

Hydrodynamic stability analysis using *Nektar++*

AIM-ED

23rd August 2011

The aim of this tutorial is to introduce the user to the spectral/*hp* element framework *Nektar++* and its use in hydrodynamic stability analysis. Information on how to install the libraries, solvers, and utilities on your own computer is available on the webpage¹. However, for this tutorial, the software is preinstalled on the laboratory computers.

In the first section we will cover the stability analysis of a two-dimensional channel flow, through both a splitting scheme and the Stokes algorithm. We will then study the transient growth of the flow past a backward-facing step and the direct/adjoint stability analysis of a flow past a cylinder.

Task 0.1

A number of data files are provided for use in this tutorial. These are located in the read-only folder `/export/aim/nektutorial/`. You should copy this directory, and all the files within, to your home directory before continuing, using the following command:

```
cp -a /export/aim/nektutorial/ $HOME/
```

1 Two-dimensional Channel flow

Linear stability analysis is a technique that allows us to determine the asymptotic stability of a flow. By decomposing the velocity and pressure in the Navier-Stokes equations as a summation of a base flow (\mathbf{U}, \mathbf{P}) and perturbation $(\mathbf{u}', \mathbf{v}')$, such that $\mathbf{u} = \mathbf{U} + \epsilon \mathbf{u}'$, $p = P + \epsilon p'$, with $\epsilon \ll 1$, we derive the linearised Navier-Stokes equations,

$$\frac{\partial \mathbf{u}'}{\partial t} + \mathbf{U} \cdot \nabla \mathbf{u}' + \mathbf{u}' \cdot \nabla \mathbf{U} = -\nabla p' + \frac{1}{Re} \nabla^2 \mathbf{u}' + \mathbf{f}', \quad (1)$$

$$\nabla \cdot \mathbf{u}' = 0. \quad (2)$$

We will consider a parallel base flow through a 2-D channel (known as Poiseuille flow) at Reynolds number $Re = 7500$. The velocity has the following analytic form:

$$\mathbf{U} = (y + 1)(1 - y)\mathbf{e}_x \quad (3)$$

¹www.nektar.info

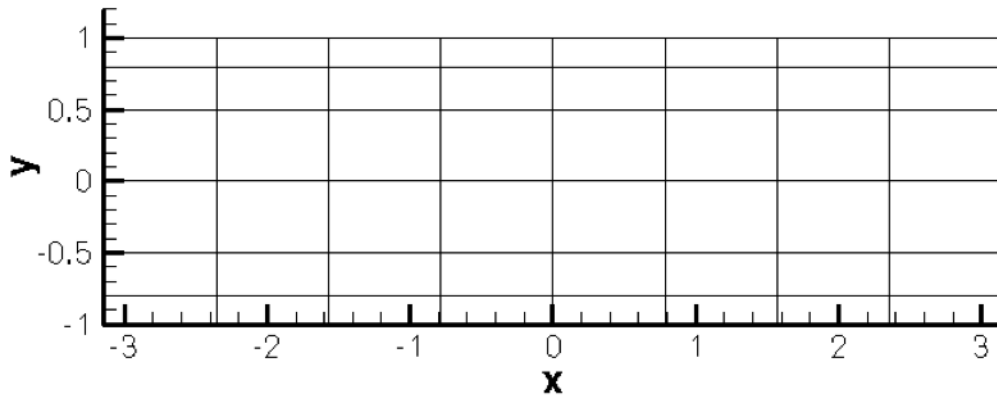


Figure 1: 48 quadrilaterals mesh

The domain is $\Omega = [-\pi, \pi] \times [-1, 1]$ and it is composed by 48 quadrilateral elements as shown in figure 1. The problem has been made non-dimensional using the centreline velocity and the channel half-height.

This mesh was created using the software *Gmsh* and the first step is to convert it into a suitable input format so that it can be processed by the *Nektar++* libraries.

In the tutorial folder **Channel/Geometry** you will find the following files:

- **Channel.geo**- *Gmsh* file containing the geometry of the problem
- **Channel.msh**- *Gmsh* generated mesh data listing mesh vertices and elements.
- **Channel.xml** - *Nektar++* session file generated from **Channel.msh** using a *Nektar++* utility.

Task 1.1

Convert the *Gmsh* geometry provided into the XML *Nektar++* format

- **Channel.msh** can be generated using *Gmsh* by running the following command:

```
gmsh -2 Channel.geo
```

- **Channel.xml** can be generated using the **MeshConvert** pre-processing tool:

```
MeshConvert Channel.msh Channel.xml
```

Examine the **Channel.xml** file you have just created. Only the mesh and default expansions are defined at present.

1.1 Computation of the base flow

We must first create an appropriate base flow. Since, in hydrodynamic stability theory, it is assumed that the base flow is incompressible, it can be computed by using the incompressible Navier-Stokes solver (**IncNavierStokesSolver**).

The specified boundary conditions will be no-slip on the walls and periodic for the inflow/outflow. In this case, since it is not a constant pressure gradient that drives the flow, it is necessary to use a constant body-force in the streamwise direction. It can be shown that this should be equal to 2ν .

In the folder **Channel/Base** you will find the file **Channel-Base.xml** which contains the geometry described above along with the necessary parameters to solve the problem. The **GEOMETRY** section defines the mesh of the problem and it is generated automatically as you have seen in the previous task. The expansion type and order is specified in the **EXPANSIONS** section. An expansion basis is applied to a geometry composite² specified in the **GEOMETRY** section. A default entry is always included by the **MeshConvert** utility. In this case the composite **C[0]** refers to the set of all elements. The **FIELDS** attribute specifies the fields for which this expansion should be used. The **TYPE** attribute specifies the kind of the polynomial basis functions to use in the expansion. For example,

```
<EXPANSIONS>
  <E COMPOSITE="C[0]" NUMMODES="8" FIELDS="u,v,p" TYPE="MODIFIED"/>
</EXPANSIONS>.
```

If we examine **Channel-Base.xml**, we can see how to define the conditions of the particular problem to solve. These are all enclosed in a **CONDITIONS** section. This section contains a number of entries:

1. **Solver information** (**SOLVERINFO**) such as the equation, the projection type (**Continuous** or **Discontinuous Galerkin**), the evolution operator (**Nonlinear** for non-linear Navier-Stokes, **Direct**, **Adjoint** or **TransientGrowth** for linearised forms) and the analysis driver to use (**Standard**, **Arpack** or **ModifiedArnoldi**), along with other properties. The solver properties are specified as quoted attributes and have the form

```
<I PROPERTY="[STRING]" VALUE="[STRING]" />
```

Task 1.2

In the **SOLVERINFO** section of **Channel-Base.xml**:

- set **EQTYPE** to **UnsteadyNavierStokes** to select the unsteady incompressible Navier-Stokes equations,
- set the **EvolutionOperator** to select the non-linear Navier-Stokes,
- set the **Projection** property to continuous Galerkin,
- set the **Driver** to perform standard time-integration.

2. The **parameters** are specified as name-value pairs:

```
<P> [KEY] = [VALUE] </P>
```

Parameters may be used within other expressions, such as function definitions, boundary conditions or the definition of other subsequently defined parameters.

²by composite we mean a collection of mesh entities, but we specifically require a collection of mesh elements here

Task 1.3

Declare parameters **Re** and **Kinvis** which sets the Reynolds number to 7500 and the kinematic viscosity, ν , to $1/Re$.

3. The declaration of the **variable(s)**.

```
<VARIABLES>
  <V ID="0"> u </V>
  <V ID="1"> v </V>
  <V ID="2"> p </V>
</VARIABLES>
```

4. The specification of **boundary regions** in terms of composites defined in the **GEOMETRY** section and the conditions applied on those boundaries. Boundary regions have the form

```
<B ID="[INDEX]"> [COMPOSITE-ID] </B>
```

The **boundary conditions** enforced on a region take the following format and must define the condition for each variable specified in the **VARIABLES** section to ensure the problem is well-posed.

```
<REGION REF="[B-REGION-INDEX]">
  <[TYPE] VAR="[VARIABLE_1]" VALUE="[EXPRESSION_1]" />
  <[TYPE] VAR="[VARIABLE_2]" VALUE="[EXPRESSION_2]" />
  ...
</REGION>
```

The **REF** attribute for a boundary condition region should correspond to the **ID** of the desired **BOUNDARYREGION**.

5. The definition of the (time- and) space-dependent functions, in terms of x , y , z and t , such as initial conditions, forcing functions, and exact solutions.

```
<FUNCTION NAME="[NAME]">
  <E VAR="[VARIABLE_1]" VALUE="[EXPRESSION]" />
  <E VAR="[VARIABLE_2]" VALUE="[EXPRESSION]" />
  ...
</FUNCTION>
```

Alternatively, one can specify the function using an external *Nektar++* field file.

```
<FUNCTION NAME="[NAME]">
  <F FILE="[FILENAME]" />
</FUNCTION>
```

Task 1.4

Define a body forcing function in the streamwise direction (called **BodyForce**): $\mathbf{f} = 2\nu\mathbf{e}_x$.

Task 1.5

Define a function called `ExactSolution`. For the Poiseuille flow with a streamwise forcing term the exact solution is:

$$U = (y + 1)(1 - y) \quad (4)$$

$$V = 0 \quad (5)$$

$$P = 0 \quad (6)$$

Note: We have not specified an initial condition, since it will be zero by default.

This completes the specification of the problem.

Task 1.6

Compute the base flow using the `Channel-Base.xml` session file:

```
IncNavierStokesSolver Channel-Base.xml
```

At the end of the simulation, the fields will be written to a binary file `Channel-Base.fld` and the L_2 error (using the given exact solution) and the L_∞ error will be printed on the terminal for each of the variables.

To visualise the flow fields, we will convert the `.fld` file into VTK format and use *Paraview*.

Task 1.7

Convert the file:

```
FldToVtk Channel-Base.xml Channel-Base.fld
```

Now run *Paraview*:

```
paraview
```

Use File ->Open, to select the VTK file, click the 'Apply' button to render the geometry, and select each field in turn from the left-most drop-down menu on the toolbar to visualise the output.

1.2 Stability analysis

After computing the base flow it is now possible to calculate the eigenvalues and the eigenmodes of the linearised Navier-Stokes equations. Two different algorithms can be used to solve the equations: the splitting scheme (`VelocityCorrectionScheme`) and the Stokes algorithm (`CoupledLinearisedNS`). We will consider both cases, highlighting the similarities and differences of these two methods. In this tutorial we will use the Implicitly Restarted Arnoldi Method (IRAM), which is implemented in the open-source *ARPACK* library. Alternatively, one can use the modified Arnoldi algorithm.³

³Int. J. Numer. Meth. Fluids, 2008; **57**:1435-1458

First, we will compute the leading eigenvalues and eigenvectors using the splitting scheme method. In the **Channel/Stability** folder there is a file called **Channel_VCS.xml**. This is similar to **Channel.xml**, but contains additional instructions to perform the direct stability analysis.

Note: The entire **GEOMETRY** section, and **EXPANSIONS** section must be identical to that used to compute the base flow.

Task 1.8

Configure the following additional options and run the linear stability analysis for the channel:

1. **EvolutionOperator** must now be set to **Direct** to activate the forward linearised Navier-Stokes system.
2. **Driver** must be set to **Arpack** to use *Arpack* eigenvalue analysis.
3. A restart file is provided to accelerate communications. Set the **InitialConditions** function to be read from **Channel.rst**. The solution will then converge after 16 iterations.
4

Note: The restart file is a field file (same format as **.fld** files) that contains the eigenmode of the system.

4. The base flow file (**Channel-Base.fld**), computed in the previous section, was copied into the **Channel/Stability** folder and renamed **Channel_VCS.bse**.

Now specify a function called **BaseFlow** which reads this file.

5. We can instruct ARPACK to converge onto specific eigenvalues through the solver property **ArpackProblemType**: the ones with the largest magnitude (**LargestMag**), largest real part (**LargestReal**) or largest imaginary part (**LargestImag**). In our case we are interested in computing the eigenvalues with the largest magnitude that determinate the stability of the flow.
6. Set the parameters for the IRAM algorithm:
 - **kdim=16**: dimension of Krylov-space,
 - **nvec=2**: number of requested eigenvalues,
 - **nits=500**: number of maximum allowed iterations,
 - **evtol=1e-6**: accepted tolerance on the eigenvalues and it determines the stopping criterion of the method.

Run the solver to perform the analysis

```
IncNavierStokesSolver Channel_VCS.xml
```

The eigenvalues are computed in the exponential form $Me^{i\theta}$ where $M = |\lambda|$ is the magnitude, while $\theta = \arctan(\lambda_i/\lambda_r)$ the phase.

$$\lambda_{1,2} = 1.00224e^{\pm 0.24984i} \quad (7)$$

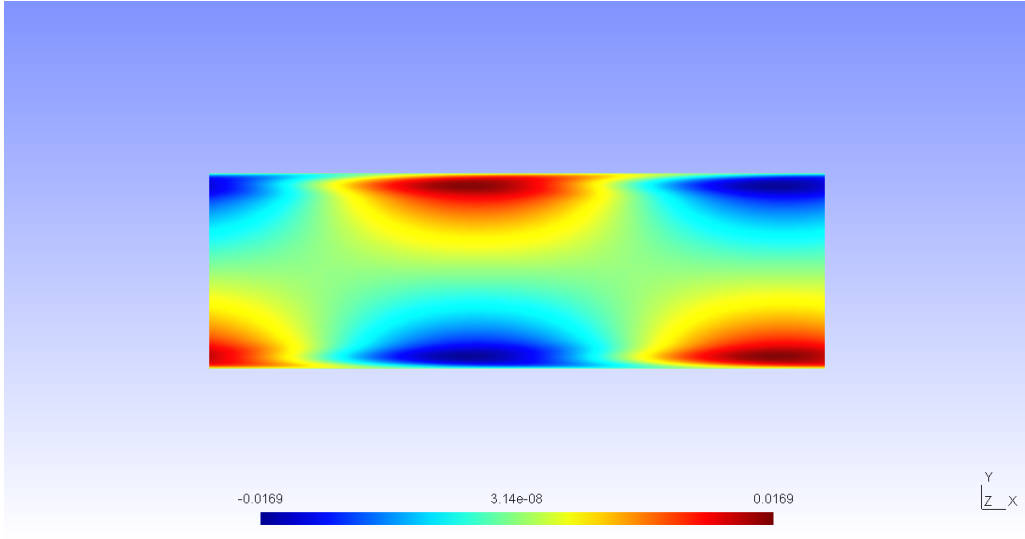


Figure 2: u' -component of the eigenmode

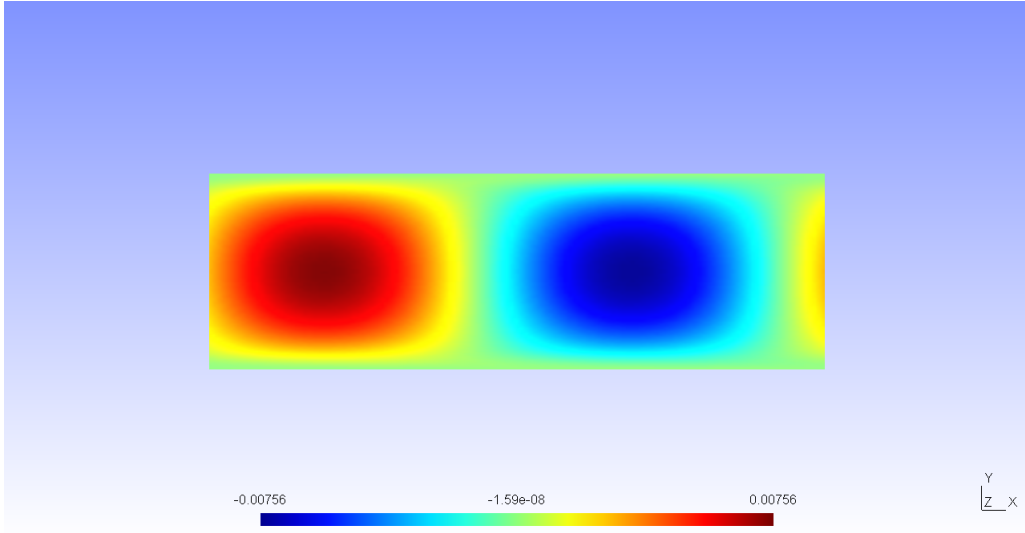


Figure 3: v' -component of the eigenmode

It is interesting to consider more general quantities that do not depend on the time length chosen for each iteration T . For this purpose we consider the growth rate $\sigma = \ln(M)/T$ and the frequency $\omega = \theta/T$.

Figures 2 and 3 show the profile of the computed eigenmode. The eigenmodes associated with the computed eigenvalues are stored in the files `Channel_VCS_eig_0` and `Channel_VCS_eig_1`. It is possible to convert this file into VTK format in the same way as previously done for the base flow earlier.

Task 1.9

Verify that for the channel flow case :

$$\sigma = 2.23711 \times 10^{-3}$$
$$\omega = \pm 2.498413 \times 10^{-1}$$

and that the eigenmodes match those given in figures 2 and 3.

This values are in accordance with the literature, in fact in Canuto et al., 1988 suggests 2.23497×10^{-3} and 2.4989154×10^{-1} for growth and frequency, respectively.

1.2.1 Stokes Algorithm

It is possible to perform the same stability analysis using a different method based on the Stokes algorithm. This method requires the solution of the full velocity-pressure system, meaning that the velocity matrix system and the pressure system are coupled, in contrast to the splitting/projection schemes. It is easy to extend this method to solve the unsteady Navier-Stokes equations introducing into the Stokes problem an unsteady term \mathbf{u}_f that modifies the weak Laplacian operator into a weak Helmholtz operator. Furthermore, the non-linear terms are explicitly advanced in time and treated as a forcing function to the Stokes solver.

Inside the folder `Channel/Stability` there is a file called `Channel.Coupled.xml` that contains all the necessary parameters that should be defined. As with the previous example, it is possible to specify the base flow putting the `Channel.Coupled.bse` in the working directory. However, it is also possible to specify the base flow directly from session file if it has an analytical expression like in our simple case. This second method will be presented in this section. Even in this case, the geometry, the type and number of modes are the same of the previous simulations.

Task 1.10

Edit the file `Channel.Coupled.xml`:

- Set the `SolverType` property to `CoupledLinearisedNS` in order to solve the linearised Navier-Stokes equations using *Nektar++*'s coupled solver.
- the `EQTYPE` must be set to `SteadyLinearisedNS` and the `Driver` to `Arpack`.
- Set the `InitialVector` property to `Random` to initialise the IRAM with a random initial vector. In this case the function `InitialConditions` will be ignored.
- To compute the eigenvalues with the largest magnitude, specify `LargestMag` in the property `ArpackProblemType`.

It is important to note that the use of the coupled solver requires that **only the velocity component variables** are specified, while the pressure can be directly computed through their values.

Task 1.11

Continue modifying `Channel_Coupled.xml`:

- In this case it is a body force that provides the driving of the flow. This is done by the setting the previously introduced function `BodyForce` to $\cos(y)$ for the u component and $\sin(y)$ for the v component.
- Finally, it is necessary to set up the base flow. For the `SteadyLinearisedNS` coupled solver, this is defined through a function called `AdvectionVelocity`. The u component must be set up to $1 - y^2$, while the v -component to zero.

Now run the solver to compute the eigenvalues.

Using the Stokes algorithm, we are computing the leading eigenvalue of the inverse of the evolution operator \mathcal{L}^{-1} . Therefore the eigenvalues of \mathcal{L} are the inverse of the computed values⁵. However, it is interesting to note that these values are different from those calculated with the Splitting Scheme Method, producing an apparent inconsistency. However, this can be explained considering that the largest eigenvalues associated to the operator \mathcal{L} correspond the ones that are clustered near the origin of the complex plane if we consider the spectrum of \mathcal{L}^{-1} . Therefore, eigenvalues with a smaller magnitude may be present but are not associated with the largest magnitude eigenvalue of operator \mathcal{L} . In order to verify this issue, let us search for the eigenvalue with a different criterion, for example, the largest imaginary part.

Task 1.12

Set up the Solver Info tag `ArpackProblemType` to `LargestImag` and run the simulation again.

In this case, it is easy to see that the eigenvalues of the evolution operator \mathcal{L} are the same ones computed in the previous section with the time-stepping approach. It is interesting to note that this method converges much quicker than the time-stepping algorithm. However, building the coupled matrix that allows us to solve the problem can take a non-negligible computational time for more complex cases.

2 Backward-facing step

In this section we will perform a transient growth analysis of the flow over a backward-facing step. This is an important case which allows us to understand the effects of separation due to abrupt changes of geometry in an open flow. The transient growth analysis consists of computing the maximum energy growth, $G(\tau)$, attainable over all possible initial conditions $\mathbf{u}'(0)$ for a specified time horizon τ . It can be demonstrated that it is equivalent to calculating the largest eigenvalue of $\mathcal{A}^*(\tau)\mathcal{A}(\tau)$, with \mathcal{A} and \mathcal{A}^* being the direct and the adjoint operators, respectively. Also note that the eigenvalue must necessarily be real since $\mathcal{A}^*(\tau)\mathcal{A}(\tau)$ is self-adjoint in this case.

The files for this section can be found in the `backward-facing_step` directory.

⁵ \mathcal{L} is the evolution operator $d\mathbf{u}/dt = \mathcal{L}\mathbf{u}$

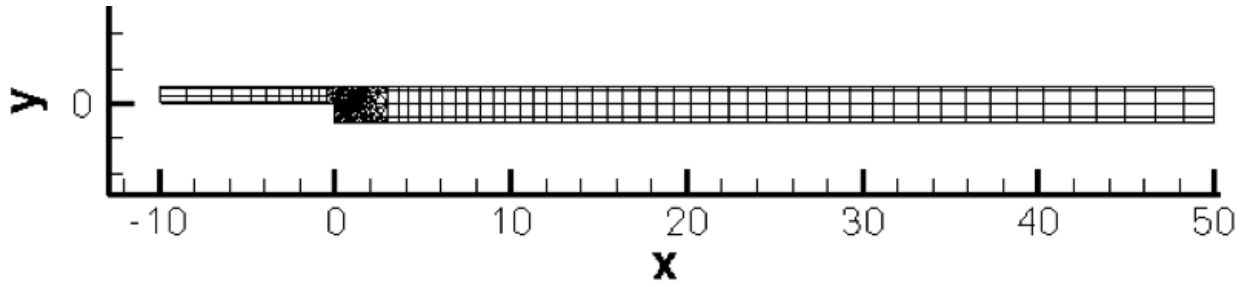


Figure 4: Mesh used for the backward-facing step

- Folder Geometry
 - `bfs.geo` - *Gmsh* file that contains the geometry of the problem
 - `bfs.msh` - *Gmsh* generated mesh data listing mesh vertices and elements.
- Folder Base
 - `bfs-Base.xml` - *Nektar++* session file, generated with the `MeshConvert` utility, for computing the base flow.
 - `bfs-Base.fld` - *Nektar++* field file that contains the base flow, generated using `bfs-Base.xml`.
- Folder Stability
 - `bfs_tg.xml` - *Nektar++* session file, generated with `MeshConvert`, for performing the transient growth analysis.
 - `bfs_tg.bse` - *Nektar++* field file that contains the base flow. It is the same as the `.fld` file present in the folder `Base`.

Figure 4 shows the mesh we will use for the computation, along with a detailed view of the step edge in figure 5. The geometry is non-dimensionalised by the step height. The domain has an inflow length of 10 upstream of the step edge and a downstream channel of length 50. The mesh consist of $N = 430$ elements. Note that in this case the mesh is composed of both triangular and quadrilateral elements. A refined triangular unstructured mesh is used near the step to capture the separation effects, whereas the inflow/outflow channels have a structure similar to the previous example. Therefore in the `EXPANSION` section of the `bfs-Base.xml` file, two composites (`C[0]` and `C[1]`) are present. For this example, we will use the `MODIFIED` basis with 7th-order polynomials.

We will perform simulations at $Re = 500$, since it is well-known that for this value the flow presents a strong convective instability.

The file `bfs.bse` is the output of the base-flow computation that should be run for a non-dimensional time of $t \geq 300$ to ensure that the solution is steady.

Task 2.1

Convert the base flow field file `bfs.bse` into VTK format to look at the profile of the base flow. Note the separation at the step-edge and the reattachment downstream.

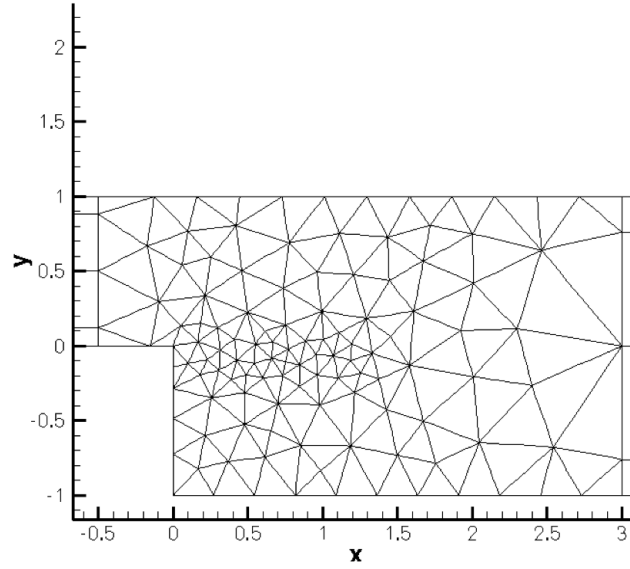


Figure 5: Detail of the mesh used for the backward-facing step nearby the step

We will now perform transient growth analysis with a Krylov subspace of `kdim=4`. The parameters and properties needed for this are present in the file `bfs_tg.xml` in `backward-facing_step/Stability`. In this case the `Arpack` library was used to compute the largest eigenvalue of the system and the corresponding eigenmode. We will compute the maximum growth for a time horizon of $\tau = 1$, usually denoted $G(1)$.

Task 2.2

Configure the `bfs_tg.xml` session for performing transient growth analysis:

- Set the `EvolutionOperator` to `TransientGrowth`.
- Define a parameter `FinalTime` that is equal to 1 (this is the time horizon).
- Set the number of steps (`NumSteps`) to be the ratio between the final time and the time step.
- Since the simulations take several iterations to converge, use the restart file `bfs_tg.rst` for the initial condition. This file contains an eigenmode of the system.

Now run the simulation

```
IncNavierStokesSolver bfs_tg.xml
```

Initially, the solution will be evolved forward in time using the operator \mathcal{A} , then backward in time through the adjoint operator \mathcal{A}^* . It will converge quickly after 4 iterations and the leading eigenvalue should be $\lambda = 3.236204$. This corresponds to the largest possible transient growth at the time horizon $\tau = 1$. The leading eigenmode is shown in figures 6 and 7. This is the optimal initial condition which will lead to the greatest growth when evolved under the linearised Navier-Stokes equations.

We can visualise graphically the optimal growth, recalling that the energy of the perturbation field any given time t is defined by means of the inner product:

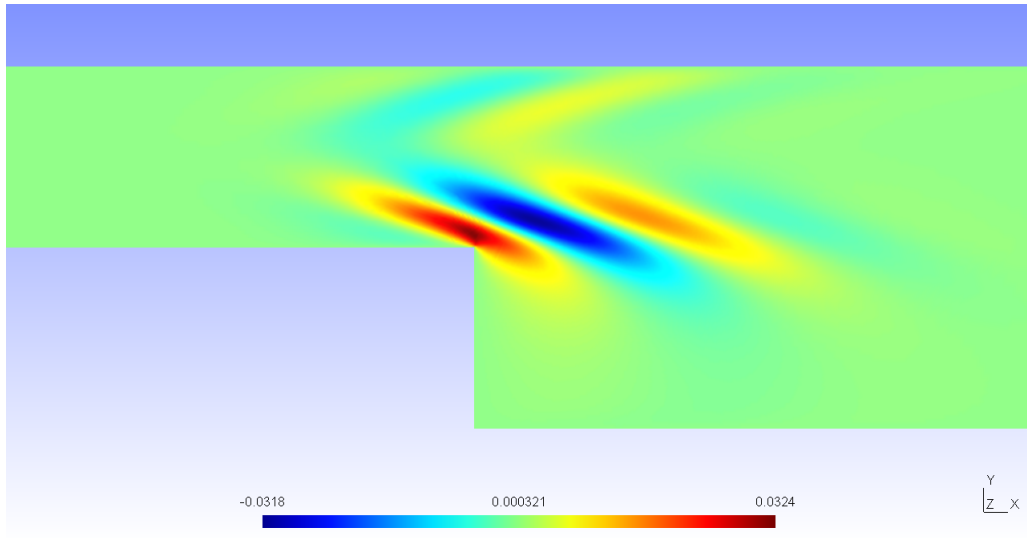


Figure 6: u' -component of the eigenmode

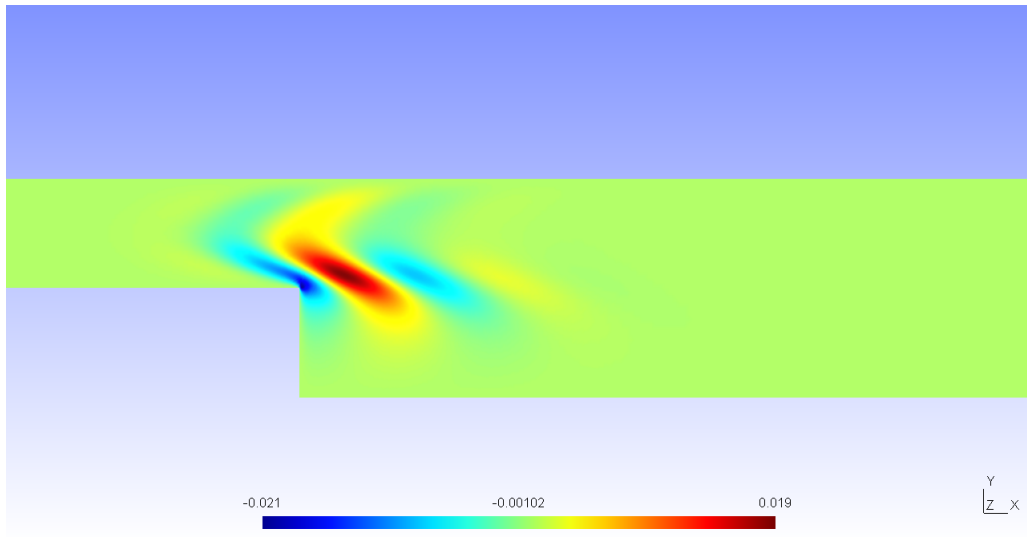


Figure 7: v' -component of the eigenmode

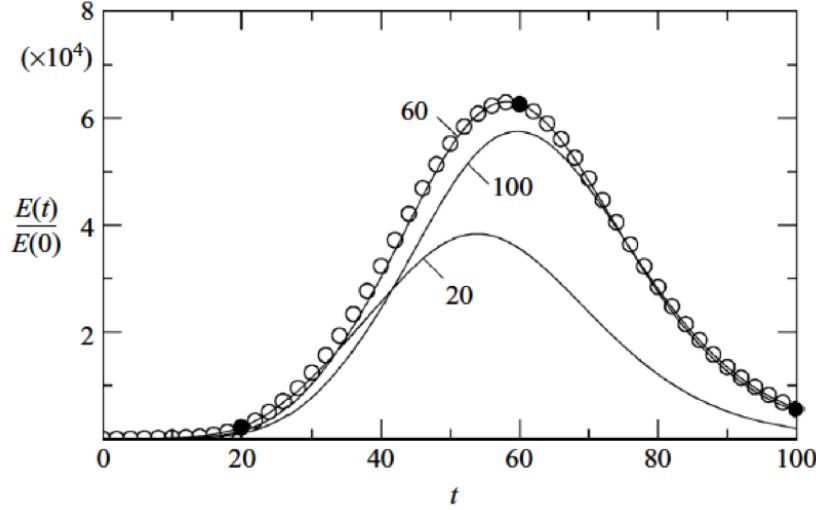


Figure 8: Envelope of two-dimensional optima at $Re = 500$ together with curves of linear energy evolution starting from the three optimal initial conditions for specific values of τ 20, 60 and 100. Figure reproduced from J. Fluid. Mech. (2008), vol 603, pp. 271-304.

$$E(\tau) = \frac{1}{2}(\mathbf{u}'(t), \mathbf{u}'(t)) \quad (8)$$

$$= \frac{1}{2} \int_{\Omega} \mathbf{u}' \cdot \mathbf{u}' dv \quad (9)$$

The solver can output the evolution of the energy of the perturbation in time by setting the parameter `IO_EnergySteps`. This will write a `.mld` file containing the energy every specified number of steps. Repeating these simulations for different τ with the optimal initial perturbation as the initial condition, it is possible to create curves like those shown in figure 8. Each curve necessarily meets the optimal growth envelope (denoted by the circles) at its corresponding value of τ , and never exceeds it.

Task 2.3

(Optional) Try generating a curve for the initial condition computed in the previous task.

Hint: You will need to switch back to using the `Standard` driver and use the `Direct` evolution operator for this task.

3 Flow past a cylinder

As a final example we will compute the direct and adjoint modes of a two-dimensional flow past a cylinder. We will investigate a case in the subcritical regime ($Re = 42$), below the on-set of the Bernard-von K rm n vortex shedding that is observed when the Reynolds number is above the critical value $Re_c \simeq 47$; this analysis is important because it allows us to study the sensitivity of the flow, much like that reported by Giannetti and Luchini (J. Fluid Mech., 2007; **592**:177-194). Due to the complex nature of the flow and the demanding computational time that is required, only

some basic information will be presented in this section, mainly to show the potential of the code for stability analysis.

In the folder `Cylinder/Geometry` there are the *Gmsh* files (`Cylinder.geo` and `Cylinder.msh`) used to create a suitable mesh for the direct and adjoint stability analysis. The mesh is shown in figure 9 and a detailed view around the cylinder is shown in figure 10. This mesh is made up of 780 quadrilateral elements.

Note: It is important to note that stability and transient growth calculations, in particular, have a strong dependence on the domain size as reported by Cantwell and Barkley (Physical Review E, 2010; **82**); moreover, poor mesh design can lead to incorrect results. Specifically, the mesh must be sufficiently refined around the cylinder in order to capture the separation of the flow and abrupt variations in the size of the elements should be avoided.

`Cylinder-Base.xml` can be found inside the `Cylinder/Base` folder. This is the *Nektar++* file generated using *MeshConvert* and augmented with all the configuration settings that are required. In this case, CFL conditions can be particularly restrictive and the time step must be set around 8×10^{-4} . We will be using Reynolds number $Re = 42$ for this study.

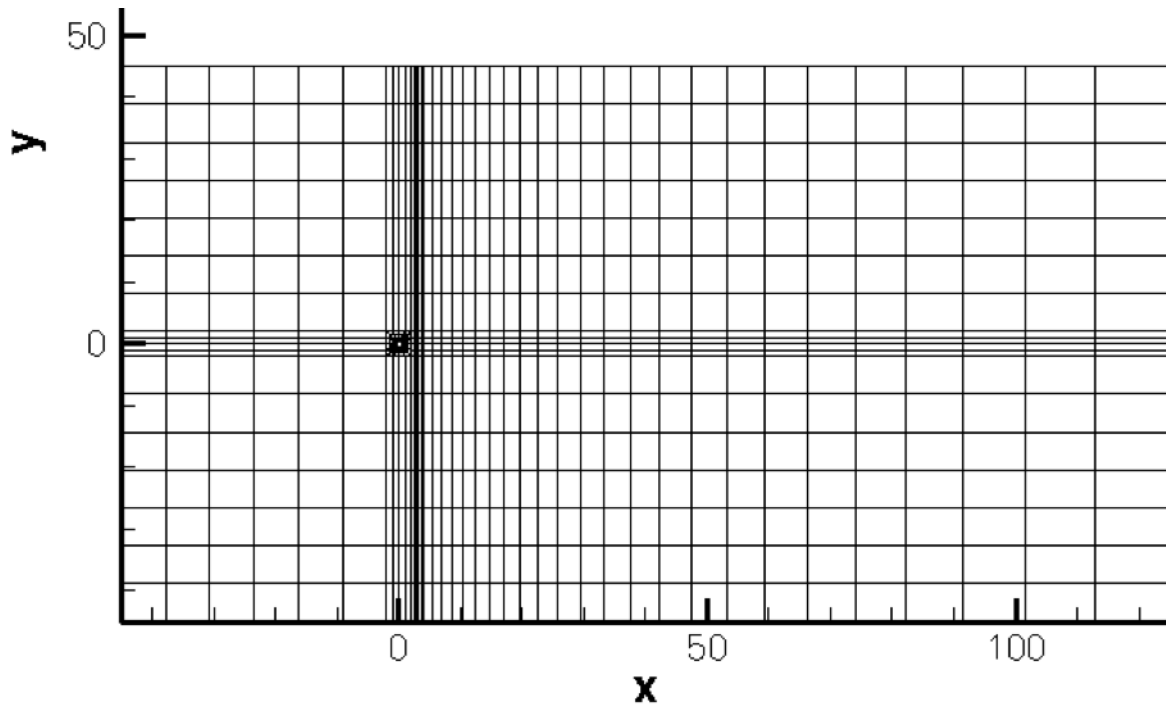


Figure 9: Mesh used for the direct stability analysis

The supplied file `Cylinder-Base.bse` is the converged base flow required for the analysis and is the result of running `Cylinder-Base.xml`. To have a steady solution it was necessary to evolve the fields for a non-dimensional time $\tau \geq 300$ and it is very important to be sure that the solution is steady. This can be verified by putting several history points on the centre line of the flow and monitoring their variation.

Task 3.1

Convert the base flow into VTK format and visualise the profile of the flow past a cylinder in *Paraview*.

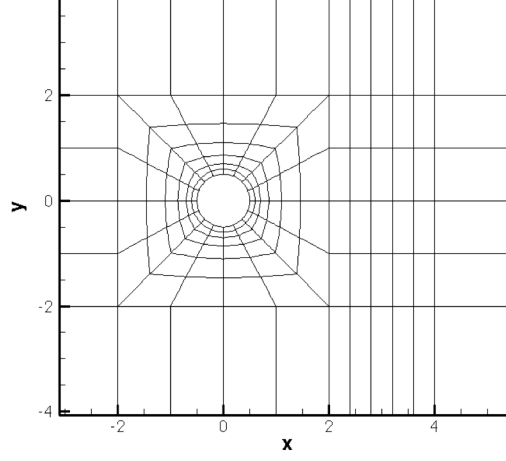


Figure 10: Detail of the mesh around the cylinder

In the folder `Cylinder/Stability/Direct` there are the files that are required for the direct stability analysis. Since, the computation would normally take several hours to converge, we will use a restart file and a Krylov-space of just $\kappa = 4$. Therefore, it will be possible to obtain the eigenvalue and the corresponding eigenmode after 4 iterations.

Task 3.2

Define a Kyrlov space of 4 and compute the eigenvalues and the eigenvectors of the problem using the restart file `Cylinder_Direct.rst`. Plot the leading eigenvector in *Paraview*. This should look like the solution shown in figures 11 and 12.

The leading eigenvalues show a growth rate of $\sigma = -2.101955 \times 10^{-2}$ and a frequency $\omega = \pm 7.265859 \times 10^{-1}$.

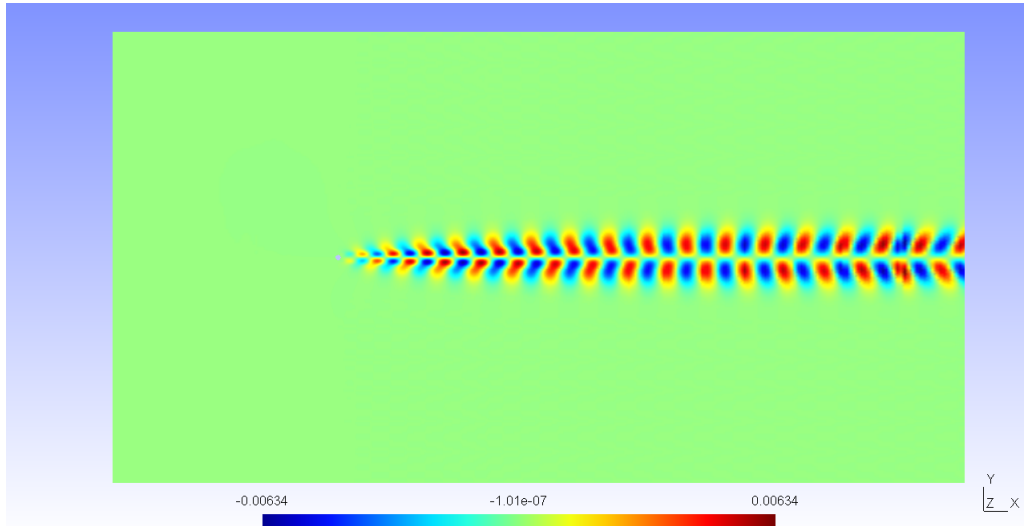


Figure 11: u' -component of the eigenmode

After the direct stability analysis, it is now interesting to compute the eigenvalues and eigenvectors of the adjoint operator \mathcal{A}^* that allows us to evaluate the effects of generic initial conditions and

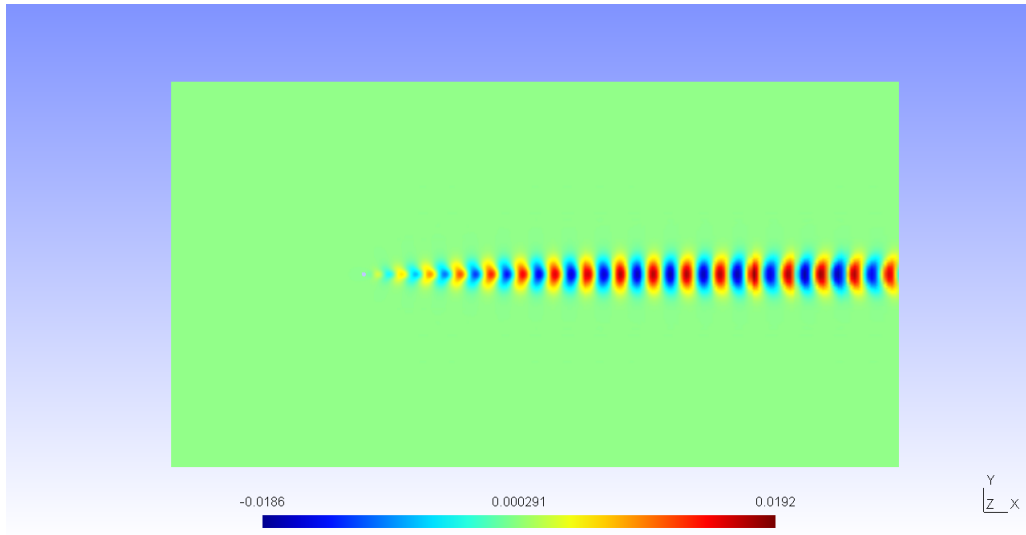


Figure 12: v' -component of the eigenmode

forcing terms on the asymptotic behaviour of the solution of the linearised equations. In the folder `Cylinder/Stability/Adjoint` there is the file `Cylinder_Adjoint.xml` that is used for the adjoint analysis.

Task 3.3

Set the `EvolutionOperator` to `Adjoint`, the Krylov space to 4 and compute the leading eigenvalue and eigenmode of the adjoint operator, using the restart file `Cylinder_Adjoint.rst`

Plot the leading eigenmode in *Paraview*.

The solution should converge after 4 iterations, giving the eigenvalues of the system as $\lambda_{1,2} = 0.980495 \times e^{\pm i0.727502}$ with a growth rate equal to $\sigma = -1.969727 \times 10^{-2}$ and a frequency $\omega = \pm 7.275024 \times 10^{-1}$. However, the most interesting thing to observe is the shape of the eigenmode. In particular, in spatially developing flows the eigenmodes of the direct stability operator tend to be located downstream while the eigenmodes of the adjoint operator tend to be located upstream, as can be seen in figures 13 and 14. From the profiles of the eigemodes, it can be deduced that the regions with the maximum receptivity for the momentum forcing and mass injection are near the wake of the cylinder, close to the upper and lower sides of the body surface, in accordance with results reported in the literature.

This completes the tutorial.

4 Installing *Nektar++* on your computer.

If you are interested in pursuing this sort of work further, you can install *Nektar++* on your own laptop by downloading and compiling the source code from the website: www.nektar.info. Detailed installation instructions are provided on the website and you are welcome to ask for help if you get stuck!

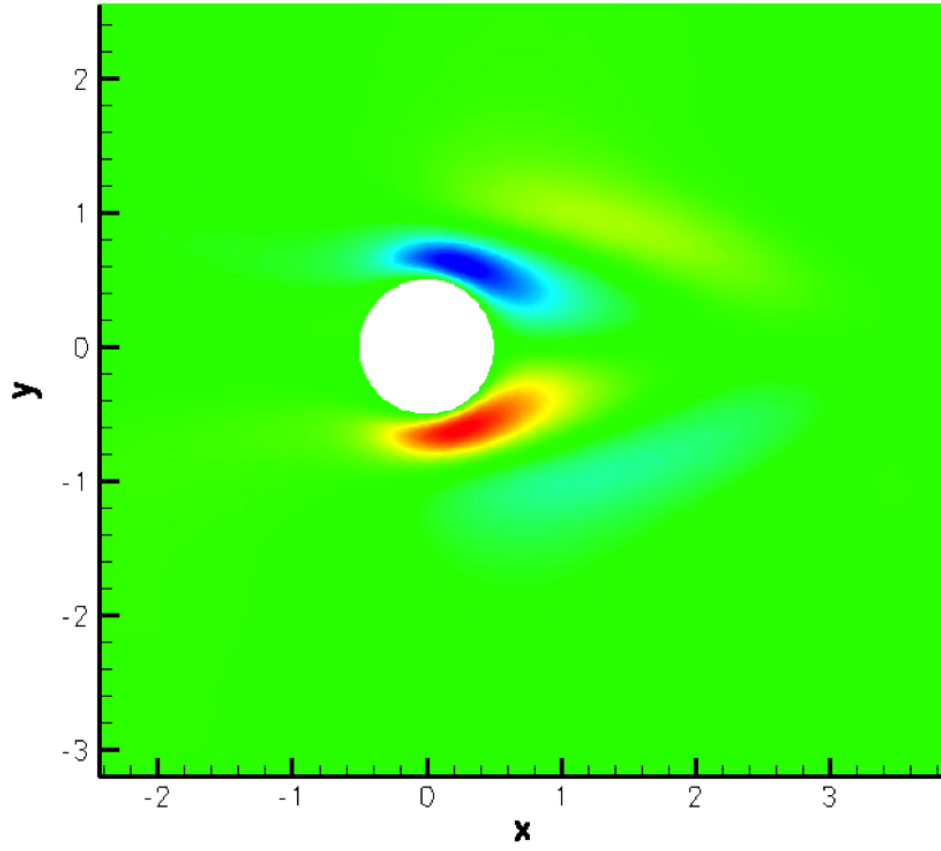


Figure 13: Close-up of the u^* -component of the adjoint eigenmode.

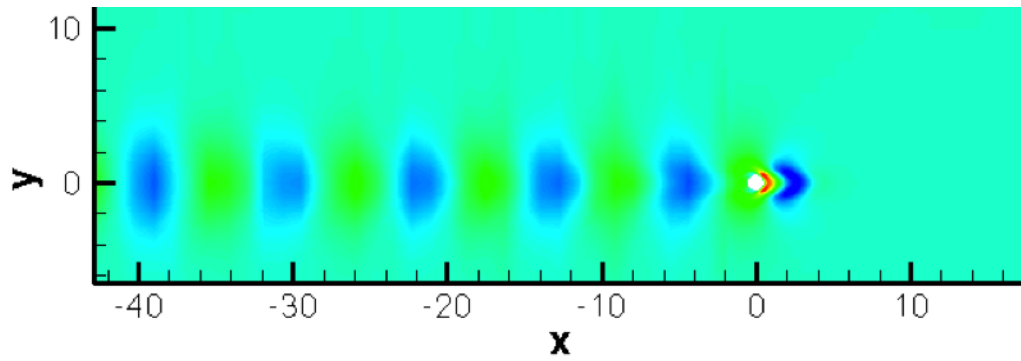


Figure 14: The v^* -component of the adjoint eigenmode extends far upstream of the cylinder