

MEK1100 - Mandatory assignment 2

William Dugan

May 2, 2022

The full python code can be found at the end of the document.

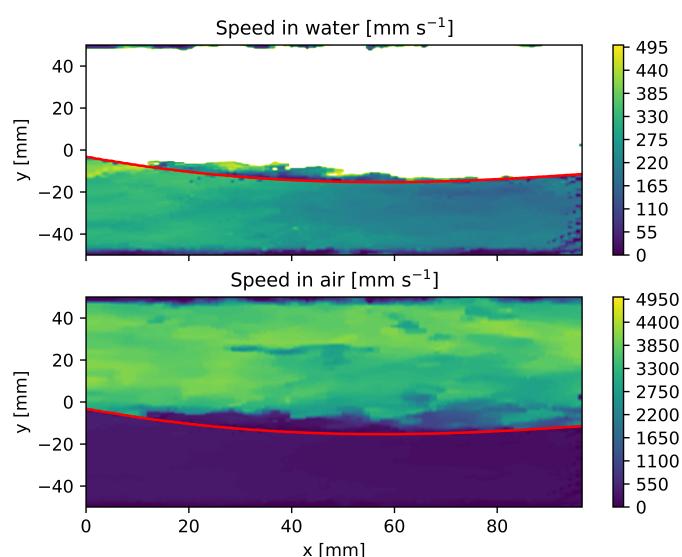


Figure 1: Contour plot of $\sqrt{u^2 + v^2}$. Red line indicating border between gas and liquid.

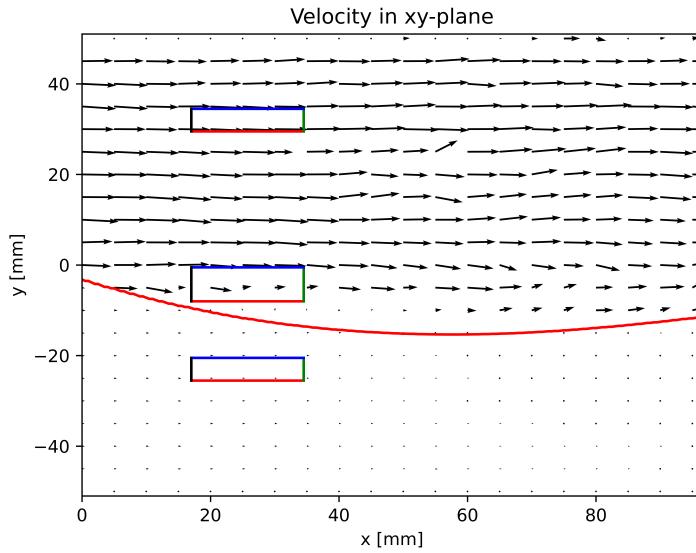


Figure 2: Quiver plot of velocity $ui + vj$. Red line indicating border between gas and liquid.

Let $\mathbf{v} = ui + vj + wk$ and $\mathbf{v}^* = ui + vj$. The divergence is

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

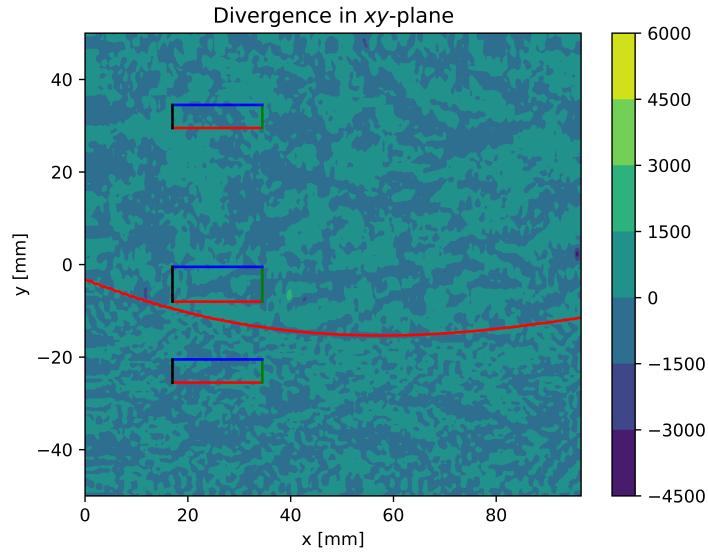


Figure 3: Contour plot of $\nabla \cdot \mathbf{v}^*$. Red line indicating border between gas and liquid.

Since both the gas and the liquid are modeled as incompressible fluids, the divergence should be zero. This means that

$$\begin{aligned}\nabla \cdot \mathbf{v} &= \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0 \\ \implies \frac{\partial w}{\partial z} &= -\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) \\ \implies w &= -\int \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) dz.\end{aligned}$$

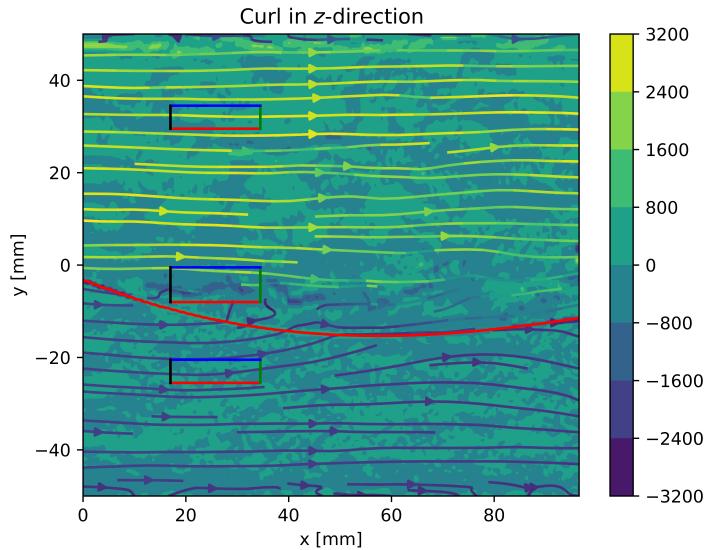


Figure 4: Contour plot of $(\nabla \times \mathbf{v}^*) \cdot \mathbf{k}$. Red line indicating border between gas and liquid.

BLABLA forklaring oppg E - Beskriv strømningen for både gassfasen og væskefasen. Legg spesielt merke til strømningen ved veggen.

The code in `integration.py` yields the following result:

```
-----
Rectangle 1

Circulation, line integral:      2695.51
Circulation side 1:             70100.52
Circulation side 2:             266.27
Circulation side 3:             -68332.86
Circulation side 4:             661.57

Circulation, surface integral:   2621.56
Percentage error:                2.74 %

Integrated flux:                 104.85
Flux side 1:                     1556.87
Flux side 2:                     21664.57
Flux side 3:                     -2059.68
Flux side 4:                     -21056.91
-----
Rectangle 2

Circulation, line integral:      -60976.60
Circulation side 1:              198.48
Circulation side 2:              300.22
Circulation side 3:              -61243.46
Circulation side 4:              -231.83

Circulation, surface integral:   -61482.54
Percentage error:                -0.83 %

Integrated flux:                 -6476.94
Flux side 1:                     -5187.56
Flux side 2:                     14782.53
Flux side 3:                     -4074.05
Flux side 4:                     -11997.86
-----
Rectangle 3

Circulation, line integral:      9.52
Circulation side 1:              5133.35
Circulation side 2:              207.91
Circulation side 3:              -5410.04
Circulation side 4:              78.30

Circulation, surface integral:   -12.21
Percentage error:                228.29 %

Integrated flux:                 -124.57
Flux side 1:                     -195.57
Flux side 2:                     1536.82
Flux side 3:                     284.94
Flux side 4:                     -1750.76
-----
```

BLABLA forklaring oppg F - Diskuter forskjellen mellom de tre rektanglene. - For å få en bedre forståelse av sirkulasjonen angir du verdien til kurveintegralene langs hver side av rektanglene. Passer disse resultatene med hva du hadde forventet når du ser på hastighetsfeltet i området?

BLABLA forklaring oppg G - Diskuter hvilken implikasjon disse svarene har for den integrerte fluksen over rektanglene orientert i z-retning - For å få en bedre forståelse av fluksen skal du angi verdien til kurveintegralene langs hver side av rektanglene. Passer disse resultatene med hva du hadde forventet når du ser på hastighetsfeltet i området

File: plots.py

```
1 from matplotlib.patches import Rectangle
2 import matplotlib.pyplot as plt
3 import scipy.io
4 import numpy as np
5
6 # Reading data
7 f = scipy.io.loadmat('data.mat')
8 x, y, u, v, xit, yit = (f[index].transpose() for index in ['x', 'y', 'u', 'v', 'xit', 'yit'])
9
10 # Checking dimension of data
11 for arr in (x, y, u, v):
12     assert arr.shape == (194, 201), 'Wrong dimension'
13
14 for arr in (xit, yit):
15     assert arr.shape == (194, 1), 'Wrong dimension'
16
17 assert (y[0,1]-y[0,0], x[1,0]-x[0,0]) == (0.5, 0.5), 'Wrong dx or dy'
18
19 assert (y[-1,0], y[-1,-1]) == (-50.0, 50.0), 'Wrong range of y'
20
21 # Indices of rectangles
22 indicies = ((34, 159, 69, 169), (34, 84, 69, 99), (34, 49, 69, 59))
23
24 def draw_rectangles(
25     indicies: tuple[tuple[int, int, int, int], ...],
26     x: np.ndarray,
27     y: np.ndarray,
28     ax: plt.Axes
29 ) -> None:
30     """
31         Draws the rectangles with lower left corner (x[x1,y1], y[x1,y1])
32         and upper right corner (x[x2,y2], y[x2,y2]).
33
34     Parameters
35     -----
36     indicies : tuple[tuple[int, int, int, int], ...]
37         Nested n-tuple with indicies for coordinates
38     x : np.ndarray
39         x component of xy-meshgrid
40     y : np.ndarray
41         y component of xy-meshgrid
42     ax : plt.Axes
43         Axes object where the rectangle is drawn.
44     """
45     for x1, y1, x2, y2 in indicies:
46         x1, y1, x2, y2 = x[x1,y1], y[x1,y1], x[x2,y2], y[x2,y2]
47         ax.plot([x1, x2], [y1, y1], color='r')
48         ax.plot([x2, x2], [y1, y2], color='g')
49         ax.plot([x1, x2], [y2, y2], color='b')
50         ax.plot([x1, x1], [y1, y2], color='k')
51
52 def main():
53     # Plotting speed as contourplot
54     v_mag = np.sqrt(u**2 + v**2)
55
56     fig, ax = plt.subplots(2, 1, sharex=True, sharey=True)
57
58     clines = np.linspace(0, 500, 101)
59     cs = ax[0].contourf(x, y, v_mag, clines, vmin=0)
```

```

60     ax[0].plot(xit, yit, 'r-')
61     ax[0].set_ylabel('y [mm]')
62     ax[0].set_title('Speed in water [mm s$^{-1}]$', fontsize=11)
63     plt.colorbar(cs, ax=ax[0])
64
65     clines = np.linspace(0, 5000, 101)
66     cs = ax[1].contourf(x, y, v_mag, clines)
67     ax[1].plot(xit, yit, 'r-')
68     ax[1].set_xlabel('x [mm]')
69     ax[1].set_ylabel('y [mm]')
70     ax[1].set_title('Speed in air [mm s$^{-1}$]', fontsize=11)
71     plt.colorbar(cs, ax=ax[1])
72
73     plt.savefig('figures/task_b.png', dpi=1200)
74
75     # Plotting velocity as quiverplot
76     fig, ax = plt.subplots()
77
78     skip = (slice(None, None, 10), slice(None, None, 10))
79     ax.quiver(x[skip], y[skip], u[skip], v[skip])
80     ax.plot(xit, yit, 'r-')
81
82     draw_rectangles(indicies, x, y, ax)
83
84     ax.set(
85         xlabel='x [mm]',
86         ylabel='y [mm]',
87         xlim=(0, np.max(x)),
88         ylim=(-51., 51.),
89         title='Velocity in xy-plane'
90     )
91
92     plt.savefig('figures/task_c.png', dpi=1200)
93
94     # Calculating divergence
95     dudx = np.gradient(u, 0.5, axis=0)
96     dvdy = np.gradient(v, 0.5, axis=1)
97     div = dudx + dvdy
98
99     # Plotting divergence as contourplot
100    fig, ax = plt.subplots()
101    cs = ax.contourf(x, y, div)
102    plt.colorbar(cs, ax=ax)
103    ax.plot(xit, yit, 'r-')
104
105    draw_rectangles(indicies, x, y, ax)
106
107    ax.set(
108        xlabel='x [mm]',
109        ylabel='y [mm]',
110        xlim=(0, np.max(x)),
111        ylim=(-50., 50.),
112        title='Divergence in $xy$-plane'
113    )
114
115    plt.savefig('figures/task_d.png', dpi=1200)
116
117    # Calculating curl
118    dvdx = np.gradient(v, 0.5, axis=0)
119    dudy = np.gradient(u, 0.5, axis=1)
120    curl_z = dvdx - dudy
121
```

```

122 # Plotting curl in z direction and streamlines as contourplot
123 fig, ax = plt.subplots()
124 cs = ax.contourf(x, y, curl_z)
125 plt.colorbar(cs, ax=ax)
126 ax.plot(xit, yit, 'r-')
127 ax.streamplot(
128     x.transpose(),
129     y.transpose(),
130     u.transpose(),
131     v.transpose(),
132     color=u.transpose()
133 )
134
135 draw_rectangles(indicies, x, y, ax)
136
137 ax.set(
138     xlabel='x [mm]',
139     ylabel='y [mm]',
140     xlim=(0, np.max(x)),
141     ylim=(-50., 50.),
142     title='Curl in $z$-direction'
143 )
144
145 plt.savefig('figures/task_e.png', dpi=1200)
146
147 if __name__ == '__main__':
148     main()
149     plt.show()

```

File: integration.py

```

1 import numpy as np
2 from plots import u, v, indicies
3
4 def circulation_line_integral(
5     u: np.ndarray,
6     v: np.ndarray,
7     p: tuple[tuple[int, int, int, int], ...]
8 ) -> tuple[list[float], float]:
9     """
10     Calculates the circulation around rectangle over field  $\mathbf{F} = ui + vj$ 
11     with positive orientation as four line integrals.
12
13     Parameters
14     -----
15     u : np.ndarray
16         x component of field
17     v : np.ndarray
18         y component of field
19     p : tuple[tuple[int, int, int, int], ...]
20         Nested n-tuple with indicies for coordinates
21
22     Returns
23     -----
24     parts : list[float]
25         Circulation over each line
26     total_circulation : float
27         Total circulation around rectangle
28     """
29     x1, y1, x2, y2 = p
30     dt = 0.50
31

```

```

32     parts = np.zeros(4)
33     parts[0] = np.sum(u[x1:x2+1, y1]*dt)
34     parts[1] = np.sum(v[x2, y1:y2+1]*dt)
35     parts[2] = -np.sum(u[x1:x2+1, y2]*dt)
36     parts[3] = -np.sum(v[x1, y1:y2+1]*dt)
37
38     return parts, np.sum(parts)
39
40 def circulation_surface_integral(
41     u: np.ndarray,
42     v: np.ndarray,
43     p: tuple[tuple[int, int, int, int], ...]
44 ) -> float:
45     """
46     Calculates the circulation around rectangle over field  $F = ui + vj$ 
47     with positive orientation as a single surface integral.
48
49     Parameters
50     -----
51     u : np.ndarray
52         x component of field
53     v : np.ndarray
54         y component of field
55     p : tuple[tuple[int, int, int, int], ...]
56         Nested n-tuple with indicies for coordinates
57
58     Returns
59     -----
60     circulation : float
61         Total circulation around rectangle
62     """
63     x1, y1, x2, y2 = p
64     dt = 0.50
65     curl_z = np.gradient(v, dt, axis=0) - np.gradient(u, dt, axis=1)
66
67     return np.sum(curl_z[x1:x2+1, y1:y2+1]*dt**2)
68
69 def flux_line_integral(
70     u: np.ndarray,
71     v: np.ndarray,
72     p: tuple[tuple[int, int, int, int], ...]
73 ) -> float:
74     """
75     Calculates the integrated flux of the velocity field  $v = ui + vj$ 
76     with positive orientation as four line integrals.
77
78     Parameters
79     -----
80     u : np.ndarray
81         x component of field
82     v : np.ndarray
83         y component of field
84     p : tuple[tuple[int, int, int, int], ...]
85         Nested n-tuple with indicies for coordinates
86
87     Returns
88     -----
89     parts : list[float]
90         Flux over each line
91     flux : float
92         Total flux around rectangle
93     """

```

```

94     x1, y1, x2, y2 = p
95     dt = 0.50
96
97     parts = np.zeros(4)
98     parts[0] = -np.sum(v[x1:x2+1, y1]*dt)
99     parts[1] = np.sum(u[x2, y1:y2+1]*dt)
100    parts[2] = np.sum(v[x1:x2+1, y2]*dt)
101    parts[3] = -np.sum(u[x1, y1:y2+1]*dt)
102
103    return parts, np.sum(parts)
104
105 def main():
106     with open('integration_out.txt', 'w') as file:
107
108         for i, rec in enumerate(indicies, start=1):
109             sides, total = circulation_line_integral(u, v, rec)
110             surf = circulation_surface_integral(u, v, rec)
111             sides_flux, flux = flux_line_integral(u, v, rec)
112
113             file.write(f'-----\n')
114             file.write(f'Rectangle {i}\n')
115
116             file.write('\n')
117             file.write(f'Circulation, line integral:      {total:.2f}\n')
118
119             for j, side in enumerate(sides, start=1):
120                 file.write(f'Circulation side {j}:           {side:.2f}\n')
121
122             file.write('\n')
123             file.write(f'Circulation, surface integral: {surf:.2f}\n')
124             file.write(f'Percentage error:                {(total-surf)/(total)*100:.2f} %\n')
125
126             file.write('\n')
127             file.write(f'Integrated flux:                  {flux:.2f}\n')
128
129             for k, side in enumerate(sides_flux, start=1):
130                 file.write(f'Flux side {k}:                   {side:.2f}\n')
131
132             file.write(f'-----')
133
134 if __name__ == '__main__':
135     main()

```