MULTIGRID SOLVER FOR STEADY STATE-STOKES EQUATION

TIANHAO WU

1700010607

This is the report of the project of **Numerical Algebra(2018-2019Autumn)**.

1. Numercial Schemes For steady-state stokes equation

In this report we consider the following PDE problem:

$$\begin{aligned}
&-\Delta\vec{u} + \nabla p = \vec{F}\\
&div\vec{u} = 0\\
&\frac{\partial u}{\partial n} = b, y = 0\\
&\frac{\partial u}{\partial n} = t, y = 1\\
&\frac{\partial v}{\partial n} = l, x = 0\\
&\frac{\partial v}{\partial n} = r, x = 1\\
&u = 0, x = 0, 1\\
&v = 0, y = 0, 1
\end{aligned}$$

Here $\Omega = (0,1) \times (0,1)$ is a square domain on the plane, while n is the External normal vector.

If we suggest the solution is

$$\begin{aligned}
u(x,y) &= (1 - cos(2\pi x))sin(2\pi y)\\
v(x,y) &= -(1 - cos(2\pi y))sin(2\pi x)\\
p(x,y) &= x^3/3 - 1/12
\end{aligned}$$

so that

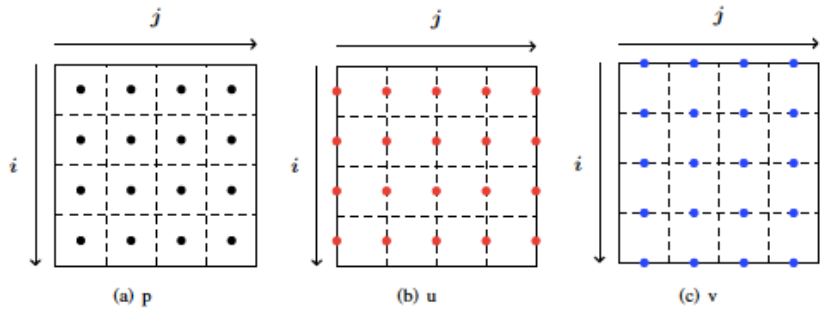
$$\begin{aligned}
f(x,y) &= -4\pi^2(2cos(2\pi x) - 1)sin(2\pi y) + x^2\\
g(x,y) &= 4\pi^2(2cos(2\pi y) - 1)sin(2\pi x)
\end{aligned}$$

1.1. **MAC Discretization.** Let $\vec{u} = (u, v)$ and $\vec{F} = (f, g)$.We rewrite the Stokes equation into corordinate-wise

$$\begin{aligned} -\Delta u + \partial_x p &= f \\ -\Delta v + \partial_y p &= g \\ \partial_x u + \partial_y v &= 0 \end{aligned}$$

The domain Ω is decomposed into small squares with size h. We use two dimension uniform grids for Ω as a typical setting.

Standard central difference discretizaton of Δ and ∂_x, ∂_y at vertices of the uniform grid will not give a stable discretization of Stoke equations. Let i be the row



index and j be the column index ,u(1:n,1:n+1),v(1:n+1,1:n),p(1:n,1:n):

$$\begin{aligned} u_{i,j} &= u((i-0.5)h, (j-1)h) \\ v_{i,j} &= v((i-1)h, (j-0.5)h) \\ p_{i,j} &= p((i-0.5)h, (j-0.5)h) \end{aligned}$$

Using this index system, the MAC scheme can be written as

$$\begin{aligned} \frac{4u_{i,j} - u_{i-1,j} - u_{i+1,j} - u_{i,j-1} - u_{i,j+1}}{h^2} + \frac{p_{i,j} - p_{i,j-1}}{h} &= f_{i,j} \\ \frac{4v_{i,j} - v_{i-1,j} - v_{i+1,j} - v_{i,j-1} - v_{i,j+1}}{h^2} + \frac{p_{i,j} - p_{i-1,j}}{h} &= g_{i,j} \\ \frac{u_{i,j+1} - u_{i,j}}{h} + \frac{v_{i,j} - v_{i+1,j}}{h} &= 0 \end{aligned}$$

Since central difference schemes are used, it is easy to see that the above scheme has second order truncation error at interior nodes. For Neumann boundary condition $\partial u/\partial n|_{\partial\Omega} = t$, the modified stencil is

$$\frac{3u_{1,j} - u_{1,j-1} - u_{1,j+1} - u_{2,j}}{h^2} + \frac{p_{1,j} - p_{1,j-1}}{h} = f_{1,j} + \frac{t_j}{h}$$

To get the matrix form, we use the matlab reshape function to reshape u,v,p into vector U,P. Notice that the first and the last colume of u is defined by the

Direchlet boudary condition and don't need to be updated, so U does not contain these two columns. Similarly, U dose not contain the first and last row of v.

$$U = [reshape(u(:, 2:n), n^2 - n, 1); reshape(v(2:n, :), n^2 - n, 1)]$$
$$P = reshape(p, n^2, 1)$$

As a consequence the matrix form of the discrete $-\Delta, \nabla$ is:

$$-\Delta = A = \frac{1}{h^2}\begin{bmatrix} I_{N-1} \otimes H + G \otimes I_N & 0 \\ 0 & H \otimes I_{N-1} + I_N \otimes G \end{bmatrix},$$
$$\nabla = B = \frac{1}{h}\begin{bmatrix} T \otimes I_N \\ I_N \otimes T \end{bmatrix}.$$

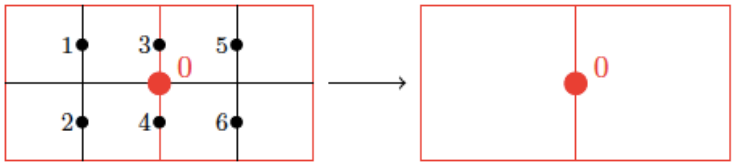
Finally we can write the discrete problem in the matrix form with familiar notation:

$$\begin{bmatrix} A & B \\ B^t & 0 \end{bmatrix}\begin{bmatrix} u_h \\ p_h \end{bmatrix} = \begin{bmatrix} r_n \\ 0 \end{bmatrix}$$

2. Multigrid Method For mac scheme

This section is based on Long Chen's paper: "Programming of MAC scheme for stokes equations".

2.0.1. *Transfer Operators.* This section we define the restriction and prolongation operators, and calculate the explicit matrix form.
The restriction at point 0 will be

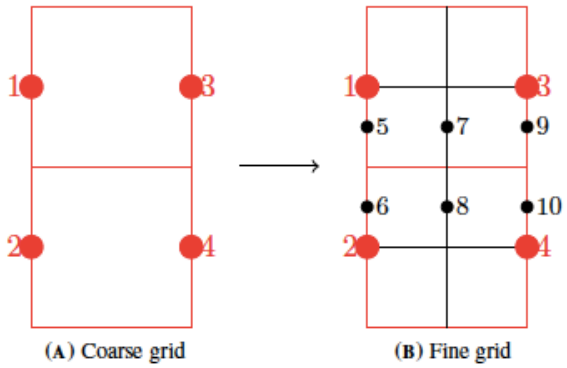


(A) Fine grid (B) Coarse grid

FIGURE 3. Sketch of restriction for vertical edge center dof 0 on coarse grid, (A) fine grid points $1, \cdots, 6$ and coarse grid point 0 on fine grid. (B). coarse grid point 0.

$$u_0 = \frac{1}{8}(u_1 + u_2 + 2u_3 + 2u_4 + u_5 + u_6)$$

Note that the restriction operators are only defined for interior points, since Direchlet boundary conditions are provided for edges. The restriction for v is similar. We don't need the restriction for p. for the prolongation operators:



(A) Coarse grid (B) Fine grid

FIGURE 4. Sketch of prolongation for vertical edge center dofs. (A) coarse grid points $1, \cdots, 4$ on coarse grid. (B). coarse grid points $1, \cdots, 4$ and fine grid points $5, \cdots, 10$ on fine grid.

$$u_5 = \frac{3}{4}u_1 + \frac{1}{4}u_2, u_6 = \frac{3}{4}u_2 + \frac{1}{4}u_1$$
$$u_9 = \frac{3}{4}u_3 + \frac{1}{4}u_4, u_{10} = \frac{3}{4}u_4 + \frac{1}{4}u_3$$
$$u_7 = \frac{1}{2}(u_5 + u_9), u_8 = \frac{1}{2}(u_6 + u_{10})$$

Next we calculate the matrix form of the two operators:
Again note that:

$$U = [reshape(u(:, 2:n), n^2 - n, 1); reshape(v(2:n, :), n^2 - n, 1)]$$
$$P = reshape(p, n^2, 1)$$

Let $h = \frac{1}{2N}$, the prolongation operator:

$$I_{2h}^h : \Omega^{2h} \to \Omega^h$$

$$\begin{bmatrix} U^h \\ P^h \end{bmatrix} = I_{2h}^h \begin{bmatrix} U^{2h} \\ P^{2h} \end{bmatrix},$$

Define:

$$J = \frac{1}{4}\begin{bmatrix} 2 & & & & & & \\ 3 & & & & & & \\ 1 & 1 & & & & & \\ & 3 & & & & & \\ & 3 & 1 & & & & \\ & 1 & 3 & & & & \\ & & 3 & \dots & & & \\ & & 1 & \dots & & & \\ & & & \dots & 1 & & \\ & & & \dots & 3 & & \\ & & & & 3 & 1 & \\ & & & & 1 & 3 & \\ & & & & & 2 & \end{bmatrix}_{2N\times N}$$

$$K = \frac{1}{2}\begin{bmatrix} 1 & & & \\ 2 & & & \\ 1 & 1 & & \\ & 2 & & \\ & 1 & \dots & \\ & & \dots & \\ & & \dots & 1 \\ & & & 2 \\ & & & 1 \end{bmatrix}_{(2N-1)\times(N-1)}$$

$$L = \begin{bmatrix} 1 & & & \\ 1 & & & \\ & 1 & & \\ & 1 & & \\ & & \dots & \\ & & \dots & \\ & & & 1 \\ & & & 1 \end{bmatrix}_{2N\times N}$$

$$I_{2h}^{h} = diag(K \otimes J, J \otimes K, L \otimes L)$$

The restriction operator:

$$I_h^{2h} : \Omega^h \to \Omega^{2h}$$
$$\begin{bmatrix} U^{2h} \\ P^{2h} \end{bmatrix} = I_h^{2h} \begin{bmatrix} U^h \\ P^h \end{bmatrix},$$

Let

$$H = \frac{1}{4}K^T = \begin{bmatrix} 1 & 2 & 1 & & & & & & \\ & & 1 & 2 & 1 & & & & \\ & & & & \dots & \dots & \dots & & \\ & & & & & & 1 & 2 & 1 \end{bmatrix}_{(N-1)\times(2N-1)}$$

$$I_h^{2h} = diag(H \otimes L^T, L^T \otimes H, \frac{1}{16}L^T \otimes L^T)$$

2.1. **V-cycle.** The Full Multigrid cycle:

Algorithm 1 U=Vcycle(A,U,F)

```
if not converge then
    Relax v_1 times on A^h U^h = F^h
    if Ω^h is not the coarsest grid then
        F^{2h} ← I_{2h}^h (F^h − A^h U^h)
        h ← 2h
        U^h ← 0
        Relax v_1 times on A^h U^h = F^h
    if Ω^h is not the finest grid then
        h ← h/2
        U^h ← U^h + I_h^{2h} U^{2h}
        Relax v_2 times on A^h U^h = F^h
```

3. Methods of Relaxation

3.1. **Modified Distributive Gauss-Seidel Relexation.** The idea of the distributive relaxation is to transform the principle operators to the main diagonal and apply the equation-wise decoupled relaxation.
Denote by

$$L = \begin{bmatrix} A & B \\ B & 0 \end{bmatrix}, and\ M = \begin{bmatrix} I & B \\ 0 & -B^TB \end{bmatrix}$$

then

$$LM = \begin{bmatrix} A & AB - BB^TB \\ B & B^TB \end{bmatrix} \approx \begin{bmatrix} A & 0 \\ B & B^TB \end{bmatrix} := \widetilde{LM}$$

By " $\approx$ " here we mean that the commutator $E = AB - BB^TB$ is small or if of low rank.
Since in the continuous level, with certain assumptions on the smoothness and boundary conditions, E can be manipulate as

$$(-\Delta + \nabla div)\nabla = curlcurl\nabla = 0$$

So the matrix $M\widetilde{LM}^{-1}$ will be an good approximation of $L^{-1} = M(LM)^{-1}$. It defines an iterative method:

$$x^{k+1} = x^k + M\widetilde{LM}^{-1}(b - Lx^k)$$

where $x^k = (u^k, p^k), b = (F, 0)$.

The so-called DGS smoother comes from consecutively Gauss-Seidel relaxation applied to the operator$\widetilde{LM}$. That is to solve:

$$\begin{bmatrix} A & 0 \\ B & B^TB \end{bmatrix} \begin{bmatrix} \delta w \\ \delta q \end{bmatrix} = \begin{bmatrix} r_u \\ r_p \end{bmatrix}$$

approximately and update it through the distributive matrix M.

Algorithm 2 $[u^{k+1}, p^{k+1}]\leftarrow$DGS$(u^k, p^k, f, g)$

1: $u^{k+1/2} = u^k + \widehat{A}^{-1}(f - Au^k - Bp^k)$
2: $A_p = B^TB$
3: $\delta q = \widehat{A_p}^{-1}(g - B^Tu^{k+1/2})$
4: $u^{k+1} = u^{k+1/2} + B\delta q$
5: $p^{k+1} = p - B^T(B\delta q)$

3.2. **Matrix-free implementation.** When implement the Gauss-Seidel iteration for finite difference methods, the direct update version is more efficient. The update for velocity u(i,j) for i=2:n-1,j=2:n is

$$u_{i,j} = (h^2 f_{i,j} + u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - h(p_{i,j} - p_{i,j-1}))/4$$

For near boundary subscript i,j, the stencil should be modified to impose the boundary condition.

After the update of velocity, we need to update the residual for the continuity equation compute $r_p = g - B^Tu = g + divu$, which can be done efficiently using the matlab conv finction. Then apply Gauss-Seidel iteration to solve $A_p\delta q = r_p$, because $A_p = B^TB$ is the laplacian of pressure with Neumann boundary condition, so it will be:

$$\delta q_{i,j} = (h^2 r_p i, j + \delta q_{i-1,j} + \delta q_{i+1,j} + \delta q_{i,j-1} + \delta q_{i,j+1})/4$$

The update for u,v,p will be

$$u_{i,j} = u_{i,j} + (\delta q_{i,j} - \delta q_{i,j-1})/h$$
$$p_{i,j} = p_{i,j} - (4\delta q_{i,j} - \delta q_{i-1,j} - \delta q_{i+1,j} - \delta q_{i,j-1} - \delta q_{i,j+1})/h^2$$

For boundary and corner cells, the stencil should be modified accordingly.

3.3. **Uzawa Iteration Method.** Set $P_0, k = 0$:

(1) Solve $AU_{k+1} = F - BP_k$;
(2) Update pressure $P_{k+1} = P_k + \alpha(B^t U_{k+1})$;
(3) If not converge, return to the fist line

Observed that the parameter α satisfy:

$$P_{k+1} = P_k + \alpha(B^t A^{-1}(F - BP_k)) = (I - \alpha B^t A^{-1} B)P_k + \alpha B^t A^{-1} F,$$

Which is equivalent to:

$$B^t A^{-1} BP = B^t A^{-1} F$$

Now we proof that the best α is α=argmin$\rho(I - \alpha B^t A^{-1} B)$:

$$\alpha_* = \frac{2}{\lambda_{min}(B^t A^{-1} B) + \lambda_{max}(B^t A^{-1} B)}.$$

Proof. Denote $M = B^t A^{-1} B$.
Then

$$Spec(I - \alpha M) = \{1 - \alpha\lambda | \lambda \in Spec(M)\},$$

Set $\lambda_1 = min\ \sigma(M)$, $\lambda_2 = max\ \sigma(M)$,So

$$min \quad \rho(I - \alpha M)$$

$$\Leftrightarrow min \ \ max\{|1 - \alpha\lambda_1|, |1 - \alpha\lambda_2|\} \tag{1}$$

$$\begin{aligned} & max\{|1 - \alpha\lambda_1|, |1 - \alpha\lambda_2|\} \\ & \frac{|1 - \alpha\lambda_1| + |1 - \alpha\lambda_2|}{2} \\ & |\alpha(\lambda_1 - \lambda_2)| \end{aligned}$$

For the equation to hold, we have:

$$\alpha = \frac{2}{\lambda_1 + \lambda_2}$$

□

3.4. **Inexact Uzawa iteration method.** Set $P_0, k = 0$:

(1) Solve $AU_{k+1} = F - BP_k$, get the inexact solution $\hat{U}_{k+1}$;
(2) Update pressure $P_{k+1} = P_k + \alpha(B^t\hat{U}_{k+1})$;
(3) If not converge, return to the fist line.

Let $\hat{U}_{k+1}$ be the approximate solution for $AU_{k+1} = F - BP_k$. Define:

$$\delta_k = A\hat{U}_{k+1} - F + BP_k,$$

If

$$||\delta_k|| \leq \tau||B^t\hat{U}_k||,$$

For proper α, the inexact Uzawa iteration method converge for sufficiently small τ.

4. Numerical Result

The following results were tested on MacBook Pro.

4.1. **Problem 1.** In problem 1, we use the v-cycle scheme and DGS to relax $A^hU^h = F^h - B^hP^h$. We use diffenrent v_1,v_2,L to test the speed and accuracy.The shutdown standard used is $|r_h|_2/|F_h|_2 < 1e-8$

Code description: Run 'problem1.m' to get the result, the code is based on the previous section of modified DGS method(dgs2.m), also contains the origin version of DGS(dgs1.m).

Table 1. P1,v_1=1,v_2=1,L=log_2(N)

N=64	Time=0.084972	err=1.495080e-03	Iter=31
N=128	Time=0.224051	err=3.736297e-04	Iter=36
N=256	Time=1.007642	err=9.339875e-05	Iter=41
N=512	Time=6.673550	err=2.334925e-05	Iter=45
N=1024	Time=31.472865	err=5.837335e-06	Iter=49
N=2048	Time=149.349985	err=1.459388e-06	Iter=52

Table 2. P1,v_1=1,v_2=0,L=log_2(N)

N=64	Time=0.059459	err=1.495079e-03	Iter=40
N=128	Time=0.191346	err=3.736291e-04	Iter=48
N=256	Time=0.987053	err=9.339848e-05	Iter=56
N=512	Time=5.125184	err=2.334906e-05	Iter=64
N=1024	Time=24.141318	err=5.837228e-06	Iter=71
N=2048	Time=127.509244	err=1.459304e-06	Iter=79

TABLE 3. P1,v_1=1,v_2=0,L=log_2(N)-1

N=64	Time=0.075684	err=1.495081e-03	Iter=56
N=128	Time=0.271118	err=3.736296e-04	Iter=62
N=256	Time=1.333705	err=9.339872e-05	Iter=67
N=512	Time=5.790284	err=2.334921e-05	Iter=71
N=1024	Time=27.773149	err=5.837285e-06	Iter=76
N=2048	Time=126.125273	err=1.459327e-06	Iter=80

TABLE 4. P1,v_1=1,v_2=0,L=log_2(N)-2

N=64	Time=0.235350	err=1.495077e-03	Iter=167
N=128	Time=0.851425	err=3.736283e-04	Iter=181
N=256	Time=3.301743	err=9.339820e-05	Iter=191
N=512	Time=15.429253	err=2.334899e-05	Iter=199
N=1024	Time=70.734955	err=5.837237e-06	Iter=206
N=2048	Time=343.913267	err=1.459340e-06	Iter=212

As we can see, an increase in v_1,v_2 will result in a decrease in the number of iterations, but has little effect on speed. And the decrease of L(L< log_2(N)-1) will greatly affect the convergence speed and the number of iterations. All parameters have nearly no effect on the error of the converged results.

4.2. **Problem 2.** In problem 2, we use the v-cycle scheme and Uzawa method to relax U^h, P^h. We use diffenrent v_1,v_2,α,L to test the speed and accuracy. The shutdown standard used is $|r_h|_2/|F_h|_2 < 1e-8$

Code description: Run 'problem2.m' to get the result, the code is based on the previous section of Uzawa method(uzawa.m).

When n is large, it takes a long time to perform any exact solution, but

TABLE 5. P2,v_1=1,v_2=0,α=1,L=log_2(N)

N=64	Time=0.036949	err=1.495079e-03	Iter=1
N=128	Time=0.125797	err=3.736291e-04	Iter=1
N=256	Time=0.426041	err=9.339849e-05	Iter=1
N=512	Time=2.257153	err=2.334907e-05	Iter=1
N=1024	Time=10.426311	err=5.837241e-06	Iter=1
N=2048	Time=105.729440	err=1.459335e-06	Iter=1

choosing the appropriate parameters can make iteration=1, so the overall speed is fast. Since the error after one iteration is already quite accurate(1e-13), a proper reduction in L will result in a slight increase in speed. Speed is very sensitive to

TABLE 6. P2,v_1=1,v_2=0,α=1,L=log_2(N)-2

N=64	Time=0.030631	err=1.495079e-03	Iter=1
N=128	Time=0.089446	err=3.736291e-04	Iter=1
N=256	Time=0.428933	err=9.339849e-05	Iter=1
N=512	Time=2.075981	err=2.334907e-05	Iter=1
N=1024	Time=9.620768	err=5.837241e-06	Iter=1
N=2048	Time=103.311387	err=1.459334e-06	Iter=1

TABLE 7. P2,v_1=1,v_2=0,α=1,L=log_2(N)-4

N=64	Time=0.021054	err=1.495079e-03	Iter=1
N=128	Time=0.093496	err=3.736291e-04	Iter=1
N=256	Time=0.411697	err=9.339849e-05	Iter=1
N=512	Time=2.188154	err=2.334907e-05	Iter=1
N=1024	Time=10.185860	err=5.837241e-06	Iter=1
N=2048	Time=119.164785	err=1.459334e-06	Iter=1

parameters when N is large(especially for α, α=0.9 or 1.1 will make iteration > 1, and slow down the speed).

4.3. **Problem 3.** In problem 3, we use the v-cycle scheme and inexact Uzawa method to relax U^h, P^h. We use diffenrent v_1,v_2,α,L to test the speed and accuracy.The shutdown criterion used is $|r_h|_2/|F_h|_2 < 1e-8$, the shutdown critertion for inexact Uzawa method used is iteration = v, or $|r_h|_2 < \tau|B_h^T U_h|_2$
Code description: Run 'problem3.m' to get the result, the code is based on the previous section of inexact Uzawa method(inexact_uzawa.m).

TABLE 8. P3,v_1=2,v_2=0,α=0.35,L=log_2(N),first shutdown criterion

N=64	Time=0.056172	err=1.495079e-03	Iter=36
N=128	Time=0.206244	err=3.736294e-04	Iter=39
N=256	Time=1.051388	err=9.339895e-05	Iter=43
N=512	Time=5.227819	err=2.334930e-05	Iter=49
N=1024	Time=25.024408	err=5.837304e-06	Iter=56
N=2048	Time=115.851110	err=1.459338e-06	Iter=62

The second shutdown criterion for inexact Uzawa iteration method is not very stable. So i prefer using the first criterion. When using the first criterion, the selection of α should belong to a range($[0.2, 0.8]$). Too small will affect the convergence speed, If it is too large, it may not converge(a>1.5).

TABLE 9. P3,v_1=2,v_2=0,α=0.8,L=log_2(N),first shutdown criterion

N=64	Time=0.091761	err=1.495079e-03	Iter=41
N=128	Time=0.222120	err=3.736292e-04	Iter=42
N=256	Time=0.995999	err=9.339880e-05	Iter=44
N=512	Time=4.734277	err=2.334922e-05	Iter=50
N=1024	Time=26.268441	err=5.837300e-06	Iter=56
N=2048	Time=120.307309	err=1.459336e-06	Iter=62

TABLE 10. P3,α=0.4,L=log_2(N),τ=1,second shutdown criterion

N=64	Time=0.034121	err=1.495079e-03	Iter=1
N=128	Time=0.098278	err=3.736291e-04	Iter=1
N=256	Time=0.393438	err=9.339849e-05	Iter=1
N=512	Time=2.080113	err=2.334907e-05	Iter=1
N=1024	Time=10.271136	err=5.837241e-06	Iter=1
N=2048	Time=113.308395	err=1.459334e-06	Iter=1

4.4. **Problem 4.** In problem 4, we use the v-cycle scheme to solve $A^hU^h = F^h - B^hP^h$, and use the Uzawa method to update P .We use diffenrent v,α,L to test the speed and accuracy.The shutdown criterion used is $|r_h|_2/|F_h|_2 < 1e-8$
Code description: Run 'problem4.m' to get the result, the code is based on the previous section of inexact Uzawa method(inexact_uzawa.m) and the vcycle scheme(function vcyc which lies behind problem4.m)
We can see that this is the fastest of the four methods. N=2048 takes less than

TABLE 11. P4,v=2,L=5,α=1.1

N=64	Time=0.037634	err=1.495079e-03	Iter=25
N=128	Time=0.112634	err=3.736291e-04	Iter=24
N=256	Time=0.474461	err=9.339849e-05	Iter=22
N=512	Time=2.853006	err=2.334907e-05	Iter=22
N=1024	Time=10.956992	err=5.837234e-06	Iter=20
N=2048	Time=44.119860	err=1.459306e-06	Iter=19

50 seconds. $\alpha = 1.1$ is almost the best parameter for this problem, But the perturbation of α does not have much effect on speed. Here we have made a little optimization, that is, when the grid is small, we turn to the exact solution.

4.5. **Error Comparison of 4 methods.** The following four figures shows $u - u_{exact}, v - v_{exact}, p - p_{exact}$

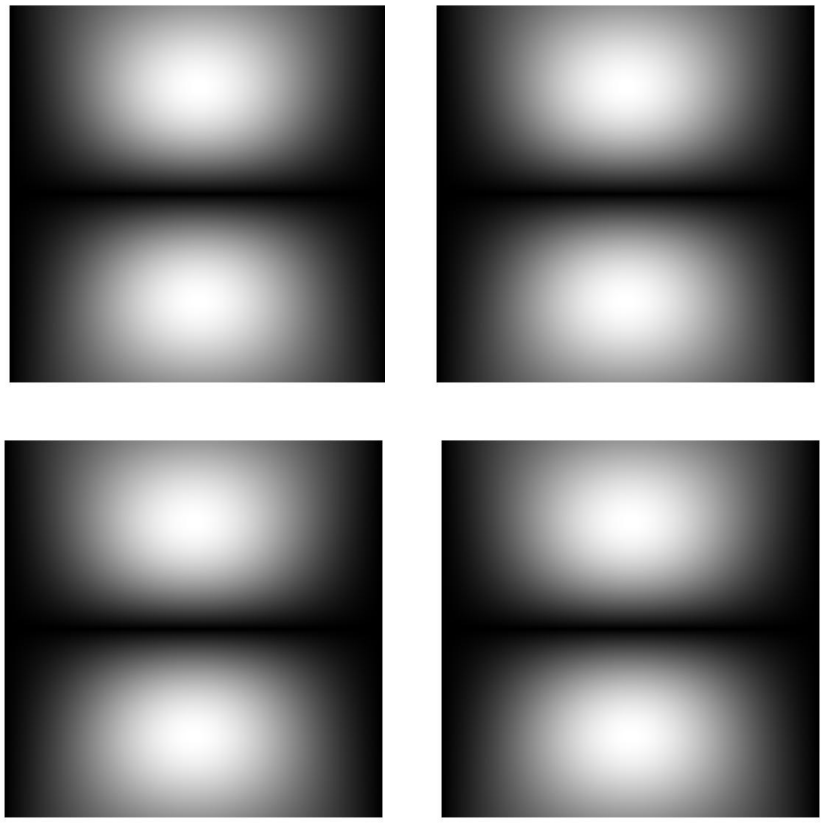


FIGURE 1. The error of u for problem 1,2,3,4 respectively

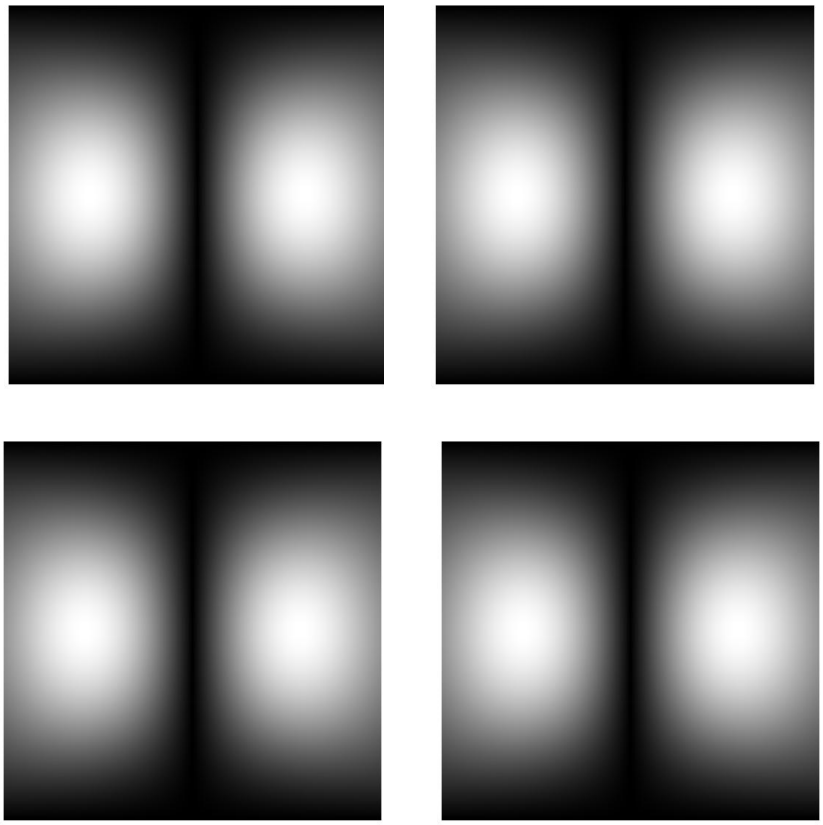


FIGURE 2. The error of v for problem 1,2,3,4 respectively

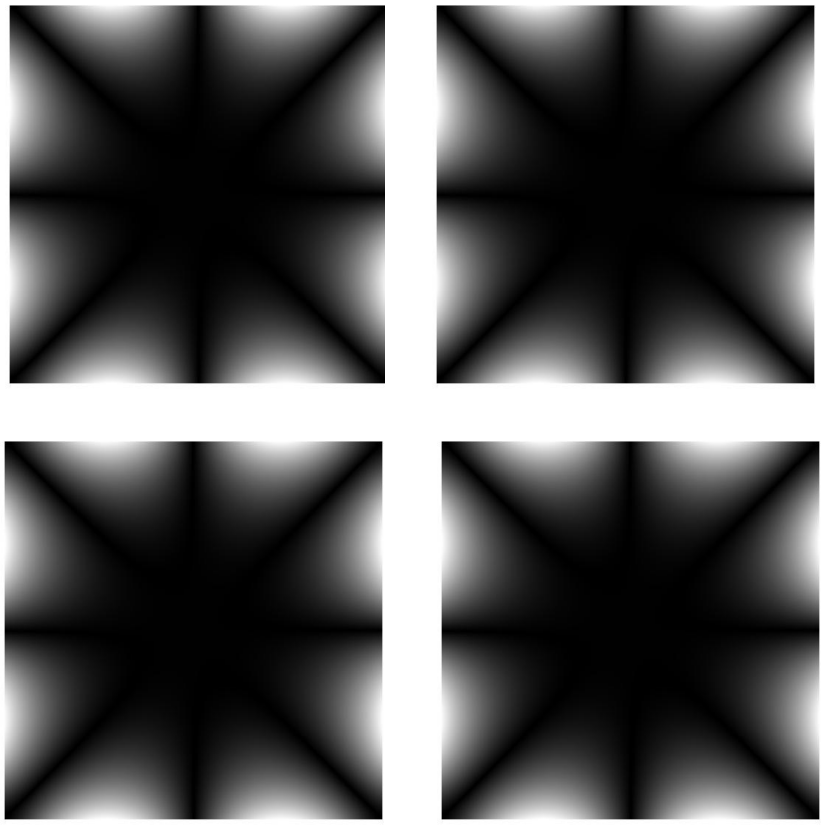


FIGURE 3. The error of p for problem 1,2,3,4 respectively

REFERENCES

[1] A. Brandt and N. Dinar. Multigrid solutions to elliptic flow problems. In S. Parter, editor, *Numerical Methods for Partial Differential Equations*, pages 53–147. Academic Press, New York, 1979. 3, 4

[2] F. Harlow and J. Welch. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Physics of fluids*, 8(12):2182, 1965. 1