# Homework Assignment 5

Ran Li

April 24, 2018

## Problem 1: Direct Inverse Method for Poisson Equation

- Solving the following Poisson Equation:

$$\nabla^2 p = -8\pi^2 \cos(2\pi x)\cos(2\pi y), \quad x, y \in [0,1]\times[0,1] \tag{1}$$

with Neumann Boundary Condition:

$$\nabla p \cdot \mathbf{n} = 0 \tag{2}$$

According to this setup, Domain in which the problem is defined could be visulized as following picture:

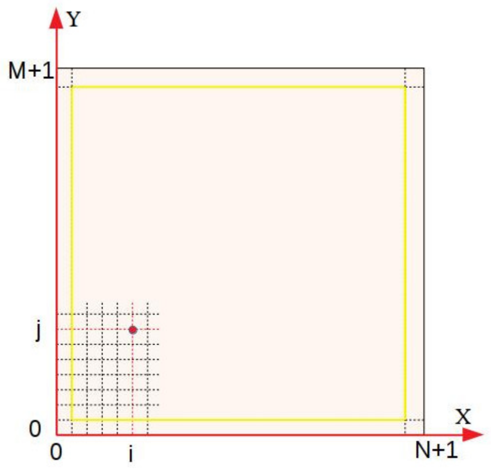

Figure 1: Physical Domain & Coresponding Computation Domain

- Algorithm based on Direct Inverse method is developed as follows:

1. Our original equaiton could be re-organized into matrix format, denote function on RHS as R, then:

$$D2x[p] + [p]D2y^T = R \tag{3}$$

2. Derive 1st order operator matrices $D1x,\ D1y$ with 2nd order accurate scheme. To ensure accuracy, for interior nodes, we applied central difference scheme:

$$\frac{\partial p}{\partial x}|_{i,j} = \frac{p_{i+1,j} - p_{i-1,j}}{2\Delta x},\ \frac{\partial p}{\partial y}|_{i,j} = \frac{p_{i,j+1} - p_{i,j-1}}{2\Delta y} \tag{4}$$

As for boundary nodes, we applied 2nd order one-sided difference scheme:

$$i = 1: \quad \frac{\partial p}{\partial x}|_1 = \frac{-3p_{1,j} + 4p_{2,j} - p_{3,j}}{2\Delta x} \tag{5}$$

$$i = N+1: \quad \frac{\partial p}{\partial x}|_{N+1} = \frac{3p_{N+1,j} - 4p_{N,j} + p_{N-1,j}}{2\Delta x} \tag{6}$$

$$j = 1: \quad \frac{\partial p}{\partial y}|_1 = \frac{-3p_{i,1} + 4p_{i,2} - p_{i,3}}{2\Delta y} \tag{7}$$

$$j = M+1: \quad \frac{\partial p}{\partial y}|_{M+1} = \frac{3p_{i,M+1} - 4p_{i,M} + p_{i,M-1}}{2\Delta y} \tag{8}$$

3. From discretization scheme above, derive 1st order difference operator matrices $D1x,\ D1y$ :

$$D1x = \begin{bmatrix} -3 & 4 & -1 & 0 & \dots & 0 \\ -1 & 0 & 1 & 0 & \dots & 0 \\ 0 & -1 & 0 & 1 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \dots & \dots & -1 & 0 & 1 \\ 0 & \dots & 0 & 1 & -4 & 3 \end{bmatrix}_{(N+1)\times(N+1)} \tag{9}$$

$$D1y = \begin{bmatrix} -3 & 4 & -1 & 0 & \dots & 0 \\ -1 & 0 & 1 & 0 & \dots & 0 \\ 0 & -1 & 0 & 1 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \dots & \dots & -1 & 0 & 1 \\ 0 & \dots & 0 & 1 & -4 & 3 \end{bmatrix}_{(M+1)\times(M+1)} \tag{10}$$

4. Using $D1x,\ D1y$ , derive 2nd order difference operator matrices $D2x,\ D2y$:

$$D2x = D1xD1x; \quad D2y = D1yD1y \tag{11}$$

5. Since direct inverse method takes only interior grid information into calculation, 2nd order difference operator matrices $D2x,\ D2y$ need to be modified to include influence from boundary conditions. Assume that elements in $D2x$ are denoted as $a_{ij}$, then following modification must be applied according to Neumann boundary condition:

$$a^*_{22} \ = \ a_{22} + \frac{4}{3}a_{21},\ a^*_{23} = a_{23} - \frac{1}{3}a_{21},\ a^*_{32} = a_{32} + \frac{4}{3}a_{31},\ a^*_{33} = a_{33} - \frac{1}{3}a_{31}$$

$$a^*_{N-1,N-1} = a_{N-1,N-1} - \frac{1}{3}a_{N-1,N+1},\ a^*_{N-1,N} = a_{N-1,N} + \frac{4}{3}a_{N-1,N+1}$$

$$a^*_{N,N-1} = a_{N,N-1} - \frac{1}{3}a_{N,N+1},\ a^*_{N,N} = a_{N,N} + \frac{4}{3}a_{N,N+1} \tag{12}$$

Same operation shall be applied to $D2Y$.

6. Triangulate the modified 2nd order difference operator matrices $D2x,\ D2y$ and we will get eigenvalue matrices $\Lambda_x,\ \Lambda_y$ and eigenvector matrices $P,\ Q$ , from which we could acquire modified $p$ and $R$:

$$\hat{p} = P^{-1}pQ,\quad \hat{R} = P^{-1}RQ \tag{13}$$

7. From equation in step 1 we know that modified $p$ and $R$ shall satisify:

$$\Lambda_x[\hat{p}] + [\hat{p}]\Lambda_y = \hat{R} \tag{14}$$

which indicate a clear and easy set of equations:

$$\hat{p}_{i,j} = \frac{\hat{R_{i,j}}}{\lambda_x^i + \lambda_y^j} \tag{15}$$

8. Eventually from $\hat{p}$, we could derive the actual full solution to the problem:

$$p = P\hat{p}Q^{-1} \tag{16}$$

- Compare Numerical Result with Analytical Solution:

1. Comparison of overall solutions:

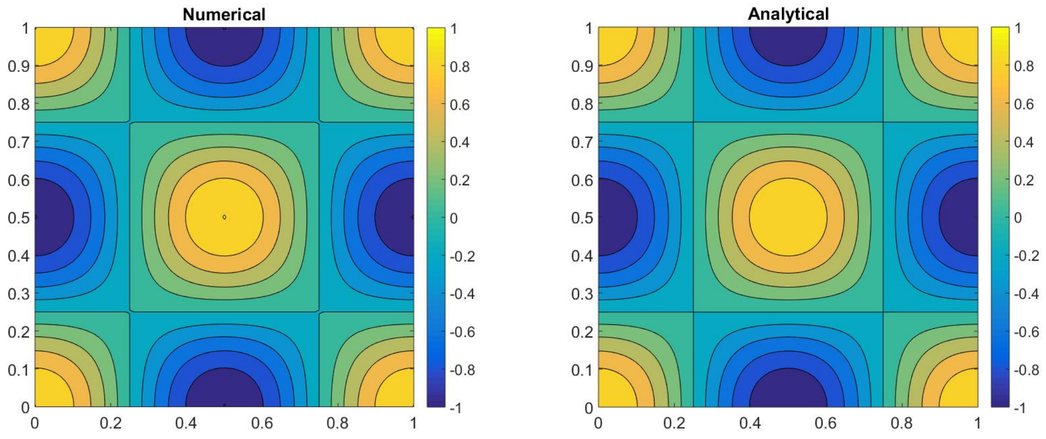

Figure 2: Comparison of Solutions from numerical simulation and analytical approach

2. Comparison of 2 cross-sections:

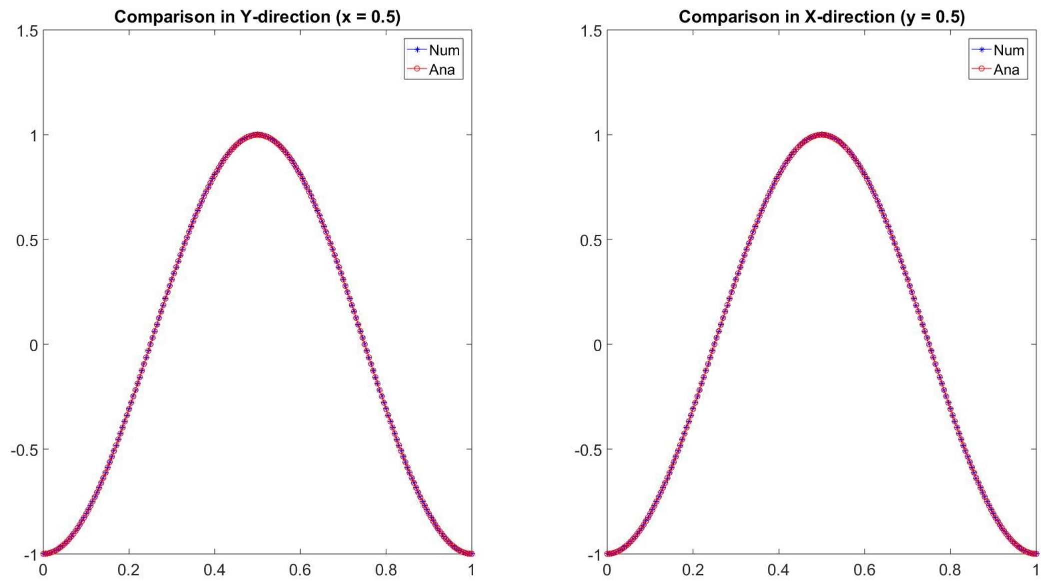

Figure 3: Comparison of results along 2 directions

- MATLAB Code for this problem is attached at the end of this report.

## Problem 2: Time-split Method for Pressure-Velocity formulated N-S Equation

- Apply following time-split method to solve N-S equation

Predictor momemtum equation is formulated as what remains after we pick out pressure gradient term from original N-S equation:

$$\frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} = \frac{1}{2}\left(3\mathbf{N}^n - \mathbf{N}^{n-1}\right) + \frac{\nu}{2}\left(\nabla^2 \mathbf{u}^* + \nabla^2 \mathbf{u}^n\right) \tag{17}$$

This predictor momemtum equation shall satisfy boundary conditions:

$$\mathbf{u} * \cdot \mathbf{n} = \mathbf{u}_B \cdot \mathbf{n},\ \mathbf{u}^* \cdot \mathbf{t} = \left(\mathbf{u}_B + \frac{\Delta t}{\rho}\nabla\phi^{n+1}\right) \cdot \mathbf{t} \tag{18}$$

Pressure Poisson Equation (PPE) which act as corrector is formulated as follow: ($\phi$ is actually pseudo pressure)

$$\nabla^2 \phi^{n+1} = \frac{\rho}{\Delta t}\nabla \cdot \mathbf{u}^*,\ \nabla\phi^{n+1} \cdot \mathbf{n} = 0 \tag{19}$$

Correction step is described as:

$$\mathbf{u}^{n+1} = \mathbf{u}^* - \frac{\rho}{\Delta \mathrm{t}} \nabla \phi^{n+1} \tag{20}$$

- Algorithm formulation is described down below:

1. Non-dimensionalize the problem first with scales enlisted below:

$$\mathbf{u}, \mathbf{v} \sim V,\ x, y \sim L,\ \Delta t \sim T = \frac{L}{V} \tag{21}$$

   Thus the momentum equation17 becomes:

$$\frac{V}{T} \frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} = \frac{V^2}{L} \frac{1}{2} \left(3\mathbf{N}^n - \mathbf{N}^{n-1}\right) + \frac{\nu V}{L^2} \frac{1}{2} \left(\nabla^2 \mathbf{u}^* + \nabla^2 \mathbf{u}^n\right)$$

   Thus we could acquire:

$$\frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} = \frac{1}{2} \left(3\mathbf{N}^n - \mathbf{N}^{n-1}\right) + \frac{\nu}{LV} \cdot \frac{1}{2} \left(\nabla^2 \mathbf{u}^* + \nabla^2 \mathbf{u}^n\right),\ Re = \frac{LV}{\nu} \tag{22}$$

   Eventually:

$$\frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} = \frac{1}{2} \left(3\mathbf{N}^n - \mathbf{N}^{n-1}\right) + \frac{1}{2Re} \left(\nabla^2 \mathbf{u}^* + \nabla^2 \mathbf{u}^n\right) \tag{23}$$

2. Re-organize the momentum equation into matrix format:

$$\left(I - \frac{\Delta t}{2Re} A_x\right) \left(I - \frac{\Delta t}{2Re} A_y\right) \left(\mathbf{u}^* - \mathbf{u}^n\right) = \frac{\Delta t}{2} \left[3\mathbf{N}^n - \mathbf{N}^{n-1}\right] + \frac{\Delta t}{Re} \nabla^2 \mathbf{u}^n \tag{24}$$

3. Intialize the 2 velocity fields and 1 pseudo pressure field as all 0. Then change x-direction velocity along the top to be 1. Computational domain set up is alike to the one we used for problem 1. (figure down below)

4. Set up time step length $dt$ and maximum time $t_{max}$ this simulation would reach. Define all difference operator matrices to be applied in our computation:

5. **START - Loop 1:** While $t_{current} \leq t_{max}$, then initialize computational margins

   (a) Static Solver: $\bar{Error} = 1$ to enter the next loop.

   (b) Transient Solver: $Counter = 1$

6. **START - Loop 2:**

   (a) Static Solver: While $\bar{Error} \leq Margin$, in my code I set the margin to $10^{-8}$.

   (b) Transient Solver: While $Counter \leq 100$

Hint on Transient: As trial run suggested, after around 100 steps of calculation for each time step, residue (pressure gradient term $\nabla\phi$) would converge. So I run 100 steps for each time step $n$

7. Compute RHS of momemtum equation24. This process includes calculating non-linear convection term $\mathbf{N}^n$ and $\mathbf{N}^{n+1}$ and linear viscous term $\frac{\Delta t}{Re}\nabla^2\mathbf{u}^n$ with initial velocity field.
8. Apply Thomas's Algorithm to solve implicit equation24, this would advance the time from $t_n$ to $t_{n+1/3}$ and then $t_{n+2/3}$ which means going from $\mathbf{u}^n$ to $\mathbf{u}^*$. Code for Thomas solver is attached at the back of this report.
9. Derive divergence term $\nabla\cdot\mathbf{u}^*$ and then derive the full RHS of PPE19. Use Direct Inverse method developed in problem 1 to solve for updated pressure corrector $\phi^{n+1}$.
10. Correct $\mathbf{u}^*$ with $\phi^{n+1}$ using correction equation20, then we would acquire $\mathbf{u}^{n+1}$.
11. Update iteration condition:
    (a) Static Solver: $Er\bar{r}or = max(|\mathbf{u}^{n+1} - \mathbf{u}^n|)$
    (b) Transient Solver: $Counter = Counter + 1$
12. **End - Loop 2**
13. Increase iteration step number: $n = n + 1$. Take record of velocity field $\mathbf{u}$ and pseudo pressure field $\phi$ . Update non-linear convection term $\mathbf{N}^{n-1}$ to $\mathbf{N}^n$ in last step.
14. **End - Loop 1**
15. Post-processing of results.

- Comparison with Results Provided:

1> Steady State Solution:

According to problem requirement, velocity$u$, $v$ and vorticity $\omega$ along 3 direction in the computational field are compared. Please note that boundary conditions are not exactly the same in 2 cases, reference case have driving velocity of 100 will my simulation is non-dimensionalized so 100 in referencial figure corresponds to 1 in simulation.

Horizontal Velocity Component $u$:

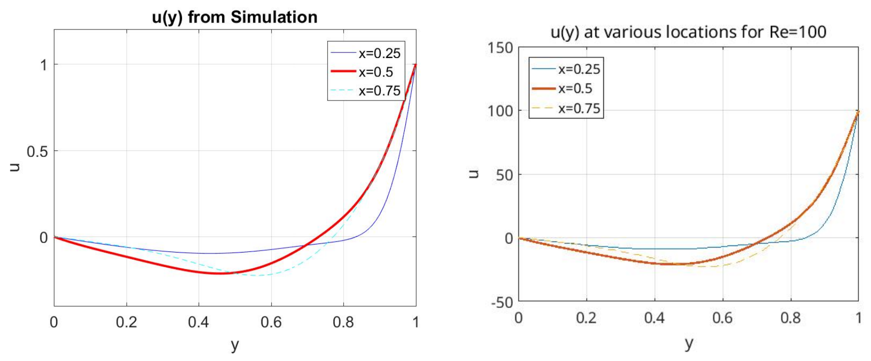

Figure 4: Steady State: Horizontal Velocity Compare

Vertical Velocity Component $v$:

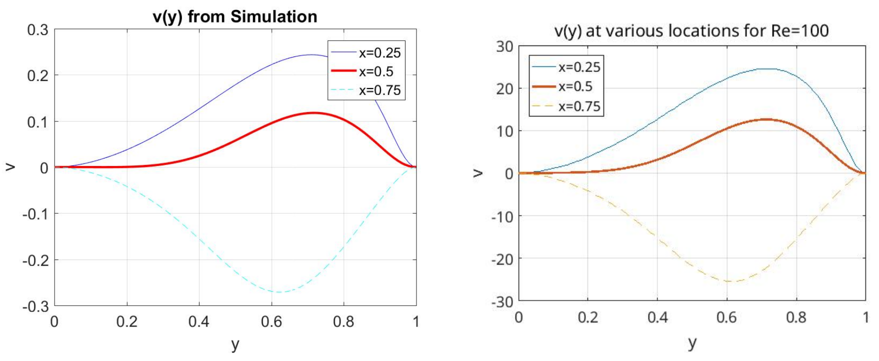

Figure 5: Steady State: Horizontal Velocity Compare

Vorticity $\omega$:

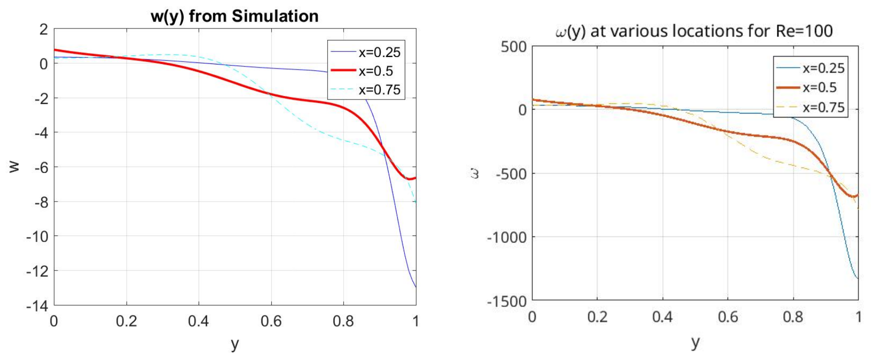

Figure 6: Steady State: Vorticity Compare

Also to better illustrate the final flow field, stream function contour $\psi$ and vorticity field $\omega$ are generated. Stream function is calculated using a online MATLAB code found from: http://pordlabs.ucsd.edu/matlab/stream. It's generated by Dr. Kirill Pankratov to compute stream function using given velocity field.

Stream function contour:

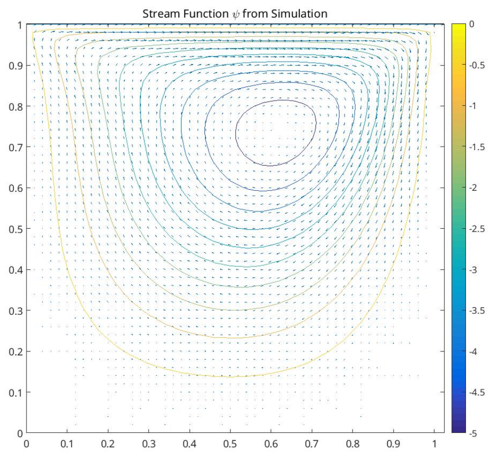

Figure 7: Stream function contour from Ran's Simulation

Vorticity contour:

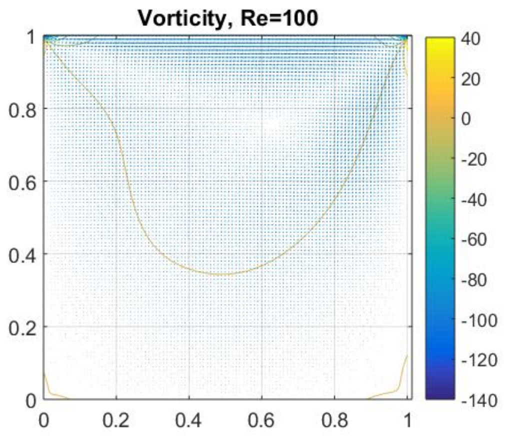

Figure 8: Vorticity contour from Ran's Simulation

2> Transient Solution:

As required by the problem, change of velocity components at given location in flow field w.r.t. time is plotted against reference solution. The comparison is shown down below:

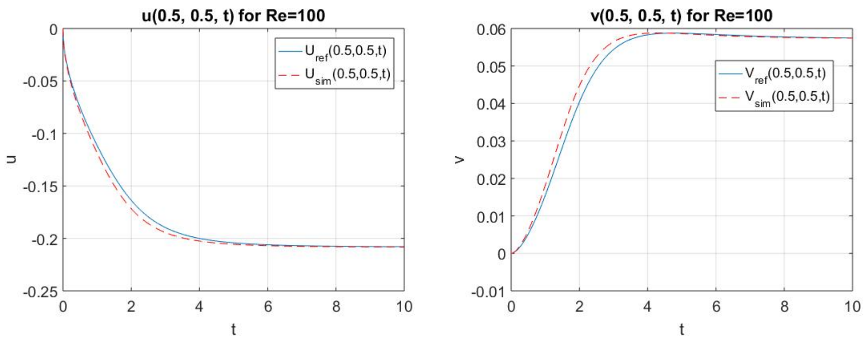

Figure 9: Comparison of velocity change w.r.t. time at location $(0.5, 0.5)$