SOLVING STOKES EQUATION WITH MAC METHOD

ABSTRACT. In this notes, we summarize numerical methods for solving Stokes equations on rectangular grid, and solve it by multigrid vcycle method with distributive Gauss-Seidel relaxation as smoothing. The numerical methods we concerned are MAC scheme, nonconforming rotate bilinear FEM and nonconforming rotate bilinear FVM.

1. PROBLEM STATEMENT

We consider Stokes equation

$$\begin{cases} -\mu\Delta \boldsymbol{u} + \nabla p = \boldsymbol{f} & \text{in } \Omega, \\ \nabla \cdot \boldsymbol{u} = 0 & \text{in } \Omega. \\ \boldsymbol{u} = 0 & \text{on } \partial\Omega \end{cases} \tag{1.1}$$

where $\boldsymbol{u} = (u, v)^t$, and $\boldsymbol{f} = (f_1, f_2)^t$.

2. MAC DISCRETIZATION

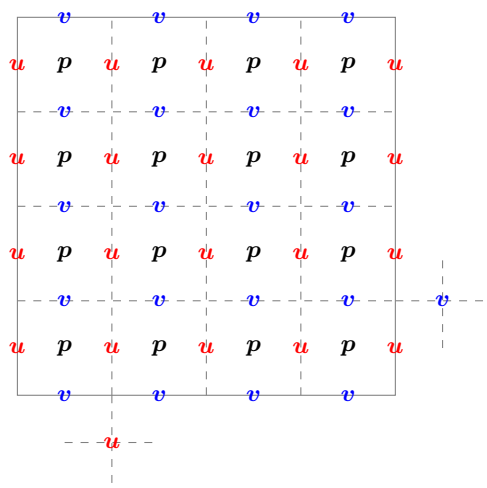


FIGURE 1. Index for p, u, v.

Suppose we have a rectangular decomposition, for each cell, the degree of freedoms for u and v are located on the vertical edge centers and horizontal edge centers, respectively, and the degree of freedoms for pressure p are located at cell centers. Suppose that $u^{i,j}$, $v^{i,j}$, and $p^{i,j}$ is approximations of the point values $u(x_i, y_{j+\frac{1}{2}})$, $v(x_{i+\frac{1}{2}}, y_j)$, and $p(x_{i+\frac{1}{2}}, y_{j+\frac{1}{2}})$, respectively; See Figure 2 for the position of $u^{i,j}$, $v^{i,j}$, and $p^{i,j}$. The MAC

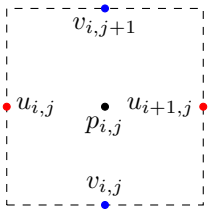


FIGURE 2. locations for dof of u, v, p.

scheme is written as ($\mu = 1$)

$$\frac{4u^{i,j} - u^{i-1,j} - u^{i+1,j} - u^{i,j-1} - u^{i,j+1}}{h^2} + \frac{p^{i,j} - p^{i-1,j}}{h} = f_1^{i,j} \tag{2.1}$$

$$\frac{4v^{i,j} - v^{i-1,j} - v^{i+1,j} - v^{i,j-1} - v^{i,j+1}}{h^2} + \frac{p^{i,j} - p^{i,j-1}}{h} = f_2^{i,j} \tag{2.2}$$

$$\frac{u^{i+1,j} - u^{i,j}}{h} + \frac{v^{i,j+1} - v^{i,j}}{h} = 0 \tag{2.3}$$

It's easy to see that the above scheme has second order truncation error.

2.1. **Dirichlet boundary condition.**

(1) Square domain

How to handle the boundary condition seems not sensitive for the finite difference method. For the grid points outside of the domain (ghost points), we can approximate it through either linear or quadratical extropolation, with the use of boundary conditions for $\boldsymbol{u}$.

(2) Lshape domain

The Lshape domain we are going to consider is the following. The red solid circle and the blue solid circle are the points near boundary for u and v, respectively.

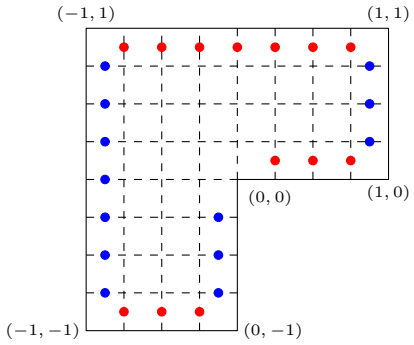


FIGURE 3. Boundary points of MAC scheme on L shape domain. The red solid circle "●" denote grid points for u, and blue ones "●" denote grid points for v.

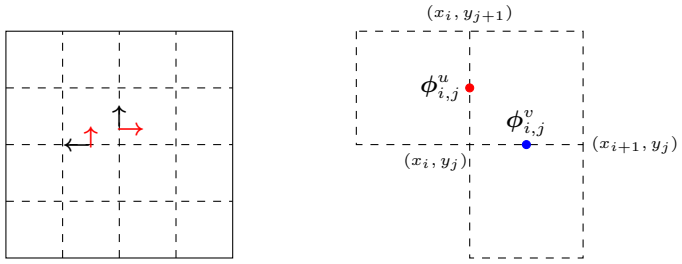
2.2. **Neumann boundary condition.** We haven't tested such case.

3. Derivation of MAC scheme from "Stream Function-Vorticity-Pressure" formulation

In this section, we will provide details on deriving the MAC scheme from mixed finite element methods.

3.1. **Derivation of the scheme.**

- **Step 1**: Grid orientation, bases in RT_0, and matrix incidence.



(A) Grid orientation **(B)** Bases in RT_0

FIGURE 4. Grid orientation and bases in RT_0.

Let $\mathcal{T}$ denote a rectangular decomposition of Ω with NT rectangles, NE edges including NE_0 interior edges and N nodes of which N_0 are interior nodes, and grid spacing h for x and y direction. For each edge $e_{i,j}$, we fix a tangent direction $\boldsymbol{t}_{i,j}$ and norm direction $\boldsymbol{n}_{i,j}$. To derive a consistent scheme with MAC scheme, we take the normal direction of vertical and horizontal edges the same with $x-$ and $y-$ axes, and the tangent direction $\boldsymbol{t}_{i,j}$ will be taken such that $(\boldsymbol{n}_{i,j}, \boldsymbol{t}_{i,j})$ are oriented like the (x, y) axes of the coordinate system; See Figure 4 **(A)** for the grid orientation. For convenience, we also adopt notations $e^u_{i,j}$, $e^v_{i,j}$, $\boldsymbol{n}^u_{i,j}$, and $\boldsymbol{n}^v_{i,j}$ etc., to distinguish quantities on vertical and horizontal edge centers.

For RT_0 elements defined on rectangular grids, we associate with each interior vertical edge $e^u_{i,j}$ and horizontal edge $e^v_{i,j}$ a dual functionl $\Phi^u_{i,j}(\boldsymbol{u}) = \frac{1}{h}\int_{e^u_{i,j}} \boldsymbol{u}\cdot\boldsymbol{n}^u_{i,j}$ and $\Phi^h_{i,j}(\boldsymbol{u}) = \frac{1}{h}\int_{e^v_{i,j}} \boldsymbol{u}\cdot\boldsymbol{n}^v_{i,j}$, respectively. The corresponding bases in RT_0 are denoted by $\boldsymbol{\phi}^u_{i,j}$ and $\boldsymbol{\phi}^v_{i,j}$. With such bases in RT_0, we then expend element $\boldsymbol{u}_h = \begin{pmatrix} u_h \\ v_h \end{pmatrix} \in RT_0(\mathcal{T})$ as

$$\boldsymbol{u}_h = \sum u_{i,j}\boldsymbol{\phi}^u_{i,j} + \sum v_{i,j}\boldsymbol{\phi}^v_{i,j}.$$

Clearly, we have $u_{i,j} = u_h(x_i, y_{j+\frac{1}{2}})$ and $v_{i,j} = v_h(x_{i+\frac{1}{2}}, y_j)$, and thus degree of freedoms $u_{i,j}$ and $v_{i,j}$ in RT_0 are approximations of nodal values at points $(x_i, y_{j+\frac{1}{2}})$ and $(x_{i+\frac{1}{2}}, y_j)$.

Denote by $\mathcal{C}$ the $NE\times N$ edge-node incidence matrix of element $\tau \in \mathcal{T}$, where

$$\mathcal{C}_{i,j} := \begin{cases} +1 & \text{if edge } i \text{ is directed into node } j, \\ -1 & \text{if edge } i \text{ is directed outof node } j, \\ 0 & \text{if edge } i \text{ does not meet node } j. \end{cases}$$

Denote by $\mathcal{D}$ the $NT \times NE$ element-edge incidence matrix of element $\tau \in \mathcal{T}$, where

$$\mathcal{D}_{i,j} := \begin{cases} +1 & \text{if normal direction of edge } j \text{ is directed outof element } i. \\ -1 & \text{if normal direction of edge } j \text{ is directed into element } i, \\ 0 & \text{if normal direction of edge } j \text{ does not meet element } i. \end{cases}$$

Note that $\mathcal{C}$ and $\mathcal{D}$ are discrete version of curl and div operators without scaling.

- **Step 2**: Mixed formulation of Stream Function-Vorticity-Pressure" system.

In order to distinguish curl operator acting on vector and scalar quantities, for scalar ω and vector $\boldsymbol{u} = (u, v)^t$, we denote

$$\mathrm{curl}\,\omega = \left(\partial_y \omega, \partial_x \omega\right), \ \mathrm{rot}\,\boldsymbol{u} = \partial_x v - \partial_y u.$$

Define the vorticity $\omega = \mathrm{rot}\,\boldsymbol{u}$. Notice the divergence free condition and make the use the following identity

$$-\Delta = -\,\mathrm{grad}\,\mathrm{div} + \mathrm{curl}\,\mathrm{rot}, \tag{3.1}$$

Stokes system (1.1) can be written as

$$\begin{cases} \omega = \mathrm{rot}\,\boldsymbol{u}, & \text{in } \Omega, \\ \mathrm{curl}\,\omega + \mathrm{grad}\,p = \boldsymbol{f}, & \text{in } \Omega, \\ -\mathrm{div}\,\boldsymbol{u} = 0, & \text{in } \Omega, \\ \boldsymbol{u} = 0, & \text{on } \partial\Omega. \end{cases} \tag{3.2}$$

Since

$$\mathrm{curl} : H^1(\Omega) \to H(\mathrm{div}, \Omega),$$

the dual operator rot of curl can be understood as

$$\mathrm{rot} : H(\mathrm{div}, \Omega) \to H^1(\Omega).$$

$$H(\mathrm{curl}) \underset{\mathrm{rot}}{\overset{\mathrm{curl}}{\rightleftarrows}} H(\mathrm{div}) \underset{\mathrm{grad}}{\overset{\mathrm{div}}{\rightleftarrows}} L^2$$

The weak formulation of the system (3.2) can be written as

Find $(\omega, \boldsymbol{u}, p) \in H(\mathrm{curl}; \Omega) \times H_0(\mathrm{div}; \Omega) \times L_0^2(\Omega)$, such that

$$\begin{cases} (\omega, \tau) - (\boldsymbol{u}, \mathrm{curl}\,\tau) = 0, & \text{for } \tau \in H(\mathrm{curl}; \Omega), \\ (\mathrm{curl}\,\omega, \boldsymbol{v}) - (p, \mathrm{div}\,\boldsymbol{v}) = (\boldsymbol{f}, \boldsymbol{v}), & \text{for } \boldsymbol{v} \in H_0(\mathrm{div}; \Omega), \\ -(\mathrm{div}\,\boldsymbol{u}, q) = 0, & \text{for } q \in L_0^2(\Omega). \end{cases} \tag{3.3}$$

If we approximate spaces $H(\mathrm{curl}; \Omega)$, $H_0(\mathrm{div}; \Omega)$ and $L_0^2(\Omega)$ with conforming linear finite element space W_h, Raviart-Thomas element of lowest order $\boldsymbol{U}_h$, and piecewise constant space Q_h, respectively, then discretized mix formulation can be written as

Find $(\omega_h, \boldsymbol{u}_h, p_h) \in W_h \times \boldsymbol{U}_h \times Q_h$, such that

$$\begin{cases} (\omega_h, \tau_h) - (\boldsymbol{u}_h, \mathrm{curl}\,\tau_h) = 0, & \text{for } \tau_h \in W_h, \\ (\mathrm{curl}\,\omega_h, \boldsymbol{v}_h) - (p_h, \mathrm{div}\,\boldsymbol{v}_h) = (\boldsymbol{f}, \boldsymbol{v}_h), & \text{for } \boldsymbol{v}_h \in \boldsymbol{U}_h, \\ -(\mathrm{div}\,\boldsymbol{u}_h, q_h) = 0, & \text{for } q_h \in Q_h. \end{cases} \tag{3.4}$$

- Step 3 Matrix form and Mass lumping
  Written in matrix form, we have

$$\begin{cases} M_{\mathrm{curl}}\omega - \mathcal{C}^t H^{-1} M_{\mathrm{div}} \boldsymbol{u} = 0, \\ M_{\mathrm{div}} H^{-1}\mathcal{C}\omega - H\mathcal{D}^t p = \boldsymbol{f}, \\ -\mathcal{D}H\boldsymbol{u} = 0, \end{cases} \tag{3.5}$$

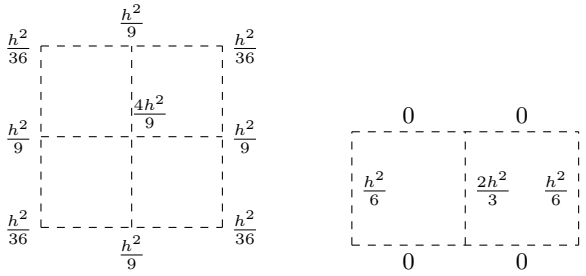
  where $H = diag(h)$, M_{curl} and M_{div} are the scaling matrix, mass matrix in $H(\mathrm{curl};\Omega)$ and mass matrix in $H(\mathrm{div};\Omega)$, respectively. If we define the discrete curl and div operator with scaling as $C = H^{-1}\mathcal{C}$, $B = -\mathcal{D}H$, then system (3.5) is replaced by

$$\begin{pmatrix} -M_{\mathrm{curl}} & C^t M_{\mathrm{div}} & 0 \\ M_{\mathrm{div}} C & 0 & B^t \\ 0 & B & 0 \end{pmatrix} \begin{pmatrix} \omega \\ \boldsymbol{u} \\ p \end{pmatrix} = \begin{pmatrix} 0 \\ \boldsymbol{f} \\ 0 \end{pmatrix}. \tag{3.6}$$

  Eliminate ω, we have

$$\begin{pmatrix} M_{\mathrm{div}} C M_{\mathrm{curl}}^{-1} C^t M_{\mathrm{div}} & B^t \\ B & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{u} \\ p \end{pmatrix} = \begin{pmatrix} \boldsymbol{f} \\ 0 \end{pmatrix}. \tag{3.7}$$

  The stencil for mass matrix in $H(\mathrm{curl})$ and $H(\mathrm{div})$ are the following:



(A) Stencil for M_{curl}. **(B)** Stencil for M_{div}.

FIGURE 5. Stencil of mass matrix in $H(\mathrm{curl})$ and $H(\mathrm{div})$.

If we use mass lumping for mass matrix in $H(\mathrm{div})$ and $H(\mathrm{curl})$, then

$$M_{\mathrm{curl}} = diag(h^2),\ M_{\mathrm{div}} = diag(h^2).$$

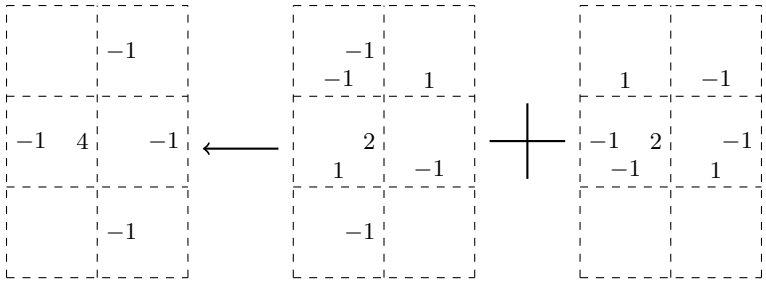
and notice that $\mathcal{D}^t\mathcal{D}\boldsymbol{u} = 0$, then the above system can then be simplied as

$$\begin{pmatrix} \mathcal{C}\mathcal{C}^t + \mathcal{D}^t\mathcal{D} & B^t \\ B & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{u} \\ p \end{pmatrix} = \begin{pmatrix} \boldsymbol{f} \\ 0 \end{pmatrix}. \tag{3.8}$$

If we denote $\mathcal{A}$ as the discreted Laplacian matrix without scaling in MAC scheme, we can check row by row that for interior points,

$$\mathcal{A} = \mathcal{C}\mathcal{C}^t + \mathcal{D}^t\mathcal{D}. \tag{3.9}$$

- Step 4. Stencil verification
  We give stencil verification for a vertical edge center, and similarly, we can give one for a horizontal edge center. Finally, we get MAC scheme from mixed formulation.



(A) $\mathcal{A}$ (B) $\mathcal{C}\mathcal{C}^t$ (C) $\mathcal{D}^t\mathcal{D}$

FIGURE 6. Stencil verification

3.2. **Basis for $H(\mathrm{curl})$ and $H(\mathrm{div})$.**

3.2.1. *Two Dimensional.* For two dimensional case, the basis for $H(\mathrm{curl})$ is the same with $H(\mathrm{grad})$. We take $[0,1]\times[0,1]$ as the reference element. The bilinear bases can be written as

$$(1-\xi)(1-\eta),\ \xi(1-\eta),\ \xi\eta,\ (1-\xi)\eta,$$

corresponding to vertexes $(0,0),\ (1,0),\ (1,1),\ (0,1)$. For the RT_0 element, the element shape function can be written as

$$\begin{pmatrix} a_1+a_2\xi \\ b_1+b_2\eta \end{pmatrix},$$

and the bases are

$$\begin{pmatrix} 1-\xi \\ 0 \end{pmatrix},\ \begin{pmatrix} \xi \\ 0 \end{pmatrix},\ \begin{pmatrix} 0 \\ 1-\eta \end{pmatrix},\ \begin{pmatrix} 0 \\ \eta \end{pmatrix},$$

corresponding to edge centers $(0,\frac{1}{2}),\ (1,\frac{1}{2}),\ (\frac{1}{2},0),\ (\frac{1}{2},1)$.

3.2.2. *Three Dimensional.* Again, take $[0,1]\times[0,1]\times[0,1]$ as the reference element. For the ND_1 element, the element shape function can be written as

$$\begin{pmatrix} a_1+a_2\eta+a_3\zeta+a_4\eta\zeta \\ b_1+b_2\xi+b_3\zeta+b_4\xi\zeta \\ c_1+c_2\xi+c_3\eta+c_4\xi\eta \end{pmatrix},$$

Bases for $H(\mathrm{curl})$ are located on the 12 edge centers, and for example, bases on edges align with x-axis,

$$\begin{pmatrix} \eta\zeta \\ 0 \\ 0 \end{pmatrix},\ \begin{pmatrix} (1-\eta)\zeta \\ 0 \\ 0 \end{pmatrix}\ \begin{pmatrix} \eta(1-\zeta) \\ 0 \\ 0 \end{pmatrix},\ \begin{pmatrix} (1-\eta)(1-\zeta) \\ 0 \\ 0 \end{pmatrix}.$$

For the RT_0 element, the element shape function can be written as

$$\begin{pmatrix} a_1+a_2\xi \\ b_1+b_2\eta \\ c_1+c_2\zeta \end{pmatrix},$$

and the bases are

$$\begin{pmatrix} 1-\xi \\ 0 \\ 0 \end{pmatrix},\ \begin{pmatrix} \xi \\ 0 \\ 0 \end{pmatrix},\ \begin{pmatrix} 0 \\ 1-\eta \\ 0 \end{pmatrix},\ \begin{pmatrix} 0 \\ \eta \\ 0 \end{pmatrix},\ \begin{pmatrix} 0 \\ 0 \\ 1-\zeta \end{pmatrix},\ \begin{pmatrix} 0 \\ 0 \\ \zeta \end{pmatrix},$$

corresponding to face centers $(0, \frac{1}{2}, \frac{1}{2})$, $(1, \frac{1}{2}, \frac{1}{2})$, $(\frac{1}{2}, 0, \frac{1}{2})$, $(\frac{1}{2}, 1, \frac{1}{2})$, $(\frac{1}{2}, \frac{1}{2}, 0)$, $(\frac{1}{2}, \frac{1}{2}, 1)$.

3.3. **Extensions and theoretical issues.** Let us propose several questions in relation to the above derivation.

3.3.1. *Convergence analysis of the MAC scheme.* Nicoladias and Xiaonan Wu wrote some papers years ago on the convergence analysis of MAC scheme. The derivation in the previous section would provide another approach for error analysis.

We should try to carry out the details of such an analysis.

In there paper, they derive the scheme from finite volume method, the new idea can be thought as a finite element approach to give an analysis.

3.3.2. *A new MAC scheme based on triangular Ravriat-Thomas elements.* If we consider a uniform triangulalation and the associated lowest order Raviat Thomas and edge elements for both H(div) and H(curl), we should be able to derive a new class of MAC finite difference scheme for Stokes.

Let see what's the difference for this new class of MAC finite difference scheme for Stokes.

- For each rectangular cell, we have one more dof, say $w_{i,j}$, which is defined at the cell center $(x_{i+\frac{1}{2}}, y_{i+\frac{1}{2}})$, with the meaning of flux on the diagonal edge inside the rectangular cell.
- The stencil for each edge on the triangluar grid has more components (for each edge $e_{i,j}$, the stencil involves edges connected with the two vertexes i and j). Also, the stencil components for $\mathcal{D}^t\mathcal{D}$ will be contained in the stencil components of $\mathcal{C}\mathcal{C}^t$. One question is how could we choose the scaling for $\mathcal{D}^t\mathcal{D}$ to make a convergence algorithm? In the previous section, we simply take $\mathcal{D}^t\mathcal{D}$ and get the five-point stencil.

3.3.3. *Different boundary conditions.* If we do not have pure Dirichlet boundary, then the above procedure may have problems to take care of the boundary conditions (such as Neumann)? How did the CVD people to deal with different boundary conditions for stream-function-voticity-pressure formulation? If this approach only works for pure Dirichet problem, can we say that the MAC scheme may not work for other boundary conditions such as Neumann?

I need to work out a way to implement MAC for Neumann problems. Let's keep on working and see what problems will happen.

4. Distributive Gauss-Seidel Relaxation

4.1. **Distributive Gauss-Seidel Relaxation for Stokes Problem.** Suppose we solve equations

$$\mathcal{L}U = F \tag{4.1}$$

where

$$\mathcal{L} = \begin{pmatrix} -\mu\Delta & \nabla \\ -\nabla\cdot & 0 \end{pmatrix}, \; U = \begin{pmatrix} \boldsymbol{u} \\ p \end{pmatrix}, \; F = \begin{pmatrix} \boldsymbol{f} \\ 0 \end{pmatrix}.$$

For one step of the iterative method is given by

$$U^{k+1} = U^k + \mathcal{B}(F - \mathcal{L}U^k), \tag{4.2}$$

where $\mathcal{B}$ is an approximation of $\mathcal{L}^{-1}$.

In order to construct a good approximation $\mathcal{B}$, we first introduce a distributive matrix

$$\mathcal{M} = \begin{pmatrix} I & \nabla \\ 0 & \mu\Delta_p \end{pmatrix}.$$

Since

$$-\mu\Delta\nabla + \nabla\mu\Delta = (-\mu\Delta + \nabla\mu\nabla\cdot)\nabla = (\mu\nabla \times \nabla\times)\nabla = 0,$$

therefore $\mathcal{L}\mathcal{M} = \begin{pmatrix} -\mu\Delta & 0 \\ -\nabla\cdot & -\Delta_p \end{pmatrix}$ is lower triangluar matrix. Thus

$$\mathcal{L}^{-1} = \mathcal{M}(\mathcal{L}\mathcal{M})^{-1} \tag{4.3}$$

$$= \mathcal{M} \begin{pmatrix} -\mu\Delta & 0 \\ -\nabla\cdot & -\Delta_p \end{pmatrix}^{-1} \tag{4.4}$$

will be a good candidate for $\mathcal{B}$.

Question: given $\boldsymbol{u}^k$ and p^k, how to update it by computing $\delta\boldsymbol{u}^k$ and δp^k ?

We first form the residual $r_{\boldsymbol{u}}$ and r_p as

$$r_{\boldsymbol{u}} = \boldsymbol{f} - \nabla p^k + \mu\Delta\,\boldsymbol{u}^k,\ r_p = \nabla\cdot\boldsymbol{u}^k.$$

Then we update $\delta w_{\boldsymbol{u}}^k$ and δw_p^k by solving

$$\mathcal{L}\mathcal{M} \begin{pmatrix} \delta w_{\boldsymbol{u}}^k \\ \delta w_p^k \end{pmatrix} = \begin{pmatrix} r_{\boldsymbol{u}} \\ r_p \end{pmatrix}, \tag{4.5}$$

finially obtain $\delta\boldsymbol{u}^k$ and δp^k by

$$\begin{pmatrix} \delta\boldsymbol{u}^k \\ \delta p^k \end{pmatrix} = \mathcal{M} \begin{pmatrix} \delta w_{\boldsymbol{u}}^k \\ \delta w_p^k \end{pmatrix} = \begin{pmatrix} \delta w_{\boldsymbol{u}}^k + \nabla\delta w_p^k \\ \mu\Delta_p\,\delta w_p^k \end{pmatrix}.$$

Hence $\boldsymbol{u}^{k+1}$ and p^{k+1} can be updated as

$$\boldsymbol{u}^{k+1} = \boldsymbol{u}^k + \delta w_{\boldsymbol{u}}^k + \nabla\delta w_p^k, \tag{4.6}$$

$$p^{k+1} = p^k + \mu\Delta_p\,\delta w_p^k. \tag{4.7}$$

We now give detail calculation of equation (4.5).

(1) The first equation is $-\mu\Delta\,\delta w_{\boldsymbol{u}}^k = r_{\boldsymbol{u}}$. We define an intermedia velocity

$$\boldsymbol{u}^{k+\frac{1}{2}} = \boldsymbol{u}^k + \delta w_{\boldsymbol{u}}^k.$$

We don't compute $\delta w_{\boldsymbol{u}}^k$ exactly, instead, starting from $\boldsymbol{u}^k$ and p^k, we get $\boldsymbol{u}^{k+\frac{1}{2}}$ approximately by solving

$$-\mu\Delta\,\boldsymbol{u}^{k+\frac{1}{2}} = \boldsymbol{f} - \nabla p^k$$

with one Gauss-Seidel relaxation.

(2) The second equation is $-\Delta\,\delta w_p = r_p + \nabla\cdot\delta w_{\boldsymbol{u}}^k = \nabla\cdot\boldsymbol{u}^{k+\frac{1}{2}}$. Finally, we can update $\boldsymbol{u}^{k+\frac{1}{2}}$ with an approximate solution of the above difference equation and update p^{k+1} by keeping the residual of the momentum equation unchanged.

Even though we introduce operators such as $\mathcal{L}$ and $\mathcal{M}$, we don't use it in the implementation. The algorithm can be summarized as the following:

Algorithm 4.1. Distributive Gauss-Seidel: Given $(\boldsymbol{u}^k, p^k)$

- **Step 1**: Relax momentum equations to get intermedia velocity $\boldsymbol{u}^{k+\frac{1}{2}}$
  Solve momentum equation
  $$-\mu\Delta\,\boldsymbol{u}^{k+\frac{1}{2}} = \boldsymbol{f} - \nabla p^k$$
  approximately by one Gauss-Seidel relaxation.
- **Step 2**: Update velocity cellwisely and pressure patchwisely.
  - **Step 2.1** For each cell τ, project $\boldsymbol{u}^{k+\frac{1}{2}}|_\tau$ on to local divergence free space on τ.
    $$\boldsymbol{u}^{k+1}|_\tau = \boldsymbol{u}^{k+\frac{1}{2}}|_\tau + \nabla\,\widetilde{\delta w_p^k},$$
    where $\widetilde{\delta w_p^k}$ is an approximation of the following difference equation
    $$-\Delta\,\delta w_p^k = \nabla\cdot\boldsymbol{u}^{k+\frac{1}{2}}$$
  - **step 2.2** Correct pressure for the current cell τ and its neighboring cells.
    $$p^{k+1} = p^k - \mu\,\nabla\cdot\boldsymbol{u}^{k+\frac{1}{2}}.$$

4.2. **Another look at the Algorithm in a variational framework.** First step, we solve:

$$-\mu\Delta\,\boldsymbol{u}^{k+\frac{1}{2}} = \boldsymbol{f} - \nabla p^k$$

The second step is try to bring $\boldsymbol{u}^{k+1/2}$ back to divergence free space by a correction in $r_h \in Q_h$

$$\boldsymbol{u}^{k+1} = \boldsymbol{u}^{k+1/2} + \nabla_h r_h. \tag{4.8}$$

We require:

$$(\nabla\cdot\boldsymbol{u}^{k+1}, q_h) = (\nabla\cdot\boldsymbol{u}^{k+1/2}, q_h) - (\nabla_h r_h, \nabla_h q_h) = 0.$$

Thus, we are lead to solve for $r_h \in Q_h$ such that

$$(\nabla_h r_h, \nabla_h q_h) = (\nabla\cdot\boldsymbol{u}^{k+1/2}, q_h), \quad \forall q_h \in Q_h. \tag{4.9}$$

Questions:

(1) In the finite difference setting, ∇_h is easy to compute. In the finite element setting, the evaluation of ∇_h may involve some mass matrix?
(2) Although (4.9) looks like a discrete Poisson equation, we do not seem to need to impose any boundary condition for r_h because ∇_h should be injective operator due to the inf-sup condition.

Due to the definition of ∇_h, $\boldsymbol{u}^{k+1}$ given by the formula (4.8) naturally satisfy the right boundary condition. It is equivalent to finding $\boldsymbol{u}^{k+1} \in V_h$ such that

$$(\boldsymbol{u}^{k+1}, \boldsymbol{v}_h) = (\boldsymbol{u}^{k+1/2}, \boldsymbol{v}_h) - (\nabla\cdot\boldsymbol{v}_h, r_h), \quad \forall \boldsymbol{v}_h \in V_h. \tag{4.10}$$

The question remains how to find p^{k+1}. One natural way to do it is as follows:

$$-\Delta_h\boldsymbol{u}^{k+1} + \nabla_h p^{k+1} = f_h$$

Namely,

$$-\Delta_h(\boldsymbol{u}^{k+1/2} + \nabla_h r_h) + \nabla_h p^{k+1} = f_h$$

and

$$-\Delta_h\nabla_h r_h + \nabla_h(p^{k+1} - p^k) = 0. \tag{4.11}$$

The question is: can we solve for p^{k+1} from the above equation? How about this:

$$-(\Delta_h \nabla_h r_h, q_h) + (\nabla_h (p^{k+1} - p^k), q_h) = 0.$$

or

$$(\nabla_h p^{k+1}, \nabla_h q_h) = (\nabla_h p^k, \nabla_h q_h) - a(\nabla_h r_h, \nabla_h q_h), \quad \forall q_h \in Q_h.$$

This looks like that we can solve p^{k+1} as long as we can evaluate ∇_h easily:

- How about using mass lumping to evaluate ∇_h?

Brandt's approach is based on the following sequence of approximations:

$$-\Delta_h \nabla_h r_h \approx -\nabla_h \nabla \cdot \nabla_h r_h = \nabla_h \nabla \cdot \boldsymbol{u}^{k+1/2}. \tag{4.12}$$

This leads to the defining formula for

$$\nabla_h (p^{k+1} - p^k + \nabla \cdot \boldsymbol{u}^{k+1/2}) = 0.$$

Namely

$$p^{k+1} = p^k - \nabla \cdot \boldsymbol{u}^{k+1/2}.$$

Or equivalently,

$$a(\boldsymbol{u}^{k+1}, \boldsymbol{v}_h) + (\nabla_h p^{k+1}, \boldsymbol{v}_h) = (\boldsymbol{f}_h, \boldsymbol{v}_h).$$

or

$$a(\boldsymbol{u}^{k+1/2} + \nabla_h r_h, v_h) + (\nabla_h p^{k+1}, \boldsymbol{v}_h) = (\boldsymbol{f}_h, \boldsymbol{v}_h).$$

But

$$a(\boldsymbol{u}^{k+1/2}, \boldsymbol{v}_h) + (\nabla_h p^k, \boldsymbol{v}_h) = (\boldsymbol{f}_h, \boldsymbol{v}_h).$$

We have

$$a(\nabla_h r_h, \boldsymbol{v}_h) + (\nabla_h (p^{k+1} - p^k), \boldsymbol{v}_h) = 0.$$

4.3. **On the mixed methods based on H(div) and H(curl).** . By (3.4), we have

$$\omega_h = \mathrm{curl}_h u_h.$$

We then have

$$\begin{cases} -(\Delta_h u_h, \boldsymbol{v}_h) - (p_h, \mathrm{div}\, \boldsymbol{v}_h) = (\boldsymbol{f}, \boldsymbol{v}_h), & \text{for } \boldsymbol{v}_h \in \boldsymbol{U}_h, \\ -(\mathrm{div}\, \boldsymbol{u}_h, q_h) = 0, & \text{for } q_h \in Q_h. \end{cases} \tag{4.13}$$

Here:

$$-\Delta_h = \nabla_h \nabla \cdot + \mathrm{curl}\ \mathrm{curl}_h$$

and

$$\Delta_h \nabla_h r_h = (\nabla_h \nabla \cdot + \mathrm{curl}\ \mathrm{curl}_h) \nabla_h r_h = \nabla_h \nabla \cdot \nabla_h r_h$$

where the last identity holds since

$$\mathrm{curl}_h \nabla_h = \mathrm{curl}^* (-\nabla \cdot)^* = -(\nabla \cdot \mathrm{curl})^* = 0.$$

Conclusion:

> Algorithm 4.1 converges in one iteration for the scheme (4.13) with any triple discrete spaces (of any order) for H(curl), H(div) and L^2! Namely the solution of (4.13) is reduced to solve exactly 3 discrete Laplacians!

The problem is that we need to be able to compute $rmcurl_h$ locally for this algorithm to be fesible. We can accomplish this by mass-lumping, which will degrade the accuracy of the scheme. But we can also use the mass-lumped scheme for constructing preconditioner.

4.4. **Distributive Gauss-Seidel for MAC discretization of Stokes equation.** We can write MAC discretization of (2.1)-(2.3) in the following Matrix form:

$$\begin{pmatrix} A & B^t \\ B & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{u} \\ p \end{pmatrix} = \begin{pmatrix} \boldsymbol{f} \\ 0 \end{pmatrix} \tag{4.14}$$

Where A approximate $-\Delta$, B approximates $-\operatorname{div}$, and B^t approximate grad. Denote

$$L = \begin{pmatrix} A & B^t \\ B & 0 \end{pmatrix}, \; M = \begin{pmatrix} I & B^t \\ 0 & X \end{pmatrix}$$

then

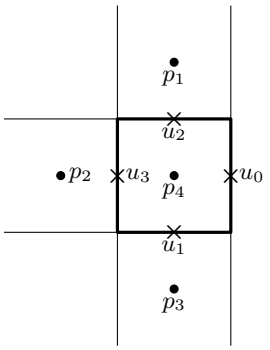
$$LM = \begin{pmatrix} A & AB^t + B^t X \\ B & BB^t \end{pmatrix}$$

4.4.1. *Choice of $X = -BB^t$:* Check $(A - B^tB)B^t$ for MAC scheme Clearly, $B\mathcal{C} = 0$, which is a discrete version of $\operatorname{div}\operatorname{curl} = 0$. Therefore

$$(A - B^tB)B^t = \mathcal{C}S^{-1}\mathcal{C}^tB^t = 0$$

for interior points, with $S = diag(h^2)$.

4.4.2. *How to compute BB^tp without computing the whole matrix?* When we update the pressure equations, we need to know the stencil used for pressure on the boundary and corner cells. Take a boundary case as an example.



$$BB^tp = \frac{u_0 - u_3}{h} + \frac{u_2 - u_1}{h} = \frac{-\frac{p_4 - p_2}{h}}{h} + \frac{\frac{p_1 - p_4}{h} - \frac{p_4 - p_3}{h}}{h} = \frac{-3p_4 + p_1 + p_2 + p_3}{h^2}.$$

What is the idea behind this? What will the stencil be for the case of $u_0 \neq 0$?

4.4.3. *Exact update formular for MAC scheme when applied with DGS method.* For MAC discretization of Stokes equation, How to get formular for updating $\vec{\boldsymbol{u}}$ and p ?

We proceed one cell at a time. Recall Algorithm 4.1 that δw_p^k satisfy

$$-\Delta\, \delta w_p^k = r_c \tag{4.15}$$

and

$$u^{k+1} = u^{k+\frac{1}{2}} - \partial_x \delta w_p^k.$$

(1) The first step is to update velocity.

Equation (4.15) can't be solved exactly. we can approximate $\partial_x \delta w_p^k$ Locally. Let's consider calculation of $(\partial_x \delta w_p^k)^{i,j}$. From the five point difference scheme for δw_p^k, we have

$$\frac{4(\delta w_p^k)^{i,j} - (\delta w_p^k)^{i-1,j} - (\delta w_p^k)^{i+1,j} - (\delta w_p^k)^{i,j-1} - (\delta w_p^k)^{i,j+1}}{h^2} = r_c.$$

Thus we can get

$$(\delta w_p^k)^{i,j} = \frac{h^2 r_c}{4} + \frac{1}{4}[(\delta w_p^k)^{i-1,j} + (\delta w_p^k)^{i+1,j} + (\delta w_p^k)^{i,j-1} + (\delta w_p^k)^{i,j+1}].$$

Since we are locally solving the above equation, we can assume the values of $(\delta w_p^k)^{i-1,j}, (\delta w_p^k)^{i+1,j}, (\delta w_p^k)^{i,j-1}, (\delta w_p^k)^{i,j+1}$ are the same, then we have

$$(\partial_x \delta w_p)^{i,j} = -\frac{h r_c}{4}.$$

Similarly, we can get

$$(\partial_x \delta w_p)^{i+1,j} = \frac{h r_c}{4},\ (\partial_y \delta w_p)^{i,j} = -\frac{h r_c}{4},\ (\partial_y \delta w_p)^{i,j+1} = \frac{h r_c}{4}.$$

(2) The next step is to update pressure.

The motivation is to keep the residual of momentum equation unchanged.

The value for updating pressure has been tagged on Figure 7. We only need to verify that the momentum equation have been balanced at all the red and blue points after updating pressure in the center cell. Take the red points as an example. The red points can be classified in to two categories. The first category is the points at $(i, j-1), (i, j+1), (i+1, j-1), (i+1, j+1)$. The second category is the points at $(i, j), (i+1, j)$. The momentum equation at $(i, j+\frac{1}{2})$ is:

$$\frac{4u^{i,j} - u^{i-1,j} - u^{i+1,j} - u^{i,j-1} - u^{i,j+1}}{h^2} + \frac{p^{i,j} - p^{i-1,j}}{h} = f_1^{i,j}$$

After updating velocity and pressure, the equation is written as:

$$\frac{4(u^{i,j} - \delta) - u^{i-1,j} - (u^{i+1,j} + \delta) - u^{i,j-1} - u^{i,j+1}}{h^2} + \tag{4.16}$$

$$\frac{(p^{i,j} + \frac{4\delta}{h}) - (p^{i-1,j} - \frac{\delta}{h})}{h} = f_1^{i,j} \tag{4.17}$$

The momentum equation can also be checked for the first category points in the same way.

Using the same way, we can give the update of velocity and pressure on the boundary and corner points. At the boundary cells and corner cells, the stencil for pressure p are $(3, -1, -1, -1)$ and $(2, -1, 1)$, respectively. So the quntity of udpate velocity should be changed to $\frac{h r_c}{3}$ and $\frac{h r_c}{2}$ accordingly.

4.5. **Multigrid.**

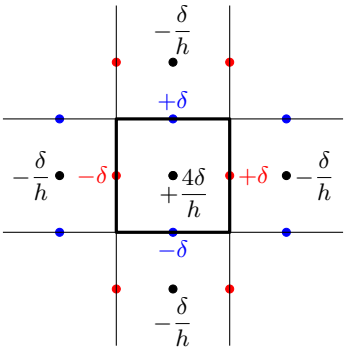
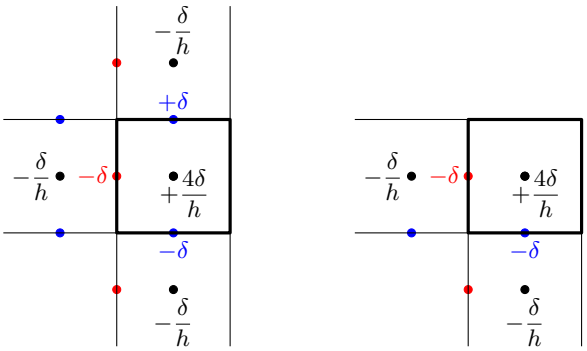


FIGURE 7. Update stencil for u, v, p at interior cells with $\delta = -\frac{hr_c}{4}$.



(A) Boundary cell **(B)** Corner cell

FIGURE 8. Update stencil for u, v, p at boundary cells with $\delta = -\frac{hr_c}{3}$, and corner cells with $\delta = -\frac{hr_c}{2}$.

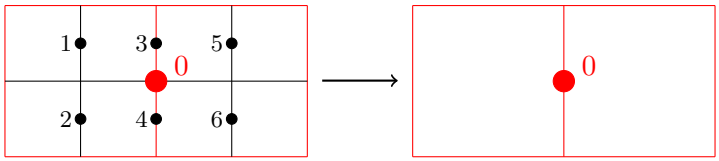
4.5.1. *Prolongation and restriction.* At u- and v-grid points, we consider six point restrictions, and at p-grid points, a four-point cell-centered restriction. In stencil notation, the restriction operators are

$$R^u_{h,2h} = \frac{1}{8}\begin{pmatrix} 1 & 2 & 1 \\ & * & \\ 1 & 2 & 1 \end{pmatrix}, \; R^v_{h,2h} = \frac{1}{8}\begin{pmatrix} 1 & & 1 \\ 2 & * & 2 \\ 1 & & 1 \end{pmatrix}, \; R^p_{h,2h} = \frac{1}{4}\begin{pmatrix} 1 & & 1 \\ & * & \\ 1 & & 1 \end{pmatrix}.$$

Let's explan what's mean of the restriction operator $R^u_{h,2h}$. Look at Figure 9 for a vertical edge center degree of freedom. Point 0 are degree of freedom for u on coarse grid, and points $1, \cdots, 6$ are degree of freedoms on the fine grid. Therefor the restriction at point 0 will be

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

The explaination for the restriction operators $R^v_{h,2h}$ and $R^p_{h,2h}$ are the same. $R^u_{h,2h}$ and $R^v_{h,2h}$ are only defined for interior edges, since the dof is prescribed via Dirichlet boundary conditions for boundary edges.



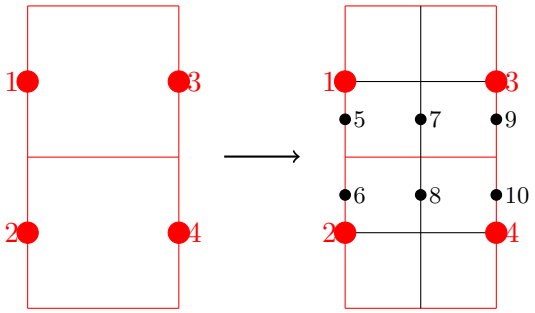
(A) Fine grid **(B)** Coarse grid

FIGURE 9. 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.

For the prolongation operators, we typically apply bilinear interpolation of neighboring coarse-grid unknowns in the staggered grid. See Figure 10. To compute the values at points $5, \cdots, 10$, we first calculate a bilinear function using coarse grid points $1, \cdots, 4$, and the value for fine grid points $5, \cdots, 10$ are just the evaluation of the bilinear function at these points. One can eaily derive that:

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

Following the same the approach, one can also derive formular for computing horizontal edge center degree of freedoms on fine grid from coarse grid solution. Piecewise constant (first-order) interpolation for the p variables.



(A) Coarse grid **(B)** Fine grid

FIGURE 10. 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.

4.5.2. *$\mathcal{V}$-cycle.* A sketch of matlab routine for $\mathcal{V}$-cycle. This code can be written in matrix free form, i.e, we don't have to write the whole matrix of MAC discretization. One can treat solution u, v, p and right hand side f_1, f_2 as matrix.

```
[ru0,rv0,rp0]=formResMAC(f1,f2,u,v,p);
r0 = max([max(abs(ru0)),max(abs(rv0)),max(abs(rp0))]); % initial residual.
%% V-cycle
k = 1; tol = 1e-6; err = 1;
while (err > tol) && (k<=maxIt)
    % form residual
    [ru{level},rv{level},rp{level}] = formRes(f1,f2,u,v,p);
    % presmoothing
    for i = level:-1:3
        [eu{i},ev{i},ep{i}] = StokesDGS(0,0,0,nu,ru{i},rv{i},rp{i},smoothStep);
        [tempru,temprv,temprp] = formRes(ru{i},rv{i},rp{i},eu{i},ev{i},ep{i});
        [ru{i-1},rv{i-1},rp{i-1}] = restrict(tempru,temprv,temprp);
    end
    % solve on coarse grid
    [eu{2},ev{2},ep{2}] = StokesMACDGS(0,0,0,nu,ru{2},rv{2},20,rp{2});
    for i = 3:level
        [tempu,tempv,tempp] = prolongation(eu{i-1},ev{i-1},ep{i-1});
        eu{i} = eu{i} + tempu;
        ev{i} = ev{i} + tempv;
        ep{i} = ep{i} + tempp;
        % post-smoothing
        [eu{i},ev{i},ep{i}] = StokesDGS(eu{i},ev{i},ep{i},ru{i},rv{i},rp{i},smoothStep);
    end
    u = u + eu{level};
    v = v + ev{level};
    p = p + ep{level};
    err = max([max(abs(ru{level})),max(abs(rv{level})),max(abs(rp{level}))])/r0;
    fprintf('V-cycle for MAC,iteration = %2.0u, err = %12.8g \n', k, err);
    k = k + 1;
end
```

Remark 4.1. Here are some suggestions for implementation. You must be make sure each component of you subroutine is correct.

(1) Make sure DGS smoothing subroutine works

Before you do $\mathcal{V}$-cycle, be sure you have already had a DGS smoothing subroutine (e.g., in matlab systle, one may name this subroutine as StokesDGS.m) for Stokes equations in hand. You also need to make sure that you have succeed in implementing the DGS smoothing subroutine. That means you should be able to get the correct numerical solution (second order for both velocity and pressure in maximum norm.) when you simply use DGS smoothing subroutine as a solver with thousands of iteration steps.

(2) Make sure prolongation and form residual subroutine (one may name it as prolongation.m and formRes.m) works.

You can code a two-level subroutine first. Solve the Stokes equations on coarse grid with your StokesDGS.m subroutine (with thousands of iterations), and prolongate the solution on coarse grid to get a fine grid solution. Then solve the residual equation on fine grid with your StokesDGS.m (with thousands of iterations), and add the error you obtained back to the solution on fine grid. Check if you have get the correct numerical solution.

(3) For the coarse grid solver, you can simply use you StokesDGS.m subroutine. 10-20 steps iterations should be able to give a better solution on the coarse grid (e.g., 4 by 4 coarse grid.).
(4) Be careful when you compute the error for pressure. you need to normalize or presribed a value for pressure. To be precise, before you compute error for pressure, you need to modify the pressure, e.g, at the last pressure cell by adding a constant in the following way:

$$p = p - p(end) + exactp(end)$$

5. Numerical Results

5.1. **Multigrid $\mathcal{V}$-cycle.** We consider Stokes problem (1.1) on unit square with the exact solution

$$\boldsymbol{u} = \begin{pmatrix} (1-\cos(2\pi x))\sin(2\pi y) \\ -(1-\cos(2\pi y))\sin(2\pi x) \end{pmatrix}, \; p = \frac{1}{3}x^3$$

We present the iteration steps for $\mathcal{V}$-cycle and $\mathcal{W}$-cycle with DGS as smoothing. We use 3 steps of pre-smoothing and post-smoothing for $\mathcal{V}$-cycle, and 2 steps of pre-smoothing and post-smoothing for $\mathcal{W}$-cycle. level in the table denotes the levels used in $\mathcal{V}$-cycle. The implementation of $\mathcal{W}$-cycle is a two-grid method with $\mathcal{V}$-cycle as coarse grid solver. The tolerance to exit cycles is that the relative residual less than $1e$-6.

(A) $\mathcal{V}$-cycle with 3 steps DGS smoothing

# level	h	# DOF	iterStep	time (s)
6	$\frac{1}{64}$	12160	8	0.188
7	$\frac{1}{128}$	48896	8	0.652
8	$\frac{1}{256}$	196096	9	2.89
9	$\frac{1}{512}$	785408	9	12.70

(B) $\mathcal{W}$-cycle with 2 steps DGS smoothing

# level	h	# DOF	iterStep	time (s)
6	$\frac{1}{64}$	12160	6	0.129
7	$\frac{1}{128}$	48896	6	0.406
8	$\frac{1}{256}$	196096	6	1.57
9	$\frac{1}{512}$	785408	7	8.05

Table 1. Iteration steps for $\mathcal{V}$-cycle and $\mathcal{W}$-cycle, and CPU cost

5.2. **GMRES with V-cycle as preconditioning.** Inside $\mathcal{V}$-cycle, 2 steps of pre-smoothing and post-smoothing are used.

# level	h	# DOF	iterStep	time (s)
6	$\frac{1}{64}$	12160	6	0.245
7	$\frac{1}{128}$	48896	6	0.518
8	$\frac{1}{256}$	196096	7	2.466
9	$\frac{1}{512}$	785408	7	10.883

TABLE 2. Iteration steps and CPU cost for GMRES with $\mathcal{V}$-cycle as preconditioning

6. CAN WE EXTEND DGS METHOD TO SOLVE THE EQUATIONS DISCRETIZED FROM MIXED FEM FOR STOKES EQUATION?

7. WHAT'S THE RELATIONSHIP BETWEEN YEE'S SCHEME AND MAC SCHEME?

Two questions:

(1) Can we design MAC-like scheme for Maxwell equation on uniform grid?
(2) Can DGS apply to solve the positive definite Maxwell equation?

$$\nabla \times \nabla \times u = f \tag{7.1}$$

$$\nabla \cdot u = 0 \tag{7.2}$$