

MEK1100 - Mandatory assignment 2

William Dugan

May 12, 2022

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

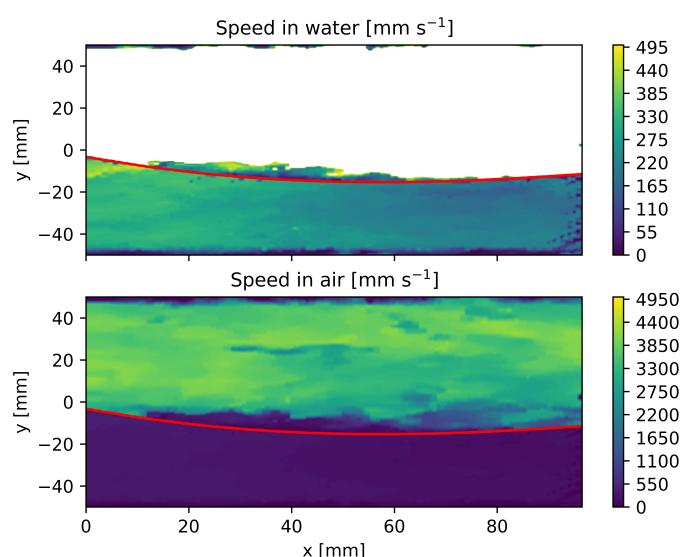


Figure 1: Contour plot of $\sqrt{u^2 + v^2}$. Red line indicating border between gas and liquid.

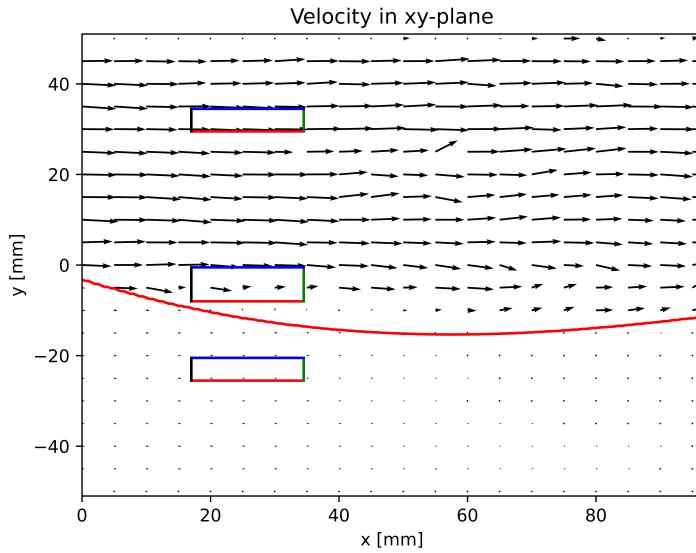


Figure 2: Quiver plot of velocity $ui + vj$. Red line indicating border between gas and liquid.

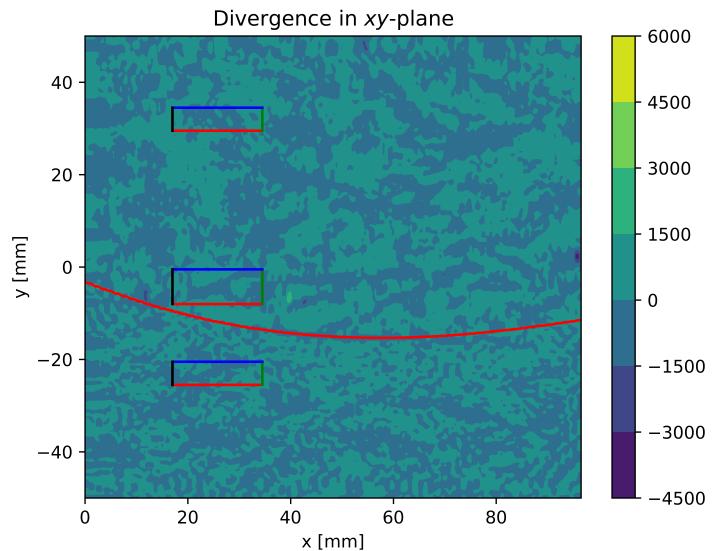


Figure 3: Contour plot of $\nabla \cdot \mathbf{v}^*$. Red line indicating border between gas and liquid.

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

$$\begin{aligned}\nabla \cdot \mathbf{v}^* &= \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \\ \nabla \cdot \mathbf{v} &= \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = \nabla \cdot \mathbf{v}^* + \frac{\partial w}{\partial z}\end{aligned}$$

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

$$\nabla \cdot \mathbf{v}^* + \frac{\partial w}{\partial z} = 0 \implies \frac{\partial w}{\partial z} = -\nabla \cdot \mathbf{v}^*$$

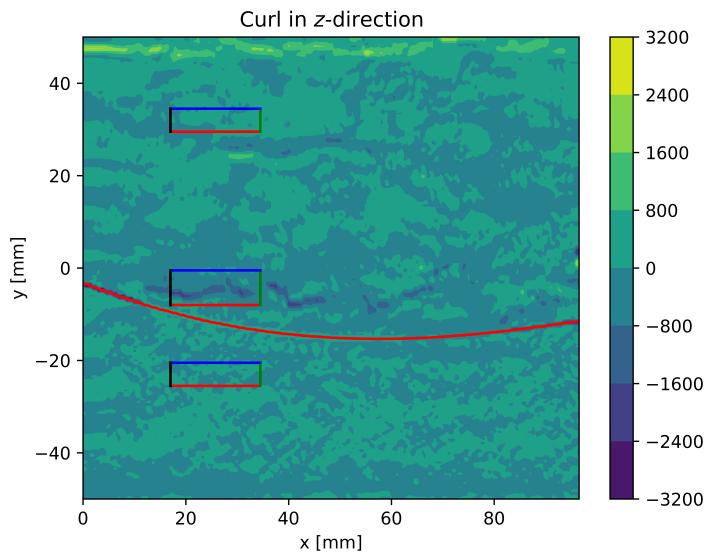


Figure 4: Contour plot of $(\nabla \times \mathbf{v}^*) \cdot \mathbf{k}$. Red line indicating border between gas and liquid.

From figure 5 we observe that the streamlines mostly follows the expected flow of an irrotational field. There are some deviations at the air-water border, but the curl is still around zero in this area (seen in figure 4). At the borders we observe that the flow is not straight. This is due to the friction force between the fluids and the walls.

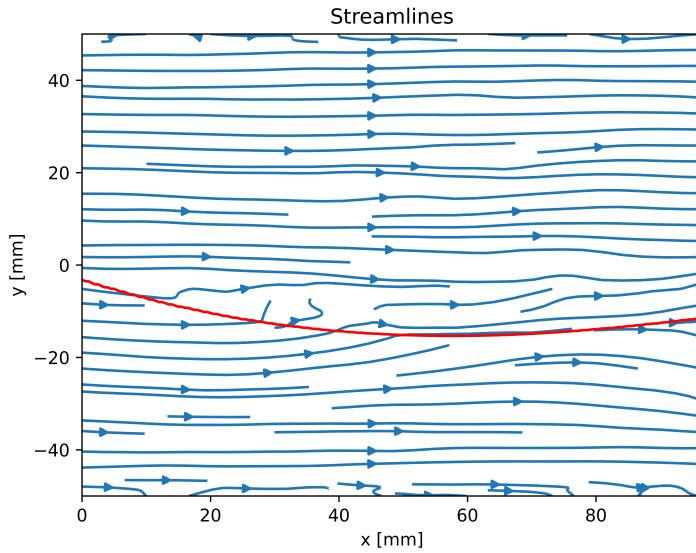


Figure 5: streamlines for \mathbf{v}^* . Red line indicating border between gas and liquid.

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
-----
```

For the rectangles in the part of the field located in the air, the difference between the line integral and the surface integral is negligible. For all three rectangles, the circulation is a lot larger in x direction, as expected. The values for the circulation around each side of the rectangles are as expected, with larger values where the velocity is higher. For the rectangle located in the water (rectangle 3), the difference between the line integral and the surface integral is very large, yet both values are a lot smaller than rectangle 1 and 2, since the velocity is a lot lower in the water. The error is likely due to our limited resolution ($\Delta x = \Delta y = 0.5\text{mm}$).

The divergence theorem states

$$\oint_C \mathbf{F} \cdot d\mathbf{s} = \int_V \nabla \cdot \mathbf{F} dV$$

Remembering that the fluid is incompressible, yet having $\nabla \cdot \mathbf{v}^* \neq 0$ results in the surface integral $\int_\sigma \nabla \cdot \mathbf{v}^* d\sigma \neq 0$. This means that the integrated flux in z direction for \mathbf{v} would be $-\int_\sigma \nabla \cdot \mathbf{v}^* d\sigma$ to ensure that $\int_\sigma \nabla \cdot \mathbf{v} d\sigma = 0$.

For all three rectangles we observe that the integrated flux is a large positive value for side number two and a large negative value for side number four. This is reasonable as there is a lot of flow in through the left side and out through the right. Rectangle 2 has a large negative integrated flux due to its placement close to the separation surface between water and air.

File: plots.py

```
1 import matplotlib.pyplot as plt
2 import scipy.io
3 import numpy as np
4
5 # Reading data
6 f = scipy.io.loadmat('data.mat')
7 x, y, u, v, xit, yit = (f[index].transpose() for index in ['x', 'y', 'u', 'v', 'xit', 'yit'])
8
9 # Checking dimension of data
10 for arr in (x, y, u, v):
11     shape = arr.shape
12     assert shape == (194, 201), 'Wrong dimension'
13     assert shape[0]*shape[1] == 38994, 'Wrong number of points in xy plane'
14
15 for arr in (xit, yit):
16     assert arr.shape == (194, 1), 'Wrong dimension'
17
18 assert (y[0,1]-y[0,0], x[1,0]-x[0,0]) == (0.5, 0.5), 'Wrong dx or dy'
19
20 assert (y[-1,0], y[-1,-1]) == (-50.0, 50.0), 'Wrong range of y'
21
22 # Indices of rectangles
23 indicies = ((34, 159, 69, 169), (34, 84, 69, 99), (34, 49, 69, 59))
24
25 def draw_rectangles(
26     indicies: tuple[tuple[int, int, int, int]],
27     x: np.ndarray,
28     y: np.ndarray,
29     ax: plt.Axes
30 ) -> None:
31     """
32         Draws the rectangles with lower left corner (x[x1,y1], y[x1,y1])
33         and upper right corner (x[x2,y2], y[x2,y2]).
34
35     Parameters
36     -----
37     indicies : tuple[tuple[int, int, int, int]]
38         Nested n-tuple with indicies for coordinates
39     x : np.ndarray
40         x component of xy-meshgrid
41     y : np.ndarray
42         y component of xy-meshgrid
43     ax : plt.Axes
44         Axes object where the rectangle is drawn.
45     """
46     for x1, y1, x2, y2 in indicies:
47         x1, y1, x2, y2 = x[x1,y1], y[x1,y1], x[x2,y2], y[x2,y2]
48         ax.plot([x1, x2], [y1, y1], color='r')
49         ax.plot([x2, x2], [y1, y2], color='g')
50         ax.plot([x1, x2], [y2, y2], color='b')
51         ax.plot([x1, x1], [y1, y2], color='k')
52
53 def main():
54     # Plotting speed as contourplot
55     v_mag = np.sqrt(u**2 + v**2)
56
57     fig, ax = plt.subplots(2, 1, sharex=True, sharey=True)
58
59     clines = np.linspace(0, 500, 101)
```

```

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

```

```

122     # Plotting curl in z direction
123     fig, ax = plt.subplots()
124
125     cs = ax.contourf(x, y, curl_z)
126     plt.colorbar(cs, ax=ax)
127     ax.plot(xit, yit, 'r-')
128
129     draw_rectangles(indicies, x, y, ax)
130
131     ax.set(
132         xlabel='x [mm]',
133         ylabel='y [mm]',
134         xlim=(0, np.max(x)),
135         ylim=(-50., 50.),
136         title='Curl in $z$-direction'
137     )
138
139
140     plt.savefig('figures/task_e_curl.png', dpi=1200)
141
142     # Plotting streamlines
143     fig, ax = plt.subplots()
144
145     ax.plot(xit, yit, 'r-')
146     ax.streamplot(
147         x.transpose(),
148         y.transpose(),
149         u.transpose(),
150         v.transpose(),
151     )
152
153     ax.set(
154         xlabel='x [mm]',
155         ylabel='y [mm]',
156         xlim=(0, np.max(x)),
157         ylim=(-50., 50.),
158         title='Streamlines'
159     )
160
161     plt.savefig('figures/task_e_streamlines.png', dpi=1200)
162
163 if __name__ == '__main__':
164     main()
165     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 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 through each line
91     flux : float
92         Total flux through 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()

```