PROGRAMMING OF MAC SCHEME FOR STOKES EQUATIONS

LONG CHEN

In this notes, we explain the implementation detail of MAC, the most popular finite difference method, for the Stokes equations

$$\begin{cases} -\mu\Delta \boldsymbol{u} + \nabla p = \boldsymbol{f} & \text{in } \Omega, \\ -\nabla \cdot \boldsymbol{u} = 0 & \text{in } \Omega. \end{cases} \tag{1}$$

1. MAC DISCRETIZATION

We use two dimensional uniform grids for $\Omega = (0,1)^2$ as a typical setting. The generalization to domains composed by rectangles and to three dimensional domains composed by cubes is straightforward but with extra notation.

Let $\boldsymbol{u} = (u, v)$ and $\boldsymbol{f} = (f_1, f_2)$. We rewrite the Stokes equations into coordinate-wise

$$-\Delta u + \partial_x p = f_1, \tag{2}$$

$$-\Delta v + \partial_y p = f_2, \tag{3}$$

$$\partial_x u + \partial_y v = 0. \tag{4}$$

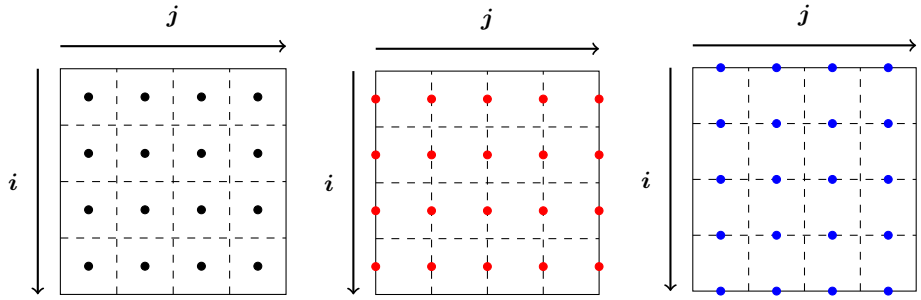
The domain $\Omega = (0,1)^2$ is decomposed into small squares with size h. Standard central difference discretization of Δ and ∂_x at vertices will not give a stable discretization of Stokes equations due to the failure of discrete inf-sup condition. To see this, one can view the 5-point stencil as a numerical quadrature of using Q_1 element for Laplacian operator and thus discretization at grid points is equivalent to use $Q_1 - Q_1$ unstable pair. Similarly changing pressure discretization to centers of cells corresponds to $Q_1 - P_0$ which is also unstable.

The idea of MAC, Marker and Cell, is to place the unknown of (u, v, p) in different locations. Specifically the pressure p is located in the center of each cell and the x-component velocity u on the middle points of vertical edges (red dots) and the y-component velocity v on middle points of horizontal edges; see the following figure.

The MAC scheme is to discretize the x-coordinate momentum equation (2) at vertical edges, the y-coordinate momentum equation (3) at horizontal edges, and the continuity equation (4) at cell centers using central difference schemes.

Let us use the geometry consistent matrix system to store the variables. Let n be the number of cells in one direction. In the geometry consistent matrix system, i.e., `meshgrid` system, the subscript i is the row index and j is the column index, running from $1:n$ or $1:n+1$. The pressure is then discretized to a matrix `p(1:n,1:n)`, and the velocity is `u(1:n,1:n+1)`, and `v(1:n+1,1:n)`. One can easily write out the mapping from the index (i,j) to the coordinate (x_j, y_i) for different variables. Note that it is not consistent with the coordinate consistent system where i corresponds to x_i and j to y_j. The argument to use the geometry consistent system is that once the mapping ((i,j) to (x_j, y_i)) is fixed, in almost all places of coding, we operate in the algebraic level and such indices system is

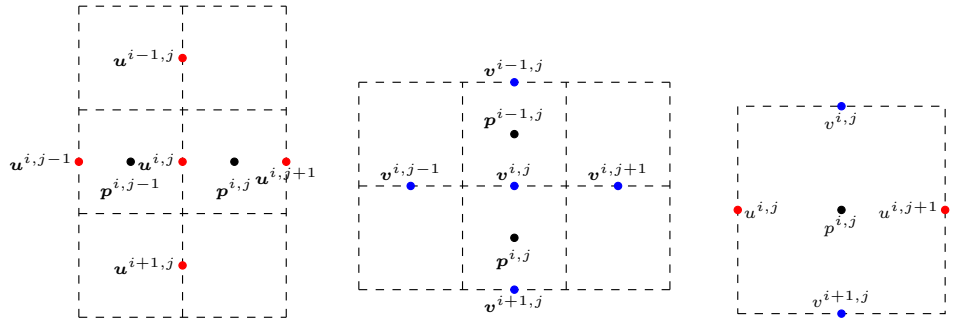
Date: March 13, 2014.



(a) p (b) u (c) v

FIGURE 1. Location and indices of (u, v, p) variables.

much intuitive to traverse in the matrix. The readers are encouraged to read Chapter: Programming of Finite Difference Methods for detailed discussion of the `meshgrid` system.



(A) equation for u **(B)** equation for v **(C)** equation for p

FIGURE 2. Indices of MAC equations

For interior nodes, the MAC scheme can be written as

```
(4*u(i,j) - u(i-1,j) - u(i+1,j) - u(i,j-1) - u(i,j+1))/h^2 ...
+ (p(i,j) - p(i,j-1))/h = f1(i,j);
(4*v(i,j) - v(i-1,j) - v(i+1,j) - v(i,j-1) - v(i,j+1))/h^2 ...
+ (p(i-1,j) - p(i,j))/h = f2(i,j);
(u(i,j+1) - u(i,j))/h + (v(i,j) - v(i+1,j))/h = 0.
```

Since central difference schemes are used, it is easy to see that the above scheme has second order truncation error for interior nodes.

We then discuss the discretization of boundary conditions. For Dirichlet type boundary condition, one can impose it in one direction by fixing the value laying on the boundary and by extrapolation on the other direction. Let us take unknown u as an example. On edges $x = 0$ and $x = 1$, the value is given by the boundary condition and no equation is discretized on these points. On edges $y = 0$ and $y = 1$, however, there is no unknowns of u on that edge and we need to modify the stencil at $y = h/2, 1 - h/2$. As an example, consider the discretization at the node $(1, j)$ near the top of the boundary. We introduce the ghost value at $y = 1+h/2$ which is artificially recorded in `u(0,j)`. Then we can discretize

the momentum equation at $(1, j)$ using the ghost point. The value at `u(0,j)` can be eliminated by the linear extrapolation, i.e, requiring `(u(0,j) + u(1,j))/2 = g_D(xj,1)`. Therefore the modified discretization for `u` is

```
1       5*u(1,j) - u(2,j) - u(1,j-1) - u(1,j+1) + h*(p(1,j) - p(1,j-1)) ...
2     = h^2*f1(1,j) + 2*gD;
```

The stencil for u-unknowns near the horizontal boundaries will be denoted by the short notation $(5, -1, -1, -1, -2)$.

We can also modify the discretization by quadratic extrapolation for the Dirichlet type boundary condition, that is, use $u^{1/2,j}, u^{1,j}, u^{2,j}$ to fit a quadratic function and evaluate at $u^{0,j}$, we get $u^{0,j} = -2u^{1,j} + \frac{1}{3}u^{2,j} + \frac{8}{3}u^{1/2,j}$, and obtain the modified boundary scheme should be:

```
1   (6*u(1,j) - 4/3*u(2,j) - u(1,j-1) - u(1,j+1)) + h*(p(1,j) - p(1,j-1)) ...
2  = h^2*f1(1,j) + 8/3*gD;
```

This stencil is denoted by $(6, -\frac{4}{3}, -1, -1, -\frac{8}{3})$. The quadratic extrapolation will lead to a better rate of convergence and the detailed error analysis can be found in my recent paper.

For Neumann boundary condition $\partial u/\partial n|_{\partial\Omega} = g_N$, the ghost value will be eliminated by the central difference discretization $(u^{0,j} - u^{1,j})/h = g_N(x_j, 1)$ and the modified stencil is

```
1    (3*u(1,j) - u(2,j) - u(1,j-1) - u(1,j+1)) + h*(p(1,j) - p(1,j-1)) ...
2   = h^2*f1(1,j) + h*gN;
```

Unlike the Dirichlet boundary condition, similar modification is needed for all grids points on or near the boundary edges and for points near corners two ghost degree of freedom (dof) should be introduced; see the lecture notes on finite difference method.

The mappings from the index (i, j) to the coordinate (x_j, y_i) for u, v, p are bettered coded as subroutines and will be needed only in the evaluation of boundary conditions and the right hand side.

2. Implementation of DGS for MAC

We first recall the DGS derived in Chapter: Multigrid Methods for Stokes Equations.

Algorithm 2.1. $[\boldsymbol{u}^{k+1}, p^{k+1}] \leftarrow \mathtt{DGS}(\boldsymbol{u}^k, p^k)$

(1) Relax momentum equations

$$\boldsymbol{u}^{k+\frac{1}{2}} = \boldsymbol{u}^k + \hat{A}^{-1}(\boldsymbol{f} - A\boldsymbol{u}^k - B^T p^k),$$

(2) Relax transformed continuity equations

$$\delta q = \hat{A}_p^{-1}(0 - B\boldsymbol{u}^{k+\frac{1}{2}}).$$

(3) Transform the correction back to the original variables

$$\boldsymbol{u}^{k+1} = \boldsymbol{u}^{k+\frac{1}{2}} + B^T\delta q,$$
$$p^{k+1} = p^k - BB^T\delta q.$$

In the DGS iteration, only algebraic manipulation not the geometric realization of the indices is needed. The neighboring indices of (i, j) for u, v, p can be found in Fig 2 in the previous section.

When implement the Gauss-Seidel iteration for finite difference methods, the direct update version is more efficient than the correction version. In the correction version, we need to compute the residual and then compute the inverse of the lower or upper triangular matrix. These two can be merged and some calculation get canceled. Then the update for velocity `u(i,j)` for `i=2:n-1,j=2:n` is simply

```
u(i,j) = (h^2*f1(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 boundary and near boundary subscript `i,j`, the stencil should be modified to impose the boundary condition. Red-black ordering can be used to vectorize the Gauss-Seidel iteration. See Chapter: Programming of Finite Difference Methods.

After the update of velocity of the momentum equation, we need to update the residual for the continuity equation, i.e. compute $r_p = \operatorname{div} u$ which can be done efficiently using the index relation of `p` and `u,v`; see Fig. 2 (C). Then we apply Gauss-Seidel to solve the elliptic equation $A_p \delta q = r_p$. Here A_p is the discrete Laplacian of pressure with Neumann boundary condition, i.e., $(4,-1,-1,-1,-1)$ for interior cells and $(3,-1,-1,-1)$ for boundary cells and $(2,-1,-1)$ for corners. One Gauss-Seidel iteration for interior cells will be

```
dq(i,j) = (h^2*rp(i,j) + dq(i-1,j) + dq(i+1,j) + dq(i,j-1) + dq(i,j+1))/4;
```

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

Then we should bring the correction `dq` to `u, v, p` through the distributive matrix. Note that B^T is just the discrete gradient. The update of u will be

```
u(i,j) = u(i,j) - (dq(i,j) - dq(i,j-1))/h;
```

The update of pressure requires the computation of $A_p \delta q$. Its matrix-free version is

```
p(i,j) = p(i,j) - (4*dq(i,j)-dq(i-1,j)-dq(i+1,j)-dq(i,j-1)-dq(i,j+1))/h^2;
```

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

3. Transfer operators

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

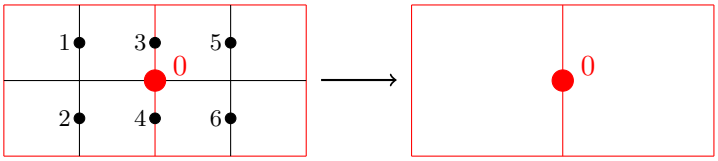
$$R^u_{h,2h} = \frac{1}{8}\begin{pmatrix} 1 & 2 & 1 \\ & * & \\ 1 & 2 & 1 \end{pmatrix}, \quad R^v_{h,2h} = \frac{1}{8}\begin{pmatrix} 1 & & 1 \\ 2 & * & 2 \\ 1 & & 1 \end{pmatrix}, \quad R^p_{h,2h} = \frac{1}{4}\begin{pmatrix} 1 & & 1 \\ & * & \\ 1 & & 1 \end{pmatrix}.$$

Let's explan what's mean of the restriction operator $R^u_{h,2h}$. Look at Figure 3 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.

For the prolongation operators, we typically apply bilinear interpolation of neighboring coarse-grid unknowns in the staggered grid. See Figure 4. To compute the values at points $5, \cdots, 10$, we first calculate a bilinear function using coarse grid points $1, \cdots, 4$, and the



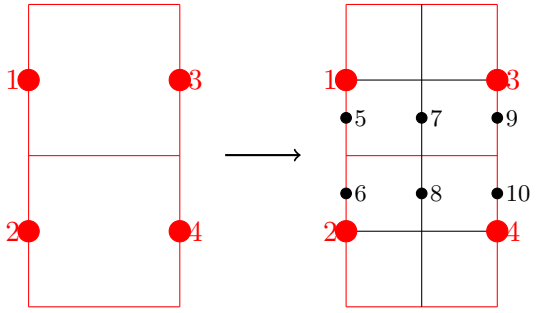
(A) Fine grid **(B)** Coarse grid

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.

value for fine grid points $5, \cdots, 10$ are just the evaluation of the bilinear function at these points. One can easily 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 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.

ACKNOWLEDGMENT

Thanks for Dr. Ming Wang for the help on the preparation of the tex file, especially those nice figures.