

MEK 1100

Obligatorisk oppgave 2

Jonas Semprini Næss

4. mai 2020

a.)

Vi er gitt fire matriser og to vektorer, henholdsvis navngitt X, Y, U, V og XIT, YIT .

Videre ønsker vi å vise at hver matrise har 194 punkter i x -retning og 201 punkter i y -retning. Dette kan bevises ved å bruke `*array*.shape` funksjonen i python (jmf. linje 16 - 19 i kildekoden), hvilket gir følgende print.

```
(201, 194)
(201, 194)
(201, 194)
(201, 194)
```

Viktig bemerkning er at python bytter om indeksering på aksene slik at y -aksen får nullte indeks og x -aksen får første indeks.

Tilleggsvis skal dimensjonen på vektorene XIT, YIT sjekkes likeledes, og gir (jmf. linje 20 - 21 i kildekoden).

```
(1, 194)
(1, 194)
```

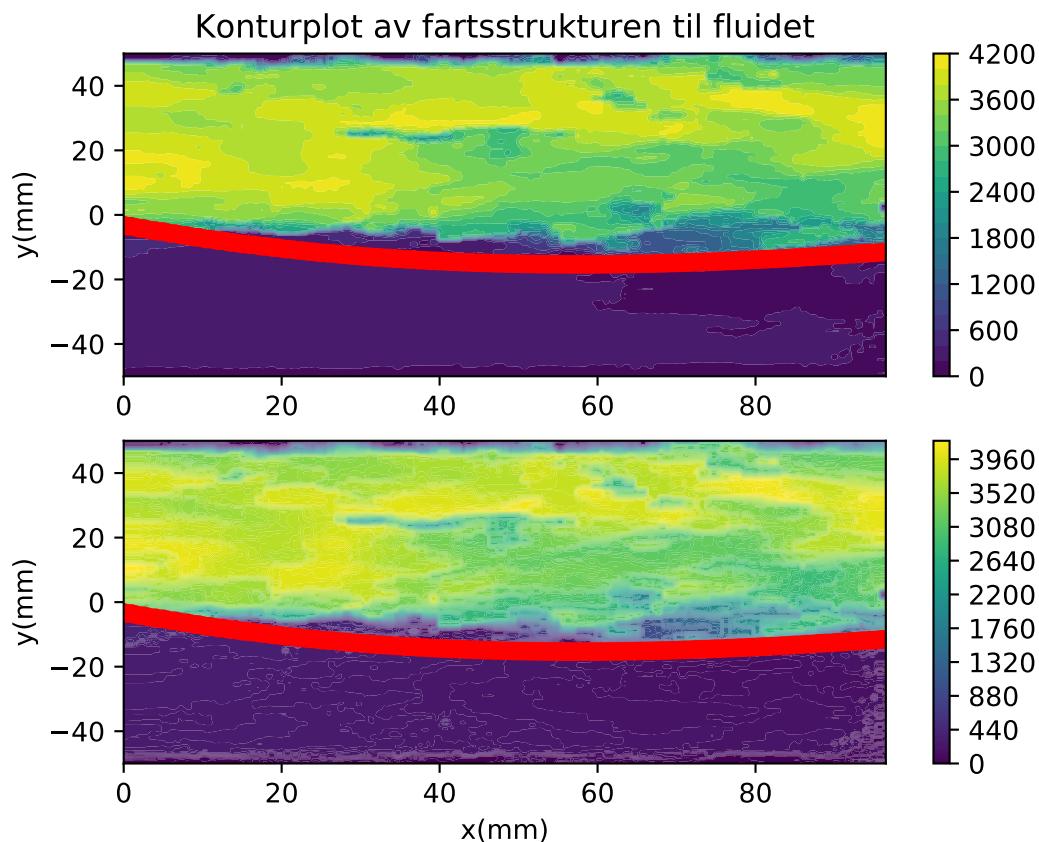
Hvilket viser at vektorene har 194 punkter i x -retning.

For å sjekke at griddet er regulært med 0.5 mm intervall i begge retninger innfører vi en testfunksjon (jmf. linje 25 - 34 i kildekoden). Hvis $\Delta x \neq 0.5$ vil vi få en assert feil som forteller at et intervall i matrisen ikke oppfyller kriteriet.

Videre har vi også valgt å transponere Y -matrisen slik at hver array har verdier $\Delta y = 100$. Dette testes på lik måte ved hjelp av en assert statement.

b.)

I denne del-oppgaven ønsker vi å undersøke strømhastigheten til hastighetfeltet ved hjelp av et konturplot (jmf. linje 52 - 71 i kildekoden). Det gir følgende bilde.

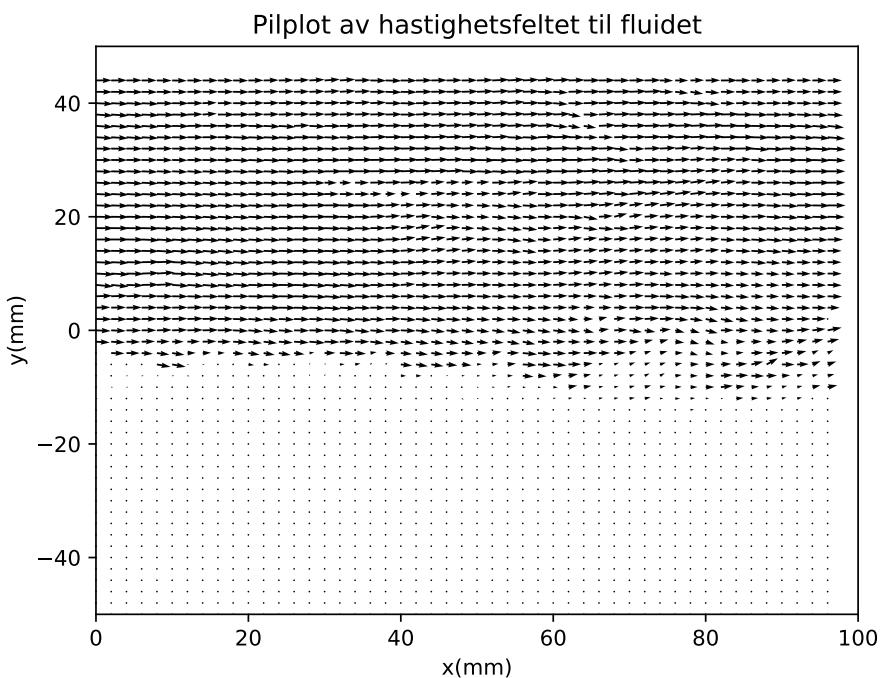


Konturplot av strømhastigheten $\xi = \|\mathbf{v}\| = \sqrt{u^2 + v^2}$

Her har vi delt opp plotet i to deler slik at man lettere kan se fargefordelingen, og skilleflaten som er markert med røde prikker.

c.)

Når vi nå skal plote et pilplot av hastighetsfeltet $\mathbf{v} = u\mathbf{i} + v\mathbf{j} + w\mathbf{k}$ bruker vi kommandoen `plt.quiver`, hvor antall inngangsverdier er regulert. I dette eksempelet plotter vi kun hver fjerde pil (jmf. linje 73 - 87 i kildekoden).

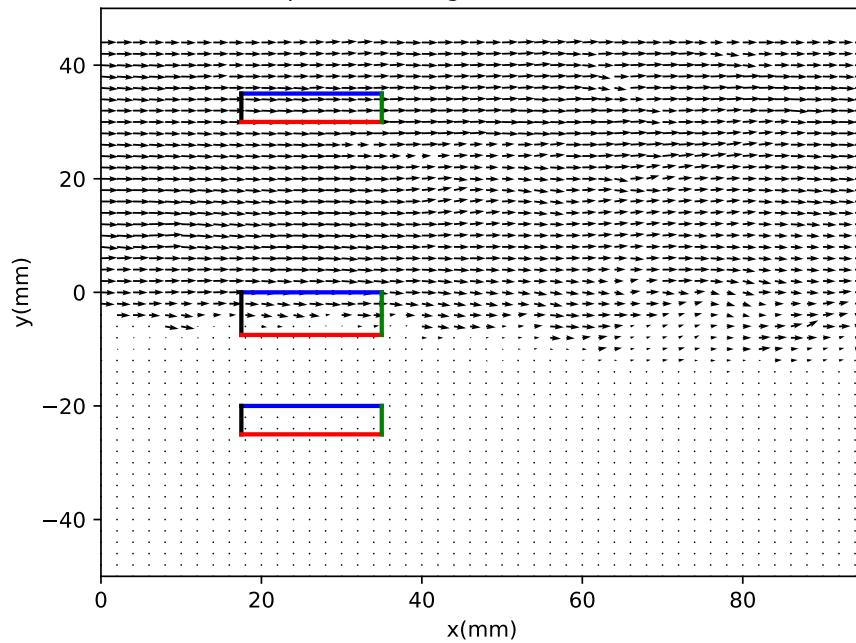


Pilplot av hastighetsfeltet \mathbf{v}

Vi ser at store deler av pilene observeres som punkter i nedre halvdel av plotet. Dette skyldes at hastigheten er markant større i luft enn vann.

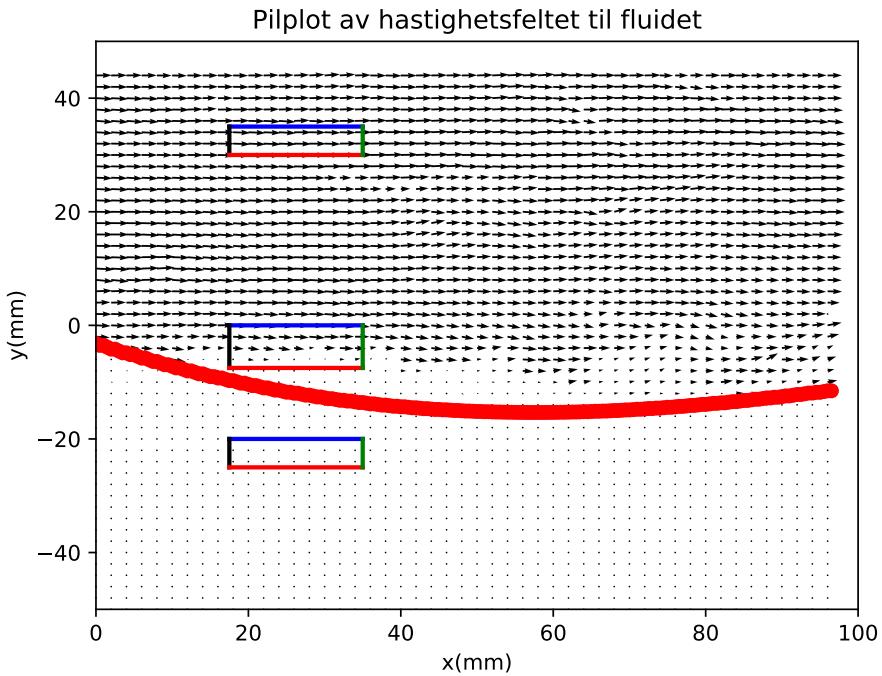
Videre i oppgaven skal vi tegne inn tre rektangler hvorav to befinner seg i gassfasen og et ligger i væskefasen. Dette kan gjøres som vist i funksjonen `def square()`, i kildekoden (linje 89 - 105). Det gir følgende bilde

Pilplot av hastighetsfeltet til fluidet



Pilplot av hastighetsfeltet \mathbf{v} med rektanger

Tegner vi også med skilleflaten,



Pilplot av hastighetsfeltet \mathbf{v} med rektanger, og skilleflate.

ser vi enda tydeligere at hastigheten mellom luft og vann utgjør en stor forskjell.

d.)

Divergensen til et hastighetsfelt er gitt ved

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

men etter hvordan vi betrakter forsøket er $\frac{\partial}{\partial z} w = 0$. Tilleggsvis ser vi på strømingen som inkompressibel, hvilket betyr at et vilkårlig lite utdrag av fluidet ikke vil endre tetthet over tid i strømningen gjennom et vilkårlig volum. Da har vi at

$$\frac{D\rho}{dt} = 0$$

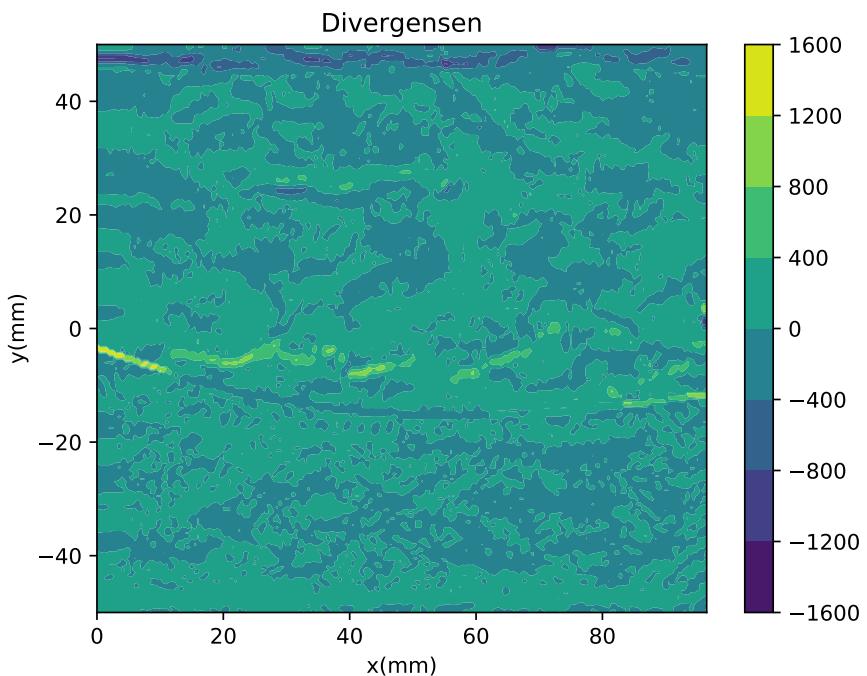
som helhetlig i kontinuitetslikningen gir

$$\frac{D\rho}{dt} + \rho \nabla \cdot \mathbf{v} = 0$$

hvilket betyr at $\nabla \cdot \mathbf{v} = 0$ for et inkompressibelt fluid.

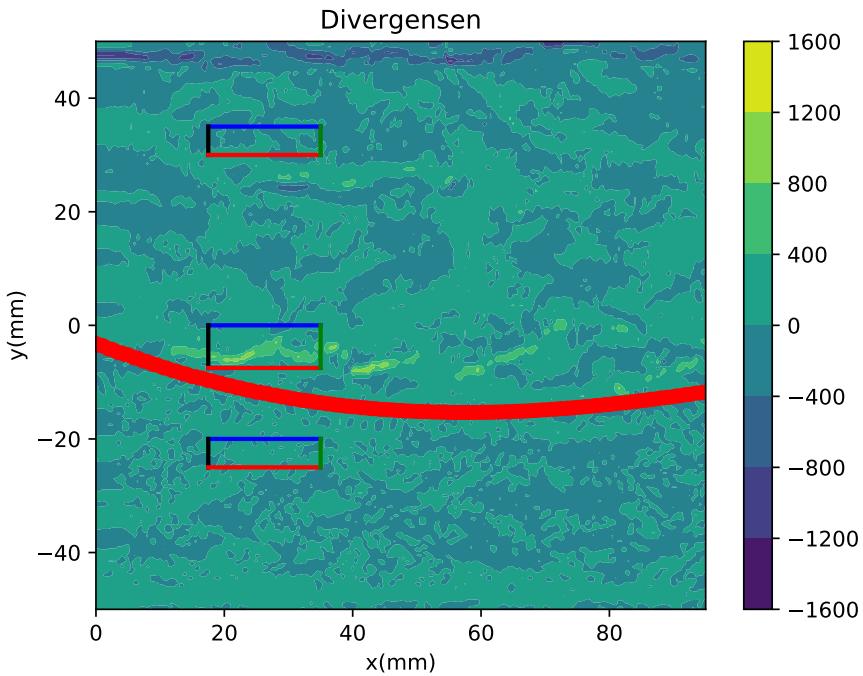
Ser vi på det praktiske eksemplet kan vi simulere tankegangen vist i kildekoden (fra linje 108 - 121). Det er viktig å merke seg at divergensen vil være forskjellig fra null i dette tilfellet på grunn av måten vi tolker eksperimentet på (som betyr at vi kun ser på hastighetsfeltet i xy -retning, dermed uten z -orientering).

Et konturplot av divergensen, kan se noe slik ut



Konturplot av divergensen $\nabla \cdot \mathbf{v}$.

hvor vi har brukt funksjonen `def divergens()` likeledes som i utregningen. Tegner vi videre inn rektanglene og skilleflaten



Konturplot av divergensen med rektangler og skilleflate $\nabla \cdot \mathbf{v}$.

Siden divergensen i teorien er lik null kan vi skrive z-komponenten som

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

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

hvilket i følge forsøket gir $\frac{\partial}{\partial z} w = -\nabla \cdot \mathbf{v}$.

e.)

Virvingen til et hastighetsfelt er gitt ved

$$\nabla \times \mathbf{v} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ u & v & w \end{vmatrix}$$

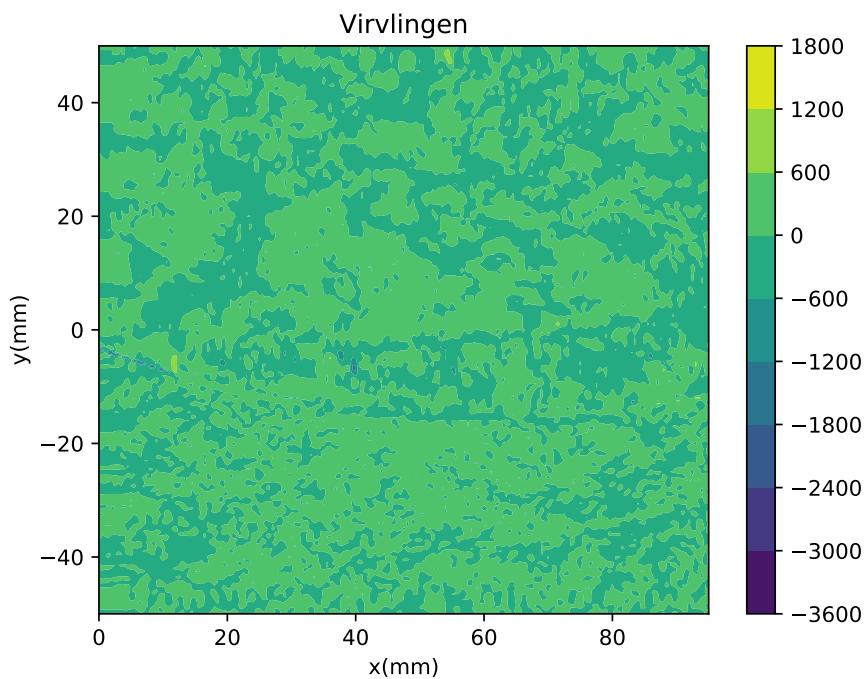
som gir

$$\mathbf{k} \left(\frac{\partial}{\partial x} v - \frac{\partial}{\partial y} u \right)$$

lager vi for ordenhetensskyld et nytt hastighetsfelt $\xi = vi - uj$ får vi at komponenten normalt på xy -planet

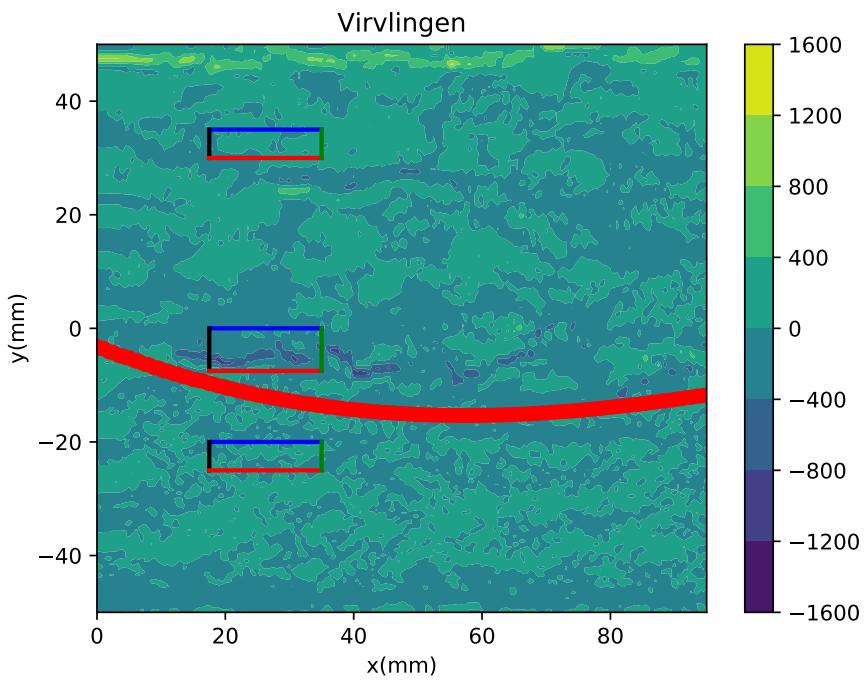
$$\mathbf{k} \left(\frac{\partial}{\partial x} v - \frac{\partial}{\partial y} u \right) = \mathbf{k} \nabla \cdot \boldsymbol{\xi}$$

Videre ønsker vi å lage et plot av virvlingen, som vist i funksjonen `def virveling()` (jmf. linje 123 - 138). Da får vi følgende bilde



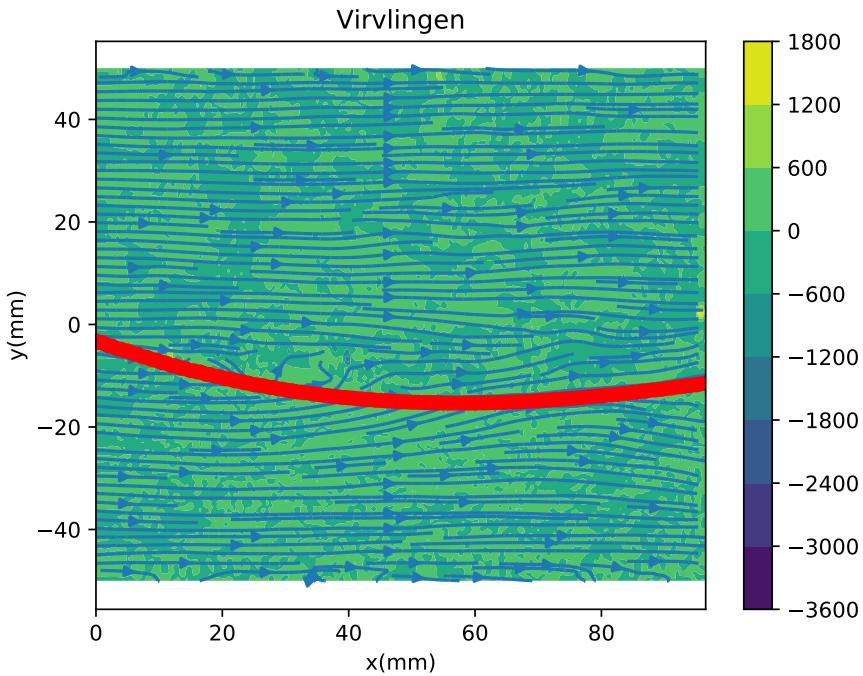
Konturplot av virvlingen $\nabla \times \mathbf{v}$.

legger vi til rektanglene og skilleflaten



Konturplot av virvlingen $\nabla \times \mathbf{v}$, med rektangler og skilleflate.

Utvider vi plotet ved å legge til strømlinjene for hastighetsfeltet får vi



Konturplot av virvlingen $\nabla \times \mathbf{v}$, med strømlinjer og skilleflate.

f.)

Kurveintegral:

For å beregne sirkulasjonen direkte som et kurveintegral gjennom et rektangulært tversnitt deler vi opp kurveintegralet i fire intervaller

$$\oint_D \mathbf{v} \cdot d\mathbf{r} = \sum_{i=1}^j \int_{D_i} \mathbf{v} \cdot d\mathbf{r}$$

for $j = 4$ og der $d\mathbf{r} = dx\mathbf{i} + dy\mathbf{j}$. Hjørnene i rektangelet $(x_0, y_0) \wedge (x_1, y_1)$ utgjør delintervallene vi integrerer over.

$$\oint_D \mathbf{v} \cdot d\mathbf{r} = \int_{x_0}^{x_1} \mathbf{v} \cdot dx\mathbf{i} + \int_{y_0}^{y_1} \mathbf{v} \cdot dy\mathbf{j} - \int_{x_1}^{x_0} \mathbf{v} \cdot dx\mathbf{i} - \int_{y_1}^{y_0} \mathbf{v} \cdot dy\mathbf{j}$$

Simulerer vi tankegangen gjennom et script må vi integrere numerisk. I dette tilfellet velger vi å bruke trapesmetoden som tilnærming til det fullstendige integralet (jmf. linje 140 - 159 i kildekoden). Det gir verdiene

Rektangel 1, kurveintegral: 3095.20
Rektangel 2, kurveintegral: -58924.49
Rektangel 3, kurveintegral: -46.59

Flateintegral:

En annen metode for å beregne sirkulasjonen er å regne det ut indirekte som et flateintegral. Vi velger derfor i denne oppgaven å benytte Stokes sats.

$$\oint_D \mathbf{v} \cdot d\mathbf{r} = \iint_S \nabla \times \mathbf{v} \, dS$$

Her transformerer vi sirkulasjonsintegralet til et dobbeltintegral av virvingen til hastighetsfeltet over en overflate S . Grensene på integralet vil likt som i eksempelet for kurveintegralet være hjørnene i rektangelet.

Implementering av metoden vist i linje 167-177 i kildekoden, indikerer at man må ta en numerisk tilnærming av integralet

$$\iint_S \nabla \times \mathbf{v} \, dS \approx \sum_{i=x0}^{x1} \sum_{j=y0}^{y1} \nabla \times \mathbf{v}_{(i,j)} \Delta x \Delta y$$

som ved gjennomkjøring gir verdiene

Rektangel 1, flateintegral: 3668.34
Rektangel 2, flateintegral: -61565.49
Rektangel 3, flateintegral: -44.41

Sammenligner vi verdiene med kurveintegralet ser vi at avviket er størst for rektangel 1, men jevner seg ut for rektangel 2 og 3. Dette har med at rektangel 1 befinner seg fast i gassfasen, hvor strømning er rask og ukontrollert. Videre befinner rektangel 2 seg i grensen mellom gass og væske hvilket betyr at strømning delvis er mer kontrollert og jevnt fordelt. Dermed sier det seg selv at presisjonen er best for rektangel 3 ettersom at hele rektangelet befinner seg i væskefasen.

Dessuten er griddet regulert slik at man ikke får et helhetlig bilde av strømningen som fører til presisjonsfeil.

Ser vi nærmere på sirkulasjonen rundt sidene på rektangelene har vi verdiene.

Rektangel 1: 68324.97915, -66076.77821, 245.22033,
601.77503)

Rektangel 2: (772.42854, -59952.63240, 109.89303, 145.82570)

Rektangel 3: (4976.79022, -5284.32880 198.71382, 62.23635)

Det er tilsynelatende større forskjeller på verdiene langs både bredden og høyden i rektangel 1 enn antatt, med tanke på at pilplotet gir et bilde av en mer eller mindre “kontinuerlig” strømning i det området. Dermed blir det avvik i både positiv og negativ xy -retning, ut fra en formodning om at strømningen er mer eller mindre lik i begge retninger (hvilket i teorien ville gitt en summering til null).

For rektangel 2 er verdiene mer variert, hvilket gir mening når man ser på hvor i plotet sidene befinner seg. De to nederste sidene i x og y retning (rød, grønn) er tidvis i en mellomfase av fluid og gass som logisk vil gi en variert strømning langs sidene.

Dermed er det naturlig at verdiene for rektangel 3 har størst presisjon, grunnet at strømningen beveger seg parallelt med sidene på rektangelet. Dette er vanskelig å se direkte ut i fra pilplotet, men verdiene indikerer at strømningen langs sidene er av “relativt” likt omfang.

g.)

Gauss’ sats sier at

$$\oint_S \mathbf{v} \cdot d\mathbf{S} = \int_V \nabla \cdot \mathbf{v} \, dV$$

hvor S er den flaten som omgir volumet V . Siden vi ønsker å beregne det lukkede kurveintegralet direkte, bruker vi en lignende teknikk som i oppgave e.) og utvider uttrykket

$$\begin{aligned} \int_V \nabla \cdot \mathbf{v} \, dV &= \oint_S (\mathbf{v} \cdot \mathbf{n}) \, dS \\ \int_V \nabla \cdot \mathbf{v} \, dV &= \int_{S_1} v \cdot \mathbf{n} \, dx + \int_{S_2} u \cdot \mathbf{n} \, dy - \int_{S_3} v \cdot \mathbf{n} \, dx - \int_{S_4} u \cdot \mathbf{n} \, dy \end{aligned}$$

Her gjenspeiler fortegnet på integralet til hvilken retning normalvektoren til overflateelementet $S_1 \dots S_4$ peker.

Implementer vi tanken i et script får vi følgende output (jmf. linje 183 - 205 i kildekoden).

```
Rektangel 1, integrert fluks: 1647.88
Rektangel 2, integrert fluks: -5123.52
Rektangel 3, integrert fluks: -211.25
```

Hvor verdiene svarer til forholdet mellom brutto og netto-innstrømning gjennom flateelementet (altså negativ fluks betyr netto > brutto, og vice versa).

Når det kommer til hvilken implikasjon dette har for rektanglene orientert i z -retning, må vi se på hva divergensen for det totale hastighetsfeltet er. Det

er antat at vi arbeider med et inkrompressibelt hastighetsfelt hvilket betyr at $\nabla \cdot \mathbf{v} = 0 \Rightarrow \int_V \nabla \cdot \mathbf{v} dV = 0$, som betyr at fluksen gjennom rektanglene(prismer) i z -retning må summere til null.

Sluttvis ønsker vi som i oppgave g.) å se på fluksen langs hver side i rektanglene. Det gir

```
Rektangel 1: (1614.3774, 19677.6893, 33.5013, -19677.6893)
Rektangel 2: (-5157.0236, 13726.6069, 33.5013, -13726.6069)
Rektangel 3: (-244.7548, 1409.1568, 33.5013, -1409.1568)
```

Her samsvarer verdiene nogenlunde med hastighetsfeltet i c.), hvor det opplagt er mest turbulent strømning gjennom rektangel 2. Imidlertid er det ikke heldekkende å kun se på hastighetsfeltet i sin helhet, men dersom man i tillegg ser på strømningslinjene fra e.) får man et tydeligere bilde av strømningen gjennom rektanglene.

Kildekode:

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import matplotlib.patches as patches
4 import scipy.io as sio
5 from scipy.integrate import *
6 # Dette virker med versjon 7 MAT-filer
7
8 data = sio.loadmat('data.mat')
9 x = data.get('x')
10 y = data.get('y')
11 u = data.get('u')
12 v = data.get('v')
13 xit = data.get('xit')
14 yit = data.get('yit')
15
16 print(x.shape)
17 print(y.shape)
18 print(u.shape)
19 print(v.shape)
20 print(xit.shape)
21 print(yit.shape)
22
23 xi = xit[-1]
24
25 def avstand(x, y):
26     yT = np.transpose(y)
27     grid = 0
28     omkrets = 0
29     for i in range(len(x)):
30         if (x[i] - x[i-1]) == 0.5:
31             grid = 0.5
```

```

32     if (yT[0, -1] - yT[0,0]) == 100:
33         omkrets = 100
34     return grid,omkrets
35
36 avstand(xi, y)
37
38
39
40 def test_grid():
41     avstand2 = 0.5
42     omkrets2 = 100
43     resultat1 = avstand(xi,y)[0]
44     resultat2 = avstand(xi,y)[1]
45     tol = 1e-7
46     success = abs(resultat1 - avstand2) < tol
47     success2 = abs(resultat2 - omkrets2) < tol
48     assert success, success2
49
50 test_grid()
51
52 def kontur_h(u, v, xit, yit):
53     plt.subplot(2,1,1)
54
55     fart = np.sqrt(u**2 + v**2)
56     CS = plt.contourf(x, y, fart, 20)
57     plt.colorbar()
58     plt.plot(xit, yit, 'ro')
59
60     plt.ylabel("y(mm)")
61     plt.title("Konturplot av fartsstrukturen til fluidet")
62
63     plt.subplot(2,1,2)
64
65     CS = plt.contourf(x, y, fart, 210)
66     plt.colorbar()
67     plt.plot(xit, yit, 'ro')
68     plt.xlabel("x(mm)")
69     plt.ylabel("y(mm)")
70     plt.savefig("konturplot2.pdf")
71     plt.show()
72
73 def pilplot(x, y, u, v):
74
75     x_start = y_start = 0
76     x_slutt = 191
77     y_slutt = 201
78     steg = 4
79
80     plt.quiver(x[x_start:x_slutt:steg], y[y_start:y_slutt:steg], y[x_start
81 :x_slutt:steg], y[y_start:y_slutt:steg], \
82         u[x_start:x_slutt:steg], v[y_start:y_slutt:steg], v[x_start:x_slutt:
83         steg], y[y_start:y_slutt:steg])
84
85     plt.xlabel('x(mm)')

```

```

84     plt.ylabel('y(mm)')
85     plt.title('Pilplot av hastighetsfeltet til fluidet')
86     #plt.savefig('pilplot2.pdf')
87     #plt.show()
88
89 def square(x, y, c0, c1):
90
91     x0 = x[c0[1]][c0[0]]
92     y0 = y[c0[1]][c0[0]]
93
94     x1 = x[c1[1]][c1[0]]
95     y1 = y[c1[1]][c1[0]]
96
97     plt.plot([x0, x1], [y1, y1], linewidth=2, color='b')
98     plt.plot([x0, x0], [y0, y1], linewidth=2, color='k')
99     plt.plot([x0, x1], [y0, y0], linewidth=2, color='r')
100    plt.plot([x1, x1], [y0, y1], linewidth=2, color='g')
101
102
103 square1 = square(x, y, (35, 160), (70, 170))
104 square2 = square(x, y, (35, 85), (70, 100))
105 square3 = square(x, y, (35, 50), (70, 60))
106
107
108 def divergens(u, v):
109
110     dudx = np.gradient(u, axis=0)
111     dudy = np.gradient(v, axis=1)
112     div = dudx + dudy
113
114     CS1 = plt.contourf(x, y, div)
115     plt.colorbar()
116     plt.xlabel('x(mm)')
117     plt.ylabel('y(mm)')
118     plt.title('Divergensen')
119     #plt.savefig('divergens.pdf')
120     #plt.show()
121     return div
122
123 def virvling(u,v):
124
125     dudy = np.gradient(u, axis=0)
126     dvdx = np.gradient(v, axis=1)
127     curlz = dvdx - dudy
128
129     plt.contourf(x, y, curlz)
130     plt.colorbar()
131     plt.streamplot(x,y, u, v, density=2)
132     plt.xlabel('x(mm)')
133     plt.ylabel('y(mm)')
134     plt.title('Virvlingen')
135     plt.savefig('virvling2.pdf')
136     #plt.show()
137

```

```

138     return curlz
139
140 def kurveintegral(c0, c1, c2, c3):
141
142     x0 = x[c1, c0]
143     y0 = y[c1, c0]
144
145     x1 = x[c3, c2]
146     y1 = y[c3, c2]
147
148     n_x = int(abs(x1 - x0)*2)
149     n_y = int(abs(y1 - y0)*2)
150
151     xn = np.linspace(x0, x1, n_x)
152     yn = np.linspace(y0, y1, n_y)
153
154     I = np.trapz(u[c1, c0:c2], xn)
155     I2 = np.trapz(-u[c3, c0:c2], xn)
156     I3 = np.trapz(v[c1:c3, c2], yn)
157     I4 = np.trapz(-v[c1:c3, c0], yn)
158
159     return I + I2 + I3 + I4
160
161
162 print(f"Rektangel 1, kurveintegral: {kurveintegral(35, 160, 70, 170):.2f}")
163 print(f"Rektangel 2, kurveintegral: {kurveintegral(35, 85, 70, 100):.2f}")
164 print(f"Rektangel 3, kurveintegral: {kurveintegral(35, 50, 70, 60):.2f}")
165
166
167 def flateintegral(c0, c1, c2, c3):
168
169     dudy = np.gradient(u, 0.5, axis=0)
170     dvdx = np.gradient(v, 0.5, axis=1)
171     curlz = dvdx - dudy
172     S = 0
173
174     for i in range(c0, c2 + 1):
175         for j in range(c1, c3 + 1):
176             S+= curlz[j, i]*0.5*0.5
177     return S
178
179 print(f"Rektangel 1, flateintegral: {flateintegral(35, 160, 70, 170):.2f}")
180 print(f"Rektangel 2, flateintegral: {flateintegral(35, 85, 70, 100):.2f}")
181 print(f"Rektangel 3, flateintegral: {flateintegral(35, 50, 70, 60):.2f}")
182
183 def integrertfluks(c0, c1, c2, c3):
184     x0 = x[c1, c0]
185     y0 = y[c1, c0]
186
187     x1 = x[c3, c2]
188     y1 = y[c3, c2]
189
190     n_x = int(abs(x1 - x0)*2)
191     n_y = int(abs(y1 - y0)*2)

```

```

192     xn = np.linspace(x0, x1, n_x)
193     yn = np.linspace(y0, y1, n_y)
194
195     I = - np.trapz(v[c1, c0:c2], xn)
196     I2 = np.trapz(u[c1:c3, c2], yn)
197     I3 = np.trapz(v[c0, c0:c2], xn)
198     I4 = - np.trapz(u[c1:c3, c2], yn)
199
200     return I + I2 + I3 + I4
201
202
203 print(f"Rektangel 1, integrert fluks: {integrertfluks(35,160, 70, 170):.2f}")
204 print(f"Rektangel 2, integrert fluks: {integrertfluks(35,85, 70, 100):.2f}")
205 print(f"Rektangel 3, integrert fluks: {integrertfluks(35,50, 70, 60):.2f}")
206
207
208 kontur_h(u, v, xit, yit)
209 pilplot(x, y, u, v)
210
211 print(divergens(u,v)[-1])
212 print(virvling(u,v)[-1])
213
214 plt.plot(xit, yit, 'ro')
215 virvling(u, v)
216
217 plt.xlim(0, 95)
218 plt.ylim(-50, 50)
219
220 plt.show()

```