Istanbul Technical University
Naval Architecture and Marine Engineering
UUM 601E - Advanced Computational Fluid Dynamics

Project - 1

Bugra Ugur YAZICI
508182010

Lecturer:
Prof.Mehmet Sahin

Istanbul, December 2020

Table of Contents

Introduction

In this report, backward step geometry is analysed with the incompressible Navier-Stokes equations that govern the incompressible viscous fluid flow. MAC scheme is implemented to solve both the Stokes flow and Navier-Stokes flow at Re=100 within the backward step geometry with the dimensions of $[0, 5] \times [0, 1]$.

In the **Numerical Method** section, geometry definition, governing equations, boundary and initial conditions of the problem will be defined. After that discretization of the incompressible Navier-Stokes equations using MAC scheme is explained with the supplementary figures.

In the **Numerical Results** section, Stokes flow and Navier-Stokes flow at Re=100 is compared using different grid sizes. Once results are computed and satisfactory, u-velocity component solutions are compared along (x = 3) line with the results of Erturk [1] at Re = 100. Moreover, the data available in NINOVA (X=3_PROFILE_RE=100.dat and X=3_PROFILE_STOKES.dat) will be used for validation purposes. In addition to these validation graphs, the contour plots of primitive variables are provided for all cases as well as streamlines. Since quasi-steady solution for the Re=100 is implemented, different incremental values of the reynolds number are investigated whether these delta values have an effect on the final solution.

c++ programming language and PETSc Library are used to create coefficient matrix and using a parallel sparse direct solver MUMPS: MUltifrontal Massively Parallel sparse direct Solver obtain numerical results. Installation process, observations while implementing native PETSc direct and iterative solvers and MUMPS library will be explained in order to reduce the burden for new users.

1. Numerical Method

Geometry and Boundary Conditions

Geometry of the problem can be seen in Figure-1.1. Total unitless dimension of the domain is expressed as $[0, 5] \times [0, 1]$. Therefore; it has a rectangular shape whose longer edge is located at x axis.

Boundary conditions are:

$$u(0, y) = -24(1 - y)(0.5 - y) \; where \; y > 0.5 \; v(0, y) = 0 \tag{1.1}$$

$$u(0, y) = 0 \; where \; y \leq 0.5 \; v(0, y) = 0 \tag{1.2}$$

$$u(x, 0) = 0 \; v(x, 0) = 0 \tag{1.3}$$

$$u(x, 1) = 0 \; v(x, 1) = 0 \tag{1.4}$$

$$u_x|_{x=5} = p \; v_{xx}|_{x=5} = 0 \tag{1.5}$$

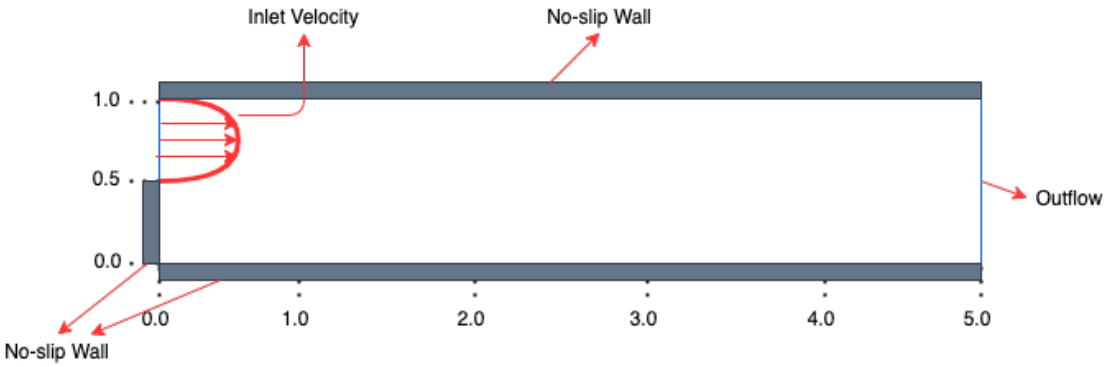

Figure 1.1: Problem Geomery and BCs

Eq.(1) corresponds with the *Inlet Velocity*
Eq.(2) corresponds with the *Left edge of the domain where* $y \leq 0.5$ where No-slip Wall boundary condition is implemented.
Eq.(3) corresponds with the *Bottom edge of the domain* where No-slip Wall boundary condition is applied.
Eq.(4) corresponds with the *Top edge of the domain* where No-slip boundary condition is applied.
Finally, Eq.(5) corresponds with the *Right edge of the domain* where outflow boundary condition is applied.

Incompressible Navier Stokes Equation

The incompressible Navier-Stokes equations that govern the incompressible viscous fluid flow in the Cartesian coordinate system can be written in dimensionless form in finite-difference approximations as follows:

Continuity equation

$$\underbrace{\frac{u_{i+1,j} - u_{ij}}{\Delta x}}_{\frac{\partial u}{\partial x}} + \underbrace{\frac{v_{i,j+1} - v_{ij}}{\Delta y}}_{\frac{\partial v}{\partial y}} = 0$$

Momentum Equation Along the x-axis

$$Re\underbrace{\frac{u_{i,j}^{n+1}-u_{i,j}^{n}}{\Delta t}}_{\frac{\partial u}{\partial t}}+Re\left[\underbrace{u_{i,j}^{n}\underbrace{\frac{u_{i+1,j}^{n+1}-u_{i-1,j}^{n+1}}{2\Delta x}}_{\frac{\partial u}{\partial x}}}_{\frac{\partial (uu)}{\partial x}}\right]+Re\left[\underbrace{\overline{v}_{i,j}^{n}\underbrace{\frac{u_{i,j+1}^{n+1}-u_{i,j-1}^{n+1}}{2\Delta y}}_{\frac{\partial u}{\partial y}}}_{\frac{\partial (uv)}{\partial x}}\right]+\underbrace{\frac{P_{i,j}^{n+1}-P_{i-1,j}^{n+1}}{\Delta x}}_{\frac{\partial P}{\partial x}}$$

$$=\underbrace{\frac{u_{i+1,j}^{n+1}-2u_{i,j}^{n+1}+u_{i-1,j}^{n+1}}{\Delta x^2}}_{\frac{\partial^2 u}{\partial x^2}}+\underbrace{\frac{u_{i,j+1}^{n+1}-2u_{i,j}^{n+1}+u_{i,j-1}^{n+1}}{\Delta y^2}}_{\frac{\partial^2 u}{\partial y^2}}$$

$$\overline{v}_{i,j}^{n}=\frac{v_{i,j}^{n}+v_{i,j+1}^{n}+v_{i-1,j+1}^{n}+v_{i-1,j}^{n}}{4}$$

Momentum Equation Along the y-axis

$$Re\underbrace{\frac{v_{i,j}^{n+1}-v_{i,j}^{n}}{\Delta t}}_{\frac{\partial v}{\partial t}}+Re\left[\underbrace{\overline{u}_{i,j}^{n}\underbrace{\frac{v_{i+1,j}^{n+1}-v_{i-1,j}^{n+1}}{2\Delta x}}_{\frac{\partial v}{\partial x}}}_{\frac{\partial (uv)}{\partial x}}\right]+Re\left[\underbrace{v_{i,j}^{n}\underbrace{\frac{v_{i,j+1}^{n+1}-v_{i,j-1}^{n+1}}{2\Delta y}}_{\frac{\partial v}{\partial y}}}_{\frac{\partial (vv)}{\partial y}}\right]+\underbrace{\frac{P_{i,j}^{n+1}-P_{i,j-1}^{n+1}}{\Delta y}}_{\frac{\partial P}{\partial y}}$$

$$=\underbrace{\frac{v_{i+1,j}^{n+1}-2v_{i,j}^{n+1}+v_{i-1,j}^{n+1}}{\Delta x^2}}_{\frac{\partial^2 v}{\partial x^2}}+\underbrace{\frac{v_{i,j+1}^{n+1}-2v_{i,j}^{n+1}+v_{i,j-1}^{n+1}}{\Delta y^2}}_{\frac{\partial^2 v}{\partial y^2}}$$

$$\overline{u}_{i,j}^{n}=\frac{u_{i,j}^{n}+u_{i+1,j}^{n}+u_{i+11,j-1}^{n}+u_{i,j-1}^{n}}{4}$$

In these equations (u,v) represents the velocity vector components, p is the pressure and Re is the dimensionless Reynolds number. The primitive variables can be arranged as shown in the next section Figure 1.2.

Local Numbering of Primitive Variables

In order to define a local numbering that is consistent with the starting index of common scientific programming languages such as Python and C++, first index of the primitive variables starts from zero. It is much more easier to debug the code and find errors in this numbering. However,

this numbering lacks in computing right number of the element/finite difference points. Therefore; total number of the discretized domain is different from primitive indexing and starts from one. Local numbering of the primitive variables and global number of points are shown in Figure 1.3. Since MAC scheme is staggered, distribution of the local variables are different from the collocated schemes.

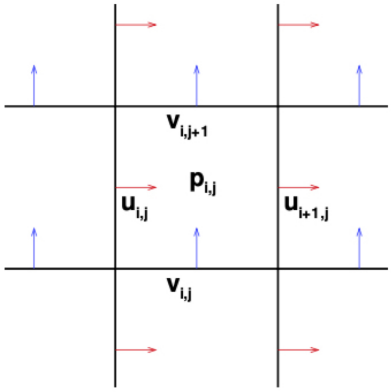

Figure 1.2: The arrangement of primitive variables.

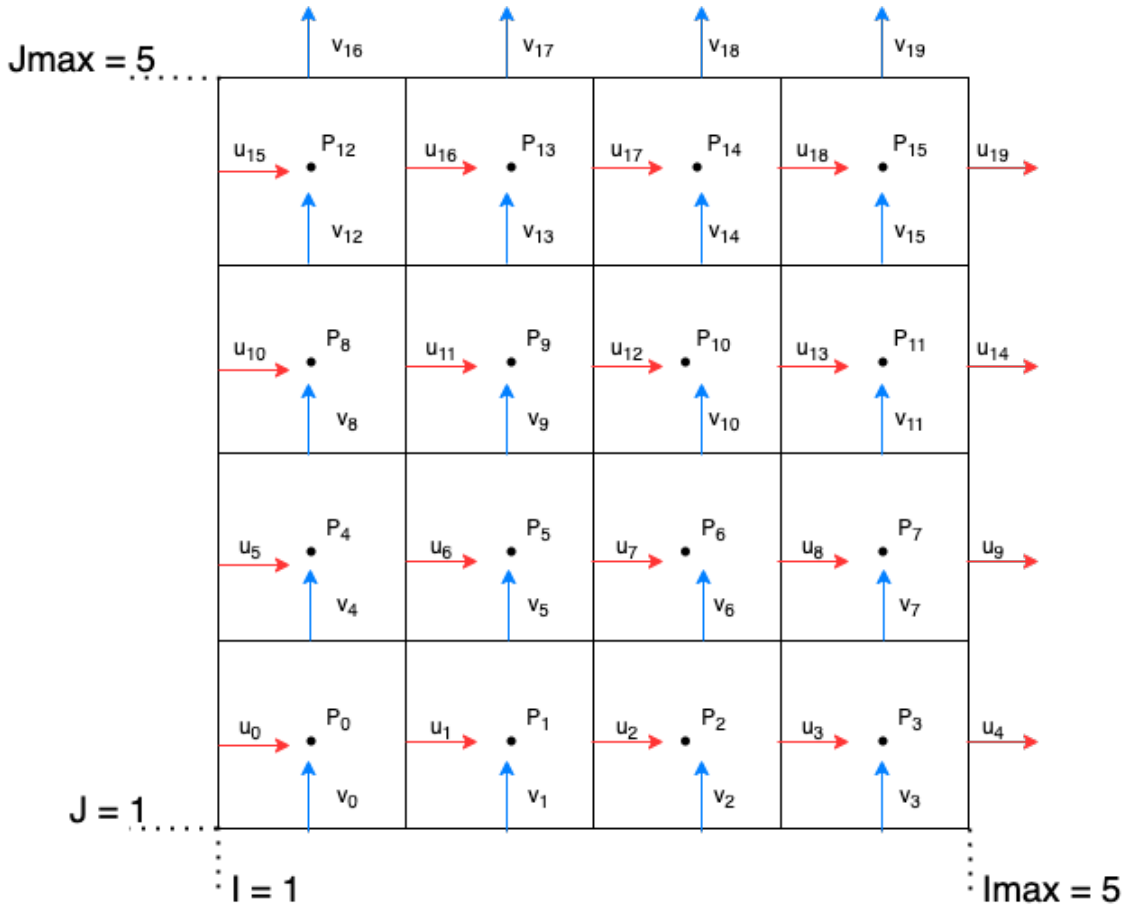

Figure 1.3: Local numbering of the primitive variables in the Project

2. Numerical Results

Before starting numerical analyses, coefficient matrix and RHS should be checked in order to prevent code to blow up. For 2D Navier Stokes problem, coefficient matrix, primitive variables and RHS take the form as follows:

$$\begin{bmatrix} A11 & A12 & A13 \\ A21 & A22 & A23 \\ A31 & A32 & A33 \end{bmatrix} \quad \begin{bmatrix} u \\ v \\ p \end{bmatrix} \quad = \quad \begin{bmatrix} RHS \\ \\ \end{bmatrix}$$

In Figure 2.1, pattern of the coefficient matrix for $I_{max} = 5$ and $J_{max} = 5$ can be found. Using these values, it can easily calculated that dimensions of the matrix is 33x33.

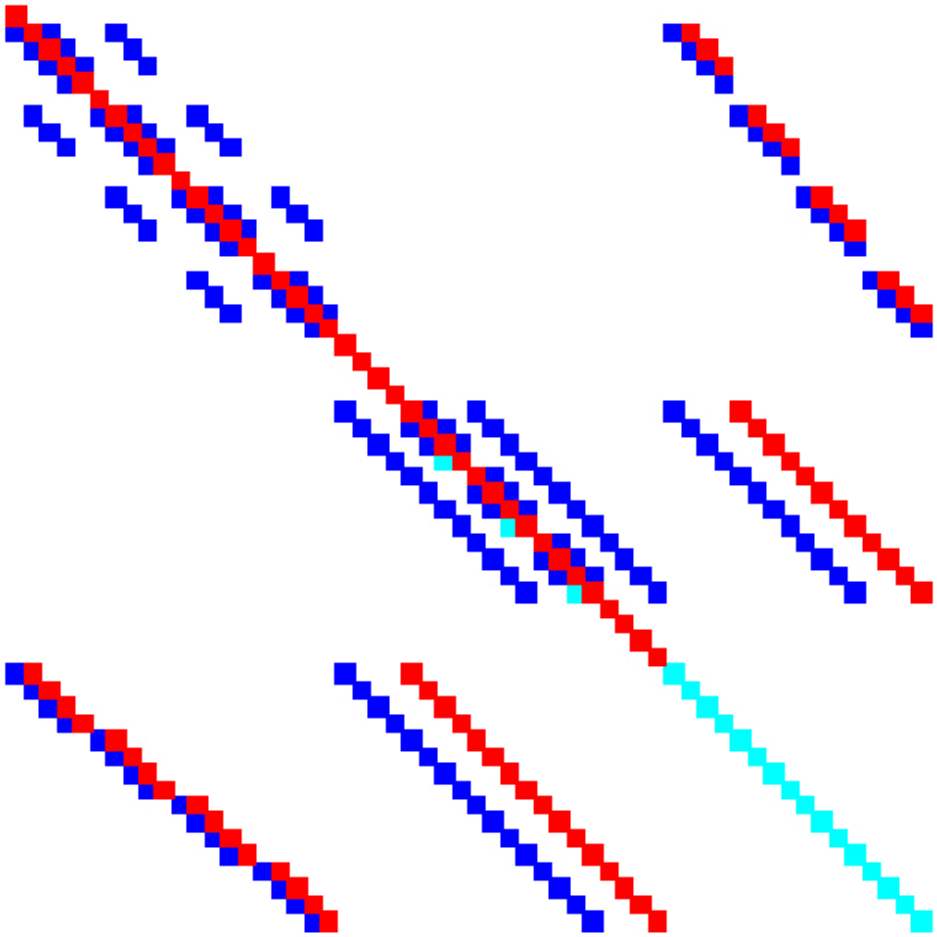

Figure 2.1: PETSc output of the pattern for coefficient matrix with the dimensions of 33x33.

In Figure 2.1, red boxes mean that corresponding matrix element is equivalent to 1, blue boxes mean all the values except for 0 and 1 and cyan boxes mean value within the element is zero. Even if the matrix is sparse and all empty elements should not be kept in the memory, PETSc gives following error while trying to solve A.x = RHS :

```
[0]PETSC ERROR: Matrix is missing diagonal entry 12960
```

In order not to have this type of error, diagonal entries that have the value of zero should be explicitly defined. Therefore; this type of entries are seen as cyan in the Figure 2.1.

Native PETSc Solvers

Many linear sparse matrix solver options have been implemented so as to obtain correct results. In Table 2.1, list of the solvers and preconditioners used within this project is given. It is concluded that PETSc direct and iterative solvers fail for this case. However, third-party external parallel direct solver MUMPS gives accurate and robust results for the backward step problem defined in the Section 1.

Grid Size	Solver	Precondition	Results	Comments
3x3	PETSc Direct Solver	LU	Success	-
3x3	PETSc Direct Solver	Cholesky	Success	-
5x5	PETSc Direct Solver	LU	Fail	All elements are inf. in solution matrix
5x5	PETSc Direct Solver	Cholesky	Fail	Gives correct results for both u and P values, fails at v.
5x5	PETSc GMRES Solver	Jacobi	Fail	Gives correct results for both u and P values, fails at v.
5x5	PETSc GMRES Solver	ILU	Fail	Gives correct results for both u and P values, fails at v.
5x5	PETSc GMRES Solver	ICC	Fail	Gives correct results for both u and P values, fails at v.
81x81	PETSc Direct Solver	LU/Cholesky	Fail	All elements are inf. in solution matrix
81x81	PETSc GMRES Solver	All	Fail	Relative residuals are increasing after each iteration.
All	MUMPS Direct Solver	LU/Cholesky	Success	-

Table 2.1: Outlook of the different types of solvers/preconditioners implemented in the project.

In order not to use $PETSC\_DEFAULT$ values for the Krylov Solvers of the PETSc, Relative and Absolute Errors of the matrix are explicitly defined as 1e-10 and 1e-12, respectively. Output of the code for 81x81 grid using GMRES iterative solver with Jacobi preconditioner after 10.000 iterations is as follows:

```
10000 KSP Residual norm 6.804800537856e+00
KSP Object: 1 MPI processes
  type: gmres
    restart=30, using Classical (unmodified) Gram-Schmidt Orthogonalization
    with no iterative refinement
    happy breakdown tolerance 1e-30
  maximum iterations=10000, initial guess is zero
  tolerances:  relative=1e-10, absolute=1e-12, divergence=10000.
  left preconditioning
  using PRECONDITIONED norm type for convergence test
PC Object: 1 MPI processes
  type: jacobi
  linear system matrix = precond matrix:
  Mat Object: 1 MPI processes
    type: seqaij
    rows=19360, cols=19360
    total: nonzeros=120644, allocated nonzeros=286400
    total number of mallocs used during MatSetValues calls=12640
      not using I-node routines
```

Even if iterative refinement option is selected as "refine_always", no improvements in the residual norm is observed. When 1.e-10 is defined explicitly in the zero diagonals of A33 block, residual norm becomes even worse with the value of 6.928204414295e+00.

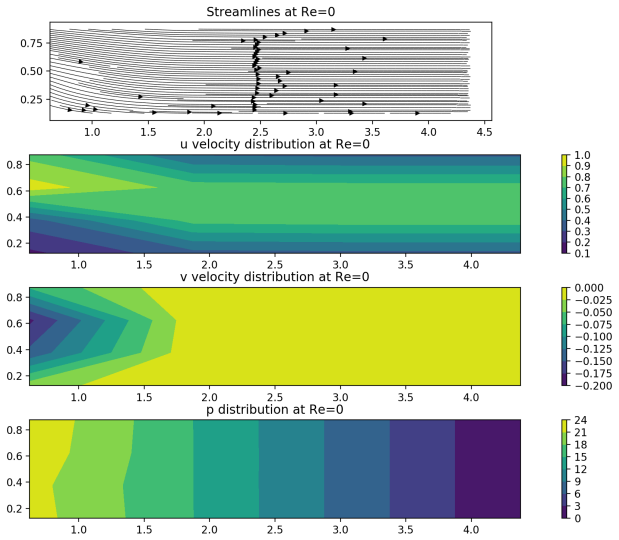

Figure 2.2: Results of GMRES with Jacobi Preconditioner for 5x5 grid.

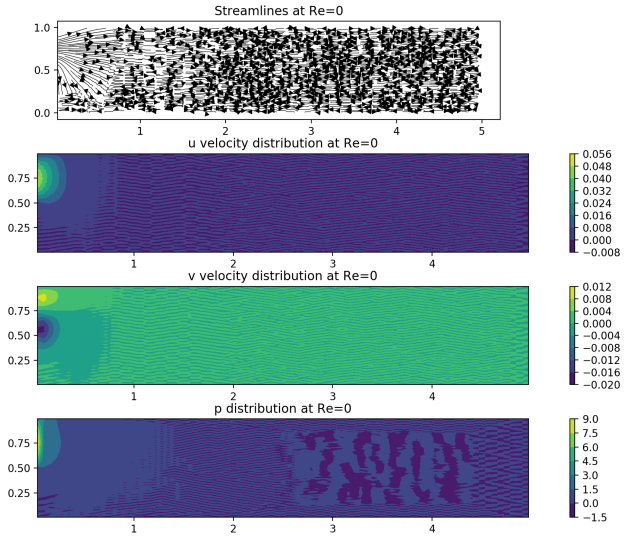

Figure 2.3: Results of GMRES with Jacobi Preconditioner for 81x81 grid.

PETSc External Solver: MUMPS

Different grid sizes for Stokes solution are defined to examine the grid size effect on the results. Therefore; u-velocity components @x=3.0 for four different grids of 41x41, 81x81, 161x161 and 201x201 are compared with the result that is shared on Ninova. Following results are obtained PETSc external direct solver library MUMPS with LU factorization.

Results of Stokes Flow (Re=0)

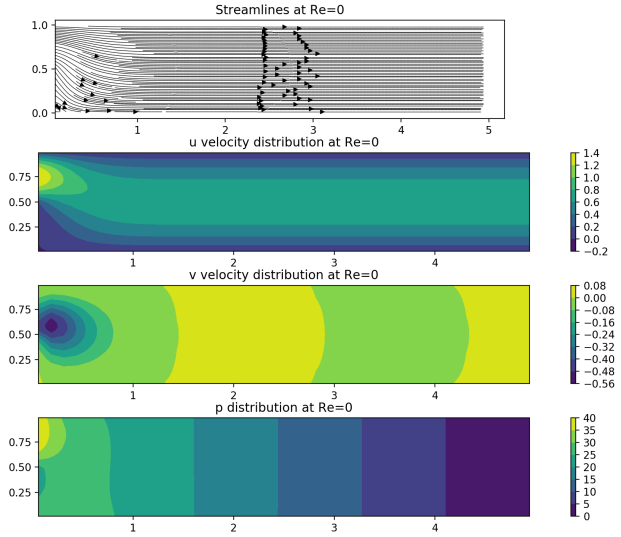

Figure 2.4: Stokes flow results of MUMPS with LU Preconditioner for 41x41 grid.

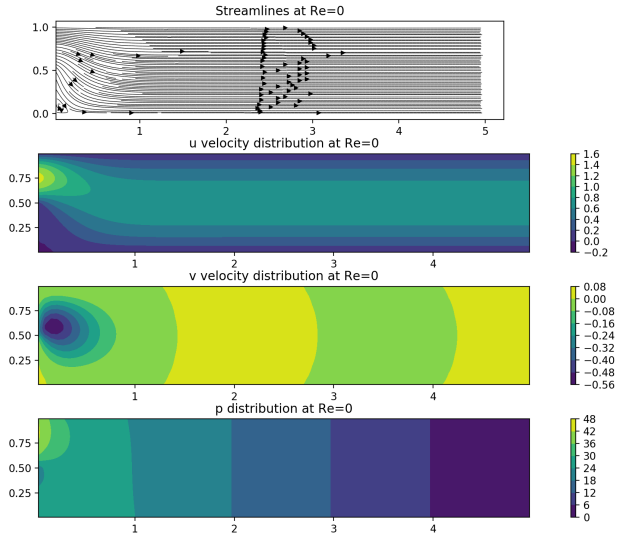

Figure 2.5: Stokes flow results of MUMPS with LU Preconditioner for 81x81 grid.

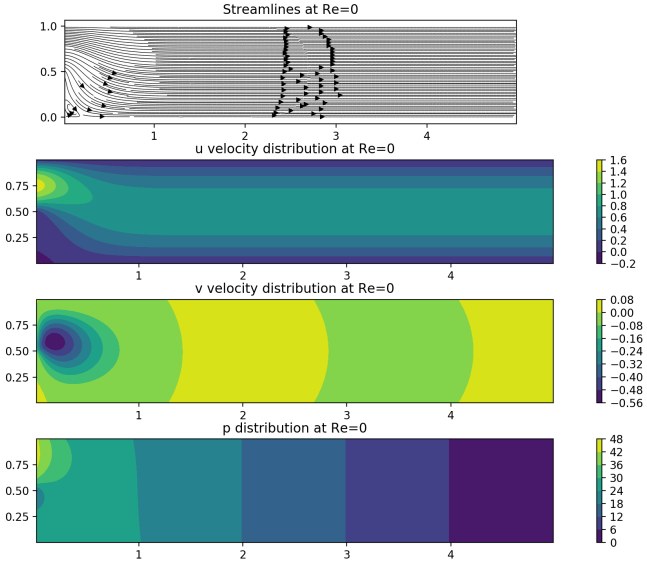

Figure 2.6: Stokes flow results of MUMPS with LU Preconditioner for 161x161 grid.

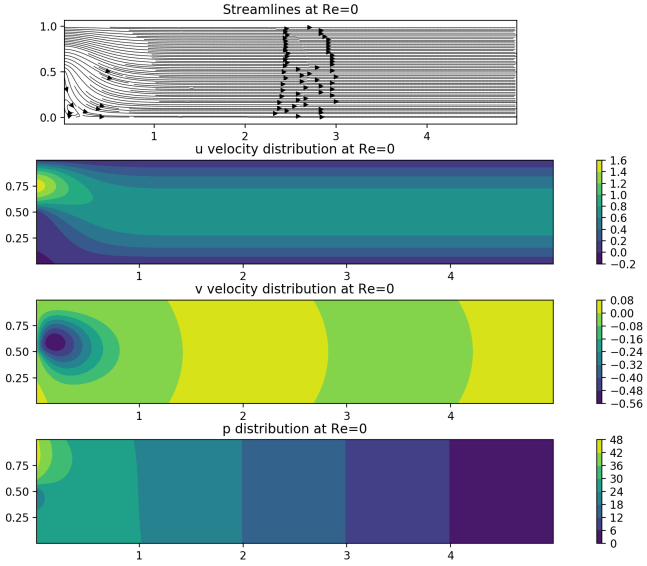

Figure 2.7: Stokes flow results of MUMPS with LU Preconditioner for 201x201 grid.

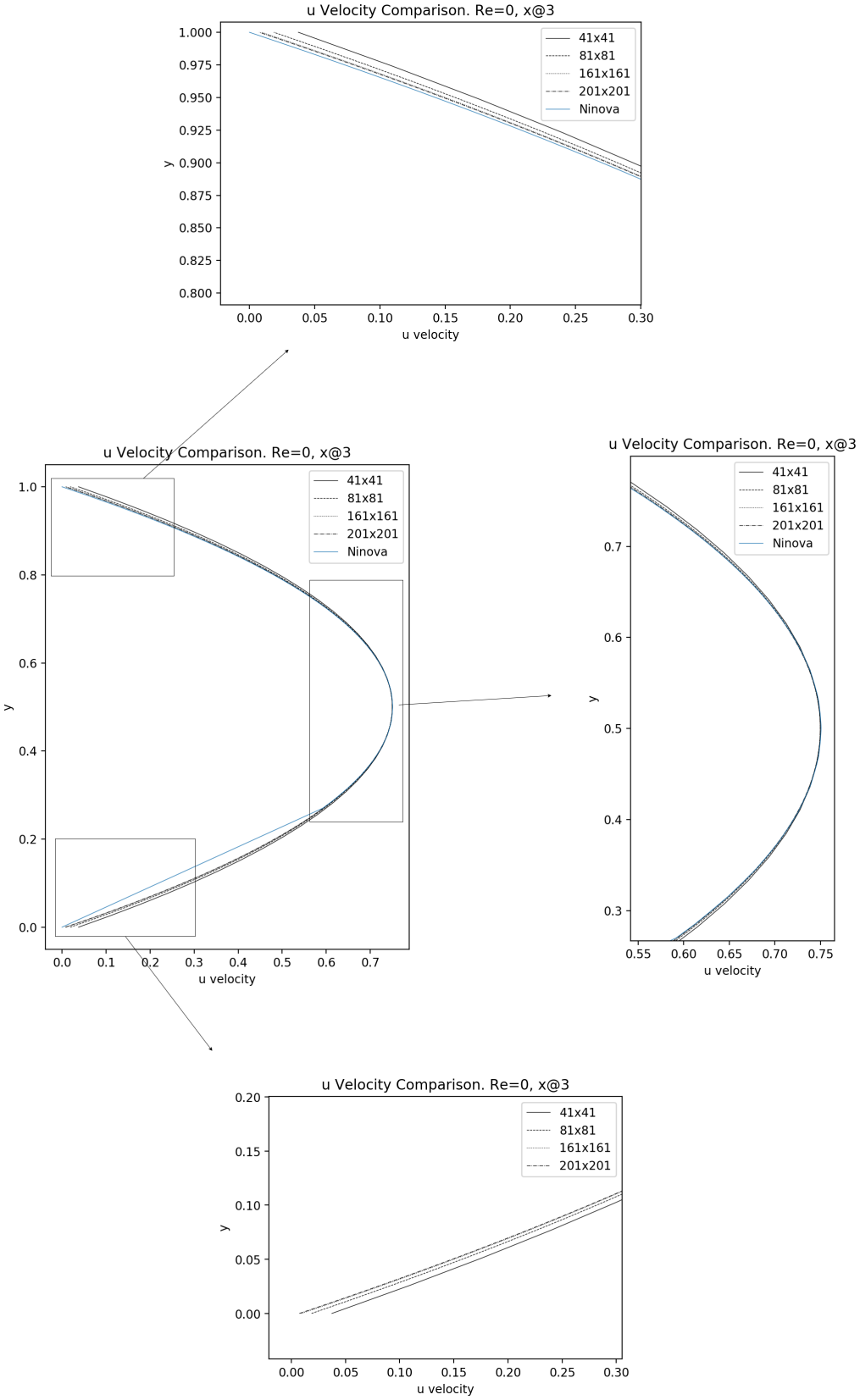

Figure 2.8: u velocity component comparison for Stokes flow x=3.

Results of Navier-Stokes Flow (Re=100)

Different grid sizes as defined for Stokes solution are also defined in Navier-Stokes problem to examine the grid size effect on the results. In addition to grid size, $\Delta Reynolds$ effect on the final solution is also studied. Therefore; u-velocity components @x=3.0 for four different grids of 41x41, 81x81, 161x161 and 201x201; and three different $\Delta Reynolds$ of 10, 25 and 50 are compared with the result that is shared on Ninova. Following results are obtained PETSc external direct solver library MUMPS with LU factorization.

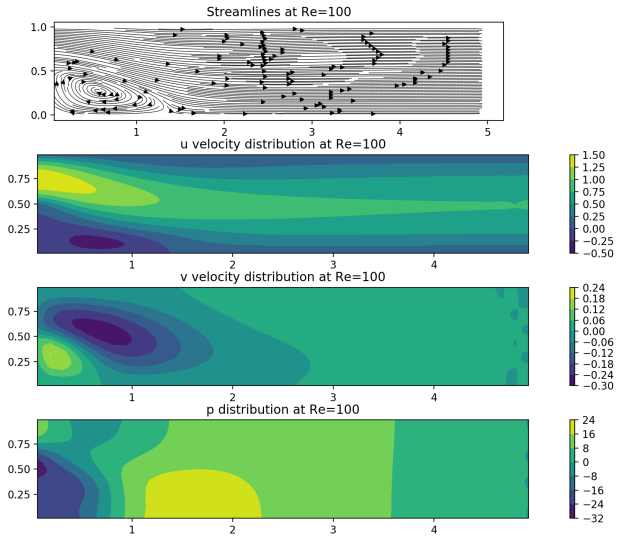

Figure 2.9: Navier-Stokes results(Re=100) of MUMPS with LU Preconditioner for 41x41 grid.

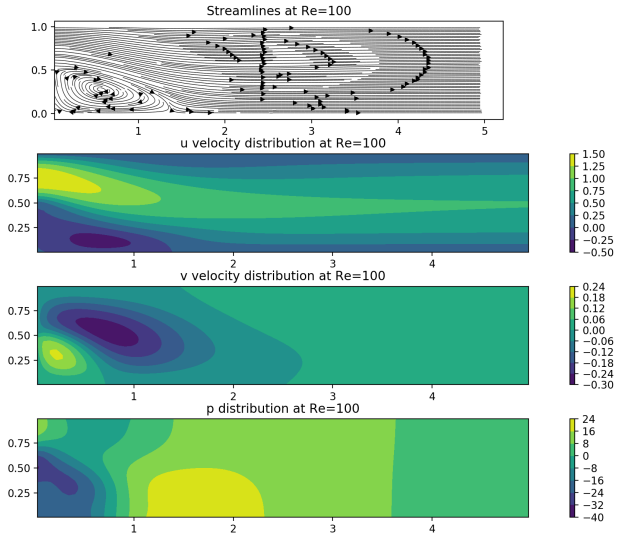

Figure 2.10: Navier-Stokes results(Re=100) of MUMPS with LU Preconditioner for 81x81 grid.

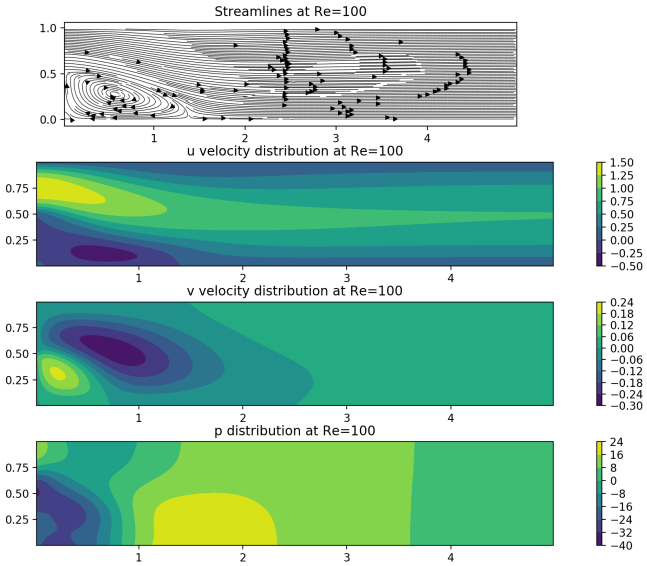

Figure 2.11: Navier-Stokes results(Re=100) of MUMPS with LU Preconditioner for 161x161 grid.

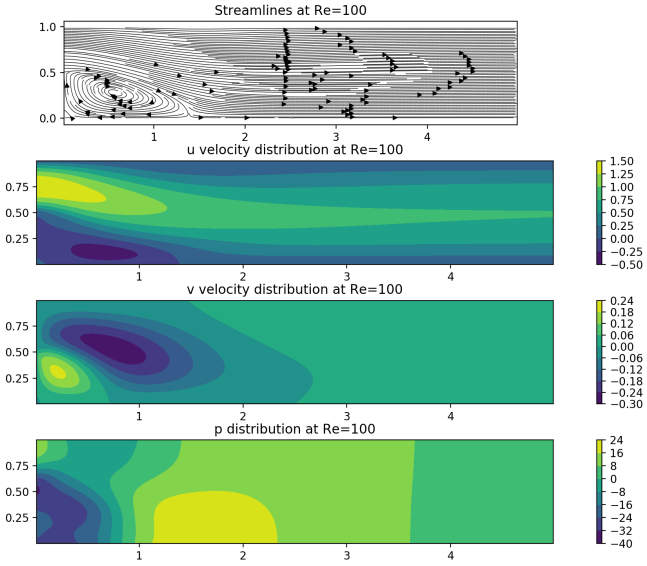

Figure 2.12: Navier-Stokes results(Re=100) of MUMPS with LU Preconditioner for 201x201 grid.

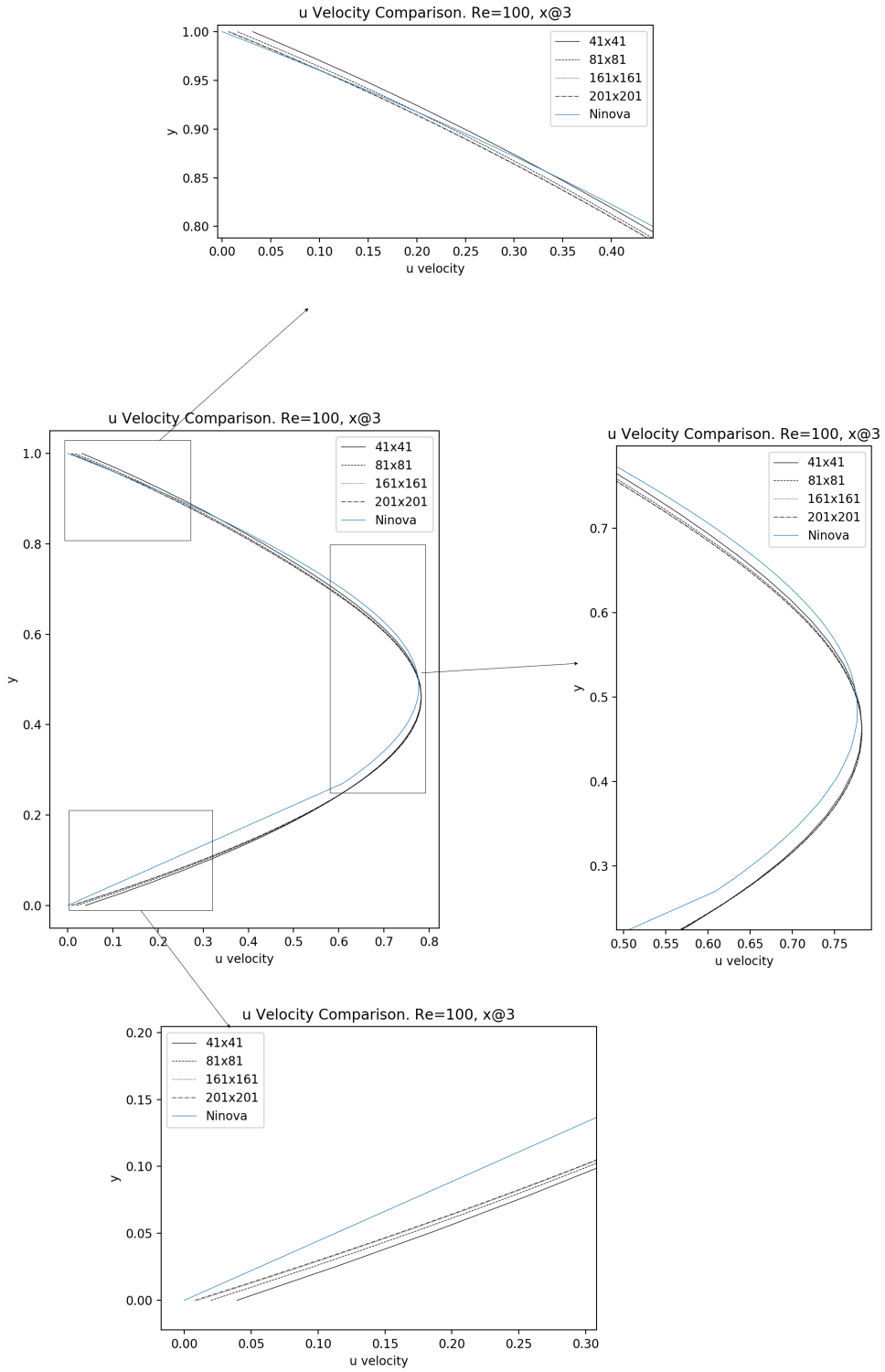

Figure 2.13: u velocity component comparison for N-S flow Re=100, x=3.

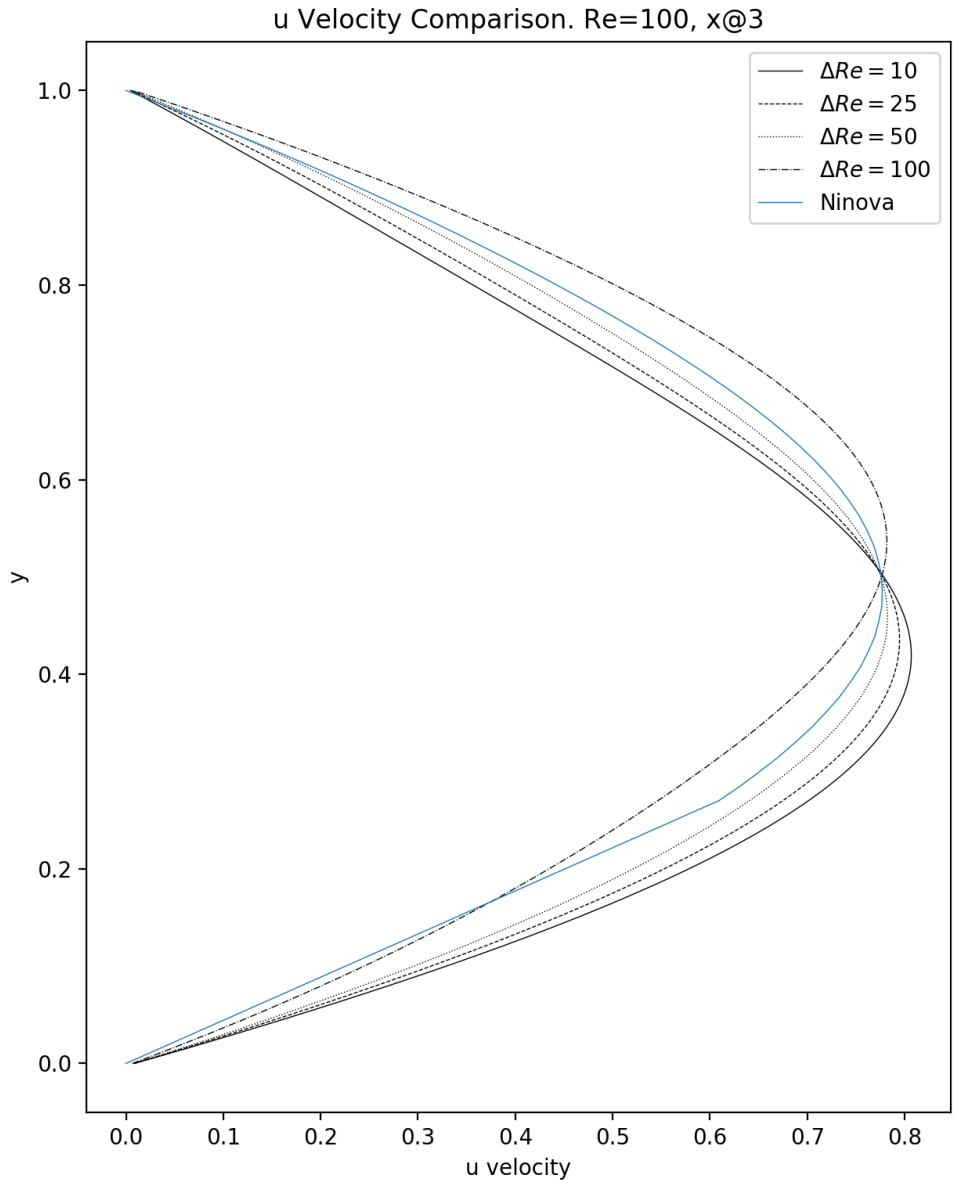

Figure 2.14: Effect of ΔRe compared with the reference solution.

3. Conclusion

Backward step geometry with given boundary conditions is analyzed with native PETSc and its external parallel direct sparse solver MUMPS. Results are presented using streamline plots and contour plots of the primitive variables for different grid sizes. It can be easily seen that 41x41 and 81x81 grids are not sufficient enough to represent the flow field accurately. Vortices generated in the bottom left are of the domain can not be observed aforementioned grid sizes.

Moreover, effect of the different increment of Reynolds number for Re=100 is presented. $\Delta Re = 50$ gives best results for amongst 10, 25 and 100 as shown in Figure 2.14. There is a observable difference between the values that are shared in a .dat file in NINOVA. This difference can be eliminated by using more dense grid which is not in the scope of this report.

Herebelow, directory tree of the project can be reviewed. Using Cmake with template files that can be found under cmake-modules folder makes it easier to compile the whole code. The installation process of PETSC and details on how to use the Cmake can be found in Appendix.

```
buy-cfd
├── build
│   ├── CmakeFiles
│   ├── MakeFile ..... File created by Cmake to compile program.
│   └── mac ..... .exe file
├── cmake-modules
│   └── FindPETSc.cmake ..... Link
├── results
│   ├── legacy ..... Results of the iterative/direct solvers of native PETSc
│   ├── correct ..... Results of the MUMPS solver
│   ├── X=3-PROFILE-RE=100.dat ..... Results of NS problem shared on NINOVA
│   └── X=3-PROFILE-STOKES.dat .... Results of Stokes problem shared on NINOVA
├── hw1.cpp .. ..... c++ code
└── plot-util.py .. ..... Python matplotlib library util for plotting
```

Even if c++ and PETSc have steep learning curve, they are very useful and handy since the community of the language and library is very active. One can easily find answers to the problem that may occur both in developing the code and runtime since PETSc is well-documented library and most of the problems have already been answered in its FAQ section of website.

In the meantime, it is worth mentioning that the problem appears in both direct/iterative solvers of PETSc while trying to solve coefficient matrix of this report. This problem couldn't be solved after trying multiple database options of PETSc library. Therefore; I highly recommend new users of PETSc to utilize MUMPS instead of PETSc to eliminate these errors and use their time efficiently.

Appendix - A

PETSc Installation

Following instructions are valid for unix environments (MacOS and Linux):

```
cd <Enter>
git clone -b release https://gitlab.com/petsc/petsc.git petsc
cd petsc
```

Listing 1: Cloning PETSc source code from git

Before compiling the source code, user should define a configuration. –with-X argument is necessary to visualize matrix pattern as shown in Figure 2.1. For MacOS, XQuarts should also be installed and run before running the written program.

```
# Compile Native PETSc
./configure --with-cc=gcc --with-cxx=g++ --with-fc=0 --with-fortran-bindings=0
--download-mpich --download-f2cblaslapack --with-X=1
# Compile with MUMPS
./configure --with-cc=gcc --with-cxx=g++ --with-fc=gfortran --download-mpich
--download-f2cblaslapack --download-mumps --download-scalapack
--download-parmetis --download-metis --download-ptscotch
--download-blacs=yes --with-X=1
```

Listing 2: Configuring PETSc before compilation

If gfortran compiler does not work, use following line to install gfortran explicitly:

```
brew install --cask gfortran
```

Listing 3: Installing gfortran in case any error occurs

```
make PETSC_DIR=/Users/bugur/petsc PETSC_ARCH=arch-darwin-c-debug all
```

Listing 4: Compilation of the PETSc with pre-defined configuration

After compilation is finished, environment variables should be defined in bashrc as follows:

```
export PETSC_DIR="/Users/bugur/petsc"
export PETSC_ARCH="arch-darwin-c-debug"
```

Listing 5: Defining environment variables

In order to write customized code, users can follow the steps in the link. It is the most basic way to compile PETSc code. Just copy paste the Makefile.user in the related folder of PETSc installation and write make in command line. However, in order this alternative to work, you should have pkg-config in your computer.

```
brew install pkg-config
```

Listing 6: Installing pkg-config

Compilation of the Customized Code

Installation process should be finished successfully after each step explained above is done. Cmake is utilized in this project. Example Cmake file for compilation process of your customized code.

```
cmake_minimum_required(VERSION 3.8)
project(MAC)

# This is the main folder of the project:
set (PARENT_DIR /Users/bugur/Desktop/ACFD/buy-cfd/)

# FindPETSc.cmake is located in this folder:
message("/../cmake-modules")
list (APPEND CMAKE_MODULE_PATH "${PARENT_DIR}/cmake-modules")

# Additional compile options
set(CMAKE_CXX_FLAGS "-std=c++17")
find_package(Matplot++ REQUIRED)
find_package (PETSc REQUIRED)

message (STATUS "PETSC COMPILER ${PETSC_COMPILER}")
message (STATUS "CPP COMPILER ${PETSC_COMPILER}")

# Include directories
include_directories (${PETSC_INCLUDES})
add_definitions (${PETSC_DEFINITIONS})
add_definitions (-g)
message (STATUS "PETSC_DEFINITIONS ${PETSC_DEFINITIONS}")

# hw1.cpp is the only code that the project is written
set(SOURCE_FILES hw1.cpp)
add_executable(mac hw1.cpp)
target_link_libraries(mac PUBLIC Matplot++::matplot)
target_link_libraries(mac PUBLIC PETSc)
```

Listing 7: Cmake file used in project

Create this file as CmakeLists.txt in the root folder of the project. So as to create a build folder under root folder and run cmake, use following command block. "MakeFile" which enables users to compile the code will be created.

```
mkdir build
cd build/
cmake ..
```

Listing 8: Running

Example Usage of the Code Written for the Project

Basic usage of the code in order to display matrix pattern with 5x5 grid size:

```
./mac -Imax 5 -Jmax 5 -mat_view draw -draw_pause -1
```

Final Reynolds number, delta Reynolds number, grid size, solver preconditioner type can be defined explicitly:

```
# Running GMRES iterative solver with Incomplete LU Factorization
./mac ksp_type preonly -pc_type lu -Imax 201 -Jmax 201

# Running GMRES iterative solver with Incomplete LU Factorization
./mac -ksp_type gmres -pc_type ilu -Imax 201 -Jmax 201

# In order to see matrix convergence add -ksp_monitor
./mac -ksp_type gmres -pc_type ilu -Imax 201 -Jmax 201 -ksp_monitor

# Direct Solver with LU Factorization:
#      -ksp_type preonly -pc_type lu :
# Direct Solver with cholesky Factorization:
#     -ksp_type preonly -pc_type cholesky
# Direct Solver with Incomplete LU Factorization
#     -ksp_type preonly -pc_type ilu
# Direct Solver with Incomplete Cholesky Factorization
#     -ksp_type preonly -pc_type icc
```

Listing 9: Running native direct/iterative solvers of PETSc

When MUMPS parallel solver is used, users can define number of processors to use. Herebelow, -latestRe is the final Reynolds number, -deltaRe is the increment of the Reynolds number and -log_summary is PETSc function to show summary of the computation.

```
mpiexec -np 2 ./mac -Imax 201 -Jmax 201 -latestRe 100 -deltaRe 100
-pc_factor_mat_solver_type mumps -ksp_type preonly -pc_type lu -log_summary
```

Log output of the code is as follows:

```
##########################################################
#                                                        #
#                       WARNING!!!                       #
#                                                        #
#   This code was compiled with a debugging option.      #
#   To get timing results run ./configure                #
#   using --with-debugging=no, the performance will      #
#   be generally two or three times faster.              #
#                                                        #
##########################################################
```

```
./mac on a arch-darwin-c-debug named bugur-MacBook-Pro.local with 1 processor, by Unknown Sun Dec 27 19 49 16 2020
Using Petsc Release Version 3.14.0, unknown

                         Max       Max/Min     Avg       Total
Time (sec):           2.552e-01     1.000   2.552e-01
Objects:              1.100e+01     1.000   1.100e+01
Flop:                 8.315e+03     1.000   8.315e+03  8.315e+03
Flop/sec:             3.259e+04     1.000   3.259e+04  3.259e+04
Memory:               2.255e+05     1.000   2.255e+05  2.255e+05
MPI Messages:         0.000e+00     0.000   0.000e+00  0.000e+00
MPI Message Lengths:  0.000e+00     0.000   0.000e+00  0.000e+00
MPI Reductions:       0.000e+00     0.000

Flop counting convention: 1 flop = 1 real number operation of type (multiply/divide/add/subtract)
                            e.g., VecAXPY() for real vectors of length N --> 2N flop
                            and VecAXPY() for complex vectors of length N --> 8N flop

Summary of Stages:   ----- Time ------  ----- Flop ------  --- Messages ---  -- Message Lengths --
                        Avg     %Total     Avg     %Total    Count   %Total     Avg         %Total
 0:      Main Stage: 2.5487e-01  99.9%  8.3150e+03 100.0%  0.000e+00   0.0%  0.000e+00        0.0%  0.000e+00   0.0

------------------------------------------------------------------------------------------------------------------------
See the 'Profiling' chapter of the users' manual for details on interpreting output.
Phase summary info:
   Count: number of times phase was executed
   Time and Flop: Max - maximum over all processors
                  Ratio - ratio of maximum to minimum over all processors
   Mess: number of messages sent
   AvgLen: average message length (bytes)
   Reduct: number of global reductions
   Global: entire computation
   Stage: stages of a computation. Set stages with PetscLogStagePush() and PetscLogStagePop().
      %T - percent time in this phase         %F - percent flop in this phase
      %M - percent messages in this phase     %L - percent message lengths in this phase
      %R - percent reductions in this phase
   Total Mflop/s: 10e-6 * (sum of flop over all processors)/(max time over all processors)
------------------------------------------------------------------------------------------------------------------------


      ##########################################################
      #                                                        #
      #                       WARNING!!!                       #
      #                                                        #
      #   This code was compiled with a debugging option.      #
      #   To get timing results run ./configure                #
      #   using --with-debugging=no, the performance will      #
      #   be generally two or three times faster.              #
      #                                                        #
      ##########################################################


Event                Count      Time (sec)     Flop                              --- Global ---  --
                   Max Ratio  Max     Ratio   Max  Ratio  Mess   AvgLen  Reduct  %T %F %M %L %R  %T
------------------------------------------------------------------------------------------------------------------------

--- Event Stage 0: Main Stage
```

```
VecView                1 1.0 9.4318e-04 1.0 0.00e+00 0.0 0.0e+00 0.0e+00 0.0e+00  0  0  0  0  0   0  0  0  0  0      0
VecCopy                1 1.0 3.0041e-05 1.0 0.00e+00 0.0 0.0e+00 0.0e+00 0.0e+00  0  0  0  0  0   0  0  0  0  0      0
VecSet                 3 1.0 6.2943e-05 1.0 0.00e+00 0.0 0.0e+00 0.0e+00 0.0e+00  0  0  0  0  0   0  0  0  0  0      0
VecAssemblyBegin       2 1.0 7.8678e-06 1.0 0.00e+00 0.0 0.0e+00 0.0e+00 0.0e+00  0  0  0  0  0   0  0  0  0  0      0
VecAssemblyEnd         2 1.0 1.0014e-05 1.0 0.00e+00 0.0 0.0e+00 0.0e+00 0.0e+00  0  0  0  0  0   0  0  0  0  0      0
MatSolve               1 1.0 5.1742e-03 1.0 7.70e+03 1.0 0.0e+00 0.0e+00 0.0e+00  2 93  0  0  0   2 93  0  0  0      1
MatLUFactorSym         1 1.0 1.3020e-03 1.0 0.00e+00 0.0 0.0e+00 0.0e+00 0.0e+00  1  0  0  0  0   1  0  0  0  0      0
MatLUFactorNum         1 1.0 8.6670e-03 1.0 6.17e+02 1.0 0.0e+00 0.0e+00 0.0e+00  3  7  0  0  0   3  7  0  0  0      0
MatAssemblyBegin       1 1.0 5.0068e-06 1.0 0.00e+00 0.0 0.0e+00 0.0e+00 0.0e+00  0  0  0  0  0   0  0  0  0  0      0
MatAssemblyEnd         1 1.0 7.3910e-05 1.0 0.00e+00 0.0 0.0e+00 0.0e+00 0.0e+00  0  0  0  0  0   0  0  0  0  0      0
MatGetRowIJ            1 1.0 2.0027e-05 1.0 0.00e+00 0.0 0.0e+00 0.0e+00 0.0e+00  0  0  0  0  0   0  0  0  0  0      0
MatGetOrdering         1 1.0 5.1904e-04 1.0 0.00e+00 0.0 0.0e+00 0.0e+00 0.0e+00  0  0  0  0  0   0  0  0  0  0      0
MatView                2 1.0 1.7607e-01 1.0 0.00e+00 0.0 0.0e+00 0.0e+00 0.0e+00 69  0  0  0  0  69  0  0  0  0      0
KSPSetUp               1 1.0 1.2875e-05 1.0 0.00e+00 0.0 0.0e+00 0.0e+00 0.0e+00  0  0  0  0  0   0  0  0  0  0      0
KSPSolve               1 1.0 1.7519e-02 1.0 8.32e+03 1.0 0.0e+00 0.0e+00 0.0e+00  7100  0  0  0  7100  0  0  0  0      0
PCSetUp                1 1.0 1.1899e-02 1.0 6.17e+02 1.0 0.0e+00 0.0e+00 0.0e+00  5  7  0  0  0   5  7  0  0  0      0
PCApply                1 1.0 5.2180e-03 1.0 7.70e+03 1.0 0.0e+00 0.0e+00 0.0e+00  2 93  0  0  0   2 93  0  0  0      1
------------------------------------------------------------------------------------------------------------------------
```

Listing 10: Example output of the program