Popularizing linear and nonlinear global approaches to hydrodynamic instabilities : A review and a simple implementation for the wake of a cylinder

D. Fabre[(a)] **et al.**
IMFT, University of Toulouse

This is the abstract. This article illustrates preparation of ASME paper using LaTeX2ε. An abstract for an ASME paper should be less than 150 words and is normally in italics. Please use this template to test how your figures will look on the printed journal page of the Journal of Mechanical Design. The Journal will no longer publish papers that contain errors in figure resolution. These usually consist of unreadable or fuzzy text, and pixilation or rasterization of lines. This template identifies the specifications used by JMD some of which may not be easily duplicated; for example, ASME actually uses Helvetica Condensed Bold, but this is not generally available so for the purpose of this exercise Helvetica is adequate. However, reproduction of the journal page is not the goal, instead this exercise is to verify the quality of your figures. Notice that this abstract is to be set in 9pt Times Italic, single spaced and right justified.

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.

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).

[1] Sorry Laurette !

2 Linear stability analysis : equations and methods

2.1 Computing a base-flow with Newton iteration

Navier-Stokes equations and weak form We start from the general problem of a flow field $[\mathbf{u}, p]$ satisfying the incompressible Navier-Stokes equations on a domain Ω,

$$\partial_t \mathbf{u} = NS(\mathbf{u}; p) \equiv -\mathbf{u}\cdot\nabla\mathbf{u} - \nabla p + \frac{2}{Re}\nabla\cdot\mathbf{D}(\mathbf{u}) \quad [\text{with} \nabla\cdot\mathbf{u} = 0]. \tag{1}$$

with suitable boundary conditions on the frontier $\partial\Omega$ of the domain. Here $\mathbf{D}(\mathbf{u})$ is the rate-of-strain tensor defined as

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

In the framework of finite element methods, it is convenient to write the equation in weak form. Prior to this we define a scalar product as follows, for either scalar or vectorial quantities (ϕ_1, ϕ_2):

$$\langle \phi_1, \phi_2 \rangle = \int_\Omega \overline{\phi_1}\cdot\phi_2 \, \mathrm{d}\Omega$$

The weak form of the Navier-Stokes equations is readily defined by introducing test functions $[\mathbf{v}, q]$ associated with the momentum and continuity equations, and integrating over the domain[2]

$$\forall[\mathbf{v}, q], \quad \partial_t \langle \mathbf{v}, \mathbf{u} \rangle = \langle \mathbf{v}, NS(\mathbf{u}; p) \rangle + \langle q, \nabla\cdot\mathbf{u} \rangle . \tag{2}$$

Newton iteration We look for a steady base-flow $(\mathbf{u}_b; p_b)$ satisfying the steady Navier-Stokes equations, i.e. $\mathcal{NS}(\mathbf{u}_b, p_b) = 0$. 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

$$[\mathbf{u}_b, p_b] = [\mathbf{u}_b^g, p_b^g] + [\delta\mathbf{u}_b, \delta p_b]. \tag{3}$$

Injecting into the Navier-Stokes equation in weak form (2) and developing up to linear terms in terms of the perturbation lead to $NS(\mathbf{u}_b^g, p_b^g) + NSL_{\mathbf{u}_b^g}(\delta\mathbf{u}_b, \delta p_b)$, which can also be written in weak form :

$$\begin{aligned} &\left\langle \mathbf{v}, NS(\mathbf{u}_b^g) \right\rangle + \left\langle q, \nabla\cdot\mathbf{u}_b^g \right\rangle \\ &+ \left\langle \mathbf{v}, NSL_{\mathbf{u}_b^g}(\delta\mathbf{u}_b, \delta p_b) \right\rangle + \langle q, \nabla\cdot\delta\mathbf{u}_b \rangle = 0. \end{aligned} \tag{4}$$

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

$$NSL_{\mathbf{U}}(\mathbf{u}; p) = -\mathcal{C}(\mathbf{U}, \mathbf{u}) - \nabla p + \frac{2}{Re}\nabla\cdot\mathbf{D}(\mathbf{u}), \tag{5}$$

and C is the convection operator defined by

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

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*.

2.2 Linear stability

Direct eigenvalue problem We consider the stability using the classical ansatz

$$\mathbf{u} = \mathbf{u}_b + \varepsilon\hat{\mathbf{u}}e^{\lambda t} \tag{7}$$

Where $\lambda = \sigma + i\omega$ is the eigenvalue, σ the amplification rate, ω the oscillation rate, $\hat{\mathbf{u}}$ the eigenmode, and ε a small parameter. The eigenmodes is governed by the linear problem $\lambda\hat{\mathbf{u}} = NSL_{\mathbf{u}_b}(\hat{\mathbf{u}}, \hat{p})$, or, in weak form :

$$\lambda\langle \mathbf{v}, \hat{\mathbf{u}} \rangle = \left\langle \mathbf{v}, NSL_{\mathbf{u}_b}(\hat{\mathbf{u}}, \hat{p}) \right\rangle + \langle q, \nabla\cdot\hat{\mathbf{u}} \rangle . \tag{8}$$

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

$$\lambda B\hat{X} = A\hat{X} \tag{9}$$

Where A is the matrix resulting from the discretization of $NSL_{\mathbf{u}_b}$, i.e. the very same matrix as appearing in the Newton computation of the base flow, and B is a 'weight' matrix associated to the scalar product $\langle v, u \rangle = \int \overline{\mathbf{v}}\cdot\mathbf{u} \, \mathrm{d}\Omega$.

Adjoint eigenvalue problem and structural sensitivity Developed in the two past decades, the concept adjoint modes has now become an unavoidable complement to the linear global stability approach. We here give a short summary of the definition and usefulness of this concept. We can first define the *adjoint linearised Navier-Stokes operator* $NSL^\dagger$ defined by the property:

$$\begin{aligned} \forall(\mathbf{u}, p; \mathbf{v}, q), \left\langle NSL_{\mathbf{U}}^\dagger(\mathbf{v}, q), \mathbf{u} \right\rangle + \langle \nabla\cdot\mathbf{v}, p \rangle \\ = \langle \mathbf{v}, NSL_{\mathbf{U}}(\mathbf{u}, p) \rangle + \langle q, \nabla\cdot\mathbf{u} \rangle . \end{aligned} \tag{10}$$

We can then define the adjoint eigenmodes as the solutions to the eigenvalue problem

$$\forall(\mathbf{u}, p), \quad \lambda^\dagger\langle \hat{\mathbf{v}}, \mathbf{u} \rangle = \left\langle NSL_{\mathbf{U}}^\dagger(\hat{\mathbf{v}}, \hat{q}), \mathbf{u} \right\rangle + \langle \nabla\cdot\hat{\mathbf{v}}, p \rangle \tag{11}$$

[2] In the simple presentation given here we have omitted the issue of boundary conditions. Details on way boundary conditions can be incorporated in the weak formulation through integration by parts can be found in the appendix.

It can be shown (Schmid & Henningson, 2001) that the adjoint eigenvalues $\lambda^\dagger$ are the complex conjugates of the direct eigenvalues (λ).

Although the concept of adjoint operator may sound complicate, the resolution of the adjoint problem using finite elements methods is actually extremely easy. In effect, the scalar product used in the definition of the weak formulation and that appearing in the definition of adjoint being the same, the weak formulations of both problems are thus identical when exchanging the test functions and the unknown functions. Thus the matricial form of the discretized version of (11 is deduced from the one of the direct problem by a simple transpose of the matrix :

$$\overline{\lambda}^\dagger B\hat{X}\dagger = A^T\hat{X}\dagger. \tag{12}$$

Adjoint eigenmodes are a powerful tool for investigating problems such as transient growth, control, sensitivity (see the reviews of Chomaz, Schmid, Luchini). The simplest physical interpretation of an adjoint eigenmode is as follows : it corresponds to the initial condition which has maximum projection along the direction of the corresponding eigenmode. Thus, the adjoint of the most amplified mode corresponds to the optimal perturbation which will maximize the growth of energy in the limit of large time. In effect, one can prove that for $t \to \infty$ the asymptotic behaviour of a solution with initial condition $\mathbf{u}_i$ is given as :

$$\mathbf{u}(t) \approx \frac{\langle \hat{\mathbf{u}}^\dagger, \mathbf{u}_i \rangle}{\langle \hat{\mathbf{u}}^\dagger, \hat{\mathbf{u}} \rangle} e^{\lambda t}\hat{\mathbf{u}}$$

. The choice $\mathbf{u}_i = \hat{\mathbf{u}}^\dagger$ is the initial condition of norm unity which maximizes the first factor in this expression.

The adjoint eigenmode also allows to deduce a quantity called the *structural sensitivity* and defined as $S(x,y) = ||\hat{\mathbf{u}}^\dagger||||\hat{\mathbf{u}}||$ which has became popular in the recent years. This quantity is a direct and practical measure of the effect of perturbations of the linear operator on the eigenvalue. The region of the flow where $S(x,y)$ is thus the region where the instability mechanism originates, and is often referred to as the *wavemaker region*

Iterative methods for eigenvalue computations When it comes to the numerical resolution of generalized eigenvalue problems such as $AX = \lambda BX$ (or its adjoint version), several methods are possible. Direct methods to compute the whole spectrum are both costly prohibitive and useless. A popular alternative is iterative methods which allow to compute a limited set of eigenvalues located in the vicinity of a "shift" value λ_{shift}. The simplest version of this method is the simple shift-invert iteration, which consists of solving iteratively the system

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

It is easy to show that this quickly asymptotes to $X^{n+1} \approx (\lambda^{*-1})^n\hat{X}$ where $\hat{X}$ is the eigenmode with largest λ^{*-1} (i.e. the one with eigenvalue λ closest to the shift).

When one has a good estimation of the eigenvalue, this method converges very rapidly and is very efficient, but it can only provide a single eigenvalue. If one wants to compute a larger number of eigenvalues, one can revert to a generalized version of iterative methods, called Arnoldi methods. The shift-invert version of the Arnoldi method is in fact the most commonly used method of the current time and is at the basis of both the popular matlab solver eigs and the build-in eigenvalue solver of FreeFem. Our implementation in StabFem allows to chose between single eigenvalue computation (power method) and multiple eigenvalue computation (Arnoldi). The selection is made according to the parameter "nev" transmitted to the driver.

2.3 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. The most common 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 method. 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 grid-point 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*.

3 Illustration for the wake of a cylinder

Problem description We will now illustrate the applicati

Mesh adaptation procedure

Base flow Having thus produced an convenient mesh, we can now illustrate the properties of the base flow as function

```
baseflow = SF_Init('Mesh_Cylinder_Large.edp');
baseflow = SF_BaseFlow(baseflow,'Re',1);
baseflow = SF_BaseFlow(baseflow,'Re',10);
baseflow = SF_BaseFlow(baseflow,'Re',60);
baseflow = SF_Adapt(baseflow,'Hmax',10,'InterpError',0.01);
[ev,em] = SF_Stability(baseflow,'shift',0.04+0.74i,'type','S')
baseflow = SF_Adapt(baseflow,em,'Hmax',10,'InterpError',0.01);
plotFF(baseflow,'mesh'); plotFF(baseflow,'ux');

Re_Range = [2 : 2: 50]; Drag_tab = []; Lx_tab = [];
    for Re = Re_Range
        baseflow = SF_BaseFlow(baseflow,'Re',Re);
        Drag_tab = [Drag_tab,baseflow.Drag];
        Lx_tab = [Lx_tab,baseflow.Lx];
    end
plot(Re_Range,2*Drag_tab,'b+-');
plot(Re_Range,Lx_tab,'b+-');

Re_Range = [40 : 2: 100];lambda_branch=[];
baseflow=SF_BaseFlow(baseflow,'Re',40);
[ev,em] = SF_Stability(baseflow,'shift',-.03+.72i,'nev',1,'type','D');
    for Re = Re_Range
        baseflow = SF_BaseFlow(baseflow,'Re',Re);
        [ev,em] = SF_Stability(baseflow,'nev',1,'shift','cont');
        lambda_branch = [lambda_branch ev];
    end
plot(Re_Range,real(lambda_branch),'b+-');
plot(Re_Range,imag(lambda_branch)/(2*pi),'b+-');
```

Fig. 1. Illustration of the computation of base-flow properties with StabFem (from script)

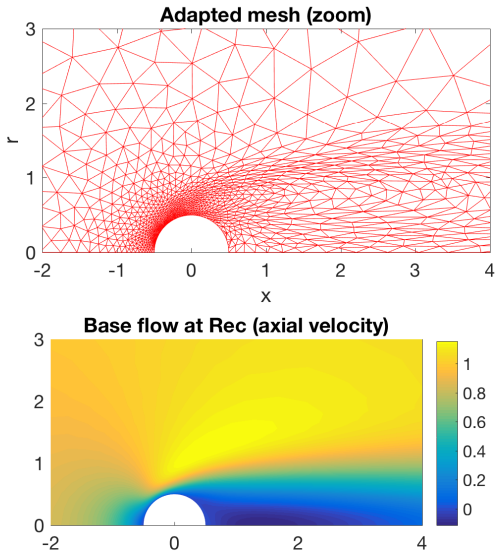

Fig. 2. Adapted mesh (a) and base flow (axial velocity component) (b) for the flow over a cylinder at $Re = Re_c = 46.7$.

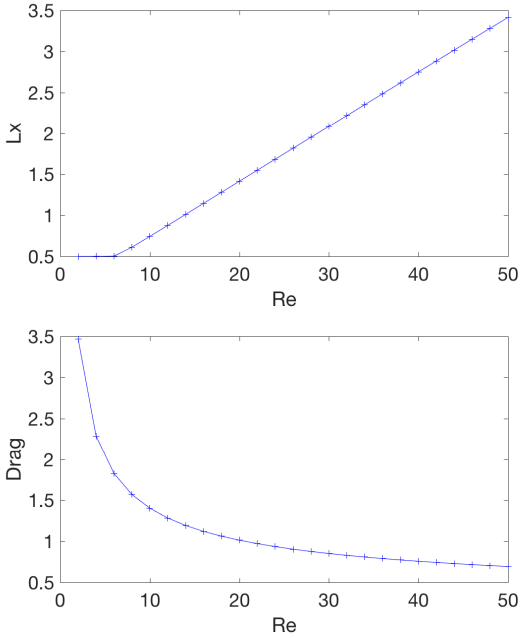

Fig. 3. Recirculation length L_x (a) and drag C_x (b) of the base flow over a cylinder as function of Re.

of Reynolds number. Figure (1) shows how to compute and plot with StabFem the two most commonly studied quantities, namely the drag coefficient $C_x(Re)$ and the recirculation length $Lx(Re)$, namely the location of the stagnation point at the rear of the recirculation region. Note that the object `baseflow` is defined as a structure with fields `Drag` and `Lx`. The resulting plots are given in figure (3), and are in good agreement with known results for this classical problem. In particular, for low Reynolds, the recirculation $L_x(Re)$

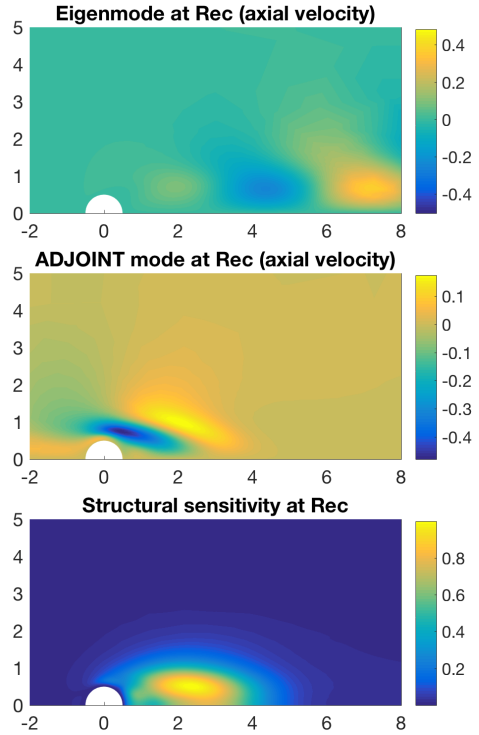

Fig. 4. Eigenmode (a) (axial velocity component), Adjoint mode (b) and structural sensitivity (c) for the cylinder's wake at $Re = Re_c = 46.7$.

is equal to 0.5 (which is the radius of the cylinder) indicating the absence of a recirculation region. The latter appears for $Re > 4.8$, in accordance with known results.

Linear stability results Matlab driver...

4 Nonlinear global stability approaches

Review on nonlinear global stability approaches In the past decade, efforts have been devoted to extend the range of validity of global approaches into the nonlinear rgime for $Re > Re_c$, with the double objective to describe the properties of the limit cycle reached after saturation and to derive amplitude equations describing the transitent dynamics towards this cycle. The first milestone in that direction is the work of Sipp & Lebedev who used a weakly nonlinear development in terms of the distance to the threshold $Re - Re_c$. However, although derived on rigorous grounds, the weakly nonlinear development has a very limited range of validity and already fails for $Re \geq 47$. More recently, Mantic-lugo et al. proposed an alternative approach termed *Self-consistent model* which succeeds in predicting the dynamics up to $Re = 120$ and more. This model was build as a linear stability approach build on a *mean flow* which includes a retroaction of the eigenmode. However, if one is interested in the limit cycle and not the transient, the approach is actually equivalent to a Fourier series decomposition of the flow with respect to time (an approach also known as *harmonic balance*) retaining only the two first terms.

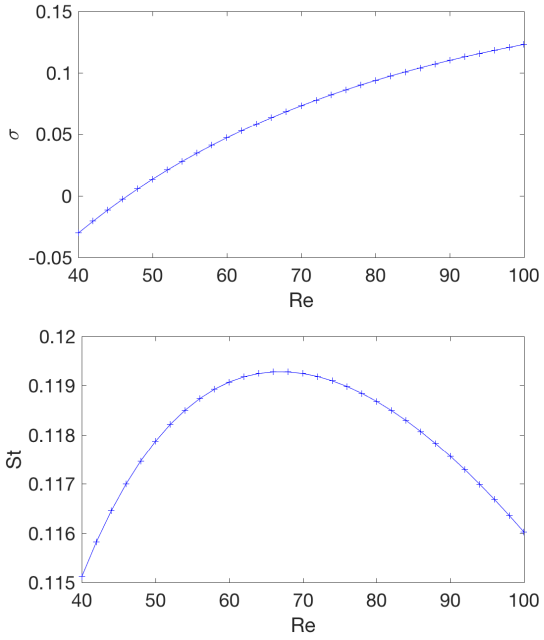

Fig. 5. Growth rate σ (a) and Strouhal number $St = (a\omega)/(2\pi U_0)$ (b) as function of Reynolds number

In the next section we thus present this model and propose a direct resolution method. The relation with the initial method of Mantic-Lugo et al. will be discussed afterwards.

Harmonic balance : equations and procedure In this section, to simplify the notations, The Navier-Stokes equations will be written symbolically as $\partial_t \mathbf{u} = NS(\mathbf{u})$, therefore droping the systematic reference of the incompressibility constraint and associated pressure field.

We thus start with a Fourier decomposition of the limit cycle under the form :

$$\mathbf{u} = \mathbf{u}_0 + \mathbf{u}_1 e^{i\omega t} + \mathbf{u}_{-1} e^{-i\omega t} \tag{13}$$

Here $\mathbf{u}_0$ is the mean flow (averaged over time), $\mathbf{u}_1$ is the mode (fourrier component at the fundamental frequency), $\mathbf{u}_{-1} = \overline{\mathbf{u}_1}$ is its complex conjugate, and ω is the (real) oscillation frequency of the limit cycle (which is not known a priori). Injecting this ansartz into the Navier-Stokes equations and taking the mean value and first Fourier component leads to the following coupled equations :

$$NS(\mathbf{u}_0) - \mathcal{C}(\mathbf{u}_1, \overline{\mathbf{u}}_1) = 0, \tag{14}$$

$$i\omega \mathbf{u}_1 = NSL_{\mathbf{u}_0}(\mathbf{u}_1). \tag{15}$$

Note that the Fourier description of the limit cycle is invariant with respect to the phase ϕ (i.e. the equations are unchanged when replacing $[\mathbf{u}_1, \mathbf{u}_{-1}]$ by $[\mathbf{u}_1 e^{i\varphi}, \mathbf{u}_{-1} e^{-i\varphi}]$. To

allow resolution we thus need an extra equation to fix this phase. Several choices are possible, but a convenient one is to set the lift associated with the imaginary part of $\mathbf{u}_1$ to lift. Physically, this means that the instant $t = 0$ will correspond to a maximum of lift. This condition reads :

$$\Im F_y(\mathbf{u}_1) = 0 \tag{16}$$

With $F_y(\mathbf{u_1}) = \int_{\Gamma_w} \left[-p_1\mathbf{n} + \frac{2}{Re}\mathbf{D}(\mathbf{u_1})\cdot\mathbf{n}\right]\cdot\mathbf{e_y} = 0$.

Mathematically, the problem **??** for the unknown $[\mathbf{u}_0, \mathbf{u}_1, \omega]$ is well posed, and can be solved using a Newton iteration just as explained for the base flow in section 2. To allow resolution, we separate the mode into real and imaginary part, by writing $\mathbf{u}_1 = \mathbf{u}_{1,r} + i\mathbf{u}_{1,i}$. We then assume that we know a 'guess'

$$[\mathbf{u}_0, \mathbf{u}_{1,r}, \mathbf{u}_{1,i}, \omega] = [\mathbf{u}_0^g, \mathbf{u}_{1,r}^g, \mathbf{u}_{1,i}^g, \omega^g] + [\delta\mathbf{u}_0, \delta\mathbf{u}_{1,r}, \delta\mathbf{u}_{1,i}, \delta\omega]$$

.

Injecting and developing up to linear order leads to the following

$$\begin{aligned}
&NS(\mathbf{u}_0^g) - [\mathcal{C}(\mathbf{u}_{1,r}^g, \mathbf{u}_{1,r}^g) + \mathcal{C}(\mathbf{u}_{1,i}^g, \mathbf{u}_{1,i}^g)] \\
&+ NSL_{\mathbf{u}_0^g}(\delta\mathbf{u}_0) - 2[\mathcal{C}(\mathbf{u}_{1,r}^g, \delta\mathbf{u}_{1,r}) + \mathcal{C}(\mathbf{u}_{1,i}^g, \delta\mathbf{u}_{1,i})] = 0; \\
&NSL_{\mathbf{u}_0^g}(\mathbf{u}_{1,r}^g) - \omega^g\mathbf{u}_{1,i}^g \\
&+ NSL_{\mathbf{u}_0^g}(\delta\mathbf{u}_{1,r}^g) - \mathcal{C}(\delta\mathbf{u}_0^g, \mathbf{u}_{1,r}) - \delta\omega^g\delta\mathbf{u}_{1,i}^g - \omega^g\delta\mathbf{u}_{1,r} = 0; \\
&NSL_{\mathbf{u}_0^g}(\mathbf{u}_{1,i}^g) + \omega^g\mathbf{u}_{1,i}^g \\
&+ NSL_{\mathbf{u}_0^g}(\delta\mathbf{u}_{1,i}^g) - \mathcal{C}(\delta\mathbf{u}_0^g, \mathbf{u}_{1,i}) + \delta\omega^g\delta\mathbf{u}_{1,i}^g + \omega^g\delta\mathbf{u}_{1,r} = 0; \\
&F_y(\mathbf{u}_{1,i}^g) + F_y(\delta\mathbf{u}_{1,i}) = 0.
\end{aligned} \tag{17}$$

After discretization, this leads to a linear problem with the form $AX = Y$ where X is the discretized version of the unknowns $[\delta\mathbf{u}_0, \delta\mathbf{u}_{1,r}, \delta\mathbf{u}_{1,i}, \delta\omega]$, and A is a matrix of dimension $3N_{dof} + 1$.

Results for the wake of a cylinder

Link with the Self-Consistent approach We now discuss the link between the harmonic balance approach presented above and the self-consistent approach as introduced by Mantic-Lugo et al. The starting point of this model is a pseudo-eigenmode expansion of the flow as follows :

$$\mathbf{u} = \mathbf{u}_m + A\left[\tilde{\mathbf{u}}_1 e^{\sigma_{SC}t + i\omega_{SC}t} + \tilde{\mathbf{u}}_{-1} e^{\sigma_{SC}t - i\omega_{SC}t}\right] \tag{18}$$

Where $\mathbf{u}_m$ is a *mean flow* defined by phase-averaging over the cycle, $\tilde{\mathbf{u}}_1$ is a pseudo-eigenvector which is normalized by the condition $||\tilde{\mathbf{u}}_1|| = 1/\sqrt{2}$, A is an amplitude parameter directly related to the energy of the oscillating flow, and $\lambda = \sigma_{SC} + i\omega_{SC}$ is a pseudo-eigenvalue which depends upon the parameter A. The mean flow is then governed by the same equation (14) written above, while $\tilde{\mathbf{u}}_1$ is the solution of an eigenvalue problem with the same form as (15) except that $i\omega$ is replaced by $\sigma_{SC} + i\omega_{SC}$. The self-consistent model has the following properties :

- For $A \ll 1$ it is equivalent to the linear eigenvalue problem (**??**), and the generalized eigenvalue coincides with the one predicted by linear stability : $\sigma_{SC} + i\omega_{SC} = \sigma_{lin} + i\omega_{lin}$.
- For $\sigma_{SC} = 0$ (corresponding to a specific choice of the amplitude $A = A_c$), the expansion (18) is equivalent to the harmonic-balance decomposition (13)
- For $0 < A < A_c$, the resolution leads to a relation $\sigma_{SC}(A); \omega_{SC}(A)$ such that $0 < \sigma_{SC}(A) < \sigma_{lin}$. Although in this case the expansion 18 cannot represent the flow for all t, Mantic-Lugo et al. argued that the relation between σ_{SC} and A can be used to build an amplitude equation which captures the transient approach to the limit cycle.

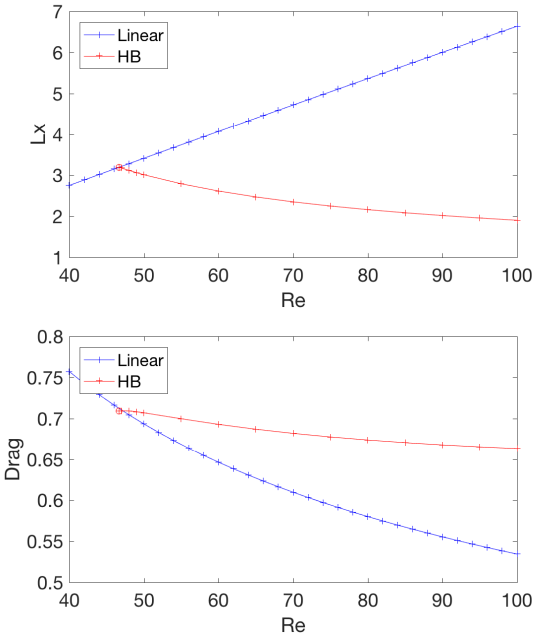

Fig. 6. Harmonic-balance results : mean-flow characteristics

A Appendix : details on mesh convergence

Illustration of mesh convergence procedure for a cylinder We illustrate the procedure in the case of a cylinder. Figure 1 gives a retranscription of the sequence of commands (from *SCRIPT_CYLINDER_ADAPTMESH_BASEFLOW*) and the output produced. First we build an initial mesh (line 1), and compute base flow solutions for increasing values of the Reynolds number up to $Re = 60$ (lines 3-9).

Then we perform the mesh adaptation (line 13). The produced outputs (lines 14-17) gives information about the resulting mesh. Note the values h_{min} and h_{max} of the smaller and larger edges, as well as the local grid size at four points

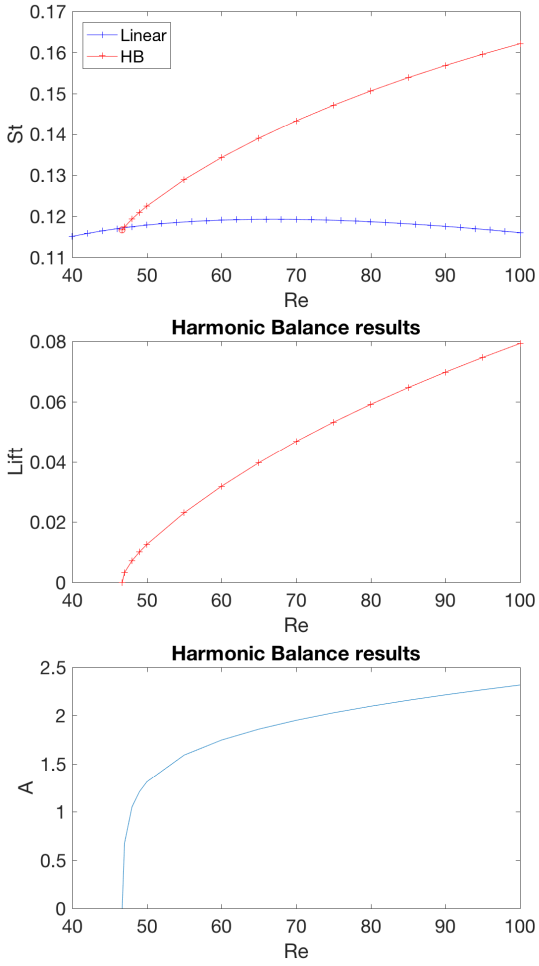

Fig. 7. Harmonic-balance results : mode characteristics

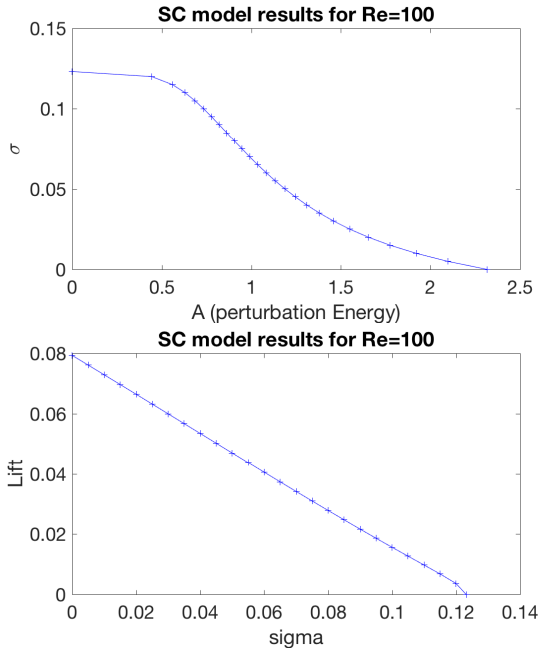

Fig. 8. SELF CONSISTENT APPROACH FOR THE WAKE OF A CYLINDER, $Re = 60$

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). Finally, the base flow is automatically recomputed on the resulting mesh (line 18-19). Finally, lines 21-22 plot the mesh and base flow structure, producing the result displayed in figure (2).

B Detail of the weak formulation

$$\forall(\mathbf{v}; q), \tag{19}$$
$$\int \left[\mathbf{v}\cdot\mathcal{C}(\mathbf{u}_b^g, \delta\mathbf{u}_b) + \nabla\cdot\delta\mathbf{u}_b q - \nabla\cdot\mathbf{v}\delta p_b + \tfrac{2}{Re}\mathbf{D}(\delta\mathbf{u}_b) : \mathbf{D}(\mathbf{v})\right]$$
$$+ \int \left[\mathbf{v}\cdot(\mathbf{u}_b^g\cdot\nabla\mathbf{u}_b^g) + \nabla\cdot\mathbf{u}_b^g q - \nabla\cdot\mathbf{v}p_b^g + \tfrac{2}{Re}\mathbf{D}(\mathbf{u}_b^g) : \mathbf{D}(\mathbf{v})\right] = 0$$

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

> baseflow=SF_Adapt(baseflow,'Hmax',10,'InterpError',0.01);
     ### 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 SF_BaseFlow : computing base flow for Re = 60
      # Base flow converged in 5 iterations ; Drag = 0.65805; Lx = 4.0737

> plotFF(baseflow,'mesh');
```

Fig. 9. Illustration of the procedure to compute a base flow and an adapted mesh using StabFem (from script *SCRIPT_CYLINDER.m*.

```
> [ev,em] = SF_Stability(baseflow,'shift',0.04+0.76i,'nev',1,'type','D');
> [baseflow,em]=SF_Adapt(baseflow,em,'Hmax',10,'InterpError',0.005);
      ### ADAPT mesh to base flow AND MODE  ; InterpError = 0.005  ; Hmax = 10
      #   Number of points np = 13458 ; Ndof = 120030
      #  deltamin, deltapax : 0.0089891 , 13.367
      #  delta_(A,B,C,D) : 0.020536 , 0.082852 , 0.07316 , 0.11914
      ### FUNCTION SF_BaseFlow : computing base flow for Re = 60
      # Base flow converged in 4 iterations ; Drag = 0.64695; Lx = 4.0715
      ### FUNCTION SF_Stability : computation of 1 eigenvalues/modes (DIRECT) with FF solver
      # Stability calculation converged in 4 iterations, lambda = 0.047152+0.74808i.

> [ev,em] = SF_Stability(baseflow,'shift',0.04+0.76i,'nev',1,'type','S');
> [baseflow,em]=SF_Adapt(baseflow,em,'Hmax',10,'InterpError',0.005);
      ### ADAPT mesh to base flow AND SENSITIVITY ; InterpError = 0.005  ; Hmax = 10
      #   Number of points np = 4355 ; Ndof = 38559
      #  deltamin, deltapax : 0.0058536 , 13.1164
      #  delta_(A,B,C,D) : 0.01049 , 0.091861 , 0.13319 , 0.89372
      ### FUNCTION SF_BaseFlow : computing base flow for Re = 60
      # Base flow converged in 3 iterations ; Drag = 0.64758; Lx = 4.0725
      ### FUNCTION SF_Stability : computation of 1 eigenvalues/modes (DIRECT) with FF solver
      # Stability calculation converged in 3 iterations (D+A+S), lambda = 0.047165+0.74823i.
```

Fig. 10. Illustration of the procedure to adapt the mesh for stability calculations with StabFem (from script *SCRIPT_CYLINDER.m*.

Fig. 11. Illustration of the procedure to adapt the mesh for stability calculations with StabFem (from script *SCRIPT_CYLINDER.m*.