MEK 4300
Mandatory Assignment

Andreas Slyngstad

12. mai 2015

Assignment I

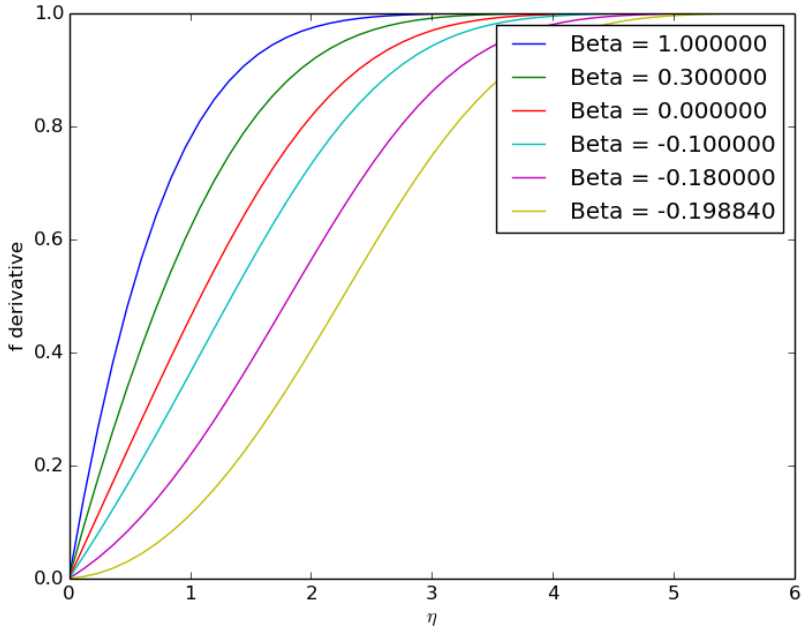
In this assignment we are tasked to implement a non-linear solver for the Falkner-Skan equation

$$f''' + ff'' + \beta\Big(1 - (f'^2)\Big) = 0$$

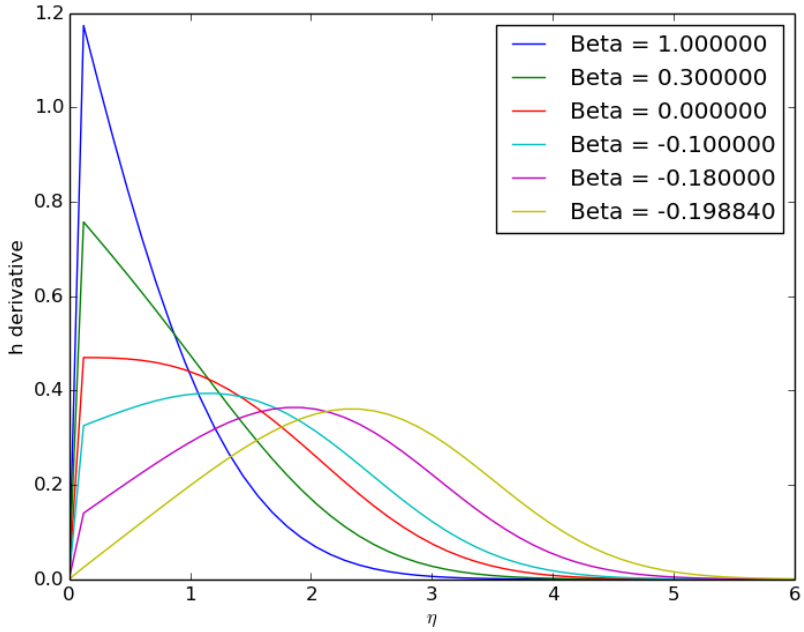
$$\beta = \frac{2m}{m+1}$$

Varying the parameter β gives the following results

Plot of the f derivative for different β



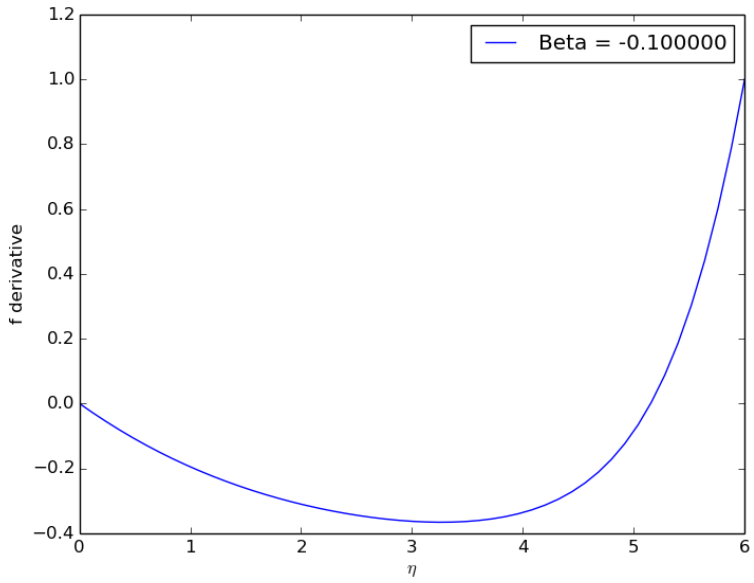
Plot of the h derivative for different β



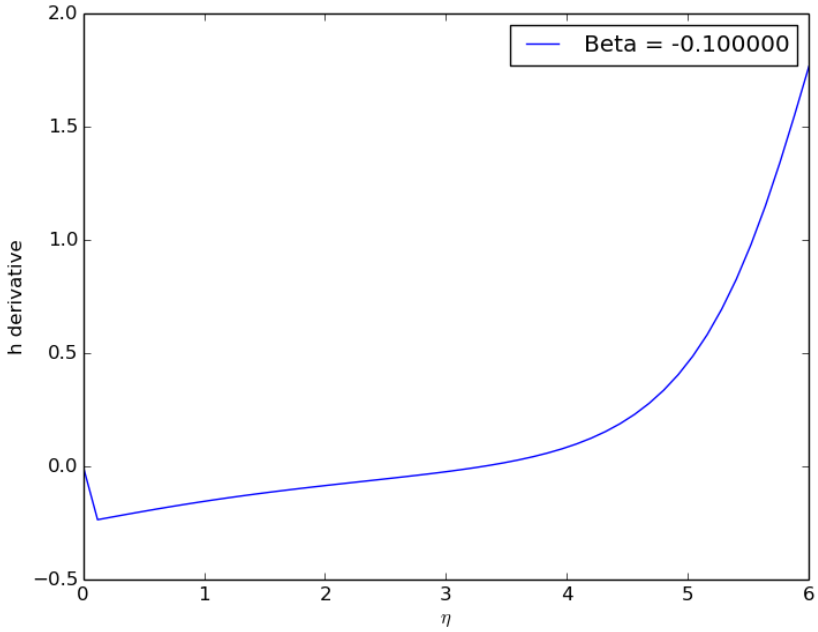
For some reason my program doesn't add the second first point in the plot, making the graph take a leap to the first value. Other than that, I conclude that the program reproduces the figure 4-11 presented in White.

We are also tasked to find a solution which showes negative walues for $f'(0)$. As observed in the plots, with the right guess, we get a solution as mentioned. This indicating a backflow close to the wall.

Plot of the f t β



Plot of the h with negative valuesβ



1 Assignmen II

In this exercise we are looking at a flow around a cylinder with a circular cross-section. First for a steady flow, then for a time dependent flow. I have not succeeded implementing the time dependent flow.
For the steady flow we have the following inflow condition

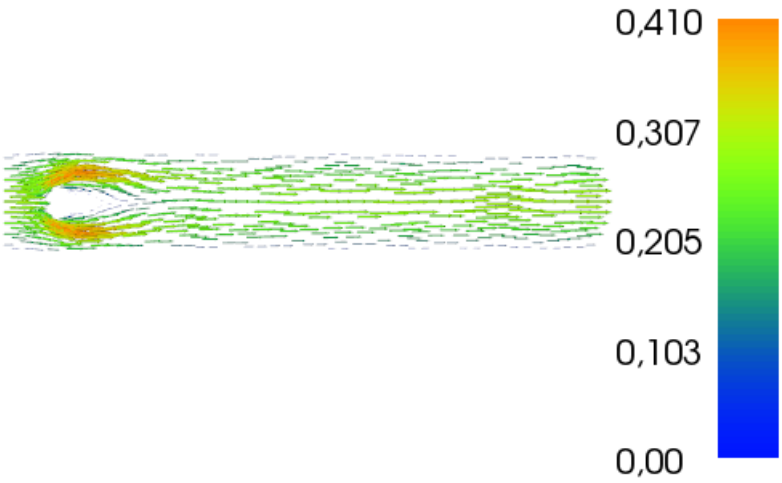
$$U(0,y) = 4U_m y\frac{(H-y)}{H^2}, \qquad V = 0$$

We are requested to compute the drag/lift coefficients, length of the recirculation zone L_a and the pressure difference ΔP The coefficients can be calculated the following way.

$$c_D = \frac{2F_w}{\rho \overline{U} D} \qquad c_F = \frac{2F_a}{\rho \overline{U} D}$$

While ΔP is defined as $\Delta P = P(x_a, y_a, t) - P(x_e, y_e, t)$, where these to points represent the front and end point of the cylinder $(x_a, y_a) = (0.15, 0.2), (x_b, y_b) = (0.25, 0.2)$ Implementing a Falkner-Skan solver for the system, I get the following velocity field and terminal output.

Velocity field β




```
Calulated values: Cd = 5.234797, Cl = -0.007819 pdiff = 0.014665
```

The velocity field seems like a good representation of the flow around the cylinder. We got a faster flow as the fluid passes the cylinder, and a calmer recirculation zone behind.

Assignment III

In this assignment we are looking at a fully developed turbulent channel flow. We are presented with the following

$$0 = \nu\frac{\partial^2\bar{u}}{\partial y^2} - \frac{1}{\rho}\frac{\partial\bar{p}}{\partial x} - \frac{\partial\overline{u'v'}}{\partial y}$$
$$\overline{u'v'} = -\nu_t\frac{\partial\bar{u}}{\partial y} \qquad \nu_t = l^2|\frac{\partial\bar{u}}{\partial y}| \qquad y^+ = \frac{yv^*}{\nu} = 0$$
$$l = \kappa y\Big(1 - \Big(-\frac{y^+}{A}\Big)\Big)$$

We start up by using Hint 1, which states that the pressure gradient is constant. We integrate the momentum equation along the channel. Due to how Hint 2 presents how the mesh can be made, we must take in to account that the mesh is reversed, here h is the distance from the wall towards the centre of the channel.

$$\int_0^h \Big(\nu\frac{\partial^2\bar{u}}{\partial y^2} - \frac{1}{\rho}\frac{\partial\bar{p}}{\partial x} - \frac{\partial\overline{u'v'}}{\partial y}\Big)dy = 0$$
$$\nu\frac{\partial\bar{u}}{\partial y}\Big|_0^h - \frac{1}{\rho}\frac{\partial\bar{p}}{\partial x}h - \bar{u}\bar{v}\Big|_0^h = 0$$

Knowing that $\frac{\partial\bar{u}}{\partial y} = 0$ in the centre, that $l = 0$ for y $= 0$ and that $v^* = \sqrt{\nu\frac{\partial\bar{u}}{\partial y}}$ at the wall, we end up with

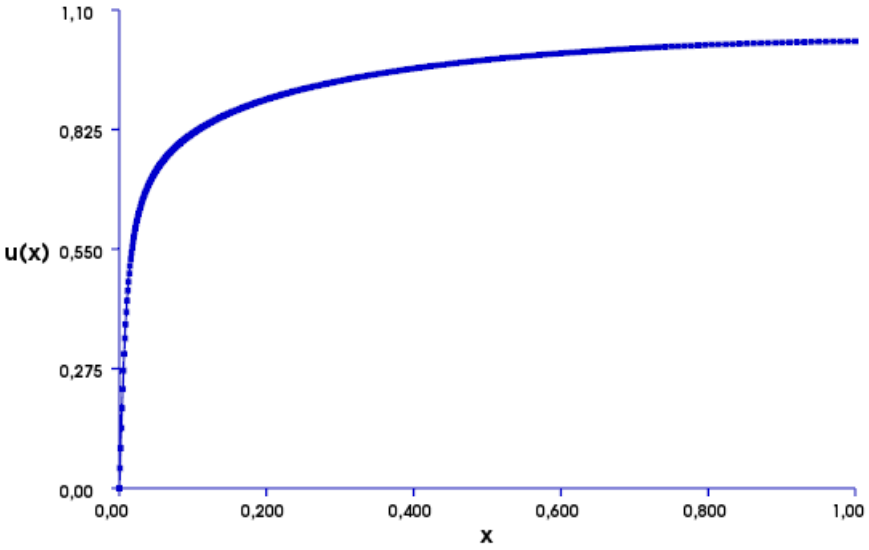
$$\frac{1}{\rho}\frac{\partial\bar{p}}{\partial x} = -\frac{(v^*)^2}{h}$$

Now we have to take a look at the variational form of the problem. It's important to note that the simple v is here a testfunction, not a part of the velocity. Using presented information, and our results from Hint 1 we get the following

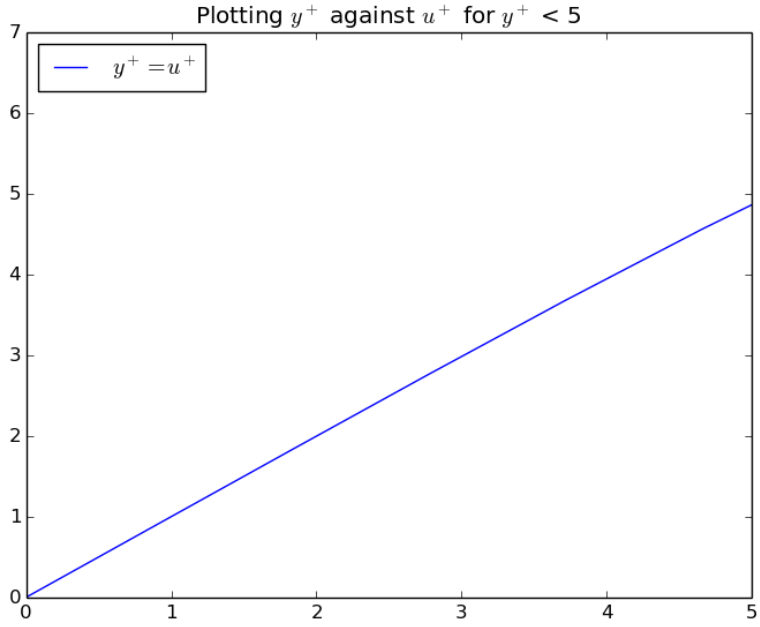
$$\int_\Omega l^2\frac{\partial}{\partial y}\frac{\partial\bar{u}}{\partial y}|\frac{\partial u}{\partial y}|vdx = -\int_\Omega l^2|\frac{\partial\bar{u}}{\partial y}|\frac{\partial\bar{u}}{\partial y}\frac{\partial v}{\partial y}dx + l^2|\frac{\partial\bar{u}}{\partial y}|\frac{\partial\bar{u}}{\partial y}v\Big|^{\partial\Omega} \qquad |\frac{\partial\bar{u}}{\partial y}|\frac{\partial\bar{u}}{\partial y}v\Big|^{\partial\Omega} = 0$$
$$\int_\Omega \Big(\nu\frac{\partial^2\bar{u}}{\partial y^2}v - \frac{1}{\rho}\frac{\partial\bar{p}}{\partial x}v - \frac{\partial\overline{u'v'}}{\partial y}v\Big)dx = 0$$
$$\nu\int_\Omega \nabla u\cdot\nabla v dx - \int_\Omega\frac{(v^*)^2}{h}vdx + \int_\Omega l^2|\frac{\partial\bar{u}}{\partial y}|\frac{\partial\bar{u}}{\partial y}\frac{\partial v}{\partial y}vdx = 0$$

We make a skewed mesh with high concentration of node points close to the wall, because this is the area we need most of our nodes to capture what's going on. Regarding nodes, we are asked how many nodes are needed to ensure $y^+ < 1$ by using a regularly spaced mesh. And the answer to that is all too many, which we can't represent on an ordinary computer. With the skewed mesh we get the following result.

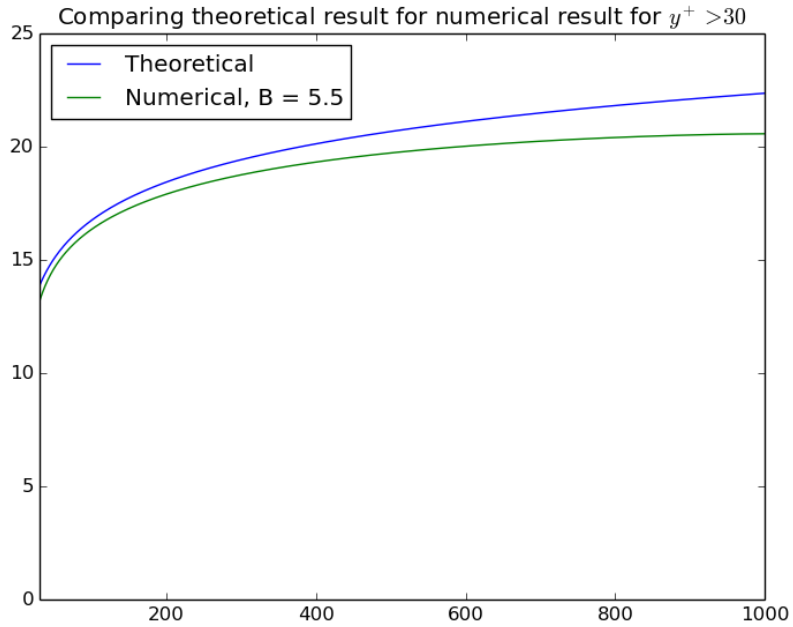
Plot of the numerical solution



Theoretical result agains numerical problem



Theoretical result agains numerical problem



We are supposed to compare our numerical solution to two theoretical results presented in the exercise.

$$y^+ = u^+ \qquad y^+ < 5$$
$$u^+ = \frac{1}{\kappa} ln y^+ + B \qquad y^+ > 30$$

Looking at plot number 2, we can see that our numerical solution looks good in terms of the first theoretical result. We clearly see linearity in the specified domain.
For the second theoretical result we see that our numerical results is a good approximation around the wall, but falls short as the distance to the wall increases. This is not too surprising due to the theoretical solution is bounded to $y^+ > 30$. Hence the theoretical and numerical solutions should be closer to each other at the wall. Calculating the turbulent viscosity at the centre of the channel yields $\nu_t = 0.0011820$. I don't find this value reasonable due to the fact that we know from experiments that alot of turbulence occur at that particular area, so the numerical result and experiments differ.

Assignment IV

- Reynolds Averaged Navier Stokes equations
  We start the derivation with our base, the Navier Stokes equations
  Defining a velocity field $\mathbf{v} = (u, v, w) = (u_1, u_2, u_3)$
  We now express the velocity components as a sum of their mean value and their corresponding fluctuation $u = \bar{u} + u', v = \bar{v} + v', w = \bar{w} + w'$ by direct insertion

$$\nabla \cdot \mathbf{V} = \frac{\partial u_i}{\partial x_i} = 0$$
$$\rho \frac{D\mathbf{v}}{Dt} = \rho \frac{Du_i}{Dt} = -\frac{\partial p}{\partial x_i} + \mu \frac{\partial^2 u_i}{\partial x_j \partial x_j}$$

$$\frac{\partial(\bar{u}_i + u_i')}{\partial x_i} = 0$$
$$\rho \frac{D(\bar{u}_i + u_i')}{Dt} = \rho\big(\frac{\partial(\bar{u}_i + u_i')}{\partial t} + (\bar{u}_j + u_j')\frac{\partial(\bar{u}_i + u_i')}{\partial x_j}\big) = -\frac{\partial(\bar{p} + p')}{\partial x_i} + \mu \frac{\partial^2(\bar{u}_i + u_i')}{\partial x_j \partial x_j}$$

We observe that the convective term can be rewritten, and that the last term is zero due to the continuum equation.

$$(\bar{u}_j + u_j')\frac{\partial(\bar{u}_i + u_i')}{\partial x_j} = \frac{\partial(\bar{u}_i + u_i')(\bar{u}_j + u_j')}{\partial x_j} - (\bar{u}_i + u_i')\frac{\partial(\bar{u}_j + u_j')}{\partial x_j}$$
$$\rho\big(\frac{\partial(\bar{u}_i + u_i')}{\partial t} + \frac{\partial(\bar{u}_i + u_i')(\bar{u}_j + u_j')}{\partial x_j} = -\frac{\partial(\bar{p} + p')}{\partial x_i} + \mu \frac{\partial^2(\bar{u}_i + u_i')}{\partial x_j \partial x_j}$$

We know time-average the whole equation, getting rid of some of the terms and we end up with the following result

$$\rho\big(\frac{\partial \bar{u}_i}{\partial t} + \frac{\bar{u}_i \bar{u}_j}{\partial x_j} + \frac{\partial \overline{u_i' u_j'}}{\partial x_j}\big) = -\frac{\partial \bar{p}}{\partial x_i} + \mu \frac{\partial^2 \bar{u}_i}{\partial x_j \partial x_j}$$

To help out making the equation more compact, we apply the chain rule on the mean convection part, and yet again observing that the last term is zero due to our continuum relation. We end up with the final result

$$\frac{\bar{u}_i \bar{u}_j}{\partial x_j} = \frac{\bar{u}_j \partial \bar{u}_i}{\partial x_j} + \frac{\bar{u}_i \partial \bar{u}_j}{\partial x_j} = \frac{\bar{u}_j \partial \bar{u}_i}{\partial x_j}$$
$$\rho\big(\frac{\partial \bar{u}_i}{\partial t} + \frac{\bar{u}_j \partial \bar{u}_i}{\partial x_j}\big) = -\frac{\partial \bar{p}}{\partial x_i} + \frac{\partial}{\partial x_j}\big(\mu \frac{\bar{u}_i}{\partial x_j} - \rho \overline{u_i' u_j'}\big)$$

- The Turbulence Kinetic-Energy Equation
  We use the same notations and defined velocity fields in this derivation. We start up with the substituted Navier Stokes equations, but this time we multiply them by u'

$$u_i'\big(\frac{\partial(\bar{u}_i+u_i')}{\partial t}+(\bar{u}_j+u_j')\frac{\partial(\bar{u}_i+u_i')}{\partial x_j}\big)=-\frac{u_i'}{\rho}\frac{\partial(\bar{p}+p')}{\partial x_i}+\nu u_i'\frac{\partial^2(\bar{u}_i+u_i')}{\partial x_\partial x_j}$$

  Time averaging the equation, we will continue manipulating the terms in the x-direction to easier see what's going on

$$\overline{u'\frac{\partial u'}{\partial t}}+\bar{u}\overline{u'\frac{\partial u'}{\partial x}}+\overline{u'^2\frac{\partial u'}{\partial x}}+\bar{v}\overline{u'\frac{\partial u'}{\partial y}}+\overline{u'v'}\frac{\partial\bar{u}}{\partial y}+\overline{u'v'\frac{\partial u'}{\partial y}}+\bar{w}\overline{u'\frac{\partial u'}{\partial z}}+\overline{u'w'}\frac{\partial\bar{u}}{\partial z}+$$
$$\overline{u'w'\frac{\partial u'}{\partial z}}=-\overline{\frac{u'}{\rho}\frac{\partial p'}{\partial x}}+\nu\overline{u'\big(\frac{\partial^2u'}{\partial x^2}+\frac{\partial^2u'}{\partial y^2}+\frac{\partial^2u'}{\partial z^2}\big)}$$

  The next step is to rewrite the left hand side of the equation. We do this to aquire a similar form to kinetic energy.

$$\frac{1}{2}\frac{\partial\overline{u'^2}}{\partial t}+\bar{u}\frac{1}{2}\frac{\partial\overline{u'^2}}{\partial x}+\bar{v}\frac{1}{2}\frac{\partial\overline{u'^2}}{\partial y}+\bar{w}\frac{1}{2}\frac{\partial\overline{u'^2}}{\partial z}+\overline{u'^2}\frac{\partial\bar{u}}{\partial x}+\overline{u'v'}\frac{\partial\bar{u}}{\partial y}+\overline{u'w'}\frac{\partial\bar{u}}{\partial z}+$$
$$\Big[\overline{u'^2\frac{\partial u'}{\partial x}}+\overline{u'v'\frac{\partial u'}{\partial y}}+\overline{u'w'\frac{\partial u'}{\partial z}}\Big]$$

  The last term marked in brackets can be rewritten, and by using continuity term, the system can be written as

$$\frac{1}{2}\frac{\partial\overline{u'^2}}{\partial t}+\bar{u}\frac{1}{2}\frac{\partial\overline{u'^2}}{\partial x}+\bar{v}\frac{1}{2}\frac{\partial\overline{u'^2}}{\partial y}+\bar{w}\frac{1}{2}\frac{\partial\overline{u'^2}}{\partial z}=-\overline{u'^2}\frac{\partial\bar{u}}{\partial x}-\overline{u'v'}\frac{\partial\bar{u}}{\partial y}-\overline{u'w'}\frac{\partial\bar{u}}{\partial z}$$
$$-\overline{\frac{u'}{\rho}\frac{\partial p'}{\partial x}}-\frac{1}{2}\Big[\frac{\partial\overline{u'^3}}{\partial x}+\frac{\overline{u'^2v'}}{\partial y}+\frac{\partial\overline{u'^2w'}}{\partial z}\Big]+\nu\overline{u'\nabla^2u'}$$

  By using the exact same approach for the v and w component of $\mathbf{v}$, we end up with similar results. This will turn into something long and nasty, so we will do two things to make the equation more compact. First we define $K=\frac{1}{2}(\overline{u'u'}+\overline{v'v'}+\overline{w'w'})$. Second we add up the components and by using index notation we get a more pleasant equation to look at.

$$\frac{DK}{Dt}=-\frac{\partial}{\partial x_i}\Big[\overline{u_i'(\frac{1}{2}u_j'u_j'+\frac{p'}{\rho})}\Big]-\overline{u_i'u_j'}\frac{\partial\bar{u}_j}{\partial x_i}+\frac{\partial}{\partial x_i}\Big[\nu u_j'\Big(\frac{\partial u_i'}{\partial x_j}+\frac{\partial u_j'}{\partial x_i}\Big)\Big]-\nu\overline{\frac{\partial u_j'}{\partial x_i}\Big(\frac{\partial u_i'}{\partial x_j}+\frac{\partial u_j'}{\partial x_i}\Big)}$$

Code appendix

Exercise I

```
from dolfin import *
from numpy import zeros
import matplotlib.pyplot as plt

L = 6
mesh = IntervalMesh(50, 0, L)
V = FunctionSpace(mesh,'CG', 1)
VV = V*V
vf, vh = TestFunctions(VV) #different testfunc for both equations

xeta = Function(V)
xeta = interpolate(Expression("x[0]"), V)
xpoint  = xeta.vector().array()
#Remember that x[0] = eta....
xd = xpoint[1]-xpoint[0]

def newton(beta):

        bc0 = DirichletBC(VV, Constant((0,0)), "std::abs(x[0]) < 1e-10")
        bc1 = DirichletBC(VV.sub(1), Constant(1), "std::abs(x[0]-%d) < 1e-10" % L
        #Sub 1 is the space for H, where f' = 1 at infinite

        #Guess first solution
        fh_ = interpolate(Expression(("x[0]", "1/L*x[0]"), L = L), VV)

        #L = L due to state what L is in Expression
        f_, h_ = split(fh_)

        F = h_*vf*dx -f_.dx(0)*vf*dx - inner(grad(h_),grad(vh))*dx + f_*h_.dx(0)*

        solve(F==0, fh_, [bc0,bc1])

        f_, h_ = fh_.split(True) #Splitting solutions for f and h
        hp  = h_.vector().array()       #Solution of h derivative

        hder = zeros(len(xpoint)) #h derivative
        for i in range(len(hder)-1):
                hder[i] = (hp[i+1]- hp[i])/xd

        return hp, hder

beta = [1, 0.3, 0, -0.1, -0.18,-0.198838, -0.19884]

def plot():
        for i in range(len(beta)):
                hp, hder = newton(Constant(beta[i]))

                fig1 = plt.figure(1)
                ax1 = fig1.add_subplot(111)
                ax1.plot(xpoint, hp)
```

```
                fig2 = plt.figure(2)
                ax2 = fig2.add_subplot(111)
                ax2.plot(xpoint, hder)

        ax1.set_xlabel("$\eta$")
        ax1.set_ylabel("f derivative")
        ax1.legend(["Beta = %f" % b for b in beta])
        ax2.set_xlabel("$\eta$")
        ax2.set_ylabel("h derivative")
        ax2.legend(["Beta = %f" % b for b in beta])


plot()
plt.show()
```

Exercise II

```
from dolfin import *

mesh = Mesh("tube2.xml")
mesh = refine(mesh) #To get a smoother solution, original was too rough

V = VectorFunctionSpace(mesh, 'CG', 2)
Q = FunctionSpace(mesh, 'CG', 1)
VQ = V*Q
u, p= TrialFunctions(VQ)
v, q = TestFunctions(VQ)
#These are used for the initial guess in Falkner-Skan
ut, pt= TrialFunctions(VQ)
vt, qt = TestFunctions(VQ)


#Condition
rho = 1.0
nu = 10**-3
D =0.1
R = D/2
Re = 20
Um = 0.3
H = 0.41

#Inflow
u_in = Expression(("4*Um*x[1]*(H-x[1])/(H*H)", "0.0"), H=H, Um = Um)

def top(x, on_boundary): return x[1] > (H-DOLFIN_EPS)
def bottom(x, on_boundary): return x[1] < DOLFIN_EPS
def circle(x, on_boundary): return sqrt((x[0] - 0.20)**2 + (x[1] - 0.20)**2) < (R

bc0 = DirichletBC(VQ.sub(0), Constant((0, 0)), top)
bc1 = DirichletBC(VQ.sub(0), Constant((0, 0)), bottom)
bc2 = DirichletBC(VQ.sub(0), Constant((0, 0)), circle)
inflow = DirichletBC(VQ.sub(0), u_in, "x[0] < DOLFIN_EPS")
bcs = [bc0, bc1, bc2, inflow]


F1 = nu*inner(grad(ut), grad(vt))*dx - q*div(ut)*dx - div(vt)*pt*dx + Constant(0)
uguess = Function(VQ)

solve(lhs(F1) == rhs(F1), uguess, bcs=bcs)

k = 0
error = 1
uh_1 = Function(VQ) #The new unknown function

while k < 10 and error > 1e-12:
        F = nu*inner(grad(u), grad(v))*dx + inner(grad(u)*uguess.sub(0), v)*dx -
        solve(lhs(F) == rhs(F), uh_1 , bcs=bcs)
        error = errornorm(uguess, uh_1)
        uguess.assign(uh_1) #updates the unknown for next it.
        k += 1
```

```
us, ps = uguess.split()

#Making parameters to calculate forces and coefficients
n = -FacetNormal(mesh) #Fetching normal
x1 = as_vector((1,0)) #unit vectors
x2 = as_vector((0,1))
nx = dot(n, x1)#Fetching the x and y components of the normal
ny = dot(n, x2)
nn = as_vector((ny, -nx))

cir = AutoSubDomain(circle) #Making a domain from our circle function
ff = FacetFunction("size_t", mesh) #Function with a value for each facet
ff.set_all(0)
cir.mark(ff, 1)
ds = ds[ff]
#ds is in Fenics, representing in use for integration over facet at boundary
un = dot(nn, us)

#Now we calculate the unknowns
rud = (2*Um)/3.0

#Drag
Fd = assemble((rho*nu*dot(grad(un), n)*ny - ps*nx)*ds(1), exterior_facet_domains=
#Lift
Fl = assemble(-(rho*nu*dot(grad(un), n)*nx + ps*ny)*ds(1), exterior_facet_domains
#Drag coefficient
Cd = (2*Fd)/(rho*rud*rud*D)
#Lift coefficient
Cl = (2*Fl)/(rho*rud*rud*D)
#Pressure difference
pressure = ps.compute_vertex_values()
pdiff = pressure[5] - pressure[7] #
mc = mesh.coordinates()
print mc[8][0]

print('Calulated values: Cd = %f, Cl = %f pdiff = %f' % ( Cd, Cl, pdiff))
```

Exercise III

```
from numpy import arctan
from dolfin import *
import matplotlib.pyplot as plt

v_star = 0.05
kappa = 0.41
Re = 1000
A = 26
ht =  1 #Top wall
hb = -1 #Bottom wall
nu = v_star/Re

mesh = IntervalMesh(250,0,1)
x = mesh.coordinates()
x[:,0] = ht -arctan(pi*(x[:,0])) / arctan(pi)

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

#y = Expression("x[0]")
#y = interpolate(y, V)
#plot(y) For understanding
#interactive()

bcs = DirichletBC(V, 0, "std::abs(x[0]) < DOLFIN_EPS")

l = Expression("kappa*x[0]*(1- exp(-x[0]*v_star/(nu*A)))", kappa = kappa, nu = nu

F = -nu*inner(grad(u), grad(v))*dx + (v_star)**2/ht*v*dx
a = lhs(F)
L = rhs(F)

u_ = Function(V)
solve(a==L, u_, bcs)

F_ = nu*inner(grad(u_), grad(v))*dx - (v_star)**2/ht*v*dx + l*l*abs(u_.dx(0))*u_.

#Newtons method
solve(F_== 0, u_, bcs)
plot(u_)
interactive()

#Testing the theoretical values
y_star = x*Re
u_v = u_.compute_vertex_values()
u_star = u_v/v_star

plt.figure(1)
plt.plot(y_star, u_star, label = "$y^{+} = u^{+}$")
plt.axis([0, 5, 0, 7])
plt.title("Plotting $y^{+}$ against $u^{+}$ for $y^{+}$ < 5")
plt.legend(loc="upper left")
```

```
B = 5.5
ut = interpolate(Expression("1/kappa * log(x[0]*Re) + B", B = B, kappa = kappa, R
u_val = ut.compute_vertex_values()

plt.figure(2)
plt.plot(y_star, u_val,'b', label = "Theoretical")
plt.plot(y_star, u_star, 'g', label = "Numerical, B = %1.1f" % B)
plt.axis([30 ,1000, 0, 25])
plt.title("Comparing theoretical result for numerical result for $y^{+} > 30$")
plt.legend(loc='upper left')

plt.show()

u_deri = project(u_.dx(0),V)
li = interpolate(l, V)
li = li.compute_vertex_values()


vt = li[0]**2*u_deri(1) #blir null, men fysikken sier at det er turb, spred, der,
print vt
```