MEK1100 - Mandatory assignment 2

William Dugan

May 11, 2022

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

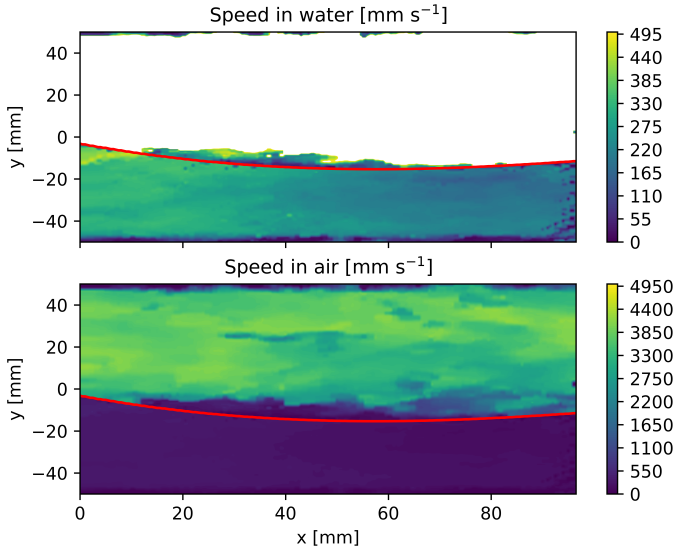

Figure 1: Contour plot of $\sqrt{u^2 + v^2}$. Red line indicating border between gas and liquid.

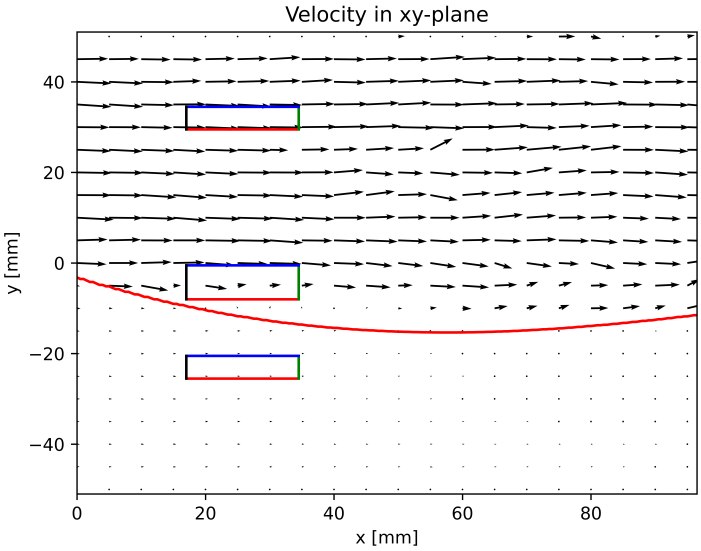

Figure 2: Quiver plot of velocity $u\boldsymbol{i} + v\boldsymbol{j}$. Red line indicating border between gas and liquid.

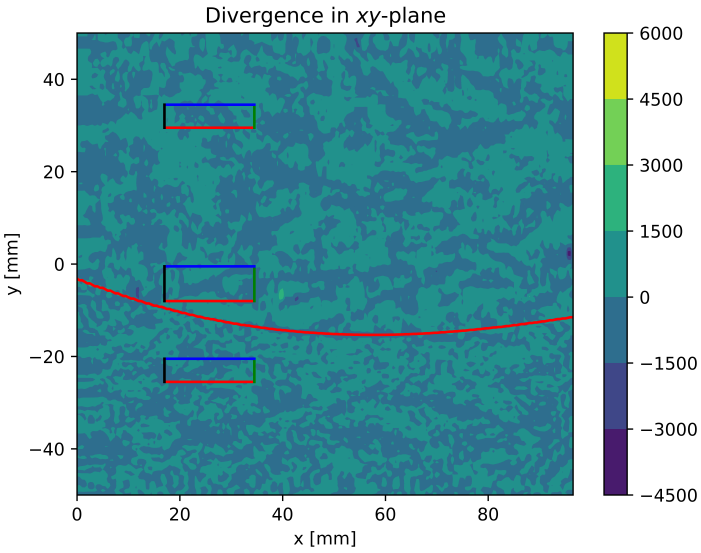

Figure 3: Contour plot of $\nabla \cdot \boldsymbol{v}^*$. Red line indicating border between gas and liquid.

Let $\boldsymbol{v} = u\boldsymbol{i} + v\boldsymbol{j} + w\boldsymbol{k}$ and $\boldsymbol{v}^* = u\boldsymbol{i} + v\boldsymbol{j}$. The divergence is

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

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

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

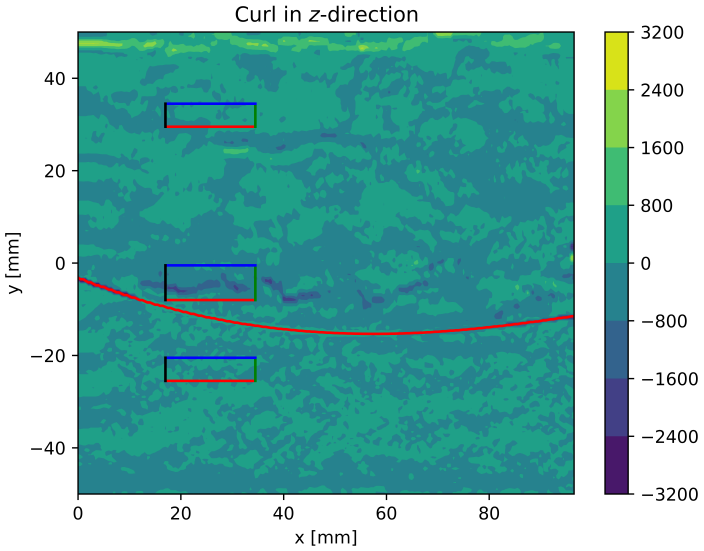

Figure 4: Contour plot of $(\nabla \times \boldsymbol{v}^*) \cdot \boldsymbol{k}$. Red line indicating border between gas and liquid.

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 fuilds and the walls.

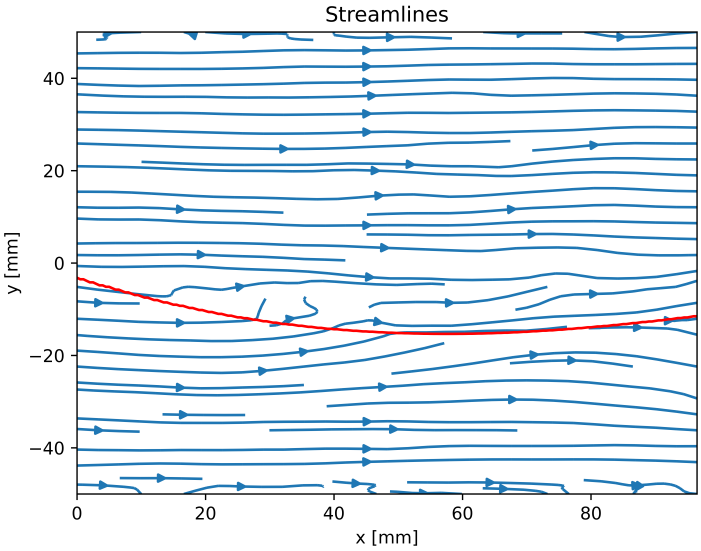

Figure 5: streamlines for $\boldsymbol{v}^*$. Red line indicating border between gas and liquid.

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 ($\Delta x = \Delta y = 0.5$mm).

Since $(\boldsymbol{\nabla} \times \boldsymbol{v}^*) \cdot \boldsymbol{k} \neq 0$, the surface integral $\iint_\sigma (\boldsymbol{\nabla} \times \boldsymbol{v}^*) \cdot \boldsymbol{k} d\sigma \neq 0$. This means that the integrated flux in z direction for $\boldsymbol{v}$ would be $-\iint_\sigma (\boldsymbol{\nabla} \times \boldsymbol{v}^*) \cdot \boldsymbol{k} d\sigma$ to ensure that $\iint_\sigma (\boldsymbol{\nabla} \times \boldsymbol{v}) \cdot \boldsymbol{n} d\sigma = 0$.

For rectangle 1, the integrated flux is very roughly zero as the net flow through the rectangle is approximately the same on all four sides. For rectangle 2, we observe that there is a lot of flow in through side 4 (marked in black in figure 2). This is as expected. For rectangle 3, there is mostly flow in through the top, and out through the bottom.

File: plots.py

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

# Reading data
f = scipy.io.loadmat('data.mat')
x, y, u, v, xit, yit = (f[index].transpose() for index in ['x', 'y', 'u', 'v', '
    xit', 'yit'])

# Checking dimension of data
for arr in (x, y, u, v):
    shape = arr.shape
    assert shape == (194, 201), 'Wrong dimension'
    assert shape[0]*shape[1] == 38994, 'Wrong numper of points in xy plane'

for arr in (xit, yit):
    assert arr.shape == (194, 1), 'Wrong dimension'

assert (y[0,1]-y[0,0], x[1,0]-x[0,0]) == (0.5, 0.5), 'Wrong dx or dy'

assert (y[-1,0], y[-1,-1]) == (-50.0, 50.0), 'Wrong range of y'

# Indicies of rectangles
indicies = ((34, 159, 69, 169), (34, 84, 69, 99), (34, 49, 69, 59))

def draw_rectangles(
        indicies: tuple[tuple[int, int, int, int]],
        x: np.ndarray,
        y: np.ndarray,
        ax: plt.Axes
    ) -> None:
    """
    Draws the rectangles with lower left corner (x[x1,y1], y[x1,y1])
    and upper right corner (x[x2,y2], y[x2,y2]).

    Parameters
    ----------
    indicies : tuple[tuple[int, int, int, int]]
        Nested n-tuple with indicies for coordinates
    x : np.ndarray
        x component of xy-meshgrid
    y : np.ndarray
        y component of xy-meshgrid
    ax : plt.Axes
        Axes object where the rectangle is drawn.
    """
    for x1, y1, x2, y2 in indicies:
        x1, y1, x2, y2 = x[x1,y1], y[x1,y1], x[x2,y2], y[x2,y2]
        ax.plot([x1, x2], [y1, y1], color='r')
        ax.plot([x2, x2], [y1, y2], color='g')
        ax.plot([x1, x2], [y2, y2], color='b')
        ax.plot([x1, x1], [y1, y2], color='k')

def main():
    # Plotting speed as contourplot
    v_mag = np.sqrt(u**2 + v**2)

    fig, ax = plt.subplots(2, 1, sharex=True, sharey=True)

    clines = np.linspace(0, 500, 101)
```

```
    cs = ax[0].contourf(x, y, v_mag, clines, vmin=0)
    ax[0].plot(xit, yit, 'r-')
    ax[0].set_ylabel('y [mm]')
    ax[0].set_title('Speed in water [mm s$^{-1}$]', fontsize=11)
    plt.colorbar(cs, ax=ax[0])

    clines = np.linspace(0, 5000, 101)
    cs = ax[1].contourf(x, y, v_mag, clines)
    ax[1].plot(xit, yit, 'r-')
    ax[1].set_xlabel('x [mm]')
    ax[1].set_ylabel('y [mm]')
    ax[1].set_title('Speed in air [mm s$^{-1}$]', fontsize=11)
    plt.colorbar(cs, ax=ax[1])

    plt.savefig('figures/task_b.png', dpi=1200)

    # Plotting velocity as quiverplot
    fig, ax = plt.subplots()

    skip = (slice(None, None, 10), slice(None, None, 10))
    ax.quiver(x[skip], y[skip], u[skip], v[skip])
    ax.plot(xit, yit, 'r-')

    draw_rectangles(indicies, x, y, ax)

    ax.set(
        xlabel='x [mm]',
        ylabel='y [mm]',
        xlim=(0, np.max(x)),
        ylim=(-51., 51.),
        title='Velocity in xy-plane'
    )

    plt.savefig('figures/task_c.png', dpi=1200)

    # Calculating divergence
    dudx = np.gradient(u, 0.5, axis=0)
    dvdy = np.gradient(v, 0.5, axis=1)
    div = dudx + dvdy

    # Plotting divergence as contourplot
    fig, ax = plt.subplots()
    cs = ax.contourf(x, y, div)
    plt.colorbar(cs, ax=ax)
    ax.plot(xit, yit, 'r-')

    draw_rectangles(indicies, x, y, ax)

    ax.set(
        xlabel='x [mm]',
        ylabel='y [mm]',
        xlim=(0, np.max(x)),
        ylim=(-50., 50.),
        title='Divergence in $xy$-plane'
    )

    plt.savefig('figures/task_d.png', dpi=1200)

    # Calculating curl
    dvdx = np.gradient(v, 0.5, axis=0)
    dudy = np.gradient(u, 0.5, axis=1)
    curl_z = dvdx - dudy
```

```
122
123     # Plotting curl in z direction
124     fig, ax = plt.subplots()
125
126     cs = ax.contourf(x, y, curl_z)
127     plt.colorbar(cs, ax=ax)
128     ax.plot(xit, yit, 'r-')
129
130     draw_rectangles(indicies, x, y, ax)
131
132     ax.set(
133         xlabel='x [mm]',
134         ylabel='y [mm]',
135         xlim=(0, np.max(x)),
136         ylim=(-50., 50.),
137         title='Curl in $z$-direction'
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
```

```
        x component of field
    v : np.ndarray
        y component of field
    p : tuple[tuple[int, int, int, int]]
        Nested n-tuple with indicies for coordinates

    Returns
    -------
    parts : list[float]
        Circulation over each line
    total_circulation : float
        Total circulation around rectangle
    """
    x1, y1, x2, y2 = p
    dt = 0.50

    parts = np.zeros(4)
    parts[0] = np.sum(u[x1:x2+1, y1]*dt)
    parts[1] = np.sum(v[x2, y1:y2+1]*dt)
    parts[2] = -np.sum(u[x1:x2+1, y2]*dt)
    parts[3] = -np.sum(v[x1, y1:y2+1]*dt)

    return parts, np.sum(parts)

def circulation_surface_integral(
        u: np.ndarray,
        v: np.ndarray,
        p: tuple[tuple[int, int, int, int]]
    ) -> float:
    """
    Calculates the circulation around rectangle over field F = ui + vj
    with positive orientation as a single surface integral.

    Parameters
    ----------
    u : np.ndarray
        x component of field
    v : np.ndarray
        y component of field
    p : tuple[tuple[int, int, int, int]]
        Nested n-tuple with indicies for coordinates

    Returns
    -------
    circulation : float
        Total circulation around rectangle
    """
    x1, y1, x2, y2 = p
    dt = 0.50
    curl_z = np.gradient(v, dt, axis=0) - np.gradient(u, dt, axis=1)

    return np.sum(curl_z[x1:x2+1, y1:y2+1]*dt**2)

def flux_line_integral(
        u: np.ndarray,
        v: np.ndarray,
        p: tuple[tuple[int, int, int, int]]
    ) -> float:
    """
    Calculates the integrated flux of the velocity field v = ui + vj
    with positive orientation as four line integrals.

```

```
    Parameters
    ----------
    u : np.ndarray
        x component of field
    v : np.ndarray
        y component of field
    p : tuple[tuple[int, int, int, int]]
        Nested n-tuple with indicies for coordinates

    Returns
    -------
    parts : list[float]
        Flux through each line
    flux : float
        Total flux through rectangle
    """
    x1, y1, x2, y2 = p
    dt = 0.50

    parts = np.zeros(4)
    parts[0] = -np.sum(v[x1:x2+1, y1]*dt)
    parts[1] = np.sum(u[x2, y1:y2+1]*dt)
    parts[2] = np.sum(v[x1:x2+1, y2]*dt)
    parts[3] = -np.sum(u[x1, y1:y2+1]*dt)

    return parts, np.sum(parts)

def main():
    with open('integration_out.txt', 'w') as file:

        for i, rec in enumerate(indicies, start=1):
            sides, total = circulation_line_integral(u, v, rec)
            surf = circulation_surface_integral(u, v, rec)
            sides_flux, flux = flux_line_integral(u, v, rec)

            file.write(f'-------------------------------------------\n')
            file.write(f'Rectangle {i}\n')

            file.write('\n')
            file.write(f'Circulation, line integral:     {total:.2f}\n')

            for j, side in enumerate(sides, start=1):
                file.write(f'Circulation side {j}:             {side:.2f}\n')

            file.write('\n')
            file.write(f'Circulation, surface integral:  {surf:.2f}\n')
            file.write(f'Percentage error:               {(total-surf)/(total)
    *100:.2f} %\n')

            file.write('\n')
            file.write(f'Integrated flux:                {flux:.2f}\n')

            for k, side in enumerate(sides_flux, start=1):
                file.write(f'Flux side {k}:                    {side:.2f}\n')

        file.write(f'-------------------------------------------')

if __name__ == '__main__':
    main()
```