MEK1100 - Oblig 2

Hans-Petter Harveg

April 2018

Oppsummering

Vi analyserer et dataset målt i Hydrodynamisk labratorium ved UiO. Vi gjør først noen tester på datasettet, deretter benytter vi nummeriske metoder for analyse av datasettet. Vi regner ut typiske størrelser og bruker Stokes, Greens sats og Gauss. Resultatene viser noe avvik, noe som er forventet når man tar høyde for nummeriske feil og unøyaktighet i målinger.

Besvarelse

a)

I denne oppgaven har jeg lastet inn datafilen og skrevet testfunksjoner for de gitte oppgavene. Spesifikt har jeg skrevet følgende funksjoner

- *get_matrix_sizes()*: skriver ut dimensjonene på matrisen ved hjelp av funksjonen *shate()*.
- *test_pixel_spread()*: sjekker at bredden mellom pixlene er 0.5. Fordi vi har en matrise med x-verider og en med y-verdier sjekker jeg begge matrisene. Funksjonen returnerer *False* dersom den finner en verdi $\Delta x \neq 0.5$.
- *test_y_range()*: sjekker at dataen spenner høyden på røret.

Kjøreeksempelet ser ut som følger

```
Size of [ yit ]: 194 1
Size of [ xit ]: 194 1
Size of [ v ]: 194 201
Size of [ y ]: 194 201
Size of [ x ]: 194 201
Size of [ u ]: 194 201

Grid is evenly spread, looks good!

Spread in y direction:
[[-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. ]]
```

Koden i sin helhet ligger under kildekode.

b)

I denne oppgaven har jeg laget to plot

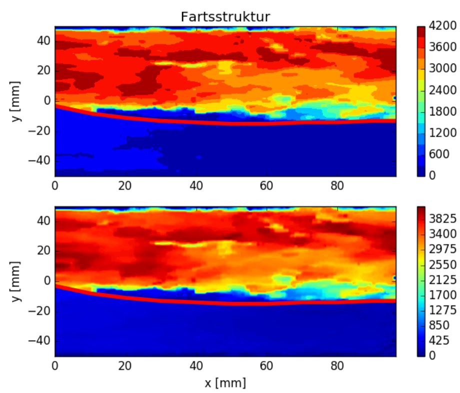

Figure 1: Fartstruktur i vann og luft. Den rød linjen markerer skillet mellom vann og luft.

c)

I denne oppgaven plotter jeg hastighetene ved hjelp av funksjonen *quiver()*. Fordi antallet punkgter er høyt viser jeg kun hver 5. pil. Grunnen til at pilene vises som punkter er fordi forskjellden i hastighet her svært forskjellig i vann og luft.

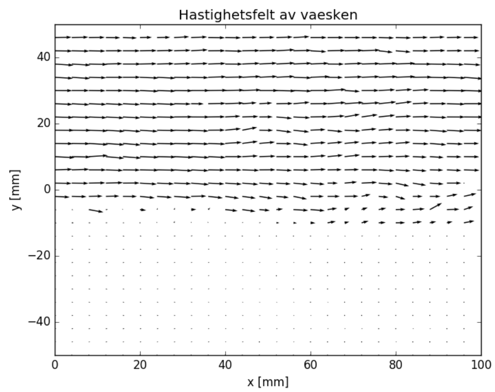

Figure 2: Fordi hastigheten er så liten i vannet i forhold til i lufta er det vanskelig å se vektorpilene.

Ved å bruke indeksene gitt i oppgaven henter jeg ut koordinater fra datasettet. Disse bruker jeg til å plotte tre firkanter inn i figuren.

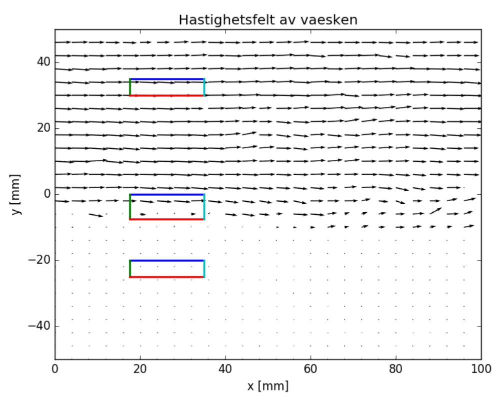

Figure 3: Fordi hastigheten er så liten i vannet i forhold til i lufta er det vanskelig å se vektorpilene.

d)

Divergensen til $\vec{v}$ er gitt ved

$$\nabla \cdot \vec{v} = \frac{\partial}{\partial x}u + \frac{\partial}{\partial y}v + \frac{\partial}{\partial z}w, \tag{1}$$

men måten eksperimentet er satt opp på gir $\frac{\partial}{\partial z}w = 0$. Imkompressibel vil si at når man følger en fluidpartikkel gjennom et hastighetsfelt vil den ha konstant tetthet

$$\frac{D\rho}{dt} = 0. \tag{2}$$

Derav fra kontinuitetsligningen har vi at

$$\frac{D\rho}{dt} + \rho\nabla \cdot \vec{v} = 0, \tag{3}$$

hvor vi har at $\nabla \cdot \vec{v} = 0$ for imkomprissibelt fluid. Dette betyr at $\nabla \cdot \vec{v} = 0$

$$\frac{\partial}{\partial z} w = -\left(\frac{\partial}{\partial x} u + \frac{\partial}{\partial y} v\right) \tag{4}$$

e)

Virvlingen til $\vec{v}$ er gitt ved

$$\nabla \cdot \vec{v} = \begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ \partial_x & \partial_y & \partial_z \\ u & v & w \end{vmatrix}. \tag{5}$$

Som gir

$$\hat{k}\left(\frac{\partial}{\partial_x} v - \frac{\partial}{\partial_y} u\right), \tag{6}$$

altså komponenten *normalt på xy-planet*. Hvis vi definerer ett nytt hastighetsfelt

$$\vec{v}' = v\hat{i} - u\hat{j}, \tag{7}$$

får vi at komponenten *normalt på xy-planet* som

$$\hat{k}\left(\frac{\partial}{\partial x} v - \frac{\partial}{\partial y} u\right) = \hat{k}\nabla \cdot \hat{v} \tag{8}$$

f)

Kurveintegral

Sirulasjonen er gitt ved

$$\oint_C \vec{v} \cdot d\vec{r} = \sum_{i=1}^{N} \int_{C_i} \vec{v} \cdot d\vec{r}, \tag{9}$$

for $N = 4$, der $d\vec{r} = dx\hat{i} + dy\hat{j}$. Vi deler så opp i fire kurver og legger sammen bidragene. Hvis vi lar x_1, y_1 og x_2, y_2 representere hjørnene i firkantene får vi

$$\oint_C \hat{v} \cdot d\hat{r} = \int_{x_1}^{x_2} \vec{v} \cdot dx\hat{i} + \int_{y_1}^{y_2} \vec{v} \cdot dy\hat{j} + \int_{x_2}^{x_1} \vec{v} \cdot dx\hat{i} + \int_{y_2}^{y_1} \vec{v} \cdot dy\hat{j} \tag{10}$$

Flateintegral

Med Greens sats ser vi på/blir kurveintegralet et flateintegral. Hvis vi setter

$$\vec{v} = F(x,y)\hat{i} + G(x,y)\hat{j} = u\hat{i} + v\hat{j} \tag{11}$$

$$\iint_S \left(\frac{\partial}{\partial x}v - \frac{\partial}{\partial y}u\right)dxdy = \oint_\gamma u\ dx + v\ dy = \oint_\gamma \vec{v}\cdot d\vec{r} \tag{12}$$

g)

For fluks gjennom en kurve har vi

$$\int_\gamma \vec{v}\cdot\vec{n}ds = \int_\gamma v_x dy - v_y dx. \tag{13}$$

Hvis vi, som i forrige oppgave deler hvert kvadrat inn i fire linjer får vi

$$a = 0 \tag{14}$$

Kildekode

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

class DataSet():
    def __init__(self, filename):
        filedata = sio.loadmat(filename)
        self.data = {"x":filedata.get("x"), "y":filedata.get("y"),
    "u":filedata.get("u"), "v":filedata.get("v"), "xit":filedata.
    get("xit"), "yit":filedata.get("yit") }


    def get_data(self):
        return (self.data["x"], self.data["y"], self.data["u"],
    self.data["v"], self.data["xit"], self.data["yit"])


    def get_matrices_sizes(self):
        for key, value in self.data.items():
            [a, b] = np.shape(value)
            print "Size of [", key, "]:", b, a

        return True


    def test_grid_spread(self):
        [x, y, u, v, xit, yit] = self.get_data()

        print ""
        for iy in range(201):
            prev_x = 0
```

```
            prev_y =0
            for ix in range(1,192):
                if x[iy][ix]-prev_x != 0.5:
                    return False
                prev_x = x[iy][ix]

                if y[iy][ix]-prev_y != 0.5:
                    return False
                prev_y = y[iy][ix]

        print ""
        print "Grid is eavenly spread, looks good!"

        return True


    def test_y_range(self):
        [x, y, u, v, xit, yit] = self.get_data()

        print ""
        print "Spread in y direction:"
        print y


    def plot_contours(self):
        [x, y, u, v, xit, yit] = self.get_data()

        c = np.sqrt(u**2 + v**2)

        plt.subplot(2,1,1)
        plt.contourf(x, y, c, 15)
        plt.colorbar()

        dataset.set_plot_desc("Fartsstruktur", "", "y [mm]")

        self.plot_line_of_seperation()

        plt.subplot(2,1,2)
        plt.contourf(x, y, c, 200)
        plt.colorbar()

        dataset.set_plot_desc("", "x [mm]", "y [mm]")

        self.plot_line_of_seperation()

        plt.show()


    def plot_line_of_seperation(self):
        plt.plot([0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 96],[-3,
    -8, -11, -13, -14, -15, -15, -14, -14, -13, -13], linewidth=4,
    color="r")


    def plot_quiver(self, x_0, x_1, y_0, y_1, step, render_squares)
    :
        [x, y, u, v, xit, yit] = self.get_data()
```

```

        plt.quiver(x[x_0:x_1:step, y_0:y_1:step], y[x_0:x_1:step, y_0:y_1:step], u[x_0:x_1:step, y_0:y_1:step], v[x_0:x_1:step, y_0:y_1:step])

        dataset.set_plot_desc("Hastighetsfelt av vaesken", "x [mm]", "y [mm]")

        if render_squares:
            self.plot_squares()

        plt.axis([0, 100, -50, 50])

        plt.show()


    def plot_square(self, c0, c1):
        [x, y, u, v, xit, yit] = self.get_data()

        x0 = x[c0[1]][c0[0]]
        y0 = y[c0[1]][c0[0]]

        x1 = x[c1[1]][c1[0]]
        y1 = y[c1[1]][c1[0]]

        plt.plot([x0, x1], [y1, y1], linewidth=2.0, color='b')
        plt.plot([x0, x0], [y0, y1], linewidth=2.0, color='g')
        plt.plot([x0, x1], [y0, y0], linewidth=2.0, color='r')
        plt.plot([x1, x1], [y0, y1], linewidth=2.0, color='c')


    def plot_squares(self):
        self.plot_square([35, 160],[70, 170])
        self.plot_square([35, 85],[70, 100])
        self.plot_square([35, 50],[70, 60])


    def plot_divergence(self):
        [x, y, u, v, xit, yit] = self.get_data()
        div = self.divergence([u, v])
        div = sio.divergence(u, v)
        plt.contourf(x, y, div)
        plt.colorbar()

        self.plot_squares()

        dataset.set_plot_desc("Divergensen", "x [mm]", "y [mm]")

        plt.show()


    def plot_zcomp(self, render_quiver):
        [x, y, u, v, xit, yit] = self.get_data()
        div = self.divergence([u, -v])
        plt.contourf(x, y, div)
        plt.colorbar()

```

```python
        x0 = y0 = 0
        x1 = 191
        y1 = 201
        step = 5

        if render_quiver:
            plt.quiver(x[x0:x1:step, y0:y1:step], y[x0:x1:step, y0:y1:step], u[x0:x1:step, y0:y1:step], v[x0:x1:step, y0:y1:step])

        self.plot_squares()

        dataset.set_plot_desc("Divergensen", "x [mm]", "y [mm]")

        plt.show()


    def divergence(self, f):
        num_dims = len(f)
        return np.ufunc.reduce(np.add, [np.gradient(f[i], axis=i) for i in range(num_dims)])


    def divergence(self, f):
        num_dims = len(f)
        return np.ufunc.reduce(np.add, [np.gradient(f[i], axis=i) for i in range(num_dims)])


    def set_plot_desc(self, title, xlabel, ylabel):
        plt.title(title)
        plt.xlabel(xlabel)
        plt.ylabel(ylabel)


# Problem a)
dataset = DataSet("data.mat")
dataset.get_matrices_sizes()
dataset.test_grid_spread()
dataset.test_y_range()

# Problem b)
dataset.plot_contours()

# Problem c)
dataset.plot_quiver(0, 194, 0, 201, 8, False)
dataset.plot_quiver(0, 194, 0, 201, 8 ,True)

# Problem d)
#dataset.plot_divergence()

# Problem e)
#dataset.plot_zcomp(False)
#dataset.plot_zcomp(True)

# Problem f)

# Problem g)
```