MEK 1100 - Oblig 1

Joakim Flatby

27. april 2017

1

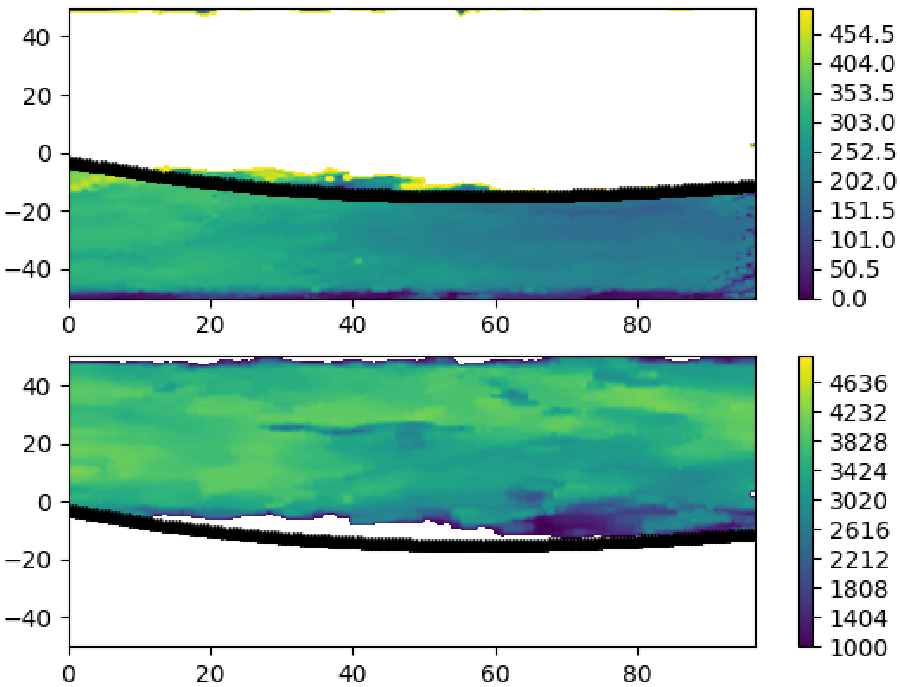
a)

Ved å bruke numpy.shape ser vi at matrisene og vektorene har riktig antall punkter. Ved å printe x og y ser vi at intervallene er 0.5mm og at y går fra -50mm til 50mm.

```
import scipy.io as sio
import numpy as np

data = sio.loadmat("data.mat")
x = data.get("x")
y = data.get("y")
u = data.get("u")
v = data.get("v")
xit = data.get("xit")
yit = data.get("yit")

print np.shape(x)
print np.shape(y)
print np.shape(u)
print np.shape(v)
print np.shape(xit)
print np.shape(yit)
print x
print y

"""
1x-193:oblig2 joakimflatby$ python oppg_a.py
(201, 194)
(201, 194)
(201, 194)
(201, 194)
(1, 194)
(1, 194)
[[  0.     0.5    1.   ...,  95.5   96.    96.5]
 [  0.     0.5    1.   ...,  95.5   96.    96.5]
 [  0.     0.5    1.   ...,  95.5   96.    96.5]
 ...,
 [  0.     0.5    1.   ...,  95.5   96.    96.5]
 [  0.     0.5    1.   ...,  95.5   96.    96.5]
 [  0.     0.5    1.   ...,  95.5   96.    96.5]]
[[-50.  -50.  -50.  ..., -50.  -50.  -50. ]
 [-49.5 -49.5 -49.5 ..., -49.5 -49.5 -49.5]
 [-49.  -49.  -49.  ..., -49.  -49.  -49. ]
 ...,
 [ 49.   49.   49.  ...,  49.   49.   49. ]
 [ 49.5  49.5  49.5 ...,  49.5  49.5  49.5]
 [ 50.   50.   50.  ...,  50.   50.   50. ]]
"""
```

b)

```python
import scipy.io as sio
import numpy as np
import matplotlib.pylab as plt

data = sio.loadmat("data.mat")
x = data.get("x")
y = data.get("y")
u = data.get("u")
v = data.get("v")
xit = data.get("xit")
yit = data.get("yit")

velocity_components = np.sqrt(u**2 + v**2)
plt.figure()

plt.subplot(2, 1, 1)
plt.plot(xit, yit, "k*")
CS = plt.contourf(x, y, velocity_components, np.linspace(0, 500, 100))
plt.colorbar(CS)

plt.subplot(2, 1, 2)
plt.plot(xit, yit, "k*")
CS2 = plt.contourf(x, y, velocity_components, np.linspace(1000, 5000, 100))
plt.colorbar(CS2)

plt.savefig("oppg_b.png")
plt.show()
```

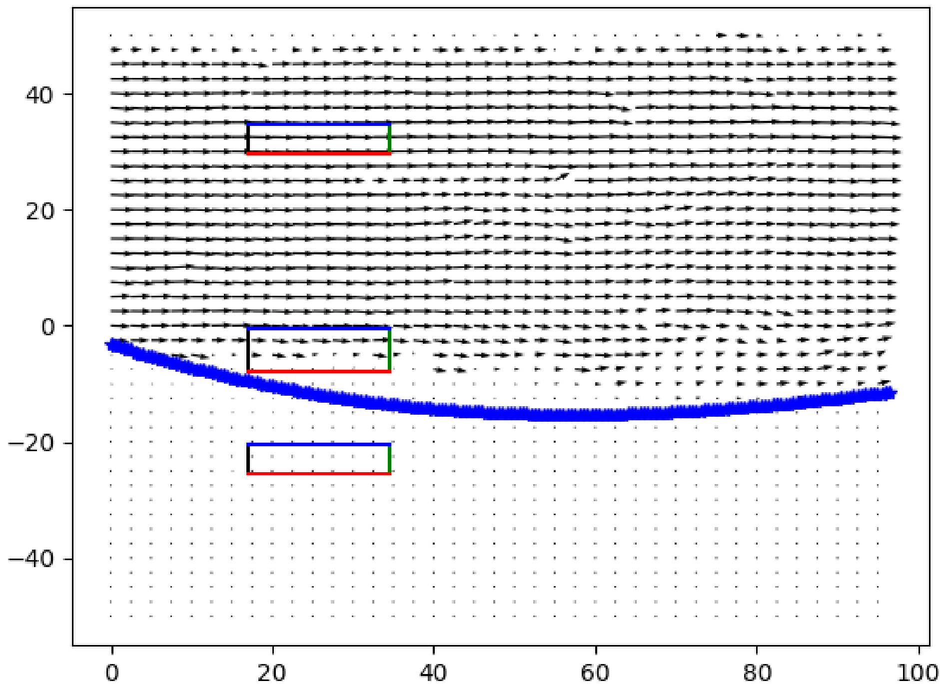


c)

```python
import scipy.io as sio
import matplotlib.pylab as plt

data = sio.loadmat("data.mat")
x = data.get("x")
y = data.get("y")
u = data.get("u")
v = data.get("v")
xit = data.get("xit")
yit = data.get("yit")

def draw_rects():
    ## Rect 1
    corner1 = (x[159, 34], y[159, 34])
    corner2 = (x[169, 69], y[169, 69])

    plt.plot([corner1[0], corner1[0]], [corner1[1], corner2[1]], "k")
    plt.plot([corner2[0], corner2[0]], [corner1[1], corner2[1]], "g")
    plt.plot([corner1[0], corner2[0]], [corner1[1], corner1[1]], "r")
    plt.plot([corner1[0], corner2[0]], [corner2[1], corner2[1]], "b")

    ## Rect 2
    corner1 = (x[84, 34], y[84, 34])
    corner2 = (x[99, 69], y[99, 69])

    plt.plot([corner1[0], corner1[0]], [corner1[1], corner2[1]], "k")
    plt.plot([corner2[0], corner2[0]], [corner1[1], corner2[1]], "g")
    plt.plot([corner1[0], corner2[0]], [corner1[1], corner1[1]], "r")
    plt.plot([corner1[0], corner2[0]], [corner2[1], corner2[1]], "b")

    ## Rect 3
    corner1 = (x[49, 34], y[49, 34])
    corner2 = (x[59, 69], y[59, 69])

    plt.plot([corner1[0], corner1[0]], [corner1[1], corner2[1]], "k")
    plt.plot([corner2[0], corner2[0]], [corner1[1], corner2[1]], "g")
    plt.plot([corner1[0], corner2[0]], [corner1[1], corner1[1]], "r")
    plt.plot([corner1[0], corner2[0]], [corner2[1], corner2[1]], "b")

plt.figure()

draw_rects()

plt.plot(xit, yit, "b*")
skip_num = 5
plt.quiver(x[::skip_num, ::skip_num], y[::skip_num, ::skip_num], u[::skip_num, ::skip_num], v[::skip_num, ::skip_num])

plt.savefig("oppg_c.png")
plt.show()
```

funksjonen draw_rects() bruker jeg i alle de neste oppgavene uten å definere eller importere funksjonen, ettersom alle opprinnelig lå i samme fil.

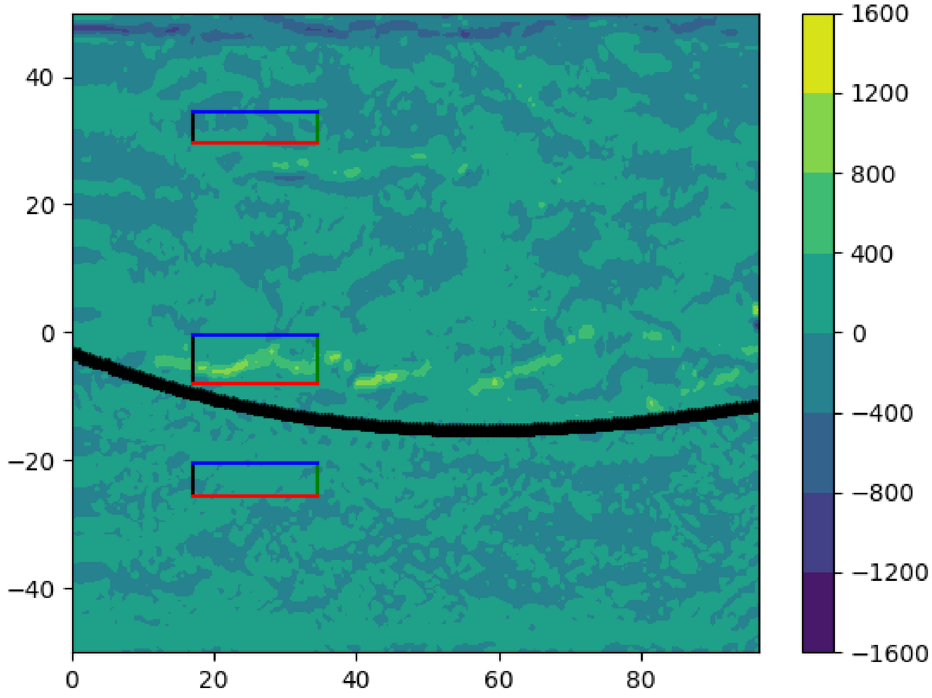
Pilene i væskefasen er så små at man ikke engang kan se retningen på de, men jeg føler at å lage et plot til med annerledes proposjoner bare vil være forvirrende(Hvertfall med de måtene jeg prøvde på..). Dette plottet viser at luften går mye fortere enn vannet, som er tilfellet.



d)

```
import scipy.io as sio
import numpy as np
import matplotlib.pyplot as plt

data = sio.loadmat("data.mat")
x = data.get("x")
y = data.get("y")
u = data.get("u")
v = data.get("v")
xit = data.get("xit")
yit = data.get("yit")

plt.figure()

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


div = dudx + dvdy

draw_rects()

divergence = plt.contourf(x, y, div)
plt.colorbar(divergence)

plt.plot(xit, yit, "k*")

```

```
plt.savefig("oppg_d.png")
plt.show()
```



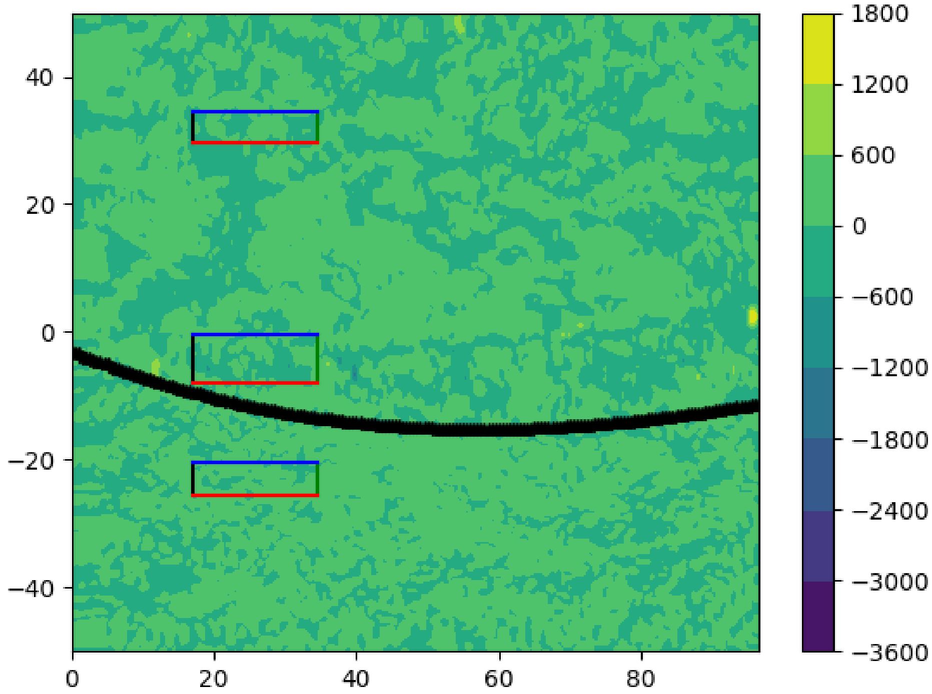
Divergensen til $u\vec{i} + v\vec{j}$ er ikke lik som divergensen til v fordi den mangler w-komponenten. v er definert ved $v = u\vec{i} + v\vec{j} + w\vec{k}$

Konsekvensen av at gassen og væsken er inkompressible er at divergensen til v er 0. Dermed skjønner vi at w har verdier som cancele ut verdiene vi fikk for divergensen til $u\vec{i} + v\vec{j}$, ettersom divergensen til $u\vec{i} + v\vec{j} + w\vec{k}$ er 0

e)

```
import scipy.io as sio
import numpy as np
import matplotlib.pyplot as plt

data = sio.loadmat("data.mat")
x = data.get("x")
y = data.get("y")
u = data.get("u")
v = data.get("v")
xit = data.get("xit")
yit = data.get("yit")

## Oppgave E:
```

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

curl = dvdx - dudy

draw_rects()

vir = plt.contourf(x, y, curl)
plt.colorbar(vir)

plt.plot(xit, yit, "k*")

plt.savefig("oppg_e.png")
plt.show()
```



f)

Jeg får ikke samme resultat med flate- og kurveintegral.. Jeg tror det er flateintegralet som er feil, men får det ikke til.

```
import scipy.io as sio
import numpy as np
import matplotlib.pyplot as plt
from scipy import integrate

```

```
data = sio.loadmat("data.mat")
x = data.get("x")
y = data.get("y")
u = data.get("u")
v = data.get("v")
xit = data.get("xit")
yit = data.get("yit")

## Kurveintegral
rect1_sides = np.zeros(4)
for i in u[159, 34:70]:
    rect1_sides[2] += i*0.5
for i in v[159:170, 69]:
    rect1_sides[1] += i*0.5
for i in u[169, 34:70]:
    rect1_sides[0] -= i*0.5
for i in v[159:170, 34]:
    rect1_sides[3] -= i*0.5


rect2_sides = np.zeros(4)
for i in u[84, 34:70]:
    rect2_sides[2] += i*0.5
for i in v[84:100, 69]:
    rect2_sides[1] += i*0.5
for i in u[99, 34:70]:
    rect2_sides[0] -= i*0.5
for i in v[84:100, 34]:
    rect2_sides[3] -= i*0.5

rect3_sides = np.zeros(4)
for i in u[49, 34:70]:
    rect3_sides[2] += i*0.5
for i in v[49:60, 69]:
    rect3_sides[1] += i*0.5
for i in u[59, 34:70]:
    rect3_sides[0] -= i*0.5
for i in v[49:60, 34]:
    rect3_sides[3] -= i*0.5


rect1 = sum(rect1_sides)
rect2 = sum(rect2_sides)
rect3 = sum(rect3_sides)

print "Rect 1: %f" %(rect1)
print "Rect 2: %f" %(rect2)
print "Rect 3: %f" %(rect3)

"""
1x-193:oblig2 joakimflatby$ python oppg_f.py
Rect 1: 2695.514093
Rect 2: -60976.600162
Rect 3: 9.521016

"""

## Flateintegral

def flate(x1,x2,y1,y2):
    curl = (np.gradient(v,axis=0))-(np.gradient(u,axis=1))
    s = 0
    for i in range (y1,y2):
        for j in range (x1,x2):
            s += curl[i,j]*0.25
    return s

print "Rect 1: %f" %(flate(34, 69, 159, 169))
print "Rect 2: %f" %(flate(34, 69, 84, 99))
print "Rect 3: %f" %(flate(34, 69, 49, 59))

"""
Rect 1: -700.923318
```

```
Rect 2: -6505.544669
Rect 3: 139.104254
"""
```