Global Stability Analysis: Channel Flow

Tutorials

July 4, 2026

Department of Aeronautics, Imperial College London, UK
Scientific Computing and Imaging Institute, University of Utah, USA

2

Chapter 1

Introduction

The aim of this tutorial is to introduce the user to the spectral/*hp* element framework *Nektar++* and its use for global stability computations. Information on how to install the libraries, solvers, and utilities on your own computer is available on the webpage www.nektar.info.

Task 1.1

Prepare for the tutorial. Make sure that you have:

- Installed and tested *Nektar++* v5.10.0from a binary package, or compiled it from source. By default binary packages will install all executables in `/usr/bin`. If you compile from source they will be in the sub-directory `dist/bin` of the `build` directory you created in the *Nektar++* source tree. We will refer to the directory containing the executables as `$NEK` for the remainder of the tutorial.
- Downloaded the tutorial files: `http://doc.nektar.info/tutorials/5.10.0/flow-stability/channel/flow-stability-channel.tar.gz` Unpack it using `unzip flow-stability-channel.tar.gz` to produce a directory `flow-stability-channel` with subdirectories called `tutorial` and `complete` We will refer to the `tutorial` directory as `$NEKTUTORIAL`.

Task 1.2

Additionally, you should also install

- a visualization package capable of reading VTK files, such as ParaView (which can be downloaded from here) or VisIt (downloaded from here). Alternatively, you can generate Tecplot formatted .dat files for use with Tecplot.
- a plotting program capable of reading data from ASCII text files, such as GNUPlot or MATLAB.

Optionally, you can install the open-source mesh generator *Gmsh* (which can be downloaded from here) to generate the meshes for the tutorial examples yourself. However, pre-generated meshes are provided.

In this tutorial, we will cover the stability analysis of a two-dimensional channel flow, through both a splitting scheme (the Velocity Correction Scheme) and the direct inversion algorithm (also referred to as the Coupled Linearised Navier-Stokes solver). We will briefly show the application of the stability tools presented to a three-dimensional channel flow test case.

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 $(\mathbf{U}, P)$ and perturbation $(\mathbf{u}', p')$, such that $\mathbf{u} = \mathbf{U} + \epsilon\mathbf{u}'$, $p = P + \epsilon p'$, with $\epsilon \ll 1$, we derive the linearised Navier-Stokes equations,

$$\frac{\partial \mathbf{u}'}{\partial t} + \mathbf{U}\cdot\nabla\mathbf{u}' + \mathbf{u}'\cdot\nabla\mathbf{U} = -\nabla p' + \frac{1}{Re}\nabla^2\mathbf{u}' + \mathbf{f}', \tag{1.1}$$

$$\nabla\cdot\mathbf{u}' = 0. \tag{1.2}$$

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:

$$\mathbf{U} = (y+1)(1-y)\mathbf{e_x} \tag{1.3}$$

The domain is $\Omega = [-\pi, \pi] \times [-1, 1]$ and it is composed by 48 quadrilateral elements as shown in figure 1.1. 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.

The files in the `$NEKTUTORIAL` directory are as follows:

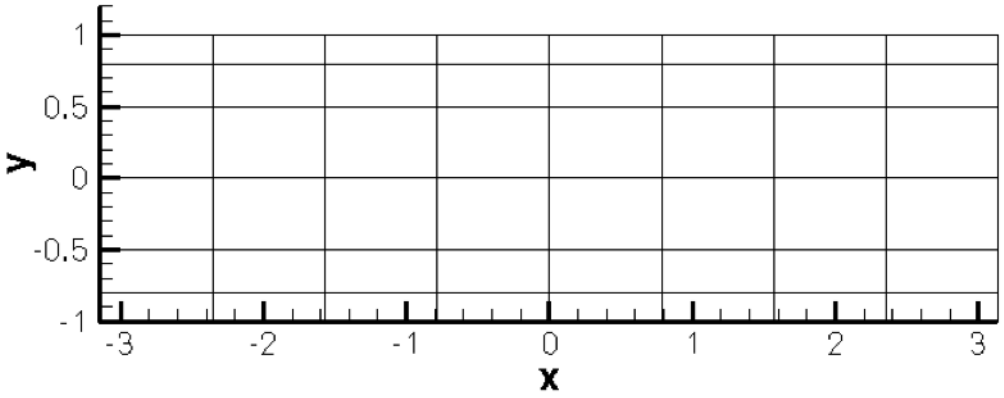

Figure 1.1 48 quadrilaterals mesh

- Folder `geometry`
  - `Channel.geo` - *Gmsh* file that contains the geometry of the problem
  - `Channel.msh` - *Gmsh* generated mesh data listing mesh vertices and elements.
- Folder `base`
  - `Channel-Base.xml` - *Nektar++* session file, generated with the `$NEK/NekMesh` utility, for computing the base flow.
- Folder `stability/VCS`
  - `Channel-VCS.xml` - *Nektar++* session file, generated with `$NEK/NekMesh`, for performing the stability analysis.
  - `Channel-VCS.rst` - *Nektar++* field file that contains a set of initial conditions closer to the solution in order to achieve faster convergence.
- Folder `stability/Coupled`
  - `Channel-Coupled.xml` - *Nektar++* session file, generated with `$NEK/NekMesh`, for performing the stability analysis.
- Folder `stability3D`
  - `PPF_R10000_3D.xml` - *Nektar++* session file for performing coupled 3D stability analysis at $Re = 10000$.
  - `PPF_R15000_3D.xml` - *Nektar++* session file for performing coupled 3D stability analysis at $Re = 15000$.

1.1 Mesh generation

The first step is to generate a mesh that is readable by *Nektar++*. The files necessary in this section can be found in `$NEKTUTORIAL/geometry/`. To achieve this task we use *Gmsh* in conjunction with the *Nektar++* pre-processing utility called `$NEK/NekMesh`. Specifically, we first generate the mesh in figure 1.1 using *Gmsh* and successively we convert it into a suitable *Nektar++* format using `$NEK/NekMesh`.

Task 1.3

Convert the *Gmsh* geometry provided into the XML *Nektar++* format and with two periodic boundaries

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

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

- `Channel.xml` can be generated using the `$NEK/NekMesh` pre-processing tool:

  ```
  $NEK/NekMesh Channel.msh Channel.xml
  ```

- `Channel-al.xml` can be generated using the module `peralign` available with the pre-processing tool `$NEK/NekMesh`:

  ```
  $NEK/NekMesh -m peralign:surf1=2:surf2=3:dir=x Channel.xml
  Channel-al.xml
  ```

  where `surf1` and `surf2` correspond to the periodic physical surface IDs specified in *Gmsh* (in our case the inflow has a physical ID=2 while the outflow has a physical ID=3) and `dir` is the periodicity direction (i.e. the direction normal to the inflow and outflow boundaries - in our case x).

Examine the `Channel.xml` and `Channel-al.xml` files you have just created. Only the mesh and default expansions are defined at present and the only difference between the two files is the ordering of the edges in the section composite ID=3 which has been re-ordered in order to apply periodic boundary conditions correctly.

Warning

There is currently an issue when using the coupled solver and periodic edges which is being investigated. For achieving the correct channel flow stability results when using the Coupled Linearised Navier-Stokes algorithm (see section 3.2), please use the files provided in the folder `$NEKTUTORIAL/stability/Coupled`.

Chapter 2

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, this can be computed using the incompressible Navier-Stokes solver (`$NEK/IncNavierStokesSolver`).

Tip

Note that the incompressible Navier-Stokes solver (`$NEK/IncNavierStokesSolver`) executable encapsulates the nonlinear equations as well as the stability tools. Therefore, when you setup either a nonlinear incompressible problem or an incompressible stability problem you should use the same executable - i.e.:

`$NEK/IncNavierStokesSolver file.xml`.

The instructions for running one or the other are specified on the XML file.

For the problem considered here, 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 `$NEKTUTORIAL/base` you will find the file `Channel-Base.xml` which contains the geometry along with the necessary parameters to solve the problem. The `GEOMETRY` section defines the mesh of the problem and it is generated automatically as you have seen in the previous task. The expansion type and order is specified in the `EXPANSIONS` section. An expansion basis is applied to a geometry *composite*, where by *composite* we mean a collection of mesh entities (specifically here, a collection of mesh elements), specified in the `GEOMETRY` section. A default entry is always included by the `$NEK/NekMesh` 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 be used in the expansion. For example,

```
<EXPANSIONS>
    <E COMPOSITE="C[0]" NUMMODES="11" FIELDS="u,v,p" TYPE="GLL_LAGRANGE"/>
</EXPANSIONS>.
```

Note that all the results obtained in this tutorial refers to the expansion parameters just defined (i.e. `NUMMODES="11" FIELDS="u,v,p" TYPE="GLL_LAGRANGE"`).

In order to complete the problem definition and generate the base flow we need to specify a section called `CONDITIONS` in the session file. If we examine `Channel-Base.xml`, we can see how to define the conditions of the particular problem to solve.

In particular, the `CONDITIONS` section contains the following 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`[1], `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

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

   

   Task 2.1

   In the `SOLVERINFO` section of `Channel-Base.xml`:
   Note: The bits to be completed are identified by . . . in this file.

   - set `EQTYPE` to `UnsteadyNavierStokes` to select the unsteady incompressible Navier-Stokes equations,
   - set the `EvolutionOperator` to `Nonlinear` in order to select the non-linear Navier-Stokes,
   - set the `Projection` property to `Continuous` in order to select the continuous Galerkin approach,
   - set the `Driver` to `Standard` in order to perform standard time-integration.

2. The **parameters** (`PARAMETERS`) are specified as name-value pairs:

[1] in this case the term *Direct* refers to the direct stability analysis (opposed to the adjoint analysis) and it has no relation with the Coupled Linearised Navier-Stokes algorithm that will be explained in the next section

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

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

Task 2.2

Declare the two parameters `Re`, that represents the Reynolds number, and `Kinvis`, which represents the kinematic viscosity. Now set the Reynolds number to 7500 and the kinematic viscosity to $1/Re$ - i.e.

```
<P> Re = 7500 </P>
<P> Kinvis = 1/Re </P>
```

Note that you can put previously defined parameters in the `VALUE` entry which can be an expression.

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

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

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

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

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.

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

The `REF` attribute for a boundary condition region should correspond to the `ID="[INDEX]"` of the desired **boundary region** specified in the `BOUNDARYREGIONS` section.

5. The definition of the (time- and) space-dependent functions (`FUNCTION`), in terms of x, y, z and t, such as initial conditions, forcing functions, and exact solutions. The `VARIABLES` represent the components of the specific function in a specified direction and they must be the same for every function.

```
<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. For example, this will be used to specify the `InitialConditions` or `ExactConditions`.

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

Task 2.3

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

$$U = (y+1)(1-y) \tag{2.1}$$
$$V = 0 \tag{2.2}$$
$$P = 0 \tag{2.3}$$

Note: You need to use the first definition of `FUNCTION` where you can set an `EXPRESSION`.

Tip

If you are interested in the meaning of the other parameters and options present in the XML file, they should be available in the User-Guide. If not - just ask and we should be able to answer!

Task 2.4

Define a body forcing function in the streamwise direction (called `BodyForce`): $f_x = 2\nu = 2 * \texttt{Kinvis}$.

Note that for using the body force you need the following additional tag outside the section `CONDITIONS`:

```
<FORCING>
    <FORCE TYPE="Body">
        <BODYFORCE> BodyForce </BODYFORCE>
    </FORCE>
</FORCING>
```

It is possible to specify an arbitrary initial condition. In this case, it was decided to start from the exact solution of the problem in order to have a steady state in just few iterations. If the initial condition is not specified, it will be set to zero by default.

This completes the specification of the problem.

Task 2.5

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

```
$NEK/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.

In particular, the terminal screen should look like this:

```
=====================================================================
               EquationType: UnsteadyNavierStokes
               Session Name: Channel-Base
               Spatial Dim.: 2
         Max SEM Exp. Order: 11
             Expansion Dim.: 2
            Projection Type: Continuous Galerkin
                  Advection: explicit
                  Diffusion: explicit
                  Time Step: 0.001
               No. of Steps: 1000
        Checkpoints (steps): 500
           Integration Type: IMEXOrder3
=====================================================================
Initial Conditions:
  - Field u: (y+1)*(1-y)
  - Field v: 0
  - Field p: 0
Writing: "Channel-Base_0.chk"
Steps: 100      Time: 0.1          CPU Time: 1.296s
Steps: 200      Time: 0.2          CPU Time: 0.440151s
Steps: 300      Time: 0.3          CPU Time: 0.440857s
Steps: 400      Time: 0.4          CPU Time: 0.438776s
Steps: 500      Time: 0.5          CPU Time: 0.441416s
Writing: "Channel-Base_1.chk"
Steps: 600      Time: 0.6          CPU Time: 0.439318s
Steps: 700      Time: 0.7          CPU Time: 0.438448s
Steps: 800      Time: 0.8          CPU Time: 0.443955s
Steps: 900      Time: 0.9          CPU Time: 0.443197s
Steps: 1000     Time: 1            CPU Time: 0.440219s
Writing: "Channel-Base_2.chk"
```

```
Time-integration  : 5.26234s
Writing: "Channel-Base.fld"
-------------------------------------------
Total Computation Time = 6s
-------------------------------------------
L 2 error (variable u) : 1.7664e-12
L inf error (variable u) : 3.59475e-12
L 2 error (variable v) : 4.79197e-13
L inf error (variable v) : 1.12599e-11
L 2 error (variable p) : 1.68712e-11
L inf error (variable p) : 5.2737e-12
```

The final step regarding the base flow is to visualise the flow fields. Specifically, we need to convert convert the `.fld` file into a format readable by a visualisation post-processing tool. In this tutorial we decided to convert the `.fld` file into a VTK format and to use the open-source visualisation package called *Paraview* .

Task 2.6

Convert the file:

```
$NEK/FieldConvert Channel-Base.xml Channel-Base.fld
Channel-Base.vtu
```

Now open *Paraview* and 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.
Note: You can also open this type of file in VisIt.

In figure 2.1 we show how the base flow just computed should look like.

Tip

Note that *Nektar++* supports also Tecplot. To obtain a Tecplot-readable file you can run the following command:

```
$NEK/FieldConvert Channel-Base.xml Channel-Base.fld
Channel-Base.dat
```

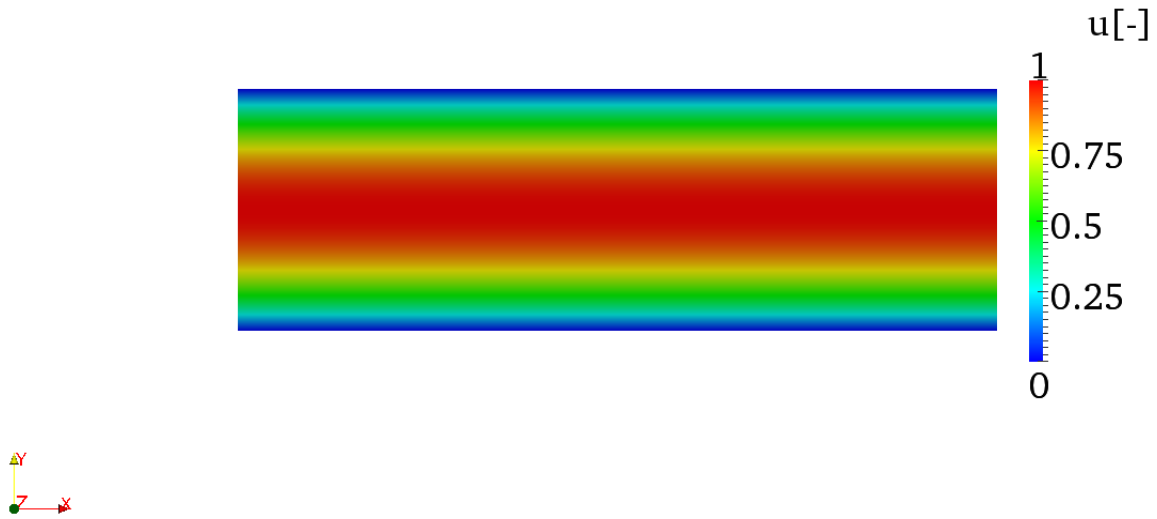

Figure 2.1 u-component of the velocity

Chapter 3

Stability Analysis

After having computed 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 Velocity Correction Scheme (`VelocityCorrectionScheme`) and
- the Coupled Linearised Navier-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 library *ARPACK* and the modified Arnoldi algorithm[1] that is also available in *Nektar++*.

3.1 Velocity Correction Scheme

First, we will compute the leading eigenvalues and eigenvectors using the velocity correction scheme method. In the `$NEKTUTORIAL/stability` folder there is a file called `Channel-VCS.xml`. This file is similar to `Channel-Base.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.

[1] International Journal for Numerical Methods in Fluids, 2008; **57**:1435-1458

Task 3.1

Configure the following additional `SOLVERINFO` options which are related to the stability analysis.

1. set the `EvolutionOperator` to `Direct` in order to activate the forward linearised Navier-Stokes system.
2. set the `Driver` to `Arpack` in order to use the *ARPACK* eigenvalue analysis.
3. Instruct ARPACK to converge onto specific eigenvalues through the solver property `ArpackProblemType`. In particular, set `ArpackProblemType` to `LargestMag` to get the eigenvalues with the largest magnitude (that determines the stability of the flow).

   Note: It is also possible to select the eigenvalue with the largest real part by setting `ArpackProblemType` to (`LargestReal`) or with the largest imaginary part by setting `ArpackProblemType` to (`LargestImag`).

Task 3.2

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.

Task 3.3

Configure the two `FUNCTION` called `InitialConditions` and `BaseFlow`.

1. A restart file is provided to accelerate communications. Set the `InitialConditions` function to be read from `Channel-VCS.rst`. The solution will then converge after 16 iterations after it has populated the Krylov subspace.

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

   Note: Since the simulations often take hundreds of iterations to converge, we will not initialise the IRAM method with a random vector during this tutorial. Normally, a random vector would be used by setting the SolverInfo option `InitialVector` to `Random`.

2. The base flow file (`Channel-Base.fld`), computed in the previous section, should be copied into the `Channel/Stability` folder and renamed `Channel-VCS.bse`. Now specify a function called `BaseFlow` which reads this file.

Task 3.4

Run the solver to perform the analysis

```
$NEK/IncNavierStokesSolver Channel-VCS.xml
```

At the end of the simulation, the terminal screen should look like this:

```
Iteration 16, output: 0, ido=99
Converged in 16 iterations
Converged Eigenvalues: 2
         Magnitude   Angle       Growth      Frequency
EV: 0 1.00112     0.124946    0.0022353   0.249892
Writing: "Channel-al_eig_0.fld"
EV: 1 1.00112     -0.124946   0.0022353   -0.249892
Writing: "Channel-al_eig_1.fld"
L 2 error (variable u) : 0.0367941
L inf error (variable u) : 0.0678149
L 2 error (variable v) : 0.0276887
L inf error (variable v) : 0.0649249
L 2 error (variable p) : 0.00512347
L inf error (variable p) : 0.00135455
```

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

$$\lambda_{1,2} = 1.00112e^{\pm 0.249892i}. \tag{3.1}$$

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 $\omega = \theta/T$.

Figure 3.1 shows the profile of the computed eigenmode. The eigenmodes associated with the computed eigenvalues are stored in the files `Channel_VCS_eig_0.fld` and `Channel_VCS_eig_1.fld`. It is possible to convert this file into VTK format in the same way as previously done for the base flow.

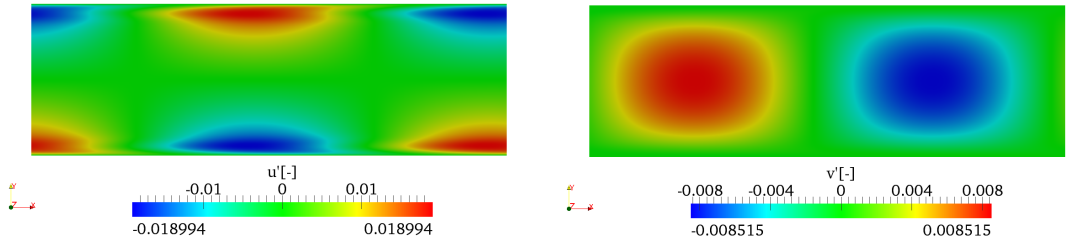

Figure 3.1 u'- and v'-component of the eigenmode.

Task 3.5

Verify that for the channel flow case :

$$\sigma = 2.2353 \times 10^{-3}$$
$$\omega = \pm 2.49892 \times 10^{-1}$$

and that the eigenmodes match those given in figures 3.1.

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.

Tip

Note that *Nektar++* implements also the modified Arnoldi algorithm. You can try to use it for this test case by setting `Driver` to `ModifiedArnoldi`. You can now try to re-run the simulation and verify that the modified Arnoldi algorithm provides a results that is consistent with the previous computation obtained with Arpack.

3.2 Coupled Linearised Navier-Stokes algorithm

Note: Remember to use the files provided in the folder `Stability/Coupled` for this case.

It is possible to perform the same stability analysis using a different method based on the Coupled Linearised Navier-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 velocity correction scheme/splitting schemes.

Inside the folder `$NEKTUTORIAL/stability` there is a file called `Channel-Coupled.xml` that contains all the necessary parameters that should be defined. In this case we will specify the base flow through an analytical expression. Even in this case, the geometry, the type and number of modes are the the same of the previous simulations.

Task 3.6

Edit the file `Channel-Coupled.xml`:
Note: As before the bits to be completed are identified by . . . in this file.

- 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 is implicitly evaluated.

Task 3.7

Continue modifying `Channel-Coupled.xml`:

- 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.

For the coupled solver, it is also necessary to define the following additional tag outside of the `CONDITIONS` tag:

```
<FORCING>
    <FORCE TYPE="StabilityCoupledLNS">
    </FORCE>
</FORCING>
```

This has already been set up in the XML file. This is necessary to tell *Nektar++* to use the previous solution as the right hand side vector for each Arnoldi iteration.

Task 3.8

Now run the solver to compute the eigenvalues

```
$NEK/IncNavierStokesSolver Channel-Coupled.xml
```

The terminal screen should look like this:

```
=======================================================================
                 Solver Type: Coupled Linearised NS
=======================================================================
        Arnoldi solver type    : Arpack
        Arpack problem type    : LM
        Single Fourier mode    : false
        Beta set to Zero       : false
        Shift (Real,Imag)      : 0,0
        Krylov-space dimension : 64
        Number of vectors      : 4
        Max iterations         : 500
        Eigenvalue tolerance   : 1e-06
=======================================================
Initial Conditions:
  - Field u: 0 (default)
  - Field v: 0 (default)
Matrix Setup Costs: 0.565916
Multilevel condensation: 0.098134
        Inital vector       : random
Iteration 0, output: 0, ido=-1
Writing: "Channel-Coupled.fld"
Iteration 20, output: 0, ido=1
Writing: "Channel-Coupled.fld"
Iteration 40, output: 0, ido=1
Writing: "Channel-Coupled.fld"
Iteration 60, output: 0, ido=1
Writing: "Channel-Coupled.fld"
Iteration 65, output: 0, ido=99

Converged in 65 iterations
Converged Eigenvalues: 4
         Real        Imaginary
EV:  0  -0.000328987            -0
Writing: "Channel-Coupled_eig_0.fld"
EV:  1   -0.00131595            -0
Writing: "Channel-Coupled_eig_1.fld"
EV:  2   -0.00296088            -0
Writing: "Channel-Coupled_eig_2.fld"
```

```
EV:  3   -0.00526379            -0
Writing: "Channel-Coupled_eig_3.fld"
L 2 error (variable u) : 2.58891
L inf error (variable u) : 1.00401
L 2 error (variable v) : 0.00276107
L inf error (variable v) : 0.0033678
```

Using the Stokes algorithm, we are computing the leading eigenvalue of the inverse of the operator $\mathcal{L}^{-1}$. Therefore the eigenvalues of $\mathcal{L}$ are the inverse of the computed values[2]. However, it is interesting to note that these values are different from those calculated with the Velocity Correction Scheme, producing an apparent inconsistency. However, this can be explained considering that the largest eigenvalues associated to the operator $\mathcal{L}$ correspond to 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 $\mathcal{L}$. One solution is to consider a large Krylov dimension specified by kdim and the number of eigenvalues to test using nvec. This will however take more iterations. Another alternative is to use shifting but in this case it will make a real problem into a complex one (we shall show an example later). Finally, another alternative is to search for the eigenvalue with a different criterion, for example, the largest imaginary part.

Task 3.9

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

```
=========================================================================
                 Solver Type: Coupled Linearised NS
=========================================================================
        Arnoldi solver type    : Arpack
        Arpack problem type    : LI
        Single Fourier mode    : false
        Beta set to Zero       : false
        Shift (Real,Imag)      : 0,0
        Krylov-space dimension : 64
        Number of vectors      : 4
        Max iterations         : 500
        Eigenvalue tolerance   : 1e-06
=========================================================
Initial Conditions:
  - Field u: 0 (default)
  - Field v: 0 (default)
Matrix Setup Costs: 0.557085
Multilevel condensation: 0.101482
        Inital vector       : random
```

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

```
Iteration 0, output: 0, ido=-1
Writing: "Channel-Coupled.fld"
Iteration 20, output: 0, ido=1
Writing: "Channel-Coupled.fld"
Iteration 40, output: 0, ido=1
Writing: "Channel-Coupled.fld"
Iteration 60, output: 0, ido=1
Writing: "Channel-Coupled.fld"
Iteration 65, output: 0, ido=99

Converged in 65 iterations
Converged Eigenvalues: 4
         Real        Imaginary
EV:  0    0.00223509      0.249891
Writing: "Channel-Coupled_eig_0.fld"
EV:  1    0.00223509     -0.249891
Writing: "Channel-Coupled_eig_1.fld"
EV:  2    -0.0542748      0.300562
Writing: "Channel-Coupled_eig_2.fld"
EV:  3    -0.0542748     -0.300562
Writing: "Channel-Coupled_eig_3.fld"
L 2 error (variable u) : 2.58891
L inf error (variable u) : 1.00401
L 2 error (variable v) : 0.00276107
L inf error (variable v) : 0.0033678
```

In this case, it is easy to 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 (apart from round-off errors). It is interesting to note that this method converges much quicker that 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.

3.3 Three-dimensional Channel flow

Now that we have presented the various stability-analysis tools present in *Nektar++*, we conclude showing the capabilities of the code in three spatial dimensions. In the folder `$NEKTUTORIAL/stability3D` there are the files that are required for the stability analysis - note that we do not show the geometry and the base flow generation (we will be using the exact solution) since we have already presented these features in the previous tutorials.

The case considered is similar to the channel flow presented in section 2. However, in this case the Reynolds number is set to 10000. In order to run a three-dimensional simulation, we can either run the full 3D solver by creating a 3D geometry or use a 2D geometry and specify the use of a Fourier expansion in the third direction. The last method is also known as 3D homogenous 1D approach. Here we will present this approach.

Specifically, we use a 2D geometry and we add the various parameters necessary to use

the Fourier expansion. Note that in the 2D plane we will use a `MODIFIED` expansion basis with `NUMMODES=11`.

Task 3.10

In the file `$NEKTUTORIAL/stability3D/PPF_R10000_3D.xml`, make the following changes:

- Add a `SOLVERINFO` tag called `HOMOGENEOUS` and set it to `1D`.
- Add two additional `SOLVERINFO` tags called `ModeType` and `BetaZero` and set them to `SingleMode` and `True`, respectively.
- Add two `PARAMETERS` called `HomModesZ` and `LZ` and set them to `2` and `1`, respectively.
- Add two other `PARAMETERS` called `realShift` and `imagShift` and set them to `0.003` and `0.2`, respectively.

Now run the solver - the terminal screen should look like this:

```
=====================================================================
                 Solver Type: Coupled Linearised NS
=====================================================================
        Arnoldi solver type    : Modified Arnoldi
        Single Fourier mode    : true
        Beta set to Zero       : true (overrides LHom)
        Shift (Real,Imag)      : 0.003,0.2
        Krylov-space dimension : 64
        Number of vectors      : 2
        Max iterations         : 500
        Eigenvalue tolerance   : 1e-06
=====================================================
Initial Conditions:
  - Field u: 0 (default)
  - Field v: 0 (default)
  - Field w: 0 (default)
Writing: "PPF_R10000_3D_0.chk"
Matrix Setup Costs: 1.97987
Multilevel condensation: 0.427631
        Inital vector       : random
Iteration: 0
Iteration: 1 (residual : 4.89954)
Iteration: 2 (residual : 3.64295)
Iteration: 3 (residual : 2.54314)
....
Iteration: 20 (residual : 1.35156e-05)
Iteration: 21 (residual : 1.64786e-06)
Iteration: 22 (residual : 1.92473e-07)
```

```
Writing: "PPF_R10000_3D.fld"
L 2 error (variable u) : 3.01846
L inf error (variable u) : 2.25716
L 2 error (variable v) : 1.8469
L inf error (variable v) : 0.985775
L 2 error (variable w) : 5.97653e-06
L inf error (variable w) : 1.2139e-05
EV:  0      0.518448      -26.6405    0.00373022      0.162477
Writing: "PPF_R10000_3D_eig_0.fld"
EV:  1      0.518448       26.6405    0.00373022      0.237523
Writing: "PPF_R10000_3D_eig_1.fld"
Warning: Level 0 assertion violation
Complex Shift applied. Need to implement Ritz re-evaluation of eigenvalue.
Only one half of complex value will be correct
```

Now convert the two files containing the eigenvectors and visualise them in *Paraview* or VisIt - the solution should look like the one below:

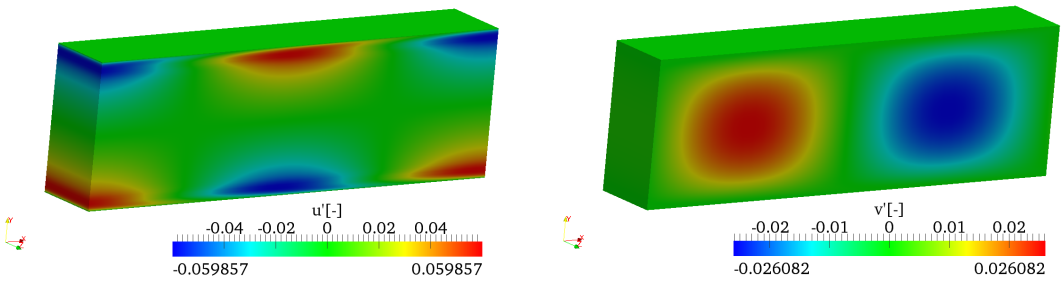

Figure 3.2 u'- and v'-component of the eigenmode.

Task 3.11

The complete input file `$NEKTUTORIAL/stability3D/PPF_R15000_3D.xml` has been provided to show a full 3D unstable eigenmode where β is not zero. Run this file and see that you obtain the eigenvalue $0.00248682 \pm -0.158348i$

Task 3.12

You can now see what the difference when not using an imaginary shifting. Set the parameters imagShift=0, kdim=384 and nvec=196.

This should take $O(500)$ iterations to complete and hidden in the list of eigenvalues should be the unstable values $0.00248662 \pm 0.158347i$. They were eigevalues 132 and 133 in my run.

3.4 Solutions

Completed solutions to the tutorials are available in the `completed` directory.

This completes the tutorial.