Réunion "StabFem"

D. Fabre

IMFT, groupe Interface

6 juillet 2018

StabFem : présentation du logiciel

Using StabFem for Linear stability

The future of StabFem

StabFem : Cahier des charges

- StabFem : an open-source and easy-to-use software allowing a large range of computations in fluid mechanics.

StabFem : Cahier des charges

- StabFem : an open-source and easy-to-use software allowing a large range of computations in fluid mechanics.
- Initially oriented towards Global Stability Approches (Linear & Nonlinear) but actually allowing a larger number of computations (DNS, linear acoustics, etc...)

StabFem : Cahier des charges

- StabFem : an open-source and easy-to-use software allowing a large range of computations in fluid mechanics.
- Initially oriented towards Global Stability Approches (Linear & Nonlinear) but actually allowing a larger number of computations (DNS, linear acoustics, etc...)
- Multi-platform (Unix, MacOs, Windows) and designed to run on "light" computers (Laptops...)

StabFem : Cahier des charges

- StabFem : an open-source and easy-to-use software allowing a large range of computations in fluid mechanics.
- Initially oriented towards Global Stability Approches (Linear & Nonlinear) but actually allowing a larger number of computations (DNS, linear acoustics, etc...)
- Multi-platform (Unix, MacOs, Windows) and designed to run on "light" computers (Laptops...)
- Easy to use/install,

StabFem : Cahier des charges

- StabFem : an open-source and easy-to-use software allowing a large range of computations in fluid mechanics.
- Initially oriented towards Global Stability Approches (Linear & Nonlinear) but actually allowing a larger number of computations (DNS, linear acoustics, etc...)
- Multi-platform (Unix, MacOs, Windows) and designed to run on "light" computers (Laptops...)
- Easy to use/install,
- Easy to customize to a variety of cases (incompressible, compressible, fixed/free objects, free surfaces,...)

StabFem : Cahier des charges

- StabFem : an open-source and easy-to-use software allowing a large range of computations in fluid mechanics.
- Initially oriented towards Global Stability Approches (Linear & Nonlinear) but actually allowing a larger number of computations (DNS, linear acoustics, etc...)
- Multi-platform (Unix, MacOs, Windows) and designed to run on "light" computers (Laptops...)
- Easy to use/install,
- Easy to customize to a variety of cases (incompressible, compressible, fixed/free objects, free surfaces,...)
- Freeware, based on two softwares : FreeFem++ and ~~Matlab~~

StabFem : Cahier des charges

- StabFem : an open-source and easy-to-use software allowing a large range of computations in fluid mechanics.
- Initially oriented towards Global Stability Approches (Linear & Nonlinear) but actually allowing a larger number of computations (DNS, linear acoustics, etc...)
- Multi-platform (Unix, MacOs, Windows) and designed to run on "light" computers (Laptops...)
- Easy to use/install,
- Easy to customize to a variety of cases (incompressible, compressible, fixed/free objects, free surfaces,...)
- Freeware, based on two softwares : FreeFem++ and ~~Matlab~~
- Developed as a collaborative project (IMFT, Università di Salerno, ONERA, UPFL, ...)

StabFem : Cahier des charges

- StabFem : an open-source and easy-to-use software allowing a large range of computations in fluid mechanics.
- Initially oriented towards Global Stability Approches (Linear & Nonlinear) but actually allowing a larger number of computations (DNS, linear acoustics, etc...)
- Multi-platform (Unix, MacOs, Windows) and designed to run on "light" computers (Laptops...)
- Easy to use/install,
- Easy to customize to a variety of cases (incompressible, compressible, fixed/free objects, free surfaces,...)
- Freeware, based on two softwares : FreeFem++ and ~~Matlab~~
- Developed as a collaborative project (IMFT, Università di Salerno, ONERA, UPFL, ...)
- Maintained on Github

https://github.com/erbafdavid/StabFem

FreeFem++

Les méthodes d'éléments finis sont bien adaptés aux problèmes de stabilité globale.
Le logiciel *FreeFem++* est un outil très populaire dans la communauté de la stabilité hydrodynamique.

- ☺ Syntaxe intuitive basée directement sur la formulation faible.

FreeFem++

Les méthodes d'éléments finis sont bien adaptés aux problèmes de stabilité globale.
Le logiciel *FreeFem++* est un outil très populaire dans la communauté de la stabilité hydrodynamique.

- ☺ Syntaxe intuitive basée directement sur la formulation faible.
- ☺ Mailleur puissant et adaptatif (movemesh, adaptmesh, etc...)

FreeFem++

Les méthodes d'éléments finis sont bien adaptés aux problèmes de stabilité globale.
Le logiciel *FreeFem++* est un outil très populaire dans la communauté de la stabilité hydrodynamique.

- ☺ Syntaxe intuitive basée directement sur la formulation faible.
- ☺ Mailleur puissant et adaptatif (movemesh, adaptmesh, etc...)
- ☺ 2D et 3D.

FreeFem++

Les méthodes d'éléments finis sont bien adaptés aux problèmes de stabilité globale.
Le logiciel *FreeFem++* est un outil très populaire dans la communauté de la stabilité hydrodynamique.

- ☺ Syntaxe intuitive basée directement sur la formulation faible.
- ☺ Mailleur puissant et adaptatif (movemesh, adaptmesh, etc...)
- ☺ 2D et 3D.
- ☺ Grand choix de solveurs séquentiels ou parallèles (Mumps, Petsc/Slepc, ..)

FreeFem++

Les méthodes d'éléments finis sont bien adaptés aux problèmes de stabilité globale.
Le logiciel *FreeFem++* est un outil très populaire dans la communauté de la stabilité hydrodynamique.

- ☺ Syntaxe intuitive basée directement sur la formulation faible.
- ☺ Mailleur puissant et adaptatif (movemesh, adaptmesh, etc...)
- ☺ 2D et 3D.
- ☺ Grand choix de solveurs séquentiels ou parallèles (Mumps, Petsc/Slepc, ..)
- ☹ Interface graphique limitée (ffglut).

FreeFem++

Les méthodes d'éléments finis sont bien adaptés aux problèmes de stabilité globale.
Le logiciel *FreeFem++* est un outil très populaire dans la communauté de la stabilité hydrodynamique.

- ☺ Syntaxe intuitive basée directement sur la formulation faible.
- ☺ Mailleur puissant et adaptatif (movemesh, adaptmesh, etc...)
- ☺ 2D et 3D.
- ☺ Grand choix de solveurs séquentiels ou parallèles (Mumps, Petsc/Slepc, ..)
- ☹ Interface graphique limitée (ffglut).
- ☹ Language interprété : pas adapté à la programmation fonctionnelle (mais des macros puissantes).

FreeFem++

Les méthodes d'éléments finis sont bien adaptés aux problèmes de stabilité globale.
Le logiciel *FreeFem++* est un outil très populaire dans la communauté de la stabilité hydrodynamique.

- ☺ Syntaxe intuitive basée directement sur la formulation faible.
- ☺ Mailleur puissant et adaptatif (movemesh, adaptmesh, etc...)
- ☺ 2D et 3D.
- ☺ Grand choix de solveurs séquentiels ou parallèles (Mumps, Petsc/Slepc, ..)
- ☹ Interface graphique limitée (ffglut).
- ☹ Language interprété : pas adapté à la programmation fonctionnelle (mais des macros puissantes).
- ☹ Syntaxe "chatouilleuse" et débuggage parfois complexe...

Pourquoi interfacer FreeFem++ avec un autre logiciel ?

- Stratégie avant "StabFem" (2010-2017) :

  Chaine de calcul : Solveurs FreeFem++ / Scripts Shell / Post-traitement Tecplot / Gnuplot....

  => 50 programmes quasiment identiques + 10 manières d'effectuer le post-traitement,

  on ne s'y retrouve plus...

Pourquoi interfacer FreeFem++ avec un autre logiciel ?

- Stratégie avant "StabFem" (2010-2017) :

  Chaine de calcul : Solveurs FreeFem++ / Scripts Shell / Post-traitement Tecplot / Gnuplot....

  => 50 programmes quasiment identiques + 10 manières d'effectuer le post-traitement,

  on ne s'y retrouve plus...

- Nécessité d'une surcouche "driver" pour piloter les calculs et tracer les résultats en mode "terminal" ou par l'intermédiaire de scripts.

Pourquoi interfacer FreeFem++ avec un autre logiciel ?

- Stratégie avant "StabFem" (2010-2017) :

  Chaine de calcul : Solveurs FreeFem++ / Scripts Shell / Post-traitement Tecplot / Gnuplot....

  => 50 programmes quasiment identiques + 10 manières d'effectuer le post-traitement,

  on ne s'y retrouve plus...

- Nécessité d'une surcouche "driver" pour piloter les calculs et tracer les résultats en mode "terminal" ou par l'intermédiaire de scripts.
- Idée (à terme) : 1 travail (1 article) = 1 unique script Matlab pour reproduire tous les calculs et générer toutes les figures (cf. Basilisk...)

Pourquoi interfacer FreeFem++ avec un autre logiciel ?

- Stratégie avant "StabFem" (2010-2017) :

  Chaine de calcul : Solveurs FreeFem++ / Scripts Shell / Post-traitement Tecplot / Gnuplot....

  => 50 programmes quasiment identiques + 10 manières d'effectuer le post-traitement,

  on ne s'y retrouve plus...

- Nécessité d'une surcouche "driver" pour piloter les calculs et tracer les résultats en mode "terminal" ou par l'intermédiaire de scripts.
- Idée (à terme) : 1 travail (1 article) = 1 unique script Matlab pour reproduire tous les calculs et générer toutes les figures (cf. Basilisk...)
- Choix actuel : Drivers en Matlab (solution non satisfaisante a terme, pb. licences...)

Articulation FreeFem / Matlab

- Etage 1 : Solveurs FreeFem++ "briques de base".
  Un solveur par "classe de problèmes" (2D incompressible, 2D compressible, Axisymétrique incompressible...) et par "type de calcul" (calcul d'un champ de base, stabilité linéaire, ...)

Articulation FreeFem / Matlab

- Etage 1 : Solveurs FreeFem++ "briques de base".
  Un solveur par "classe de problèmes" (2D incompressible, 2D compressible, Axisymétrique incompressible...) et par "type de calcul" (calcul d'un champ de base, stabilité linéaire, ...)
- Etage 2 : drivers Matlab "génériques"
  Un unique driver pour chaque "type de calcul" , le choix du bon solveur est fait en fonction des paramètres optionnels fournis au driver.

Articulation FreeFem / Matlab

- Etage 1 : Solveurs FreeFem++ "briques de base".
  Un solveur par "classe de problèmes" (2D incompressible, 2D compressible, Axisymétrique incompressible...) et par "type de calcul" (calcul d'un champ de base, stabilité linéaire, ...)
- Etage 2 : drivers Matlab "génériques"
  Un unique driver pour chaque "type de calcul" , le choix du bon solveur est fait en fonction des paramètres optionnels fournis au driver.
- Etage 3 : Les boucles sur les paramètres et la génération des figures sont faites dans un "script principal".

Articulation FreeFem / Matlab

- Etage 1 : Solveurs FreeFem++ "briques de base".
  Un solveur par "classe de problèmes" (2D incompressible, 2D compressible, Axisymétrique incompressible...) et par "type de calcul" (calcul d'un champ de base, stabilité linéaire, ...)
- Etage 2 : drivers Matlab "génériques"
  Un unique driver pour chaque "type de calcul" , le choix du bon solveur est fait en fonction des paramètres optionnels fournis au driver.
- Etage 3 : Les boucles sur les paramètres et la génération des figures sont faites dans un "script principal".
- Les solveurs et les drivers génériques sont dans des répertoires communs **SOURCES_MATLAB/** et **SOURCES_FREEFEM/**,

Articulation FreeFem / Matlab

- Etage 1 : Solveurs FreeFem++ "briques de base".
  Un solveur par "classe de problèmes" (2D incompressible, 2D compressible, Axisymétrique incompressible...) et par "type de calcul" (calcul d'un champ de base, stabilité linéaire, ...)
- Etage 2 : drivers Matlab "génériques"
  Un unique driver pour chaque "type de calcul" , le choix du bon solveur est fait en fonction des paramètres optionnels fournis au driver.
- Etage 3 : Les boucles sur les paramètres et la génération des figures sont faites dans un "script principal".
- Les solveurs et les drivers génériques sont dans des répertoires communs **SOURCES_MATLAB/** et **SOURCES_FREEFEM/**,
- Les programmes spécifiques à chaque cas d'étude sont dans un répertoire spécifique.
  Exemple : le répertoire "CYLINDRE" contient les programmes suivants :

Articulation FreeFem / Matlab

- Etage 1 : Solveurs FreeFem++ "briques de base".
  Un solveur par "classe de problèmes" (2D incompressible, 2D compressible, Axisymétrique incompressible...) et par "type de calcul" (calcul d'un champ de base, stabilité linéaire, ...)
- Etage 2 : drivers Matlab "génériques"
  Un unique driver pour chaque "type de calcul" , le choix du bon solveur est fait en fonction des paramètres optionnels fournis au driver.
- Etage 3 : Les boucles sur les paramètres et la génération des figures sont faites dans un "script principal".
- Les solveurs et les drivers génériques sont dans des répertoires communs **SOURCES_MATLAB/** et **SOURCES_FREEFEM/**,
- Les programmes spécifiques à chaque cas d'étude sont dans un répertoire spécifique.
  Exemple : le répertoire "CYLINDRE" contient les programmes suivants :
  1. Mesh_Cylinder.edp -> Génération du maillage

Articulation FreeFem / Matlab

- Etage 1 : Solveurs FreeFem++ "briques de base".
  Un solveur par "classe de problèmes" (2D incompressible, 2D compressible, Axisymétrique incompressible...) et par "type de calcul" (calcul d'un champ de base, stabilité linéaire, ...)
- Etage 2 : drivers Matlab "génériques"
  Un unique driver pour chaque "type de calcul" , le choix du bon solveur est fait en fonction des paramètres optionnels fournis au driver.
- Etage 3 : Les boucles sur les paramètres et la génération des figures sont faites dans un "script principal".
- Les solveurs et les drivers génériques sont dans des répertoires communs **SOURCES_MATLAB/** et **SOURCES_FREEFEM/**,
- Les programmes spécifiques à chaque cas d'étude sont dans un répertoire spécifique.
  Exemple : le répertoire "CYLINDRE" contient les programmes suivants :
  1. Mesh_Cylinder.edp -> Génération du maillage
  2. Macros_StabFem.edp -> Macros case-dependant (conditions limites et post-traitement)

Articulation FreeFem / Matlab

- Etage 1 : Solveurs FreeFem++ "briques de base".
  Un solveur par "classe de problèmes" (2D incompressible, 2D compressible, Axisymétrique incompressible...) et par "type de calcul" (calcul d'un champ de base, stabilité linéaire, ...)
- Etage 2 : drivers Matlab "génériques"
  Un unique driver pour chaque "type de calcul" , le choix du bon solveur est fait en fonction des paramètres optionnels fournis au driver.
- Etage 3 : Les boucles sur les paramètres et la génération des figures sont faites dans un "script principal".
- Les solveurs et les drivers génériques sont dans des répertoires communs **SOURCES_MATLAB/** et **SOURCES_FREEFEM/**,
- Les programmes spécifiques à chaque cas d'étude sont dans un répertoire spécifique.
  Exemple : le répertoire "CYLINDRE" contient les programmes suivants :
  1. Mesh_Cylinder.edp -> Génération du maillage
  2. Macros_StabFem.edp -> Macros case-dependant (conditions limites et post-traitement)
  3. SCRIPT_CYLINDER.m -> Script "Maitre".

Articulation FreeFem / Matlab

- Etage 1 : Solveurs FreeFem++ "briques de base".
  Un solveur par "classe de problèmes" (2D incompressible, 2D compressible, Axisymétrique incompressible...) et par "type de calcul" (calcul d'un champ de base, stabilité linéaire, ...)
- Etage 2 : drivers Matlab "génériques"
  Un unique driver pour chaque "type de calcul" , le choix du bon solveur est fait en fonction des paramètres optionnels fournis au driver.
- Etage 3 : Les boucles sur les paramètres et la génération des figures sont faites dans un "script principal".
- Les solveurs et les drivers génériques sont dans des répertoires communs **SOURCES_MATLAB/** et **SOURCES_FREEFEM/**,
- Les programmes spécifiques à chaque cas d'étude sont dans un répertoire spécifique.
  Exemple : le répertoire "CYLINDRE" contient les programmes suivants :
  1. Mesh_Cylinder.edp -> Génération du maillage
  2. Macros_StabFem.edp -> Macros case-dependant (conditions limites et post-traitement)
  3. SCRIPT_CYLINDER.m -> Script "Maitre".

Articulation FreeFem / Matlab

- Etage 1 : Solveurs FreeFem++ "briques de base".
  Un solveur par "classe de problèmes" (2D incompressible, 2D compressible, Axisymétrique incompressible...) et par "type de calcul" (calcul d'un champ de base, stabilité linéaire, ...)
- Etage 2 : drivers Matlab "génériques"
  Un unique driver pour chaque "type de calcul" , le choix du bon solveur est fait en fonction des paramètres optionnels fournis au driver.
- Etage 3 : Les boucles sur les paramètres et la génération des figures sont faites dans un "script principal".
- Les solveurs et les drivers génériques sont dans des répertoires communs **SOURCES_MATLAB/** et **SOURCES_FREEFEM/**,
- Les programmes spécifiques à chaque cas d'étude sont dans un répertoire spécifique.
  Exemple : le répertoire "CYLINDRE" contient les programmes suivants :
  1. Mesh_Cylinder.edp -> Génération du maillage
  2. Macros_StabFem.edp -> Macros case-dependant (conditions limites et post-traitement)
  3. SCRIPT_CYLINDER.m -> Script "Maitre".

Remarques : les programmes FreeFem doivent pouvoir être utilisés directement en dehors du driver StabFem, notamment pour faciliter le développement/débuggage...)

Les contributeurs "utilisateurs" (ex. étudiant M1/M2) ne travaillent qu'à l'étage 3 et ne devraient travailler que sur ces 3 fichiers.

Les contributeurs "développeurs" travaillent aux étages inférieurs...

Format d'échange des données

- Format de fichier d'échange ".ff2m", généré par FreeFem++ et relu par Matlab

Format d'échange des données

- Format de fichier d'échange ".ff2m", généré par FreeFem++ et relu par Matlab
- Example d'en-tête d'un fichier .ff2m :

```
1 ### Data generated by Freefem++ ;
2 Temperature
3 Format :
4 P1 T
```

Ligne 4 : "TypeField1 NameField1 TypeField2 NameField2... " TypeField can be "real" (scalar data), "real.N" (vectorial data), "P1" (data associated to mesh), ...

Format d'échange des données

- Format de fichier d'échange ".ff2m", généré par FreeFem++ et relu par Matlab
- Example d'en-tête d'un fichier .ff2m :

```
1 ### Data generated by Freefem++ ;
2 Temperature
3 Format :
4 P1 T
```

Ligne 4 : "TypeField1 NameField1 TypeField2 NameField2... " TypeField can be "real" (scalar data), "real.N" (vectorial data), "P1" (data associated to mesh), ...

- Le fichier est lu et importé sous forme d'une *structure matlab*

Format d'échange des données

- Format de fichier d'échange ".ff2m", généré par FreeFem++ et relu par Matlab
- Example d'en-tête d'un fichier .ff2m :

```
1 ### Data generated by Freefem++ ;
2 Temperature
3 Format :
4 P1 T
```

Ligne 4 : "TypeField1 NameField1 TypeField2 NameField2... " TypeField can be "real" (scalar data), "real.N" (vectorial data), "P1" (data associated to mesh), ...

- Le fichier est lu et importé sous forme d'une *structure matlab*
- Illustration : cas "EXAMPLE_Lshape"

StabFem : présentation du logiciel

Using StabFem for Linear stability

The future of StabFem

Stabilité globale : qu'es aquò ? ?

Les problèmes d'instabilités sont omniprésents en mécanique des fluides.

Leur résolution fait appel à une classe de méthodes numériques spécifiques, complémentaires aux approches de simulation directe, qui sont actuellement en plein développement.

On parle *stabilité globale* quand la géométrie du problème necessite une résolution en 2D (ou en 3D).

(par opposition à *stabilité locale* quand des directions d'invariance (écoulement parallèle par ex.) permettent de rammener le problème à une résolution en 1D).

Stabilité globale : liste des types de problèmes et cas-tests

- CYLINDER -> 2D incompressible, objet fixe
- CYLINDER_VIV -> 2D incompressible, objet mobile
  (Stage Diogo Ferrera-Sabino)
- IMPACTINGJET -> 2D incompressible, 3D stability.
  (with David LoJacono)
- CYLINDER_Compressible -> 2D compressible
  (Javier Serra, Vincenzo Citro...)
- BIRDCALL -> 2D-axisymmetric, incompressible or "augmented incompressible"
  (with R. Longobardi, V. Citro....)
- POROUS_DISK -> 2D-axisymmetric, with porous object
  (stage Adrien Rouvière)
- LiquidBridges -> 2D axi, with deformable free surface
  (stage Nabil Achour)
- ROTATING_POLYGONS
  (with Jérôme Mougel...)
- ....

=> Illustration dans le cas CYLINDER

First step : Generation of a mesh and "guess" base flow

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

What the `SF_Init` driver does :

- Runs the relevant FreeFem++ program `Mesh_Cylinder.edp` with the corresponding parameters (here size of the domain),

  This program generates the following output files : `mesh.msh` (mesh data), `mesh.ff2m` (mesh information), `SF_Init.ff2m` (auxiliary information), `BaseFlow_init.txt` and `BaseFlow_init.ff2` ("guess" base flow).

First step : Generation of a mesh and "guess" base flow

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

What the `SF_Init` driver does :

- Runs the relevant FreeFem++ program `Mesh_Cylinder.edp` with the corresponding parameters (here size of the domain),

  This program generates the following output files : `mesh.msh` (mesh data), `mesh.ff2m` (mesh information), `SF_Init.ff2m` (auxiliary information), `BaseFlow_init.txt` and `BaseFlow_init.ff2` ("guess" base flow).

- Reads all the output files,

First step : Generation of a mesh and "guess" base flow

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

What the `SF_Init` driver does :

- Runs the relevant FreeFem++ program `Mesh_Cylinder.edp` with the corresponding parameters (here size of the domain),

  This program generates the following output files : `mesh.msh` (mesh data), `mesh.ff2m` (mesh information), `SF_Init.ff2m` (auxiliary information), `BaseFlow_init.txt` and `BaseFlow_init.ff2` ("guess" base flow).

- Reads all the output files,
- Returns a matlab "structure" object containing all the data needed for post-processing and subsequent usage.

Computation of a Base flow : principle

We look for a steady base-flow $(\mathbf{u}_b; p_b)$ satisfying the steady Navier-Stokes equations, i.e. $NS(\mathbf{u}_b, p_b) = 0$.
Suppose that we have a 'guess' for the base flow $[\mathbf{u}_b^g, p_b^g]$ which almost satisfies the equations. We look for a better approximation under the form

$$[\mathbf{u}_b, p_b] = [\mathbf{u}_b^g, p_b^g] + [\delta\mathbf{u}_b, \delta p_b] = 0. \tag{1}$$

Injecting into the Navier-Stokes equation lead to

$$NS(\mathbf{u}_b^g, p_b^g) + NSL_{\mathbf{u}_b^g}(\delta\mathbf{u}_b, \delta p_b)$$

Where NSL is the linearised Navier-Stokes operator.
$=>$ matricial problem with the form $A \cdot \delta X = Y$. The procedure of Newton iteration is to solve iteratively this set of equations up to convergence.

Computation of a Base flow : implementation

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

What the `SF_Init` driver does :

- Copies the previous base flow into file `BaseFlow_guess.txt` which will be read by Freefem++,

Computation of a Base flow : implementation

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

What the `SF_Init` driver does :

- Copies the previous base flow into file `BaseFlow_guess.txt` which will be read by Freefem++,
- Runs the relevant FreeFem++ solver (here `Newton_2D.edp`) with the corresponding parameters (here the value of Re),

Computation of a Base flow : implementation

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

What the `SF_Init` driver does :

- Copies the previous base flow into file `BaseFlow_guess.txt` which will be read by Freefem++,
- Runs the relevant FreeFem++ solver (here `Newton_2D.edp`) with the corresponding parameters (here the value of Re),
- Reads all the generated output files (here `BaseFlow.ff2m`),

Computation of a Base flow : implementation

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

What the `SF_Init` driver does :

- Copies the previous base flow into file `BaseFlow_guess.txt` which will be read by Freefem++,
- Runs the relevant FreeFem++ solver (here `Newton_2D.edp`) with the corresponding parameters (here the value of Re),
- Reads all the generated output files (here `BaseFlow.ff2m`),
- Returns a matlab "structure" object containing all the data needed for post-processing and subsequent usage.

Mesh adaptation

Linear stability

$$\mathbf{u} = \mathbf{u}_b + \epsilon \hat{\mathbf{u}} e^{\lambda t} \tag{2}$$

The eigenmodes is governed by the linear problem

$$\lambda \hat{\mathbf{u}} = NSL_{\mathbf{u}_b}(\hat{\mathbf{u}}, \hat{p})$$

,
After discretization we end up with an eigenvalue problem with the matricial form

$$\lambda B \hat{X} = A \hat{X} \tag{3}$$

Iterative method : single-mode shift-invert iteration

$$X^n = (A - \lambda_{shift} B)^{-1} B X^{n-1}$$

Generalization : Arnoldi

Eigenvalue computation : implementation

`SF_Stability(bf,'shift',0.04` + `0.74i,'nev',1,' type ' , 'D' ) ;`

What the `SF_Stability` driver does :

- Copies the base flow into file `BaseFlow.txt` which will be needed by Freefem++,

Eigenvalue computation : implementation

`SF_Stability(bf,'shift',0.04` + `0.74i,'nev',1,' type ' , 'D' ) ;`

What the `SF_Stability` driver does :

- Copies the base flow into file `BaseFlow.txt` which will be needed by Freefem++,
- Runs the FreeFem++ solver (here `Stab_2D.edp`) with the corresponding parameters (shift, number of eigenvalues, direct eigenmode),

Eigenvalue computation : implementation

`SF_Stability(bf,'shift',0.04` + `0.74i,'nev',1,' type ' , 'D' ) ;`

What the `SF_Stability` driver does :

- Copies the base flow into file `BaseFlow.txt` which will be needed by Freefem++,
- Runs the FreeFem++ solver (here `Stab_2D.edp`) with the corresponding parameters (shift, number of eigenvalues, direct eigenmode),
- Reads all the generated output files (here `Eigenmode.ff2m`),

Eigenvalue computation : implementation

`SF_Stability(bf,'shift',0.04` + `0.74i,'nev',1,' type ' , 'D' ) ;`

What the `SF_Stability` driver does :

- Copies the base flow into file `BaseFlow.txt` which will be needed by Freefem++,
- Runs the FreeFem++ solver (here `Stab_2D.edp`) with the corresponding parameters (shift, number of eigenvalues, direct eigenmode),
- Reads all the generated output files (here `Eigenmode.ff2m`),
- Returns a matlab "structure" object containing all the data needed for post-processing and subsequent usage.

StabFem : présentation du logiciel

Using StabFem for Linear stability

The future of StabFem

Recent progress

- Multi-platform objective : MacOs OK ; Unix OK ; Windows 10 currently 50 % compatible.
  main issues with windows : cp = copy,...
- Plotting options : recent intergration of "pdeplot2dff" from Markus "chloros" in place of pdeplot/pdetools .
  other solutions for plotting : tecplot converter, vtk converter, ...
- Compatibility with Octave : currently 50 % compatible.
  Main issues with octave : importdata, plotting (now solved), inputParser (now solved).
- Translation in Python ? ?

Besoins

- Maintaining a fully opensource (Matlab-Octave or Python ?) and fully multiplatform version (windows).
- Managing a list of test cases (non-regression tests, etc...)
- Help simplifying/rationalizing the programation style.
- Gestion of errors / debugging / "verbosity" ...
- Upgrading to 3D / parallel computation ? (currently not priority)
- Support with github (/ gitlab ?)
- Documentation
  Automatic generation from comments in programs ?
  Doxygen ? ?