# 4EE11
# Programming Project: Chaotic mixing

January 7, 2009

## 1 Goal of the project

The goal of this project is

1. to learn the basic elements of Fortran 95,
2. to develop a tool for tracking interfaces in 2D in Fortran 95,
3. to use this tool to study the efficiency of mixing in a rectangular cavity flow.

We will be using the Silverfrost Fortran 95 compiler for Windows including the plato3 interface for building projects. It can be installed[1] from the Campus software site of the TU/e.

## 2 Work plan

A possible work plan is as follows:

1. Read the theory guide [1] and discuss/solve the problems 1, 2, 11, 13 and 14 of the same document.
2. Study Chapters 1 to 6 and Secs. 8.1–8.3. of the book of Chapman [2] according to the 'first read' in the study guide [3].
3. Write a Fortran 95 program that computes and writes the length of the interface and the area of the domain surrounded by the interface from a given set of points. Make the program such that
   (a) the coordinates of the points can either be read from a file or filled in the program.
   (b) the output is written to a file,
4. Use/adapt the program in 3. to verify that the errors in the length and area for a circular blob are given by the Eqs. (13) and (14) in the theory guide [1], respectively. In particular verify the quadratic decrease of the error with the number of points. You can do that by running the program several times with a different number of points $n$ or add a loop in the program itself.

[1] Install the version for Win32 and DO NOT INSTALL the version for the .NET framework.

5. Study Chapter 7 the book of Chapman [2] according the first read in the study guide [3].

6. Solve problem 8 of the theory guide [1].

7. Adapt the program in 3. such that it becomes more 'modular': try to identify one or more subtasks and put these into subroutines and functions and make the main program as 'empty' as possible. Also add a function that computes the angle $\alpha$ between two elements and compute $\alpha$ in all nodal points for a simple interface, such as 6-sided regular polygon and verify the result. .

8. Study Secs. 11.1 and 13.1 of the book of Chapman [2].

9. Write a module that defines the kinds of the `real` having precision $p = 6$ ('single precision') and $p = 15$ ('double precision'). `use` this module in your program and define all you constants and variables to be 'double precision'.

10. Restructure the program such that it consists of a set of modules and a small main program, making all interfaces explicit.

11. Study the remaining part of Chapter 8 and Chapter 9 the book of Chapman [2] according the first read in the study guide [3].

12. Adapt the program in 10. such that the arrays are defined dynamically, i.e. the size is determined at run time from the input parameters. Is it a good idea to define the size of the arrays according to the number of nodes at the start and increase the size if needed?

13. Study Chapter 12 of the book of Chapman [2].

14. Discuss/solve the problems 3 and 4 of the theory guide [1].

15. Write a module (name `interface_tracking_m`, for example) that contains the following:

    (a) A derived type (name `interface_t`, for example) that contains all the necessary information that defines an interface that needs to be tracked and updated[2].

    (b) A subroutine that allocates all the arrays in a 'structure' of type `interface_t` and initializes all the variables in the structure[3]

    (c) A subroutine that deallocates all the arrays in a 'structure' of type `interface_t` and set all the scalar variables in the structure to some default value (for example zero)[4]

    (d) A function that computes the length of the interface[5].

    (e) A function that computes the area of the domain in between the (closed) interface.

[2] In OOP (Object-Oriented Programming) parlance a 'structure' defined by a derived type such as `interface_t` is called an *object*

[3] In OOP parlance such a routine is called a *constructor*.

[4] In OOP parlance such a routine is called a *destructor*.

[5] In OOP parlance a routine that operates on an object and 'does something' with it, represents a *method*.

(f) A function or subroutine that computes the angle $\alpha$ between all consecutive elements in the interface.

16. Rewrite the program in 12. such that it 'uses' the module `interface_tracking_m` written in 15.

17. Add a subroutine (name `track_points`, for example) to the module `interface_tracking_m` that updates the points in the interface if going from a time $t_1$ to $t_2$ using the second-order Runge-Kutta scheme as discussed in the theory guide [1]. In order to easily switch between different problems, the velocity field should be supplied to the routine `track_points` using a function or subroutine in the argument list (See Section 7.5 of the book of Chapman [2]).

18. Write a program that starts with a number of points on a circle and tracks these points for three time units with the following velocity field:

$$\begin{aligned} u(x, y) &= x \\ v(x, y) &= -y \end{aligned} \tag{1}$$

Try various initial positions of the circle. Plot the results using Matlab.

19. Discuss/solve the problems 5, 6 and 7 of the theory guide [1].

20. Add a subroutine (name `split_elements`, for example) to the module `interface_tracking_m` that adds points to the interface if the distance between the points becomes too large or if the interface becomes too curved as described in the theory guide [1]. Use the routine `track_points`, as defined in 17., to compute the coordinates of the newly added points! Use the velocity field Eq. (1) to test the routine.

21. Discuss/solve the problems 9 and 10 of the theory guide [1].

22. Write a module that specifies a function or subroutine for the velocity field in the time-periodic flow in a cavity as specified in the theory guide [1].

23. Write a main program that uses both your module `interface_tracking_m` and the module for the velocity field in the cavity. The function or subroutine for the velocity field should be part of the argument list of the routine `track_points`.

24. Run the program developed in 23. for several, initially circular, blobs for a number of periods with varying dimensionless displacement $D$. Check that the area of the blobs remains approximately constant. Plot the length of the interface as a function of time. Is the length of the interface growing exponentially? How does the growth rate depend on the position of the blob? Is the mixing process efficient for values of $D$ that are used?

25. Discuss/solve problem 12 of the theory guide [1].

# 3 Time schedule

**week 1** Items 1 and (part of) 2 of the work plan.

**week 2** Items 2–7 of the work plan.

**week 3** Items 8 to (part of) 15 of the work plan.

**week 4** Items 15–22 of the work plan.

**week 5** Items 23–25 of the work plan. Finish and 'beautify' the code. Finish the final report.

# 4 Finalization

The final code developed for item 23 of the work plan and a report containing

1. answers to all the problems in the theory guide,
2. a discussion, including computational results, of item 24.

must be submitted for finalization of this project.

# References


[1] M.A. Hulsen and P.D. Anderson. Chaotic mixing in a rectangular cavity: theoretical background.

[2] S.J. Chapman. *Fortran 95/2003 for Scientists and Engineers*. McGraw-Hill, London, 2007.

[3] M.A. Hulsen. Study guide for the book: Stephen J. Chapman "Fortran 95/2003 for Scientists and Engineers".