cosets modulo \( n \) with respect to \( q \).

Proof. A cyclic code \( C \) of length \( n \) over \( \mathbb{F}_q \) is uniquely determined by its set of zeros \( Z(C) \) by Proposition 7.2.52. The set of zeros is a disjoint union of cyclotomic cosets modulo \( n \) with respect to \( q \) by Proposition 7.2.53. Hence a cyclic code is uniquely determined by a choice of a subset of all \( N \) cyclotomic cosets. There are \( 2^N \) of such subsets. \( \diamond \)
7.2. DEFINING ZEROS

Example 7.2.62 There are 3 cyclotomic cosets modulo 7 with respect to 2. Hence there are 8 binary cyclic codes of length 7 with generator polynomials

\[ 1, m_0, m_1, m_3, m_0m_1, m_0m_3, m_1m_3, m_0m_1m_3. \]

7.2.5 Exercises

7.2.1 Show that \( f(X) = 1 + X^2 + X^5 \) is irreducible in \( \mathbb{F}_2[X] \). Give a principal representation of the product of \( \beta = 1 + x + x^4 \) and \( \gamma = 1 + x^3 + x^4 \) in the factor ring \( \mathbb{F}_2[X]/(f(X)) \) by division by \( f(X) \) with rest. Give a table of the principal representation and the Zech logarithm of the powers \( x^i \) for \( i \in \{*, 0, 1, \ldots, 30\} \). Compute the addition of \( \beta \) and \( \gamma \) by means of the exponential representation.

7.2.2 What is the smallest field extension of \( \mathbb{F}_q \) that has an element of order 37 in case \( q = 2, 3 \) and 5? Show that the degree of the extension is always a divisor of 36 for any prime power \( q \) not divisible by 37.

7.2.3 Determine \( \Phi_q(X) \) in characteristic zero. Let \( p \) be an odd prime. Show that \( \Phi_q(X) \) is irreducible in \( \mathbb{F}_p[X] \) if and only if \( p \equiv 2 \mod 3 \).

7.2.4 Let \( \alpha \) be an element of order 8 in an extension of \( \mathbb{F}_5 \). Give all possible choices of the minimal polynomial \( m_1(X) \). Compute the coefficients of \( m_i(X) \) for all \( i \) between 0 and 7.

7.2.5 Let \( \alpha \) be an element of order \( n \) in \( \mathbb{F}_{q^n}^* \). Let \( m_1(X) \) be the minimal polynomial of \( \alpha \) in \( \mathbb{F}_q \). Estimate the total number of arithmetical operations in \( \mathbb{F}_q \) to compute the minimal polynomial \( m_i(X) \) by means of Proposition 7.2.44 if \( \gcd(i, n) = 1 \) as a function of \( n \) and \( e \). Compare this complexity with the computation by means of Proposition 7.2.48.

7.2.6 Let \( C \) be a cyclic code of length 7 over \( \mathbb{F}_q \). Show that \( \{1, 2, 4\} \) is a complete defining set if \( q \) is even.

7.2.7 Compute the zeros of the code of Example 7.1.5.

7.2.8 Show that \( \alpha = 5 \) is an element of order 6 in \( \mathbb{F}_7 \). Give the coefficients of the generator polynomial of the cyclic [6,3] code over \( \mathbb{F}_7 \) with \( \alpha, \alpha^2 \) and \( \alpha^3 \) as zeros.

7.2.9 Consider the binary cyclic code \( C \) of length 31 and generator polynomial \( 1 + X^2 + X^5 \) of Exercise 7.1.6. Let \( \alpha \) be a zero of this polynomial. Then \( \alpha \) has order 31 by Exercise 7.2.1.

1) Determine the coefficients of \( m_1(X), m_3(X) \) and \( m_5(X) \) with respect to \( \alpha \).
2) Determine \( Z(C) \) and \( Z(C^+) \).

7.2.10 Let \( C \) be a cyclic code over \( \mathbb{F}_5 \) with \( m_1(X)m_2(X) \) as generator polynomial. Determine \( Z(C) \) and \( Z(C^+) \).

7.2.11 What is the number of ternary cyclic codes of length 8?

7.2.12 [CAS] Using a function \texttt{MoebiusMu} from GAP and Magma write a program that computes the number of irreducible polynomials of given degree as per Proposition 7.2.19. Check your result with the use of the function \texttt{IrreduciblePolynomialsNr} in GAP.
7.2.13 [CAS] Take the field $\mathbb{GF}(2^{10})$ and its primitive element $a$. Compute the Zech logarithm of $a^{100}$ with respect to $a$ using commands ZechLog both in GAP and Magma.

7.2.14 [CAS] Using the function CyclotomicCosets in GAP/GUAVA write a function that takes as an input the code length $n$, the field size $q$ and a list of integers $L$ and computes dimension of a $q$-ary cyclic code which definig set is $\{a^i | i \in L\}$ for some primitive $n$-th root of unity $a$ (predefined in GAP is fine).

7.3 Bounds on the minimum distance

The BCH bound is a lower bound for the minimum distance of a cyclic code. Although this bound is tight in many cases, it is not always the true minimum distance. In this section several improved lower bounds are given but not one of them gives the true minimum distance in all cases. In fact computing the true minimum distance of a cyclic code is a hard problem.

7.3.1 BCH bound

Definition 7.3.1 Let $C$ be an $\mathbb{F}_q$-linear code. Let $\tilde{C}$ be an $\mathbb{F}_{q^m}$-linear code in $\mathbb{F}_{q^m}^n$. If $C \subseteq \tilde{C} \cap \mathbb{F}_q^n$, then $C$ is called a subfield subcode of $\tilde{C}$, and $\tilde{C}$ is called a super code of $C$. If equality holds, then $C$ is called the restriction (by scalars) of $\tilde{C}$.

Remark 7.3.2 Let $I$ be a defining set for the cyclic code $C$. Then

$$c(a^i) = c_0 + c_1 a^i + \cdots + c_j a^{ij} + \cdots + c_{n-1} a^{i(n-1)} = 0$$

for all $i \in I$. Let $l = |I|$. Let $\tilde{H}$ be the $l \times n$ matrix with entries

$$\left( a^{ij} | i \in I, j = 0, 1, \ldots, n-1 \right).$$

Let $\tilde{C}$ be the $\mathbb{F}_{q^m}$-linear code with $\tilde{H}$ as parity check matrix. Then $C$ is a subfield subcode of $\tilde{C}$, and it is in fact its restriction by scalars. Any lower bound on the minimum distance of $\tilde{C}$ holds a fortiori for $C$.

This remark will be used in the following proposition on the BCH (Bose-Chaudhuri-Hocquenghem) bound on the minimum distance for cyclic codes.

Proposition 7.3.3 Let $C$ be a cyclic code that has at least $\delta - 1$ consecutive elements in $Z(C)$. Then the minimum distance of $C$ is at least $\delta$.

Proof. The complete defining set $C$ contains $\{b \leq i \leq b+\delta - 2\}$ for a certain $b$. We have seen in Remark 7.3.2 that $\left( a^{ij} | b \leq i \leq b+\delta - 2, 0 \leq j < n \right)$ is a parity check matrix of a code $\tilde{C}$ over $\mathbb{F}_q^m$ that has $C$ as a subfield subcode. The $j$-th column has entries $a^{b+\delta - 2} a^{ij}, 0 \leq i \leq \delta - 2$. The code $\tilde{C}$ is generalized equivalent with the code with parity check matrix $\tilde{H}' = (a^{ij})$ with $0 \leq i \leq \delta - 2, 0 \leq j < n$, by the linear isometry that divides the $j$-th coordinate by $a^{b+\delta - 2}$ for $0 \leq j < n$. Let $x_j = a^{ij-1}$ for $1 \leq j \leq n$. Then $H' = (x_j^i | 0 \leq i \leq \delta - 2, 1 \leq j \leq n)$ is a
Definition 7.3.4 A cyclic code with defining set \( \{b, b+1, \ldots, b+\delta-2\} \) is called a BCH code with designed minimum distance \( \delta \). The BCH code is called narrow sense if \( b = 1 \), and it is called primitive if \( n = q^m - 1 \).

Example 7.3.5 Consider the binary cyclic Hamming code \( C \) of length 7 of Example 7.2.28. The complete defining set of \( C \) is \( \{1, 2, 4\} \) and contains two consecutive elements. So 3 is a lower bound for the minimum distance. This is equal to the minimum distance. Let \( D \) be the binary cyclic code of length 7 with defining set \( \{0, 3\} \). Then the complete defining set of \( D \) is \( \{0, 3, 5, 6\} \). So 5, 6, 7 are three consecutive elements in \( Z(D) \), since 7 \( \equiv 0 \mod 7 \). So 4 is a lower bound for the minimum distance of \( D \). In fact equality holds, since \( D \) is the dual of \( C \) that is the \([7, 3, 4]\) binary simplex code.

Example 7.3.6 Consider the \([6,3]\) cyclic code \( C \) over \( \mathbb{F}_7 \) of Example 7.2.56. The zeros of the generator polynomial are \( \alpha, \alpha^2 \) and \( \alpha^3 \). So \( Z(C) = \{1, 2, 3\} \) and \( d(C) \geq 4 \). Now \( g(x) = 6 + x + 3x^2 + x^3 \) is a codeword of weight 4. Hence the minimum distance is 4.

Remark 7.3.7 If \( \alpha \) and \( \beta \) are both elements of order \( n \), then there exist \( r, s \) in \( \mathbb{Z}_n \) such that \( \beta = \alpha^r \) and \( \alpha = \beta^s \) and \( rs \equiv 1 \mod n \) by Theorem 7.2.36. If \( C \) is the cyclic code with defining set \( I \) with respect to \( \alpha \), and \( D \) is the cyclic code with defining set \( I \) with respect to \( \beta \), then \( C \) and \( D \) are equivalent under the permutation \( \sigma \) of \( \mathbb{Z}_n \) such that \( \sigma(0) = 0 \) and \( \sigma(i) = ir \) for \( i = 1, \ldots, n-1 \). Hence a cyclic code that is given by its zero set is defined up to equivalence by the choice of an element of order \( n \).

Example 7.3.8 Consider the binary cyclic code \( C_1 \) of length 17 and \( m_1(X) \) as generator polynomial. Then

\[
Z_1 = \{-, 1, 2, -4, -, -3, 8, 9, -, -, -13, -, -15, 16\}
\]

is the complete defining set of \( C_1 \), where the spacing \(-\) indicates a gap. The BCH bound gives 3 as a lower bound for the minimum distance of \( C_1 \). The code \( C_3 \) with generator polynomial \( m_3(X) \) has complete defining set

\[
Z_3 = \{-, -, -3, -5, 6, 7, -, -, 10, 11, 12, -, 14, -, -\}. 
\]

Hence 4 is a lower bound for \( d(C_3) \). The cyclic codes of length 17 with generator polynomial \( m_i(X) \) are equivalent if \( i \neq 0 \), by Remark 7.3.7, since the order of \( \alpha^i \) is 17. Hence \( d(C_1) = d(C_3) \geq 4 \). In Example 7.4.2 it will be shown that the minimum distance is in fact 5.

The following definition does not depend on the choice of an element of order \( n \).
Definition 7.3.9 A subset of $\mathbb{Z}_n$ of the form \( \{ b + ia | 0 \leq i \leq \delta - 2 \} \) for some integers \( a, b \) and \( \delta \) with \( \gcd(a, n) = 1 \) and \( \delta \leq n + 1 \) is called consecutive of period \( a \). Let \( I \) be a subset of \( \mathbb{Z}_n \). The number \( \delta_{BCH}(I) \) is the largest integer \( \delta \leq n+1 \) such that there is a consecutive subset of \( I \) consisting of \( \delta - 1 \) elements. Let \( C \) be a cyclic code of length \( n \). Then \( \delta_{BCH}(Z(C)) \) is denoted by \( \delta_{BCH}(C) \).

Theorem 7.3.10 The minimum distance of \( C \) is at least \( \delta_{BCH}(C) \).

Proof. Let \( \alpha \) be an element of order \( n \) in an extension of \( \mathbb{F}_q \). Suppose that the complete defining set of \( C \) with respect to \( \alpha \) contains the set \( \{ b + ia | 0 \leq i \leq \delta - 2 \} \) of \( \delta - 1 \) elements for some integers \( a \) and \( b \) with \( \gcd(a, n) = 1 \). Let \( \beta = \alpha^b \). Then \( \beta \) is an element of order \( n \) and there is an element \( c \in \mathbb{Z}_n \) such that \( ac = 1 \), since \( \gcd(a, n) = 1 \). Hence \( \{ bc + ia | 0 \leq i \leq \delta - 2 \} \) is a defining set of \( C \) with respect to \( \beta \) containing \( \delta - 1 \) consecutive elements. Hence the minimum distance of \( C \) is at least \( \delta \) by Proposition 7.3.3.

Remark 7.3.11 One easily sees a consecutive set of period one in \( Z(C) \) by writing the elements in increasing order and the gaps by a spacing as done in Example 7.3.8. Suppose \( \gcd(a, n) = 1 \). Then there exists a \( b \) such that \( ab \equiv 1 \pmod{n} \). A consecutive set of period \( a \) is seen by considering \( b \cdot Z(C) \) and its consecutive sets of period 1. In this way one has to inspect \( \varphi(n) \) complete defining sets for its consecutive sets of period 1. The complexity of this computation is at most \( \varphi(n)|Z(C)| \) in the worst case. But quite often it is much less in case \( b \cdot Z(C) = Z(C) \).

Example 7.3.12 This is a continuation of Example 7.3.8. The complete defining set \( Z_3 \) of the code \( C_3 \) has \( \{ 5, 6, 7 \} \) as largest consecutive subset of period 1 in \( \mathbb{Z}_{17} \). Now \( 3 \cdot 6 \equiv 1 \pmod{17} \) and \( 6 \cdot \{ 5, 6, 7 \} = \{ 13, 2, 8 \} \) is a consecutive subset of period 6 in \( \mathbb{Z}_{17} \) of three elements contained in the complete defining set \( Z_1 \) of the code \( C_1 \). Now \( b \cdot Z_1 \) is equal to \( Z_1 \) or \( Z_3 \) for all \( 0 < b < 17 \). Hence \( \delta_{BCH}(C_1) = \delta_{BCH}(C_3) = 4 \).

Example 7.3.13 Consider the binary BCH code \( C_b \) of length 15 and with defining set \( \{ b, b + 1, b + 2, b + 3 \} \) for some \( b \). So its designed distance is 5. Take \( \alpha \) in \( \mathbb{F}_{16}^\ast \) with \( \alpha^4 = 1 + \alpha \) as primitive element. Then \( m_0(X) = 1 + X, m_1(X) = 1 + X + X^4, m_3(X) = 1 + X + X^2 + X^3 + X^4 \) and \( m_5(X) = 1 + X + X^2 \). If \( b = 1 \), then the complete defining set is \( \{ 1, 2, 3, 4, 6, 8, 9, 12 \} \) so \( \delta_{BCH}(C_1) = 5 \). The generator polynomial is \( g_1(X) = m_1(X)m_3(X) = 1 + X^4 + X^6 + X^7 + X^8 \) as is shown in Example 7.1.39 and has weight 5. So the minimum distance of \( C_1 \) is 5.

If \( b = 0 \), then \( \delta_{BCH}(C_0) = 6 \). The generator polynomial is
\[
g_0(X) = m_0(X)m_1(X)m_3(X) = 1 + X + X^4 + X^5 + X^6 + X^9
\]
and has weight 6. So the minimum distance \( C_0 \) is 6.

If \( b = 2 \) or \( b = 3 \), then \( \delta_{BCH}(C_2) = 7 \). The generator polynomial is \( g_2(X) = 1 + X + X^2 + X^4 + X^5 + X^8 + X^{10} \) and has weight 7. So the minimum distance \( C_2 \) is 7. If \( b = 4 \) or \( b = 5 \), then \( \delta_{BCH}(C_4) = 15 \). The generator polynomial is \( g_4(X) = 1 + X + X^2 + X^4 + X^8 + X^{12} + X^{13} + X^{14} \) and has weight 15. So the minimum distance \( C_4 \) is 15.
7.3. **BOUNDS ON THE MINIMUM DISTANCE**

**Example 7.3.14** Consider the primitive narrow sense BCH code of length 15 over $\mathbb{F}_{16}$ with designed distance 5. So the defining set is $\{1, 2, 3, 4\}$. Then this is also the complete defining set. Take $\alpha$ with $\alpha^4 = 1 + \alpha$ as primitive element. Then the generator polynomial is given by

$$(X - \alpha)(X - \alpha^2)(X - \alpha^3)(X - \alpha^4) = \alpha^{10} + \alpha^3X + \alpha^6X^2 + \alpha^{13}X^3 + X^4.$$ 

In all these cases of the previous two examples the minimum distance is equal to the BCH bound and equal to the weight of the generator polynomial. This is not always the case as one see in Exercise 7.3.8

**Example 7.3.15** Although BCH codes are a special case of codes defined through roots as in Section 7.2.4, GAP and Magma have special functions for constructing these. In GAP/GUAVA we proceed as follows. 

```plaintext
> C:=BCHCode(17,3,GF(2));
a cyclic [17,9,5] BCH code, delta=3, b=1 over GF(2)
> DesignedDistance(C);
3
> MinimumDistance(C);
5
```

Syntax is `BCHCode(n,delta,F)`, where $n$ is the length, $\delta$ is $\delta$ in Definition 7.3.4, and $F$ is the ground field. So here we constructed the narrow-sense BCH code. One can give the parameter $b$ explicitly, by the command `BCHCode(n,b,delta,F)`. The designed distance for a BCH code is printed in its description, but can also be called separately as above. Note that code $C$ coincides with the code $CR$ from Example 12.5.17.

In Magma we proceed as follows.

```plaintext
> C:=BCHCode(GF(2),17,3); // [17, 9, 5] "BCH code (d = 3, b = 1)" // Linear Code over GF(2)
> a:=RootOfUnity(17,GF(2));
> C:=CyclicCode(17,[a^3],GF(2));
> BCHBound(C);
4
```

We can also provide $b$ giving it as the last parameter in the `BCHCode` command. Note that there is a possibility in Magma to compute the BCH bound as above.

### 7.3.2 Quadratic residue codes

**Example 7.3.16** Consider an $\mathbb{F}_q$-linear cyclic code of length $n = (q^r - 1)/(q-1)$ with defining set $\{1\}$. Let $\alpha$ be an element of order $n$ in $\mathbb{F}_q^*$. The minimum distance of the code is at least 2, by the BCH bound. If $\gcd(r, q-1) = i > 1$, then $i$ divides $n$, since

$$n = \frac{q^r - 1}{q - 1} = q^{r-1} + \cdots + q + 1 \equiv r \mod (q-1).$$

Let $j = n/i$. Let $c_0 = -\alpha^j$. Then $c_0 \in \mathbb{F}_q^*$, since $j(q-1) = n(q-1)/i$ is a multiple of $n$. So $c(x) = c_0 + x^j$ is a codeword of weight 2 and the minimum
distance is 2. Now consider the case with \( q = 3 \) and \( r = 2 \) in particular. Then \( \alpha \in F_9^* \) is an element of order 4 and \( c(x) = 1 + x^2 \) is a codeword of the ternary cyclic code of length 4 with defining set \{1\}. So this code has parameters \([4,2,2]\).

**Proposition 7.3.17** Let \( n = (q^r - 1)/(q-1) \). If \( r \) is relatively prime with \( q - 1 \), then the \( F_q \)-linear cyclic code of length \( n \) with defining set \{1\} is a generalized \([n,n-r,3]\) Hamming code.

**Proof.** Let \( \alpha \) be an element of order \( n \) in \( F_q^* \). The minimum distance of the code is at least 2 by the BCH bound. Suppose there is a codeword \( c(x) \) of weight 2 with nonzero coefficients \( c_i \) and \( c_j \) with \( 0 \leq i < j < n \). Then \( c(\alpha) = 0 \). So \( c_i \alpha^i + c_j \alpha^j = 0 \). Hence \( \alpha^{j-i} = -c_i/c_j \). Therefore \( \alpha^{j-i}(q-1) = 1 \), since \( -c_i/c_j \in F_q^* \). Now \( n|\(j-i)\(q-1)\) \), but since \( \text{gcd}(n,q-1) = \text{gcd}(r,q-1) = 1 \) by assumption, it follows that \( n|\(j-i) \), which is a contradiction. Hence the minimum distance is at least 3. Therefore the parameters are \([n,n-r,3]\) and the code is equivalent with the Hamming code \( H_r(q) \) by Proposition 2.5.19.

**Corollary 7.3.18** The simplex code \( S_r(q) \) is equivalent with a cyclic code if \( r \) is relatively prime with \( q - 1 \).

**Proof.** The dual of a cyclic code is cyclic by Proposition 7.1.6 and a simplex code is by definition the dual of a Hamming code. So the statement is a consequence of Proposition 7.3.17.

**Proposition 7.3.19** The binary cyclic code of length 23 with defining set \{1\} is equivalent to the binary \([23,12,7]\) Golay code.

**Proof.***

**Proposition 7.3.20** The ternary cyclic code of length 11 with defining set \{1\} is equivalent to the ternary \([11,6,5]\) Golay code.

**Proof.***

*** Show that there are two generator polynomials of a ternary cyclic code of length 11 with defining set \{1\}, depending on the choice of an element of order 11. Give the coefficients of these generator polynomials. ***

### 7.3.4 Exercises

**7.3.1** Let \( C \) be the binary cyclic code of length 9 and defining set \{0,1\}. Give the BCH bound of this code.

**7.3.2** Show that a nonzero binary cyclic code of length 11 has minimum distance 1, 2 or 11.

**7.3.3** Choose the primitive element as in Exercise 7.2.9. Give the coefficients of the generator polynomial of a cyclic \( H_5(2) \) Hamming code and give a word of weight 3.
7.4 Improvements of the BCH bound

7.4.1 Hartmann-Tzeng bound

Proposition 7.4.1 Let $C$ be a cyclic code of length $n$ with defining set $I$. Let $U_1$ and $U_2$ be two consecutive sets in $\mathbb{Z}_n$ consisting of $\delta_1 - 1$ and $\delta_2 - 1$ elements, respectively. Suppose that $U_1 + U_2 \subseteq I$. Then the minimum distance of $C$ is at least $\delta_1 + \delta_2 - 2$.

Proof. This is a special case of the forthcoming Theorem 7.4.19 and Proposition 7.4.20. ***direct proof***

Example 7.4.2 Consider the binary cyclic code $C_3$ of length 17 and defining set $\{3\}$ of Example 7.3.8. Then Proposition 7.4.1 applies with $U_1 = \{5, 6, 7\}$, $U_2 = \{0, 5\}$, $\delta_1 = 4$ and $\delta_2 = 3$. Hence the minimum distance of $C_3$ is at least 5. The factorization of $1 + X^{17}$ in $\mathbb{F}_2[X]$ is given by

$$(1 + X)(1 + X^3 + X^4 + X^5 + X^8)(1 + X + X^2 + X^4 + X^6 + X^7 + X^8).$$

Let $\alpha$ be a zero of the second factor. Then $\alpha$ is an element of $\mathbb{F}_2^8$ of order 17. Hence $m_1(X)$ is the second factor and $m_3(X)$ is the third factor. Now $1 + x^3 + x^4 + x^5 + x^8$ is a codeword of $C_1$ of weight 5. Furthermore $C_1$ and $C_3$ are equivalent. Hence $d(C_3) = 5$.

Definition 7.4.3 For a subset $I$ of $\mathbb{Z}_n$, let $\delta_{HT}(I)$ be the largest number $\delta$ such that there exist two nonempty consecutive sets $U_1$ and $U_2$ in $\mathbb{Z}_n$ consisting of $\delta_1 - 1$ and $\delta_2 - 1$ elements, respectively, with $U_1 + U_2 \subseteq I$ and $\delta = \delta_1 + \delta_2 - 2$. Let $C$ be a cyclic code of length $n$. Then $\delta_{HT}(Z(C))$ is denoted by $\delta_{HT}(C)$. 

7.4.2 Hartmann-Tzeng bound

Proposition 7.4.1 Let $C$ be a cyclic code of length $n$ with defining set $I$. Let $U_1$ and $U_2$ be two consecutive sets in $\mathbb{Z}_n$ consisting of $\delta_1 - 1$ and $\delta_2 - 1$ elements, respectively. Suppose that $U_1 + U_2 \subseteq I$. Then the minimum distance of $C$ is at least $\delta_1 + \delta_2 - 2$.

Proof. This is a special case of the forthcoming Theorem 7.4.19 and Proposition 7.4.20. ***direct proof***

Example 7.4.2 Consider the binary cyclic code $C_3$ of length 17 and defining set $\{3\}$ of Example 7.3.8. Then Proposition 7.4.1 applies with $U_1 = \{5, 6, 7\}$, $U_2 = \{0, 5\}$, $\delta_1 = 4$ and $\delta_2 = 3$. Hence the minimum distance of $C_3$ is at least 5. The factorization of $1 + X^{17}$ in $\mathbb{F}_2[X]$ is given by

$$(1 + X)(1 + X^3 + X^4 + X^5 + X^8)(1 + X + X^2 + X^4 + X^6 + X^7 + X^8).$$

Let $\alpha$ be a zero of the second factor. Then $\alpha$ is an element of $\mathbb{F}_2^8$ of order 17. Hence $m_1(X)$ is the second factor and $m_3(X)$ is the third factor. Now $1 + x^3 + x^4 + x^5 + x^8$ is a codeword of $C_1$ of weight 5. Furthermore $C_1$ and $C_3$ are equivalent. Hence $d(C_3) = 5$.

Definition 7.4.3 For a subset $I$ of $\mathbb{Z}_n$, let $\delta_{HT}(I)$ be the largest number $\delta$ such that there exist two nonempty consecutive sets $U_1$ and $U_2$ in $\mathbb{Z}_n$ consisting of $\delta_1 - 1$ and $\delta_2 - 1$ elements, respectively, with $U_1 + U_2 \subseteq I$ and $\delta = \delta_1 + \delta_2 - 2$. Let $C$ be a cyclic code of length $n$. Then $\delta_{HT}(Z(C))$ is denoted by $\delta_{HT}(C)$. 

7.3.4 Choose the primitive element as in Exercise 7.2.9. Consider the binary cyclic code $C$ of length 31 and generator polynomial $m_0(X)m_1(X)m_3(X)m_5(X)$. Show that $C$ has dimension 15 and $\delta_{BCH}(C) = 8$. Give a word of weight 8.

7.3.5 Determine $\delta_{BCH}(C)$ for all the binary cyclic codes $C$ of length 17.

7.3.6 Show the existence of a binary cyclic code of length 127, dimension 64 and minimum distance at least 21.

7.3.7 Let $C$ be the ternary cyclic code of length 13 with complete defining set $\{1, 3, 4, 9, 10, 12\}$. Show that $\delta_{BCH}(C) = 5$ and that it is the true minimum distance.

7.3.8 Consider the binary code $C$ of length 21 and defining set $\{1\}$.

1) Show that there are exactly two binary irreducible polynomials of degree 6 that have as zeros elements of order 21.

2) Show that the BCH bound and the minimum distance are both equal to 3.

3) Conclude that the minimum distance of a cyclic code is not always equal to the minimal weight of the generator polynomials of all equivalent cyclic codes.
Theorem 7.4.4 The Hartmann-Tzeng bound. Let $I$ be the complete defining set of a cyclic code $C$. Then the minimum distance of $C$ is at least $\delta_{HT}(I)$.

Proof. This is a consequence of Definition 7.4.3 and Proposition 7.4.1.

Proposition 7.4.5 Let $I$ be a subset of $\mathbb{Z}_n$. Then $\delta_{HT}(I) \geq \delta_{BCH}(I)$.

Proof. If we take $U_1 = U$, $U_2 = \{0\}$, $\delta_1 = \delta$ and $\delta_2 = 2$ in the HT bound, then we get the BCH bound.

Remark 7.4.6 In computing $\delta_{HT}(I)$ one considers all $a \cdot I$ with $\gcd(a, n) = 1$ as in Remark 7.3.11. So we may assume that $U_1$ is a consecutive set of period one. Let $S(U_1) = \{i \in \mathbb{Z}_n | i + U_1 \subseteq I\}$ be the shift set of $U_1$. Then $U_1 + S(U_1) \subseteq I$. Furthermore if $U_1 + U_2 \subseteq I$, then $U_2 \subseteq S(U_1)$. Take a consecutive subset $U_2$ of $S(U_1)$. This gives all desired pairs $(U_1, U_2)$ of consecutive subsets in order to compute $\delta_{HT}(I)$.

Example 7.4.7 Consider Example 7.4.2. Then $U_1$ is a consecutive subset of period one of $\mathbb{Z}_3$ and $U_2 = S(U_1)$ is a consecutive subset of period five. And $U_1' = \{1, 2\}$ is a consecutive subset of period one of $\mathbb{Z}_7$ and $U_2' = S(U_1') = \{0, 7, 14\}$ is a consecutive subset of period seven. The choices of $(U_1, U_2)$ and $(U_1', U_2')$ are both optimal. Hence $\delta_{HT}(Z_1) = 5$.

Example 7.4.8 Let $C$ be the binary cyclic code of length 21 and defining set $\{1, 3, 7, 9\}$. Then $I = \{-, 1, 2, 3, 4, -, 6, 7, 8, 9, -, 11, 12, -, 14, 15, 16, -, 18, -, -\}$ is the complete defining set of $C$. From this we conclude that $\delta_{BCH}(I) \geq 5$ and $\delta_{HT}(I) \geq 6$. By considering $5 \cdot I$ one concludes that in fact equalities hold. But we show in Example 7.4.17 that the minimum distance of $C$ is strictly larger than 6.

7.4.2 Roos bound

The Roos bound is first formulated for arbitrary linear codes and afterwards applied to cyclic codes.

Definition 7.4.9 Let $a, b \in \mathbb{F}_q^n$. Define the star product $a * b$ by the coordinate wise multiplication:

$$a * b = (a_1 b_1, \ldots, a_n b_n).$$

Let $A$ and $B$ be subsets of $\mathbb{F}_q^n$. Define

$$A * B = \{ a * b \mid a \in A, b \in B \}.$$

Remark 7.4.10 If $A$ and $B$ are subsets of $\mathbb{F}_q^n$, then

$$(A * B)^\perp = \{ c \in \mathbb{F}_q^n \mid (a * b) \cdot c = 0 \text{ for all } a \in A, b \in B \}$$
is a linear subspace of \( F_q \). But if \( A \) and \( B \) are linear subspaces of \( F_q^n \), then \( A \ast B \) is not necessarily a linear subspace. See Example 9.1.3.

Consider the star product combined with the inner product. Then

\[
(a \ast b) \cdot c = \sum_{i=1}^{n} a_i b_i c_i.
\]

Hence \( a \cdot (b \ast c) = (a \ast b) \cdot c \).

**Proposition 7.4.11** Let \( C \) be an \( F_q \)-linear code of length \( n \). Let \((A, B)\) be a pair of \( F_{q^m} \)-linear codes of length \( n \) such that \( C \subseteq (A \ast B)^\perp \). Assume that \( A \) is not degenerate and \( k(A) + d(A) + d(B^\perp) \geq n + 3 \). Then \( d(C) \geq k(A) + d(B^\perp) - 1 \).

**Proof.** Let \( a = k(A) - 1 \) and \( b = d(B^\perp) - 1 \). Let \( c \) be a nonzero element of \( C \) with support \( I \). If \( |I| \leq b \), then take \( i \in I \). There exists an \( a \in A \) such that \( a_i \neq 0 \), since \( A \) is not degenerate. So \( a \ast c \) is not zero. Now \((c \ast a) \cdot b = c \cdot (a \ast b)\) by Remark 7.4.10 and this is equal to zero for all \( b \) in \( B \), since \( C \subseteq (A \ast B)^\perp \).

Hence \( a \ast c \) is a nonzero element of \( B^\perp \) of weight at most \( b \). This contradicts \( d(B^\perp) > b \). So \( b < |I| \).

If \( |I| \leq a + b \), then we can choose index sets \( I_- \) and \( I_+ \) such that \( I_- \subseteq I \subseteq I_+ \) and \( I_- \) has \( b \) elements and \( I_+ \) has \( a + b \) elements. Recall from Definition 4.4.10 that \( A(I_+ \setminus I_-) \) is defined as the space \( \{ a \in A | a_i = 0 \text{ for all } i \in I_+ \setminus I_- \} \). Now \( k(A) > a \) and \( I_+ \setminus I_- \) has \( a \) elements. Hence \( A(I_+ \setminus I_-) \) is not zero. Let \( a \) be a nonzero element of \( A(I_+ \setminus I_-) \). The vector \( c \ast a \) is an element of \( B^\perp \) and has support in \( I_- \). Furthermore \( |I_-| = b < d(B^\perp) \), hence \( a \ast c = 0 \), so \( a_i = 0 \) for all \( i \in I_+ \). Therefore \( a \) is a nonzero element of \( A \) of weight at most \( n - |I_+| = n - (a + b) \), which contradicts the assumption \( d(A) > n - (a + b) \). So \( |I| > a + b \). Therefore \( d(C) \geq a + b + 1 = k(A) + d(B^\perp) - 1 \).

In order to apply this proposition to cyclic codes some preparations are needed.

**Definition 7.4.12** Let \( U \) be a subset of \( \mathbb{Z}_n \). Let \( \alpha \) be an element of order \( n \) in \( \mathbb{F}_{q^m}^* \). Let \( C_U \) be the code over \( F_{q^m} \) of length \( n \) generated by the elements \((1, \alpha, \ldots, \alpha^{(n-1)})\) for \( i \in U \). Then \( U \) is called a generating set of \( C_U \). Let \( d_U \) be the minimum distance of the code \( C_U \).

**Remark 7.4.13** Notice that \( C_U \) and its dual are codes over \( F_{q^m} \). Every subset \( U \) of \( \mathbb{Z}_n \) is a complete defining set with respect to \( q^m \), since \( n \) divides \( q^m - 1 \), so \( q^m U = U \). Furthermore \( C_U \) has dimension \( |U| \). The code \( C_U \) is cyclic, since \( \sigma(1, \alpha, \ldots, \alpha^{(n-1)}) = \alpha^{-1}(1, \alpha, \ldots, \alpha^{(n-1)}) \).

\( U \) is the complete defining set of \( C_U \). So \( d_U \geq \delta_{BCH}(U) \). Beware that \( d_U \) is by definition the minimum distance of \( C_U \) over \( F_{q^m} \) and not of the cyclic code over \( F_q \) with defining set \( U \).

**Remark 7.4.14** Let \( U \) and \( V \) be subsets of \( \mathbb{Z}_n \). Let \( w \in U + V \). Then \( w = u + v \) with \( u \in U \) and \( v \in V \). So

\[
(1, \alpha^w, \ldots, \alpha^{w(n-1)}) = (1, \alpha^u, \ldots, \alpha^{u(n-1)}) \ast (1, \alpha^v, \ldots, \alpha^{v(n-1)})
\]

Hence

\[
C_U \ast C_V \subseteq C_{U+V}.
\]

Therefore \( C \subseteq (C_U \ast C_V)^\perp \) if \( C \) is a cyclic code with \( U + V \) in its defining set.
Remark 7.4.15 Let $U$ be a subset of $\mathbb{Z}_n$. Let $\bar{U}$ be a consecutive set containing $U$. Then $\bar{U}$ is the complete defining set of $C^{\perp}_U$. Hence $\mathbb{Z}_n \setminus \{-i|i \in \bar{U}\}$ is the complete defining set of $C_U$ by Proposition 7.2.58. Then $\mathbb{Z}_n \setminus \{-i|i \in \bar{U}\}$ is a consecutive set of size $n - |\bar{U}|$ that is contained in the defining set of $C_U$. Hence the minimum distance of $C_U$ is at least $n - |\bar{U}| + 1$ by the BCH bound.

Proposition 7.4.16 Let $U$ be a nonempty subset of $\mathbb{Z}_n$ that is contained in the consecutive set $\bar{U}$. Let $V$ be a subset of $\mathbb{Z}_n$ such that $|U| \leq |U| + dv - 2$. Let $C$ be a cyclic code of length $n$ such that $U + V$ is in the set of zeros of $C$. Then the minimum distance of $C$ is at least $|U| + dv - 1$.

Proof. Let $A$ and $B$ be the cyclic codes with generating sets $U$ and $V$, respectively. Then $A$ has dimension $|U|$ by Remark 7.4.13 and its minimum distance is at least $n - |U| + 1$ by Remark 7.4.15. A generating matrix of $A$ has no zero column, since otherwise $A$ would be zero, since $A$ is cyclic; but $A$ is not zero, since $U$ is not empty. So $A$ is not degenerate. Moreover $d(B^{\perp}) = dv$, by Definition 7.4.12. Hence $k(A) + d(A) + d(B^{\perp}) \geq |U| + (n - |U| + 1) + dv$ which is at least $n + 3$, since $|U| \leq |U| + dv - 2$. Finally $C \subseteq (A \ast B)^{\perp}$ by Remark 7.4.14. Therefore all assumptions of Proposition 7.4.25 are fulfilled. Hence $d(C) \geq k(A) + d(B^{\perp}) - 1 = |U| + dv - 1$.

Example 7.4.17 Let $C$ be the binary cyclic code of Example 7.4.8. Let $U = 4 \cdot \{0, 1, 3, 5\}$ and $V = \{2, 3, 4\}$. Then $\bar{U} = 4 \cdot \{0, 1, 2, 3, 4, 5\}$ is a consecutive set and $dv = 4$. By inspection of the table

| + 0 4 12 20 |
| 2 2 6 14 1 |
| 3 3 7 15 2 |
| 4 4 8 16 3 |

we see that $U + V$ is contained in the complete defining set of $C$. Furthermore $|\bar{U}| = 6 = |U| + dv - 2$. Hence $d(C) \geq 7$ by Proposition 7.4.16. The alternative choice with $U' = 4 \cdot \{0, 1, 2, 3, 5, 6\}$, $U'' = 4 \cdot \{0, 1, 2, 3, 4, 5, 6\}$ and $V' = \{3, 4\}$ gives $d(C) \geq 8$ by the BCH bound. This in fact is the true minimum distance.

Definition 7.4.18 Let $I$ be a subset of $\mathbb{Z}_n$. Denote by $\delta_R(I)$ the largest number $\delta$ such that there exist nonempty subsets $U$ and $V$ of $\mathbb{Z}_n$ and a consecutive set $\bar{U}$ with $U \subseteq \bar{U}$, $U + V \subseteq I$ and $|U| \leq |U| + dv - 2 = \delta - 1$. Let $C$ be a cyclic code of length $n$. Then $\delta_R(Z(C))$ is denoted by $\delta_R(C)$.

Theorem 7.4.19 The Roos bound. The minimum distance of a cyclic code $C$ is at least $\delta_R(C)$.

Proof. This is a consequence of Proposition 7.4.16 and Definition 7.4.18.

Proposition 7.4.20 Let $I$ be a subset of $\mathbb{Z}_n$. Then $\delta_R(I) \geq \delta_{HT}(I)$.

Proof. Let $U_1$ and $U_2$ be nonempty consecutive subsets of $\mathbb{Z}_n$ of sizes $\delta_1 - 1$ and $\delta_2 - 1$, respectively. Let $U = \bar{U} = U_1$ and $V = U_2$. Now $dv = \delta_2 \geq 2$, since $V$ is not empty. Hence $|U| \leq |U| + dv - 2$. Applying Proposition 7.4.16 gives $\delta_R(I) \geq |U| + dv - 1 \geq \delta_1 + \delta_2 - 2$. Hence $\delta_R(I) \geq \delta_{HT}(I)$.

Example 7.4.21 Examples 7.4.8 and 7.4.17 give a subset $I$ of $\mathbb{Z}_{21}$ such that $\delta_{BCH}(I) < \delta_{HT}(I) < \delta_R(I)$. 


7.4.3 AB bound

Remark 7.4.22 In 3.1.2 we defined for every subset \( I \) of \( \{1, \ldots, n\} \) the projection map \( \pi_I : \mathbb{F}_q^n \to \mathbb{F}_q^r \) by \( \pi_I(x) = (x_{i_1}, \ldots, x_{i_t}) \), where \( I = \{i_1, \ldots, i_t\} \) and \( 1 \leq i_1 < \ldots < i_t \leq n \). We denoted the image of \( \pi_I \) by \( A(I) \) and the kernel of \( \pi_I \) by \( \ker(I) \), that is \( A(I) = \{a \in A | a_i = 0 \ \text{for all} \ i \in I\} \). Suppose that \( \dim A = k \) and \( |I| = t \). If \( t < d(A^+) \), then \( \dim A(I) = k - t \) by Lemma 4.4.13, and therefore \( \dim A(I) = t \).

The following proposition is known for cyclic codes as the \( AB \) or the \textit{van Lint-Wilson} bound.

Proposition 7.4.23 Let \( A, B \) and \( C \) be linear codes of length \( n \) over \( \mathbb{F}_q \) such that \( (A * B) \perp C \) and \( d(A^+) > a > 0 \) and \( d(B^+) > b > 0 \). Then \( d(C) \geq a + b \).

Proof. Let \( c \) be a nonzero codeword in \( C \) with support \( I \), that is to say \( I = \{i | c_i \neq 0\} \). Let \( t = |I| \). Without loss of generality we may assume that \( a \leq b \). We have that

\[
\dim(A_I) + \dim(B_I) \geq \begin{cases} 
2t & \text{if } t \leq a \\
\ a + t & \text{if } a < t \leq b \\
\ a + b & \text{if } b < t 
\end{cases}
\]

by Remark 7.4.22. But \( (A * B) \perp C \), so \( (c * A)_I \perp B_I \). Moreover

\[
\dim((c * A)_I) = \dim(A_I),
\]

since \( c_i \neq 0 \) for all \( i \in I \). Therefore

\[
\dim(A_I) + \dim(B_I) \leq |I| = t.
\]

This is only possible in case \( t \geq a + b \). Hence \( d(C) \geq a + b \).

Example 7.4.24 Consider the binary cyclic code of length 21 and defining set \( \{0, 1, 3, 7\} \). Then the complete defining set of this code is given by

\[
I = \{0, 1, 2, 3, 4, -6, 7, 8, -8, -11, 12, -14, 1, -16, -1, -2, -3, -7\}.
\]

We leave it as an exercise to show that \( \delta_{\text{BCH}}(I) = \delta_{\text{HT}}(I) = \delta_{\text{R}}(I) = 6 \). Application of the AB bound to \( U = \{1, 2, 3, 6\} \) and \( V = \{0, 1, 5\} \) gives that the minimum distance is at least 7. The minimum distance is at least 8, since it is an even weight code.

Remark 7.4.25 Let \( C \) be an \( \mathbb{F}_q \)-linear code of length \( n \). Let \( (A, B) \) be a pair of \( \mathbb{F}_q^r \)-linear codes of length \( n \). Let \( a = k(A) - 1 \) and \( b = d(B^+) - 1 \), then one can restate the conditions of Proposition as follows: If \( (1) \ A * B \perp C \), \( (2) \ k(A) > a \), \( (3) \ d(B^+) > b \), \( (4) \ d(A) + a + b > n \) and \( (5) \ d(A^+) > 1 \), then \( d(C) \geq a + b + 1 \).

The original proof given by Van Lint and Wilson of the Roos bound is as follows. Let \( A \) be a generator matrix of \( A \). Let \( A_I \) be the submatrix of \( A \) consisting of the columns indexed by \( I \). Then \( \text{rank}(A_I) = \dim(A_I) \). Condition \( (5) \) implies that \( A \) has no zero column, so \( \text{rank}(A_I) \geq 1 \) for all \( I \) with at least one element. Let \( I \) be an index set such that \( |I| \leq a + b \), then any two words of \( A \) differ in
at least one place of \( I \), since \( d(A) > n - (a + b) \geq n - |I| \), by Condition (4). So \( A \) and \( A' \) have the same number of codewords, so \( \text{rank}(A_I) \geq k(A) \geq a + 1 \). Hence for any \( I \) such that \( b < |I| \leq a + b \) we have that \( \text{rank}(A_I) \geq |I| - b + 1 \). Let \( B \) be a generator matrix of \( B \). Then Condition (3) implies:

\[
\text{rank}(B_I) = \begin{cases} 
|I| & \text{if } |I| \leq b \\
\geq b & \text{if } |I| > b
\end{cases}
\]

by Remark 7.4.22. Therefore,

\[
\text{rank}(A_I) + \text{rank}(B_I) > |I| \text{ for } |I| \leq a + b
\]

Now let \( c \) be a nonzero element of \( C \) with support \( I \), then \( \text{rank}(A_I) + \text{rank}(B_I) \leq |I| \), as we have seen in the proof of Proposition 7.4.23. Hence \(|I| > a + b\), so \( d(C) > a + b \).

**Example 7.4.26** In this example we show that the assumption that \( A \) is non-degenerate is necessary. Let \( A, B \subseteq C \) be the binary codes with generating matrices \((011), (111)\) and \((100)\), respectively. Then \( A \ast C \subseteq B \subseteq C \) and \( k(A) = 1, d(A) = 2, n = 3 \) and \( d(B) = 3 \), so \( k(A) + d(A) + d(B) = 6 = n + 3 \), but \( d(C) = 1 \).

### 7.4.4 Shift bound

**Definition 7.4.27** Let \( I \) be a subset of \( \mathbb{Z}_n \). A subset \( A \) of \( \mathbb{Z}_n \) is called independent with respect to \( I \) if it can be obtained by the following rules:

1. (I.1) the empty set is independent with respect to \( I \).
2. (I.2) if \( A \) is independent with respect to \( I \) and \( A \) is a subset of \( I \) and \( b \in \mathbb{Z}_n \) is not an element of \( I \), then \( A \cup \{b\} \) is independent with respect to \( I \).
3. (I.3) if \( A \) is independent with respect to \( I \) and \( c \in \mathbb{Z}_n \), then \( c + A \) is independent with respect to \( I \), where \( c + A = \{c + a \mid a \in A \} \).

**Remark 7.4.28** The name "shifting" refers to condition (I.3). A set \( A \) is independent with respect to \( I \) if and only if there exists a sequence of sets \( A_1, \ldots, A_s \) and elements \( a_1, \ldots, a_{s-1} \) and \( b_0, b_1, \ldots, b_{s-1} \) in \( \mathbb{Z}_n \) such that \( A_1 = \{b_0\} \) and \( A = A_s \) and furthermore

\[
A_{i+1} = (a_i + A_i) \cup \{b_i\}
\]

and

\[
a_i + A_i \text{ is a subset of } I \text{ and } b_i \text{ is not an element of } I.
\]

Then

\[
A_i = \{b_{i-1} + \sum_{j=1}^{i-1} a_j \mid l = 1, \ldots, i \},
\]

and all \( A_i \) are independent with respect to \( I \).

Let \( i_1, i_2, \ldots, i_w \) and \( j_1, j_2, \ldots, j_w \) be new sequences which are obtained from the sequences \( a_1, \ldots, a_{w-1} \) and \( b_0, b_1, \ldots, b_{w-1} \) by:

\[
i_w = 0, \quad i_{w-1} = a_1, \ldots, \quad i_{w-k} = a_1 + \cdots + a_k \quad \text{and} \quad j_k = b_{k-1} - i_{w-k+1}.
\]
These data can be given in the following table

<table>
<thead>
<tr>
<th></th>
<th>$j_1$</th>
<th>$j_2$</th>
<th>$j_3$</th>
<th>...</th>
<th>$j_{w-1}$</th>
<th>$j_w$</th>
<th>+</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_{w-1}$</td>
<td>$A_w$</td>
<td>$i_1 + j_1$</td>
<td>$i_1 + j_2$</td>
<td>...</td>
<td>$i_1 + j_{w-1}$</td>
<td>$b_{w-1}$</td>
<td>$i_1$</td>
</tr>
<tr>
<td>$a_{w-2}$</td>
<td>$A_{w-1}$</td>
<td>$i_2 + j_1$</td>
<td>$i_2 + j_2$</td>
<td>...</td>
<td>$b_{w-2}$</td>
<td>$i_2$</td>
<td></td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td></td>
</tr>
<tr>
<td>$a_2$</td>
<td>$A_3$</td>
<td>$a_1 + a_2 + b_0$</td>
<td>$a_2 + b_1$</td>
<td>$b_2$</td>
<td>$i_{w-2}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$a_1$</td>
<td>$A_2$</td>
<td>$a_1 + b_0$</td>
<td>$b_1$</td>
<td>$i_{w-1}$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$A_1$</td>
<td>$b_0$</td>
<td>$i_w$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

with the elements of $A_i$ as rows in the middle part. The enumeration of the $A_i$ is from the bottom to the top, and the $b_i$ are on the diagonal. In the first row and the last column the $j_i$ and the $i_k$ are tabulated, respectively. The sum $i_k + j_i$ is given in the middle part.

By this transformation it is easy to see that a set $A$ is independent with respect to $I$ if and only if there exist sequences $i_1, i_2, \ldots, i_w$ and $j_1, j_2, \ldots, j_w$ such that $A = \{i_1 + j_l \mid 1 \leq l \leq w\}$ and

$$i_k + j_l \in I \text{ for all } l + k \leq w \quad \text{and} \quad i_k + j_l \notin I \text{ for all } l + k = w + 1.$$  

So the entries in the table above the diagonal are elements of $I$, and on the diagonal are not in $I$.

Notice that in this formulation we did not assume that the sets

$$\{i_k \mid 1 \leq k \leq w\}, \quad \{j_l \mid 1 \leq l \leq w\}$$

and $A$ have size $w$, since this is a consequence of this definition. If for instance $i_k = i_{k'}$ for some $1 \leq k < k' \leq w$, then $i_k + j_{w+1-k'} = i_{k'} + j_{w+1-k'} \notin I$, but $i_k + j_{w+1-k'} \in I$, which is a contradiction.

**Definition 7.4.29** For a subset $Z$ of $\mathbb{Z}_n$, let $\mu(Z)$ be the maximal size of a set which is independent with respect to $Z$. Define the *shift bound* bound for a subset $I$ of $\mathbb{Z}_n$ as follows:

$$\delta_S(I) = \min\{ \mu(Z) \mid I \subseteq Z \subseteq \mathbb{Z}_n, Z \neq \mathbb{Z}_n \text{ and } Z \text{ a complete defining set } \}.$$

**Theorem 7.4.30** The minimum distance of $C(I)$ is at least $\delta_S(I)$.

The proof of this theorem will be given at the end of this section.

**Proposition 7.4.31** The following inequality holds:

$$\delta_S(I) \geq \delta_{HT}(I).$$

**Proof.** There exist $\delta$, $s$ and $a$ such that $\gcd(a, n) = 1$ and $\delta_{HT}(I) = \delta + s$ and

$$\{i + j + ka \mid 1 \leq j < \delta, 0 \leq k \leq s\} \subseteq I.$$  

Suppose $Z$ is a complete defining set which contains $I$ and is not equal to $\mathbb{Z}_n$. Then there exists a $\delta' \geq \delta$ such that $i + j \in Z$ for all $1 \leq j < \delta'$ and $i + \delta' \notin Z$.  

The set \( \{i + j + ka \mid 1 \leq j < \delta, k \in \mathbb{Z}_n \} \) is equal to \( \mathbb{Z}_n \), since gcd\((a, n) < \delta\). So there exist \( s' \geq s \) and \( j' \) such that \( i + j + ka \in \mathbb{Z}_l \) for all \( 1 \leq j < \delta \) and \( 0 \leq k \leq s' \), and \( 1 \leq j' \leq \delta \) and \( i + j' + (s' + 1)a \not\in \mathbb{Z}_l \). Let \( w = \delta + s' \). Let \( i_k = (k-1)a \) for all \( 1 \leq k \leq s' + 1 \), and \( i_k = \delta' - \delta - s' - 1 + k \) for all \( k \) such that \( s' + 2 \leq k \leq \delta + s' \). Let \( j_l = i + l \) for all \( 1 \leq l \leq \delta - 1 \), and let \( j_l = i + j' + (l - \delta + 1)a \) for all \( l \) such that \( \delta \leq l \leq \delta + s' \). Then one easily checks that \( i_k + j_l \in \mathbb{Z}_l \) for all \( k + l \leq w \), and \( i_k + j_w, k+1 = i + j' + (s' + 1)a \not\in \mathbb{Z}_l \) for all \( 1 \leq k \leq s' + 1 \), and \( i_k + j_w, k+1 = i + \delta' \not\in \mathbb{Z}_l \) for all \( s' + 2 \leq k \leq \delta + s' \). So we have a set which is independent with respect to \( \mathbb{Z}_l \) and has size \( w = \delta + s' \geq \delta + s \). Hence \( \mu(Z) \geq \delta + s \) for all complete defining sets \( Z \) which contain \( I \) and are not equal to \( \mathbb{Z}_n \). Therefore \( \delta_S(I) \geq \delta_{HT}(I) \). 

\[
\text{Example 7.4.32} \quad \text{The binary Golay code of length 23 can be defined as the cyclic code with defining set \{1\}, see Proposition 7.3.19. In this example we show that the shift bound is strictly greater than the HT bound and is still not equal to the minimum distance. Let } I_i \text{ be the cyclotomic coset of } i. \text{ Then} \\
Z_1 = \{-1, 1, 2, 3, 4, -6, -8, 9, -12, 13, -16, -18, -19, -20, -21, -22\},
\]
and
\[
Z_5 = \{-1, -3, -5, -7, -9, -11, -14, -15, -17, -19, -20, -21, -22\}
\]
Then \( \delta_{BCH}(Z_1) = \delta_{HT}(Z_1) = 5 \).

Let \( (a_1, \ldots, a_5) = (1, -1, -3, 7, 4, 13) \) and \( (b_0, \ldots, b_5) = (5, 5, 14, 5, 5) \). Then the \( A_{s+1} = (A_s + a_i) \cup \{b_i\} \) are given in the rows of the middle part of the following table

<table>
<thead>
<tr>
<th>( a_i )</th>
<th>( A_{s+1} )</th>
<th>( i_{s+1} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>( A_6 )</td>
<td>2 3 6 8 18 5 -3</td>
</tr>
<tr>
<td>4</td>
<td>( A_5 )</td>
<td>12 13 16 18 5 7</td>
</tr>
<tr>
<td>7</td>
<td>( A_4 )</td>
<td>8 9 12 14 3</td>
</tr>
<tr>
<td>-3</td>
<td>( A_3 )</td>
<td>1 2 5 4</td>
</tr>
<tr>
<td>-1</td>
<td>( A_2 )</td>
<td>4 5 -1</td>
</tr>
<tr>
<td>1</td>
<td>( A_1 )</td>
<td>5 0</td>
</tr>
</tbody>
</table>

with the \( a_i \) in the first column and the \( b_i \) in the diagonal. The corresponding sequence \((i_1, \ldots, i_6) = (-3,3,7,-4,-1,0)\) is given in the last column of the table and \((j_0, \ldots, j_6) = (5,6,9,11,-2,8)\) in second row. So \( Z_1 \) has an independent set of size 6. In fact this is the maximal size of an independent set of \( Z_1 \). 

Hence \( \mu(Z_1) = 6 \). The defining sets \( Z_0, Z_1 \) and \( Z_5 \) and their unions are complete, and these are the only ones. Let \( Z_{0,1} = Z_0 \cup Z_1 \), then \( Z_{0,1} \) has an independent set of size 7, since \( A_6 \) is independent with respect to \( Z_1 \) and also with respect to \( Z_{0,1} \), and \(-2 + A_6 = \{0, 1, 4, 6, 16, 3\}\) is a subset of \( Z_{0,1} \) and \( 5 \not\in Z_{0,1} \), so \( A_7 = \{1, 3, 4, 6, 16, 3, 5\}\) is independent with respect to \( Z_{0,1} \). Furthermore \( Z_{1,5} = Z_1 \cup Z_5 \) contains a sequence of 22 consecutive elements, so \( \mu(Z_{1,5}) \geq 23 \). Therefore \( \delta_S(Z_1) = 6 \). But the minimum distance of the binary Golay code is 7, since otherwise there would be a word \( c \in C(Z_1) \) of weight 6, so \( c \in C(Z_{0,1}) \), but \( \delta_S(Z_{0,1}) \geq 7 \), which is a contradiction.
7.4. IMPROVEMENTS OF THE BCH BOUND

Example 7.4.33 Let \( n = 26, F = F_{27}, \) and \( F_0 = F_3. \) Let \( 0, 13, 14, 16, 17, 22, 23 \) and \( 25 \) be the elements of \( I. \) Let \( U = \{0, 3, 9, 12\} \) and \( V = \{13, 14\}. \) Then \( d_V = 3 \) and \( \overline{U} = \{0, 3, 6, 9, 12\}, \) so \( |\overline{U}| = 5 \leq 4 + 3 - 2. \) Moreover \( I \) contains \( U + V. \) Hence \( \delta_R(I) \geq 4 + 3 - 1 = 6, \) but in fact \( \delta_S(I) = 5. \)

Example 7.4.34 ***Example of \( \delta_R(I) < \delta_S(I). ***

Example 7.4.35 It is necessary to take the minimum of all \( \mu(Z) \) in the definition of the shift bound. The maximal size of an independent set with respect to a complete defining set \( I \) is not a lower bound for the minimum distance of the cyclic code with \( I \) as defining set, as the following example shows. Let \( F \) be a finite field of odd characteristic. Let \( \alpha \) be a non-zero element of \( F \) of even order \( n. \) Let \( I = \{2, 4, \ldots, n - 2\} \) and \( I = \{0, 2, 4, \ldots, n - 2\}. \) Then \( I \) and \( I \) are complete and \( \mu(I) = 3, \) since \( \{2, 0, 1\} \) is independent with respect to \( I, \) but \( \mu(I) = 2. \)

***Picture of interrelations of the several bounds.

One way to get a bound on the weight of a codeword \( c = (c_0, \ldots, c_{n-1}) \) is obtained by looking for a maximal non-singular square submatrix of the matrix of syndromes \( (S_{ij}). \) For cyclic codes we get in this way a matrix, with entries \( S_{ij} = \sum c_k \alpha^{k(i+j)} \), which is constant along back-diagonals.

Suppose \( \gcd(n, q) = 1. \) Then there is a field extension \( F_q^m \) of \( F_q \) such that \( F_q^m \) has an element \( \alpha \) of order \( n. \) Let \( a_i = (1, \alpha^i, \ldots, \alpha^{i(n-1)}) \). Then \( \{a_i \mid i \in \mathbb{Z}_n\} \) is a basis of \( F_q^m. \)

Consider the following generalization of the definition of a syndrome 6.2.2.

Definition 7.4.36 The syndrome of a word \( y \in F_0^n \) with respect to \( a_i \) and \( b_j \) is defined by

\[
S_{i,j}(y) = y \cdot a_i \ast b_j.
\]

Let \( S(y) \) be the syndrome matrix with entries \( S_{i,j}(y). \)

Notice that \( a_i \ast a_i = a_i \ast a_j \) for all \( i, j \in \mathbb{Z}_n. \) Hence \( S_{i,j} = S_{i+j}. \)

Lemma 7.4.37 Let \( y \in F_0^n. \) Let \( I = \{i + j \mid i, j \in \mathbb{Z}_n \} \) and \( y \cdot a_i \ast b_j = 0 \}. \) If \( A \) is independent with respect to \( I, \) then \( \text{wt}(y) \geq |A|. \)

Proof. Suppose \( A \) is independent with respect to \( I \) and \( w = \text{rank}(M) \) elements, then there exist sequences \( i_1, \ldots, i_w \) and \( j_1, \ldots, j_w \) such that \( A \) consists of the pairs \( (i_1, j_1), (i_1, j_2), \ldots, (i_k, j_k) \) for all \( k + l \leq w \) and \( (i_k, j_l) \notin I \) for all \( k + l = w + 1. \) Consider the \( (w \times w) \) matrix \( M \) with entries \( M_{kl} = S_{i_k+j_l}(y). \)

By the assumptions we have that \( M \) is a matrix such that \( M_{k,l} = 0 \) for all \( k + l \leq w \) and \( M_{k,l} \neq 0 \) for all \( k + l = w + 1, \) that is to say with zeros above the back-diagonal and non-zeros on the back-diagonal, so \( M \) has rank \( w. \) Moreover \( M \) is a submatrix of the matrix \( S(y) \) which can be written as a product:

\[
S(y) = HD(y)HT,
\]

where \( H \) is the matrix with the \( a_i \) as row vectors, \( D(y) \) is the diagonal matrix with the entries of \( y \) on the diagonal. Now the rank of \( H \) is \( n, \) since the \( a_0, \ldots, a_{n-1} \) is a basis of \( F_q^n. \) Hence

\[
|A| = w = \text{rank}(M) \leq \text{rank}(S(y)) \leq \text{rank}(D(y)) = \text{wt}(y).
\]
Remark 7.4.38 Let $C_i$ be a code with $Z_i$ as defining set for $i = 1, 2$. If $Z_1 \subseteq Z_2$, then $C_2 \subseteq C_1$.

Lemma 7.4.39 Let $I$ be a complete defining set for the cyclic code $C$. If $y \in C$ and $y \notin D$ for all cyclic codes $D$ with complete defining sets $Z$ which contain $I$ and are not equal to $I$, then $wt(y) \geq \mu(I)$.

Proof. Define

$$Z = \{i + j | i, j \in \mathbb{Z}_n, y \cdot a_i \ast b_j = 0\}.$$ 

***Then $Z$ is a complete defining set. ***

Clearly $I \subseteq Z$, since $y \in C$ and $I$ is a defining set of $C$. Let $D$ be the code with defining set $Z$. Then $y \in D$. If $I \neq Z$, then $y \notin D$ by the assumption, which is a contradiction. Hence $I = Z$, and $wt(y) \geq \mu(I)$, by Lemma 7.4.37.

Proof. Let $y$ be a non-zero codeword of $C$. Let $Z$ be equal to $\{i + j | i, j \in \mathbb{Z}_n, y \cdot a_i \ast b_j = 0\}$. Then $Z \neq \mathbb{Z}_n$, since $y$ is not zero and the $a_i$’s generate $\mathbb{F}_q^m$. The theorem now follows from Lemma 7.4.39 and the definition of the shift bound.

Remark 7.4.40 The computation of the shift bound is quite involved, and is only feasible the use of a computer. It makes sense if one classifies codes with respect to the minimum distance, since in order to get $\delta_S(I)$ one gets at the same time the $\delta_S(J)$ for all $I \subseteq J$.

7.4.5 Exercises

7.4.1 Consider the binary cyclic code of length 15 and defining set $\{3, 5\}$. Compute the complete defining set $I$ of this code. Show that $\delta_{BCH}(I) = 3$ and $\delta_{HT}(I) = 4$ is the true minimum distance.

7.4.2 Consider the binary cyclic code of length 35 and defining set $\{1, 5, 7\}$. Compute the complete defining set $I$ of this code. Show that $\delta_{BCH}(I) = \delta_{HT}(I) = 6$ and $\delta_R(I) \geq 7$.

7.4.3 Let $m$ be odd and $n = 2^m - 1$. Melas’s code is the binary cyclic code of length $n$ and defining set $\{1, -1\}$. Show that this code is reversible, has dimension $k = n - 2m$ and that the minimum distance is at least five.

7.4.4 Let $-1$ be a power of $q$ modulo $n$. Then every cyclic code over $\mathbb{F}_q$ of length $n$ is reversible.

7.4.5 Let $n = 2^{2m} + 1$ with $m > 1$. Zetterberg’s code is the binary cyclic code of length $n$ and defining set $\{1\}$. Show that this code is reversible, has dimension $k = n - 4m$ and that the minimum distance is at least five.

7.4.6 Consider the ternary cyclic code of length 11 and defining set $\{1\}$. Compute the complete defining set $I$ of this code. Show that $\delta_{BCH}(I) = \delta_{HT}(I) = \delta_S(I) = 4$. Let $I' = \{0\} \cup I$. Show that $\delta_{BCH}(I') = \delta_{HT}(I) = 4$ and $\delta_S(I) \geq 5$.

7.4.7 Let $q$ be a power of a prime and $n$ a positive integer such that $\gcd(n, q) = 1$. Write a computer program that computes the complete defining set $Z$ modulo $n$ with respect to $q$ and the bounds $\delta_{BCH}(Z)$, $\delta_{HT}(Z)$, $\delta_R(Z)$ and $\delta_S(Z)$ for a given defining set $I$ in $\mathbb{Z}_n$. 

228 CHAPTER 7. CYCLIC CODES
7.5 Locator polynomials and decoding cyclic codes

7.5.1 Mattson-Solomon polynomial

Definition 7.5.1 Let \( \alpha \in \mathbb{F}_{q^m}^* \) be a primitive \( n \)-th root of unity. The Mattson-Solomon (MS) polynomial \( A(Z) \) of

\[
a(x) = a_0 + a_1 x + \cdots + a_{n-1} x^{n-1}
\]

is defined by

\[
A(Z) = \sum_{i=1}^{n} A_i Z^{n-i}, \quad \text{where} \quad A_i = a(\alpha^i) \in \mathbb{F}_{q^m}.
\]

Here too we adopt the convention that the index \( i \) is computed modulo \( n \).

The MS polynomial \( A(Z) \) is the discrete Fourier transform of \( a(x) \). In order to compute the inverse discrete Fourier transform, that is the coefficients of \( a(x) \) in terms of the \( A(Z) \) we need the following lemma on the sum of a geometric sequence.

Lemma 7.5.2 Let \( \beta \in \mathbb{F}_{q^m} \) be a zero of \( X^n - 1 \). Then

\[
\sum_{i=1}^{n} \beta^i = \begin{cases} 
  n & \text{if } \beta = 1 \\
  0 & \text{if } \beta \neq 1.
\end{cases}
\]

Proof. If \( \beta = 1 \), then \( \sum_{i=1}^{n} \beta^i = n \). If \( \beta \neq 1 \), then using the formula for the sum of a geometric series \( \sum_{i=1}^{n} \beta^i = (\beta^{n+1} - \beta) / (\beta - 1) \) and \( \beta^{n+1} = \beta \) gives the desired result.

Proposition 7.5.3

1) The inverse transform is given by \( a_i = \frac{1}{n} A(\alpha^i) \).
2) \( A(Z) \) is the MS polynomial of a word \( a(x) \) coming from \( \mathbb{F}_q^n \) if and only if \( A_{jq} = A_j^q \) for all \( j = 1, \ldots, n \).
3) \( A(Z) \) is the MS polynomial of a codeword \( a(x) \) of the cyclic code \( C \) if and only if \( A_j = 0 \) for all \( j \in Z(C) \) and \( A_{jq} = A_j^q \) for all \( j = 1, \ldots, n \).

Proof.

1) Expanding \( A(\alpha^i) \) and using the definitions gives

\[
A(\alpha^i) = \sum_{j=1}^{n} A_j \alpha^{i(n-j)} = \sum_{j=1}^{n} a(\alpha^j) \alpha^{i(n-j)} = \sum_{j=1, k=0}^{n} a_k \alpha^k \alpha^{i(n-j)}.
\]

Using \( \alpha^n = 1 \), interchanging the order of summation and using Lemma 7.5.2 with \( \beta = \alpha^{k-i} \) gives

\[
\sum_{k=0}^{n-1} a_k \sum_{j=1}^{n} \alpha^{(k-i)j} = na_i.
\]
2) If $A(Z)$ is the MS polynomial of $a(x)$, then using Proposition 7.2.40 gives

$$A_j^q = a(a^j)^q = a(a^{qj}) = A_{qj},$$

since the coefficients of $a(x)$ are in $\mathbb{F}_q$.

Conversely, suppose that $A_{jq} = A_j^q$ for all $j = 1, \ldots, n$. Then using (1) gives

$$a_i^q = (\frac{1}{n} A(\alpha^i))^q = \frac{1}{n} \sum_{j=1}^n A_j^q \alpha^{q(j-i)} = \frac{1}{n} \sum_{j=1}^n A_j \alpha^{q(n-j)}.$$

Using the fact that multiplication with $q$ is a permutation of $\mathbb{Z}_n$ gives that the above sum is equal to

$$\frac{1}{n} \sum_{j=1}^n A_j \alpha^{q(n-j)} = a_i.$$

Hence $a_i^q = a_i$ and $a_i \in \mathbb{F}_q$ for all $i$. Therefore $a(x)$ is coming from $\mathbb{F}_q^n$.

3) $A_j = 0$ if and only if $a(a^j) = 0$ by (1). Together with (2) and the definition of $Z(C)$ this gives the desired result.

Another proof of the BCH bound can be obtained with the Mattson-Solomon polynomial.

**Proposition 7.5.4** Let $C$ be a narrow sense BCH code with defining minimum distance $\delta$. If $A(Z)$ is the MS polynomial of $a(x)$ a nonzero codeword of $C$, then the degree of $A(Z)$ is at most $n - \delta$ and the weight of $a(x)$ is at least $\delta$.

**Proof.** Let $a(x)$ be a nonzero codeword of $C$. Let $A(Z)$ be the MS polynomial of $a(x)$, then $A_i = a(\alpha^i) = 0$ for all $i = 1, \ldots, \delta - 1$. So the degree of $A(Z)$ is at most $n - \delta$. We have that $a_i = A(\alpha^i)/n$ by (1) of Proposition 7.5.3. The number of zero coefficients of $a(x)$ is the number zeros of $A(Z)$ in $\mathbb{F}_{q^n}$, which is at most $n - \delta$. Hence the weight of $a(x)$ is at least $\delta$.

**Example 7.5.5** Let $a(x) = 6 + x + 3x^2 + x^3$ a codeword of the cyclic code of length 6 over $\mathbb{F}_7$ of Example 7.1.24. Choose $\alpha = 3$ as primitive element. Then $A(Z) = 4 + Z + 3Z^2$ is the MS polynomial of $a(x)$.

### 7.5.2 Newton identities

**Definition 7.5.6** Let $a(x)$ be a word of weight $w$. Then there are indices $0 \leq i_1 < \cdots < i_w < n$ such that

$$a(x) = a_{i_1} x^{i_1} + \cdots + a_{i_w} x^{i_w}$$

with $a_{i_j} \neq 0$ for all $j$. Let $x_j = \alpha^{i_j}$ and $y_j = a_{i_j}$. Then the $x_j$ are called the locators and the $y_j$ the corresponding values. Furthermore

$$A_i = a(\alpha^i) = \sum_{j=1}^w y_j x_j^i.$$

Consider the product

$$\sigma(Z) = \prod_{j=1}^w (1 - x_j Z).$$

Then $\sigma(Z)$ has as zeros the reciprocals of the locators, and is sometimes called the locator polynomial. Sometimes this name is reserved for the monic polynomial that has the locators as zeros.
Proposition 7.5.7 Let \( \sigma(Z) = \sum_{i=0}^{w} \sigma_i Z^i \) be the locator polynomial of the locators \( x_1, \ldots, x_w \). Then \( \sigma_i \) is the \( i \)-th elementary symmetric function in these locators:

\[
\sigma_i = (-1)^i \sum_{1 \leq j_1 < j_2 < \cdots < j_i \leq w} x_{j_1} x_{j_2} \cdots x_{j_i}.
\]

Proof. This is proved by induction on \( w \) and is left to the reader as an exercise.

The following property of the MS polynomial is called the generalized Newton identity and gives the reason for these definitions.

Proposition 7.5.8 For all \( i \) it holds that

\[
A_{i+w} + \sigma_1 A_{i+w-1} + \cdots + \sigma_w A_i = 0.
\]

Proof. Substitute \( Z = 1/x_j \) in the equation

\[
1 + \sigma_1 Z + \cdots + \sigma_w Z^w = \prod_{j=1}^{w} (1 - x_j Z)
\]

and multiply by \( y_j x_j^{i+w} \). This gives

\[
y_j x_j^{i+w} + \sigma_1 y_j x_j^{i+w-1} + \cdots + \sigma_w y_j x_j^i = 0.
\]

Summing on \( j = 1, \ldots, w \) yields the desired result of Proposition 7.5.8.

Example 7.5.9 Let \( C \) be the cyclic code of length 5 over \( \mathbb{F}_{16} \) with defining set \( \{1, 2\} \). Then this defining set is complete. The polynomial

\[
X^4 + X^3 + X^2 + X + 1
\]

is irreducible over \( \mathbb{F}_2 \). Let \( \beta \) be a zero of this polynomial in \( \mathbb{F}_{16} \). Then the order of \( \beta \) is 5. The generator polynomial of \( C \) is

\[
(X + \beta)(X + \beta^2) = X^2 + (\beta + \beta^2)X + \beta^3.
\]

So \( (\beta^3, \beta + \beta^2, 1, 0, 0) \in C \) and

\[
(\beta + \beta^2 + \beta^3, 1 + \beta, 0, 1, 0) = (\beta + \beta^2)(\beta^3, \beta + \beta^2, 1, 0, 0) + (0, \beta^3, \beta + \beta^2, 1, 0)
\]

is an element of \( C \). These codewords together with their cyclic shifts and their nonzero scalar multiples give \((5 + 5) \times 15 = 150\) words of weight 3. In fact these are the only codewords of weight 3, since it is an \([5, 3, 3]\) MDS code and \( A_3 = \binom{5}{3}(16 - 1) \) by Remark 3.2.15. Propositions 7.5.3 and 7.5.8 give another way to proof this. Consider the set of equations:

\[
\begin{align*}
A_1 &+ \sigma_1 A_3 + \sigma_2 A_2 + \sigma_3 A_1 = 0 \\
A_5 &+ \sigma_1 A_4 + \sigma_2 A_3 + \sigma_3 A_2 = 0 \\
A_1 &+ \sigma_1 A_5 + \sigma_2 A_4 + \sigma_3 A_3 = 0 \\
A_2 &+ \sigma_1 A_1 + \sigma_2 A_5 + \sigma_3 A_4 = 0 \\
A_3 &+ \sigma_1 A_2 + \sigma_2 A_1 + \sigma_3 A_5 = 0 
\end{align*}
\]
If \( A_1, A_2, A_3, A_4 \) and \( A_5 \) are the coefficients of the MS polynomial of a codeword, then \( A_1 = A_2 = 0 \). If \( A_3 = 0 \), then \( A_i = 0 \) for all \( i \). So we may assume that \( A_3 \neq 0 \). The above equations imply \( A_4 = \alpha_1 A_3, A_5 = (\sigma_1^2 + \sigma_2)A_3 \) and

\[
\begin{align*}
\sigma_1^3 + \sigma_3 &= 0 \\
\sigma_1^2 \sigma_2 + \sigma_2^2 + \sigma_1 \sigma_3 &= 0 \\
\sigma_1^2 \sigma_3 + \sigma_2 \sigma_3 + 1 &= 0.
\end{align*}
\]

Substitution of \( \sigma_3 = \sigma_1^3 \) in the remaining equations yields

\[
\begin{align*}
\sigma_1^4 + \sigma_1^2 \sigma_2 + \sigma_2^2 &= 0 \\
\sigma_1^3 + \sigma_1^2 \sigma_2 + 1 &= 0.
\end{align*}
\]

Multiplying the first equation with \( \sigma_1 \) and adding to the second one gives

\[1 + \sigma_1 \sigma_2^2 = 0.\]

Thus \( \sigma_1 = \sigma_2^{-2} \) and

\[\sigma_2^{10} + \sigma_2^5 + 1 = 0.\]

This last equation has 10 solutions in \( \mathbb{F}_{16} \), and we are free to choose \( A_3 \) from \( \mathbb{F}_{16}^* \). This gives in total 150 solutions.

### 7.5.3 APGZ algorithm

Let \( C \) be a cyclic code of length \( n \) such that the minimum distance of \( C \) is at least \( \delta \) by the BCH bound. In this section we will give a decoding algorithm for such a code which has an efficient implementation and is used in practice. This algorithm corrects errors of weight at most \((\delta - 1)/2\), whereas the true minimum distance can be larger than \( \delta \).

The notion of a syndrome was already given in the context of arbitrary codes in Definition 6.2.2. Let \( \alpha \) be a primitive \( n \)-th root of unity. Let \( C \) be a cyclic code of length \( n \) with \( 1, \ldots, \delta - 1 \) in its complete defining set. Let \( h_i = (1, \alpha^i, \ldots, \alpha^{i(n-1)}) \). Consider \( C \) as the subfield subcode of the code with parity check matrix \( \tilde{H} \) with rows \( h_i \) for \( i \in Z(C) \) as in Remark 7.3.2. Let \( c = (c_0, \ldots, c_{n-1}) \in C \) be the transmitted word, so \( c(x) = c_0 + \cdots + c_{n-1}x^{n-1} \). Let \( r \) be the received word with \( w \) errors and \( w \leq (\delta - 1)/2 \). So \( r(x) = c(x) + e(x) \) and \( wt(e(x)) = w \). The syndrome \( S_i \) of \( r(x) \) with respect to the row \( h_i \) is equal to

\[ S_i = r(\alpha^i) = e(\alpha^i) \text{ for } i \in Z(C), \]

since \( c(\alpha^i) = 0 \) for all \( i \in Z(C) \). The syndrome of \( r \) is \( s = r\tilde{H}^T \). Hence \( s_i = S_i \) for all \( i \in Z(C) \) and these are also called the known syndromes, since the receiver knows \( S_i \) for all \( i \in Z(C) \). The unknown syndromes are defined by \( S_i = e(\alpha^i) \) for \( i \not\in Z(C) \).

Let \( A(Z) \) be the MS polynomial of \( e(x) \). Then

\[ S_i = r(\alpha^i) = e(\alpha^i) = A_i \text{ for } i \in Z(C). \]

The receiver knows all \( S_1, S_2, \ldots, S_{2w} \), since \( \{1, 2, \ldots, \delta - 1\} \subseteq Z(C) \) and \( 2w \leq \delta - 1 \).
Let $\sigma(Z)$ be the error-locator polynomial, that is the locator polynomial

$$\sigma(Z) = \prod_{j=1}^{w}(1 - x_j Z)$$

of the error positions

$$\{x_1, \ldots, x_w\} = \{ \alpha^i \mid e_i \neq 0 \}.$$

Let $\sigma_i$ be the $i$-th coefficient of $\sigma(Z)$ and form the following set of generalized Newton identities of Proposition 7.5.8 with

$$S_i = A_i \begin{cases} S_{w+1} + \sigma_1 S_w + \cdots + \sigma_w S_1 = 0 \\ S_{w+2} + \sigma_1 S_{w+1} + \cdots + \sigma_w S_2 = 0 \\ \vdots & \vdots \\ S_{2w} + \sigma_1 S_{2w-1} + \cdots + \sigma_w S_w = 0. \end{cases} \quad (7.1)$$

The algorithm of Arimoto-Peterson-Gorenstein-Zierler (APGZ) solves this system of linear equations in the variables $\sigma_j$ by Gaussian elimination. The fact that this system has a unique solution is guaranteed by the following.

**Proposition 7.5.10** The matrix $(S_{i+j-1}|1 \leq i, j \leq v)$ is nonsingular if and only if $v = w$ the number of errors.

**Proof.** \( \text{***}(S_{i+j-1}) = HD(e)H^T \text{ as in the proof of Lemma 7.4.37***} \)

After the system of linear equations is solved, we know the error-locator polynomial

$$\sigma(Z) = 1 + \sigma_1 Z + \sigma_2 Z^2 + \cdots + \sigma_w Z^w$$

which has as its zeros the reciprocals of the error locations. Finding the zeros of this polynomial is done by inspecting all values of $F_{q^m}$.

**Example 7.5.11** Let $C$ be the binary narrow sense BCH code of length 15 and designed minimum distance 5 with generator polynomial $1 + X^4 + X^6 + X^7 + X^8$ as in Example 7.3.13. Let

$$r = (0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0)$$

be a received word with respect to the code $C$ with 2 errors. Then $r(x) = x + x^3 + x^4 + x^7 + x^{13}$ and $S_1 = r(\alpha) = \alpha^{12}$ and $S_3 = r(\alpha^3) = \alpha^7$. Now $S_2 = S_1^2 = \alpha^9$ and $S_4 = S_1^4 = \alpha^3$. The system of equations becomes:

$$\begin{cases} \alpha^7 + \alpha^9 \sigma_1 + \alpha^{12} \sigma_2 = 0 \\ \alpha^3 + \alpha^7 \sigma_1 + \alpha^9 \sigma_2 = 0 \end{cases}$$

Which has the unique solution $\sigma_1 = \alpha^{12}$ and $\sigma_2 = \alpha^3$. So the error-locator polynomial is

$$1 + \alpha^{12} Z + \alpha^3 Z^2$$

which has $\alpha^{-3}$ and $\alpha^{-10}$ as zeros. Hence

$$e = (0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0)$$

is the error and

$$c = (0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0)$$

is the codeword sent.
7.5.4 Closed formulas

Consider the system of equations (7.1) as linear in the unknowns $\sigma_1, \ldots, \sigma_w$ with coefficients in $S_1, \ldots, S_{2w}$. Then

$$\sigma_i = \frac{\Delta_i}{\Delta_0}$$

where $\Delta_i$ is the determinant of a certain $w \times w$ matrix according to Cramer’s rule. Then the $\Delta_i$ are polynomials in the $S_i$. Conclude that

$$\det \begin{pmatrix} 1 & Z & \ldots & Z^w \\ S_{w+1} & S_w & \ldots & S_1 \\ \vdots & \vdots & \ddots & \vdots \\ S_{2w} & S_{2w-1} & \ldots & S_w \end{pmatrix} = \Delta_0 + \Delta_1 Z + \cdots + \Delta_w Z^w$$

is a closed formula of the generic error-locator polynomial. Notice that the constant coefficient of the generic error-locator polynomial is not 1.

Example 7.5.12 Consider the narrow-sense BCH code with defining minimum distance 5. Then $\{1, 2, 3, 4\}$ is the defining set, so the syndromes $S_1, S_2, S_3$ and $S_4$ of a received word are known. We have to solve the system of equations

$$\begin{cases} S_2\sigma_1 + S_1\sigma_2 = -S_3 \\ S_3\sigma_1 + S_2\sigma_2 = -S_4 \end{cases}$$

Now Cramer’s rule gives that

$$\sigma_1 = \frac{\begin{vmatrix} -S_3 & S_1 \\ -S_4 & S_2 \end{vmatrix}}{\begin{vmatrix} S_2 & S_1 \\ S_3 & S_2 \end{vmatrix}} = \frac{S_1S_4 - S_2S_3}{S_2^2 - S_1S_3}$$

and similarly

$$\sigma_2 = \frac{S_2^2 - S_2S_4}{S_2^2 - S_1S_3}$$

The generic error-locator polynomial is

$$\det \begin{pmatrix} 1 & Z & Z^2 \\ S_3 & S_2 & S_1 \\ S_4 & S_3 & S_2 \end{pmatrix} = (S_1^2 - S_1S_3) + (S_1S_4 - S_2S_3)Z + (S_3^2 - S_2S_4)Z^2.$$
Example 7.5.13 Let $C$ be the narrow sense BCH code over $\mathbb{F}_{16}$ of length 15 and designed minimum distance 5 as in Example 7.3.14. Let
\[ r = (\alpha^5, \alpha^8, \alpha^{11}, \alpha^{10}, \alpha^7, \alpha^{12}, 1, \alpha, \alpha^{12}, \alpha^{14}, \alpha^{12}, \alpha^2, 0) \]
be a received word with respect to the code $C$ with 2 errors. Then $S_1 = \alpha^{12}$, $S_2 = \alpha^7$, $S_3 = 0$ and $S_4 = \alpha^2$. The formulas $S_2^{1} = S_2$ and $S_4^{1} = S_4$ of Example 7.5.11 are no longer valid, since this code is defined over $\mathbb{F}_{16}$ instead of $\mathbb{F}_2$. By the formulas in Example 7.5.12 the error-locator polynomial is
\[ 1 + Z + \alpha^{10}Z^2 \]
which has $\alpha^{-2}$ and $\alpha^{-8}$ as zeros. In this case the error positions are known, but the error values need some extra computation, since the values are not binary. This could be done by considering the error positions as erasures along the lines of Section 6.2.2. The next section gives an alternative with Forney’s formula.

7.5.5 Key equation and Forney’s formula

Consider the narrow sense BCH code $C$ with designed minimum distance $\delta$. So the defining set is $\{1, \ldots, \delta - 1\}$. Let $c(x) \in C$ be the transmitted codeword. Let $r(x) = c(x) + e(x)$ be the received word with error $e(x)$. Suppose that the number of errors $w = \text{wt}(e(x))$ is at most $(\delta - 1)/2$. The support of $e(x)$ will be denoted by $I$, that is $e_i \neq 0$ if and only if $i \in I$. So the error-locator polynomial is
\[ \sigma(Z) = \prod_{i \in I} (1 - \alpha^i Z) \]
with coefficients $\sigma_0 = 1$, $\sigma_1, \ldots, \sigma_w$.

Definition 7.5.14 The syndromes are $S_j = r(\alpha^j)$ for $1 \leq j \leq \delta - 1$. The syndrome polynomial $S(Z)$ is defined by
\[ S(Z) = \sum_{j=1}^{\delta-1} S_j Z^{j-1}, \]

Remark 7.5.15 The syndrome $S_j$ is equal to $e(\alpha^j)$, since $c(\alpha^j) = 0$, for all $j = 1, \ldots, \delta - 1$. Furthermore $2w \leq \delta - 1$. The Newton identities
\[ S_k + \sigma_1 S_{k-1} + \cdots + \sigma_w S_{k-w} = 0 \quad \text{for } k = w + 1, \ldots, 2w \]
imply that the $(k-1)$st coefficient of $\sigma(Z)S(Z)$ is zero for all $k = w + 1, \ldots, 2w$, since
\[ \sigma(Z)S(Z) = \sum_k \left( \sum_{i+j=k} \sigma_i S_j \right) Z^{k-1} \]
Hence there exist polynomials $q(Z)$ and $r(Z)$ such that
\[ \sigma(Z)S(Z) = r(Z) + q(Z)Z^{2w}, \quad \deg(r(Z)) < w. \]
In the following we will identify the remainder $r(Z)$.
Definition 7.5.16 The error-evaluator polynomial \( \omega(Z) \) is defined by

\[
\omega(Z) = \sum_{i \in I} e_i \alpha^i \prod_{i \neq j \in I} (1 - \alpha^j Z).
\]

Proposition 7.5.17 Let \( \sigma'(Z) \) be the formal derivative of \( \sigma(Z) \). Then the error values are given by Forney’s formula:

\[
e_i = -\frac{\omega(\alpha^{-i})}{\sigma'(\alpha^{-i})}
\]

for all error positions \( \alpha^i \).

Proof. Differentiating

\[
\sigma(Z) = \prod_{i \in I} (1 - \alpha^i Z)
\]

gives

\[
\sigma'(Z) = \sum_{i \in I} -\alpha^i \prod_{i \neq j \in I} (1 - \alpha^j Z).
\]

Hence

\[
\sigma'(\alpha^{-i}) = -\alpha^i \prod_{i \neq j \in I} (1 - \alpha^{-j})
\]

which is not zero. Substitution of \( \alpha^{-i} \) in \( \omega(Z) \) gives \( \omega(\alpha^{-i}) = -e_i \sigma'(\alpha^{-i}) \).

Remark 7.5.18 The polynomial \( \sigma(Z) \) has simple zeros. Hence \( \beta \) is not a zero of \( \sigma'(Z) \) if \( \beta \) is a zero of \( \sigma(Z) \), by Lemma 7.2.8. So the denominator in Proposition 7.5.17 is not zero. This proposition implies that \( \beta \) is not a zero of \( \omega(Z) \) if \( \beta \) is a zero of \( \sigma(Z) \). Hence the greatest common divisor of \( \sigma(Z) \) and \( \omega(Z) \) is one.

Proposition 7.5.19 The error-locator polynomial \( \sigma(Z) \) and the error-evaluator polynomial \( \omega(Z) \) satisfy the Key equation:

\[
\sigma(Z)S(Z) \equiv \omega(Z)(\text{mod } Z^{\delta-1}).
\]

Moreover if \( (\sigma_1(Z), \omega_1(Z)) \) is another pair of polynomials that satisfy the Key equation and such that \( \deg \omega_1(Z) < \deg \sigma_1(Z) \leq (\delta - 1)/2 \), then there exists a polynomial \( \lambda(Z) \) such that \( \sigma_1(Z) = \lambda(Z)\sigma(Z) \) and \( \omega_1(Z) = \lambda(Z)\omega(Z) \).

Proof. We have that \( S_j = r(\alpha^j) = e(\alpha^j) \) for all \( j = 1, 2, \ldots, \delta - 1 \). Using the definitions, interchanging summations and the sum formula for a geometric series we get

\[
S(Z) = \sum_{j=1}^{\delta-1} e(\alpha^j)Z^{j-1} = \sum_{j=1}^{\delta-1} \sum_{i \in I} e_i \alpha^{ij} Z^{j-1}
\]

\[
= \sum_{i \in I} e_i \alpha^i \sum_{j=1}^{\delta-1} (\alpha^i Z)^{j-1} = \sum_{i \in I} e_i \alpha^i \frac{1 - (\alpha^i Z)^{\delta-1}}{1 - \alpha^i Z}.
\]
Hence
\[ \sigma(Z)S(Z) = \prod_{j \in I} (1 - \alpha^j Z)S(Z) = \sum_{i \in I} e_i \alpha^i \left(1 - (\alpha^i Z)^{\delta - 1}\right) \prod_{i \neq j \in I} (1 - \alpha^j Z). \]
Therefore
\[ \sigma(Z)S(Z) \equiv \sum_{i \in I} e_i \alpha^i \prod_{i \neq j \in I} (1 - \alpha^j Z) \equiv \omega(Z) \pmod{Z^{\delta - 1}}. \]
Suppose that we have another pair \((\sigma_1(Z), \omega_1(Z))\) such that
\[ \sigma_1(Z)S(Z) \equiv \omega_1(Z) \pmod{Z^{\delta - 1}} \]
and \(\deg \omega_1(Z) < \deg \sigma_1(Z) \leq (\delta - 1)/2\). Then
\[ \sigma(Z)\omega_1(Z) \equiv \sigma_1(Z)\omega(Z) \pmod{Z^{\delta - 1}} \]
and the degrees of \(\sigma(Z)\omega_1(Z)\) and \(\sigma_1(Z)\omega(Z)\) are strictly smaller than \(\delta - 1\). Hence
\[ \sigma(Z)\omega_1(Z) = \sigma_1(Z)\omega(Z). \]
The greatest common divisor of \(\sigma(Z)\) and \(\omega(Z)\) is one by Remark 7.5.18. Therefore there exists a polynomial \(\lambda(Z)\) such that \(\sigma_1(Z) = \lambda(Z)\sigma(Z)\) and \(\omega_1(Z) = \lambda(Z)\omega(Z)\).

**Remark 7.5.20** In Remark 7.5.15 it is shown that the Newton identities give the Key equation \(\sigma(Z)S(Z) \equiv r(Z) \pmod{Z^{\delta - 1}}\). In Proposition 7.2 a new proof of the Key equation is given where the remainder \(r(Z)\) is identified as the error-evaluator polynomial \(\omega(Z)\). Conversely the Newton identities can be derived form this second proof.

**Example 7.5.21** Let \(C\) be the narrow sense BCH code of length 15 over \(F_{16}\) of designed minimum distance 5 and let \(r\) be the received word as in Example 7.5.13. The error-locator polynomial is \(\sigma(Z) = 1 + Z + \alpha^{10}Z^2\) which has \(\alpha^{-2}\) and \(\alpha^{-8}\) as zeros. The syndrome polynomial is \(S(Z) = \alpha^{12} + \alpha^7Z + \alpha^2Z^4\). Then
\[ \sigma(Z)S(Z) = \alpha^{12} + \alpha^2Z + \alpha^2Z^4 + \alpha^{12}Z^5. \]
Proposition 7.5.19 implies
\[ \omega(Z) \equiv \sigma(Z)S(Z) \equiv \alpha^{12} + \alpha^2Z \pmod{Z^4}. \]
Hence \(\omega(Z) = \alpha^{12} + \alpha^2Z\), since \(\deg(\omega(Z)) < \deg(\sigma(Z)) = 2\). Furthermore \(\sigma'(Z) = 1\). The error values are therefore
\[ e_2 = \omega(\alpha^{-2}) = \alpha^{11} \text{ and } e_8 = \omega(\alpha^{-8}) = \alpha^8 \]
by Proposition 7.5.17.

**Remark 7.5.22** Consider the BCH code \(C\) with \(\{b, b + 1, \ldots, b + \delta - 2\}\) as defining set. The syndromes are \(S_j = e(\alpha^j)\) for \(b \leq j \leq b + \delta - 2\). Adapt the above definitions as follows. The syndrome polynomial \(S(Z)\) is defined by
\[ S(Z) = \sum_{j=b}^{b+\delta-2} S_j Z^{j-b}, \]
CHAPTER 7. CYCLIC CODES

The error-evaluator polynomial \( \omega(Z) \) is defined by

\[
\omega(Z) = \sum_{i \in I} e_i \alpha^{ib} \prod_{i \neq j \in I} (1 - \alpha^j Z). 
\]

Show that the error-locator polynomial \( \sigma(Z) \) and the error-evaluator polynomial \( \omega(Z) \) satisfy the Key equation:

\[
\sigma(Z) \equiv \omega(Z) \pmod{Z^{d-1}}.
\]

Show that the error values are given by Forney’s formula:

\[
e_i = -\frac{\omega(\alpha^{-i})}{\alpha^{i(b-1)}\sigma'(\alpha^{-i})}
\]

for all error positions \( i \).

7.5.6 Exercises

7.5.1 Consider \( A(Z) = 2 + 6Z + 2Z^2 + 5Z^3 \) in \( \mathbb{F}_7[Z] \). Show that \( A(Z) \) is the MS polynomial of codeword \( a(x) \) of a cyclic code of length 6 over \( \mathbb{F}_7 \) with primitive element \( \alpha = 3 \). Compute the zeros and coefficients of \( a(x) \).

7.5.2 Give a proof of Proposition 7.5.7.

7.5.3 In case \( w = 2 \) we have that \( \sigma_1 = -x_1 x_2 \), \( \sigma_2 = x_1 x_2 \) and \( A_i = y_1 x_1^i + y_2 x_2^i \). Substitute these formulas in the Newton identities in order to check their validity.

7.5.4 Let \( C \) be the binary narrow sense BCH code of length 15 and designed minimum distance 5 as in Example 7.5.11. Let

\[
r = (1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1)
\]

be a received word with respect to the code \( C \) with 2 errors. Find the codeword sent.

7.5.5 Consider the narrow-sense BCH code with defining minimum distance 7. Then the syndromes \( S_1, S_2, \ldots, S_6 \) of a received word are known. Compute the coefficients of the generic error-locator polynomial. Show that in the binary case the generic error-locator polynomial becomes

\[
(S_3 + S_4^2) + (S_1 S_3 + S_1^2) Z + (S_5 + S_1^2 S_3) Z^2 + (S_3^2 + S_1 S_5 + S_1^2 S_3 + S_3^2) Z^3,
\]

by using \( S_2 = S_1^2 \), \( S_4 = S_1^4 \) and \( S_6 = S_2^4 \) and after division by the common factor \( S_3^2 + S_1 S_5 + S_1^2 S_3 + S_3^2 \).

7.5.6 Let \( C \) be the narrow sense BCH code of length 15 over \( \mathbb{F}_{16} \) of designed minimum distance 5 as in Examples 7.5.13 and 7.5.21. Let

\[
r = (\alpha^8, \alpha^7, \alpha^1, \alpha^{11}, \alpha^3, \alpha^5, \alpha^{10}, \alpha^{11}, \alpha^{10}, \alpha^7, \alpha^4, \alpha^{10}, 0, 1, \alpha^5)
\]

be a received word with respect to the code \( C \) with 2 errors. Find the error positions. Determine the error values by Forney’s formula.

7.5.7 Show the validity of the Key equation and Forney’s formula as claimed in Remark 7.5.22.
7.6 Notes

6.4.2: iterated HT, iterated Roos bound.

6.4.3: symmetric Roos bound.

In many cases of binary codes of length at most 62 the shift bound is equal to the minimum distance, see [?]. For about 95\% of all ternary codes of length at most 40 the shift bound is equal to the minimum distance, see [?].

In a discussion with B.-Z. Shen we came to the following generalization of independent sets and the shift bound, see also Shen and Tzeng [?] and Augot, Charpin and Sendrier [?] on generalized Newton identities.

Lemma 7.4.37 is a generalization of a theorem of van Lint and Wilson [?, Theorem 11].

Generalization of shift bound for linear codes.

Linear complexity and the pseudo rank bound.

Shift bound for gen. Hamming weights.

Conjecture of (non) existence of asymptotically good cyclic codes Assmus, Turyn 1966.

***Blahut’s theorem, Massey in Festschrift on DFT and PS polynomial***

Fundamental iterative algorithm.
Chapter 8

Polynomial codes

Ruud Pellikaan

****

8.1 RS codes and their generalizations

Reed-Solomon codes will be introduced as special cyclic codes. We will show that these codes are MDS and can be obtained by evaluating certain polynomials. This gives rise to a generalization of these codes. Fractional transformations are defined and related to the automorphism group of generalized Reed-Solomon code.

8.1.1 Reed-Solomon codes

Consider the following definition of Reed-Solomon codes over the finite field $\mathbb{F}_q$.

**Definition 8.1.1** Let $\alpha$ be a primitive element of $\mathbb{F}_q$. Let $n = q - 1$. Let $b$ and $k$ be non-negative integers such that $0 \leq b, k \leq n$. Define the generator polynomial $g_{b,k}(X)$ by

$$g_{b,k}(X) = (X - \alpha^b) \cdots (X - \alpha^{b+k+n-k-1}).$$

The Reed-Solomon (RS) code $RS_k(n,b)$ is by definition the $q$-ary cyclic code with generator polynomial $g_{b,k}(X)$. In the literature the code is also denoted by $RS_b(n,k)$.

**Proposition 8.1.2** The code $RS_k(n,b)$ has length $n = q - 1$, is cyclic, linear and MDS of dimension $k$. The dual of $RS_k(n,b)$ is equal to $RS_{n-k}(n,n-b+1)$.

**Proof.** The code $RS_k(n,b)$ is of length $q - 1$, cyclic and linear by definition. The degree of the generator polynomial is $n - k$, so the dimension of the code is $k$ by Proposition 7.1.21. The complete defining set is $\{b, b+1, \ldots, b+n-k-1\}$.
and has \( n - k \) consecutive elements. Hence the minimum distance \( d \) is at least \( n - k + 1 \) by the BCH bound of Proposition 7.3.3. The generator polynomial \( g_{b,k}(X) \) has degree \( n - k \), so \( g_{b,k}(x) \) is a codeword of weight at most \( n - k + 1 \). Hence \( d \) is at most \( n - k + 1 \). Also the Singleton bound gives that \( d \) is at most \( n - k + 1 \). Hence \( d = n - k + 1 \) and the code is MDS. Another proof that the parameters are \([n,k,n-k+1]\) will be given in Proposition 8.1.14.

The complete defining set of \( RS_k(n,b) \) is the subset \( U \) consisting of \( n - k \) consecutive elements:

\[
U = \{b,b+1,\ldots,b+n-k-1\}.
\]

Hence \( Z_n \setminus \{-i|i \in U\} \) is the complete defining set of the dual of \( RS_k(n,b) \) by Proposition 7.2.58. But

\[
Z_n \setminus \{-i|i \in U\} = Z_n \setminus \{n - (b + n - k - 1),\ldots,n - (b + 1),n - b\} = \\
\{n - b + 1,n - b + 2,\ldots,n - b + k\}
\]

is the complete defining set of \( RS_{n-k}(n,n-b+1) \).

Another description of RS codes will be given by evaluating polynomials.

**Definition 8.1.3** Let \( f(X) \in \mathbb{F}_q[X] \). Let \( ev(f(X)) \) be the evaluation of \( f(X) \) defined by

\[
ev(f(X)) = (f(1),f(\alpha),\ldots,f(\alpha^{n-1})).
\]

**Proposition 8.1.4** We have that

\[
RS_k(n,b) = \{ ev(X^{n-b+1}f(X)) \mid f(X) \in \mathbb{F}_q[X], \ deg(f) < k \}.
\]

**Proof.** The dual of \( RS_k(n,b) \) is \( RS_{n-k}(n,n-b+1) \) by Proposition 8.1.2, which has \( \{n - b + 1,\ldots,n - b + k\} \) as complete defining set. So \( RS_{n-k}(n,n-b+1) \) has \( H = (\alpha^i|n - b + 1 \leq i \leq n - b + k, 0 \leq j \leq n - 1) \) as parity check matrix, by Remark 7.3.2 and the proof of Proposition 7.3.3. That means that \( H \) is a generator matrix of \( RS_k(n,k) \). The rows of \( H \) are \( ev(X^i) \) for \( n - b + 1 \leq i \leq n - b + k \). So they generate the space \{ \( ev(X^{n-b+1}f(X)) \mid \deg(f) < k \) \}. □

**Example 8.1.5** Consider \( RS_3(7,1) \). It is a cyclic code over \( \mathbb{F}_8 \) with generator polynomial

\[
g_{1,3}(X) = (X - \alpha)(X - \alpha^2)(X - \alpha^3)(X - \alpha^4)
\]

where \( \alpha \) is a primitive element of \( \mathbb{F}_8 \) satisfying \( \alpha^4 = \alpha + 1 \). Then

\[
g_{1,3}(X) = \alpha^3 + \alpha X + X^2 + \alpha^3 X^3 + X^4
\]

In the second description we have that

\[
RS_3(7,1) = \{ ev(f(X)) \mid f(X) \in \mathbb{F}_q[X], \ deg(f) < 3 \}
\]

The matrix in Exercise 7.1.5 is obtained by evaluating the monomials \( 1, X \) and \( X^2 \) at \( \alpha^j \) for \( j = 0,1,\ldots,6 \). It is a generating matrix of \( RS_3(7,1) \).
8.1. RS CODES AND THEIR GENERALIZATIONS

8.1.2 Extended and generalized RS codes

Definition 8.1.6 The extended RS code $ERS_k(n, b)$ is the extension of the code $RS_k(n, b)$.

The code $ERS_k(n, 1)$ has also a description by means of evaluations.

Proposition 8.1.7 We have that

$$ERS_k(n, 1) = \{ (f(1), f(α), \ldots, f(α^{n-1}), f(0)) \mid f(X) ∈ \mathbb{F}_q[X], \deg(f) < k \}. $$

Proof. If $C$ is a code of length $n$, then by Definition 3.1.6 the extended code $C^e$ is given by

$$C^e = \{ (c, -\sum_{i=0}^{n-1} c_i) \mid c ∈ C \}. $$

So we have to show that

$$f(0) + f(1) + f(α) + \cdots + f(α^{n-1}) = 0$$

for all polynomials $f(X) ∈ \mathbb{F}_q[X]$ of degree at most $k - 1$. By linearity it is enough to show that this is the case for all monomials of degree at most $k - 1$.

Let $f(X)$ be the monomial $X^i$ with $0 ≤ i < n$. Then

$$\sum_{j=0}^{n-1} f(α^j) = \sum_{j=0}^{n-1} α^{ij} = \begin{cases} n & \text{if } i = 0, \\ 0 & \text{if } 0 < i < n, \end{cases}$$

by Lemma 7.5.2. Now $n = q - 1 = -1$ in $\mathbb{F}_q$. So in both cases we have that this sum is equal to $-f(0)$.

Definition 8.1.8 Let $𝔽$ be a field. Let $f(X) = f_0 + f_1X + \cdots + f_kX^k$ be an element of $𝔽[X]$ and $α ∈ 𝔽$. Then the evaluation of $f(X)$ at $α$ is given by

$$f(α) = f_0 + f_1α + \cdots + f_kα^k.$$ 

Let

$$L_k = \{ f(X) ∈ 𝔽_q[X] \mid \deg(f(X)) ≤ k \}. $$

The evaluation map

$$ev_{k, α} : L_k → 𝔽$$

is given by $ev_{k, α}(f(X)) = f(α)$. Furthermore the evaluation at infinity is defined by $ev_{k, ∞}(f(X)) = f_k$.

Remark 8.1.9 The evaluation map is linear. Furthermore $ev_{k, ∞}(f(X)) = 0$ if and only if $f(X)$ has degree at most $k - 1$ for all $f(X) ∈ L_k$. The map $ev_{k, α}$ does not depend on $k$ if $α ∈ 𝔽$. The notation $f(∞)$ will be used instead of $ev_{k, ∞}(f(X))$, but notice that this depends on $k$ and the implicit assumption that $f(X)$ has degree at most $k$.

Definition 8.1.10 Let $n$ be an arbitrary integer such that $1 ≤ n ≤ q$. Let $a$ be an $n$-tuple of mutually distinct elements of $𝔽_q∪\{∞\}$. Let $b$ be an $n$-tuple of nonzero elements of $𝔽_q$. Let $k$ be an arbitrary integer such that $0 ≤ k ≤ n$. The generalized RS code $GRS_k(a, b)$ is defined by

$$GRS_k(a, b) = \{ (f(a_1)b_1, f(a_2)b_2, \ldots, f(a_n)b_n) \mid f(X) ∈ 𝔽_q[X], \deg(f) < k \}. $$
The following two examples show that the generalized RS codes are indeed generalizations of both RS codes as extended RS codes.

**Example 8.1.11** Let \( \alpha \) be a primitive element of \( \mathbb{F}_q^* \). Let \( n = q - 1 \). Define \( a_j = \alpha^{j-1} \) and \( b_j = a_j^{n-b+1} \) for \( j = 1, \ldots, n \). Then \( RS_k(n,b) = GRS_k(a,b) \).

**Example 8.1.12** Let \( \alpha \) be a primitive element of \( \mathbb{F}_q^* \). Let \( n = q \) and \( a_1 = 0 \) and \( b_1 = 1 \). Define \( a_j = \alpha^{j-1} \) and \( b_j = a_j^{n-b+1} \) for \( j = 2, \ldots, n \). Then \( ERS_k(n,1) = GRS_k(a,b) \).

**Example 8.1.13** The BCH code over \( \mathbb{F}_q \) with defining set \( \{b, b+1, \ldots, b+\delta-2\} \) and length \( n \), can be considered as a subfield subcode over \( \mathbb{F}_q \) of a generalized RS code over \( \mathbb{F}_{q^n} \) where \( m \) is such that \( n \) divides \( q^m - 1 \).

**Proposition 8.1.14** Let \( 0 \leq k \leq n \leq q \). Then \( GRS_k(a,b) \) is an \( \mathbb{F}_q \)-linear MDS code with parameters \( [n,k,n-k+1] \).

**Proof.** Notice that a linear code \( C \) stays linear under the linear map \( c \mapsto (b_1 c_1, \ldots, b_n c_n) \), and that the parameters remain the same if the \( b_i \) are all nonzero. Hence we may assume without loss of generality that \( b \) is the all ones vector.

Consider the evaluation map \( ev_{k-1,a} : L_{k-1} \rightarrow \mathbb{F}_q^n \) defined by

\[
ev_{k-1,a}(f(X)) = (f(a_1), f(a_2), \ldots, f(a_n)).
\]

This map is linear and \( L_{k-1} \) is a linear space of dimension \( k \). Furthermore \( GRS_k(a,b) \) is the image of \( L_{k-1} \) under \( ev_{k-1,a} \).

Suppose that \( a_j \in \mathbb{F}_q \) for all \( j \). Let \( f(X) \in L_{k-1} \) and \( ev_{k-1,a}(f(X)) = 0 \). Then \( f(X) \) is of degree at most \( k-1 \) and has \( n \) zeros. But \( k-1 < n \) by assumption. So \( f(X) \) is the zero polynomial. Hence the restriction of the map \( ev_{k-1,a} \) to \( L_{k-1} \) is injective, and \( GRS_k(a,b) \) has the same dimension \( k \) as \( L_{k-1} \).

Let \( c \) be a nonzero codeword of \( GRS_k(a,b) \) of weight \( d \). Then there exists a nonzero polynomial \( f(X) \) of degree at most \( k-1 \) such that \( ev_{k-1,a}(f(X)) = c \).

The zeros of \( f(X) \) among the \( a_1, \ldots, a_n \) correspond to the zero coordinates of \( c \). So the number of zeros of \( f(X) \) among the \( a_1, \ldots, a_n \) is equal to the number of zero coordinates of \( c \), which is \( n - d \). Hence \( n - d \leq \deg f(X) \leq k-1 \), that is \( d \geq n-k+1 \).

The evaluation of the polynomial \( f(X) = \prod_{i=1}^{k-1}(X-a_i) \) gives an explicit codeword of weight \( n-k+1 \). Also the Singleton bound gives that \( d \leq n-k+1 \). Therefore the minimum distance of the generalized RS code is equal to \( n-k+1 \) and the code is MDS.

In case \( a_j = \infty \) for some \( j \), then \( a_i \in \mathbb{F}_q \) for all \( i \neq j \). Now \( f(a_j) = 0 \) implies that the degree of \( f(X) \) is a most \( k-2 \). So the above proof applies for the remaining \( n-1 \) elements and polynomials of degree at most \( k-2 \). \( \diamond \)

**Remark 8.1.15** The monomials \( 1, X, \ldots, X^{k-1} \) form a basis of \( L_{k-1} \). Suppose that \( a_j \in \mathbb{F}_q \) for all \( j \). Then evaluating these monomials gives a generator matrix with entries \( a_j^{-1} b_j \) of the code \( GRS_k(a,b) \). If \( b \) is the all ones vectors, then the matrix \( G_k(a) \) of Proposition 3.2.10 is a generator matrix of \( GRS_k(a,b) \). If \( a_j = \infty \), then \( ev_{k-1,a}(b_i X^{i-1}) = 0 \) for all \( i < k-1 \) and \( ev_{k-1,a}(b_i X^{k-1}) = b_j \). Hence \( (0, \ldots, 0, b_j)^T \) is the corresponding column vector of the generator matrix.
**Remark 8.1.16** A generalized RS code is MDS by Proposition 8.1.14. So any $k$ positions can be used to encode systematically. That means that there is a generator matrix $G$ of the form $(I_k|P)$, where $I_k$ is the $k \times k$ identity matrix and $P$ a $k \times (n-k)$ matrix. The next proposition gives an explicit description of $P$.

**Proposition 8.1.17** Let $b$ be an $n$-tuple of nonzero elements of $\mathbb{F}_q$. Let $a$ be an $n$-tuple of mutually distinct elements of $\mathbb{F}_q \cup \{\infty\}$. Define $[a_i,a_j] = a_i - a_j$, $[\infty,a_j] = 1$ and $[a_i,\infty] = -1$ for $a_i,a_j \in \mathbb{F}_q$. Then $\text{GRS}_k(a,b)$ has a generator matrix of the form $(I_k|P)$, where

$$p_{ij} = \frac{b_{j+k} \prod_{t=1,t\neq i}^{k}[a_{j+k},a_i]}{b_i \prod_{t=1,t\neq i}^{k}[a_i,a_t]}$$

for $1 \leq i \leq k$ and $1 \leq j \leq n-k$.

**Proof.** Assume first that $b$ is the all ones vector. Let $g_i$ be the $i$-th row of this generator matrix. Then this corresponds to a polynomial $g_i(X)$ of degree at most $k-1$ such that $g_i(a_j) = 1$ and $g_i(a_t) = 0$ for all $1 \leq t \leq k$ and $t \neq i$. By the Lagrange Interpolation Theorem there is a unique polynomial with these properties and it is given by

$$g_i(X) = \frac{\prod_{t=1,t\neq i}^{k}[X-a_t]}{\prod_{t=1,t\neq i}^{k}[a_i,a_t]}.$$  

Notice that if $a_i = \infty$, then $g_i(X)$ satisfies also the required conditions, since $[\infty,a_i] = [\infty, a_i] = 1$ by definition and $g_i(X)$ is a monic polynomial of degree $k-1$ so $g_i(\infty) = 1$. Hence $P_{ij} = g_i(a_j+k)$ is of the described form also in case $a_j+k = \infty$.

For arbitrary $b$ we have to multiply the $j$-th column of $G$ with $b_j$. In order to get the identity matrix back, the $i$-th row is divided by $b_i$. $\diamond$

**Corollary 8.1.18** Let $(I_k|P)$ be the generator matrix of the code $\text{GRS}_k(a,b)$. Then

$$p_{iu}p_{jv}[a_1,a_{k+u}][a_j,a_{k+v}] = p_{ju}p_{iv}[a_j,a_{k+u}][a_i,a_{k+v}]$$

for all $1 \leq i,j \leq k$ and $1 \leq u,v \leq n-k$.

**Proof.** This is left as an exercise. $\diamond$

In Section 3.2.1 both generalized Reed-Solomon and Cauchy codes were introduced as examples of MDS codes. The following corollary shows that in fact these codes are the same.

**Corollary 8.1.19** Let $a$ be an $n$-tuple of mutually distinct elements of $\mathbb{F}_q$. Let $b$ be an $n$-tuple of nonzero elements of $\mathbb{F}_q$. Let

$$c_i = \begin{cases} 
  b_i \prod_{t=1,t\neq i}^{k}[a_i,a_t] & \text{if } 1 \leq i \leq k, \\
  b_i \prod_{t=1}^{k}[a_i,a_t] & \text{if } k+1 \leq i \leq n.
\end{cases}$$

Then $\text{GRS}_k(a,b) = C_k(a,c)$. 

---

8.1. RS CODES AND THEIR GENERALIZATIONS
Proposition 8.1.21 Let \( \mathbf{b}^\perp \) be the vector with entries
\[
b^\perp_j = \frac{1}{b_j \prod_{\ell \neq j}[a_j, a_\ell]} \]
for \( j = 1, \ldots, n \). Then \( GRS_{n-k}(\mathbf{a}, \mathbf{b}^\perp) \) is the dual code of \( GRS_k(\mathbf{a}, \mathbf{b}) \).

Proof. Let \( G = (I_k|P) \) be the generator matrix of \( GRS_k(\mathbf{a}, \mathbf{b}) \) with \( P \) as obtained in Proposition 8.1.17. In the same way \( GRS_{n-k}(\mathbf{a}, \mathbf{b}^\perp) \) has a generator matrix \( H \) of the form \( (Q|I_{n-k}) \) with
\[
Q_{ij} = \frac{c_j \prod_{t=k+1, t \neq i+k}[a_j, a_t]}{c_{i+k} \prod_{t=k+1, t \neq i+k}[a_{i+k}, a_t]} \]
for \( 1 \leq i \leq n-k \) and \( 1 \leq j \leq k \). After substituting the values for \( b^\perp_j \) and canceling the same terms in numerator and denominator we see that \( Q = -P^T \). Hence \( H \) is a parity check matrix of \( GRS_k(\mathbf{a}, \mathbf{b}) \) by Proposition 2.3.30. \( \Box \)

Example 8.1.22 This is a continuation of Example 8.1.11. Let \( \mathbf{b} \) be the all ones vector. Then \( RS_k(n, 1) = GRS_k(\mathbf{a}, \mathbf{b}) \) and \( RS_k(n, 0) = GRS_k(\mathbf{a}, \mathbf{a}) \). Furthermore the dual of \( RS_k(n, 1) \) is \( RS_{n-k}(n, 0) \) by Proposition 8.1.2. So \( RS_k(n, 1)^\perp = GRS_{n-k}(\mathbf{a}, \mathbf{a}) \). Alternatively, Proposition 8.1.21 gives that the dual of \( GRS_k(\mathbf{a}, \mathbf{b}) \) is equal to \( GRS_{n-k}(\mathbf{a}, \mathbf{c}) \) with \( c_j = 1/\prod_{\ell \neq j}(a_j - a_\ell) \). We leave it as an exercise to show that \( c_j = -a_j \) for all \( j \).

Example 8.1.23 Consider the code \( RS_3(7, 1) \). Let \( \alpha \in \mathbb{F}_8 \) be an element with \( \alpha^3 = 1 + \alpha \). Let \( \mathbf{a}, \mathbf{b} \in \mathbb{F}_8^8 \) with \( a_i = \alpha^{i-1} \) and \( \mathbf{b} \) the all ones vector. Then \( RS_3(7, 1) = GRS_3(\mathbf{a}, \mathbf{b}) \) by Remark 8.1.11. Let \( (I_3|P) \) be a generator matrix of this code. Let \( g_1(X) \) be the quadratic polynomial such that \( g_1(1) = 1, g_1(\alpha) = 0 \) and \( g_1(\alpha^2) = 0 \). Then
\[
g_1(X) = \frac{(X + \alpha)(X + \alpha^2)}{(1 + \alpha)(1 + \alpha^2)}.
\]
Hence \( g_1(\alpha^3) = \alpha^3, g_1(\alpha^4) = \alpha, g_1(\alpha^5) = 1 \) and \( g_1(\alpha^6) = \alpha^3 \) are the entries of the first row of \( P \). Continuing in this way we get
\[
P = \begin{pmatrix}
\alpha^3 & \alpha & 1 & \alpha^3 \\
\alpha^6 & \alpha^6 & 1 & \alpha^2 \\
\alpha^5 & \alpha^6 & 1 & \alpha^4
\end{pmatrix}.
\]
The dual of $RS_q(7,1)$ is $RS_q(7,0)$, by Proposition 8.1.2, which is equal to $GRS_q(a,a)$. This is in agreement with Proposition 8.1.21, since $c_j = a_j$ for all $j$.

**Remark 8.1.24** Let $b$ be an $n$-tuple of nonzero elements of $F_q$. Let $a$ be an $n$-tuple of mutually distinct elements in $F_q \cup \{\infty\}$ and $a_k = \infty$. Then $GRS_k(a,b)$ has a generator matrix of the form $(I_k | P)$, where

$$p_{ij} = \frac{c_{j+k}c_j^{-1}}{a_{j+k} - a_i}$$

for all $1 \leq i \leq k - 1$ and $1 \leq j \leq n - k$ and

$$p_{kj} = c_{j+k}c_k^{-1}$$

for $1 \leq j \leq n - k$, with

$$c_i = \begin{cases} b_i \prod_{t=1, t \neq i}^{k-1} (a_i - a_t) & \text{if } 1 \leq i \leq k - 1, \\ b_k & \text{if } i = k, \\ b_i \prod_{t=1}^{k-1} (a_i - a_t) & \text{if } k + 1 \leq i \leq n. \end{cases}$$

by Corollary 8.1.19.

### 8.1.3 GRS codes under transformations

**Proposition 8.1.25** Let $n \geq 2$. Let $a$ be in $F_q^n$ consisting of mutually distinct entries. Let $b$ be an $n$-tuple of nonzero elements of $F_q$. Let $1 \leq i, j \leq n$ and $i \neq j$. Then there exists a $b'$ in $F_q^n$ with nonzero entries and an $a'$ in $F_q^n$ consisting of mutually distinct entries such that $a'_i = 0$, $a'_j = 1$ and

$$GRS_k(a,b) = GRS_k(a',b').$$

**Proof.** We may assume without loss of generality that $b = 1$. Consider the linear polynomials $l(X) = (X-a_i)/(a_j - a_i)$ and $m(X) = (a_j - a_i)X + a_i$. Then $l(m(X)) = X$ and $m(l(X)) = X$. Now $L_k$ is the vector space of all polynomials in the variable $X$ of degree at most $k$. Then the maps $\lambda, \mu : L_{k-1} \to L_{k-1}$ defined by $\lambda(f(X)) = f(l(X))$ and $\mu(g(X)) = g(m(X))$ are both linear and inverses of each other. Hence $\lambda$ and $\mu$ are automorphisms of $L_{k-1}$. Let $a'_i = l(a_i)$ for all $t$. Then the $a'_i$ are mutually distinct, since the $a_i$ are mutually distinct and $l(X)$ defines a bijection of $F_q$. Furthermore $a'_i = l(a_i) = 0$ and $a'_j = l(a_j) = 1$. Now $\text{ev}_{k-1,a}(f(l(X)))$ is equal to

$$(f(l(a_1)), \ldots, f(l(a_n))) = (f(a'_1), \ldots, f(a'_n)) = \text{ev}_{k-1,a'}(f(X)).$$

Finally

$$GRS_k(a',1) = \{\text{ev}_{k-1,a'}(f(X)) \mid f(X) \in L_{k-1}\}$$

and $GRS_k(a,1)$ is equal to

$$\{\text{ev}_{k-1,a}(g(X)) \mid g(X) \in L_{k-1}\} = \{\text{ev}_{k-1,a}(f(l(X))) \mid f(X) \in L_{k-1}\}.$$
CHAPTER 8. POLYNOMIAL CODES

Remark 8.1.26 ***Introduction of GRS with \( a_i = \infty \) as in Remark 8.1.15. Refer to forthcoming section of AG codes on the projective line. We leave the proof of the fact that we may assume furthermore \( a'_i = \infty \) as an exercise to the reader. For this one has to consider the fractional transformations

\[
\frac{aX + b}{cX + d}
\]

with \( ad - bc \neq 0 \). The set of fractional transformations with entries in a field \( F \) form a group with the composition of maps as group operation and determine the product and inverse.

Consider the map from \( GL(2, F) \) to the group of fractional transformations with entries in \( F \) defined by

\[
\begin{pmatrix} a & b \\ c & d \end{pmatrix} \mapsto \frac{aX + b}{cX + d}.
\]

Then this map is a morphism of groups and that the kernel of this map consists of the diagonal matrices \( aI_2 \) with \( a \neq 0 \).

Remark 8.1.27 ***Definition of evaluation of a rational function.....

Let \( \phi(X) \) be a fractional transformation and \( a \in F_q \cup \{\infty\} \) and \( f(X) \in F[X] \). Then

\[
ev_{k, \phi(a)}(f(X)) = ev_{k, a}(f(\phi(X))).
\]

This follows straightforward from the definitions in case \( a \) is in \( F \) and \( a \) not a zero of the denominator of \( \phi(X) \).....

***projective transformations of the projective line***

Proposition 8.1.28 Let \( n \geq 3 \). Let \( a \) be an \( n \)-tuple of mutually distinct entries in \( F_q \cup \{\infty\} \). Let \( b \) be an \( n \)-tuple of nonzero elements of \( F_q \). Let \( i, j \) and \( l \) be three mutually distinct integers between 1 and \( n \). Then there exists a \( b' \) in \( F^n_q \) with nonzero entries and an \( n \)-tuple \( a' \) consisting of mutually distinct entries in \( F_q \cup \{\infty\} \) such that \( a'_i = 0 \), \( a'_j = 1 \) and \( a'_l = \infty \) and

\[
GRS_k(a, b) = GRS_k(a', b').
\]

Proof. This is shown similarly as the proof of Proposition 8.1.25 using fractional transformations instead and is left as an exercise. \( \diamond \)

Now suppose that a generator matrix of the code \( GRS_k(a, b) \) is given. Is it possible to retrieve \( a \) and \( b \)? The pair \( (a, b) \) is not unique by the action of the fractional transformations. The following proposition gives an answer to this question.

Proposition 8.1.29 Let \( n \geq 3 \). Let \( a \) and \( a' \) be \( n \)-tuples with mutually distinct entries in \( F_q \cup \{\infty\} \). Let \( b \) and \( b' \) be \( n \)-tuples of nonzero elements of \( F_q \). Let \( i, j \) and \( l \) be three mutually distinct integers between 1 and \( n \). If \( a'_i = a_i \), \( a'_j = a_j \), \( a'_l = \infty \) and \( GRS_k(a, b) = GRS_k(a', b') \), then \( a' = a \) and \( b' = \lambda b \) for some nonzero \( \lambda \) in \( F_q \).

Proof. The generalized RS code is MDS, so it is systematic at the first \( k \) positions and it has a generator matrix of the form \( (I_k|P) \) such that the entries of \( P \) are nonzero. Let

\[
c = (p_{11}, \ldots, p_{k1}, 1, p_{k1}/p_{k2}, \ldots, p_{k1}/p_{n(n-k)})
\]
8.1. RS CODES AND THEIR GENERALIZATIONS

Let $G = c \ast (I_k | P)$. Then $G$ is the generator matrix of a generalized equivalent code $C$. Dividing the $i$-th row of $G$ by $p_{i1}$ gives another generator matrix $G'$ of the same code $C$ such that the $(k+1)$-th column of $G'$ is the all ones vector and the $k$-th row is of the form $(0, \ldots, 1, \ldots, 1)$. So we may suppose without loss of generality that the generator matrix of generalized RS code is of the form $(I_k | P)$ with $p_{i1} = 1$ for all $i = 1, \ldots, k$ and $p_{kj} = 1$ for all $j = 1, \ldots, n - k$.

After a permutation of the positions we may suppose without loss of generality that $l = k$, $i = k + 1$ and $j = k + 2$. After a fractional transformation we may assume that $g_{k+1} = a_{k+1} = 0$, $g_{k+2} = a_{k+2} = 1$ and $g_k = a_k = \infty$ by Proposition 8.1.28.

Remark 8.1.24 gives that there exists an $n$-tuple $c$ with nonzero entries in $\mathbb{F}_q$ such that

$$p_{ij} = \frac{c_{j+k}c_i^{-1}}{a_{j+k} - a_i} \quad \text{for all } 1 \leq i \leq k - 1 \text{ and } 1 \leq j \leq n - k$$

and

$$p_{kj} = c_{j+k}c_k^{-1} \quad \text{for } 1 \leq j \leq n - k.$$  

Hence $p_{ij} = c_{j+k}c_k^{-1} = 1$. So $c_{j+k} = c_k$ for all $j = 1, \ldots, n - k$. Multiplying all entries of $c$ with a nonzero constant gives the same code. Hence we may assume without loss of generality that $c_{j+k} = c_k = 1$ for all $j = 1, \ldots, n - k$. Therefore $c_j = 1$ for all $j \geq k$.

Let $i < k$. Then $p_{i1} = c_{k+1}/(c_i(a_{k+1} - a_i)) = 1$, $c_{k+1} = 1$ and $a_{k+1} = 0$. So $p_{i1} = -1/(a_ic_i) = 1$. Hence $a_ic_i = -1$.

Likewise $p_{i2} = c_{k+2}/(c_i(a_{k+2} - a_i))$, $c_{k+2} = 1$ and $a_{k+2} = 1$. So

$$p_{i2} = \frac{1}{((1 - a_i)c_i)} = \frac{1}{(c_i + 1)}, \quad \text{since } a_ic_i = -1.$$  

Hence

$$c_i = \frac{(1 - p_{i2})}{p_{i2}} \quad \text{and} \quad a_i = \frac{p_{i2}}{(p_{i2} - 1)} \quad \text{for all } i < k.$$  

Finally $p_{ij} = c_{k+j}/(c_i(a_{k+j} - a_i))$ and $c_{k+j} = 1$. So $a_{k+j} - a_i = 1/(c_ip_{ij})$. Hence $a_{k+j} = a_i - a_i/p_{ij}$, since $a_i = -1/c_i$. Combining this with the expression for $a_i$ gives

$$a_{j+k} = \frac{p_{i2}}{(p_{i2} - 1)} \cdot \frac{(p_{ij} - 1)}{p_{ij}}.$$  

Therefore $a$ and $c$ are uniquely determined. So also $b$ is uniquely determined, since

$$b_i = \begin{cases} 
  c_i/\prod_{t=1, t \neq i}^{k-1}(a_i - a_t) & \text{if } 1 \leq i \leq k - 1, \\
  c_k & \text{if } i = k, \\
  c_i/\prod_{t=1}^{k-1}(a_i - a_t) & \text{if } k + 1 \leq i \leq n.
\end{cases}$$  

by Remark 8.1.24.

- $\text{PAut}(GRS(a, b)) = \ldots$ and $\text{MAut}(GRS(a, b)) = \ldots$.

- What is the number of GRS codes?

***
Example 8.1.30 Let $G$ be the generator matrix of a generalized Reed-Solomon code with entries in $\mathbb{F}_7$ given by

$$G = \begin{pmatrix} 6 & 1 & 1 & 6 & 2 & 2 & 3 \\ 3 & 4 & 1 & 1 & 5 & 4 & 3 \\ 1 & 0 & 3 & 3 & 6 & 0 & 1 \end{pmatrix}.$$ 

Then $\text{rref}(G) = (I_3|A)$ with

$$A = \begin{pmatrix} 1 & 3 & 3 & 6 \\ 4 & 4 & 6 & 6 \\ 3 & 1 & 6 & 3 \end{pmatrix}.$$ 

So we want to find a vector $a$ consisting of mutually distinct entries in $\mathbb{F}_7 \cup \{\infty\}$ and $b$ in $\mathbb{F}_7^*$ with nonzero entries such that $C = \text{GRS}_3(a, b)$. Now $C' = (1, 4, 3, 1, 5, 5, 6) \ast C$ has a generator matrix of the form $(I_3 | A')$ with

$$A' = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 5 & 4 & 2 \\ 1 & 4 & 3 & 6 \end{pmatrix}.$$ 

We may assume without loss of generality that $a_4 = 0$, $a_5 = 1$ and $a_3 = \infty$ by Proposition 8.1.25.

8.1.4 Exercises

8.1.1 Show that in $\text{RS}_3(7, 1)$ the generating codeword $g_{1,3}(x)$ is equal to $\alpha \text{ev}(1) + \alpha^5 \text{ev}(X) + \alpha^4 \text{ev}(X^2)$.

8.1.2 Compute the parity check polynomial of $\text{RS}_3(7, 1)$ and the generator polynomial of $\text{RS}_3(7, 1)^\perp$ by means of Proposition 7.1.37, and verify that it is equal to $g_{0,4}(X)$ according to Proposition 8.1.2.

8.1.3 Give the generator matrix of $\text{RS}_4(7, 1)$ of the form $(I_4 | P)$, where $P$ a $4 \times 3$ matrix.

8.1.4 Show directly, that is without the use of Proposition 8.1.4, that the code $\{ \text{ev}(X^{n-b+1}f(X)) \mid \deg(f) < k \}$ is cyclic.

8.1.5 Give another proof of the fact in Proposition 8.1.2 that the dual of $\text{RS}_k(n, b)$ is equal to $\text{RS}_{n-k}(n, n-b+1)$ using the description with evaluations of Proposition 8.1.4 and that the inner product of codewords of the two codes is zero.

8.1.6 Let $n = q - 1$. Let $a_1, \ldots, a_n$ be an enumeration of the elements of $\mathbb{F}_q^*$. Show that $\prod_{i \neq j}(a_j - a_i) = -1/a_j$ for all $j$.

8.1.7 Consider $\alpha \in \mathbb{F}_8$ with $\alpha^3 = 1 + \alpha$. Let $a = (a_1, \ldots, a_7)$ with $a_i = \alpha^{i-1}$ for $1 \leq i \leq 7$. Let $b = (1, \alpha^2, \alpha^4, \alpha^2, 1, 1, \alpha^4)$. Find $c$ such that the dual of $\text{GRS}_4(a, b)$ is equal to $\text{GRS}_{7-k}(a, c)$ for all $k$.

8.1.8 Determine all values of $n$, $k$ and $b$ such that $\text{RS}_k(n, b)$ is self dual.
8.1.9 Give a proof of Corollary 8.1.18.

8.1.10 Let \( n \leq q \). Let \( a \) be an \( n \)-tuple of mutually distinct elements of \( \mathbb{F}_q \), and \( r \) an \( n \)-tuple of nonzero elements of \( \mathbb{F}_q \). Let \( k \) be an integer such that \( 0 \leq k \leq n \). Show that the generalized Cauchy code \( C_k(a, r) \) is equal to \( r \ast C_k(a) \).

8.1.11 Give a proof of statements made in Remark 8.1.26.

8.1.12 Let \( u, v \) and \( w \) be three mutually distinct elements of a field \( \mathbb{F} \). Show that there is a unique fractional transformation \( \varphi \) such that \( \varphi(u) = 0 \), \( \varphi(v) = 1 \) and \( \varphi(w) = \infty \).

8.1.13 Give a proof of Proposition 8.1.28.

8.1.14 Let \( \alpha \in \mathbb{F}_8 \) be a primitive element such that \( \alpha^3 = \alpha + 1 \). Let \( G \) be the generator matrix a generalized Reed-Solomon code given by

\[
G = \begin{pmatrix}
\alpha^6 & \alpha^6 & \alpha & 1 & \alpha^4 & 1 & \alpha^4 \\
0 & \alpha^3 & \alpha^3 & \alpha^4 & \alpha^6 & \alpha^6 & \alpha^4 \\
\alpha^4 & \alpha^5 & \alpha^3 & 1 & \alpha^2 & 0 & \alpha^6
\end{pmatrix}.
\]

(1) Find \( a \) in \( \mathbb{F}_7^8 \) consisting of mutually distinct entries and \( b \) in \( \mathbb{F}_7^{18} \) with nonzero entries such that \( G \) is a generator matrix of \( GRS_3(a, b) \).

(2) Consider the \( 3 \times 7 \) generator matrix \( G' \) of the code \( RS_3(7,1) \) with entry \( \alpha^{(i-1)(j-1)} \) in the \( i \)-th row and the \( j \)-th column. Give an invertible \( 3 \times 3 \) matrix \( S \) and a permutation matrix \( P \) such that \( G' = SGP \).

(3) What is the number of pairs \((S, P)\) of such matrices?

8.2 Subfield and trace codes

8.2.1 Restriction and extension by scalars

In this section we derive bounds on the parameters of subfield subcodes. We repeat Definitions 4.4.32 and 7.3.1.

**Definition 8.2.1** Let \( D \) be an \( \mathbb{F}_q \)-linear code in \( \mathbb{F}_q^n \). Let \( C \) be an \( \mathbb{F}_{q^m} \)-linear code of length \( n \). If \( D = C \cap \mathbb{F}_q^n \), then \( D \) is called the **subfield subcode** or the **restriction (by scalars)** of \( C \), and is denoted by \( C|\mathbb{F}_q \). If \( D \subseteq C \), then \( C \) is called a **super code** of \( D \). If \( C \) is generated as an \( \mathbb{F}_{q^m} \)-linear space by \( D \), then \( C \) is called the **extension (by scalars)** of \( D \) and is denoted by \( D \otimes \mathbb{F}_{q^m} \).

**Proposition 8.2.2** Let \( G \) be a generator matrix with entries in \( \mathbb{F}_q \). Let \( D \) and \( C \) be the \( \mathbb{F}_q \)-linear and the \( \mathbb{F}_{q^m} \)-linear code, respectively with \( G \) as generator matrix. Then

\[
(D \otimes \mathbb{F}_{q^m}) = C \text{ and } (C|\mathbb{F}_q) = D.
\]
Proof. Let $G$ be a generator matrix of the $\mathbb{F}_q$-linear code $D$. Then $G$ is also a generator matrix of $D \otimes \mathbb{F}_{q^m}$ by Remark 4.4.33. Hence $(D \otimes \mathbb{F}_{q^m}) = C$.

Now $D$ is contained in $C$ and in $\mathbb{F}_{q^n}$. Hence $D \subseteq (C|\mathbb{F}_q)$. Conversely, suppose that $c \in (C|\mathbb{F}_q)$. Then $c \in \mathbb{F}_{q^n}$ and $c = xG$ for some $x \in \mathbb{F}_{q^m}$. After a permutation of the coordinates we may assume without loss of generality that $G = (I_k|A)$ for some $k \times (n-k)$ matrix $A$ with entries in $\mathbb{F}_q$. Therefore $(x, xA) = xG = c \in \mathbb{F}_{q^n}$. Hence $x \in \mathbb{F}_{q^k}$ and $c \in D$.

Remark 8.2.3 Similar statements hold as in Proposition 8.2.2 with a parity check matrix $H$ instead of a generator matrix $G$.

Remark 8.2.4 Let $D$ be a cyclic code of length $n$ over $\mathbb{F}_q$ with defining set $I$. Suppose that $\gcd(n, q) = 1$ and $n$ divide $q^m - 1$. Let $\alpha$ in $\mathbb{F}_q$ have order $n$. Let $D$ be the $\mathbb{F}_q$-linear cyclic code with parity check matrix $H = (\alpha^i|i \in I, j = 0, \ldots, n-1)$. Then $D$ is the restriction of $\tilde{D}$ by Remark 7.3.2. So $(D \otimes \mathbb{F}_{q^m}) \subseteq \tilde{D}$ and $((D \otimes \mathbb{F}_{q^m})|\mathbb{F}_q) = (\tilde{D}|\mathbb{F}_q) = D$. If $\alpha$ is not an element of $\mathbb{F}_q$, then $H$ is not defined over $\mathbb{F}_q$ and the analogous statement of Proposition 8.2.2 as mentioned in Remark 8.2.3 does hold and $(D \otimes \mathbb{F}_{q^m})$ is a proper subcode of $\tilde{D}$.

We will see that $H$ is row equivalent over $\mathbb{F}_{q^m}$ with a matrix $H$ with entries in $\mathbb{F}_q$ and $(D \otimes \mathbb{F}_{q^m}) = \tilde{D}$ if $I$ is the complete defining set of $D$.

8.2.2 Parity check matrix of a restricted code

Lemma 8.2.5 Let $h_1, \ldots, h_n \in \mathbb{F}_{q^m}$. Let $\alpha_1, \ldots, \alpha_m$ be a basis of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$. Then there exist unique elements $h_{ij} \in \mathbb{F}_q$ such that

$$ h_j = \sum_{i=1}^{m} h_{ij} \alpha_i. $$

Furthermore for all $x \in \mathbb{F}_{q^n}$

$$ \sum_{j=1}^{n} h_{ij} x_j = 0 $$

if and only if

$$ \sum_{j=1}^{n} h_{ij} x_j = 0 \text{ for all } i = 1, \ldots, m. $$

Proof. The existence and uniqueness of the $h_{ij}$ is a consequence of the assumption that $\alpha_1, \ldots, \alpha_m$ is a basis of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$. Let $x \in \mathbb{F}_{q^n}$. Then

$$ \sum_{j=1}^{n} h_{ij} x_j = \sum_{j=1}^{n} \left( \sum_{i=1}^{m} h_{ij} \alpha_i \right) x_j = \sum_{i=1}^{m} \left( \sum_{j=1}^{n} h_{ij} x_j \right) \alpha_i. $$

The $\alpha_i$ form a basis over $\mathbb{F}_q$ and the $x_j$ are elements of $\mathbb{F}_q$. This implies the statement on the equivalence of the linear equations.

Proposition 8.2.6 Let $E = (h_1, \ldots, h_n)$ be a $1 \times n$ parity check matrix of the $\mathbb{F}_{q^m}$-linear code $C$. Let $l$ be the dimension of the $\mathbb{F}_q$ linear subspace in $\mathbb{F}_{q^m}$ generated by $h_1, \ldots, h_n$. Then the dimension of $C|\mathbb{F}_q$ is equal to $n-l$. 

\end{document}
8.2. SUBFIELD AND TRACE CODES

Proof. Let $H$ be the $m \times n$ matrix with entries $h_{ij}$ as given in Lemma 8.2.5. Then $(h_{1j}, \ldots, h_{mj})$ are the coordinates of $h_j$ with respect to the basis $\alpha_1, \ldots, \alpha_m$ of $\mathbb{F}_q^m$ over $\mathbb{F}_q$. So the rank of $H$ is equal to $l$. The code $C|\mathbb{F}_q$ is the null space of the matrix $H$, by Lemma 8.2.5 and has dimension $n - \text{rank}(H)$ which is $n - l$.

Example 8.2.7 Let $\alpha \in \mathbb{F}_q$ be a primitive element such that $\alpha^2 + \alpha - 1 = 0$. Choose $\alpha_i = \alpha^i$ with $1, \alpha$ as basis. Consider the parity check matrix $E = \begin{pmatrix} 1 & \alpha & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 & \alpha^7 \end{pmatrix}$ of the $\mathbb{F}_q$-linear code $C$. Then according to Lemma 8.2.5 the parity check matrix $H$ of $C|\mathbb{F}_3$ is given by

$$H = \begin{pmatrix} 1 & 0 & 1 & 2 & 2 & 0 & 2 & 1 \\ 0 & 1 & 2 & 2 & 0 & 2 & 1 & 1 \end{pmatrix}$$

For instance $\alpha^3 = -1 - \alpha$, so $\alpha^3$ has coordinates $(-1, -1)$ with respect to the chosen basis and the transpose of this vector is the 4-th column of $H$. The entries of the row $E$ generate $\mathbb{F}_9$ over $\mathbb{F}_3$. The rank of $H$ is 2, so the dimension of $C|\mathbb{F}_3$ is 6. This is in agreement with Proposition 8.2.6.

Lemma 8.2.5 has the following consequence.

Proposition 8.2.8 Let $D$ be an $\mathbb{F}_q$-linear code of length $n$ and dimension $k$. Let $m = n - k$. If $k < n$, then $D$ is the restriction of a code $C$ over $\mathbb{F}_q^m$ of codimension one.

Proof. Let $H$ be an $(n-k) \times n$ parity check matrix of $D$ over $\mathbb{F}_q$. Let $m = n-k$. Let $k < n$. Then $m > 0$. Let $\alpha_1, \ldots, \alpha_m$ be a basis of $\mathbb{F}_q^m$ over $\mathbb{F}_q$. Define for $j = 1, \ldots, n$

$$h_j = \sum_{i=1}^{m} h_{ij} \alpha_i.$$ 

Let $E = (h_1, \ldots, h_n)$ be an $1 \times n$ parity check matrix of the $\mathbb{F}_q^m$-linear code $C$. Now $E$ is not the zero vector, since $k < n$. So $C$ has codimension one, and $D$ is the restriction of $C$ by Lemma 8.2.5.

Proposition 8.2.9 Let $C$ be an $\mathbb{F}_q^m$ linear code with parameters $[n, k, d]_{\mathbb{F}_q^m}$. Then the dimension of $C|\mathbb{F}_q$ over $\mathbb{F}_q$ is at least $n - m(n-k)$ and its minimum distance is at least $d$.

Proof. The minimum distance of $C|\mathbb{F}_q$ is at least the minimum distance of $C$, since $C|\mathbb{F}_q$ is a subset of $C$. Let $E$ be a parity check matrix of $C$. Then $E$ consists of $n-k$ rows. Every row gives rise to $m$ linear equations over $\mathbb{F}_q$ by Lemma 8.2.5. So $C|\mathbb{F}_q$ is the solution space of $m(n-k)$ homogeneous linear equations over $\mathbb{F}_q$. Therefore the dimension of $C|\mathbb{F}_q$ is at least $n - m(n-k)$.

Remark 8.2.10 ***Lower bound of Delsarte-Sidelnikov***
8.2.3 Invariant subspaces

Remark 8.2.11 Let $D$ be the restriction of an $\mathbb{F}_q$-linear code $C$. Suppose that $\mathbf{h} = (h_1, \ldots, h_n) \in \mathbb{F}_q^n$ is a parity check for $D$. So

$$h_1c_1 + \cdots + h_nc_n = 0 \text{ for all } \mathbf{c} \in D.$$ 

Then

$$\sum_{i=1}^n h_i^qc_i = \sum_{i=1}^n h_i^qc_i = (\sum_{i=1}^n h_ic_i)^q = 0$$

for all $\mathbf{c} \in D$, since $c_i^q = c_i$ for all $i$ and $\mathbf{c} \in D$. Hence $(h_1^q, \ldots, h_n^q)$ is also a parity check for the code $D$.

Example 8.2.12 This is a continuation of Example 8.2.7. Consider the parity check matrix

$$E' = \begin{pmatrix} 1 & \alpha & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 & \alpha^7 \\ 1 & \alpha^3 & \alpha^6 & \alpha^4 & \alpha^5 & \alpha^2 & \alpha^7 & \alpha^6 \\ \end{pmatrix}$$

of the $\mathbb{F}_9$-linear code $C'$. Let $D'$ be the ternary restriction of $C'$. Then according to Proposition 8.2.6 the code $D'$ is the null space of the matrix $H'$ given by

$$H' = \begin{pmatrix} 1 & 0 & 1 & 2 & 2 & 0 & 2 & 1 \\ 0 & 1 & 2 & 2 & 0 & 2 & 1 & 1 \\ 1 & 2 & 2 & 0 & 2 & 1 & 1 & 0 \\ 0 & 2 & 1 & 1 & 0 & 1 & 2 & 2 \\ \end{pmatrix}$$

The second row of $E'$ is obtained by taking the third power of the entries of the first row. So $D' = D$ by Remark 8.2.11. Indeed, the last two rows of $H'$ are linear combinations of the first two rows. Hence $H'$ and $H$ have the same rank, that is 2.

Definition 8.2.13 Extend the Frobenius map $\varphi : \mathbb{F}_q^n \to \mathbb{F}_q^n$, defined by $\varphi(x) = x^q$, to the map $\varphi : \mathbb{F}_q^n \to \mathbb{F}_q^n$, defined by $\varphi(x) = (x_1^q, \ldots, x_n^q)$. Likewise we define $\varphi(G)$ of a matrix with entries $(g_{ij})$ to be the matrix with entries $(\varphi(g_{ij}))$.

Remark 8.2.14 The map $\varphi : \mathbb{F}_q^n \to \mathbb{F}_q^n$, has the property that

$$\varphi(\alpha \mathbf{x} + \beta \mathbf{y}) = \alpha^q \varphi(\mathbf{x}) + \beta^q \varphi(\mathbf{y})$$

for all $\alpha, \beta \in \mathbb{F}_q$ and $\mathbf{x}, \mathbf{y} \in \mathbb{F}_q^n$. Hence this map is $\mathbb{F}_q$-linear, since $\alpha^q = \alpha$ and $\beta^q = \beta$ for $\alpha, \beta \in \mathbb{F}_q$. In particular, the Frobenius map is an automorphism of the field $\mathbb{F}_q$ with $\mathbb{F}_q$ as the field of elements that are point-wise fixed. Therefore it leaves also the points of $\mathbb{F}_q^n$ point-wise fixed under $\varphi$. If $\mathbf{x} \in \mathbb{F}_q^n$, then

$$\varphi(\mathbf{x}) = \mathbf{x} \text{ if and only if } \mathbf{x} \in \mathbb{F}_q^n.$$ 

Furthermore $\varphi$ is an isometry.

Definition 8.2.15 Let $\mathbb{F}$ be a subfield of $G$. The Galois group $\text{Gal}(G/\mathbb{F})$ is the group all field automorphisms of $G$ that leave $\mathbb{F}$ point-wise fixed. $\text{Gal}(G/\mathbb{F})$ is denoted by $\text{Gal}(q^n, q)$ in case $\mathbb{F} = \mathbb{F}_q$ and $G = \mathbb{F}_q^n$. A subspace $W$ of $\mathbb{F}_q^n$ is called $\text{Gal}(q^n, q)$-invariant, or just invariant, if $\tau(W) = W$ for all $\tau \in \text{Gal}(q^n, q)$. 


Remark 8.2.16 \( \text{Gal}(q^m, q) \) is a cyclic group generated by \( \varphi \) of order \( m \). Hence a subspace \( W \) is invariant if and only if \( \varphi(W) \subseteq W \).

The following two lemmas are similar to the statements for the shift operator in connection with cyclic codes in Propositions 7.1.3 and 7.1.6 but now for the Frobenius map.

Lemma 8.2.17 Let \( G \) be \( k \times n \) generator matrix of the \( \mathbb{F}_{q^m} \)-linear code \( C \). Let \( g_i \) be the \( i \)-th row of \( G \). Then \( C \) is \( \text{Gal}(q^m, q) \)-invariant if and only if \( \varphi(g_i) \in C \) for all \( i = 1, \ldots, k \).

Proof. If \( C \) is invariant, then \( \varphi(g_i) \in C \) for all \( i \), since \( g_i \in C \). Conversely, suppose that \( \varphi(g_i) \in C \) for all \( i \). Let \( c \in C \). Then \( c = \sum_{i=1}^{k} x_i g_i \) for some \( x_i \in \mathbb{F}_{q^m} \). So

\[
\varphi(c) = \sum_{i=1}^{k} x_i^q \varphi(g_i) \in C.
\]

Hence \( C \) is an invariant code.

Lemma 8.2.18 Let \( C \) be an \( \mathbb{F}_{q^m} \)-linear code. Then \( C^\perp \) is invariant if \( C \) is invariant.

Proof. Notice that

\[
\varphi(x \cdot y) = (\sum_{i=1}^{n} x_i y_i)^q = \sum_{i=1}^{n} x_i^q y_i^q = \varphi(x) \cdot \varphi(y)
\]

for all \( x, y \in \mathbb{F}_{q^m} \). Suppose that \( C \) is an invariant code. Let \( y \in C^\perp \) and \( c \in C \). Then \( \varphi^{m-1}(c) \in C \). Hence

\[
\varphi(y) \cdot c = \varphi(y) \cdot \varphi^{m}(c) = \varphi(y \cdot \varphi^{m-1}(c)) = \varphi(0) = 0,
\]

Therefore \( \varphi(y) \in C^\perp \) for all \( y \in C^\perp \), and \( C^\perp \) is invariant.

Proposition 8.2.19 Let \( C \) be \( \mathbb{F}_{q^m} \)-linear code of length \( n \). Then \( C \) is \( \text{Gal}(q^m, q) \)-invariant if and only if \( C \) has a generator matrix with entries in \( \mathbb{F}_q \) if and only if \( C \) has a parity check matrix with entries in \( \mathbb{F}_q \).

Proof. If \( C \) has a generator matrix with entries in \( \mathbb{F}_q \), then clearly \( C \) is invariant.

Now conversely, suppose that \( C \) is invariant. Let \( G \) be a \( k \times n \) generator matrix of \( C \). We may assume without loss of generality that the first \( k \) columns are independent. So after applying the Gauss algorithm we get the row reduced echelon form \( G' \) of \( G \) with the \( k \times k \) identity matrix \( I_k \) in the first \( k \) columns. So \( G' = (I_k | A) \), where \( A \) is a \( k \times (n-k) \) matrix. Let \( g'_i \) be the \( i \)-th row of \( G' \). Now \( C \) is invariant. So \( \varphi(g'_i) \in C \) and \( \varphi(g'_i) \) is an \( \mathbb{F}_{q^m} \)-linear combination of the \( g'_j \). That is one can find elements \( s_{ij} \) in \( \mathbb{F}_{q^m} \) such that

\[
\varphi(g'_i) = \sum_{j=1}^{n} s_{ij} g'_j.
\]
Let $S$ be the $k \times k$ matrix with entries $(s_{ij})$. Then

$$(I_k|\varphi(A)) = (\varphi(I_k)|\varphi(A)) = \varphi(G') = SG' = S(I_k|A) = (S|SA).$$

Therefore $I_k = S$ and $\varphi(A) = SA = A$. Hence the entries of $A$ are elements of $F_q$. So $G'$ is a generator matrix of $C$ with entries in $F_q$. The last equivalence is a consequence of Proposition 2.3.3.

**Example 8.2.20** Let $\alpha \in F_8$ be a primitive element such that $\alpha^3 = \alpha + 1$. Let $G$ be the generator matrix of the $F_8$-linear code $C$ with

$$G = \begin{pmatrix}
1 & \alpha & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \\
1 & \alpha^2 & \alpha^4 & \alpha^6 & \alpha & \alpha^3 & \alpha^5 \\
1 & \alpha^4 & \alpha^5 & \alpha^2 & \alpha^6 & \alpha^3 & \alpha
\end{pmatrix}$$

Let $g_i$ be the $i$-th row of $G$. Then $\varphi(g_i) = g_{i+1}$ for all $i = 1, 2$ and $\varphi(g_3) = g_1$. Hence $C$ is an invariant code by Lemma 8.2.17. The proof of Proposition 8.2.19 explains how to get a generator matrix $G'$ with entries in $F_2$. Let $G'$ be the row reduced echelon form of $G$. Then

$$G' = \begin{pmatrix}
1 & 0 & 0 & 1 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & 1 & 1
\end{pmatrix}$$

is indeed a binary matrix. In fact it is the generator matrix of the binary $[7, 3, 4]$ Hamming code.

**Definition 8.2.21** Let $C$ be an $F_{q^m}$-linear code. Define the codes $C^0$ and $C^*$ by

$$C^0 = \cap_{i=1}^m \varphi^i(C),$$

$$C^* = \sum_{i=1}^m \varphi^i(C).$$

**Remark 8.2.22** It is clear from the definitions that the codes $C^0$ and $C^*$ are $\text{Gal}(q^m, q)$-invariant. Furthermore $C^0$ is the largest invariant code contained in $C$, that is if $D$ is an invariant code and $D \subseteq C$, then $D \subseteq C^0$. And similarly, $C^*$ is the smallest invariant code containing $C$, that is if $D$ is an invariant code and $C \subseteq D$, then $C^* \subseteq D$.

**Proposition 8.2.23** Let $C$ be an $F_{q^m}$-linear code. Then

$$C^0 = ((C^\perp)^*)^\perp$$

**Proof.** The following inclusion holds $C^0 \subseteq C$. So dually $C^\perp \subseteq (C^0)^\perp$. Now $C^0$ is invariant. So $(C^0)^\perp$ is invariant by Lemma 8.2.18, and it contains $C^\perp$. By Remark 8.2.22 $(C^\perp)^*$ is the smallest invariant code containing $C^\perp$. Hence $(C^\perp)^* \subseteq (C^0)^\perp$ and therefore

$$C^0 \subseteq ((C^\perp)^*)^\perp.$$ 

We have $C^\perp \subseteq (C^\perp)^*$. So dually $(C^\perp)^* \subseteq C$. The code $((C^\perp)^*)^\perp$ is invariant and is contained in $C$. The largest code that is invariant and contained in $C$ is equal to $C^0$. Hence

$$((C^\perp)^*)^\perp \subseteq C^0.$$ 

Both inclusions give the desired equality. 

\[\square\]
Theorem 8.2.24 Let $C$ be an $\mathbb{F}_{q^m}$-linear code. Then $C$ and $C^0$ have the same restriction. Furthermore
\[
\dim_{\mathbb{F}_q}(C|\mathbb{F}_q) = \dim_{\mathbb{F}_{q^m}}(C^0) \quad \text{and} \quad d(C|\mathbb{F}_q) = d(C^0).
\]

Proof. The inclusion $C^0 \subseteq C$ implies $(C^0|\mathbb{F}_q) \subseteq (C|\mathbb{F}_q)$. The code $(C|\mathbb{F}_q) \otimes \mathbb{F}_{q^m}$ is contained in $C$ and is invariant. Hence
\[
(C|\mathbb{F}_q) \otimes \mathbb{F}_{q^m} \subseteq C^0,
\]
by Remark 8.2.22. So $(((C|\mathbb{F}_q) \otimes \mathbb{F}_{q^m})|\mathbb{F}_q) \subseteq (C^0|\mathbb{F}_q)$. But
\[
(C|\mathbb{F}_q) = (((C|\mathbb{F}_q) \otimes \mathbb{F}_{q^m})|\mathbb{F}_q),
\]
by Lemma 8.2.2 applied to $D = (C|\mathbb{F}_q)$. Therefore $((C|\mathbb{F}_q) \subseteq (C^0|\mathbb{F}_q)$, and with the converse inclusion above we get the desired equality $(C^0|\mathbb{F}_q) = (C|\mathbb{F}_q)$. The code $C^0$ has a $k \times n$ generator matrix $G$ with entries in $\mathbb{F}_q$, by Proposition 8.2.19, since $C^0$ is an invariant code. Then $G$ is also a generator matrix of $(C^0|\mathbb{F}_q)$, by Lemma 8.2.2. Furthermore $(C|\mathbb{F}_q) = (C^0|\mathbb{F}_q)$. Therefore
\[
\dim_{\mathbb{F}_q}(C|\mathbb{F}_q) = k = \dim_{\mathbb{F}_{q^m}}(C^0).
\]
The code $C^0$ has a parity check $H$ with entries in $\mathbb{F}_q$, by Proposition 8.2.19. Then $H$ is also a parity check matrix of $(C|\mathbb{F}_q)$ over $\mathbb{F}_q$. The minimum distance of a code can be expressed as the minimum number of columns in a parity check matrix that are dependent, by Proposition 2.3.11. Consider a $l \times m$ matrix $B$ with entries in $\mathbb{F}_q$. Then the the columns of $B$ are dependent if and only if rank($B$) < $m$. The rank $B$ is equal to the number of pivots in the row reduced echelon form of $B$. The row reduced echelon form of $B$ is unique, by Remark 2.2.18, and does not change by considering $B$ as a matrix with entries over $\mathbb{F}_{q^m}$. Therefore $d(C|\mathbb{F}_q) = d(C^0)$.

Remark 8.2.25 Lemma 8.2.5 gives us a method to compute the parity check matrix of the restriction. Proposition 8.2.23 and Theorem 8.2.24 give us another way to compute the parity check and generator matrix of the restriction of a code. Let $C$ be an $\mathbb{F}_{q^m}$-linear code. Let $H$ be a parity check matrix of $C$. Then $H$ is a generator matrix of $C^\perp$. Let $(\mathbf{h}_i, i = 1, \ldots, n-k)$ be the rows of $H$. Let $H^*$ be the matrix with the $(n-k)m$ rows $\varphi^i(\mathbf{h}_i)$, $i = 1, \ldots, n-k, j = 1, \ldots, m$. Then these rows generate $(C^\perp)^*$. Let $H_0$ be the row reduced echelon form of $H^*$ with the zero rows deleted. Then $H_0$ has entries in $\mathbb{F}_q$ and is a generator matrix of $(C^\perp)^*$, since it is an invariant code. So $H_0$ is the parity check matrix of $(C^\perp)^* = C^0$. Hence it is also the parity check matrix of $(C^0|\mathbb{F}_q) = (C|\mathbb{F}_q)$.

Example 8.2.26 Consider the parity check matrix $E$ of Example 8.2.7. Then $E^*$ is equal to the matrix $E'$ of Example 8.2.12. Taking the row reduced echelon form of $E^*$ gives indeed the parity check matrix $H$ obtained in Example 8.2.7.

8.2.4 Cyclic codes as subfield subcodes

***
CHAPTER 8. POLYNOMIAL CODES

8.2.5 Trace codes

Definition 8.2.27 The trace map $\text{Tr}_{\mathbb{F}_{q^m}} : \mathbb{F}_{q^m} \rightarrow \mathbb{F}_q$ is defined by
$$\text{Tr}_{\mathbb{F}_{q^m}}(x) = x + x^q + \cdots + x^{q^{m-1}} \quad \text{for} \quad x \in \mathbb{F}_{q^m}.$$ 

The notation $\text{Tr}_{\mathbb{F}_{q^m}}$ is abbreviated by $\text{Tr}$ in case the context is clear. This map is extended coordinatewise to a map $\text{Tr} : \mathbb{F}^n_{q^m} \rightarrow \mathbb{F}^n_q$.

Remark 8.2.28 Let $\mathbb{F}$ be a field and $\mathbb{G}$ a finite field extension of $\mathbb{F}$ of degree $m$. Then $\mathbb{G}$ is a vector space over $\mathbb{F}$ of dimension $m$. Choose a basis of $\mathbb{G}$ over $\mathbb{F}$. Let $x \in \mathbb{G}$. Then multiplication by $x$ on $\mathbb{G}$ is an $\mathbb{F}$-linear map. Let $M_x$ be the corresponding matrix of this map with respect to the chosen basis. The sum of the diagonal elements of $M_x$ is called the trace of $x$. This trace does not depend on the chosen basis and will be denoted by $\text{Tr}_{\mathbb{G}}(x)$ or by $\text{Tr}(x)$ for short.

Definition 8.2.27 of the trace for a finite extension of a finite field is an ad hoc definition. With the above generalization of the definition of the trace the ad hoc definition becomes a property.

The maps $\text{Tr} : \mathbb{F}_{q^m} \rightarrow \mathbb{F}_q$ and $\text{Tr} : \mathbb{F}^n_{q^m} \rightarrow \mathbb{F}^n_q$ are $\mathbb{F}_q$-linear.

Proposition 8.2.29 (Delsarte-Sidelnikov) Let $C$ be an $\mathbb{F}_{q^m}$-linear code. Then $(C^\perp \cap \mathbb{F}_q^n)^\perp = \text{Tr}(C)$.

Proof. ***

8.2.6 Exercises

8.2.1 Let $\alpha \in \mathbb{F}_{16}$ be a primitive element such that $\alpha^4 = \alpha + 1$. Choose $\alpha_i = \alpha^i$ with $i = 0, 1, 2, 3$ as basis. Consider the parity check matrix
$$E = \begin{pmatrix} 1 & \alpha & \alpha^2 & \alpha^3 & \cdots & \alpha^{14} \\ 1 & \alpha^2 & \alpha^4 & \alpha^6 & \cdots & \alpha^{14} \end{pmatrix}$$

of the $\mathbb{F}_{16}$-linear code $C$. Let $E'$ be the $1 \times 8$ submatrix of $E$ consisting of the first row of $E$. Let $C'$ be the $\mathbb{F}_{16}$-linear code with $E'$ as parity check matrix. Determine the the parity check matrices $H$ of $C|\mathbb{F}_q$ and of $C'|\mathbb{F}_q$ using Lemma 8.2.5 and Proposition 8.2.9. Show that $H = H'$.

8.2.2 Let $\alpha \in \mathbb{F}_{16}$ be a primitive element such that $\alpha^4 = \alpha + 1$. Give a binary parity check matrix of the binary restriction of the code $RS_4(15, 0)$. Determine the dimension of the binary restriction of the code $RS_n(15, 0)$ for all $k$.

8.2.3 Let $\alpha \in \mathbb{F}_{16}$ be a primitive element such that $\alpha^4 = \alpha + 1$. Let $G$ be the $4 \times 15$ matrix with entry $g_{ij} = \alpha^{2i}$ at the $i$-th row and the $j$-th column. Let $C$ be the code with generator matrix $G$. Show that $C$ is $\text{Gal}(16, 2)$-invariant and give a binary generator matrix of $C$.

8.2.4 Let $m$ be a positive integer and $C$ an $\mathbb{F}_{q^m}$ linear code. Let $\varphi$ be the Frobenius map of $\mathbb{F}_{q^m}$ fixing $\mathbb{F}_q$. Show that $\varphi(C)$ is an $\mathbb{F}_{q^m}$ linear code that is isometric with $C$. Give a counter example of a code $C$ that is not monomial equivalent with $\varphi(C)$.

8.2.5 Give proofs of the statements made in Remark 8.2.28.
8.3 Some families of polynomial codes

8.3.1 Alternant codes

Definition 8.3.1 Let \( a = (a_1, \ldots, a_n) \) be an \( n \)-tuple of \( n \) distinct elements of \( \mathbb{F}_{q^m} \). Let \( b = (b_1, \ldots, b_n) \) be an \( n \)-tuple of nonzero elements of \( \mathbb{F}_{q^m} \). Let \( \text{GRS}_k(a, b) \) be the generalized RS code over \( \mathbb{F}_{q^m} \) of dimension \( k \). The alternant code \( \text{ALT}_r(a, b) \) is the \( \mathbb{F}_q \)-linear restriction of \( (\text{GRS}_r(a, b))^\perp \).

Proposition 8.3.2 The code \( \text{ALT}_r(a, b) \) has parameters \([n, k, d]\) with \( k \geq n - mr \) and \( d \geq r + 1 \).

Proof. The code \( (\text{GRS}_r(a, b))^\perp \) is equal to \( \text{GRS}_{n-r}(a, c) \) by Proposition 8.1.21 with \( c_j = 1/(b_j \prod_{i \neq j} (a_j - a_i)) \) by Proposition 8.1.21, and has parameters \([n, n-r, r+1]\) by Proposition 8.1.14. So the statement is a consequence of Proposition 8.2.9.

Proposition 8.3.3 \( (\text{ALT}_r(a, b))^\perp = \text{Tr}(\text{GRS}_r(a, b)) \).

Proof. This is a direct consequence of the definition of an alternant code and Proposition 8.2.29.

Proposition 8.3.4 Every linear code of minimum distance at least 2 is an alternant code.

Proof. Let \( C \) be a code of length \( n \) and dimension \( k \). Then \( k < n \), since the minimum distance of \( C \) is at least 2. Let \( m \) be a positive integer such that \( n - k \) divides \( m \) and \( q^m \geq n \). Let \( a = (a_1, \ldots, a_n) \) be any \( n \)-tuple of \( n \) distinct elements of \( \mathbb{F}_{q^m} \). Let \( H \) be an \((n-k) \times n\) parity check matrix of \( C \) over \( \mathbb{F}_q \). Following the proof of Proposition 8.2.8, let \( \alpha_1, \ldots, \alpha_{n-k} \) be a basis of \( \mathbb{F}_{q^m} \) over \( \mathbb{F}_q \). The field \( \mathbb{F}_{q^m} \) is an extension of \( \mathbb{F}_{q^{n-k}} \), since \( n - k \) divides \( m \). Define \( b_j = \sum_{i=1}^{m} h_{ij} \alpha_i \) for \( j = 1, \ldots, n \). The minimum distance of \( C \) is at least 2. So \( H \) does not contain a zero column by Proposition 2.3.11. Hence \( b_j \neq 0 \) for all \( j \). Let \( b = (b_1, \ldots, b_n) \). Then \( C \) is the restriction of \( \text{GRS}_1(a, b)^\perp \). Therefore \( C = \text{ALT}_1(a, c) \) by definition.

Remark 8.3.5 The above proposition gives that almost all linear codes are alternant, but it gives no useful information about the parameters of the code.

***Alternant codes meet the GV bound (MacWilliams & Sloane page 337)
BCH codes are not asymptotically good?? ***
8.3.2 Goppa codes

A special class of alternant codes is given by Goppa codes.

Definition 8.3.6 Let \( L = (a_1, \ldots, a_n) \) be an \( n \)-tuple of \( n \) distinct elements of \( \mathbb{F}_{q^m} \). A polynomial \( g \) with coefficients in \( \mathbb{F}_{q^m} \) such that \( g(a_j) \neq 0 \) for all \( j \) is called a Goppa polynomial with respect to \( L \). Define the \( \mathbb{F}_q \)-linear Goppa code \( \Gamma(L, g) \) by

\[
\Gamma(L, g) = \left\{ \mathbf{c} \in \mathbb{F}_q^n \mid \sum_{j=1}^n \frac{c_j}{X - a_j} \equiv 0 \mod g(X) \right\}.
\]

Remark 8.3.7 The assumption \( g(a_j) \neq 0 \) implies that \( X - a_j \) and \( g(X) \) are relatively prime, so their greatest common divisor is 1. Euclid’s algorithm gives polynomials \( P_j \) and \( Q_j \) such that \( P_j(X)g(X) + Q_j(X)(X - a_j) = 1 \). So \( Q_j(X) \) is the inverse of \( X - a_j \) modulo \( g(X) \). We claim that

\[
Q_j(X) = g(X) - g(a_j)X^{-1}g(a_j)^{-1}.
\]

Notice that \( g(X) - g(a_j) \) has \( a_j \) as zero. So \( g(X) - g(a_j) \) is divisible by \( X - a_j \) and its fraction is a polynomial of degree one less than the degree of \( g(X) \). With the above definition of \( Q_j \) we get

\[
Q_j(X)(X - a_j) = (g(X) - g(a_j))g(a_j)^{-1} = 1 - g(X)g(a_j)^{-1} \equiv 1 \mod g(X).
\]

Remark 8.3.8 Let \( g_1 \) and \( g_2 \) be two Goppa polynomials with respect to \( L \). If \( g_2 \) divides \( g_1 \), then \( \Gamma(L, g_1) \) is a subcode of \( \Gamma(L, g_2) \).

Proposition 8.3.9 Let \( L = \mathbf{a} = (a_1, \ldots, a_n) \). Let \( g \) be a Goppa polynomial of degree \( r \). The Goppa code \( \Gamma(L, g) \) is equal to the alternant code \( \text{ALT}_r(\mathbf{a}, \mathbf{b}) \) where \( b_j = 1/g(a_j) \).

Proof. Remark 8.3.7 implies that \( \mathbf{c} \in \Gamma(L, g) \) if and only if

\[
\sum_{j=1}^n c_j \frac{g(X) - g(a_j)}{X - a_j} g(a_j)^{-1} = 0,
\]

since the left hand side is a polynomial of degree strictly smaller than the degree of \( g(X) \), and this polynomial is 0 if and only if it is 0 modulo \( g(X) \). Let \( g(X) = g_0 + g_1X + \cdots + g_rX^r \). Then

\[
\frac{g(X) - g(a_j)}{X - a_j} = \sum_{i=0}^r g_i \frac{X^i - a_j^i}{X - a_j} = \sum_{i=0}^r g_i \sum_{l=0}^{i-1} X^i - a_j^{i-1} - X^i \sum_{l=0}^{i-1} a_j^{l-1} - X^i \sum_{l=0}^{i-1} a_j^{l-1}.
\]

Therefore \( \mathbf{c} \in \Gamma(L, g) \) if and only if

\[
\sum_{j=1}^n \left( \sum_{l=i+1}^r g_2 a_j^{l-1} \right) g(a_j)^{-1} c_j = 0
\]
for all \( i = 0, \ldots, r - 1 \), if and only if \( H_1 c^T = 0 \), where \( H_1 \) is a \( r \times n \) parity check matrix with \( j \)-th column

\[
\begin{pmatrix}
g_r a_j^{r-1} + g_{r-1} a_j^{r-2} + \cdots + g_2 a_j + g_1 \\
g_r a_j^2 + g_{r-1} a_j + g_{r-2} \\
g_r a_j + g_{r-1} \\
g_r
\end{pmatrix}
\]

\( g(a_j)^{-1} \)

The coefficient \( g_r \) is not zero, since \( g(X) \) has degree \( r \). Divide the last row of \( H_1 \) by \( g_r \). Then subtract \( g_{r-1} \) times the \( r \)-th row from row \( r - 1 \). Next divide row \( r - 1 \) by \( g_r \). Continuing in this way by a sequence of elementary transformations it is shown that \( H_1 \) is row equivalent with the matrix \( H_2 \) with entry \( a_i^{j-1} g(a_j)^{-1} \) in the \( i \)-th row and the \( j \)-th column. So \( H_2 \) is the generator matrix of \( GRS_r(a, b) \), where \( b = (b_1, \ldots, b_n) \) and \( b_j = 1/g(a_j) \). Hence \( \Gamma(L, g) \) is the restriction of \( GRS_r(a, b)^\perp \). Therefore \( \Gamma(L, g) = ALT_r(a, b) \) by definition.

\[\Box\]

**Proposition 8.3.10** Let \( g \) be a Goppa polynomial of degree \( r \) over \( \mathbb{F}_{q^m} \). Then the Goppa code \( \Gamma(L, g) \) is an \([n, k, d]\) code with

\[
k \geq n - mr \quad \text{and} \quad d \geq r + 1.
\]

**Proof.** This is a consequence of Proposition 8.3.9 showing that a Goppa code is an alternant code and Proposition 8.3.2 on the parameters of alternant codes.

\[\Box\]

**Remark 8.3.11** Let \( g \) be a Goppa polynomial of degree \( r \) over \( \mathbb{F}_{q^m} \). Then the Goppa code \( \Gamma(L, g) \) has minimum distance \( d \geq r + 1 \) by Proposition 8.3.10. It is an alternant code, that is a subfield subcode of a GRS code of minimum distance \( r+1 \) by Proposition 8.3.9. This super code has several efficient decoding algorithms that correct \( \lfloor r/2 \rfloor \) errors. The same algorithms can be applied to the Goppa code to correct \( \lfloor r/2 \rfloor \) errors.

**Definition 8.3.12** A polynomial is called **square free** if all (irreducible) factors have multiplicity one.

**Remark 8.3.13** Notice that irreducible polynomials are square free Goppa polynomials. If \( g(X) \) is a square free Goppa polynomial, then \( g(X) \) and its formal derivative \( g'(X) \) have no common factor by Lemma 7.2.8.

**Proposition 8.3.14** Let \( g \) be a square free Goppa polynomial with coefficients in \( \mathbb{F}_{2^m} \). Then the binary Goppa code \( \Gamma(L, g) \) is equal to \( \Gamma(L, g^2) \).

**Proof.** (1) The code \( \Gamma(L, g^2) \) is a subcode of \( \Gamma(L, g) \), by Remark 8.3.8.

(2) Let \( c \) be a binary word. Define the polynomial \( f(X) \) by

\[
f(X) = \prod_{j=1}^{n} (X - a_j)^{c_j}
\]
So \( f(X) \) is the reciprocal locator polynomial of \( c \), it is the monic polynomial of degree \( \text{wt}(c) \) and its zeros are located at those \( a_j \) such that \( c_j \neq 0 \). Now

\[
f'(X) = \sum_{j=1}^{n} c_j (X - a_j)^{-1} \prod_{l=1, l \neq j}^{n} (X - a_l)^{c_l}.
\]

Hence

\[
\frac{f''(X)}{f(X)} = \sum_{j=1}^{n} \frac{c_j}{X - a_j}
\]

Let \( c \in \Gamma(L, g) \). Then \( f'(X)/f(X) \equiv 0 \mod g(X) \). Now \( \gcd(f(X), g(X)) = 1 \). So there exist polynomials \( p(X) \) and \( q(X) \) such that \( p(X)f(X) + q(X)g(X) = 1 \).

Hence

\[
p(X)f'(X) \equiv \frac{f'(X)}{f(X)} \equiv 0 \mod g(X).
\]

Therefore \( g(X) \) divides \( f'(X) \), since \( \gcd(p(X), g(X)) = 1 \).

Let \( f(X) = f_0 + f_1 X + \cdots + f_n X^n \). Then

\[
f'(X) = \sum_{i=0}^{n} i f_i X^{i-1} = \sum_{i=0}^{\lfloor n/2 \rfloor} f_{2i+1} X^{2i} = \left( \sum_{i=0}^{\lfloor n/2 \rfloor} f_{2i+1}^2 X^{i} \right)^2,
\]

since the coefficients are in \( \mathbb{F}_{2^m} \). So \( f'(X) \) is a square that is divisible by the square free polynomial \( g(X) \). Hence \( f'(X) \) is divisible by \( g(X)^2 \), so \( c \in \Gamma(L, g^2) \).

Therefore \( \Gamma(L, g) \) is contained in \( \Gamma(L, g^2) \). So they are equal by (1).

**Proposition 8.3.15** Let \( g \) be a square free Goppa polynomial of degree \( r \) with coefficients in \( \mathbb{F}_{2^m} \). Then the binary Goppa code \( \Gamma(L, g) \) is an \([n, k, d]\) code with

\[
k \geq n - mr \quad \text{and} \quad d \geq 2r + 1.
\]

**Proof.** This is a consequence of Proposition 8.3.14 showing that \( \Gamma(L, g) = \Gamma(L, g^2) \) and Proposition 8.3.10 on the parameters of Goppa codes. The lower bound on the dimension uses that \( g(X) \) has degree \( r \), and the lower bound on the minimum distance uses that \( g^2(X) \) has degree \( 2r \).

**Example 8.3.16** Let \( \alpha \in \mathbb{F}_8 \) be a primitive element such that \( \alpha^4 = \alpha + 1 \). Let \( a_j = \alpha^{j-1} \) be an enumeration of the seven elements of \( L = \mathbb{F}_8^* \). Let \( g(X) = 1 + X + X^2 \). Then \( g \) is a square free polynomial in \( \mathbb{F}_2[X] \) and a Goppa polynomial with respect to \( L \). Let \( a \) be the vector with entries \( a_j \). Let \( b \) be defined by \( b_j = 1/g(a_j) \). Then \( b = (1, \alpha^2, \alpha^4, \alpha^2, \alpha, \alpha, \alpha^4) \). And \( \Gamma(L, g) = ALT_2(a, b) \) by Proposition 8.3.9. Let \( k \) be the dimension and \( d \) the minimum distance of \( \Gamma(L, g) \). Then \( k \geq 1 \) and \( d \geq 5 \) by Proposition 8.3.15. In fact \( \Gamma(L, g) \) is a one dimensional code generated by \([0, 1, 1, 1, 1, 1, 1] \). Hence \( d = 6 \).

**Example 8.3.17** Let \( L = \mathbb{F}_{2^{10}} \). Consider the binary Goppa code \( \Gamma(L, g) \) with a Goppa polynomial \( g \) in \( \mathbb{F}_{2^{10}}[X] \) of degree 50 with respect to \( L = \mathbb{F}_{2^{10}} \). Then the code has length 1024, dimension \( k \geq 524 \) and minimum distance \( d \geq 51 \). If moreover \( g \) is square free, then \( d \geq 101 \).

***Goppa codes meet the GV bound, random argument***
8.3. SOME FAMILIES OF POLYNOMIAL CODES

8.3.3 Counting polynomials

The number of certain polynomials will be counted in order to get an idea of the number of Goppa codes.

Remark 8.3.18

(1) Irreducible polynomials are square free Goppa polynomials. The number of monic irreducible polynomials in \( \mathbb{F}_q[X] \) of degree \( d \) is counted by \( \text{Irr}_q(d) \) and this number is computed by means of the M"obius function as given by Proposition 7.2.19.

(2) Every monic square free polynomial \( f(X) \) over \( \mathbb{F}_q \) of degree \( r \) has a unique factorization in monic irreducible polynomials. Let \( e_i \) be the number of irreducible factors in \( f(X) \) of degree \( i \). Then \( e_1 + 2e_2 + \cdots + re_r = r \) and there are \( e_i \) ways to choose among the \( \text{Irr}_q(i) \) monic irreducible polynomials of degree \( i \). Hence the number \( S_q(r) \) of monic square free polynomials over \( \mathbb{F}_q \) of degree \( r \) is equal to

\[
S_q(r) = \sum_{\substack{e_1+2e_2+\cdots+re_r=r \\ \text{and} \\ e_i \text{ integers}}} \prod_{i=1}^{r} \left( \frac{\text{Irr}_q(i)}{e_i} \right).
\]

(3) The number \( S_{G_q}(r) \) of square free monic Goppa polynomials in \( \mathbb{F}_q[X] \) of degree \( r \) with respect to \( L = \mathbb{F}_q \) is similar, since such Goppa polynomials have no linear factors in \( \mathbb{F}_q[X] \). Hence

\[
S_{G_q}(r) = \sum_{\substack{e_2+\cdots+re_r=r \\ \text{and} \\ e_i \text{ integers}}} \prod_{i=2}^{r} \left( \frac{\text{Irr}_q(i)}{e_i} \right).
\]

Simpler formulas are obtained in the following.

Proposition 8.3.19

Let \( S_q(r) \) be the number of monic square free polynomials over \( \mathbb{F}_q \) of degree \( r \). Then \( S_q(0) = 1, S_q(1) = q \) and \( S_q(r) = q^r - q^{r-1} \) for \( r > 1 \).

Proof. Clearly \( S_q(0) = 1 \) and \( S_q(1) = q \). Since 1 is the only monic polynomial of degree zero, and \( \{a + X | a \in \mathbb{F}_q\} \) is the set of monic polynomials of degree one and they are all square free.

If \( f(X) \) is a monic polynomial of degree \( r > 1 \), but not square free, then we have a unique factorization

\[
f(X) = g(X)^2h(X),
\]

where \( g(X) \) is a monic polynomial, say of degree \( a \), and \( h(X) \) is a monic square free polynomial of degree \( b \). So \( 2a + b = r \) and \( a > 0 \). Hence the number of monic polynomials of degree \( r \) over \( \mathbb{F}_q \) that are not square free is \( q^r - S_q(r) \) and equal to

\[
\sum_{a=1}^{\lfloor r/2 \rfloor} q^a S_q(r - 2a).
\]

Therefore

\[
S_q(r) = q^r - \sum_{a=1}^{\lfloor r/2 \rfloor} q^a S_q(r - 2a).
\]

This recurrence relation with starting values \( S_q(0) = 1 \) and \( S_q(1) = q \) has the unique solution \( S_q(r) = q^r - q^{r-1} \) for \( r > 1 \). This is left as an exercise. ⋄
Proposition 8.3.20 Let \( r \leq n \leq q \). The number \( G_q(r, n) \) of monic Goppa polynomials in \( \mathbb{F}_q[X] \) of degree \( r \) with respect to \( L \) that consists of \( n \) distinct given elements in \( \mathbb{F}_q \) is given by

\[
G_q(r, n) = \sum_{i=0}^{r} (-1)^i \binom{n}{i} q^{r-i}.
\]

Proof. Let \( \mathcal{P}_q(r) \) be the set of all monic polynomials in \( \mathbb{F}_q[X] \) of degree \( r \). Then \( P_q(r) := |\mathcal{P}_q(r)| = q^r \), since \( r \) coefficients of a monic polynomial of degree \( r \) are free to choose in \( \mathbb{F}_q \). Let \( a \) be a vector of length \( n \) with entries the elements of \( L \). Let \( I \) be a subset of \( \{1, \ldots, n\} \). Define

\[
\mathcal{P}_q(r, I) = \{ f(X) \in \mathcal{P}_q(r) \mid f(a_i) = 0 \text{ for all } i \in I \}.
\]

If \( r \geq |I| \), then

\[
\mathcal{P}_q(r, I) = \mathcal{P}_q(r - |I|) \cdot \prod_{i \in I} (X - a_i),
\]

since \( f(a_i) = 0 \) if and only if \( f(X) = g(X)(X - a_i) \), and the \( a_i \) are mutually distinct. Hence

\[
P_q(r, I) := |\mathcal{P}_q(r, I)| = P_q(r - |I|) = q^{r-|I|}
\]

for all \( r \geq |I| \). So \( P_q(r, I) \) depends on \( q, r \) and only on the size of \( I \). Furthermore \( P_q(r, I) \) is empty if \( r < |I| \). The set of monic Goppa polynomials in \( \mathbb{F}_q[X] \) of degree \( r \) with respect to \( L \) is equal to

\[
\bigcap_{i=1}^{n} (\mathcal{P}_q(r) \setminus \mathcal{P}_q(r, a_i)) = \mathcal{P}_q(r) \setminus \left( \bigcup_{i=1}^{n} \mathcal{P}_q(r, a_i) \right).
\]

The principle of inclusion/exclusion gives

\[
G_q(r, n) = \sum_{I} (-1)^{|I|} P_q(r, I) = \sum_{i=0}^{r} (-1)^i \binom{n}{i} q^{r-i}.
\]

Proposition 8.3.21 Let \( r \leq n \leq q \). The number \( SG_q(r, n) \) of square free, monic Goppa polynomials in \( \mathbb{F}_q[X] \) of degree \( r \) with respect to \( L \) that consists of \( n \) distinct given elements in \( \mathbb{F}_q \) is given by

\[
SG_q(r, n) = (-1)^r \binom{n + r - 1}{r} + \sum_{i=0}^{r-1} (-1)^i \binom{n + i - 1}{i} q^{r-i}.
\]

Proof. An outline of the proof is given. The details are left as an exercise.

1) The following recurrence relation holds

\[
SG_q(r, n) = G_q(r, n) - \sum_{a=1}^{\lfloor r/2 \rfloor} G_q(a, n) \cdot SG_q(r - 2a, n)
\]

and that the given formula for \( SG_q(r, n) \) satisfies this recurrence relation.

2) The given formula satisfies the recurrence relation and the starting values.
Example 8.3.22 ****Consider polynomials over the finite field $F_{1024}$. Compute the following numbers.
(1) The number of monic irreducible polynomials of degree 50.
(2) The number of square free monic polynomials of degree 50.
(3) The number of monic Goppa polynomials of degree 50 with respect to $L$.
(4) The number of square free, monic Goppa polynomials of degree 50 with respect to $L$.
****

Question: If $\Gamma(L, g_1) = \Gamma(L, g_2)$ and ..., then $g_1 = g_2$???

***the book of Berlekamp on Algebraic coding theory.
***generating functions, asymptotics***

***Goppa codes meet the GV bound.***

8.3.4 Exercises

8.3.1 Give a proof of Remark 8.3.8.

8.3.2 Let $L = F_9$. Consider the Goppa codes $\Gamma(L, g)$ over $F_9$. Show that the only Goppa polynomials in $F_9[X]$ of degree 2 are $X^2$ and $2X^2$.

8.3.3 Let $L$ be an enumeration of the eight elements of $F_9$. Describe the Goppa codes $\Gamma(L, X)$ and $\Gamma(L, X^2)$ over $F_9$ as alternant codes of the form $ALT_1(a, b)$ and $ALT_1(a, b')$. Determine the parameters of these codes and compare these with the ones given in Proposition 8.3.15.

8.3.4 Let $g$ be a square free Goppa polynomial of degree $r$ over $F_q$. Then the Goppa code $\Gamma(L, g)$ has minimum distance $d \geq 2r + 1$ by Proposition 8.3.15. Explain how to adapt the decoding algorithm mentioned in Remark 8.3.11 to correct $r$ errors.

8.3.5 Let $L = F_{2^{11}}$. Consider the binary Goppa code $\Gamma(L, g)$ with a square free Goppa polynomial $g$ in $F_{2^{11}}[X]$ of degree 93 with respect to $L = F_{2^{11}}$. Give lower bounds on the dimension the minimum distance of this code.

8.3.6 Give a proof of the formula $S_q(r) = q^r - q^{r-1}$ for $r > 1$ by showing by induction that it satisfies the recurrence relation given in the proof of Proposition 8.3.19.

8.3.7 Give a proof of the recurrence relation given in (1) of the proof of Proposition 8.3.21 and show that the given formula for $SG_q(r, n)$ satisfies the recurrence relation.

8.3.8 Consider polynomials over the finite field $F_{2^{11}}$. Let $L = F_{2^{11}}$. Give a numerical approximation of the following numbers.
(1) The number of monic irreducible polynomials of degree 93.
(2) The number of square free monic polynomials of degree 93.
(3) The number of monic Goppa polynomials of degree 93 with respect to $L$.
(4) The number of square free, monic Goppa polynomials of degree 93 with respect to $L$. 

---

8.3. SOME FAMILIES OF POLYNOMIAL CODES 265
8.4 Reed-Muller codes

The \(q\)-ary RS code \(RS_k(n,1)\) of length \(q - 1\) was introduced as a cyclic code in Definition 8.1.1 and it was shown in Proposition 8.1.4 that it could also be described as the code obtained by evaluating all univariate polynomials over \(F_q\) of degree strictly smaller than \(k\) at all the nonzero elements of the finite field \(F_q\). The extended RS codes can be considered as the code evaluating those functions at all the elements of \(F_q\) as done in 8.1.7. The multivariate generalization of the last point view is taken as the definition of Reed-Muller codes and it will be shown that the shortened Reed-Muller codes are certain cyclic codes.

In this section, we assume \(n = q^m\). The vector space \(F^m_q\) has \(n\) elements. Choose an enumerations of its point \(F^m_q = \{P_1, \cdots, P_n\}\). Let \(P = (P_1, \ldots, P_n)\). Define the evaluation maps \(ev_P : F_q[X_1, \ldots, X_m] \rightarrow F_n^m\) by

\[
ev_P(f) = (f(P_1), \ldots, f(P_n))
\]

for \(f \in F_q[X_1, \ldots, X_m]\).

**Definition 8.4.1** The \(q\)-ary Reed-Muller code \(RM_q(u,m)\) of order \(u\) in \(m\) variables is defined as

\[
RM_q(u,m) = \{ ev_P(f) | f \in F_q[X_1, \ldots, X_m], \deg(f) \leq u \}.
\]

The dual of a Reed-Muller code is again Reed-Muller.

**Proposition 8.4.2** The dual code of \(RM_q(u,m)\) is equal to \(RM_q(u^\perp, m)\), where \(u^\perp = m(q - 1) - u - 1\).

**Proof.**

8.4.1 Punctured Reed-Muller codes as cyclic codes

The field \(F_q^m\) can be viewed as an \(m\)-dimensional vector space over \(F_q\). Let \(\beta_1, \cdots, \beta_m\) be a basis of \(F_q^m\) over \(F_q\). Then we have an isomorphism of vector spaces

\[
\varphi : F_q^m \rightarrow F_q^m
\]

such that \(\varphi(\alpha) = (a_1, \ldots, a_m)\) if and only if

\[
\alpha = \sum_{i=1}^{m} a_i \beta_i
\]

for every \(\alpha \in F_q^m\).

Choose a primitive element \(\zeta\) of \(F_q^m\), that is a generator of \(F_q^m\), which is an element of order \(q^m - 1\). Now define the \(n\) points \(P = (P_1, \ldots, P_n)\) in \(F_q^m\) by \(P_1 = 0\) and \(P_i = \varphi(\zeta^{i-1})\) for \(i = 1, \ldots, n\).

\[
P_j := (a_{1j}, a_{2j}, \ldots, a_{mj}), \quad j = 1, \cdots, n.
\]

and let \(\alpha = (a_1, \ldots, a_n)\) with

\[
\alpha_j := \sum_{i=1}^{m} a_{ij} \beta_i \quad j = 1, \cdots, n.
\]
8.4. REED-MULLER CODES

8.4.2 Reed-Muller codes as subfield subcodes and trace codes

Alternant codes are restrictions of generalized RS codes, and it is shown [?, Theorem 15] that Sudan’s decoding algorithm can be applied to this situation. Following [?] we describe the $q$-ary Reed-Muller code $RM_q(u,m)$ as a subfield subcode of $RM_{q^m}(v,1)$ for some $v$, and this last one is a RS code over $F_{q^m}$.

In this section, we assume $n = q^m$. The vector space $F_{q^m}$ has $n$ elements which are often called points, i.e, $F_{q^m} = \{P_1, \cdots, P_n\}$. Since $F_{q^m} \cong F_q^m$, the elements of $F_q^m$ exactly correspond to the points of $F_q^n$. Define the evaluation maps

$$ev_P : F_q[X_1, \ldots, X_m] \rightarrow F_q^n \quad \text{and} \quad ev_\alpha : F_q[Y] \rightarrow F_q^m$$

by $ev_P(f) = (f(P_1), \ldots, f(P_n))$ for $f \in F_q[X_1, \ldots, X_m]$ and $ev_\alpha(g) = (g(\alpha_1), \ldots, g(\alpha_n))$ for $g \in F_q[Y]$.

Recall that the $q$-ary Reed-Muller code $RM_q(u,m)$ of order $u$ is defined as

$$RM_q(u,m) = \{ ev_P(f) \mid f \in F_q[X_1, \ldots, X_m], \deg(f) \leq u \}.$$ 

Similarly the $q^m$-ary Reed-Muller code $RM_{q^m}(v,1)$ of order $v$ is defined as

$$RM_{q^m}(v,1) = \{ ev_\alpha(g) \mid g \in F_q[Y], \deg(g) \leq v \}.$$ 

The following proposition is form [?] and [?].

**Proposition 8.4.3** Let $\rho$ be the rest after division of $u^+ + 1$ by $q - 1$ with quotient $e$, that is

$$u^+ + 1 = e(q - 1) + \rho, \quad \text{where } \rho < q - 1.$$ 

Define $d = (\rho + 1)q^e$. Then $d$ is the minimum distance of $RM_q(u,m)$.

**Proposition 8.4.4** Let $n = q^m$. Let $d$ be the minimum distance of $RM_q(u,m)$. Then $RM_q(u,m)$ is a subfield subcode of $RM_{q^m}(n-d,1)$.

**Proof.** This can be shown by using the corresponding fact for the cyclic punctured codes as shown in Theorem 1 and Corollary 2 of [?]. Here we give a direct proof.

1) Consider the map of rings

$$\varphi : F_{q^m}[Y] \rightarrow F_q[X_1, \ldots, X_m]$$

defined by

$$\varphi(Y) = \beta_1 X_1 + \cdots + \beta_m X_m.$$ 

Let $Tr : F_{q^m} \rightarrow F_q$ be the trace map. This induces an $F_q$-linear map

$$F_{q^m}[X_1, \ldots, X_m] \rightarrow F_q[X_1, \ldots, X_m]$$

that we also denote by $Tr$ and which is defined by

$$Tr \left( \sum_i f_i X^i \right) = \sum_i Tr(f_i) X^i$$
CHAPTER 8. POLYNOMIAL CODES

where the multi-index notation is adopted $X^i = X_1^{i_1} \cdots X_m^{i_m}$ for $i = (i_1, \ldots, i_m) \in \mathbb{N}_0^m$. Define the $\mathbb{F}_q$-linear map

$$T : \mathbb{F}_q^m[Y] \rightarrow \mathbb{F}_q[X_1, \ldots, X_m]$$

by the composition $T = \text{Tr} \circ \varphi$.

The trace map $\text{Tr} : \mathbb{F}_q^m \rightarrow \mathbb{F}_q$ is defined by $\text{Tr}(a) = (\text{Tr}(a_{1}), \ldots, \text{Tr}(a_{n}))$.

Consider the square of maps $\mathbb{F}_q^m \rightarrow \mathbb{F}_q^m \rightarrow \mathbb{F}_q^m$.

We claim that that this diagram commutes. That means that

$$\text{ev}_P \circ T = \text{ev}_\alpha \circ \text{Tr}.$$

In order to show this it is sufficient that $\gamma Y^h$ is mapped to the same element under the two maps for all $\gamma \in \mathbb{F}_q^m$ and $h \in \mathbb{N}_0$, since the maps are $\mathbb{F}_q$-linear and the $\gamma Y^h$ generate $\mathbb{F}_q^m[Y]$ over $\mathbb{F}_q$. Furthermore it is sufficient to show this for the evaluation maps $\text{ev}_P : \mathbb{F}_q[X_1, \ldots, X_m] \rightarrow \mathbb{F}_q$ and $\text{ev}_\alpha : \mathbb{F}_q^m[Y] \rightarrow \mathbb{F}_q^m$ for all points $P \in \mathbb{F}_q^n$ and elements $\alpha \in \mathbb{F}_q^m$ such that $P = (a_1, a_2, \ldots, a_m)$ and $\alpha = \sum_{i=1}^m a_i \beta_i$. Now

$$\text{ev}_P \circ T(\gamma Y^h) = \text{ev}_P(\text{Tr}(\gamma(\beta_1 X_1 + \cdots + \beta_m X_m)^h))) =$$

$$\text{ev}_P\left(\text{Tr}\left(\sum_{i_1+\cdots+i_m=h} \left(\begin{array}{c} \gamma \beta_1 \cdots \beta_m \end{array}\right) X_1^{i_1} \cdots X_m^{i_m}\right)\right) =$$

$$\text{ev}_P\left(\sum_{i_1+\cdots+i_m=h} \left(\begin{array}{c} \gamma \beta_1 \cdots \beta_m \end{array}\right) X_1^{i_1} \cdots X_m^{i_m}\right) =$$

$$\text{ev}_P\left(\sum_{i_1+\cdots+i_m=h} \left(\begin{array}{c} \gamma \beta_1 \cdots \beta_m \end{array}\right) a_1^{i_1} \cdots a_m^{i_m}\right) =$$

$$\text{Tr}\left(\sum_{i_1+\cdots+i_m=h} \left(\begin{array}{c} \gamma \beta_1 \cdots \beta_m \end{array}\right) a_1^{i_1} \cdots a_m^{i_m}\right) =$$

$$\text{Tr}(\gamma(\beta_1 a_1 + \cdots + \beta_m a_m)^h) = \text{Tr}(\gamma a^h) = \text{Tr}(\text{ev}_\alpha(\gamma Y^h)) = \text{Tr} \circ \text{ev}_\alpha(\gamma Y^h).$$

This shows the commutativity of the diagram.

2) Let $h$ be an integer such that $0 \leq h \leq q^m - 1$. Express $h$ in radix-$q$ form

$$h = h_0 + h_1 q + \delta_2 q^2 + \cdots + h_{m-1} q^{m-1}.$$
8.4. REED-MULLER CODES

Define the weight of \( h \) as

\[
W(h) = h_0 + \delta_1 + h_2 + \cdots + h_{m-1}.
\]

We show that for every \( f \in \mathbb{F}_{q^m}[Y] \) there exists a polynomial \( g \in \mathbb{F}_q[X_1, \ldots, X_m] \) such that \( \deg(g) \leq W(h) \) and

\[
ev_P \circ T(\gamma Y^h) = ev_P(g).
\]

It is enough to show this for every \( f \) of the form \( f = \gamma Y^h \) where \( \gamma \in \mathbb{F}_{q^m} \) and \( h \) an integer such that \( 0 \leq h \leq q^m - 1 \). Consider

\[
ev_P \circ T(\gamma Y^h) = ev_P \circ T(\gamma Y^{\sum_i h_i q^i}) = ev_P \circ T\left( \gamma \prod_{t=0}^{m-1} Y^{h_t q^t} \right).
\]

Expanding this expression gives

\[
\text{Tr} \left( \gamma \prod_{t=0}^{m-1} \sum_{i_1 + \cdots + i_m = h_t} \left( \beta^i_1 \cdots \beta^i_m \right)^q a_1^{i_1} \cdots a_m^{i_m} \right).
\]

Let

\[
g = \text{Tr} \left( \gamma \prod_{t=0}^{m-1} \sum_{i_1 + \cdots + i_m = h_t} \left( \beta^i_1 \cdots \beta^i_m \right)^q X_1^{i_1} \cdots X_m^{i_m} \right).
\]

Then this \( g \) has the desired properties.

3) A direct consequence of 1) and 2) is

\[
\text{Tr}(RM_{q^m}(h, 1)) \subseteq RM_q(W(h), m).
\]

We defined \( d = (\rho + 1)q^e \), where \( \rho \) is the rest after division of \( u^1 + 1 \) by \( q - 1 \) with quotient \( e \), that is \( u^1 + 1 = e(q-1) + \rho \), where \( \rho < q - 1 \). Then \( d - 1 \) is the smallest integer \( h \) such that \( W(h) = u^1 + 1 \), see [?] Theorem 5. Hence \( W(h) \leq u^1 \) for all integers \( h \) such that \( 0 \leq h \leq d - 2 \). Therefore

\[
\text{Tr}(RM_{q^m}(d-2, 1)) \subseteq RM_q(u^1, m).
\]

4) So

\[
RM_q(u, m) \subseteq (\text{Tr}(RM_{q^m}(d-2, 1)))^\perp.
\]

5) Let \( C \) be an \( \mathbb{F}_{q^m} \)-linear code in \( \mathbb{F}_{q^m}^n \). The relation between the restriction \( C \cap F_q^n \) and the trace code \( \text{Tr}(C) \) is given by Delsarte’s theorem, see [?] chap. 7, §8 Theorem 11

\[
C \cap F_q^n = (\text{Tr}(C^\perp))^{\perp}.
\]

Application to 4) and using \( RM_{q^m}(n-d, 1) = RM_{q^m}(d-2, 1)^\perp \) gives

\[
RM_q(u, m) \subseteq RM_{q^m}(n-d, 1) \cap F_q^n.
\]

Hence \( RM_q(u, m) \) is a subfield subcode of \( RM_{q^m}(n-d, 1) \).

***Alternative proof making use of the fact that RM is an extension of a restriction of a RS code, and use the duality properties of RS codes and dual(puncture)=shorten(dual)***

\[\diamondsuit\]
Example 8.4.5 The code $RM_q(u, m)$ is not necessarily the restriction of $RM_{q^m}(n - d, 1)$. The following example shows that the punctured Reed-Muller code is a proper subcode of the binary BCH code. Take $q = 2$, $m = 6$ and $u = 3$. Then $u^\perp = 2$, $\sigma = 3$ and $\rho = 0$. So $d = 2^3 = 8$. The code $RM_2^*(3, 6)$ has parameters $[63, 42, 7]$. The binary BCH code with zeros $\zeta^i$ with $i \in \{1, 2, 3, 4, 5, 6\}$ has complete defining set the union of the sets: $\{1, 2, 4, 8, 16, 32\}$, $\{3, 6, 12, 24, 28, 33\}$, $\{5, 10, 20, 40, 17, 34\}$. So the dimension of the BCH code is: $63 - 3 \cdot 6 = 45$. Therefore the BCH code has parameters $[63, 45, 7]$ and it has the punctured RM code as a subcode, but they are not equal. This is explained by the zero $9 = 1 + 2^3$ having 2-weight equal to $2 \leq u^\perp$, whereas no element of the cyclotomic coset $\{9, 18, 36\}$ of 9 is in the set $\{1, 2, 3, 4, 5, 6\}$. The BCH code is the binary restriction of $RM_{64}^*(56, 1)$. Hence $RM_2^*(3, 6)$ is a subcode of the binary restriction of $RM_{64}^*(56, 1)$, but they are not equal.

8.4.3 Exercises

8.4.1 Show the Shift bound for $RM(q, m)^*$ considered as cyclic code is equal to the actual minimum distance.

8.5 Notes

Subfield subcodes of RS code, McEliece-Solomon.

Numerous applications of Reed-Solomon codes can be found in [135]. Twisted BCH codes by Edel.

Folded RS codes by Guruswami.

Stichtenoth-Wirtz

Cauchy and Srivastava codes, Roth-Seroussi and Dür.

Proposition 8.3.19 is due to Carlitz [37]. See also [11, Exercise (3.3)]. Proposition 8.3.21 is a generalization of Retter [98].
Chapter 9

Algebraic decoding

Ruud Pellikaan and Xin-Wen Wu

*** intro ***

9.1 Error-correcting pairs

In this section we give an algebraic way, that is by solving a system of linear equations, to compute the error positions of a received word with respect to Reed-Solomon codes. The complexity of this algorithm is $O(n^3)$.

9.1.1 Decoding by error-correcting pairs

In Definition 7.4.9 we introduced the star product $a \ast b$ for $a, b \in \mathbb{F}_q^n$ by the coordinate wise multiplication $a \ast b = (a_1 b_1, \ldots, a_n b_n)$.

Remark 9.1.1 Notice that multiplying polynomials first and then evaluating gives the same answer as first evaluating and than multiplying. That is, if $f(X), g(X) \in \mathbb{F}_q[X]$ and $h(X) = f(X)g(X)$, then $h(a) = f(a)g(a)$ for all $a \in \mathbb{F}_q$. So

$ev(f(X)g(X)) = ev(f(X)) \ast ev(g(X))$ and

$ev_a(f(X)g(X)) = ev_a(f(X)) \ast ev_a(g(X))$

for the evaluation maps $ev$ and $ev_a$.

Proposition 9.1.2 Let $k + l \leq n$. Then

$\langle GRS_k(a, b) \ast GRS_l(a, c) \rangle = GRS_{k+l-1}(a, b \ast c)$ and

$\langle RS_k(n, b) \ast RS_l(n, c) \rangle = RS_{k+l-1}(n, b + c - 1)$ if $n = q - 1$.

Proof. Now $GRS_k(a, b) = \{ev_a(f(X)) \ast b \mid f(X) \in \mathbb{F}_q[X], \deg f(X) < k \}$ and similar statements hold for $GRS_l(a, c)$ and $GRS_{k+l-1}(a, b \ast c)$. Furthermore

$(ev_a(f(X)) \ast b) \ast (ev_a(g(X)) \ast c) = ev_a(f(X)g(X)) \ast b \ast c$.

271
and \( \deg f(X)g(X) < k + l - 1 \) if \( \deg f(X) < k \) and \( \deg g(X) < l \). Hence

\[
GRS_k(a, b) \ast GRS_l(a, c) \subseteq GRS_{k+l-1}(a, b \ast c).
\]

In general equality does not hold, but we have

\[
\langle GRS_k(a, b) \ast GRS_l(a, c) \rangle = GRS_{k+l-1}(a, b \ast c),
\]

since on both sides the vector spaces are generated by the elements

\[
(ev_a(X^i) \ast b) \ast (ev_a(X^j) \ast c) = ev_a(X^{i+j}) \ast b \ast c
\]

where \( 0 \leq i < k \) and \( 0 \leq j < l \).

Let \( n = q - 1 \). Let \( \alpha \) be a primitive element of \( \mathbb{F}_q^* \). Define \( a_j = \alpha^{j-1} \) and \( b_j = a^{n-j+1} \) for \( j = 1, \ldots, n \). Then \( RS_k(n, b) = GRS_k(a, b) \) by Example 8.1.11. Similar statements hold for \( RS_l(n, c) \) and \( RS_{k+l-1}(n, b + c - 1) \). The statement concerning the star product of \( RS \) codes is now a consequence of the corresponding statement on the \( GRS \) codes.

Example 9.1.3 Let \( n = q - 1, k, l > 0 \) and \( k + l < n \). Then \( RS_k(n, 1) \) is in one-to-one correspondence with polynomials of degree at most \( k - 1 \), and similar statements hold for \( RS_l(n, 1) \) and \( RS_{k+l-1}(n, 1) \). Now \( RS_k(n, 1) \ast RS_l(n, 1) \) corresponds one-to-one with polynomials that are a product of a polynomial of degree at most \( k - 1 \) and \( l - 1 \), respectively, that is reducible polynomials over \( \mathbb{F}_q \) of degree at most \( k + l - 1 \). There exists an irreducible polynomial of degree \( k + l - 1 \), by Remark 7.2.20. Hence

\[
RS_k(n, 1) \ast RS_l(n, 1) \neq RS_{k+l-1}(n, 1).
\]

Definition 9.1.4 Let \( A \) and \( B \) be linear subspaces of \( \mathbb{F}_q^n \). Let \( r \in \mathbb{F}_q^n \). Define the kernel

\[
K(r) = \{ a \in A \mid (a \ast b) \cdot r = 0 \text{ for all } b \in B \}.
\]

Definition 9.1.5 Let \( B^\vee \) be the space of all linear functions \( \beta : B \to \mathbb{F}_q \). Now \( K(r) \) is a subspace of \( A \) and it is the kernel of the linear map

\[
S_r : A \to B^\vee
\]

defined by \( a \mapsto \beta_a \), where \( \beta_a(b) = (a \ast b) \cdot r \). Let \( a_1, \ldots, a_t \) and \( b_1, \ldots, b_m \) be bases of \( A \) and \( B \), respectively. Then the map \( S_r \) has the \( m \times t \) syndrome matrix \((b_i \ast a_j) \cdot r|1 \leq j \leq t, 1 \leq i \leq m\) with respect to these bases.

Example 9.1.6 Let \( A = RS_{t+1}(n, 1) \), \( B = RS_t(n, 0) \). Then \( A \ast B \) is contained in \( RS_{2t}(n, 0) \) by Proposition 9.1.2. Let \( C = RS_{2t}(n, 1) \). Then \( C^\perp = RS_{2t}(n, 0) \) by Proposition 8.1.2. As \( g_{n,k}(X) = g_{0,k}(X) \) for \( n = q - 1 \), by the definition of Reed-Solomon code, we further have \( C^\perp = RS_{2t}(n, 0) \). Hence \( A \ast B \subseteq C^\perp \). Let \( a_i = ev(X^{i-1}) \) for \( i = 1, \ldots , t + 1 \), and \( b_j = ev(X^j) \) for \( j = 1, \ldots , t \), and \( b_j = ev(X^j) \) for \( j = 1, \ldots , t \). Then \( a_{i_1}, \ldots , a_{i_{t+1}} \) is a basis of \( A \) and \( b_1, \ldots , b_t \) is a basis of \( B \). The vectors \( h_{i_1}, \ldots , h_{2t} \) form the rows of a parity check matrix \( H \) for \( C \). Then \( a_i \ast b_j = ev(X^{i+j-1}) = h_{i+j-1} \). Let \( r \) be a received word and \( s = rH^T \) its syndrome. Then

\[
(b_i \ast a_j) \cdot r = s_{i+j-1}.
\]
Hence to compute the kernel \( K(r) \) we have to compute the null space of the matrix of syndromes

\[
\begin{pmatrix}
    s_1 & s_2 & \cdots & s_t & s_{t+1} \\
    s_2 & s_3 & \cdots & s_{t+1} & s_{t+2} \\
    \vdots & \vdots & \ddots & \vdots & \vdots \\
    s_t & s_{t+1} & \cdots & s_{2t-1} & s_{2t}
\end{pmatrix}
\]

We have seen this matrix before as the coefficient matrix of the set of equations for the computation of the error-locator polynomial in the algorithm of APGZ 7.5.3.

**Lemma 9.1.7** Let \( C \) be an \( \mathbb{F}_q \)-linear code of length \( n \). Let \( r \) be a received word with error vector \( e \). If \( A \ast B \subseteq C^\perp \), then \( K(r) = K(e) \).

**Proof.** We have that \( r = c + e \) for some codeword \( c \in C \). Now \( a \ast b \) is a parity check for \( C \), since \( A \ast B \subseteq C^\perp \). So \( (a \ast b) \cdot c = 0 \), and hence \( (a \ast b) \cdot r = (a \ast b) \cdot e \) for all \( a \in A \) and \( b \in B \).

Let \( J \) be a subset of \( \{1, \ldots, n\} \). The subspace

\[ A(J) = \{ a \in A \mid a_j = 0 \text{ for all } j \in J \} \]

was defined in 4.4.10.

**Lemma 9.1.8** Let \( A \ast B \subseteq C^\perp \). Let \( e \) be the error vector of the received word \( r \). If \( I = \text{supp}(e) = \{ i \mid e_i \neq 0 \} \), then \( A(I) \subseteq K(r) \). If moreover \( d(B^\perp) > \text{wt}(e) \), then \( A(I) = K(r) \).

**Proof.** 1) Let \( a \in A(I) \). Then \( a_i = 0 \) for all \( i \) such that \( e_i \neq 0 \), and therefore

\[(a \ast b) \cdot e = \sum_{e_i \neq 0} a_i b_i e_i = 0\]

for all \( b \in B \). So \( a \in K(e) \). But \( K(e) = K(r) \) by Lemma 9.1.7. Hence \( a \in K(r) \). Therefore \( A(I) \subseteq K(r) \).

2) Suppose moreover that \( d(B^\perp) > \text{wt}(e) \). Let \( a \in K(r) \), then \( a \in K(e) \) by Lemma 9.1.7. Hence

\[(e \ast a) \cdot b = e \cdot (a \ast b) = 0\]

for all \( b \in B \), giving \( e \ast a \in B^\perp \). Now \( \text{wt}(e \ast a) \leq \text{wt}(e) < d(B^\perp) \) So \( e \ast a = 0 \) meaning that \( e_i a_i = 0 \) for all \( i \). Hence \( a_i = 0 \) for all \( i \) such that \( e_i \neq 0 \), that is for all \( i \in I = \text{supp}(e) \). Hence \( a \in A(I) \). Therefore \( K(r) \subseteq A(I) \) and equality holds by (1).

**Remark 9.1.9** Let \( I = \text{supp}(e) \) be the set of error positions. The set of zero coordinates of \( a \in A(I) \) contains the set of error positions by Lemma 9.1.8. For that reason the elements of \( A(I) \) are called error-locator vectors or functions. But the space \( A(I) \) is not known to the receiver. The space \( K(r) \) can be computed after receiving the word \( r \). The equality \( A(I) = K(r) \) implies that all elements of \( K(r) \) are error-locator functions.

Let \( A \ast B \subseteq C^\perp \). The basic algorithm for the code \( C \) computes the kernel \( K(r) \).
for every received word \( r \). If this kernel is nonzero, it takes a nonzero element \( a \) and determines the set \( J \) of zero positions of \( a \). If \( d(B^\perp) > wt(e) \), where \( e \) is the error-vector, then \( J \) contains the support of \( e \) by Lemma 9.1.8. If the set \( J \) is not too large, the error values are computed.

Thus we have a basic algorithm for every pair \((A, B)\) of subspaces of \( F_q^n \) such that \( A * B \subseteq C^\perp \). If \( A \) is small with respect to the number of errors, then \( K(r) = 0 \). If \( A \) is large, then \( B \) becomes small, which results in a large code \( B^\perp \), and it will be difficult to meet the requirement \( d(B^\perp) > wt(e) \).

**Definition 9.1.10** Let \( A, B \) and \( C \) be subspaces of \( F_q^n \). Then \((A, B)\) is called a \( t \)-error-correcting pair for \( C \) if the following conditions are satisfied:

1. \( A * B \subseteq C^\perp \),
2. \( \dim(A) > t \),
3. \( d(B^\perp) > t \),
4. \( d(A) + d(C) > n \)

**Proposition 9.1.11** Let \((A, B)\) be a \( t \)-error-correcting pair for \( C \). Then the basic algorithm corrects \( t \) errors for the code \( C \) with complexity \( O(n^3) \).

**Proof.** The pair \((A, B)\) is a \( t \)-error-correcting for \( C \), so \( A * B \subseteq C^\perp \) and the basic algorithm can be applied to decode \( C \).

If a received word \( r \) has at most \( t \) errors, then the error vector \( e \) with support \( I \) has size at most \( t \) and \( A(I) \) is not zero, since \( I \) imposes at most \( t \) linear conditions on \( A \) and the dimension of \( A \) is at least \( t + 1 \).

Let \( a \) be a nonzero element of \( K(r) \). Let \( J = \{ j \mid a_j = 0 \} \).

We assumed that \( d(B^\perp) > t \). So \( K(r) = A(I) \) by Lemma 9.1.8. So \( a \) is an error-locator vector and \( J \) contains \( I \).

The weight of the vector \( a \) is at least \( d(A) \), so \( a \) has at most \( n - d(A) < d(C) \) zeros by (4) of Definition 9.1.10. Hence \( |J| < d(C) \) and Proposition 6.2.9 or 6.2.15 gives the error values.

The complexity is that of solving systems of linear equations, that is \( O(n^3) \). \( \diamond \)

We will show the existence of error-correcting pairs for (generalized) Reed-Solomon codes.

**Proposition 9.1.12** The codes \( GRS_{n-2t}(a, b) \) and \( RS_{n-2t}(n, b) \) have \( t \)-error-correcting pairs.

**Proof.** Let \( C = GRS_{n-2t}(a, b) \). Then \( C^\perp = GRS_{2t}(a, c) \) for some \( c \) by Proposition 8.1.21. Let \( A = GRS_{t+1}(a, 1) \) and \( B = GRS_t(a, c) \). Then \( A * B \subseteq C^\perp \) by Proposition 9.1.2.

The codes \( A, B \) and \( C \) have parameters \([n, t + 1, n - t] \), \([n, t, n - t + 1] \) and \([n, n - 2t, 2t + 1] \), respectively, by Proposition 8.1.14.

Furthermore \( B^\perp \) has parameters \([n, n - t, t + 1] \) by Corollary 3.2.7, and has minimum distance \( t + 1 \). Hence \((A, B)\) is a \( t \)-error-correcting pair for \( C \).

The code \( RS_{n-2t}(n, b) \) is of the form \( GRS_{n-2t}(a, b) \). Therefore the pair of codes \((RS_{t+1}(n, 1), RS_t(n, n - b + 1))\) is a \( t \)-error-correcting pair for the code \( RS_{n-2t}(n, b) \). \( \diamond \)
Example 9.1.13 Choose \( \alpha \in \mathbb{F}_{16} \) such that \( \alpha^4 = \alpha + 1 \) as primitive element of \( \mathbb{F}_{16} \). Let \( C = RS_{11}(15, 1) \). Let
\[
 r = (0, \alpha^4, \alpha^3, \alpha^{14}, \alpha^1, \alpha^{10}, \alpha^7, \alpha^9, \alpha^2, \alpha^{13}, \alpha^5, \alpha^{12}, \alpha^{11}, \alpha^6, \alpha^3)
\]
be a received word with respect to the code \( C \) with 2 errors. We show how to find the transmitted codeword by means of the basic algorithm.

The dual of \( C \) is equal to \( RS_4(15, 0) \). Hence \( RS_3(15, 1) \ast RS_2(15, 0) \) is contained in \( RS_4(15, 0) \). Take \( A = RS_3(15, 1) \) and \( B = RS_2(15, 0) \). Then \( A \) is a \([15, 3, 13]\) code, and the dual of \( B \) is \( RS_3(15, 1) \) which has minimum distance 3. Therefore \((A, B)\) is a 2-error-correcting pair for \( C \) by Proposition 9.1.12. Let
\[
 H = (\alpha_i^j \mid 1 \leq i \leq 4, 0 \leq j \leq 14).
\]

Then \( H \) is a parity check matrix of \( C \). The syndrome vector of \( r \) equals
\[
(s_1, s_2, s_3, s_4) = rH^T = (\alpha^{10}, 1, 1, \alpha^{10}).
\]
The space \( K(r) \) consists of the evaluation \( ev(a_0 + a_1 X + a_2 X^2) \) of all polynomials \( a_0 + a_1 X + a_2 X^2 \) such that \( (a_0, a_1, a_2)^T \) is in the null space of the matrix
\[
\begin{pmatrix}
 s_1 & s_2 & s_3 & s_4 \\
 1 & 1 & 1 & 1
\end{pmatrix}
\approx
\begin{pmatrix}
 1 & 0 & 1 \\
 0 & 1 & \alpha^5
\end{pmatrix}.
\]
So \( K(r) = (ev(1 + \alpha^5 X + X^2)) \). The polynomial \( 1 + \alpha^5 X + X^2 \) has \( \alpha^6 \) and \( \alpha^9 \) as zeros. Hence the error positions are at the 7-th and 10-th coordinate. In order to compute the error values by Proposition 6.2.9 we have to find a linear combination of the 7-th and 10-th column of \( H \) that equals the syndrome vector. The system
\[
\begin{pmatrix}
 \alpha^6 & \alpha^9 & \alpha^{10} \\
 \alpha^{12} & \alpha^3 & 1 \\
 \alpha^5 & \alpha^{12} & 1 \\
 \alpha^9 & \alpha^5 & \alpha^{10}
\end{pmatrix}
\]
has \((\alpha^5, \alpha^5)^T\) as unique solution. That is, the error vector \( e \) has \( e_7 = \alpha^5 \), \( e_{10} = \alpha^5 \) and \( e_i = 0 \) for all \( i \not\in \{7, 10\} \). Therefore the transmitted codeword is
\[
c = r - e = (0, \alpha^4, \alpha^8, \alpha^{14}, \alpha^1, \alpha^{10}, \alpha^7, \alpha^9, \alpha^2, \alpha^{13}, \alpha^5, \alpha^{13}, \alpha^{11}, \alpha^6, \alpha^7).
\]

9.1.2 Existence of error-correcting pairs

Example 9.1.14 Let \( C \) be the binary cyclic code with defining set \( \{1, 3, 7, 9\} \) as in Examples 7.4.8 and 7.4.17. Then \( d(C) \geq 7 \) by the Roos bound 7.4.16 with \( U = \{0, 4, 12, 20\} \) and \( V = \{2, 3, 4\} \). ***This gives us an error correcting pair***

Remark 9.1.15 The great similarity between the concept of an error-correcting pair and the techniques used by Van Lint and Wilson in the AB bound one can see in the reformulation of the Roos bound in Remark 7.4.25. A special case of this reformulation is obtained if we take \( a = b = t \).

Proposition 9.1.16 Let \( C \) be an \( \mathbb{F}_q \)-linear code of length \( n \). Let \((A, B)\) be a pair of \( \mathbb{F}_q \)-linear codes of length \( n \) such that the following properties hold:
1. \((A \ast B) \perp C\)
2. \( k(A) > t \)
3. \( d(B^\perp) > t \)
4. \( d(A) + 2t > n \)
5. \( d(A^\perp) > 1 \)

Then \( d(C) \geq 2t + 1 \) and \((A, B)\) is a \( t \)-error-correcting pair for \( C \).
Proof. The conclusion on the minimum distance of $C$ is explained in Remark 7.4.25. Conditions (1), (2) and (3) are the same as in the ones in the definition of a $t$-error-correcting pair. Condition (4) in the proposition is stronger than in the definition, since $d(A) + d(C) \geq d(A) + 2t + 1 > d(A) + 2t > n$. 

Remark 9.1.17 As a consequence of this proposition there is an abundance of examples of codes $C$ with minimum distance at least $2t + 1$ that have a $t$-error-correcting pair. Take for instance $A$ and $B$ MDS codes with parameters $[n, t + 1, n - t]$ and $[n, t, n - t + 1]$, respectively. Then $k(A) > t$ and $d(B^\perp) > t$, since $B^\perp$ is an $[n, n - t, t + 1]$ code. Take $C = (A \ast B)^\perp$. Then $d(C) \geq 2t + 1$ and $(A, B)$ is a $t$-error-correcting pair for $C$. Then the dimension of $C$ is at least $n - t(t + 1)$ and is most of the time equal to this lower bound.

Remark 9.1.18 For a given code $C$ it is hard to find a $t$-error-correcting pair with $t$ close to half the minimum distance. Generalized Reed-Solomon codes have this property as we have seen in ?? and Algebraic geometry codes too as we shall see in **** ??***. We conjecture that if an $[n, n - 2t, 2t + 1]$ MDS code has a $t$-error-correcting pair, then this code is a GRS code. This is proven in the cases $t = 1$ and $t = 2$.

Proposition 9.1.19 Let $C$ be an $\mathbb{F}_q$-linear code of length $n$ and minimum distance $d$. Then $C$ has a $t$-error-correcting pair if $t \leq (n - 1)/(n - d + 2)$.

Proof. There exists an $m$ and an $\mathbb{F}_{q^n}$-linear $[n, n - d + 1, d]$ code $D$ that contains $C$, by Corollary 4.3.25. Let $t$ be a positive integer such that $t \leq (n - 1)/(n - d + 2)$. It is sufficient to show that $D$ has a $t$-error-correcting pair. Let $B$ be an $[n, t, n - t + 1]$ code with the all one vector in it. Such a code exists if $m$ is sufficiently large. Then $B^\perp$ is an $[n, n - t, t + 1]$ code. So $d(B^\perp) > t$. Take $A = (B \ast D)^\perp$. *** Now $A$ is contained in $D^\perp$, since the all one vector is in $B$, and $D^\perp$ is an $[n, d - 1, n - d + 2]$ code. So $d(A) \geq n - d + 2$. *** Now $D^\perp \subseteq A$, since the all one vector is in $B$. We have that $D^\perp$ is an $[n, d - 1, n - d + 2]$ code, so $d(A) \geq d(D^\perp) = n - d + 2$. Hence $d(A) + d(D) > n$. Let $b_1, \ldots, b_t$ be a basis of $B$ and $d_1, \ldots, d_{n - d + 1}$ be a basis of $D$. Then $x \in A$ if and only if $x \cdot (b_i \ast d_j) = 0$ for all $i = 1, \ldots, t$ and $j = 1, \ldots, n - d + 1$. This is system of $t(n - d + 1)$ homogeneous linear equations and $n - t(n - d + 1) \geq t + 1$ by assumption. Hence $k(A) \geq n - t(n - d + 1) > t$. Therefore $(A, B)$ is a $t$-error-correcting pair for $D$ and a fortiori for $C$. 

9.1.3 Exercises

9.1.1 Choose $\alpha \in \mathbb{F}_{16}$ such that $\alpha^4 = \alpha + 1$ as primitive element of $\mathbb{F}_{16}$. Let $C = RS_{11}(15,0)$. Let $$r = (\alpha, 0, \alpha^{11}, \alpha^{10}, \alpha^5, \alpha^{13}, \alpha, \alpha^8, \alpha^5, \alpha^{10}, \alpha^4, \alpha^2, 0, 0)$$ be a received word with respect to the code $C$ with 2 errors. Find the transmitted codeword.

9.1.2 Consider the binary cyclic code of length 21 and defining set $\{0, 1, 3, 7\}$. This code has minimum distance 8. Give a 3 error correcting pair for this code.

9.1.3 Consider the binary cyclic code of length 35 and defining set $\{1, 5, 7\}$. This code has minimum distance 7. Give a 3 error correcting pair for this code.
9.2 Decoding by key equation

In Section 7.5.5, we introduced Key equation. Now we introduce two algorithms which solve the key equation, and thus decode cyclic codes efficiently.

9.2.1 Algorithm of Euclid-Sugiyama

In Section 7.5.5 we have seen that the decoding of BCH code with designed minimum distance \( \delta \) is reduced to the problem of finding a pair of polynomials \((\sigma(Z), \omega(Z))\) satisfying the following key equation for a given syndrome polynomial \( S(Z) = \sum_{i=1}^{\delta-1} S_i Z^{i-1} \),

\[
\sigma(Z)S(Z) \equiv \omega(Z) \pmod{Z^{\delta-1}}
\]
such that \( \deg(\sigma(Z)) \leq t = (\delta - 1)/2 \) and \( \deg(\omega(Z)) \leq \deg(\sigma(Z)) - 1 \). Here, \( \sigma(Z) = \sum_{i=1}^{\delta-1} \sigma_i Z^{i-1} \) is the error-locator polynomial, and \( \omega(Z) = \sum_{i=1}^{\delta-1} \omega_i Z^{i-1} \) is the error-evaluator polynomial. Note that \( \sigma_1 = 1 \) by definition.

Given the key equation, the Euclid-Sugiyama algorithm (which is also called Sugiyama algorithm in the literature) finds the error-locator and error-evaluator polynomials, by an iterative procedure. This algorithm is based on the well-known Euclidean algorithm. To better understand the algorithm, we briefly review the Euclidean algorithm first. For a pair of univariate polynomials, namely, \( r_{-1}(Z) \) and \( r_0(Z) \), the Euclidean algorithm finds their greatest common divisor, which we denote by \( \gcd(r_{-1}(Z), r_0(Z)) \). The Euclidean algorithm proceeds as follows.

\[
\begin{align*}
r_{-1}(Z) &= q_1(Z)r_0(Z) + r_1(Z), \quad \deg(r_1(Z)) < \deg(r_0(Z)) \\
r_0(Z) &= q_2(Z)r_1(Z) + r_2(Z), \quad \deg(r_2(Z)) < \deg(r_1(Z)) \\
& \vdots \quad \vdots \\
r_{s-2}(Z) &= q_s(Z)r_{s-1}(Z) + r_s(Z), \quad \deg(r_s(Z)) < \deg(r_{s-1}(Z)) \\
r_{s-1}(Z) &= q_{s+1}(Z)r_s(Z).
\end{align*}
\]

In each iteration of the algorithm, the operation of \( r_{j-2}(Z) = q_j(Z)r_{j-1}(Z) + r_j(Z) \), with \( \deg(r_j(Z)) < \deg(r_{j-1}(Z)) \), is implemented by division of polynomials, that is, dividing \( r_{j-2}(Z) \) by \( r_{j-1}(Z) \), with \( r_j(Z) \) being the remainder. The algorithm keeps running, until it finds a remainder which is the zero polynomial. That is, the algorithm stops after it completes the \( s \)-iteration, where \( s \) is the smallest \( j \) such that \( r_{j+1}(Z) = 0 \). It is easy to prove that \( r_s(Z) = \gcd(r_{-1}(Z), r_0(Z)) \).

We are now ready to present the Euclid-Sugiyama algorithm for solving the key equation.

Algorithm 9.2.1 (Euclid-Sugiyama Algorithm)

**Input:** \( r_{-1}(Z) = Z^{\delta-1}, r_0(Z) = S(Z), U_{-1}(Z) = 0, \) and \( U_0(Z) = 1 \).

Proceed with the Euclidean algorithm for \( r_{-1}(Z) \) and \( r_0(Z) \), as presented above, until an \( r_s(Z) \) is reached such that

\[
\deg(r_{s-1}(Z)) \geq \frac{1}{2}(\delta - 1) \quad \text{and} \quad \deg(r_s(Z)) \leq \frac{1}{2}(\delta - 3),
\]
CHAPTER 9. ALGEBRAIC DECODING

Update

\[ U_j(Z) = q_j(Z)U_{j-1}(Z) + U_{j-2}(Z). \]

**Output:** The following pair of polynomials:

\[
\begin{align*}
\sigma(Z) &= \epsilon U_s(Z) \\
\omega(Z) &= (-1)^s \epsilon r_s(Z)
\end{align*}
\]

where \( \epsilon \) is chosen such that \( \sigma(0) = 1 \).

Then the error-locator and evaluator polynomials are given as \( \sigma(Z) = \epsilon U_s(Z) \) and \( \omega(Z) = (-1)^s \epsilon r_s(Z) \). Note that the Euclid-Sugiyama algorithm does not have to run the Euclidean algorithm completely; it has a different stopping parameter \( s \).

**Example 9.2.2** Consider the code \( C \) given in Examples 7.5.13 and 7.5.21. It is a narrow sense BCH code of length 15 over \( \mathbb{F}_{16} \) of designed minimum distance \( \delta = 5 \). Let \( r \) be the received word

\[ r = (\alpha^5, \alpha^8, \alpha^{11}, \alpha^{10}, \alpha^7, \alpha^{12}, \alpha^{11}, 1, \alpha, \alpha^{12}, \alpha^{14}, \alpha^{12}, \alpha^2, 0) \]

Then \( S_1 = \alpha^{12}, S_2 = \alpha^7, S_3 = 0 \) and \( S_4 = \alpha^2 \). So, \( S(Z) = \alpha^{12} + \alpha^7 Z + \alpha^2 Z^3 \). Running the Euclid-Sugiyama algorithm with the input \( S(Z) \), the results for each iteration are given by the following table.

<table>
<thead>
<tr>
<th>( j )</th>
<th>( r_{j-1}(Z) )</th>
<th>( r_j(Z) )</th>
<th>( U_{j-1}(Z) )</th>
<th>( U_{j}(Z) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>( Z^4 )</td>
<td>( \alpha^2 Z^3 + \alpha^7 Z + \alpha^{12} )</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>( \alpha^2 Z^3 + \alpha^7 Z + \alpha^{12} )</td>
<td>( \alpha^5 Z^2 + \alpha^{10} Z )</td>
<td>1</td>
<td>( \alpha^{13} Z )</td>
</tr>
<tr>
<td>2</td>
<td>( \alpha^5 Z^2 + \alpha^{10} Z )</td>
<td>( \alpha^2 Z + \alpha^{12} )</td>
<td>( \alpha^{13} Z )</td>
<td>( \alpha^{10} Z^2 + Z + 1 )</td>
</tr>
</tbody>
</table>

Thus, we have found the error-locator polynomial as \( \sigma(Z) = U_2(Z) = 1 + Z + \alpha^{12} Z^2 \), and the error-evaluator polynomial as \( \omega(Z) = r_2(Z) = \alpha^{12} + \alpha^2 Z \).

**9.2.2 Algorithm of Berlekamp-Massey**

Consider again the following key equation

\[ \sigma(Z)S(Z) \equiv \omega(Z) \pmod{Z^{\delta-1}} \]

such that \( \deg(\sigma(Z)) \leq t = (\delta - 1)/2 \) and \( \deg(\omega(Z)) \leq \deg(\sigma(Z)) - 1 \); and \( S(Z) = \sum_{i=1}^{\delta-1} S_i Z^{i-1} \) is given.

It is easy to show that the problem of solving the key equation is equivalent to the problem of solving the following matrix equation with unknown \( (\sigma_2, \ldots, \sigma_{t+1})^T \)
The Berlekamp-Massey algorithm which we will introduce in this section can solve this matrix equation by finding $\sigma_2, \ldots, \sigma_{t+1}$ for the following recursion
\[
S_i = - \sum_{j=2}^{t+1} \sigma_i S_{t-j+1}, \quad i = t + 1, \ldots, 2t.
\]

We should point out that the Berlekamp-Massey algorithm actually solves a more general problem, that is, for a given sequence $E_0, E_1, E_2, \ldots, E_{N-1}$ of length $N$ (which we denote by $E$ in the rest of the section), it finds the recursion
\[
E_i = - \sum_{j=1}^{L} \Lambda_j E_{i-j}, \quad i = L, \ldots, N-1,
\]
for which $L$ is smallest. If the matrix equation has no solution, the Berlekamp-Massey algorithm then finds a recursion with $L > t$.

To make it more convenient to present the Berlekamp-Massey algorithm and to prove the correctness, we denote $\Lambda(Z) = \sum_{i=0}^{\infty} \Lambda_i Z^i$ with $\Lambda_0 = 1$. The above recursion is denoted by $(\Lambda(Z), L)$, and $L = \deg(\Lambda(Z))$ is called the length of the recursion.

The Berlekamp-Massey algorithm is an iterative procedure for finding the shortest recursion for producing successive terms of the sequence $E$. The $r$-th iteration of the algorithm finds the shortest recursion $(\Lambda^{(r)}(Z), L_r)$ where $L_r = \deg(\Lambda^{(r)}(X))$, for producing the first $r$ terms of the sequence $E$, that is,
\[
E_i = - \sum_{j=1}^{L_r} \Lambda_j^{(r)} E_{i-j}, \quad i = L_r, \ldots, r-1,
\]
or equivalently,
\[
\sum_{j=0}^{L_r} \Lambda_j^{(r)} E_{i-j} = 0, \quad i = L_r, \ldots, r-1,
\]
with $\Lambda_0^{(r)} = 0$.

**Algorithm 9.2.3** (Berlekamp-Massey Algorithm)

(Initialization) $r = 0$, $\Lambda(Z) = B(Z) = 1$, $L = 0$, $\lambda = 1$, and $b = 1$.

1) If $r = N$, stop. Otherwise, compute $\Delta = \sum_{j=0}^{L} \Lambda_j E_{r-j}$
2) If $\Delta = 0$, then $\lambda \leftarrow \lambda + 1$, and go to 5).
3) If $\Delta \neq 0$ and $2L > r$, then
\[
\Lambda(Z) \leftarrow \Lambda(Z) - \Delta b^{-1} Z^L B(Z)
\]
\[\lambda \leftarrow \lambda + 1\]
and go to 5).

4) If $\Delta \neq 0$ and $2L \leq r$, then
\[
T(Z) \leftarrow \Lambda(Z) \text{ (temporary storage of } \Lambda(Z))
\]
\[
\Lambda(Z) \leftarrow \Lambda(Z) - \Delta b^{-1} Z^L B(Z)
\]
\[
L \leftarrow r + 1 - L
\]
\[
B(Z) \leftarrow T(Z)
\]
\[b \leftarrow \Delta
\]
\[\lambda \leftarrow 1
\]

5) $r \leftarrow r + 1$ and return to 1).

**Example 9.2.4** Consider again the code $C$ given in Example 9.2.2. Let $r$ be the received word
\[r = (\alpha^5, \alpha^8, \alpha^{11}, \alpha^{10}, \alpha^7, \alpha^{12}, \alpha^{11}, 1, \alpha, \alpha^{12}, \alpha^{14}, \alpha^2, 0)\]
Then $S_1 = \alpha^{12}$, $S_2 = \alpha^7$, $S_3 = 0$ and $S_4 = \alpha^2$.

Now let us compute the error-locator polynomial $\sigma(Z)$ by using the Berlekamp-Massey algorithm. Letting $E_i = S_{i+1}$, for $i = 0, 1, 2, 3$, we have a sequence $E = \{E_0, E_1, E_2, E_3\} = \{\alpha^{12}, \alpha^7, 0, \alpha^2\}$, as the input of the algorithm. The intermediate and final results of the algorithm are given in the following table.

<table>
<thead>
<tr>
<th>$r$</th>
<th>$\Delta$</th>
<th>$B(Z)$</th>
<th>$\Lambda(Z)$</th>
<th>$L$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$\alpha^{12}$</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>$1 + \alpha^{12} Z$</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>$\alpha^2$</td>
<td>1</td>
<td>$1 + \alpha^{10} Z$</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>$\alpha$</td>
<td>$1 + \alpha^{10} Z$</td>
<td>$1 + \alpha^{10} Z + \alpha^5 Z^2$</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>$1 + \alpha^{10} Z$</td>
<td>$1 + Z + \alpha^{10} Z^2$</td>
<td>2</td>
</tr>
</tbody>
</table>

The result of the last iteration the Berlekamp-Massey algorithm, $\Lambda(Z)$, is the error-locator polynomial. That is
\[
\sigma(Z) = \sigma_1 + \sigma_2 Z + \sigma_2 Z^2 = \Lambda(Z) = \Lambda_0 + \lambda_1 Z + \Lambda_2 Z^2 = 1 + Z + \alpha^{10} Z^2.
\]
Substituting this into the key equation, we then get $\omega(Z) = \alpha^{12} + \alpha^2 Z$.

**Proof of the correctness**: will be done.

**Complexity and some comparison between E-S and B-M algorithms**: will be done.
9.2.3 Exercises

9.2.1 Take \( \alpha \in \mathbb{F}^*_{16} \) with \( \alpha^4 = 1 + \alpha \) as primitive element. Let \( C \) be the BCH code over \( \mathbb{F}_{16} \), of length 15 and designed minimum distance 5, with defining set \{1, 2, 3, 4, 6, 8, 9, 12\}. The generator polynomial is \( 1 + X^4 + X^6 + X^7 + X^8 \) (see Example 7.3.13). Let

\[
\mathbf{r} = (0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0)
\]

be a received word with respect to the code \( C \). Find the syndrome polynomial \( S(Z) \). Write the key equation.

9.2.2 Consider the same code and same received word given in last exercise. Using the Berlekamp-Massey algorithm, compute the error-locator polynomial. Determine the number of errors occurred in the received word.

9.2.3 For the same code and same received word given the previous exercises, using the Euclid-Sugiyama algorithm, compute the error-locator and error-evaluator polynomials. Find the codeword which is closest to the received word.

9.2.4 Let \( \alpha \in \mathbb{F}_{16}^* \) with \( \alpha^4 = 1 + \alpha \) as in Exercise 9.2.1. For the following sequence \( \mathbf{E} \) over \( \mathbb{F}_{16} \), using the Berlekamp-Massey algorithm, find the shortest recursion for producing successive terms of \( \mathbf{E} \):

\[
\mathbf{E} = \{ \alpha^{12}, 1, \alpha^{14}, \alpha^{13}, 1, \alpha^{11} \}.
\]

9.2.5 Consider the [15, 9, 7] Reed-Solomon code over \( \mathbb{F}_{16} \) with defining set \{1, 2, 3, 4, 5, 6\}. Suppose the received word is

\[
\mathbf{r} = (0, 0, \alpha^{11}, 0, 0, \alpha^5, 0, \alpha, 0, 0, 0, 0, 0, 0).
\]

Using the Berlekamp-Massey algorithm, find the codeword which is closest to the received word.

9.3 List decoding by Sudan’s algorithm

A decoding algorithm is efficient if the complexity is bounded above by a polynomial in the code length. Brute-force decoding is not efficient, because for a received word, it may need to compare \( q^k \) codewords to return the most appropriate codeword. The idea behind list decoding is that instead of returning a unique codeword, the list decoder returns a small list of codewords. A list-decoding algorithm is efficient, if both the complexity and the size of the output list of the algorithm are bounded above by polynomials in the code length. List decoding was first introduced by Elias and Wozencraft in 1950’s.

We now describe a list decoder more precisely. Suppose \( C \) is a \( q \)-ary \([n, k, d]\) code, \( t \leq n \) is a positive integer. For any received word \( \mathbf{r} = (r_1, \cdots, r_n) \in \mathbb{F}_q^n \), we refer to any codeword \( \mathbf{c} \) in \( C \) satisfying \( d(\mathbf{c}, \mathbf{r}) \leq t \) as a \( t \)-consistent codeword. Let \( l \) be a positive integer less than or equal to \( q^k \). The code \( C \) is called \( (t, l) \)-decodable, if for any word \( \mathbf{r} \in \mathbb{F}_q^n \) the number of \( t \)-consistent codewords is at most \( l \). If for any received word, a list decoder can find all the \( t \)-consistent codewords, and the output list has at most \( l \) codewords, then the decoder is called a \( (t, l) \)-list decoder. In 1997 for the first time, Sudan proposed an efficient list-decoding algorithm for Reed-Solomon codes. Later, Sudan’s list-decoding algorithm was generalized to decoding algebraic-geometric codes and Reed-Muller codes.
9.3.1 Error-correcting capacity

Suppose a decoding algorithm can find all the \( t \)-consistent codewords for any received word. We call \( t \) the error-correcting capacity or decoding radius of the decoding algorithm. As we have known in Section ??, for any \([n,k,d]\) code, if \( t \leq \left\lfloor \frac{d-1}{2} \right\rfloor \), then there is only one \( t \)-consistent codeword for any received word. In other words, any \([n,k,d]\) code is \( (\left\lfloor \frac{d-1}{2} \right\rfloor,1) \)-decodable. The decoding algorithms in the previous sections return a unique codeword for any received word; and they achieve an error-correcting capability less than or equal to \( \left\lfloor \frac{d-1}{2} \right\rfloor \). The list decoding achieves a decoding radius greater than \( \left\lfloor \frac{d-1}{2} \right\rfloor \); and the size of the output list must be bounded above by a polynomial in \( n \).

It is natural to ask the following question: For a \([n,k,d]\) linear code \( C \) over \( \mathbb{F}_q \), what is the maximal value \( t \), such that \( C \) is \((t,l)\)-decodable for a \( l \) which is bounded above by a polynomial in \( n \)? In the following, we give a lower bound on the maximum \( t \) such that \( C \) is \((t,l)\)-decodable, which is called Johnson bound in the literature.

**Proposition 9.3.1** Let \( C \subseteq \mathbb{F}_q^n \) be any linear code of minimum distance \( d = (1 - 1/q)(1 - \beta)n \) for \( 0 < \beta < 1 \). Let \( t = (1 - 1/q)(1 - \gamma)n \) for \( 0 < \gamma < 1 \). Then for any word \( r \in \mathbb{F}_q^n \),

\[
|B_t(r) \cap C| \leq \min\left\{ \frac{n(q - 1)}{2}, \frac{(1 - \beta)(\gamma^2 - \beta)}{2(n(q - 1) - 1)} \right\} \quad \text{when} \quad \gamma > \sqrt{\beta} \\
\leq \frac{n(q - 1)}{2} \quad \text{when} \quad \gamma = \sqrt{\beta}
\]

where, \( B_t(r) = \{ x \in \mathbb{F}_q^n \mid d(x, r) \leq t \} \) is the Hamming ball of radius \( t \) around \( r \).

We will prove this proposition later. We are now ready to state the Johnson bound.

**Theorem 9.3.2** For any linear code \( C \subseteq \mathbb{F}_q^n \) of relative minimum distance \( \delta = d/n \), it is \((t,l(n))\)-decodable with \( l(n) \) bounded above by a linear function in \( n \), provided that

\[
\frac{t}{n} \leq \left( 1 - \frac{1}{q} \right) \left( 1 - \sqrt{1 - \frac{q}{q - 1}} \right).
\]

**Proof.** For any received word \( r \in \mathbb{F}_q^n \), the set of \( t \)-consistent codewords \( \{ c \in C \mid d(c, r) \leq t \} = B_t(r) \cap C \).

Let \( \beta \) be a positive real number and \( \beta < 1 \). Denote \( d = (1 - 1/q)(1 - \beta)n \). Let \( t = (1 - 1/q)(1 - \gamma)n \) for some \( 0 < \gamma < 1 \). Suppose

\[
\frac{t}{n} \leq \left( 1 - \frac{1}{q} \right) \left( 1 - \sqrt{1 - \frac{q}{q - 1}} \right)
\]

Then, \( \gamma \geq \sqrt{1 - \frac{q}{q - 1}} \cdot \frac{d}{n} = \sqrt{\beta} \). By Proposition 9.3.1, the number of \( t \)-consistent codewords, \( l(n) \), which is \( |B_t(r) \cap C| \), is bounded above by a polynomial in \( n \), here \( q \) is viewed as a constant.

**Remark 9.3.3** The classical error-correcting capability is \( t = \left\lfloor \frac{d-1}{2} \right\rfloor \). For a linear \([n,k]\) code of minimum distance \( d \), we have \( d \leq n - k + 1 \) (Note that for Reed-Solomon codes, \( d = n - k + 1 \)). Thus, the normalized capability

\[
\tau = \frac{t}{n} \leq \frac{1}{n} \cdot \left\lfloor \frac{n - k}{2} \right\rfloor \approx \frac{1}{2} - \frac{1}{2} \kappa
\]
9.3. LIST DECODING BY SUDAN’S ALGORITHM

where $\kappa$ is the code rate.

Let us compare this with the Johnson bound. From Theorem 9.3.2 and $d \leq n - k + 1$, the Johnson bound is

$$
\frac{1 - \frac{1}{q}}{1 - \sqrt{1 - \frac{q}{q+1} \delta}} \\
\approx 1 - \sqrt{\kappa}
$$

for large $n$ and large $q$. A comparison is given by the following Figure 9.1.

![Figure 9.1: Classical error-correcting capability v.s. the Johnson bound.](image)

To prove Proposition 9.3.1, we need the following lemma.

**Lemma 9.3.4** Let $v_1, \ldots, v_m$ be $m$ non-zero vectors in the real $N$-dimensional space, $\mathbb{R}^N$, satisfying that the inner product, $v_i \cdot v_j \leq 0$, for every pair of distinct vectors. Then we have the following upper bounds on $m$:

1. $m \leq 2N$.
2. If there exists a non-zero vector $u \in \mathbb{R}^N$ such that $u \cdot v_i \geq 0$ for all $i = 1, \ldots, m$. Then $m \leq 2N - 1$.
3. If there exists an $u \in \mathbb{R}^N$ such that $u \cdot v_i > 0$ for all $i = 1, \ldots, m$. Then $m \leq N$.

**Proof.** It is clear that (1) follows from (2). Suppose (2) is true. By viewing $-v_1$ as $u$, the conditions of (2) are all satisfied. Thus, we have $m - 1 \leq 2N - 1$, that is, $m \leq 2N$. 
To prove (2), we will use induction on \( N \). When \( N = 1 \), it is obvious that \( m \leq 2N - 1 = 1 \). Otherwise, by the conditions, there are non-zero real numbers \( u, v_1, v_2 \) such that \( u \cdot v_1 > 0 \) and \( u \cdot v_2 > 0 \), but \( v_1 \cdot v_2 < 0 \). This is impossible.

Now considering the result, we must be linearly dependant. Let \( S \subseteq \{1, 2, \ldots, m\} \) be a non-empty set of minimum size for which there is a relation \( \sum_{i \in S} a_i v_i = 0 \) with all \( a_i \neq 0 \). We claim that the \( a_i \)s must all be positive or all be negative. In fact, if not, we collect the terms with positive \( a_i \)s on one side and the terms with negative \( a_i \)s on the other. Then we get an equation \( \sum_{i \in S^+} a_i v_i = \sum_{j \in S^-} b_j v_j \) (which we denote by \( w \)) with \( a_i \) and \( b_j \) all are positive, where \( S^+ \) and \( S^- \) are disjoint non-empty sets and \( S^+ \cup S^- = S \). By the minimality of \( S \), \( w \neq 0 \). Thus, the inner product \( w \cdot w = 0 \). On the other hand, \( w \cdot w = (\sum_{i \in S^+} a_i v_i) \cdot (\sum_{j \in S^-} b_j v_j) = \sum_{i,j} (a_i b_j) (v_i \cdot v_j) \leq 0 \) since \( a_i b_j > 0 \) and \( v_i \cdot v_j \leq 0 \). This contradiction shows that \( a_i \)s must all be positive or all be negative. Following this, we actually can assume that \( a_i > 0 \) for all \( i \in S \) (otherwise, we can take \( a_i' = -a_i \) for a relation \( \sum_{i \in S} a_i' v_i = 0 \)).

Without loss of generality, we assume that \( S = \{1,2,\ldots,s\} \). By the linear dependance \( \sum_{i=1}^s a_i v_i = 0 \) with each \( a_i > 0 \) and minimality of \( S \), the vectors \( v_1, \ldots, v_s \) must span a subspace \( V \) of \( \mathbb{R}^N \) of dimension \( s - 1 \). Now, for \( l = s + 1, \ldots, m \), we have \( \sum_{i=1}^s a_i v_i \cdot v_l = 0 \) as \( \sum_{i=1}^s a_i v_i = 0 \). Since \( a_i > 0 \) for \( 1 \leq i \leq s \) and all \( v_i \cdot v_l \leq 0 \), we have that \( v_i \) is orthogonal to \( v_l \) for all \( i, l \) with \( 1 \leq i \leq s \) and \( s < l \leq m \). Similarly, we can prove that \( u \) is orthogonal to \( v_i \) for \( i = 1, 2, \ldots, s \). Therefore, the vectors \( v_{s+1}, \ldots, v_m \) and \( u \) are all in the dual space \( V^\perp \) which has dimension \( (N - s + 1) \). As \( s > 1 \), applying the induction hypothesis to these vectors, we have \( m - s \leq 2(N - s + 1) - 1 \). Thus, we have \( m \leq 2N - s + 1 \leq 2N - 1 \).

Now we prove (3). Suppose the result is not true, that is, \( m \geq N + 1 \). As above, \( v_1, \ldots, v_m \) must be linearly dependant \( \mathbb{R}^N \). Let \( S \subseteq \{1, 2, \ldots, m\} \) be a non-empty set of minimum size for which there is a relation \( \sum_{i \in S} a_i v_i = 0 \) with all \( a_i \neq 0 \). Again, we can assume that \( a_i > 0 \) for all \( a_i \in S \). From this, we have \( \sum_{i=1}^m a_i u \cdot v_i = 0 \). But this is impossible since for each \( i \) we have \( a_i > 0 \) and \( u \cdot v_i > 0 \). This contradiction shows \( m \leq N \).

Now we are ready to prove Proposition 9.3.1.

**Proof of Proposition 9.3.1.** We identify vectors in \( \mathbb{F}_q^n \) with vectors in \( \mathbb{R}^m \) in the following way: First, we set an ordering for the elements of \( \mathbb{F}_q \), and denote the elements as \( \alpha_1, \alpha_2, \ldots, \alpha_q \). Denote by \( \text{ord}(\beta) \) the order of element \( \beta \in \mathbb{F}_q \) under this ordering. For example, \( \text{ord}(\beta) = i \), if \( \beta = \alpha_i \). Then, each element \( \alpha_i \) \((1 \leq i \leq q)\) corresponds to the real unit vector of length \( q \) with a 1 in position \( i \) and 0 elsewhere.

Without loss of generality, we assume that \( r = (\alpha_q, \alpha_q, \ldots, \alpha_q) \). Denote by \( c_1, c_2, \ldots, c_m \) all the codewords of \( C \) that are in the Hamming ball \( B_t(r) \) where \( t = (1 - 1/q)(1 - \gamma)n \) for some \( 0 < \gamma < 1 \).

We view each vector in \( \mathbb{R}^m \) as having \( n \) blocks each having \( q \) components, where the \( n \) blocks correspond to the \( n \) positions of the vectors in \( \mathbb{F}_q^n \). For each \( l = 1, \ldots, q \), denote by \( e_l \) the unit vector of length \( q \) with 1 in the \( l \)th position and 0 elsewhere. For \( i = 1, 2, \ldots, m \), the vector in \( \mathbb{R}^m \) associated with
9.3. LIST DECODING BY SUDAN’S ALGORITHM

the codeword \( \mathbf{c}_i \), which we denote by \( \mathbf{d}_i \), has in its \( j \)th block the components of the vector \( \mathbf{c}_{\text{ord}(i)} \), where \( \mathbf{c}_{j} \) is the \( j \)th component of \( \mathbf{c} \). The vector in \( \mathbb{R}^{qn} \) associated with the word \( \mathbf{r} \in \mathbb{F}_{q}^{n} \), which we denote by \( \mathbf{s} \), is defined similarly.

Let \( \mathbf{1} \in \mathbb{R}^{qn} \) be the all 1’s vector. We define \( \mathbf{v} = \lambda \mathbf{s} + \frac{(1-\lambda)}{q} \mathbf{1} \) for \( 0 \leq \lambda \leq 1 \) that will be specified later. We observe that \( \mathbf{d}_i \) and \( \mathbf{v} \) are all in the space defined by the intersection of the hyperplanes \( \mathcal{P}_j = \{ \mathbf{x} \in \mathbb{R}^{qn} \mid \sum_{l=1}^{q} x_{j,l} = 1 \} \)

for \( j = 1, \ldots, n \). This fact implies that the vectors \( (\mathbf{d}_i - \mathbf{v}) \), for \( i = 1, \ldots, m \), are all in \( \mathcal{P} = \bigcap_{j=1}^{n} \mathcal{P}_j \) where \( \mathcal{P}_j = \{ \mathbf{x} \in \mathbb{R}^{qn} \mid \sum_{l=1}^{q} x_{j,l} = 0 \} \). As \( \mathcal{P} \) is an \( n(q-1) \)-dimensional subspace of \( \mathbb{R}^{qn} \), we have that the vectors \( (\mathbf{d}_i - \mathbf{v}) \), for \( i = 1, \ldots, m \), are all in an \( n(q-1) \)-dimensional space.

We will set the parameter \( \lambda \) so that the \( n(q-1) \)-dimensional vectors \( (\mathbf{d}_i - \mathbf{v}) \), \( i = 1, \ldots, m \), have all pairwise inner products less than 0. For \( i = 1, \ldots, m \), let \( t_i = d(\mathbf{c}_i, \mathbf{r}) \). Then \( t_i \leq t \) for every \( i \), and

\[
\begin{align*}
\mathbf{d}_i \cdot \mathbf{v} &= \lambda (\mathbf{d}_i \cdot \mathbf{s}) + \frac{(1-\lambda)}{q} (\mathbf{d}_i \cdot \mathbf{1}) \\
&= \lambda (n - t_i) + (1 - \lambda) \frac{n}{q} \geq \lambda (n - t) + (1 - \lambda) \frac{n}{q}, \quad (9.1)
\end{align*}
\]

\[
\begin{align*}
\mathbf{v} \cdot \mathbf{v} &= \lambda^2 n + 2(1 - \lambda) \frac{n}{q} + (1 - \lambda)^2 \frac{n}{q} \leq \lambda^2 n + (1 - \lambda)^2 \frac{n}{q} \leq \frac{n}{q} + \lambda^2 (1 - \frac{1}{q})n, \quad (9.2)
\end{align*}
\]

\[
\mathbf{d}_i \cdot \mathbf{d}_j = n - d(\mathbf{c}_i, \mathbf{c}_j) \leq n - d, \quad (9.3)
\]

which implies that for \( i \neq j \),

\[
(\mathbf{d}_i - \mathbf{v}) \cdot (\mathbf{d}_j - \mathbf{v}) \leq 2\lambda t - d + (1 - \frac{1}{q})(1 - \lambda)^2 n. \quad (9.4)
\]

Substituting \( t = (1 - 1/q)(1 - \gamma)n \) and \( d = (1 - 1/q)(1 - \beta)n \) into the above inequation, we have

\[
(\mathbf{d}_i - \mathbf{v}) \cdot (\mathbf{d}_j - \mathbf{v}) \leq (1 - \frac{1}{q}) n(\beta + \lambda^2 - 2\lambda \gamma). \quad (9.5)
\]

Thus, if \( \gamma > \frac{1}{2} (\frac{q}{q} + \lambda) \), we will have all pairwise inner products to be negative. We pick \( \lambda \) to minimize \( \frac{q}{q} + \lambda \) by setting \( \lambda = \sqrt{\beta} \). Now when \( \gamma > \sqrt{\beta} \), we have \( (\mathbf{d}_i - \mathbf{v}) \cdot (\mathbf{d}_j - \mathbf{v}) < 0 \) for \( i \neq j \).

9.3.2 Sudan’s algorithm

The algorithm of Sudan is applicable to Reed-Solomon codes, Reed-Muller codes, algebraic-geometric codes, and some other families of codes. In this sub-section, we give a general description of the algorithm of Sudan.

Consider the following linear code

\[
C = \{(f(P_1), f(P_2), \ldots, f(P_n)) \mid f \in \mathbb{F}_q[X_1, \ldots, X_m] \text{ and } \deg(f) < k \},
\]

where \( P_i = (x_{i1}, \ldots, x_{im}) \in \mathbb{F}_q^n \) for \( i = 1, \ldots, n \), and \( n \leq q^m \). Note that when \( m = 1 \), the code is a Reed-Solomon code or an extended Reed-Solomon code; when \( m \geq 2 \), it is a Reed-Muller code.
In the following algorithm and discussions, to simplify the statement we denote $(i_1, \ldots, i_m)$ by $(i)$, $X_1^{i_1} \cdots X_m^{i_m}$ by $X^i$, $H(X_1 + x_1, \ldots, X_m + x_m, Y + y)$ by $H(X, x, Y + y)$, $(i_1') \cdots (i_m')$ by $(i)'$, and so on.

**Algorithm 9.3.5** (The Algorithm of Sudan for List Decoding)

**INPUT:** The following parameters and a received word:
- Code length $n$ and the integer $k$;
- $n$ points in $\mathbb{F}_q^m$, namely, $P_i := (x_{i_1}, \ldots, x_{i_m}) \in \mathbb{F}_q^m, i = 1, \ldots, n$;
- Received word $r = (y_1, \ldots, y_n) \in \mathbb{F}_q^n$;
- Desired error-correcting radius $t$.

**Step 0:** Compute parameters $r, s$ satisfying certain conditions that we will give for specific families of codes in the following subsections.

**Step 1:** Find a nonzero polynomial $H(X, Y) = H(X_1, \ldots, X_m, Y)$ such that
- The $(1, \ldots, 1, k-1)$-weighted degree of $H(X_1, \ldots, X_m, Y)$ is at most $s$;
- For $i = 1, \ldots, n$, each point, $(x_i, y_i) = (x_{i_1}, \ldots, x_{i_m}, y_i) \in \mathbb{F}_q^{m+1}$, is a zero of $H(X, Y)$ of multiplicity $r$.

**Step 2:** Find all the $Y$-roots of $H(X, Y)$ of degree less than $k$, namely, $f = f(X_1, \ldots, X_m)$ of $\deg(f) < k$ such that $H(X, f)$ is a zero polynomial. For each such root, check if $f(P_i) = y_i$ for at least $n - t$ values of $i \in \{1, \ldots, n\}$. If so, include $f$ in the output list.

As we will see later, for an appropriately selected parameter $t$, the algorithm of Sudan can return a list containing all the $t$-consistent codewords in polynomial time, with the size of the output list bounded above by a polynomial in code length. So far, the best known record of error-correcting radius of list decoding by Sudan’s algorithm is the Johnson bound. In order to achieve this bound, prior to the actual decoding procedure, that is, Steps 1 and 2 of the algorithm above, a pair of integers, $r$ and $s$, should be carefully chosen, which will be used to find an appropriate polynomial $H(X, Y)$. The parameters $r$ and $s$ are independent on received words. They are used for the decoding procedure for any received word, as long as they are determined. The actual decoding procedure consists two steps: Interpolation and Root Finding. By the interpolation procedure, a nonzero polynomial $H(X, Y)$ is found. This polynomial contains all the polynomials which define the $t$-consistent codewords as its $Y$-roots. A $Y$-root of $H(X, Y)$ is a polynomial $f(X)$ satisfying that $H(X, f(X))$ is a zero polynomial. The root-finding procedure finds and returns all these $Y$-roots; thus all the $t$-consistent codewords are found.

We now explain the terms: weighted degree and multiplicity of a zero of a polynomial, which we have used in the algorithm. Given integers $a_1, a_2, \ldots, a_l$, the
Step 1 of the algorithm seeks a nonzero polynomial for all where is a zero of . The (, , , )-weighted degree of a polynomial is the maximal (, , , )-weighted degree of its terms.

For a polynomial , it is clear that 0 is a zero of , i.e., if and only if . We say 0 is a zero of multiplicity of , provided that and . For a nonzero value such that is a zero of multiplicity of , we say 0 is a zero of multiplicity of , provided that is a zero of multiplicity of . Similarly, for a multivariate polynomial , if and only if for all with , , , such that , then there exists with , , . A point is a zero of multiplicity of , if and only if for all with , , and for every , , , with , , such that , we have . The (, ) polynomial is nonzero polynomial in . It is easy to prove that (we leave the proof to the reader as an exercise), for = and ,

where

Step 1 of the algorithm seeks a nonzero polynomial such that its -weighted degree is at most , and for each , is a zero of multiplicity of . Based on the discussion above, this can be done by solving a system consisting of the following homogeneous linear equations in unknowns (which are coefficients of ),

for all , and for every with , and for every such that . And

for every with .

### 9.3.3 List decoding of Reed-Solomon codes

A Reed-Solomon code can be defined as a cyclic code generated by a generator polynomial (see Definition 8.1.1) or as an evaluation code (see Proposition 8.1.4). For the purpose of list decoding by Sudan’s algorithm, we view Reed-Solomon
codes as evaluation codes. Note that since any non-zero element \( \alpha \in \mathbb{F}_q \) satisfies \( \alpha^n = \alpha \), we have \( ev(X^n f(X)) = ev(f(X)) \) for any \( f(X) \in \mathbb{F}_q[X] \), where \( n = q - 1 \). Therefore,

\[
RS_k(n,1) = \{ f(x_1), f(x_2), \ldots, f(x_n) \mid f(X) \in \mathbb{F}_q[X], \deg(f) < k \}
\]

where \( x_1, \ldots, x_n \) are \( n \) distinct nonzero elements of \( \mathbb{F}_q \).

In this subsection, we consider the list decoding of Reed-Solomon codes \( RS_k(n,1) \) and extended Reed-Solomon codes \( ERS_k(n,1) \), that is, the case \( m = 1 \) of the general algorithm, Algorithm 9.3.5. As we will discuss later, Sudan’s algorithm can be adapted to list decoding generalized Reed-Solomon codes (see Definition 8.1.10).

The correctness and error-correcting capability of list-decoding algorithm are dependent on the parameters \( r \) and \( s \). In the following, we first prove the correctness of the algorithm for appropriate choice of \( r \) and \( s \). Then we calculate the error-correcting capability.

We can prove the correctness of the list-decoding algorithm by proving: (1) There exists a nonzero polynomial \( H(X,Y) \) satisfying the conditions given in Step 1 of Algorithm 9.3.5; and (2) All the polynomials \( f(X) \) satisfying the conditions in Step 2 are the \( Y \)-roots of \( H(X,Y) \), that is, \( Y - f(X) \) divides \( H(X,Y) \).

**Proposition 9.3.6** Consider a pair of parameters \( r \) and \( s \).

(1) If \( r \) and \( s \) satisfy

\[
n \left( \frac{r + 1}{2} \right) < \frac{s(s + 2)}{2(k - 1)}.
\]

Then, a nonzero polynomial \( H(X,Y) \) sought in Algorithm 9.3.5 does exist.

(2) If \( r \) and \( s \) satisfy

\[
r(n - t) > s.
\]

Then, for any polynomial \( f(X) \) of degree at most \( k - 1 \) such that \( f(x_i) = y_i \) for at least \( n - t \) values of \( i \in \{1,2,\ldots,n\} \), the polynomial \( H(X,Y) \) is divisible by \( Y - f(X) \).

**Proof.** We first prove (1). As discussed in the previous subsection, a nonzero polynomial \( H(X,Y) \) exists as long as we have a nonzero solution of a system of homogeneous linear equations in unknowns \( \alpha_i \), i.e., the coefficients of \( H(X,Y) \). A nonzero solution of the system exists, provided that the number of equations is strictly less than the number of unknowns. From the precise expression of the system (see the end of last subsection), it is easy to calculate the number of equations, which is \( n \left( \frac{r + 1}{2} \right) \). Next, we compute the number of unknowns. This number is equal to the number of monomials \( X^{i_1}Y^{i_2} \) of
(1, k - 1)-weighted degree at most s, which is
\[
\left\lfloor \frac{x}{k-1} \right\rfloor s - i_2(k-1) \\
\sum_{i_2=0}^{\left\lfloor \frac{x}{k-1} \right\rfloor} \sum_{i_1=0}^{x - \left\lfloor \frac{x}{k-1} \right\rfloor} 1 \\
= \sum_{i_2=0}^{\left\lfloor \frac{x}{k-1} \right\rfloor} (s + 1 - i_2(k-1)) \\
= (s + 1) \left( \left\lfloor \frac{x}{k-1} \right\rfloor + 1 \right) - \frac{k-1}{2} \left\lfloor \frac{x}{k-1} \right\rfloor \left( \left\lfloor \frac{x}{k-1} \right\rfloor + 1 \right) \\
\geq \left( \left\lfloor \frac{x}{k-1} \right\rfloor + 1 \right) \left( \frac{s}{2} + 1 \right) \\
\geq \frac{s}{k-1}. \quad \frac{s}{k-1}
\]
where \( \left\lfloor x \right\rfloor \) stands for the maximal integer less than or equal to \( x \). Thus, we proved (1).

We now prove (2). Suppose \( H(X, f(X)) \) is not zero polynomial. Denote \( h(X) = H(X, f(X)) \). Let \( I = \{i \mid 1 \leq i \leq n \text{ and } f(x_i) = y_i \} \). We have \( |I| \geq n - t \).

For any \( i = 1, \ldots, n \), as \( (x_i, y_i) \) is a zero of \( H(X, Y) \) of multiplicity \( r \), we can express \( H(X, Y) = \sum_{j_1+j_2 \geq r} \gamma_{j_1,j_2}(X-x_i)^{j_1}(Y-y_i)^{j_2} \).

Now, for any \( i \in I \), we have \( f(X) - y_i = (X-x_i)f_1(X) \) for some \( f_1(X) \), because \( f(x_i) = y_i \). Thus, we have
\[
h(X) = \sum_{j_1+j_2 \geq r} \gamma_{j_1,j_2}(X-x_i)^{j_1}(f(X)-y_i)^{j_2} = \sum_{j_1+j_2 \geq r} \gamma_{j_1,j_2}(X-x_i)^{j_1+j_2}(f_1(X))^{j_2}.
\]
This implies that \((X-x_i)^r\) divides \( h(X) \). Therefore, \( h(X) \) has a factor \( g(X) = \prod_{i \in I} (X-x_i)^r \), which is a polynomial of degree at least \( r(n-t) \).

On the other hand, since \( H(X, Y) \) has \((1, k-1)\)-weighted degree at most \( s \) and the degree of \( f(X) \) is at most \( k-1 \), the degree of \( h(X) \) is at most \( s \), which is less than \( r(n-t) \). This is impossible. Therefore, \( H(X, f(X)) \) is a zero polynomial, that is, \( Y - f(X) \) divides \( H(X, Y) \).

**Proposition 9.3.7** If \( t \) satisfies \((n-t)^2 > n(k-1)\), then there exist \( r \) and \( s \) satisfying both
\[
n(r+1) \left( \begin{array}{c} r+1 \\ 2 \end{array} \right) < \frac{s}{2(k-1)}
\]
and \( r(n-t) > s \).

**Proof.** Set \( s = r(n-t) - 1 \). It suffices to prove that there exists \( r \) satisfying
\[
r \left( \begin{array}{c} r+1 \\ 2 \end{array} \right) < \frac{(r(n-t)-1)(r(n-t)+1)}{2(k-1)}
\]
which is equivalent to the following inequivalent
\[
((n-t)^2 - n(k-1)) \cdot r^2 - n(k-1) \cdot r - 1 > 0.
\]
Since \((n-t)^2 - n(k-1) > 0\), any integer \( r \) satisfying
\[
r > \frac{n(k-1) + \sqrt{n^2(k-1)^2 + 4(n-t)^2 - 4n(k-1)}}{2(n-t)^2 - 2n(k-1)}
\]
satisfies the inequivalent above. Therefore, for the list-decoding algorithm to be correct it suffices by setting the integers \( r \) and \( s \) as
\[
r = \left\lfloor \frac{n(k-1) + \sqrt{n^2(k-1)^2 + 4(n-t)^2 - 4n(k-1)}}{2(n-t)^2 - 2n(k-1)} \right\rfloor + 1
\]
and \( s = r(n - t) - 1 \).

We give the following result, Theorem 9.3.8, which is a straightforward corollary of the two propositions.

**Theorem 9.3.8** For a \([n, k]\) Reed-Solomon or extended Reed-Solomon code the list-decoding algorithm, Algorithm 9.3.5, can correctly find all the codewords \( \mathbf{c} \) within distance \( t \) from the received word \( \mathbf{r} \), i.e., \( d(\mathbf{r}, \mathbf{c}) \leq t \), provided

\[
t < n - \sqrt{n(k - 1)}.
\]

**Remark 9.3.9** Note that for a \([n, k]\) Reed-Solomon code, the minimum distance \( d = n - k + 1 \) which implies that \( k - 1 = n - d \). Substituting this into the bound on error-correcting capability in the theorem above, we have

\[
\frac{t}{n} < 1 - \frac{d}{n}.
\]

This shows that the list-decoding of Reed-Solomon codes achieves the Johnson bound (see Theorem 9.3.2).

Regarding the size of the output list of the list-decoding algorithm, we have the following theorem.

**Theorem 9.3.10** Consider a \([n, k]\) Reed-Solomon or extended Reed-Solomon code. For any \( t < n - \sqrt{n(k - 1)} \) and any received word, the number of \( t \)-consist codewords is \( O(n^{3k}) \).

**Proof.** From Proposition 9.3.6, we actually have proved that the number \( N \) of the \( t \)-consist codewords is bounded from above by the degree \( \deg_Y(H(X,Y)) \). Since the \((1, k-1)\)-weighted degree of \( H(X,Y) \) is at most \( s \), we have \( N \leq \deg_Y(H(X,Y)) \leq \left\lfloor \frac{s}{k-1} \right\rfloor \). By the choices of \( r \) and \( s \), \( \frac{s}{k-1} = O\left(\frac{n(k-1)(n-t)}{k-1}\right) = O(n(n-t)) \). Corresponding to the largest permissible value of \( t \) for the \( t \)-consist codewords, we can choose \( n - t = \left\lfloor \sqrt{n(k-1)} \right\rfloor + 1 \). Thus,

\[
N = O(n(n-t)) = O(\sqrt{n^{3(k-1)}}) = O(n^{3k}).
\]

Let us analyze the complexity of the list decoding of a \([n, k]\) Reed-Solomon code. As we have seen, the decoding algorithm consists of two main steps. Step 1 is in fact reduced to a problem of solving a system of homogeneous linear equations, which can be implemented making use of Gaussian elimination with time complexity \( O\left(\left(\frac{s(s+2)}{2(k-1)}\right)^3\right) = O(n^3) \) where \( \frac{s(s+2)}{2(k-1)} \) is the number of unknowns of the system of homogeneous linear equations, and \( s \) is given as in Proposition 9.3.7.

The second step is a problem of finding \( Y \)-roots the polynomial \( H(X,Y) \). This can be implemented by using a fast root-finding algorithm proposed by Roth and Ruckenstein in time complexity \( O(nk) \).
9.3. LIST DECODING BY SUDAN’S ALGORITHM

9.3.4 List Decoding of Reed-Muller codes

We consider the list decoding of Reed-Muller codes in this subsection. Let $n = q^m$ and $P_1, \ldots, P_n$ be an enumeration of the points of $\mathbb{F}_q^n$. Recall that the $q$-ary Reed-Muller code $RM_q(u, m)$ of order $u$ in $m$ variables is defined as

$$RM_q(u, m) = \{(f(P_1), \ldots, f(P_n)) \mid f \in \mathbb{F}_q[X_1, \ldots, X_m], \deg(f) \leq u\}.$$ 

Note that when $m = 1$, the code $RM_q(u, 1)$ is actually an extended Reed-Solomon code.

From Proposition 8.4.4, $RM_q(u, m)$ is a subfield subcode of $RM_{q^m}(n - d, 1)$ where $d$ be the minimum distance of $RM_q(u, m)$, that is $RM_q(u, m) \subseteq RM_{q^m}(n - d, 1) \cap \mathbb{F}_q^n$.

Here $RM_{q^m}(n - d, 1)$ is an extended Reed-Solomon code over $\mathbb{F}_{q^m}$ of length $n$ and dimension $k = n - d + 1$. We now give a list-decoding algorithm for $RM_q(u, m)$ as follows.

Algorithm 9.3.11 (List-Decoding Algorithm for Reed-Muller Codes)

**INPUT:** Code length $n$ and a received word $r = (y_1, \ldots, y_n) \in \mathbb{F}_q^n$.

**Step 0:**
- **(1)** Compute the minimum distance $d$ of $RM_q(u, m)$ and a parameter $t = \lceil n - \sqrt{n(n-d)} - 1 \rceil$.
- **(2)** Construct the extension field $\mathbb{F}_{q^m}$ using an irreducible polynomial of degree $m$ over $\mathbb{F}_q$.
- **(3)** Generate the code $RM_{q^m}(n - d, 1)$.
- **(4)** Construct a parity check matrix $H$ over $\mathbb{F}_q$ for the code $RM_q(u, m)$.

**Step 1:** Using the list-decoding algorithm for Reed-Solomon codes over $\mathbb{F}_{q^m}$, find $\mathcal{L}^{(1)}$, the set of all codewords $c \in RM_{q^m}(n - d, 1)$ satisfying $d(c, r) \leq t$.

**Step 2:** For every $c \in \mathcal{L}^{(1)}$, check if $c \in \mathbb{F}_q^n$, if so, append $c$ to $\mathcal{L}^{(2)}$.

**Step 3:** For every $c \in \mathcal{L}^{(2)}$, check if $Hc^T = 0$, if so, append $c$ to $\mathcal{L}$. Output $\mathcal{L}$.

From Theorems 9.3.8 and 9.3.10, we have the following theorem.

**Theorem 9.3.12** Denote by $d$ the minimum distance of the $q$-ary Reed-Muller code $RM_q(u, m)$. Then $RM_q(u, m)$ is $(t, l)$-decodable, provided that $t < n - \sqrt{n(n-d)}$ and $l = O(\sqrt{(n-d)m^3})$.

The algorithm above correctly finds all the $t$-consistent codewords for any received vector $r \in \mathbb{F}_q^n$. 
Remark 9.3.13 Note that Algorithm 9.3.11 outputs a set of \( t \)-consistent codewords of the \( q \)-ary Reed-Muller code defined by the enumeration of points of \( \mathbb{F}_q^m \), say \( P_1, P_2, \ldots, P_n \), specified in Section 7.4.2. If \( \mathcal{RM}_q(u,m) \) is defined by another enumeration of the points of \( \mathbb{F}_q^m \), say \( P'_1, P'_2, \ldots, P'_n \), we can get the correct \( t \)-consistent codewords by the following steps: (1) Find the permutation \( \pi \) such that \( P_i = P'_{\pi(i)} \), \( i = 1, 2, \ldots, n \), and the inverse permutation \( \pi^{-1} \). (2) Let \( r^* = (r_{\pi(1)}, r_{\pi(2)}, \ldots, r_{\pi(n)}) \). Then, go to Steps 0-2 of Algorithm 9.3.11 with \( r^* \). (3) For every codeword \( c = (c_1, c_2, \ldots, c_n) \in \mathcal{L} \), let \( \pi^{-1}(c) = (c_{\pi^{-1}(1)}, c_{\pi^{-1}(2)}, \ldots, c_{\pi^{-1}(n)}) \). Then, \( \pi^{-1}(\mathcal{L}) = \{ \pi^{-1}(c) \mid c \in \mathcal{L} \} \) is the set of \( t \)-consistent codewords of \( \mathcal{RM}_q(u,m) \).

Now, let us consider the complexity of Algorithm 9.3.11. In Step 0, to construct the extension field \( \mathbb{F}_{q^m} \), it is necessary to find an irreducible polynomial \( g(x) \) of degree \( m \) over \( \mathbb{F}_q \). It is well known that there are efficient algorithms for finding irreducible polynomials over finite fields. For example, a probabilistic algorithm proposed by V. Shoup in 1994 can find an irreducible polynomial of degree \( m \) over \( \mathbb{F}_q \) with expected number of \( \mathcal{O}((m^2 \log m + m \log q) \log m \log m \log m) \) field operations in \( \mathbb{F}_q \).

To generate the Reed-Solomon code \( GRS_{n-d+1}(a,1) \) over \( \mathbb{F}_{q^m} \), we need to find a primitive element of \( \mathbb{F}_{q^m} \). With a procedure by I.E. Shparlinski in 1993, a primitive element of \( \mathbb{F}_{q^m} \) can be found in deterministic time \( \mathcal{O}((q^m)^{1/4+\epsilon}) = \mathcal{O}(n^{1/4+\epsilon}) \), where \( n = q^m \) is the length of the code, \( \epsilon \) denotes an arbitrary positive number.

Step 1 of Algorithm 9.3.11 can be implemented using the list-decoding algorithm in for Reed-Solomon code \( GRS_{n-d+1}(a,1) \) over \( \mathbb{F}_{q^m} \). From the previous subsection, it can be implemented to run in \( \mathcal{O}(n^3) \) field operations in \( \mathbb{F}_{q^m} \).

So, the implementation of Algorithm 9.3.11 requires \( \mathcal{O}(n) \) field operations in \( \mathbb{F}_q \) and \( \mathcal{O}(n^3) \) field operations in \( \mathbb{F}_{q^m} \).

9.3.5 Exercises

9.3.1 Let \( P(X_1, \ldots, X_l) = \sum_{i_1, \ldots, i_l} a_{i_1, \ldots, i_l} X_1^{i_1} \cdots X_l^{i_l} \) be a polynomial in variables \( X_1, \ldots, X_l \) with coefficients \( a_{i_1, \ldots, i_l} \) in a field \( \mathbb{F} \). Prove that for any \( (a_1, \ldots, a_l) \in \mathbb{F}^l \),

\[
P(X_1 + a_1, \ldots, X_l + a_l) = \sum_{j_1, \ldots, j_l} \beta_{j_1, \ldots, j_l} X_1^{j_1} \cdots X_l^{j_l}
\]

where

\[
\beta_{j_1, \ldots, j_l} = \sum_{j_1 \geq j_1} \cdots \sum_{j_l \geq j_l} \binom{j_1}{j_1} \cdots \binom{j_l}{j_l} a_{j_1, \ldots, j_l} a_1^{j_1-j_1} \cdots a_l^{j_l-j_l}.
\]

9.4 Notes

Many cyclic codes have error-correcting pairs, for this we refer to Duursma and Kötter [53, 54].
The algorithms of Berlekamp-Massey [11, 79] and Sugiyama [118] both have $O(t^2)$ as an estimation of the complexity, where $t$ is the number of corrected errors. In fact the algorithms are equivalent as shown in [50, 65]. The application of a fast computation of the gcd of two polynomials in [4, Chap. 16, §8.9] in computing a solution of the key equation gives as complexity $O(t \log^2(t))$ by [69, 104].
Chapter 10

Cryptography

Stanislav Bulygin

This chapter is aiming at giving an overview of topics from cryptography. In particular, we cover symmetric as well as asymmetric cryptography. When talking about symmetric cryptography, we concentrate on a notion of a block cipher, as a mean to implement symmetric cryptosystems in practical environments. Asymmetric cryptography is represented by the RSA and El Gamal cryptosystems, as well as code-based cryptosystems due to McEliece and Niederreiter. We also take a look at other aspects such as authentication codes, secret sharing, and linear feedback shift registers. The material of this chapter is quite basic, but we elaborate more on several topics. Especially we should connections to codes and related structures where applicable. The basic idea of algebraic attacks on block ciphers is considered in the next chapter, Section 11.3.

10.1 Symmetric cryptography and block ciphers

10.1.1 Symmetric cryptography

This section is devoted to the Symmetric cryptosystems. The idea behind these is quite simple and thus was basically known for quite along time. The task is to convey a secret between two parties, called traditionally Alice and Bob, so that figuring the secret out is not possible without knowledge of some additional information. This additional information is called a secret key and is supposed to be known only to the two communicating parties. The secrecy of the transmitted message rests entirely upon the knowledge of this secret key, and thus if an adversary or an eavesdropper, traditionally called Eve, is able to find out the key, then the whole secret communication is corrupted. Now let us take a look at the formal definition.

Definition 10.1.1 The symmetric cryptosystem is defined by the following data:

- The plaintext space $\mathcal{P}$ and the ciphertext space $\mathcal{C}$. 
• \( \{ E_e : \mathcal{P} \to \mathcal{C} | e \in K \} \) and \( \{ D_d : \mathcal{C} \to \mathcal{P} | d \in K \} \) are the sets of encryption and decryption transformations, which are bijections from \( \mathcal{P} \) to \( \mathcal{C} \) and from \( \mathcal{C} \) to \( \mathcal{P} \) resp.

• The above transformations are parametrized by the key space \( K \).

• Given an associated pair \((e, d)\), so that a property \( \forall p \in \mathcal{P} : D_d(E_e(p)) = p \) holds, knowing \( e \) it is "computationally easy" to find out \( d \) and vise versa.

The pair \((e, d)\) is called secret key. Moreover, \( e \) is called the encryption key and \( d \) is called the decryption key.

Note that often the counterparts \( e \) and \( d \) coincide. This gives a reason for the name "symmetric". There exist also cryptosystems in which knowledge of an encryption key \( e \) does not reveal (i.e. it is "computationally hard" to find) an associated decryption key \( d \). So encryption keys can be made public, and such cryptosystems are called asymmetric or public see Section 10.2.

Of course, one should specify exactly what are \( \mathcal{P}, \mathcal{C}, K \) and the transformations.

Let us take a look at a concrete example.

**Example 10.1.2** The first use of a symmetric cryptosystem is conventionally attributed to Julius Caesar. He used the following cryptosystem for communication with his generals, which is historically called Caesar cipher. Let \( \mathcal{P} \) and \( \mathcal{C} \) be the sets of all strings composed of letters from the English (Latin for Caesar) alphabet \( \mathcal{A} = \{A, B, C, \ldots, Z\} \). Let \( K = \{0, 1, 2, \ldots, 25\} \). Now an encryption transformation \( E_e \) given a plaintext \( p = (p_1, \ldots, p_n) , p_i \in \mathcal{A} , i = 1, \ldots, n \) does the following. For each \( i = 1, \ldots, n \) one determines a position of \( p_i \) in the alphabet \( \mathcal{A} \) ("A" being 0, "B" being 1, \ldots, "Z" being 25). Next one finds a letter in \( \mathcal{A} \) that stands \( e \) positions to the left, thus finding a letter \( c_i \); one needs to overlap if the beginning of \( \mathcal{A} \) is reached. So with the enumeration of \( \mathcal{A} \) as above, we have \( c_i = p_i - e \mod 26 \). In this way a ciphertext \( c = (c_1, \ldots, c_n) \) is obtained. Decryption key is given by \( d = -e \mod 26 \), or, equivalently, for decryption one needs to shift letters \( e \) positions to the right.

Julius Caesar used \( e = 3 \) for his cryptosystem. Let us consider an example. For the plaintext \( p = "BRUTUS IS AN ASSASSIN" \), the ciphertext (if we ignore spaces during the encryption) looks like \( c = "YORQRP FP XK XOOXOOFK" \). To decrypt one simply shifts back 3 positions to the right.

### 10.1.2 Block ciphers. Simple examples

The above is a simple example of a so-called substitution cipher, which is in turn an instance of a block cipher. Block ciphers, among other things, provide a practical realization of symmetric cryptosystems. They can also be used for constructing other cryptographic primitives, like pseudorandom number generators, authentication codes (Section 10.3), hash functions. The formal definition follows.

**Definition 10.1.3** The \( n \)-bit block cipher is defined as a mapping \( E : \mathcal{A}^n \times K \to \mathcal{A}^n \), where \( \mathcal{A} \) is an alphabet set and \( K \) is the key space and for each \( k \in K \) the mapping \( E(\cdot, k) =: E_k : \{0,1\}^n \to \{0,1\}^n \) is invertible. \( E_k \) is the encryption transformation for the key \( k \), and \( E_k^{-1} = D_k \) is the decryption transformation. If \( E_k(p) = c \), then \( c \) is the ciphertext of the plaintext \( p \) under the key \( k \).
It is common to work with the binary alphabet, i.e. $A = \{0, 1\}$. In such case, ideally we would like to have a block cipher that is random in the sense that it implements all $2^n!$ bijections from $\{0, 1\}^n$ to $\{0, 1\}^n$. In practice, though, it is quite expensive to have such a cipher. So when designing a block cipher we care that it behaves like a random one, i.e. for a randomly chosen key $k \in \mathcal{K}$ the encryption transformation $E_k$ should appear random. If one is able to find distinctions of $E_k$, where $k$ is in some subset $\mathcal{K}_{\text{weak}}$, from the random transformation, then it is an evidence of a weakness of the cipher. Such subset $\mathcal{K}_{\text{weak}}$ is called the subset of weak keys; we will turn back to this later when talking about DES.

Now we present several simple examples of block ciphers. We consider permutation and substitution ciphers that were used quite intensively in the past (see Notes) and some fundamental ideas thereof appear also in the modern ciphers.

**Example 10.1.4** *(Permutation or transposition cipher)* The idea of this cipher is to partition the plaintext into blocks and perform a permutation of elements in that block. More formally, partition the plaintext into blocks of the form $p = p_1 \ldots p_t$ and then permute: $c = E_k(p) = p_{k(1)}, \ldots, p_{k(t)}$. A number $t$ is called a period of the cipher. The key space $\mathcal{K}$ now is the set of all permutations of $\{1, \ldots, t\} : \mathcal{K} = S_t$. For example let the plaintext be $p = \text{"CODING AND CRYPTO"}$, let $t = 5$, and $k = (4, 2, 5, 3, 1)$. If we remove the spaces and partition $p$ into 3 blocks we obtain $c = \text{"INCODDCGAN}\text{TORYP"}$. Used alone permutation cipher does not provide good security (see below), but in combination with other techniques it is used also in modern ciphers to provide diffusion in a ciphertext.

**Example 10.1.5** We can use Sage system to run the previous example. The code looks as follows.

```python
> S = AlphabeticStrings()
> E = TranspositionCryptosystem(S,5)
> K = PermutationGroupElement('(4,2,5,3,1)')
> L = E.inverse_key(K)
> M = S("CODINGANDCRYPTO")
> e = E(K)
> c = E(L)
> e(M)
INCODDCGAN\text{TORYP}
> c(e(M))
CODINGANDCRYPTO
```

One can also choose a random key for encryption:

```python
> KR = E.random_key()
> KR
(1,4,2,3)
> LR = E.inverse_key(KR)
> LR
(1,3,2,4)
> eR = E(KR)
> cR = E(LR)
> eR(M)
IDCONDNGACTPRYO
```
Example 10.1.6 (Substitution cipher) An idea behind monoalphabetic substitution cipher is to provide substitution of every symbol in a plaintext by some other symbol from a chosen alphabet. Formally, let \( \mathcal{A} \) be the alphabet, so that plaintexts and ciphertexts are composed of symbols from \( \mathcal{A} \). For the plaintext \( p = p_1 \ldots p_n \) the ciphertext \( c \) is obtained as \( c = E_k(p) = k(p_1), \ldots, k(p_n) \). The key space now is the set of all permutations of \( \mathcal{A} : K = S_{|\mathcal{A}|} \). In Example 10.1.2 we have already seen an instance of such a cipher. There \( k \) was chosen to be \( k = (23, 24, 25, 0, 1, \ldots, 21, 22) \). Again, used alone monoalphabetic cipher is insecure, but as a basic idea is used in modern ciphers to provide confusion in a ciphertext.

There is also a polyalphabetic substitution cipher. Let the key \( k \) be defined as a sequence of permutations on \( \mathcal{A} : k = (k_1, \ldots, k_t) \), where \( t \) is the period. Then every \( t \) symbols of the plaintext \( p \) are mapped to \( t \) symbols of the ciphertext \( c \) as \( c = k_1(p_1), \ldots, k_t(p_t) \). Simplifying \( k_i \) to shifting by \( l_i \) symbols to the right we obtain \( c_i = p_i + l_i \mod |\mathcal{A}| \). Such cipher is called simple Vigenère cipher.

Example 10.1.7 The Sage-code for a substitution cipher encryption is given below.

```sage
> S = AlphabeticStrings()
> E = SubstitutionCryptosystem(S)
> K = E.random_key()
> K
ZYNJQHLBSPEDCMAXVRUTIKGF
> L = E.inverse_key(K)
> M = S("CODINGANDCRYPTO")
> e = E(K)
> e(M)
NDJSMLZMJNWGARD
> c = E(L)
Here the string ZYNJQHLBSPEDCMAXVRUTIKGF shows the permutation of the alphabet. Namely, the letter A is mapped to Z, the letter B is mapped to Y etc.
One can also provide the permutation explicitly as follows
> K = S(’MHKENLQSCDFGBIAYOUTZXJVWPR’)
> e = E(K)
> e(M)
KAECIQAEMKPRYZA
A piece of code for working with the simple Vigenère cipher is provided below.
> S = AlphabeticStrings()
> E = VigenereCryptosystem(S,15)
> K = S(’XSPUDFOQLRMRDJS’)
> L = E.inverse_key(K)
> M = S("CODINGANDCRYPTO")
> e = E(K)
> e(M)
ZGSCQLODOTDPSCG
> c = E(L)
> c(e(M))
```
Table 10.1: Frequencies of the letters in the English language

<table>
<thead>
<tr>
<th>Letter</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>11.1607%</td>
</tr>
<tr>
<td>A</td>
<td>8.4966%</td>
</tr>
<tr>
<td>R</td>
<td>7.5809%</td>
</tr>
<tr>
<td>T</td>
<td>7.1635%</td>
</tr>
<tr>
<td>N</td>
<td>6.6544%</td>
</tr>
<tr>
<td>S</td>
<td>5.7351%</td>
</tr>
<tr>
<td>L</td>
<td>4.5388%</td>
</tr>
<tr>
<td>C</td>
<td>3.3844%</td>
</tr>
<tr>
<td>D</td>
<td>3.1671%</td>
</tr>
</tbody>
</table>

Note that here the string XSPUDFOQLMRDJS defines 15 permutations: one per position. Namely, every letter is an image of the letter A at that position. So at the first position A is mapper to X (therefore, e.g. B is mapped to Y), at the second position A is mapped to S and so on.

The ciphers above used alone do not provide security, as has already been mentioned. One way to break such ciphers is to use statistical methods. For the permutation ciphers note that they do not change frequency of occurrence of each letter of an alphabet. Comparing frequencies obtained from a ciphertext with a frequency distribution of the language used, one can figure out that he/she deals with the ciphertext obtained with a permutation cipher. Moreover, for cryptanalysis one may try to look for anagrams - words in which letters are permuted. If the eavesdropper is able to find such anagrams and solve them, then he/she is pretty close to breaking such a cipher (Exercise 10.1.1). Also if the eavesdropper has an access to an encryption device and is able to produce ciphertexts for plaintext of his/her choice (chosen-plaintext attack), then he/she can simply choose plaintexts, such that figuring out the period and the permutation becomes easy.

For monoalphabetic substitution ciphers one also notes that although letters are changed, the frequency with which they occur does not change. So the eavesdropper may compare frequencies in a long-enough ciphertext with a frequency distribution of the language used and thus figure out, how letters of an alphabet were mapped to obtain a ciphertext. For example for the English alphabet one may use frequency analysis of words occurring in "Concise Oxford Dictionary" (http://www.askoxford.com/asktheexperts/faq/aboutwords/frequency), see Table 10.1. Note that since positions of the symbols are not altered, the eavesdropper may not only look at frequencies of the symbols, but also for combinations of symbols. In particular look for pieces of a ciphertext that correspond to frequently used words like "the", "we", "in", "at" etc. For polyalphabetic ciphers one needs to find out the period first. This can be done by the so-called Kasiski method. When the period is determined, one can proceed with the
frequency analysis as above performed separately for all sets of positions that stand at distance $t$ at each other.

10.1.3 Security issues

As we have seen, block ciphers provide us with a mean to convey secret messages as per symmetric scheme. It is clear that an eavesdropper tries to get insight in this secret communication. The question that naturally arises is "What does it mean to break a cipher?" or "When the cipher is considered to be broken?". In general we consider a cipher to be totally broken if the eavesdropper is able to recover the secret key, thus compromising the whole secret communication. We consider a cipher to be partially broken if the eavesdropper is able to recover (a part of) a plaintext from a given ciphertext, thus compromising the part of the communication. In order to describe actions of the eavesdropper more formally, different assumptions on the eavesdropper’s abilities and scenarios of attacks are introduced.

Assumptions:

- The eavesdropper has an access to all ciphertexts that are transmitted through the communication channel. He/she is able to extract these ciphertexts and use them further for his/her disposal.
- The eavesdropper has a full description of the block cipher itself, i.e. he/she is aware of how the encryptions constituting the cipher act.

The first assumption is natural to assume, since communication in the modern world (e.g. via the Internet) assumes huge amount of information to be transmitted between an enormous variety of parties. Therefore, it is impossible to provide secure channels for all such transmissions. The second one is also quite natural, as for most block ciphers that are proposed in the recent time there is a full description publicly available either as a legitimate standard or as a paper/report.

Attack scenarios:

- **ciphertext-only**: The eavesdropper does not have any additional information, only an intercepted ciphertext.
- **known-plaintext**: Some amount of plaintext-ciphertext pairs encrypted with one particular yet unknown key are available to the eavesdropper.
- **chosen-plaintext and chosen-ciphertext**: The eavesdropper has an access to plaintext-ciphertext pairs with a specific eavesdropper’s choice of plaintexts and ciphertexts resp.
- **adaptive chosen-plaintext and adaptive chosen-ciphertext**: The choice of the special plaintexts resp. ciphertext in the previous scenario depends on some prior processing of pairs.
- **related-key**: The eavesdropper is able to do encryptions with unknown yet related keys, with the relations known to the eavesdropper.
10.1. SYMMETRIC CRYPTOGRAPHY AND BLOCK CIPHERS

Note that the last three attacks are quite hard to realize in a practical environment and sometimes even impossible. Nevertheless, studying these scenarios provides more insight in the security properties of a considered cipher. When undertaking an attack on a cipher one thinks in terms of complexity. Recall from Definition 6.1.4 that there is always time or processing as well as memory or storage complexities. Another type of complexity one deals with here is data complexity, which is an amount of pre-knowledge (e.g. plain-/ciphertexts) needed to mount an attack.

The first thing to think of when designing a cipher is to choose block/key length, so that brute force attacks are not possible. Let us take a closer look here. If the eavesdropper is given $2^n$ plaintext-ciphertext pairs encrypted with one secret key, then he/she entirely knows the encryption function for a given secret key. This implies that $n$ should not be chosen too small, as then simply composing a codebook of associated plaintexts-ciphertexts is possible. For modern block ciphers, block length 128 bits is common. On the other side, if the eavesdropper is given just one plaintext-ciphertext pair $(p, c)$, he/she may proceed as follows. Try every key from $K$ (assume now that $K = \{0, 1\}^l$) until he/she finds $k$ that maps $p$ to $c$: $E_k(p) = c$. Validate $k$ with another pair (or several pairs) $(p', c')$, i.e. check whether $E_k(p') = c'$. If validation fails, then discard $k$ and move further in $K$. One expects to find a valid key after searching through a half of $\{0, 1\}^l$, i.e. after $2^{l-1}$ trials. This observation implies that key space should not be too small, as then exhaustive search of such kind is possible. For modern ciphers key lengths of 128, 192, 256 bits are applied. Smaller block-length, like 64 bits, are also employed in light-weight ciphers that are used for resource constraint devices.

Let us now discuss two main types of security that exist out there for cryptosystems in general.

**Definition 10.1.8**

- **Computational security.** Here one considers a cryptosystem to be secure (computationally) if the number of operations needed to break the cryptosystem is so large that cannot be executed in practice, similarly for memory. Usually one measures such a number by the best attacks available for a given cryptosystem and thus claiming computational security. Another similar idea is to show that breaking a given cryptosystem is equivalent to solving some problem, that is believed to be hard. Such security is called provable security or sometimes reductionist security.

- **Unconditional security.** Here one assumes that an eavesdropper has an unlimited computational power. If one is able to prove that even having this unlimited power, an eavesdropper is not able to break a given cryptosystem, then it is said that the cryptosystem is unconditionally secure or that it provides perfect secrecy.

Before going to examples of block ciphers, let us take a look at the security criteria that are usually used, when estimating security capabilities of a cipher.

**Security criteria:**

- **state-of-the-art security level:** One gets more confident in a cipher’s security if known up-to-date attacks, both generic and specialized, do not break the cipher faster, than the exhaustive search. The more such attacks are considered, the more confidence one gets. Of course, one cannot
be absolutely confident here as new, unknown before, attacks may appear that would impose real threat.

- **block and key size**: As we have seen above, small block and key sizes make brute force attacks possible, so in this respect, longer blocks and key provide more security. On the other hand, longer blocks and key imply more costs in implementing such a cipher, i.e. encryption time and memory consumption may rise considerably. So there is a trade-off between security and ease/speed of an implementation.

- **implementation complexity**: In addition to the previous point, one should also take care for efficient implementation of encryption/decryption mappings depending on an environment. For example different methods may be used for hardware and software implementations. Special care is to be taken, when one deals with hardware units with very limited memory (e.g. smartcards).

- **others**: Things like data expansion and error propagation also play a role in applications and should be taken into account accordingly.

### 10.1.4 Modern ciphers. DES and AES

In Section 10.1.2 we considered basic ideas for block ciphers. Next, let us consider two examples of modern block ciphers. The first one - DES (Data Encryption Standard) - was proposed in 1976 and was used until late 1990s. Due to short key length, it became possible to implement an exhaustive search attack, so the DES was no longer secure. In 2001 the cipher Rijndael proposed by Belgian cryptographers Joan Daemen and Vincent Rijmen was adopted as the Advanced Encryption Standard (AES) in the USA and is now widely used for protecting classified governmental documents. In commerce AES also became the standard *de facto*.

We start with DES, which is an instance of a Feistel cipher, which is in turn an iterative cipher.

**Definition 10.1.9** An iterative block cipher is a block cipher which performs sequentially a certain key dependant transformation $F_k$. This transformation is called round transformation and the number of rounds $N_r$ is a parameter of an iterative cipher. It is also common to expand the initial private key $k$ to subkeys $k_i, i=1,\ldots,N_r$, where each $k_i$ is used as a key for $F$ at round $i$. A procedure for obtaining the subkeys from the initial key is called a key schedule. For each $k_i$ the transformation $F$ should be invertible to allow decryption.

**DES**

**Definition 10.1.10** A Feistel cipher is an iterative cipher, where encryption is done as follows. Divide $n$-bit plaintext $p$ into two parts - left and right - $(l_0, r_0)$ ($n$ is assumed to be even). A transformation $f : \{0, 1\}^{n/2} \times K' \rightarrow \{0, 1\}^{n/2}$ is chosen ($K'$ may differ from $K$). The initial secret key is expanded to obtain the subkeys $k_i, i=1,\ldots,N_r$. Then for every $i=1,\ldots,N_r$ a pair $(l_i, r_i)$ is obtained from the previous pair $(l_{i-1}, r_{i-1})$ as follows: $l_i = r_{i-1}, r_i = l_{i-1} \oplus f(r_{i-1}, k_i)$. Here "⊕" means bitwise addition of $\{0, 1\}$-vectors. The ciphertext is taken as $(r_{N_r}, l_{N_r})$ rather than $(l_{N_r}, r_{N_r})$. 

On Figure 10.1 one can see the scheme of Feistel cipher encryption. Note that $f(\cdot, k_i)$ need not be invertible (Exercise 10.1.5). Decryption is done analogously with the reverse order of subkeys: $k_N, \ldots, k_1$. DES is a Feistel cipher that operates on 64-bit blocks and needs a 56-bit key. Actually the key is given initially in 64 bits, of which 8 bits can be used as parity checks. DES has 16 rounds. The subkeys $k_1, \ldots, k_{16}$ are 48 bits long. The transformation $f$ from Definition 10.1.10 is chosen as

$$f(r_i, k_i) = P(S(E(r_i) \oplus k_i)).$$

(10.1)

Here $E : \{0, 1\}^{32} \rightarrow \{0, 1\}^{48}$ is an expansion transformation that expands 32-bit vector to a 48-bit one in order to fit the size of $k_i$ when doing bitwise addition. Next $S$ is a substitution transformation that acts as follows. First
divide 48-bit vector \( E(r_{i-1}) \oplus k_i \) into 8 6-bit blocks. For every block perform a (non-linear) substitution that takes 6 bits and outputs 4 bits. Thus at the end one has a 32-bit vector obtained by a concatenation of the results from the substitution \( S \). The substitution \( S \) is an instance of an S-box, a carefully chosen non-linear transformation that makes relation between its input and output complex, thus adding confusion to the encryption transformation (see below for the discussion). Finally, \( P \) is a permutation of a 32-bit vector.

**Algorithm 10.1.11 (DES encryption)**

**Input:** The 64-bit plaintext \( p \) and the 64-bit key \( k \).

**Output:** The 64-bit ciphertext \( c \) corresponding to \( p \).

1. Use the parity check bits \( k_8, k_{16}, \ldots, k_{64} \) to detect errors in 8-bit subblocks of \( k \).
   - If no errors are detected then obtain 48-bit subkeys \( k_1, \ldots, k_{16} \) from \( k \) using key schedule.
2. Take \( p \) and perform an initial permutation \( IP \) to \( p \). Divide the 64-bit vector \( IP(p) \) into halves \( (l_0, r_0) \).
3. For \( i = 1, \ldots, 16 \) do
   - \( l_i := r_{i-1} \).
   - \( f(r_{i-1}, k_i) := P(S(E(r_{i-1}) \oplus k_i)), \) with \( S, E, P \) as explained after (10.1).
   - \( r_i := l_{i-1} \oplus f(r_{i-1}, k_i) \).
4. Interchange the last halves \( (l_{16}, r_{16}) \rightarrow (r_{16}, l_{16}) = c' \).
5. Perform the permutation inverse to the initial one to \( c' \), the result is the ciphertext \( c := IP^{-1}(c') \).

Let us now give a brief overview of DES properties. First of all, we mention two basic features that any modern block cipher provides and definitely should be taken into account when designing a block cipher.

- **Confusion.** When an encryption transformation of a block cipher makes relations among a plaintext, a ciphertext, and a key, as complex as possible, it is said that such cipher adds to the encryption process. Confusion is usually achieved by non-linear transformations realized by S-boxes.

- **Diffusion.** When an encryption transformation of a block cipher makes every bit of a ciphertext dependent on every bit of a plaintext and on every bit of a key, it is said that such cipher adds to the encryption process. Diffusion is usually achieved by permutations. See Exercise 11.3.1 for a concrete example.

Empirically, DES has the above features, so in this respect appears to be rather strong. Let us discuss some other features of DES and some attacks that exist out there for DES. Let \( DES_k(\cdot) \) be an encryption transformation defined by DES as per Algorithm 10.1.11 for a key \( k \). DES has 4 weak keys, in this context these are the keys \( k \) for which \( DES_k(DES_k(\cdot)) \) is the identity mapping, which,
of course, violates the criteria mentioned above. Moreover for each of these weak keys DES has \(2^{32}\) fixed points, i.e. plaintexts \(p\) such that \(DES_k(p) = p\). There are 6 pairs of semi-weak keys (dual keys), i.e. pairs \((k_1, k_2)\) such that \(DES_{k_1}(DES_{k_2}(\cdot))\) is the identity mapping. Similarly to weak keys, 4 out of 12 semi-weak keys have \(2^{32}\) anti-fixed points, i.e. plaintexts \(p\) such that \(DES_k(p) = \bar{p}\), where \(\bar{p}\) is the bitwise complement of \(p\). It is also known that DES encryptions are not closed under composition, i.e. do not form a group. This is quite important as otherwise using multiple DES encryptions would be less secure, than otherwise is believed.

If the eavesdropper is able to work under huge data complexity, several known-plaintext attacks become possible. The most well-known of them related to DES are linear and differential cryptanalysis. The linear cryptanalysis was proposed by Mitsuru Matsui in early 1990s and is based on the idea of an approximation of a cipher with an affine function. In order to implement this attack for DES one needs \(2^{43}\) known plaintext-ciphertext pairs. Existence of such an attack is an evidence of theoretic weakness of DES. Similar observation is applicable to the differential cryptanalysis. The idea of this general method is to carefully explore how differences in inputs to certain parts of an encryption transformation affects outputs of these parts. Usually the focus is on the S-Boxes. An eavesdropper is trying to find a bias in differences distribution, which would allow him/her to distinguish a cipher from a random permutation. In the DES-situation the eavesdropper needs \(2^{55}\) known or \(2^{47}\) chosen plaintext-ciphertext pairs in order to mount such an attack. These attacks do not bear any practical threat to DES. Moreover, performing exhaustive search on the entire key space of size \(2^{56}\) is practically faster, than the attacks above.

AES

Next we present the basic description and properties of the Advanced Encryption Standard (AES). AES is a successor of DES and was proposed, because DES was not considered to be secure anymore. A new cipher for the Standard should have had larger key/block size and be resistant to linear and differential cryptanalysis that imposed theoretically a threat for the DES. The cipher Rijndael adapted for the Standard satisfies these demands. It operates on blocks of length 128 bits and keys of length 128, 192, or 256 bits. We will concentrate on AES, which employs keys of length 128 bits - the most common setting used. AES is an instance of a substitution-permutation network. We give a definition next.

**Definition 10.1.12** The substitution-permutation network (SP-network) is the iterative block cipher with layers of S-boxes interchanged with layers of permutations (or P-Boxes), see Figure 10.2. It is required that S-boxes are invertible.

Note that in the definition of an SP-network we demand that S-boxes are invertible transformations in contrast to Feistel ciphers, where S-boxes do not have to be invertible, see the discussion after Definition 10.1.10. Sometimes invertibility of S-boxes is not required, which makes the definition wider. If we recall the notions of confusion and diffusion, we see that SP-networks exactly reflect these notions: S-Boxes provide local confusion and then bit permutations of affine maps provide diffusion.
The description of the AES follows. As has already been said, AES operates on 128-bit blocks and 128-bit keys (standard version). For convenience these 128-bit vectors are considered as $4 \times 4$ arrays of bytes (8-bits). AES-128 (key length is 128 bits) has 10 rounds. We know that AES is an SP-network, so let us describe its substitution and diffusion (permutation) layers.

AES substitution layer is based on 16 S-boxes each acting on a separate byte of the square representation. In AES terminology the S-box is called SubBytes. One S-box performs its substitution in three steps:

1. **Inversion::** Consider an input byte $b_{\text{input}}$ (a $\{0,1\}$-vector of length 8) as an element of $\mathbb{F}_{256}$. This is done via the isomorphism $\mathbb{F}_2[a]/(a^8 + a^4 + a^3 + a + 1) \cong \mathbb{F}_{256}$, so that $\mathbb{F}_{256}$ can be regarded as an 8-dimensional vector space over $\mathbb{F}_2$. **Appendix 8.** If $b_{\text{input}} \neq 0$, then the output of this step is $b_{\text{inverse}} = b_{\text{input}}^{-1}$ otherwise $b_{\text{inverse}} = 0$.

2. **$\mathbb{F}_2$-linear mapping::** Consider $b_{\text{inverse}}$ again as a vector from $\mathbb{F}_2^8$. The output of this step is given by $b_{\text{linear}} = L(b_{\text{inverse}})$, where $L$ is an invertible $\mathbb{F}_2$-linear mapping given by a prescribed circulant matrix.

3. **S-box constant:** The output of the entire S-box is obtained as $b_{\text{output}} = b_{\text{linear}} + c$, where $c$ is an S-box constant.

Thus, in essence, each S-box applies inversion and then the affine transformation to an 8-bit input block yielding 8-bit output block. It is easy to see that S-box so defined is invertible.

The substitution layer acts locally on each individual byte, whereas diffusion layer acts on the entire square array. The diffusion layer consists of two consecutive linear transformations. The first one, called ShiftRows, shifts the $i$-th row of the array by $i - 1$ positions to the left. The second one, called MixColumns, is given by a $4 \times 4$ matrix $M$ over $\mathbb{F}_{256}$ and transforms every column $C$ of the array to a column $MC$. The matrix $M$ is the parity check matrix of an MDS code.
code, cf. Definition 3.2.2 and was introduced to follow the so-called wide trail strategy and precludes the use of linear and differential cryptanalysis. Let us now describe the encryption process of AES.

**Algorithm 10.1.13 (AES encryption)**

**Input:** The 128-bit plaintext $p$ and the 128-bit key $k$

**Output:** The 128-bit ciphertext $c$ corresponding to $p$.

1. Perform initial key addition: $w := p \oplus k \equiv \text{AddRoundKey}(p, k)$.
2. Expand the initial key $k$ to subkeys $k_1, \ldots, k_{10}$ using key schedule.
3. For $i = 1, \ldots, 9$ do
   - Perform S-box substitution: $w := \text{SubBytes}(w)$.
   - Shift the rows: $w := \text{ShiftRows}(w)$.
   - Transform the columns with the MDS matrix $M$: $w := \text{MixColumns}(w)$.
   - Add the round key: $w := \text{AddRoundKey}(w, k_i) = w \oplus k_i$.

   # The last round does not have MixColumns.
4. Perform S-box substitution: $w := \text{SubBytes}(w)$.
5. Shift the rows: $w := \text{ShiftRows}(w)$.
6. Add the round key: $w := \text{AddRoundKey}(w, k_{10}) = w \oplus k_{10}$.
7. The ciphertext is $c := w$.

The key schedule is designed similarly to the encryption and is omitted here. All the details on the components, the key schedule, and the reverse cipher for decryption can be found in the literature, see Notes. The reverse cipher is quite straightforward as it has to undo invertible affine transformations and the inversion in $F_{256}$.

Let us discuss some properties of AES. First of all, we note that AES possesses confusion and diffusion properties. The use of S-boxes provides sufficient resistance to linear and differential cryptanalysis that was one of the major concerns when replacing DES. The use of the affine mapping in the S-box among other things removes fixed points. In the diffusion layer the diffusion is done separately for rows and columns. It is remarkable that in contrast to the DES, where the encryption is mainly described via table look-ups, AES description is very algebraic. All transformations described as either field inversion or a matrix multiplication. Of course, in real-world applications some operations like S-box are nevertheless realized as table look-ups. Still the simplicity of the AES description has been in discussion since the selection process, where the future AES Rijndael took part. Highly algebraic nature of the AES description boosted a new branch in cryptanalysis called algebraic cryptanalysis. We address this issue in the next chapter, see Section 11.3.
10.1.5 Exercises

10.1.1 It is known that the following ciphertext is obtained with a permutation cipher of period 6 and contains an anagram of a famous person’s name (spaces are ignored by encryption): "AAASSNIFNOSECRSAAKIWNOSN". Find the original plaintext.

10.1.2 Sequential composition of several permutation ciphers with periods \( t_1, \ldots, t_s \) is called compound permutation (compound transposition). Show that the compound permutation cipher is equivalent to a simple permutation cipher with the period \( t = \text{lcm}(t_1, \ldots, t_s) \).

10.1.3 [CAS] The Hill cipher is defined as follows. One encodes a length-\( n \) block \( p = (p_1, \ldots, p_n) \), which is assumed to consist of elements from \( \mathbb{Z}_n \), with an invertible \( n \times n \) matrix \( H = (h_{ij}) \) as \( c_i = \sum_{j=1}^{n} h_{ij} p_j \). Therewith one obtains the cryptogram \( c = (c_1, \ldots, c_n) \). The decryption is done analogously using \( H^{-1} \). Write a procedure that implements the Hill cipher. Compare your implementation with the \texttt{HillCryptosystem} class from Sage.

10.1.4 The following text is encrypted with a monoalphabetic substitution cipher. Decrypt it. The following ciphertext using frequency analysis and Table 10.1:

AI QYWX YRHIVXEOI MQQIHMEXI EGXMSR. XLI IRIQC MW ZIVC GOSWI!

Hint: Decrypting small words and using first may be very useful. Also use Table 10.1.

10.1.5 Show that in the definition of a Feistel cipher the transformation \( f \) need not be invertible to ensure encryption, in a sense that the round function is invertible even if \( f \) is not. Also show that performing encryption starting at \((r_{N_r}, l_{N_r})\) with the reverse order of subkeys, yields \((l_0, r_0)\) at the end, thus providing a decryption.

10.1.6 It is known that the expansion transformation \( E \) of DES has the complementary property, i.e. for every input \( x \) it holds that \( E(\overline{x}) = \overline{E(x)} \). It is also known that \( k \) expands to \( k_1, \ldots, k_{16} \). Knowing this show that

a. The entire DES transformation also possesses the complementary property: \( \forall p \in \{0, 1\}^{64}, k \in \{0, 1\}^{56} : DES_k(p) = \overline{DES_k(\overline{p})} \).

Using (a.) show

b. It is possible to reduce exhaustive search complexity from \( 2^{55} \) (half the key-space size) to \( 2^{54} \).

10.2 Asymmetric cryptosystems

In Section 10.1 we considered symmetric cryptosystems. As we may see, for a successful communication Alice and Bob are required to keep their encryption/decryption keys secret. Only the channel itself is assumed to be eavesdropped. For Alice and Bob to set a secret communication it is necessary to
convey encryption/decryption keys. This can be done, e.g. by means of a trusted courier or some very secure channel (like specially secured telephone line) that is considered to be strongly secure. This paradigm suited well for diplomatic and military communication: the amount of communicating parties in these scenarios was quite limited; in addition, usually communicating parties could afford sending a trusted courier in order to keep keys secret or provide some highly protected channel for exchanging keys. In 1970s with the beginning of electronic communication it became apparent that such an exchanging mechanism is absolutely inefficient. This is mainly due to a drastic increase in the number of communicating parties. It is not only diplomats or high order military officials that wish to set secret communication, but usual users (e.g. companies, banks, social networks users) who would like to be able to do business over some large distributed network. Suppose that there is \( n \) users which potentially are willing to communicate with each other secretly. Then it is possible to share secret keys between every pair of users. There are \( n(n - 1)/2 \) pairs of users, so one would need this number of exchanges in a network to set the communication. Note that already for \( n = 1,000 \), we have \( n(n - 1)/2 = 499,500 \), which is of course not something we would like to do. Another option would be to set some trusted authority in the middle who would store secret keys for every user and then if Alice would like to send a plaintext \( p \) to Bob, she would send \( c_{\text{Alice}} = E_{K_{\text{Alice}}}(p) \) to the trusted authority Tim. Tim would decrypt \( p = D_{K_{\text{Alice}}}(c_{\text{Alice}}) \) and send \( c_{\text{Bob}} = E_{K_{\text{Bob}}}(p) \) to Bob who is able then to decrypt \( c_{\text{Bob}} \) with his secret key \( K_{\text{Bob}} \). An obvious drawback of this approach is that Tim knows all the secret keys, and thus is able to read (and alter!) all the plaintexts, which is of course not desirable. Another disadvantage is that for a large network it could be hard to implement the trusted authority of this kind as it has to take part in every communication between users and thus can get overwhelmed.

A solution to the problem above was proposed by Diffie and Hellman in 1976. This was the starting point for asymmetric cryptography. The idea is that if Alice wants to communicate with some other parties, she generates an encryption/decryption pair \((e, d)\) in such a way that knowing \( e \) it is computationally infeasible to obtain \( d \). This is quite different from symmetric cryptography, where \( e \) and \( d \) are (computationally) the same. Motivation for the name "asymmetric cryptosystem" as oppose to "symmetric cryptosystem" should be clear now. So what Alice does, she publishes her encryption key \( e \) in some public repository and keeps \( d \) secret. If Bob wants to send a plaintext \( p \) to Alice, he simply finds her public key \( e = c_{\text{Alice}} \) in the repository and uses it for encryption: \( c = E_{e}(p) \). Now Alice is able to decrypt with her private key \( d = d_{\text{Alice}} \).

Note that due to assumptions we have on the pair \((e, d)\), Alice is the only person who is able to decrypt \( c \). Indeed, Eve is able to know \( c \) and an encryption key \( e \) used, but she is not able to get \( d \) for decryption. Remarkably, even Bob himself is not able to restore his plaintext \( p \) from \( c \) if he loses or deletes it beforehand! The formal definition follows.

**Definition 10.2.1** The asymmetric cryptosystem is defined by the following data:

- The plaintext space \( \mathcal{P} \) and the ciphertext space \( \mathcal{C} \).
- \( \{ E_{e} : \mathcal{P} \rightarrow \mathcal{C} | e \in \mathcal{K} \} \) and \( \{ D_{d} : \mathcal{C} \rightarrow \mathcal{P} | d \in \mathcal{K} \} \) are the sets of encryption and decryption transformations resp., which are bijections from \( \mathcal{P} \) to \( \mathcal{C} \).
The above transformations are parameterized by the key space $K$.

- Given an associated pair $(e, d)$, so that a property $\forall p \in \mathcal{P} : D_d(E_e(p)) = p$ holds, knowing $e$ it is "computationally hard" to find out $d$.

Here, the encryption key $e$ is called public and the decryption key $d$ is called private.

The core issue in the above definition is having a property that knowledge of $e$ practically does not shed any light on $d$. The study of this issue led to the notion of a one-way function. We say that a function $f : X \rightarrow Y$ is one-way if it is "computationally easy" to compute $f(x)$ for any $x \in X$, but for $y \in \text{Im}(f)$, it is "computationally hard" to find $x \in X$ such that $f(x) = y$. Note that one may compute $Y' = \{ f(x) | x \in \mathbb{Z} \subset X \}$, where $\mathbb{Z}$ is some small subset of $X$ and then invert elements from $Y'$. Still $Y'$ is essentially small compared to $\text{Im}(f)$, so for randomly chosen $y \in \text{Im}(f)$ the above assumption should hold. Theoretically it is not known if one-way functions exist, but in practice there are several candidates that are believed to be one-way. We discuss this a bit later.

The above notion of one-way function solves half of the problem. Namely, if Bob sends Alice an encrypted plaintext $c = E(p)$, where $E$ is one-way, Eve is not able to find $p$ as she is not able to invert $E$. But Alice faces then the same problem! Of course we would like to provide Alice with means to invert $E$ and find $p$. Here the notion of a trapdoor one-way function comes in hand. A one-way function $f : X \rightarrow Y$ is said to be one-way trapdoor, if there is some additional information, called trapdoor, having which it is "computationally easy" for $y \in \text{Im}(f)$ to find $x \in X$ such that $f(x) = y$. Now if Alice possesses such a trapdoor for $E$ she is able to obtain $p$ from $c$.

**Example 10.2.2** We now give examples of functions that are believed to be one-way.

1. The first is $f : \mathbb{Z}_n \rightarrow \mathbb{Z}_n$ defined by $f(x) = x^a \mod n$. If we take $a = 3$ it is easy to compute $x^3 \mod n$, but having $y \in \mathbb{Z}_n$ it is believed to be hard to compute $x$ such that $y = x^3 \mod n$. For suitably chosen $a$ and $n$ this function is used in RSA cryptosystem, Section 10.2.1. For $a = 2$ one obtains the so-called Rabin scheme. It can be shown that in fact factoring $n$ is equivalent to inverting $f$ in this case. Since factoring of integers is considered to be a hard computational problem, it is believed that $f$ is one-way. For RSA it is believed that inverting $f$ is as hard as factoring, although no rigorous proof is known. In both schemes above it is assumed that $n = pq$, where $p$ and $q$ are (suitably chosen) primes, and this fact is a public knowledge, but $p$ and $q$ are kept secret. One-way property relies on hardness of factoring $n$, i.e. finding $p$ and $q$. For Alice the knowledge of $p$ and $q$ is a trapdoor using which he is able to invert $f$. Thus $f$ is believed to be trapdoor one-way function.

2. The second example is $g : \mathbb{F}_q^* \rightarrow \mathbb{F}_q^*$ defined by $g(x) = a^x$, where $a$ generates the multiplicative group $\mathbb{F}_q^*$. The problem of inverting $g$ is called discrete logarithm problem (DLP) in $\mathbb{F}_q$. It is a basis for El Gamal scheme, Section
10.2. ASYMMETRIC CRYPTOSYSTEMS

10.2.2. The DLP problem is believed to be hard in general, thus $g$ is believed to be one way, since for given $x$ computing $a^x$ in $\mathbb{F}_q^*$ is easy. One may also use domains different from $\mathbb{F}_q$ and try to solve DLP there, on some discussion on that cf. Section 10.2.2.

3. Consider a function $h : \mathbb{F}_q^k \rightarrow \mathbb{F}_q^n$, $k < n$ defined as $\mathbb{F}_q^k \ni m \mapsto mG + e \in \mathbb{F}_q^n$, where $G$ is a generator matrix of an $[n,k,d]$ linear code and $wt(e) \leq t \leq (d-1)/2$. So $h$ defines an encoding function for the code defined by $G$. When inverting $h$ one faces the problem of bounded distance decoding, which is believed to be hard. The function $h$ is a basis for the McEliece and Niederreiter cryptosystems, see Sections 10.6 and ??.

4. In the last example we consider a function $z : \mathbb{F}_q^n \rightarrow \mathbb{F}_q^m$, $n \geq m$ defined as $\mathbb{F}_q^n \ni x \mapsto F(x) = (f_1(x), \ldots, f_m(x)) \in \mathbb{F}_q^m$, where $f_i$'s are non-linear polynomials over $\mathbb{F}_q$. Inverting $z$ means finding a solution of $F(X) = 0$ of a system of non-linear equations. This problem is known to be NP-hard even if $f_i$'s are quadratic and $q = 2$. The function $z$ is a basis of multivariate cryptosystems, see Section 10.2.3.

Before going to consider concrete examples of asymmetric cryptosystems, we would like to note that there is a vital necessity of authentication in asymmetric cryptosystems. Indeed, imagine that Eve can not only intercept and read messages, but also alter the repository, where public keys are stored. Suppose Alice is willing to communicate a plaintext $p$ to Bob. Assume that Eve is aware of this intention and is able to substitute Bob’s public key $e_{Bob}$ with her key $e_{Eve}$ for which she has the corresponding decryption key $d_{Eve}$. Alice, not knowing that the key was replaced, takes $e_{Eve}$ and encrypts $c = E_{e_{Eve}}(p)$. Eve intercepts $c$ and decrypts $p = D_{d_{Eve}}(c)$. So now Eve knows $p$. After that she may either encrypt $p$ with Bob’s $e_{Bob}$ and send the ciphertext to him, or even replace $p$ with some other $p'$. As a result, not only Eve gets the secret message $p$, but Bob can be misinformed by the message $p'$, which, as he thinks, comes from Alice. Fortunately there are ways of providing means to tackle this problem. They include use of a third trusted party (TTP) and digital signatures. Digital signatures are asymmetric analogous of (message) authentication codes, Section 10.3. These are out of scope of this introductory chapter.

The last remark concerns the type of security that asymmetric cryptosystems provide. Note that opposed to symmetric cryptosystems, some of which can be shown to be unconditionally secure, asymmetric cryptosystems can only be computationally secure. Indeed, having Bob’s public key $e_{Bob}$ Eve can simply encrypt all possible plaintexts until she finds $p$ such that $E_{e_{Bob}}(p)$ coincides with the ciphertext $c$ that she observed.

10.2.1 RSA

Now we consider an example of one of the most used asymmetric cryptosystems - RSA, named after its creators R. Rivest, A. Shamir, and L. Adleman. This cryptosystem was proposed in 1977 shortly after Diffie and Hellman invented asymmetric cryptography. It is based on hardness of factorizing integers and up to now withstood cryptanalysis, although some of the attacks suggest careful choosing of public/private key and its size. First we present the RSA itself:
how one chooses a public/private key pair, how encryption/decryption is done
and why it works. Then we consider a concrete example with small numbers.
Finally we discuss some security issues. In this and the following subsection we
denote plaintext with $m$, because historically $p$ and $q$ are reserved in context of
RSA.

**Algorithm 10.2.3 (RSA key generation)**

**Output:** RSA public/private key pair $((e, n), d)$.

1. Choose two distinct primes $p$ and $q$.
2. Compute $n = pq$ and $\phi(n) = (p - 1)(q - 1)$.
3. Select a number $e$, $1 < e < \phi$, such that $\gcd(e, \phi) = 1$.
4. Using extended Euclidean algorithm, compute $d$ such that $ed \equiv 1 \pmod{\phi}$.
5. The key pair is $((e, n), d)$.

The integers $e$ and $d$ above are called *encryption* and *decryption* exponent resp.;
the integer $n$ is called *modulus*. For encryption Alice uses the following algo-
rithm.

**Algorithm 10.2.4 (RSA encryption)**

**Input:** Plaintext $m$ and Bob’s encryption exponent $e$ together with the modulus $n$.

**Output:** Ciphertext $c$.

1. Represent $m$ as an integer $0 \leq m < n$.
2. Compute $c = m^e \pmod{n}$.
3. The ciphertext for sending to Bob is $c$.

For decryption Bob uses the following

**Algorithm 10.2.5 (RSA decryption)**

**Input:** Ciphertext $c$, the decryption exponent $d$, and the modulus $n$.

**Output:** Plaintext $m$.

1. Compute $m = c^d \pmod{n}$.
2. The plaintext is $m$.

Let us see why Bob gets initial $m$ as a result of decryption. Since $ed \equiv 1 \pmod{\phi}$, there exists an integer $s$ such that $ed = 1 + s\phi$. For $\gcd(m, p)$ there are two possibilities: either $1$ or $p$. If $\gcd(m, p) = 1$, then due to Fermat’s little theorem we have $m^{p-1} \equiv 1 \pmod{p}$. Raising both sides to $(q-1)$-th power and multiplying by $m$ we have $m^{1+s(p-1)(q-1)} \equiv m \pmod{p}$. Now using $ed = 1 + s\phi = 1 + s(p-1)(q-1)$ we have $m^{ed} \equiv m \pmod{p}$. For the case $\gcd(m, p) = p$ we get the last congruence right away. The same argument can be applied to $q$, so we obtain analogously $m^{ed} \equiv m \pmod{q}$. Using the Chinese remainder theorem we then get $m^{ed} = m \pmod{n}$. So indeed $c^d = (m^e)^d = m \pmod{n}$.
10.2. ASYMMETRIC CRYPTOSYSTEMS

Example 10.2.6 Consider an example of RSA as is described in the algorithms above with some small values. First let us choose the primes $p = 5519$ and $q = 4651$. So our modulus is $n = pq = 25668869$, thus $\phi = (p-1)(q-1) = 25658700$. Take $e = 29$ as an encryption exponent, $\gcd(29, \phi) = 1$. Using Euclidian algorithm obtain $e \cdot (-3539131) + 4 \cdot \phi = 1$, so take $d = -3539131 \mod \phi = 22119569$. The key pair now is $((e, n), d) = ((29, 25668869), 22119569)$.

Suppose Alice wants to transmit a plaintext message $m = 7847098$ to Bob. She takes his public key $e = 29$ and computes $c = m^e \pmod{n} = 22152327$. She sends $c$ to Bob. After obtaining $c$ Bob computes $c^d \pmod{n} = m$.

Example 10.2.7 Magma computer algebra system (cf. Appendix ??) gives an opportunity to compute an RSA modulus of a given bit-length. For example, if we want to construct a “random” RSA modulus of bit-length 25, we should write:

```plaintext
> RSAModulus(25);
26827289 1658111
```

Here the first number is the random RSA modulus $n$ and the second one is a number $e$, such that $\gcd(e, \phi(n)) = 1$. We can also specify the number $e$ explicitly (below $e = 29$):

```plaintext
> n:=RSAModulus(25,29);
> n;
19579939
```

One can further factorize $n$ as follows:

```plaintext
> Factorization(n);
[ <3203, 1>, <6113, 1> ]
```

This means that $p = 3203$ and $q = 6113$ are prime factors of $n$ and $n = pq$. We can also use extended Euclidian algorithm to recover $d$ as follows:

```plaintext
> e:=29; phi:=25658700;
> ExtendedGreatestCommonDivisor(e, phi);
1 -3050975 4
```

So here 1 is the $\gcd$ and $d = -3539131$, as was computed in the example above.

As has already been mentioned, the RSA relies on hardness of factoring integers. Of course, if Eve is able to factor $n$, then she is able to produce $d$ and thus decrypt all ciphertexts. The open question is whether breaking RSA leads to a factoring algorithm for $n$. The problem of breaking RSA is called the RSA problem. There is no rigorous proof, though, that breaking RSA is equivalent to factoring. Still it can be shown that computing decryption exponent $d$ and factoring are equivalent. Note that in principle for an attacker it might be unnecessary to compute $d$ in order to figure out $m$ from $c$ given $(e, n)$. Nevertheless, even though there is no rigorous proof of equivalence, RSA is believed to be as hard as factoring. Now we briefly discuss some other things that need to be taken into the consideration, when choosing parameters for the RSA.

1. For fast encryption using small encryption exponent is desirable, e.g. $e = 3$. The possibility for an attack exists then, if this exponent is used for sending the same message even to different recipients with different moduli. There is also concern about small decryption exponent. For example, if bitlength of $d$ is approximately $1/4$ of bitlength of $n$, then there is an efficient way to get $d$ from $(e, n)$. 
2. As to the primes \( p \) and \( q \) one should take the following into account. First of all \( p - 1 \) and \( q - 1 \) should not have small factors, as then factoring \( n \) with Pollard’s \( p - 1 \) algorithm is possible. Then, in order to avoid elliptic curve factoring, \( p \) and \( q \) should be roughly of the same bitlength. On the other side if the difference \( p - q \) is too small then techniques like Fermat factorization become feasible.

3. In order to avoid problems as in (1.), different padding schemes are proposed that add certain amount of randomness to ciphertexts. Thus the same message will be encrypted to one of ciphertexts from some range.

An important remark to make is that using so-called quantum computers, which are large enough, it is possible to solve the factorization problem in polynomial time. See Notes for references. The same problem exists for the cryptosystems based on the DLP, which are described in the next subsection. Problems (3.) and (4.) from Example 10.2.2 are not known to be susceptible to quantum computer attacks. Together with some other hard problems, they make a foundation for the post-quantum cryptography that deals with cryptosystems resistant to quantum computer attacks. See Notes for references.

10.2.2 Discrete logarithm problem and public-key cryptography

In the previous subsection we considered the asymmetric cryptosystem RSA based on hardness of factorizing integers. As has already been noted in Example 10.2.2, there is also a possibility to use hardness of finding discrete logarithms as a basis for an asymmetric cryptosystem. General DLP is defined below.

Definition 10.2.8 Let \( G \) be a finite cyclic group of order \( g \). Let \( \alpha \) be a generator of this group so that \( G = \{ \alpha^i \mid 1 \leq i \leq g \} \). The discrete logarithm problem (DLP) in \( G \) is the problem of finding \( 1 \leq x \leq g \) from \( a^x = \alpha^x \), where \( a \in G \) is given.

For cryptographic purposes a group \( G \) should possess two main properties: 1.) the operation in \( G \) should be efficiently performed and 2.) the DLP in \( G \) should be difficult to solve (see Exercise 10.2.4). Cyclic groups that are widely used in cryptography include the multiplicative group \( \mathbb{F}_q^* \) of the finite field \( \mathbb{F}_q \) (in particular the multiplicative group \( \mathbb{Z}_p^* \) for \( p \) prime), a group of points on an elliptic curve over a finite field. Other possibilities that exist out there are the group of units \( \mathbb{Z}_n^* \) for a composite \( n \), the Jacobian of a hyperelliptic curve defined over a finite field, and the class group of an imaginary quadratic number field, see Notes.

Here we consider classical El Gamal scheme based on the DLP. As we will see the following description will do for any cyclic group with "efficient description". Initially the multiplicative group of a finite field was used.

Algorithm 10.2.9 (El Gamal key generation)
Output: El Gamal public/private key pair \(((G, \alpha, h), a)\).

1. Choose some cyclic group \( G \) of order \( g = \text{ord}(G) \), where the group operation is done efficiently, and then choose its generator \( \alpha \).

2. Select a random integer \( a \) such that \( 1 \leq a \leq g - 2 \) and compute \( h = \alpha^a \).
3. The key pair is \((G, \alpha, h), a)\).

Note that \(G\) and \(\alpha\) can be fixed in advance for all users, so only \(h\) becomes a public key. For encryption Alice uses the following algorithm.

**Algorithm 10.2.10** (*El Gamal encryption*)

**Input:** Plaintext \(m\) and Bob’s encryption public key \(h\) together with \(\alpha\) and the group description of \(G\).

**Output:** Ciphertext \(c\).

1. Represent \(m\) as an element of \(G\).
2. Select random \(b\) such that \(1 \leq b \leq g - 2\), where \(g = \text{ord}(G)\) and compute \(c_1 = \alpha^b\) and \(c_2 = m \cdot h^b\).
3. The ciphertext for sending to Bob is \(c = (c_1, c_2)\).

For decryption Bob uses the following

**Algorithm 10.2.11** (*El Gamal decryption*)

**Input:** Ciphertext \(c\), the private key \(a\) together with \(\alpha\) and the group description of \(G\).

**Output:** Plaintext \(m\).

1. In \(G\) compute \(m = c_2 \cdot c_1^{-a} = c_2 \cdot c_1^{g-1-a}\), where \(g = \text{ord}(G)\).
2. The plaintext is \(m\).

Let us see why we get initial \(m\) as a result of decryption. Using \(h = \alpha^a\) we have

\[c_2 \cdot c_1^{-a} = m \cdot h^b \cdot \alpha^{-ab} = m \cdot \alpha^{ab} \cdot \alpha^{-ab} = m.\]

**Example 10.2.12** For this example let us take the group \(\mathbb{Z}_p^*\) where \(p = 8053\) with a generator \(\alpha = 2\). Let us choose a private key to be \(a = 3117\). Compute \(h = \alpha^a \mod p = 3030\). So the public key is \(h = 3030\) and the private key is \(a = 3117\).

Suppose Alice wants to encrypt a message \(m = 1734\) for Bob. For this she chooses a random \(b = 6809\) computes \(c_1 = \alpha^b \mod p = 3540\) and \(c_2 = m \cdot h^b \mod p = 7336\). So her ciphertext is \(c = (3540, 7336)\). Upon receiving \(c\) Bob computes \(c_2 \cdot c_1^{p-1-a} \mod p = 7336 \cdot 3540^{-3935} \mod 8053 = 1734\).

Now we briefly discuss some issues connected with the El Gamal scheme.

- **Message expansion:** It should be noted that oppose to the RSA scheme, ciphertexts in El Gamal are twice as large as plaintexts. So we have that El Gamal scheme actually has a drawback of providing message expansion by factor of 2.

- **Randomization:** Note that in Algorithm 10.2.10 we used randomization to compute a ciphertext. Randomization in encryption gives an advantage that the same message is mapped to different ciphertexts with different encryption runs. This in turn makes chosen-plaintext attack more difficult. We will see another example of an asymmetric scheme with randomized encryption in Section 10.6, where we discuss McEliece scheme based on error-correcting codes.
• Security reliance: The problem of breaking El Gamal scheme is equivalent to the so-called (generalized) Diffie-Hellman problem, which is the problem of finding \( \alpha^{ab} \in G \) given \( \alpha^a \in G \) and \( \alpha^b \in G \). Obviously enough, if one is able to solve the DLP, then one is able to solve the Diffie-Hellmann problem, i.e. DLP is polytime reducible to the Diffie-Hellmann problem (cf. Definition 6.1.22). It is not known whether these two problems are computationally equivalent. Nevertheless, it is believed that breaking El Gamal is as hard as solving DLP.

• As we have mentioned before, El Gamal scheme is vulnerable to quantum computer attacks. See Notes.

10.2.3 Some other asymmetric cryptosystems

So far we have seen examples of asymmetric cryptosystems based on hardness of factoring integers (Section 10.2.1) and solving DLP in the multiplicative group of a finite field (Section 10.2.2). Other examples that will be covered are McEliece scheme that is based on hardness of decoding random linear codes (Section 10.6) and solving DLP in a group of points of an elliptic curve over a finite field (Section ??). In this subsection we would like to briefly mention what are other alternatives that exist out there.

The first direction we consider here is the so-called multivariate cryptography. Here cryptosystems based on hardness of solving the multivariate quadratic (MQ) problem. This problem is the problem of finding a solution \( x = (x_1, \ldots, x_n) \in \mathbb{F}_q^n \) to the system

\[
y_1 = f_1(X_1, \ldots, X_n), \\
\vdots \\
y_m = f_m(X_1, \ldots, X_n),
\]

where \( f_i \in \mathbb{F}_q[X_1, \ldots, X_n], \deg f_i = 2, i = 1, \ldots, m \) and the vector \( y = (y_1, \ldots, y_m) \in \mathbb{F}_q^m \) is given. This problem is known to be NP-hard, so is thought to be a good source of a one-way function. The trapdoor is added by choosing \( f_i \)'s having some structure that is kept secret and allows decryption that e.g. boils down to univariate factorization over a larger field. To an eavesdropper, though, the system above with such a trapdoor should appear random. So the idea is that the eavesdropper can do no better than solve a random quadratic system over a finite field which is believed to be a hard problem. The cryptosystems and digital schemes in this category include e.g. Hidden Field Equations (HFE), SFLASH, Unbalanced Oil and Vinegar (UOV), Step-wise Triangular Schemes (STS), and some others. Some of those were broken and several modification were proposed to overcome the attacks (e.g. PFLASH, enSTS). At present it is not quite clear whether it is possible to design a secure multivariate cryptosystem. A lot of research in this area, though, gives a basis for optimism.

Another well-known example of a cryptosystem based on an NP-hard problem is the knapsack cryptosystem. This cryptosystem was the first concrete realization of an asymmetric scheme and was proposed in 1978 by Merkle and Hellman. The knapsack cryptosystem is based on a well-known NP-hard subset sum problem. Namely the problem by given the set of positive integers \( A = \{a_1, \ldots, a_n\} \) and the positive integer \( s \) to find a subset of \( A \), such that the sum of elements from \( A \) yields \( s \). The idea of Merkle and Hellman was to make so-called super increasing sequences, for which the above problem is easily solved, appear as
10.3. AUTHENTICATION, ORTHOGONAL ARRAYS, AND CODES

A random set $A$, thus providing a trapdoor. So an eavesdropper supposedly has nothing better to do as to deal with well-known hard problem. This initial proposal was broken by Shamir and later an improved version was broken by Brickell. These attacks are based on integer lattices and made quite a shake in cryptographic community at that time. There are some other types of cryptosystems out there: polynomial based “Poly-Cracker”-type, lattice based, hash based, and group based. Therefore, we may summarize that active research is being conducted in order to provide alternatives to widely used cryptosystems.

10.2.4 Exercises

10.2.1 a. Given primes $p = 5081$ and $q = 6829$ and an encryption exponent $e = 37$ find the corresponding decryption exponent and encrypt the message $m = 29800110$.

b. Let $e$ and $m$ be as above. Generate (e.g. with Magma) a random RSA modulus $n$ of bit-length 25. For these $n, e, m$ find the corresponding decryption exponent via factorizing $n$; encrypt $m$.

10.2.2 Show that a number $\lambda = \text{lcm}(p - 1, q - 1)$ that is called universal exponent of $n$, can be used instead of $\phi$ in Algorithms 10.2.3 and 10.2.5.

10.2.3 Generate a public/private key pair for El Gamal scheme with $G = \mathbb{Z}_{7121}^\ast$ and encrypt a message $m = 5198$ using this scheme.

10.2.4 Give an example of a finite cyclic group where the DLP problem is easy to solve.

10.2.5 Show that using the same $b$ in Algorithm 10.2.10 at least for two different encryptions is insecure, namely if $c'$ and $c''$ are two ciphertexts that correspond to $m'$ and $m''$, which were encrypted with the same $b$, then knowing one of the plaintexts yields the other.

10.3 Authentication, orthogonal arrays, and codes

10.3.1 Authentication codes

In Section 10.1 we dealt with the problem of secure communication between two parties by means of symmetric cryptosystems. In this section we address another important problem, the problem of data source authentication. So we are now interested in providing means for Bob to make sure that a (encrypted) message he received from Alice indeed was sent by her and was not altered during the transmission. In this section we consider so-called authentication codes that provide tools necessary to ensure authentication. These codes are analyzed in terms of unconditional security (see Definition 10.1.8). For practical purposes one is more interested in computational security. Analogous to authentication codes for this purpose are message authentication codes (MACs). It is also to be noted that authentication codes are, in a sense, symmetric based, i.e. a secretly shared key is needed to provide such an authentication. There is also asymmetric analogue (Section 10.2) called a digital signature. In this model everybody can
verify Alice’s signature by publicly available verification algorithm. Let us go on now to the formal definition of an authentication code.

**Definition 10.3.1** An authentication code is defined by the following data:

- A set of source states \( S \).
- A set of authentication tags \( T \).
- A set of keys, the keyspace \( K \).
- A set of authentication maps \( A \) parameterized by \( K \): for each \( k \in K \) there is an authentication map \( a_k : S \to T \).

We also define a message space \( M = S \times T \).

The idea of authentication is as follows. Alice and Bob secretly agree on some secret key \( k \in K \) for their communication. Suppose that Alice wants to transmit a message \( s \) which per definition above is called a source state. Note that now we are not interested in providing secrecy for \( s \) itself, but rather in providing means of authentication for \( s \). For the transmission Alice adds an authentication tag to \( s \) by \( t = a_k(s) \). She then sends concatenated message \((s, t)\). Usually \((s, t)\) is an encrypted message, maybe also encoded for error-correction, but for us it does not play a role here. Suppose Bob receives \((s', t')\). He separates \( s' \) and \( t' \) and checks whether \( s' = a_k(t') \). If the check succeeds, he accepts \( s' \) as a valid message that came from Alice otherwise he rejects it. If no intrusion occurred we have \( s = s' \) and \( t = t' \) and the check trivially succeeds. But what if Eve wants to alter the message and make Bob believe that the altered by her choice message still originates from Alice? There are two types of Eve’s malicious actions one usually considers.

- **Impersonation**: Eve sends some message \((s, t)\) with an intention that Bob accepts it as Alice’s message, i.e. she aims at passing the check \( s = a_k(t) \) with high probability, where the key \( k \) is unknown to Eve.
- **Substitution**: Eve intercepts Alice’s message \((s, t)\). Now she wants to substitute instead another message \((s', t')\), where \( s' \neq s \) such that \( a_k(s') = t' \) for the key \( k \) unknown to Eve.

As has already been said authentication codes are studied from the point of view of unconditional security, i.e. we assume that Eve has unbounded computational power. In this case we need to show that no matter how much computational power Eve has, she cannot succeed in the above attack scenarios with a large probability. Therefore, we need to estimate probabilities of success of impersonation \( P_I \) and substitution \( P_S \), given probability distributions \( p_S \) and \( p_K \) of the source states set and key space resp. The probabilities \( P_I \) and \( P_S \) are also called deception probabilities. Note that \( P_I \) as well as \( P_S \) are computed in assumption that Eve tries to maximize her chances of deception. In reality Eve might want not only to maximize her probability to pass the check, but also she might have some preference as to which message she wants to substitute for Alice’s one. For example, intercepting Alice’s message \((s, t)\), where \( s = "Meeting is at seven" \), she would like to send something like \((s', t')\), where \( s' = "Meeting is at six" \). Thus \( P_I \) and \( P_S \) actually provide an upper bound on Eve’s chances of success.
Let us first compute $P_I$. Let us compute what is the probability of some message $(s, t)$ to be validated by Bob, when some private key $k_0 \in K$ is used. In fact for Eve every key $k$ that maps $s$ to $t$ will do. So, $\Pr(a_{k_0}(s) = t) = \sum_{k \in K} a_{k}(s) = \Pr(k)$. Now in order to maximize her chances, Eve should choose $(s, t)$ with $\Pr(a_{k_0}(s) = t)$ largest possible, i.e.

$$P_I = \max \{ \Pr(a_{k_0}(s) = t) | s \in S, t \in T \}.$$ 

Note that $P_I$ depends only on the distribution $p_K$ and not on $p_S$.

Computing $P_S$ is a bit trickier. We obtain that conditional probability $\Pr(a_{k_0}(s') = t'|a_{k_0}(s) = t)$ of the fact that Eve’s message $(s', t'), s' \neq s$ passes the check once valid message $(s, t)$ is known is

$$\Pr(a_{k_0}(s') = t'|a_{k_0}(s) = t) = \frac{\Pr(a_{k_0}(s') = t', a_{k_0}(s) = t)}{\Pr(a_{k_0}(s) = t)} = \frac{\sum_{k \in K, a_k(s) = t'} p_K(k)}{\sum_{k \in K, a_k(s) = t} p_K(k)}.$$

Having $(s, t)$, Eve maximizes her chances by choosing $(s', t'), s' \neq s$, such that the corresponding conditional probability is maximal. To reflect this, introduce $p_{s,t} := \max \{ \Pr(a_{k_0}(s') = t'|a_{k_0}(s) = t) | s' \in S \setminus \{s\}, t' \in T \}$. Now in order to get $P_S$ we need to take weighted average of $p_{s,t}$ according to the distribution $p_S$:

$$P_S = \sum_{(s, t) \in M} p_M(s, t)p_{s,t},$$

where the distribution $p_M$ is obtained as $p_M(s, t) = p_S(s)p(t|s) = p_S(s) \times \Pr(a_{k_0}(s) = t)$. The value $\Pr(a_{k_0}(s) = t)$ is called pay-off of a message $(s, t)$, we denote it as $\pi(s, t)$. Also $Pr(a_{k_0}(s') = t'|a_{k_0}(s) = t)$ is a pay-off of a message $(s, t)$ given a valid message $(s', t')$, we denote it as $\pi_{s,t}(s', t')$.

For convenience one may think of an authentication code as of array, which rows are indexed by $K$, columns by $S$ and an entry $(k, s)$ for $k \in K, s \in S$ has a value $a_k(s)$, see Exercise 10.3.1.

We have discussed some basic things about authentication codes. So the question now is what are important criteria for a good authentication code. These are summarized below:

1. The deception probabilities must be small, so that eavesdropper’s chances are low.
2. $|S|$ should be large to facilitate authentication of potentially large number of source states.
3. Note that since we are studying authentication codes from the point of view of unconditional security, the secret key should be used only one, and then changed for the next transmission as in one-time pad cf. Example 10.3.1. Thus $|K|$ should be minimized, because key values have to be transmitted every time. E.g. if $K = \{0, 1\}^t$, then keys of length $\log_2 |K| = l$ are to be transmitted.

Let us now concentrate on item (1); items (2) and (3) are considered in the next sub-sections, where different constructions of authentication codes are presented. We would like to see what values can be achieved by $P_I$ and $P_S$ and under which circumstances do they achieve minimal possible values. Basic results are collected in the following proposition.
Proposition 10.3.2 Let the authentication code with the data $S, T, K, A, p_S, p_K$ be fixed. We have

1. $P_I \geq 1/|T|$. Moreover, $P_I = 1/|T|$ iff $\pi(s, t) = 1/|T|$ for all $s \in S, t \in T$.

2. $P_S \geq 1/|T|$. Moreover, $P_S = 1/|T|$ iff $\pi_{s,t}(s', t') = 1/|T|$ for all $s, s' \in S, s \neq s' ; t, t' \in T$.

3. $P_I = P_S = 1/|T|$ iff $\pi(s, t)\pi_{s,t}(s', t') = 1/|T|^2$ for all $s, s' \in S, s \neq s' ; t, t' \in T$.

Proof.

1. For a fixed source state $s \in S$ we have

\[ \sum_{t \in T} \pi(s, t) = \sum_{t \in T} \sum_{k \in K : \pi_k(s) = t} p_K(k) = \sum_{k \in K} p_K(k) = 1. \]

Thus for every $s \in S$ there exists an authentication tag $t = t(s) \in T$, such that $\pi(s, t(s)) \geq 1/|T|$. Now the claim follows by the computation of $P_I$ we made above. Note that equality is possible iff $\pi(s, t) = 1/|T|$ for all $s \in S, t \in T$.

2. For different fixed source states $s, s' \in S$ and a tag $t \in T$, such that $(s, t)$ is valid we have

\[ \sum_{t' \in T} \pi_{s,t}(s', t') = \sum_{t' \in T} \sum_{k \in K : \pi_k(s') = t'} p_K(k) \]

\[ = \sum_{k \in K : \pi_k(s) = t} \sum_{k \in K : \pi_k(s') = t'} p_K(k) \]

\[ = \sum_{k \in K : \pi_k(s) = t} p_K(k). \]

So for every $s', s, t, s \neq s'$ there exists a tag $t' = t'(s') ; \pi_{s,t}(s', t'(s')) \geq 1/|T|$. Now the claim follows by the computation of $P_S$ we made above. Note that equality is possible iff $\pi_{s,t}(s', t') = 1/|T|$ for all $s \in S, t \in T$, due to the definition of $p_{s,t}$.

3. If $P_I = P_S = 1/|T|$, then $\pi(s, t) = 1/|T|$ for all $s \in S, t \in T$ and $\pi_{s,t}(s', t') = 1/|T|$ for all $s, s' \in S, s \neq s' ; t, t' \in T$. For all $s, s' \in S, s \neq s' ; t, t' \in T$ we have $\pi(s, t)\pi_{s,t}(s', t') = 1/|T|^2$.

If $\pi(s, t)\pi_{s,t}(s', t') = 1/|T|^2$ for all $s, s' \in S, s \neq s' ; t, t' \in T$, then due to the equality

\[ \pi(s, t) = \pi(s, t) \sum_{t' \in T} \pi_{s,t}(s', t') = \sum_{t' \in T} \pi(s, t)\pi_{s,t}(s', t') = \sum_{t' \in T} \frac{1}{|T|^2} = \frac{1}{|T|^2}, \]

so $P_I = 1/|T|$ by (1). Now

\[ \pi_{s,t}(s', t') = \frac{1}{|T|^2\pi(s, t)} = \frac{1}{|T|}. \]

So $P_S = 1/|T|$ by (2).
As a straightforward consequence we have

**Corollary 10.3.3** With the notation as above and assuming that \( p_k \) is the uniform distribution (keys are equiprobable), we have \( P_I = P_S = 1/|T| \) iff

\[
|\{ k \in \mathcal{K} : a_k(s') = t', a_k(s) = t \}| = \frac{|\mathcal{K}|}{|T|^2},
\]

for all \( s, s' \in \mathcal{S}, s \neq s'; t, t' \in \mathcal{T} \).

### 10.3.2 Authentication codes and other combinatorial objects

**Authentication codes from orthogonal arrays**

Now we take a look at certain combinatorial objects, called orthogonal arrays that can be used for constructing authentication systems. A bit later we also consider a construction that uses error-correcting codes. For the definitions and basic properties of orthogonal arrays the reader is referred to Chapter 5, Section 5.5.1. What is important for us is that orthogonal arrays yield a construction of authentication codes in quite a natural way. The next proposition shows a relation between orthogonal arrays and authentication codes.

**Proposition 10.3.4** If there exists an orthogonal array \( OA(n, l, \lambda) \) with symbols from the set \( \mathcal{N} \) with \( n \) elements, then one can construct an authentication code with \( |\mathcal{S}| = l, |\mathcal{K}| = \lambda n^2, \mathcal{T} = \mathcal{N} \) and thus \( |\mathcal{T}| = n \), for which \( P_I = P_S = 1/n \). Conversely, if there exists an authentication code with the above parameters, then there exists an orthogonal array \( OA(n, l, \lambda) \).

**Proof.** Consider \( OA(n, l, \lambda) \) as an array representation of an authentication code from Section 5.5.1. Moreover, set \( p_K \) to be uniform, i.e. \( p_K(k) = 1/(\lambda n^2) \) for every \( k \in \mathcal{K} \). Then values of parameters of such a code easily follow. In order to obtain values for \( P_I \) and \( P_S \) use Corollary 10.3.3. Indeed, \( |\{ k \in \mathcal{K} : a_k(s') = t', a_k(s) = t \}| = \lambda \) by the definition of an orthogonal array, but \( \lambda = |\mathcal{K}|/|\mathcal{T}|^2 \). The claim now follows. The converse if proved analogously.

Let us now consider which criteria should be met by orthogonal arrays in order to produce good authentication codes. Parameters estimates for orthogonal arrays in terms of authentication codes parameters \( n, l, \lambda \) follow directly from the above proposition.

- If we set that deception probabilities should be at most some value: \( P_I \leq \epsilon, P_S \leq \epsilon \), then an orthogonal array should have \( n \geq 1/\epsilon \).
- As we can always remove some columns from an orthogonal array and still obtain one after removal, we demand that \( l \geq |\mathcal{S}| \).
- \( \lambda \) should be minimized under constraints imposed by the previous two items. This is due to the fact that we would like to keep key space size as low as possible, as has already been noted in the previous sub-section.
Finally, we present without proofs two characterization results, which say that if one wants to construct authentication codes with minimal deception probabilities, one cannot avoid using orthogonal arrays.

**Theorem 10.3.5** Assume there exists an authentication code defined by $S, T, K, A, p_K, p_S$ with $|T| = n$ and $P_T = P_S = 1/n$. Then:

1. $|K| \geq n^2$. The equality is achieved iff there exists an orthogonal array $OA(n,l,1)$ with $l = |S|$ and $p_K(k) = 1/n^2$ for every $k \in K$.

2. $|K| \geq l(n - 1) + 1$. The equality is achieved iff there exists an orthogonal array $OA(n,l,\lambda)$ with $l = |S|, \lambda = (l(n - 1) + 1)/n^2$ and $p_K(k) = 1/(l(n - 1) + 1)$ for every $k \in K$.

**Authentication codes from error-correcting codes**

As we have seen above, if one wants to keep deception probabilities minimal, one has to deal with orthogonal arrays. A significant drawback of this approach is that the key space grows linearly in size of the source state set. In particular we have from Theorem 10.3.5 (2.) that $|K| > l \geq |S|$. This means that amount of information that needs to be transmitted secretly is larger than the one that is allowed to go through a public channel. The same problem occurs in the one-time pad scheme, Example ???. Of course, this is not quite practical.

In this sub-section we consider so-called almost universal and almost strongly universal hash functions. By means of these functions it is possible to construct authentication codes with deception probabilities slightly larger than minimal, but size of the source state set of which grows exponentially in the key space size. This gives an opportunity to work with much shorter keys sacrificing security threshold a bit.

Next we give a definition of an almost universal hash function.

**Definition 10.3.6** Let $X$ and $Y$ be some sets of cardinality $n$ and $m$ respectively. Consider the family $H$ of functions $f : X \to Y$. Denote $N := |H|$. We call a family $H$ $\epsilon$-almost universal, if for every two different $x_1, x_2 \in X$ the number of functions $f$ from $H$ such that $f(x_1) = f(x_2)$ is $\leq \epsilon N$. Notation for such a family is $\epsilon - AU(N,n,m)$.

There is a natural connection between almost universal hash functions and error-correcting codes as is shown next.

**Proposition 10.3.7** The existence of one of the two objects below implies the existence of the other:

1. $H = \epsilon - AU(N,n,m)$ family of almost universal hash functions.

2. An $m$-ary error-correcting code $C$ of length $N$, cardinality $n$ and relative minimum distance $d/N \geq 1 - \epsilon$.

**Proof.** Let us first describe $\epsilon - AU(N,n,m)$ as an array, similarly to how we have done it for orthogonal arrays. Rows of the representation array are indexed by functions from $H$ and columns by the set $X$. On the place indexed by $f \in H$ and $x \in X$ we write $f(x) \in Y$. Now the equivalence becomes clear.
Indeed, consider this array also as a code-book for an error-correcting code $C$, so that the codewords are written in columns. It is clear that the length is the number of rows, $N$, cardinality is the number of columns, $n$. Entries of the array take their values from $Y$, thus $C$ is an $m$-ary code. Now the definition of $H$ implies that for any two codewords $x_1$ and $x_2$ (columns), the number of positions where they agree is $\leq \epsilon N$. But $d(x_1, x_2)$ is the number of positions where they disagree, so $d(x_1, x_2) \geq (1-\epsilon)N$, so $d/N \geq 1-\epsilon$. The reverse implication is proven analogously.

Next we define almost strongly universal hash functions that are used for authentication.

**Definition 10.3.8** Let $X$ and $Y$ be sets of cardinality $n$ and $m$ respectively. Consider a family $H$ of functions $f : X \to Y$. Denote $N := |H|$. We call a family $H \epsilon$-almost strongly universal, if the following two conditions hold:

1. For every $x \in X$ and $y \in Y$ the number of functions $f$ from $H$ such that $f(x) = y$ is $N/m$.

2. For every two different $x_1,x_2 \in X$ and every $y_1,y_2 \in Y$ the number of functions $f$ from $H$ such that $f(x_i) = y_i$, $i = 1,2$ is $\leq \epsilon \cdot N/m$.

Notation for such a family is $\epsilon - ASU(N,n,m)$.

Almost strongly universal hash functions are nothing but authentication codes with some conditions on the deception probabilities. The following proposition is quite straightforward and is left to the reader as an exercise.

**Proposition 10.3.9** If there exists a family $H$ which is $\epsilon - ASU(N,n,m)$, then there exists an authentication code with $K = H, S = X, T = Y$, $p_K$ a uniform distribution, such that $P_1 = 1/m$ and $P_2 \leq \epsilon$.

Note that if $\epsilon = 1/m$ in Definition 10.3.8, then from Proposition 10.3.9, 10.3.2 (2), 10.3.4 we see that $\epsilon - ASU(N,n,m)$ is actually an orthogonal array. The problem with orthogonal arrays has already been mentioned above. Note that with almost strongly universal hash functions we have more freedom, as we can make $\epsilon$ a bit larger, but gaining in other parameters, as we will see below. So for us it is interesting to be able to construct good $ASU$-families. There are two methods of doing so based on coding theory:

1. Construct $AU$-families from codes as per Proposition 10.3.7 and then use Stinson’s composition method, Theorem 10.3.10 below.

2. Construct $ASU$-families directly from error-correcting codes.

Here we consider (1). For (2) see the Notes. The next result due to Stinson enables one to construct $ASU$-families from $AU$-families and some previously constructed $ASU$-families; we omit the proof thereof.

**Theorem 10.3.10** Let $X,Y,U$ be sets of cardinality $n,m,u$ resp. Let $H_1$ be an $AU$-family $\epsilon_1 - AU(N_1,n,u)$ of functions $f_1 : X \to U$ and let $H_2$ be an $ASU$-family $\epsilon_2 - ASU(N_2,u,m)$ of functions $f_2 : U \to Y$. Consider a family $H$ of all possible compositions thereof: $H = \{ f \mid f = f_2 \circ f_1, f_i \in H_i, i = 1,2 \}$. Then $H$ is $\epsilon - ASU(N,n,m)$, where $\epsilon = \epsilon_1 + \epsilon_2 - \epsilon_1 \epsilon_2$ and $N = N_1 N_2$. 
CHAPTER 10. CRYPTOGRAPHY

Table 10.2: For Exercise 10.3.1

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

One example of the idea (1.) that employs Reed-Solomon codes is given in Exercise 10.3.2. Note that from Exercise 10.3.2 and Proposition 10.3.9 follows that there exists an authentication code with $|S| = |K|^{ (2/5)|K|^{ 1/5}}$ (set $a = 2b$) and $P_I = 1/|T|, P_S = 2/|T|$. So by allowing the probability of substitution deception to rise just two times from the minimal value, we obtain that $|S|$ grows exponentially in $|K|$, which was not possible with orthogonal arrays, where always $|K| > |S|$.

10.3.3 Exercises

10.3.1 An authentication code is represented by the array in Table 10.2 (cf. Sections 10.3.1, 5.5.1). The distributions $p_K$ and $p_S$ are given as follows: $p_S(1) = p_S(3) = 1/4, p_S(2) = 1/2; p_K(1) = p_K(2) = p_K(3) = 1/6, p_K(4) = 1/2$. Compute $P_I$ and $P_S$.

*Hint:* For computing the sums use the following: e.g. for the sum

$$\sum_{k \in K: a_k(s) = t} p_K(k)$$

look at the column corresponding to $s$ and look at which rows entry $t$ appear, then sum up probabilities that correspond to marked rows (they are indexed by keys).

10.3.2 Consider a $q$-ary $[q,k,q-k+1]$ Reed-Solomon code.

- Construct the corresponding $AU$-family using Proposition 10.3.7. What are parameters thereof?

It is known that for natural numbers $a,b : a \geq b$ and $q$ a prime power there exists an $ASU$-family $1/q^b - ASU(q^{a+b}, q^a, q^b)$. Using Stinson’s composition, Theorem 10.3.10,

- prove that there exists an $ASU$-family $2/q^b - ASU(q^{2a+b}, q^{a+b}, q^b)$ with $\epsilon < 1/q^a + 1/q^b$.

10.4 Secret sharing

In the model of symmetric (Section 10.1) and asymmetric (Section 10.2) cryptography a one-to-one relation between Alice and Bob is assumed, maybe with
10.4. SECRET SHARING

A trusted party in the middle. This means that Alice and Bob have necessary pieces of secret information to convey the communication between them. Sometimes it is necessary to distribute this secret information among several participants. Possible scenarios of such applications are: distributing the secret information among the participants in a way so that even if some participants lost some pieces of this secret information it is still possible to reconstruct the whole secret; also sometimes shared responsibility is required, i.e. some action is to be triggered only when several participants combine their secret pieces of information to form the one that triggers that action. Examples of the latter one could be triggering some military action (e.g. missile launch) by several authorized persons (e.g. a president, higher military officials) or opening a bank vault by several top-officials of a bank. In this section we consider mathematical means to achieve the goal. The schemes providing such functionality are called secret sharing schemes. We consider in detail the first such scheme proposed by Adi Shamir in 1979. Then we also briefly demonstrate how error-correcting codes can be used for a construction of linear secret sharing schemes.

In secret sharing schemes shares are produced from the secret to be shared. These shares are then assigned to participants of the scheme. The idea is that if several authorized participants gather in a group that is large enough they should be able to reconstruct the secret using knowledge of their shares. On the contrary if a group is too small or some outsiders decide to find out the secret, their knowledge should not be enough to figure it out. This leads to the following definition.

**Definition 10.4.1** Let \( S_i, i = 1, \ldots, n \) be the shares that are produced from the secret \( S \). Consider a collection of \( n \) participants where each participant is assigned his/her share \( S_i \). A \((t,n)\) threshold scheme is a scheme where every group of \( t \) (or more) participants out of \( n \) can obtain the secret \( S \) using their shares. On the other hand any group of less than \( t \) participants should not be able to obtain \( S \).

We next present the Shamir’s secret sharing scheme that is a classical example of a \((t,n)\) threshold scheme for any \( n \) and \( t \leq n \).

**Algorithm 10.4.2** (Shamir’s secret sharing scheme)

**Set-up:** taking \( n \) as input prepare the scheme for \( n \) participants

1. Choose some prime power \( q > n \) and fix a working field \( \mathbb{F}_q \) that will be used for all the operations in the scheme.

2. Assign to the \( n \) participants \( P_1, \ldots, P_n \) some distinct non-zero elements \( x_1, \ldots, x_n \in \mathbb{F}_q^* \).

**Input:** The threshold value \( t \), the secret information \( S \) in some form.

**Output:** The secret \( S \) is shared among \( n \) participants.

**Generation and distribution of shares:**

1. Encode a secret to be shared as an element \( S \in \mathbb{F}_q \). If this is not possible redo the Set-up phase with a larger \( q \).

2. Choose randomly \( t-1 \) elements \( a_1, \ldots, a_{t-1} \in \mathbb{F}_q \). Assign \( a_0 := S \) and form the polynomial \( f(X) = \sum_{i=0}^{t-1} a_i X^i \in \mathbb{F}_q[X] \).
3. For $i = 1, \ldots, n$ do
   
   - compute value $y_i = f(x_i), i = 1, \ldots, n$ and assign the value $y_i$ to $P_i$.

**Computing the secret from the shares:**

1. Any $t$ participants $P_1, \ldots, P_t$, pull their shares $y_1, \ldots, y_t$ together and then using e.g. Lagrange interpolation with $t$ interpolation points $(x_i, y_i), \ldots, (x_i, y_t)$ restore $f$ and thus $a_0 = S = f(0)$.

The part "Computing the secret from the shares" is clearly justified by the following formulas of Lagrange interpolation (w.l.o.g. the first $t$ participants pull their shares):

$$f(X) = \sum_{i=1}^{t} y_i \prod_{j \neq i} \frac{X - x_j}{x_i - x_j},$$

so that $f(x_i) = y_i, i = 1, \ldots, t$ and $f$ is a unique polynomial of degree $\leq t - 1$ with this property. Of course the participants do not have to reconstruct the whole $f$, they just need to know $a_0$ that can be computed as

$$S = a_0 = \sum_{i=1}^{t} c_i y_i, \quad c_i = \prod_{j \neq i} \frac{x_j}{x_j - x_i}. \quad (10.2)$$

So every $t$ or more participants can recover the secret value $S = f(0)$. On the other hand it is possible to show that for any $t - 1$ shares (w.l.o.g. the first ones) $(x_i, y_i), i = 1, \ldots, t - 1$ and any $a \in \mathbb{F}_q$ there exists a polynomial $f_a$ such that its evaluation at 0 is $a$. Indeed, take $f_a(X) = a + X f_a(X)$, where $f_a(X)$ is the Lagrange polynomial of degree $\leq t - 2$ such that $f_a(x_i) = (y_i - a)/x_i, i = 1, \ldots, t - 1$ (recall that $x_i$'s are non-zero). Then $\deg f_a \leq t - 1, f_a(x_i) = y_i$, and $f_a(0) = a$. So this means that any $t - 1$ (or less) participants have no information about $S$: the best they can do is to guess the value of $S$, the probability of such a guess is $1/q$. This is because, to their knowledge, $f$ can be any of $f_a$'s.

**Example 10.4.3** Let us construct a $(3, 6)$ Shamir’s threshold scheme. Take $q = 8$ and fix the field $\mathbb{F}_8 = \mathbb{F}_2[\alpha]/(\alpha^3 + \alpha + 1)$. Element $\alpha$ is a generating element of $\mathbb{F}_8^*$. For $i = 1, \ldots, 6$ assign $x_i = \alpha^i$ to the participant $P_i$. Suppose that the secret $S = \alpha^5$ is to be shared. Choose $a_1 = \alpha^3, a_2 = \alpha^5$, so that $f(X) = \alpha^5 + \alpha^3 X + \alpha^6 X^2$. Now evaluate $y_1 = f(\alpha) = \alpha^4, y_2 = f(\alpha^2) = \alpha^3, y_3 = f(\alpha^3) = \alpha^6, y_i = f(\alpha^i) = \alpha^5$. For every $i = 1, \ldots, 6$ assign $y_i$ as a share for $P_i$. Now suppose that the participants $P_2, P_3,$ and $P_5$ decide to pull their shares together and obtain $S$. As in (10.2) they compute

$$c_2 = \frac{y_2}{x_2 - x_5}, \frac{y_3}{x_3 - x_5} - 1, c_3 = 1, c_5 = 1.$$  

Accordingly, $c_2 y_2 + c_3 y_3 + c_5 y_5 = \alpha^5 = S$. On the other hand, due to the explanation above, any 2 participants cannot deduce $S$ from their shares. In other words, any element of $\mathbb{F}_8$ is equally likely for them to be the secret.

See Exercise 10.4.1 for a simple construction of a $(t, t)$ threshold scheme.
these are columns and they are not to be confused with the usual notation for rows of $G$. Choose some information vector $a \in \mathbb{F}_q^k$ such that $S = ag_i^\perp$, where $S$ is the secret information. Then compute $s = (s_0, s_1, \ldots, s_{n-1}) = aG$. Now $s_0 = S$ and $s_1, \ldots, s_{n-1}$ can be used as shares. The next result characterizes a situation when the secret $S$ can be obtained from the shares.

**Proposition 10.4.4** With the notation as above, let $s_1, \ldots, s_m$ be some shares for $1 \leq m \leq n - 1$. These shares can reconstruct the secret $S$ iff $c^\perp = (1, 0, \ldots, 0, c_{i_1}, 0, \ldots, 0, c_{i_m}, 0, \ldots, 0) \in C^\perp$, where at least one $c_{i_j} \neq 0$.

**Proof.** The claim follows from the fact that $G \cdot c^\perp = 0$ and that the secret $S = ag_i^\perp$ can be obtained if $g_i^\perp$ is a linear combination of $g_{i_1}^\perp, \ldots, g_{i_m}^\perp$. ∎

If we carefully look one more time at Shamir’s scheme it is not a surprise that it can be seen as the above construction with Reed-Solomon code as a code $C$. Indeed, choose $N = q-1$ and set $x_i = \alpha^i$ where $\alpha$ is a primitive element of $\mathbb{F}_q$. It is then quite easy to see that encoding the secret and shares via the polynomial $f$ as in Algorithm 10.4.2 is equivalent to encoding via the Reed-Solomon code $RS_t(N, 1)$, cf. Definition 8.1.1 and Proposition 8.1.4. The only nuance is that in general we may assign some $n \leq N$ shares and not all $N$. Now we need to see that every collection of $t$ shares reconstructs the secret. Using the above notation, let $s_1, \ldots, s_i$ be the shares pulled together. According to Proposition 10.4.4 the dual of $C = RS_t(N, 1)$ should contain a codeword with the 1 at the first position and at least one non-zero element at positions $i_1, \ldots, i_t$. From Proposition 8.1.2 we have that $RS_t(N, 1)^\perp = RS_{N-t}(N,N)$ and $RS_{N-t}(N,N)$ is an MDS $[N,N-t,t+1]$ code. We use now Corollary 3.2.14 with the $t+1$ positions $1, i_1, \ldots, i_t$ and we are guaranteed to have a prescribed codeword. Therefore every collection of $t$ shares reconstructs the secret. Having $x_i = \alpha^i$ is not really a restriction (Exercise 10.4.3).

In general the problem of constructing secret sharing schemes can be reduced to finding codewords of minimum weight in a dual code as per Proposition 10.4.4. There are more advanced constructions based on error-correcting codes, in particular based on AG-code, see Notes for the references.

It is clear that if a group of participants can recover the secret by combining their shares, then any group of participants containing this group can also recover the secret. We call a group of participants a **minimal access set**, if the participants of this group can recover the secret with their shares, while any proper subgroup of participants can not do so. From the preceding discussions, it is clear that there is one-to-one correspondence between the set of minimal access sets and the set of minimal weight codewords of the dual code $C^\perp$ whose first coordinate is 1. Therefore, for a secret sharing scheme based on a code $C$, the problem of determining the access structure of the secret sharing scheme is reduced to the problem of determining the set of minimal weight codewords whose first coordinate is 1. It is obvious that the shares for the participants depend on the selection of the generator matrix $G$ of the code $C$. However, by Proposition ??, the selection of generator matrix does not affect the access structure of the secret sharing scheme.

Note that the set of minimal weight codewords whose first coordinate is 1 is a subset of the set of all minimal weight codewords. The problem of determining the set of all minimal weight codewords of a code is known as the **covering**
problem. This problem is a hard problem for an arbitrary linear code. In the following, let us have some more discussions on the access structure of secret sharing schemes based on special classes of linear codes. It is clear that for any participant, he (she) must be in at least one minimal access set. This is true for any secret sharing scheme. Now, we further ask the following question: Given a participant \( P_i \), how many minimal access sets are there which contain \( P_i \)? This question is solved if the dual code of the code used by the secret sharing scheme is a constant weight code. In the following proposition, we suppose \( C \) is a \( q \)-ary \([n,k]\) code, and \( G = (g'_0, g'_1, \ldots, g'_{n-1}) \) is a generator matrix of \( C \).

**Proposition 10.4.5** Suppose \( C \) is a constant weight code. Then, in the secret sharing scheme based on \( C^\perp \), there are \( q^k-1 \) minimal access sets. Moreover, we have the following:

1. If \( g'_i \) is a scalar multiple of \( g'_0 \), \( 1 \leq i \leq n-1 \), then every minimal access set contains the participant \( P_i \). Such a participant is called a dictatorial participant.
2. If \( g'_i \) is not a scalar multiple of \( g'_0 \), \( 1 \leq i \leq n-1 \), then there are \( (q-1)q^{k-1} \) minimal access sets which contain the participant \( P_i \).

**Proof.** ..........will be given later.......... \( \Box \)

The following problem is an interesting research problem: Identify (or construct) linear codes which are good for secret sharing, that is, the covering problem can be solved, or the minimal weight codewords can be well characterized. Several classes of linear codes which are good for secret sharing have been identified, see the papers by C. Ding and J. Yuan.

### 10.4.1 Exercises

10.4.1 Suppose that some trusted party \( T \) wants to share a secret \( S \in \mathbb{Z}_m \) between two participants \( A \) and \( B \). For this, \( T \) generates some random number \( a \in \mathbb{Z}_m \) and assigns it to \( A \). \( T \) then assigns \( b = S - a \mod m \) to \( B \).

- Show that the scheme above is a \((2,2)\) threshold scheme. This scheme is an example of a **split-knowledge** scheme.

- Generalize the idea above to construct a \((t,t)\) threshold scheme for arbitrary \( t \).

10.4.2 Construct a \((4,7)\) Shamir’s threshold scheme and share the bit-string "1011" using it.

**Hint:** Represent the bit-string "1011" as an element of a finite field with more than 7 elements.

10.4.3 Remove the restriction on \( x_i \) being equal to \( \alpha^i \) in the Reed-Solomon construction of Shamir’s scheme by using Proposition 3.2.10.
10.5 Basics of stream ciphers. Linear feedback shift registers

In Section 10.1 we have seen how block ciphers are used for construction of symmetric cryptosystems. Here we give some basics of stream ciphers, i.e. ciphers that proceed information bitwise as oppose to blockwise. Stream ciphers are usually faster than block ciphers and have lower requirements on implementation costs. Nevertheless, stream ciphers appear to be more susceptible to cryptanalysis. Therefore, much care should be put in designing a secure cipher. In this section we concentrate on stream cipher design that involves the linear feedback shift register (LFSR) as one of the building blocks.

The difference between block and stream ciphers is quite vague, since a block cipher can be turned to a stream one using some special mode of operation. Nevertheless, let us see what are characterizing features of such ciphers. A stream cipher is defined via its stream of states \( S \), the keystream \( K \), and the stream of outputs \( C \). Having an input (plaintext) stream \( P \), one would like to obtain \( C \) using \( S \) and \( K \) by operating successively on individual units of these streams. The streams \( C \) and \( K \) are obtained using some key, either secret or not. If these units are binary bits, we are dealing with the binary cipher.

Consider an infinite sequence (a stream) of key bits \( k_1, k_2, k_3, \ldots \), and a stream of plaintext bits \( p_1, p_2, p_3, \ldots \). Then we can form a ciphertext stream by simply adding the key stream and the plaintext stream bitwise:

\[
  c_i = p_i \oplus k_i, i = 1, 2, 3, \ldots
\]

One can stop at some moment \( n \) thus having the \( n \)-bit ciphertext from the \( n \)-bit key and the \( n \)-bit plaintext. If \( k_i \)'s are chosen uniformly at random and independently, we have the one-time pad scheme. It can be shown that in the one-time pad if an eavesdropper only possesses the ciphertext, he/she cannot say anything about the plaintext. In other words, the knowledge of the ciphertext does not shed any additional light on the plaintext for an eavesdropper. Moreover, an eavesdropper even knowing \( n \) key bits is completely uncertain about the \( (n + 1) \)-th bit. This is a classical example of unconditionally secure cryptosystem, cf. Definition 10.1.8.

Although the above idea yields provable guarantees for security it has an essential drawback: a key should be at least as long as a plaintext, which is a usual thing in unconditionally secure systems, see also Section 10.3.1. Clearly this requirement is quite impractical. That is why what is usually done is the following. One starts with some bitstring of a fixed size called a seed, and then by making some operations with this string obtains some larger string (it can be infinite theoretically), which should “appear random” to an eavesdropper. Note that since the seed is finite we cannot talk about unconditional security anymore, only computational. Indeed, having long enough key stream in the known-plaintext scenario, it is in principle possible to run an exhaustive search on all possible seeds to find out the one that gives rise to the given key stream. In particular all the successive bits of the key stream will be known.

Now let us present two commonly used types of stream ciphers: synchronous and self-synchronizing. Let \( P = \{p_0, p_1, \ldots \} \) be the plaintext stream, \( K = \{k_0, k_1, \ldots \} \) be the keystream, \( C = \{c_0, c_1, \ldots \} \) be the ciphertext stream, and \( S = \{s_0, s_1, \ldots \} \) be the state stream. The synchronous stream cipher synchronous
stream cipher is defined as follows:

\[
\begin{align*}
    s_{i+1} &= f(s_i, k), \\
    k_i &= g(s_i, k), \\
    c_i &= h(k_i, p_i), i = 0, 1, \ldots.
\end{align*}
\]

Here \(s_0\) is the initial state and \(f\) is the state function, which generates a next state from the previous one and also depends on a key. Now \(k_i\)'s form the key stream via the function \(g\). Finally the ciphertext is formed by applying the output function \(h\) to the bits \(k_i\) and \(p_i\). This cipher is called synchronous, since both Alice and Bob need to use the same key stream \((k_i)\) i.e., if some (non-)malicious insertions/deletions occur, the synchronization is lost, so additional means for providing synchronization are necessary. Note that usually the function \(h\) is just a bitwise addition of streams \((k_i)\) and \((p_i)\). It is also very common for stream ciphers to have an initialization phase, where only the states \(s_i\) are updated first and the update and output starts to happen only at some later point of time. Therewith the key stream \((k_i)\) gets more complicated and dependent on more state bits.

The self-synchronizing stream cipher is defined as

\[
\begin{align*}
    s_i &= (c_{i-t}, \ldots, c_{i-1}), \\
    k_i &= g(s_i, k), \\
    c_i &= h(k_i, p_i), i = 0, 1, \ldots.
\end{align*}
\]

Here \((c_{t-1}, \ldots, c_{-1})\) is a non-secret initial state. So the encryption/decryption depends only on some number of ciphertext bits, therefore the output stream is able to recover from deletions/insertions.

Observe that if \(h\) is a bitwise addition modulo 2, then the stream ciphers described above follow the idea of the one-time pad. The difference is that now one obtains the key stream \((k_i)\), not fully randomly, but as a pseudorandom expansion of an initial state (seed) \(s_0\). The LFSR is used as a building block in many stream ciphers that facilitates such a pseudorandom expansion. The LFSRs have an advantage that they can be efficiently implemented in hardware. Also the outputs of LFSRs have nice statistical properties. Moreover, LFSRs are closely related to so-called linear recurring sequences that are readily studied via algebraic methods.

Schematically an LFSR can be presented as in Figure 10.3

Let us figure out what is going on the diagram. First the notation. A square box is a delay box sometimes called ”flip-flop”. Its task is to pass its stored value further after each unit of time set by a synchronizing clock. A circle with the value \(a_i\) in it performs AND operation or multiplication modulo 2 on the input with the prescribed \(a_i\). The plus sign in a circle clearly means the XOR operation or addition modulo 2. Now the square boxes are initialized with some values, namely the box \(D_i\) gets some value \(s_i \in \{0, 1\}, i = 0, \ldots, L - 1\). When the first time unit comes to an end the following happens: the value \(s_0\) becomes an output bit. Then all values \(s_i, i = 1, \ldots, L - 1\) are shifted from \(D_i\) to \(D_{i-1}\). Simultaneously for each \(i = 0, \ldots, L - 1\) the value \(s_i\) goes to an AND-circle, gets multiplied with \(a_i\), and then all these products are summed up by means of plus-circles, so that the sum \(\oplus_{i=0}^{L-1} a_i s_i\) is formed. This sum is written to \(D_{L-1}\) and is called \(s_2\). The same procedure takes place at the end of the next time unit: now \(s_1\) is the output, the remaining values are shifted,
10.5. BASICS OF STREAM CIPHERS. LINEAR FEEDBACK SHIFT REGISTERS

and \( s_{L+1} = \oplus_{i=1}^{L} a_{i-1} s_i \) is written to \( D_{L-1} \). Analogously one proceeds further. The name “Linear Feedback Shift Register” is clear now: we use only linear operations here (multiplication by \( a_i \)’s and addition), the values that appear in \( D_0, \ldots, D_{L-2} \) give feedback to \( D_{L-1} \) by means of a sum of the type described, also the values are being shifted from \( D_i \) to \( D_{i-1} \).

Algebraically LFSRs are studied via the notion of linear recurring sequences, which we introduce next.

**Definition 10.5.1** Let \( L \) be a positive integer, let \( a_0, \ldots, a_{L-1} \) be some values from \( \mathbb{F}_2 \). A sequence \( S \), which first \( L \) elements are \( s_0, \ldots, s_{L-1} \) that are values from \( \mathbb{F}_2 \) and the defining rule is

\[
s_{L+i} = a_{L-1} s_{L+i-1} + a_{L-2} s_{L+i-2} + \cdots + a_0 s_i, \quad i \geq 0,
\]

is called the \((L\text{-th order})\) homogeneous linear recurring sequence in \( \mathbb{F}_2 \). The elements \( s_0, \ldots, s_{L-1} \) are said to form the initial state sequence.

Obviously, a homogeneous linear recurring sequence represents an output of some LFSR and vice versa, so we will use the both notions interchangeably. Another important notion that comes along with the linear recurring sequences is the following.

**Definition 10.5.2** Let \( S \) be an \( L \)-th order homogeneous linear recurring sequence in \( \mathbb{F}_2 \) defined by (10.3). Then the polynomial

\[
f(X) = X^L + a_{L-1}X^{L-1} + \cdots + a_0 \in \mathbb{F}_2[X]
\]

is called the characteristic polynomial of \( S \).

**Remark 10.5.3** The characteristic polynomial is also sometimes defined as \( g(X) = 1 + a_{L-1}X + \cdots + a_0X^L \) and is called connection or feedback polynomial. We have \( g(X) = f(1/X) \). Everything that will be said about \( f(X) \) in the sequel remains true also for \( g(X) \).
Example 10.5.4 On Figure 10.4 (a), (b), and (c) the diagrams for LFSRs with the characteristic polynomials \( X^2 + X + 1 \), \( X^2 + 1 \), \( X^2 + X \) are depicted. We removed the circles and sometimes also connected lines, since we are working in \( \mathbb{F}_2 \), so \( a_i \in \{0, 1\} \). The table for the case (a) looks like this

<table>
<thead>
<tr>
<th>( D_0 )</th>
<th>( D_1 )</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

So we see that the output sequence is actually periodic with period 3. The value 3 for the period is maximum one can get for \( L = 2 \). This is due to the fact that \( X^2 + X + 1 \) is irreducible and moreover primitive, see below Theorem 10.5.8.

For the case (b) we have

<table>
<thead>
<tr>
<th>( D_0 )</th>
<th>( D_1 )</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

and the period is 2. For the case (c) we have

<table>
<thead>
<tr>
<th>( D_0 )</th>
<th>( D_1 )</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

So the output sequence here is not periodic, but is ultimately periodic, i.e. periodic starting at position 2 and the period here is 1. The non-periodicity is due to the fact that for \( f(X) = X^2 + X \), \( f(0) = 0 \), see Theorem 10.5.8.

Example 10.5.5 Let us see how one can handle LFSRs in Magma. In Magma one works with a connection polynomial (Remark 10.5.3). For example we are given the connection polynomial \( f = X^6 + X^4 + X^3 + X + 1 \) and initial state sequence \((s_0, s_1, s_2, s_3, s_4, s_5) = (0, 1, 1, 1, 0, 1)\), then the next state
(s_1, s_2, s_3, s_4, s_5, s_6) can be computed as
> P<X>:=PolynomialRing(GF(2));
> f:=X^6+X^4+X^3+X+1;
> S:=[GF(2)|0,1,1,1,0,1];
> LFSRStep(f,S);
[ 1, 1, 1, 0, 1, 1 ]

By writing
> LFSRSequence(f,S,10);
[ 0, 1, 1, 1, 0, 1, 1, 1, 1, 0 ]
we get the next 10 state values s_6,...,s_{15}.

In Sage one can do the same in the following way
> con_poly=[GF(2)(i) for i in [1,0,1,1,0,1]]
> init_state=[GF(2)(i) for i in [0,1,1,1,0,1]]
> n=10
> lfsr_sequence(con_poly, init_state, n)
[0, 1, 1, 1, 0, 1, 1, 1, 1, 0]

So one has to provide the connection polynomial via its coefficients.

As we have mentioned, the characteristic polynomial plays an essential role
in determining the properties of a linear recurring sequence and the associated
LFSR. Next we summarize all the results concerning a characteristic polynomial,
but first let us make precise the notions of periodic and ultimately periodic
sequences.

Definition 10.5.6 Let \( S = \{ s_i \}_{i \geq 0} \) be a sequence such that there exists a
positive integer \( P \) such that \( s_{P+i} = s_i \ \forall i = 0,1,... \). Such a sequence is called
periodic and \( P \) is a period of \( S \). If the property \( s_{P+i} = s_i \) holds for all \( i \) starting
from some non-negative \( P_0 \), then such a sequence is called ultimately periodic
also with a period \( P \). Note that a periodic sequence is also ultimately periodic.

Remark 10.5.7 Note that periodic and ultimately periodic sequences have
many periods. It turns out that the least period always divides any other
period. We will refer to the term period meaning the least period of a sequence.

Now the main result follows.

Theorem 10.5.8 Let \( S \) be an \( L \)-th order homogeneous linear recurring se-
quenCe and let \( f(X) \in \mathbb{F}_2[X] \) be its characteristic polynomial. The following
holds:

1. \( S \) is an ultimately periodic sequence with period \( P \leq 2^L - 1 \).
2. If \( f(0) \neq 0 \), then \( S \) is periodic.
3. If \( f(X) \) is irreducible over \( \mathbb{F}_2 \), then \( S \) is periodic with period \( n \) such that
   \( P | (2^L - 1) \).
4. If \( f(X) \) is primitive *** recall the definition? *** over \( \mathbb{F}_2 \), then \( S \) is
   periodic with period \( P = 2^L - 1 \).

Definition 10.5.9 A homogeneous linear recurring sequence \( S \) with \( f(X) \) a
primitive characteristic polynomial is called maximal period sequence in \( \mathbb{F}_2 \) or
an m-sequence.
The notions and results above can be generalized to the case of arbitrary finite field \( \mathbb{F}_q \). It is notable that one can compute the characteristic polynomial of an \( L \)-th order homogeneous linear recurring sequence \( S \) by knowing any sub-sequence of length at least \( 2L \) by means of an algorithm by Berlekamp and Massey, which is essentially the one from Section 9.2.2. More details have to be provided here to show the connection. The explanation in 9.2.2 is a bit too technical. Maybe we should introduce a simple version of BM in Chapter 6, which we could then use here? See also Exercise 10.5.3.

Naturally one is interested in obtaining sequences with large period. Therefore \( m \)-sequences have primary application. These sequences have nice statistical properties. For example the distribution of patterns that have length \( \leq L \) is almost uniform. The notion of linear complexity is used as a tool for investigating the statistical properties of outputs of LFSRs. Roughly speaking, a linear complexity of a sequence is a minimal \( L \) such that there exists a homogeneous linear recurring sequence with the characteristic polynomial of degree \( L \). Because of nice statistical properties LFSRs can be used as pseudorandom bit generators, see Notes.

An obvious cryptographic drawback of LFSRs is the fact that the whole output sequence can be reconstructed by having just \( 2L \) bits of it, where \( L \) is the linear complexity of the sequence. This obstructs using LFSRs as cryptographic primitives, in particular as key stream generators. Nevertheless, one can use LFSRs in certain combinations, add non-linearity and obtain quite effective and secure key stream generators for stream ciphers. Let us briefly describe the three possibilities of such combinations.

- **Nonlinear combination generator.** Here one transmits outputs of \( l \) LFSRs \( L_1, \ldots, L_l \) to a non-linear function \( f \) with \( l \) inputs. The output of \( f \) becomes then the key stream. The function \( f \) should be chosen to be correlation immune, i.e. there should be no correlation between the output of \( f \) and outputs of any small subset of \( L_1, \ldots, L_l \).

- **Nonlinear filter generator.** Here the \( L \) delay boxes at every time unit end give their values to a non-linear function \( g \) with \( L \) inputs. The output of \( g \) becomes then the key stream. The function \( g \) is chosen in a way that its algebraic representation is dense.

- **Clock-controlled generator.** Here outputs of one LFSRs control the clocks of other LFSRs that compose the cipher. In this way the non-linearity is introduced.

For some examples of the above, see Notes.

### 10.5.1 Exercises

10.5.1 Consider an example of a synchronous cipher defined by the following data. The initial state is \( s_0 = 10010 \). The function \( f \) shifts its argument by 3 positions to the right and adds 01110 bitwise. Now \( g \) is defined to sum up bits with positions 2, 4, and 5 modulo 2 to obtain a keystream bit. Compute the first 6 key stream bits of such a cipher.

10.5.2 a. The polynomial \( f(X) = X^4 + X^3 + 1 \) is primitive over \( \mathbb{F}_2 \). Draw a diagram of an LFSR that has \( f \) as the characteristic polynomial. Let
10.6. PKC SYSTEMS USING ERROR-CORRECTING CODES

Let \((s_0, s_1, s_2, s_3)\) be the initial state. Compute the output of such LFSR up to the point when it is seen that the output sequence is periodic. What is the period?

b. Rewrite (a.) in terms of a connection polynomial. Take the same initial state and compute (e.g. with Magma) enough output sequence values to see the periodicity.

10.5.3 Let \(s_0, \ldots, s_{2L-1}\) be the first \(2L\) bits of an \(L\)-th order homogeneous linear recurring sequence defined by (10.3). If it is known that the matrix

\[
\begin{pmatrix}
s_0 & s_1 & \ldots & s_{L-1} \\
s_1 & s_2 & \ldots & s_L \\
\vdots & \vdots & \ddots & \vdots \\
s_{L-1} & s_L & \ldots & s_{2L-1}
\end{pmatrix}
\]

is invertible, show that it is possible to compute \(a_0, \ldots, a_{L-1}\), i.e. to find out the structure of the underlying LFSR.

10.5.4 [CAS] The shrinking generator is an example of the clock-controlled generator. The shrinking generator is composed of two LFSR’s \(L_1\) and \(L_2\). The output of \(L_1\) controls the output of \(L_2\) in the following way: if the output bit of \(L_1\) is one, then the output bit of \(L_2\) is taken as an output of the whole generator. If the output of \(L_1\) is zero, then the output of \(L_2\) is discarded. So, in other words, the output of the generator forms a subsequence of the output of \(L_2\) and this subsequence is masked by the 1’s in the output of \(L_1\). Write a procedure that implements the shrinking generator. Then use the output of the shrinking generator as a key-stream \(k\), and define a stream cipher with it, i.e. a ciphertext is formed as \(c_i = p_i \oplus k_i\), where \(p\) is the plaintext stream. Compare your simulation results with the ones obtained with the ShrinkingGeneratorCipher class from Sage.

10.6 PKC systems using error-correcting codes

In this section we consider the public key encryption schemes due to McEliece (Section 10.6.1) and Niederreiter (Section 10.6.2). Both of these encryption schemes rely on hardness of decoding random linear codes as well as hardness of distinguishing a code with the prescribed structure from a random one. As we have seen, the problem of the nearest codeword decoding is NP-hard. So the McEliece cryptosystem is one of the proposals to use an NP-hard problem as a basis, for some others see Section 10.2.3.

As has been mentioned at the end of Section 10.2.1, quantum computer attacks impose a potential threat for classical cryptosystems like RSA (Section 10.2.1) and those based on the DLP problem (Section 10.2.2). On the other side, no significant advantages of using a quantum computer in attacking the code-based schemes of McEliece and Niederreiter are known. Therefore, this area of cryptography attracted quite a lot of attention in the last years. See Notes on the recent developments.
10.6.1 McEliece encryption scheme

Now let us consider the public key cryptosystem by McEliece. It was proposed in 1978 and is in fact one of the oldest public key cryptosystems. The idea of the cryptosystem is to take a class of codes $C$ for which there is an efficient bounded distance decoding algorithm. The secret code $C \in C$ is given by a $k \times n$ generator matrix $G$. This $G$ is scrambled into $G' = SGP$ by means of a $k \times k$ invertible matrix $S$ and an $n \times n$ permutation matrix $P$. Denote by $C'$ the code with the generator matrix $G'$. Now $C'$ is equivalent to $C$, cf. Definition 2.5.15. The idea of scrambling is that the code $C'$ should appear random to an attacker, so it should not be possible to use the efficient decoding algorithm available for $C$ to decrypt messages. More formally we have the following procedures that define the encryption scheme as in Algorithms 10.1, 10.2, and 10.3. Note that in these algorithms when we say “choose” we mean “choose randomly from an appropriate set”.

**Algorithm 10.1 McEliece key generation**

**Input:**
- System parameters:
  - Length $n$
  - Dimension $k$
  - Alphabet size $q$
  - Error-correcting capacity $t$
  - A class $C$ of $[n,k]$ $q$-ary linear codes that have an efficient decoder that can correct up to $t$ errors

**Output:** McEliece public/private key pair $(P_K, S_K)$.  

**Begin**

Choose $C \in C$ represented by a generator matrix $G$ and equipped with an efficient decoder $D_C$.

Choose an invertible $q$-ary $k \times k$ matrix $S$.

Choose an $n \times n$ permutation matrix $P$.

Compute $G' := SGP$ [a generator matrix of an equivalent $[n,k]$ code]

$P_K := G'$.

$S_K := (D_C, S, P)$.

**Return** $(P_K, S_K)$.

**End**

Let us see why the decryption procedure really yields a correct message from a ciphertext. We have $c_1 = cP^{-1} = mSG + eP^{-1}$. Now since $\text{wt}(eP^{-1}) = \text{wt}(e) = t$, we have $c_2 = D_C(c_1) = mS$. The last step is then trivial.

Initially McEliece proposed to use the class of binary Goppa codes (cf. Section 8.3.2) as the class $C$. Interestingly enough, this class turned out to be pretty much the only secure choice up to now. See Section 10.6.3 for the discussion. As we saw in the procedures above, decryption is just a decoding with the code generated by $G'$. So if we are successful in “masking”, for instance a binary Goppa code $C$, as a random code $C'$, then the adversary is faced with the problem of correcting $t$ errors in a random code, which is assumed to be hard, if $t$ is large enough. More on that in Section 10.6.3. Let us consider a specific example.
### Algorithm 10.2 McEliece encryption

**Input:**
- Plaintext \( m \)
- Public key \( P_K = G' \)

**Output:** Ciphertext \( c \).

**Begin**

Represent \( m \) as a vector from \( \mathbb{F}_q^k \).

Choose randomly a vector \( e \in \mathbb{F}_q^n \) of weight \( t \).

Compute \( c := m G' + e \). \{encode and add noise; \( c \) is of length \( n \} \)

**return** \( c \)

**End**

### Algorithm 10.3 McEliece decryption

**Input:**
- Ciphertext \( c \)
- private key \( S_K = (D, S, P) \)

**Output:** Plaintext \( m \).

**Begin**

Compute \( c_1 := c P^{-1} \).

Compute \( c_2 := D(c_1) \).

Compute \( c_3 := c_2 S^{-1} \).

**return** \( c_3 \)

**End**

---

**Example 10.6.1 [CAS]** We use Magma to construct a McEliece encryption scheme based on a binary Goppa code, encrypt a message with it and then decrypt. First we construct a Goppa code of length 31, dimension 16, efficiently correcting 3 errors (see also Example 12.5.23):

```plaintext
> q:=2^5;
> P<x>:=PolynomialRing(GF(q));
> g:=x^3+x+1;
> a:=PrimitiveElement(GF(q));
> L:=[a^i : i in [0..q-2]];
> C:=GoppaCode(L,g); // a [31,16,7] binary Goppa code
> C2:=GoppaCode(L,g^2);
> n:=#L; k:=Dimension(C);
```

Note that we had to define the code \( C2 \) generated by the square of the Goppa polynomial \( g \). Although the two codes are equal, we need the code \( C2 \) later for decoding. *** add references *** Now the key generation part:

```plaintext
> G:=GeneratorMatrix(C);
> S:=Random(GeneralLinearGroup(k,GF(2)));
> Determinant(S); // indeed an invertible map
1
> p:=Random(Sym(n)); // a random permutation of an n-set
> F:=PermutationMatrix(GF(2), p); // its matrix
> GPublic:=S*G*P; // our public generator matrix
```

After we have obtained the public key, we can encrypt a message:
> MessageSpace:=VectorSpace(GF(2),k);
> m:=Random(MessageSpace);
> m;
(1 1 0 0 1 1 1 0 0 0 0 1 1 1 0 0)
> m2:=m*GPublic;
> e:=C ! 0; e[10]:=1; e[20]:=1; e[25]:=1; // add 3 errors
> c:=m2+e;
Let us decrypt using the private key:
> c1:=c*P^-1;
> bool,c2:=Decode(C2,c1: Al:="Euclidean");
> IS:=InformationSet(C);
> ms:=MessageSpace ! [c2[i]: i in IS];
> m_dec:=ms*S^-1;
> m_dec;
(1 1 0 0 1 1 1 0 0 0 0 1 1 1 0 0)
We see that m_dec=m. Note that we applied the Euclidian algorithm for decoding
a Goppa code (**reference**), but we had to apply it to the code generated
by g^2 to be able to correct all three errors. Since as a result of decoding we
obtained a codeword, not the message it encodes, we had to find an information
set and then extract a subvector at positions that correspond to this set (our
generator matrices are in a standard form, so we simply take the subvector).

10.6.2 Niederreiter’s encryption scheme

The scheme proposed by Niederreiter in 1986 is dual to the one of McEliece.
Namely, instead of using generator matrices and words, this scheme uses parity
check matrices and syndromes. Although different in terms of parameter sizes
and efficiency of en-/decryption, the two schemes of McEliece and Niederreiter
actually can be shown to have equivalent security, see the end of this section.
We now present how keys are generated and how en-/decryption is performed in
the Niederreiter scheme in Algorithms 10.4, 10.5, and 10.6. Note that in these
algorithms we use the syndrome decoder. Recall that the notion of a syndrome
decoder is equivalent to the notion of a minimum distance decoder ***add this
section***.

The correctness of the en-/decryption procedures is shown analogously to the
McEliece scheme, see Exercise 10.6.1. The only difference is that here we use
a syndrome decoder, which returns a vector with the smallest non-zero weight
that has the input syndrome, whereas in the case of McEliece the output of
a decoder is the codeword closest to the given word. Let us take a look at a
specific example.

Example 10.6.2 [CAS] We are working in Magma as in Example 10.6.1 and
are considering the same binary Goppa code from there. So the first 8 lines that
define the code are the same; we just add
> t:=Degree(g);
Now the key generation part is quite similar as well:
> H:=ParityCheckMatrix(C);
> S:=Random(GeneralLinearGroup(n-k,GF(2)));
> p:=Random(Sym(n));P:=PermutationMatrix(GF(2), p);
> HPublic:=S*H*P; // our public parity check matrix
Algorithm 10.4 Niederreiter key generation

**Input:**
- System parameters:
  - Length $n$
  - Dimension $k$
  - Alphabet size $q$
  - Error-correcting capacity $t$
  - A class $\mathcal{C}$ of $[n,k]$ $q$-ary linear codes that have an efficient syndrome decoder that corrects up to $t$ errors

**Output:** Niederreiter public/private key pair $(P_K, S_K)$.

**Begin**
Choose $C \in \mathcal{C}$ represented by a parity check matrix $H$ and equipped with an efficient decoder $D_C$.
Choose an invertible $q$-ary $(n-k) \times (n-k)$ matrix $S$.
Choose an $n \times n$ permutation matrix $P$.
Compute $H' := SHP$ \{a parity check matrix of an equivalent $[n,k]$ code\}
$P_K := H'$.
$S_K := (D_C, S, P)$.
**return** $(P_K, S_K)$.
**End**

Algorithm 10.5 Niederreiter encryption

**Input:**
- Plaintext $m$
- Public key $P_K = H'$

**Output:** Ciphertext $c$.

**Begin**
Represent $m$ as a vector from $\mathbb{F}_q^n$ of weight $t$. *** notation! ***
Compute $c := H'm^T$. \{ciphertext is a syndrome\}
**return** $c$
**End**

Algorithm 10.6 Niederreiter decryption

**Input:**
- Ciphertext $c$
- private key $S_K = (D_C, S, P)$

**Output:** Plaintext $m$.

**Begin**
Compute $c_1 := S^{-1}c$.
Compute $c_2 := D_C(c_1)$ \{The decoder returns an error vector of weight $t$.\}
Compute $c_3 := P^{-1}c_2$.
**return** $c_3$
**End**
The encryption is a bit trickier than in Example 10.6.1, since our messages now are vectors of length $n$ and of weight $t$.

```plaintext
> MessageSpace:=Subsets(Set([1..n]), t);
> mm:=Random(MessageSpace);
> mm:=[i: i in mm]; m:=C ! [0: i in [1..n]];
> // insert errors at given positions
> for i in mm do
>   m[i]:=1;
> end for;
> c:=m*Transpose(HPublic); // the ciphertext
```

The decryption part is also a bit tricker, because the decoding function of Magma expects a word, not a syndrome. So we have to find a solution to the parity check linear system and then pass this solution to the decoding function.

```plaintext
> c1:=c*Transpose(S^-1);
> c22:=Solution(Transpose(H),c1); // find any solution
> bool,c2:=Decode(C2,c22:Al:="Euclidean");
> m_dec:=(c22-c2)*Transpose(P^-1);
One may see that $m=m_{\text{dec}}$ holds.
```

Now we will show that in fact the Niederreiter and McEliece encryption schemes have equivalent security. In order to do so, we assume that we have generated the two schemes from the same secret code $C$ with a generator matrix $G$ and a parity check matrix $H$. Assume further that the private key of the McEliece scheme is $(S,G,P)$ and for the Niederreiter scheme is $(M,H,P)$, so that the public keys are $G'=SGP$ and $H'=MHP$ respectively. Let $z=yH'^T$ be the ciphertext obtained by encrypting $y$ with the Niederreiter scheme and $c=mG'+e$ be the ciphertext obtained from $m$ with the McEliece scheme. Equivalence now means that if one is able to recover $y$ from the constructed $z$, we are able to recover $e$ and thus $m$ from its ciphertext $c=mG'+e$.

Analogously, assume that for any $m$ and $e$ of weight $\leq t$ we can recover them from $c=mg'+e$. Now we want to recover $y$ of weight $\leq t$ from $z=yH'^T$. For $e=y$ we have

$$yH'^T = eH'^T = mg'H'^T + eH'^T = cH'^T = z,$$

with $c=mg'+e$, since $G'H'^T = SGPPT^TH^TM^T = SGH^TM^T = 0$, due to $PP^T=Id_n$ and $GH^T=0$. So if we can recover such $y$ from the above constructed $z$, we are able to recover $e$ and thus $m$ from its ciphertext $c=mg'+e$.

10.6.3 Attacks

There are two types of attacks one may think of for code-based cryptosystems:
1. **Generic decoding attacks.** One tries to recover \( m \) from \( c \) using the code \( C' \).

2. **Structural attacks.** One tries to recover \( S, G, P \) from the code \( C' \) given by \( G' \) in the McEliece scheme or \( S, H, P \) from in the Niederreiter scheme.

Consider the McEliece encryption scheme. In the attack of type (1), the attacker tries to directly decode the ciphertext \( c \) using the code \( C' \) generated by the public generator matrix \( G' \). Assuming \( C' \) is a random code, one may obtain complexity estimates for this type of attack. The best results in this direction are obtained using the family of algorithms that improve on the information set decoding (ISD), see Section 6.2.3.

Recall that the idea of the (probabilistic) ISD is to find an error-free information set \( I \) and then decode as \( c = r(I)G' \) for the received word \( r \). Here the matrix \( G' \) is a generator matrix of an equivalent code that is systematic at the positions of \( I \). In order to avoid confusion with the public generator matrix, we denote it by \( G \) in the following. The first improvement of the ISD due to Lee and Brickell is in allowing some small number \( p \) of errors to occur in the set \( I \). So we no longer require \( r(I) = c(I) \), but allow \( r(I) = c(I) + e(I) \) with \( wt(e(I)) \leq p \). We can now modify the algorithm in Algorithm 6.2 as in Algorithm 10.7. Note that since we know the number of errors occurred, the if-part has changed also.

**Algorithm 10.7 Lee-Brickell ISD**

**Input:**
- Generator matrix \( G \) of an \([n, k]\) code \( C \)
- Received word \( r \)
- Number of errors \( t \) occurred, so that \( d(r, C) = t \)
- Number of trials \( N_{\text{trials}} \)
- Parameter \( p \)

**Output:** A codeword \( c \in C \), such that \( d(r, c) = t \) or “No solution found”

**Begin**

\[ c := 0; \]
\[ N_{\text{tr}} := 0; \]
repeat
\[ N_{\text{tr}} := N_{\text{tr}} + 1; \]
Choose a subset \( I \) of \([1, \ldots, n]\) of cardinality \( k \).
if \( G(I) \) is invertible then
\[ \tilde{G} := G(I)^{-1}G \]
for \( e(I) \) of weight \( \leq p \) do
\[ \tilde{c} = (r(I) + e(I))\tilde{G} \]
if \( d(\tilde{c}, r) = t \) then
return \( \tilde{c} \)
end if
end for
end if
until \( N_{\text{tr}} < N_{\text{trials}} \)
return “No solution found”
**End**
Remark 10.6.3 In Algorithm 10.7 one may replace choosing a set $I$ by choosing every time a random permutation matrix $\Pi$. Then one may find $\text{rref}(G\Pi)$ therewith obtaining an information set. One must keep track of the applied permutations $\Pi$ in order to “go back” after finding a solution in this way.

The probability of success in one trial of the Lee-Brickell variant is

$$p_{LB} = \frac{k\binom{n-k}{t-p}}{\binom{n}{t}}$$

compared to the original one of the probabilistic ISD

$$p_{ISD} = \frac{(n-k)}{\binom{n}{t}}.$$ 

Since in the for-loop of Algorithm 10.7 we have to run $2^p$ times, $p$ should be a small constant. In fact for small $p$, like $p = 2$, one obtains complexity improvement, although not asymptotical, but quite relevant for practice.

There is a rich list of further improvements due to many researchers in the field, see Notes. The improvements basically consider different configurations of where a small number $p$ of errors is allowed to be present, where only a block of $l$ zeroes should be present, etc. Further, the choice of the next set $I$ can be optimized, for example by changing just one element in the current $I$ in a clever way. With all these techniques in mind, one obtains quite a considerable improvement of the ISD in practical attacks on the McEliece cryptosystem. In fact the original proposal of McEliece to use $[1024,524]$ binary Goppa codes correcting 50 errors is not a secure choice any more; one has to increase the parameters of the Goppa codes used.

Example 10.6.4 [CAS] Magma contains implementations of the “vanilla” probabilistic ISD, which was also considered in the original paper of McEliece, as well as Lee-Brickell’s variant and several other improved algorithms. Let us try to attack the toy example considered in Example 10.6.1. So we copy all the instructions responsible for the code construction, key generation, and encryption.

> ... // as in Example \ref{ex-CAS-McEliece}

Then we use commands

> CPublic:=LinearCode(GPublic);
> McEliecesAttack(CPublic,c,3);
> LeeBrickellsAttack(CPublic,c,3,2);

to mount our toy attack. For this specific example it takes no time to execute both attacks. In both commands first we pass the code, then the received word, and then the number of errors to be corrected. In \texttt{LeeBrickellsAttack} the last parameter is exactly the parameter $p$ from Algorithm 10.7; we set it to 2.

We can correct errors with random codes. Below is the example:

> C:=RandomLinearCode(GF(2),50,10);
> c:=Random(C); r:=c;
> r[2]:=r[2]+1; r[17]:=r[17]+1; r[26]:=r[26]+1;
> McEliecesAttack(C,r,3);
> LeeBrickellsAttack(C,r,3,2);
Apart from decoding being hard for the public code \( C' \), it should be impossible to deduce the structure of the code \( C \) from the public \( C' \). Structural attacks of (2.) aim at exploiting this structure. As we have mentioned, the choice of binary Goppa codes turned out to be pretty much the only secure choice up to now. There were quite a few attempts to propose other classes of codes for which efficient decoding algorithms are known. Alas, all of these proposals were broken, we just name a few: Generalized Reed-Solomon (GRS) codes, Reed-Muller codes, BCH codes, algebraic-geometry codes of small genus, LDPC codes, quasi-cyclic codes; see Notes. In the next section we will consider in detail how a prominent attack on GRS works. In particular, weakness of the GRS codes suggest, due to equivalent security, the weakness of the original proposal of Niederreiter, who suggested to use these codes in his scheme.

### 10.6.4 The attack of Sidelnikov and Shestakov

Let \( C \) be the code \( GRS_k(a, b) \), where \( a \) consists of \( n \) mutually distinct entries of \( \mathbb{F}_q \) and \( b \) consists of nonzero entries, cf. Definition 8.1.10. If this code is used in the McEliece PKC, then for an attacker the code \( C' \) with generator matrix \( G' = SGP \) is known, where \( S \) is an invertible \( k \times k \) matrix and \( P = \Pi D \) with \( \Pi \) an \( n \times n \) permutation matrix and \( D \) an invertible diagonal matrix. The code \( C' \) is equal to \( GRS_k(a', b') \), where \( a' = a \Pi \) and \( b' = b P \). In order to decode \( GRS_k(a', b') \) up to \( \lfloor (n - k + 1)/2 \rfloor \) errors it is enough to find \( a' \) and \( b' \). The \( S \) is not essential in masking \( G \), since \( G' \) has a unique row equivalent matrix \((I_k | A')\) that is in reduced row echelon form. Here \( A' \) is a generalized Cauchy matrix (Definition 3.2.17), but it is a priori not evident how to recover \( a' \) and \( b' \) from this.

The code is MDS hence all square submatrices of \( A' \) are invertible by Remark 3.2.16. In particular all entries of \( A' \) are nonzero. After multiplying the coordinates with nonzero constants we get a code which is generalized equivalent with the original one, and is again of the form \( GRS_k(a'', b'') \), since \( \mathbf{r} \ast GRS_k(a', b') = GRS_k(a', b' \ast \mathbf{r}) \). So without loss of generality it may be assumed that the code has a generator matrix of the form \((I_k | A')\) such that the last row and the first column of \( A' \) consists of ones.

Without loss of generality it may be assumed that \( a_{k-1} = \infty \), \( a_k = 0 \) and \( a_{k+1} = 1 \) by Proposition 8.1.25. Then according to Proposition 8.1.17 and Corollary 8.1.19 there exists a vector \( \mathbf{c} \) with entries \( c_i \) given by

\[
c_i = \begin{cases} 
b_i \prod_{t=1, t \neq i}^{k} (a'_t - a'_i) & \text{if } 1 \leq i \leq k, \\
b'_i \prod_{t=1}^{k} (a'_t - a'_i) & \text{if } k + 1 \leq i \leq n,
\end{cases}
\]

such that \( A' \) has entries \( a'_{ij} \) given by

\[
a'_{ij} = \frac{c_{j+k-1} c_i^{-1}}{a_{j+k-1} - a_i}
\]

for \( 1 \leq i \leq k - 1 \), \( 1 \leq j \leq n - k + 1 \) and

\[
a'_{ik} = c_{j+k-1} c_i^{-1}
\]

for \( 1 \leq j \leq n - k + 1 \).
Example 10.6.5 Let $G'$ be the generator matrix of a code $C'$ with entries in $\mathbb{F}_7$ given by

$$G' = \begin{pmatrix} 6 & 1 & 1 & 6 & 2 & 2 & 3 \\ 3 & 4 & 1 & 1 & 5 & 4 & 3 \\ 1 & 0 & 3 & 3 & 6 & 0 & 1 \end{pmatrix}.$$ 

Then $\text{rref}(G') = (I_3|A')$ with

$$A' = \begin{pmatrix} 1 & 3 & 3 & 6 \\ 4 & 4 & 6 & 6 \\ 3 & 1 & 6 & 3 \end{pmatrix}.$$ 

$G'$ is a public key and it is known that it is the generator matrix of a generalized Reed-Solomon code. So we want to find $a$ in $\mathbb{F}_7^7$ consisting of mutually distinct entries and $b$ in $\mathbb{F}_7^7$ with nonzero entries such that $C' = GRS_3(a,b)$. Now $C = (1,4,3,1,5,5,6) \cdot C'$ has a generator matrix of the form $(I_3|A)$ with

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 5 & 4 & 2 \\ 1 & 4 & 3 & 6 \end{pmatrix}.$$ 

We may assume without loss of generality that $a_1 = 0$ and $a_2 = 1$ by Proposition 8.1.25.

10.6.5 Exercises

10.6.1 Show the correctness of the Niederreiter scheme.

10.6.2 Using methods of Section 10.6.3 attack larger McEliece schemes. In the Goppa construction take

- $m = 8, r = 16$
- $m = 9, r = 5$
- $m = 9, r = 7$

Make observations that would answer the following questions:

- Which attack is faster: the plain ISD or Lee-Brickell’s variant?

- What is the role of the parameter $p$? What is the optimal value of $p$ in these experiments?

- Does the execution time differ from one run to the other or it stays the same?

- Is there any change in execution time, when the attacks are done for random codes with the same parameters as above?

Try to experiment with other attack methods implemented in Magma: LeonsAttack, SternsAttack, CanteautChabaudsAttack.

Hint: For constructing Goppa polynomials use the command PrimitivePolynomial.
10.6.3 Consider binary Goppa codes of length 1024 and Goppa polynomial of degree 50.
(1) Give an upper bound of the number of these codes.
(2) What is the fraction of the number of these codes with respect to all binary \([1024, 524]\) codes?
(3) What is the minimum distance of a random binary \([1024, 524]\) code according to the Gilbert-Varshamov bound?

10.6.4 Give an estimate of the complexity of decoding 50 errors of a received word with respect to a binary \([1024, 524, 101]\) code by means of covering set decoding.

10.6.5 Let \(\alpha \in \mathbb{F}_8\) be a primitive element such that \(\alpha^3 = \alpha + 1\). Let \(G'\) be the generator matrix given by

\[
G' = \begin{pmatrix}
\alpha^6 & \alpha^6 & \alpha & 1 & \alpha^4 & 1 & \alpha^4 \\
0 & \alpha^3 & \alpha^4 & \alpha^6 & \alpha^6 & \alpha^4 \\
\alpha^4 & \alpha^5 & \alpha^3 & 1 & \alpha^2 & 0 & \alpha^6
\end{pmatrix}
\]

(1) Find \(a\) in \(\mathbb{F}_8^7\) consisting of mutually distinct entries and \(b\) in \(\mathbb{F}_8^7\) with nonzero entries such that \(G'\) is a generator matrix of \(\text{GRS}_3(a, b)\).

(2) Consider the \(3 \times 7\) generator matrix \(G\) of the code \(\text{RS}_3(7, 1)\) with entry \(\alpha^{(i-1)(j-1)}\) in the \(i\)-th row and the \(j\)-th column. Give an invertible \(3 \times 3\) matrix \(S\) and a permutation matrix \(P\) such that \(G' = SGP\).

(3) What is the number of pairs \((S, P)\) of such matrices?

10.7 Notes

Some excellent references for introduction to cryptography are [87, 117, 35].

10.7.1 Section 10.1

Computational security concerns with practical attacks on cryptosystems, whereas unconditional security works with probabilistic models, where an attacker is supposed to possess unlimited computing power. A usual claim when working with unconditional security would be to give an upper bound of attacker’s success probability. This probability is independent on the computing power of an attacker and bears ”absolute value”. For instance in the case of Shamir’s secret sharing scheme (Section 10.4) no matter how much computing power does a group of \(t - 1\) participants have, it does not have better to do as to guess a value of the secret. Probability of such a success is \(1/q\). More on these issues can be found in [117].

A couple of remarks on block ciphers that were used in the past. Jefferson cylinder invented by Thomas Jefferson in 1795 and independently by Etienne Bazeries is a polyalphabetic cipher used by the U.S. army in 1923-1942 and had the name M-94. It was constructed as a rotor with 20–36 discs with letters, each of which provided a substitution at a corresponding position. For its time it had quite good cryptographic properties. Probably the best known historic cipher is German’s Enigma. It had been used for commercial purposes already in 1920s, but became famous for its use during the World War II by the Nazi German military. Enigma is also a rotor-based polyalphabetic cipher. More on historical ciphers in [87].
The Kasiski method aims at recovering period of a polyalphabetic substitution cipher. Here one encrypts repeated portions of a plaintext with the same keyword. More details in [87].

The National Bureau of Standards (NBS, later became National Institute of Standards and Technology - NIST) initiated development of DES (Data Encryption Standard) in early 1970s. IBM’s cryptography department and in particular its leaders Dr. Horst Feistel (recall Feistel cipher) and Dr. W. Tuchman contributed the most to the development. The evaluation process was also facilitated by the NSA (National Security Agency). The standard was finally approved and published in 1977 [90]. A lot of controversy accompanied DES since its appearance. Some experts claimed that the developers could have intentionally added some design trapdoors to the cipher, so that its cryptanalysis would have been possible by them, but not by the others. The key size 56-bits also raised concern, which eventually led to the need to adopt a new standard. Historical remarks on DES and its development can be found in [112].

Differential and linear cryptanalysis turned out to be the most successful theoretical attacks on DES. For initial papers on these attacks, see [15, 82]. The reader may also visit http://www.ciphersbyritter.com/RES/DIFFANA.HTM for more references and history of the differential cryptanalysis. We also mention that differential cryptanalysis may have been known to the developers long before Adi Shamir published his paper in 1990.

Since DES encryptions do not form a group, the use of a triple application of DES was proposed that was called triple DES [66]. Although no effective cryptanalysis against triple DES was proposed it is rarely used due to its slow compared to AES implementation.

In the middle of 1990s it became apparent to the cryptography community that the DES did not provide sufficient security level anymore. So NIST announced a competition for a cipher that would replace DES and became the AES (Advanced Encryption Standard). The main criteria that were imposed for the future AES were resistance to linear and differential cryptanalysis, faster and more effective (compared to DES) implementation, ability to work with 128-bit plaintext blocks and 128, 192, 256-bit keys; the number of rounds was not specified. After five years of the selection process, the cipher Rijndael proposed by the Belgian researchers Joan Daemen and Vincent Rijmen won. The cipher was officially adopted for the AES in 2001 [91]. Because resistance to linear and differential cryptanalysis was one of the milestones in the design of AES, J. Daemen and V. Rijmen carefully studied this question and showed how such resistance can be achieved within Rijndael. In the design they used what they called wide trail strategy - a method devised specifically to counter the linear and differential cryptanalysis. Description of the AES together with the discussion of underlying design decisions and theoretical discussion can be found in [45]; for the wide trail strategy see [46].

As to attacks on AES, up to now there is no attack out there that could break AES at least theoretically, i.e. faster than the exhaustive search, in a scenario where the unknown key stays the same. Several attacks, though, work on non-full AES that performs less than 10 rounds. For example Boomerang type of attacks are able to break 5–6 rounds of AES-128 much faster, than the exhaustive search. For 6 rounds the Boomerang attack has data complexity of $2^{71}$ 128-bit blocks, memory complexity of $2^{33}$ blocks and time complexity of $2^{71}$ AES encryptions. This attack is mounted under a mixture of chosen plaintext and adaptive chosen ciphertext scenarios. Some other attacks also can attack 5–7 rounds. Among them are the Square attack, proposed by Daemen and Rijmen themselves, collision attack, partial sums, impossible
differentials. For an overview of attacks on Rijndael see [40]. There are recent works on related key attacks on AES, see [1]. It is possible to attack 12 rounds of AES-192 and 14 rounds of AES-256 in the related key scenario. Still, it is quite questionable by the community on whether one may consider these as a real threat.

We would like to mention several other recent block ciphers. The cipher Serpent is an instance of an SP-network and was the second in the AES competition that was won by Rijndael. As was prescribed by the selection committee it also operates on 128-bit blocks and key of sizes 128, 192, and 256 bits. Serpent has a strong security margin, prescribing 32 rounds. Some information online: http://www.cl.cam.ac.uk/~rja14/serpent.html. Next, the cipher Blowfish proposed in 1993 is an instance of a Feistel cipher, has 16 rounds and operates on 64-bit blocks and default key size of 128 bits. Blowfish is up to now resistant to cryptanalysis and its implementation is rather fast, although has some limitations that preclude its use in some environments. Information online: http://www.schneier.com/blowfish.html. A successor of Blowfish proposed by the same person - Bruce Schneier - Twofish was one of the five finalists in the AES competition. It has the same block and key sizes as all the AES contestants and has 16 rounds. Twofish is also a Feistel cipher. This cipher is also believed to resist cryptanalysis. Information online http://www.schneier.com/twofish.html. It is noticeable that all these ciphers are in public domain and are free for use in any software/hardware implementations. A light-weight block cipher PRESENT that operates on plaintext block of only 64 bits and key length of 80 and 128 bits; PRESENT has 31 rounds [2]. There exist proposals with even smaller block lengths, see http://www.ecrypt.eu.org/lightweight/index.php/Block_ciphers.

10.7.2 Section 10.2

The concept of the asymmetric cryptography was introduced by Diffie and Hellman in 1976 [48]. For an introduction to the subject and survey of results see [87, 89]. The notion of a one-way as well as trapdoor one-way function was also introduced by Diffie and Hellman in the same paper [48].

Rabin’s scheme from Example 10.2.2 was introduced by Rabin in [97] in 1979 and ElGamal scheme was presented in [55]. The notion of a digital signature was also presented in the pioneering work [48], see also [88].

The RSA scheme was introduced in 1977 by Rivest, Shamir, and Adleman [100]. In the same paper they showed that computing the decryption exponent and factoring are equivalent. There is no known polynomial time algorithm for factoring integers. Still there are quite a few algorithms out there that have sub-exponential complexity. For a survey of existing methods, see [96]. Asymptotically the best known sub-exponential algorithm is general number field sieve, and it has an expected running time of $O(\exp((\frac{64}{27}b)^{1/3} \log b^{2/3}))$, where $b$ is a bit length of a number $n$ that is to be factored [36].

Development of factoring algorithms changed requirements on the RSA key size through the time. In their original paper [100] the authors suggested the use of 200 decimal digit modulus (664 bits). The sizes of 336 and 512 bits were also used. In 2010 the result of factoring RSA-768 was announced. Used at present modulus of 1024 bits raises many questions on whether it may be considered secure. Therefore, for long-term security the key size of 2048 or even 3072 bits are to be used. Quite remarkable is the work of Shor [7, 8] who proposed an algorithm that can solve integer factorization problem in polynomial time on a quantum computer.

The use of $\mathbb{Z}_n$ in ElGamal scheme was proposed in [83]. For the use of Jacobian of
a hyperelliptic curve see [41]. There are several methods for solving DLP, we name just a few: Baby-step giant-step, Pollard’s rho and Pollard’s lambda (or kangaroo), Pohlig-Hellman, index calculus. The fastest algorithms for solving DLP for \( \mathbb{Z}_p^* \) and \( F_{2^m} \) are variations of the index calculus algorithm. All of the above algorithms applied to the multiplicative group of a finite field do not have polynomial complexity. For an introduction to these methods the reader may consult [41]. Index calculus is an algorithm with sub-exponential complexity. These developments on DLP solving algorithms affected the key size of El Gamal scheme. In practice the key size grew from 512 to 768 and finally to 1024. At the moment using 1024-bit key for El Gamal is considered to be a standard. The mentioned work of Shor [108] also solve the DLP problem in polynomial time. Therefore existence of a large enough quantum computer jeopardizes the ElGamal scheme.

Despite doubts of some researcher of a possibility to construct a large enough quantum computer, the area of post-quantum cryptography has evolved that incorporates cryptosystems that are potentially resistant to quantum computer attacks. See [12] for an overview of the area, which includes lattice based, hash based, coding based, and multivariate based cryptography. Some references to multivariate asymmetric systems, digital schemes, and their cryptanalysis can be found in [49]. The knapsack cryptosystem by Merkle and Hellman has an interesting history. It was one of the first asymmetric cryptosystems. Its successful cryptanalysis showed that only reliance on hardness of the underlying problem may be misleading. For an interesting historical development, see the survey chapter of Diffie [47].

### 10.7.3 Section 10.3

Authentication codes were initially proposed by MacWilliams, Gilbert, and Sloan [58]. Introductory material on authentication codes is well exposed in [117]. Message authentication codes (MACs) are widely used in practice for authentication purposes. MACs are keyed hash functions. In the case of MACs one demands from a hash function to provide compression (a message of arbitrary size is mapped to a fixed size vector), ease of computation (it should be easy to compute an image knowing the key), and computation-resistance (practical impossibility to compute an image without knowing the key, even having some pairs element-image in disposal). More on MACs in [87].

Results and discussion of the relation between authentication codes and orthogonal arrays is in [117, 116, 115]. Proposition 10.3.7 is due to Bierbrauer, Johansson, Kabatianskii, and Smeets [13]. By adding linear structure to the source state set, key space, tag space, and authentication mappings one obtains linear authentication codes that can be used in the study of distributed authentication systems [103].

### 10.7.4 Section 10.4

The notion of a secret sharing scheme was first introduced in 1979 by Shamir [106] and independently by Blakely [22]. We mention here some notions that were not mentioned in the main text. A secret sharing scheme is called perfect if knowledge of shares from an unauthorized group (e.g. a group of \( < t \) participants in Shamir’s scheme) does not reduce the uncertainty about the secret itself. In terms of entropy function it can be stated like this: \( H(S|A) = 0 \), where \( S \) is the secret and \( A \) is an unauthorized set, moreover we have \( H(S|B) = H(S) \) for \( B \) being an authorized set. In perfect secret sharing schemes it holds that the size of each share is at least the size of the secret.
10.7. NOTES

If equality holds such a system is called ideal. The notion of a secret sharing scheme can be generalized via the notion of an access structure. Using access structures one prescribes which subsets of participants can reconstruct the secret (authorized subset) and which cannot (unauthorized subset). The notion of a distribution of shares can also be formalized. More details on these notions and treatment using probability theory can be found e.g. in [117].

McEliece and Sarwate [85] were the first to point out the connection between Shamir’s scheme and the Reed-Solomon codes construction. Some other works on relations between coding theory and secret sharing schemes include [70, 81, 94, 138]. More recent works concern applications of AG-codes to this subject. We mention the chapter of Duursma [52] and the work of Chen and Cramer [39]. In the latter two references the reader can also find the notion of secure multi-party computation, see [137]. The idea here is that several participants wish to compute the value of some publicly known function evaluated at their values (like shares in the above). The thing is that each of the participants should not be able to know the values of other participants by the known computed value of the public function and his/her own value. We also mention that as was the case with authentication codes, information theoretic perfectness can be traded off to obtain a system where shares are smaller than the secret [23].

10.7.5 Section 10.5

Introductory material on LFSRs with discussion of practical issues can be found in [87]. The notion of linear complexity is treated in [102], see also materials online at http://www.ciphersbyritter.com/RES/LINCOMPL.HTM. A thorough exposure of linear recurring sequences is in [74].

Some examples of non-cryptographic use of LFSRs, namely randomization in digital broadcasting: Advanced Television Systems Committee (ATSC) standard for digital television format (http://www.atsc.org/guide_default.html), Digital Audio Broadcasting (DAB) digital radio technology for broadcasting radio stations, Digital Video Broadcasting - Terrestrial (DVB-T) is the European standard for the broadcast transmission of digital terrestrial television (http://www.dvb.org/technology/standards/).

An example of nonlinear combination generator: E0 is a stream cipher used in the Bluetooth protocol, see e.g. [64]; of nonlinear filter generator Knapsack generator [87]; of clock-controlled generators: A5/1 and A5/2 are stream ciphers used to provide voice privacy in the Global System for Mobile communications (GSM) cellular telephone protocol http://web.archive.org/web/20040712061808/www.ausmobile.com/downloads/technical/Security+in+the+GSM+system+01052004.pdf.

10.7.6 Section 10.6

***
Chapter 11

The theory of Gröbner bases and its applications

Stanislav Bulygin

In this chapter we deal with methods in coding theory and cryptology based on polynomial system solving. As the main tool for this we use the theory of Gröbner bases that is a well-established instrument in computational algebra. In Section 11.1 we give a brief overview of the topic of polynomial system solving. We start with relatively easy methods of linearization and extended linearization. Then we give basics of more involved theory of Gröbner bases.

The problem we are dealing with in this chapter is the problem of polynomial system solving. We formulate this problem as follows: let $\mathbb{F}$ be a field and let $\mathbb{P} = \mathbb{F}[X_1, \ldots, X_n]$ be a polynomial ring over $\mathbb{F}$ in $n$ variables $X = (X_1, \ldots, X_n)$. Let $f_1, \ldots, f_m \in \mathbb{P}$. We are interested in finding a set of solutions $S \subseteq \mathbb{F}^n$ to the polynomial system defined as

$$f_1(X_1, \ldots, X_n) = 0,$$

$$\ldots$$

$$f_m(X_1, \ldots, X_n) = 0.$$  \hspace{1cm} (11.1)

In other words, $S$ is composed of those elements from $\mathbb{F}^n$ that satisfy all the equations above.

In terms of algebraic geometry this problem may be formulated as follows. Given an ideal $I \subseteq \mathbb{P}$, find the variety $V_{\mathbb{F}}(I)$ which it defines:

$$V_{\mathbb{F}}(I) = \{ x = (x_1, \ldots, x_n) \in \mathbb{F}^n | f(x) = 0 \ \forall f \in I \}.$$  \hspace{1cm} (11.1)

Since now we are interested in applications to coding and cryptology, we will be working over finite fields and often we would like solutions of corresponding systems to lie in these finite fields, rather than in an algebraic closure. Recall that for every element $\alpha \in \mathbb{F}_q$ the following holds: $\alpha^q = \alpha$. This means that if we add an equation $X^q - X = 0$ to a polynomial system (11.1), we are guaranteed that solutions for the $X$-variable lie in $\mathbb{F}_q$.

After introducing tools for the polynomial system solving in Section 11.1, we
give two concrete applications in Sections 11.2 and 11.3. In Section 11.2 we con-
sider applications of Gröbner bases techniques to decoding linear codes, whereas
Section 11.3 deals with methods of algebraic cryptanalysis of block ciphers. Due
to space limitations many interesting topics related to these areas are not con-
sidered. We provide their short overview with references in the Notes section.

11.1 Polynomial system solving

11.1.1 Linearization techniques

We know how to solve systems of linear equations efficiently. Gaussian elimi-
nation is a standard tool for this job. If we are given a system of non-linear
equations, a natural solution would be to try to reduce this problem to a lin-
ear one, which we know how to solve. This simple idea leads to a technique
that is called linearization. This technique works as follows: we replace every
monomial occurring in a non-linear (polynomial) equation by a new variable.
At the end we obtain a linear system with the same number of equations, but
many new variables. The hope is that by solving this linear system we are able
to get a solution to our initial non-linear problem. It is better to illustrate this
approach on a concrete example.

Example 11.1.1 Consider a quadratic system in two unknowns \( x \) and \( y \) over
the field \( \mathbb{F}_3 \):

\[
\begin{align*}
  x^2 - y^2 - x + y &= 0 \\
  -x^2 + x - y + 1 &= 0 \\
  y^2 + y + x &= 0 \\
  x^2 + x + y &= 0
\end{align*}
\]

Introduce new variables \( a := x^2 \) and \( b := y^2 \). Therewith we have a linear system:

\[
\begin{align*}
  a - b - x + y &= 0 \\
  -a + x - y + 1 &= 0 \\
  b + y + x &= 0 \\
  a + x + y &= 0
\end{align*}
\]

This system has a unique solution, which may be found with the Gaussian
elimination: \( a = b = x = y = 1 \). Moreover, the solution on \( a \) and \( b \) is consistent
with the conditions \( a = x^2, b = y^2 \). So although the system is quadratic, we are
still able to solve it purely with methods of linear algebra.

It must be noted that the linearization technique works very seldom. Usually
the number of variables (i.e. monomials) in the system is much larger than the
number of equations. Therefore one has to solve an underdetermined linear
system, which has many solutions, among which it is hard to find a “real” one
that stems from the original non-linear system.

Example 11.1.2 Consider a system in three variables \( x, y, z \) over the field \( \mathbb{F}_{16} \):

\[
\begin{align*}
  xy + yz + xz &= 0 \\
  xyz + x + 1 &= 0 \\
  xy + y + z &= 0
\end{align*}
\]
It may be shown that over $\mathbb{F}_{16}$ this system has a unique solution $(1, 0, 0)$. If we replace monomials in this system with new variables we will end up with a linear system of 3 equations and 7 variables. This system is full rank. In particular the variables $x, y, z$ are now free variables which yield values for other variables. So such linear system has $16^3$ solutions, of which only one will provide a legitimate solution for the initial system. Other solutions do not have any meaning. E.g. we may show that assignment $x = 1, y = 1, z = 1$ implies that “variable” $xy$ should be 0, and of course this cannot be true, since both $x$ and $y$ are 1. So using linearization technique boils down to sieving the set $\mathbb{F}_{16}^3$ for a right solution, but this is nothing more than an exhaustive search for the initial system.

So the problem with the linearization technique is that we do not have enough linearly independent equations for solving. Here is where the idea of eXtended Linearization (XL) comes in hand. The idea of XL is to multiply initial equations by all monomials up to given degree (hopefully not too large) to generate new equations. Of course new variables will appear, since new monomials will appear. Still if the system is “nice” enough we may generate necessary number of linearly independent equations to obtain a solution. Namely, we hope that after “extending” our system with new equations and doing Gaussian elimination, we will be able to find a univariate equation. Then we can solve it and plug in obtained values and then proceed with a simplified system.

**Example 11.1.3** Consider a small system in two unknowns $x, y$ over the field $\mathbb{F}_4$:

\[
\begin{align*}
  x^2 + y + 1 &= 0 \\
  xy + y &= 0
\end{align*}
\]

It is clear that the linearization technique does not work so well in this case, since the number of variables (3) is larger than the number of equations (2). Now multiply the two equations first with $x$ and then with $y$. Therewith we obtain four new equations, which have the same solution as the initial ones, so we may add them to the system. The new equations are:

\[
\begin{align*}
  x^3 + xy + x &= 0 \\
  x^2 y + xy &= 0 \\
  x^2 y + y^2 + y &= 0 \\
  xy^2 + y^2 &= 0
\end{align*}
\]

Here again the number of equations is lower than the number of variables. Still, by ordering monomials in the way that $y^2$ and $y$ go leftmost in the matrix representation of a system and doing Gaussian elimination, we encounter a univariate equation $y^2 = 0$ (check this!). So we have a solution for $y$, namely $y = 0$. After substituting $y = 0$ in the first equation we have $x^2 + 1 = 0$, which is again a univariate equation. Over $\mathbb{F}_4$ is has a unique solution $x = 1$. So by using linear algebra and univariate equation solving, we were able to obtain the solution $(1, 0)$ for the system.

Algorithm 11.1 explains more formally how the XL works. In our example it was enough to set $D = 3$. Usually one has to go much higher to get the result.

In the next section we consider a technique of Gröbner basis, which is a more powerful tool. In some sense, it is a refined and improved version of the XL.
Algorithm 11.1 XL\((F, D)\)

Input:
- A system of polynomial equations \(F = \{f_1 = 0, \ldots, f_m = 0\}\) of total degree \(d\) over the field \(\mathbb{F}\) in variables \(x_1, \ldots, x_n\);
- Parameter \(D\);

Output: a solution to the system \(F\) or the message “no solution found”

Begin
\(D_{\text{current}} := d;\)
\(\text{Sol} := \emptyset;\)
repeat
  \textbf{Extend}: Multiply each equation \(f_i \in F\) with all monomials of degree \(\leq D_{\text{current}} - d\); Denote the system so obtained by \(\text{Sys}\);
  \textbf{Linearize}: assign each monomial appearing in \(\text{Sys}\) a new variable, order the new variables such that \(x_i^a\) go left-most in the matrix representation of a system in blocks \(\{x_i, x_i^2, \ldots\}\); \(\text{Sys} := \text{Gauss}(\text{Sys})\);
  if exists a univariate equation \(f(x_i) = 0\) in \(\text{Sys}\) then
    solve \(f(x_i) = 0\) over \(\mathbb{F}\) and obtain \(a_i : f(a_i) = 0\);
    \(\text{Sol} := \text{Sol} \cup \{(i, a_i)\}\);
    if \(|\text{Sol}| = n\) then
      return \(\text{Sol}\);
    end if
    \(\text{Sys} := \text{Sys}\) with \(a_i\) substituted for \(x_i\);
  else
    \(D_{\text{current}} := D_{\text{current}} + 1;\)
  end if
until \(D_{\text{current}} = D + 1\)
return “no solution found”
End
11.1.2 Gröbner bases

In the previous section we saw how one can solve systems of non-linear equations using linearization techniques. Speaking the language of algebraic geometry, we want to find elements of the variety $V(f_1, \ldots, f_m)$, where $V(f_1, \ldots, f_m) := \{a \in \mathbb{F}^n : \forall 1 \leq i \leq m, f_i(a) = 0\}$ and $\mathbb{F}$ is a field. The target object of this section, Gröbner basis technique, gives an opportunity to find this variety and also solve many other important problems like for example ideal membership, i.e. deciding whether a given polynomial may be obtained as a polynomial combination of the given set of polynomials. As we will see, the algorithm for finding Gröbner bases generalizes Gaussian elimination for linear systems on one side and Euclidean algorithm of finding the GCD of two univariate polynomials on the other side. We will see how this algorithm, the Buchberger’s algorithm, works and how Gröbner bases can be applied for finding a variety (system solving) and some other problems. First of all, we need some definitions.

**Definition 11.1.4** Let $R := \mathbb{F}[x_1, \ldots, x_n]$ be a polynomial ring in $n$ variables over the field $\mathbb{F}$. An ideal in $R$ is a subset of $R$ with the following properties:

- $0 \in I$;
- for every $f, g \in I : f + g \in I$;
- for every $h \in R$ and every $f \in I : h \cdot f \in R$.

So the ideal $I$ is a subset of $R$ closed under addition and closed under multiplication with elements from $R$. Let $f_1, \ldots, f_m \in R$. It is easy to see that the object $\langle f_1, \ldots, f_m \rangle := \{a_1f_1 + \cdots + a_mf_m | a_i \in R \ \forall i\}$ is an ideal. We say that $\langle f_1, \ldots, f_m \rangle$ is an ideal generated by the polynomials $f_1, \ldots, f_m$. From commutative algebra it is know that every ideal $I$ has a finite system of generators, i.e. $I = \langle f_1, \ldots, f_m \rangle$ for some $f_1, \ldots, f_m \in I$. A Gröbner basis, that we define later, is a system of generators with special properties.

A monomial in $R$ is a product of the form $x_1^{a_1} \cdots x_n^{a_n}$ with $a_1, \ldots, a_n$ being non-negative integers. Denote $X = \{x_1, \ldots, x_n\}$ and by $Mon(X)$ the set of all monomials in $R$.

**Definition 11.1.5** A monomial ordering on $R$ is any relation $>$ on $Mon(X)$ such that

- $>$ is a total ordering on $Mon(X)$, i.e. any two elements of $Mon(X)$ are comparable;
- $>$ is multiplicative, i.e. $X^\alpha > X^\beta$ implies $X^\alpha \cdot X^\gamma > X^\beta \cdot X^\gamma$ for all vectors $\gamma$ with non-negative integer entries, here $X^\alpha = x_1^{a_1} \cdots x_n^{a_n}$;
- $>$ is a well-ordering, i.e. every non-empty subset of $Mon(X)$ has a minimal element.

**Example 11.1.6** Here are some orderings that are frequently used in practice.

1. Lexicographic ordering induced by $x_1 > \cdots > x_n : X^\alpha >_{\text{lex}} X^\beta$ if and only if there exists an $s$ such that $\alpha_1 = \beta_1, \ldots, \alpha_{s-1} = \beta_{s-1}, \alpha_s > \beta_s$. 

---

11.1. POLYNOMIAL SYSTEM SOLVING
2. **Degree reverse lexicographic ordering** induced by $x_1 > \cdots > x_n : X^\alpha >_{dp} X^\beta$ if and only if $|\alpha| : = \alpha_1 + \cdots + \alpha_n > |\beta| : = \beta_1 + \cdots + \beta_n$ or if $|\alpha| = |\beta|$ and there exists an $s$ such that $\alpha_n = \beta_n, \ldots, \alpha_{n-s+1} = \beta_{n-s+1}, \alpha_{n-s} < \beta_{n-s}$.

3. **Block ordering or product ordering**. Let $X$ and $Y$ be two ordered sets of variables, $>_1$ a monomial ordering on $\mathbb{F}[X]$ and $>_2$ a monomial ordering on $\mathbb{F}[Y]$. The block ordering on $\mathbb{F}[X,Y]$ is the following: $X^\alpha_1 Y^\beta_1 > X^\alpha_2 Y^\beta_2$ if and only if $X^\alpha_1 >_1 X^\alpha_2$ or if $X^\alpha_1 =_1 X^\alpha_2$ and $Y^\beta_1 >_2 Y^\beta_2$.

**Definition 11.1.7** Let $>$ be a monomial ordering on $R$. Let $f : = \sum c_\alpha X^\alpha$ be a non-zero polynomial from $R$. Let $\alpha_0$ be such that $c_{\alpha_0} \neq 0$ and $X^\alpha_0 > X^\alpha$ for all $\alpha \neq \alpha_0$ with $c_\alpha \neq 0$. Then $lc(f) : = c_{\alpha_0}$ is called the **leading coefficient** of $f$, $lm(f) : = X^\alpha_0$ is called the **leading monomial** of $f$, $lt(f) : = c_{\alpha_0} X^\alpha_0$ is called the **leading term** of $f$, moreover $tail(f) : = f - lt(f)$.

Having these notions we are ready to define the notion of a Gröbner basis.

**Definition 11.1.8** Let $I$ be an ideal in $R$. The **leading ideal** of $I$ with respect to $>$ is defined as $L_>(I) : = \langle lt(f) | f \in I, f \neq 0 \rangle$. The $L_>(I)$ is abbreviated by $L(I)$ if it is clear which ordering is meant. A finite subset $G = \{g_1, \ldots, g_m\}$ of $I$ is called a **Gröbner basis** for $I$ with respect to $>$ if $L_>(I) = \langle lt(g_1), \ldots, lt(g_m) \rangle$.

We say that the set $F = \{f_1, \ldots, f_m\}$ is a Gröbner basis if $F$ is a Gröbner basis of the ideal $\langle F \rangle$.

**Remark 11.1.9** Note that a Gröbner basis of an ideal is not unique. The so-called **reduced** Gröbner basis of an ideal is unique. By this one means a Gröbner basis $G$ in which all elements have leading coefficient equal to 1 and no leading term of an element $g \in G$ divides any of the terms of $g'$, where $g \neq g' \in G$.

Historically the first algorithm for computing Gröbner bases was proposed by Bruno Buchberger in 1965. In fact the very notion of the Gröbner basis was introduced by Buchberger in his Ph.D. thesis and was named after his Ph.D. advisor Wolfgang Gröbner. In order to be able to formulate the algorithm we need two more definitions.

**Definition 11.1.10** Let $f, g \in R \setminus \{0\}$ be two non-zero polynomials, and let $lm(f)$ and $lm(g)$ be leading monomials of $f$ and $g$ resp. w.r.t some monomial ordering. Denote $m : = lcm(lm(f), lm(g))$. Then the **s-polynomial** of these two polynomials is defined as

\[
spoly(f,g) = m/lm(f) \cdot f - lc(f) \cdot lc(g) \cdot m/lm(g) \cdot g.
\]

**Remark 11.1.11**

1. If $lm(f) = x_1^{a_1} \cdots x_n^{a_n}$ and $lm(g) = x_1^{b_1} \cdots x_n^{b_n}$, then $m = x_1^{c_1} \cdots x_n^{c_n}$, where $c_i = \max(a_i, b_i)$ for all $i$. Therewith $m/lm(f)$ and $m/lm(g)$ are monomials.

2. Note that if we write $f = lc(f) \cdot lm(f) + f'$ and $g = lc(g) \cdot lm(g) + g'$, where $lm(f') < lm(f)$ and $lm(g) < lm(g')$, then $spoly(f,g) = m/lm(f) \cdot (lc(f) \cdot lm(f) + f') - lc(f)/lc(g) \cdot m/lm(g) \cdot (lc(g) \cdot lm(g) + g') = m \cdot lc(f) + m/lm(f) \cdot f' - m \cdot lc(f) - lc(f)/lc(g) \cdot m/lm(g) \cdot g' = m/lm(f) \cdot f' - lc(f)/lc(g) \cdot m/lm(g) \cdot g'$. Therewith we “canceled out” the leading terms of both $f$ and $g$. 
Example 11.1.12 In order to understand this notion better, let us see what are the s-polynomials in the case of linear and univariate polynomials

linear: Let $R = \mathbb{Q}[x, y, z]$ and a monomial ordering being lexicographic with $x > y > z$. Let $f = 3x + 2y - 10z, g = x + 5y - 5z$, then $\text{lm}(f) = \text{lm}(g) = x, m = x$. $\text{spoly}(f, g) = f - 3/1 \cdot g = 3x + 2y - 10z - 3x - 15y + 15z = -13y + 5z$ and this is exactly what one would do to cancel out the variable $x$ during the Gaussian elimination.

univariate: Let $R = \mathbb{Q}[x]$. Let $f = 2x^5 - x^3, g = x^2 - 10x + 1$, then $m = x^5$ and $\text{spoly}(f, g) = f - 2/1 \cdot x^3 \cdot g = 2x^5 - x^3 - 2x^5 + 20x^4 - 2x^3 = 20x^4 - 3x^3$ and this is the first step in polynomial division algorithm, which is used in the Euclidean algorithm for finding $\gcd(f, g)$.

To define the next notion we need for the Buchberger’s algorithm, we use the following result.

Theorem 11.1.13 Let $f_1, \ldots, f_m \in R \setminus \{0\}$ be non-zero polynomials in the ring $R$ endowed with a monomial ordering $<$ and let $f \in R$ be some polynomial. Then there exist polynomials $a_1, \ldots, a_m, h \in R$ with the following properties:

1. $f = a_1 \cdot f_1 + \cdots + a_m \cdot f_m + h$;
2. $\text{lm}(f) \geq \text{lm}(a_i \cdot f_i)$ for $f \neq 0$ and every $i$ such that $a_i \cdot f_i \neq 0$;
3. if $h \neq 0$, then $\text{lm}(h)$ is not divisible by any of $\text{lm}(a_i \cdot f_i)$.

Moreover, if $G = \{f_1, \ldots, f_m\}$ is a Gröbner basis, then the polynomial $h$ is unique.

Definition 11.1.14 Let $F = \{f_1, \ldots, f_m\} \subset R$ and $f \in R$. We define the normal form of $f$ w.r.t $F$ to be any $h$ from Theorem 11.1.13. Notation is $\text{NF}(f | F) := h$.

Remark 11.1.15 1. If $R = F[x]$ and $f_1 := g \in R$, then $\text{NF}(f | \langle g \rangle)$ is exactly the remainder of division of the univariate polynomial $f$ by the polynomial $g$. So the notion of a normal form generalizes the notion of a remainder for the case of a multivariate polynomial ring.

2. The notion of a normal form is uniquely defined only if $f_1, \ldots, f_m$ is a Gröbner basis.

3. Normal form has a very important property: $f \in I \iff \text{NF}(f | G) = 0$, where $G$ is a Gröbner basis of $I$. So by computing a Gröbner basis of the given ideal $I$ and then computing the normal form of the given polynomial $f$ we may decide whether $f$ belongs to $I$ or not.

The algorithm for computing a normal form proceeds as in Algorithm 11.2. In Algorithm 11.2 the function $\text{Exists_LT_Divisor}(F, h)$ returns an index $i$ such that $\text{lt}(f_i)$ divides $\text{lt}(h)$ if such index exists and 0 otherwise. Note that the algorithm may also be adapted so that it returns the polynomial combination of $f_i$’s such that together with $h$ it satisfies conditions (1)–(3) of Theorem 11.1.13.
Algorithm 11.2 \text{NF}(f|F)

\begin{itemize}
  \item Polynomial ring \( R \) with monomial ordering \(< \)
  \item Set of polynomials \( F = \{ f_1, \ldots, f_m \} \subset R \)
  \item Polynomial \( f \in R \)
\end{itemize}

Output: a polynomial \( h \) which satisfies (1)–(3) of Theorem 11.1.13 for the set \( F \) and the polynomial \( f \) with some \( a_i \)'s from \( R \)

\begin{algorithm}
  \begin{algorithmic}
    \State \( h := f \);
    \State \( i := \text{Exists_LT_Divisor}(F, h) \);
    \While {\( h \neq 0 \) and \( i \) do}
      \State \( h := h - \text{lt}(h)/\text{lt}(f_i) \cdot f_i \);
      \State \( i := \text{Exists_LT_Divisor}(F, h) \);
    \EndWhile
    \State \text{return} \( h \)
  \end{algorithmic}
\end{algorithm}

Example 11.1.16 Let \( R = \mathbb{Q}[x, y] \) and the monomial ordering being lexicographic ordering with \( x > y \). Let \( f = x^2 - y^3 \) and \( F = \{ f_1, f_2 \} = \{ x^2 + x + y, x^3 + xy + y^3 \} \). At the beginning of Algorithm 11.2 \( h = f \). Now \( \text{Exists_LT_Divisor}(F, h) = 1 \) so we enter the while-loop. In the while-loop following assignment is made \( h := h - \text{lt}(h)/\text{lt}(f_1) \cdot f_1 = x^2 - y^3 - x^2/(x^2 + x + y) = -x - y^3 - y \). We compute again \( \text{Exists_LT_Divisor}(F, h) = 0 \). So we do not enter in the second loop and \( h = -x - y^3 - y \) is a normal form of \( f \) we looked for.

Now we are ready to formulate the Buchberger’s algorithm for finding a Gröbner basis of an ideal: Algorithm 11.3.

The main idea of the algorithm is: if after “canceling” leading terms of the current pair (also called a critical pair) we cannot “divide” the result by the current set, then add the result to this set and add all new pairs to the set of critical pairs. The next example shows the algorithm in action.

Example 11.1.17 We take as a basis Example 11.1.16. So \( R = \mathbb{Q}[x, y] \) with the monomial ordering being lexicographic ordering with \( x > y \), and we have \( f_1 = x^3 + x + y, f_2 = x^3 + xy + y^3 \). Initialization phase yields \( G := \{ f_1, f_2 \} \) and \( \text{Pairs} := \{(f_1, f_2)\} \). Now we enter the while-loop. We have to compute \( h = \text{NF}(\text{spoly}(f_1, f_2)|G) \). First, \( \text{spoly}(f_1, f_2) = x \cdot f_1 - f_2 = x^3 + x^2 + xy - x^3 - xy - y^3 = x^2 - y^3 \). As we know from Example 11.1.16, \( \text{NF}(x^2 - y^3|G) = -x - y^3 - y \) and is non-zero. Therefore, we add \( f_3 := h \) to \( G \) and add pairs \( (f_3, f_1) \) and \( (f_3, f_2) \) to \( \text{Pairs} \). Recall that pair \( (f_1, f_2) \) is no longer in \( \text{Pairs} \), so now we have two elements there.

In the second run of the loop we take the pair \( (f_3, f_1) \) and remove it from \( \text{Pairs} \). Now \( \text{spoly}(f_3, f_1) = -xy^3 - xy + x + y \) and \( \text{NF}(-xy^3 - xy + x + y|G) = y^6 + 2y^4 - y^3 + y^2 =: f_4 \). We update the sets \( G \) and \( \text{Pairs} \). Now \( \text{Pairs} = \{(f_3, f_2), (f_4, f_1), (f_4, f_3), (f_4, f_2)\} \) and \( G = \{ f_1, f_2, f_3, f_4 \} \). Next take the pair \( (f_3, f_2) \). For this pair \( \text{spoly}(f_3, f_2) = -x^2 y^3 - x^2 y + xy + y^3 \) and \( \text{NF}(\text{spoly}(f_3, f_2)|G) = 0 \). It may be shown that likewise all the other pairs from the set \( \text{Pairs} \) reduce to 0 w.r.t \( G \). Therefore, the algorithm outputs \( G = \{ f_1, f_2, f_3, f_4 \} \) as a Gröbner basis of \( \langle f_1, f_2 \rangle \) w.r.t lexicographic ordering.
11.1. POLYNOMIAL SYSTEM SOLVING

Algorithm 11.3 Buchberger($F$)

Input:
- Polynomial ring $R$ with monomial ordering $<$
- Normal form procedure NF
- Set of polynomials $F = \{f_1, \ldots, f_m\} \subset R$

Output: Set of polynomials $G \subset R$ such that $G$ is a Gröbner basis of the ideal generated by the set $F$ w.r.t monomial ordering $<$

Begin
\[ G := \{f_1, \ldots, f_m\}; \]
while $Pairs \neq \text{empty}$ do

Select a pair $(f, g) \in Pairs$;

Remove the pair $(f, g)$ from $Pairs$;

\[ h := \text{NF}(\text{spoly}(f, g)|G); \]
if $h \neq 0$ then

for all $p \in G$ do

Add pair $(h, p)$ to $Pairs$;
end for

Add $h$ to $G$;
end if
end while
return $G$
End

Example 11.1.18 [CAS] Now we will show how to compute the above examples in Singular and Magma. In Singular one has to execute the following code:
> ring r=0,(x,y),lp;
> poly f1=x^2+x+y;
> poly f2=x^3+xy+y^3;
> ideal I=f1,f2;
> ideal GBI=std(I);
> GBI;
\[ \text{GBI}[1]=y^6+2y^4-y^3+y^2 \]
\[ \text{GBI}[2]=x+y^3+y \]

One may request computation of the reduced Gröbner basis by switching on the option $\text{option(redSB)}$. In the above example $\text{GBI}$ is already reduced. Now if we compute the normal form of $f_1-f_2$ w.r.t $\text{GBI}$ it should be zero.
> NF(f1-f2,GBI);
0

It is also possible to track computations for small examples using $\text{LIB"teachstd.lib";}$. One should add this line at the beginning of the above piece of code together with the line $\text{printlevel=1;}$, which makes program comments visible. Then one should use $\text{standard(I)}$ instead of $\text{std(I)}$ to see the run in detail. Similarly, $\text{NFMora(f1-f2,I)}$ should be used instead of $\text{NF(f1-f2,I)}$.

In Magma the following piece of code does the job:
> P<x,y> := PolynomialRing(Rationals(), 2, "lex");
> I:=[x^2+x+y,x^3+x*y+y^3];
> G:=GroebnerBasis(I);
> NormalForm(I[1]-I[2],G);
Now that we have introduced techniques necessary to compute Gröbner bases, let us demonstrate one of the main applications of Gröbner bases, namely polynomial system solving. The following result shows how one can solve a polynomial system of equations, provided one can compute a Gröbner basis w.r.t lexicographic ordering.

**Theorem 11.1.19** Let \( f_1(X) = \cdots = f_m(X) = 0 \) be a system of polynomial equations defined over \( \mathbb{F}[X] \) with \( X = (x_1, \ldots, x_n) \), such that it has finitely many solutions \(^1\). Let \( I = (f_1, \ldots, f_m) \) be an ideal defined by the polynomials in the system and let \( G \) be a Gröbner basis for \( I \) with respect to \( >_{tp} \) induced by \( x_n < \cdots < x_1 \). Then there are elements \( g_1, \ldots, g_n \in G \) such that

\[
\begin{align*}
g_n & \in \mathbb{F}[x_n], \quad \text{lt}(g_n) = c_0 x_n^{m_n}, \\
g_{n-1} & \in \mathbb{F}[x_{n-1}, x_n], \quad \text{lt}(g_{n-1}) = c_{n-1} x_{n-1}^{m_{n-1}}, \\
& \ldots \\
g_1 & \in \mathbb{F}[x_1, \ldots, x_n], \quad \text{lt}(g_1) = c_1 x_1^{m_1}.
\end{align*}
\]

for some positive integers \( m_i, i = 1, \ldots, n \) and elements \( c_i \in \mathbb{F} \setminus \{0\}, i = 1, \ldots, n \).

It is clear how to solve the system \( I \) now. After computing \( G \), first solve a univariate equation \( g_n(x_n) = 0 \). Let \( a_1^{(n)}, \ldots, a_k^{(n)} \) be the roots. For every \( a_i^{(n)} \), then solve \( g_{n-1}(x_{n-1}, a_i^{(n)}) = 0 \) to find possible values for \( x_{n-1} \). Repeat this process until all the coordinates of all candidate solutions are found. The candidates form a finite set \( \text{Can} \subseteq \mathbb{F}^n \). Test all other elements of \( G \) on whether they vanish at elements of \( \text{Can} \). If there is some \( g \in G \) that does not vanish at some \( c \in \text{Can} \), then discard \( c \) from \( \text{Can} \). Since the number of solutions is finite the above procedure terminates.

**Example 11.1.20** Let us be more specific and give a concrete example of how Theorem 11.1.19 can be applied. Turn back to Example 11.1.17. Suppose we want to solve a system of equations \( x^2 + x + y = 0, x^3 + xy + y^3 = 0 \) over the rationals. We compute a Gröbner basis of the corresponding ideal and obtain that elements \( f_1 = y^6 + 2y^4 - y^3 + y^2 \) and \( f_3 = -x - y^3 - y \) belong to the Gröbner basis. Since \( f_4 \) has finitely many solutions (at most 6 over the rationals) and for every fixed value of \( y \) \( f_3 \) has exactly one solution of \( x \), we actually know that our system has finitely many solutions, both over the rationals and its algebraic closure. In order to find solutions, we have to solve the univariate equation \( y^6 + 2y^4 - y^3 + y^2 = 0 \) for \( y \). If we factorize, we obtain \( f_4 = y^2(y^4 + 2y^2 - y + 1) \), where \( y^4 + 2y^2 - y + 1 \) is irreducible over \( \mathbb{Q} \). So from the equation \( f_4 = 0 \) we only get \( y = 0 \) as a solution. Then for \( y = 0 \) the equation \( f_3 = 0 \) yields \( x = 0 \). Therefore, over rationals the given system has a unique solution \((0, 0)\).

**Example 11.1.21** Let us consider another example. Consider the following

\(^1\)Rigorously speaking, we require the system to have finitely many solutions in \( \bar{\mathbb{F}} \). Such systems (ideals) are called zero-dimensional.
system over \( \mathbb{F}_2 \):

\[
\begin{align*}
xy + x + y + z &= 0, \\
xz + yz + y &= 0, \\
x + yz + z &= 0, \\
x^2 + x &= 0, \\
y^2 + y &= 0, \\
z^2 + z &= 0.
\end{align*}
\]

Note that the field equations \( x^2 + x = 0, y^2 + y = 0, z^2 + z = 0 \) make sure that any solution for the first three equations actually lies in \( \mathbb{F}_2 \). Since \( \mathbb{F}_2 \) is a finite field, we automatically get that the system above has finitely many solutions (in fact not more than \( 2^3 = 8 \)). One can show that reduced Gröbner basis (see Remark 11.1.9) w.r.t lexicographic ordering with \( x > y > z \) of the corresponding ideal is: \( G = \{ z^2 + z, y + z, x \} \). From this we obtain that the system in question has two solutions: \((0,0,0) \) and \((0,1,1) \).

In Sections 11.2 and 11.2 we will see many more situations when Gröbner bases are applied in the solving context. Gröbner basis techniques are also used for answering many other important questions. To end this section, we give one such application. *** should this go to Section 11.3? ***

**Example 11.1.22** Sometimes it is needed to obtain explicit equations relating certain variables from given implicit ones. The following example is quite typical in algebraic cryptanalysis of block ciphers. One of the main building blocks of modern block ciphers are the so-called S–Boxes, local non-linear transformations that in composition with other, often linear, mappings compose a secure block cipher. Suppose we have an S–Box \( S \) that transforms a 4-bit vector into a 4-bit vector in a non-linear way as follows. Consider a non-zero binary vector \( x \) as an element in \( \mathbb{F}_{2^4} \) via an identification of \( \mathbb{F}_{2^4} \) and \( \mathbb{F}_{2^4} = \mathbb{F}_2[a]/(a^4 + a + 1) \) done in a usual way, so that e.g. a vector \((0,1,0,0)\) is mapped to the primitive element \( a \), and \((0,1,0,1)\) is mapped to \( a + a^3 \). Now if \( x \) is considered as an element of \( \mathbb{F}_{2^4} \) the S–Box \( S \) maps it to \( y = x^{-1} \) and then considers it again as a vector via the above identification. The zero vector is mapped to the zero vector. Not going deeply into details, we just state that such a transformation can be represented over \( \mathbb{F}_2 \) as a system of quadratic equations that implicitly relate the input variables \( x \) with the output variables \( y \). The equations are

\[
\begin{align*}
y_0 x_0 + y_3 x_1 + y_2 x_2 + y_1 x_3 + 1 &= 0, \\
y_1 x_0 + y_0 x_1 + y_3 x_1 + y_2 x_2 + y_3 x_2 + y_1 x_3 + y_2 x_3 &= 0, \\
y_0 x_0 + y_1 x_1 + y_0 x_2 + y_3 x_2 + y_2 x_3 + y_3 x_3 &= 0, \\
y_3 x_0 + y_2 x_1 + y_1 x_2 + y_0 x_3 + y_3 x_3 &= 0,
\end{align*}
\]

together with the field equations \( x_i^2 + x_i = 0 \) and \( y_i^2 + y_i = 0 \) for \( i = 0, \ldots, 3 \). The equations do not describe the part when 0 is mapped to 0, so only the inversion is modeled.

In certain situations it is more preferable to have explicit relations that would show how the output variables \( y \) depend on the input variables \( x \). For this the following technique is used. Consider the above equations as polynomials in the same polynomial ring \( \mathbb{F}_2[y_0, \ldots, y_3, x_0, \ldots, x_3] \) with \( y_0 > \cdots > y_3 > x_0 > \cdots > x_3 \) w.r.t the block ordering with blocks being \( y \)- and \( x \)-variables, and the ordering is degree reverse lexicographic in each block (see Example 11.1.6). In
this ordering, each monomial in $y$-variables will be larger than any monomial in $x$-variable, regardless of their degree. This ordering is a so-called elimination ordering for $y$ variables. The reduced Gröbner basis of the ideal composed of the above equations is

$$x_i^2 + x_i,$$

the field equations on $x$;

$$(x_0 + 1) \cdot (x_1 + 1) \cdot (x_2 + 1) \cdot (x_3 + 1),$$

which provides that $x$ should not be the all-zero vector, and

$$y_3 + x_1x_2x_3 + x_0x_3 + x_1x_3 + x_2x_3 + x_1 + x_2 + x_3,$$

$$y_2 + x_0x_2x_3 + x_0x_1 + x_0x_2 + x_0x_3 + x_2 + x_3,$$

$$y_1 + x_0x_1x_3 + x_0x_1 + x_0x_2 + x_1x_2 + x_1x_3 + x_3,$$

$$y_0 + x_0x_1x_2 + x_1x_2x_3 + x_0x_2 + x_1x_2 + x_0 + x_1 + x_2 + x_3,$$

which give explicit relations on $y$ in terms of $x$. Interestingly enough, the field equations together with the latter explicit equations describe the entire S–Box transformation, so the case $0 \mapsto 0$ is also covered.

Using similar techniques one may obtain other interesting properties of ideals, which come in hand in different applications.

11.1.3 Exercises

11.1.1 Let $R = \mathbb{Q}[x, y, z]$ and let $F = \{f_1, f_2\}$ with $f_1 = x^2 + xy + z^2, f_2 = y^2 + z$ and let $f = x^3 + 2y^3 - z^3$. The monomial ordering is degree lexicographic. Compute $\text{NF}(f|F)$. Use the procedure $\text{NFMora}$ from the Singular’s $\text{teachstd.lib}$ to check your result.

11.1.2 Let $R = \mathbb{F}_2[x, y, z]$ and let $F = \{f_1, f_2\}$ with $f_1 = x^2 + xy + z^2, f_2 = xy + z^2$. The monomial ordering is lexicographic. Compute a Gröbner basis of $\langle F \rangle$. Use the procedure $\text{standard}$ from the Singular’s $\text{teachstd.lib}$ to check your result.

11.1.3 [CAS] Recall that in Example 11.1.20 we came to the conclusion that the only solution over the rationals for the system is $(0, 0)$. Use Singular’s library $\text{solve.lib}$ and in particular the command $\text{solve}$ to find also complex solutions of this system.

11.1.4 Upgrade Algorithm 11.2 so that it also returns $a_i$’s from Theorem 11.1.19.

11.1.5 Prove the so-called product criterion: if polynomials $f$ and $g$ are such that $\text{lm}(f)$ and $\text{lm}(g)$ are co-prime, then $\text{NF}(\text{spoly}(f, g)\langle\{f, g\}\rangle) = 0$.

11.1.6 Do the following sets constitute a Gröbner basis:

---

2By setting printing level appropriately, procedures of teachstd.lib enable tracking their run. Therewith one may see exactly what a corresponding algorithm is doing.
11.2. DECODING CODES WITH GRÖBNER BASES

1. \( F_1 := \{ xy + 1, yz + x + y + 2 \} \subset \mathbb{Q}[x, y, z] \) with the degree ordering being degree lexicographic?

2. \( F_2 := \{ x + 20, y + 10, z + 12, u + 1 \} \subset \mathbb{F}_{23}[x, y, z, u] \) with the degree ordering being the block ordering with blocks \((x, y)\) and \((z, u)\) and degree reverse lexicographic ordering inside the blocks?

11.2 Decoding codes with Gröbner bases

As the first application of the Gröbner basis method we consider decoding linear codes. For the clarity of presentation we make an emphasis on cyclic codes. We consider Cooper’s philosophy or the power sums method in Section 11.2.1 and the method of generalized Newton identities in Section 11.2.2. In Section 11.2.3 we provide a brief overview of methods for decoding general linear codes.

11.2.1 Cooper’s philosophy

Now we will give an introduction to the so-called Cooper’s philosophy or the power sums method. This method uses the special form of a parity-check matrix of a cyclic code. The main idea is to write these parity check equations with unknowns for error positions and error values and then solve with respect to these unknowns by adding some natural restrictions on them. Let \( F = F_q^m \) be the splitting field of \( X^n - 1 \) over \( F_q \). Let \( a \) be a primitive \( n \)-th root of unity in \( F \). If \( i \) is in the defining set of a cyclic code \( C \) (Definition ??), then

\[
(1, a^i, \ldots, a^{(n-1)i})^T c = c_0 + c_1 a^i + \cdots + c_{n-1} a^{(n-1)i} = c(a^i) = 0,
\]

for every codeword \( c \in C \). Hence \((1, a^i, \ldots, a^{(n-1)i})\) is a parity check of \( C \). Let \( \{ i_1, \ldots, i_r \} \) be a defining set of \( C \). Then a parity check matrix \( H \) of \( C \) can be represented as a matrix with entries in \( F \) (see also Section 7.5.3):

\[
H = \begin{pmatrix}
1 & a^{i_1} & a^{2i_1} & \cdots & a^{(n-1)i_1} \\
1 & a^{i_2} & a^{2i_2} & \cdots & a^{(n-1)i_2} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & a^{i_r} & a^{2i_r} & \cdots & a^{(n-1)i_r}
\end{pmatrix}.
\]

Let \( c, r \) and \( e \) be the transmitted codeword, the received word and the error vector, respectively. Then \( r = c + e \). Denote the corresponding polynomials by \( c(x), r(x) \) and \( e(x) \), respectively. If we apply the parity check matrix to \( r \), we obtain

\[
s^T := Hr^T = H(e^T + e^T) = He^T + He^T = He^T,
\]

since \( He^T = 0 \), where \( s \) is the syndrome vector. Define \( s_i = r(a^i) \) for all \( i = 1, \ldots, n \). Then \( s_i = e(a^i) \) for all \( i \) in the complete defining set, and these \( s_i \) are called the known syndromes. The remaining \( s_i \) are called the unknown syndromes. We have that the vector \( s \) above has entries \( s = (s_{i_1}, \ldots, s_{i_r}) \). Let \( t \) be the number of errors that occurred while transmitting \( c \) over a noisy channel. If the error vector is of weight \( t \), then it is of the form \( e = (0, \ldots, 0, e_{j_1}, 0, \ldots, 0, e_{j_2}, 0, \ldots, 0, e_{j_t}, 0, \ldots, 0) \).
More precisely there are \( t \) indices \( j_l \) with \( 1 \leq j_1 < \cdots < j_t \leq n \) such that \( e_{j_l} \neq 0 \) for all \( l = 1, \ldots, t \) and \( e_j = 0 \) for all \( j \) not in \{ \( j_1, \ldots, j_t \) \}. We obtain

\[
s_{iu} = r(a^{iu}) = e(a^{iu}) = \sum_{l=1}^{t} e_j(a^{iu})^{j_l}, \quad 1 \leq u \leq r. \tag{11.2}
\]

The \( a^{j_1}, \ldots, a^{j_t} \) but also the \( j_1, \ldots, j_t \) are the error locations, and the \( e_{j_1}, \ldots, e_{j_t} \) are the error values. Define \( z_l = a^{j_l} \) and \( y_l = e_{j_l} \). Then \( z_1, \ldots, z_t \) are the error locations and \( y_1, \ldots, y_t \) are the error values and the syndromes in (11.2) become generalized power sum functions

\[
s_{iu} = \sum_{l=1}^{t} y_l z_l^{iu}, \quad 1 \leq u \leq r. \tag{11.3}
\]

In the binary case the error values are \( y_i = 1 \), and the syndromes are the ordinary power sums.

Now we give a description of Cooper’s philosophy. As the receiver does not know how many errors occurred, the upper bound \( t \) is replaced by the error-correcting capacity \( c \) and some \( z_l \)’s are allowed to be zero, while assuming that the number of errors is at most the error-correcting capacity \( c \). The following variables are introduced: \( X_1, \ldots, X_r, Z_1, \ldots, Z_e \) and \( Y_1, \ldots, Y_e \), where \( X_u \) stands for the syndrome \( s_{iu}, 1 \leq u \leq r \); \( Z_l \) stands for the error location \( z_l \) for \( 1 \leq l \leq t \), and 0 for \( t < l \leq e \); and finally \( Y_l \) stands for the error value \( y_l \) for \( 1 \leq l \leq t \), and any element of \( \mathbb{F}_q \setminus \{0\} \) for \( t < l \leq e \). The syndrome equations (11.2) are rewritten in terms of these variables as power sums:

\[
f_u := \sum_{l=1}^{e} Y_l Z_l^{iu} - X_u = 0, \quad 1 \leq u \leq r.
\]

We also add some other equations in order to specify the range of values that can be achieved by our variables, namely:

\[
e_u := X_u^{q^m} - X_u = 0, \quad 1 \leq u \leq r,
\]

since \( s_j \in \mathbb{F} \); *** add field equations in the Appendix ***

\[
\eta_l := Z_l^{q+1} - Z_l = 0, \quad 1 \leq l \leq e,
\]

since \( a^{j_l} \) are either \( n \)-th roots of unity or zero; and

\[
\lambda_l := Y_l^{q-1} - 1 = 0, \quad 1 \leq l \leq e,
\]

since \( y_l \in \mathbb{F}_q \setminus \{0\} \). We obtain the following set of polynomials in the variables \( X = (X_1, \ldots, X_r), Z = (Z_1, \ldots, Z_e) \) and \( Y = (Y_1, \ldots, Y_e) \):

\[
F_C = \{ f_u, e_u, \eta_l, \lambda_l : 1 \leq u \leq r, 1 \leq l \leq e \} \subset \mathbb{F}_q[X, Z, Y]. \tag{11.4}
\]

The zero-dimensional ideal \( I_C \) generated by \( F_C \) is called the CRHT-syndrome ideal associated to the code \( C \), and the variety \( V(F_C) \) defined by \( F_C \) is called the CRHT-syndrome variety, after Chen, Reed, Helleseth and Truong. We have
11.2. DECODING CODES WITH GRÖBNER BASES

Initially decoding of cyclic codes was essentially brought to finding the reduced Gröbner basis of the CRHT-ideal. Unfortunately, the CRHT-variety has many spurious elements, i.e. elements that do not correspond to error positions/values. It turns out that adding more polynomials to the CRHT-ideal gives an opportunity to eliminate these spurious elements. By adding polynomials

$$\chi_{l,m} := Z_l Z_m p(n, Z_l, Z_m) = 0, \quad 1 \leq l < m \leq e$$

we ensure that for all \( l \) and \( m \) either \( Z_l \) and \( Z_m \) are distinct or at least one of them is zero. The resulting set of polynomials is

$$F'_C := \{ f_u, \epsilon_u, \eta_i, \lambda_i, \chi_{l,m} : 1 \leq u \leq r, 1 \leq i \leq e, 1 \leq l < m \leq e \} \subset F_q[X, Z, Y].$$

The ideal generated by \( F'_C \) is denoted by \( I'_C \). By investigating the structure of \( I'_C \) and its reduced Gröbner basis with respect to lexicographic order induced by \( X_1 < \cdots < X_r < Z_e < \cdots < Z_1 < Y_1 < \cdots < Y_e \), the following result may be proven.

**Theorem 11.2.1** Every cyclic code \( C \) possesses a general error-locator polynomial \( L_C \). This means that there exists a unique polynomial \( L_C \in F_q[X_1, \ldots, X_r, Z] \) that satisfies the following two properties:

- \( L_C = Z^r + a_{r-1} Z^{r-1} + \cdots + a_0 \) with \( a_j \in F_q[X_1, \ldots, X_r] \), \( 0 \leq j \leq e - 1 \);
- given a syndrome \( s = (s_1, \ldots, s_r) \in \mathbb{F}^r \) corresponding to an error of weight \( t \leq e \) and error locations \( \{k_1, \ldots, k_t\} \), if we evaluate the \( X_u = s_{i_u} \) for all \( 1 \leq u \leq r \), then the roots of \( L_C(s, Z) \) are exactly \( a^{k_1}, \ldots, a^{k_t} \) and \( 0 \) of multiplicity \( e - t \), in other words

$$L_C(s, Z) = Z^{e-t} \prod_{i=1}^{t} (Z - a^{k_i}).$$

Moreover, \( L_C \) belongs to the reduced Gröbner basis of the ideal \( I'_C \) and its is a unique element, which is a univariate polynomial in \( Z_e \) of degree \( e \). *** check this ***

Having this polynomial, decoding of the cyclic code \( C \) reduces to univariate factorization. The main effort here is finding the reduced Gröbner basis of \( I'_C \). In general this is infeasible already for moderate size codes. For small codes, though, it is possible to apply this technique successfully.

**Example 11.2.2** As an example we consider finding the general error locator polynomial for a binary cyclic BCH code \( C \) with parameters \([15, 7, 5]\) that corrects 2 errors. This code has \( \{1, 3\} \) as a defining set. So here \( q = 2, m = 4, \) and \( n = 15 \). The field \( \mathbb{F}_{16} \) is the splitting field of \( X^{15} - 1 \) over \( \mathbb{F}_2 \). In the above description we have to write equations for all syndromes that correspond to elements in
the complete defining set. Note that we may write the equations only for the
elements from the defining set \{1, 3\} as all the others are just consequences of
those. Following the description above we write generators \( F_C' \) of the ideal \( I_C' \)
in the ring \( \mathbb{F}_2[X_1, X_2, Z_1, Z_2] \):

\[
\begin{align*}
Z_1 + Z_2 - X_1, & \quad Z_1^3 + Z_2^3 - X_2, \\
X_1^{16} - X_1, & \quad X_2^{16} - X_2, \\
Z_1^{16} - Z_1, & \quad Z_2^{16} - Z_2, \\
Z_1 Z_2 p(15, Z_1, Z_2).
\end{align*}
\]

We suppress the equations \( \lambda_1 \) and \( \lambda_2 \) as error values are over \( \mathbb{F}_2 \). In order
to find the general error locator polynomial we compute the reduced Gröbner
basis \( G \) of the ideal \( I_C' \) with respect to the lexicographical order induced by
\( X_1 < X_2 < Z_2 < Z_1 \). The elements of \( G \) are:

\[
\begin{align*}
X_1^{16} + X_1, \\
X_2 X_1^{15} + X_2, \\
X_2^8 + X_2 X_1^{12} + X_2^2 X_1^3 + X_2 X_1^6, \\
Z_1 X_1^{15} + Z_2, \\
Z_2^2 + Z_2 X_1 + X_2 X_1^{14} + X_1^2, \\
Z_1 + Z_2 + X_1.
\end{align*}
\]

According to Theorem 11.2.1 the general error correcting polynomial \( L_C \) is a
unique element of \( G \) of degree 2 with respect to \( Z_2 \). So \( L_C \in \mathbb{F}_2[X_1, X_2, Z] \) is

\[
L_C(X_1, X_2, Z) = Z^2 + Z X_1 + X_2 X_1^{14} + X_1^2.
\]

Let us see how decoding using \( L_C \) works. Let

\[
r = (1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1)
\]

be a received word with 2 errors. In the field \( \mathbb{F}_{16} \) with a primitive element \( a \),
such that \( a^4 + a + 1 = 0, a \) is also a 15-th root of unity. Then the syndromes
are \( s_1 = a^2 \), \( s_3 = a^{14} \). Plug them into \( L_C \) in place of \( X_1 \) and \( X_2 \) and obtain:

\[
L_C(Z) = Z^2 + a^2 Z + a^6.
\]

Factorizing yields \( L_C = (Z + a)(Z + a^5) \). According to Theorem 11.2.1, exponents
1 and 5 show exactly the error locations minus 1. So that errors occurred
on positions 2 and 6.

**Example 11.2.3** [CAS] All the computations in the previous example may be
undertaken using the library `decodegb.lib` of Singular. The following Singular-
code yields the CRHT-ideal and its reduced Gröbner basis.

```plaintext
> LIB "decodegb.lib";
> // binary cyclic [15,7,5] code with a defining set (1,3)
> list defset=1,3; // defining set
> int n=15; // length
> int e=2; // error-correcting capacity
> int q=2; // base field size
> int m=4; // degree extension of the splitting field
> int sala=1; // indicator to add additional equations as in (11.5)
```
11.2. DECODING CODES WITH GRÖBNER BASES

> def A=sysCRHT(n,defset,e,q,m,sala);
> setring A; // set the polynomial ring for the system 'crht'
> option(redSB); // compute reduced Groebner basis
> ideal red_crht=std(crht);

Now, inspecting the ideal red_crht we see which polynomial should we take as a general error-locator polynomial according to Theorem 11.2.1.

> poly gen_err_loc_poly=red_crht[5];

At this point we have to change to a splitting field in order to do our further computations.

> list l=ringlist(basering);
> l[1][4]=ideal(a4+a+1);
> def B=ring(l);
> setring B;
> poly gen_err_loc_poly=imap(A,gen_err_loc_poly);

We can now process our received vector and compute the syndromes:

> matrix rec[1][n]=1,1,0,1,0,0,0,0,0,0,1,1,0,1;
> matrix checkrow1[1][n];
> matrix checkrow3[1][n];
> int i;
> number work=a;
> for (i=0; i<=n-1; i++) {
    > checkrow1[1,i+1]=work^i;
    > work=a^3;
    > for (i=0; i<=n-1; i++){
        > checkrow3[1,i+1]=work^i;
    > }
> }
> for (i=0; i<=n-1; i++) {
    > checkrow1[1,i+1]=work^i;
> }
> // compute syndromes
> matrix s1mat=checkrow1*transpose(rec);
> matrix s3mat=checkrow3*transpose(rec);
> number s1=number(s1mat[1,1]);
> number s3=number(s3mat[1,1]);

One can now substitute and solve

> poly specialized_gen=substitute(gen_err_loc_poly,X(1),s1,X(2),s3);
> factorize(specialized_gen);

[1]:
  [1]=1
  [2]=[Z(2)+(a)]
  [3]=[Z(2)+(a-2+a)]

[2]:
  1,1,1

One can also check that a^-5=a^-2+a.

So we have seen that it is theoretically possible to encode all the information needed for decoding a cyclic code in one polynomial. Finding this polynomial, though, is a quite challenging task. Moreover, note that the polynomial coefficients $a_j \in F_q[X_1,\ldots,X_r]$ may be quite dense, so it may be a problem even just to store the polynomial $L_C$. The method, nevertheless, provides efficient closed formulas for small codes that are relevant in practice. This method can be adapted to correct erasures and to find the minimum distance of a code.
More information on these issues is in Notes.

11.2.2 Newton identities based method

In Section 7.5.2 and Section 7.5.3 we have seen how Newton identities can be used for efficient decoding of cyclic codes up to half the BCH bound. Now we want to generalize this method and be able to decode up to half the minimum distance. In order to correct more errors we have to pay a price. Systems we have to solve are no longer linear, but quadratic. This is exactly where Gröbner basis techniques come into play.

Let us recall necessary notions. Note that we change the notation a bit, as it will be convenient for the generalization. The error-locator polynomial is defined by

$$\sigma(Z) = \prod_{l=1}^{t}(Z - z_l).$$

If this product is expanded

$$\sigma(Z) = Z^t + \sigma_1 Z^{t-1} + \cdots + \sigma_{t-1} Z + \sigma_t,$$

then the coefficients $\sigma_i$ are the elementary symmetric functions in the error locations $z_1, \ldots, z_t$.

$$\sigma_i = (-1)^i \sum_{1 \leq j_1 < j_2 < \cdots < j_i \leq t} z_{j_1} z_{j_2} \ldots z_{j_i}, \quad 1 \leq i \leq t.$$

The syndromes $s_i$ and the coefficients $\sigma_i$ satisfy the following generalized Newton identities, see Proposition 7.5.8:

$$s_i + \sum_{j=1}^{t} \sigma_j s_{i-j} = 0, \quad \text{for all } i \in \mathbb{Z}_n. \quad (11.7)$$

Now suppose that the complete defining set of the cyclic code contains the $2t$ consecutive elements $b, \ldots, b + 2t - 1$ for some $b$. Then $d \geq 2t + 1$ by the BCH bound. Furthermore the set of equations (11.7) for $i = b + t, \ldots, b + 2t - 1$ is a system of $t$ linear equations in the unknowns $\sigma_1, \ldots, \sigma_t$ with the known syndromes $s_{b+t}, \ldots, s_{b+2t-1}$ as coefficients. Gaussian elimination solves the system of equations with complexity $O(t^3)$. In this way we obtain the APGZ decoding algorithm, see Section 7.5.3. See Example 7.5.11 for the algorithm in action on a small example.

One may go further and obtain closed formulas or solve the decoding problem via the key equation, see Section ?? Section ??%. All the above mentioned algorithms from Chapter 7 decode up to the BCH error-correcting capacity, which is often strictly smaller than the true capacity. A general method was outlined by Berlekamp, Tzeng, Hartmann, Chien, and Stevens, where the unknown syndromes were treated as variables. We have

$$s_{i+n} = s_i, \quad \text{for all } i \in \mathbb{Z}_n,$$

since $s_{i+n} = r(a^{i+n}) = r(a^i)$. Furthermore

$$s_i^q = (e(a^i))^q = e(a^{iq}) = s_{qi}, \quad \text{for all } i \in \mathbb{Z}_n,$$
and
\[ \sigma_i^m = \sigma_i, \quad \text{for all } 1 \leq i \leq t. \]

So the zeros of the following set of polynomials Newton_i in the variables \( S_1, \ldots, S_n \) and \( \sigma_1, \ldots, \sigma_t \) are considered.

\[
\text{Newton}_i \begin{cases} 
\sigma_i^m - \sigma_i, & \text{for all } 1 \leq i \leq t, \\
S_{i+n} - S_i, & \text{for all } i \in \mathbb{Z}_n, \\
S_i^n - S_{ni}, & \text{for all } i \in \mathbb{Z}_n, \\
S_i + \sum_{j=1}^{t} \sigma_j S_{i-j}, & \text{for all } i \in \mathbb{Z}_n.
\end{cases}
\] (11.8)

Solutions of Newton_i are called generic, formal or one-step. Computing these solutions is considered as a preprocessing phase which has to be performed only one time. For the actual decoder for every received word \( r \) the variables \( S_i \) are specialized to the actual value \( s_i(r) \) for \( i \in S_C \). Alternatively one can solve Newton_i together with the polynomials \( S_i - s_i(r) \) for \( i \in S_C \). This is called online decoding. Note that obtaining general error-locator polynomial as in the previous subsection is an example of formal decoding: this polynomial has to be found only once.

**Example 11.2.4** Let us consider an example of decoding using Newton identities and such that the APGZ algorithm is not applicable. We consider a 3-error correcting cyclic code of length 31 with a defining set \{1, 5, 7\}. Note that BCH error-correcting capacity of this code is 2. We are aiming now at correcting 3 errors. Let us write the corresponding ideal:

\[
\begin{align*}
\sigma_1 S_{31} + \sigma_2 S_{30} + \sigma_3 S_{29} + S_1, \\
\sigma_1 S_1 + \sigma_2 S_{31} + \sigma_3 S_{30} + S_2, \\
\sigma_1 S_2 + \sigma_2 S_1 + \sigma_3 S_{31} + S_3, \\
\sigma_1 S_{i-1} + \sigma_2 S_{i-2} + \sigma_3 S_{i-3} + S_i, 4 \leq i \leq 31, \\
\sigma_i^3 + \sigma_i, i = 1, 2, 3, \\
S_{i+31} + S_i, & \text{ for all } i \in \mathbb{Z}_{31}, \\
S_i^2 + S_{2i}, & \text{ for all } i \in \mathbb{Z}_{31}.
\end{align*}
\]

Note that the equations \( S_{i+31} = S_i \), and \( S_i^2 = S_{2i} \) imply,

\[
\begin{align*}
S_1^2 + S_2, & \quad S_1^4 + S_4, & \quad S_1^6 + S_8, & \quad S_1^{16} + S_{16}, \\
S_2^2 + S_6, & \quad S_2^4 + S_{12}, & \quad S_2^8 + S_{24}, & \quad S_2^{16} + S_{17}, \\
S_3^2 + S_{10}, & \quad S_3^4 + S_{20}, & \quad S_3^8 + S_9, & \quad S_3^{16} + S_{18}, \\
S_4^2 + S_{14}, & \quad S_4^4 + S_{28}, & \quad S_4^8 + S_{25}, & \quad S_4^{16} + S_{19}, \\
S_5^2 + S_6, & \quad S_5^4 + S_{12}, & \quad S_5^8 + S_{24}, & \quad S_5^{16} + S_{17}, \\
S_6^2 + S_{14}, & \quad S_6^4 + S_{28}, & \quad S_6^8 + S_{25}, & \quad S_6^{16} + S_{19}, \\
S_7^2 + S_{16}, & \quad S_7^4 + S_{30}, & \quad S_7^8 + S_{27}, & \quad S_7^{16} + S_{23}, \\
S_8^2 + S_{22}, & \quad S_8^4 + S_{31}, & \quad S_8^8 + S_{29}, & \quad S_8^{16} + S_{21}, \\
S_{10}^2 + S_{30}, & \quad S_{15}^2 + S_{31}, & \quad S_{25}^2 + S_{27}, & \quad S_{30}^2 + S_{23}.
\end{align*}
\]

Our intent is to write \( \sigma_1, \sigma_2, \sigma_3 \) in terms of known syndromes \( S_1, S_5, S_7 \). The next step would be to compute the reduced Gröbner basis of this system with respect to some elimination order induced by \( S_{31} > \cdots > S_8 > S_6 > S_4 > \cdots > S_2 > \sigma_1 > \sigma_2 > \sigma_3 > S_7 > S_5 > S_1 \). Unfortunately the computation is quite time consuming and the result is too huge to illustrate the idea. Rather, we
do online decoding, i.e. for a concrete received \( r \) compute syndromes \( S_1, S_5, S_7 \), plug the values into the system and then find \( \sigma \)'s. Let

\[ r = (0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1) \]

be a received word with three errors. So the known syndromes we need are \( s_1 = a^7, s_5 = a^{25} \) and \( s_7 = a^{29} \). Substitute these values into the system above and compute the reduced Gröbner basis of the system. The reduced Gröbner basis with respect to the degree reverse lexicographic order (here it is possible to go without an elimination order, see Remark ??) restricted to the variables \( \sigma_1, \sigma_2, \sigma_3 \) is

\[
\begin{align*}
\sigma_3 + a^5, \\
\sigma_2 + a^3, \\
\sigma_1 + a^7
\end{align*}
\]

Corresponding values for \( \sigma \)'s gives rise to the error locator polynomial:

\( \sigma(Z) = Z^3 + a^7Z^2 + a^3Z + a^5. \)

Factoring this polynomial yields three roots: \( a^4, a^{10}, a^{22} \), which indicate error positions 5, 11, and 23.

Note also that we could have worked only with the equations for \( S_1, S_5, S_7, S_3, S_11, S_{15}, S_{31} \), but the Gröbner basis computation is harder then.

Example 11.2.5 [CAS] The following program fulfills the above computation using decodegb.lib from Singular.

\[
> \text{LIB}"\text{decodegb.lib}"; \\
> \text{int n}=31; // length \\
> \text{list defset}=1,5,7; // defining set \\
> \text{int t}=3; // number of errors \\
> \text{int q}=2; // base field size \\
> \text{int m}=5; // degree extension of the splitting field \\
> \text{def A}=\text{sysNewton(n,defset,t,q,m)}; \\
> \text{setring A}; \\
> \text{// change the ring to work in the splitting field} \\
> \text{list l}=\text{ringlist(basering)}; \\
> \text{l}[1][4]=\text{ideal}(a5+a2+1); \\
> \text{def B}=\text{ring(l)}; \\
> \text{setring B}; \\
> \text{ideal newton}=\text{imap(A,newton)}; \\
> \text{matrix rec[1][n]}=0,0,1,0,0,1,1,1,0,1,0,1,1,0,1,0,0,0,1,0,0,1,1,0,0,1,1,0,0,1; \\
> \text{// compute the parity-check rows for defining set (1,5,7)} \\
> \text{// similarly to the example with CRHT} \\
> \text{...} \\
> \text{// compute syndromes s1,s5,s7} \\
> \text{// analogously to the CRHT-example} \\
> \text{...} \\
> \text{// substitute the known syndromes in the system} \\
> \text{ideal specialize_newton}; \\
> \text{for (i=1; i<=size(newton); i++)} \{ \\
> \text{specialize_newton[i]}=\text{substitute(newton[i],S(1),s1,S(S),s5,S(7),s7)}; \\
> \}
\]
11.2. DECODING CODES WITH GRÖBNER BASES

> option(redSB);
> // find sigmas
> ideal red_spec_newt=std(specialize_newton);
> // identify values of sigma_1, sigma_2, and sigma_3
> // find the roots of the error-locator
> ring solve=(2,a),Z,lp;minpoly=a5+a2+1;
> poly error_loc=Z3+(a^4+a2)*Z2+(a^3)*Z+(a^2+1); // sigma's plugged in
> factorize(error_loc);

So as we see, by using Gröbner basis it is possible to go beyond the BCH error-correcting capacity. The price paid is the complexity of solving quadratic, as opposed to linear, systems. *** more stuff in notes ***

11.2.3 Decoding arbitrary linear codes

Now we will outline a couple of ideas that may be used for decoding arbitrary linear codes up to the full error-correcting capacity.

Decoding affine variety codes with Fitzgerald-Lax

The following method generalizes ideas of Cooper's philosophy to arbitrary linear codes. In this approach the main notion is the affine variety code. Let $P_1, \ldots, P_n$ be points in $\mathbb{F}_q^n$. It is possible to compute a Gröbner basis of an ideal $I \subseteq \mathbb{F}_q[U_1, \ldots, U_s]$ of polynomials that vanish exactly at these points. Define $I_q := I + (X_1^q - X_1, \ldots, X_s^q - X_s)$. So $I_q$ is a 0-dimensional ideal. We have $V(I_q) = \{P_1, \ldots, P_n\}$. An affine variety code $C(I, L) = \phi(L)$ is an image of the evaluation map

$$\phi : R \to \mathbb{F}_q^n,$$

$$\tilde{f} \mapsto (f(P_1), \ldots, f(P_n)),$$

where $R := \mathbb{F}_q[U_1, \ldots, U_s]/I_q$, $L$ is a vector subspace of $R$ and $\tilde{f}$ is the coset of $f$ in $\mathbb{F}_q[U_1, \ldots, U_s]$ modulo $I_q$. It is possible to show that every $q$-ary linear $[n,k]$ code, or equivalently its dual, can be represented as an affine variety code for certain choice of parameters. See Exercise 11.2.2 for such a construction in the case of cyclic codes.

In order to write a system of polynomial equations similar to the one in Section 11.2.1 one needs to generalize the CRHT approach to affine codes. Similarly to the CRHT method the system of equations (or equivalently the ideal) is composed of the "parity-check" part and the "constraints" part. Parity-check part is constructed according to the evaluation map $\phi$. Now, as can be seen from Exercise 11.2.2, the points $P_1, \ldots, P_n$ encode positions in a vector, similarly to how $a^i$ encode positions in the case of a cyclic code, $a$ being a primitive $n$-th root of unity. Therefore, one needs to add polynomials $(g_l(X_{k1}, \ldots, X_{ks}))_{l=1, \ldots, m, k=1, \ldots, t}$ for every error position. Adding other natural constraints, like field equations on error values, and then computing a Gröbner basis of the combined ideal $\mathcal{I}_C$ w.r.t certain elimination ordering, it is possible to recover both error positions (i.e. values of "error points") and error values.

In general, finding $I$ and $L$ is quite technical and it turns out that for random codes this method is quite poor, because of the complicated structure of $\mathcal{I}_C$. The method may be quite efficient, though, if a code has more structure, like in the case of geometric codes (e.g. Hermitian codes). We mention also that there
are improvements of the approach of Fitzgerald and Lax, which follow the same idea as the improvements for the CRHT-method. Namely, one adds polynomials that ensure that the error locations are different. It can be proven that affine variety codes possess the so-called **multi-dimensional general error-locator polynomial**, which is a generalization of the general error-locator polynomial from Theorem 11.2.1.

**Decoding by embedding in an MDS code**

Now we briefly outline a method that provides a system for decoding that is composed of at most quadratic equations. The main feature of the method is that we do not need field equations for the solution to lie in a correct domain. Let \( C \) be an \( \mathbb{F}_q \)-linear \([n,k]\) code with error correcting capacity \( e \). Choose a parity check matrix \( H \) of \( C \). Let \( h_1, \ldots, h_r \) be the rows of \( H \). Let \( b_1, \ldots, b_n \) be a basis of \( \mathbb{F}_q^n \). Let \( B_s \) be the \( s \times n \) matrix with \( b_1, \ldots, b_s \) as rows, then \( B = B_n \). We say that \( b_1, \ldots, b_n \) is an ordered MDS basis and \( B \) an MDS matrix if all the \( s \times s \) submatrices of \( B_s \) have rank \( s \) for all \( s = 1, \ldots, n \). Note that an MDS basis for \( \mathbb{F}_q^n \) always exists if \( n \leq q \). By extending an initial field to a sufficiently large degree, we may assume that an MDS basis exists there.

Since the parameters of a code do not change when going to a scalar extension, we may assume that our code \( C \) is defined over this sufficiently large \( \mathbb{F}_q \) with \( q \geq n \). Each row \( h_i \) is then a linear combination of the basis \( b_1, \ldots, b_n \), that is there are constants \( a_{ij} \in \mathbb{F}_q \) such that \( h_i = \sum_{j=1}^n a_{ij} b_j \). In other words \( H = AB \) where \( A \) is the \( r \times n \) matrix with entries \( a_{ij} \). For every \( i \) and \( j \), \( b_i * b_j \) is a linear combination of the basis vectors \( b_1, \ldots, b_n \), so there are constants \( \mu_{ij} \in \mathbb{F}_q \) such that \( b_i * b_j = \sum_{l=1}^n \mu_{ij}^l b_l \). The elements \( \mu_{ij}^l \in \mathbb{F}_q \) are called the **structure constants** of the basis \( b_1, \ldots, b_n \). Linear functions \( U_i \) in the variables \( U_1, \ldots, U_n \) are defined as \( U_{ij} = \sum_{l=1}^n \mu_{ij}^l U_l \).

**Definition 11.2.6** For the received vector \( r \) the ideal \( J(r) \) in the ring \( \mathbb{F}_q[U_1, \ldots, U_n] \) is generated by the elements

\[
\sum_{i=1}^n a_{ij} U_i = s_j(r) \quad \text{for} \quad j = 1, \ldots, r,
\]

where \( s(r) \) is the syndrome of \( r \). The ideal \( I(t, U, V) \) in the ring \( \mathbb{F}_q[U_1, \ldots, U_n, V_1, \ldots, V_t] \) is generated by the elements

\[
\sum_{j=1}^t U_{ij} V_j = U_{i,t+1} \quad \text{for} \quad i = 1, \ldots, n.
\]

Let \( J(t, r) \) be the ideal in \( \mathbb{F}_q[U_1, \ldots, U_n, V_1, \ldots, V_t] \) generated by \( J(r) \) and \( I(t, U, V) \).

Now we are ready to state the main result of the method.

**Theorem 11.2.7** Let \( B \) be an MDS matrix with structure constants \( \mu_{ij}^l \) and linear functions \( U_{ij} \). Let \( H \) be a parity check matrix of the code \( C \) such that \( H = AB \) as above. Let \( r = c + e \) be a received word with \( c \) in \( C \) the codeword sent and \( e \) the error vector. Suppose that the weight of \( e \) is not zero and at most \( e \). Let \( t \) be the smallest positive integer such that \( J(t, r) \) has a solution \( (u, v) \) over \( \mathbb{F}_q \). Then \( wt(e) = t \) and the solution is unique satisfying \( u = Be \). The error vector is recovered as \( e = B^{-1}u \).
So as we see, although we did not impose any field equations neither on $U$– nor on $V$–variables, we still are able to obtain a correct solution. For the case of cyclic codes by going to a certain field extension $\mathbb{F}_q$ it may be shown that the system $I(t,U,V)$ actually defines the generalized Newton identities. Therefore one of the corollaries of the above theorem that it is actually possible to work without the field equations in the method of Newton identities.

Decoding by normal form computations

Another method for arbitrary linear codes has a different approach to how one represents code-related information. Below we outline the idea for binary codes. Let $[X]$ be a commutative monoid generated by $X = \{X_1, \ldots, X_n\}$. The following mapping associates a vector of reduced exponents to a monomial:

$$\psi : [X] \to \mathbb{F}_2^n, \quad \prod_{i=1}^n X_i^{a_i} \mapsto (a_1 \mod 2, \ldots, a_n \mod 2).$$

Now, let $\{w_1, \ldots, w_k\}$ be rows of a generator matrix $G$ of the binary $[n,k]$ code $C$ with the error-correcting capacity $e$. Consider an ideal $I_C \subseteq K[X_1, \ldots, X_n]$, where $K$ is an arbitrary field:

$$I := \langle X_{w_1}^a - 1, \ldots, X_{w_k}^a - 1, X_1^2 - 1, \ldots, X_n^2 - 1 \rangle.$$

So the ideal $I_C$ encodes the information about the code $C$. The next theorem shows how one decodes using $I_C$.

**Theorem 11.2.8** Let $GB$ be the reduced Gröbner basis of $I_C$ w.r.t some degree compatible monomial ordering $<$. If $\text{wt}(\psi(NF(X^n, GB))) \leq e$, then $\psi(NF(X^n, GB))$ is the error vector corresponding to the received word $\psi(X^n)$, i.e. $\psi(X^n) - \psi(NF(X^n, GB))$ is the codeword of $C$, closest to $\psi(X^n)$.

Note that $I_C$ is a binomial ideal, and therefore $GB$ is also a binomial ideal. For binomial ideals a normal form of a monomial is again a monomial. So the computation in the theorem above are well-defined. Using the special structure of $I_C$ it is possible to improve on Gröbner basis computations to obtain $GB$, compared to usual techniques.

It is remarkable that the code-related information as well as a solution to the decoding problem is represented by exponents of monomials, whereas in all the methods we considered before these data are encoded as values of certain variables.

### 11.2.4 Exercises

**11.2.1** [CAS] Consider a binary cyclic code of length 21 with a defining set $(1,3,7,9)$. This code has parameters $[21,7,8]$, see Example 7.4.8 and Example 7.4.17. The BCH bound is 5, so we cannot correct more than 2 errors with the methods from Chapter 7. Use the full error-correction capacity and correct 3 errors in some random codeword using methods from Section 11.2.1, Section 11.2.2, and `decodegb.lib` from Singular. Note that finding the general error-locator polynomial is very intense, therefore use online decoding in the CRHT-method: plug in concrete values of syndromes before computing a Gröbner basis.

**11.2.2** Show how a cyclic code may be considered as an affine variety code from Section 11.2.3.
11.2.3 Using a method of normal forms decode one error in a random codeword of the Hamming code (Example 2.2.14). Try different coefficient fields, as well as different monomial orderings. Do you always get the same result?

11.3 Algebraic cryptanalysis

In the previous section we have seen how polynomial system solving (via Gröbner bases) is used in the problem of decoding linear codes. In this section we briefly highlight another interesting application of polynomial system solving. Namely, we will be talking about algebraic cryptanalysis of block ciphers. Block ciphers were introduced in Chapter 10 as one of the main tools for providing secure symmetric communication. There we also mentioned that there exist methods for cryptanalyzing block ciphers, i.e. distinguishing them from random permutations and using this for recovering secret key used for the encryption. Traditional methods of cryptanalysis are statistical in nature. A cryptanalyst or attacker queries a cipher seen as a black-box and set up with an unknown key with (possibly chosen) plaintexts and receives corresponding ciphertexts. By collecting many such pairs a cryptanalyst hopes to find statistical patterns that would distinguish the cipher in question from a random permutation. Algebraic cryptanalysis takes another approach. In this approach a cryptanalyst writes down a system of polynomial equations over a finite field (usually \( \mathbb{F}_2 \)), which corresponds to the cipher in question via modeling operations done by the cipher during the encryption process (and also key schedule) as algebraic (polynomial) equations. Therewith the obtained system of equations reflects the encryption process; plaintext and ciphertext are parameters of the system; key is the unknown variable represented e.g. by bit variables. After plugging in actual plaintext/ciphertext the system should yield the unknown secret key as a solution. In theory, provided that plaintext- and key lengths coincide, an attacker needs only one pair of plaintext/ciphertext to recover the key\(^3\). This feature distinguishes algebraic approach from the statistical one, where an attacker usually needs many pairs to observe some statistical pattern.

We proceed as follows. In Section 11.3.1 we describe a toy cipher, which will then be used to illustrate the idea outlined above. We will see how to write equations for the toy cipher in Section 11.3.2. We will also see that it may be possible to write equations in different ways, which can be important for actual solving. In Section 11.3.3 we address the question of writing equations for an arbitrary S-Box.

11.3.1 Toy example

As a toy block cipher we will take an iterative (Definition 10.1.9) block cipher (Definition 10.1.3) with text/key length of 16 bits and a two-round encryption. Our toy cipher is an SP-network (Definition 10.1.12). Namely in every round we have a layer of local substitutions (S-Boxes) followed by a permutation layer. Specifically, the encryption algorithm proceeds as in Algorithm 11.4.

\(^3\)He/she may need a few pairs in case the size of a plaintext and key do not coincide
element of the field $\mathbb{F}_{16} \cong \mathbb{F}_2[x]/(x^4 + x + 1)$. The SBox then takes this number and outputs an inverse in $\mathbb{F}_{16}$ for non-zero inputs, or $0 \in \mathbb{F}_{16}$ otherwise. The so obtained number is then interpreted again as a vector over $\mathbb{F}_2$ of length 4. Now the permutation layer represented by $\text{Perm}$ acts on the entire 16-bit state vector. The bit at position $i, 0 \leq i \leq 15$ is moved to position $\text{Pos}(i)$, where

$$
\text{Pos}(i) = \begin{cases} 
4 \cdot i \mod 15, & 0 \leq i \leq 14, \\
15, & i = 15.
\end{cases}
$$

(11.9)

So $\text{Perm}(w) = (w_{\text{Pos}(1)}, \ldots, w_{\text{Pos}(15)})$. Interestingly enough, this permutation provides optimal diffusion in a sense that full dependency is achieved already after 2 rounds, see Exercise 11.3.1.

Schematically the encryption process of our toy cipher is depicted on Figure ...

Algorithm 11.4 Toy cipher encryption

**Input:** A 16-bit plaintext $p$ and a 16-bit key $k$.

**Output:** A 16-bit ciphertext $c$.

**Begin**

Perform initial key addition: $w := p \oplus k = \text{AddKey}(p,k)$.

for $i = 1, \ldots, 2$ do

Perform S-box substitution: $w := \text{SBox}(w)$.

Perform a permutation $w := \text{Perm}(w)$.

Add the key: $w := \text{AddKey}(w,k) = w \oplus k$.

end for

The ciphertext is $c := w$.

return $c$

**End**

11.3.2 Writing down equations

Now let us turn to the question of how to write a system of equations that describes the encryption algorithm as in Algorithm 11.4. We would like to write equations on the bit level, i.e. over $\mathbb{F}_2$. Denote by $p = (p_0, \ldots, p_{15})$ and $c = (c_0, \ldots, c_{15})$ the plaintext and ciphertext variables that appear as parameters in our system. Then $k = (k_0, \ldots, k_{15})$ are unknown key variables. Let $x_i = (x_{i,0}, \ldots, x_{i,15}), i = 0, 1$ be the variables representing result of bitwise key addition, $y_i = (y_{i,0}, \ldots, y_{i,15}), i = 1, 2$ be variables representing outcome of the S-Boxes, and $z_i = (z_{i,0}, \ldots, z_{i,15}), i = 1, 2$ be results of the permutation layer. Now we can write the encryption process as the following system:

$$
\begin{align*}
x_0 &= p + k, \\
y_i &= \text{SBox}(x_{i-1}), & i &= 1, 2, \\
z_i &= \text{Perm}(y_i), & i &= 1, 2, \\
x_1 &= z_1 + k, \\
c &= z_2 + k.
\end{align*}
$$

(11.10)

Here $\text{SBox}$ and $\text{Perm}$ are some polynomial functions that act on variable-vectors according to Algorithm 11.4.
There are three operations that are performed in the algorithm: bitwise key addition, substitution via four 4-bit S-Boxes, and the permutation. The key addition is represented trivially as above and one can write it on the bit level as, e.g. in the initial key addition: \( x_{a,j} = p_j + k_j, 0 \leq j \leq 15 \). The permutation \( Perm \) also does not pose any problem. According to (11.9) we have that the blocks \( z_i = Perm(y_i), i = 1, 2 \) above are written as
\[
z_{i,j} = y_{i,Pos^{-1}(j)}, 0 \leq j \leq 15,
\]
where \( Pos^{-1}(j) \) may be easily computed and in fact in this case we have \( Pos^{-1} = Pos \).

An interesting question is how to write equations over \( \mathbb{F}_2 \) that would describe the S-Box transformation \( SBox \). Since \( SBox \) is composed of four parallel S-Boxes that perform inversion in \( \mathbb{F}_{16} \), we may concentrate on writing equations for one S-Box. Let \( a = (a_0, a_1, a_2, a_3) \) be input bits of the S-Box and \( b = (b_0, b_1, b_2, b_3) \) are the output bits. The way we defined S-Box, we should consider \( a \neq 0 \) as an element of \( \mathbb{F}_4 \) and then compute \( b = a^{-1} \) in \( \mathbb{F}_{16} \). Afterwards we regard \( b \) as a vector in \( \mathbb{F}_4 \). The all-zero vector is mapped to the all-zero vector. The \( SBox \) of \( a \) in \( \mathbb{F}_{16} \) is also does not pose any problem. According to (11.9) we have that the block \( z = Perm(y), i = 1, 2 \) above are written as
\[
z_{i,j} = y_{i,Pos^{-1}(j)}, 0 \leq j \leq 15,
\]
where \( Pos^{-1}(j) \) may be easily computed and in fact in this case we have \( Pos^{-1} = Pos \).

In order to fully describe the S-Box we must recall that our bit variables \( a_i \)'s and \( b_j \)'s live in \( \mathbb{F}_2 \). Therefore the field equations \( a_i^2 + a_i = 0 \) and \( b_i^2 + b_i = 0 \)
for 0 ≤ i ≤ 3 have to be added. So now we have obtained exactly the implicit equations as in Example 11.1.22.

By adding field equations for all participating variables to the equations we introduced above, we obtain a full description of the toy cipher in assumption that no zero-inversion occurs in S-Boxes, a probability of this event is computed introduced above, we obtain a full description of the toy cipher in assumption. By adding field equations for all participating variables to the equations we obtained equations as in Example 11.1.22.

for 0 ≤ i ≤ 3 have to be added. So now we have obtained exactly the implicit equations as in Example 11.1.22.

Going back to Example 11.1.22 we recall that it is possible to obtain explicit relations between the inputs and outputs. Note also that these relations now also include the case 0 → 0, if we remove the equation (a_0+1)(a_1+1)(a_2+1)(a_3+1) = 0. These explicit equations are:

\[
\begin{align*}
 b_0 &= a_0a_2a_3 + a_0a_2 + a_1a_2 + a_0 + a_1 + a_2 + a_3, \\
 b_1 &= a_0a_2a_3 + a_0a_1 + a_0a_2 + a_1a_2 + a_1a_3 + a_3, \\
 b_2 &= a_0a_2a_3 + a_0a_1 + a_0a_2 + a_0a_3 + a_2 + a_3, \\
 b_3 &= a_1a_2a_3 + a_0a_3 + a_1a_3 + a_2a_3 + a_1 + a_2 + a_3.
\end{align*}
\]

These equations may be useful in the following approach. By having explicit equations of degree three that describe the S-Boxes, one may obtain equations of degree 3·3 = 9 in the key variables only. Indeed, one should do consecutive substitutions from equation to equation in the system (11.10). One proceeds by substituting corresponding bit variables from x_0 = p + k to y_1 = SBox(x_0), therewith obtaining relations of the form y_1 = f(p, k) of degree three in k (p is assumed to be known as usual). Then substitute y_1 = f(p, k) to z_1 = Perm(y_1) and then these to x_1 = z_1 + k. One obtains relations of the form x_1 = g(p, k) again of degree three in k. Now the next substitution of x_1 = g(p, k) to y_2 = SBox(x_1) increases the degree. Namely, because g is of degree three and SBox is of degree three we obtain equations y_2 = h(p, k) of degree 3·3 = 9. The following substitutions do not increase the degree, since all the following equations are linear.

The reason for us wanting to obtain such equations in key variables only is a possibility to use more than one pair of plain-/ciphertext encoded with the same unknown key. By doing the above process for each such pair, we obtain each time 16 equations of degree 9 in the key variables k (the key stays the same). Note that if we would use the implicit representation we could not eliminate the “intermediate” variables, such as x_0, y_1, z_1, etc. Moreover, these intermediate variables depend on parameters p (and c), so these variables are all different for different plaintext/ciphertext pairs. The idea of the latter approach is to keep the number of variables as small as possible, but increase the number of equations that relate them. In the theory and practice of solving polynomial systems it has been noted that solving more overdetermined (i.e. more equations than variables) systems has a positive effect on complexity and thus on the success of solving a system in question.

Still, degree-9 equations are too hard to attack. We would like to reduce the degree of our equations. Below we outline a general principle, known as the “meet-in-the-middle” principle, to reduce the degree. As the name suggests, we would like to obtain some relations between variables in the middle, rather than at the end of encryption. For this we need to invert the second half of a cipher in question. In our case this means to invert the second round. We have
already noted that $\text{Perm} = \text{Perm}^{-1}$. Also since the S-Box transformation is an inversion in $\mathbb{F}_{16}$ with $0 \mapsto 0$ we have that $S\text{Box} = S\text{Box}^{-1}$. Now similarly with the above substitution procedure, we do “forward” substitutions:

$$x_0 = p + k \mapsto y_1 = S\text{Box}(x_0) \mapsto z_1 = Perm(y_1),$$

obtaining at the end equations $z_1 = F(p, k)$ of degree 3, and then “backward” substitutions

$$z_2 = c + k \mapsto y_2 = Perm(z_2) \mapsto x_1 = S\text{Box}(y_2) \mapsto z_1 = x_1 + k,$$

obtaining equations $z_1 = G(c, k)$ also of degree 3. Equating the two one obtains a system of 16 equations $F(p, k) = G(c, k)$ of degree 3 in key variables $k$ only. Repeating this process for each plain-/ciphertext pair, one may obtain as many equations (each time a multiple of 16) as one wants. One should not forget, of course, to include the field equations each time to make sure that the values of variables stay in $\mathbb{F}_2$. Exercise 11.3.4 elaborates on solving using this approach.

11.3.3 General S-Boxes

In the previous section we have seen how to write equations for the S-Box given by the inversion function in the field $\mathbb{F}_{16}$. Although this idea was employed in the AES, a widely used cipher, cf. Section 10.1.4, this is not a standard way to define S-Boxes in block ciphers. Usually S-Boxes are defined via so-called look-up tables, i.e. tables which explicitly prescribe an output value to a given input value. Whereas we used algebraic structure of the toy cipher in Section 11.3.1 to derive equations, it is still not clear from that exposition how to write S-Box equations in the more general case of look-up table definitions.

As an illustrating example we will use a 3-bit S-Box. This S-Box is even smaller than the one employed in our toy cipher. Still it has been proposed in one of the so-called light-weight block ciphers PrintCIPHER. The look-up table for this S-Box, call it $S_x$, is as follows:

<table>
<thead>
<tr>
<th>$x$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S(x)$</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>6</td>
<td>7</td>
<td>4</td>
<td>5</td>
<td>2</td>
</tr>
</tbody>
</table>

Here we used decimal representation for length-3 binary vectors. For example, the S-Box maps the vector $2 = (0, 1, 0)$ to the vector $3 = (1, 1, 0)$.

One method we can use to obtain explicit relations for the output values is as follows. The S-Box $S$ is a function $S : \mathbb{F}_2^3 \rightarrow \mathbb{F}_2^3$, which can be seen as a collection of functions $S_i : \mathbb{F}_2^3 \rightarrow \mathbb{F}_2$, $i = 0, 1, 2$ mapping input vectors to the bits at position 0, 1, and 2 resp. It is known that *** recall?! *** that each function defined over a finite field is actually a polynomial function.

Let us find a polynomial describing the function $S_0$. The look-up table in this case is as follows:

<table>
<thead>
<tr>
<th>$x$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_0(x)$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Denote by $x_0, x_1, x_2$ the input bits. We have

$$S_0(x_0, x_1, x_2) = S_0(0, 0, 0) \cdot (x_0 - 1)(x_1 - 1)(x_2 - 1) +$$
$$+ S_0(1, 0, 0) \cdot x_0(x_1 - 1)(x_2 - 1) +$$
$$+ S_0(0, 1, 0) \cdot (x_0 - 1)x_1(x_2 - 1) + S_0(1, 1, 0) \cdot x_0x_1(x_2 - 1) +$$
$$+ S_0(0, 0, 1) \cdot (x_0 - 1)(x_1 - 1)x_2 + S_0(1, 0, 1) \cdot x_0(x_1 - 1)x_2 +$$
$$+ S_0(0, 1, 1) \cdot (x_0 - 1)x_1x_2 + S_0(1, 1, 1) \cdot x_0x_1x_2.$$
Indeed, by assigning concrete values \((v_0, v_1, v_2)\) to \((x_0, x_1, x_2)\) we obtain:

\[
S_0(x_0, x_1, x_2) = x_0(x_1 - 1)(x_2 - 1) + (x_0 - 1)x_1(x_2 - 1) + (x_0 - 1)(x_1 - 1)x_2 + (x_0 - 1)x_1x_2 + x_0 + x_1 + x_2.
\]

Analogously we obtain polynomial expressions for \(S_1\) and \(S_2\):

\[
S_1(x_0, x_1, x_2) = x_0x_2 + x_1 + x_2,
\]

\[
S_2(x_0, x_1, x_2) = x_0x_1 + x_2.
\]

Another technique based on linear algebra gives an opportunity to obtain different relations between input and output variables. We are interested in relations of as low degree as possible and usually these are quadratic relations. Let us demonstrate how to obtain bilinear relations for the S-Box \(S\). Denote \(y_i = S_i, i = 0, 1, 2\). So we are interested in finding relations of the form:

\[
\sum_{0 \leq i, j \leq 3} a_{ij}x_iy_j = 0.
\]

In order to do this, we treat coefficients \(a_{ij}\)'s as variables. Each assignment of values to \((x_0, x_1, x_2)\) yields a unique assignment of values to \((y_0, y_1, y_2)\) according to the look-up table. Each assignment of \((x_0, x_1, x_2)\) and thus of \((y_0, y_1, y_2)\) provides us with a linear equation in \(a_{ij}\)'s by plugging in assigned values in the relation:

\[
\sum_{0 \leq i, j \leq 3} a_{ij}x_iy_j = 0,
\]

which should hold for every assignment. We may use \(2^3 = 8\) assignments for the \(x\)-variables to get 8 linear equations in \(3 \cdot 3 = 9\) variables \(a_{ij}\)'s. Each non-trivial solution of this homogeneous linear system provides us with a non-trivial bilinear relation between the \(x\)- and \(y\)-variables. Exercise 11.3.5 works out the details of this approach for the example of \(S\). We just mention that, e.g., \(x_0y_2 + x_1y_0 + x_1y_1 + x_2y_1 + x_2y_2 = 0\) is one such bilinear relation. There exist overall 3 linearly independent bilinear relations. Using exactly the same idea one may find other relations, e.g., general quadratic:

\[
\sum_{0 \leq i, j \leq 3} a_{ij}x_iy_j + \sum_{0 \leq i<j \leq 3} b_{ij}x_ix_j + \sum_{0 \leq i<j \leq 3} c_{ij}y_iy_j + \sum_{0 \leq i \leq 3} d_i + \sum_{0 \leq i \leq 3} e_iy_i = 0
\]

and others that may be of interest. Clearly, techniques of this section apply also to other S-Boxes defined by look-up tables. See Exercise 11.3.6 for the treatment of the S-Box coming from the block cipher PRESENT.

### 11.3.4 Exercises

#### 11.3.1
Prove that in the toy cipher of Section 11.3.1 every ciphertext bit depends on every plaintext bit.

#### 11.3.2
Considering that inputs to the S-Boxes of the toy cipher are all uniformly distributed and independent random values, what is the probability that no zero-inversion occurs during the encryption?

#### 11.3.3
[CAS] Using Magma and/or Singular and/or SAGE/PolyBoRi write an equation system representing the toy cipher from Section 11.3.1. When defining a base ring for your Gröbner bases computations think/experiment on the following questions:

- which ordering of variables works better?
• which monomial ordering is better? try e.g. lexicographic, degree reverse lexicographic;
• does the result of the computation change when changing the ordering? why?
• what happens if you remove the field equations?
• try explicit vs. implicit representations for the S-Box;

11.3.4 Work out the meet-in-the-middle approach of Section 11.3.2. For the substitution use the command \texttt{subst} in Singular.

11.3.5 Find bilinear relations for the S-Box $S$ using the linear algebra approach from Section 11.3.3. Compose a matrix for the homogeneous system as described in the text. The rows will be indexed by assignments of $(x_0, x_1, x_2)$ and columns by indexes $(i, j)$ of the variables $a_{i,j}$ that are coefficients for $x_ify_j$. Show that rank of this matrix is 7 and thus you can get 3 linearly independent solutions. Write down 3 linearly independent bilinear relations for $S$.

11.3.6 An S-Box in the block cipher PRESENT is a non-linear transformation of 4-bit vectors. Its look-up table is as follows:

<table>
<thead>
<tr>
<th>$x$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>SBox($x$)</td>
<td>12</td>
<td>5</td>
<td>6</td>
<td>11</td>
<td>9</td>
<td>0</td>
<td>10</td>
<td>13</td>
<td>3</td>
<td>14</td>
<td>15</td>
<td>8</td>
<td>4</td>
<td>7</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

• Write down equations that relate input bits explicitly to output bits. What is the degree of these equations?
• Find all linearly independent bilinear relations and general quadratic relations between inputs and outputs.

11.4 Notes
Chapter 12

Coding theory with computer algebra packages

Stanislav Bulygin

In this chapter we give a brief overview of the three computer algebra systems: Singular, Magma, and GAP. We concentrate our attention on things that are useful for the book. On other topics, as well as language semantics and syntax the reader is referred to the corresponding web-cites.

12.1 Singular

As is cited at www.singular.uni-kl.de: “SINGULAR is a Computer Algebra System for polynomial computations with special emphasis on the needs of commutative algebra, algebraic geometry, and singularity theory”. In the context of this book, we use some functionality provided for AG-codes (brnoeth.lib), decoding linear code via polynomial system solving (decodegb.lib), teaching cryptography (crypto.lib, atkins.lib) and Gröbner bases (teachstd.lib). Singular can be downloaded free of charge from http://www.singular.uni-kl.de/download.html for different platforms (Linux, Windows, Mac OS). The current version is 3-0-4 *** to change at the end *** The web-cite provides an online manual at http://www.singular.uni-kl.de/Manual/latest/index.htm. Below we provide the list of commands that can be used to work with objects presented in this book together with short descriptions. Examples of use can be looked at from the links given below. More examples occur throughout the book at the corresponding places. The functionality mentioned above is provided via libraries and not the kernel function to load a library in Singular one has to type (brnoeth.lib as an example):

> LIB"brnoeth.lib";


Description: Implementation of the Brill-Noether algorithm for solving the
CHAPTER 12. CODING THEORY WITH COMPUTER ALGEBRA PACKAGES

Riemann-Roch problem and applications in Algebraic Geometry codes. The computation of Weierstrass semigroups is also implemented. The procedures are intended only for plane (singular) curves defined over a prime field of positive characteristic. For more information about the library see the end of the file brnoeth.lib.

Selected procedures:

- NSplaces: computes non-singular places with given degrees
- BrillNoether: computes a vector space basis of the linear system $L(D)$
- Weierstrass: computes the Weierstrass semigroup of $C$ at $P$ up to $m$
- AGcode_L: computes the evaluation AG code with divisors $G$ and $D$
- AGcode_Omega: computes the residual AG code with divisors $G$ and $D$
- decodeSV: decoding of a word with the basic decoding algorithm
- dual_code: computes the dual code

decodegb.lib: Decoding and minimum distance of linear codes with Gröbner bases by S. Bulygin (…)

Description: In this library we generate several systems used for decoding cyclic codes and finding their minimum distance. Namely, we work with the Cooper’s philosophy and generalized Newton identities. The original method of quadratic equations is worked out here as well. We also (for comparison) enable to work with the system of Fitzgerald-Lax. We provide some auxiliary functions for further manipulations and decoding. For an overview of the methods mentioned above, see “Decoding codes with GB” section of the manual. For the vanishing ideal computation the algorithm of Farr and Gao is implemented.

Selected procedures:

- sysCRHT: generates the CRHT-ideal as in Cooper’s philosophy
- sysNewton: generates the ideal with the generalized Newton identities
- syndrome: computes a syndrome w.r.t. the given check matrix
- sysQE: generates the system of quadratic equations for decoding
- errorRand: inserts random errors in a word
- randomCheck: generates a random check matrix
- mindist: computes the minimum distance of a code
- decode: decoding of a word using the system of quadratic equations
- decodeRandom: a procedure for manipulation with random codes
- decodeCode: a procedure for manipulation with the given code
- vanishId: computes the vanishing ideal for the given set of points

Description: The library contains procedures to compute the discrete logarithm, primality-tests, factorization included elliptic curve methods. The library is intended to be used for teaching purposes but not for serious computations. Sufficiently high printlevel allows to control each step, thus illustrating the algorithms at work.

atkins.lib: Procedures for teaching Elliptic Curve cryptography (primality test) by S. Steidel (http://www.singular.uni-kl.de/Manual/latest/sing_1281.htm#SEC1340)

Description: The library contains auxiliary procedures to compute the elliptic curve primality test of Atkin and the Atkin’s Test itself. The library is intended to be used for teaching purposes but not for serious computations. Sufficiently high printlevel allows to control each step, thus illustrating the algorithms at work.


Description: The library is intended to be used for teaching purposes, but not for serious computations. Sufficiently high printlevel allows to control each step, thus illustrating the algorithms at work. The procedures are implemented exactly as described in the book 'A SINGULAR Introduction to Commutative Algebra' by G.-M. Greuel and G. Pfister (Springer 2002).

Selected procedures:

- tail: tail of f
- leadminomial: leading monomial as poly (also for vectors)
- monomialLcm: lcm of monomials m and n as poly (also for vectors)
- spoly: s-polynomial of f [symmetric form]
- NFMor: normal form of i w.r.t Mora algorithm
- prodcrit: test for product criterion
- chaincrit: test for chain criterion
- standard: standard basis of ideal/module

12.2 Magma

"Magma is a large, well-supported software package designed to solve computationally hard problems in algebra, number theory, geometry and combinatorics" – this is a formulation given at the official web-site http://magma.maths.usyd.edu.au/magma/. The current version is 2.15-7 *** to change at the end ***. In this book we use illustrations with Magma for different coding constructions: general, as well as more specific, such as AG-codes, also some machinery for working with algebraic curves, as well as a few procedures for cryptography. Although Magma is a non-commercial system, it is still not free of charge: one has to purchase a license to work with it. Details can be found at http://magma.maths.usyd.edu.au/magma/Ordering/ordering.shtml. Still one can try to run simple Magma-code in the so-called "Magma-Calculator"
Next we describe briefly some procedures that come in hand while dealing with objects from this book. We list only a few commands to give a flavor of functionality. One can get a lot more from the manual.

### 12.2.1 Linear codes

Full list of commands with descriptions can be found at [http://magma.maths.usyd.edu.au/magma/htmlhelp/text1667.htm](http://magma.maths.usyd.edu.au/magma/htmlhelp/text1667.htm)

- **LinearCode**: constructs a linear codes as a vector subspace
- **PermutationCode**: permutes positions in a code
- **RepetitionCode**: constructs a repetition code
- **RandomLinearCode**: constructs random linear code
- **CyclicCode**: constructs a cyclic code
- **ReedMullerCode**: constructs a Reed-Muller code
- **HammingCode**: constructs a Hamming code
- **BCHCode**: constructs a BCH code
- **ReedSolomonCode**: constructs a Reed-Solomon code
- **GeneratorMatrix**: yields the generator matrix
- **ParityCheckMatrix**: yields the parity check matrix
- **Dual**: constructs the dual code
- **GeneratorPolynomial**: yields the generator polynomial of the given cyclic code
- **CheckPolynomial**: yields the check polynomial of the given cyclic code
- **Random**: yields a random codeword
- **Syndrome**: yields a syndrome of a word
- **Distance**: yields distance between words
- **MinimumDistance**: computes minimum distance of a code
- **WeightEnumerator**: computes the weight enumerator of a code
- **ProductCode**: constructs a product code from the given two
- **SubfieldSubcode**: constructs a subfield subcode
- **McElieceAttack**: Runs basic attack on the McEliece cryptosystem
- **GriesmerBound**: provides the Griesmer bound for the given parameters
12.2. MAGMA

- **SpherePackingBound**: provides the sphere packing bound for the given parameters
- **BCHBound**: provides the BCH bound for the given cyclic code
- **Decode**: decode a code with standard methods
- **MattsonSolomonTransform**: computes the Mattson-Solomon transform
- **AutomorphismGroup**: computes the automorphism group of the given code

12.2.2 AG-codes

Full list of commands with descriptions can be found at [http://magma.maths.usyd.edu.au/magma/htmlhelp/text1686.htm](http://magma.maths.usyd.edu.au/magma/htmlhelp/text1686.htm)

- **AGCode**: constructs an AG-code
- **AGDualCode**: constructs a dual AG-code
- **HermitianCode**: constructs a Hermitian code
- **GoppaDesignedDistance**: returns designed Goppa distance
- **AGDecode**: basic algorithm for decoding an AG-code

12.2.3 Algebraic curves

Full list of commands with descriptions can be found at [http://magma.maths.usyd.edu.au/magma/htmlhelp/text1686.htm](http://magma.maths.usyd.edu.au/magma/htmlhelp/text1686.htm)

- **Curve**: constructs a curve
- **CoordinateRing**: computes the coordinate ring of the given curve with Gröbner basis techniques
- **JacobianMatrix**: computes the Jacobian matrix
- **IsSingular**: test if the given curve has singularities
- **Genus**: computes genus of a curve
- **EllipticCurve**: constructs an elliptic curve
- **AutomorphismGroup**: computes the automorphism of the given curve
- **FunctionField**: computes the function field of the given curve
- **Valuation**: computes a valuation of the given function w.r.t the given place
- **GapNumbers**: yields gap numbers
- **Places**: computes places of the given curve
- **RiemannRochSpace**: computes the Riemann-Roch space
- **Basis**: computes a sequence containing a basis of the Riemann-Roch space $L(D)$ of the divisor $D$.

- **CryptographicCurve**: given the finite field computes an elliptic curve $E$ over a finite field together with a point $P$ on $E$ such that the order of $P$ is a large prime and the pair $(E, P)$ satisfies the standard security conditions for being resistant to MOV and Anomalous attacks.

### 12.3 GAP

In this subsection we consider GAP computational discrete algebra system that is “a system for computational discrete algebra, with particular emphasis on Computational Group Theory”. GAP stands for Groups, Algorithms, Programming, [http://www.gap-system.org](http://www.gap-system.org). Although primary concern of GAP is computations with groups, it also provides coding-oriented functionality via the GUAVA package, [http://www.gap-system.org/Packages/guava.html](http://www.gap-system.org/Packages/guava.html). GAP can be downloaded for free from [http://www.gap-system.org/Download/index.html](http://www.gap-system.org/Download/index.html). The current GAP version is 4.4.12, the current GUAVA version is 3.9.

*** to change at the end *** As before, we only list here some procedures to provide an understanding of which things can be done with GUAVA/GAP. Package GUAVA is included as follows:

```gap
> LoadPackage("guava");
```

Online manual for guava can be found at [http://www.gap-system.org/Manuals/pkg/guava3.9/htm/chap0.html](http://www.gap-system.org/Manuals/pkg/guava3.9/htm/chap0.html)

Selected procedures:

- **RandomLinearCode**: constructs a random linear code
- **GeneratorMatCode**: constructs a linear code via its generator matrix
- **CheckMatCode**: constructs a linear code via its parity check matrix
- **HammingCode**: constructs a Hamming code
- **ReedMullerCode**: constructs a Reed-Muller code
- **GeneratorPolCode**: constructs a cyclic code via its generator polynomial
- **CheckPolCode**: constructs a cyclic code via its check polynomial
- **RootsCode**: constructs a cyclic code via roots of the generator polynomial
- **BCHCode**: constructs a BCH code
- **ReedSolomonCode**: constructs a Reed-Solomon code
- **CyclicCodes**: returns all cyclic codes of given length
- **EvaluationCode**: construct an evaluation code
- **AffineCurve**: sets a framework for working with an affine curve
- **GoppaCodeClassical**: construct a classical geometric Goppa code
- **OnePointAGCode**: construct a one-point AG-code
- **PuncturedCode**: construct a punctured code for the given
- **DualCode**: constructs the dual code to the given
- **UUVCCode**: constructs a code via the \((u|u+v)\)-construction
- **LowerBoundMinimumDistance**: yields the best lower bound on the minimum distance available
- **UpperBoundMinimumDistance**: yields the best upper bound on the minimum distance available
- **MinimumDistance**: yields the minimum distance of the given code
- **WeightDistribution**: yield the weight distribution of the given code
- **Decode**: general decoding procedure

## 12.4 Sage

Sage framework provides an opportunity to use strengths of many open-source computer algebra systems (among them are Singular and GAP) for developing effective code for solving different mathematical problems. The general framework is made possible through the python-interface. Sage is thought as an open-source alternative to commercial systems, such as Magma, Maple, Mathematica, and Matlab. Sage provides tools for a wide variety of algebraic and combinatorial objects among other things. For example functionality for coding theory and cryptography is present, as well as functionality for working with algebraic curves. The web-page of the project is [http://www.sagemath.org/](http://www.sagemath.org/). One can download Sage from [http://www.sagemath.org/download.html](http://www.sagemath.org/download.html). The reference manual for Sage is available at [http://www.sagemath.org/doc/reference/](http://www.sagemath.org/doc/reference/). Now we briefly describe some commands that may come in hand while working with this book.

### 12.4.1 Coding Theory

Manual available at [http://www.sagemath.org/doc/reference/coding.html](http://www.sagemath.org/doc/reference/coding.html) Coding-functionality of Sage has a lot in common with the one of GAP/GUAVA. In fact, for many commands Sage uses implementations available from GAP.

*Selected procedures:*

- **LinearCodeFromCheckMatrix**: constructs a linear code via its parity check matrix
- **RandomLinearCode**: constructs a random linear code
- **CyclicCodeFromGeneratingPolynomial**: constructs a cyclic code via its generator polynomial
- **QuadraticResidueCode**: constructs a quadratic residue cyclic code
- **ReedSolomonCode**: constructs a Reed-Solomon code
12.4.2 Cryptography


Selected procedures/classes:
- SubstituteCryptosystem: defines a substitution cryptosystem/cipher
- VigenereCryptosystem: defines the Vigenere cryptosystem/cipher
- \texttt{lfsr	extunderscore sequence}: produces an output of the given LFSR
- \texttt{SR}: returns a small scale variant for the AES

12.4.3 Algebraic curves


Selected procedures/classes:
- \texttt{EllipticCurve	extunderscore finite	extunderscore field}: constructs an elliptic curves over a finite field
- \texttt{trace	extunderscore of	extunderscore frobenius}: computes the trace of Frobenius of an elliptic curve
- \texttt{cardinality}: computes the number of rational points of an elliptic curve
- \texttt{HyperellipticCurve	extunderscore finite	extunderscore field}: constructs a hyperelliptic curves over a finite field

12.5 Coding with computer algebra

12.5.1 Introduction

12.5.2 Error-correcting codes

Example 12.5.1 Let us construct Example 2.1.6 for \( n = 5 \) using GAP/GUAVA. First, we need to define the list of codewords

\[
\texttt{M := Z(2)^0 * [ [1,1,0,0,0], [1,0,1,0,0], [1,0,0,1,0], [1,0,0,0,1], [0,1,1,0,0], [0,1,0,1,0], [0,1,0,0,1], [0,0,1,1,0], [0,0,1,0,1], [0,0,0,1,1] ];}
\]

In GAP \( Z(q) \) is a primitive element of the field \( GF(q) \). So multiplying the list \( M \) by \( Z(2)^0 \) we make sure that the elements belong to \( GF(2) \). Now construct the code:

\[
\texttt{C:=ElementsCode(M,"Example 2.1.6 for n=5",GF(2));}
\]
Example 12.5.2 Let us construct the Hamming $[7, 4, 3]$ code in GAP/GUAVA and Magma. Both systems have a built-in command for this. In GAP
\[
\text{\texttt{C:=HammingCode(3,GF(2));}}
\]
a linear $[7, 4, 3]$ Hamming $(3, 2)$ code over GF(2)
Here the syntax is HammingCode($r$, GF($q$)) where $r$ is the redundancy and GF($q$) is the defining alphabet. We can extract a generator matrix as follows
\[
\text{\texttt{M:=GeneratorMat(C);}}
\]
\[
\text{\texttt{Display(M);}}
\]
1 1 1 . . . .
1 . . 1 1 . .
. 1 . 1 . 1 .
1 1 . 1 . 1
Two semicolons indicate that we do not want an output of a command be printed on the screen. Display provides a nice way to represent objects.
In Magma we do it like this
\[
\text{\texttt{C:=HammingCode(GF(2),3);}}
\]
\[
\text{\texttt{C;}}
\]
$[7, 4, 3]$ "Hamming code (r = 3)" Linear Code over GF(2)
Generator matrix:
\[
\begin{bmatrix}
1 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 1 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 1 & 1 & 0 & 1
\end{bmatrix}
\]
So here the syntax is reverse.

Example 12.5.3 Let us construct $[7, 4, 3]$ binary Hamming code via its parity check matrix. In GAP/GUAVA we proceed as follows
\[
\text{\texttt{H1:=Z(2)^0*[\{[1,0,0,0,1,1,0],[0,1,0,0,0,1,1],[0,0,1,0,1,1,1],[0,0,0,1,1,0,1]\};}}
\]
\[
\text{\texttt{H:=TransposedMat(H1);}}
\]
\[
\text{\texttt{C:=CheckMatCode(H,GF(2));}}
\]
a linear $[7, 4, 1..3]1$ code defined by check matrix over GF(2)
We can now check the property of the check matrix:
\[
\text{\texttt{G:=GeneratorMat(C);}}
\]
\[
\text{\texttt{Display(G+H1);}}
\]
We can also compute syndromes:

\[ c := \text{CodewordNr}(C, 7); \]
\[ \begin{array}{cccccccc}
1 & 1 & 0 & 0 & 1 & 1 & 0 \\
\end{array} \]
\[ \text{Syndrome}(C, c); \]
\[ \begin{array}{cccc}
0 & 0 & 0 \\
\end{array} \]
\[ e := \text{Codeword}("1000000"); \]
\[ \text{Syndrome}(C, c + e); \]
\[ \begin{array}{cccc}
1 & 0 & 0 \\
\end{array} \]

So we have taken the 7th codeword in the list of codewords of \( C \) and showed that its syndrome is 0. Then we introduced an error at the first position: the syndrome is non-zero.

In Magma one can generate codes only by vector subspace generators. So the way to generate a code via its parity check matrix is to use the \texttt{Dual} command, see Example 12.5.4. So we construct the Hamming code as in Example 12.5.2 and then proceed as above.

\[ C := \text{HammingCode}(\text{GF}(2), 3); \]
\[ H := \text{ParityCheckMatrix}(C); \]
\[ H; \]
\[ \begin{array}{cccccccc}
1 & 0 & 0 & 1 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 1 & 1 \\
\end{array} \]
\[ G := \text{GeneratorMatrix}(C); \]
\[ G \times \text{Transpose}(H); \]
\[ \begin{array}{cccc}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{array} \]

Syndromes are handled as follows:

\[ c := \text{Random}(C); \]
\[ \text{Syndrome}(c, C); \]
\[ (0 \ 0 \ 0) \]
\[ V := \text{AmbientSpace}(C); \]
\[ e := V \times [1, 0, 0, 0, 0, 0, 0]; \]
\[ r := c + e; \]
\[ \text{Syndrome}(r, C); \]
\[ (1 \ 0 \ 0) \]

Here we have taken a random codeword of \( C \) and computed its syndrome. Now, \( V \) is the space where \( C \) is defined, so the error vector \( e \) sits there, which is indicated by the prefix \( V! \).

**Example 12.5.4** Let us start again with the binary Hamming code and see how dual codes are constructed in GAP and Magma. In GAP we have

\[ C := \text{HammingCode}(3, \text{GF}(2)); \]
\[ CS := \text{DualCode}(C); \]

\[ a \text{ linear } [7, 3, 4]_{2} \text{ dual code} \]
\[ G := \text{GeneratorMat}(C); \]
\[ H := \text{GeneratorMat}(CS); \]
\[ \text{Display}(G \times \text{TransposedMat}(H)); \]

\[ \ldots \]

\[ \ldots \]
The same can be done in Magma. Moreover, we can make sure that the dual of the Hamming code is the predefined simplex code:

```plaintext
> C:=HammingCode(GF(2),3);
> CS:=Dual(C);
> G:=GeneratorMatrix(CS);
> S:=SimplexCode(3);
> H:=ParityCheckMatrix(S);
> G*Transpose(H);
[0 0 0 0]
[0 0 0 0]
[0 0 0 0]
```

**Example 12.5.5** Let us work out some examples in GAP and Magma that illustrate the notions of permutation equivalency and permutation automorphism group. As a model example we take as usual the binary Hamming code. Next we show how equivalency can be checked in GAP/GUAVA:

```plaintext
> C:=HammingCode(3,GF(2));;
> p:=(1,2,3)(4,5,6,7);
> CP:=PermutedCode(C,p);
a linear [7,4,3]1 permuted code
> IsEquivalent(C,CP);
true
```

So codes $C$ and $CP$ are equivalent. We may compute the permutation that brings $C$ to $CP$:

```plaintext
> CodeIsomorphism( C, CP );
(4,5)
```

Interestingly, we obtain that $CP$ can be obtained from $C$ just by $(4,5)$. Let us check if this is indeed true:

```plaintext
> CP2:=PermutedCode(C,(4,5));;
> Display(GeneratorMat(CP)*TransposedMat(CheckMat(CP2)));
```

So indeed the codes $CP$ and $CP2$ are the same. The permutation automorphism group can be computed via:

```plaintext
> AG:=AutomorphismGroup(C);
Group([ (1,2)(5,6), (2,4)(3,5), (2,3)(4,6,5,7), (4,5)(6,7), (4,6)(5,7) ])
> Size(AG)
168
```

So the permutation automorphism group of $C$ has 5 generators and 168 elements.
In Magma there is no immediate way to define permuted codes. We still can compute a permutation automorphism group, which is called a permutation group there:

```plaintext
> C:=HammingCode(GF(2),3);
> PermutationGroup(C);
Permutation group acting on a set of cardinality 7
```
Order = 168 = 2^3 * 3 * 7
(3, 6)(5, 7)
(1, 3)(4, 5)
(2, 3)(4, 7)
(3, 7)(5, 6)

12.5.3 Code constructions and bounds

Example 12.5.6 In this example we go through the above constructions in GAP and Magma. As a model code we consider the [15,11,3] binary Hamming code.

> C:=HammingCode(4,GF(2));;
> CP:=PuncturedCode(C);
a linear [14,11,2]1 punctured code
> CP5:=PuncturedCode(C,[11,12,13,14,15]);
a linear [10,10,1]0 punctured code
So PuncturedCode(C) punctures C at the last position and there is also a possibility to give the positions explicitly. The same syntax is for the shortening construction.

> CS:=ShortenedCode(C);
a linear [14,10,3]2 shortened code
> CS5:=ShortenedCode(C,[11,12,13,14,15]);
a linear [10,6,3]2..3 shortened code
Next we extend a code and check the property described in Proposition 3.1.11.

> CE:=ExtendedCode(C);
a linear [16,11,4]2 extended code
> CEP:=PuncturedCode(CE,);
> C=CEP;
true
A code C can be extended i times via ExtendedCode(C,i). Next take the shortened code augment and lengthen it.

> CSA:=AugmentedCode(CS,);
> d:=MinimumDistance(CSA,);
> CSA;
a linear [14,11,2]1 code, augmented with 1 word(s)
> CSL:=LengthenedCode(CS,);
a linear [15,11,2]1..3 code, lengthened with 1 column(s)
By default the augmentation is done by the all-one vector. One can specify the vector v to augment with explicitly by AugmentedCode(C,v). One can also do extension in the lengthening construction i times by LengthenedCode(C,i).

Now we do the same operations in Magma.

> C:=HammingCode(GF(2),4);
> CP:=PunctureCode(C, 15);
> CPS:=PunctureCode(C, {11..15});
> CS:=ShortenCode(C, 15);
> CSS:=ShortenCode(C, {11..15});
> CE:=ExtendCode(C);
> CEP:=PunctureCode(CE,16);
> C eq CEP;
true
12.5. CODING WITH COMPUTER ALGEBRA

> CSA:=AugmentCode(CS);
> CSL:=LengthenCode(CS);
One can also expurgate a code as follows.
> CExp:=ExpurgateCode(C);
> CExp;
[15, 10, 4] Cyclic Linear Code over GF(2)
Generator matrix:
\[
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\
\end{bmatrix}
\]
We see that in fact the code \( C_{\text{Exp}} \) has more structure: it is cyclic, i.e. a cyclic shift of every codeword is again a codeword, cf. Chapter 7. One can also expurgate codewords from the given list \( L \) by \( \text{ExpurgateCode}(C,L) \). In GAP this is done via \( \text{ExpurgatedCode}(C,L) \).

Example 12.5.7 Let us demonstrate how direct product is constructed in GAP and Magma. We construct a direct product of the binary \([15,11,3]\) Hamming code with itself. In GAP we do

> C:=HammingCode(4,GF(2));;
> CProd:=DirectProductCode(C,C);
a linear \([225,121,9]15..97\) direct product code
In Magma:
> C:=HammingCode(GF(2),4);
> CProd:=DirectProduct(C,C);

Example 12.5.8 Now we go through some of the above constructions using GAP and Magma. As model codes for summands we take binary \([7,4,3]\) and \([15,11,3]\) Hamming codes. In GAP the direct sum and the \((u|u+v)\)-construction are implemented.

> C1:=HammingCode(3,GF(2));;
> C2:=HammingCode(4,GF(2));;
> C:=DirectSumCode(C1,C2);
a linear \([22,15,3]2\) direct sum code
In Magma along with the above commands, a command for the juxtaposition is defined. The syntax of the commands is as follows:

> C1:=HammingCode(GF(2),3);
> C2:=HammingCode(GF(2),4);
> C:=DirectSum(C1,C2);
> CJ:=Juxtaposition(C2,C2); // [30, 11, 6] Cyclic Linear Code over GF(2)
> CPL:=PlotkinSum(C1,C2);
Example 12.5.9 Let us construct a concatenated code in GAP and Magma. We concatenate a Hamming \([17,15,3]\) code over \(\mathbb{F}_{16}\) and the binary \([7,4,3]\) Hamming code. In GAP we do the following
\[
> O:=\text{[HammingCode}(2,\text{GF}(16))];;
> I:=\text{[HammingCode}(3,\text{GF}(2))];;
> C:=\text{BZCodeNC}(O,I);
\]
a linear \([119,60,9]\) Blokh Zyablov concatenated code

In Magma we proceed as below
\[
> O:=\text{HammingCode}(\text{GF}(16),2);
> I:=\text{HammingCode}(\text{GF}(2),3);
> C:=\text{ConcatenatedCode}(O,I);
\]

Example 12.5.10 Magma provides a way to construct an MDS code with parameters \([q+1,k,q-k+2]\) over \(\mathbb{F}_q\) given the prime power \(q\) and positive integer \(k\). Example follows
\[
> C:=\text{MDSCode}(\text{GF}(16),10); //[17, 10, 8] Cyclic Linear Code over GF(2^4)
\]

Example 12.5.11 GAP and Magma provide commands that give an opportunity to compute some lower and upper bounds in size and minimum distance of codes, as well as stored tables for best known codes. Let us take a look how this functionality is handled in GAP first. The command \(\text{UpperBoundSingleton}(n,d,q)\) gives an upper bound on size of codes of length \(n\), minimum distance \(d\) defined over \(\mathbb{F}_q\). This applies also to non-linear codes. E.g.:
\[
> \text{UpperBoundSingleton}(25,10,2);
65536
\]
In the same way one can compute the Hamming, Plotkin, and Griesmer bounds:
\[
> \text{UpperBoundHamming}(25,10,2);
2196
> \text{UpperBoundPlotkin}(25,10,2);
1280
> \text{UpperBoundGriesmer}(25,10,2);
512
\]
Note that GAP does not require \(qd > (q-1)n\) as in Theorem 3.2.29. If \(qd > (q-1)n\) is not the case, shortening is applied. One can compute an upper bound which is a result of several bounds implemented in GAP
\[
> \text{UpperBound}(25,10,2);
1280
\]
Since Griesmer bound is not in the list with which \(\text{UpperBound}\) works, we obtain larger value. Analogously one can compute lower bounds
\[
> \text{LowerBoundGilbertVarshamov}(25,10,2);
16
\]
Here \(16 = 2^4\) is the size of the binary code of length 25 with the minimum distance at least 10.

One can access built-in tables (although somewhat outdated) as follows:
\[
\text{Display(BoundsMinimumDistance}(50,25,\text{GF}(2)));
\]
\[
\text{rec(}
 n := 50,
\)
12.5. CODING WITH COMPUTER ALGEBRA

k := 25,
q := 2,
references := rec(
construction := false,
lowerBound := 10,
lowerBoundExplanation := [ "Lb(50,25)=10, by taking subcode of:", "Lb(50,27)=10, by extending:", "Lb(49,27)=9, reference: EB3" ],
upperBound := 12,
upperBoundExplanation := [ "Ub(50,25)=12, by a one-step Griesmer \ bound from:", "Ub(37,24)=6, by considering shortening to:", "Ub(28,15)=6, otherwise extending would contradict:", "Ub(29,15)=7,reference: Ja" ]
)

In Magma one can compute the bounds in the following way
> GriesmerBound(GF(2),25,10):
> PlotkinBound(GF(2),25,10);
>> PlotkinBound(GF(2),25,10);

Runtime error in 'PlotkinBound': Require n <= 2*d for even weight binary case
> PlotkinBound(GF(2),100,51);
34
> SingletonBound(GF(2),25,10):
> SpherePackingBound(GF(2),25,10):
> GilbertVarshamovBound(GF(2),25,10);
9
> GilbertVarshamovLinearBound(GF(2),25,10);
16

Note that the result on the Plotkin bound is different from the one computed by
GAP, since Magma implements an improved bound treated in Remark 3.2.32.
The colon at the end of line suppresses the output. Access to built-in database
for given \ n \ and \ d \ is done as follows:
> BDLCLowerBound(GF(2),50,10);
27
> BDLCUpperBound(GF(2),50,10);
29

The corresponding commands for given \ n, k \ and \ k, d \ start with prefixes BKLC
and BLLC respectively.

12.5.4 Weight enumerator

Example 12.5.12 This example illustrates some functionality available for
weight distribution computations in GAP and Magma. In GAP one can compute
the weight enumerator of a code as well as the weight enumerator for its
dual via the MacWilliams identity.
> C:=HammingCode(4,GF(2));;
> CodeWeightEnumerator(C);
One interesting feature available in GAP is drawing weight histograms. It works as follows:

```plaintext
> WeightDistribution(C);
[ 1, 0, 0, 35, 105, 168, 280, 435, 435, 280, 168, 105, 35, 0, 0, 1 ]
> WeightHistogram(C);

+--------+--+--+--+--+--+--+--+--+--+--------+--+
<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
</table>
In Magma the analogous functionality looks as follows:

```plaintext
> C:=HammingCode(GF(2),4);
> WeightEnumerator(C);
$.1^15 + 35*$.1^12*$.2^3 + 105*$.1^11*$.2^4 + 168*$.1^10*$.2^5 + 280*$.1^9*$.2^6 \+ 435*$.1^8*$.2^7 + 435*$.1^7*$.2^8 + 280*$.1^6*$.2^9 + 168*$.1^5*$.2^10 + \$
105*$.1^4*$.2^11 + 35*$.1^3*$.2^12 + $.2^15
> W:=WeightDistribution(C);
> MacWilliamsTransform(15,11,2,W);
[ <0, 1>, <8, 15> ]
```

So **WeightEnumerator(C)** actually returns the homogeneous weight enumerator with $.1$ and $.2$ as variables.
12.5. CODING WITH COMPUTER ALGEBRA

12.5.5 Codes and related structures

12.5.6 Complexity and decoding

Example 12.5.13 In GAP/GUAVA and Magma for general linear code the idea of Definition 2.4.10 (1) is employed. In GAP such a decoding goes as follows

```gap
> C:=RandomLinearCode(15,5,GF(2));;
> MinimumDistance(C);
5
> # can correct 2 errors
> c:="11101"*C; # encoding
> c in C;
true
> r:=c+Codeword("01000000100000");
> c1:=Decodeword(C,r);;
> c1 = c;
true
> m:=Decode(C,r); # obtain initial message word
[ 1 1 1 0 1 ]
```

One can also obtain the syndrome table that is a table of pairs coset leader / syndrome by `SyndromeTable(C)`.

The same idea is realized in Magma as follows.

```magma
> C:=RandomLinearCode(GF(2),15,5); // can be [15,5,5] code
> # can correct 2 errors
> c:=Random(C);
> e:=AmbientSpace(C) ! [0,1,0,0,0,0,0,1,0,0,0,0,0,0,0];
> r:=c+e;
> result,c1:=Decode(C,r);
> result; // does decoding succeed?
true
> c1 eq c;
true
```

There are more advanced decoding methods for general linear codes. More on that in Section 10.6.

12.5.7 Cyclic codes

Example 12.5.14 We have already constructed finite fields and worked with them in GAP and Magma. Let us take a look one more time at those notions and show some new. In GAP we handle finite fields as follows.

```gap
> G:=GF(2^5);
> a:=PrimitiveRoot(G);
Z(2^5)
> DefiningPolynomial(G);
x_1^5+x_1^2+Z(2)^0
> result,c1:=Decodeword(C,r);
> result; // does decoding succeed?
true
> c1 eq c;
true
```

Pretty much the same functionality is provided in Magma

```magma
> G:=GF(2^5);
```
398 CHAPTER 12. CODING THEORY WITH COMPUTER ALGEBRA PACKAGES

\[
\begin{align*}
\texttt{a:=PrimitiveElement(G);} \\
\texttt{DefiningPolynomial(G);} \\
\texttt{$.1^5+$.1^2+1} \\
\texttt{b:=G.1;} \\
\texttt{a eq b;} \\
\texttt{true} \\
\texttt{// define explicitly} \\
\texttt{P<x>:=PolynomialRing(GF(2));} \\
\texttt{p:=x^5+x^2+1;} \\
\texttt{F<z>:=ext<GF(2)|p>;} \\
\texttt{F;} \\
\texttt{Finite field of size 2^5}
\end{align*}
\]

Example 12.5.15 Minimal polynomials are computed in GAP as follows:
\[
\begin{align*}
\texttt{a:=PrimitiveUnityRoot(2,17);} \\
\texttt{MinimalPolynomial(GF(2),a);} \\
\texttt{x_1^8+x_1^7+x_1^6+x_1^4+x_1^2+x_1+Z(2)^0} \\
\texttt{In Magma it is done analogous} \\
\texttt{a:=RootOfUnity(17,GF(2));} \\
\texttt{MinimalPolynomial(a,GF(2));} \\
\texttt{x^8 + x^7 + x^6 + x^4 + x^2 + x + 1}
\end{align*}
\]

Example 12.5.16 Some example on how to compute the cyclotomic polynomial in GAP and Magma follow. In GAP
\[
\begin{align*}
\texttt{CyclotomicPolynomial(GF(2),10);} \\
\texttt{x_1^4+x_1^3+x_1^2+x_1+Z(2)^0} \\
\texttt{In Magma it is done as follows} \\
\texttt{CyclotomicPolynomial(10);} \\
\texttt{$.1^4 - $.1^3 + $.1^2 - $.1 + 1} \\
\texttt{Note that in Magma the cyclotomic polynomial is always defined over \( \mathbb{Q} \).}
\end{align*}
\]

Example 12.5.17 Let us construct cyclic codes via roots in GAP and Magma. In GAP/GUAVA we proceed as follows.
\[
\begin{align*}
\texttt{C:=GeneratorPolCode(h,17,GF(2));} & \quad \# h \text{ is from Example 6.1.41} \\
\texttt{CR:=RootsCode(17,[1],2);} & \\
\texttt{MinimumDistance(CR);} & \\
\texttt{CR;} \\
\texttt{a cyclic [17,9,5]3..4 code defined by roots over GF(2)} \\
\texttt{C=CR;} & \\
\texttt{true} \\
\texttt{C2:=GeneratorPolCode(g,17,GF(2));} & \quad \# g \text{ is from Example 6.1.41} \\
\texttt{CR2:=RootsCode(17,[3],2);} & \\
\texttt{C2=CR2;} & \\
\texttt{true} \\
\texttt{So we generated first a cyclic code which generator polynomial has a (predefined) primitive root of unity as a root. Then we took the first element, which is not in the cyclotomic class of 1 and that is 3. We constructed a cyclic code with a primitive root of unity cubed as a root of the generator polynomial. Note that these results are in accordance with Example 12.5.15. We can also compute the number of all cyclic codes of the given length, as e.g.} \\
\texttt{NrCyclicCodes(17,GF(2));}
\end{align*}
\]
8

In Magma we do the construction as follows:

```magma
> a:=RootOfUnity(17,GF(2));
> C:=CyclicCode(17,[a],GF(2));
```

**Example 12.5.18** We can compute the Mattson-Solomon transform in Magma. This is done as follows:

```magma
> F<x> := PolynomialRing(SplittingField(x^17-1));
> f:=x^15+x^3+x;
> A:=MattsonSolomonTransform(f,17);
> A;
$.1^{216}*x^{16} + $.1^{177}*x^{15} + $.1^{214}*x^{14} + $.1^{99}*x^{13} + \$
$.1^{181}*x^{12} + $.1^{173}*x^{11} + $.1^{182}*x^{10} + $.1^{198}*x^{9} + \$
$.1^{108}*x^{8} + $.1^{107}*x^{7} + $.1^{218}*x^{6} + $.1^{91}*x^{5} + $.1^{54}*x^{4}\$
 + $.1^{109}*x^{3} + $.1^{27}*x^{2} + $.1^{141}*x + 1
> InverseMattsonSolomonTransform(A,17) eq f;
true
```

So for the construction we need a field that contains a primitive n-th root of unity. We can also compute the inverse transform.

### 12.5.8 Polynomial codes

**Example 12.5.19** Now we describe constructions of Reed-Solomon codes in GAP/ GUAVA and Magma. In GAP we proceed as follows:

```magma
> C:=ReedSolomonCode(31,5);
a cyclic [31,27,5]3..4 Reed-Solomon code over GF(32)
```

A construction of the extended code is somewhat different from the one defined in Definition 8.1.6. GUAVA first constructs by $\text{ExtendedReedSolomonCode}(n,d)$ first $\text{ReedSolomonCode}(n-1,d-1)$ and then extends it. The code is defined over $GF(n)$, so n should be a prime power.

```magma
> CE:=ExtendedReedSolomonCode(31,5);
a linear [31,27,5]3..4 extended Reed Solomon code over GF(31)
The generalized Reed-Solomon codes are handled as follows.

```magma
> R:=PolynomialRing(GF(2^5));;
> a:=Z(2^5);
> L:=List([1,2,3,6,7,10,12,16,20,24,25,29],i->Z(2^5)\^i);
> CG:=GeneralizedReedSolomonCode(L,4,R);;
```

So we define the polynomial ring R and the list of points L. Note that such a construction corresponds to the construction from Definition 8.1.10 with $b = 1$.

In Magma we proceed as follows:

```magma
> C:=ReedSolomonCode(31,5);
a:=PrimitiveElement(GF(2^5));
> A:=[a\^i:i in [1,2,3,6,7,10,12,16,20,24,25,29]];
> B:=[a\^i:i in [1,2,1,2,1,2,1,2,1,2,1,2]];
> CG:=GRSCode(A,B,4);
```

So Magma give an opportunity to construct the generalized Reed-Solomon codes with arbitrary $b$ which entries are non-zero.

**Example 12.5.20** In Magma one can compute subfield subcodes. This is done as follows:
> a:=RootOfUnity(17,GF(2));
> C:=CyclicCode(17,[a],GF(2^8)); // splitting field size 2^8
> CSS:=SubfieldSubcode(C);
> C2:=CyclicCode(17,[a],GF(2));
> C2 eq CSS;
true
> CSS_4:=SubfieldSubcode(C,GF(4)); // [17, 13, 4] code over GF(2^2)
By default the prime subfield is taken for the construction.

Example 12.5.21 *** GUAVA slow!!! ***
In Magma we can compute a trace code as is shown below:
> C:=HammingCode(GF(16),3);
> CT:=Trace(C);
> CT:Minimal;
[273, 272] Linear Code over GF(2)
We can also specify a subfield to restrict to by giving it as a second parameter
in Trace.

Example 12.5.22 In GAP/GUAVA in order to construct an alternant code
we proceed as follows
> a:=Z(2^5);
> P:=List([1,2,3,6,7,10,12,16,20,24,25,29],i->a^i);
> B:=List([1,2,1,2,1,2,1,2,1,2,1,2],i->a^i);
> CA:=AlternantCode(2,B,P,GF(2));
a linear [12,5,3..4]3..6 alternant code over GF(2)
By providing an extension field as the last parameter in AlternantCode, one
constructs an extension code (as per Definition 8.2.1) of the one defined by the
base field (in our example it is GF(2)), rather than the restriction-construction
as in Definition 8.3.1.
In Magma one proceeds as follows.
> a:=PrimitiveElement(GF(2^5));
> A:=[a^i:i in [1,2,3,6,7,10,12,16,20,24,25,29]]; 
> B:=[a^i:i in [1,2,1,2,1,2,1,2,1,2,1,2]]; 
> CA:=AlternantCode(A,B,2);
> CG:=GRSCode(A,B,2);
> CGS:=SubfieldSubcode(Dual(CG));
> CA eq CGS;
true
Here one can add a desired subfield for the restriction as in Definition 8.3.1 via
giving it as another parameter at the end of the parameter list for AlternantCode.

Example 12.5.23 In GAP/GUAVA one can construct a Goppa code as fol-
low.
> x:=Indeterminate(GF(2),"x");
> g:=x^3+x+1;
> C:=GoppaCode(g,15);
a linear [15,3,7]6..8 classical Goppa code over GF(2)
So the Goppa code C is constructed over the field, where the polynomial g is
defined. There is also a possibility to provide the list of non-roots L explicitly
via GoppaCode(g,L).
In Magma one needs to provide the list L explicitly.
\[ P<x>:=\text{PolynomialRing}(\text{GF}(2^5)); \]
\[ G:=x^3+x+1; \]
\[ a:=\text{PrimitiveElement}(\text{GF}(2^5)); \]
\[ L:=\{a^i : i \in [0..30]\}; \]
\[ C:=\text{GoppaCode}(L,G); \]
\[ C:=\text{Minimal}; \]
\[ [31, 16, 7] \text{ "Goppa code (r = 3)" Linear Code over GF(2)} \]
The polynomial \( G \) should be defined in the polynomial ring over the extension. The command \( C:=\text{Minimal} \) only displays the description for \( C \), no generator matrix is displayed.

Example 12.5.24 Now we show how binary Reed-Muller code can be constructed in GAP/GUAVA and also we check the property from the previous proposition.
\[ u:=5; \]
\[ m:=7; \]
\[ C:=\text{ReedMullerCode}(u,m); \]
\[ C2:=\text{ReedMullerCode}(m-u-1,m); \]
\[ CD:=\text{DualCode}(C); \]
\[ C = C2; \]
\[ true \]
In Magma one can do the above analogously:
\[ u:=5; \]
\[ m:=7; \]
\[ C:=\text{ReedMullerCode}(u,m); \]
\[ C2:=\text{ReedMullerCode}(m-u-1,m); \]
\[ CD:=\text{Dual}(C); \]
\[ CD eq C2; \]
\[ true \]

12.5.9 Algebraic decoding
Chapter 13

Bézout's theorem and codes on plane curves

Ruud Pellikaan

In this section affine and projective plane curves are defined. Bézout’s theorem on the number of points in the intersection of two plane curves is proved. A class of codes from plane curves is introduced and the parameters of these codes are determined. Divisors and rational functions on plane curve will be discussed.

13.1 Affine and projective space

lines planes quadrics coordinate transformations pictures

13.2 Plane curves

Let $\mathbb{F}$ be a field and $\overline{\mathbb{F}}$ its algebraic closure. By an affine plane curve over $\mathbb{F}$ we mean the set of points $(x, y) \in \mathbb{F}^2$ such that $F(x, y) = 0$, where $F \in \mathbb{F}[X,Y]$. Here $F = 0$ is called the defining equation of the curve. The $\mathbb{F}$-rational points of the curve with defining equation $F = 0$ are the points $(x, y) \in \mathbb{F}^2$ such that $F(x, y) = 0$. The degree of the curve is the degree of $F$.

Two plane curves with defining equations $F = 0$ and $G = 0$ have a component in common with defining equation $H = 0$, if $F$ and $G$ have a nontrivial factor $H$ in common, that is $F = BH$ and $G = AH$ for some $A, B \in \mathbb{F}[X,Y]$, and the degree of $H$ is not zero.

A curve with defining equation $F = 0$, $F \in \mathbb{F}[X,Y]$, is called irreducible if $F$ is not divisible by any $G \in \mathbb{F}[X,Y]$ such that $0 < \deg(G) < \deg(F)$, and absolutely irreducible if $F$ is irreducible when considered as a polynomial in $\mathbb{F}[X,Y]$.

The partial derivative with respect to $X$ of a polynomial $F = \sum f_{ij}X^iY^j$ is defined by

$$F_X = \sum if_{ij}X^{i-1}Y^j.$$
A point \((x, y)\) on an affine curve with equation \(F = 0\) is singular if \(F_X(x, y) = F_Y(x, y) = 0\), where \(F_X\) and \(F_Y\) are the partial derivatives of \(F\) with respect to \(X\) and \(Y\), respectively. A regular point of a curve is a nonsingular point of the curve. A regular point \((x, y)\) on the curve has a well-defined tangent line to the curve with equation

\[ F_X(x, y)(X - x) + F_Y(x, y)(Y - y) = 0. \]

**Example 13.2.1** The curve with defining equation \(X^2 + Y^2 = 0\) can be considered over any field. The polynomial \(X^2 + Y^2\) is irreducible in \(F_3[\![X, Y]\!]\) but reducible in \(F_9[\![X, Y]\!]\) and \(F_5[\![X, Y]\!]\). The point \((0, 0)\) is an \(F\)-rational point of the curve over any field \(F\), and it is the only singular point of this curve if the characteristic of \(F\) is not two.

A projective plane curve of degree \(d\) with defining equation \(F = 0\) over \(F\) is the set of points \((x : y : z)\in \mathbb{P}^2(\bar{F})\) such that \(F(x, y, z) = 0\), where \(F\in \mathbb{F}[X, Y, Z]\) is a homogeneous polynomial of degree \(d\).

Let \(F = \sum f_{ij}X^iY^j \in \mathbb{F}[X, Y]\) be a polynomial of degree \(d\). The homogenization \(F^*\) of \(F\) is an element of \(\mathbb{F}[X, Y, Z]\) and is defined by

\[ F^* = \sum f_{ij}X^iY^jZ^{d-i-j}. \]

Then \(F^*(X, Y, Z) = Z^dF(X/Z, Y/Z)\). If \(F = 0\) defines an affine plane curve of degree \(d\), then \(F^* = 0\) is the equation of the corresponding projective curve. A point at infinity of the affine curve with equation \(F = 0\) is a point of the projective plane in the intersection of the line at infinity and the projective curve with equation \(F^* = 0\). So the points at infinity on the curve are all points \((x : y : 0)\in \mathbb{P}^2(\bar{F})\) such that \(F^*(x, y, 0) = 0\).

A projective plane curve is reducible, respectively absolutely irreducible, if its defining homogeneous polynomial is irreducible, respectively absolutely irreducible.

A point \((x : y : z)\) on a projective curve with equation \(F = 0\) is singular if \(F_X(x, y, z) = F_Y(x, y, z) = F_Z(x, y, z) = 0\), and regular otherwise. Through a regular point \((x : y : z)\) on the curve passes the tangent line with equation

\[ F_X(x, y, z)X + F_Y(x, y, z)Y + F_Z(x, y, z)Z = 0. \]

If \(F\in \mathbb{F}[X, Y, Z]\) is a homogeneous polynomial of degree \(d\), then Euler’s equation

\[ XF_X + YF_Y + ZF_Z = dF \]

holds. So the two definitions of the tangent line to a curve in the affine and projective plane are consistent with each other.

A curve is called regular or nonsingular if all its points are regular. In Corollary 13.3.13 it will be shown that a regular projective plane curve is absolutely irreducible.

**Remark 13.2.2** Let \(F\) be a polynomial in \(\mathbb{F}[X, Y]\) of degree \(d\). Suppose that \(\mathbb{F}\) has at least \(d + 1\) elements. Then there exists an affine change of coordinates
such that the coefficients of $U^d$ and $V^d$ in $F(U,V)$ are 1. This is seen as follows. The projective curve with the defining equation $F^* = 0$ intersects the line at infinity in at most $d$ points. Then there exist two $F$-rational points $P$ and $Q$ on the line at infinity and not on the curve. Choose a projective transformation of coordinates which transforms $P$ and $Q$ into $(1 : 0 : 0)$ and $(0 : 1 : 0)$, respectively. This change of coordinates leaves the line at infinity invariant and gives a polynomial $F(U,V)$ such that the coefficients of $U^d$ and $V^d$ are not zero. An affine transformation can now transform these coefficients into 1. If for instance $F = X^2Y + XY^2 \in F_4[X,Y]$ and $\alpha$ is a primitive element of $F_4$, then $X = U + \alpha V$ and $Y = \alpha U + V$ gives $F(U,V) = U^3 + V^3$. Similarly, for all polynomials $F, G \in F[X,Y]$ of degree $l$ and $m$ there exists an affine change of coordinates such that the coefficients of $V^m$ and $V^m$ in $F(U,V)$ and $G(U,V)$, respectively, are 1.

Example 13.2.3 The Fermat curve $F_m$ is a projective plane curve with defining equation

$$X^m + Y^m + Z^m = 0.$$ 

The partial derivatives of $X^m + Y^m + Z^m$ are $mX^{m-1}$, $mY^{m-1}$, and $mZ^{m-1}$. So considered as a curve over the finite field $F_q$, it is regular if $m$ is relatively prime to $q$.

Example 13.2.4 Suppose $q = r^2$. The Hermitian curve $H_r$ over $F_q$ is defined by the equation

$$U^{r+1} + V^{r+1} + 1 = 0.$$ 

The corresponding homogeneous equation is

$$U^{r+1} + V^{r+1} + W^{r+1} = 0.$$ 

Hence it has $r+1$ points at infinity and it is the Fermat curve $F_m$ over $F_q$ with $r = m - 1$. The conjugate of $a \in F_q$ over $F_r$ is obtained by $\bar{a} = a^r$. So the equation can also be written as

$$U\bar{U} + V\bar{V} + W\bar{W} = 0.$$ 

This looks like equating a Hermitian form over the complex numbers to zero and explains the terminology.

We will see in Section 3 that for certain constructions of codes on curves it is convenient to have exactly one point at infinity. We will give a transformation such that the new equation of the Hermitian curve has this property. Choose an element $b \in F_q$ such that $b^{r+1} = -1$. There are exactly $r+1$ of these, since $q = r^2$. Let $P = (1 : b : 0)$. Then $P$ is a point of the Hermitian curve. The tangent line at $P$ has equation $U + bV = 0$. Multiplying with $b$ gives the equation $V = bU$. Substituting $V = bU$ in the defining equation of the curve gives that $W^{r+1} = 0$. So $P$ is the only intersection point of the Hermitian curve and the tangent line at $P$. New homogeneous coordinates are chosen such that this tangent line becomes the line at infinity. Let $X_1 = W$, $Y_1 = U$ and $Z_1 = bU - V$. Then the curve has homogeneous equation

$$X_1^{r+1} = b^rY_1^rZ_1 + bY_1Z_1^r - Z_1^{r+1}.$$
in the coordinates $X_1$, $Y_1$ and $Z_1$. Choose an element $a \in \mathbb{F}_q$ such that $a^r + a = -1$. There are $r$ of these. Let $X = X_1$, $Y = bY_1 + aZ_1$ and $Z = Z_1$. Then the curve has homogeneous equation

$$X^{r+1} = Y^r Z + YZ^r$$

with respect to $X$, $Y$ and $Z$. Hence the Hermitian curve has affine equation

$$X^{r+1} = Y^r + Y$$

with respect to $X$ and $Y$. This last equation has $(0 : 1 : 0)$ as the only point at infinity.

To see that the number of affine $\mathbb{F}_q$-rational points is $r + (r+1)(r^2 - r) = r^3$ one argues as follows. The right side of the equation $X^{r+1} = Y^r + Y$ is the trace from $\mathbb{F}_q$ to $\mathbb{F}_r$. The first $r$ in the formula on the number of points corresponds to the elements of $\mathbb{F}_r$. These are exactly the elements of $\mathbb{F}_q$ with zero trace. The remaining term corresponds to the elements in $\mathbb{F}_q$ with a nonzero trace, since the equation $X^{r+1} = \beta$, $\beta \in \mathbb{F}_r^*$, has exactly $r+1$ solutions in $\mathbb{F}_q$.

**Example 13.2.5** The Klein curve has homogeneous equation

$$X^3 Y + Y^3 Z + Z^3 X = 0.$$

More generally we define the curve $\mathcal{K}_m$ by the equation

$$X^m Y + Y^m Z + Z^m X = 0.$$

Suppose that $m^2 - m + 1$ is relatively prime to $q$. The partial derivatives of the left side of the equation are $mX^{m-1} Y + Z^m$, $mY^{m-1} Z + X^m$ and $mZ^{m-1} X + Y^m$. Let $(x : y : z)$ be a singular point of the curve $\mathcal{K}_m$. If $m$ is divisible by the characteristic, then $x^m = y^m = z^m = 0$. So $x = y = z = 0$, a contradiction. If $m$ and $q$ are relatively prime, then $x^m y = -my^m z = m^2 z^m x$. So

$$(m^2 - m + 1)z^m x = x^m y + y^m z + z^m x = 0.$$ 

Therefore $z = 0$ or $x = 0$, since $m^2 - m + 1$ is relatively prime to the characteristic. But $z = 0$ implies $x^m = -my^m z = 0$. Furthermore $y^m = -mz^{m-1} x$. So $x = y = z = 0$, which is a contradiction. Similarly $x = 0$ leads to a contradiction. Hence $\mathcal{K}_m$ is nonsingular if $\gcd(m^2 - m + 1, q) = 1$.

### 13.3 Bézout’s theorem

The *principal theorem of algebra* says that a polynomial of degree $m$ in one variable with coefficients in a field has at most $m$ zeros. If the field is algebraically closed and if the zeros are counted with multiplicities, then the total number of zeros is equal to $m$. Bézout’s theorem is a generalization of the principal theorem of algebra from one to several variables. It can be stated and proved in any number of variables. But only the two variable case will be treated, that is to say the case of plane curves.

First we recall some wellknown notions from commutative algebra.
13. BÉZOUT’S THEOREM

Let $R$ be a commutative ring with a unit. An ideal $I$ in $R$ is called a prime ideal if $I \neq R$ and for all $f, g \in R$ if $fg \in I$, then $f \in I$ or $g \in I$.

Let $F$ be a field. Let $F$ be a polynomial in $\mathbb{F}[X, Y]$ which is not a constant, and let $I$ be the ideal in $\mathbb{F}[X, Y]$ generated by $F$. Then $I$ is a prime ideal if and only if $F$ is irreducible.

Let $R$ be a commutative ring with a unit. A nonzero element $f$ of $R$ is called a zero divisor if $fg = 0$ for some $g \in R$, $g \neq 0$. The ring $R$ is called an integral domain if it has no zero divisors.

Let $S$ be a commutative ring with a unit. Let $I$ be an ideal in $S$. The factor ring of $S$ modulo $I$ is denoted by $S/I$. Then $I$ is a prime ideal if and only if $S/I$ is an integral domain.

Let $R$ be an integral domain. Define the relation $\sim$ on the set of pairs $\{(f, g) \mid f, g \in R, \ g \neq 0\}$ by $(f_1, g_1) \sim (f_2, g_2)$ if and only if there exists an $h \in R$, $h \neq 0$ such that $f_1g_2h = g_1f_2h$. This is an equivalence relation. Its classes are called fractions. The class of $(f, g)$ is denoted by $f/g$ or $\frac{f}{g}$ and $f$ is called the numerator and $g$ the denominator. The field of fractions or quotient field of $R$ consists of all fractions $f/g$ where $f, g \in R$ and $g \neq 0$ and is denoted by $\mathbb{Q}(R)$. This indeed is a field with addition and multiplication defined by

$$\frac{f_1}{g_1} + \frac{f_2}{g_2} = \frac{f_1g_2 + f_2g_1}{g_1g_2} \quad \text{and} \quad \frac{f_1}{g_1} \cdot \frac{f_2}{g_2} = \frac{f_1f_2}{g_1g_2}.$$

Example 13.3.1 The quotient field of the integers $\mathbb{Z}$ is the rationals $\mathbb{Q}$. The quotient field of the ring of polynomials $F[X_1, \ldots, X_m]$ is called the field of rational functions (in $m$ variables) and is denoted by $F(X_1, \ldots, X_m)$.

Remark 13.3.2 If $R$ is a commutative ring with a unit, then matrices with entries in $R$ and the determinant of a square matrix can be defined as when $R$ is a field. The usual properties for matrix addition and multiplication hold. If moreover $R$ is an integral domain, then a square matrix $M$ of size $n$ has determinant zero if and only if there exists a nonzero $r \in R^n$ such that $rM = 0$. This is seen by considering the same statement over the quotient field $\mathbb{Q}(R)$, where it is true, and clearing denominators.

Furthermore we define an algebraic construction which is called the resultant of two polynomials that measures whether they have a factor in common.

Definition 13.3.3 Let $R$ be a commutative ring with a unit. Then $R[Y]$ is the ring of polynomials in one variable $Y$ with coefficients in $R$. Let $F$ and $G$ be two polynomials in $R[Y]$ of degree $l$ and $m$, respectively. Then $F = \sum_{i=0}^{l} f_i Y^i$ and $G = \sum_{j=0}^{m} g_j Y^j$, where $f_i, g_j \in R$ for all $i, j$. Define the Sylvester matrix $Sylv(F, G)$ of $F$ and $G$ as the square matrix of size $l + m$ by

$$Sylv(F, G) = \begin{pmatrix}
    f_0 & f_1 & \cdots & f_l & 0 & \cdots & 0 & 0 \\
    0 & f_0 & f_1 & \cdots & f_l & 0 & \cdots & 0 \\
    \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\
    0 & \cdots & 0 & f_0 & f_1 & \cdots & f_l & 0 \\
    0 & 0 & \cdots & 0 & f_0 & f_1 & \cdots & f_l \\
    g_0 & g_1 & \cdots & g_m & 0 & \cdots & 0 \\
    0 & g_0 & g_1 & \cdots & g_m & 0 & \cdots & 0 \\
    \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\
    0 & \cdots & 0 & g_0 & g_1 & \cdots & g_m 
\end{pmatrix}.$$
The first $m$ rows consist of the cyclic shifts of the first row
\[ f_0 \ f_1 \ \ldots \ f_t \ 0 \ \ldots \ 0 \]
and the last $l$ rows consist of the cyclic shifts of row $m+1$
\[ g_0 \ g_1 \ \ldots \ g_m \ 0 \ \ldots \ 0 \]

The determinant of $\text{Sylv}(F,G)$ is called the \textit{resultant} of $F$ and $G$ and is denoted by $\text{Res}(F,G)$.

\textbf{Proposition 13.3.4} If $R$ is an integral domain and $f$ and $g$ are elements of $R[Y]$, then $\text{Res}(F,G) = 0$ if and only if $F$ and $G$ have a nontrivial common factor.

\textbf{Proof.} If $F$ and $G$ have a nontrivial common factor, then $F = BH$ and $G = AH$ for some $A, B$ and $H$ in $R[Y]$, where $H$ has nonzero degree. So $AF = BG$ for some $A$ and $B$, where $\deg(A) < m = \deg(G)$ and $\deg(B) < l = \deg(F)$. Write $A = \sum a_i Y^i$, $F = \sum f_j Y^j$, $B = \sum b_r Y^r$ and $G = \sum g_s Y^s$. Rewrite the equation $AF - BG = 0$ as a system of equations
\[ \sum_{i+j=k} a_i f_j - \sum_{r+s=k} b_r g_s = 0 \quad \text{for} \quad k = 0, 1, \ldots, l + m - 1, \]
or as a matrix equation
\[ (a, -b) \text{Sylv}(F,G) = 0, \]
where $a = (a_0, a_1, \ldots, a_{m-1})$, and $b = (b_0, b_1, \ldots, b_{l-1})$. Hence the rows of the matrix $\text{Sylv}(F,G)$ are dependent in case $F$ and $G$ have a common factor and so its determinant is zero. Thus we have shown that if $F$ and $G$ have a nontrivial common factor, then $\text{Res}(F,G) = 0$. The converse is also true. This is proved by reversing the argument. 

\textbf{Corollary 13.3.5} If $F$ is an algebraically closed field and $F$ and $G$ are elements of $F[Y]$, then $\text{Res}(F,G) = 0$ if and only if $F$ and $G$ have a common zero in $F$.

After this introduction on the resultant, we are in a position to prove a weak form of Bézout’s theorem.

\textbf{Proposition 13.3.6} Two plane curves of degree $l$ and $m$ that do not have a component in common, intersect in at most $lm$ points.

\textbf{Proof.} A special case of Bézout is $m = 1$. A line, which is not a component of a curve of degree $l$, intersects this curve in at most $l$ points. Stated differently, suppose that $F$ is a polynomial in $X$ and $Y$ with coefficients in a field $F$ and has degree $l$, and $L$ is a nonconstant linear form. If $F$ and $L$ have more than $l$ common zeros, then $L$ divides $F$ in $F[X,Y]$. A more general special case is if $F$ is a product of linear terms. So if one of the curves is a union of lines and the other curve does not contain any of these lines as a component, then the number of points in the intersection is at most $lm$. This follows from the above special case. The third special case is: if $G = XY - 1$ and $F$ is arbitrary. Then
the first curve can be parameterized by \( X = T, Y = 1/T \); substituting this in \( F \) gives a polynomial in \( T \) and \( 1/T \) of degree at most \( l \), multiplying by \( T^i \) gives a polynomial of degree at most \( 2l \), and therefore the intersection of these two curves has at most \( 2l \) points. It is not possible to continue like this, that is to say by parameterizing the second curve by rational functions in \( T \): \( X = X(T) \) and \( Y = Y(T) \).

The proof of the general case uses elimination theory. Suppose that we have two equations in two variables of degree \( l \) and \( m \), respectively, and we eliminate one variable. Then we get a polynomial in one variable of degree at most \( lm \) having as zeros the first coordinates of common zeros of the two original polynomials. In geometric terms, we have two curves of degree \( l \) respectively \( m \) in the affine plane, and we project the points of the intersection to a line. If we can show that we get at most \( lm \) points on this line and we can choose the projection in such a way that no two points of the intersection project to one point on the line, then we are done.

We may assume that the field is algebraically closed, since by a common zero \((x, y)\) of \( F \) and \( G \), we mean a pair \((x, y)\) \( \in \mathbb{F}^2 \) such that \( F(x, y) = G(x, y) = 0 \). Let \( F \) and \( G \) be polynomials in the variables \( X \) and \( Y \) of degree \( l \) and \( m \), respectively, with coefficients in a field \( \mathbb{F} \), and which do not have a common factor in \( \mathbb{F}[X,Y] \). Then they do not have a nontrivial common factor in \( \mathbb{F}[Y] \), where \( R = \mathbb{F}[X] \), so \( \text{Res}(F,G) \) is not zero by Proposition 13.3.4. By Remark 13.2.2 we may assume that, after an affine change of coordinates, \( F \) and \( G \) are monic and have degree \( l \) and \( m \), respectively, as polynomials in \( Y \) with coefficients in \( \mathbb{F}[X] \). Hence \( F = \sum_{i=0}^{l} f_i Y^i \) and \( G = \sum_{j=0}^{m} g_j Y^j \), where \( f_i \) and \( g_j \) are elements of \( \mathbb{F}[X] \) of degree at most \( l - i \) and \( m - j \), respectively, and \( f_l = g_m = 1 \). The square matrix \( \text{Sylv}(F,G) \) of size \( l + m \) has entries in \( \mathbb{F}[X] \).

Taking the determinant gives the resultant \( \text{Res}(F,G) \) which is an element of \( R = \mathbb{F}[X] \), that is to say a polynomial in \( X \) with coefficients in \( \mathbb{F} \).

The degree is at most \( lm \). This can be seen by homogenizing \( F \) and \( G \). Then \( F^* = \sum_{i=1}^{l} f_i^* y^i \) where \( f_i^* \) is a homogeneous polynomial in \( X \) and \( Z \) of degree \( l - i \), and similarly for \( G^* \). The determinant \( D(X,Z) \) of the corresponding Sylvester matrix is homogeneous of degree \( lm \), since

\[
D(TX,TZ) = T^{lm} D(X,Z).
\]

This is seen by dividing the rows and columns of the matrix by appropriate powers of \( T \).

We claim that the zeros of the polynomial \( \text{Res}(F,G) \) are exactly the projection of all points in the intersection of the curves defined by the equations \( F = 0 \) and \( G = 0 \). Thus we claim that \( x \) is a zero of \( \text{Res}(F,G) \) if and only if there exists an element \( y \in \mathbb{F} \) such that \((x, y)\) is a common zero of \( F \) and \( G \).

Let \( F(x) \) and \( G(x) \) be the polynomials in \( \mathbb{F}[Y] \), which are obtained from \( F \) and \( G \) by substituting \( x \) for \( X \). The polynomials \( F(x) \) and \( G(x) \) have again degree \( l \) and \( m \) in \( Y \), since we assumed that \( F \) and \( G \) are monic polynomials in \( Y \) of degrees \( l \) and \( m \), respectively. Now

\[
\text{Res}(F(x),G(x)) = \text{Res}(F,G)(x),
\]
that is to say it does not make a difference if we substitute \( x \) for \( X \) first and take the resultant afterwards, or take the resultant first and make the substitution afterwards. The degree of \( F \) and \( G \) has not diminished after the substitution. Let \( (x, y) \) be a common zero of \( F \) and \( G \). Then \( y \) is a common zero of \( F(x) \) and \( G(x) \), so \( \text{Res}(F(x), G(x)) = 0 \) by Corollary 13.3.5, and therefore \( \text{Res}(F, G)(x) = 0 \).

For the proof of the converse statement, one reads the above proof backwards.

Now we know that \( \text{Res}(F, G) \) is not identically zero and has degree at most \( lm \), and therefore \( \text{Res}(F, G) \) has at most \( lm \) zeros. There is still a slight problem, it may happen that for a fixed zero \( x \) of \( \text{Res}(F, G) \) there exists more than one \( y \) such that \( (x, y) \) is a zero of \( F \) and \( G \). This occasionally does happen. We will show that after a suitable coordinate change this does not occur.

For every zero \( x \) of \( \text{Res}(F, G) \) there are at most \( \min\{l, m\} \) elements \( y \) such that \( (x, y) \) is a zero of \( F \) and \( G \). Therefore \( F \) and \( G \) have at most \( \min\{lm, l^2m \} \) zeros in common, hence the collection of lines, which are incident with two distinct points of these zeros, is finite. Hence we can find a point \( P \) that is not in the union of this finite collection of lines. Furthermore there exists a line \( L \) incident with \( P \) and not incident with any of the common zeros of \( F \) and \( G \). In fact almost every point \( P' \) and line \( L' \), incident with \( P' \), have the above mentioned properties. Choose homogeneous coordinates such that \( P = (0 : 1 : 0) \) and \( L \) is the line at infinity. If \( P_1 = (x, y_1) \) and \( P_2 = (x, y_2) \) are zeros of \( F \) and \( G \), then the line with equation \( X - xZ = 0 \) through the corresponding points \( (x : y_1 : 1) \) and \( (x : y_2 : 1) \) in the projective plane, contains also \( P \). This contradicts the choice made for \( P \). So for every zero \( x \) of \( \text{Res}(f, g) \) there exists exactly one \( y \) such that \( (x, y) \) is a zero of \( F \) and \( G \). Hence \( F \) and \( G \) have at most \( lm \) common zeros. This finishes the proof of the weak form of Bézout’s theorem.

There are several reasons why the number of points in the intersection could be less than \( lm \): \( \mathbb{F} \) may not be algebraically closed; points of the intersection may lie at infinity; and multiplicities may occur.

Take for instance \( F = X^2 - Y^2 + 1, G = Y \) and \( F = \mathbb{F}_3 \). Then the two points of the intersection lie in \( \mathbb{F}_3^2 \) and not in \( \mathbb{F}_3 \). Let \( H = Y - 1 \). Then the two lines defined by \( G \) and \( H \) have no intersection in the affine plane. The homogenized polynomials \( G^* = G \) and \( H^* = Y - Z \) define curves in the projective plane which have exactly \( (1 : 0 : 0) \) in their intersection. Finally the line with equation \( H = 0 \) is the tangent line to the conic defined by \( F \) at the point \( (0, 1) \), and this point has to be counted with multiplicity 2.

In order to define the multiplicity of a point of intersection we have to localize the ring of polynomials.

**Definition 13.3.7** Let \( P = (x, y) \in \mathbb{F}^2 \). Let \( \mathbb{F}[X, Y]_P \) be the subring of the field of fractions \( \mathbb{F}(X, Y) \) consisting of all fractions \( A/B \) such that \( A, B \in \mathbb{F}[X, Y] \) and \( B(P) \neq 0 \). The ring \( \mathbb{F}[X, Y]_P \) is called the localization of \( \mathbb{F}[X, Y] \) at \( P \).

We explain the use of localization for the definition of the multiplicity by analogy to the multiplicity of a zero of a polynomial in one variable. Let \( F = (X - a)^eG \), where \( a \in \mathbb{F}, F, G \in \mathbb{F}[X] \) and \( G(a) \neq 0 \). Then \( a \) is a zero of \( F \) with multiplicity \( e \). The dimension of \( \mathbb{F}[X]/(F) \) as a vector space over \( \mathbb{F} \) is equal to the degree...
of $F$. But the element $G$ is invertible in the localization $\mathbb{F}[X]_a$ of $\mathbb{F}[X]$ at $a$. So the ideal generated by $F$ in $\mathbb{F}[X]_a$ is equal to the ideal generated by $(X - a)^e$. Hence the dimension of $\mathbb{F}[X]_a/(F)$ over $\mathbb{F}$ is equal to $e$.

**Definition 13.3.8** Let $P$ be a point in the intersection of two affine curves $\mathcal{X}$ and $\mathcal{Y}$ defined by $F$ and $G$, respectively. The *intersection multiplicity* $I(P; \mathcal{X}, \mathcal{Y})$ of $P$ at $\mathcal{X}$ and $\mathcal{Y}$ is defined by

$$I(P; \mathcal{X}, \mathcal{Y}) = \dim \mathbb{F}[X,Y]_P/(F,G).$$

Without proof we state several properties of the intersection multiplicity.

After a projective change of coordinates it may be assumed that the point $P = (0,0)$ is the origin of the affine plane. There is a unique way to write $F$ as the sum of its homogeneous parts

$$F = F_d + F_{d+1} + \cdots + F_l,$$

where $F_i$ is homogeneous of degree $i$, and $F_d \neq 0$ and $F_l \neq 0$. The homogeneous polynomial $F_d$ defines a union of lines over $\overline{\mathbb{F}}$, which are called the tangent lines of $\mathcal{X}$ at $P$. The point $P$ is regular point if and only if $d = 1$. The tangent line to $\mathcal{X}$ at $P$ is defined by $F_1 = 0$ if $d = 1$. Similarly

$$G = G_e + G_{e+1} + \cdots + G_m.$$

If the tangent lines of $\mathcal{X}$ at $P$ are distinct from the tangent lines of $\mathcal{Y}$ at $P$, then the multiplicity of $P$ is equal to $de$. In particular, if $P$ is a regular point of both curves and the tangent lines are distinct, then $d = e = 1$ and the intersection multiplicity is $1$.

The Hermitian curve over $\mathbb{F}_q$, with $q = r^2$, has the property that every line in the projective plane with coefficients in $\mathbb{F}_q$ intersects the Hermitian curve in $r + 1$ distinct points or in exactly one point with multiplicity $r + 1$.

**Definition 13.3.9** A *cycle* is a formal sum $\sum m_P P$ of points of the projective plane $\mathbb{P}^2(\overline{\mathbb{F}})$ with integer coefficients $m_P$ such that for finitely many $P$ its coefficient $m_P$ is not zero. The *degree* of a cycle is defined by $\deg(\sum m_P P) = \sum m_P$. If the projective plane curves $\mathcal{X}$ and $\mathcal{Y}$ are defined by the equations $F = 0$ and $G = 0$, respectively, then the *intersection cycle* $\mathcal{X} \cdot \mathcal{Y}$ is defined by

$$\mathcal{X} \cdot \mathcal{Y} = \sum I(P; \mathcal{X}, \mathcal{Y}) P.$$

Proposition 13.3.6 implies that this indeed is a cycle, that is to say there are only finitely many points $P$ such that $I(P; \mathcal{X}, \mathcal{Y})$ is not zero.

**Example 13.3.10** Consider the curve $\mathcal{X}$ with homogeneous equation

$$X^a Y^c + Y^{b+c} Z^{a-b} + X^d Z^{a+c-d} = 0$$

with $d < b < a$. Let $\mathcal{L}$ be the line with equation $X = 0$. The intersection of $\mathcal{L}$ with $\mathcal{X}$ consists of the points $P = (0 : 0 : 1)$ and $Q = (0 : 1 : 0)$. The origin of the affine plane is mapped to $P$ under the mapping $(x,y) \mapsto (x : y : 1)$. The affine equation of the curve is

$$X^a Y^c + Y^{b+c} + X^d = 0.$$
The intersection multiplicity at \( P \) of \( \mathcal{X} \) and \( \mathcal{L} \) is equal to the dimension of \( \mathbb{F}[X,Y]_0/(X, X^a Y^c + Y^{b+c} + X^d) \), which is \( b + c \).

The origin of the affine plane is mapped to \( Q \) under the mapping \((x, z) \mapsto (x : 1 : z)\). The affine equation of the curve becomes now

\[
X^a + Z^{a-b} + X^d Z^{a+c-d} = 0.
\]

The intersection multiplicity at \( P \) of \( \mathcal{X} \) and \( \mathcal{L} \) is equal to the dimension of \( \mathbb{F}[X,Y]_0/(X, X^a + Z^{a-b} + X^d Z^{a+c-d}) \), which is \( a - b \). Therefore

\[
\mathcal{X} \cdot \mathcal{L} = (b + c)P + (a - b)Q.
\]

Let \( \mathcal{M} \) be the line with equation \( Y = 0 \). Let \( \mathcal{N} \) be the line with equation \( Z = 0 \). Let \( R = (1 : 0 : 0) \). One shows similarly that

\[
\mathcal{X} \cdot \mathcal{M} = dP + (a + c - d)R \quad \text{and} \quad \mathcal{X} \cdot \mathcal{N} = aQ + cR.
\]

We state now as a fact the following strong version of Bézout’s theorem.

**Theorem 13.3.11** If \( \mathcal{X} \) and \( \mathcal{Y} \) are projective plane curves of degrees \( l \) and \( m \), respectively, that do not have a component in common, then

\[
\deg(\mathcal{X} \cdot \mathcal{Y}) = lm.
\]

**Corollary 13.3.12** Two projective plane curves of positive degree have a point in common.

**Corollary 13.3.13** A regular projective plane curve is absolutely irreducible.

**Proof.** If \( F = GH \) is a factorization of \( F \) with factors of positive degree, we get

\[
F_X = G_X H + GH_X
\]

by the product or Leibniz rule for the partial derivative. So \( F_X \) is an element of the ideal generated by \( G \) and \( H \), and similarly for the other two partial derivatives. Hence the set of common zeros of \( F_X, F_Y, F_Z \) and \( F \) contains the set of common zeros of \( G \) and \( H \). The intersection of the curves with equations \( G = 0 \) and \( H = 0 \) is not empty, by Corollary 13.3.12 since \( G \) and \( H \) have positive degrees. Therefore the curve has a singular point.

**Remark 13.3.14** Notice that the assumption that the curve is a projective plane curve is essential. The equation \( X^2Y - X = 0 \) defines a regular affine plane curve, but is clearly reducible. However one gets immediately from Corollary 13.3.13 that if \( F = 0 \) is an affine plane curve and the homogenization \( F^* \) defines a regular projective curve, then \( F \) is absolutely irreducible. The affine curve with equation \( X^2Y - X = 0 \) has the points \((1 : 0 : 0)\) and \((0 : 1 : 0)\) at infinity, and \((0 : 1 : 0)\) is a singular point.
13.4 Codes on plane curves

Let $G$ be an irreducible element of $\mathbb{F}_q[X,Y]$ of degree $m$. Let $P_1, \ldots, P_n$ be $n$ distinct points in the affine plane over $\mathbb{F}_q$ which lie on the plane curve defined by the equation $G = 0$. So $G(P_j) = 0$ for all $j = 1, \ldots, n$. Consider the code

$$E(l) = \{ (F(P_1), \ldots, F(P_n)) \mid F \in \mathbb{F}_q[X,Y], \deg(F) \leq l \}.$$ 

Let $V_l$ be the vector space of all polynomials in two variables $X, Y$ and coefficients in $\mathbb{F}_q$, and of degree at most $l$. Let $\mathcal{P} = \{P_1, \ldots, P_n\}$. Consider the evaluation map $ev_P : \mathbb{F}_q[X,Y] \rightarrow \mathbb{F}_q^n$ defined by $ev_P(F) = (F(P_1), \ldots, F(P_n))$. Then this is a linear map that has $E(l)$ as image of $V_l$.

**Proposition 13.4.1** Let $k$ be the dimension and $d$ the minimum distance of the code $E(l)$. Suppose $lm < n$. Then

$$d \geq n - lm.$$ 

**Proof.** The monomials of the form $X^\alpha Y^\beta$ with $\alpha + \beta \leq l$ form a basis of $V_l$. Hence $V_l$ has dimension $\binom{l+2}{2}$.

Let $F \in V_l$. If $G$ is a factor of $F$, then the corresponding codeword $ev_P(F)$ is zero. Conversely, if $ev_P(F) = 0$, then the curves with equation $F = 0$ and $G = 0$ have degree $l' \leq l$ and $m$, respectively, and have the $n$ points $P_1, \ldots, P_n$ in their intersection. Bézout’s theorem and the assumption $lm < n$ imply that $F$ and $G$ have a common factor. But $G$ is irreducible. Therefore $F$ is divisible by $G$. So the kernel of the evaluation map, restricted to $V_l$, is equal to $GV_{l-m}$, which is zero if $l < m$. Hence $k = \binom{l+2}{2}$ if $l < m$, and

$$k = \binom{l+2}{2} - \binom{l-m+2}{2} = lm + 1 - \binom{m-1}{2}$$

if $l \geq m$.

The same argument with Bézout gives that a nonzero codeword has at most $lm$ zeros, and therefore has weight at least $n - lm$. This shows that $d \geq n - lm$. ◇

**Example 13.4.2** Conics, reducible and irreducible............................

**Remark 13.4.3** If $F_1, \ldots, F_k$ is a basis for $V_l$ modulo $GV_{l-m}$, then

$$(F_i(P_j) \mid 1 \leq i \leq k, 1 \leq j \leq n)$$

is a generator matrix of $E(l)$. So it is a parity check matrix for $C(l)$, the dual of $E(l)$. The minimum distance $d^\perp$ of $C(l)$ is equal to the minimal number of dependent columns of this matrix. Hence for all $t < d^\perp$ and every subset $Q$ of
P consisting of t distinct points $P_1, \ldots, P_t$, the corresponding $k \times t$ submatrix must have maximal rank $t$. Let $L_t = V_l / GV_{l-m}$. Then the evaluation map $ev_Q$ induces a surjective map from $L_t$ to $F^t_q$. The kernel is the space of all functions $F \in V_l$ which are zero at the points of $Q$ modulo $GV_{l-m}$, which we denote by $L_l(Q)$. So $\dim(L_l(Q)) = k - t$.

Conversely, the dimension of $L_l(Q)$ is at least $k - t$ for all $t$-subsets $Q$ of $\mathcal{P}$. But in order to get a bound for $d^{\perp}$, we have to know that $\dim(L_l(Q)) = k - t$ for all $t < d^{\perp}$. The theory developed so far is not sufficient to get such a bound. The theorem of Riemann-Roch in the theory of algebraic curves gives an answer to this question. See Section ??; Section ?? gives another, more elementary, solution to this problem.

Notice that the following inequality hold for the codes $E(l)$:

$$k + d \geq n + 1 - g,$$

where $g = (m - 1)(m - 2)/2$. In Section 7 we will see that $g$ is the (arithmetic) genus. In Sections 3-6 the role of $g$ will be played by the number of gaps of the (Weierstrass) semigroup of a point at infinity.

13.5 Conics, arcs and Segre

Proposition 13.5.1

$$m(3, q) = \begin{cases} q + 1 & \text{if } q \text{ is odd} \\ q + 2 & \text{if } q \text{ is even.} \end{cases}$$

Proof. We have seen that $m(3, q)$ is at least $q + 1$ for all $q$ in Example ???. If case $q$ is even, then $m(3, q)$ is least $q + 2$ by in Example 3.2.12. ***Segre***

13.6 Qubic plane curves

13.6.1 Elliptic curves

13.6.2 The addition law on elliptic curves

13.6.3 Number of rational points on an elliptic curve

Manin’s proof, Chahal

13.6.4 The discrete logarithm on elliptic curves

13.7 Quartic plane curves

13.7.1 Flexes and bitangents

13.7.2 The Klein quartic

13.8 Divisors

In the following, $\mathcal{X}$ is an irreducible smooth projective curve over an algebraically closed field $F$. 
Definition 13.8.1 A divisor is a formal sum $D = \sum_{P \in X} n_P P$, with $n_P \in \mathbb{Z}$ and $n_P = 0$ for all but a finite number of points $P$. The support of a divisor is the set of points with nonzero coefficient. A divisor $D$ is called effective if all coefficients $n_P$ are non-negative (notation $D \geq 0$). The degree $\deg(D)$ of the divisor $D$ is $\sum n_P$.

Definition 13.8.2 Let $X$ and $Y$ be projective plane curves defined by the equations $F = 0$ and $G = 0$, respectively, then the intersection divisor $X \cdot Y$ is defined by

$$X \cdot Y = \sum I(P; X, Y)P,$$

where $I(P; X, Y)$ is the intersection multiplicity of Definition 13.8.1.

Bézout’s theorem tells us that $X \cdot Y$ is indeed a divisor and that its degree is $lm$ if the degrees of $X$ and $Y$ are $l$ and $m$, respectively.

Let $v_P = \text{ord}_P$ be the discrete valuation defined for functions on $X$ in Definition 13.8.3.

Definition 13.8.3 If $f$ is a rational function on $X$, not identically 0, we define the divisor of $f$ to be

$$(f) = \sum_{P \in X} v_P(f)P.$$

So, in a sense, the divisor of $f$ is a bookkeeping device that tells us where the zeros and poles of $f$ are and what their multiplicities and orders are.

Theorem 13.8.4 The degree of a divisor of a rational function is 0.

Proof. Let $X$ be a projective curve of degree $l$. Let $f$ be a rational function on the curve $X$. Then $f$ is represented by a quotient $A/B$ of two homogeneous polynomials of the same degree, say $m$. Let $Y$ and $Z$ be the hypersurfaces defined by the equations $A = 0$ and $B = 0$, respectively. Then $v_P(f) = I(P; X, Y) - I(P; X, Z)$, since $f = a/b = (a/h^m)(b/h^m)^{-1}$, where $H$ is a homogeneous linear form representing $h$ such that $H(P) \neq 0$. Hence

$$(f) = X \cdot Y - X \cdot Z.$$

So $(f)$ is indeed a divisor and its degree is zero, since it is the difference of two intersection divisors of the same degree $lm$.

Example 13.8.5 Look at the curve of Example 13.8.1. We saw that $f = x/(y+z)$ has a pole of order 2 in $Q = (0 : 1 : 1)$. The line $L$ with equation $X = 0$ intersects the curve in three points, namely $P_1 = (0 : \alpha : 1)$, $P_2 = (0 : 1 + \alpha : 1)$ and $Q$. So $X \cdot L = P_1 + P_2 + Q$. The line $M$ with equation $Y = 0$ intersects the curve in three points, namely $P_3 = (1 : 0 : 1)$, $P_4 = (\alpha : 0 : 1)$ and $P_5 = (1 + \alpha : 0 : 1)$. So $X \cdot M = P_3 + P_4 + P_5$. The line $N$ with equation $Y + Z = 0$ intersects the curve only in $Q$. So $X \cdot N = 3Q$. Hence $(x/(y+z)) = P_1 + P_2 - 2Q$ and $(y/(y+z)) = P_3 + P_4 + P_5 - 3Q$.

In this example it is not necessary to compute the intersection multiplicities since they are a consequence of Bézout’s theorem.
Example 13.8.6 Let \( X \) be the Klein quartic with equation \( X^3Y + Y^3Z + Z^3X = 0 \) of Example 13.2.5. Let \( P_1 = (0 : 0 : 1) \), \( P_2 = (1 : 0 : 0) \) and \( Q = (0 : 1 : 0) \). Let \( L \) be the line with equation \( X = 0 \). Then \( L \) intersects \( X \) in the points \( P_1 \) and \( Q \). Since \( L \) is not tangent in \( Q \), we see that \( I(Q; X, L) = 1 \). So the intersection multiplicity of \( X \) and \( L \) in \( P_1 \) is 3, since the multiplicities add up to 4. Hence \( X \cdot L = 3P_1 + Q \). Similarly we get for the lines \( M \) and \( N \) with equations \( Y = 0 \) and \( Z = 0 \), respectively, \( X \cdot M = 3P_2 + P_1 \) and \( X \cdot N = 3Q + P_2 \). Therefore \( (x/z) = 3P_1 - P_2 - 2Q \) and \( (y/z) = P_1 + 2P_2 - 3Q \).

Definition 13.8.7 The divisor of a rational function is called a principal divisor. We shall call two divisors \( D \) and \( D' \) linearly equivalent if and only if \( D - D' \) is a principal divisor; notation \( D \equiv D' \).

This is indeed an equivalence relation.

Definition 13.8.8 Let \( D \) be a divisor on a curve \( X \). We define a vector space \( L(D) \) over \( \mathbb{F} \) by

\[
L(D) = \{ f \in \mathbb{F}(X)^* \mid (f) + D \geq 0 \} \cup \{0\}.
\]

The dimension of \( L(D) \) over \( \mathbb{F} \) is denoted by \( l(D) \).

Note that if \( D = \sum_{i=1}^{s} n_i P_i - \sum_{j=1}^{r} m_j Q_j \) with all \( n_i, m_j > 0 \), then \( L(D) \) consists of 0 and the functions in the function field that have zeros of multiplicity at least \( m_j \) at \( Q_j \) \( (1 \leq j \leq r) \) and that have no poles except possibly at the points \( P_i \), with order at most \( n_i \) \( (1 \leq i \leq r) \). We shall show that this vector space has finite dimension.

First we note that if \( D \equiv D' \) and \( g \) is a rational function with \( (g) = D - D' \), then the map \( f \mapsto fg \) shows that \( L(D) \) and \( L(D') \) are isomorphic.

Theorem 13.8.9

(i) \( l(D) = 0 \) if \( \deg(D) < 0 \),

(ii) \( l(D) \leq 1 + \deg(D) \).

Proof. (i) If \( \deg(D) < 0 \), then for any function \( f \in \mathbb{F}(X)^* \), we have \( \deg((f) + D) < 0 \), that is to say, \( f \notin L(D) \).

(ii) If \( f \) is not 0 and \( f \in L(D) \), then \( D' = D + (f) \) is an effective divisor for which \( L(D') \) has the same dimension as \( L(D) \) by our observation above. So without loss of generality \( D \) is effective, say \( D = \sum_{i=1}^{s} n_i P_i \), \( (n_i \geq 0 \text{ for } 1 \leq i \leq r) \). Again, assume that \( f \) is not 0 and \( f \in L(D) \). In the point \( P_i \), we map \( f \) onto the corresponding element of the \( n_i \)-dimensional vector space \( (t_i^{-n_i} \mathcal{O}_{P_i})/\mathcal{O}_{P_i} \), where \( t_i \) is a local parameter at \( P_i \). We thus obtain a mapping of \( f \) onto the direct sum of these vector spaces; (map the 0-function onto 0). This is a linear mapping. Suppose that \( f \) is in the kernel. This means that \( f \) does not have a pole in any of the points \( P_i \), that is to say, \( f \) is a constant function. It follows that

\[
l(D) \leq 1 + \sum_{i=1}^{s} n_i = 1 + \deg(D).
\]
13.9. DIFFERENTIALS ON A CURVE

Example 13.8.10 Look at the curve of Examples ?? and 13.8.5. We saw that $f = x/(y+z)$ and $g = y/(y+z)$ are regular outside $Q$ and have a pole of order 2 and 3, respectively, in $Q = (0 : 1 : 1)$. So the functions 1, $f$ and $g$ have mutually distinct pole orders and are elements of $L(3Q)$. Hence the dimension of $L(3Q)$ is at least 3. We will see in Example 13.10.3 that it is exactly 3.

13.9 Differentials on a curve

Let $X$ be an irreducible smooth curve with function field $\mathbb{F}(X)$.

Definition 13.9.1 Let $V$ be a vector space over $\mathbb{F}(X)$. An $\mathbb{F}$-linear map $D : \mathbb{F}(X) \rightarrow V$ is called a derivation if it satisfies the product rule

$$D(fg) = fD(g) + gD(f).$$

Example 13.9.2 Let $X$ be the projective line with function field $\mathbb{F}(X)$. Define $D(F) = \sum i a_i x^{i-1}$ for a polynomial $F = \sum a_i x^i \in \mathbb{F}[X]$ and extend this definition to quotients by

$$D\left(\frac{E}{G}\right) = \frac{GD(F) - FD(G)}{G^2}.$$

Then $D : \mathbb{F}(X) \rightarrow \mathbb{F}(X)$ is a derivation.

Definition 13.9.3 The set of all derivations $D : \mathbb{F}(X) \rightarrow V$ will be denoted by $\text{Der}(X, V)$. We denote $\text{Der}(X, V)$ by $\text{Der}(X)$ if $V = \mathbb{F}(X)$.

The sum of two derivations $D_1, D_2 \in \text{Der}(X, V)$ is defined by $(D_1 + D_2)(f) = D_1(f) + D_2(f)$. The product of $D \in \text{Der}(X, V)$ with $f \in \mathbb{F}(X)$ is defined by $(fD)(g) = fD(g)$. In this way $\text{Der}(X, V)$ becomes a vector space over $\mathbb{F}(X)$.

Theorem 13.9.4 Let $t$ be a local parameter at a point $P$. Then there exists a unique derivation $D_t : \mathbb{F}(X) \rightarrow \mathbb{F}(X)$ such that $D_t(t) = 1$. Furthermore $\text{Der}(X)$ is one dimensional over $\mathbb{F}(X)$ and $D_t$ is a basis element for every local parameter $t$.

Definition 13.9.5 A rational differential form or differential on $X$ is an $\mathbb{F}(X)$-linear map from $\text{Der}(X)$ to $\mathbb{F}(X)$. The set of all rational differential forms on $X$ is denoted by $\Omega(X)$.

Again $\Omega(X)$ becomes a vector space over $\mathbb{F}(X)$ in the obvious way. Consider the map

$$d : \mathbb{F}(X) \rightarrow \Omega(X),$$

where for $f \in \mathbb{F}(X)$ the differential $df : \text{Der}(X) \rightarrow \mathbb{F}(X)$ is defined by $df(D) = D(f)$ for all $D \in \text{Der}(X)$. Then $d$ is a derivation.

Theorem 13.9.6 The space $\Omega(X)$ has dimension 1 over $\mathbb{F}(X)$ and $dt$ is a basis for every point $P$ with local parameter $t$.

So for every point $P$ and local parameter $t_P$, a differential $\omega$ can be represented in a unique way as $\omega = f_P dt_P$, where $f_P$ is a rational function. The obvious definition for “the value “ of $\omega$ in $P$ by $\omega(P) = f_P(P)$ has no meaning, since it depends on the choice of $t_P$. Despite of this negative result it is possible to say whether $\omega$ has a pole or a zero at $P$ of a certain order.
Definition 13.9.7 Let \( \omega \) be a differential on \( X \). The order or valuation of \( \omega \) in \( P \) is defined by \( \text{ord}_P(\omega) = v_P(\omega) = v_P(f_P) \). The differential form \( \omega \) is called regular if it has no poles. The regular differentials on \( X \) form an \( \mathbb{F}[X] \)-module, which we denote by \( \Omega[X] \).

This definition does not depend on the choices made.

If \( X \) is an affine plane curve defined by the equation \( F = 0 \) with \( F \in \mathbb{F}[X,Y] \), then \( \Omega[X] \) is generated by \( dx \) and \( dy \) as an \( \mathbb{F}[X] \)-module with the relation \( f_x dx + f_y dy = 0 \).

Example 13.9.8 We again look at the curve \( X \) in \( \mathbb{P}^2 \) given by \( X^3 + Y^3 + Z^3 = 0 \) in characteristic unequal to three. We define the sets \( U_x \) by \( U_x = \{(x : y : z) \in X \mid y \neq 0, z \neq 0\} \) and similarly \( U_y \) and \( U_z \). Then \( U_x, U_y, \) and \( U_z \) cover \( X \) since there is no point on \( X \) where two coordinates are zero. It is easy to check that the three representations

\[
\omega = \left( \frac{y}{z} \right)^2 \frac{d}{d\left( \frac{y}{z} \right)} \quad \text{on} \quad U_x, \quad \eta = \left( \frac{x}{y} \right)^2 \frac{d}{d\left( \frac{x}{y} \right)} \quad \text{on} \quad U_y, \quad \zeta = \left( \frac{x}{y} \right)^2 \frac{d}{d\left( \frac{z}{x} \right)} \quad \text{on} \quad U_z
\]

define one differential on \( X \). For instance, to show that \( \eta \) and \( \zeta \) agree on \( U_y \cap U_z \) one takes the equation \( (x/z)^3 + (y/z)^3 + 1 = 0 \), differentiates, and applies the formula \( d(f^{-1}) = -f^{-2} df \) to \( f = z/x \).

The only regular functions on \( X \) are constants, so one cannot represent this differential as \( g df \) with \( f \) and \( g \) regular functions on \( X \).

Now the divisor of a differential is defined as for functions.

Definition 13.9.9 The divisor \( (\omega) \) of the differential \( \omega \) is defined by

\[
(\omega) = \sum_{P \in X} v_P(\omega)P.
\]

Of course, one must show that only finitely many coefficients in \( (\omega) \) are not zero.

Let \( \omega \) be a differential and \( W = (\omega) \). Then \( W \) is called a canonical divisor. If \( \omega' \) is another nonzero differential, then \( \omega' = f\omega \) for some rational function \( f \). So \( (\omega') = W' \equiv W \) and therefore the canonical divisors form one equivalence class. This class is also denoted by \( W \). Now consider the space \( \mathcal{L}(W) \). This space of rational functions can be mapped onto an isomorphic space of differential forms by \( f \mapsto f\omega \). By the definition of \( \mathcal{L}(W) \), the image of \( f \) under the mapping is a regular differential form, that is to say, \( \mathcal{L}(W) \) is isomorphic to \( \Omega[X] \).

Definition 13.9.10 Let \( X \) be a smooth projective curve over \( \mathbb{F} \). We define the genus \( g \) of \( X \) by \( g = l(W) \).

Example 13.9.11 Consider the differential \( dx \) on the projective line. Then \( dx \) is regular at all points \( P_a = (a : 1) \), since \( x - a \) is a local parameter in \( P_a \) and \( dx = d(x - a) \). Let \( Q = (1 : 0) \) be the point at infinity. Then \( t = 1/x \) is a local parameter in \( Q \) and \( dx = -t^{-2} dt \). So \( v_Q(dx) = -2 \). Hence \( (dx) = -2Q \) and \( l(-2Q) = 0 \). Therefore the projective line has genus zero.
The genus of a curve will play an important role in the following sections. For methods with which one can determine the genus of a curve, we must refer to textbooks on algebraic geometry. We mention one formula without proof, the so-called Plücker formula.

**Theorem 13.9.12** If $X$ is a nonsingular projective curve of degree $m$ in $\mathbb{P}^2$, then

$$g = \frac{1}{2}(m - 1)(m - 2).$$

**Example 13.9.13** The genus of a line and a nonsingular conic are zero by Theorem 13.9.12. In fact a curve of genus zero is isomorphic to the projective line. For example the curve $X$ with equation $XZ - Y^2 = 0$ of Example ?? is isomorphic to $\mathbb{P}^1$ where the isomorphism is given by $(x : y : z) \mapsto (x : y) = (y : z)$ for $(x : y : z) \in X$. The inverse map is given by $(u : v) \mapsto (u^2 : uv : v^2)$.

**Example 13.9.14** So the curve of Examples ??, 13.8.5 and 13.9.8 has genus $1$ and by the definition of genus, $L(W) = \mathbb{F}$, so regular differentials on $X$ are scalar multiples of the differential $\omega$ of Example 13.9.8.

For the construction of codes over algebraic curves that generalize Goppa codes, we shall need the concept of residue of a differential at a point $P$. This is defined in accordance with our treatment of local behavior of a differential $\omega$.

**Definition 13.9.15** Let $P$ be a point on $X$, $t$ a local parameter at $P$ and $\omega = f \, dt$ the representation of $\omega$. The function $f$ can be written as $\sum a_i t^i$. We define the residue $\text{Res}_P(\omega)$ of $\omega$ in the point $P$ to be $a_{-1}$.

One can show that this algebraic definition of the residue does not depend on the choice of the local parameter $t$.

One of the basic results in the theory of algebraic curves is known as the **residue theorem**. We only state the theorem.

**Theorem 13.9.16** If $\omega$ is a differential on a smooth projective curve $X$, then

$$\sum_{P \in X} \text{Res}_P(\omega) = 0.$$ 

### 13.10 The Riemann-Roch theorem

The following theorem, known as the **Riemann-Roch theorem** is not only a central result in algebraic geometry with applications in other areas, but it is also the key to the new results in coding theory.

**Theorem 13.10.1** Let $D$ be a divisor on a smooth projective curve of genus $g$. Then, for any canonical divisor $W$

$$l(D) - l(W - D) = \deg(D) - g + 1.$$ 

We do not give the proof. The theorem allows us to determine the degree of canonical divisors.
Corollary 13.10.2 For a canonical divisor \( W \), we have \( \deg(W) = 2g - 2 \).

Proof. Everywhere regular functions on a projective curve are constant, that is to say, \( \mathcal{L}(0) = \mathbb{F} \), so \( l(0) = 1 \). Substitute \( D = W \) in Theorem 13.10.1 and the result follows from Definition 13.9.10.

Example 13.10.3 It is now clear why in Example 13.8.10 the space \( \mathcal{L}(3Q) \) has dimension 3. By Example 13.9.14 the curve \( X \) has genus 1, the degree of \( W - 3Q \) is negative, so \( l(W - 3Q) = 0 \). By Theorem 13.10.1 we have \( l(3Q) = 3 \).

At first, Theorem 13.10.1 does not look too useful. However, Corollary 13.10.2 provides us with a means to use it successfully.

Corollary 13.10.4 Let \( D \) be a divisor on a smooth projective curve of genus \( g \) and let \( \deg(D) > 2g - 2 \). Then

\[
l(D) = \deg(D) - g + 1.
\]

Proof. By Corollary 13.10.2, \( \deg(W - D) < 0 \), so by Theorem 13.8.9(i), \( l(W - D) = 0 \).

Example 13.10.5 Consider the code of Theorem ???. We embed the affine plane in a projective plane and consider the rational functions on the curve defined by \( G \). By Bézout’s theorem, this curve intersects the line at infinity, that is to say, the line defined by \( Z = 0 \), in \( m \) points. These are the possible poles of our rational functions, each with order at most \( l \). So, in the terminology of Definition 13.8.8, we have a space of rational functions, defined by a divisor \( D \) of degree \( lm \). Then Corollary 13.10.4 and Theorem ?? imply that the curve defined by \( G \) has genus at most equal to \( \binom{m}{2} \). This is exactly what we find from the Plücker formula 13.9.12.

Let \( m \) be a non-negative integer. Then \( l(mp) \leq l((m - 1)p) + 1 \), by the argument as in the proof of Theorem 13.8.9.

Definition 13.10.6 If \( l(mp) = l((m - 1)p) \), then \( m \) is called a (Weierstrass) gap of \( P \). A non-negative integer that is not a gap is called a nongap of \( P \).

The number of gaps of \( P \) is equal to the genus \( g \) of the curve, since \( l(iP) = i + 1 - g \) if \( i > 2g - 2 \), by Corollary 13.10.4 and

\[
1 = l(0) \leq l(P) \leq \cdots \leq l((2g - 1)p) = g.
\]

If \( m \in \mathbb{N}_0 \), then \( m \) is a nongap of \( P \) if and only if there exists a rational function which has a pole of order \( m \) in \( P \) and no other poles. Hence, if \( m_1 \) and \( m_2 \) are nongaps of \( P \), then \( m_1 + m_2 \) is also a nongap of \( P \). The nongaps form the Weierstrass semigroup in \( \mathbb{N}_0 \). Let \( \rho_i | i \in \mathbb{N} \) be an enumeration of all the nongaps of \( P \) in increasing order, so \( \rho_1 = 0 \). Let \( f_i \in \mathcal{L}(\rho_i P) \) be such that \( v_p(f_i) = -\rho_i \) for \( i \in \mathbb{N} \). Then \( f_1, \ldots, f_i \) provide a basis for the space \( \mathcal{L}(\rho_i P) \). This will be the approach of Sections 3-7.

The term \( l(W - D) \) in Theorem 13.10.1 can be interpreted in terms of differentials. We introduce a generalization of Definition 13.8.8 for differentials.
13.11. CODES FROM ALGEBRAIC CURVES

**Definition 13.10.7** Let $D$ be a divisor on a curve $X$. We define

$$
\Omega(D) = \{\omega \in \Omega(X) \mid (\omega) - D \geq 0\}
$$

and we denote the dimension of $\Omega(D)$ over $\mathbb{F}$ by $\delta(D)$, called the *index of speciality* of $D$.

The connection with functions is established by the following theorem.

**Theorem 13.10.8** $\delta(D) = l(W - D)$.

**Proof.** If $W = (\omega)$, we define a linear map $\phi: L(W - D) \to \Omega(D)$ by $\phi(f) = f\omega$. This is clearly an isomorphism. \hfill \Box

**Example 13.10.9** If we take $D = 0$, then by Definition 13.9.10 there are exactly $g$ linearly independent regular differentials on a curve $X$. So the differential of Example 13.9.8 is the only regular differential on $X$ (up to a constant factor) as was already observed after Theorem 13.9.12.

13.11 Codes from algebraic curves

We now come to the applications to coding theory. Our alphabet will be $\mathbb{F}_q$. Let $\mathbb{F}$ be the algebraic closure of $\mathbb{F}_q$. We shall apply the theorems of the previous sections. A few adaptations are necessary, since for example, we consider for functions in the coordinate ring only those that have coefficients in $\mathbb{F}_q$. If the affine curve $X$ over $\mathbb{F}_q$ is defined by a prime ideal $I$ in $\mathbb{F}_q[X_1, \ldots, X_n]$, then its coordinate ring $\mathbb{F}_q[X]$ is by definition equal to $\mathbb{F}_q[X_1, \ldots, X_n]/I$ and its function field $\mathbb{F}_q(X)$ is the quotient field of $\mathbb{F}_q[X]$. It is always assumed that the curve is *absolutely irreducible*, this means that the defining ideal is also prime in $\mathbb{F}[X_1, \ldots, X_n]$. Similar adaptations are made for projective curves. Notice that $F(x_1, \ldots, x_n) = F(x_1^q, \ldots, x_n^q)$ for all $F \in \mathbb{F}_q[X_1, \ldots, X_n]$. So if $(x_1, \ldots, x_n)$ is a zero of $F$ and $F$ is defined over $\mathbb{F}_q$, then $(x_1^q, \ldots, x_n^q)$ is also a zero of $F$. Let $F_r: \mathbb{F} \to \mathbb{F}$ be the *Frobenius map* defined by $F_r(x) = x^q$. We can extend this map coordinatewise to points in affine and projective space. If $X$ is a curve defined over $\mathbb{F}_q$ and $P$ is a point of $X$, then $F_r(P)$ is also a point of $X$, by the above remark. A divisor $D$ on $X$ is called rational if the coefficients of $P$ and $F_r(P)$ in $D$ are the same for any point $P$ of $X$. The space $L(D)$ will only be considered for rational divisors and is defined as before but with the restriction of the rational functions to $\mathbb{F}_q(X)$. With these changes the stated theorems remain true over $\mathbb{F}_q$ in particular the theorem of Riemann-Roch 13.10.1.

Let $X$ be an absolutely irreducible nonsingular projective curve over $\mathbb{F}_q$. We shall define two kinds of algebraic geometry codes from $X$. The first kind generalizes Reed-Solomon codes, the second kind generalizes Goppa codes. In the following, $P_1, P_2, \ldots, P_n$ are rational points on $X$ and $D$ is the divisor $P_1 + P_2 + \cdots + P_n$. Furthermore $G$ is some other divisor that has support disjoint from $D$. Although it is not necessary to do so, we shall make more restrictions on $G$, namely

$$
2g - 2 < \deg(G) < n.
$$
Definition 13.11.1 The linear code $C(D, G)$ of length $n$ over $\mathbb{F}_q$ is the image of the linear map $\alpha : \mathcal{L}(G) \rightarrow \mathbb{F}_q^n$ defined by $\alpha(f) = (f(P_1), f(P_2), \ldots, f(P_n))$. Codes of this kind are called geometric Reed-Solomon codes.

Theorem 13.11.2 The code $C(D, G)$ has dimension $k = \deg(G) - g + 1$ and minimum distance $d \geq n - \deg(G)$.

Proof. (i) If $f$ belongs to the kernel of $\alpha$, then $f \in \mathcal{L}(G - D)$ and by Theorem 13.8.9(ii), this implies $f = 0$. The result follows from the assumption $2g - 2 < \deg(G) < n$ and Corollary 13.10.4. 
(ii) If $\alpha(f)$ has weight $d$, then there are $n - d$ points $P_i$, say $P_{i_1}, P_{i_2}, \ldots, P_{i_{n-d}}$, for which $f(P_i) = 0$. Therefore $f \in \mathcal{L}(G - E)$, where $E = P_{i_1} + \cdots + P_{i_{n-d}}$. Hence $\deg(G) - n - d \geq 0$.

Note the analogy with the proof of Theorem ??.

Example 13.11.3 Let $X$ be the projective line over $\mathbb{F}_q$. Let $n = q^m - 1$. We define $P_0 = (0 : 1), P_\infty = (1 : 0)$ and we define the divisor $D$ as $\sum_{j=0}^{n} P_j$, where $P_j = (\beta^j : 1), (1 \leq j \leq n)$. We define $G = aP_0 + bP_\infty$, $a \geq 0$, $b \geq 0$. (Here $\beta$ is a primitive $n$th root of unity.) By Theorem 13.10.1, $\mathcal{L}(G)$ has dimension $a + b + 1$ and one immediately sees that the functions $(x/y)^i, -a \leq i \leq b$, form a basis of $\mathcal{L}(G)$. Consider the code $C(D, G)$. A generator matrix for this code has as rows $(\beta^i, \beta^{2i}, \ldots, \beta^{ni})$ with $-a \leq i \leq b$. One easily checks that $(c_1, c_2, \ldots, c_n)$ is a codeword in $C(D, G)$ if and only if $\sum_{j=1}^{n} c_j (\beta^j)^i = 0$ for all $l$ with $a < l < n - b$. It follows that $C(D, G)$ is a Reed-Solomon code. The subfield subcode with coordinates in $\mathbb{F}_q$ is a BCH code.

Example 13.11.4 Let $X$ be the curve of Examples ??, 13.8.5, 13.8.10 and 13.10.3. Let $G = 3Q$, where $Q = (0 : 1 : 1)$. We take $n = 8$, so $D$ is the sum of the remaining rational points. The coordinates are given by

<table>
<thead>
<tr>
<th></th>
<th>$Q$</th>
<th>$P_1$</th>
<th>$P_2$</th>
<th>$P_3$</th>
<th>$P_4$</th>
<th>$P_5$</th>
<th>$P_6$</th>
<th>$P_7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>$\alpha$</td>
<td>$\overline{\alpha}$</td>
<td>1</td>
<td>$\alpha$</td>
</tr>
<tr>
<td>$y$</td>
<td>1</td>
<td>$\alpha$</td>
<td>$\overline{\alpha}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$z$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

where $\overline{\alpha} = \alpha^2 = 1 + \alpha$. We saw in Examples 13.8.10 and 13.10.3 that 1, $x/(y+z)$ and $y/(y+z)$ are a basis of $(3Q)$ over $\mathbb{F}$ and hence also over $\mathbb{F}_4$. This leads to the following generator matrix for $C(D, G)$:

$$
\begin{pmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 0 & 0 & 1 & \alpha & \overline{\alpha} & 1 & \alpha & \overline{\alpha} \\
\overline{\alpha} & \alpha & 0 & 0 & 1 & 1 & 1
\end{pmatrix}
$$

By Theorem 13.11.2, the minimum distance is at least 5 and of course, one immediately sees from the generator matrix that $d = 5$.

We now come to the second class of algebraic geometry codes. We shall call these codes geometric Goppa codes.

Definition 13.11.5 The linear code $C^*(D, G)$ of length $n$ over $\mathbb{F}_q$ is the image of the linear map $\alpha^* : \Omega(G - D) \rightarrow \mathbb{F}_q^n$ defined by

$$
\alpha^*(\eta) = (\text{Res}_{P_1}(\eta), \text{Res}_{P_2}(\eta), \ldots, \text{Res}_{P_n}(\eta)).
$$

The parameters are given by the following theorem.
Theorem 13.11.6 The code \( C^*(D,G) \) has dimension \( k^* = n - \deg(G) + g - 1 \) and minimum distance \( d^* \geq \deg(G) - 2g + 2 \).

Proof. Just as in Theorem 13.11.2, these assertions are direct consequences of Theorem 13.10.1 (Riemann-Roch), using Theorem 13.10.8 (making the connection between the dimension of \( \Omega(G) \) and \( l(W - G) \)) and Corollary 13.10.2 (stating that the degree of a canonical divisor is \( 2g - 2 \)).

Example 13.11.7 Let \( L = \{\alpha_1, \ldots, \alpha_n\} \) be a set of \( n \) distinct elements of \( \mathbb{F}_{q^m} \). Let \( g \) be a polynomial in \( \mathbb{F}_{q^m}[X] \) which is not zero at \( \alpha_i \) for all \( i \). The (classical) Goppa code \( \Gamma(L, g) \) is defined by

\[
\Gamma(L, g) = \{ c \in \mathbb{F}_{q^n} | \sum c_i X - \alpha_i \equiv 0 \pmod{g} \}.
\]

Let \( P_i = (\alpha_i : 1) \), \( Q = (1 : 0) \) and \( D = P_1 + \cdots + P_n \). If we take for \( E \) the divisor of zeros of \( g \) on the projective line, then \( \Gamma(L, g) = C^*(D, E - Q) \) and

\[
c \in \Gamma(L, g) \text{ if and only if } \sum c_i X - \alpha_i \in \Omega(E - Q - D).
\]

This is the reason that some authors extend the definition of geometric Goppa codes to subfield subcodes of codes of the form \( C^*(D,G) \). It is a well-known fact that the parity check matrix of the Goppa code \( \Gamma(L, g) \) is equal to the following generator matrix of a generalized RS code

\[
\begin{pmatrix}
g(\alpha_1)^{-1} & \cdots & g(\alpha_n)^{-1} \\
g_1(\alpha_1)^{-1} & \cdots & g_1(\alpha_n)^{-1} \\
\vdots & \cdots & \vdots \\
g_r(\alpha_1)^{-1} & \cdots & g_r(\alpha_n)^{-1}
\end{pmatrix},
\]

where \( r \) is the degree of the Goppa polynomial \( g \). So \( \Gamma(L, g) \) is the subfield subcode of the dual of a generalized RS code. This is a special case of the following theorem.

Theorem 13.11.8 The codes \( C(D,G) \) and \( C^*(D,G) \) are dual codes.

Proof. From Theorem 13.11.2 and Theorem 13.11.6 we know that \( k + k^* = n \). So it suffices to take a word from each code and show that the inner product of the two words is 0. Let \( f \in \mathcal{L}(G) \), \( \eta \in \Omega(G - D) \). By Definitions 13.11.1 and 13.11.5, the differential \( f\eta \) has no poles except possibly poles of order 1 in the points \( P_1, P_2, \ldots, P_n \). The residue of \( f\eta \) in \( P_i \) is equal to \( f(P_i)\text{Res}_{P_i}(\eta) \). By Theorem 13.9.16, the sum of the residues of \( f\eta \) over all the poles, that is to say, over the points \( P_i \), is equal to zero. Hence we have

\[
0 = \sum_{i=1}^{n} f(P_i)\text{Res}_{P_i}(\eta) = \langle \alpha(f), \alpha^*(\eta) \rangle.
\]
Theorem 13.11.9 Let $X$ be a curve defined over $\mathbb{F}_q$. Let $P_1, \ldots, P_n$ be $n$ rational points on $X$. Let $D = P_1 + \cdots + P_n$. Then there exists a differential form $\omega$ with simple poles at the $P_i$ such that $\text{Res}_{P_i}(\omega) = 1$ for all $i$. Furthermore

$$C^*(D, G) = C(D, W + D - G)$$

for all divisors $G$ that have a support disjoint from the support of $D$, where $W$ is the divisor of $\omega$.

So one can do without differentials and the codes $C^*(D, G)$. However, it is useful to have both classes when treating decoding methods. These use parity checks, so one needs a generator matrix for the dual code.

In the next paragraph we treat several examples of algebraic geometry codes. It is already clear that we find some good codes. For example from Theorem 13.11.2 we see that such codes over a curve of genus 0 (the projective line) are MDS codes. In fact, Theorem 13.11.2 says that $d \geq n - k + 1 - g$, so if $g$ is small, we are close to the Singleton bound.

13.12 Rational functions and divisors on plane curves

This section will be finished together with the correction of Section 7.

Example 13.12.1 Consider the curve $X$ with homogeneous equation

$$X^aY^c + Y^{b+c}Z^{a-b} + X^dZ^{a+c-d} = 0$$

with $d < b < a$ as in Example 13.3.10. The divisor of the rational function $x/z$ is

$$\left( \frac{x}{z} \right) = (X \cdot \mathcal{L}) - (X \cdot \mathcal{N}) = (b + c)P - bQ - cR.$$ 

The divisor of the rational function $y/z$ is

$$\left( \frac{y}{z} \right) = (X \cdot \mathcal{M}) - (X \cdot \mathcal{N}) = dP - aQ - (a - d)R.$$ 

Hence the divisor of $(x/z)^\alpha(y/z)^\beta$ is

$$(b + c)\alpha + d\beta)P + (-b\alpha - a\beta)Q + (-c\alpha + (a - d)\beta)R.$$ 

It has only a pole at $Q$ if and only if $c\alpha \leq (a - d)\beta$. (This will serve as a motivation for the choice of the basis of $R$ in Proposition ??.)

13.13 Resolution or normalization of curves

13.14 Newton polygon of plane curves
13.15 Notes

Goppa submitted his seminal paper [?] in June 1975 and it was published in 1977. Goppa also published three more papers in the eighties [?, ?, ?] and a book [?] in 1991.

Most of this section is standard textbook material. See for instance [?, ?, ?, ?] to mention a few. Section 13.4 is a special case of Goppa’s construction and comes from [?]. The Hermitian curves in Example 13.2.4 and their codes have been studied by many authors. See [?, ?, ?, ?]. The Klein curve goes back to F. Klein [?] and has been studied thoroughly, also over finite fields in connection with codes. See [?, ?, ?, ?, ?, ?].
Chapter 14

Curves
14.1 Algebraic varieties

14.2 Curves

14.3 Curves and function fields

14.4 Normal rational curves and Segre’s problems

14.5 The number of rational points
   14.5.1 Zeta function
   14.5.2 Hasse-Weil bound
   14.5.3 Serre’s bound
   14.5.4 Ihara’s bound
   14.5.5 Drinfeld-Vlăduţ bound
   14.5.6 Explicit formulas
   14.5.7 Oesterlé’s bound

14.6 Trace codes and curves

14.7 Good curves
   14.7.1 Maximal curves
   14.7.2 Shimura modular curves
   14.7.3 Drinfeld modular curves
   14.7.4 Tsfasman-Vlăduţ-Zink bound
   14.7.5 Towers of Garcia-Stichtenoth
14.8 Applications of AG codes

14.8.1 McEliece crypto system with AG codes

14.8.2 Authentication codes

Here we consider an application of AG-codes to authentication. Recall that in Chapter 10, Section 10.3.1 we started to consider authentication codes that are constructed via almost universal and almost strongly universal hash functions. They, in turn, can be constructed using error-correction codes. We recall two methods of constructing authentication codes (almost strongly universal hash function to be precise) from error-correcting codes:

1. Construct AU-families from codes as per Proposition 10.3.7 and then use Stinson’s composition method, Theorem 10.3.10.

2. Construct ASU-families directly from error-correcting codes.

As an example we mentioned ASU-families constructed as in (1.) using Reed-Solomon codes, Exercise 10.3.2. Now we would like to move on and present a general construction of almost universal hash functions that employs AG-codes. The following proposition formulates the result we need.

**Proposition 14.8.1** Let \( C \) be an algebraic curve over \( \mathbb{F}_q \) with \( N+1 \) rational points \( P_0, P_1, \ldots, P_N \). Fix \( P = P_i \) for some \( i = 0, \ldots, N \) and let \( WS(P) = \{0, w_1, w_2, \ldots\} \) be the Weierstraß semigroup of \( P \). Then for each \( j \geq 1 \) one can construct an almost universal hash family \( \epsilon^{-U(N,q^j,q)} \), where \( \epsilon \leq w_j/N \).

**Proof.** Indeed, construct an AG-code \( C = C_L(D,w_jP) \), where the divisor \( D \) is defined as \( D = \sum_{k \neq i} P_k \) and \( P = P_i \). So \( C \) is obtained as an image of the evaluation map for the functions that have a pole only at \( P \) and its order is bounded by \( w_j \). From ?? we have that length of \( C \) is \( N \), \( \dim C = \dim L(w_jP) = j \), and \( d(C) \geq N - \deg(w_jP) = N - w_j \). So \( 1 - d(C)/N \leq w_j \) and now the claim easily follows.

As an example of this proposition, we show next how one can obtain AU-families from Hermitian curves.

**Proposition 14.8.2** For every prime power \( q \) and every \( i \leq q \), Hermitian curve \( y^q + y = x^{q+1} \) over \( \mathbb{F}_{q^2} \) yields

\[
\frac{i}{q^2} - U(q^3, q^{2+i}, q^2).
\]

**Proof.** Recall from ?? that Hermitian curve over \( \mathbb{F}_{q^2} \) has \( q^3 + 1 \) rational points \( P_1, \ldots, P_{q^2}, P_\infty \). Construct \( C = C_L(D, w_iP) \), where \( P = P_\infty \) is a place at infinity, \( D = \sum_{i=1}^{q^2} P_i \), and \( WS(P) = \{0, w_1, w_2, \ldots\} \). It is known that the Weierstraß semigroup \( WS(P) \) is generated by \( q \) and \( q+1 \). Let us show that \( w_{\left\lfloor \frac{i}{q^2} \right\rfloor} = iq \) for all \( i \leq q \). We proceed by induction. For \( i = 1 \) we have \( w_1 = q \), which is obviously true. Then suppose that for some \( i \geq 1 \) we have \( w_{\left\lfloor \frac{i}{q^2} \right\rfloor} = (i-1)q \) and want to prove \( w_{\left\lfloor \frac{i}{q^2} \right\rfloor} = iq \). Clearly, for this we need to show that there is exactly \( i-1 \) non-gaps between \( (i-1)q \) and \( iq \) (these numbers themselves are not included in the count). So for the non-gaps \( aq + b(q+1) \)
that lie between \((i-1)q\) and \(iq\) we have: \((i-1)q < aq + b(q+1) < iq\). Thus, automatically, \(a < i\). We have then
\[
(i-a-1)\frac{q}{q+1} < b < (i-a)\frac{q}{q+1},
\]
so from here we see that \(0 < a < i - 1\), because for \(a = i - 1\) we have \(b < q/(q+1)\), which is not possible. So there are \(i - 1\) values of \(a\), namely \(0, \ldots, i - 2\), which could give rise to a non-gap. The interval from (14.1) has length \(q/(q+1) < 1\). So it may contain at most one integer. If \(i-a < q+1\), then \((i-a-1)q/(q+1) < i-a-1 < (i-a)q/(q+1)\). And thus an integer \(i-a-1\) is always in that interval if \(i-a < q+1\). But for \(0 < a < i - 1\), the condition \(i-a < q+1\) is always full filled, since \(i \leq q\) by the assumption. Thus for every \(0 < a < i - 1\), there exists exactly one \(b = i-a-1\), such that \(aq + b(q+1)\) lies between \((i-1)q\) and \(iq\). It is also easily seen that all these non-gaps are different. So, indeed, \(w_{i+1} = iq\) for all \(i \leq q\).

Now the claim follows form Proposition 14.8.1.

As a consequence we have

**Corollary 14.8.3** Let \(a, b\) be positive integers such that \(b \leq a \leq 2b\) and \(q^a\) is a square. Then there exists
\[
\frac{2}{q^b} = SU(q^{5a/2+b}, q^{2a^2/2}, q^b).
\]

**Proof.** Do the "Hermitian" construction from the previous proposition over \(F_{q^a}\) and \(i = q^{a-b}\). Then the claim follows from Theorem 10.3.10 and Exercise 10.3.2.

*** Suzuki curves? ***

To get some feeling about all these, the reader is advised to solve Exercise 14.8.1. Now we move to (2.). We would like to show the direct construction of Xing et. al ?? that uses AG-codes.

**Theorem 14.8.4** Let \(C\) be an algebraic curve over \(F_q\) of genus \(g\) and \(L\) be some set of rational points of \(C\). Let \(G\) be a positive divisor such that \(|R| > \deg(G) \geq 2g + 1\) and \(R \cap \text{supp}(G) = \emptyset\). Then there exists \(\epsilon = \text{ASU}(N, n, m)\) with \(N = q|R|, n = q^{\deg(G)+g+1}, m = q\), and \(\epsilon = \deg(G)/|R|\).

**Proof.** Consider the set \(H = \{h_{(P,\alpha)} : \mathcal{L}(G) \to F_q[h_{(P,\alpha)}(f) = f(P) + \alpha, f \in \mathcal{L}(G)\}\). Take \(H\) as functions in the definition of an ASU-family; set \(X = \mathcal{L}(G), Y = F_q\). Then \(|X| = \dim \mathcal{L}(G) = \deg(G) - g + 1\), because \(\deg(G) \geq 2g + 1 > 2g - 1\).

It can be shown (see Exercise 14.8.2) that if \(\deg(G) \geq 2g + 1\), then \(|H| = q|R|\). It is also easy to see that for any \(a \in \mathcal{L}(G)\) and any \(b \in F_q\) there exists exactly \(|R| = |H|/q\) functions from \(H\) that map \(a\) to \(b\). This proves the first part of being ASU. As to the second part consider
\[
m = \max_{a_1 \neq a_2 \in \mathcal{L}(G), b_1, b_2 \in F_q} |\{h_{(P,\alpha)} \in H[h_{(P,\alpha)}(a_1) = b_1; h_{(P,\alpha)}(a_2) = b_2]\} =
\max_{a_1 \neq a_2 \in \mathcal{L}(G), b_1, b_2 \in F_q} |\{(P, \alpha) \in R \times F_q|(a_1 - a_2 - b_1 + b_2)(P) = 0; a_2(P) + \alpha = b_2\}|
\]
As $a_1 - a_2 \in \mathcal{L}(G) \setminus \{0\}$ and $b_1 - b_2 \in \mathbb{F}_q$ we see that $a_1 - a_2 - b_1 + b_2 \in \mathcal{L}(G) \setminus \{0\}$. Note that there cannot be more than $\deg(G)$ zeros of $a_1 - a_2 - b_1 + b_2$ among points in $R$ (cf. ??). Since $\alpha$ in $(P, \alpha)$ is uniquely determined by $P \in R$, we see that there are at most $\deg(G)$ pairs $(P, \alpha) \in R \times \mathbb{F}_q$ that satisfy both $(a_1 - a_2 - b_1 + b_2)(P) = 0, a_2(P) + \alpha = b_2$. In other words,

$$m \leq \deg(G) = \frac{\deg(G) \cdot |H|}{|R| \cdot q}.$$ 

We can take now $\epsilon = \deg(G)/|R|$ in Definition 10.3.8. \hfill \diamond

Again we present here a concrete result coming from Hermitian codes.

**Corollary 14.8.5** Let $q$ be a prime power and let an integer $q^3 > d \geq q(q-1)+1$ be given. Then there exists

$$d \frac{q^3}{q^2} = \text{ASU}(q^5, q^{d-q(q-1)/2+1}, q^2).$$

**Proof.** Consider again the Hermitian curve over $\mathbb{F}_{q^2}$. Take any rational point $P$ and construct $C = C_{\mathcal{L}}(D, G)$, where $D = \sum_{P' \neq P} P'$ is the sum of all remaining rational points (there is $q^3$ of them), and $G = dP$. Then the claim follows directly from the previous theorem. \hfill \diamond

For a numerical example we refer again to Exercise 14.8.1.

14.8.3 Fast multiplication in finite fields

14.8.4 Correlation sequences and pseudo random sequences

14.8.5 Quantum codes

14.8.6 Exercises

14.8.1 Suppose we would like to obtain an authentication code with $P_S = 2^{-20} \geq P_I$ and $\log |S| \geq 2^{34}$. Give the parameters of such an authentication code using the following constructions. Compare the results.

- OA-construction as per Theorem 10.3.5.
- RS-construction as per Exercise 10.3.2.
- Hermitian construction as per Corollary 14.8.3.
- Hermitian construction as per Corollary 14.8.5.

14.8.2 Let $\mathcal{H} = \{h_{(P, \alpha)} : \mathcal{L}(G) \to \mathbb{F}_q | h_{(P, \alpha)}(f) = f(P) + \alpha, f \in \mathcal{L}(G)\}$ as in the proof of Theorem 14.8.4. Prove that if $\deg(G) \geq 2y + 1$, then $|\mathcal{H}| = q|R|.$

14.9 Notes
Bibliography

[1]

[2]


Index

action, 40
adjacent, 123
Adleman, 311
AES, 302
algebra, 14
algorithm
  APGZ, 233
  basic, 273
  efficient, 35
  Euclidean, 191, 277
  Sudan, 286
  Sugiyama, 277
alphabet, 18
anti-symmetric, 136
arc, 95
Arimoto, 233
arrangement, 98
  central, 98
  essential, 98
  graphic, 127
  simple, 98
array
  orthogonal, 159, 160
  linear, 160
atom, 141
atomic, 141
attack
  adaptive chosen-ciphertext, 300
  adaptive chosen-plaintext, 300
  chosen-plaintext, 300
  ciphertext
    chosen, 300
    ciphertext-only, 300
    known-plaintext, 300
    related-key, 300
authentication code, 318
  MAC, 317
  message, 317
automorphism, 41
  monomial, 42
  permutation, 42
axiom
  circuit elimination, 130
  independence augmentation, 128
balanced, 84
ball, 19
basis, 22
  external, 131
  internal, 131
Berlekamp, 334
bilinear, 30
binary cipher, 329
binomial
  Gaussian, 39, 93
Bose, 214
bound
  AB, 223
  BCH, 214
  Bush, 161
  Gilbert, 72
  Gilbert-Varshamov, 73
  greatest lower, 139
  Griesmer, 68
  Hamming, 69
  HT, 220
  Johnson, 282
  least upper, 139
  Plotkin, 71
  Roos, 222
  shift, 225
  Singleton, 63
  sphere-covering, 70
  sphere-packing, 69
  Varshamov, 73
bound:redundancy, 74
box
  delay, 330
  S, 304
Brickell, 317
bridge, 124
broken
partially, 300
totally, 300
capacity, 14, 37
error-correcting, 34, 282
chain
extension of, 142
maximal, 142
of length, 137
channel
binary symmetric, 36
q-ary symmetric, 36
character, 85
principal, 85
characteristic, 201
Chaudhuri, 214
Chinese remainder, 207
cipher
alphabetic substitution, 298
block, 296
Caesar, 296
Feistel, 302
iterative block, 302
permutation, 297
self-synchronizing stream, 330
stream, 329
substitution, 296, 298
transposition, 297
Vigenère, 298
ciphertext, 295, 296
confusion, 298
diffusion, 297
circuit, 130
class
parallel, 91
closed
algebraically, 202
closure, 145
cocircuit, 130
code
alternant, 259
augmented, 51
BCH, 215
narrow sense, 215
primitive, 215
block, 18
Cauchy, 67
genrated, 67
concatenated, 61
constant weight, 72
convolutional, 18
cycle, 127
degenerate, 93
dual, 30
complementary, 31
self, 31
weakly self, 31
error-correcting, 13
expurgated, 52
extended, 48
extension by scalars, 104, 251
Gallager, 127
genus, 102
geometric Goppa, 422
Golay, 60
ternary, 32
Goppa, 260
graph, 127
sparse, 127
Tanner, 127
Hamming, 22, 29
hull, 31
inner, 61
lengthened, 52
linear, 21
low-density parity check, 127
maximum distance separable, 64
MDS, 64
Melas, 228
orthogonal, 30, 31
self, 31
outer, 61
projective, 94
punctured, 47, 107
reduced, 145
Reed-Muller, 266
Reed-Solomon, 241
extended, 243
generalized, 66, 243
geometric, 422
residual, 68
restricted, 47
restriction by scalars, 214, 251
reverse, 196
shortened, 50, 107
simplex, 29
sub, 22
even weight, 22
subfield, 214, 251
super, 22, 214, 251
INDEX

minimum, 19
distribution
  weight, 79
division with rest, 191
divisor, 415
  canonical, 418
degree, 415
effective, 415
greatest common, 191
  principal, 416
  support, 415
DLP, 310
domain
  Euclidean, 191
dual, 30, 93
  self
    formally, 88
    quasi, 44
edge, 123
  acyclic, 124
El Gamal, 314
Elias, 281
elimination
  Gaussian, 24
embedded, 92
encoder, 18
  systematic, 25
encoding, 23, 195
encryption, 296
  AES, 307
  confusion, 304
  DES, 304
  diffusion, 304
  El Gamal, 315
  McEliece, 337
  Niederreiter, 339
  RSA, 312
end, 123
entropy function, 75
enumerator
  weight, 79
    average, 84
    coset leader, 35
    extended, 105
    generalized, 113
    homogeneous, 79
equality
  modular, 142
equation
  Key, 236
  equivalent, 41
    generalized, 42
    monomial, 42
    permutation, 42
  error, 34
    decoding, 15, 34
    number of, 27, 34
    undetected, 89
  Euclid, 191, 277
  Euclid-Sugiyama, 277
  Euler, 206
evaluation, 243
expand, 302
explicit, 26, 91
  exponent
    universal, 317
failure
  decoding, 15, 34
family
  almost universal, 322
feedback, 331
Feistel, 302
field
  Galois, 203
  prime, 201
  splitting, 202
  sub, 201
finite
  locally, 137
flat, 145
force
  brute, 301
form
  differential, 417
formula
  closed, 234
  deletion-contraction, 126, 134
  deletion-restriction, 149
  Möbius inversion, 138
  Stirling, 75
Forney, 236
Fourier, 229
free
  square, 261
Frobenius, 254
function
  Euler’s phi, 140
  Möbius, 137
one-way, 310
trapdoor, 310
state, 330
sum, 138
symmetric, 231
zeta, 157

Galois, 254
gap, 420
Weierstrass, 420
Gauss, 24, 39
generator
  clock-controlled, 334
  nonlinear combination, 334
  nonlinear filter, 334
  shrinking, 335
generic, 234
genus, 418
Gilbert, 72, 73
girth, 126
Golay, 60
good
  asymptotically, 75
Goppa, 260
Gorenstein, 233
graph, 123
  coloring, 124
  connected, 127
  contraction, 126
  deletion, 126
  planar, 124
  simple, 123
greatest common divisor, 277
Griesmer, 68
group
  automorphism, 42
  monomial, 42
  permutation, 42
  dihedral, 197
Galois, 254
general linear, 38
symmetric, 40

Hamming, 14, 16, 18
Hartmann, 220
Hellman, 309, 316
hierarchy
  weight, 111
Hocquenghem, 214
hyperplane, 92, 97

homogeneous, 97
projective, 92

ideal, 191
  generated, 191
identity
  generalized Newton, 231
MacWilliams, 85
image, 40
impersonation, 318
implicit, 26, 91
incident, 92, 123
independent, 224
index, 160
inequality
  semimodular, 142
  triangle, 19
information theory, 14
inner product, 30
interpolation
  Lagrange, 245
interval, 136
invariant, 40, 43, 254
  permutation, 43
inversion, 306
isometry, 40
  linear, 40
isomorphic
  graph, 124
  matroid, 129
  poset, 141
isthmus
  graph, 124
  matroid, 129

Jefferson cylinder, 345
Johnson, 282
join, 139
juxtaposition, 56

Kasiski method, 299
kernel, 272
key
  decryption, 296, 310
  dual, 305
  encryption, 296, 310
  private, 309
  public, 309
  schedule, 302
  secret, 296
size, 302
weak, 297, 304
semi, 305
key generation
El Gamal, 314
Niederreiter, 339
RSA, 312
key space, 310
keyspace, 318
keystream, 329
LaGrange, 245
lattice, 139
free, 147
geometric, 141
modular, 141
of flats, 145
rank, 141
semimodular, 141
super solvable, 147
Leibniz, 202
length, 18
level, 142
levels, 160
LFSR, 329
line
affine, 90
at infinity, 91
parallel, 91
projective, 90
lines, 91
Lint, van, 223
locator, 230
error, 233, 273
logarithm, 203
Zech, 203
loop, 144
graph, 123
matroid, 128
Möbious, 137
MacWilliams, 85
map
$\mathbb{F}_2$-linear, 306
authentication, 318
evaluation, 242, 243
Frobenius, 254
trace, 258
Massey, 334
matrix
Cauchy, 67
generalized, 67
diagonal, 40
generator, 23
incidence, 127
monomial, 40
parity check, 26
permutation, 40
reduced, 145
syndrome, 227, 272
matroid, 128
basis, 128
cographic, 131
cycle, 130
dimension, 128
dual, 129
free, 128
graphic, 131
rank, 128
realizable, 129
representable, 129
simple, 128, 144
uniform, 128
Mattson, 229
maximum, 138
McEliece, 295
meet, 139
Melas, 228
memory, 301
Merkle, 316
message, 18
method
Stinson’s composition, 323
minimum, 138
modulation, 16
monotone, 141
strictly, 141
morphism
graph, 124
matroid, 129
Muller, 266
neighbor, 123
Newton, 201, 231
Niederreiter, 295
nondegenerate, 30
nongap, 420
operations
elementary row, 24
INDEX

order, 158, 159
  linear, 136
  partial, 136
output, 329

pad
  one-time, 329

pair
  error-correcting, 274

parallel, 144
  graph, 123
  matroid, 128

parameter, 21
  parametric, 91

parity check, 26
path, 126
pay-off, 319
perfect, 70
period, 297, 333
periodic, 333
  ultimately, 333

permutation
  substitution, 305

Peterson, 233

pivots, 25
plaintext, 295, 296

plane
  affine, 91
  projective, 91

Plotkin, 71

point
  anti-fixed, 305
  at infinity, 91
  fixed, 305
  rational, 157

points, 91

polynomial
  characteristic, 146, 331
    two variable, 146
  chromatic, 124
  coboundary, 146
  cyclotomic, 207
  error-evaluator, 236
  error-locator, 233
  generator, 193
  locator, 230
  M"obius, 146
  Mattson-Solomon, 229
  minimal, 205
  parity check, 197

  Poincaré, 146
  syndrome, 235
  Tutte, 131
  Whitney rank generating, 131

poset, 136

position
  error, 34, 233
  general, 95
  primitive, 203
  cryptographic, 296

principle
  inclusion/exclusion, 99
    of inclusion/exclusion, 140

probabilistic, 14

probability
  correct decoding, 36
  cross-over, 36
  deception, 318
  decoding error, 36
  decoding failure, 36
  error, 14, 36
  retransmission, 89

problem
  Diffie-Hellman, 316
  discrete logarithm, 310, 314
  DLP, 310, 314
  multivariate quadratic, 316
  RSA, 313
  subset sum, 316

processing, 301

product, 202
  direct, 53
  Kronecker, 53
  star, 220, 271
  tensor, 53

projection plane, 92

property
  Jordan-H"older, 142
  pseudorandom, 334

quotient, 191

radius
  covering, 69
  decoding, 34, 282

rate
  error, 36
    information, 14, 18
  rational, 91
  reciprocal, 197
monic, 197
redundant, 13, 18
Reed, 241, 266
reflexive, 136
register
  linear feedback shift, 329
relation
  reverse, 139
representation
  exponential, 203
  principal, 201
residue, 419
restriction, 149
retransmission, 26, 89
reverse, 196
Riemann, 419
Rijmen, 302
Rijndael, 302
ring
  factor, 191
Rivest, 311
Roch, 419
row reduced echelon form, 24
scheme
  El Gamal, 310
  Rabin, 310
  secret sharing, 325
    ideal, 349
    perfect, 348
    threshold, 325
secrecy
  perfect, 301
security
  computational, 301
  provable, 301
  state-of-the-art, 301
  unconditional, 301
seed, 329
semigroup
  Weierstrass, 420
sequence, 333
  initial state, 331
  linear recurring, 330
    homogeneous, 331
  maximal period, 333
  super increasing, 316
set
  check, 28
  complete defining, 211
defining, 211
generating, 221
information, 25
root, 211
shift, 220
zero, 211
Shamir, 311, 317, 325
Shannon, 14, 37
Shestakov, 343
shift
  cyclic, 189
Sidelnikov, 258, 343
sieve
  number field, 347
Singleton, 63
size, 160
Solomon, 229, 241
space, 92
  ciphertext, 295, 309
  key, 296
  message, 318
  null, 273
  plaintext, 295, 309
  projective, 92
spectrum
  weight, 79
sphere, 19
split-knowledge scheme, 328
square
  Latin, 158
  Greek, 159
  mutually orthogonal, 159
standard
  advanced encryption, 302
  AES, 302
  data encryption, 302
  DES, 302
state, 329
  initial, 330
  source, 318
Stinson, 323
storage, 301
strategy
  wide trail, 307, 346
strength, 160
structure
  access, 349
subcode
  minimal, 156
subgraph, 126
subset
  independent
  maximal, 128
subspace
  affine, 92
substitution, 298, 318
Sudan, 281, 286
Sugiyama, 277
sum
  direct, 56
support, 21, 111
symmetric, 30
syndrome, 27, 227
  known, 232
system
  projective, 93
    equivalent, 94
    simple, 93
systematic, 25, 196
tag
  authentication, 318
time, 301
transform
  discrete Fourier, 229
transformation
  decryption, 296
  encryption, 296
  fractional, 248
  projective, 94
  round, 302
transitive, 136
trapdoor, 310
trivial, 189
Tzeng, 220
value, 230
  error, 34
Vandermonde, 65
variety
  affine, 157
Varshamov, 73
vector
  error, 27, 34
Venn-diagram, 16
vertex, 123

Weierstrass, 420
weight, 21, 111
  constant, 30
even, 22
generalized, 111
minimum, 21
Whitney number, 148
  first kind, 148
  second kind, 148
Wilson, 223
word
  message, 18
  source, 18
Wozencraft, 281
Zetterberg, 228
Zierler, 233