Simulating Flow through a Lid Driven cavity using Python & NumPy

Nipun Shashank Kothare
Satyam Singh Thakur
(Team – 48)

Abstract:
This python code solves a problem statement known as Lid Driven Cavity (LDC) defined in the fluid dynamics field. Fluid dynamic problems can be solved using computational methods known as CFD, and the algorithm for this problem is from [Sharma,2018]. Each algorithm step is defined in code blocks with the calling of functions wherever necessary. Since the computations are mathematically intensive on arrays, the numpy library is utilized. The computational process involves multiple iterative calculations; hence the focus is on achieving computational efficiency. The computed results are verified with benchmarked results from Ghia et. al.

Problem Statement:
Lid-driven cavity flow is probably the most commonly used problem for testing an in-house Navier-Stokes (NS) solver. This is shown in the figure as a square cavity with the left, right, and bottom walls as stationary. The top wall, called here the lid, acts as a long conveyor belt and moves horizontally with a constant velocity of U0. The motion results in a lid-driven recirculating flow inside the cavity. The cavity is represented by a closed 2D Cartesian square domain of size L1=L2, with all the boundaries as the solid walls. The figure also shows the initial and boundary conditions for the non-dimensional computational set-up of the problem. For a Reynolds number of 100 and a grid size of *42×42*, present and discuss a figure for the velocity-vector, velocity streamlines, and pressure-contour in the flow domain.

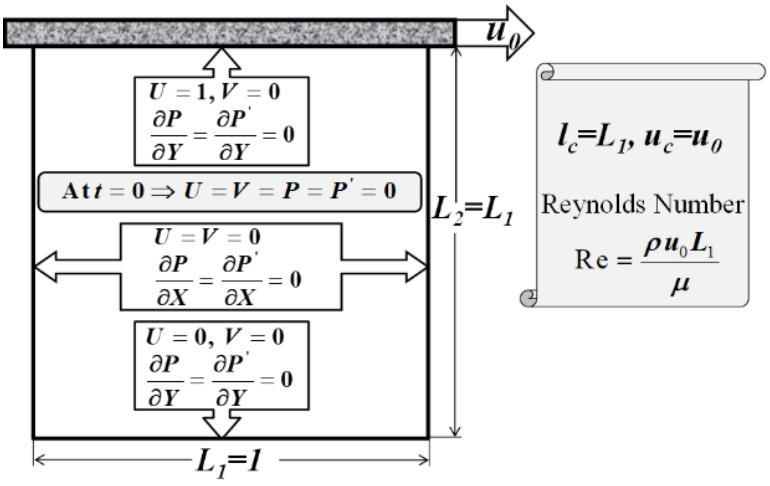


Figure 1: Computational domain and boundary conditions for the lid-driven cavity flow (left) & Expected output flow contour from benchmark results of Ghia et al. (right)

Scientific Methodology:

Using the flux-based solution methodology of the CFD for the FOU scheme, develop a python program for a 2D unsteady NS solver on a *uniform* staggered grid. The length of the cavity L_1 is the characteristic length, and the lid velocity U_0 is the characteristic velocity scale. The governing parameter for the isothermal flow is Reynolds number $Re=\rho U_0 L_1/\mu$. The Re is implemented in the code by a computational set-up as $\rho=U_0=L_2=1$ and $\mu=1/Re$. After the CFD development, run the code with a convergence tolerance of $\varepsilon_{st}=10^{-3}$ for the steady-state and $\varepsilon=10^{-8}$ for the mass-conservation.

Algorithm: The algorithm is based on Example 9.1 of Sharma [2017].

Since we are operating on a symmetric 2D grid problem, a 2D matrix is required to represent values of a scalar parameter at each point. For a 1D vector, each component has its own 2D matrix. Hence a 3D matrix is required to represent all values of a 1D vector. The third dimension points to different components of the vector with k=0 for x component, k=1 for y-component. The same logic can be extended to vectorization in 3D problems where k=2 for z-component. For even further advanced vector analysis, involving 2D Tensors, each element of the 2D tensor matrix itself holds a 2D matrix value for the grid points. Hence a 4D matrix is required for such components as we see in the algorithm.

$$u[i,j], v[i,j] \rightarrow u[i,j,k] \quad i,j \; define \; grid \; point \; location$$
$$k = 0 \; for \; u, 1 \; for \; v$$

Variables 50

x	7X1	u	6X7	mux	5X6	mvx	6X5
y	7X1	u_old	6X7	m+ux	5X6	m+vx	6X5
dx	6X1	u_p	6X7	m-ux	5X6	m-vx	6X5
dy	6X1	v	7X6	aux	5X6	avx	6X5
a	6X6	v_old	7X6	dux	5X6	dvx	6X5
mx	6X6	v_p	7X6	muy	5X6	mvy	6X5
my	6X6			m+uy	5X6	m+vy	6X5
mpx	6X6	p	7X7	m-uy	5X6	m-vy	6X5
mpy	6X6	p_old	7X7	auy	5X6	auy	6X5
Spm	6X6	p_p	7X7	duy	5X6	duy	6X5
mcx	6X6	p_c	7X7	Au	5X6	Av	6X5
mcy	6X6	p_c_old	7X7	Du	5X6	Dv	6X5
Scm	6X6			Su	5X6	Sv	6X5

Variables 24

x	7X2	u	7X7X2	mux	mvx	6X6X2X2	
y		v		muy	mvy		
dx	6X2	u_old	7X7X2	m+ux	m+vx	6X6X2X2	
dy		v_old		m+uy	m+vy		
mx	6X6X2	u_p	7X7X2	m-ux	m-vx	6X6X2X2	
my		v_p		m-uy	m-vy		
a	6X6			aux	avx	6X6X2X2	
mpx	6X6X2	p	7X7	auy	auy		
mpy		p_old	7X7	dux	dvx	6X6X2X2	
Spm	6X6	p_p	7X7	duy	duy		
mcx	6X6X2	p_c	7X7	Au	Av	6X6X2	
mcy		p_c_old	7X7	Du	Dv	6X6X2	
Scm	6X6			Su	Sv	6X6X2	

Figure 2 No. of arrays involved before & after vectorization of the algorithm

Accounting for high dimensions in vectors & grid structures, an effective tool is required to process multi-dimensional arrays. Numpy arrays are highly effective in this developing arrays of higher dimensions. They are also well-suited for matrix operations which is an intensive requirement for this algorithm. We are able to reduce the no. of variables to half by operating on multi-dimensional arrays. For the case of 4D arrays, 2 additional indices k & l are utilized.

```
for(i=1:imax-1); p(i,1)=p(i,2)
```

↓

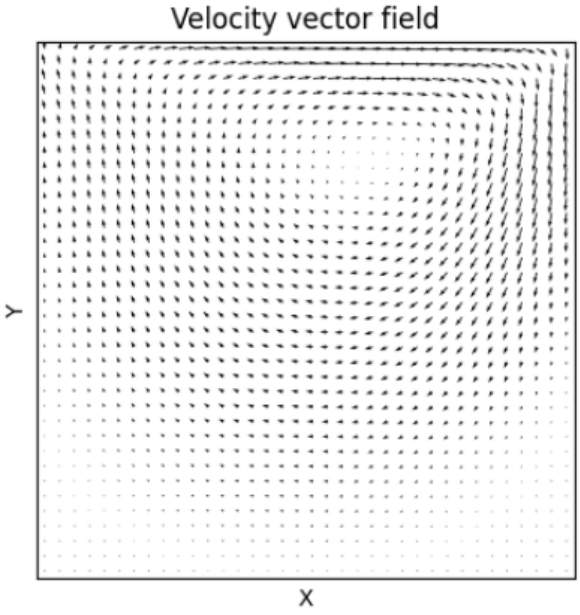
```
p(1:imax-1,1)=p(1:imax-1,2);
```

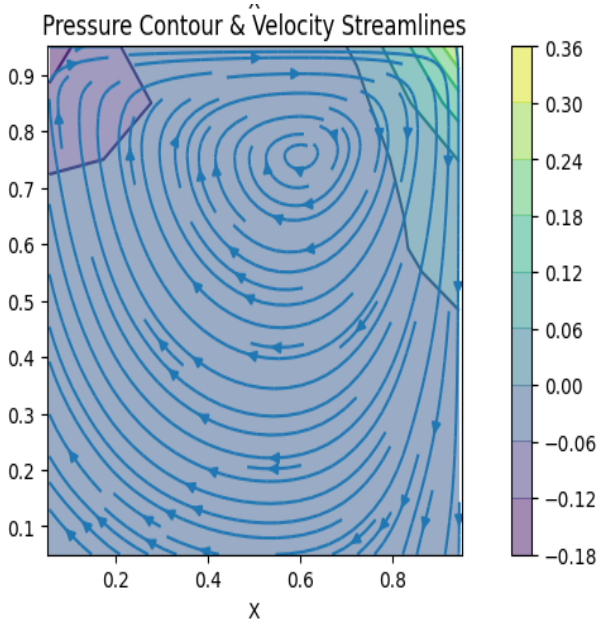
Grid	% Time Improve
7x7	95
12*12	97
22*22	98

Since the coding involved is computationally expensive, especially on larger grid sizes, it is critical to reduce computational load. The iterative nature of evaluation of the matrices in the algorithm are key to this high computational load. It is more efficient to utilize array indexing and hence the use of for-loops is minimized. There is a significant reduction in computational time due to indexing by up to 98%. The different stages involved n the algorithm are control parameters, initialization, grid generation, time step evaluation, predictor step, corrector step & finally plotting of the results.

Results & Observations:

We observe rotational flow in the lid driven cavity as expected. The flow moves from the top left corner to the top right corner and forms a vortex in the right half of the cavity. The flow in the remaining cavity is centred around this vortex. The U-velocity contour suggests that the U velocity is negative below the vortex with the lowest negative values achieved immediately below the vortex which observes an increased flow to compensate for the positive flow on the other side of the vortex in a mass balanced closed system. The pressure contour suggests a pressure gradient along the upper wall with a higher pressure at the top right and a lower pressure at the top left. Applying Bernoulli's on the 2 points, we will find that this pressure difference is to compensate for the effects of vorticity generated in the flow.





Conclusion:

Based on verification with benchmark results of Ghia[1982], the results obtained from the program are in compliance. It can be hence be concluded that the program can successfully simulate flow for the LDC problem.

References:

Sharma A., (2017), Introduction to Computational Fluid Dynamics: Development, Application, and Analysis, Ane Books Pvt. Ltd., New Delhi, Chapter 9, pp. 302-306.

Ghia U., Ghia K. N., and Shin C. T. (1982). High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method, J. Comp. Phys., vol. 48, pp. 387-411.