Popularizing linear and nonlinear approaches to global flow instabilities : A review and a simple implementation for the wake of a cylinder (and of a sphere ?)

D. Fabre[(a)], P. Bonnefis[(a)], V. Citro[(b)] & F. Giannetti[(b)]

[a] *IMFT, University of Toulouse*
[b] *University of Salerno*

Abstract

This template helps you to create a properly formatted LaTeX manuscript.

Keywords: Global instabilities, wake instabilities, finite elements

1. Introduction

The destabilization of a flow as a parameter is varied, leading from a steady solution to a unsteady one, is an ubiquous situation in fluid dynamics. Specific analytical and numerical methods suited to the description have emerged in the second part of the 19th century and continuously evolved up to the present day. When the flow has no analytical expression, the stability problem requires a numerical resolution, and when there are at least two spatial variables, the class of methods suited to solve such problems are generally called "global stability approaches."[1]

Among all instability problems, one of the most famous and most studied is the flow around a 2D cylinder, which becomes unsteady for $Re \approx 47$ leading to the well-known Bnard-Von Karman vortex alley. This situation has served as a benchmark in the development of this class of methods. If one is only interested in predicting the stability or instability of a flow, it is enough to conduct an *linear stability analysis* which is the fundamental brick of global stability approaches. Beyond this simple question, in the past two decades, a number of extensions have been developed and popularized. *Adjoint methods* are an important extension (Giannetti & Luchini ; Marquet et al) ; they can give insight into the sensitivity of the flow to intrinsic or extrinsic contributions. *Nonlinear stability approaches* (Sipp & Lebedev; Mantic-Lugo et al) have also been developed in order to extend the range of applicability of the approach towards large amplitude perturbations.

[1] Sorry Laurette !

The objective of the present work is to contribute to the popularization of such methods in two ways.

First, we give a concise but self-contained exposition of the main concepts and specific numerical methods pertaining to global stability, including basic linear stability, adjoint-based sensitivity, as well as the most recent nonlinear developments.

Secondly we offer an open-source and user-friendly software called "StabFem" to perform such calculations. The software combines program written in both FreeFem++ and Matlab languages. FreeFem++ is used to generate and adapt the meshes and to solve the various linear problems arising in the analysis. Matlab is used as a driver to monitor the computations, perform the required loops over parameters, and plot the results.

In the present paper the concepts are introduced and the software is demonstrated for the reference case of the incompressible, two-dimensional flow around a cylinder, but the software is easily customizable to a variety of other situations (compressible, three-dimensional, etc..).

Although we don't claim to invent any radically new method, our exposition and implementation contains a number of originalities making the computation particularly efficient in terms of computational time and memory (all the figures of the paper can be produced in only a few minutes on a standard laptop). The most notable originalities are the systematic use of mesh adaptation (§2 and 3), the use of simple shift-invert instead of Arnoldi (§3), and a reformulation and simplification of the nonlinear self-consistent approach of Mantic-Lugo et al in terms of a harmonic balance (§4).

2. Computing a base-flow with Newton iteration

Governing equations. Navier-Stokes equations :

$$\partial_t \mathbf{u} = \mathcal{NS}(\mathbf{u}) \equiv \mathbf{u}\cdot\nabla\mathbf{u} - \nabla p + 2\mu\nabla\cdot\mathbf{D}(\mathbf{u}) \qquad \text{with the constraint } \nabla\cdot\mathbf{u} = 0. \quad (1)$$

$\mathbf{D}(\mathbf{u})$ is the rate-of-strain tensor defined as

$$\mathbf{D}(\mathbf{u}) = 1/2\left(\nabla\mathbf{u} + \nabla^T\mathbf{u}\right)$$

Note that the notation $\mathcal{NS}(\mathbf{u})$ is a shortcut as it is defined as an operator acting on the velocity field $\mathbf{u}$, not on the full flow field $(\mathbf{u}, p)$.

Base-flow equations. We look for a steady base-flow $(\mathbf{u}_b; p_b)$ satisfying the Navier-Stokes equations :

$$\mathcal{NS}(\mathbf{u}_b) = 0 \quad (2)$$

Newton iteration. Suppose that we have a 'guess' for the base flow $\mathbf{u}_b^g$ which almost satisfies the equations. We look for a better approximation under the form

$$\mathbf{u}_b = \mathbf{u}_b^g + \delta\mathbf{u}_b \tag{3}$$

Injecting into the equation and developing up to linear order in the perturbations leads to :

$$\mathcal{NSL}(\mathbf{u}_b^g) + \mathcal{NSL}_{\mathbf{u}_b^g}(\delta\mathbf{u}_b) = 0 \tag{4}$$

Where $\mathcal{NSL}$ is the linearised Navier-Stokes operator, defined by its action on a flow field $(\mathbf{u}; p)$ as follows :

$$\mathcal{NSL}_{\mathbf{U}}(\mathbf{u}; p) = \mathcal{C}(\mathbf{U}, \mathbf{u}) - \nabla p + 2\nu\nabla\cdot\mathbf{D}(\mathbf{u}) \quad (\text{ with } \nabla\cdot\mathbf{u} = 0\,). \tag{5}$$

C is the convection operator defined by

$$\mathcal{C}(\mathbf{U}, \mathbf{u}) = \mathbf{U}\cdot\nabla\mathbf{u} + \mathbf{u}\cdot\nabla\mathbf{U} \tag{6}$$

The problem can be set into weak form by multiplying by multiplying Eq. 4 by a test function $\mathbf{v}$ and the associated divergence constraint by a test function q. After a few integration by parts we are led to :

$$\begin{aligned}\forall(\mathbf{v}; q), \qquad & \int [\mathbf{v}\cdot\mathcal{C}(\mathbf{u}_b^g, \delta\mathbf{u}_b) + \delta\mathbf{u}_b\cdot\nabla q - \mathbf{v}\cdot\nabla\delta p_b + 2\mu\mathbf{D}(\delta\mathbf{u}_b) : \mathbf{D}(\delta\mathbf{v})] \\ & + \int [\mathbf{v}\cdot(\mathbf{u}_b^g\cdot\nabla\mathbf{u}_b^g) + \mathbf{u}_b^g\cdot\nabla q - \mathbf{v}\cdot\nabla p_b^g + 2\mu\mathbf{D}(\mathbf{u}_b^g) : \mathbf{D}(\delta\mathbf{v})] = 0\end{aligned} \tag{7}$$

This problem can now be discretized by projecting upon a basis of Taylor-Hood (P2-P2-P1) finite elements. Noting δX the discretization of $(\delta\mathbf{u}_b; \delta p_b)$ this eventually leads to a 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.

In our implementation, the algorithm is written in the Freefem++ solver *Newton_2D.edp* which is wrapped by the Matlab driver *Freefem_BaseFlow.m*.

Mesh adaptation procedure. As for any numerical method, a crucial point in the numerical efficiency is the design of the mesh. The finite element method allows to use unstructured mesh and hence to locally adapt the refinement. Following Sipp Lebedev, the usual procedure is to decompose the domain into several parts with different grid densities ; for instance for the wake of a cylinder, we will design a near-wall region with very small size, a "wake" region with intermediate mesh size, and an outer region with large mesh size. The inconvenient is that the design relies on an a priori expectation of the regions where gradients will be large.

In our implementation, we used an automatic mesh adaptation procedure which provides an optimal mesh adapted to the structure of the flow. The implementation relies on the AdaptMesh procedure of the FreeFem++ software. This procedure is detailed in detail in ref. []. In short, the classical Delaunay-Voronoi algorithm produces a mesh with gridpoint distribution specified by a

Metric matrix $\mathcal{M}$. The AdaptMesh algorithm consists of using as a metric the *Hessian* (second-order spatial derivative) of an objective function u_h defined over the domain, i.e. $\mathcal{M} = \nabla\nabla u_h$. The precision can be controlled by specifying an objective value for the interpolation error of the function on the new mesh.

To build an optimal mesh for the base-flow calculation, the idea is to use as the objective function u_h the solution $\mathbf{u}_b$ itself, as computed on a previous mesh.The base flow is then recomputed on the adapted mesh, providing a better approximation of the solution. The procedure can be repeated a few steps to ensure a right convergence. In our implementation, the whole process is performed using the Matlab driver *Freefem_AdaptMesh.m*.

Example for the cylinder. We illustrate the procedure in the case of a cylinder (see program *SCRIPT_CYLINDER_ADAPTMESH_BASEFLOW*)

First we build an initial mesh, and compute base flow solutions for increasing values of the Reynolds number up to $Re = 60$:

```
> baseflow=FreeFem_Init('Mesh_Cylinder_Large.edp');
    ### INITIAL MESH CREATED WITH np = 2207 points
> baseflow=FreeFem_BaseFlow(baseflow,'Re',1);
    ### FUNCTION FreeFem_BaseFlow : computing base flow for Re = 1
    # Base flow converged in 6 iterations ; Drag = 5.9876; Lx = 0.501
> baseflow=FreeFem_BaseFlow(baseflow,'Re',10);
    ### FUNCTION FreeFem_BaseFlow : computing base flow for Re = 10
    # Base flow f converged in 5 iterations ; Drag = 1.4519; Lx = 0.73191
> baseflow=FreeFem_BaseFlow(baseflow,'Re',60);
    ### FUNCTION FreeFem_BaseFlow : computing base flow for Re = 60
    # Base flow converged in 6 iterations ; Drag = 0.66348; Lx = 3.6989
```

Then we perform the mesh adaptation. The following listing shows the output of the software. Note the values h_{min} and h_{max} of the smaller and larger edges, as well as the local grid size at four points A,B,C,D defined as $(x_A, y_A) = (0.5, 0)$ (at the surface of the cylinder at the position of maximum shear), $(x_B, y_B) = (0.5, 2.5)$ (at the location of the peak of structural sensitivity, see next section), $(x_C, y_C) = (0., 4)$ (in the near wake), and $(x_D, y_D) = (0., 10)$ (in the the far wake).

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

    ### ADAPT mesh to base flow ; InterpError = 0.01  ; Hmax = 10
    #   Number of points np = 1352 ; Ndof = 11862
    #  h_min, h_max : 0.018824 , 10.2895
    #  h_(A,B,C,D) : 0.027467 , 0.28644 , 0.45948 , 0.89245
    ### FUNCTION FreeFem_BaseFlow : computing base flow for Re = 60
    # Base flow converged in 5 iterations ; Drag = 0.65805; Lx = 4.0737
```

The resulting mesh and base flow are plotted on figure 1.

Base flow characteristics.

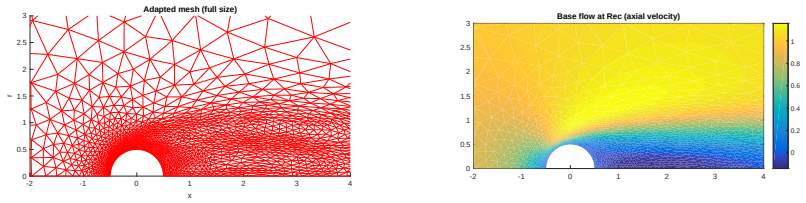


Figure 1: Adapted mesh (a) and base flow (axial velocity component) (b) for the flow over a cylinder at $Re = Re_c = 46.7$.

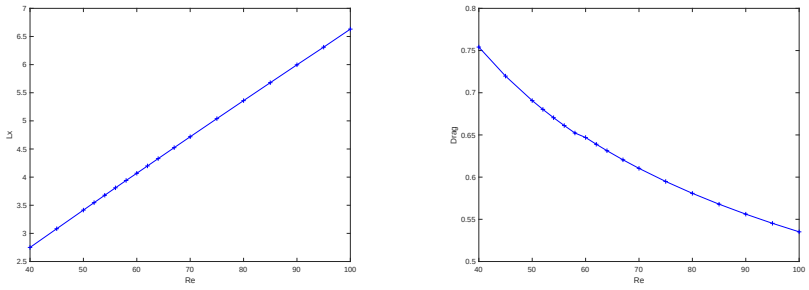


Figure 2: Recirculation length L_x (a) and drag C_x (b) of the base flow over a cylinder at $Re = Re_c = 46.7$.

3. Computing an eigenmode with a shift-invert method

We consider the stability using the classical ansatz
$\mathbf{u} = \mathbf{u}_b + \hat{\mathbf{u}} e^{\lambda t}$
Where $\lambda = \sigma + i\omega$ is the eigenvalue, σ the amplification rate, ω the oscillation rate, $\hat{\mathbf{u}}$ the eigenmode.
The eigenmode is governed by the linear problem
After discretization we end up

Stability equations. Eigenvalue problem
Adjoint eigenvalue problem
Sensitivity

Iterative methods. These de Andrea Fani

Implementation. Matlab driver...

3.1. Results

3.2. Application

4. Computing the limit cycle for $Re > Re_c$ with Harmonic Balance

Appendix A. Appendix : details on mesh convergence

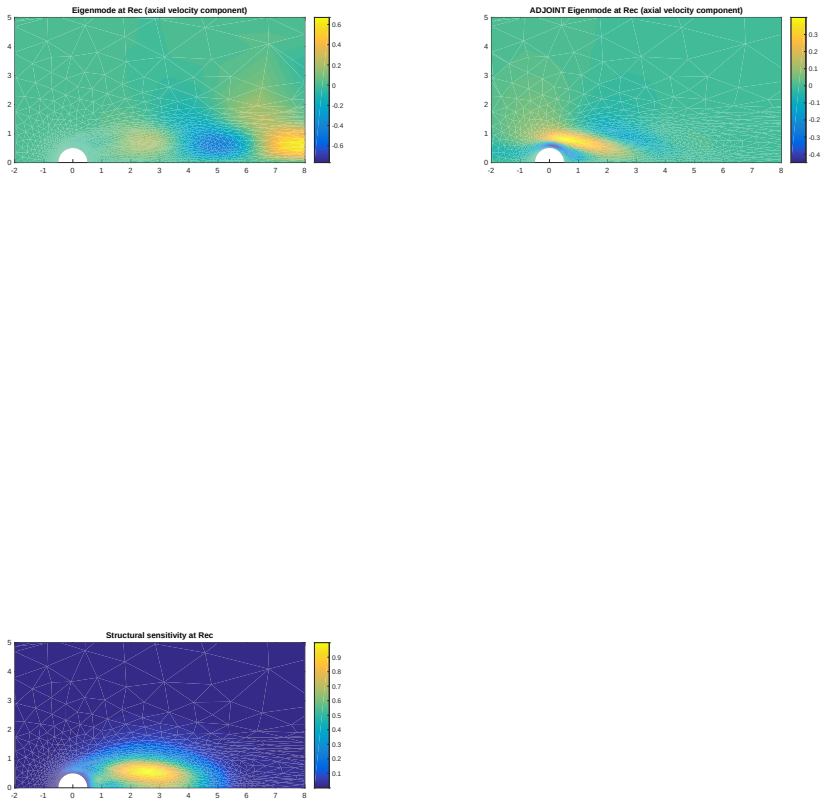


Figure 3: Eigenmode (a) (axial velocity component), Adjoint eigenmode (b) and structural sensitivity (c) for the cylinder's wake at $Re = Re_c = 46.7$.

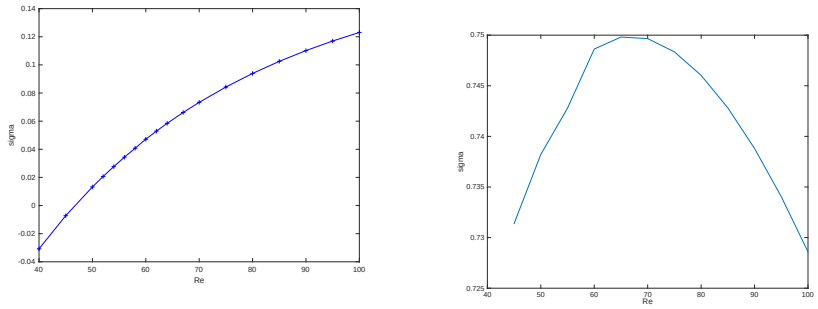
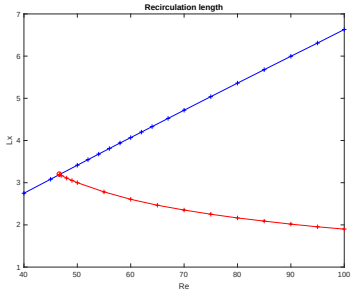
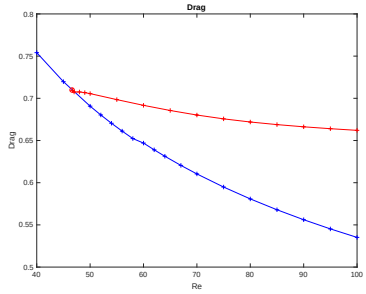
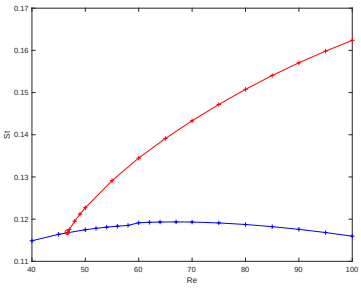
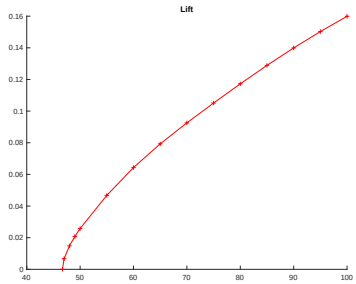


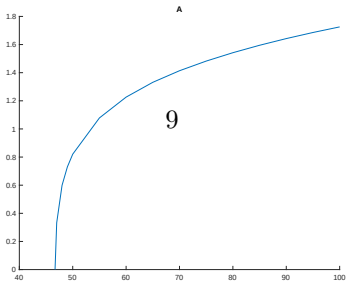
Figure 4: Growth rate σ (a) and Strouhal number $St = (a\omega)/(2\pi U_0)$ (b) as function of Reynolds number


Recirculation length
Lx
Re


Drag
Drag
Re


St
Re


Lift


A

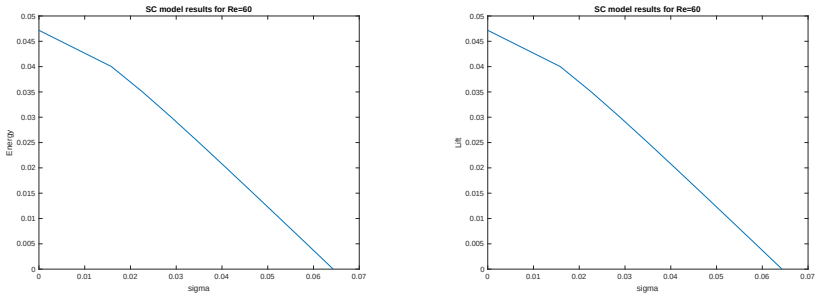


Figure 6: SELF CONSISTENT APPROACH FOR THE WAKE OF A CYLINDER, $Re = 60$