Final Project Report
Multigrid Solver for Helmholtz/Poission Equations

Yu-hsuan Shih yhs264

1 Introduction

In this project, I developed two geometric multigrid solvers, one for Helmholtz

$$\boldsymbol{u} - c\Delta\boldsymbol{u} = \boldsymbol{f}$$

and the other one for Poission Equation

$$-\Delta u = f$$

on a 2D periodic domain using the PETSc library. The solvers are used to construct an approximate projection method preconditioner for solving the incompressible time-dependent Stokes equation on a staggered-grid

$$\begin{aligned}&\boldsymbol{u}_t(\boldsymbol{x},t) + \boldsymbol{\nabla} p(\boldsymbol{x},t) = \mu\boldsymbol{\nabla}^2\boldsymbol{u}(\boldsymbol{x},t) + \boldsymbol{f}(\boldsymbol{x},t),\\ &\boldsymbol{\nabla}\cdot\boldsymbol{u}(\boldsymbol{x},t) = 0,\end{aligned}$$

where $\boldsymbol{u}(\boldsymbol{x},t)$ is the velocity, $p(\boldsymbol{x},t)$ is the pressure, $f(\boldsymbol{x},t)$ is a force density, and μ is the coefficient of viscosity.

2 Numerical Scheme

Applying Crank-Nickolson to the diffusion term and midpoint rule to the pressure and source terms to solve the spatially-discretized equations with time step size Δt, gives the following discrete system for $\boldsymbol{u}^{n+1}$ and $p^{n+\frac{1}{2}}$,

$$\frac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^n}{\Delta t} + \boldsymbol{\nabla} p^{n+\frac{1}{2}} = \mu\boldsymbol{\nabla}^2\left(\frac{\boldsymbol{u}^{n+1} + \boldsymbol{u}^n}{2}\right) + \boldsymbol{f}^{n+\frac{1}{2}} \tag{1}$$

$$\boldsymbol{\nabla}\cdot\boldsymbol{u}^{n+1} = 0. \tag{2}$$

Denote $\boldsymbol{D} = (D^x, D^y)$, $\boldsymbol{G} = (G^x, G^y)$, $\boldsymbol{L} = (L^x, L^y)$ as discrete divergence, gradient and Laplace operators. The system can be written as

$$\begin{bmatrix} I - \frac{\mu\Delta t}{2}L^x & 0 & \Delta t G^x \\ 0 & I - \frac{\mu\Delta t}{2}L^y & \Delta t G^y \\ -\Delta t D^x & -\Delta t D^y & 0 \end{bmatrix} \begin{bmatrix} u^{n+1} \\ v^{n+1} \\ p^{n+\frac{1}{2}} \end{bmatrix} = \begin{bmatrix} \left(I + \frac{\mu\Delta t}{2}L^x\right)u^n + \Delta t f_u^{n+\frac{1}{2}} \\ \left(I + \frac{\mu\Delta t}{2}L^y\right)v^n + \Delta t f_v^{n+\frac{1}{2}} \\ 0 \end{bmatrix}. \tag{3}$$

This is a method of lines discretization which is second order in time, If all the spatial discrete operators $D^x, D^y, G^x, G^y, L^x, L^y$ are second order in space and a time step $\Delta t = \mathcal{O}(h)$, where $h = h_x = h_y$ is the grid spacing in x and y direction, is used, the numerical solution will converge in second-order.

2.1 Projection Method as a Preconditioner

A pressure-free projection method is used as a preconditioner to solve linear system (3) in this project. It is an approximate solver for (3) and in fact, if we are in a periodic domain and all the steps involved are calculated exactly, this preconditioner will be an exact solve to the linear system and so the iterative solver will converge in one iteration.

There are three main steps in the pressure-free projection method:

1. Solve an analog of (1) for an intermediate quantity $\boldsymbol{u}^*$,

$$\frac{\boldsymbol{u^*} - \boldsymbol{u}^n}{\Delta t} = \mu \boldsymbol{\nabla}^2 \left(\frac{\boldsymbol{u^*} + \boldsymbol{u}^n}{2} \right) + \boldsymbol{f}^{n+\frac{1}{2}}. \tag{4}$$

   Note that to obtain $\boldsymbol{u}^*$, there are two linear systems to solve:

$$\left(I - \frac{\mu \Delta t}{2} L^x \right) u^* = \Delta t f_u$$

   and

$$\left(I - \frac{\mu \Delta t}{2} L^y \right) v^* = \Delta t f_v.$$

2. Project this intermediate quantity onto the space of divergence-free fields to get $\boldsymbol{u}^{n+1}$. The projection is done by solving

$$\begin{aligned} \boldsymbol{u}^* &= \boldsymbol{u}^{n+1} + \Delta t \boldsymbol{G} \phi \\ D\boldsymbol{u}^{n+1} &= 0, \end{aligned}$$

   which is equivalent to solving a poission equation

$$-\boldsymbol{D}\boldsymbol{G}\phi = -L^c = -\frac{1}{\Delta t} \boldsymbol{D}\boldsymbol{u}^*$$

   and updating $\boldsymbol{u}^{n+1}$ by removing the non divergence-free part of $\boldsymbol{u}^*$,

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^* - \Delta t \boldsymbol{G} \phi$$

3. Correct the pressure by plugging $\boldsymbol{u}^*$ back to (4) and compare it to the original system (1),

$$\begin{aligned} \boldsymbol{G} p^{n+\frac{1}{2}} &= \left(I - \frac{\mu \Delta t}{2} \boldsymbol{L} \right) \boldsymbol{G} \phi \\ & \boldsymbol{G}\phi - \frac{\mu \Delta t}{2} \boldsymbol{L}\boldsymbol{G}\phi. \end{aligned}$$

   If $\boldsymbol{L}\boldsymbol{G} = \boldsymbol{G}L^c$ which is the case when having periodic domain, then the update formula can be reduced to

$$p^{n+\frac{1}{2}} = \left(I - \frac{\mu \Delta t}{2} L^c \right) \phi.$$

To summarize, in this projection method, there are two helmholtz equations to solve for intermediate value $\boldsymbol{u}^*$ and one poission equation to solve for the correction of divergence-free $\boldsymbol{u}^{n+1}$. These linear systems can be efficiently solved by using geometric multigrid techniques (See section 3).

2.2 Staggered-grid Discretization

On a staggered-grid, scalar variable pressure, p, is defined at cell centers and components of vector variables, velocities $\boldsymbol{u} = (u, v)$, source term $\boldsymbol{f} = (f_u, f_v)$, are defined at the corresponding faces of the grid.

The divergence of $\boldsymbol{u} = (u, v)$ is approximated at cell centers by $\boldsymbol{Du} = (D^x u + D^y v)$ with

$$(D^x u)_{i,j} = \frac{u_{i+\frac{1}{2},j} - u_{i-\frac{1}{2},j}}{h}$$

and

$$(D^y v)_{i,j} = \frac{u_{i,j+\frac{1}{2}} - u_{i,j-\frac{1}{2}}}{h};$$

the gradient of p is approximated at the x and y edges of the grid cells by $\boldsymbol{G}p = (G^x p, G^y p)$ with

$$(G^x u)_{i-\frac{1}{2},j} = \frac{p_{i,j} - p_{i-1,j}}{h}$$

and

$$(G^y u)_{i,j-\frac{1}{2}} = \frac{p_{i,j} - p_{i,j-1}}{h};$$

the Laplacian of $\boldsymbol{u} = (L^x u, L^y v)$ and Laplacisn of $\phi = L^c p$ is discretized using the standard five-point discrete Laplacian,

$$(L^x u)_{i-\frac{1}{2},j} = \frac{u_{i+\frac{1}{2},j} + u_{i-\frac{3}{2},j} + u_{i-\frac{1}{2},j+1} + u_{i-\frac{1}{2},j-1} - 4u_{i-\frac{1}{2},j}}{h^2},$$

$$(L^y u)_{i,j-\frac{1}{2}} = \frac{u_{i,j+\frac{1}{2}} + u_{i,j-\frac{3}{2}} + u_{i+1,j-\frac{1}{2}} + u_{i-1,j-\frac{1}{2}} - 4u_{i,j-\frac{1}{2}}}{h^2},$$

and

$$(L^c \phi)_{i,j} = \frac{u_{i,j+1} + u_{i,j-1} + u_{i+1,j} + u_{i-1,j} - 4u_{i,j}}{h^2},$$

. Note that all of these discrete operators are $\mathcal{O}(h^2)$.

3 Multigrid Solver

The two Helmholtz (face-centered) and a poission (cell-centered) subsolvers in the projection method can be solved by using standard geometric multigrid techniques.

In this project, the coarsening factor I used for multigrid is 2, and the grid is coarsened until there are only 2 cells in each direction on the domain.

There are 3 major steps in Multigrid: (i) smoothing, (ii) restriction, and (iii) prolongation. The two types of degree of freedom, cell-centered and face-centered, have different restriction and prolongation operators:

Cell-Centered For cell-centered degree of freedom, the restriction is done by averaging 2^2 fine cells to the corresponding coarse cell and the prolongation is done by simply injecting the coarse cell value to the overlaying 2^2 fine cells.

Faced-Centered Referenced from [1], I used a 6-point stencil for the restriction operator on a face-centered degree of freedom. For prolongation, either a 2-point stencil or a 4-point stencil (Right) is used. See Figure 1 for illustration.

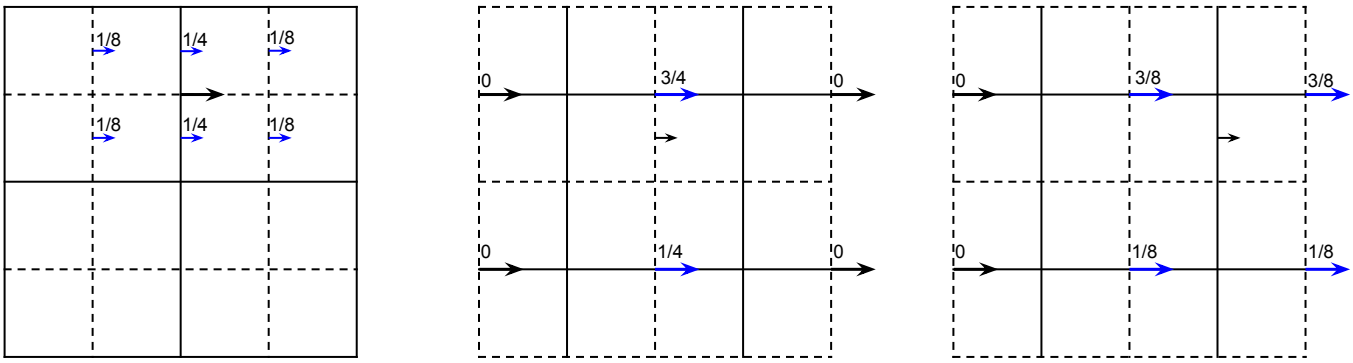

Figure 1: The 6-point stencil that used for the restriction operation (Left), the 2-point stencil (Middle) and 4-point stencil for prolongation operation for face-centered degree of freedome.

4 Computational Complexity

Assuming N by N grid points are used to discretize the domain. Since the linear system in (3) is sparse with $\mathcal{O}(N^2)$ nonzeros, the matrix vector multiplication requires $\mathcal{O}(N^2)$ floating point operations (flops). The dominate cost for the projection method preconditioner is solving the three linear systems (other operations are all of $\mathcal{O}(N^2)$). By using multigrid, the amount of flops involved is either $\mathcal{O}(N^2 \log(N))$ or $\mathcal{O}(N^2)$. The latter is the case when fixed amount of V-cycles are used or fixed relative tolerance is specified. Therefore, the computational complexity of applying the preconditioner is $\mathcal{O}(N^2 \log(N))$ or $\mathcal{O}(N^2)$. Let number of iterations be k, the cost of GMRES will grows as $\mathcal{O}(k^2)$ and so the computational complexity is $\mathcal{O}(k^2 + N^2)$. If using a restart GMRES, the complexity then becomes $\mathcal{O}(N^2)$. In conclution, one preconditioned iteration requires $\mathcal{O}(N^2)$ or $\mathcal{O}(N^2 \log(N))$ flops.

5 Code Implementation

In PETSc, to solve a linear system, one needs to construct a **KSP** object which contains information of (i) matrix A defining the linear system, (ii) the iterative/direct solver to solve the system, and (iii) the preconditioner (**PC**) to use if using an iterative method. There are many built-in solvers and preconditioners that users can use, such as GMRES, FGMRES, ..., etc, but one can also define their own.

The whole code structure that I built for this project is demonstrated in Figure 2. Three KSP objects along with three PCs are declared. The first KSP object defines the linear system (3) which uses the projection method preconditioner. The preconditioner is implemented by constructing a PC with type **PCShell**, which along with it, there's a function **ShellPCApply** defined. Inside the ShellPCApply function, I implemented the algorithm of projection method. The three linear systems that need to be solved when applying the preconditioner are implemented by further defining two KSP objects, one for Helmholtz equation, one for poission equation. Since PETSc already have an implementation of multigrid methods, i.e. the grid hierarchy, communication between levels of grids, the KSP objects for the smoothers on each level are defined. To customize the multrigrid solver, I specified the restriction and prolongation operators using **PCMGSetRestriction**, **PCMGSetInterpolation**.

	D_x		D_y		G_x		G_y	
N	Rel Err E_N	$p \approx \log_2 \frac{E_{2N}}{E_N}$	E_N	p	E_N	p	E_N	p
8	2.550e-02	n/a	2.550e-02	n/a	9.968e-02	n/a	9.968e-02	n/a
16	6.413e-03	1.9914	6.413e-03	1.9914	2.550e-02	1.9668	2.550e-02	1.9668
32	1.606e-03	1.9975	1.606e-03	1.9975	6.413e-03	1.9914	6.413e-03	1.9914
64	4.015e-04	2.0000	4.015e-04	2.0000	1.606e-03	1.9975	1.606e-03	1.9975

Table 1: The relative error and estimated order of convergence of discrete spatial operators, G^x, G^y, D^x, D^y

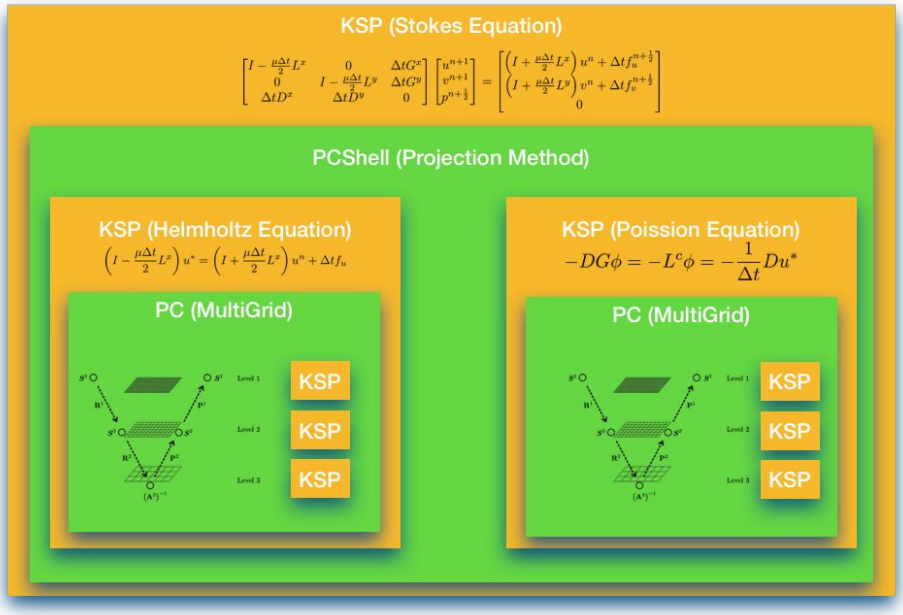

Figure 2: This figure shows how the linear system and the preconditioner are implemented using KSP and PC objects in the PETSc library.

6 Code Validation

6.1 Discrete Spatial Operators

The spatial discrete operators that defines the linear system (3), $I - \frac{\mu\Delta t}{2}L^x$, $I - \frac{\mu\Delta t}{2}L^y$, G^x, G^y, D^x, D^y are verified using

$$
\begin{aligned}
u(x,t) &= e^{-\mu t}\cos(2\pi x)\sin(2\pi y) \\
v(x,t) &= -e^{-\mu t}\sin(2\pi x)\cos(2\pi y) \\
p(x,t) &= \frac{1}{4}e^{-2\mu t}\left(\cos(4\pi x) + \cos(4\pi y)\right)
\end{aligned}
$$

and compare the application of the discrete operators of them with the continuous version of the operators. The results is reported in Table 1 and Table 2. We can indeed see that the numerical solutions are consistent with the analytical solution and are all converging in 2nd order.

	$I - \frac{\mu\Delta t}{2} L_x$		$I - \frac{\mu\Delta t}{2} L_x$	
N	Rel Err E_N	$p \approx \log_2 \frac{E_{2N}}{E_N}$	E_N	p
8	4.911e-02	n/a	4.911e-02	n/a
16	1.247e-02	1.9776	1.247e-02	1.9776
32	3.129e-03	1.9947	3.129e-03	1.9947
64	7.831e-04	1.9984	7.831e-04	1.9984

Table 2: The relative error and estimated order of convergence of discrete spatial operators, $I - \frac{\mu\Delta t}{2} L^x$, $I - \frac{\mu\Delta t}{2} L^y$

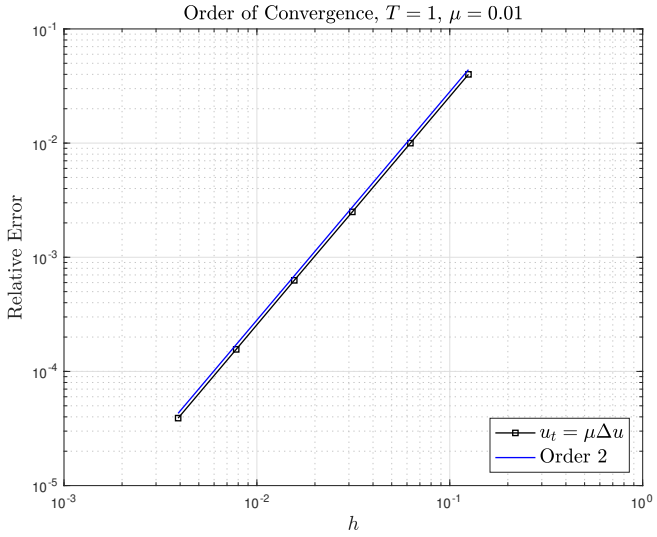

Figure 3: Order of convergence for solving diffusion equation $u_t = \mu\Delta u$.

6.2 Temporal Discretization (Without Incompressiblity Condition)

6.2.1 Pure diffusion

To verify the Crank-Nicholson implementation, I ignore the pressure and source terms and solve the pure diffusion equation $u_t = \mu\Delta u$, where we have

$$\left(I - \frac{\mu\Delta t}{2} L^x\right) u^{n+1} = \left(I + \frac{\mu\Delta t}{2} L^x\right) u^n$$

to solve for each time step.

Since Crank-Nicholson is unconditionally stable, we don't have CFL restriction and I set it to 1. From Figure 3, we can see that the solution is converging and reaches relative error around 10^{-4} at number of grid points $N_x = N_y = 256$ and it's converging in 2nd order. Note that once the number of grid points get larger (the smaller the discretization error), the requirement of the accuracy from the linear system also gets higher. Therefore, I have to set the relative tolerance smaller in order to get second order of convergence.

6.2.2 First Order Oridinary Differential Equation

When coefficient of viscousity μ equals to 0 and the pressure term is ignored, the equation reduced to a first order ode $u_t = f(\boldsymbol{x}, t)$, where we have $u^{n+1} = u^n + \Delta t f^{n+\frac{1}{2}}$ to solve for each time step. The right

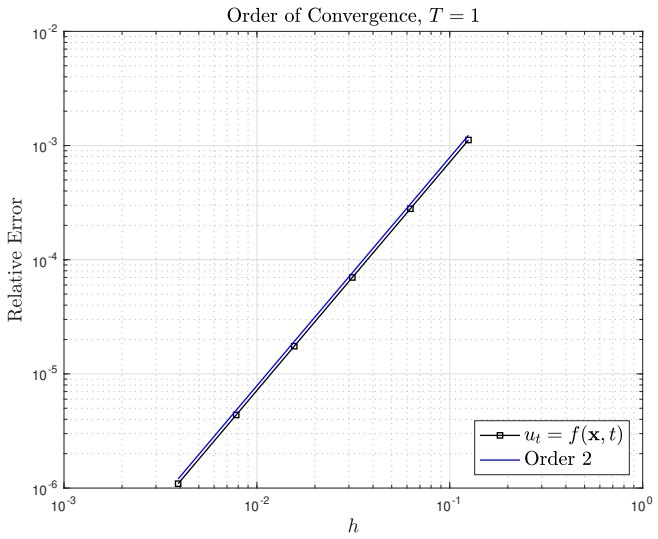

Figure 4: Order of convergence for solving $u_t = f(\boldsymbol{x}, t)$.

hand side I used here is

$$f(x, y, t) = -e^{-t} \sin(2\pi x) \sin(2\pi y).$$

From Figure 4, we can see that the numerical solution is converging and has 2nd order of convergence.

6.3 Construction of the Linear Sytem (3)

The matrix defines the linear system (3) is constructed by using submatrixes, which are the discrete spatial operators that I've verified the correctness in Section 6.1. In this section, I apply the matrix to true solution $\boldsymbol{u}(\boldsymbol{x}, t + \Delta t) = (u(\boldsymbol{x}, t + \Delta t), v(\boldsymbol{x}, t + \Delta t))$, $p(x, t + \frac{\Delta t}{2})$ and compare with the right hand side vector evalutated by $\boldsymbol{u}(\boldsymbol{x}, t)$ and $\boldsymbol{f}(\boldsymbol{x}, t)$, i.e.,

$$\| \begin{bmatrix} I - \frac{\mu \Delta t}{2} L^x & 0 & \Delta t G^x \\ 0 & I - \frac{\mu \Delta t}{2} L^y & \Delta t G^y \\ -\Delta t D^x & -\Delta t D^y & 0 \end{bmatrix} \begin{bmatrix} u(\boldsymbol{x}, t + \Delta t) \\ v(\boldsymbol{x}, t + \Delta t) \\ p(\boldsymbol{x}, t + \frac{\Delta t}{2}) \end{bmatrix} - \begin{bmatrix} \left(I + \frac{\mu \Delta t}{2} L^x\right) u(\boldsymbol{x}, t) + \Delta t f_u^{n+\frac{1}{2}} \\ \left(I + \frac{\mu \Delta t}{2} L^y\right) u(\boldsymbol{x}, t) + \Delta t f_v^{n+\frac{1}{2}} \\ 0 \end{bmatrix} \|_2.$$

From Figure 5, we can see the norm of the residual is converging in 2nd order. This verifies the local truncation error of the numerical scheme, $\mathcal{O}(h^2) + \mathcal{O}(\Delta t^2)$, and also tells the correct implementation of submatrixes' combination.

6.4 Projection Method Preconditioner

As mentioned is Section 2.1, if the domain is periodic, applying an accurately calculated preconditioner results in a direct solve to the problem and it's in fact a 2nd order numerical scheme for the equations. To verify the implementation of the preconditioner, I used the same set of manufactured solutions in Section 6.1 and Figure 6 indeed shows that the scheme is 2nd order both in pressure and velocity. Note that in PETSc, to use a preconditioner as a direct solver, we can simply do it by setting the type of the KSP object to KSPPREONLY.

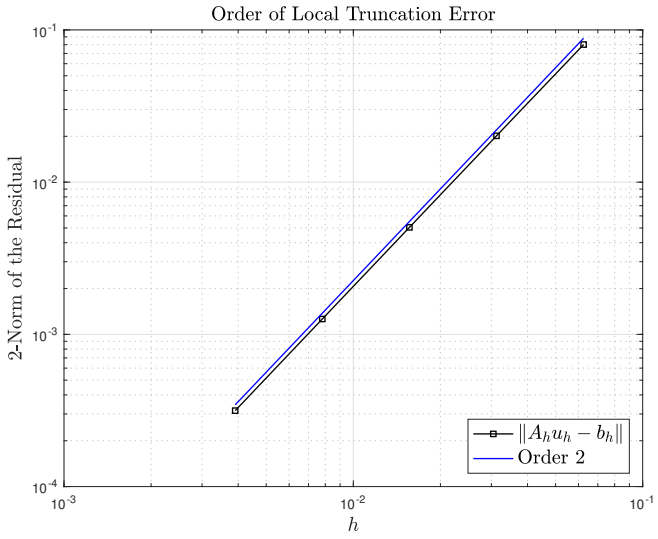

Figure 5: Local truncation error for the numerical scheme discussed in Section 2.

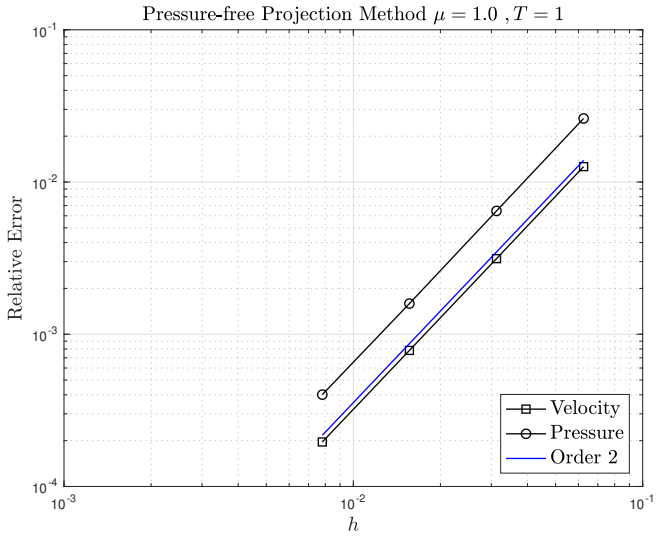

Figure 6: Order of convergence of pressure-free projection method on a periodic domain.

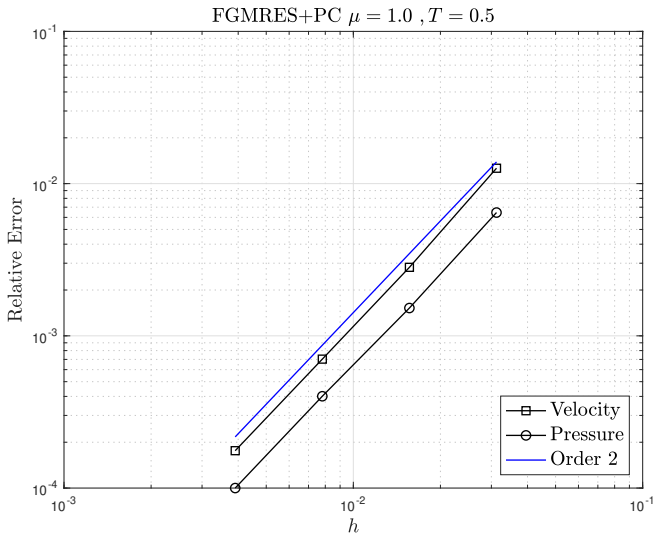

Figure 7: Order of convergence of pressure-free projection method on a periodic domain.

7 Numerical Results

All the numerical experiments in this section are done with manufactured solution

$$\begin{aligned} u(\boldsymbol{x}, t) &= 1 - 2\cos(2\pi(x-t))\sin(2\pi(y-t)) \\ v(\boldsymbol{x}, t) &= 1 + 2\sin(2\pi(x-t))\cos(2\pi(y-t)) \\ p(\boldsymbol{x}, t) &= -(\cos(4\pi(x-t)) + \cos(4\pi(y-t))), \end{aligned}$$

where $\boldsymbol{f}(\boldsymbol{x}, t)$ is calculated correspondingly.

7.1 Order of Accuracy

One further convergence result is showed in Figure 7. This is the result obtained by using GMRES with the projection method preconditioner to solve (3). A 2nd ordered of convergence is again observed.

7.2 Multigrid subsolvers

7.2.1 The error reduction factor ρ

One nice property of multigrid method is that with a good smoothing, the reduction factor after one application of V-cycle is independent of the grid size h

$$\| \text{ Residual after 1 V-cycle } \|_2 \leq \rho \| \text{ Original Residual } \|_2$$

. Therefore, we can achieve a give relative tolerance in a fixed number of cycles no matter how many grid points we are using.

To illustrate this empirically, an experiment is done with the poission multigrid solver in which I used a weighted Jacobi smoother with 3 smoothing sweeps on each level and the relative tolerance of the solver is set to 10^{-6}. The result is recorded in Table 3. We can indeed see that the reduction factors for all N tested are all around 0.09.

N	32		64		128		256	
#V-Cycles	Residual	ρ	Residual	ρ	Residual	ρ	Residual	ρ
0	1.43E+00	n/a	1.10E+00	n/a	7.63E-01	n/a	4.71E-01	n/a
1	3.19E-01	0.22	4.44E-01	0.40	5.91E-01	0.78	7.23E-01	1.53
2	1.68E-02	0.05	2.24E-02	0.05	2.93E-02	0.05	3.60E-02	0.05
3	1.57E-03	0.09	2.04E-03	0.09	2.67E-03	0.09	3.30E-03	0.09
4	1.30E-04	0.08	1.60E-04	0.08	2.03E-04	0.08	2.50E-04	0.08
5	1.08E-05	0.08	1.41E-05	0.09	1.87E-05	0.09	2.34E-05	0.09
6	1.02E-06	0.09	1.24E-06	0.09	1.61E-06	0.09	2.04E-06	0.09

Table 3: The relative residual and the reduction factor after a number of V-cycles for different number of grid points $N = 32, 64, 128, 256$.

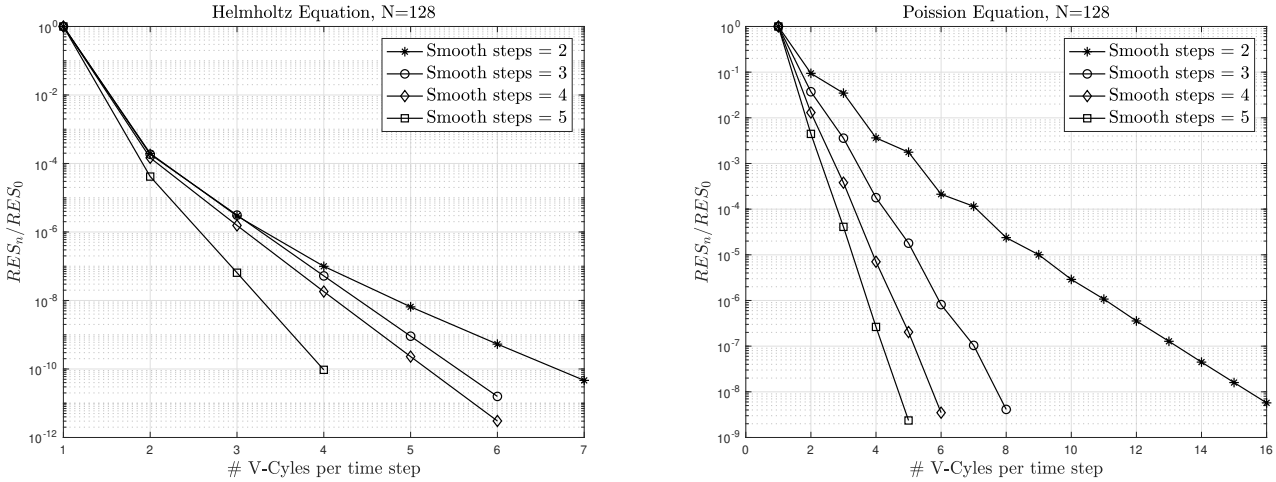

Figure 8: The residual reduction for different number of smoothing steps, 2, 3, 4, 5, applied to each level.

7.2.2 The effect of the number of smoothing sweeps

It is known that the larger the number of smoothing steps, the better the smoothing effect on each level. This will give us a bigger reduction on the residual and hence reduce the number of V-Cycles to reach the relative tolerance condition of the linear system. Figure 8 shows how the reduction factor can be reduced by increasing smoothing sweeps for Helmholtz and Poission Equations. One can observe that for Poission equation, moving from 2 to 3 sweeps reduces half of the V-Cycles, while for Helmholtz equation, only 3 V-Cylcles are reduced from 2 to 5 smoothing sweeps. Since performing one V-Cycles requires communication between levels of grids, it is desirable to reduce number of iterations. However, increasing smoothing sweeps on each level will also double, triple, ..., amount of work per cycle. Therefore, it is not an obvious decision to make and it might also depend from case to case, like the two different behavior showed in the figure.

7.3 Pressure-free Projection Method Preconditioner

7.3.1 The effect of accuracy of the preconditioner

The total amount of work solving the problem can be approximately captured by number of FGMRES iterations plus number of V-Cycles that is done through the whole simulation. Higher accuracy one required for the preconditioner would lead to more V-Cycles involved in the multigrid subsolvers while at the same

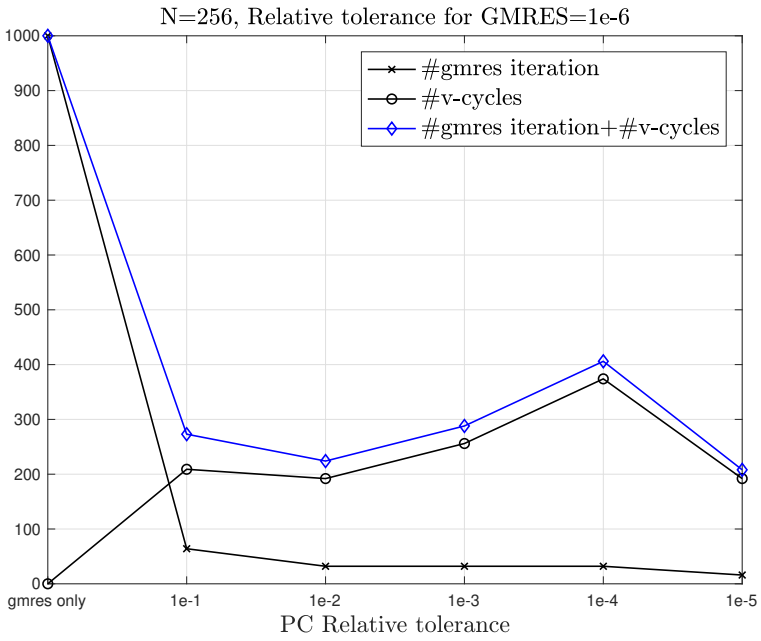

Figure 9: The number of FGMRES iterations and V-Cycles that are done through the simulation until $T = 0.125$. Note that for the first data point, where no preconditioner is used, the iteration number is actually much larger (≥ 8000) and is converging slowly that I terminate the simulation at some point.

time it reduces the number of the GMRES iterations. Figure 9 shows the relationship. Relative tolerance 10^{-5} and 10^{-2} seems to be optimal in this case.

8 Conclusion

The pressure-free projection method preconditioner helps the convergence of the iterative solver a lot. It reduces a large amount of FGMRES iterations. Noticing that it is quite a complicate preconditioner and one may think it is too expensive to use (three linear systems to solve), the multigrid techniques give a big help here. By using it, the amount of work for solving that three linear systems are drastically reduced.

In this project, the code is implemented only for periodic domain, which in fact is not a most exciting case since a stand-alone projection method can already solve the problem. For other boundary conditions, it is harder to construct such method and would be intersting to explore how it would work as a preconditioner.

References

[1] Cai, Mingchao, Andy Nonaka, John B. Bell, Boyce E. Griffith, and Aleksandar Donev. "Efficient variable-coefficient finite-volume Stokes solvers." Communications in Computational Physics 16, no. 5 (2014): 1263-1297.

[2] Griffith, Boyce E. "An accurate and efficient method for the incompressible Navier–Stokes equations using the projection method as a preconditioner." Journal of Computational Physics 228, no. 20 (2009): 7565-7595.