Chapter 1

Introduction

Computations with large finite or infinite groups are usually very tedious and time consuming. In many cases the computations carried out are very mechanical and error prone when carried out by hand. Such computations can often be carried out more easily by computer. For more complicated tasks one needs to design and implement new algorithms. For groups in particular, this includes operations with group elements (multiplication, inversion, conjugation, etc.) or other important properties (subgroup structure, conjugacy classes, etc.). The first problem is deciding how elements should be represented in the computer. Often a group is defined *intrinsically*, that is, defined implicitly by requiring some properties on the elements (e.g., the fixed point subgroup of another group). For computations with group elements, such a definition is not very useful, since it provides no group elements other than the identity. In such cases one needs an *extrinsic* definition for the group, such as a presentation or a matrix representation.

We design and implement algorithms for computation with groups of Lie type. Algorithms for element arithmetic in the Steinberg presentation of untwisted groups of Lie type, and for conversion between this presentation and linear representations, were given in [12] (building on work of [15] and [26]). We extend this work to twisted groups, including groups that are not quasisplit.

A twisted group of Lie type is the group of rational points of a twisted form of a reductive linear algebraic group. These forms are classified by Galois cohomology. In order to compute the Galois cohomology, we develop a method for computing the cohomology of a finitely presented group Γ on a finite group A. This method is of interest in its own right. We then extend this method to the Galois cohomology of reductive linear algebraic groups.

Let G be a reductive linear algebraic group defined over a field k. A twisted group of Lie type $G_{\boldsymbol{\alpha}}(k)$ is uniquely determined by the cocycle $\boldsymbol{\alpha}$ of the Galois group of K on $A := \mathrm{Aut}_K(G)$, the group of K-algebraic automorphisms where

K is a finite Galois extension of k. We give algorithms for computing the relative root system of $G_{\boldsymbol{\alpha}}(k)$, the root subgroups, and the root elements, as well as algorithms for the computing of relations between root elements. This enables us to compute inside the normal subgroup $G_{\boldsymbol{\alpha}}(k)^\dagger$ of $G_{\boldsymbol{\alpha}}(k)$ generated by the root elements. We apply our algorithms to several examples, including ${}^2\mathrm{E}_{6,1}(k)$ and ${}^{3,6}\mathrm{D}_{4,1}(k)$. In this application, the field k need not be specified, one only needs to assume some properties of k.

As an application, we develop an algorithm for computing all twisted maximal tori of a finite group of Lie type. The order of such a torus is computed as a polynomial in q, the order of the field k. We also compute the orders of the factors in a decomposition of the torus as a direct product of cyclic subgroups. For a given field k, we compute the maximal tori of $G_{\boldsymbol{\beta}}(k)$ as subgroups of $G_{\boldsymbol{\beta}}(K)$ over some extension field K, and then use the effective version of Lang's Theorem [11] to conjugate the torus to a k-torus, which is a subgroup of $G_{\boldsymbol{\beta}}(k)$.

Using this information on the maximal tori, we provide an algorithm for computing all Sylow subgroups of a finite group of Lie type. If p is not the characteristic of the field, the Sylow subgroup is computed as a subgroup of the normaliser of a k-torus.

All algorithms presented here have been implemented by the author in Magma [5].