Christophe Blomsen
chriskbl@student.matnat.uio.no

2. mai 2020

Innhold

Figurer

Kode

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)

```
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')

print(f"x shape er {np.shape(x)}")
print(f"y shape er {np.shape(y)}")
print(f"u shape er {np.shape(u)}")
print(f"v shape er {np.shape(v)}")
print(f"xit shape er {np.shape(xit)}")
print(f"yit shape er {np.shape(yit)}")

print(x)
print(y)
```

Utskrift til terminalen blir

```
x shape er (201, 194)
y shape er (201, 194)
u shape er (201, 194)
v shape er (201, 194)
xit shape er (1, 194)
yit shape er (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 rettninger. 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

```
velocity = np.sqrt(u**2 + v**2)

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

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

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

Det produserer følgende plot

c)

Velger å bruke vært femte element i pilplottet

Kode 3: Kode til oppgave c

```
def rectangle(x1, x2, y1, y2):
    position1 = (x[x2, x1], y[x2, y1])
    position2 = (x[y2, y1], y[y2, y1])

    # Bottom
    plt.plot([position1[0], position2[0]], [position1[1], position1[1]], "r")

    # Right
    plt.plot([position2[0], position2[0]], [position1[1], position2[1]], "g")

    # Top
    plt.plot([position1[0], position2[0]], [position2[1], position2[1]], "b")

    # Left
    plt.plot([position1[0], position1[0]], [position1[1], position2[1]], "k")


def draw_rectangles():
    rectangle1_values = [34, 159, 69, 169]
    rectangle(rectangle1_values[0], rectangle1_values[1],
              rectangle1_values[2], rectangle1_values[3])

    rectangle2_values = [34, 84, 69, 100]
    rectangle(rectangle2_values[0], rectangle2_values[1],
              rectangle2_values[2], rectangle2_values[3])

    rectangle3_values = [34, 49, 69, 59]
    rectangle(rectangle3_values[0], rectangle3_values[1],
              rectangle3_values[2], rectangle3_values[3])

draw_rectangles()
plt.plot(xit, yit, "k*")
num_skip = 5
```

```
plt.quiver(x[::num_skip, ::num_skip], y[::num_skip, ::num_skip],
           u[::num_skip, ::num_skip], v[::num_skip, ::num_skip])

plt.title("Oppgave c)")
plt.xlabel("x")
plt.ylabel("y")

plt.savefig("oppgave_c.png")
```

Denne kodesnutten produserer følgede 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 *dvdy*.

Kode 4: Oppgave d

```
dudx = np.gradient(u, 0.5, axis=0)
dvdy = np.gradient(v, 0.5, axis=1)

divergence = dudx + dvdy
print(f"Divergensen er {divergence}")

plt.contourf(x, y, divergence)
plt.colorbar()
plt.title("Oppgave d)")
plt.savefig("oppgave_d.png")
```

Denne kodesnutten produserer da følgende plot.
Divergensen til $u\mathbf{i} + v\mathbf{j} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}$, så det mangler w komponenten for at det skal være likt med $\nabla \cdot \mathbf{v}$
Siden gassen og væska er inkompressible så betyr det at $\nabla \cdot \mathbf{v} = 0$. Dette medfører

$$\nabla \cdot \mathbf{v} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0$$

$$\frac{\partial w}{\partial z} = -\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right)$$

Da vil hastighetskompenten

$$w = \int \left(\frac{\partial v}{\partial y} + \frac{\partial u}{\partial x}\right) \mathrm{d}z$$

Dette integralet kan ikke være være 0 derfor må divergensen til $\mathbf{v}$ være forskjellig fra divergensen til $u\mathbf{i} + v\mathbf{j}$

e)

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

```python
dudy = np.gradient(u, 0.5, axis=0)
dvdx = np.gradient(v, 0.5, axis=1)

curl_v = dvdx - dudy

curl_plot = plt.contourf(x, y, curl_v)
plt.streamplot(x, y, u, v, color="orange")
plt.colorbar()

plt.title("Oppgave e)")

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

Konturplottet av denne virvlingskomponenten kan finnes her. Observerer at strømningen skaper sirkulasjon mellom gass- og væskefasen, spesielt rundt det miderste rektangelet. Ser også at strømningen trekkes til veggen på grunn av friksjoni, og at det er større friksjon i væskefasen.

f)

```python
def line_integral(x1, y1, x2, y2):
    side1 = 0
    side2 = 0
    side3 = 0
    side4 = 0
    dx = 0.5
    dy = 0.5
    for k in u[y1, x1:x2+1]:
        side1 += k*dx
    for k in v[x2, y1:y2+1]:
        side2 += k*dy
    for k in u[y2, x1:x2+1]:
        side3 -= k*dx
    for k in v[x1, y1:y2+1]:
        side4 -= k*dy
    sumation = side1 + side2 + side3 + side4
    return sumation, side1, side2, side3, side4

def surface_integral(x1, y1, x2, y2):
    integral = 0
    dx = 0.5
    dy = 0.5
    for i in range(x1, x2+1):
        for j in range(y1, y2+1):
            integral += curl_v[j, i]*dx*dy
    return integral

line_integral_1, side1_1, side2_1, side3_1, side4_1 = line_integral(34, 159, 69, 169)
line_integral_2, side1_2, side2_2, side3_2, side4_2 = line_integral(34, 84, 69, 99)
line_integral_3, side1_3, side2_3, side3_3, side4_3 = line_integral(34, 49, 69, 59)

integral_1 = surface_integral(34, 159, 69, 169)
integral_2 = surface_integral(34, 84, 69, 99)
integral_3 = surface_integral(34, 49, 69, 59)

print(f"Sirkulasjonen til rektangel 1 ble {line_integral_1}")
print(f"Sirkulasjonen til rektangel 2 ble {line_integral_2}")
print(f"Sirkulasjonen til rektangel 3 ble {line_integral_3}")

print(f"""Rektangel 1 har side verdier: Side 1={side1_1},
      Side 2={side2_1}, Side 3={side3_1}, Side 4={side4_1}""")
print(f"""Rektangel 2 har side verdier: Side 1={side1_2},
      Side 2={side2_2}, Side 3={side3_2}, Side 4={side4_2}""")
print(f"""Rektangel 3 har side verdier: Side 1={side1_3},
      Side 2={side2_3}, Side 3={side3_3}, Side 4={side4_3}""")
```

```
Sirkulasjonen til rektangel 1 ble 1796.013421486005
Sirkulasjonen til rektangel 2 ble -60206.56400779366
Sirkulasjonen til rektangel 3 ble -143.18708039114364
Rektangel 1 har side verdier: Side 1=70100.52387861427,
Side 2=-100.99982042140701, Side 3=-68332.85609978675, Side 4=129.34546307988458
Rektangel 2 har side verdier: Side 1=198.47559740489203,
Side 2=919.0821556116496, Side 3=-61243.46477849595, Side 4=-80.65698231424645
Rektangel 3 har side verdier: Side 1=5133.347850903836,
Side 2=175.1650519009061, Side 3=-5410.039721925995, Side 4=-41.660261269891
```

Som nevnt i oppgave e så ser vi nå med hjelp av kurveintegral at sirkulasjon er størst i det midterste rektangelet og minst i det nederste.

g)

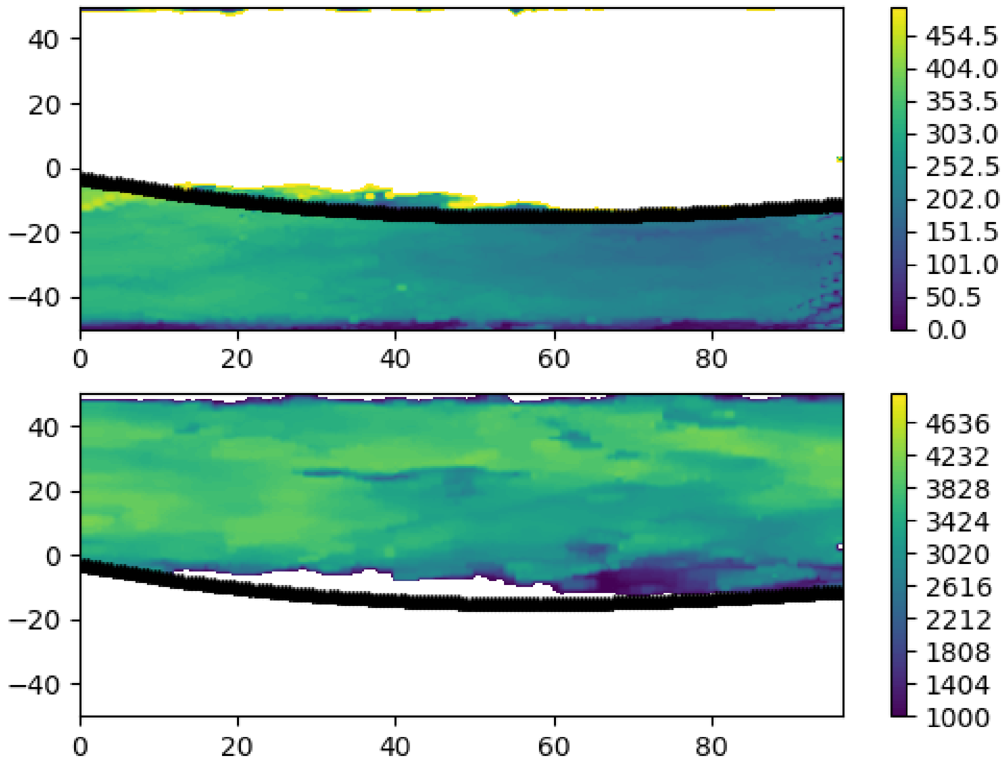
```python
def gauss(x1, y1, x2, y2):
    side1 = 0
    side2 = 0
    side3 = 0
    side4 = 0
    dx = 0.5
    dy = 0.5
    dz = 1

    for k in v[y1, x1:x2+1]:
        side1 -= k*dx*dz

    for k in u[x2, y1:y2+1]:
        side2 += k*dy*dz

    for k in v[y2, x1:x2+1]:
        side3 += k*dx*dz

    for k in u[x1, y1:y2+1]:
        side4 -= k*dy*dz

    sumation = side1 + side2 + side3 + side4
    return sumation, side1, side2, side3, side4

gauss_1, gauss_side1_1, gauss_side2_1, gauss_side3_1, gauss_side4_1 = gauss(34, 159, 69, 169)
gauss_2, gauss_side1_2, gauss_side2_2, gauss_side3_2, gauss_side4_2 = gauss(34, 84, 69, 99)
gauss_3, gauss_side1_3, gauss_side2_3, gauss_side3_3, gauss_side4_3 = gauss(34, 49, 69, 59)

print(f"Fluksen til kurveintegral 1 ble {gauss_1}")
print(f"Fluksen til kurveintegral 2 ble {gauss_2}")
print(f"Fluksen til kurveintegral 3 ble {gauss_3}")

print(f"""Rektangel 1 har side verdier: Side 1={gauss_side1_1},
    Side 2={gauss_side2_1}, Side 3={gauss_side3_1}, Side 4={gauss_side4_1}""")
print(f"""Rektangel 2 har side verdier: Side 1={gauss_side1_2},
    Side 2={gauss_side2_2}, Side 3={gauss_side3_2}, Side 4={gauss_side4_2}""")
print(f"""Rektangel 3 har side verdier: Side 1={gauss_side1_3},
    Side 2={gauss_side2_3}, Side 3={gauss_side3_3}, Side 4={gauss_side4_3}""")
```

```
Fluksen til kurveintegral 1 ble 104.8526049082102
Fluksen til kurveintegral 2 ble -6476.93918209796
Fluksen til kurveintegral 3 ble -124.56866604496236
Rektangel 1 har side verdier: Side 1=1556.8679439413959,
Side 2=21664.567474322168, Side 3=-2059.6771847938708, Side 4=-21056.905628561482
Rektangel 2 har side verdier: Side 1=-5187.564033067891,
Side 2=14782.532896182345, Side 3=-4074.0522144394345, Side 4=-11997.85583077298
Rektangel 3 har side verdier: Side 1=-195.5701479258336,
Side 2=1536.8217966413547, Side 3=284.9436464350764, Side 4=-1750.7639611955597
```

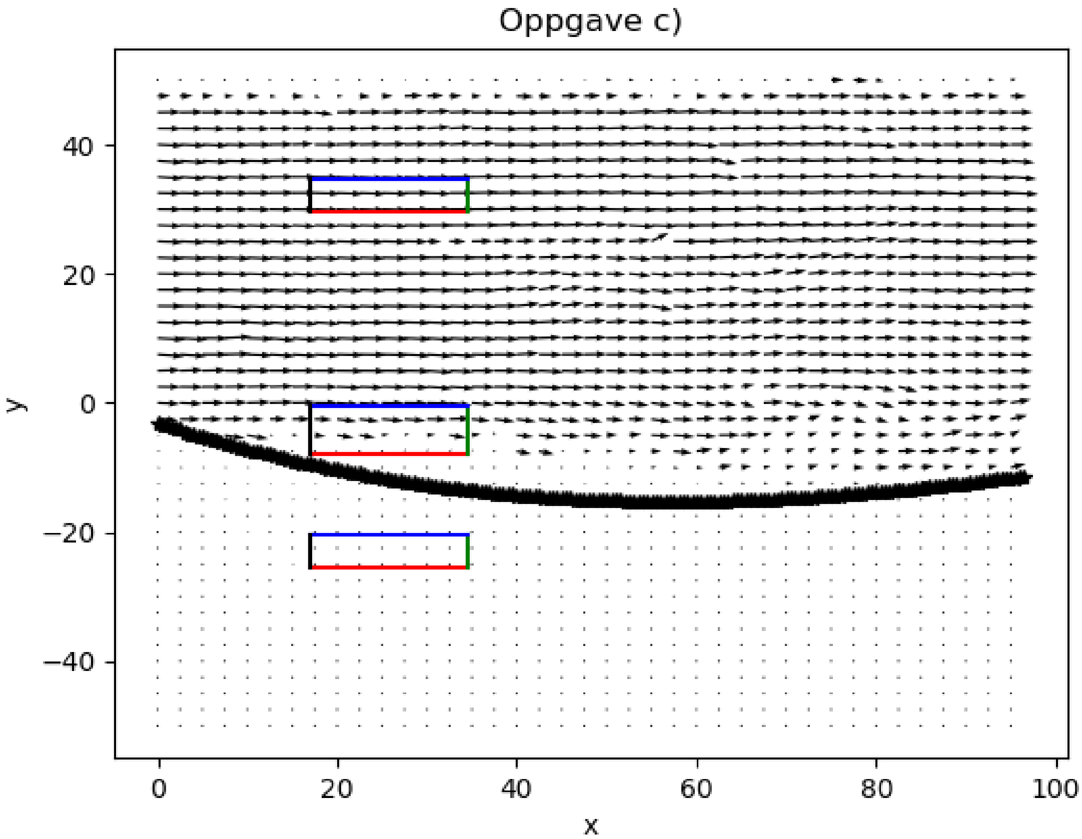
Anvender gauss sats så ser vi at fluksen i z-retning er like stor bare motsatt retning. Disse resultatene ser fornuftige ut siden det er side 2 og 4 som har størst verdi.

Plots

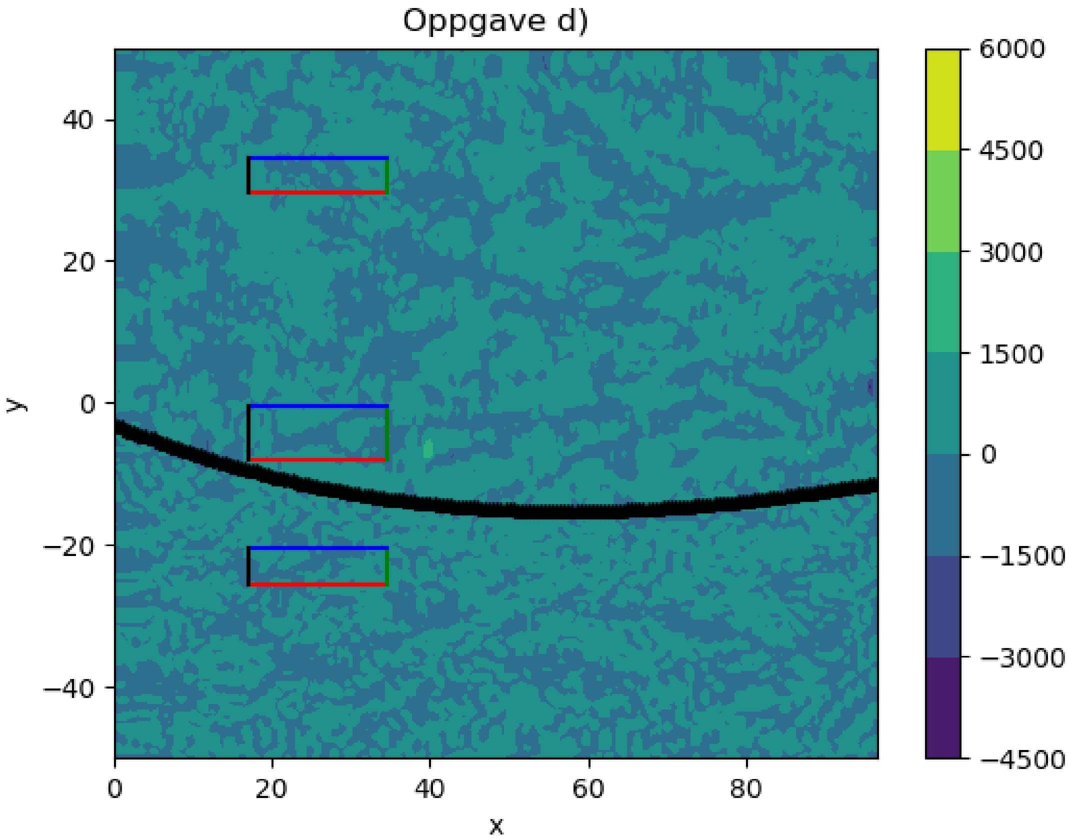
Figur 1: Graf til oppgave b



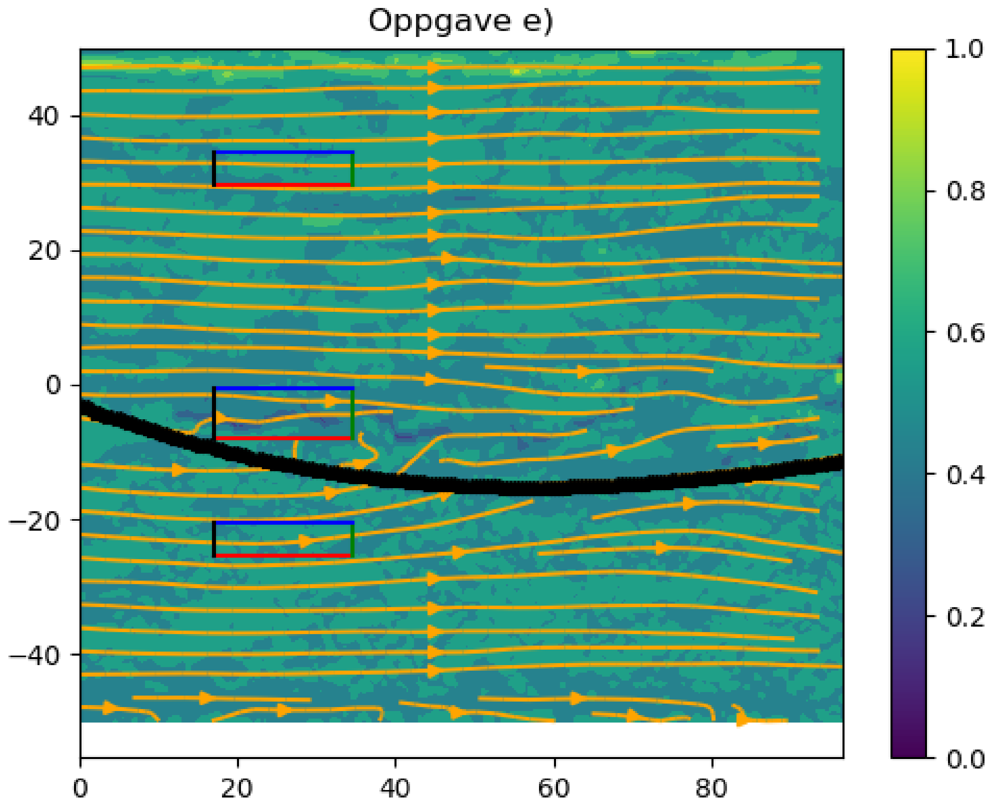
Figur 2: Graf til oppgave c



Figur 3: Graf til oppgave d



Figur 4: Graf til oppgave e



1 Appendiks

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

#Oppgave a
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(f"x shape er {np.shape(x)}")
print(f"y shape er {np.shape(y)}")
print(f"u shape er {np.shape(u)}")
print(f"v shape er {np.shape(v)}")
print(f"xit shape er {np.shape(xit)}")
print(f"yit shape er {np.shape(yit)}")

print(x)
print(y)

#Opggave b
velocity = np.sqrt(u**2 + v**2)

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

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

plt.savefig("oppgave_b.png")
plt.show()

#Oppgave c
def rectangle(x1, x2, y1, y2):
    position1 = (x[x2, x1], y[x2, y1])
    position2 = (x[y2, y1], y[y2, y1])

    # Bottom
    plt.plot([position1[0], position2[0]], [position1[1], position1[1]], "r")

    # Righ
    plt.plot([position2[0], position2[0]], [position1[1], position2[1]], "g")

    # Top
    plt.plot([position1[0], position2[0]], [position2[1], position2[1]], "b")

    # Left
    plt.plot([position1[0], position1[0]], [position1[1], position2[1]], "k")

def draw_rectangles():
    rectangle1_values = [34, 159, 69, 169]
    rectangle(rectangle1_values[0], rectangle1_values[1],
              rectangle1_values[2], rectangle1_values[3])

    rectangle2_values = [34, 84, 69, 99]
    rectangle(rectangle2_values[0], rectangle2_values[1],
              rectangle2_values[2], rectangle2_values[3])

    rectangle3_values = [34, 49, 69, 59]
    rectangle(rectangle3_values[0], rectangle3_values[1],
              rectangle3_values[2], rectangle3_values[3])
```

```python


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

plt.title("Oppgave c)")
plt.xlabel("x")
plt.ylabel("y")

plt.savefig("oppgave_c.png")

#Oppgave d

dudx = np.gradient(u, 0.5, axis=1)
dvdy = np.gradient(v, 0.5, axis=0)

divergence = dudx + dvdy
print(f"Divergensen er {divergence}")

plt.contourf(x, y, divergence)
plt.colorbar()
plt.title("Oppgave d)")
plt.savefig("oppgave_d.png")
plt.close()

#Oppgave e
dudy = np.gradient(u, 0.5, axis=0)
dvdx = np.gradient(v, 0.5, axis=1)

curl_v = dvdx - dudy

curl_plot = plt.contourf(x, y, curl_v)
plt.streamplot(x, y, u, v, color="orange")
plt.colorbar()

plt.title("Oppgave e)")

plt.savefig("oppgave_e.png")
plt.show()

#Oppgave f

def line_integral(x1, y1, x2, y2):
    side1 = 0
    side2 = 0
    side3 = 0
    side4 = 0
    dx = 0.5
    dy = 0.5

    for k in u[y1, x1:x2+1]:
        side1 += k*dx

    for k in v[x2, y1:y2+1]:
        side2 += k*dy

    for k in u[y2, x1:x2+1]:
        side3 -= k*dx

    for k in v[x1, y1:y2+1]:
        side4 -= k*dy

    sumation = side1 + side2 + side3 + side4
    return sumation, side1, side2, side3, side4


def surface_integral(x1, y1, x2, y2):
    integral = 0
    dx = 0.5
    dy = 0.5
```

```python
    for i in range(x1, x2+1):
        for j in range(y1, y2+1):
            integral += curl_v[j, i]*dx*dy

    return integral


line_integral_1, side1_1, side2_1, side3_1, side4_1 = line_integral(34, 159, 69, 169)
line_integral_2, side1_2, side2_2, side3_2, side4_2 = line_integral(34, 84, 69, 99)
line_integral_3, side1_3, side2_3, side3_3, side4_3 = line_integral(34, 49, 69, 59)

integral_1 = surface_integral(34, 159, 69, 169)
integral_2 = surface_integral(34, 84, 69, 99)
integral_3 = surface_integral(34, 49, 69, 159)

print(f"Sirkulasjonen til rektangel 1 ble {line_integral_1}")
print(f"Sirkulasjonen til rektangel 2 ble {line_integral_2}")
print(f"Sirkulasjonen til rektangel 3 ble {line_integral_3}")

print(f"""Rektangel 1 har side verdier: Side 1={side1_1},
      Side 2={side2_1}, Side 3={side3_1}, Side 4={side4_1}""")
print(f"""Rektangel 2 har side verdier: Side 1={side1_2},
      Side 2={side2_2}, Side 3={side3_2}, Side 4={side4_2}""")
print(f"""Rektangel 3 har side verdier: Side 1={side1_3},
      Side 2={side2_3}, Side 3={side3_3}, Side 4={side4_3}""")

#Oppgave g
def gauss(x1, y1, x2, y2):
    side1 = 0
    side2 = 0
    side3 = 0
    side4 = 0
    dx = 0.5
    dy = 0.5

    for k in v[y1, x1:x2+1]:
        side1 -= k*dx

    for k in u[y1:y2+1, x2]:
        side2 += k*dy

    for k in v[y2, x1:x2+1]:
        side3 += k*dx

    for k in u[y1:y2+1, x1]:
        side4 -= k*dy

    sumation = side1 + side2 + side3 + side4
    return sumation, side1, side2, side3, side4


gauss_1, gauss_side1_1, gauss_side2_1, gauss_side3_1, gauss_side4_1 = gauss(34, 159, 69, 169)
gauss_2, gauss_side1_2, gauss_side2_2, gauss_side3_2, gauss_side4_2 = gauss(34, 84, 69, 99)
gauss_3, gauss_side1_3, gauss_side2_3, gauss_side3_3, gauss_side4_3 = gauss(34, 49, 69, 59)

print(f"Fluksen til kurveintegral 1 ble {gauss_1}")
print(f"Fluksen til kurveintegral 2 ble {gauss_2}")
print(f"Fluksen til kurveintegral 3 ble {gauss_3}")

print(f"""Rektangel 1 har side verdier: Side 1={gauss_side1_1},
      Side 2={gauss_side2_1}, Side 3={gauss_side3_1}, Side 4={gauss_side4_1}""")
print(f"""Rektangel 2 har side verdier: Side 1={gauss_side1_2},
      Side 2={gauss_side2_2}, Side 3={gauss_side3_2}, Side 4={gauss_side4_2}""")
print(f"""Rektangel 3 har side verdier: Side 1={gauss_side1_3},
      Side 2={gauss_side2_3}, Side 3={gauss_side3_3}, Side 4={gauss_side4_3}""")
```