Iterative methods for solution of algebraic systems

Discretization of the Navier-Stokes equations results to a set of algebraic equations which can be written in a general form as

$$L\,U = b \qquad \begin{cases} L: \text{ discretized operator} \\ b: \text{ forcing + B.C.} \\ U: \text{ vector of unknowns at all nodes} \end{cases}$$

Usually number of unknowns, and consequently L, is too large to solve the equation by means of direct methods. Therefore, iterative methods are used and the equation above is recasted as

$$U^{k+1} = f(U^k),$$

where k denotes the iteration step. Below we describe different iterative methods in a general form.

Let us decompose operator L as $L = P + A$ and rewrite the original equation in the form of an iterative algorithm as

$$P\,U^{k+1} = b - A\,U^k.$$

This can be rearranged as

$$P\,(U^{k+1} - U^k) = b - L\,U^k$$

Alternative way of writing:

$$P\,\Delta U^k = -R^k \quad \text{where} \quad \begin{cases} \Delta U^k = U^{k+1} - U^k \\ \\ R^k = L\,U^k - b \quad \text{(Residual)} \end{cases}$$

Iterations are performed until the residual R^k has decayed some order of magnitude. Different methods correspond to different choice of P and A. While the iterative methods are simple to derive, implement, and analyze, convergence is only guaranteed for a limited choice of matrices A and P.

Convergence properties of iterative methods:

Let $\overline{U}$ be the exact solution of $LU = b$.

$$\Rightarrow L\overline{U} \equiv (P + A)\overline{U} = b \Rightarrow P\overline{U} = b - A\overline{U}$$

Subtraction $PU^{k+1} = b - AU^k$ gives

$$\Rightarrow \; P\left(\overline{U} - U^{k+1}\right) = -A\left(\overline{U} - U^k\right).$$

Let us define the error at iteration step no. k as $e^k \equiv \overline{U} - U^k$

$$\Rightarrow \; Pe^{k+1} = -Ae^k$$

$$\Rightarrow \; e^{k+1} = -P^{-1}Ae^k = \{A = L - P\} = \left(I - P^{-1}L\right)e^k \equiv Ge^k$$

where $G = (I - P^{-1}L)$ is the iteration matrix.

One can easily observe that

$$e^{k+1} = Ge^k = GGe^{k-1} = \cdots = G^{k+1}e^0$$

Note that we can write $Ge^k = \lambda_j e^k$, where λ is the eigenvalue of matrix G, which means that for a decaying error it is necessary that

$$\rho(G) < 1,$$

where $\rho(G) = \max\{|\lambda_1|, \ldots, |\lambda_n|\}$ is the spectral radius of G.

Jacobi method:

Consider the eq. system $LU = b$ and decompose L as

$$L = L_D + L_L + L_U$$

where

$$L_D = \begin{bmatrix} L_{1,1} & & & \\ & \ddots & & 0 \\ & & L_{i,i} & \\ & 0 & & \ddots \\ & & & L_{N,N} \end{bmatrix}, L_L = \begin{bmatrix} 0 & & & \\ L_{2,1} & \ddots & & 0 \\ \vdots & & 0 & \\ \vdots & & & \ddots \\ L_{N,1} & \cdots & L_{N,N-1} & 0 \end{bmatrix}$$

$$L_U = \begin{bmatrix} 0 & L_{1,2} & \cdots & \cdots & L_{1,N} \\ & \ddots & & & \vdots \\ & & 0 & & \vdots \\ & 0 & & \ddots & L_{N-1,N} \\ & & & & 0 \end{bmatrix}$$

Jacobi method can be formulated as

$$L_D U^{k+1} = b - (L_L + L_U)U^k$$

or

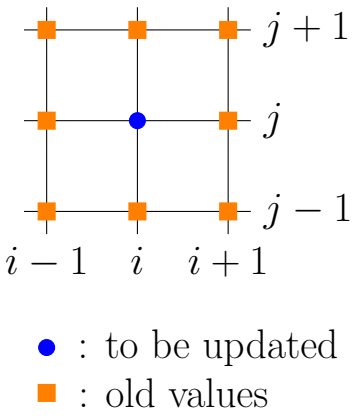
$$L_D \Delta U^k = -R^k$$

where $P = L_D$ and $A = L_L + L_U$.

This can be written as

$$U_i^{k+1} = \frac{1}{L_{i,i}} \left(b_i - \sum_{j \neq i} L_{i,j} U_j^k \right)$$

It means that at each iteration step we use only values from the previous step.



Gauss-Seidel method:

Gauss-Seidel iteration can be formulated as

$$(L_D + L_L)U^{k+1} = b - L_U U^k$$

or

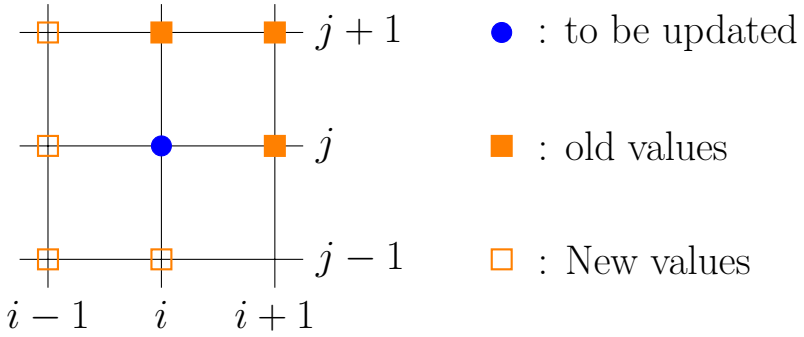
$$(L_D + L_L)\Delta U^k = -R^k$$

Here $P = L_D + L_L$ and $A = L_U$.

This can be written as

$$U_i^{k+1} = \frac{1}{L_{i,i}} \left(b_i - \sum_{j>i} L_{i,j} U_j^k - \sum_{j<i} L_{i,j} U_j^{k+1} \right)$$

It means that at each iteration step we use values from previous step and those already updated at current step.



Guass-Seidel method has a faster convergence rate than Jacobi method:

$$\left|\lambda^{GS}\right| \leq \left|\lambda^{J}\right|$$

If L is a tridiagonal matrix

$$\left|\lambda^{GS}\right| = \left|\lambda^{J}\right|^2$$

meaning that Gauss-Seidel is two times faster than Jacobi method for this case. Further, Gauss-Seidel scheme requires less memory

since there is no need to save the data at two successive iteration steps as the old data are replaced by new ones as soon as they are computed.

Example: Laplace eq. on cartesian grid

The Laplace equation on a Cartesian grids can be discretized as

$$\frac{\Phi_{i+1,j} - 2\phi_{i,j} + \Phi_{i-1,j}}{\Delta x^2} + \frac{\Phi_{i,j+1} - 2\Phi_{i,j} + \Phi_{i,j-1}}{\Delta y^2} = 0$$

We can solve for $\Phi_{i,j}$,

$$\Phi_{i,j} = \frac{\Phi_{i+1,j} + \Phi_{i-1,j} + \beta^2 \left(\Phi_{i,j+1} + \Phi_{i,j-1}\right)}{2\left(1 + \beta^2\right)}$$

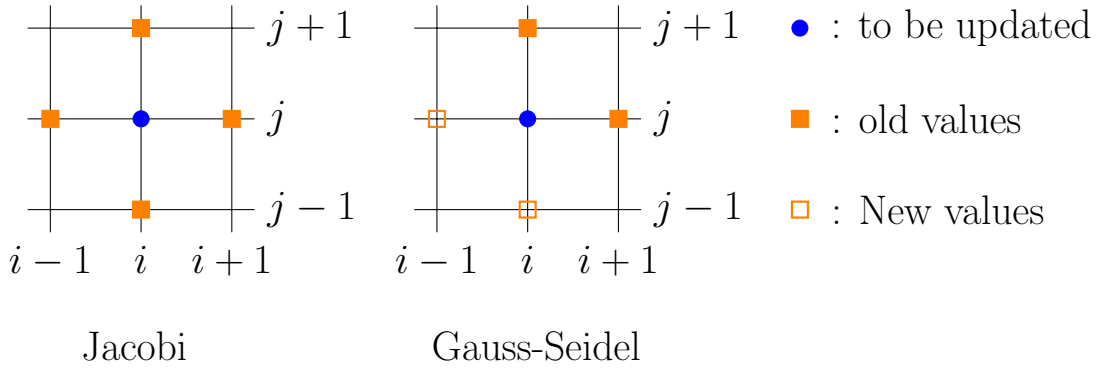
where $\beta = \Delta x / \Delta y$.

Jacobi method gives

$$\Phi_{i,j}^{k+1} = \frac{\Phi_{i+1,j}^{k} + \Phi_{i-1,j}^{k} + \beta^2 \left(\Phi_{i,j+1}^{k} + \Phi_{i,j-1}^{k}\right)}{2\left(1 + \beta^2\right)}$$

Gauss-Seidel method gives

$$\Phi_{i,j}^{k+1} = \frac{\Phi_{i+1,j}^{k} + \Phi_{i-1,j}^{k+1} + \beta^2 \left(\Phi_{i,j+1}^{k} + \Phi_{i,j-1}^{k+1}\right)}{2\left(1 + \beta^2\right)}$$



Jacobi Gauss-Seidel

Required amount of work

This iteration method has a slow convergence, it is possible to show

$$\|\mathbf{\Phi} - \mathbf{\Phi}^k\| \leq \rho^k \|\mathbf{\Phi} - \mathbf{\Phi}^0\|$$

where

$$\rho = 1 - \mathcal{O}\left(h^2\right) \qquad h = \Delta x = \Delta y$$

i.e. the error is reduced by $\mathcal{O}\left(h^2\right)$ each iteration. Requiring an error reduction to $\mathcal{O}\left(h^2\right)$ the number of iterations must satisfy $\rho^k \sim h^2$ which gives

$$\Rightarrow k \ln(\rho) \sim \ln(h^2)$$

$$\Rightarrow k \sim \frac{\ln(h^2)}{\ln(\rho)} \sim \frac{\ln(h^2)}{\ln(1-h^2)}$$

$$\{\ln(1-\epsilon) \approx -\epsilon\} \Rightarrow k \sim \frac{\ln(h^2)}{-h^2}$$

Consider a grid with total $N = n \times n$ nodes,

$$h = \Delta x = \Delta y = \frac{1}{n-1} \approx n^{-1} = N^{-1/2}$$

which implies that

$$k \sim \frac{\ln(N^{-1})}{-N^{-1}} = N \ln(N)$$

since the work per iteration is $\mathcal{O}(N)$, the total work is

$$W = \mathcal{O}\left(N^2 \ln(N)\right)$$

Convergence acceleration: Successive Over-Relaxation (SOR)

Consider equations system $L\ U = b$.

Decompose L as

$$L = L_L + L_D + L_U$$

.

We have $L\ U - b = 0$. We can write

$$L_D\ U = L_D\ U - \omega(L\ U - b)$$

Re-arranging the terms gives

$$(L_D + \omega L_L)\ U = [\omega L_U + (1-\omega)L_D]U + \omega b$$

The iteration algorithm reads

$$(L_D + \omega L_L)\ U^{k+1} = [\omega L_U + (1-\omega)L_D]U^k + \omega b$$

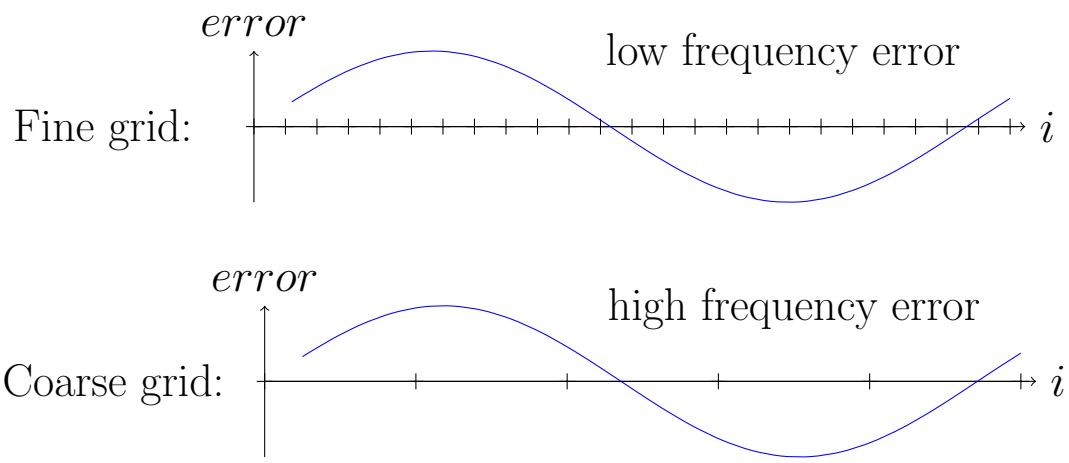
This can be written as

$$U_i^{k+1} = (1-\omega)U^k + \frac{\omega}{L_{i,i}}\left(b_i - \sum_{j>i} L_{i,j}U_j^k - \sum_{j<i} L_{i,j}U_j^{k+1}\right)$$

$\omega = 1$ corresponds to the Guass-Seidel method.

Convergence acceleration: Multigrid method

We note that the Gauss-Seidel provides good smoothing of the local error, but converges slowly since it takes time for boundary information to propagate into the domain. Further, the low-frequency errors decay much slower compared to the those with high frequency. Note that an oscillation with a low frequency on a fine grid will be experienced as a one with high frequency on a coarse grid. The reason is that the function value changes faster when moving from one grid point to the next on a corse grid.



This observation is the basis of the so-called *multigrid* method. Here, by solving the equations on different grids with different degree of coarseness the decay of low-frequent errors is accelerated.

Example: two-level multigrid

To demonstrate different steps of the multigrid let us consider the equation $L\Phi_{i,j} = b_{i,j}$

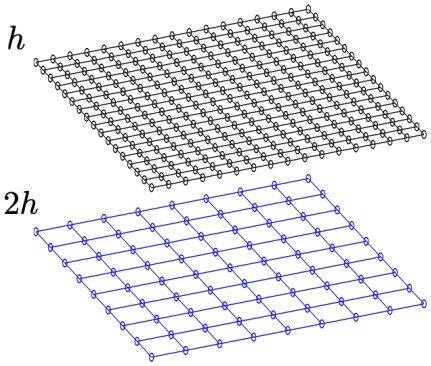
Define the correction to the solution towards the true discrete solution as

$$\begin{array}{lcl} \Phi_{i,j} & = & \Phi^k_{i,j} + \Delta\Phi_{i,j} \\ \Phi^k_{i,j} & - & \text{solution at iteration level } k \\ \Delta\Phi_{i,j} & - & \text{correction to } \Phi^k_{i,j} \text{ to true discrete solution} \end{array}$$

This implies that

$$L\Delta\Phi_{i,j} = b_{i,j} - L\Phi^k_{i,j} \equiv -R_{i,j}.$$

For simplicity, we consider a two-level multigrid consisting of one level of coarsening of the original mesh. This residual equation is solved on a coarser grid and correction is interpolated on a finer grid.

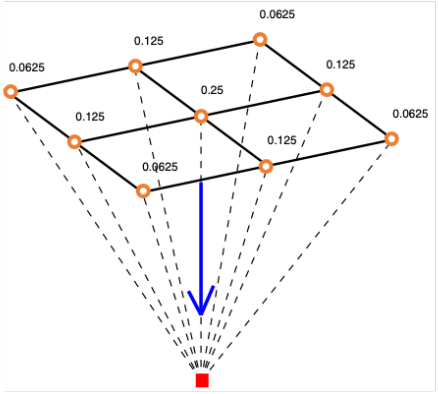


Multigrid algorithm requires transfer of data between different levels of grid. We define I as the transfer operator between grids:

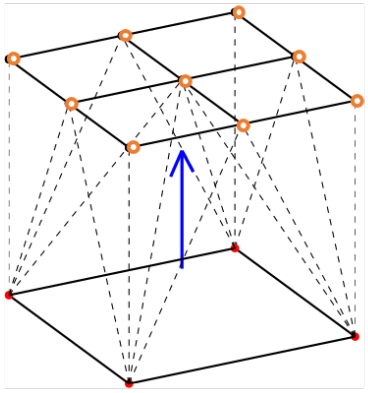
I_h^{2h} : restriction operator from grid h to grid $2h$
Ex.: injection (every other point in each direction)

I_{2h}^h : prolongation operator from grid $2h$ to grid h
Ex.: bilinear interpolation, intermediate values
-average of right and left (or top and bottom)
-average of averages

Restriction (I_h^{2h})



Prolongation (I_{2h}^h)



We define the following two level scheme:

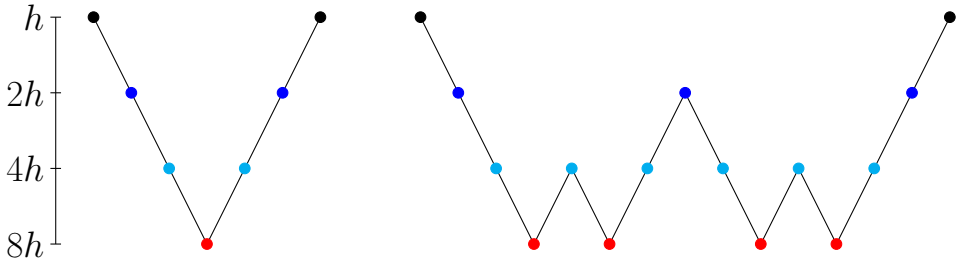
1. iterate k times $L\Phi_{i,j} = b_{i,j}$ on the fine grid.
2. compute residual $R_{i,j} = L\Phi^k_{i,j} - b_{i,j}$.
3. iterate k times $L\Delta\Phi_{i,j} = -I^2_1 R_{i,j}$ on the coarse grid. $\Delta\Phi^0_{i,j} = 0$ (starting value).
4. correct solution on the fine grid $\Phi_{i,j} = \Phi^k_{i,j} + I^1_2 \Delta\Phi_{i,j}$ and go to step 1.

At the end of step 1 convergence is checked.

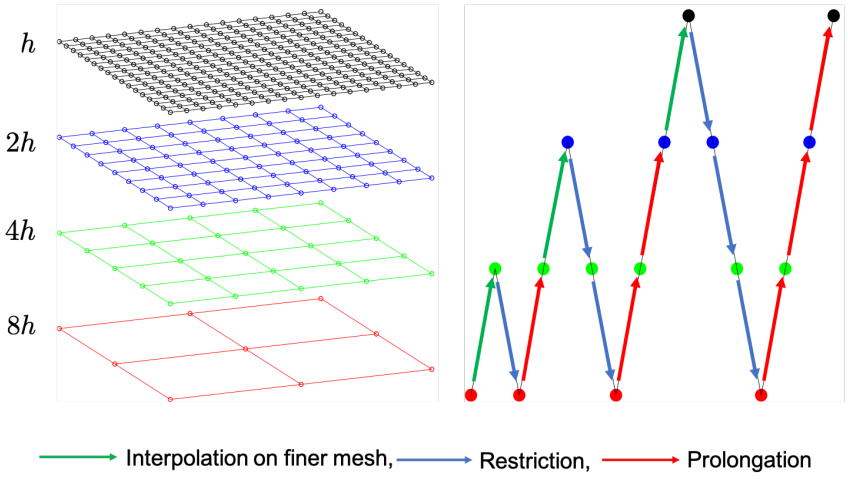
A useful relation between the restriction and prolongation operators is $I^h_{2h} = c\left(I^{2h}_h\right)$, where c is a real number.

The work of the multigrid method is estimated to be $W \sim \mathcal{O}\left(N \ln N\right)$, a substantial improvement over the Gauss-Seidel method ($\mathcal{O}\left(N^2 \ln N\right)$). Here $N = n \times n$ is the total number of the grid points.

Examples of MG cycles



V-cycle MG W-cycle MG



Full multigrid

In *Full Multigrid* (FMG) approach the computations start at coarsest grid level. The work for FMG is estimated to be $W \sim \mathcal{O}(N)$.