

MEK 1100 - Oblig 1

Joakim Flatby

27. april 2017

1

a)

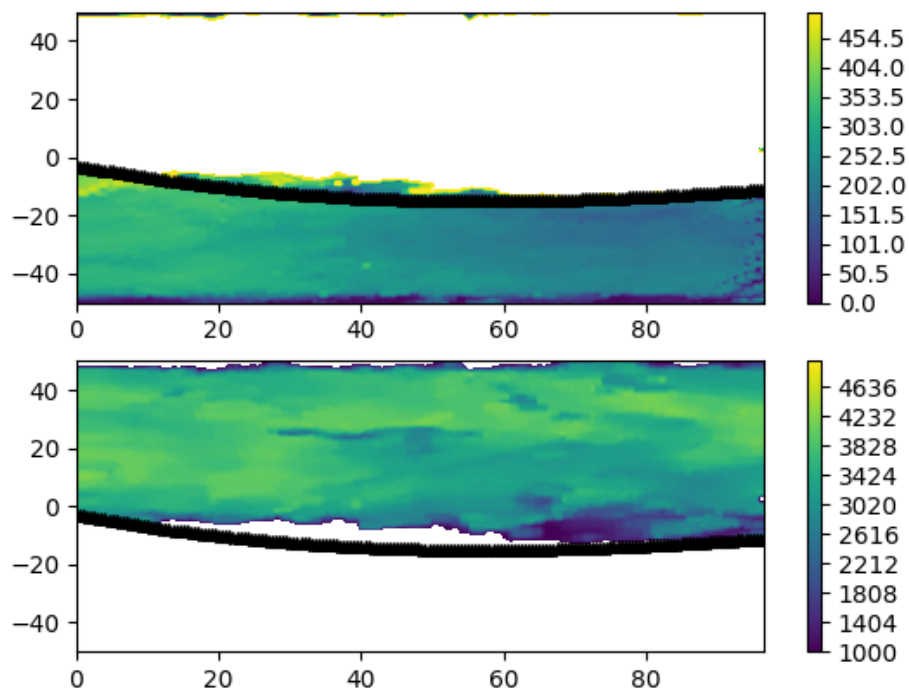
Ved å bruke `numpy.shape` ser vi at matrisene og vektorene har riktig antall punkter.

Ved å printe `x` og `y` ser vi at intervallene er 0.5mm og at `y` går fra -50mm til 50mm.

```
1 import scipy.io as sio
2 import numpy as np
3
4 data = sio.loadmat("data.mat")
5 x = data.get("x")
6 y = data.get("y")
7 u = data.get("u")
8 v = data.get("v")
9 xit = data.get("xit")
10 yit = data.get("yit")
11
12 print np.shape(x)
13 print np.shape(y)
14 print np.shape(u)
15 print np.shape(v)
16 print np.shape(xit)
17 print np.shape(yit)
18 print x
19 print y
20
21 """
22 lx-193:oblig2 joakimflatby$ python oppg-a.py
23 (201, 194)
24 (201, 194)
25 (201, 194)
26 (201, 194)
27 (1, 194)
28 (1, 194)
29 [[ 0.    0.5    1.    ..., 95.5  96.   96.5]
30 [ 0.    0.5    1.    ..., 95.5  96.   96.5]
31 [ 0.    0.5    1.    ..., 95.5  96.   96.5]
32 ...,
33 [ 0.    0.5    1.    ..., 95.5  96.   96.5]
34 [ 0.    0.5    1.    ..., 95.5  96.   96.5]
35 [ 0.    0.5    1.    ..., 95.5  96.   96.5]]
36 [[-50. -50. -50. ..., -50. -50. -50. ]
37 [-49.5 -49.5 -49.5 ..., -49.5 -49.5 -49.5]
38 [-49. -49. -49. ..., -49. -49. -49. ]
39 ...,
40 [ 49.  49.  49. ...,  49.  49.  49. ]
41 [ 49.5 49.5 49.5 ..., 49.5 49.5 49.5]
42 [ 50.  50.  50. ...,  50.  50.  50. ]]
43 """
```

b)

```
1 import scipy.io as sio
2 import numpy as np
3 import matplotlib.pyplot as plt
4
5 data = sio.loadmat("data.mat")
6 x = data.get("x")
7 y = data.get("y")
8 u = data.get("u")
9 v = data.get("v")
10 xit = data.get("xit")
11 yit = data.get("yit")
12
13 velocity_components = np.sqrt(u**2 + v**2)
14 plt.figure()
15
16 plt.subplot(2, 1, 1)
17 plt.plot(xit, yit, "k*")
18 CS = plt.contourf(x, y, velocity_components, np.linspace(0, 500, 100))
19 plt.colorbar(CS)
20
21 plt.subplot(2, 1, 2)
22 plt.plot(xit, yit, "k*")
23 CS2 = plt.contourf(x, y, velocity_components, np.linspace(1000, 5000, 100))
24 plt.colorbar(CS2)
25
26 plt.savefig("oppg-b.png")
27 plt.show()
```

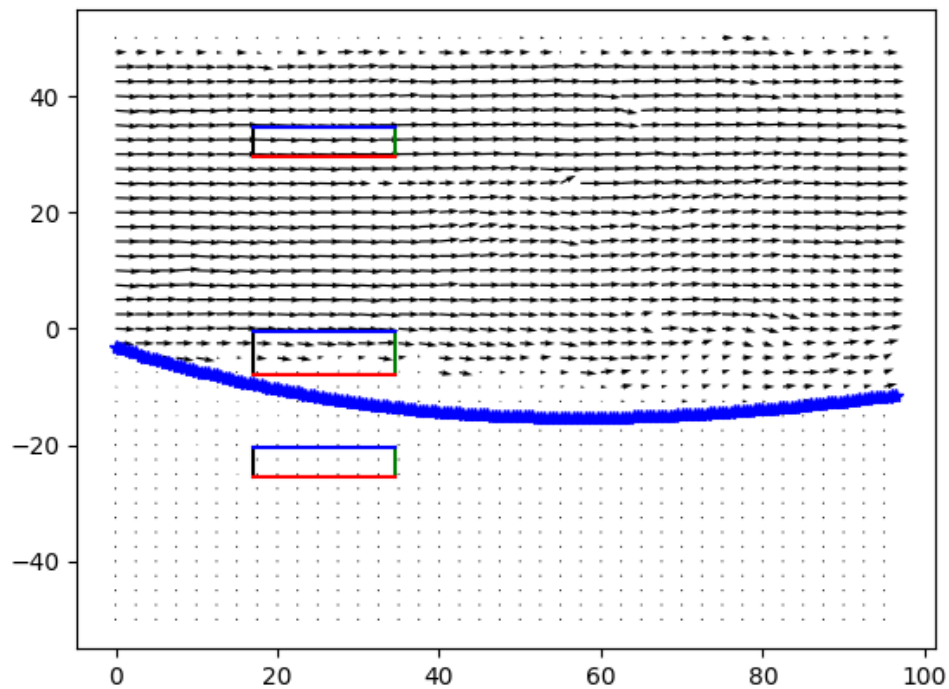


c)

```
1 import scipy.io as sio
2 import matplotlib.pyplot as plt
3
4 data = sio.loadmat("data.mat")
5 x = data.get("x")
6 y = data.get("y")
7 u = data.get("u")
8 v = data.get("v")
9 xit = data.get("xit")
10 yit = data.get("yit")
11
12 def draw_rects():
13     ## Rect 1
14     corner1 = (x[159, 34], y[159, 34])
15     corner2 = (x[169, 69], y[169, 69])
16
17     plt.plot([corner1[0], corner1[0]], [corner1[1], corner2[1]], "k")
18     plt.plot([corner2[0], corner2[0]], [corner1[1], corner2[1]], "g")
19     plt.plot([corner1[0], corner2[0]], [corner1[1], corner1[1]], "r")
20     plt.plot([corner1[0], corner2[0]], [corner2[1], corner2[1]], "b")
21
22     ## Rect 2
23     corner1 = (x[84, 34], y[84, 34])
24     corner2 = (x[99, 69], y[99, 69])
25
26     plt.plot([corner1[0], corner1[0]], [corner1[1], corner2[1]], "k")
27     plt.plot([corner2[0], corner2[0]], [corner1[1], corner2[1]], "g")
28     plt.plot([corner1[0], corner2[0]], [corner1[1], corner1[1]], "r")
29     plt.plot([corner1[0], corner2[0]], [corner2[1], corner2[1]], "b")
30
31     ## Rect 3
32     corner1 = (x[49, 34], y[49, 34])
33     corner2 = (x[59, 69], y[59, 69])
34
35     plt.plot([corner1[0], corner1[0]], [corner1[1], corner2[1]], "k")
36     plt.plot([corner2[0], corner2[0]], [corner1[1], corner2[1]], "g")
37     plt.plot([corner1[0], corner2[0]], [corner1[1], corner1[1]], "r")
38     plt.plot([corner1[0], corner2[0]], [corner2[1], corner2[1]], "b")
39
40 plt.figure()
41
42 draw_rects()
43
44 plt.plot(xit, yit, "b*")
45 skip_num = 5
46 plt.quiver(x[::skip_num, ::skip_num], y[::skip_num, ::skip_num], u[::skip_num, ::skip_num], v[::skip_num, ::skip_num])
47
48 plt.savefig("oppg-c.png")
49 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 forttere enn vannet, som er tilfellet.



d)

```

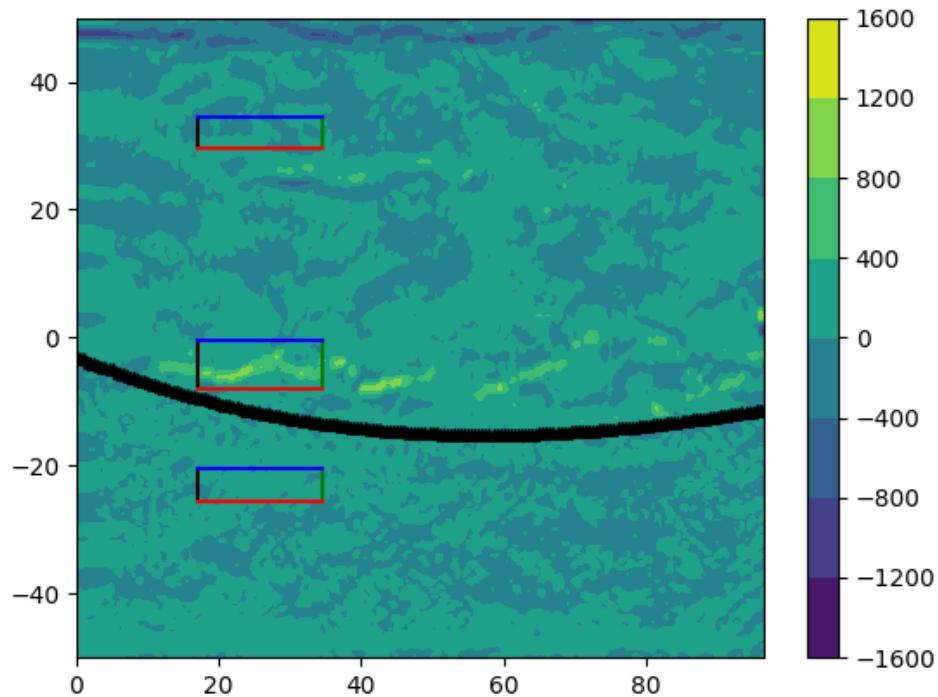
1 import scipy.io as sio
2 import numpy as np
3 import matplotlib.pyplot as plt
4
5 data = sio.loadmat("data.mat")
6 x = data.get("x")
7 y = data.get("y")
8 u = data.get("u")
9 v = data.get("v")
10 xit = data.get("xit")
11 yit = data.get("yit")
12
13 plt.figure()
14
15 dux = np.gradient(u, axis=0)
16 dvdy = np.gradient(v, axis=1)
17
18
19 div = dux + dvdy
20
21 draw_rects()
22
23 divergence = plt.contourf(x, y, div)
24 plt.colorbar(divergence)
25
26 plt.plot(xit, yit, "k*")
27

```

```

28 plt.savefig("oppg.d.png")
29 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 canceler 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)

```

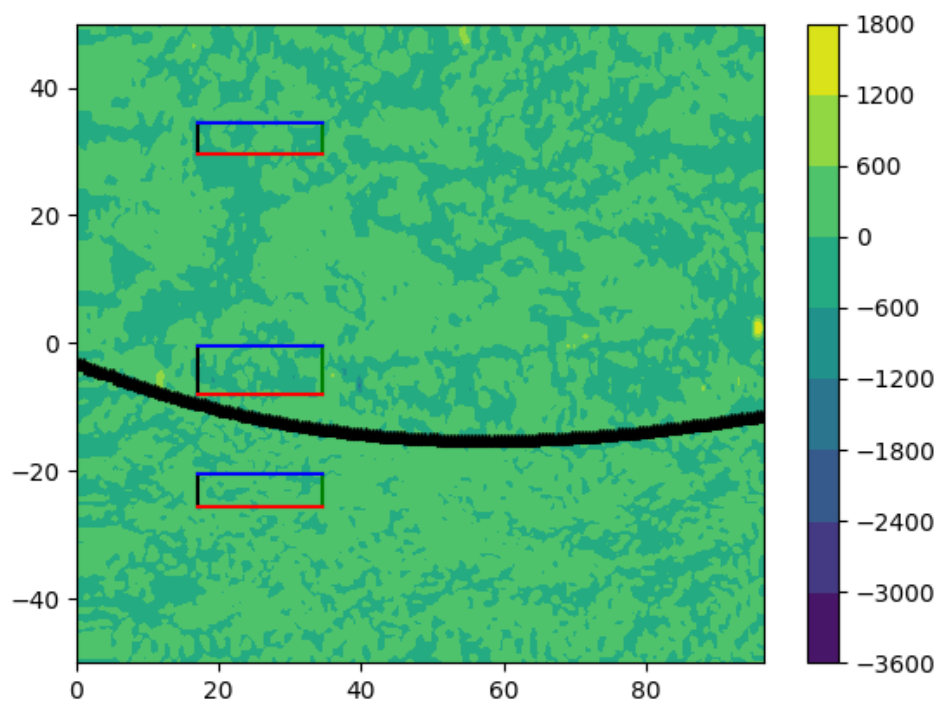
1 import scipy.io as sio
2 import numpy as np
3 import matplotlib.pyplot as plt
4
5 data = sio.loadmat("data.mat")
6 x = data.get("x")
7 y = data.get("y")
8 u = data.get("u")
9 v = data.get("v")
10 xit = data.get("xit")
11 yit = data.get("yit")
12
13 ## Oppgave E:

```

```

14
15 dudy = np.gradient(u, axis=1)
16 dvdx = np.gradient(v, axis=0)
17
18 curl = dvdx - dudy
19
20 draw_rects()
21
22 vir = plt.contourf(x, y, curl)
23 plt.colorbar(vir)
24
25 plt.plot(xit, yit, "k*")
26
27 plt.savefig("oppg-e.png")
28 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.

```

1 import scipy.io as sio
2 import numpy as np
3 import matplotlib.pyplot as plt
4 from scipy import integrate
5

```

```

6 data = sio.loadmat("data.mat")
7 x = data.get("x")
8 y = data.get("y")
9 u = data.get("u")
10 v = data.get("v")
11 xit = data.get("xit")
12 yit = data.get("yit")
13
14 ## Kurveintegral
15 rect1_sides = np.zeros(4)
16 for i in u[159, 34:70]:
17     rect1_sides[2] += i*0.5
18 for i in v[159:170, 69]:
19     rect1_sides[1] += i*0.5
20 for i in u[169, 34:70]:
21     rect1_sides[0] -= i*0.5
22 for i in v[159:170, 34]:
23     rect1_sides[3] -= i*0.5
24
25
26 rect2_sides = np.zeros(4)
27 for i in u[84, 34:70]:
28     rect2_sides[2] += i*0.5
29 for i in v[84:100, 69]:
30     rect2_sides[1] += i*0.5
31 for i in u[99, 34:70]:
32     rect2_sides[0] -= i*0.5
33 for i in v[84:100, 34]:
34     rect2_sides[3] -= i*0.5
35
36 rect3_sides = np.zeros(4)
37 for i in u[49, 34:70]:
38     rect3_sides[2] += i*0.5
39 for i in v[49:60, 69]:
40     rect3_sides[1] += i*0.5
41 for i in u[59, 34:70]:
42     rect3_sides[0] -= i*0.5
43 for i in v[49:60, 34]:
44     rect3_sides[3] -= i*0.5
45
46
47 rect1 = sum(rect1_sides)
48 rect2 = sum(rect2_sides)
49 rect3 = sum(rect3_sides)
50
51 print "Rect 1: %f" %(rect1)
52 print "Rect 2: %f" %(rect2)
53 print "Rect 3: %f" %(rect3)
54
55 """
56 lx-193:oblig2 joakimflatby$ python oppg-f.py
57 Rect 1: 2695.514093
58 Rect 2: -60976.600162
59 Rect 3: 9.521016
60
61 """
62
63 ## Flateintegral
64
65 def flate(x1,x2,y1,y2):
66     curl = (np.gradient(v,axis=0))-(np.gradient(u,axis=1))
67     s = 0
68     for i in range (y1,y2):
69         for j in range (x1,x2):
70             s += curl[i,j]*0.25
71     return s
72
73 print "Rect 1: %f" %(flate(34, 69, 159, 169))
74 print "Rect 2: %f" %(flate(34, 69, 84, 99))
75 print "Rect 3: %f" %(flate(34, 69, 49, 59))
76
77 """
78 Rect 1: -700.923318

```

79	Rect 2: -6505.544669
80	Rect 3: 139.104254
81	" "