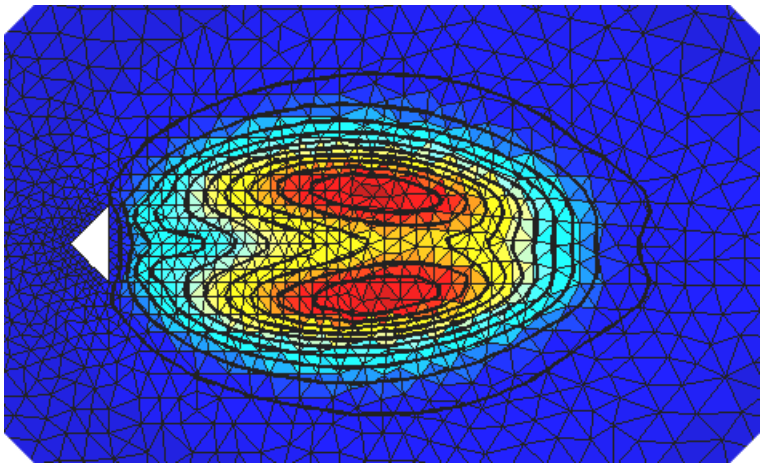
Incompressible Navier-Stokes Stability and Structural Sensitivity Analysis

- using FreeFem++ and Matlab -

Stefano Boccelli

- Feb. 2015 -



1 Quick Introduction

This is simply a collection of scripts I've made while preparing the exam of *Fluid Flows Instability and Turbulence*. My goal was implementing something that I was studying and seeing if I could produce those cool structural sensitivity plots!

The results are far from accurate, but I think I've caught the fluid dynamic behavior I was hunting!! I've uploaded my scripts only for your own joy and in the behalf that someone in the universe could maybe benefit from them. Do not trust on my results too much.. *And do not even trust on my comments that much, these are merely notes and observations of a student and might be totally misleading!*

My scripts compute the stability and structural sensitivity of a flow whose geometry is created via a **FreeFem++** input file. FreeFem++ also computes the solution to Navier-Stokes equation, providing a flow around which the problem can be linearized to compute eigenpairs. Stiffness and Mass matrices for linearized Navier-Stokes variational formulations are exported with FreeFem++ and then **Matlab** does the rest:

- importing mesh, matrices, fixed point solution etc
- assembling and solving the generalized eigenvalue problem
- computing structural sensitivity
- plotting colored streamlines over everything

Why didn't you use GNU/Octave, you proprietary-software slave?! Almost all the scripts work under Octave, except a couple of things about plots (i.e. the `TriScatteredInterp` function), but the point is that Octave is unfortunately way too slow -compared to Matlab- in dealing with big linear algebra problems.. Sorry GPL friends..

PS: there's interesting document on the net, called *A Comparative Evaluation of Matlab, Octave, FreeMat, and Scilab on Tara*, comparing various general purpose scientific computation software, see [1].

In this file:

- **Linear Stability Analysis** of a base flow
- **Structural Sensitivity**: a quick overview
- Computing the base flow with Finite Elements (**FreeFem++**)

- **Matlab** and the computation of eigenpairs
- **Results**, 2D flow around slender bodies: cylinder, triangle and two cylinders side by side
- **Final Remarks**

Contents

2 Linear Stability of a base flow

The linear stability of a base flow $\vec{U}$ can be studied by adding to the flow a small perturbation $\vec{u}$. This leads to a linearized Navier-Stokes system, whose eigenvalues determine the stability of the small perturbation.

First of all, write the velocity as $\vec{U} + \vec{u}$, where $\vec{u}$ is supposed to be small and hence the second order terms can be neglected, thus obtaining a linear system where the unknown is the perturbation $\vec{u}$:

$$\begin{cases} \frac{\partial \vec{u}}{\partial t} + (\vec{U} \cdot \vec{\nabla})\vec{u} + (\vec{u} \cdot \vec{\nabla})\vec{U} - \frac{1}{Re}\nabla^2 \vec{u} + \vec{\nabla} p = 0 \\ \vec{\nabla} \cdot \vec{u} = 0 \end{cases} \tag{1}$$

then one can express the solution of that *linear* system as $\vec{u} = \vec{u}_i e^{\lambda_i t}$, where λ_i is the i^{th} eigenvalue and $\vec{u}_i$ the corresponding eigenfunction. Substitute it, simplify the omnipresent term $e^{\lambda_i t}$ and you'll obtain the direct eigenvalue problem:

$$\begin{cases} \lambda_i \vec{u}_i + (\vec{U} \cdot \vec{\nabla})\vec{u}_i + (\vec{u}_i \cdot \vec{\nabla})\vec{U} - \frac{1}{Re}\nabla^2 \vec{u}_i + \vec{\nabla} p_i = 0 \\ \vec{\nabla} \cdot \vec{u}_i = 0 \end{cases} \tag{2}$$

The boundary conditions to be imposed on such a problem are homogeneous Dirichlet everywhere: both on the external boundaries and the walls. *Note that imposing zero perturbation on external boundaries means that we are not introducing energy in the system from the boundaries.*

Those BCs would require a pretty big domain, which implies quite long calculation times on average personal computers. In this document I've used quite humble domains.

The discretized problem (by FEM for example, see section 4) has a huge number of eigenvalues. In this work, I've calculated around 100 eigenvalues for each flow (pay attention, be sure the most unstable ones have been computed!)

2.1 Meaning of the eigenvalues

While talking to my friend, who studies structural engineering and is quite comfortable with eigenvectors related to buckling and structural instability problems, he asked me something like "what on Earth do Navier Stokes eigenvalues mean??" I think the answer is quite useful to be remarked here.

Let's say you have a nonlinear dynamical system, or a nonlinear governing equation.. and let's say that you've found a solution. You might need to know whether

your solution is stable or not. Think of a compressed rod: the bifurcation diagram shows a Pitchfork bifurcation and after a certain critical load the "straight" solution is no longer stable, the rod buckles etc! Or.. think at the 2D fluid flow around a cylinder: one can compute a stationary solution at $Re = 50$, obtaining something colored and cool that is apparently OK, but we know that at this regime the system have passed a Hopf bifurcation and the computed solution is unstable (a Von Karman wake generates etc).

One way to test the stability of a solution is doing the same thing we do to old rusty watches that point a wrong hour: we gently knock them and see if a different solution appears. By writing the unknown of our Navier Stokes system as $\vec{U} + \vec{u}$ we are adding such a "gentle perturbation" to the computed solution $\vec{U}$. We now have a new equation governing the evolution of $\vec{u}$.

Since the system is *linear*, the solution can be written as a combination of eigenvectors: $\vec{u} = \sum_i A_i e^{\lambda_i t} \vec{u}_i$. The meaning of the eigenvalues for the linearized problem should now be clear: **they represent the exponential growth rate of a perturbation**!

Example: compressed rod

We can study the linear stability of a compressed rod in the very same way. The governing equation is something like:

$$EJw_{/xx} - Pw = 0 \quad \rightarrow w_{/xx} - \alpha^2 w = 0 \tag{3}$$

where EJ is the rod stiffness, P is the applied load and w the deviation out of the symmetry plane. Now, if you have your solution you can expand it as $w = w + \hat{w}$ where $\hat{w}$ is supposed to be a small perturbation. You end up with a differential equation in the unknown $\hat{w}$ (cause w is known, remember?). This equation is linear (because if it wasn't yet, the hypothesis that $\hat{w}$ is small will kill higher order therms) so you can write $\hat{w} = e^{\lambda_i x} \hat{w}_i$ and solve the eigenvalue problem. Those eigenvalues represent the growth rate of the perturbation $\hat{w}$ with respect to the independent variable of the problem, that is x in this case! An unstable solution will cause some small perturbations to grow exponentially wrt x, thus deviating a lot from the original result!

3 Structural Sensitivity

The Structural Sensitivity analysis aims to answer the question "*which perturbation produces the biggest stability variation in a flow?*" To answer this question, we introduce a perturbation in the operator of the Navier Stokes equations linearized around a computed base flow. Perturbing the operator implies obtaining perturbed eigenvalues and eigenvectors: **the idea is searching for the perturbation that generates the maximum variation to the most unstable eigenvalue**.

So.. referring to the section 2, first of all one linearizes the Navier Stokes system around a base flow $\vec{U}$ and computes eigenpairs, then introducing a localized perturbation on the Oseen operator (linearized Navier Stokes operator) in the form of a Dirac delta function, one can express:

$$s(x,y) = \frac{||\vec{v}_i(x,y)||\ ||\vec{u}_i(x,y)||}{|\int_\Omega \vec{v}_i^* \cdot \vec{u}_i|}; \tag{4}$$

The vectors $\vec{v}_i$ and $\vec{u}_i$ represent the adjoint and direct eigenfunction, corresponding to the i^{th} eigenvalue. If you choose the most unstable eigenvalue, this function is called **structural sensitivity parameter**, see [3].

The direct eigenfunctions can be computed as previously showed, while for the adjoint ones there are two ways:

1. formulating and solving the adjoint eigenvalue problem
2. computing the hermitian matrices of the direct discretized eigenvalue problem and then solving the resulting problem.

See section 4 for more about implementation.

4 Solution via Finite Element Method

4.1 Domain discretization

The domain discretization have been performed with the FreeFem++ built-in grid generator, specifying domain borders and number of elements for every side. FreeFem++ is able to compute eigenpairs (as explained in FreeFem++ manual), but I decided to export the results for further operations with Matlab. FreeFem++ let the user choose between various mesh formats to be exported.

4.2 Base Flow computation

The first step in our calculation (right after the domain distcretization, which is the *very* first step) is the computation of a base flow, around which we want to linearize the Navier Stokes equations. I've choosed to use P1 (polinomial, 1^{st} order) finite elements for pressure and P1-bubble for velocity field. For sure this choice brings in a certain inaccuracy, but that was almost a forced choice for reasons linked to the *hard and heavy* eigenvalue computation step.

Navier Stokes equations have been linearized with Newton method and a continuation method was implemented to reach convergence with much more ease: a first simulation is performed with a very low Reynolds number, then the computed solution is used as initial guess for the a problem with increased Reynolds and so on until reaching the target Re.

4.3 Generalized Eigenvalue Problem

Finite element discretization of Navier-Stokes equations leads to the generalized eigenvalue problem:

$$(A - \lambda_i B)\vec{u}_i = 0 \tag{5}$$

whose A and B matrices have a typical block shape commonly arising in mixed finite element discretization: $A = \begin{bmatrix} K & C \\ C^T & 0 \end{bmatrix} \quad B = \begin{bmatrix} M & 0 \\ 0 & 0 \end{bmatrix}$

Due to the absence of time derivatives for pressure, the B matrix is singular and this requires ad-hoc methods to be implemented. As shown by *Cliffe, Garratt e Spence* in [2], it's possible to study the reduced problem:

$$Q_2^T(K - \lambda M)Q_2\vec{z} = 0 \tag{6}$$

where the Q_2 matrix is obtained from the *qr decomposition* of C:

$$C = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R \\ 0 \end{bmatrix} \tag{7}$$

After solving the reduced problem, the original problem eigenvectors are readily found:

$$\vec{u} = Q_2 \vec{z} \tag{8}$$

It's easy to show that the adjoint eigenpairs come from the eigenvalue problem:

$$Q_2^T (K^H - \lambda M^H) Q_2 \vec{z}_{adj} = 0 \tag{9}$$

one can obtain it by recalling that the adjoint operator of a matrix product is the matrix hermitian:

$$(\vec{y}, A\vec{x}) = \vec{y} \cdot A\vec{x} = \vec{y}^H A\vec{x} = (\vec{y}^H A\vec{x})^{H^H} = ... = (A^H \vec{y}, \vec{x}) \tag{10}$$

4.4 FreeFem++ operations

FreeFem++ was used first of all to export the mesh and the computed base flow. Then the command `varf` was used to create the matrices K, C and M, that were exported for further operations with Matlab.

4.5 Matlab Scripts

Matlab was used to compute eigenpairs for stability analysis and structural sensitivity. The real bottleneck of the whole calculation process was shown to depend on mesh size.

- on "usual" and quite poor meshes (the one you can see in the results section), the most time consuming calculation was multiplying the Q_2, K and M matrices, created while constructing the reduced eigenvalue problem
- on bigger meshes the most requiring part was the `eigs()` algorithm, a couple of times also crashed the system by completely filling up the RAM memory.

In order to let ordinary laptops perform those calculations, it was necessary to keep matrices as small as possible, thus implementing P1 finite elements for pressure and P1-bubble for the velocity field. A better choice would be P1 // P2 or higher

order polynomials. The P1 // P1-bubble is the absolute minimum requirement to obtain a numerically stable problem. To obtain fair results, also relatively loose meshes require calculation times of about one night.

Note that FreeFem++ imposes boundary conditions via *penalty methods*. FreeFem imposes the so called TGV (Tres Grand Valeur = Very Big Value) to elements to whom a Dirichlet BC is imposed. By imposing a very big element in the stiffness or mass matrix ($TGV = 10^{30}$) in the diagonal position corresponding to a Dirichlet node, the other elements of the same row and column lose their importance. The result is almost the same as eliminating every non-diagonal component for Dirichlet elements, thus imposing a "constraint". Since programming languages memorize sparse matrices in row format or column format, accessing both rows and columns is quite an expensive operation. TGV method is much faster that more clean and polite methods and yet preserves symmetry, but generates very bad-scaled matrices and that turns out to be a problem for eigenpairs-calculating algorithms! To solve this problem, we accept a little waste of time and cycle over Dirichlet nodes, imposing TGV-ed elements to 1 and the rest of the row and column to 0.

To compute eigenpairs of sparse matrices, Octave and Matlab provide the `eigs` function. `eigs` works with both dense and sparse matrices and computes a definite number of eigenvalues and eigenvectors, allowing the user to choose for largest or smallest magnitude or largest/smallest real/imaginary part, or even the closest to a specified point.

For every flow, I've calculated the 100 smallest magnitude eigenpairs. For stability analysis purposes, one should compute the eigenvalues with smallest real part, but the `eigs` algorithm in *smallest real* mode is much much much slower than it's *smallest module* counterpart.

5 How to run the scripts

Of course you must have a version of Matlab and FreeFem++ on your system. Also, if you're not using a UNIX operating system you may need to modify the name of the directories, into the scripts, replacing ./ with a backslash. The rest is pretty simple: open Matlab and run the *STRUCTURAL_SENSITIVITY.m* script! The script will do the following, with very little user interaction:

1. run a FreeFem++ script (`.edp` extension). Modify the proper line in *STRUCTURAL_SENSITIVITY.m* to call the correct .edp file, you can choose between a few geometries I created.

2. the FreeFem++ script exports the mesh, computes the base flow with a continuation method and then exports the Stiffness and Mass matrices. You can enable a flag into the script to perform an evolutive simulation. Just be sure to disable it if you want to perform some structural sensitivity analysis, or your base flow won't be a stationary NS solution, but the last time-step instead!

   When the FreeFem++ computation is over, you'll be asked to *ESC the process.*. just click on the image and press *ESC*.

3. the main script imports mesh, solution and computes the center of mass and area for every triangle.

4. 100 eigenpairs are calculated and plotted

5. structural sensitivity is plotted **for a given eigenvalue**. You must look at the eigenvalues (by typing `diag(Lamb)`), locate the most unstable one and specify to the file called *stru_sens.m* that you want to compute the structural sensitivity over *that* eigenvalue, by setting the variable `eigPosStru` (or something like that) to the proper eigenvalue position in `diag(Lamb)`.

6 Results

Let me now show some numerical results. I've calculated eigenpairs and structural stability for some 2D flows in the vicinity of a a bifurcation point. In this section you can find the 100 smallest module eigenvalues, some direct and adjoint eigenvectors associated to the most unstable eigenvalue and structural sensitivity plots overimposed on current streamlines.

6.1 Flow past a cylinder

Here are the numerical results for the flow around a cylinder, the true king of bluff-bodies. Simulation was performed for $Re = 50$, a little above the Hopf bifurcation point of the Von Karman wake. As one could imagine, the eigenvalues plot shows two complex conjugate eigenvalues right at the right of the imaginary axis.

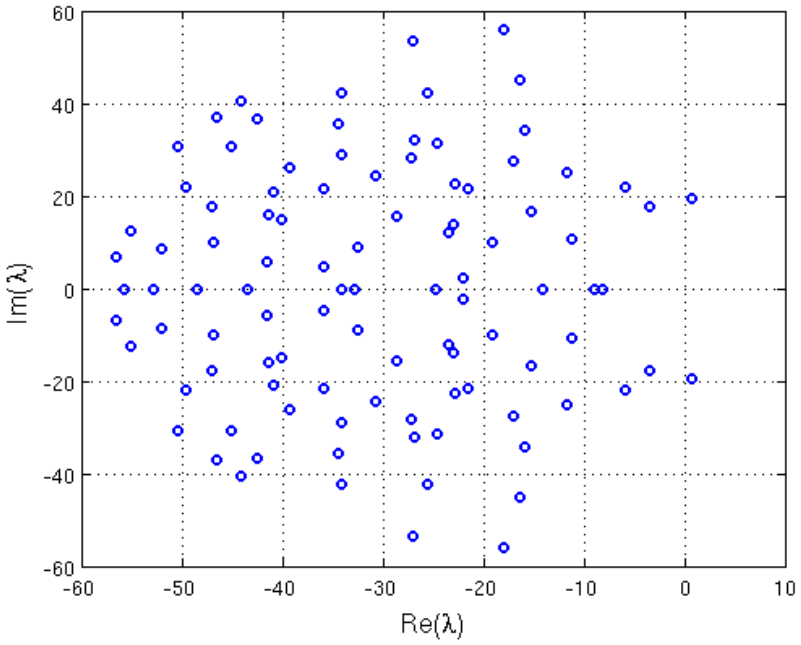


Figure 1: *100 eigenvalues, $Re = 50$*

Here are the eigenvectors of the unstable eigenvalues.

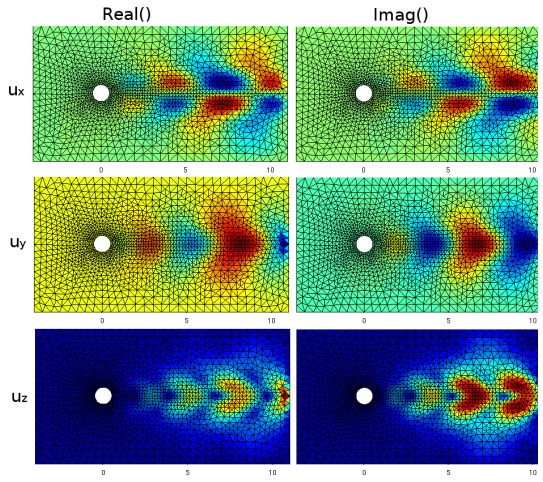


Figure 2: *Direct eigenvector. Left column: real part, right column: imaginary part.*

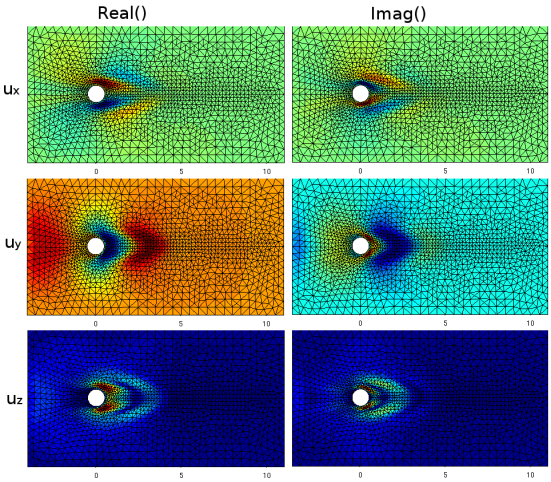


Figure 3: *Adjoint eigenvector. Left column: real part, right column: imaginary part.*

Figures 4 and 5 show the structural sensitivity parameter for the unstable eigenvalue.

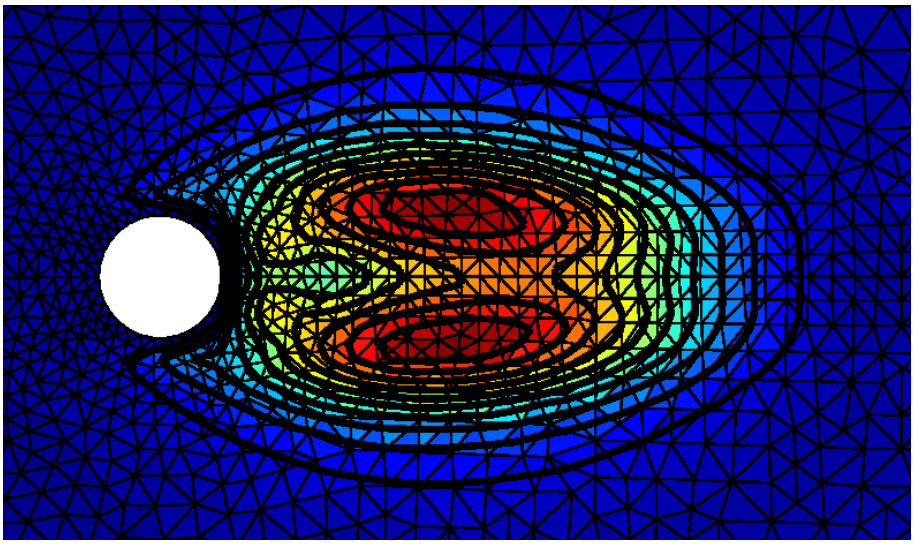


Figure 4: *Structural sensitivity parameter.*

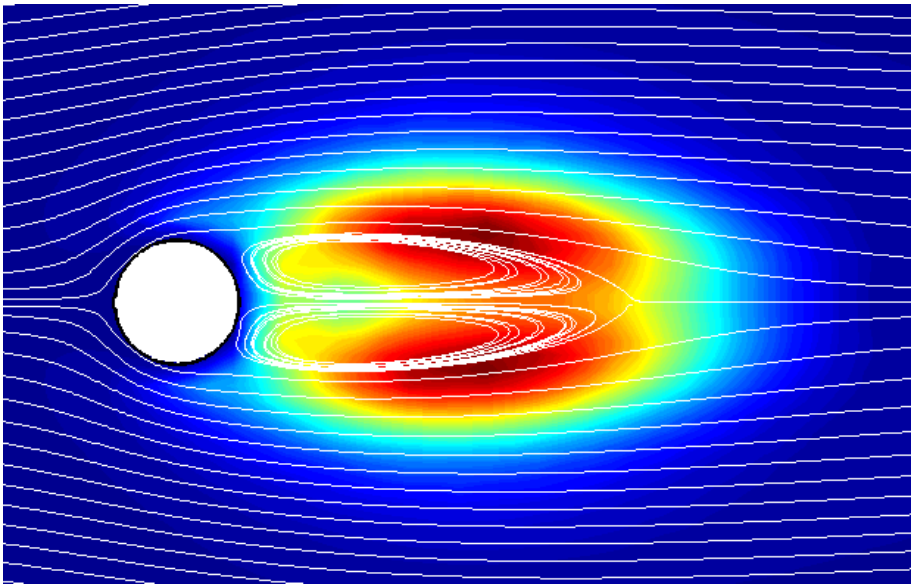


Figure 5: *Structural sensitivity parameter overimposed on base flow streamlines.*

6.2 Flow past a triangular bluff-body

Here are the results for a flow over a right angled triangle with two equal sides and the third of length L. The Reynolds number was defined as: $Re = \frac{UL}{\nu} = 50$. In figure 9 is shown the coolest mode: it's the imaginary part of the vertical velocity component of an adjoint eigenvector.

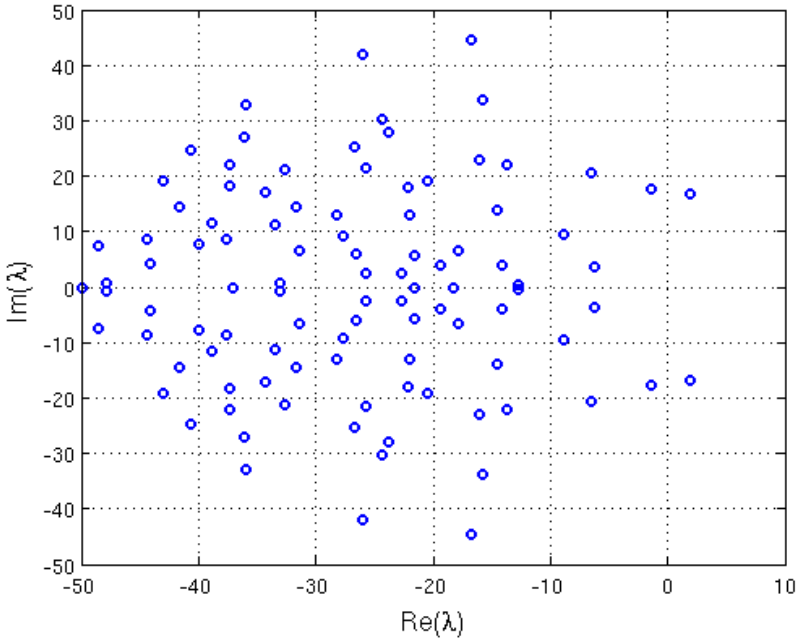


Figure 6: *100 eigenvalues, Re = 50.*

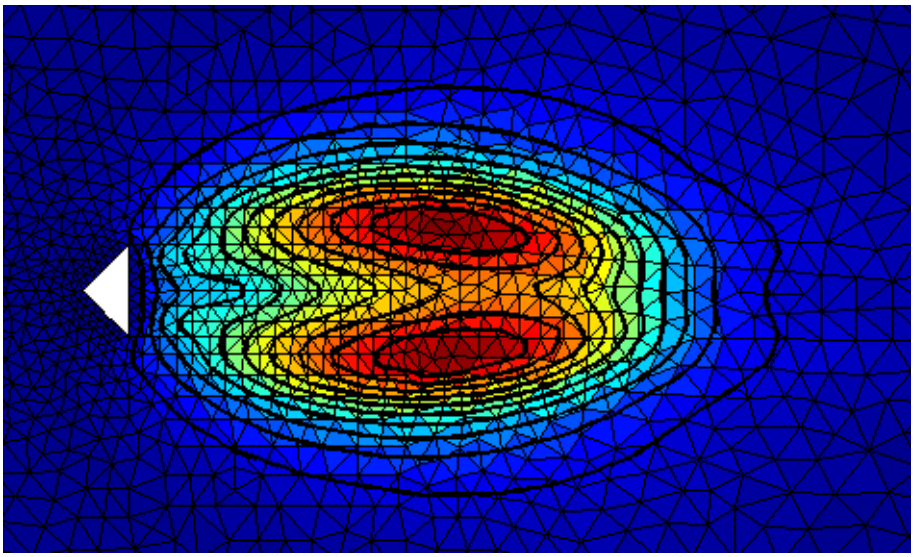


Figure 7: *Structural Sensitivity, contour*

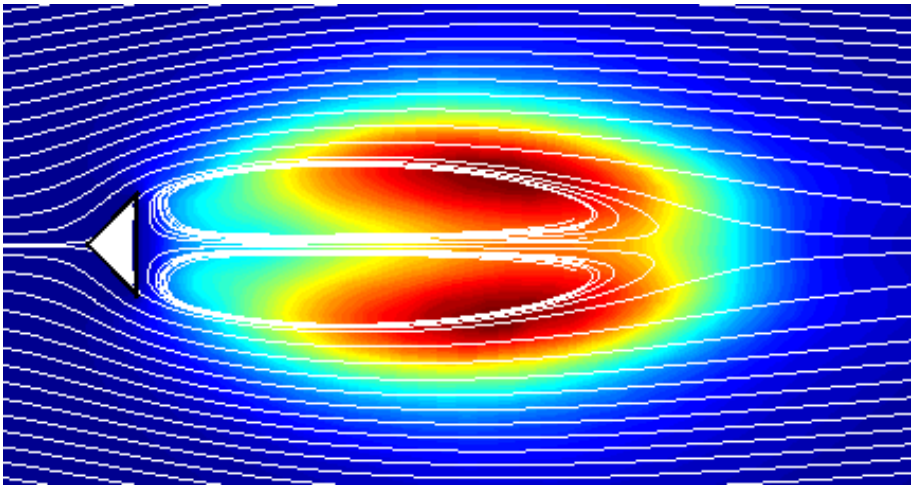


Figure 8: *Structural sensitivity parameter overimposed on base flow streamlines*

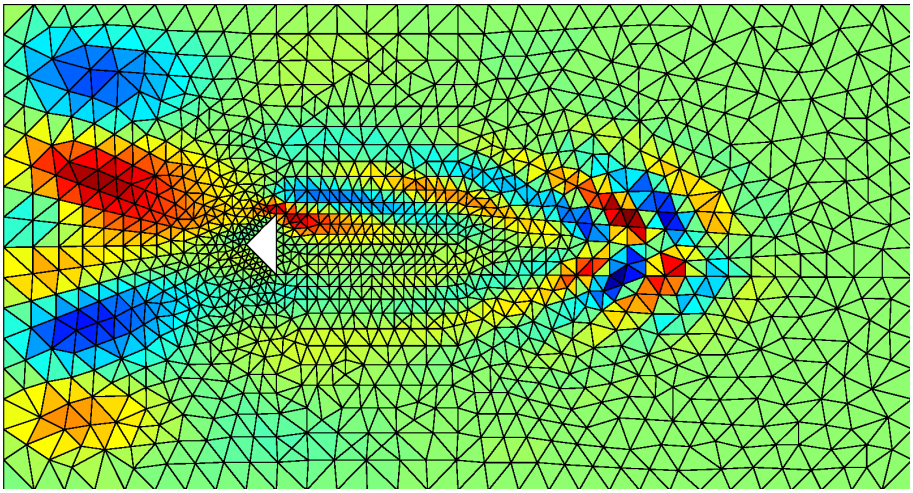


Figure 9: *The coolest eigenvector*

6.3 Flow past two side by side cylinders

I've tried to apply the scripts to the flow past two side by side cylinders. This problem is a little computationally harder than the previous ones, since the presence of two cylinders enlarges minimum acceptable dimensions for the domain and also doubles the fine zone of the mesh! That's why I had to perform the calculations on a quite loose mesh.

The problem is defined by two parameters: the Reynolds number Re and the ratio "gap over cylinder diameter" g/D. Figure 10 shows a bifurcation diagram for this problem, computed by *Carini, Auteri e Giannetti* in [4].

I've choosed to analyze flow between *in-phase shedding* and *asymmetric mode* regimes, so the parameters $Re = 60$ and $g/D = 1$ were chosen. **Note that the very loose mesh may cause a certain inaccuracy into the bifurcation's position!** The streamlines plot shows that the base flow computation have converged on the asymmetric mode and the eigenvalue diagram confirms that:

i) two *unstable* complex conjugate eigenvalues (Hopf bifurcation, in-phase shedding mode) tell us that our Re and g/D parameters are pointing to an instationary region.

ii) a real *stable* eigenvalue close to the imaginary axis (Pitchfork bifurcation) represents a stationary mode. It's probably the asymmetric mode, positioned nearby the imaginary axis for the parameters are not very close to the bifurcation point.

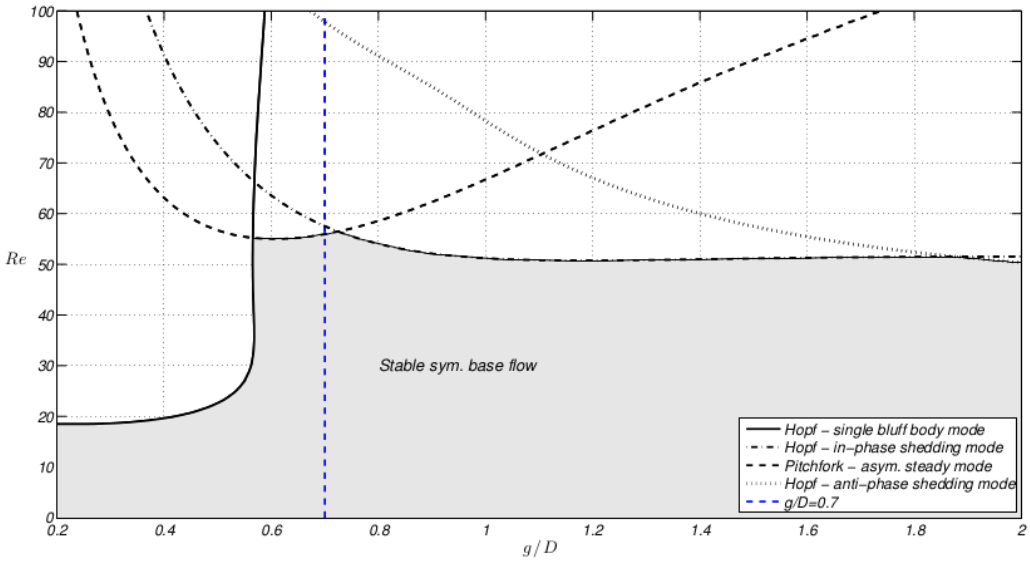


Figure 10: *Bifurcation Diagram from [4].*

Structural sensitivity was computed for the modes associated to the most unstable **real** eigenvalue. A more trustworthy computation would require to be closer to the bifurcating point (that means less unstable eigenvalues).

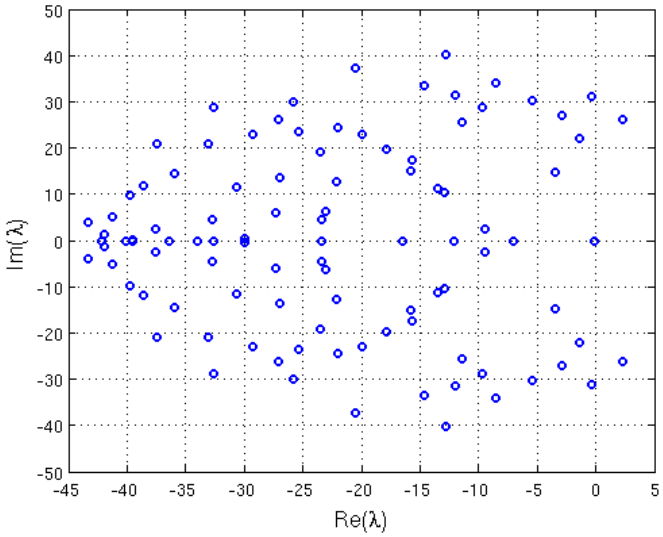


Figure 11: *100 eigenvalues for $Re = 60$, $g/D = 1$.*

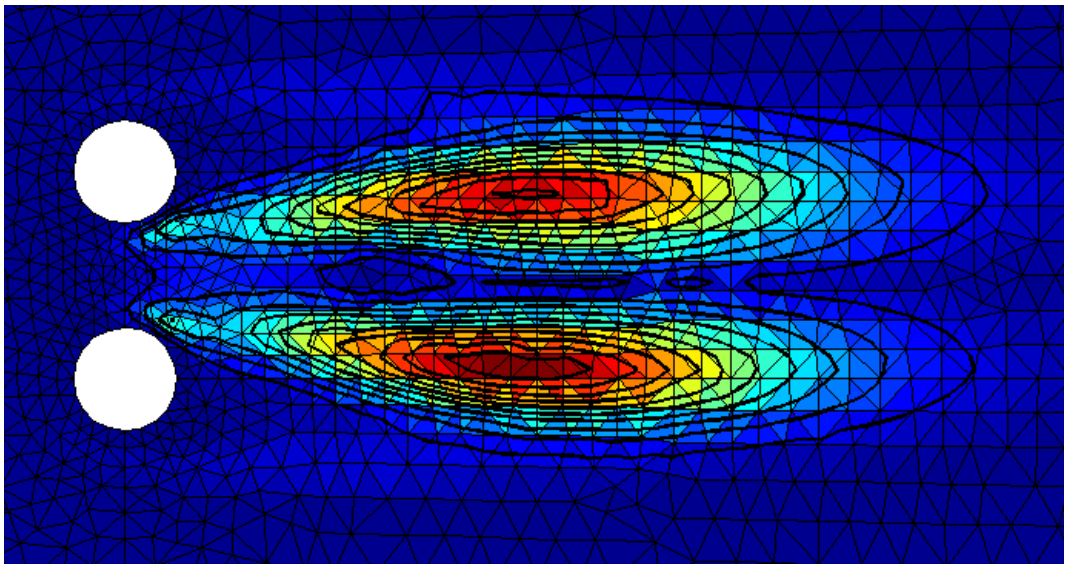


Figure 12: *Structural Sensitivity for the Pitchfork eigenvalue.*

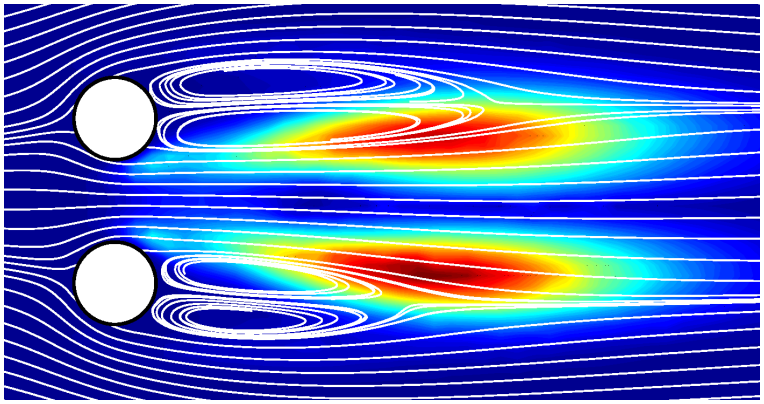


Figure 13: *Structural Sensitivity overimposed on base flow streamlines.*

Note that: in figure 13, streamlines cross the bodies: it's just a numerical error due to an inaccurate streamline plotting!

7 Final Remarks

The obtained results show up to be qualitatively correct, whereas much better results would be obtained by refining the mesh, augmenting the calculation domain and by choosing higher order polynomials for finite elements shape functions (at least, we should use P1//P2 elements).
The most computationally hard part was:

for small-size meshes, the linear system creation by the matrix products $Q_2 K Q_2^T$ and $Q_2 M Q_2^T$ was the most time-consuming part

for bigger meshes, the eigenpairs computation filled completely my 4Gb RAM, eventually restarting the operating system window manager.

Note that FreeFem++ also permits to calculate eigenpairs: a direct calculation by FreeFem++ will probably result in a big saving in computation time.

Calculations were performed on a Thinkpad T410 laptop, Intel Core i5, 4Gb RAM and the shown simulations took around 4 to 6 hours to be performed.

References

[1] Matthew W.Brewster, Matthias K.Gobbert, *A Comparative Evaluation of Matlab, Octave, FreeMat, and Scilab on Tara*, Technical Report HPCF-2011-10

[2] K.A.Cliffe, T.J.Garratt, A.Spence, *Eigenvalues of Block Matrices Arising From Problems in Fluid Mechanics*, 1993, Center for Research on Parallel Computation

[3] F.Giannetti, P.Luchini, *Structural sensitivity of the first instability of the cylinder wake*, Journal of Fluid Mechanics, **581**, 2007

[4] M.Carini, F.Auteri, F.Giannetti *Codimension 2 bifurcation analysis of the flow past two side-by-side cylinders*, European Fluid Mechanics Conference 9, 2012