MEK1100 - Oblig 2

Dag Arne Lydvo

25. april 2018

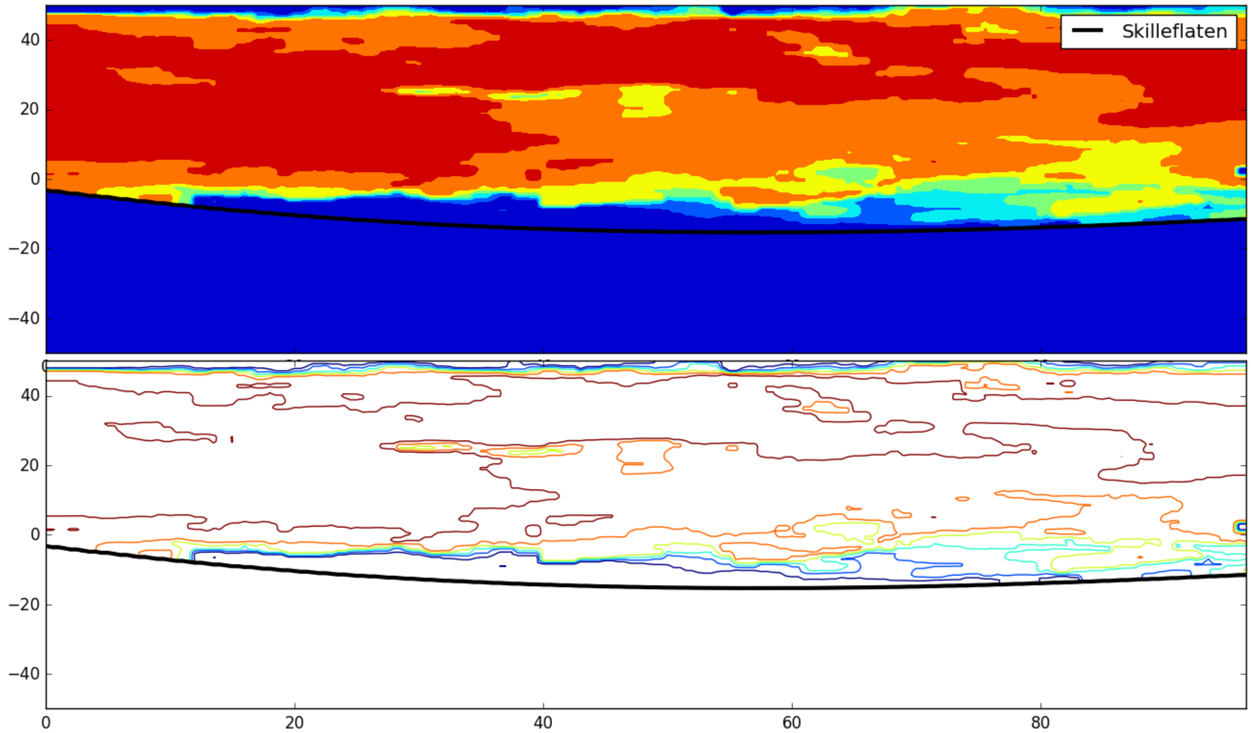
Oppgave a)

```
# -*- coding: utf-8 -*-
import pandas as pd
import scipy.io as sio
import numpy as np
import matplotlib.pyplot as plt

data = sio.loadmat("data.mat")
x = pd.DataFrame(data["x"]).T
y = pd.DataFrame(data["y"]).T
u = pd.DataFrame(data["u"]).T
v = pd.DataFrame(data["v"]).T
xit = pd.DataFrame(data["xit"]).T
yit = pd.DataFrame(data["yit"]).T

#Sjekke matriser og vektorer
print "Matrise x(x,y):",x.shape
print "Matrise y(x,y):",y.shape
print "Matrise u(x,y):",u.shape
print "Matrise v(x,y):",v.shape
print "Matrise xit(x,y):",xit.shape
print "Matrise yit(x,y):",yit.shape

#Sjekke intervallet i gridet
intervallx = pd.Series(x[0]).diff(periods=1)
intervally = pd.Series(y.iloc[0,:]).diff(periods=1)
s = 0
for i,j in zip(intervallx,intervally):
    if i == 0.5 and j == 0.5:
        s += 1
if s == len(intervallx)-1:
    print "Intervaller er 0.5 "

#Sjekke spennet p  y koordinatene
if abs(y[0].min()+y[0].max()) == 100:
    print "y-koordinatene spenner hele diameteren til r ret."
```

Kjøreeksempel:
In [147]: run oblig2a.py
Matrise x(x,y): (194, 201)
Matrise y(x,y): (194, 201)
Matrise u(x,y): (194, 201)
Matrise v(x,y): (194, 201)
Matrise xit(x,y): (194, 1)
Matrise yit(x,y): (194, 1)
Intervaller er 0.5

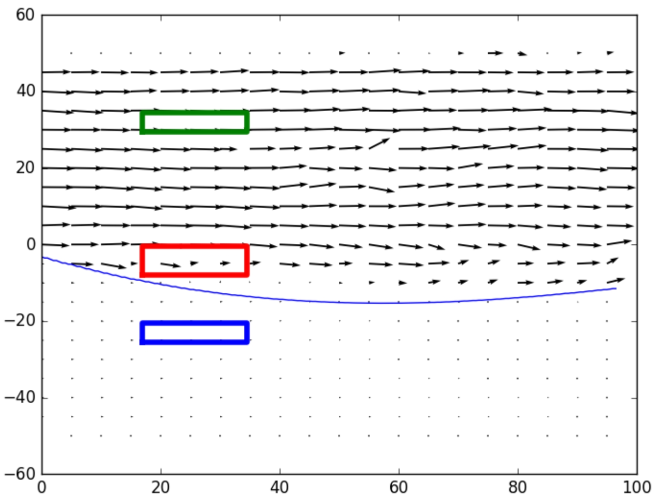
y-koordinatene spenner hele diameteren til røret.

Oppgave b)




```
from  oblig2a import *

z = np.sqrt(u**2+v**2)

plt.subplot(2, 1, 1)
plt.plot(xit,yit,"k",linewidth=3)
plt.contourf(x,y,z)
plt.legend(["Skilleflaten"])
plt.colorbar()
plt.subplot(2,1,2)
plt.plot(xit,yit,"k",linewidth=3)
plt.contour(x,y,z)
plt.colorbar()
plt.show()
```

c)

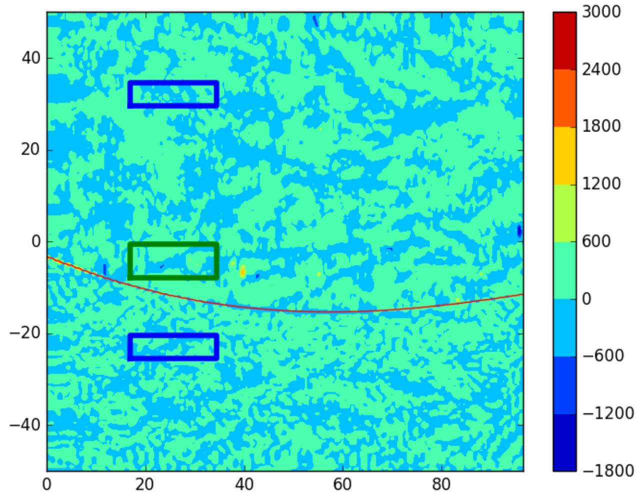



```
# -*- coding: utf-8 -*-
from oblig2a import *

#Velger ut verdier for pilplott
x2 = x.iloc[0::10,0::10]
y2 = y.iloc[0::10,0::10]
u2 = u.iloc[0::10,0::10]
v2 = v.iloc[0::10,0::10]

def rectangle(ix1,iy1,ix2,iy2):
    yy1 =  y[iy1 -1][0]
    xx1 =  x[0][ix1-1]
    xx2 = x[0][ix2-1]
    yy2 = y[iy2 -1][0]
    xx3 = xx2
    yy3 = yy1
    xx4 = xx1
    yy4 = yy2
    xvalues = [xx1,xx3,xx2,xx4,xx1]
    yvalues = [yy1,yy3,yy2,yy4,yy1]
    return xvalues,yvalues

rec1 = rectangle(35,160,70,170)
rec2 = rectangle(35,85,70,100)
rec3 = rectangle(35,50,70,60)

plt.plot(xit,yit)
plt.plot(rec1[0],rec1[1],rec2[0],rec2[1],rec3[0],rec3[1],"b",linewidth=4)
plt.quiver(x2,y2,u2,v2)
plt.show()
```

d)

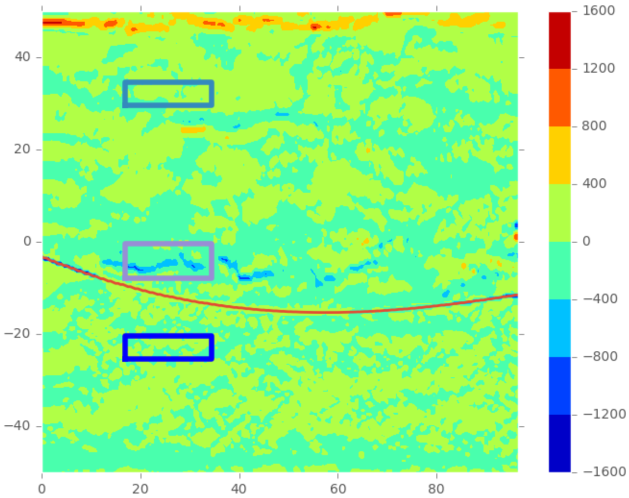
```
from oblig2a import *
```

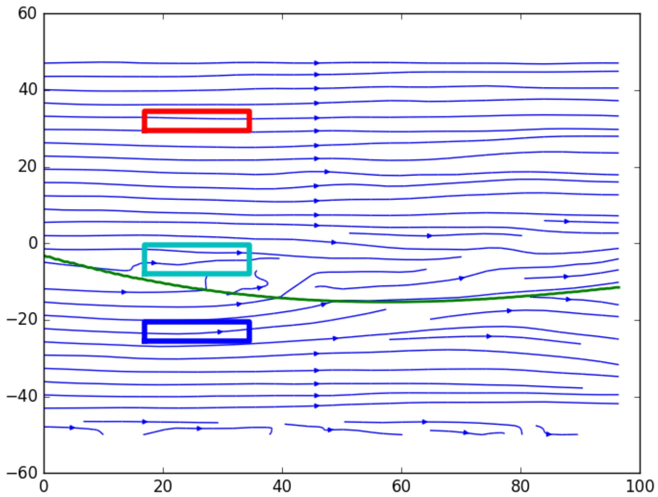
```
from oblig2c import rectangle

dudx = np.gradient(u, axis=0)
dvdy = np.gradient(v, axis=1)
div = dudx + dvdy

rec1 = rectangle(35,160,70,170)
rec2 = rectangle(35,85,70,100)
rec3 = rectangle(35,50,70,60)
plt.contourf(x,y,div)
plt.plot(rec1[0],rec1[1],rec2[0],rec2[1],rec3[0],rec3[1],"b",linewidth=4)
plt.plot(xit,yit)
plt.colorbar()
plt.show()
```



Divergensen til $u\vec{i} + v\vec{j}$ er ikke det samme som divergensen til $\vec{v}$, da hastighets komponenten w er inkludert i $\vec{v}$. Da både gassen og væska er inkompressible vil divergensen til $\vec{v}$ bli null. Divergensen til hastighetskomponenten w vil da være den negative til divergensen til $u\vec{i} + v\vec{j}$.

e)






```python
from oblig2c import *

dudy = np.gradient(u, axis=1)
dvdx = np.gradient(v, axis=0)
vr = dvdx - dudy


plt.contourf(x,y,vr)
plt.plot(xit,yit,linewidth=2)
plt.plot(rec1[0],rec1[1],rec2[0],rec2[1],rec3[0],rec3[1],"b",linewidth=4)
plt.colorbar()
plt.show()

```

```
x = x.values
y = y.values
u = u.values
v = v.values
plt.streamplot(x.T,y.T,u.T,v.T)
plt.plot(xit,yit,linewidth=2)
plt.plot(rec1[0],rec1[1],rec2[0],rec2[1],rec3[0],rec3[1],"b",linewidth=4)
plt.show()
```

Det oppstår turbulens ved skilleflaten mellom væsken og gassen, og noe turbulens ved bunnen av røret da det er friksjon mellom røret og vannet.

f)

Bruker Stokes sats:

```
import pandas as pd
import scipy.io as sio
import numpy as np
import matplotlib.pyplot as plt
import pprint as p
data = sio.loadmat("data.mat")
x = pd.DataFrame(data["x"]).T
y = pd.DataFrame(data["y"]).T
u = pd.DataFrame(data["u"]).T
v = pd.DataFrame(data["v"]).T
xit = pd.DataFrame(data["xit"])
xit = xit.T
yit = pd.DataFrame(data["yit"])
yit = yit.T

#Kurveintegral
def kurvint(x1,y1,x2,y2):
    #De fire sidene i rektangelet ganget med dx,dy = 0.5
    ix1 = u.iloc[:,y1-1][x1-1:x2]*0.5
    ix2 = u.iloc[:,y2-1][x1-1:x2]*0.5
    iy1 = v.iloc[x1-1,:][y1-1:y2]*0.5
    iy2 = v.iloc[x2-1,:][y1-1:y2]*0.5
    I = ix1.sum()-ix2.sum()+iy1.sum()-iy2.sum()
    return I,ix1.sum(),iy1.sum(),-ix2.sum(),-iy2.sum()

Irec1,x11,y11,x12,y12 = kurvint(35,160,70,170)  #Kurveintegralet og de fire sidene.
Irec2,x21,y21,x22,y22 = kurvint(35,85,70,100)
Irec3,x31,y31,x32,y32 = kurvint(35,50,70,60)


dudy = np.gradient(u, axis=1)
dvdx = np.gradient(v, axis=0)
vr = dvdx - dudy
vr = pd.DataFrame(vr)

#Flateintegral
def flatint(x1,y1,x2,y2):
    s = 0
    for i in range(y1-1,y2):
        ix1 = vr.iloc[:,i][x1-1:x2]*(0.5**2)
        I = ix1.sum()
        s += I
    return s

I1 = flatint(35,160,70,170)
I2 = flatint(35,85,70,100)
I3 = flatint(35,50,70,60)
print "Kurveintegral rektangel 1,2 og 3:","%.4f,%.4f,%.4f" %(Irec1,Irec2,Irec3)
print "Flateintegralet for rektangel 1,2 og 3: %.4f, %.4f, %.4f" %(I1,I2,I3)
```

```
print "————————————"
print "Sidene i rektangel 1:","x1:%.4f,y1:%.4f,x2:%.4f,y2:%.4f" %(x11,y11,x12,y12)
print "Sidene i rektangel 2:","x1:%.4f,y1:%.4f,x2:%.4f,y2:%.4f" %(x21,y21,x22,y22)
print "Sidene i rektangel 3:","x1:%.4f,y1:%.4f,x2:%.4f,y2:%.4f" %(x31,y31,x32,y32)
```

Kjøreeksempel:
In [177]: run oblig2f.py Kurveintegral rektangel 1,2 og 3: 839.8215,-61113.3782,-562.9048
Flateintegralet for rektangel 1,2 og 3: 1310.7793, -30741.2705, -6.1072
————————

Sidene i rektangel 1: x1:70100.5239,y1:-661.5727,x2:-68332.8561,y2:-266.273
Sidene i rektangel 2: x1:198.4756,y1:231.8276,x2:-61243.4648,y2:-300.2166
Sidene i rektangel 3: x1:5133.3479,y1:-78.3029,x2:-5410.0397,y2:-207.9100

Kurve og flate integralene blir ikke det samme, forskjellen er nok på grunn av bruk av numerisk metode. Analytisk skulle de være like. Vi ser sirkulasjonen i de to lange sidene i første rektangel blir veldig store, mens y flatene blir små. Dette gir mening da rektangelet ligger i gassen og hastigheten til gassen er veldig høy.
Det midterste rektangelet ligger mellom gassen og væsken og får derfor høy negativ sirkulasjon i den delen som ligger i gassen, mens siden som ligger i væsken får lav sirkulasjon.
Sirkulasjonen i rektangelet som ligger i væsken, blir relativt mindre, da væsken har lavere hastighet.
Resultatet passer godt med det en skulle forvente fra hastighetsfeltet.

g)

```
from oblig2a import *

z = np.sqrt(u**2+v**2)

plt.subplot(2, 1, 1)
plt.plot(xit,yit,"k",linewidth=3)
plt.contourf(x,y,z)
plt.legend(["Skilleflaten"])
plt.colorbar()
plt.subplot(2,1,2)
plt.plot(xit,yit,"k",linewidth=3)
plt.contour(x,y,z)
plt.colorbar()
plt.show()
```

Kjøreeksempel:
In [180]: run oblig2g.py
Total fluks: -1110.4711,-12046.2933,303.3157
Rektangel 1 sider: 1556.8679 21664.5675 -2059.6772 -21664.56
Rektangel 2 sider: -5187.5640 14782.5329 -4074.0522 -14782.5
Rektangel 3 sider: -195.5701 1536.8218 284.9436 -1536.8218

Tallverdien hos de lange flater og korte flater er ganske like, dette fordi det strømmer ca. like mye gass,væske inn i rektangelet som ut. Det midterste rektangelet har noe større fluks da det er mer turbulens i skilleflaten.