

Christophe Blomsen
chriskbl@student.matnat.uio.no

1. mai 2020

Innhold

a)	1
b)	1
c)	2
d)	3
e)	3
f)	4
g)	4

Figurer

1	Graf til oppgave b	6
2	Graf til oppgave c	7
3	Graf til oppgave d	8
4	Graf til oppgave e	9

Kode

1	Kode oppgave a)	1
2	Kode oppgave b	2
3	Kode til oppgave c	2
4	Oppgave d	3

a)

All kode vil ligge i samme python fil så de relevante kode snuttene er tilgjengelig på de tilsvarende deloppgavene. Hele filen kan finnes i Appendikset.

Kode 1: Kode oppgave a)

```
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 print(np.shape(x))
14 print(np.shape(y))
15 print(np.shape(u))
16 print(np.shape(v))
17 print(np.shape(xit))
18 print(np.shape(yit))
19
20 print(x)
21 print(y)
```

Utskrift til terminalen blir

```
x shape is (201, 194)
y shape is (201, 194)
u shape is (201, 194)
v shape is (201, 194)
xit shape is (1, 194)
yit shape is (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. ]]
```

Ser da at griddet i xy -planet har et regulært intervall på 0.5 mm i begge retninger. Samt at y -koordinatene spenner ut hele diameteren til røret.

b)

Observerer fra presentasjonen av eksperimentet at det er væskefase i den nedre halvdelen av røret og gassfase i den andre halvdelen.

Kode 2: Kode oppgave b

```
1 velocity = np.sqrt(u**2 + v**2)
2
3 plt.subplot(2, 1, 1)
4 plt.plot(xit, yit, "k*")
5 water_bender = plt.contourf(x, y, velocity, np.linspace(0, 500, 100))
6 plt.colorbar(water_bender)
7
8 plt.subplot(2, 1, 2)
9 plt.plot(xit, yit, "k*")
10 air_bender = plt.contourf(x, y, velocity, np.linspace(1000, 5000, 100))
11 plt.colorbar(air_bender)
12
13 plt.savefig("oppgave_b.png")
14 plt.show()
```

Det produserer følgende [plot](#)

c)

Velger å bruke vært femte element i pilplottet

Kode 3: Kode til oppgave c

```
1 def rectangle(x1, x2, y1, y2):
2     position1 = (x[x2, x1], y[x2, y1])
3     position2 = (x[y2, y1], y[y2, y1])
4
5     # Bottom
6     plt.plot([position1[0], position2[0]], [position1[1], position1[1]], "r")
7
8     # Right
9     plt.plot([position2[0], position2[0]], [position1[1], position2[1]], "g")
10
11     # Top
12     plt.plot([position1[0], position2[0]], [position2[1], position2[1]], "b")
13
14     # Left
15     plt.plot([position1[0], position1[0]], [position1[1], position2[1]], "k")
16
17
18 def draw_rectangles():
19     rectangle1_values = [34, 159, 69, 169]
20     rectangle(rectangle1_values[0], rectangle1_values[1],
21               rectangle1_values[2], rectangle1_values[3])
22
23     rectangle2_values = [34, 84, 69, 100]
24     rectangle(rectangle2_values[0], rectangle2_values[1],
25               rectangle2_values[2], rectangle2_values[3])
26
27     rectangle3_values = [34, 49, 69, 59]
28     rectangle(rectangle3_values[0], rectangle3_values[1],
29               rectangle3_values[2], rectangle3_values[3])
30
31 draw_rectangles()
32 plt.plot(xit, yit, "k*")
33 num_skip = 5
```

```

34 plt.quiver(x[::num_skip, ::num_skip], y[::num_skip, ::num_skip],
35            u[::num_skip, ::num_skip], v[::num_skip, ::num_skip])
36
37 plt.title("Oppgave c)")
38 plt.xlabel("x")
39 plt.ylabel("y")
40
41 plt.savefig("oppgave-c.png")

```

Denne kodesnutten produserer følgende [plot](#)

d)

I numpy pakken til python så finnes det flere finne funksjoner, i denne oppgaven så blir `numpy.gradient` funksjonen brukt. Det den gjør er å regne ut gradienten til arrays. Hvis man også bruker keyword argumentet `axis` så kan man velge hvilken av komponente du vil ha. I kodesnutten under er da dette oppnådd med at vi i `dudx` kun trekker ut x -komponente fra gradienten til u tilsvarende for `dvdv`.

Kode 4: Oppgave d

```

1 dudx = np.gradient(u, 0.5, axis=0)
2 dvdy = np.gradient(v, 0.5, axis=1)
3
4 divergence = dudx + dvdy
5 print(f"The divergence is {divergence}")
6
7 plt.contourf(x, y, divergence)
8 plt.colorbar()
9 plt.title("Oppgave d)")
10 plt.savefig("oppgave-d.png")

```

Denne kodesnutten produserer da følgende [plot](#). Divergensen til \mathbf{v} er ikke den samme som divergensen til $u\mathbf{i} + v\mathbf{j}$. Dette fordi divergensen til \mathbf{v} er

$$\nabla \cdot \mathbf{v} = \frac{\partial v}{\partial x}$$

e)

`np.gradient` blir brukt på samme måte i denne oppgaven for å finne virvlingen.

```

1 dudy = np.gradient(u, 0.5, axis=0)
2 dvdx = np.gradient(v, 0.5, axis=1)
3
4 curl_v = dvdx - dudy
5
6 curl_plot = plt.contourf(x, y, curl_v)
7 plt.streamplot(x, y, u, v, color="orange")
8 plt.colorbar()
9
10 plt.title("Oppgave e)")
11
12 plt.savefig("oppgave-e.png")
13 plt.show()

```

Konturplottet av denne virvlingskomponenten kan finnes [her](#). Observerer at strømmingen skaper sirkulasjon mellom gass- og væskefasen, spesielt rundt det miderste rektangelet. Ser også at strømmingen trekkes til veggen på grunn av friksjon.

f)

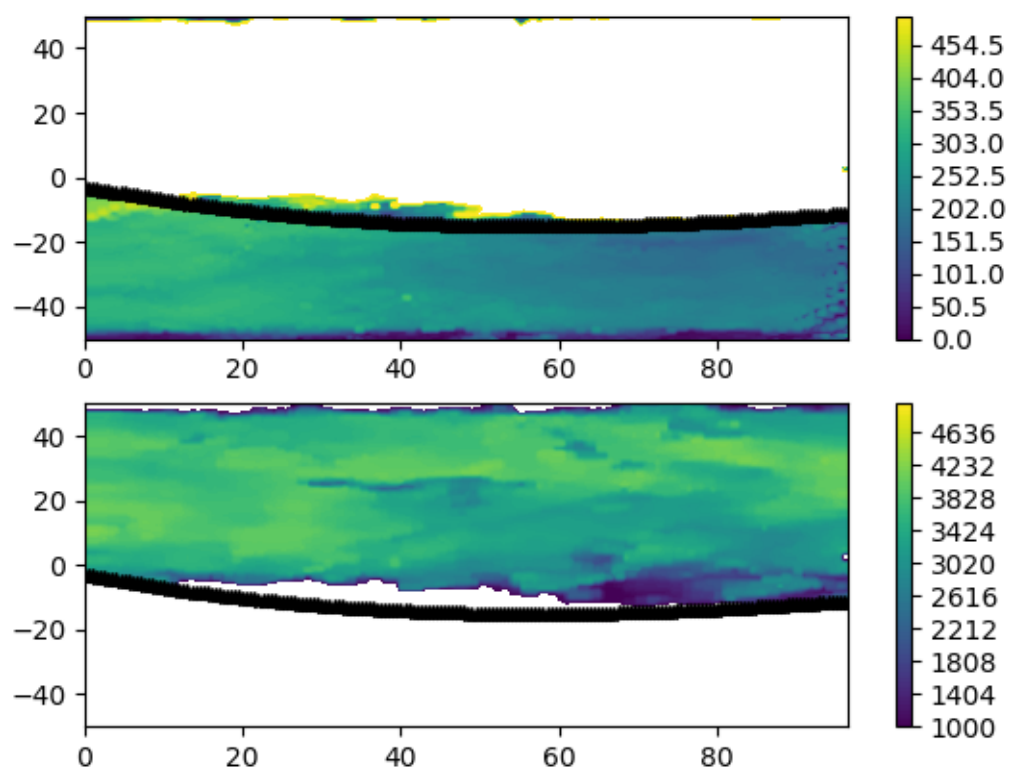
```
1 def line_integral(x1, y1, x2, y2):
2     side1 = 0
3     side2 = 0
4     side3 = 0
5     side4 = 0
6     dx = 0.5
7     dy = 0.5
8
9     for k in u[y1, x1:x2+1]:
10         side1 += k*dx
11
12     for k in v[x2, y1:y2+1]:
13         side2 += k*dy
14
15     for k in u[y2, x1:x2+1]:
16         side3 -= k*dx
17
18     for k in v[x1, y1:y2+1]:
19         side4 -= k*dy
20
21     summation = side1 + side2 + side3 + side4
22     return summation
23
24
25 def surface_integral(x1, y1, x2, y2):
26     integral = 0
27     dx = 0.5
28     dy = 0.5
29     for i in range(x1, x2+1):
30         for j in range(y1, y2+1):
31             integral += curl_v[j, i]*dx*dy
32
33     return integral
34
35
36 print(line_integral(34, 159, 69, 169), line_integral(34, 84, 69, 99),
37       line_integral(34, 49, 69, 59))
38 print(surface_integral(34, 159, 69, 169), surface_integral(34, 84, 69, 99),
39       surface_integral(34, 49, 69, 59))
```

g)

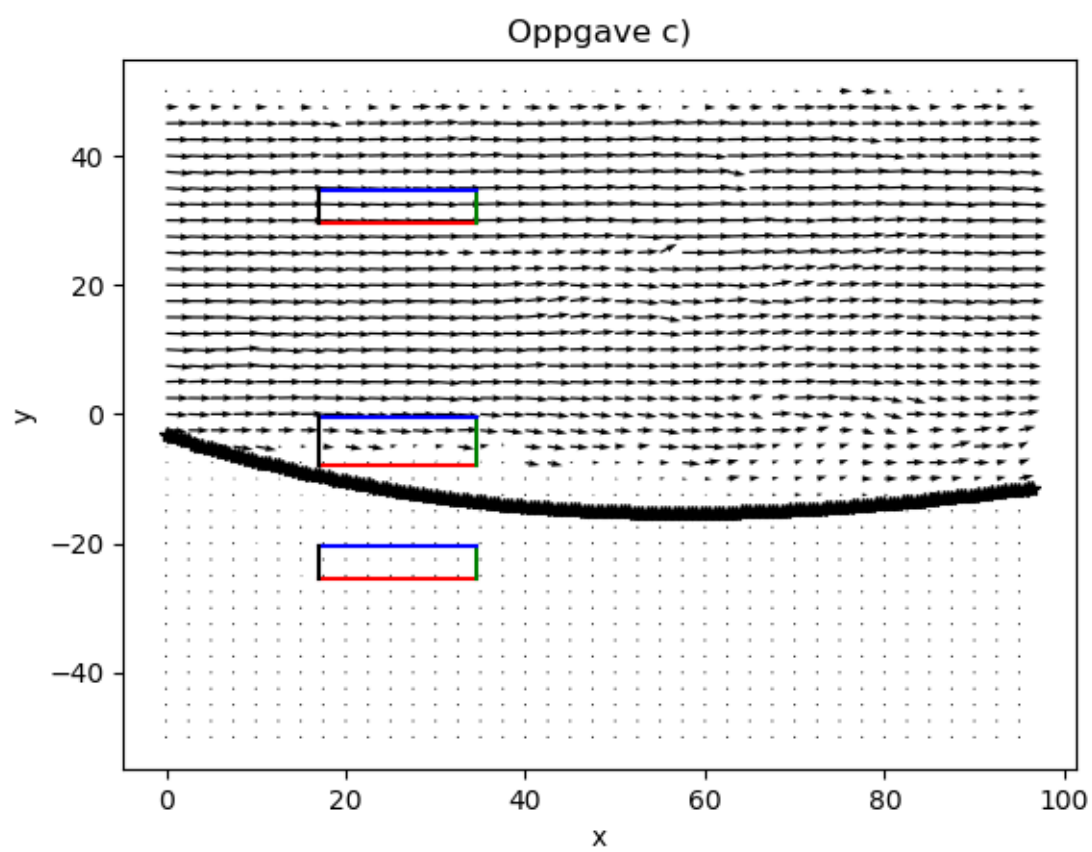
```
1 def gaus(x1, y1, x2, y2):
2     side1 = 0
3     side2 = 0
4     side3 = 0
5     side4 = 0
6     dx = 0.5
7     dy = 0.5
8     dz = 1
9
10    for k in v[y1, x1:x2+1]:
11        side1 -= k*dx*dz
12
13    for k in u[x2, y1:y2+1]:
```

```
14     side2 += k*dy*dz
15
16     for k in v[y2, x1:x2+1]:
17         side3 += k*dx*dz
18
19     for k in u[x1, y1:y2+1]:
20         side4 -= k*dy*dz
21
22     summation = side1 + side2 + side3 + side4
23     return summation
```

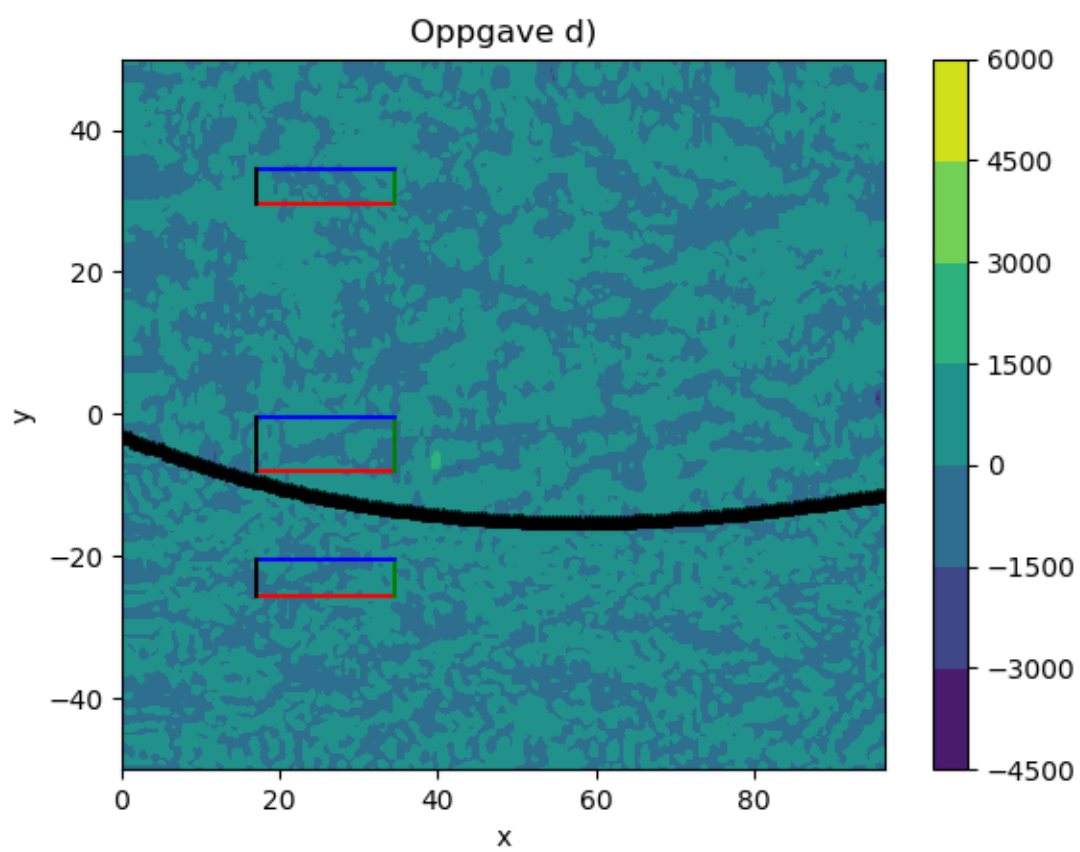
Figur 1: Graf til oppgave b



Figur 2: Graf til oppgave c



Figur 3: Graf til oppgave [d](#)



Figur 4: Graf til oppgave e

