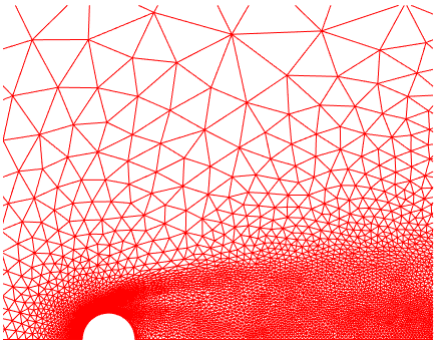

StabFem Documentation

Manual

IMFT, v0.2 May 2018

Contents

Previous Note

StabFem is a software to perform Global Stability calculations in Fluid Mechanics, which is developed for both research and education purposes. It can be downloaded at `https://github.com/erbafdavid/StabFem`. Additionally, one will previously need to install *Matlab* and *FreeFem++* on the running system. For now, *StabFem* works perfectly in *Ubuntu16.06*. On other Operating Systems, it's not sure that it will work.

This document is a collaboration of different people and is intended to explain, the best way possible, the different parts of *StabFem*. Perhaps as the reader will read it, some chapters are uncompleted or complicated to understand. The reader is invited to contribute in order to render the documentation more understandable.

Chapter 1

General View of *StabFem*

StabFem has a main directory where all the projects are located and where the mutual scripts are located too. The directory is composed by the following particular project directories:

1. ACOUSTICS_PIPES;
2. CYLINDER;
3. CYLINDER_VIV;
4. DISK_IN_PIPE.
5. etc...

The mutual directories are the followings:

1. SOURCES_FREEFEM;
2. SOURCES_MATLAB;
3. Documentation.

These directories are used by the several projects of *StabFem*.

Attention: When you do some change in the mutual directory files, we have to assure that it will work on all the projects.

1.1 General Features

- With *StabFem* we can solve forced problems, eigenvalue problems, etc...
- *StabFem* allows us to save the scripts used to generate the different data and graphs, unabling to reproduce the same results (useful for repeat figures for articles, confirming results, etc...).

1.2 What is attended from this manual document

This document presents this first chapter describing what can be found in a *StabFem* project, a chapter describing the running logic of a *StabFem* program and the main common files and a chapter for each project, describing succinctly their running features.

1.3 Notes and tips for users and developers

When contributing to the project you can intervene at two levels:

- Contributring as a *User* means that you will use the available solvers and drivers for a class of problems already integrated in the project. For instance, if you wish to study the wake flow around an elliptical 2D body, which uses the same solvers than the case of a circular cylinder already present in the base.

  In this case, you will create a directory for your case (for instance *ELLIPSE/*) containing : The mesh generator freefem script (for instance *Mesh_Ellipse.edp*), the macro file *Macros_Stabfem.edp* containing the case-dependent boundary condition and postprocessing detail, and a number of Mablab scripts. O,n the other hand, you will suposedly not modify thecommon files in the source directories (with the exception of the file *SOURCES_MATLAB/SF_Start.m*).

- Contributing as a *Developer* means that you wish to help integrate new Freefem solver into the project (for instance, you want to solve a 2D problem in the Bousinesq approximation, you already have a set of Freefem solvers for this case and you wish to integrate them in the project).

1.3.1 FAQ

Here is a kind of FAQ of the projecft, regerouping in random order notes, tips and "good practize recomendations" for users and developers.

- At installation (git clone) it is recommended that you install the whole StabFem project in your home directory and that you do not remove or displace the parts of the project that you don't (immediately) need. Otherwise this will perturb the operation of the version manager git. In short : if you there are parts you think you don't need, don't remove them, just ignore them !

- good usage of the "verbosity"' parameter ...

1.3.2 Note on the usage of github

The project is supported by the git subversion manager program. Here are a fexw tips/recommendations :

If you are contributing as a user

- After the initial git clone, you may want to get the latest development of the main branch using *git pull*. This will only update the sources in the common repositories, not your own files.

  If you have modified the *SF_Start.m* file and/or made minor modifications to other files that you wish to keep in your local version but do not want to export to the main branch odf the project, the procedure is to do successively :

  git stash

  git pulll

  git stash apply.

  (don't be afraid, any "mistake" with git can be undone !)

- If you want to incorporate your work in the project repository (normally after everything is validated...) you should do the following :

  git add "files". *(please add only source files of your case directory, such as freefem mesh generator and macro scripts, an example matlab script producing sample results for your case, and possibly matlab functions you have developed for your own case and are not sure they will work for other cases)*

  git commit -a

  git push

  For this last step you need to be register as a contributor, just ask !

- (...)

If you are contributing as a developer

- The best way is to creat a "fork" or a "branch" (still not sure what is the most efficient)
- Once you have your own branch/fork, use gut commit / git push as frequently as you wish !
- If you want to merge with the main branch, create a *pull request*.
- (...)

1.4 Notes/tips about FreeFem

All the Freefem tricks which are not in the manual....

Part I

Stabfem principles : learning by examples

Chapter 2

The basics of StabFem

In this chapter we explain the basic functionalities of StabFem. We recommend that you study the basic example EXAMPLE_Lshape.m.

This directory contains three freefem programs and a demo matlab script doing solving the following problem :

2.1 Simple problem : solution of heat equation in a L-shaped domain

1. Solve the steady heat conduction equations on a L-shaped domain with constant volumic source term P and homogeneous Dirichlet boiundary conditions :

$$\nabla^2 T = P \text{ for } \mathbf{x} \in \Omega; \quad T = 0 \text{ for } \mathbf{x} \in \Gamma$$

2. Solve the unsteady heat conduction equations on a L-shaped domain with zero source term P and time-periodic Dirichlet boundary conditions :

$$\partial T_1/\partial t = \nabla^2 T_1 \text{ for } \mathbf{x} \in \Omega; \quad T_1 = T_w e^{i\omega t} \text{ for } \mathbf{x} \in \Gamma$$

2.2 Matlab script and results

The programs are sufficiently short to be fully given in these pages. See figures **??**, **??**, **??**.

The reader already familiar with FreeFm++ should not have any provblem in undersanding the programs. If not, we recommand that you study the FreeFem++ Manual (chapter "Learning by examples").

The results are given in figure **??**.

```
% This script is a basic example which illustrates the main
    functionalities of the StabFem Software
clear all; close all;
run('../SOURCES_MATLAB/SF_Start.m');ffdatadir = './';verbosity=100;

% Generation of the mesh
Ndensity =40;
ffmesh=SF_Mesh('Lshape_Mesh.edp','Params',Ndensity)
figure();plotFF(ffmesh,'title','Mesh for L-shape body');
set(gca,'FontSize', 18); saveas(gca,'FIGURES/Lshape_Mesh','png');

pause

% importation and plotting of a real P1 field : temperature
heatS=SF_Launch('Lshape_Steady.edp','Mesh',ffmesh)
figure();plotFF(heatS,'T','title', 'Solution of the steady heat equation
    on a L-shaped domain');
set(gca,'FontSize', 18); saveas(gca,'FIGURES/Lshape_T0','png');

pause

% importation and plotting of data not associated to a mesh : temperature
    along a line
heatCut = importFFdata('Heat_1Dcut.ff2m')
figure();plot(heatCut.Xcut,heatCut.Tcut);
title('Temperature along a line : T(x,y=0.25)')
set(gca,'FontSize', 18); saveas(gca,'FIGURES/Lshape_T0_Cut','png');

pause

% importation and plotting of a complex P1 field : temperature for
    unsteady problem
heatU=SF_Launch('Lshape_Unsteady.edp','Params',100,'Mesh',ffmesh,'DataFile
    ','Heat_Unsteady.ff2m')
figure();plotFF(heatU,'Tc.re','title',['Ti: real(colors) part'])
%figure();plotFF(heatU,'Tc.re','contour','Tc.im','title',['Ti: ' char(13)
    'real(colors) and imaginary(levels) parts'])
set(gca,'FontSize', 18); saveas(gca,'FIGURES/Lshape_Tc','png');
```

Figure 2.1: Matlab program SCRIPT_Lshape.m)

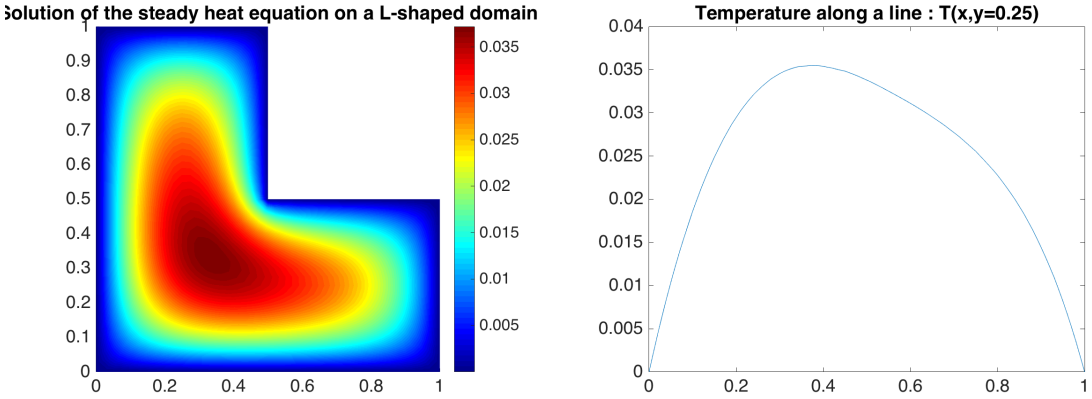

Figure 2.2: Solution for the steady hrat equation in a L-shape domain : (a) Temperature field $T(x, y)$. (b) Temperature $T(x, 0.25)$ along a horizontal line.

```
1  /// Program Lshape_Mesh.edp : generation of the mesh for the basic StabFem
       example
2  include "Macros_StabFem.edp";
3
4  int nn;
5  cout << "Enter the mesh density : " << endl;
6  cin >> nn;
7  cout << "### Mesh density nn =  " << nn << endl;
8
9  border a(t=0,1){x=t;y=0;label=1;};
10 border b(t=0,0.5){x=1;y=t;label=1;};
11 border c(t=0,0.5){x=1-t;y=0.5;label=1;};
12 border d(t=0.5,1){x=0.5;y=t;label=1;};
13 border e(t=0.5,1){x=1-t;y=1;label=1;};
14 border f(t=0,1){x=0;y=1-t;label=1;};
15
16 mesh th = buildmesh ( a(nn) + b(.5*nn) + c(.5*nn) +d(.5*nn) + e(.5*nn) + f
       (nn));
17 IFMACRO(FREEFEMPLOTS,YES)
18 plot(th);
19 ENDIFMACRO
20
21 SFWriteMesh(th);
```

Figure 2.3: FreeFem++ program Lshape_Mesh.edp)

```
/// Program Lshape_Steady.edp : resolution of the steady heat equation for
    the basic StabFem example
include "Macros_StabFem.edp";

// Read the mesh
mesh th = readmesh("mesh.msh");
fespace Vh(th,P1);

// Solve the Poisson Equation
Vh u,v;
func so= 1.0;
real cpu=clock();
solve Poisson(u,v,solver=LU)=int2d(th)(dx(u)*dx(v) + dy(u)*dy(v)) - int2d(
    th)( so*v)+on(1,u=0);

// Generates the data file for the StabFem driver with 2D mesh-organized
    data
{
        ofstream file("Data.ff2m");
    file << "### Data generated by Freefem++ ; " << endl;
    file << "Temperature in a L-Shape domain" << endl;
    file << "datatype EXAMPLE datastoragemode RePl " << endl;
        file << "P1 T" << endl << endl ;
        for (int j=0;j<u[].n ; j++)
        file << u[][j] << endl;
}

// exports as well in ".txt" format (for subsequent usage by FreeFem)
{
    ofstream file2("Data.txt");
    file2 << u[];
}




// Generates a second data file for the StabFem driver with data on a "
    slice"
{
        ofstream file("Heat_1Dcut.ff2m");
    file << "### Data generated by Freefem++ ; " << endl;
    file << "Temperature in a L-Shape domain: 1D cut along y=0.25" << endl
        ;
    file << "DataType EXAMPLE" << endl;
    int NN = 101;
        file << "real." << NN << " Xcut" << "  real." << NN << " Tcut" <<
            endl << endl ;
        for (int j=0;j<NN ; j++) { file << 1./(NN-1)*j << endl;} ;
        for (int j=0;j<NN ; j++) {file  << u(1./(NN-1)*j,.25) << endl;};
    file << endl;
}

// Freefem plot if required
IFMACRO(FREEFEMPLOTS,YES)
plot(th,u);
ENDIFMACRO
```

Figure 2.4: FreeFem++ program Lshape_Steady.edp)

2.3 How does it work ?

2.3.1 Analysis of the example

Let us observe and explain the most important steps of the example matlab script SCRIPT_Lshape.m.

- The first step is **creating and importing a mesh**. This is done by the function SF_Mesh.m invoked at line 7 of the script. Basically this function does the following operations :

  1. Launch the FreeFem++ program Lshape_Mesh.edp with the *entry parameter* [1] $Ndensity = 40$ (here corresponding to the mesh density).
  2. Import the data produced by the FreeFem++ Macro SFWriteMesh (more explanations in paragraph 3.3)
  3. Return the data as a Matlab *structure* object called ffmesh

  Execution of this step leads to the following :

```
>> ffmesh=SF_Mesh('Lshape_Mesh.edp','Params',Ndensity)
       ### INITIAL MESH CREATED WITH np = 1426 points

ffmesh =

         DataDescription: 'Mesh for a L-shaped domain'
                datatype: 'Mesh'
                meshtype: '2D'
                      np: 1426
                      nt: 2690
             problemtype: 'EXAMPLE'
                       R: 1
     InitialMeshDensity: 40
         meshgeneration: 0
                  points: [3x1426 double]
```

  This structure objects has several fields which will serve for plotting data obtained on this mesh. See more in section 2.3.

- The second step is **launching a FreeFem program using this mesh and importing results**. This is done by the generic driver function SF_Launch.m invoked at line 12 of the script. Basically this function does the following operations :

  1. Launch the FreeFem++ program Lshape_Steady.edp (here with no *entry parameter*).
  2. Import the data produced by the FreeFem++. By default the driver will look for a file called Data.ff2m. This file is the one created by the FreeFem program in lines 14-22 of the Freefem program **??**.

     Looking inside the file Data.ff2m, the header (i.e. first four lines) is as follows :

[1] Here and in the sequel, *entry parameter* means a value you will be invited to type on the keyboard if using directly FreeFem++ in the way you are used to. Just try !

```
### Data generated by Freefem++ ;
Temperature in a L-Shape domain
datatype EXAMPLE datastoragemode RePl
P1 T
```

, meaning that this files contains a P1 mesh-organised field called 'T'.

3. Return the data as a matlab *structure* object called heatS.

Execution of this step leads to the following :

```
>> heatS=SF_Launch('Lshape_Steady.edp','Mesh',ffmesh)

heatS =

        mesh: [1x1 struct]
    filename: 'Data.txt'
    datatype: 'Temperature'
           T: [1426x1 double]
```

Where we can see that the structure has field called 'T' (containing the imported data) and a field called 'mesh' which is actually the mesh object previously constructed and passed to the driver as a parameter as a field.

- The third step which is important to explain ist **importation of results previously generated by Freefem**. This is done by the generic driver function importFFdata.m invoked at line 12 of the script.

  Here, besides the file Data.edp previously imported, our program has generated a second data file called Heat_1Dcut.ff2m. The header of this file is as follows :

```
### Data generated by Freefem++ ;
Temperature in a L-Shape domain: 1D cut along y=0.25
DataType EXAMPLE
real.101 Xcut  real.101 Tcut
```

  , meaning that this files contains two vectors of dimension 101 called X and T.

  Execution of this step leads to the following :

```
>> heatCut = importFFdata('Heat_1Dcut.ff2m')

heatCut =

    datatype: 'Temperature'
        Xcut: [101x1 double]
        Tcut: [101x1 double]
```

  Where we can see that the data is imported as a structure with two fields corresponding to the required data.

The importFFdata.m functions is the core of the interface, and is actually internally called by the higher-levels drivers SF_Mesh.m and SF_Launch.m.

2.3.2 Explanations of the `.ff2m` exchange format

As can be seen in the previous short examples, the basis of the FreeFem/Matlab interface relies in the file exchange format .ff2m and the importation "wizard" importFFdata.m which reads these files.

Let us explain how such files are constituted. As we have observed, the header of a .ff2m file has the following structure :

```
1 ### Data generated by Freefem++ ;
2 (Description)
3 'property1 value1 property2 value2' (...)
4 'Datatype_1' 'DataName_1' 'Datatype_2' 'DataName_2'    ( ... )
5
6 ( numerical data ... )
```

.

Line 2 contains a description string which is imported as field 'datadescription'.

Line 3 contains a series of property names and alphanumeric (string) values.

Line 4 explains the nature of the data which is contained in the file. There can be as many data objects as you want organised in the order of your choice. The recognized types are currently as follows :

P1 for mesh-structured data stored as P1 finite-elements [2]

P1c for mesh-structured *complex* data stored as P1 finite-elements

real for real scalars,

complex for complex scalars,

real.N for real vectors of dimension N not associated to a mesh,

complex.N for complex vectors of dimension N not associated to a mesh,

P1surf for real data defined along a frontier of the mesh,

P1surfc for complex data defined along a frontier of the mesh.

2.3.3 Explanations about the mesh

Examination of the simple example here shows that the mesh generation produces four files (here all generated by the macro SF_writemesh defined in file Macros_StabFem.edp)

- file mesh.txt contains the mesh in the native FreeFem format. Namely, information about the vertices, the boundaries and the triangles. This mesh is imported in stabfem using importFFmesm.m , and it is also needed for subsequent Freefem++ programs.
- file mesh.ff2m contains informations about the mesh structure, to be imported by StabFem (it is read by importFFmesm.m)
- file SF_Init.ff2m contains informations about the mesh geometry, to be imported by StabFem as well (it is also read by importFFmesm.m)

[2] P2 data is not yet supported, the solution is to convert everything to P1 before exportation.

- file SF_geom.edp contains definitions of case-dependent geometrical parameters which may be used by the FreeFem++ solvers. This file is designed to be *included* in the header of the FreeFem++ solvers.

Note that files mesh.ff2m and SF_Init.ff2m may seem redundant... the difference is that SF_Init.ff2m is created only once while mesh.ff2m is recreated each time the mesh is modified (adapted, splitted, etc...) .

Chapter 3

Using StabFem for globals stability calculations

The stabfem software was initially designed to perforrm global stability calculations (linear and nonlinear). This chapter explains this throufh the example of the wake of a cylinder.

The theory is fully explained in the submitted paper [...]. In this chapter we explain the implementation issues.

Newt of the chapter is initial version of Diogo,n to be recast)

A *StabFem* project have a directory, for example 🗀`StabFemm>CYLINDER` where it can be found:

1. The `*.m` files corresponding to the *Matlab* scripts. Normally, a main script can be found, e.g. SCRIPT_CYLINDER_DEMO.m ;
2. A directory 🗀`StabFemm>CYLINDER > WORK` (if not, it will be created when *Matlab* script run) where it can be found the output data;
3. All the `*.edp` specific to the project and to be editexecuted automatically by the *Matlab* interface. E.g.: The Mesh*.edp, the Macros_StabFem.edp, Param_Adaptmesh.edp, etc.

3.1 Running logic of a *StabFem* project

When one executes the main script `*.m`, both local and share scripts are executed. For a commum project, the following steps done:

1. Launch the share script SF_Start.m:

```
run('../SOURCES_MATLAB/SF_Start.m');
```

creating the following directories as global variables and adding the *sfdir* to the *matlab* paths:

```
ff = '/PRODCOM/Ubuntu16.04/freefem/3.51/gcc-5.4-mpich_3.2/bin/
    FreeFem++'; % on IMFT network
sfdir = '~/StabFem/SOURCES_MATLAB/'; % where to find the matlab
     drivers
ffdir = '~/StabFem/SOURCES_FREEFEM/'; % where to find the
    freefem scripts
ffdatadir = './WORK/';
addpath(sfdir);
```

It also creates the SF_Geom.edp need for *FreeFemm++*.

2. Then, a mesh and a baseflow are generated for the project with the help of the share script SF_Init.edp located in StabFemm>SOURCES_MATLAB,e.g.:

```
baseflow=SF_Init('Mesh_Cylinder.edp', [-40 80 40]);
```

The detailed input/output parameters are discussed in next chapter (To do...). It will execute Mesh*.edp and generate in the current path the mesh.msh, mesh.ff2m, mesh_init.msh, SF_Init.ff2m, BaseFlow_init.txt and BaseFlow_init.ff2m files.

The path StabFemm>CYLINDER > WORK>BASEFLOWS is created. This is where all the base flow for the different Reynolds numbers will be stored.

3. The baseflows for for different parameters (*Re*, Porosity,...) are generated by the command:

```
baseflow=SF_BaseFlow(baseflow,'Re',10);
```

executing the commom scritpt SF_BaseFlow.m. This script will read the *baseflow.mesh.problemtype* parameter and execute the corresponding Newton routing `Newton*.edp` located at StabFemm>SOURCES_FREE in order to generate the corresponding baseflows in their path.

4. Then, a adaptation of the mesh is made to refine and converge the results for the correct baseflow, with the following command:

```
baseflow=SF_Adapt(baseflow,'Hmax',10,'InterpError',0.005);
```

The refine parameters are written in Param_Adaptmesh.edp(why?) in the current path. Adapt_Mode.edp is executed from *ffdir* to refine the mesh. (Some files are created...explain...)

5. A mesh adaptation taking into account the eigenmode can be done: for that, first a solution have to be commuted first giving the eigenmodes. Then the mesh adaptation can be done like it was done in last step.

```
[ev,em] = SF_Stability(baseflow,'shift',0.04+0.74i,'nev',1,'
    type','D');
[baseflow,em]=SF_Adapt(baseflow,em,'Hmax',10,'InterpError'
    ,0.01);
```

Here, the eigenvalue problem has been solved with a shitf-and-invert iteration process, detail in next chapter((to do)). SF_Stability.m is once again located at *sfdir*.

6. After the former step, once wisely used, post-processing can be made. At that stage, each project have its particularities and it will be detailed in their dedicated chapters.

3.2 Create a *StabFem* project

In order to create a project in *StabFem* one has to start to code the scripts in *FreeFEM++*.

> Attention: When creating the different `*.edp` files, one has to pay attention on the compulsory inputs and output of the *Matlab* interface.

Then, the *Mathlab* scripts, with the previously presented style, have to be created.

> Attention: The different `*.m` files created in the particular directory will be used only in your project, so you can used them as you like; but, once the common files are used, they must not be changed without careful examination of the impact on the other projects.

Commonly, the following files are needed in the current file:

***FreeFEM++* file: SCRIPT_*.edp** Here, the problem is defined. See chapter (to do) for more details. It can be a eigenvalue problem, a forced problem, etc. To define the problem, see the FreeFEM++ documentation [**?**][1].

***FreeFEM++* file: Mesh_*.edp** File where the mesh of the problem and the convenient files are generated.

***FreeFEM++* file: Macros_StabFem.edp** Macros are a powerful tool in *FreeFEM++*. In this file all the Macros specific of the project are created. This macros will be used both by the `*.edp` scripts of one's problem and by the common scripts located at `StabFemm>SOURCES_FREEFEM`.

***Matlab* file: mains_cript.m** This script will be organise like in the previously presented style. Then, the a different treatment is given to each problem, and one can be inspired by the project already created.

[1] http://www.freefem.org/ff++/ftp/freefem++doc.pdf

Chapter 4

Using StabFem to perform DNS (tipe-stepping resolution of NS equations)

Not yet implemented but coming very soon...

Chapter 5

Plotting data with StabFem

5.1 Plotting with StabFem : `plotFF.m`

5.2 Native FreeFem plots (ffglut)

this is enabled/disabled with the MACRO FREEFEMPLOTS

5.3 Interfacing with Tecplot

`exportFF_Tecplot.m`

5.4 Interfacing with Paraview

Javier did that ???

5.5 Alternative idea

The problem with plotFF is that is is built upon pdeplot and requires the pdetools toolbox.

An alternative solution would be to adapt the solution of chloros based on patch, which has the advantage of operating under OCTAVE.

See more there : `https://github.com/samplemaker/freefem_matlab_octave_plot`

Part II

Description of the test-cases

Chapter 6

Stability of the flow around a 2D cylinder

Directory in the StabFem project : `CYLINDER`

Main contributors : D. Fabre, V. Citro, D. Ferreira-Sabino

Reference : A practical review on linear and nonlinear global approaches to stability analysis, D. Fabre et al., *submitted to Rev. Appl. Mech (2018)*

Chapter 7

Stability of the flow around a spring-mounted

Directory in the StabFem project : `CYLINDER_VIV`

Main contributors : Diogo

Reference :

Chapter 8

Stability of the flow around a flying menhir

Directory in the StabFem project : `MENHIR`

Main contributors : Obelix & Cie.

Reference : Asterix et le coup du menhir, R. Goscinny & A. Udezo. Dargaud Ed.

Chapter 9

Capillary oscillations of a liquid bridge

Directory in the StabFem project : `LiquidBridges`

Main contributors : D. Fabre.

Reference : Chireux et al., Phys. Fluids, 2015.

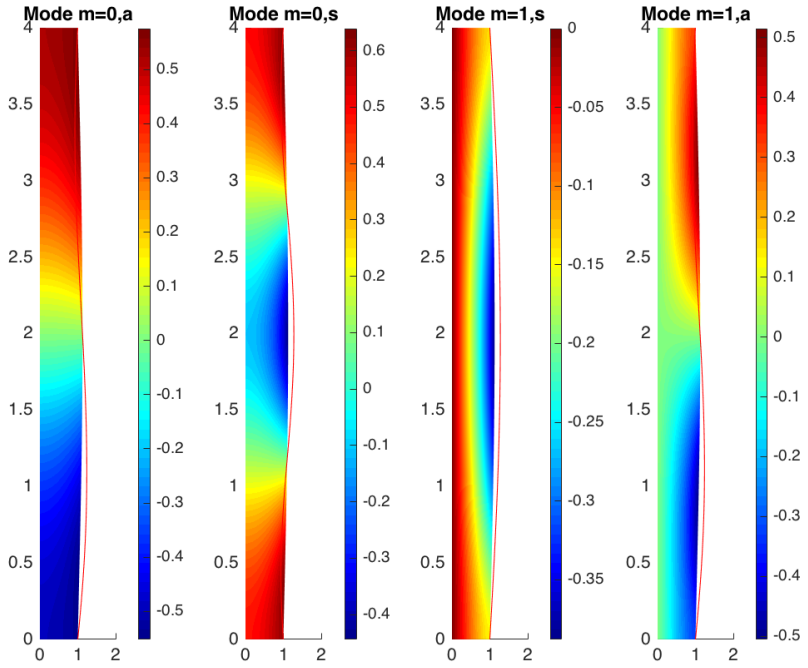

Figure 9.1: Oscillation modes of a liquid bridge of aspect ratio $L/R = 4$ and reduced volume $V^{=}$...

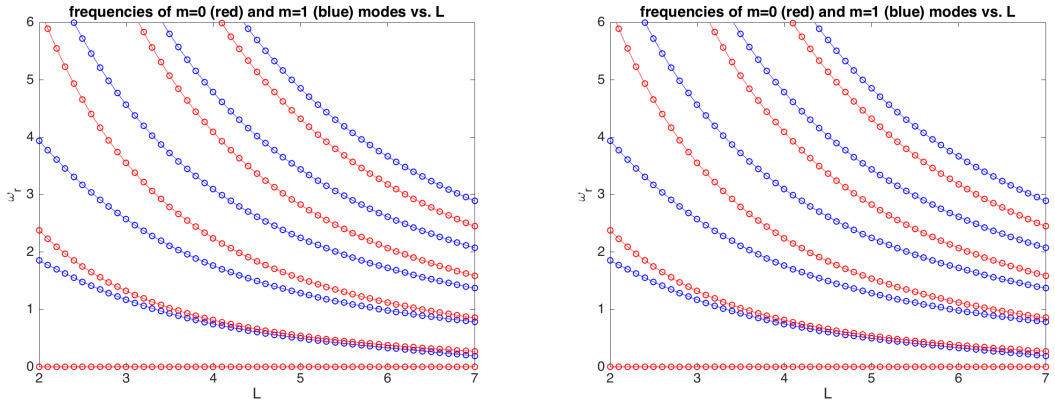

Figure 9.2: Oscillation frequencies of a liquid bridge resulting from the coalescence of two spherical droplets as function of $L^* = L/R$ (figures 11,12 of Chireux et al.)

Chapter 10

Instabilities of a potential free-surface vortex flow

Directory in the StabFem project : `ROTATINGPOLYGONS`

Main contributors : J. Mougel, D. Fabre

Reference : Mougel et al., JFM 2018

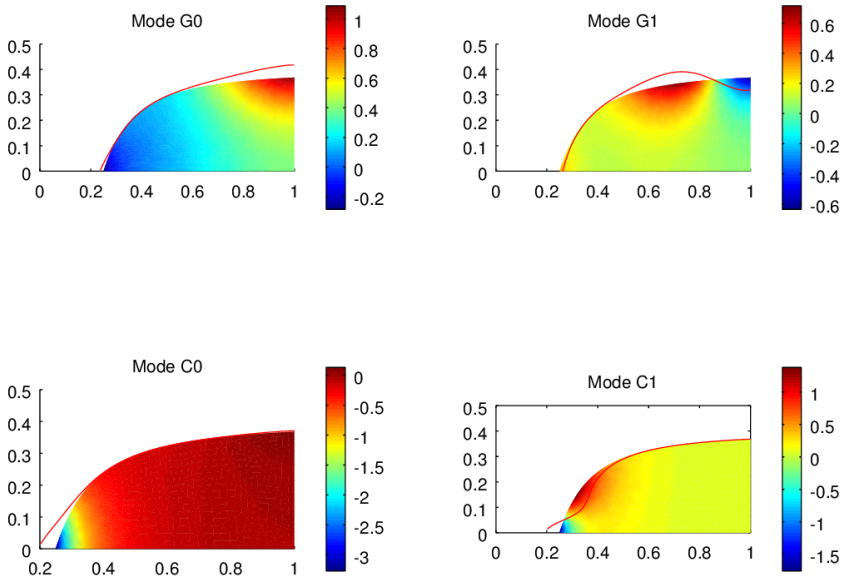

Figure 10.1: Oscillation modes of a potential vortex for $a = H/R = 0.3$ and $m = 3$ (figure 5, 6 of Mougel et al.).

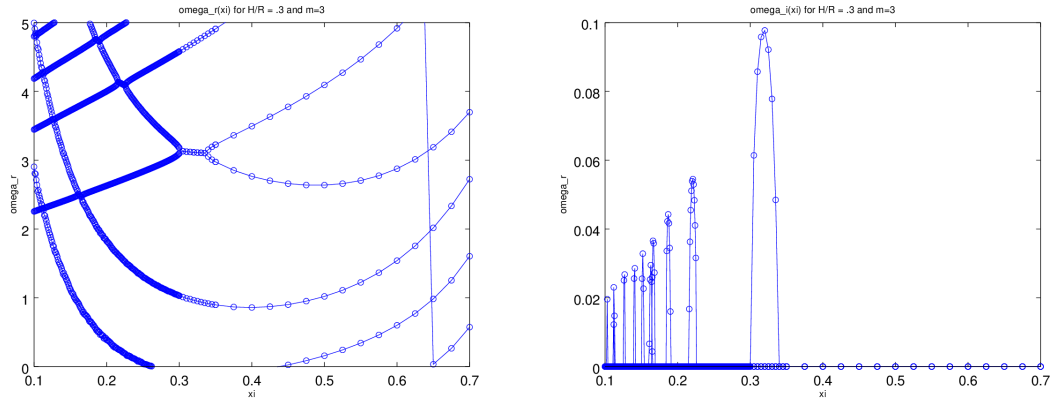

Figure 10.2: Oscillation frequencies and amplification rates for f a potential vortex for $m = 3$ (figure 4 of Mougel et al.).

Part III

Appendix : documentation of the source programs

Appendix A

Mutual Matlab Files

In this section we give the documentation of the Matlab drivers of the project.

We have to find a way to generate automatically this documentation by extracting the documentation directly from the programs (Doxygen ?)

A.1 General drivers and FF/Matlab interface tools

A.1.1 SF_Mesh.m

A.1.2 SF_Launch.m

A.1.3 importFFMesh.m

A.1.4 importFFData.m

A.2 Stability-oriented drivers

A.2.1 SF_Init.m

A.2.2 SF_BaseFlow.m

A.2.3 SF_Stability.m

A.2.4 SF_Adapt.m

A.3 Plotting programs

A.4 Auxiliary programs

A.4.1 mysystem.m

A.4.2 mycp.m

Appendix B

Mutual FreeFEM++ Files