

Computational Fluid Dynamics  
SG2212 Project Report

David Ahnlund

## Question 1

The kronecker operator (**kron**) is an operator for tensor product, which falls convinient when doing finite differences in higher dimensions than the 1D case. As we study the 2D case in this lid-driven cavity project the 5 point stencil is considered:

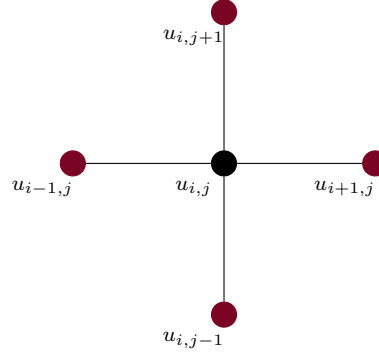


Figure 1: The five point stencil in 2D.

If we construct a finite difference scheme for the laplace operator we get:

$$\nabla^2 u = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h_x^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h_y^2} \quad (1)$$

$$\left\{ \text{and if } h_x = h_y = h \implies \frac{u_{i+1,j} + u_{i-1,j} - 4u_{i,j} + u_{i,j+1} + u_{i,j-1}}{h^2} \right\} \quad (2)$$

The above expression we can transform into a vector form in size  $(N_x, N_y) \mapsto N_x \cdot N_y$ , using the indexing that one step in  $y$ -direction is equivalent to one row in the matrix *i.e.*  $+N_x$  steps.

We can now define the second partial derivative in the separate directions:

$$S_x = \frac{1}{h_x^2} \begin{bmatrix} -1 & 1 & & & \\ 1 & -2 & 1 & & \\ & 1 & -2 & \ddots & \\ & & \ddots & \ddots & \\ & & & 1 & -1 \end{bmatrix} \quad S_y = \frac{1}{h_y^2} \begin{bmatrix} -1 & 1 & & & \\ 1 & -2 & 1 & & \\ & 1 & -2 & \ddots & \\ & & \ddots & \ddots & \\ & & & 1 & -1 \end{bmatrix}$$

Forming a matrix by the size  $(N_x \cdot N_y) \times (N_x \cdot N_y)$  is now done by the kronecker product.

The operation done by the kronecker product essentially does the following in this case. Consider the identity matrix in the shape  $N_y \times N_y$  ( $I_y$ ), and the  $S_x$  from above. We get:

$$\text{kron}(I_y, S_x) = I_y \otimes S_x = \begin{bmatrix} [S_x] & 0 & \dots & \\ 0 & [S_x] & 0 & \dots \\ \vdots & & \ddots & \\ & & & [S_x] \end{bmatrix} \quad \text{shape is } (N_y \cdot N_x) \times (N_y \cdot N_x)$$

and for the other term in the sum of kronecker products we have:

$$S_y \otimes I_x = \begin{bmatrix} S_y^{1,1}[I_x] & 0 & \dots & \\ 0 & S_y^{2,2}[I_x] & 0 & \dots \\ \vdots & & \ddots & \\ & & & S_y^{N_y, N_y}[I_x] \end{bmatrix} \quad \text{shape is } (N_y \cdot N_x) \times (N_y \cdot N_x)$$

We can then see that in the simplified example where  $N_x = N_y = N$  and  $h_x = h_y = h = 1$ , and no specific boundaries are set, we get:

$$I_y \otimes S_x + S_y \otimes I_x = \begin{bmatrix} -4 & 1 & \dots & 1 & \dots & \\ 1 & -4 & 1 & \dots & 1 & \dots \\ 0 & \ddots & \ddots & \ddots & & \\ & & & 1 & \dots & 1 & -4 \end{bmatrix} \quad (3)$$

which describes the case stated in Equation 2. Note that the 1's in matrix in Equation 3 are  $N_x$  values apart, with zero-values in between.

### Question 2

Experimental stability condition for  $\Delta t$  (with parameters  $N_x = N_y = 30$ ,  $L_y = L_x = 1$  and  $Re = 25$ ) was empirically found to be

$$\Delta t_{\max} \approx 0.006971$$

when integrated up to  $T = 50$ .

### Question 3

### Question 4

Placing a probe in the domain centre gives the following velocity  $U$  along the time line to  $T = 100$ :

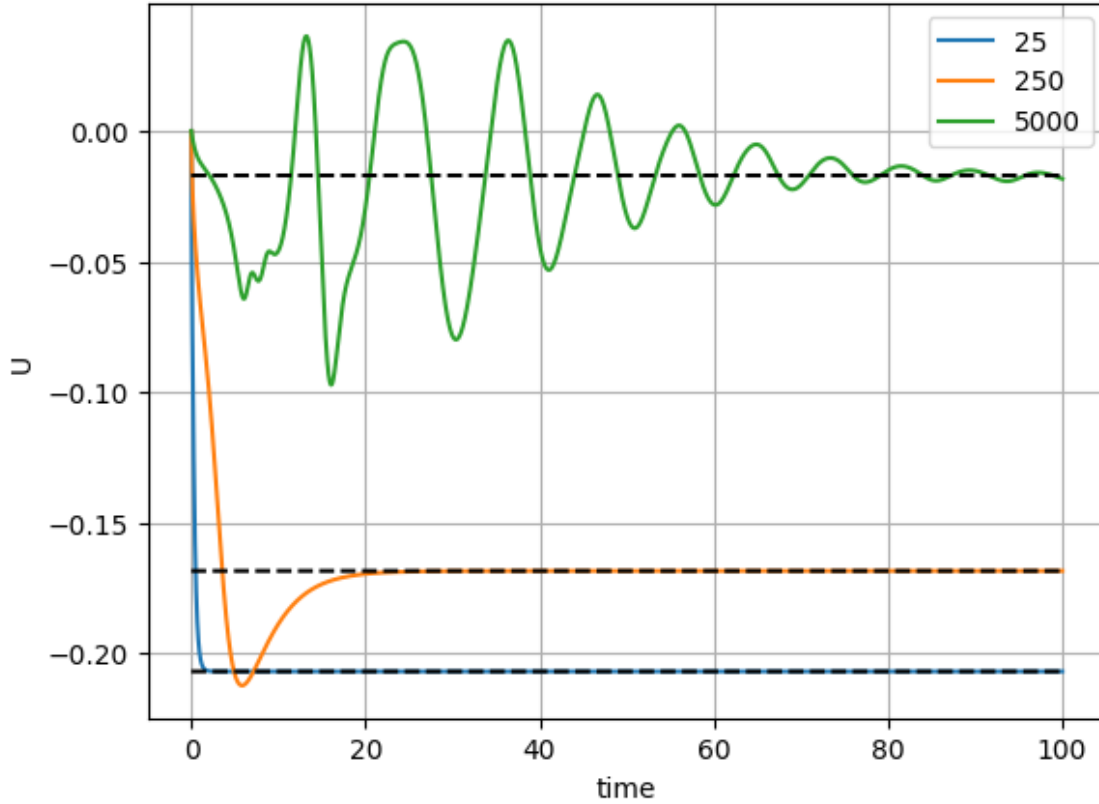


Figure 2:  $U$  in middle of domain for different  $Re$ -values

Clearly the cases with lower Reynold's number converges to a steady flow quicker than the cases with a higher number. This is due to the amount of turbulence is greater in the case with higher  $Re$ ; the inertial forces are greater in comparison to the viscous forces when the  $Re$  increases.

To in a more concrete way decide when a steady flow is reached in the location of the probe, the derivative of  $U$  w.r.t  $t$  is taken into account. When setting up two conditions, which essentially measures when the relative derivative and deviation from the median doesn't overshoot 5%. With the following lines of code:

```
for col in range(uvel.shape[1]):
    cond1 = np.abs(uvel[1:, col] - np.median(uvel, axis = 0)[col]) < \
        0.05 * np.abs(np.median(uvel, axis = 0)[col])
    cond2 = np.abs(rel_deriv[:, col]) < 0.05

    cond = cond1 * cond2
```

The following times for steady flow was estimated, when looking at the first time point when both conditions were fulfilled:

```
Time to steady flow when Re = 25: 1.1998011998011997 s
Time to steady flow when Re = 250: 13.217413217413217 s
Time to steady flow when Re = 5000: 96.35719635719636 s
```

### Question 5

Using OpenFoam in each of the cases A-C the following data where obtained: Figure 3 shows both a solution from the Python implementation as well as OpenFOAM solution with the data visualized in matplotlib.

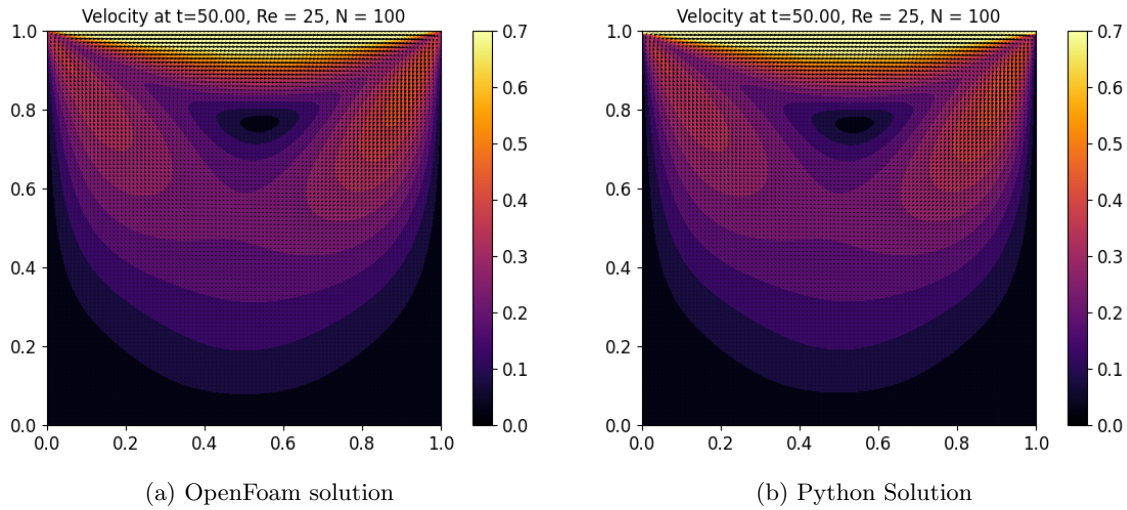


Figure 3: Case A

For a more detailed comparison we can observe the velocity magnitude along the diagonal of the domain, so called a *plot over line* visualization. For case A we have the following:

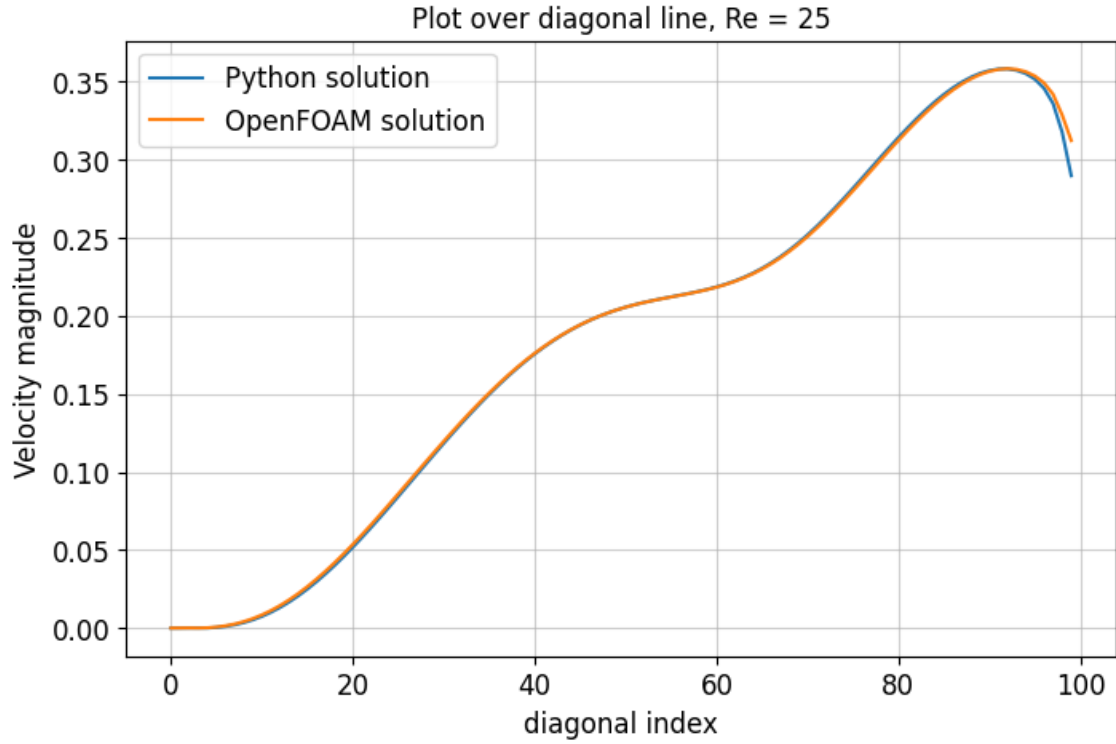


Figure 4: Plot over line (Case A): Python against OpenFOAM

Likewise we can compare case B and C in a similar manner:

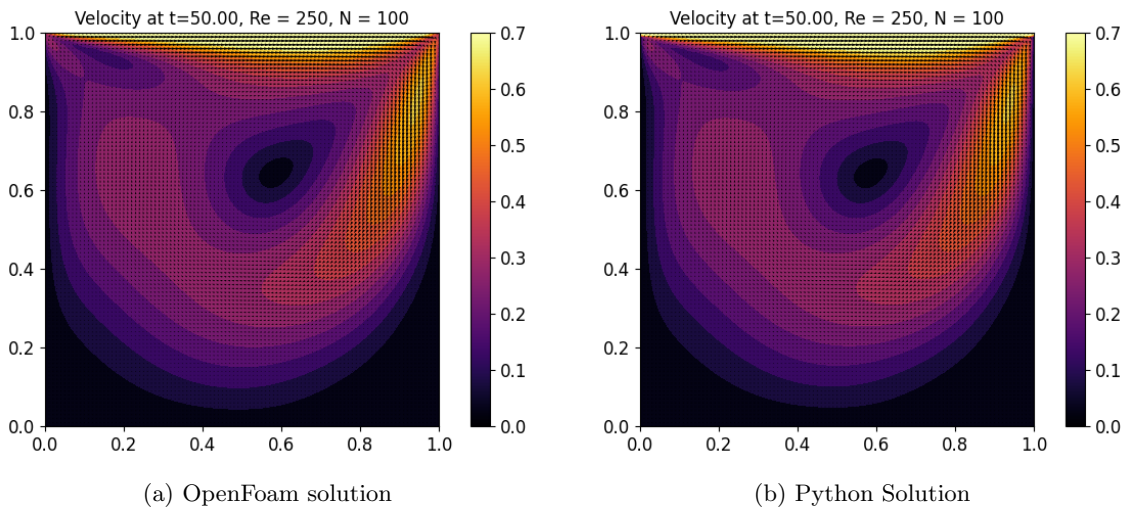


Figure 5: Case B

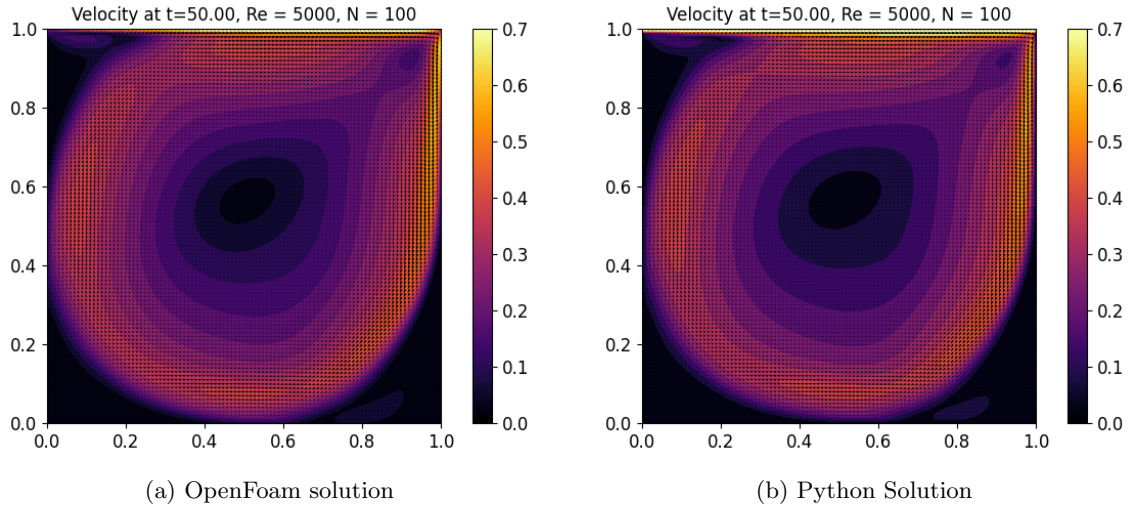


Figure 6: Case C

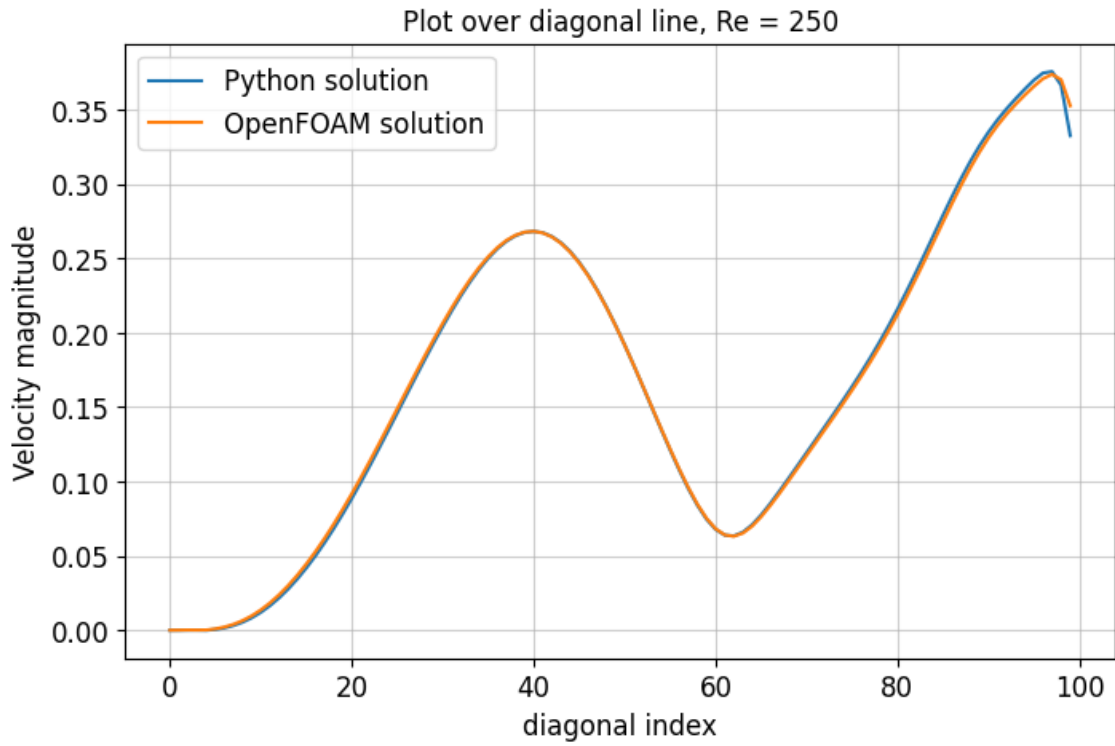


Figure 7: Plot over line (Case B): Python against OpenFOAM

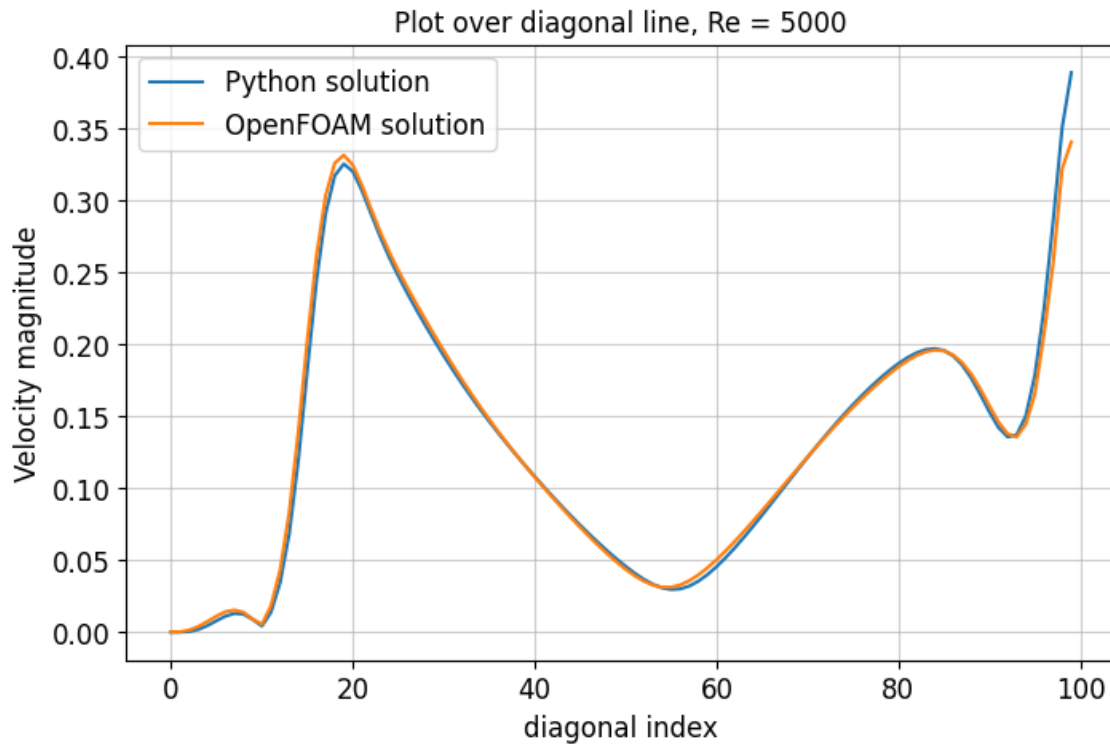


Figure 8: Plot over line (Case C): Python against OpenFOAM

We can conclude that the Python and OpenFOAM solutions converges, and notably the largest differences was found close to the boundaries.

Comparing the  $L_2$ -norm between the different steady solutions we got the following data:

Running case for  $Re = 25$   
 Relative error: 0.04912371466615295

Running case for  $Re = 250$   
 Relative error: 0.05397593894314182

Running case for  $Re = 5000$   
 Relative error: 0.09147688319684655

## Question 6

When also including a lower wall with the same velocity as the upper lid, the following solution was obtained at  $T = 50$ :



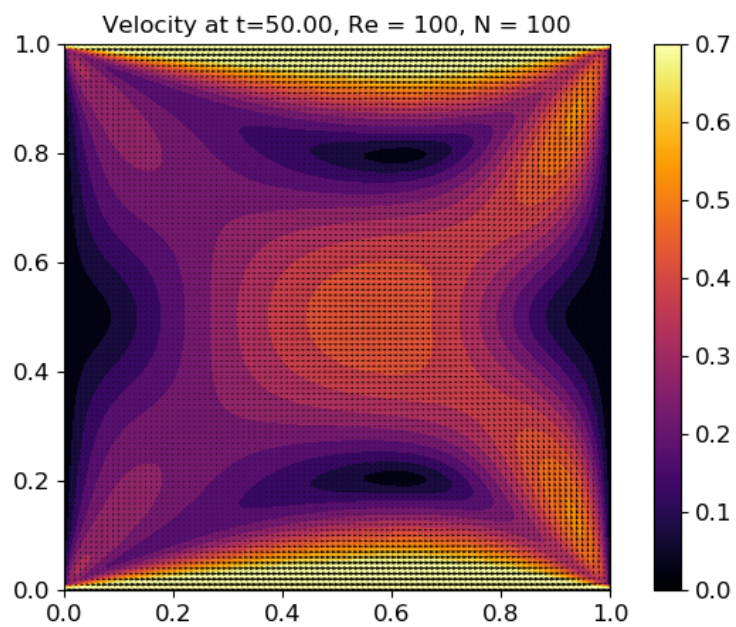


Figure 9: Case D