MEK4300 Lecture notes

Mikael Mortensen

Aug 20, 2020

WHITE - VISCOUS FLUID FLOW

These lecture notes are created using Jupyter notebooks. The notes contain live code examples that can be modified and run. The idea is for the user to get an interactive experience that hopefully will make the education both more interesting, as well as better quality. Learning by doing, programming is understanding. The notes are best read with html online. However, they can still be downloaded as a single pdf from the PDF button on the right.

PDF

On all html pages in these notes there are some buttons in the top right corner. The last button lets you download the page you are currently surfing as a jupyter notebook or a pdf. The notebook can then be opened, modified and run locally, if you are in an environment with jupyter installed. However, the different pages in these lecture notes are connected, with some links going from one page to another. So you probably want to open all these notebooks in a folder where they all live together. To this end you can for example clone the github repository (third link above) where these notes are stored. The notebooks can then be run if you have a proper environment. However, if you do not have a proper jupyter environment, then there is also first button, which looks like a rocket. The rockets first link will take you to Binder, where you will have all notebooks available in the same good environment. This means, in particular, that the environment will have FEniCS installed for you.

Please note that if you read these notes in jupyter, then some equations and figures will look weird, or unrendered, still in their source, or latex form. This is because all equations in need of reference (especially across different pages) must be written in MyST Markdown format. This gives nice equations and references in html and pdf, but the notebooks unfortunately look unrendered. It is not wrong, it is just that jupyter still does not know how to render MyST Markdown.

So what about these live computational cells? On most pages of these notes there are computational cells, where code can be typed and executed, like this:

```
a = 'Hello world!'
```

```
print(a)
```

```
Hello world!
```

Confused? Well, that's probably because the variable `a` was declared in the hidden cell above (press `Click to show`). In notebooks the cells are connected, and a variable defined in one cell will be available in consecutive cells.

If you are reading these notes in jupyter, then these cells are simply regular code cells. However, on the web (if this note you are reading is an html-file) the cells appear first to be dead, and you will be unable to edit them, which is a drag:-(Luckily, at this point you can turn again to the rocket button and press `Live Code`. This will in turn load a Binder environment that makes it possible to run the cells interactively, while still remaining in the html-version of this book, without moving to Binder. Please note that the first time a binder image is requested for a new version of this book, it may take some time to build and the *Waiting for kernel...* message seems to be hanging. However, after loading the first time, subsequent runs on this and other pages should be much faster.

Finally, note that in these lecture notes we will make extensive use of FEniCS as a tool for solving the various partial differential equations that arise. FEniCS makes use of the finite element method (FEM), and some knowledge about FEM is an advantage, yet no requirement, when reading these notes. We import all functionality of FEniCS by importing from `dolfin`, as seen below. Note that `dolfin` is the C++ module of FEniCS, which has been wrapped and made usable from Python. But note that this import must be done on each notebook. It is not enough to import dolfin here, and then use it on consecutive pages, like the *Couette chapter*. For this reason you'll also find the following code bit also on, e.g., the *Implementation* section in the Couette notes.

```
from dolfin import *
```

CHAPTER

THREE

CHAPTER 3

Chapter 3 in [Whi06] considers simple solutions of incompressible laminar shear flows. Incompressible flows are described by the physical laws for conservation of mass Newton's second law of motion). Together these are usually termed the *Navier-Stokes* equations that mathematically can be represented as

$$\begin{aligned} \frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} &= -\frac{1}{\rho}\nabla p + \nu \nabla^2 \boldsymbol{u} + \boldsymbol{f} \\ \nabla \cdot \boldsymbol{u} &= 0, \end{aligned} \tag{3.1}$$

where $\boldsymbol{u}$, p, ρ, ν and $\boldsymbol{f}$ are the velocity vector, pressure, density, kinematic viscosity and body forces respectively. The equations as written are independent of coordinate system, but they look exactly the same using Cartesian coordinates. Of equally great importance, at least in chapter 3, are the Navier-Stokes equations in cylindrical coordinates. The cylindrical coordinates, r, θ, z, are given in terms of the Cartesian coordinates x, y, z as

$$\begin{aligned} x &= r\cos\theta, \\ y &= r\sin\theta, \\ z &= z. \end{aligned}$$

The Cartesian position vector is thus

$$\boldsymbol{x} = r\cos\theta \boldsymbol{i} + r\sin\theta \boldsymbol{j} + z\boldsymbol{k}.$$

The unit vectors in cylindrical coordinates are

$$\begin{aligned} \boldsymbol{i}_r &= \frac{\frac{\partial \boldsymbol{x}}{\partial r}}{|\frac{\partial \boldsymbol{x}}{\partial r}|} = \cos\theta \boldsymbol{i} + \sin\theta \boldsymbol{j}, \\ \boldsymbol{i}_\theta &= \frac{\frac{\partial \boldsymbol{x}}{\partial \theta}}{|\frac{\partial \boldsymbol{x}}{\partial \theta}|} = -\sin\theta \boldsymbol{i} + \cos\theta \boldsymbol{j}, \\ \boldsymbol{i}_z &= \frac{\frac{\partial \boldsymbol{x}}{\partial z}}{|\frac{\partial \boldsymbol{x}}{\partial z}|} = \boldsymbol{k}. \end{aligned}$$

The velocity vectors in Cartesian and cylindrical coordinates read respectively

$$\begin{aligned} \boldsymbol{u}(x, y, z, t) &= u_x \boldsymbol{i} + u_y \boldsymbol{j} + u_z \boldsymbol{k} \\ \boldsymbol{u}(r, \theta, z, t) &= u_r \boldsymbol{i}_r + u_\theta \boldsymbol{i}_\theta + u_z \boldsymbol{i}_z \end{aligned}$$

The divergence of the velocity vector $\nabla \cdot \boldsymbol{u}$ in Cartesian and cylindrical coordinates are given respectively by

$$\begin{aligned} \nabla \cdot \boldsymbol{u} &= \frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} + \frac{\partial u_z}{\partial z} \\ \nabla \cdot \boldsymbol{u} &= \frac{1}{r}\frac{\partial r u_r}{\partial r} + \frac{1}{r}\frac{\partial u_\theta}{\partial \theta} + \frac{\partial u_z}{\partial z} \end{aligned}$$

The Laplacian in Cartesian and cylindrical coordinates are given, respectively, as

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$$
$$\nabla^2 = \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial}{\partial r}\right) + \frac{1}{r^2}\frac{\partial^2}{\partial \theta^2} + \frac{\partial^2}{\partial z^2}$$

The advection $(\boldsymbol{u}\cdot\nabla)$ is given in Cartesian and cylindrical coordinates, respectively, as

$$\boldsymbol{u}\cdot\nabla = u_i\frac{\partial}{\partial x_i}$$
$$\boldsymbol{u}\cdot\nabla = u_r\frac{\partial}{\partial r} + \frac{1}{r}u_\theta\frac{\partial}{\partial \theta} + u_z\frac{\partial}{\partial z}$$

See mek2200 notes on notation for a more thorough discussion about notation.

Suggested assignment Navier-Stokes

Start from Navier-Stokes and derive the momentum equations for Cylindrical coordinates. The results are given in App B of [Whi06].

3.1 Parallel shear flows

The Navier-Stokes equations are nonlinear because of the convective term, $(\boldsymbol{u}\cdot\nabla)\boldsymbol{u}$, and in general not analytically solvable for most problems in fluid mechanics. The equations can be solved analytically when the convective term is zero and for a few other simplified flows. In chapter 3 of White we start by considering the types of flow where convection is zero.

The convection term is never negligible for turbulent flows, where the flow rapidly changes direction in a seemingly chaotic fashion. We will get back to turbulent flows in Chapter 6. For now we assume the flow is laminar. For a laminar flow the convection term will be zero for a parallel shear flow where the geometry is infinite in at least one direction and the velocity vector is parallel to the walls of the geometry. For example, a pipe flow or a plane channel flow as shown below

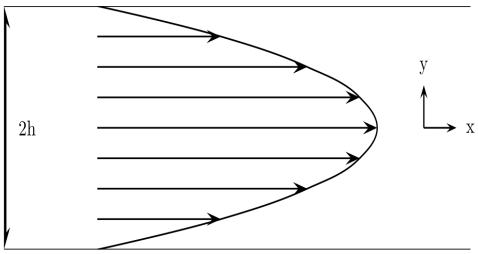


Fig. 3.1: Channel flow

where there is only one nonzero component of the velocity vector, and the velocity vector is parallel to the surrounding walls. The gradient of this velocity component is normal to the wall and as such $\boldsymbol{u}\cdot\nabla u = 0$ and the Navier-Stokes equations reduce to the linear equations

$$\frac{\partial \boldsymbol{u}}{\partial t} = -\frac{1}{\rho}\nabla p + \nu\nabla^2\boldsymbol{u} + \boldsymbol{f},$$
$$\nabla\cdot\boldsymbol{u} = 0.$$

Since the equations are linear it should not be surprising to find that there are many analytical solutions available for laminar parallel shear flows.

If the only nonzero velocity component is in the x-direction, then $\boldsymbol{u} = (u, 0, 0)$ and the Navier-Stokes equations reduce to

$$\frac{\partial u}{\partial t} = -\frac{1}{\rho}\frac{\partial p}{\partial x} + \nu\nabla^2 u + f_x,$$
$$\frac{\partial u}{\partial x} = 0.$$

which is also valid for the axial component of cylindrical coordinates (see Ch. 3-2.2). There are two types of parallel shear flows: Couette and Poiseuille. Couette flows are driven by moving walls. Friction, i.e., drag or viscous forces are then responsible for "dragging" the fluid with a direction aligned with the wall. For Poiseuille flows the driving force is a pressure gradient that can be generated using a pump.

3.1.1 Couette flows

We will first look at a steady plane Couette flow, like in Chapter 3-2.1, where the height of the plane channel is $2h$. There is no applied pressure and no gravitational forces in the x-direction, so the Navier-Stokes equations reduce further to

$$\begin{aligned} \nabla^2 u &= 0, \\ \frac{\partial u}{\partial x} &= 0, \end{aligned} \tag{3.2}$$

with boundary conditions $u(-h) = 0$ and $u(h) = U$ (see Fig. 3-1 in [Whi06]). The exact solution of these equations is

$$u(y) = \frac{U}{2}\left(1 + \frac{y}{h}\right). \tag{3.3}$$

We will now solve the Couette flow numerically using FEniCS, which is a software used for solving differential equations with the finite element method. There is an excellent tutorial for quickly getting started with FEniCS.

FEniCS solves PDEs by expressing the original problem (the PDEs with boundary and initial conditions) as a variational problem. The core of the recipe for turning a PDE into a variational problem is to multiply the PDE by a function v, integrate the resulting equation over the computational domain (typically called Ω), and perform integration by parts of terms with second-order derivatives. The function v which multiplies the PDE is in the mathematical finite element literature called a test function. The unknown function u to be approximated is referred to as a trial function. The terms test and trial function are used in FEniCS programs too.

In this course we will learn how to use the FEniCS software, but the focus will be on the physics of flow. We will use FEniCS to generate numerical solutions that we can easily play with to enhance our understanding of what the mathematical equations represent, but we will not go through the inner details of the finite element method.

In the current case of a Couette flow we will solve (3.2). We assume here for completeness that the equation has an additional constant source f

$$\nabla^2 u = f.$$

We now multiply the Poisson equation by the test function v and integrate over the domain $\Omega = [-h, h]$,

$$\int_\Omega (\nabla^2 u) v \,\mathrm{d}x = \int_\Omega f v \,\mathrm{d}x\,.$$

Note that $\mathrm{d}x$ is used to represent a volume integral. For Cartesian coordinates $\mathrm{d}x$ is equal to $\mathrm{d}x\mathrm{d}y\mathrm{d}z$, whereas for cylindrical coordinates it equals $r\mathrm{d}r\mathrm{d}\theta\mathrm{d}z$.

Next we apply integration by parts to the integrand on the left hand side with a second-order derivative,

$$\int_\Omega (\nabla^2 u) v \, \mathrm{d}x = -\int_\Omega \nabla u \cdot \nabla v \, \mathrm{d}x + \int_{\partial\Omega} \frac{\partial u}{\partial n} v \, \mathrm{d}s, \tag{3.4}$$

where $\frac{\partial u}{\partial n}$ is the derivative of u in the outward normal direction at the boundary (here at $y = \pm h$). The test function v is required to vanish on the parts of the boundary where u is known, which in the present problem implies that $v = 0$ for $y = \pm h$. The second term on the right-hand side of (3.4) therefore vanishes for the current problem and we are left with

$$-\int_\Omega \nabla u \cdot \nabla v \, \mathrm{d}x = \int_\Omega f v \, \mathrm{d}x \,,$$

which is also referred to as the weak form of the original boundary value problem ($\nabla^2 u = f$ with boundary conditions). The finite element method discretizes and solves this weak form of the problem on a domain divided into non-overlapping cells. In 2D we typically apply triangles, whereas in 3D tetrahedrons are applied. For 1D problems like the Couette flow we simply divide the computational domain Ω into non-overlapping intervals.

The final step required for building a finite element solution is to choose an appropriate function space for our solution. A function space can for example be piecewise linear functions or piecewise polynomials of a higher degree. Different function spaces can be chosen for the trial and test functions, but in general they only differ on the boundaries. The test and trial spaces $\hat{V}$ and V are in the present problem defined as

$$\hat{V} = \{v \in H^1(\Omega) : v = 0 \text{ for } y = \pm h\}, \tag{3.5}$$

$$V = \{v \in H^1(\Omega) : v = 0 \text{ for } y = -h \text{ and } v = U for y = h\}, \tag{3.6}$$

Briefly, $H^1(\Omega)$ is known as the Sobolev space containing functions v such that v^2 and $\|\nabla v\|^2$ have finite integrals over Ω. You will learn more about the Sobolev space in a basic finite element course. For now it will be sufficient to know that you need to choose a function space and for fluid flow it is usually appropriate to choose a space consisting of piecewise linear or quadratic polynomials.

Note that the proper mathematical statement of our variational problem now goes as follows: Find $u \in V$ such that

$$-\int_\Omega \nabla u \cdot \nabla v \, \mathrm{d}x = \int_\Omega f v \, \mathrm{d}x \quad \forall v \in \hat{V}. \tag{3.7}$$

Implementation

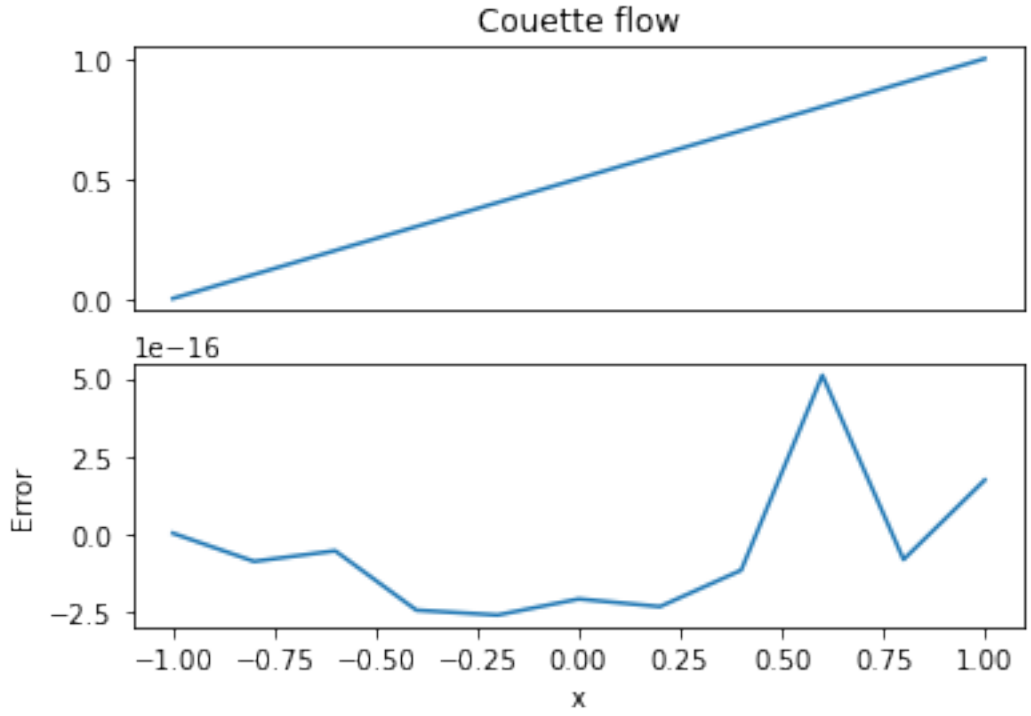
The entire implementation that solves the variational problem (3.7) is given below:

```
"""
Couette flow
"""
from dolfin import *
import matplotlib.pyplot as plt
%matplotlib inline

h = 1.
N = 10
mesh = IntervalMesh(N, -h, h)
V = FunctionSpace(mesh, 'CG', 1)
u = TrialFunction(V)
v = TestFunction(V)

def bottom(x, on_boundary):
    return near(x[0], -h) and on_boundary
```

(continues on next page)

(continued from previous page)

```
def top(x, on_boundary):
    return near(x[0], h) and on_boundary

U = 1.
bcs = [DirichletBC(V, 0, bottom),
       DirichletBC(V, U, top)]

u_ = Function(V)
solve(-inner(grad(u), grad(v))*dx == Constant(0)*v*dx, u_, bcs=bcs)

u_exact = project(Expression("U/2*(1+x[0]/h)", U=U, h=h, degree=1), V)

plt.subplot(211)
plot(u_, title="Couette flow")
plt.xticks([])
plt.subplot(212)
plot(u_ - u_exact)
plt.ylabel('Error')
plt.xlabel('x')
plt.show()
```



A more thorough tutorial for the Poisson equation is given in the online tutorial. Here we will repeat only the most necessary steps. The first thing we need to do is to import all basic FEniCS functionality into the python environment

```
from dolfin import *
```

This is typically the first line of most FEniCS scripts as it imports all necessary functionality like `Interval`, `FunctionSpace`, `Function`, `DirichletBC`, `UnitSquare` and much, much more. A computational mesh is then generated by dividing the computational domain ($\Omega = [-h, h]$) into N equally sized intervals, each of size $2h/\mathrm{N}$

```
h = 1.
N = 10
mesh = IntervalMesh(N, -h, h)
```

We now choose a piecewise linear solution by defining a function space over the mesh

```
V = FunctionSpace(mesh, "CG", 1)
u = TrialFunction(V)
v = TestFunction(V)
```

Second order polynomials or higher may be chosen by using a higher number here, but for the current problem this will not enhance the accuracy since the analytical solution is linear. The trial and test functions u and v are then declared, just as described leading up to (3.7).

The current problem assigns a value for u on both boundaries, which is mathematically termed Dirichlet boundary conditions. To implement these Dirichlet conditions we first need to specify where the boundaries are using two python functions

```
def bottom(x, on_boundary):
    return near(x[0], -h) and on_boundary

def top(x, on_boundary):
    return near(x[0], h) and on_boundary
```

Here x is an array representing position. It is an array of length equal to the number of spatial dimensions of the mesh. In our case it is an array of length 1. The boundary conditions are created using the FEniCS function `DirichletBC`.

```
U = 1.
bcs = [DirichletBC(V, 0, bottom),
       DirichletBC(V, U, top)]
```

Note that by using brackets [] the two boundary conditions are placed in a python list named bcs. The variational problem is solved by creating a `Function` to hold the solution and then calling the function `solve`.

```
u_ = Function(V)
solve(-inner(grad(u), grad(v))*dx == Constant(0)*v*dx, u_, bcs=bcs)
```

The $u\_$ `Function` has $\mathrm{N}+1$ unknowns, which are the solutions at the $\mathrm{N}+1$ nodes of the mesh. However, it is important to remember that the finite element solution is not restricted to the nodes of the mesh. The finite element solution is the piecewise continuous linear profiles, well defined over the entire elements. You can look at the discrete solution at the nodes on the command line as a numpy array using

```
w = u_.vector().get_local()
print(w)
```

```
[1.  0.9 0.8 0.7 0.6 0.5 0.4 0.3 0.2 0.1 0. ]
```

We can put the exact solution in a FEniCS `Function` by projecting a known analytical solution as an `Expression` onto the same function space as the numerical solution:

```
u_exact = project(Expression("U/2*(1+x[0]/h)", U=U, h=h), V)
```

The `Function` `u_exact` will now contain $\mathrm{N}+1$ values, just like `u_`, and if we have computed correctly these values should be more or less the same as in `u_`. The plot above of the numerical solution minus the exact solution (`plot(u_ - u_exact, title="Error couette flow")`) shows that the computed solution is exact down to machine precision $\approx 10^{-16}$. This is to be expected since the solution is linear and we are assuming a piecewise linear finite element solution.

Couette flow between axially moving concentric cylinders

The next example in Chapter 3-2.2 is the flow between axially moving concentric cylinders. The equation is the same as for the regular Couette flow in 3-2.1, but the coordinate system is cylindrical. There is no applied pressure gradient and the flow is described by the Poisson equation with no source.

$$\begin{aligned} \nabla^2 u &= 0, \\ \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial u}{\partial r}\right) &= 0. \end{aligned} \tag{3.8}$$

The two cylinders have diameters r_0 and r_1, where $r_0 < r_1$. The Dirichlet boundary conditions are $u(r = r_0) = u_0$ and $u(r = r_1) = u_1$. The solution to (3.8) can be found by integrating twice

$$u(r) = C_1 \ln(r) + C_2, \tag{3.9}$$

where the two integration constants are found using the boundary conditions leading to

$$\begin{aligned} C_1 &= \frac{U_1 - U_0}{\ln(r_1/r_0)} \\ C_2 &= U_0\frac{\ln(r_1)}{\ln(r_1/r_0)} - U_1\frac{\ln(r_0)}{\ln(r_1/r_0)} \end{aligned}$$

which inserted into (3.9) gives

$$u(r) = U_0\frac{\ln(r_1/r)}{\ln(r_1/r_0)} + U_1\frac{\ln(r/r_0)}{\ln(r_1/r_0)}. \tag{3.10}$$

Eq. (3.10) is exactly the sum of Eqs.~(3-18) and (3-19), which is a result of the governing equation being linear.

The variational formulation of the problem reads

$$-\int_\Omega \nabla u \cdot \nabla v \,\mathrm{d}x = \int_\Omega f v \,\mathrm{d}x\,,$$

but in Cylindrical coordinates the volume integral needs to be transformed ($\mathrm{d}x = r\mathrm{d}r2\pi L$, assuming the cylinder has length L) and the variational problem becomes

$$-\int_\Omega \nabla u \cdot \nabla v \, r\mathrm{d}r = \int_\Omega f v r \,\mathrm{d}r\,.$$

Note that the gradient requires no special treatment since the gradient component in the r-direction is unchanged in cylindrical coordinates.

A FEniCS implementation for axially moving concentric cylinders is shown below.

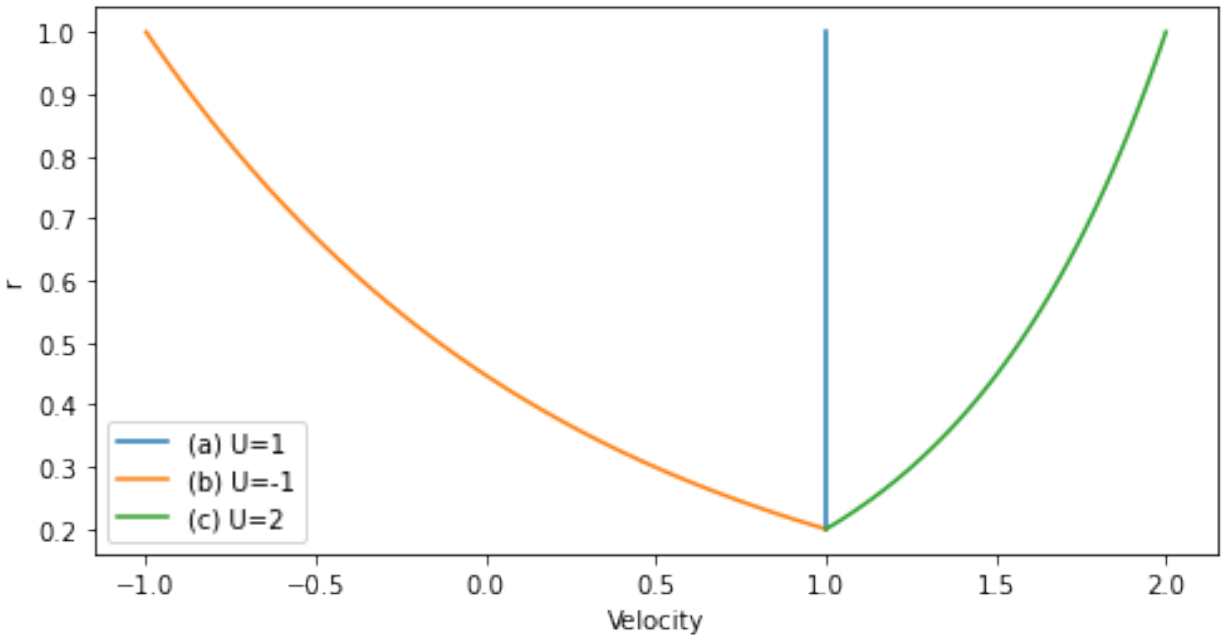
```
"""
Couette flow between two axially moving cylinders
"""
from dolfin import *

parameters['reorder_dofs_serial'] = False

r0 = 0.2
r1 = 1.0
u0 = Constant(0.1)
u1 = Constant(1)
mesh = IntervalMesh(100, r0, r1)
V = FunctionSpace(mesh, "CG", 1)
```

(continues on next page)

(continued from previous page)

```
u = TrialFunction(V)
v = TestFunction(V)

def inner_boundary(x, on_boundary):
    return near(x[0], r0) and on_boundary

def outer_boundary(x, on_boundary):
    return near(x[0], r1) and on_boundary

bc0 = DirichletBC(V, u0, inner_boundary)
bc1 = DirichletBC(V, u1, outer_boundary)

r = Expression("x[0]", degree=1)
F = inner(grad(v), grad(u)) * r * dx == Constant(0) * v * r * dx

u_ = Function(V)

# Compute all three subproblems (3-2) and plot in the same figure
plt.figure(figsize=(8,4))
for u00, u11 in zip([1, 1, 1],[1, -1, 2]):
    u0.assign(u00)
    u1.assign(u11)
    solve(F, u_, bcs=[bc0, bc1])
    plt.plot(u_.vector().get_local(), mesh.coordinates())

plt.legend(["(a) U=1", "(b) U=-1", "(c) U=2"], loc="lower left")
plt.xlabel("Velocity")
plt.ylabel("r")

u_exact = (u1-u0)*ln(r)/ln(r1/r0) + u0 - (u1-u0)*ln(r0)/ln(r1/r0)
u_exact = project(u_exact, V)
print('Error ', errornorm(u_, u_exact, degree_rise=0))
plt.show()
```

```
Error  2.7838640479591485e-05
```



Unsteady Couette flows

Transient behavior can be obtained for Couette type flows, where the velocity of the walls are either

1. Suddenly accelerated to a constant velocity $u_{wall} = U_0$
2. Oscillated as a function of time, i.e., $u_{wall} = f(t)$.

The governing equation for these types of flow is the homogeneous heat equation

$$\frac{\partial \boldsymbol{u}}{\partial t} = \nu \nabla^2 \boldsymbol{u}.$$

We consider only parallel shear flows in Cartesian coordinates, where the velocity vector is $\boldsymbol{u} = (u(y,t), 0, 0)$ and the equation is reduced to

$$\frac{\partial u}{\partial t} = \nu \frac{\partial^2 u}{\partial y^2}. \tag{3.11}$$

There are many different solutions available for this 1D heat equation, depending on the boundary and initial conditions of the problem. Most solutions make use of the Fourier series, the Fourier integral or Fourier transforms, see, e.g., [Kre99].

As a first example of flow that is suddenly accelerated to a constant velocity, consider a infinite plane normal to the y-axis located at $y = 0$ and initially at rest at $t = 0$. The plane is then suddenly accelerated to a constant velocity U_0 in the x-direction and remains at this velocity for all $t > 0$. Above the plane, for $y > 0$, the fluid is initially at rest. Friction and the no-slip boundary condition will make the flow above the plane increase in speed. Far away from the plane the velocity will be zero, i.e., $u(\infty, t) = 0$. This means that the velocity will never reach a steady state, and the boundary layer over the plane will continue to grow for all time.

We can find the solution as before using a separation of variables, but first we normalize the velocity

$$v(y,t) = \frac{u - U_0}{U_0},$$

such that the boundary condition at $y = 0$ becomes homogeneous, i.e., $v(0,t) = 0$ for $t > 0$. The PDE for v is still (3.11).

Start by separating variables

$$v(y,t) = T(t)V(y).$$

Inserted into (3.11) we get

$$\frac{\dot{T}}{\nu T} = \frac{V''}{V} = K,$$

where K is a constant. To obtain physically realistic solutions, the constant needs to be negative. This can be understood looking first at the transient part:

$$\dot{T} = K\nu T,$$

with solution

$$T(t) = e^{K\nu t}.$$

If K is positive, then the velocity will grow without bounds and without energy being added to the system. For the second equation we have

$$V'' - KV = 0.$$

If K is positive, then the solution is

$$V(y) = A\sinh(\sqrt{K}y) + B\cosh(\sqrt{K}y).$$

The boundary condition $V(0) = 0$ requires $B = 0$. The remaining term $A\sinh(\sqrt{K}y)$ grows without bounds as $y \longrightarrow \infty$, which is not in agreement with $v(\infty, t) = -1$ and we have to assume that K is a negative constant that we can write as $K = -\lambda^2$. The solutions to the separated ordinary differential equations become

$$\begin{aligned} T(t) &= e^{-\lambda^2\nu t}, \\ V(y) &= A\sin(\lambda y) + B\cos(\lambda y). \end{aligned}$$

The constants need to be found using boundary and initial conditions. The boundary condition $V(0) = 0$ leads to $B = 0$ and as such one solution reads

$$v(y,t) = A\sin(\lambda y)e^{-\lambda^2\nu t}.$$

To satisfy the initial condition we need to use more than one solution and through the superposition principle we can simply add solutions. A periodic Fourier series for v reads

$$v(y,t) = \sum_{k=1}^{\infty} A_k \sin(\lambda_k y)e^{-\lambda_k^2\nu t},$$

where A_k are constants, $\lambda_k = k\pi/L$ and L is the length of the periodic domain.

However, our domain is not periodic and it is unbounded for $y \longrightarrow \infty$. It turns out that for infinite domains and non-periodic solution, it is advantageous to work with Fourier integrals instead. A Fourier integral for our problem reads

$$v(y,t) = \int_0^{\infty} A(\lambda)\sin(\lambda y)e^{-\lambda^2\nu t}d\lambda.$$

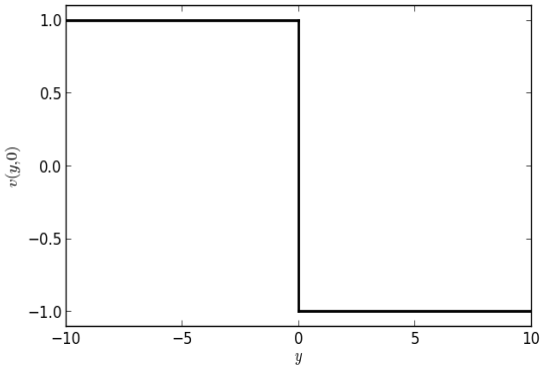


Fig. 3.2: The Heaviside function.

and the "constant" $A(\lambda)$ is a continuous function of λ. Since v only contains sinuses it is an odd function. As such, even though we are only interested in the domain $y > 0$ we can define an initial condition for $v(y, 0)$ over the entire $-\infty < y < \infty$ as an odd function

$$v(y,0) = 1 - 2\boldsymbol{H}(y) = \int_0^\infty A(\lambda)\sin(\lambda y)d\lambda, \text{ for } -\infty < y < \infty,$$

where $\boldsymbol{H(y)}$ is the Heaviside function.

This way $v(y, 0) = 1$ for $y < 0$, $v(y, 0) = -1$ for $y > 0$ and by symmetry $v(0, 0) = 0$. Through orthogonality and the initial condition we can find $A(\lambda)$ as (see [Kre99])

$$A(\lambda) = -\frac{2}{\pi}\int_0^\infty \sin(\lambda y')dy'.$$

Inserted into the total solution we get

$$v(y,t) = -\frac{2}{\pi}\int_0^\infty \int_0^\infty \sin(\lambda y')dy' \sin(\lambda y)e^{-\lambda^2\nu t}d\lambda,$$

which can be rewritten by changing the order of integration as

$$v(y,t) = -\frac{2}{\pi}\int_0^\infty \int_0^\infty \sin(\lambda y') \sin(\lambda y)e^{-\lambda^2\nu t}d\lambda dy'.$$

The inner integral can be transformed using $\sin(\lambda y)\sin(\lambda y') = 0.5(\cos\lambda(y - y') - \cos\lambda(y + y'))$

$$v(y,t) = -\frac{2}{\pi}\int_0^\infty \int_0^\infty e^{-\lambda^2\nu t}\frac{1}{2}\left(\cos\lambda(y - y') - \cos\lambda(y + y')\right)d\lambda dy'.$$

The inner integrals can then be computed exactly since

$$\int_0^\infty e^{-\lambda^2\nu t}\cos\lambda(y - y')\mathrm{d}\lambda = \sqrt{\frac{\pi}{4\nu t}}e^{-\left(\frac{y-y'}{2\sqrt{\nu t}}\right)^2},$$

and

$$\int_0^\infty e^{-\lambda^2\nu t}\cos\lambda(y + y')\mathrm{d}\lambda = \sqrt{\frac{\pi}{4\nu t}}e^{-\left(\frac{y+y'}{2\sqrt{\nu t}}\right)^2}.$$

The solution now reads

$$v(y,t) = -\sqrt{\frac{1}{4\pi\nu t}}\int_0^\infty \left(e^{-\left(\frac{y-y'}{2\sqrt{\nu t}}\right)^2} - e^{-\left(\frac{y+y'}{2\sqrt{\nu t}}\right)^2}\right)dy'.$$

Using integration by substitution, the integrals can be reorganized into error functions

$$\mathrm{erf}(z) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-u^2} \mathrm{d}u,$$

if we simply use a change of variables

$$u = \frac{y - y'}{2\sqrt{\nu t}}, \quad \frac{\mathrm{d}u}{\mathrm{d}y'} = -\frac{1}{2\sqrt{\nu t}}$$
$$u' = \frac{y + y'}{2\sqrt{\nu t}}, \quad \frac{\mathrm{d}u}{\mathrm{d}y'} = \frac{1}{2\sqrt{\nu t}}$$

The integrals become

$$\begin{aligned}\int_0^\infty e^{-\left(\frac{y-y'}{2\sqrt{\nu t}}\right)^2} dy' &= -2\sqrt{\nu t} \int_{\frac{y}{2\sqrt{\nu t}}}^{-\infty} e^{-u^2} \mathrm{d}u, \\ &= -\sqrt{\nu t \pi} \left(1 + \mathrm{erf}\left(\frac{y}{2\sqrt{\nu t}}\right)\right)\end{aligned}$$

and

$$\begin{aligned}\int_0^\infty e^{-\left(\frac{y+y'}{2\sqrt{\nu t}}\right)^2} dy' &= 2\sqrt{\nu t} \int_{\frac{y}{2\sqrt{\nu t}}}^{\infty} e^{-u^2} \mathrm{d}u, \\ &= \sqrt{\nu t \pi} \left(1 - \mathrm{erf}\left(\frac{y}{2\sqrt{\nu t}}\right)\right).\end{aligned}$$

Putting it all together we finally obtain

$$\begin{aligned}v(y,t) &= -\sqrt{\frac{1}{4\pi\nu t}} \left(-\sqrt{\nu t \pi}\right) (-2) \mathrm{erf}\left(\frac{y}{2\sqrt{\nu t}}\right), \\ &= -\mathrm{erf}\left(\frac{y}{2\sqrt{\nu t}}\right).\end{aligned}$$

The original unnormalized velocity becomes

$$u(y,t) = U_0 \left(1 - \mathrm{erf}\left(\frac{y}{2\sqrt{\nu t}}\right)\right), \tag{3.12}$$

in agreement with Eq. 3-107 in [Whi06]. Profiles for the velocity above the plane is computed below, where the first plot includes the unphysical (mirror) domain.

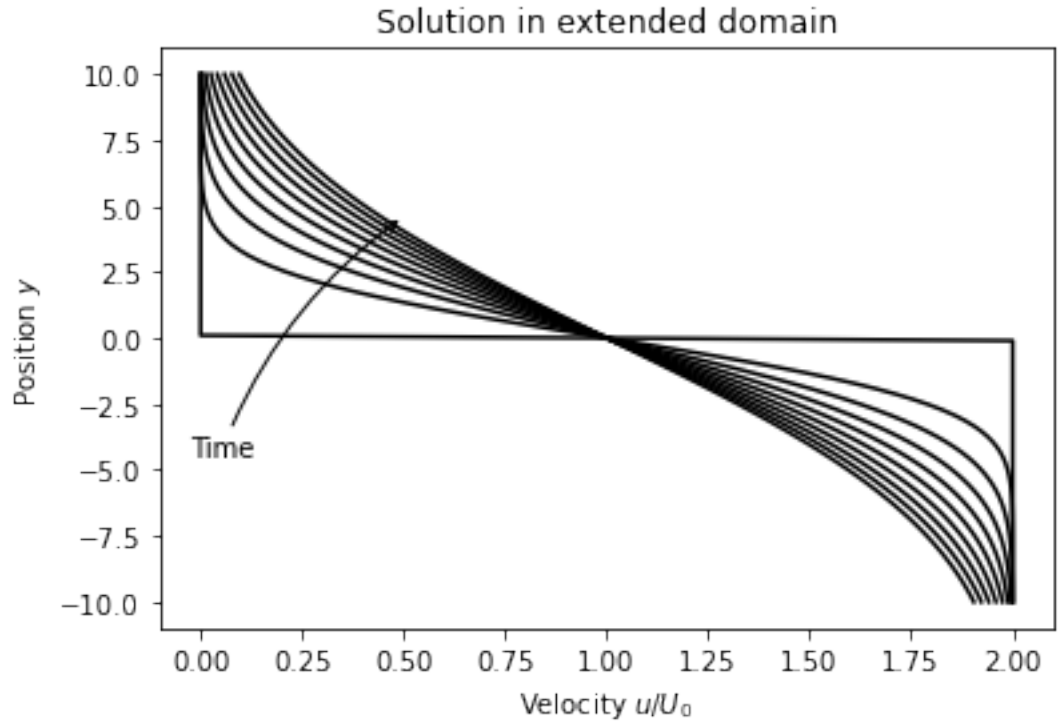
```
import numpy as np
from scipy.special import erf

nu = 0.01
U0 = 1.
for y0, title in zip((-10, 0), ('Solution in extended domain', 'Solution in true␣
↪domain')):
    y = np.linspace(y0, 10, 100)
    plt.figure(figsize=(6, 4))
    for i in range(0, 10):
        t = i * 200.
        u = U0*(1-erf(y/2/sqrt(nu*t)))
        plt.plot(u, y, "k")

    plt.annotate(r'Time', xy=(0.5, 4.6),
```

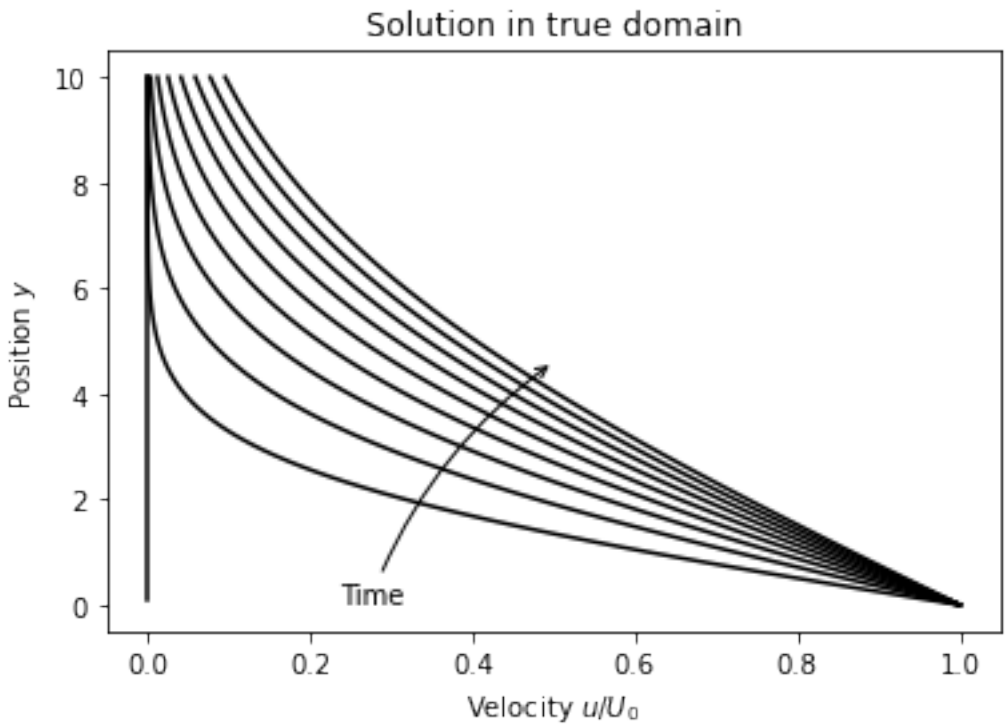
(continues on next page)

(continued from previous page)

```
                xytext=(-80, -90), textcoords='offset points',
                arrowprops=dict(arrowstyle="->", connectionstyle="arc3,rad=-.15"))
    plt.xlabel("Velocity " + r"$u/U_0$")
    plt.ylabel("Position " + "$y$")
    plt.title(title)
plt.show()
```

```
<ipython-input-4-d4d1f923fc3d>:11: RuntimeWarning: divide by zero encountered␣
↪in true_divide
  u = U0*(1-erf(y/2/sqrt(nu*t)))
<ipython-input-4-d4d1f923fc3d>:11: RuntimeWarning: divide by zero encountered␣
↪in true_divide
  u = U0*(1-erf(y/2/sqrt(nu*t)))
<ipython-input-4-d4d1f923fc3d>:11: RuntimeWarning: invalid value encountered␣
↪in true_divide
  u = U0*(1-erf(y/2/sqrt(nu*t)))
```





Fluid oscillation above an infinite plate

An infinite plate oscillates with a velocity $u(0,t) = U_0 \cos(\omega t)$, which is the boundary condition to the fluid flow set in motion above the plate. We look for a "steady state" solution $\boldsymbol{u} = (u(y), 0, 0)$, meaning that we look for a solution independent of initial conditions. This problem is also referred to as Stokes second problem.

Since the plate oscillates with frequency $\cos(\omega t)$ a justified guessed solution is

$$u(y,t) = f(y)e^{i\omega t}.$$

Inserted into the unsteady 1D heat equation we obtain

$$\begin{aligned} f i\omega \cancel{e^{i\omega t}} - \nu f'' \cancel{e^{i\omega t}} &= 0, \\ f'' - \frac{i\omega}{\nu} f &= 0 \end{aligned}$$

A general solution to this problem is

$$f(y) = Ae^{-\sqrt{\frac{i\omega}{\nu}}y} + Be^{\sqrt{\frac{i\omega}{\nu}}y},$$

where only the first leads to physically realistic real solutions for $y \longrightarrow \infty$, such that $B = 0$. The boundary condition at $y = 0$ leads to

$$u(0,t) = f(0)e^{i\omega t} = Ae^{i\omega t},$$

where the real part then requires $A = U_0$ and as such

$$u(y,t) = U_0 e^{-\sqrt{\frac{i\omega}{\nu}}y + i\omega t}$$

Unsteady flow between two infinite plates

Consider two infinite plates separated by a distance h containing fluid initially at rest at $t = 0$. At $t = 0$ the velocity of the lower plate at $y = 0$ is suddenly accelerated to a velocity of U_0 and the fluid above the plate starts to move due to friction and the no-slip boundary condition. The flow will approximate the steady linear Couette solution $U_0(1 - y/h)$ as $t \longrightarrow \infty$. Find the velocity between the plates as a function of time and space.

The velocity, $\boldsymbol{u} = (u(y), 0, 0)$, is parallel to the walls and described by the 1D heat equation

$$\begin{aligned} \frac{\partial u}{\partial t} - \nu \frac{\partial^2 u}{\partial y^2} &= 0, \quad \text{for} \quad 0 \leq y \leq h \\ u(y, 0) &= 0, \quad 0 < y < h \\ u(0, t) &= U_0, \\ u(h, t) &= 0 \end{aligned}$$

Through a shift of variables and using the steady Couette solution $v(y, t) = u(y, t) - U_0(1 - y/h)$, the problem can be redefined with homogeneous boundary conditions

$$\begin{aligned} \frac{\partial v}{\partial t} - \nu \frac{\partial^2 v}{\partial y^2} &= 0, \quad \text{for} \quad 0 \leq y \leq h \\ v(y, 0) &= -U_0(1 - y/h), \quad 0 < y < h \\ v(0, t) &= 0, \\ v(h, t) &= 0 \end{aligned}$$

We can solve this problem easily using separation of variables $v(y, t) = T(t)V(y)$. Inserted into the heat equation we obtain as before

$$\frac{\dot{T}}{\nu T} = \frac{V''}{V} = -\lambda^2$$

and two ordinary differential equations for T and V

$$\begin{aligned} T(t) &= e^{-\lambda^2 \nu t}, \\ V(y) &= A \sin(\lambda y) + B \cos(\lambda y). \end{aligned}$$

The boundary conditions give us $B = 0$ and $\lambda = \lambda_n = n\pi/h$ for $n = 1, 2, \ldots$. Using superposition we get the total solution for $v(y, t)$:

$$v(y, t) = \sum_{n=1}^{\infty} A_n \sin(\lambda_n y) e^{-\lambda_n^2 \nu t},$$

where the coefficients are found using the initial condition and orthogonality. That is, multiply $v(y, 0)$ by $\sin(\lambda_m)$, integrate from 0 to h and isolate A_n:

$$\begin{aligned} A_n &= -\frac{2U_0}{h} \int_0^h \sin(\lambda_n y)(1 - y/h) \mathrm{d}y \\ A_n &= -\frac{2U_0}{n\pi} \end{aligned}$$

The total solution is then

$$u(y, t) = U_0(1 - y/h) - \frac{2U_0}{\pi} \sum_{n=1}^{\infty} \frac{\sin(\lambda_n y)}{n} e^{-\lambda_n^2 \nu t}.$$

Suggested assignments Couette

Couette problems from White:

Problems 3-1, 3-2, 3-3, 3-4 and 3-5.

Additional Couette problems:

Homogeneous heat equation with inhomogeneous boundary conditions

Consider the homogeneous heat equation with inhomogeneous boundary conditions

$$\begin{aligned} \frac{\partial u}{\partial t} - \nu \frac{\partial^2 u}{\partial y^2} &= 0, \quad \text{for} \quad 0 \leq y \leq L \\ u(y,0) &= \phi(y), \\ u(0,t) &= g, \\ u(L,t) &= h \end{aligned} \tag{3.13}$$

where g and h are constants. Find the general solution $u(y,t)$. Hint: use a function $\boldsymbol{U}(y) = 1/L[(L-y)g + yh]$ and find the solution of $v(y,t) = u(y,t) - \boldsymbol{U}(y)$. The equation for $v(y,t)$ is the same as for $u(y,t)$, but now with homogeneous boundary conditions.

Optional (difficult), find the general solution if g and h are functions of time. Two hints to the optional assignment are

Hint 1

Use a function $\boldsymbol{U}(y,t) = 1/L[(L-y)g(t) + yh(t)]$ and find the solution of $v(y,t) = u(y,t) - \boldsymbol{U}(y,t)$. The equation for $v(y,t)$ should be obtained as an inhomogeneous heat equation with homogeneous boundary conditions.

Hint 2

Use Duhamel's principle to solve the inhomogeneous heat equation. Duhamel's principle states that the solution to the inhomogeneous heat equation can be found through solving just the homogeneous part. An inhomogeneous heat equation for $v(y,t)$ reads

$$\begin{aligned} \frac{\partial v}{\partial t} - \nu \frac{\partial^2 v}{\partial y^2} &= f(y,t), \quad \text{for} \quad 0 \leq y \leq L \\ v(y,0) &= \overline{\phi}(y), \\ v(0,t) &= v(L,t) = 0, \end{aligned}$$

We can solve instead the homogeneous part

$$\begin{aligned} \frac{\partial v^h}{\partial t} - \nu \frac{\partial^2 v^h}{\partial y^2} &= 0, \quad \text{for} \quad 0 \leq y \leq L \\ v^h(y,0) &= \overline{\phi}(y), \\ v^h(0,t) &= v^h(L,t) = 0, \end{aligned}$$

using for example a separation of variables $v^h(y,t) = S(t)V(y)$. The solution will evolve from the initial condition $\overline{\phi}$ and we thus expect the solution in a small increment of time to be expressed as $v^h(y,t) = S(t)\overline{\phi}(y)$. Duhamel's

principle then states that the solution to the inhomogeneous problem can be computed as

$$v(y,t) = v^h(y,t) + \int_0^t S(t-s)f(y,s)\mathrm{d}s,$$

where $S(t-s)f(y,s)$ is the homogeneous solution we found earlier, but computed from the initial condition $f(y,s)$ instead of $\overline{\phi}$. One may think of $\int_0^t S(t-s)f(y,s)\mathrm{d}s$ as a superposition in time. Initialize using $f(y,s)$ and then move the solution an infinitesimal time forward using the homogeneous solution. By linearity, or superposition in time, one may add up, or integrate, the solution in time.

Duhamel's principle can be justified by computation. Insert for the inhomogeneous solution in the inhomogeneous equation

$$\frac{\partial v}{\partial t} - \nu\frac{\partial^2 v}{\partial y^2} = \frac{\partial v^h}{\partial t} \cancel{- \nu\frac{\partial^2 v^h}{\partial y^2}} + \left(\frac{\partial}{\partial t} - \nu\frac{\partial^2}{\partial y^2}\right)\left(\int_0^t S(t-s)f(y,s)\mathrm{d}s\right)$$

Use Leibniz rule for differentiation under the integral sign on the last part

$$\begin{aligned}\frac{\partial v}{\partial t} - \nu\frac{\partial^2 v}{\partial y^2} &= S(t-t)f(y,t) + \int_0^t \left(\frac{\partial}{\partial t} - \nu\frac{\partial^2}{\partial y^2}\right) S(t-s)f(y,s)\mathrm{d}s, \\ &= S(0)f(y,t) = f(y,t).\end{aligned}$$

The integral is zero since the integrand $S(t-s)f(y,s)$ is known to be a solution of the homogeneous heat equation.

Computer assignments Couette

Rotating cylinders

Validate the analytical results (3-22) and (3-23) in Chapter 3-2.3 of [Whi06], for flow between rotating concentric cylinders. Can you reproduce Eq.~(3-23)?

Implement Eq. (3.13)

Consider (3.13) with suitable g and h and compare the analytical solution from the written assignment with a numerical implementation.

3.1.2 Poiseuille flows

Poiseuille flows are driven by pumps that forces the fluid to flow by modifying the pressure. Fluids flow naturally from regions of high pressure to regions of low pressure. Typical examples are cylindrical pipe flow and other duct flows. A fully developed plane channel flow is shown above. Fully developed Poiseuille flows exists only far from the entrances and exits of the ducts, where the flow is aligned parallel to the duct walls. If we assume the flow is in the x-direction, as shown in Fig. 3-6 of [Whi06], then the velocity vector is $\boldsymbol{u}(y,z,t) = (u(y,z,t),0,0)$ and Navier-Stokes equations can be simplified to

$$\begin{aligned}\frac{\partial u}{\partial t} &= \nu\nabla^2 u - \frac{1}{\rho}\frac{\mathrm{d}p}{\mathrm{d}x}, \\ \frac{\partial u}{\partial x} &= 0,\end{aligned} \tag{3.14}$$

The equations are still linear because the flow is aligned in one direction and convection is thus zero. The only difference from Couette flow is that there is a non-zero source term in the pressure gradient. The transient term on the left hand side is zero for stationary flows.

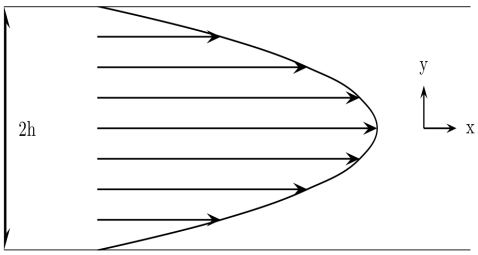


Fig. 3.3: Channel flow is a pressure driven Poiseuille flow

The total volume flow, Q, through any duct is found by integrating the velocity over the entire cross section, C, of the duct

$$Q = \int_{\mathrm{C}} u \mathrm{d}A. \tag{3.15}$$

Using (3.15) it is also possible to define an average velocity, $\overline{u}$, through the duct

$$\overline{u} = \frac{Q}{A}.$$

The average velocity is subsequently used to define the Reynolds number and certain friction factors - engineering tools used to classify duct flow.

The circular pipe

For a steady cylindrical pipe flow with radius r_0 the solution to (3.14) is found simply by integrating twice and applying boundary conditions:

$$\begin{aligned}
\frac{\nu}{r}\frac{\mathrm{d}}{\mathrm{d}r}\left(r\frac{\mathrm{d}u}{\mathrm{d}r}\right) &= \frac{1}{\rho}\frac{\mathrm{d}p}{\mathrm{d}x}, \\
\frac{\mathrm{d}}{\mathrm{d}r}\left(r\frac{\mathrm{d}u}{\mathrm{d}r}\right) &= \frac{r}{\mu}\frac{\mathrm{d}p}{\mathrm{d}x}, \\
r\frac{\mathrm{d}u}{\mathrm{d}r} &= \frac{r^2}{2\mu}\frac{\mathrm{d}p}{\mathrm{d}x} + C_1, \\
\frac{\mathrm{d}u}{\mathrm{d}r} &= \frac{r}{2\mu}\frac{\mathrm{d}p}{\mathrm{d}x} + \frac{C_1}{r}, \\
u(r) &= \frac{r^2}{4\mu}\frac{\mathrm{d}p}{\mathrm{d}x} + C_1\ln(r) + C_2,
\end{aligned}$$

where C_1 and C_2 are integration constants. The constant C_1 must be zero for the flow at the center of the pipe to remain finite. The condition at the wall $u(r_0) = 0$ gives

$$C_2 = -\frac{r_0^2}{4\mu}\frac{\partial p}{\partial x},$$

which leads to the final expression for the pipe flow

$$u(r) = -\frac{1}{4\mu}\frac{\partial p}{\partial x}\left(r_0^2 - r^2\right) \tag{3.16}$$

Combined Couette and Poiseuille flow

It is possible to combine an applied pressure gradient with moving walls. Such flows are termed combined Couette-Poiseuille flows and they are governed by the Poiseuille (3.14). The solution to a combined Couette-Poiseuille flow can be found as the sum of a Couette solution u_c using zero pressure gradient and a homogeneous Poiseuille solution u_p using zero velocity on walls. For a plane channel with $u(h) = U$ and $u(-h) = 0$ the solution is thus found by solving the two problems, for Couette

$$\nabla^2 u_c = 0, \quad u_c(h) = U,\, u_c(-h) = 0, \tag{3.17}$$

and Poiseuille

$$\nabla^2 u_p = \frac{1}{\mu}\frac{\mathrm{d}p}{\mathrm{d}x}, \quad u_p(\pm h) = 0. \tag{3.18}$$

The combined solution $u = u_c + u_p$ is the solution to the original problem. This can be seen by summing (3.17) and (3.18) and also evidently $u(h) = u_c(h) + u_p(h) = U$ and similarily for $y = -h$. The superposition is possible simply because the governing equation is linear. The principle is used also for transient flows in both Chapters (3-4) and (3-5).

Noncircular ducts

For steady and fully developed duct flows the governing equation is simply the Poisson equation

$$\nabla^2 u = \frac{1}{\mu}\frac{\mathrm{d}p}{\mathrm{d}x}. \tag{3.19}$$

The equation is linear and very easy to solve numerically for any shape of the duct. There is also a great number of analytical solutions available for noncircular duct flows, solutions that can be used to verify the quality of the numerical software. The duct solutions are often used for specifying inlet profiles to much more complicating geometries, where the flow enters the geometry through a duct.

In the code blelow we show a FEniCS code used to solve (3.19) for a rectangular duct spanned as $-a \leqslant y \leqslant a$ and $-b \leqslant z \leqslant b$. Of interest here is the computation of the very complicated analytical expression for the velocity given as

$$u(z,y) = \frac{16a^2}{\mu\pi^3}\left(-\frac{\mathrm{d}p}{\mathrm{d}x}\right)\sum_{i=1,3,5,\ldots}^{\infty}(-1)^{\frac{i-1}{2}}\left[1-\frac{\cosh(i\pi z)/2a}{\cosh(i\pi b)/2a}\right]\times\frac{\cos(i\pi y)/2a}{i^3}. \tag{3.20}$$

The FEniCS implementation of this exact solution overloads the `UserExpression` class and you should recognize (3.20) in the overloaded `eval` method. Here x is an array of the coordinates and $x[0]$ and $x[1]$ represent y and z respectively. Unfortunately, the Python `UserExpression` `u_exact` is quite slow to interpolate due to the non-vectorized for-loop. For this reason, a C++ version of the same `Expression` is implemented instead, which speeds up the program with approximately a factor of 45.

```
from dolfin import *
from numpy import cosh, cos
set_log_active(False)

dpdx = Constant(-0.01)
mu = Constant(0.01)
a = 2
b = 1

factor = 16.*a*a/mu(0)/pi**3*(-dpdx(0))
class u_exact(UserExpression):
```

(continues on next page)

(continued from previous page)

```
    def eval(self, value, x):
        u = 0
        for i in range(1, 500, 2):
            u += ((-1)**((i-1)/2)*(1-cosh(i*pi*x[1]/2./a)/
                  cosh(i*pi*b/2/a))*cos(i*pi*x[0]/2./a)/i**3)
        value[0] = u * factor

# Much faster C++ version
ue_code = '''
#include <pybind11/pybind11.h>
#include <pybind11/eigen.h>
namespace py = pybind11;

#include <dolfin/function/Expression.h>

class U : public dolfin::Expression
{
  public:

    double a, b, mu, dpdx;

  void eval(Eigen::Ref<Eigen::VectorXd> values,
            Eigen::Ref<const Eigen::VectorXd> x,
            const ufc::cell& cell) const override
    {
      double u = 0.;
      double factor = 16.*a*a/mu/pow(DOLFIN_PI, 3)*(-dpdx);
      for (std::size_t i=1; i<500; i=i+2)
        u += pow(-1, (i-1)/2 % 2)*(1.-cosh(i*DOLFIN_PI*x[1]/2./a)/
           cosh(i*DOLFIN_PI*b/2./a))*cos(i*DOLFIN_PI*x[0]/2./a)/pow(i, 3);
      values[0] = u*factor;
    }
};

PYBIND11_MODULE(SIGNATURE, m)
{
  py::class_<U, std::shared_ptr<U>, dolfin::Expression>
    (m, "U")
    .def(py::init<>())
    .def_readwrite("a", &U::a)
    .def_readwrite("b", &U::b)
    .def_readwrite("mu", &U::mu)
    .def_readwrite("dpdx", &U::dpdx);
}
'''
u_c = CompiledExpression(compile_cpp_code(ue_code).U(), a=float(a), b=float(b),
                         mu=float(mu(0)), dpdx=float(dpdx(0)), degree=1)

def main(N, degree=1):
    mesh = RectangleMesh(Point(-a, -b), Point(a, b), N, N)
    V = FunctionSpace(mesh, 'CG', degree)
    u = TrialFunction(V)
    v = TestFunction(V)
    F = inner(grad(u), grad(v))*dx + 1/mu*dpdx*v*dx
    bc = DirichletBC(V, Constant(0), DomainBoundary())
    u_ = Function(V)
    solve(lhs(F) == rhs(F), u_, bcs=bc)
```

(continues on next page)

(continued from previous page)

```
    #u_e = interpolate(u_exact(), V)  # slow, but otherwise the same as below
    u_e = interpolate(u_c, V)
    bc.apply(u_e.vector())
    u_error = errornorm(u_e, u_, degree_rise=0)
    return u_error, mesh.hmin()

E = []; h = []; degree = 1
for n in [5, 10, 20, 40, 60]:
    ei, hi = main(n, degree=degree)
    E.append(ei)
    h.append(hi)

from math import log as ln
for i in range(1, len(E)):
    r = ln(E[i]/E[i-1])/ln(h[i]/h[i-1])
    print("h=%2.2E E=%2.2E r=%.2f" %(h[i], E[i], r))
```

```
h=4.47E-01 E=6.41E-03 r=1.76
h=2.24E-01 E=1.69E-03 r=1.93
h=1.12E-01 E=4.28E-04 r=1.98
h=7.45E-02 E=1.91E-04 r=1.99
```

Analysing the error

In the code for the rectangular duct we compute the errornorm that compares the computed finite element solution with the exact solution. This error will depend on the mesh size and the order of the polynomial approximation. The error should disappear in the limit of a highly resolved mesh or for very high order polynomials. Given the polynomial order, the rate at which the error disappears can be computed by performing experiments with meshes of different resolution. We assume that the error for a mesh of element size h_i can be written as $E_i = Ch_i^r$, where C is a constant. We now want to find the order r of which the error disappears with mesh refinement. The order r can be found by computing E_i for two different h_i's, i.e., $E_i = h_i^r$ and $E_{i-1} = h_{i-1}^r$, and then isolate r:

$$r = \frac{\ln(E_i/E_{i-1})}{\ln(h_i/h_{i-1})}.$$

It is this error that is printed out at the end above. Changing the degree in the `FunctionSpace` to 2 the order should be three. Lets try it:

```
E = []; h = []; degree = 2
for n in [5, 10, 20, 40, 60]:
    ei, hi = main(n, degree=degree)
    E.append(ei)
    h.append(hi)

from math import log as ln
for i in range(1, len(E)):
    r = ln(E[i]/E[i-1])/ln(h[i]/h[i-1])
    print("h=%2.2E E=%2.2E r=%.2f" %(h[i], E[i], r))
```

```
h=4.47E-01 E=2.14E-04 r=3.13
h=2.24E-01 E=2.63E-05 r=3.03
h=1.12E-01 E=3.25E-06 r=3.02
h=7.45E-02 E=9.41E-07 r=3.05
```

Unsteady duct flows

Unsteady plane shear flows are experienced by fluids where the driving forces are suddenly or continuously changed. A fluid initially at rest will for example respond to a sudden increase in the pressure gradient (we suddenly start the pump) and the volume flow through the duct will then increase monotonically until steady state is reached. The applied pressure gradient may also vary in time. Another scenario is that a wall is suddenly or continuously set in motion, dragging with it the fluid initially at rest. Common for these flows is that the transient term in the Navier Stokes equations cannot be neglected.

Starting flow in a cylindrical pipe

When a fluid initially at rest suddenly is exposed to a constant pressure gradient, the fluid will start to move as a response to the applied forces and the flow will increase for some time until the steady state Poiseuille flow profile (3.16) is reached. For convenience we modify this equation slightly here and write it for the steady velocity u_s as

$$u_s = u_m \left(1 - \tilde{r}^2\right),$$

where

$$u_m = -\frac{r_0^2}{4\mu}\frac{\partial p}{\partial x}$$

and $\tilde{r} = r/r_0$. We use the superposition principle and split the equation into two contributions

$$u(r,t) = v(r,t) + u_s(r),$$

where $u(r,t)$ is the complete solution and $v(r,t)$ is the solution to the homogeneous (3.14) in cylinder coordinates:

$$\frac{\partial v}{\partial t} = \nu\left(\frac{\partial^2 v}{\partial r^2} + \frac{1}{r}\frac{\partial v}{\partial r}\right), \tag{3.21}$$

subject to boundary condition $v(r_0,t) = 0$ and initial condition $v(r,0) = -u_s(r)$. The equation can be non-dimensionalized by multiplying with r_0^2/ν and using $\tilde{t} = t\nu/r_0^2$ and $\tilde{r} = r/r_0$

$$\frac{\partial v}{\partial \tilde{t}} = \frac{\partial^2 v}{\partial \tilde{r}^2} + \frac{1}{\tilde{r}}\frac{\partial v}{\partial \tilde{r}}. \tag{3.22}$$

Equation (3.22) can be solved using a separation of variables $v(\tilde{r},\tilde{t}) = V(\tilde{r})T(\tilde{t})$. Inserted into (3.22) we obtain

$$\begin{aligned} V\frac{\mathrm{d}T}{\mathrm{d}\tilde{t}} &= T\left(\frac{\mathrm{d}^2 V}{\mathrm{d}\tilde{r}^2} + \frac{1}{\tilde{r}}\frac{\mathrm{d}V}{\mathrm{d}\tilde{r}}\right), \\ V\dot{T} &= T\left(V'' + \frac{1}{\tilde{r}}V'\right), \\ \frac{\dot{T}}{T} &= \frac{V'' + \frac{1}{\tilde{r}}V'}{V}, \end{aligned}$$

where the dot and apostrophe represent ordinary derivatives with respect to $\tilde{t}$ and $\tilde{r}$ respectively. Since the left hand side depends only on $\tilde{t}$ and the right hand side only on $\tilde{r}$, both sides must be equal to a constant. It turns out that the constant must be negative to satisfy the appropriate boundary conditions. We call the new constant $-\lambda^2$ and obtain the two separate ordinary differential equations

$$\begin{aligned} \dot{T} + \lambda^2 T &= 0, \\ \tilde{r}^2 V'' + \tilde{r}V' + \lambda^2\tilde{r}^2 V &= 0. \end{aligned}$$

The first equation has solution

$$T(t) = e^{-\lambda^2 \tilde{t}}.$$

The second equation can be rewritten using $x = \tilde{r}\lambda$ and chain rule differentiation leading to

$$x^2 V'' + xV' + x^2 V = 0, \tag{3.23}$$

with the apostrophe here being a derivative with respect to x. Equation (3.23) is Bessel's differential equation of zero order. The solution to Bessel's equation is the Bessel function of the first kind of order zero

$$J_0(x).$$

A solution to V becomes

$$V(\tilde{r}) = AJ_0(\lambda\tilde{r}),$$

where A is a constant. We now need to make sure that the boundary and initial conditions are satisfied. The boundary condition requires $V(1) = J_0(\lambda) = 0$. The figure below shows how the Bessel function behaves. Apparently it looks somewhat like a damped cosine function with multiple roots that we call λ_k. The first four positive roots are located at $ x = 2.4048, 5.5201, 8.6537, 11.7915$ corresponding to $\lambda_1, \lambda_2, \lambda_3, \lambda_4$.

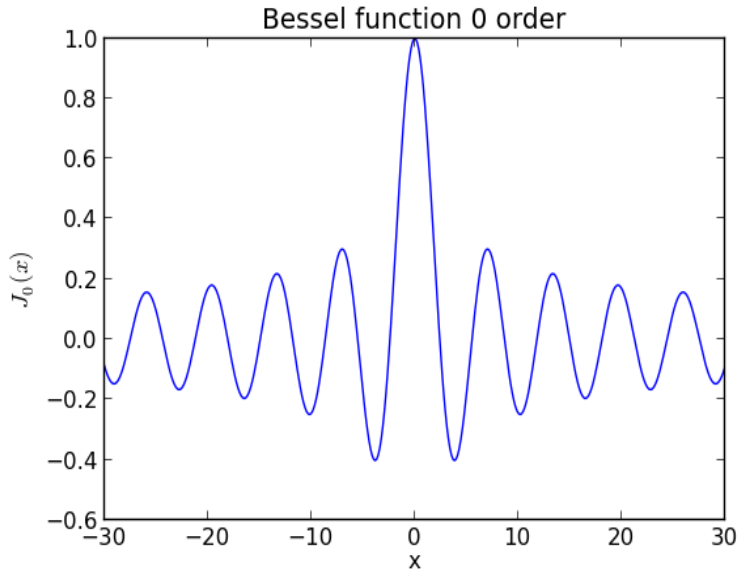


Fig. 3.4: Bessel function

One general solution is thus

$$V(\tilde{r}) = A_k J_0(\lambda_k \tilde{r})$$

with a total solution

$$v(\tilde{r}, \tilde{t}) = A_k J_0(\lambda_k \tilde{r}) e^{-\lambda_k^2 \tilde{t}}.$$

Since the equations are linear we can use superposition and obtain a total solution simply by summing over multiple possible solutions

$$v(\tilde{r}, \tilde{t}) = \sum_{k=1}^{\infty} A_k J_0(\lambda_k \tilde{r}) e^{-\lambda_k^2 \tilde{t}}.$$

We now have a solution consisting of many constants that satisfy the boundary condition. It remains to close these constants using the initial condition $v(\tilde{r},0) = -u_m(1-\tilde{r}^2)$.

We have

$$\sum_{k=1}^{\infty} A_k J_0(\lambda_k \tilde{r}) = -u_m(1-\tilde{r}^2) \tag{3.24}$$

and we want to find A_k. To this end we use orthogonality

$$\int_0^1 x J_n(\lambda_k x) J_n(\lambda_m x) dx = 0 \quad \text{for } k \neq m \tag{3.25}$$

and orthonormality

$$\int_0^1 x J_n(\lambda_k x) J_n(\lambda_k x) dx = \frac{J_{n+1}^2(\lambda_k)}{2}. \tag{3.26}$$

If (3.24) is multiplied by $\tilde{r}J_0(\lambda_m \tilde{r})$ and integrated from zero to unity we get

$$\int_0^1 \tilde{r} J_0(\lambda_m \tilde{r}) \sum_{k=1}^{\infty} A_k J_0(\lambda_k \tilde{r}) \mathrm{d}\tilde{r} = -\int_0^1 u_m(1-\tilde{r}^2)\tilde{r} J_0(\lambda_m \tilde{r}) \mathrm{d}\tilde{r}.$$

We now use (3.25) to eliminate all terms on the left hand side where $k \neq m$ and insert for (3.26). The result is a closed expression for A_k

$$A_k = -\frac{2}{J_1^2(\lambda_k)} \int_0^1 \tilde{r} J_0(\lambda_k \tilde{r}) u_m(1-\tilde{r}^2) \mathrm{d}\tilde{r}. \tag{3.27}$$

The integral on the right hand side can be evaluated exactly (details left out for now) leading to

$$A_k = -\frac{4u_m J_2(\lambda_k)}{\lambda_k^2 J_1^2(\lambda_k)} = -\frac{8u_m}{\lambda_k^3 J_1(\lambda_k)} \tag{3.28}$$

The final solution to starting flow in a cylinder, (3.21), becomes

$$u(\tilde{r},\tilde{t}) = u_m(1-\tilde{r}^2) - \sum_{k=1}^{\infty} \frac{8u_m J_0(\lambda_k \tilde{r})}{\lambda_k^3 J_1(\lambda_k)} e^{-\lambda_k^2 \tilde{t}}$$

in agreement with Eq. (3-96) in White [Whi06].

Pipe flow due to an oscillating pressure gradient

Consider now the same problem as in the previous section, but with a pressure gradient that is oscillating in time

$$\frac{\partial p}{\partial x} = -\rho K e^{i\omega t},$$

where $i = \sqrt{-1}$ and

$$e^{i\omega t} = \cos \omega t + i \sin \omega t.$$

The governing equation for the cylinder is thus

$$\frac{\partial u}{\partial t} = K e^{i\omega t} + \nu \left(\frac{\partial^2 u}{\partial r^2} + \frac{1}{r}\frac{\partial u}{\partial r} \right). \tag{3.29}$$

We now want solve (3.29) with no-slip boundary conditions on the cylinder walls. We will neglect the initial condition and look only for a long-term steady oscillatory solution on the form

$$u(r,t) = e^{i\omega t}V(r). \tag{3.30}$$

That is, we look for a solution where all the transient behaviour is governed by an oscillating pressure gradient proportional to $e^{i\omega t}$.

Inserting for (3.30) in (3.29) and using the same nomenclature as in the previous section we obtain

$$\begin{aligned} Vi\omega \cancel{e^{i\omega t}} &= K\cancel{e^{i\omega t}} + \nu\cancel{e^{i\omega t}}\left(V'' + \frac{1}{r}V'\right) \\ Vi\omega &= K + \nu\left(V'' + \frac{1}{r}V'\right) \\ r^2V'' + rV' - \frac{i\omega}{\nu}r^2V &= -\frac{Kr^2}{\nu}. \end{aligned}$$

Using $x = r\sqrt{i\omega/\nu}$ and chain rule differentiation the equation can be rewritten as

$$x^2V'' + xV' - x^2V = -\frac{Kx^2}{i\omega}, \tag{3.31}$$

where the apostrophe now represents differentiation with respect to x. This equation looks very much like a modified Bessel equation of zero order. The only difference is the right hand side, which should be zero. To get rid of the right hand side we use another change of variables

$$V = \hat{V} + \frac{K}{i\omega}$$

and insert this in (3.31) such that

$$x^2\hat{V}'' + x\hat{V}' - x^2\hat{V} = 0.$$

The solution to the modified Bessel equation of zero order is the modified Bessel function of the first kind (the second kind cannot be used since this function is infinite for $x = 0$) of zero order, I_0, so

$$\hat{V}(x) = AI_0(x),$$

or in terms of the ordinary Bessel function:

$$\hat{V}(x) = AJ_0(-ix),$$

where A is a constant. With this result we can now express the total solution as

$$u(r,t) = e^{i\omega t}\left(AI_0(r\sqrt{i\omega/\nu}) + \frac{K}{i\omega}\right). \tag{3.32}$$

The boundary condition requires $u(r_0,t) = 0$, which determines A:

$$A = -\frac{K}{i\omega I_0(r_0\sqrt{i\omega/\nu})}.$$

Inserted into (3.32) we are left with

$$u(r,t) = \frac{Ke^{i\omega t}}{i\omega}\left(1 - \frac{I_0(r\sqrt{i\omega})}{I_0(r_0\sqrt{i\omega/\nu})}\right),$$

or

$$u(r,t) = \frac{Ke^{i\omega t}}{i\omega}\left(1 - \frac{J_0(r\sqrt{-i\omega/\nu})}{J_0(r_0\sqrt{-i\omega/\nu})}\right),$$

in accordance with Eq. 3-98 in [Whi06].

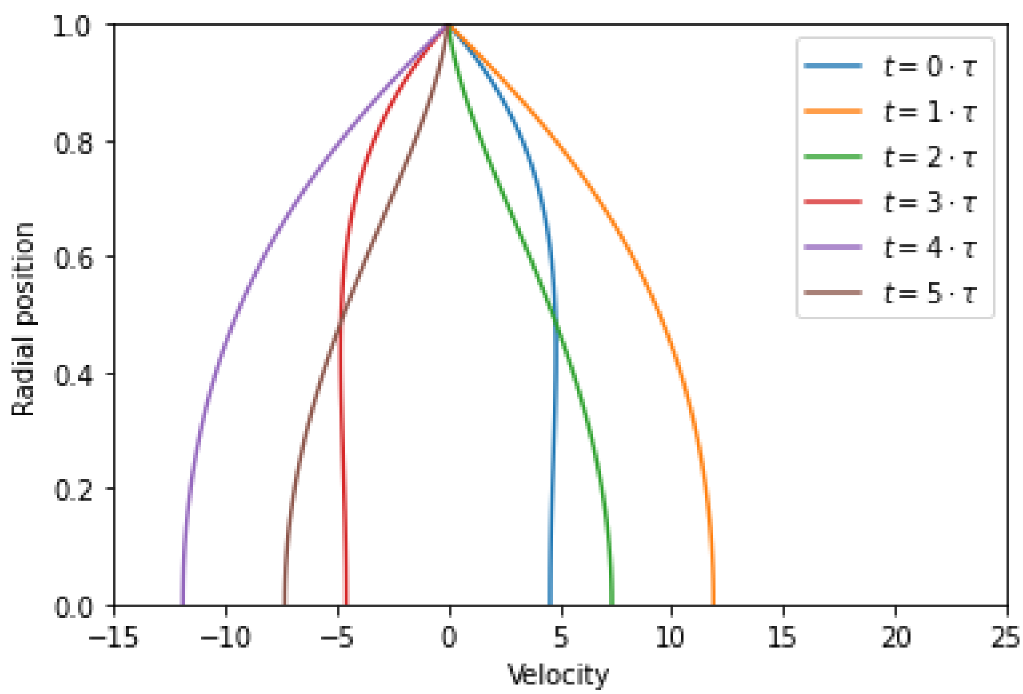
The Bessel function can be found in the toolbox `scipy.special`. The solution is computed below for a relatively large normalized $\tilde{\omega} = \omega r_0^2/\nu = 10$. A smaller value of $\tilde{\omega}$ will result in velocity profiles closer to the steady Poiseuille profile $u_{max}(1-\tilde{r}^2)$.

```
from numpy import *
import matplotlib.pyplot as plt
from scipy.special import jv
%matplotlib inline

r0 = 1
w = 0.1
K = 1.0
nu = 0.01
r = linspace(0, r0, 50)
ws = w * r0**2 / nu

def u(t):
    return K/1j/w*exp(1j*w*t)*(1-jv(0, r*sqrt(-1j*w/nu))/jv(0, r0*sqrt(-1j*w/nu)))

plt.figure(figsize=(6, 4))
ll = []
for i in range(6):
    print(1/w*i*2*pi/6)
    plt.plot(u(i/(6*w)*2*pi), r)
    ll += [r"$t = {0}\cdot \tau$".format(i)]
plt.legend(ll)
plt.axis([-15, 25, 0, 1])
plt.xlabel("Velocity")
plt.ylabel("Radial position")
plt.show()
```

```
0.0
10.471975511965978
20.943951023931955
31.415926535897928
41.88790204786391
52.35987755982989
```

```
/Users/mikaelmortensen/opt/anaconda3/envs/mek4300/lib/python3.8/site-packages/
↪numpy/core/_asarray.py:83: ComplexWarning: Casting complex values to real␣
↪discards the imaginary part
  return array(a, dtype, copy=False, order=order)
/Users/mikaelmortensen/opt/anaconda3/envs/mek4300/lib/python3.8/site-packages/
↪numpy/core/_asarray.py:83: ComplexWarning: Casting complex values to real␣
↪discards the imaginary part
  return array(a, dtype, copy=False, order=order)
/Users/mikaelmortensen/opt/anaconda3/envs/mek4300/lib/python3.8/site-packages/
↪numpy/core/_asarray.py:83: ComplexWarning: Casting complex values to real␣
↪discards the imaginary part
  return array(a, dtype, copy=False, order=order)
```

```
/Users/mikaelmortensen/opt/anaconda3/envs/mek4300/lib/python3.8/site-packages/
↪numpy/core/_asarray.py:83: ComplexWarning: Casting complex values to real␣
↪discards the imaginary part
  return array(a, dtype, copy=False, order=order)
/Users/mikaelmortensen/opt/anaconda3/envs/mek4300/lib/python3.8/site-packages/
↪numpy/core/_asarray.py:83: ComplexWarning: Casting complex values to real␣
↪discards the imaginary part
  return array(a, dtype, copy=False, order=order)
/Users/mikaelmortensen/opt/anaconda3/envs/mek4300/lib/python3.8/site-packages/
↪numpy/core/_asarray.py:83: ComplexWarning: Casting complex values to real␣
↪discards the imaginary part
  return array(a, dtype, copy=False, order=order)
```



Suggested assignments Poiseuille

Poiseuille problems from White

Problems 3-15, 3-16 and 3-17.

Computer exercise Poiseuille

Verify analytical solutions

In [Whi06] Chap. 3-3.3 there are 6 analytical solutions provided for 6 different types of ducts. Verify the analytical solutions (3-47), (3-49) and (3-52). Experiment with higher order "CG" elements (the solution is then higher order continuous piecewise polynomials) and compute the `errornorm`. Does the error vanish for higher order polynomials for all ducts or does it remain constant?

3.2 Similarity solutions

Similarity solutions are obtained when the number of independent variables describing a problem is reduced by at least one.

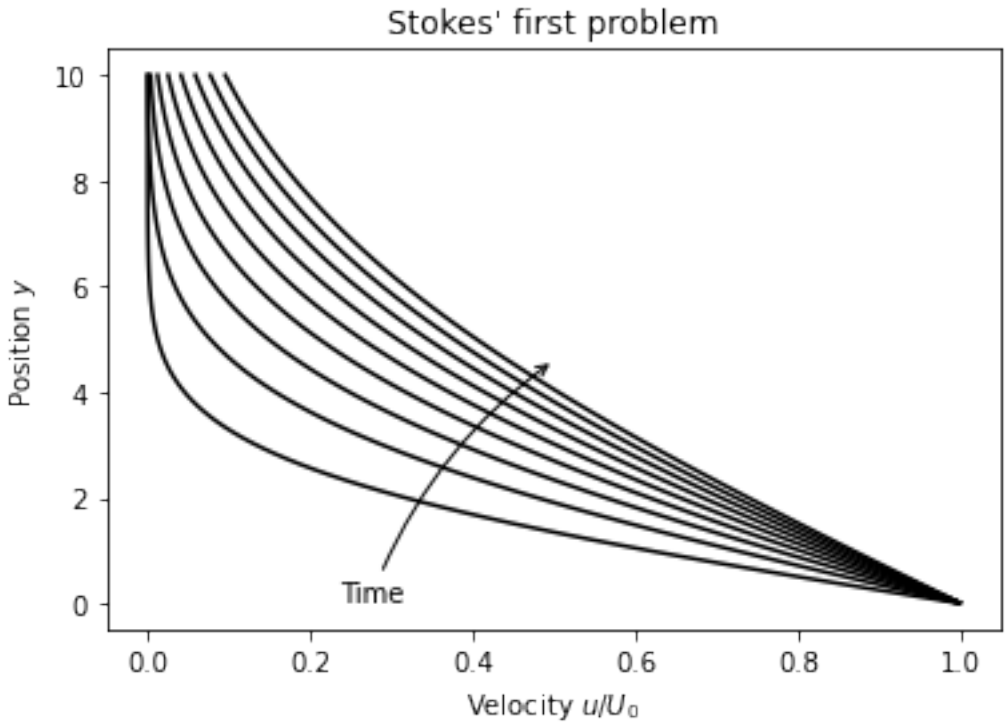
3.2.1 Stokes' first problem revisited

We will first illustrate this by revisiting Stokes' first problem - A fluid initially at rest in an infinite domain ($y \in [0, \infty)$) is suddenly set in motion by an infinite flat plate located at $y = 0$. The problem is described by the equations

$$\begin{aligned} \frac{\partial u}{\partial t} &= \nu \frac{\partial^2 u}{\partial y^2} \text{ for } y \in [0, \infty), \\ u(\infty, t) &= 0, \\ u(y, t < 0) &= 0 \\ u(0, t > 0) &= U_0 \end{aligned} \tag{3.33}$$

We have already found the solution to this problem using Fourier integrals with the solution as shown below

```
import numpy as np
from scipy.special import erf
import matplotlib.pyplot as plt
%matplotlib inline

nu = 0.01
U0 = 1.
y = np.linspace(0.01, 10, 100)
plt.figure(figsize=(6, 4))
for i in range(1, 10):
    t = i * 200.
    u = U0*(1-erf(y/2/np.sqrt(nu*t)))
    plt.plot(u, y, "k")
plt.annotate(r'Time', xy=(0.5, 4.6),
             xytext=(-80, -90), textcoords='offset points',
             arrowprops=dict(arrowstyle="->", connectionstyle="arc3,rad=-.15"))
plt.xlabel("Velocity " + r"$u/U_0$")
plt.ylabel("Position " + "$y$")
plt.title("Stokes' first problem")
plt.show()
```



The solution is repeated here

$$u(y,t) = U_0 \left(1 - \mathrm{erf}\left(\frac{y}{2\sqrt{\nu t}}\right)\right).$$

We will now look at this problem again and try to find this solution using a similarity approach. First define a similarity variable

$$\eta = \frac{y}{2\sqrt{\nu t}},$$

and observe that the solution can be written in terms of merely one variable

$$u(\eta) = U_0 \left(1 - \mathrm{erf}(\eta)\right).$$

Usually we don't know the solution and have to guess a similarity variable. Having defined η, the heat equation (3.33) can be rewritten using chain rule differentiation

$$\frac{\mathrm{d}u}{\mathrm{d}\eta}\frac{\partial \eta}{\partial t} - \nu\left(\frac{\mathrm{d}^2 u}{\mathrm{d}\eta^2}\left(\frac{\partial \eta}{\partial y}\right)^2 + \frac{\mathrm{d}u}{\mathrm{d}\eta}\frac{\partial^2 \eta}{\partial y^2}\right) = 0.$$

The partial derivatives of η are

$$\begin{aligned}
\frac{\partial \eta}{\partial t} &= -\frac{\eta}{2t}, \\
\frac{\partial \eta}{\partial y} &= \frac{1}{2\sqrt{\nu t}}, \\
\frac{\partial^2 \eta}{\partial y^2} &= 0.
\end{aligned}$$

Inserting this into the above leads to and ordinary differential equation for $u(\eta)$, which is exactly what we were hoping to achieve

$$2\eta\frac{\mathrm{d}u}{\mathrm{d}\eta} + \frac{\mathrm{d}^2 u}{\mathrm{d}\eta^2} = 0.$$

This ordinary differential equation must be solved using boundary conditions $u(0) = U_0$ and $u(\infty) = 0$. Define first $h(\eta) = \mathrm{d}u/\mathrm{d}\eta$ and rewrite the ODE as

$$2\eta h + \frac{\mathrm{d}h}{\mathrm{d}\eta} = 0.$$

Separate variables and integrate to obtain

$$\frac{\mathrm{d}u}{\mathrm{d}\eta} = Ae^{-\eta^2}.$$

Integrate one more time to obtain

$$u(\eta) = A\mathrm{erf}(\eta) + B$$

Inserting for the boundary conditions we finally get the original solution derived with much more effort in *Unsteady Couette flows*.

3.2.2 Plane stagnation flow

Another well known and much used similarity solution is the streamfunction $\psi(\boldsymbol{x}, t)$, that is most often used for two-dimensional flows, where it is defined through

$$u = \frac{\partial \psi}{\partial y}, \quad v = -\frac{\partial \psi}{\partial x}. \tag{3.34}$$

In 2D the only non-zero component of the vorticity vector, ω_z, is computed as

$$\omega_z = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y},$$

If this definition is combined with (3.34) we obtain a Poisson equation for the streamfunction

$$\nabla^2 \psi = -\omega_z. \tag{3.35}$$

The streamfunction is much used for inviscid irrotational flows, where the vorticity component $\omega_z = 0$. However, the definition of the streamfunction is equally valid for viscous flows.

An illustration of plane stagnation flow is given in Fig. (3-24) of [Whi06]. For inviscid flows the solution is trivially obtained as

$$\psi = Bxy, \quad u = Bx, \quad v = -By,$$

where B is a positive constant. Make sure to check that the solution satisfies both (3.35) (with zero right hand side) and the continuity equation

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0.$$

For viscous flows it is somewhat more complicated, since the right hand side of (3.35) is nonzero. To find the solution for viscous flows we also need to consider the momentum equations in two dimensions

$$\begin{aligned}
\underline{u\frac{\partial u}{\partial x}} + v\frac{\partial u}{\partial y} &= -\frac{1}{\rho}\frac{\partial p}{\partial x} + \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right), \\
u\frac{\partial v}{\partial x} + \underline{v\frac{\partial v}{\partial y}} &= -\frac{1}{\rho}\frac{\partial p}{\partial y} + \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right).
\end{aligned}$$

These equations are nonlinear due to the two underlined terms and coupled through the continuity equation. Note that there are three equations for the three unknowns u, v, p, so the equations are closed. We will now look for a similarity solution through a modification of the streamfunction of the form

$$\psi_\nu = Bxf(y), \quad u = \frac{\partial \psi_\nu}{\partial y} = Bx\frac{df}{dy}, \quad v = -\frac{\partial \psi_\nu}{\partial x} = -Bf.$$

Using an apostrophe for the derivative of f with respect to y one has immediately

$$\frac{\partial u}{\partial x} = Bf', \quad \frac{\partial u}{\partial y} = Bxf'', \quad \frac{\partial^2 u}{\partial x^2} = 0, \quad \frac{\partial^2 u}{\partial y^2} = Bxf''',$$
$$\frac{\partial v}{\partial x} = 0, \quad \frac{\partial v}{\partial y} = -Bf', \quad \frac{\partial^2 v}{\partial x^2} = 0, \quad \frac{\partial^2 v}{\partial y^2} = -Bf''.$$

We immediately see that the continuity equation is satisfied. Inserting for the streamfunction in (3.36) we obtain

$$B^2x(f')^2 - B^2xff'' - \nu Bxf''' = -\frac{1}{\rho}\frac{\partial p}{\partial x},$$
$$B^2ff' + \nu Bf'' = -\frac{1}{\rho}\frac{\partial p}{\partial y}$$

The last equation here states that $\partial p/\partial y$ is only a function of y and thus

$$\frac{\partial^2 p}{\partial x \partial y} = 0. \tag{3.36}$$

The momentum equation in the x-direction can be simplified by dividing through by $-\nu Bx$

$$f''' + \frac{B}{\nu}\left(ff'' - (f')^2\right) = \frac{1}{\nu Bx\rho}\frac{\partial p}{\partial x} \tag{3.37}$$

The left hand side is only a function of y and as such we need the right hand side also as a function of only y. For this to happen it is evident that the pressure derivative $\partial p/\partial x = \text{const}\, x\, h(y)$, where $h(y)$ is some function of y. However, from Eq. (3.36) it follows that h must be a constant as well and as such the right hand side of (3.37) must be constant. Since the right hand side is constant, this means that the left hand side also must be a constant.

Treating the right hand side as a constant we can compute its value using the boundary conditions $f' = 1, f'' = f''' = 0$ for $y \longrightarrow \infty$ (the first follows from freestream condition $u_{viscous} = u_{inviscid}$). Inserted for the boundary conditions we obtain finally

$$f''' + \frac{B}{\nu}\left(ff'' - (f')^2\right) = -\frac{B}{\nu}. \tag{3.38}$$

Note that (3.38) is an exact equation for stagnation flow, since no terms have been eliminated in its derivation. The equation is nonlinear, which calls for iterative numerical solution techniques. The equation can be non-dimensionalised using length- and velocity-scales $\sqrt{\nu/B}$ and $\sqrt{\nu B}$ respectively, and the dimensionless variables

$$\eta = y\sqrt{\frac{B}{\nu}}, \quad \psi = xF(\eta)\sqrt{B\nu}. \tag{3.39}$$

We can now easily obtain

$$f(y) = F\sqrt{\frac{\nu}{B}},$$
$$f' = \frac{dF}{d\eta}\frac{d\eta}{dy}\sqrt{\frac{\nu}{B}} = F'\sqrt{\frac{B}{\nu}}\sqrt{\frac{\nu}{B}} = F',$$
$$f'' = \frac{dF'}{d\eta}\frac{d\eta}{dy} = F''\sqrt{\frac{B}{\nu}},$$
$$f''' = \frac{dF''}{d\eta}\frac{d\eta}{dy} = F'''\frac{B}{\nu}.$$

Inserted into (3.38) we obtain the dimensionless equation

$$F''' + FF'' + 1 - (F')^2 = 0, \tag{3.40}$$

that can be solved using boundary conditions $F(0) = F'(0) = 0$ and $F'(\infty) = 1$.

3.2.3 Axisymmetric stagnation flow

The plane stagnation flow discussed in the previous section is stagnant for a line in the z-plane located at $x = 0$. Axisymmetric stagnation flow is similar in form, but only stagnant in a single point located at $x = y = z = 0$. This axisymmetric flow is what you obtain by pointing a garden hose directly towards a plane wall (neglecting gravity). The equations are similar to the plane flow, but may be simplified using cylindrical coordinates. A streamfunction in cylindrical coordinates can be defined through

$$u_r = -\frac{1}{r}\frac{\partial \psi}{\partial z}, \quad u_z = \frac{1}{r}\frac{\partial \psi}{\partial r} \tag{3.41}$$

For axisymmetric flows with $u_\theta = 0$ the continuity equation reads

$$\frac{1}{r}\frac{\partial r u_r}{\partial r} + \frac{\partial u_z}{\partial z} = 0.$$

Inserting for (3.41) it is easily verified that continuity is satisfied.

The streamfunction for axisymmetric inviscid flow towards a stagnation point is given as

$$\psi = -Br^2 z$$

which leads to velocity components

$$u_r = Br, \quad u_z = -2Bz$$

For viscous flows one may obtain a dimensionless similarity solution using

$$\psi = -r^2 F(\eta)\sqrt{B\nu}, \quad \eta = z\sqrt{\frac{B}{\nu}}.$$

Inserting into the momentum equation for u_r the mathematics are very much like in the previous section leading to

$$F''' + 2FF'' + 1 - (F')^2 = 0. \tag{3.42}$$

This differs from plane stagnation flow only by the factor 2.

3.2.4 Suggested assignments similarity solutions

3.3 Stokes flow

Flows at very low Reynolds numbers are often called *Stokes flow*. If the Navier-Stokes equations are nondimensionalized using the characteristic velocity U and length scale L, the equations can be simplified as

$$\begin{gathered} \mathrm{Re}\left(\frac{\partial \tilde{\boldsymbol{u}}}{\partial \tilde{t}} + (\tilde{\boldsymbol{u}}\cdot\tilde{\nabla})\tilde{\boldsymbol{u}}\right) = -\tilde{\nabla}\tilde{p} + \tilde{\nabla}^2\tilde{\boldsymbol{u}}, \\ \tilde{\nabla}\cdot\tilde{\boldsymbol{u}} = 0, \end{gathered} \tag{3.43}$$

where $\tilde{\boldsymbol{u}} = \boldsymbol{u}/U$, $\tilde{t} = tU/L$, $\tilde{\boldsymbol{x}} = \boldsymbol{x}/L$, $\tilde{p} = p/(\mu U/L)$ and $\mathrm{Re} = \rho UL/\mu$. The spatial derivative is taken with respect to $\tilde{\boldsymbol{x}}$, which explains the tilde on the nabla operator. The assumption for Stokes flow is that the Reynolds number is

very small $\mathrm{Re} << 1$, such that the acceleration term on the left hand side of Navier-Stokes can be neglected. Stokes flow is therefore governed, on dimensional form, by

$$\begin{aligned} \mu\nabla^2\boldsymbol{u} &= \nabla p, \\ \nabla\cdot\boldsymbol{u} &= 0. \end{aligned} \tag{3.44}$$

Here the first (vector) equation is a momentum equation, whereas the second $\nabla\cdot\boldsymbol{u}$ usually is referred to as the continuity eqaution.

Conveniently there is a FEniCS demo available for solving the Stokes equations in a three-dimensional cubic domain. In this demo the velocity and pressure are solved using a coupled system of equations. That is, both equations in (3.44) are solved simultaneously as opposed to in a segregated, sequential manner. Since the first equation in (3.44) is a vector equation, this means that we are actually solving 4 equations (if the domain is 3D) simultaneously. Both $\boldsymbol{u}$ and p are treated as unknowns (`TrialFunctions`) using a mixed (coupled) functionspace.

The variational formulation for the Stokes equations is found by multiplying the vector equation in (3.44) with `TestFunction` $\boldsymbol{v}$ and the continuity equation with `TestFunction` q and then integrating over the domain

$$\begin{aligned} \mu\int_\Omega \boldsymbol{v}\cdot\nabla^2\boldsymbol{u}\,\mathrm{d}x &= \int_\Omega \boldsymbol{v}\cdot\nabla p\,\mathrm{d}x, \\ \int_\Omega q\,\nabla\cdot\boldsymbol{u}\,\mathrm{d}x &= 0. \end{aligned}$$

Both the Laplacian and pressure gradient are integrated by parts

$$\begin{aligned} \mu\int_\Gamma \boldsymbol{v}\cdot(\nabla\boldsymbol{u}\cdot\boldsymbol{n})\,\mathrm{d}s - \mu\int_\Omega \nabla\boldsymbol{v}:\nabla\boldsymbol{u}\,\mathrm{d}x &= \int_\Gamma \boldsymbol{v}\cdot p\boldsymbol{n}\,\mathrm{d}s - \int_\Omega p\,\nabla\cdot\boldsymbol{v}\,\mathrm{d}x, \\ \int_\Omega q\,\nabla\cdot\boldsymbol{u}\,\mathrm{d}x &= 0, \end{aligned}$$

where $\boldsymbol{n}$ is the normal vector out of the domain and the colon represents an inner product (contraction) of two second order tensors. The boundary integral can be written as

$$\int_\Gamma \boldsymbol{v}\cdot(p\boldsymbol{I}-\mu\nabla\boldsymbol{u})\cdot\boldsymbol{n}\,\mathrm{d}s$$

and you should recognize $(p\boldsymbol{I}-\mu\nabla\boldsymbol{u})\cdot\boldsymbol{n}$ as part of the total stress $\tau = -p\boldsymbol{I}+\mu(\nabla\boldsymbol{u}+\nabla\boldsymbol{u}^T)$ acting on the surface with normal vector $\boldsymbol{n}$. In finite element communities the boundary condition $(p\boldsymbol{I}-\mu\nabla\boldsymbol{u})\cdot\boldsymbol{n}=0$ is called pseudo-traction or a do-nothing boundary condition. It is often used on outlets where we are simply interested in letting the fluid escape with as little interference of boundary conditions as possible. It is called do-noting because you don't have to do anything to enforce it, just simply omit the boundary terms and solve the two equations

$$\begin{aligned} \mu\int_\Omega \nabla\boldsymbol{v}:\nabla\boldsymbol{u}\,\mathrm{d}x - \int_\Omega p\,\nabla\cdot\boldsymbol{v}\,\mathrm{d}x &= 0, \\ \int_\Omega q\,\nabla\cdot\boldsymbol{u}\,\mathrm{d}x &= 0. \end{aligned}$$

In FEniCS we may set up these two equations (one vector and one scalar equation) to be solved in a coupled manner. The approach is given below:

```
from dolfin import *
import matplotlib.pyplot as plt
%matplotlib inline

mesh = UnitSquareMesh(20, 20)

# Build function space
```

(continues on next page)

(continued from previous page)

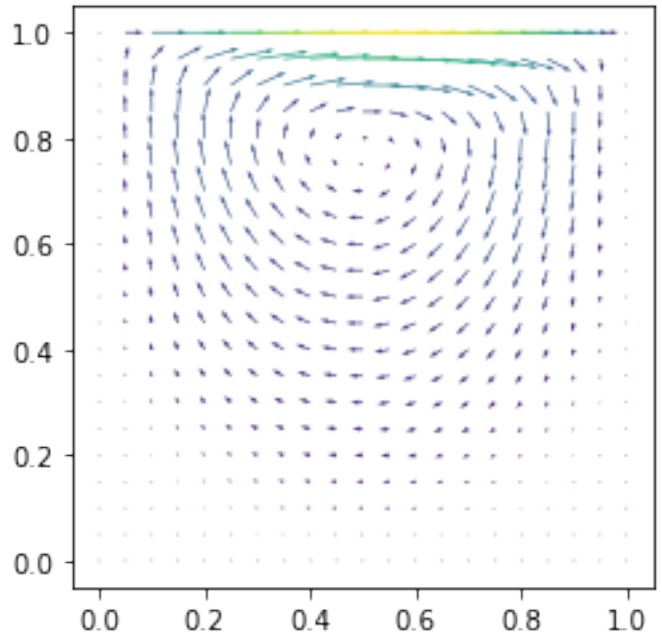
```
P2 = VectorElement("CG", mesh.ufl_cell(), 2)
P1 = FiniteElement("CG", mesh.ufl_cell(), 1)
TH = P2 * P1
VQ = FunctionSpace(mesh, TH)
u, p = TrialFunctions(VQ)
v, q = TestFunctions(VQ)

mu = Constant(0.1)
F = mu*inner(grad(v), grad(u))*dx - inner(div(v), p)*dx - inner(q, div(u))*dx
```

Note that terms involving `TestFunction v` end up in the vector equation, whereas terms containing `TestFunction q` end up in the scalar equation. All terms defined in F are bilinear, in that they all contain one `TestFunction` and one `TrialFunction`. The sign of the velocity divergence has been set to negative, because this leads to a symmetric coefficient matrix, which is favorable for iterative solvers (even though we are not using a iterative solver here).

To solve the equations we need to create a `Function` in the mixed functionspace to hold the solution, for example like this:

```
up_ = Function(VQ)
bc0 = DirichletBC(VQ.sub(0), Expression(("x[0]*(1-x[0])", "0"), degree=0),
↪"std::abs(x[1])>1-1e-12 && on_boundary")
bc1 = DirichletBC(VQ.sub(0), Constant((0, 0)), "std::abs(x[0]*(1-x[0])*x[1])<1e-12 &&␣
↪on_boundary")
solve(lhs(F) == rhs(F), up_, bcs=[bc0, bc1])
u_, p_ = up_.split()
plot(u_)
```

```
<matplotlib.quiver.Quiver at 0x7f8b51fb5ca0>
```



The boundary conditions will of course depend on the problem being solved. Here we have set homogeneous Dirichlet (no-slip) on three walls and a parabolic profile on top, which corresponds to a modified lid driven cavity. Note that boundary conditions for velocity are set using `VQ.sub(0)`, where `sub(0)` corresponds to subspace 0 of the mixed

`VQ` space. A Dirichlet boundary condition on pressure could have been set using `VQ.sub(1)`, but here that would be an over-specification.

The stream function $\psi(\boldsymbol{x}, t)$ was defined in Eq. (3.34) in Similarity solutions. The Poisson equation (3.35) for the streamfunction, can be transformed to a variational form by multiplying with a `TestFunction` ψ_v and then integrating over the domain using integration by parts

$$\int_\Omega \psi_v \nabla^2 \psi \mathrm{d}x = -\int_\Omega \psi_v \omega_z \mathrm{d}x,$$
$$-\int_\Omega \nabla(\psi_v) \cdot \nabla(\psi) \mathrm{d}x + \int_\Gamma \psi_v \nabla\psi \cdot \boldsymbol{n} \mathrm{d}s = -\int_\Omega \psi_v \omega_z \mathrm{d}x.$$

Here $\boldsymbol{n}$ is the normal vector pointing outwards from the external boundary. The normal can be used directly in FEniCS forms through, e.g., `n=FacetNormal(mesh)`.

By taking the curl of the momentum equation (3.44) and using that the curl of a gradient is always zero, an equation for the vorticity vector $\boldsymbol{\omega}$ is obtained

$$\nabla^2 \boldsymbol{\omega} = 0.$$

Likewise, by taking the divergence (not gradient as it says in White [Whi06]) of the momentum equation (3.44) we obtain an equation for the pressure

$$\nabla^2 p = 0,$$

which follows since $\nabla \cdot \nabla^2 \boldsymbol{u} = \nabla^2(\nabla \cdot \boldsymbol{u}) = 0$. For Stokes flow both the pressure and the vorticity are governed by homogeneous Laplace equations. For 2D flows we also know the equation for the streamfunction $\nabla^2\psi = -\omega_z$. Inserting this into the Laplace equation for ω_z we obtain

$$\nabla^4 \psi = 0,$$

which is a biharmonic equation for the streamfunction. Obviously, if ψ is a solution to $\nabla^4\psi = 0$, then $-\psi$ is a solution as well. In other words, the direction of flow along a streamline is arbitrary and only fixed due to boundary conditions. Reversing the direction of the flow (like we do in the mandatory assignment) will not change the streamlines, but the values will get the opposite sign.

Stokes paradox states that it is not possible to obtain steady solutions to $\nabla^4\psi = 0$ for 2D flows with both free stream and no-slip boundary conditions. Stokes paradox is, like d'Alembert's paradox, really only a paradox because the equations do not represent real physical flow, just approximations in the limit. It follows from Stokes paradox that inertia can never be neglected for 2D flows with both free stream and no-slip boundary conditions.

3.3.1 Creeping flow past a sphere

Stokes flow past a sphere of radius a is a 3D flow and thus solutions to $\nabla^4\psi = 0$ may be found even though there is no-slip on the sphere wall and free stream far away. In spherical polar coordinates the stream function, ψ, is defined through

$$u_r = \frac{1}{r^2 \sin\theta} \frac{\partial \psi}{\partial \theta}, \quad u_\theta = -\frac{1}{r \sin\theta} \frac{\partial \psi}{\partial r}. \tag{3.45}$$

The streamfunction equation becomes for spherical coordinates

$$\left(\frac{\partial^2}{\partial r^2} + \frac{1}{r} \frac{\partial^2}{\partial \theta^2} - \frac{\cot\theta}{r^2} \frac{\partial}{\partial \theta} \right)^2 \psi = 0,$$

with boundary conditions

$$\frac{\partial \psi}{\partial r} = \frac{\partial \psi}{\partial \theta} = 0, \quad \text{for} \quad r = a,$$
$$\psi = \frac{1}{2} U r^2 \sin^2\theta + \text{const}, \quad \text{for} \quad r \longleftrightarrow \infty.$$

Using separation of variables White reports that the solution to this problem reads

$$\psi = \frac{1}{4} U a^2 \sin^2\theta \left(\frac{a}{r} - \frac{3r}{a} + \frac{2r^2}{a^2} \right),$$

and thus we obtain the velocity components from (3.45)

$$u_r = U \cos\theta \left(1 + \frac{a^3}{2r^3} - \frac{3a}{2r} \right),$$
$$u_\theta = U \sin\theta \left(-1 + \frac{a^3}{4r^3} + \frac{3a}{4r} \right)$$

A few important notes

- The velocity is independent of ρ and μ, which is actually true for all creeping flows.
- There is perfect symmetry of the streamlines past the sphere. No wake and no separation.
- The velocity is everywhere smaller than the free stream. This is contrary to inviscid flows where the velocity is maximum at the top and bottom of the sphere. For Stokes flow the velocity will here be zero due to no-slip.
- The effect of the sphere can be observed at enormous distances away from the sphere. At $r = 10a$ the velocity is still 10% lower than the free stream.

The pressure can be computed from the momentum equations in radial and θ-direction

$$\frac{\partial p}{\partial r} = \mu \left(\nabla^2 u_r - \frac{2u_r}{r^2} - \frac{2}{r^2} \frac{\partial u_\theta}{\partial \theta} - \frac{2u_\theta \cot\theta}{r^2} \right),$$
$$\frac{\partial p}{\partial \theta} = \mu r \left(\nabla^2 u_\theta + \frac{2}{r^2} \frac{\partial u_r}{\partial \theta} - \frac{u_\theta}{r^2 \sin^2\theta} \right)$$

Insert for u_r and u_θ and integrate both equations, using boundary conditions, to obtain

$$p(r,\theta) = p_\infty - \frac{3\mu a U}{2r^2} \cos\theta, \tag{3.46}$$

where p_∞ is the pressure in the free stream. From (3.46) it is evident that the pressure is highest at the front (left hand side) of the sphere where $\theta = \pi$, and lowest at the rear where $\theta = 0$. The pressure difference sets up a force acting on the sphere in the streamwise direction. The total force acting on the sphere is a vector that can be computed as

$$\boldsymbol{F} = \int_\Gamma \tau \cdot \boldsymbol{n} \, \mathrm{d}A,$$

where as before the total stress is composed of pressure and friction through the stress tensor $\tau = -p\boldsymbol{I} + \mu(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T)$. The normal to the surface of the sphere is everywhere aligned with the r-direction such that we can write

$$\begin{aligned} \boldsymbol{F_p} + \boldsymbol{F_\tau} &= \int_\Gamma \left(-p\boldsymbol{I} + \mu(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T) \right) \cdot \boldsymbol{n} \, \mathrm{d}A, \\ &= \int_0^\pi (\tau_{rr} + \tau_{r\theta}) 2\pi a^2 \sin\theta \mathrm{d}\theta, \end{aligned}$$

where τ_{rr} and $\tau_{r\theta}$ are the pressure and friction stresses respectively. This follows since the pressure force is always acting normal to the plane and shear acts in the tangent plane. Remember that τ_{ij} is a force acting in the j-direction on a plane having i as the normal direction.

The pressure has already been computed. The remaining shear stress can be computed in spherical coordinates as

$$\begin{aligned}\tau_{r\theta} &= \mu\left(\frac{1}{r}\frac{\partial u_r}{\partial \theta} + \frac{\partial u_\theta}{\partial r} - \frac{u_\theta}{r}\right), \\ &= -\frac{\mu U \sin\theta}{r}\frac{3a^3}{2r^3}.\end{aligned}$$

The total force acting in the streamwise direction is computed by inserting for τ_{rr} and $\tau_{r\theta}$ leading to two simple integral as

$$\begin{aligned}F_x &= |\boldsymbol{F_p}|\cos\theta + |\boldsymbol{F_\tau}|\sin\theta \\ &= 3\mu U\pi a\left(\frac{2}{3} + \frac{4}{3}\right), \\ &= 6\mu\pi U a.\end{aligned}$$

In other words, the drag force consists of 2/3 friction and 1/3 pressure.

CHAPTER

FOUR

CHAPTER 4

Important topics of Chapter 4 [Whi06].

1. Displacement thickness
2. Momentum thickness
3. Flat plate integral analysis
4. Boundary layer equations
5. Flow separation
6. Blasius solution
7. Falkner-Skan solution

4.1 Suggested assignments chapter 4

4.1.1 Written assignments chapter 4

Derive the laminar boundary layer equations

Derive Eq. (4-32) in [Whi06] starting from the Navier-Stokes equations.

Derive nonlinear similarity solutions for boundary layers

Derive both the Blasius and Falkner-Skan equations.

4.1.2 Computer assignments chapter 4

Implement nonlinear solvers

Implement a nonlinear solver for the Blasius equation. Do the same for the Falkner-Skan equation and vary the parameter β as shown in Fig. 4-11 in [Whi06]. What happens at $\beta = -0.19884$? At $-0.19884 \leq \beta \leq 0$ there are according to White at least two possible solutions. One solution is shown in Fig. 4-11a in White. Another solution shows negative values for $f''(0)$, thus indicating backflow close to the wall. Using different initial guesses for the solution it should be possible to obtain both solutions.

Compute the flow past a rotating cylinder.

Use a rectangular domain $\Omega = [0, 6] \times [-1, 1]$ and place the cylinder at $(x, y) = (1, 0)$ with a radius of 0.25. Assume at first steady creeping flow, walls at $y = \pm 1$ and set the velocity on the left hand side ($x = 0$) to $1 - y^2$. Set the cylinder to rotate against the clock at a constant speed of 1. Compute lift and drag forces on the cylinder. Reverse the direction of the rotation and recompute lift and drag. Now include convection and solve the Steady (nonlinear) Navier-Stokes equations for the same problem. Set the rotation to oscillate with an angular speed of $\sin \pi t$ and compute the unsteady creeping flow past the cylinder. Do the same for unsteady Navier-Stokes and attempt to find the von Kármán street for a steady cylinder.

CHAPTER

SIX

CHAPTER 6

6.1 Incompressible turbulent mean flow

Turbulent flows are governed by the Navier-Stokes equations. However,

- there are no simplifications possible by dimensional analysis or through similarity solutions,
- no exact definition is known for turbulence,
- turbulent flows are not very well understood.

Turbulent flows are usually *defined* by listing their most characteristic properties. Here is what we know

- Turbulence is a *flow* property and not a *fluid* property.
- Turbulence is always three-dimensional in space and transient. There is no such thing as two-dimensional turbulence.
- Turbulence is irregular, with seemingly random and chaotic motion. However, there is order to the chaos. The flow satisfies the Navier-Stokes equations, so it is apparently deterministic. Still, N-S are extremely sensitive to initial and boundary conditions (the butterfly-effect). The slightest perturbations in initial or boundary conditions leads to a completely different solution. Thus, *turbulence is stochastic even though the Navier-Stokes equations are deterministic*.
- Turbulent flows consist to a great deal of vortices and vorticity. Vortex stretching and tilting are 3-dimensional phenomena that characterize turbulent flows and these processes can not happen in 2D. If a vortex tube is stretched it must increase its angular velocity to conserve angular momentum. This can only happen in 3D. In 2D vorticity is (unphysically) a conserved quantity.
- The vortices come in a range of scales. The size of the largest vortices (the largest scales) is determined by the geometry of the flow, whereas the size of the smallest vortices (the smallest scales) is determined by viscosity.
- The energy cascade. Energy is introduced into the flow on the largest scales and dissipated on the smallest. Dissipation ε of kinetic energy is defined as

  $$\varepsilon = \mu \boldsymbol{s} : \boldsymbol{s}, \quad \text{where strain} \quad \boldsymbol{s} = 0.5 \left(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T \right)$$

  The strain increases dramatically towards the center of a vortex tube leading to higher dissipation rates in this region.
- Turbulence occur at high Reynolds numbers. The normalized momentum equation reads

  $$\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} = -\nabla p + \frac{1}{Re} \nabla^2 \boldsymbol{u} + \boldsymbol{f}.$$

  Diffusion is stabilizing as it tends to smear out all gradients. Briefly, perturbations (disturbances) to the flow are amplified by convection and dampened by viscosity. If the Reynolds number is large, then diffusion is small and convection dominates, leading to unstable turbulent flows. If the Reynolds number is small then viscosity dominates and perturbations are killed before they can trigger turbulence.

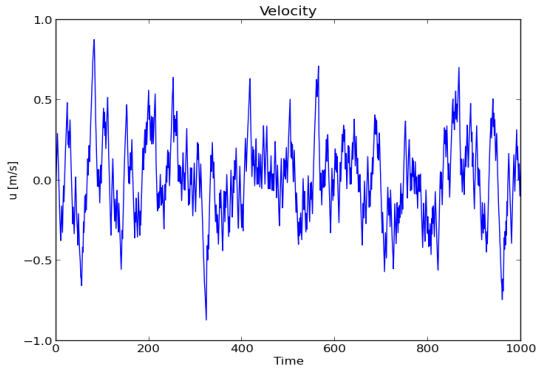


Fig. 6.1: Figure shows 1000 instantaneous measurements of one chaotic velocity component.

Engineers need to be able to predict the behaviour of turbulent flows. However, in a turbulent flow the velocity $\boldsymbol{u}(\boldsymbol{x},t)$ and pressure $p(\boldsymbol{x},t)$ should really be considered as random variables. In other words, at any point in space and time we will not be able to predict the exact value of velocity or pressure. We can, however, predict the statistical properties of these random variables. This is usually accomplished using ensemble averages or *Reynolds averaging*.

Consider first a turbulent flow that is statistically steady, i.e., the statistical properties of the flow are independent of time. This happens, e.g., for a plane channel flow or a straight pipe flow where the applied pressure gradient (the driving force) is held constant. As previously stated, in a turbulent flow the velocity vector $\boldsymbol{u}(\boldsymbol{x},t)$ can be considered a random variable. If you stand in the same spot inside the statistically steady turbulent flow and measure one velocity component, then the signal will look a lot like Gaussian noise, as can be seen in Fig. 6.1.

The *arithmetic* mean of the velocity over these samples can computed as

$$\langle u \rangle (\boldsymbol{x}) = \frac{1}{N} \sum_{i=1}^{N} u(\boldsymbol{x}, t_i)$$

where $u(\boldsymbol{x},t_i)$ is the velocity at time t_i and $N = 1000$ for the turbulent series shown. If the sampling is done in discrete intervals of time $\triangle t$, then $t_i = i\triangle t$. Furthermore, if the samples are taken at sufficiently long time intervals such that $u(\boldsymbol{x},t_i)$ is uncorrelated with $u(\boldsymbol{x},t_i+\triangle t)$ then we talk about an *ensemble* average as

$$\overline{u}(\boldsymbol{x}) = \lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} u(\boldsymbol{x}, t_i).$$

By multiplying through by $\triangle t$ in both numerator and denominator the ensemble average can also be rewritten as

$$\begin{aligned} \overline{u}(\boldsymbol{x}) &= \lim_{N\to\infty} \frac{1}{N\triangle t} \sum_{i=1}^{N} u(\boldsymbol{x}, t_i)\triangle t, \\ &\approx \frac{1}{T} \int_0^T u(\boldsymbol{x}, t)\mathrm{d}t, \end{aligned}$$

where $T = N\triangle t$ is the total sampling time, which typically needs to be larger than the largest timescales of the flow. It can be seen that for a statistically steady flow the ensemble average corresponds to the time average.

For a more general turbulent flow where the statistics can change in time more care needs to be taken in defining the ensemble average because the random variable $u(\boldsymbol{x},t)$ will in general be statistically different from the random variable $u(\boldsymbol{x},t+\triangle t)$ (e.g., $\overline{u}(\boldsymbol{x},t) \neq \overline{u}(\boldsymbol{x},t+\triangle t)$). To resolve this, consider flipping a coin. If ξ is the random variable where head gives $\xi = 1$ and tail $\xi = 0$, then the arithmetic average obviously become

$$\langle \xi \rangle = \frac{1}{N} \sum_{i=1}^{N} \xi^{(i)}$$

where $\xi^{(i)}$ is the result of flip i and as $N \to \infty$ we should get $\langle \xi \rangle = 0.5$. Now, if the coins are identical then the arithmetic average is obviously the same if one person flips a coin N times or N persons flip one coin one time. This principle is used in defining an ensemble average for turbulent flows. If, instead of sampling the random variable $u(\boldsymbol{x}, t)$ at N different time intervals, we run N identical experiments and sample $u(\boldsymbol{x}, t)$ at the same time and place in all experiments, then the definition of the ensemble average will be

$$\overline{u}(\boldsymbol{x}, t) = \lim_{N \to \infty} \frac{1}{N} \sum_{i=1}^{N} u^{(i)}(\boldsymbol{x}, t).$$

For a statistically steady flow this will be identical to the time average and the left hand side will not be dependent on time.

By using the ensemble average the instantaneous velocity and pressure may be decomposed into a mean and fluctuating component

$$\begin{aligned} \boldsymbol{u}(\boldsymbol{x}, t) &= \overline{\boldsymbol{u}}(\boldsymbol{x}, t) + \boldsymbol{u}'(\boldsymbol{x}, t), \\ p(\boldsymbol{x}, t) &= \overline{p}(\boldsymbol{x}, t) + p'(\boldsymbol{x}, t). \end{aligned}$$

Note that both the instantaneous velocity $\boldsymbol{u}(\boldsymbol{x}, t)$ and the fluctuating velocity $\boldsymbol{u}'(\boldsymbol{x}, t)$ should be considered as random variables, whereas the mean velocity $\overline{\boldsymbol{u}}(\boldsymbol{x}, t)$ is deterministic. Consider any random variable decomposed as $\phi = \overline{\phi} + \phi'$. By definition we have $\overline{\phi'} = 0$. This follows since we can take the average of both sides of Eq. (6.1) leading to

$$\begin{aligned} \overline{\phi} &= \overline{\overline{\phi} + \phi'}, \\ &= \overline{\overline{\phi}} + \overline{\phi'}, \\ &= \overline{\phi} + \overline{\phi'}, \end{aligned}$$

that can only be true if $\overline{\phi'} = 0$. (Note that the average of an average is still the average.) Some other useful averaging rules are

$$\begin{aligned} \overline{\phi' \overline{\phi}} &= \overline{\phi'}\,\overline{\phi} = 0, \\ \overline{\phi \overline{\phi}} &= \overline{\phi}\,\overline{\phi}, \\ \overline{\phi \phi} &= \overline{(\overline{\phi} + \phi')(\overline{\phi} + \phi')} = \overline{\phi}^2 + \overline{\phi' \phi'}, \\ \overline{\frac{\partial \phi}{\partial s}} &= \frac{\partial \overline{\phi}}{\partial s}. \end{aligned}$$

The last rule follows since the two operations differentiation and averaging *commute*, i.e., the derivative of an average is the same as the average of a derivative.

6.2 Derivation of Reynolds averaged Navier-Stokes (RANS) equations

Having defined the average velocity and pressure we are now in a need of equations that can be used to predict them. Equations for any statistical measure can be derived directly by manipulating the Navier-Stokes equations. The equations for $\overline{\boldsymbol{u}}$ and $\overline{p}$ are found simply by averaging NS and introducing Eq. (6.1). Averaging the continuity equation leads to

$$\overline{\nabla \cdot \boldsymbol{u}} = \nabla \cdot \overline{\boldsymbol{u}} = 0$$

simply because averaging and differentiation commutes. The average of the momentum equation can be written

$$\overline{\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \boldsymbol{u} + \boldsymbol{f}}$$

The density and viscosity are assumed constant. The first term on the left and all terms on the right are averaged by using commutation

$$\begin{aligned}\overline{\frac{\partial \boldsymbol{u}}{\partial t}} &= \frac{\partial \overline{\boldsymbol{u}}}{\partial t},\\ \overline{\frac{1}{\rho}\nabla p} &= \frac{1}{\rho}\nabla \overline{p},\\ \overline{\nu\nabla^2 \boldsymbol{u}} &= \nu\nabla^2 \overline{\boldsymbol{u}}.\end{aligned}$$

This leaves the nonlinear convection term. Using index notation we can take two fast steps using continuity and commutation

$$\begin{aligned}\overline{u_j \frac{\partial u_i}{\partial x_j}} &= \overline{\frac{\partial u_i u_j}{\partial x_j}} - \cancel{\overline{u_i \frac{\partial u_j}{\partial x_j}}}\\ &= \frac{\partial \overline{u_i u_j}}{\partial x_j}.\end{aligned}$$

Inserting now for Eq. (6.1) we have

$$\overline{u_i u_j} = \overline{(\overline{u}_i + u_i')(\overline{u}_j + u_j')} = \overline{u}_i \overline{u}_j + \overline{u_i' u_j'}$$

and thus

$$\begin{aligned}\overline{u_j \frac{\partial u_i}{\partial x_j}} &= \frac{\partial \overline{u}_i \overline{u}_j + \overline{u_i' u_j'}}{\partial x_j}\\ &= \overline{u}_j \frac{\partial \overline{u}_i}{\partial x_j} + \frac{\partial \overline{u_i' u_j'}}{\partial x_j}\end{aligned}$$

Putting it all together we obtain the Reynolds averaged Navier-Stokes equations

$$\frac{\partial \overline{\boldsymbol{u}}}{\partial t} + (\overline{\boldsymbol{u}}\cdot\nabla)\overline{\boldsymbol{u}} = -\frac{1}{\rho}\nabla\overline{p} + \nu\nabla^2\overline{\boldsymbol{u}} - \frac{\partial \overline{u_i' u_j'}}{\partial x_j} + \overline{\boldsymbol{f}}, \tag{6.1}$$

$$\nabla\cdot\overline{\boldsymbol{u}} = 0 \tag{6.2}$$

6.3 Derivation of equation for average turbulent kinetic energy

The Reynolds averaged Navier-Stokes equations are derived in the previous section. To derive the turbulent kinetic energy equation it can be advantageous to first derive an equation for the fluctuating velocity $\boldsymbol{u}'$. Such an equation can be derived by subtracting Eq. (6.1) from the instantaneous momentum equation. All terms are trivial except from the convection

$$\begin{aligned}\frac{\partial \boldsymbol{u} - \overline{\boldsymbol{u}}}{\partial t} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u} - (\overline{\boldsymbol{u}}\cdot\nabla)\overline{\boldsymbol{u}} &= -\frac{1}{\rho}\nabla(p-\overline{p}) + \nu\nabla^2(\boldsymbol{u}-\overline{\boldsymbol{u}}) + \frac{\partial \overline{u_i' u_j'}}{\partial x_j} + \boldsymbol{f} - \overline{\boldsymbol{f}},\\ \frac{\partial \boldsymbol{u}'}{\partial t} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u} - (\overline{\boldsymbol{u}}\cdot\nabla)\overline{\boldsymbol{u}} &= -\frac{1}{\rho}\nabla p' + \nu\nabla^2\boldsymbol{u}' + \frac{\partial \overline{u_i' u_j'}}{\partial x_j} + \boldsymbol{f}'.\end{aligned}$$

The convection terms are manipulated by inserting for $\boldsymbol{u}$

$$\begin{aligned}(\boldsymbol{u}\cdot\nabla)\boldsymbol{u} - (\overline{\boldsymbol{u}}\cdot\nabla)\overline{\boldsymbol{u}} &= ((\overline{\boldsymbol{u}}+\boldsymbol{u}')\cdot\nabla)(\overline{\boldsymbol{u}}+\boldsymbol{u}') - (\overline{\boldsymbol{u}}\cdot\nabla)\overline{\boldsymbol{u}},\\ &= (\overline{\boldsymbol{u}}\cdot\nabla)\boldsymbol{u}' + (\boldsymbol{u}'\cdot\nabla)(\overline{\boldsymbol{u}}+\boldsymbol{u}'),\end{aligned}$$

which gives us the final equation for the fluctuating velocity component

$$\frac{\partial \boldsymbol{u}'}{\partial t} + (\overline{\boldsymbol{u}} \cdot \nabla)\boldsymbol{u}' = -\frac{1}{\rho}\nabla p' + \nu\nabla^2\boldsymbol{u}' - (\boldsymbol{u}' \cdot \nabla)(\overline{\boldsymbol{u}} + \boldsymbol{u}') + \frac{\partial \overline{u_i' u_j'}}{\partial x_j} + \boldsymbol{f}'. \tag{6.3}$$

Using index notation throughout the equation reads

$$\frac{\partial u_i'}{\partial t} + \overline{u}_j \frac{\partial u_i'}{\partial x_j} = -\frac{1}{\rho}\frac{\partial p'}{\partial x_i} + \nu\frac{\partial^2 u_i'}{\partial x_j \partial x_j} - u_j'\frac{\partial \overline{u}_i}{\partial x_j} - u_j'\frac{\partial u_i'}{\partial x_j} + \frac{\partial \overline{u_i' u_j'}}{\partial x_j} + f_i'. \tag{6.4}$$

We now want to use this equation to derive transport equations for the Reynolds stresses and the turbulent kinetic energy. The turbulent kinetic energy is $k = 0.5\overline{\boldsymbol{u}' \cdot \boldsymbol{u}'} = 0.5\overline{u_i' u_i'}$. An equation for k can be obtained by multiplying Eq. (6.4) with u_i' and then taking the average of the resulting equation. However, here we will make the derivation of the Reynolds stress $\overline{u_i' u_j'}$ first, because then we can get the equation for $\overline{u_i' u_i'}$ simply by taking the trace of the Reynolds stress equation. Start the derivation by multiplying Eq. (6.4) by u_k' and then perform an average of the entire resulting equation. For simplicity we neglect the body force f_i

$$\overline{u_k'\left(\frac{\partial u_i'}{\partial t} + \overline{u}_j \frac{\partial u_i'}{\partial x_j} = -\frac{1}{\rho}\frac{\partial p'}{\partial x_i} + \nu\frac{\partial^2 u_i'}{\partial x_j \partial x_j} - u_j'\frac{\partial \overline{u}_i}{\partial x_j} - u_j'\frac{\partial u_i'}{\partial x_j} + \frac{\partial \overline{u_i' u_j'}}{\partial x_j}\right)}.$$

Most terms are trivially manipulated, and the last term on the rhs disappears

$$\overline{u_k'\frac{\partial u_i'}{\partial t}} + \overline{u}_j\overline{u_k'\frac{\partial u_i'}{\partial x_j}} = -\frac{1}{\rho}\overline{u_k'\frac{\partial p'}{\partial x_i}} + \nu\overline{u_k'\frac{\partial^2 u_i'}{\partial x_j \partial x_j}} - \overline{u_k' u_j'}\frac{\partial \overline{u}_i}{\partial x_j} - \overline{u_k' u_j'\frac{\partial u_i'}{\partial x_j}}.$$

Since both k and i are free indices they can be interchanged into another equation

$$\overline{u_i'\frac{\partial u_k'}{\partial t}} + \overline{u}_j\overline{u_i'\frac{\partial u_k'}{\partial x_j}} = -\frac{1}{\rho}\overline{u_i'\frac{\partial p'}{\partial x_k}} + \nu\overline{u_i'\frac{\partial^2 u_k'}{\partial x_j \partial x_j}} - \overline{u_i' u_j'}\frac{\partial \overline{u}_k}{\partial x_j} - \overline{u_i' u_j'\frac{\partial u_k'}{\partial x_j}}.$$

Noting that, e.g., $\partial u_i' u_k'/\partial t = u_i'\partial u_k'/\partial t + u_k'\partial u_i'/\partial t$, you should now be able to realize that the sum of the two previous equations will give us an equation for $\overline{u_i' u_k'}$

$$\begin{aligned}
\frac{\partial \overline{u_i' u_k'}}{\partial t} + \overline{u}_j\frac{\partial \overline{u_i' u_k'}}{\partial x_j} = &-\frac{1}{\rho}\left(\overline{u_i'\frac{\partial p'}{\partial x_k}} + \overline{u_k'\frac{\partial p'}{\partial x_i}}\right) \\
&+ \nu\left(\overline{u_i'\frac{\partial^2 u_k'}{\partial x_j \partial x_j}} + \overline{u_k'\frac{\partial^2 u_i'}{\partial x_j \partial x_j}}\right) \\
&- \overline{u_i' u_j'}\frac{\partial \overline{u}_k}{\partial x_j} - \overline{u_k' u_j'}\frac{\partial \overline{u}_i}{\partial x_j} \\
&- \overline{u_i' u_j'\frac{\partial u_k'}{\partial x_j}} - \overline{u_k' u_j'\frac{\partial u_i'}{\partial x_j}}.
\end{aligned}$$

Several terms on the rhs can be further simplified. Using the product rule the pressure terms can alternatively be written as

$$\begin{aligned}
\overline{u_i'\frac{\partial p'}{\partial x_k}} + \overline{u_k'\frac{\partial p'}{\partial x_i}} &= \overline{p'\left(\frac{\partial u_i'}{\partial x_k} + \frac{\partial u_k'}{\partial x_i}\right)} + \frac{\partial}{\partial x_j}\left(\overline{p' u_i'}\delta_{kj} + \overline{p' u_k'}\delta_{ij}\right), \\
&= 2\overline{p' s_{ik}'} + \frac{\partial}{\partial x_j}\left(\overline{p' u_i'}\delta_{kj} + \overline{p' u_k'}\delta_{ij}\right),
\end{aligned}$$

where

$$s_{ij}' = \frac{1}{2}\left(\frac{\partial u_i'}{\partial x_j} + \frac{\partial u_j'}{\partial x_i}\right)$$

and the notation with the identity tensor is used to enable a divergence form ($\partial\overline{p'u_i'}/\partial x_k = \partial\overline{p'u_i'}/\partial x_j\delta_{kj}$). This notation is used to emphasize that the term is conservative, it can neither create nor destroy $\overline{u_i'u_k'}$, but simply serves to move it around. For this reason the term is often referred to as pressure diffusion.

The two Laplacian terms in Eq. (6.5) can be simplified using the following identities

$$u_i'\frac{\partial^2 u_k'}{\partial x_j\partial x_j} = \frac{\partial}{\partial x_j}\left(\frac{\partial u_i'u_k'}{\partial x_j} - u_k'\frac{\partial u_i'}{\partial x_j}\right) - \frac{\partial u_i'}{\partial x_j}\frac{\partial u_k'}{\partial x_j}$$

and

$$u_k'\frac{\partial^2 u_i'}{\partial x_j\partial x_j} = \frac{\partial}{\partial x_j}\left(u_k'\frac{\partial u_i'}{\partial x_j}\right) - \frac{\partial u_i'}{\partial x_j}\frac{\partial u_k'}{\partial x_j}.$$

Summing up these two equations we get

$$u_i'\frac{\partial^2 u_k'}{\partial x_j\partial x_j} + u_k'\frac{\partial^2 u_i'}{\partial x_j\partial x_j} = \frac{\partial^2 u_i'u_k'}{\partial x_j\partial x_j} - 2\frac{\partial u_i'}{\partial x_j}\frac{\partial u_k'}{\partial x_j}.$$

Finally, note that due to continuity

$$u_i'u_j'\frac{\partial u_k'}{\partial x_j} + u_k'u_j'\frac{\partial u_i'}{\partial x_j} = \frac{\partial u_i'u_k'u_j'}{\partial x_j}.$$

Inserting for all the simplifications above we obtain a final form for the Reynolds stress transport equation

$$\begin{aligned}\frac{\partial\overline{u_i'u_k'}}{\partial t} + \overline{u}_j\frac{\partial\overline{u_i'u_k'}}{\partial x_j} =& -\frac{2\overline{p's_{ik}'}}{\rho} - \frac{\partial}{\partial x_j}\left(\frac{1}{\rho}(\overline{p'u_i'}\delta_{kj} + \overline{p'u_k'}\delta_{ij}) + \overline{u_i'u_k'u_j'} - \nu\frac{\partial\overline{u_i'u_k'}}{\partial x_j}\right)\\ &- 2\nu\overline{\frac{\partial u_i'}{\partial x_j}\frac{\partial u_k'}{\partial x_j}} - \overline{u_i'u_j'}\frac{\partial\overline{u}_k}{\partial x_j} - \overline{u_k'u_j'}\frac{\partial\overline{u}_i}{\partial x_j}.\end{aligned}$$

This is the same as Eq.~(6-18) in [Whi06]. Note that on the right hand side only the last two terms on the bottom line and the last on the first line are closed since they only contain $\overline{u_i'u_k'}$ and gradients of the mean velocity. This means that in our ambitious effort to find a closure for $\overline{u_i'u_k'}$ from first principles we have ended up creating even more problems for our self.

An equation for the turbulent kinetic energy k can be obtained by contracting the two free indices in Eq. (6.5) (set index $k = i$). A few terms disappear due to continuity ($s_{ii}' = 0$) and symmetry leading to

$$\begin{aligned}\frac{\partial\overline{u_i'u_i'}}{\partial t} + \overline{u}_j\frac{\partial\overline{u_i'u_i'}}{\partial x_j} =& -\frac{\partial}{\partial x_j}\left(\frac{2}{\rho}\overline{p'u_i'}\delta_{ij} + \overline{u_i'u_i'u_j'} - \nu\frac{\partial\overline{u_i'u_i'}}{\partial x_j}\right)\\ &- 2\nu\overline{\frac{\partial u_i'}{\partial x_j}\frac{\partial u_i'}{\partial x_j}} - 2\overline{u_i'u_j'}\frac{\partial\overline{u}_i}{\partial x_j}.\end{aligned}$$

If we simplify this further using $k = 0.5\overline{u_i'u_i'}$ we obtain

$$\begin{aligned}\frac{\partial k}{\partial t} + \overline{u}_j\frac{\partial k}{\partial x_j} =& -\frac{\partial}{\partial x_j}\left(\frac{1}{\rho}\overline{p'u_i'}\delta_{ij} + \frac{1}{2}\overline{u_i'u_i'u_j'} - \nu\frac{\partial k}{\partial x_j}\right)\\ &- \nu\overline{\frac{\partial u_i'}{\partial x_j}\frac{\partial u_i'}{\partial x_j}} - \overline{u_i'u_j'}\frac{\partial\overline{u}_i}{\partial x_j}.\end{aligned}$$

Evidently this equation differs in a few places from Eq.~(6-17) in \cite{white06}. However, the difference is merely due to a few rearrangements and we will now derive also a slightly different form more similar to (6-17). Note that the Laplacian term, i.e., the last of the divergence terms on the rhs of Eq. (6.5), can alternatively be rewritten using the identity

$$\nu\frac{\partial^2 u_i'u_i'}{\partial x_j\partial x_j} = 4\nu\frac{\partial s_{ij}'u_i'}{\partial x_j} - 2\nu\frac{\partial u_i'}{\partial x_j}\frac{\partial u_j'}{\partial x_i}.$$

Inserting this into Eq. (6.5) leads to

$$\frac{\partial k}{\partial t} + \overline{u}_j \frac{\partial k}{\partial x_j} = -\frac{\partial}{\partial x_j}\left(\frac{1}{\rho}\overline{p' u_i'}\delta_{ij} + \frac{1}{2}\overline{u_i' u_i' u_j'} - 2\nu \overline{s_{ij}' u_i'}\right) - 2\nu \overline{\frac{\partial u_i'}{\partial x_j} s_{ij}'} - \overline{u_i' u_j'}\frac{\partial \overline{u}_i}{\partial x_j}.$$

Furthermore, the velocity deformation tensor can be decomposed as

$$\frac{\partial u_i'}{\partial x_j} = s_{ij}' + \omega_{ij}',$$

where the anti-symmetric rotation rate tensor ω_{ij}' is given as

$$\omega_{ij}' = \frac{1}{2}\left(\frac{\partial u_i'}{\partial x_j} - \frac{\partial u_j'}{\partial x_i}\right).$$

Since the contraction of a symmetric tensor with an anti-symmetric is identically zero it follows that

$$\begin{aligned} s_{ij}' \frac{\partial u_i'}{\partial x_j} &= s_{ij}' s_{ij}' + s_{ij}' \omega_{ij}' \\ &= s_{ij}' s_{ij}'. \end{aligned}$$

The final form of the turbulent kinetic energy equation, which is also the one that is most often used in the literature, is thus

$$\frac{\partial k}{\partial t} + \overline{u}_j \frac{\partial k}{\partial x_j} = -\frac{\partial}{\partial x_j}\left(\frac{1}{\rho}\overline{p' u_i'}\delta_{ij} + \frac{1}{2}\overline{u_i' u_i' u_j'} - 2\nu \overline{s_{ij}' u_i'}\right) - 2\nu \overline{s_{ij}' s_{ij}'} - \overline{u_i' u_j'}\frac{\partial \overline{u}_i}{\partial x_j}.$$

All terms in the turbulent kinetic energy equation have their own physical interpretation

- Rate of change of k due to non-stationary statistics

$$\frac{\partial k}{\partial t}$$

- Rate of change of k due to convection

$$\overline{u}_j \frac{\partial k}{\partial x_j}$$

- Conservative transport of kinetic energy in an inhomogeneous field due respectively to the pressure fluctuations, the turbulence itself, and the viscous stresses:

$$\frac{\partial}{\partial x_j}\left(\frac{1}{\rho}\overline{p' u_i'}\delta_{ij} + \frac{1}{2}\overline{u_i' u_i' u_j'} - 2\nu \overline{s_{ij}' u_i'}\right)$$

- Rate of production of turbulence kinetic energy from the mean flow (gradient):

$$\overline{u_i' u_j'}\frac{\partial \overline{u}_i}{\partial x_j}$$

- Rate of dissipation of turbulence kinetic energy per unit mass due to viscous stresses:

$$\varepsilon = 2\nu \overline{s_{ij}' s_{ij}'}$$

6.4 Turbulence modelling

Turbulence modelling is all about finding suitable closures, or models, for the Reynolds stress tensor. The Reynolds stress tensor is symmetrical

$$\overline{u_i' u_j'} = \overline{u_j' u_i'}$$

and thus contains 6 unknown quantities. The Reynolds stress is not really a stress, but it is so called because of the way it appears in the mean momentum equation (6.1). The mean momentum equation can alternatively be rewritten as

$$\rho\left(\frac{\partial \overline{\boldsymbol{u}}}{\partial t} + (\overline{\boldsymbol{u}}\cdot\nabla)\overline{\boldsymbol{u}}\right) = \nabla\cdot\overline{\tau} + \rho\overline{\boldsymbol{f}}, \tag{6.5}$$

where

$$\overline{\tau}_{ij} = -\overline{p}\delta_{ij} + 2\mu\overline{S}_{ij} - \rho\overline{u_i' u_j'},$$

is a new term that has dimensions of stress and

$$\overline{S}_{ij} = \frac{1}{2}\left(\frac{\partial \overline{u}_i}{\partial x_j} + \frac{\partial \overline{u}_j}{\partial x_i}\right).$$

Mathematically then it appears that the total mean stress tensor has been added a third term $-\rho\overline{u_i' u_j'}$ after the regular pressure and viscous stresses. The term is called a Reynolds "stress", but it is in fact just the contribution of the fluctuating velocities to the mean nonlinear convection. (Note that ρ is required for the term to have dimensions of stress.) By stress we usually mean the internal forces that fluid particles exert on each other.

The highest level of RANS modelling is usually considered the second moment closures that solve transport equations for $\overline{u_i' u_j'}$ directly. Second moment closures are outside the scope of this course.

The by far most commonly used turbulence model is the eddy viscosity (first moment) closure

$$\overline{u_i' u_j'} = -2\nu_T\overline{S}_{ij} + \frac{2}{3}k\delta_{ij}, \tag{6.6}$$

where ν_T is a 'turbulent viscosity'. Turbulence models are usually classified by the number of additional PDE's that are used to model the turbulent viscosity.

The turbulent viscosity is a parameter of the flow and not the fluid. As mentioned before, one of the most important properties of a turbulent flow is its ability to efficiently mix momentum and scalars, like temperature. Mathematically, the enhanced mixing is manifested through the Reynolds stress and it is through Eq. (6.6) sought modelled as a diffusion process using ν_T as a positive (≥ 0) mixing rate constant. Inserted into the momentum equation the eddy viscosity gives rise to the term

$$\frac{\partial \overline{u_i' u_j'}}{\partial x_j} = -\frac{\partial}{\partial x_j}\left[\nu_t\left(\frac{\partial \overline{u}_i}{\partial x_j} + \frac{\partial \overline{u}_j}{\partial x_i}\right)\right] + \frac{2}{3}\frac{\partial k}{\partial x_i},$$

and it can be seen that the first term on the right hand side will lead to enhanced mixing (or diffusion) of $\overline{u}_i$ as long as ν_T is positive.

Having established an algebraic model for the Reynolds stress (6.6), we have reduced the number of unknowns from 6 to 2 (ν_T and k). We will now discuss the most common strategies for closing the remaining terms.

The turbulent viscosity is by dimensional reasoning the product of a velocityscale $\tilde{u}$ and a lengthscale l

$$\nu_T = \tilde{u}\cdot l,$$

where different interpretations are possible for defining such scales. Remembering that one of the 'defining' properties of turbulence is that it contains a range of scales, such a classification into one single velocity- and lengthscale may

seem questionable from the very start. Nevertheless, most turbulence models boil down to finding good approximations for these scales, and the modelled scales are then usually interpreted as the largest scales of turbulence. Many different models exist (including multiscale models, which is beside the scope for this course) and sometimes a timescale t_k is used instead of $\tilde{u}$, but then the two are related through $\tilde{u} = l/t_k$.

Turbulence models are usually classified by the number of additional PDEs that are required to establish the proper scales and we will now go through some of the most basic models.

6.4.1 Zero-equation models

Zero-equation turbulence models are models that use zero additional PDEs to model the turbulent viscosity. The Mixing length model was the first such turbulence model. For a simple plane shear flow where $\overline{u}$ is the tangential velocity and y is the wall normal direction the model reads

$$\tilde{u} = l \left| \frac{\mathrm{d}\overline{u}}{\mathrm{d}y} \right|,$$
$$\nu_T = l^2 \left| \frac{\mathrm{d}\overline{u}}{\mathrm{d}y} \right|.$$

For a plane shear flow the model is excellent as one may choose

$$l \approx y^2, \quad \text{for} \quad y^+ < 5,$$
$$l \approx \kappa y, \quad \text{for} \quad 30 < y^+ < 100,$$
$$l \approx 1, \quad \text{for} \quad y^+ > 100$$

where $\kappa = 0.41$ is von Kármán's constant and y is the distance to the wall. Further

$$v^* = \sqrt{\nu \frac{\partial \overline{u}}{\partial y}}_{\text{wall}},$$
$$y^+ = \frac{y v^*}{\nu},$$

are, respectively, the turbulent wall friction velocity and the distance to the wall in normalized 'wall' units. A Reynolds number based on the friction velocity, $Re_\tau = L \cdot v^*/\nu$, is often used to characterize flow in turbulent plane shear flows.

Several models exist that merge the three domains in Eq. (6.7) into one single continuous function. Van Driest provided a model that merged the first two inner layers using

$$l = \kappa y \left(1 - \exp\left(-\frac{y^+}{A} \right) \right), \tag{6.7}$$

where A is a constant set to 26 for flat plate flow. This model should be merged with $l = \text{constant}$ far enough from the wall. Far enough is often chosen when l from Eq. (6.7) becomes larger than a factor of the mixing layer thickness δ. An often used estimate is when $\kappa y = 0.09\delta$ or $y = 0.22\delta$.

The Mixing length model is not a good model for other flows than plane boundary layer flows. For example, in the center of a plane channel the mean velocity gradient is zero due to symmetry ($\mathrm{d}\overline{u}/\mathrm{d}y = 0$). The Mixing length model Eq. (6.7) thus predicts that the turbulent viscosity should be zero in the center of the channel. This is contrary to the experimental observation and intuition, because the turbulent intensity is actually very large in the center.

This flaws of the mixing length model first led to the suggested improvement that the velocityscale should be proportional to the turbulence intensity and not the mean velocity gradient

$$\tilde{u} = \text{const} \cdot \sqrt{k}. \tag{6.8}$$

The problem now is that this model introduces a new unknown k that requires modelling (whereas the old model used known quantities l and $\overline{u}$). Thus one additional PDE is required for closure of the turbulent viscosity.

6.4.2 One-equation models

Most one-equation turbulence models solves a transport equation for k, the turbulent kinetic energy. The equation for k has already been derived in previous assignments and it is evident that it contains even more unclosed terms. If we start by rewriting the k-equation as

$$\frac{\partial k}{\partial t} + \overline{u}_j \frac{\partial k}{\partial x_j} = -\frac{\partial T_j}{\partial x_j} + P - \tilde{\varepsilon}$$

then the terms on the right hand side are

$$\begin{aligned} T_j &= \frac{1}{\rho}\overline{p' u_i'}\delta_{ij} + \frac{1}{2}\overline{u_i' u_i' u_j'} - \nu\frac{\partial k}{\partial x_j}, \\ P &= -\overline{u_i' u_j'}\frac{\partial \overline{u}_i}{\partial x_j}, \\ \tilde{\varepsilon} &= \nu\overline{\frac{\partial u_i'}{\partial x_j}\frac{\partial u_i'}{\partial x_j}}. \end{aligned} \tag{6.9}$$

Here T_j represents turbulent transport, P is production of turbulent kinetic energy and $\tilde{\varepsilon}$ is 'pseudo'-dissipation of turbulent kinetic energy. The production P is closed since it only contains known quantities like the Reynolds stress and mean velocity gradients. The first two terms of T_j and $\tilde{\varepsilon}$ are unknown and require closure. $\tilde{\varepsilon}$ is called 'pseudo'-dissipation since the actual dissipation of k is really $2\nu\overline{s_{ij}' s_{ij}'}$. The two terms are related through the identity

$$\begin{aligned} 2\nu\overline{s_{ij}' s_{ij}'} &= \nu\overline{\frac{\partial u_i'}{\partial x_j}\frac{\partial u_i'}{\partial x_j}} + \nu\overline{\frac{\partial u_i'}{\partial x_j}\frac{\partial u_j'}{\partial x_i}}, \\ \varepsilon &= \tilde{\varepsilon} + \nu\overline{\frac{\partial u_i'}{\partial x_j}\frac{\partial u_j'}{\partial x_i}}. \end{aligned}$$

Since the last term on the rhs is small relative to the first, dissipation and 'pseudo'-dissipation are similar for most flows.

The turbulent transport term is usually modelled as gradient diffusion

$$T_j = -\left(\nu + \frac{\nu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_j},$$

where σ_k is a constant in the near vicinity of one. Like the name suggests the model enhances diffusion of k and serves to smooth out the k field faster than possible simply through regular diffusion. This corresponds well with our knowledge of turbulence as an efficient mixing process.

The 'pseudo'-dissipation rate scales like

$$\tilde{\varepsilon} \approx \frac{\tilde{u}^3}{l}.$$

Thus we can find a suitable model for $\tilde{\varepsilon}$ simply by inserting for Eq. (6.8)

$$\begin{aligned} \tilde{\varepsilon} &= \frac{(\text{const}\cdot\sqrt{k})^3}{l}, \\ \tilde{\varepsilon} &= \text{const}\cdot\frac{k^{3/2}}{l}, \end{aligned}$$

and thus

$$l = \text{const}\cdot\frac{k^{3/2}}{\tilde{\varepsilon}}.$$

By inserting for the velocity- and lengthscales we get the following model for the turbulent viscosity

$$\nu_T = \text{const} \cdot \sqrt{k} l,$$
$$\nu_T = \text{const} \cdot \frac{k^2}{\tilde{\varepsilon}}.$$

A weakness of the model is that it requires the lengthscale l to be provided from knowledge of the problem under consideration. For this reason one-equation turbulence models are often referred to as being incomplete. One-equation models are popular for systems where the mixing length can be easily specified, like a plane boundary layer or an aeroplane wing.

6.4.3 Two-equation models

Two-equation turbulence models introduce an additional PDE for the missing lengthscale. Since two-equation models do not require any parameter (like l) from previous knowledge of the system, the two-equation models are often referred to as being complete.

The most popular two-equation turbulence models are variations of the $k-\tilde{\varepsilon}$ model and the $k-\omega$ model. $k-\tilde{\varepsilon}$ models solve one PDE for k and an additional for $\tilde{\varepsilon}$. Likewise, $k-\omega$ models solve PDEs for k and ω, where $\omega \approx \tilde{\varepsilon}/k$. If k and $\tilde{\varepsilon}$ are known then

$$l = \frac{k^{3/2}}{\tilde{\varepsilon}}, \quad t_k = \frac{k}{\tilde{\varepsilon}}, \quad \tilde{u} = \frac{l}{t_k} = \sqrt{k}$$

where t_k is a turbulent timescale. The turbulent viscosity is as before determined by

$$\nu_T = \text{const} \cdot \tilde{u} \cdot l = \text{const} \cdot \frac{k^2}{\tilde{\varepsilon}}.$$

A transport equation for $\tilde{\varepsilon}$ can be derived exactly from the Navier-Stokes equations. The resulting equation contains many new unknowns and here we will not go into detail. Instead we will here simply report the entire and closed $k-\tilde{\varepsilon}$ model with its most common model 'constants'

$$\begin{aligned}
\frac{\partial \overline{u}_i}{\partial t} + \overline{u}_j \frac{\partial \overline{u}_i}{\partial x_j} &= \frac{\partial}{\partial x_j}\left[(\nu + \nu_T)\left(\frac{\partial \overline{u}_i}{\partial x_j} + \frac{\partial \overline{u}_j}{\partial x_i}\right)\right] - \frac{1}{\rho}\frac{\partial \overline{p} + \frac{2}{3}k}{\partial x_i}, \\
\frac{\partial k}{\partial t} + \overline{u}_j \frac{\partial k}{\partial x_j} &= \frac{\partial}{\partial x_j}\left[\left(\nu + \frac{\nu_T}{\sigma_k}\right)\frac{\partial k}{\partial x_j}\right] + P - \tilde{\varepsilon}, \\
\frac{\partial \tilde{\varepsilon}}{\partial t} + \overline{u}_j \frac{\partial \tilde{\varepsilon}}{\partial x_j} &= \frac{\partial}{\partial x_j}\left[\left(\nu + \frac{\nu_T}{\sigma_\varepsilon}\right)\frac{\partial \tilde{\varepsilon}}{\partial x_j}\right] + C_{\tilde{\varepsilon}_1}\frac{P\tilde{\varepsilon}}{k} - C_{\tilde{\varepsilon}_2}\frac{\tilde{\varepsilon}^2}{k}, \\
\frac{\partial \overline{u}_i}{\partial x_i} &= 0, \\
\nu_T &= C_\mu \frac{k^2}{\tilde{\varepsilon}}, \\
C_\mu = 0.09, \quad C_{\tilde{\varepsilon}_1} = 1.44, \quad C_{\tilde{\varepsilon}_2} &= 1.92, \quad \sigma_{\tilde{\varepsilon}} = 1.3, \quad \sigma_k = 1.
\end{aligned}$$

Unfortunately the model 'constants' are known to vary from flow to flow and thus often in need of tuning for optimal performance. Nevertheless, two-equation models are complete and the $k-\tilde{\varepsilon}$ model is very much in use in industry for a wide range of flows.

There is a great number of different RANS turbulence models. Some are listed on the page for NASA's Turbulence Modelling Resources and the wiki for turbulence modeling.

6.4.4 Boundary conditions

The average statistics in a turbulent flow have sharp gradients in the near vicinity of a wall. To properly resolve all statistics the location of the first inner computational node needs to be at approximately $y^+ = 1$. This restriction is quite demanding in terms of grid resolution, especially at high Reynolds number. The standard $k - \tilde{\varepsilon}$ model, Eq. (6.10), is not very accurate close to walls and many modifications exist to handle this problem. Two main approaches are

1. Do not resolve the flow further than approximately $y^+ = 30$ and use wall functions to prescribe boundary conditions inside the wall instead of at the wall.
2. Resolve the flow completely (all the way up to $y^+ = 1$) and introduce damping functions.

The wall functions are based on the log-law

$$u^+ = \frac{1}{\kappa} \ln y^+ + B,$$

which is known to be in close agreement with experiments of regular boundary layers for a range of Reynolds numbers. If the first inner node (the first node on the inside of the wall) is located at y^p, then the velocity can be computed as

$$\frac{\overline{u}^p}{u^*} = \frac{1}{\kappa} \ln \frac{y^p u^*}{\nu} + B,$$

CHAPTER

SEVEN

SUGGESTED SOLUTIONS

7.1 Chapter 3

7.1.1 Suggested solutions Couette

Solutions written assignments Couette

Problem 3-1 in White

Solve for a non-Newtonian Couette flow, where the stress tensor is computed as

$$\tau = K\left(\frac{\mathrm{d}u}{\mathrm{d}y}\right)^n,$$

where n is an integer different from 1. The momentum equation for any stress tensor in a Couette flow is

$$0 = \nabla \cdot \tau.$$

For one-dimensional Couette flow aligned with the x-direction and homogeneous in the z-direction, the equation reads

$$0 = \frac{\mathrm{d}\tau}{\mathrm{d}y},$$

which means that τ is a constant with respect to y and thus

$$K\left(\frac{\mathrm{d}u}{\mathrm{d}y}\right)^n = K_2,$$

where K_2 is a new constant. In other words

$$\frac{\mathrm{d}u}{\mathrm{d}y} = K_3 \left(= (K_2/K)^{1/n}\right),$$

and thus

$$u = K_3 y + K_4.$$

The solution is the same as for Newtonian flows regardless of n.

7.2 Chapter 4

CHAPTER

EIGHT

FENICS

FEniCS is a computing platform for solving partial differential equations (PDEs). FEniCS enables users to quickly translate scientific models into efficient finite element code. With the high-level Python and C++ interfaces to FEniCS, it is easy to get started, but FEniCS offers also powerful capabilities for more experienced programmers. FEniCS runs on a multitude of platforms ranging from laptops to high-performance clusters.

There is an excellent online tutorial that is recommended for getting started.

8.1 Numerical solution of nonlinear equations

Several derived equations in Similarity solutions (e.g., Eq. (3.42)) are nonlinear and no analytical solutions are known. Very few techniques are actually available for solving nonlinear equations analytically and as such we usually have to rely on numerical methods to obtain solutions for real fluid flows. Reynolds Averaged Navier-Stokes equations, that will be discussed thoroughly in chapter 6 of White [Whi06], are basically Navier-Stokes equations with a nonlinear viscosity coefficient. In this section we will consider two of the most important techniques for iteratively solving nonlinear equations: Picard and Newton iterations.

The general procedure for obtaining solutions to nonlinear equations is to guess an initial solution and then successively recompute new and hopefully better approximations to the solution. This is illustrated nicely with Newton's method (also called Newton-Raphson's method). Consider a nonlinear function $f(x)$ of one variable x (e.g., $f(x) = x^2 - 1$ or $f(x) = x\sin(x) - 1$), where one is interested in finding the roots x such that $f(x) = 0$. Newton's method boils down to making an initial guess x_0 that does not satisfy our equation (i.e., $f(x_0) \neq 0$) and from this initial guess we successively compute better approximations to the final root:

$$
\begin{aligned}
&\text{Guess } x_0, \quad n = 0 \\
&\text{while not } f(x_n) \approx 0 \text{ do} \\
&\qquad x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \\
&\qquad n = n + 1
\end{aligned}
$$

That is, compute x_1 from x_0 and check if the solution is close enough to the root. If not, compute x_2 using x_1 as initial condition. Repeat until $f(x_n) \approx 0$.

When PDEs are solved numerically we obtain through discretization a system of many equations for many variables. Newton's method extends easily to systems of equations as well. Consider 2 equations with two unknowns written compactly as $F(\overline{x})$

$$
F(\overline{x}) = \begin{cases} x^2 + 2xy & = 0, \\ x + 4y^2 & = 0, \end{cases}
$$

where $\overline{x} = (x, y)^T$. The vector F has two components (the two equations) and there are two unknowns. We can compute the derivative of F with respect to $\overline{x}$

$$J_{ij} = \frac{\partial F_i}{\partial x_j} = \begin{pmatrix} 2x + 2y & 2x \\ 1 & 8y \end{pmatrix}.$$

The derivative is often called the Jacobian. Newton's method for these two equations (or any system of equations) is simply

$$J(\overline{x}^k)(\overline{x}^{k+1} - \overline{x}^k) = -F(\overline{x}^k),$$

or with index notation for equation i

$$J^k_{ij}(x^{k+1}_j - x^k_j) = -F^k_i,$$

where the k index signals that both J and F are computed using the solution at iteration step k. There is no summation on repeated k's.

A discrete finite element solution of the function u can be written as

$$u(x) = \sum_{i=0}^{N} u_i v_i(x),$$

where v_i are the testfunctions. The solution has $N+1$ unknowns u_i, or "degrees of freedom", that for a linear Lagrange element are the values at the vertices of the mesh. When a PDE containing both trial- and testfunctions is assembled in FEniCS a set of $N + 1$ linear equations arise for these unknowns. We will now see how nonlinear equations can be handled by FEniCS.

8.1.1 Nonlinear Poisson equation

Consider the nonlinear Poisson equation

$$-\nabla \cdot (q(u)\nabla u) = f(u),$$

where q and f may be nonlinear functions of u. Using testfunction v and neglecting the boundary term the variational form reads

$$\int_\Omega q(u)\nabla u \cdot \nabla v \mathrm{d}x = \int_\Omega f(u) v \mathrm{d}x. \tag{8.1}$$

This is a nonlinear variational form and we need to solve it iteratively.

Newton's method

Consider first Newton's method for solving the nonlinear Poisson equation. We use the notation u^k for the known solution at iteration step k and u the unknown trialfunction. For Newton's method we write the variational form solely in terms of known functions

$$F(u^k; v) = \int_\Omega q(u^k)\nabla u^k \cdot \nabla v \mathrm{d}x - \int_\Omega f(u^k) v \mathrm{d}x.$$

$F(u^k; v)$ is a linear form in that it does not contain the unknown u, only the testfunction v. It is a *nonlinear* form in terms of the known `Function` u^k though. The Jacobian of $F(u^k; v)$ is computed is FEniCS as

```
u_ = Function(V)
u = TrialFunction(V)
J = derivative(F, u_, u)
```

A complete implementation that solves the nonlinear Poisson equation on the unit interval x=[0, 1] is shown below, where the use of this `J` should be obvious

```
from dolfin import *

mesh = UnitIntervalMesh(10)
V = FunctionSpace(mesh, 'CG', 1)
u = TrialFunction(V)
v = TestFunction(V)

bc0 = DirichletBC(V, Constant(0), "std::abs(x[0]) < 1e-10")
bc1 = DirichletBC(V, Constant(1), "std::abs(x[0]-1) < 1e-10")

def q(u):
    return 1+u**4

u_ = interpolate(Expression("x[0]", degree=1), V)
F = inner(grad(v), q(u_)*grad(u_))*dx
##solve(F == 0, u_, [bc0, bc1])

J = derivative(F, u_, u)
du = Function(V)
error = 1; k = 0
while k < 100 and error > 1e-12:
    A = assemble(J)
    b = assemble(-F)
    [bc.apply(A, b, u_.vector()) for bc in [bc0, bc1]]
    solve(A, du.vector(), b)
    u_.vector().axpy(1., du.vector())
    error = norm(du.vector())
    k += 1
    print("Iteration {} Error = {}".format(k, error))
```

```
Iteration 1 Error = 0.20207598526678652
Iteration 2 Error = 0.006786044472018965
Iteration 3 Error = 1.626130021884984e-05
Iteration 4 Error = 1.1905550866645789e-10
Iteration 5 Error = 8.698631493053902e-17
```

Note that everything below the line with `solve(F == 0, u_, [bc0, bc1])` actually can be replaced simply by using this very compact call. The part from `J = derivative(F, u_, u)` and out is included here simply to illustrate how Newton's method works in practise.

Picard iterations

For Picard iterations we have to linearize the variational form in Eq. (8.1) ourself. Let us first linearize such that the coefficient is known:

$$F(u;v) = \int_\Omega q(u^k)\nabla u \cdot \nabla v \mathrm{d}x - \int_\Omega f(u^k, u) v \mathrm{d}x$$

The first term on the rhs of $F(u;v)$ is bilinear in that it contains both trial- and testfunctions u and v. The equation is also linear in u, which is necessary for FEniCS to accept it. Note that the function f could be linear in u as well. For example, if $f(u) = u^2$, then we can linearize it like $f(u) = u \cdot u^k$. Picard iterations are implemented as Newton up to the point where the form is created. The remaining part of the Picard code reads:

```
u_ = interpolate(Expression("x[0]", degree=1), V)
F = inner(grad(v), q(u_)*grad(u))*dx
k = 0
u_1 = Function(V)
error = 1
while k < 100 and error > 1e-12:
    solve(F == Constant(0)*v*dx, u_, [bc0, bc1])
    error = errornorm(u_, u_1)
    u_1.assign(u_)
    k += 1
    print("Iteration {} Error = {}".format(k,  error))
```

```
Iteration 1 Error = 0.6227202551171986
Iteration 2 Error = 0.009279110924024897
Iteration 3 Error = 0.0009571365383617912
Iteration 4 Error = 7.170672306756823e-05
Iteration 5 Error = 1.6196475252925717e-05
Iteration 6 Error = 1.6051762319950847e-06
Iteration 7 Error = 1.5867106937166712e-07
Iteration 8 Error = 3.2011735458409474e-08
Iteration 9 Error = 2.6184233711820267e-09
Iteration 10 Error = 3.617146938878996e-10
Iteration 11 Error = 6.066104630273271e-11
Iteration 12 Error = 4.281972850229132e-12
Iteration 13 Error = 8.043617961021526e-13
```

The `Function u_1` holds the previous solution and `u_1.assign(u_)` copies all values from the newly computed `u_` to `u_1`.

Note that the Newton's method requires 5 iterations to converge, whereas Picard requires 13. This is quite typical. Newton's method is known to be very efficient when it actually finds the solution, but it often diverges if the initial guess is not close enough to the final solution. Picard is known to approach the solution more slowly, but in return it is usually more robust in that it finds the solution from a broader range of initial conditions.

8.2 Transient problems

Transient problems are usually solved in FEniCS using a finite difference approximation of the time derivative. The time dimension can be discretized using constant discrete time intervals of length $\triangle t$, and we look for solutions at the discrete times $t = [0, \triangle t, 2\triangle t, ..., T - \triangle t, T] = k\triangle t$, for $k = 0, 1, 2, ..., N-1, N$, $\triangle t = T/N$. The solutions at the $N+1$ different timesteps are similarly written as u^k. Using finite differences for the time derivative, a variational form of the heat equation reads

$$\int_\Omega \frac{u^k - u^{k-1}}{\triangle t} v \mathrm{d}x = -\nu \int_\Omega \nabla u^{k-\frac{1}{2}} \cdot \nabla v \mathrm{d}x,$$

where the right hand side is computed at the midpoint between timesteps k and $k-1$ using notation $u^{k-\frac{1}{2}} = (u^k + u^{k-1})/2$. Note that when the solution is computed, we start at the initial condition at $k = 0$, where the initial condition u^0 is known and u^1 is unknown. When u^1 is subsequently computed and known, we are ready to move on to the next solution u^2 and so on. In other words, u^k is always the unknown we are trying to compute and $u^{k-1}, u^{k-2}, ...$ are all considered to be known. In FEniCS the unknown u^k is represented in `Forms` as a `TrialFunction`, whereas all knowns are represented as `Functions`. A variational form for the heat equation in FEniCS may look like:

```
u = TrialFunction(V)
v = TestFunction(V)
u_ = Function(V)     # Known solution at k
u_1 = Function(V)    # Known solution at k-1
dt = Constant(0.1)
nu = 0.01
U = 0.5*(u+u_1)
F = inner(u - u_1, v)*dx + dt*nu*inner(grad(U), grad(v))*dx
```

The solution of the form must be placed inside a loop, advancing the solution forward in time, something like:

```
t = 0
while t < T:
    t += dt
    solve(lhs(F) == rhs(F), u_, bcs)
    # Advance solution to next timestep:
    u_1.vector()[:] = u_.vector()
```

The python functions `lhs` and `rhs` are used to extract bilinear (terms containing both trial- and testfunctions) and linear (terms containing only testfunction and no trialfunction) forms respectively. The last line of code copies all the values from `u_` to `u_1`. Note that u^k is the unknown we look for at timestep k. In the variational form u^k is represented as an unknown `TrialFunction`. However, when the variational form has been solved, the known solution can be found in the `Function` `u_` and we are then finished with timestep k. When we now move on to the next timestep, the solution we just found becomes the solution at the previous timestep, i.e., at $k-1$. This is why we copy all values from `u_` to `u_1` as our final task in the time loop.

BIBLIOGRAPHY

[Whi06] Frank M. White. *Viscous Fluid Flow*. McGraw-Hill, third edition, 2006.

[Kre99] Erwin Kreyszig. *Advanced Engineering mathematics*. Wiley, 8 edition, 1999.

[Whi06] Frank M. White. *Viscous Fluid Flow*. McGraw-Hill, third edition, 2006.

[Whi06] Frank M. White. *Viscous Fluid Flow*. McGraw-Hill, third edition, 2006.

[Whi06] Frank M. White. *Viscous Fluid Flow*. McGraw-Hill, third edition, 2006.

[Whi06] Frank M. White. *Viscous Fluid Flow*. McGraw-Hill, third edition, 2006.

[Whi06] Frank M. White. *Viscous Fluid Flow*. McGraw-Hill, third edition, 2006.

[Whi06] Frank M. White. *Viscous Fluid Flow*. McGraw-Hill, third edition, 2006.

[Whi06] Frank M. White. *Viscous Fluid Flow*. McGraw-Hill, third edition, 2006.