上海交通大学
SHANGHAI JIAO TONG UNIVERSITY

SJTU COMPUTATIONAL MARINE HYDRODYNAMICS LAB
上海交大船舶与海洋工程计算水动力学研究中心

Class-7

NA26018

Finite Element Analysis of Solids and Fluids

万德成

dcwan@sjtu.edu.cn , http://dcwan.sjtu.edu.cn/

上海交通大学

船舶海洋与建筑工程学院
海洋工程国家重点实验室

2020年

Iterative method

- Direct methods are generally not appropriate for solving large systems of equations particularly when the coefficient matrix is sparse
- In contrast, iterative methods are more appealing for these problems since the solution of the linearized system becomes part of the iterative solution process. Add to that the low computer storage and low computational cost requirements of this approach relative to the direct method
- A brief examination of basic iterative methods is provided along with an appraisal of multigrid algorithms that are generally used to address their deficiency

Iterative method

To unify the presentation of these methods, the coefficient matrix will be written in the following form

$$\mathbf{A} = \mathbf{D} + \mathbf{L} + \mathbf{U}$$

where D, L, and U refers to a diagonal, strictly lower, and strictly upper matrix, respectively

Iterative methods for solving a linear system of the type $A\phi = b$, compute a series of solutions ϕ_n that, if certain conditions are satisfied, converge to the exact solution ϕ

- **$\phi_0$is selected as the initial condition or initial guess) and an iterative procedure that computes ϕ_n from the previously computed ϕ_{n-1} field is developed**

Jacobi Method

Iterative method

- Considering the system of equations, solution process starts by assigning guessed values to the unknown vector ϕ . These guessed values are used to calculate new estimates starting with ϕ_1, then ϕ_2, and computations proceed until a new estimate for ϕ_N is computed. This represents one iteration. Results obtained are treated as a new guess for the next iteration and the solution process is repeated
- Iterations continue until the changes in the predictions between two consecutive iterations drop below a vanishing value or until a preset convergence criterion is satisfied. Once this happens the final solution is reached

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1N-1} & a_{1N} \\ a_{21} & a_{22} & \dots & a_{2N-1} & a_{2N} \\ \vdots & \vdots & \dots & \vdots & \vdots \\ a_{N1} & a_{N2} & \dots & a_{NN-1} & a_{NN} \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \vdots \\ \phi_N \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ \vdots \\ b_N \end{bmatrix}$$

Iterative method

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1N-1} & a_{1N} \\ a_{21} & a_{22} & \dots & a_{2N-1} & a_{2N} \\ \vdots & \vdots & \dots & \vdots & \vdots \\ a_{N1} & a_{N2} & \dots & a_{NN-1} & a_{NN} \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \vdots \\ \phi_N \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ \vdots \\ b_N \end{bmatrix}$$

$$\phi_j^{(n)} = \frac{1}{a_{ii}} \left(b_i - \sum_{\substack{j=1 \\ j \neq i}}^{N} a_{ij} \phi_j^{(n-1)} \right) \qquad i = 1, 2, 3, \dots, N$$

Equation indicates that values obtained during an iteration are not used in the subsequent calculations during the same iteration but rather retained for the next iteration

Iterative method

$$\phi_j^{(n)} = \frac{1}{a_{ii}} \left(b_i - \sum_{\substack{j=1 \\ j \neq i}}^{N} a_{ij} \phi_j^{(n-1)} \right) \qquad i = 1, 2, 3, \ldots, N$$

$$\begin{bmatrix} a_{11} & 0 & \ldots & 0 & 0 \\ 0 & a_{22} & \ldots & 0 & 0 \\ \vdots & \vdots & \ldots & \vdots & \vdots \\ 0 & \ldots & 0 & a_{N-1,N-1} & 0 \\ 0 & 0 & \ldots & 0 & a_{NN} \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_{N-1} \\ \phi_N \end{bmatrix}$$

$$+ \begin{bmatrix} 0 & a_{12} & \ldots & a_{1N-1} & a_{1N} \\ a_{21} & 0 & \ldots & a_{2N-1} & a_{2N} \\ \vdots & \vdots & \ldots & \vdots & \vdots \\ a_{N-1,1} & a_{N-1,2} & \ldots & 0 & a_{N-1,N} \\ a_{N1} & a_{N2} & a_{N3} & \ldots & 0 \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_{N-1} \\ \phi_N \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ \vdots \\ b_N \end{bmatrix}$$

Iterative method

Solving for ϕ_N, Eq. yields

$$\begin{bmatrix} \phi_1^{(n)} \\ \phi_2^{(n)} \\ \vdots \\ \phi_{N-1}^{(n)} \\ \phi_N^{(n)} \end{bmatrix} = \begin{bmatrix} a_{11} & 0 & \dots & 0 & 0 \\ 0 & a_{22} & \dots & 0 & 0 \\ \vdots & \vdots & \dots & \vdots & \vdots \\ 0 & \dots & 0 & a_{N-1,N-1} & 0 \\ 0 & 0 & \dots & 0 & a_{NN} \end{bmatrix}^{-1}$$

$$\left(\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ \vdots \\ b_N \end{bmatrix} - \begin{bmatrix} 0 & a_{12} & \dots & a_{1N-1} & a_{1N} \\ a_{21} & 0 & \dots & a_{2N-1} & a_{2N} \\ \vdots & \vdots & \dots & \vdots & \vdots \\ a_{N-1,1} & a_{N-1,2} & \dots & 0 & a_{N-1,N} \\ a_{N1} & a_{N2} & a_{N3} & \dots & 0 \end{bmatrix} \begin{bmatrix} \phi_1^{(n-1)} \\ \phi_2^{(n-1)} \\ \vdots \\ \phi_{N-1}^{(n-1)} \\ \phi_N^{(n-1)} \end{bmatrix} \right)$$

$$\phi^{(n)} = -\mathbf{D}^{-1}(\mathbf{L}+\mathbf{U})\phi^{(n-1)} + \mathbf{D}^{-1}\mathbf{b}$$

Iterative method

Gauss-Seidel Method

A more popular take on the Jacobi is the Gauss-Seidel method, which has better convergence characteristics. It is somewhat less expensive memory-wise since it does not require storing the new estimates in a separate array. Rather, it uses the latest available estimate of ϕ in its calculations. The iterative formula in the Gauss Seidel method is given as

$$\phi_i^{(n)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij}\phi_j^{(n)} - \sum_{j=i+1}^{N} a_{ij}\phi_j^{(n-1)} \right) \quad i = 1, 2, 3, \ldots, N$$

Matrix form is written as

$$\boldsymbol{\phi}^{(n)} = -(\mathbf{D} + \mathbf{L})^{-1}\mathbf{U}\boldsymbol{\phi}^{(n-1)} + (\mathbf{D} + \mathbf{L})^{-1}\mathbf{b}$$

Iterative method

$$\phi_i^{(n)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij}\phi_j^{(n)} - \sum_{j=i+1}^{N} a_{ij}\phi_j^{(n-1)} \right) \quad i = 1, 2, 3, \ldots, N$$

$$\phi_j^{(n)} = \frac{1}{a_{ii}} \left(b_i - \sum_{\substack{j=1 \\ j \neq i}}^{N} a_{ij}\phi_j^{(n-1)} \right) \quad i = 1, 2, 3, \ldots, N$$

- **Gauss-Seidel method uses the most recent values in its iteration, specifically all $\phi_j^{(n)}$ values for $j < i$ since by the time ϕ_i is to be calculated, the values of ϕ_1, ϕ_2, ϕ_3 ,⋯ ϕ_{i-1} at the current iteration are already calculated**

- **This approach also saves memory since the newer value is always overwriting the previous one**

Iterative method

Example

$$\begin{bmatrix} 3 & -1 & 0 & 0 \\ -2 & 6 & -1 & 0 \\ 0 & -2 & 6 & -1 \\ 0 & 0 & -2 & 7 \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \end{bmatrix} = \begin{bmatrix} 3 \\ 4 \\ 5 \\ -3 \end{bmatrix}$$

Apply 5 iterations of the Gauss-Seidel and Jacobi methods to the system of equations in last Example and compute the errors at each iteration using the exact solution

$$\boldsymbol{\phi} = \begin{bmatrix} \frac{435}{299} & \frac{408}{299} & \frac{382}{299} & -\frac{19}{299} \end{bmatrix}$$

As an initial guess start with the field $\phi = [0\ 0\ 0\ 0]$

Iterative method

$$\phi_j^{(n)} = \frac{1}{a_{ii}} \left(b_i - \sum_{\substack{j=1 \\ j \neq i}}^{N} a_{ij} \phi_j^{(n-1)} \right) \qquad i = 1, 2, 3, \ldots, N$$

Denoting with a superscript (*) values from the previous iteration, the equations to be solved in the Jacobi method are as follows

$$\phi_1 = \frac{1}{3}\left(\phi_2^* + 3\right)$$

$$\phi_2 = \frac{1}{6}\left(2\phi_1^* + \phi_3^* + 4\right)$$

$$\phi_3 = \frac{1}{6}\left(2\phi_2^* + \phi_4^* + 5\right)$$

$$\phi_4 = \frac{1}{7}\left(2\phi_3^* - 3\right)$$

Iterative method

with the error given as

$$\varepsilon = \left| \phi_{exact} - \phi_{computed} \right|$$

The solution for the first iteration is obtained as

$$\phi_1 = \frac{1}{3}(0+3) = 1 \Rightarrow \varepsilon_1 = |1.4548 - 1| = 0.4548$$

$$\phi_2 = \frac{1}{6}(0+0+4) = 0.6667 \Rightarrow \varepsilon_2 = |1.3645 - 0.6667| = 0.6978$$

$$\phi_3 = \frac{1}{6}(0+0+5) = 0.8333 \Rightarrow \varepsilon_3 = |1.2776 - 0.8333| = 0.4443$$

$$\phi_4 = \frac{1}{7}(0-3) = -0.4286 \Rightarrow \varepsilon_4 = |-0.06354 + 0.4286| = 0.3650$$

Computations proceed in the same manner with solution obtained treated as the new guess

Iterative method

The results for the first five iterations are given in Table

Iter #	ϕ_1	ε_1	ϕ_2	ε_2	ϕ_3	ε_3	ϕ_4	ε_4
0	0	1.4548	0	1.3645	0	1.2776	0	0.06354
1	1	0.4548	0.6667	0.6978	0.8333	0.4443	−0.4286	0.3650
2	1.2222	0.2326	1.1389	0.2257	0.9841	0.2935	−0.1905	0.1269
3	1.3796	7.52E−02	1.2381	0.1265	1.1812	9.64E−02	−0.1474	8.38E−02
4	1.4127	4.22E−02	1.3234	4.11E−02	1.2215	5.61E−02	−9.11E−02	2.75E−02
5	1.4411	1.37E−02	1.3411	2.34E−02	1.2593	1.83E−02	−7.96E−02	1.6E−02

Denoting with a superscript (*) values from the previous iteration, the equations to be solved in the Gauss-Seidel method are as follows:

$$\phi_i^{(n)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}\phi_j^{(n)} - \sum_{j=i+1}^{N} a_{ij}\phi_j^{(n-1)}\right) \qquad i = 1, 2, 3, \ldots, N$$

$$\phi_1 = \frac{1}{3}\left(\phi_2^* + 3\right)$$

$$\phi_2 = \frac{1}{6}\left(2\phi_1 + \phi_3^* + 4\right)$$

$$\phi_3 = \frac{1}{6}\left(2\phi_2 + \phi_4^* + 5\right)$$

$$\phi_4 = \frac{1}{7}\left(2\phi_3 - 3\right)$$

Iterative method

The solution for the first iteration is obtained as

$$\phi_1 = \frac{1}{3}(0+3) = 1 \Rightarrow \varepsilon_1 = |1.4548 - 1| = 0.4548$$

$$\phi_2 = \frac{1}{6}(2*1+0+4) = 1 \Rightarrow \varepsilon_2 = |1.3645 - 1| = 0.3645$$

$$\phi_3 = \frac{1}{6}(2*1+0+5) = 1.1667 \Rightarrow \varepsilon_3 = |1.2776 - 1.1667| = 0.1109$$

$$\phi_4 = \frac{1}{7}(2*1.1667-3) = -0.09523 \Rightarrow \varepsilon_4 = |-0.06354 + 0.09523| = 0.03169$$

Iterative method

The results for the first five iterations are given in Table (Jacobi)

Iter #	ϕ_1	ε_1	ϕ_2	ε_2	ϕ_3	ε_3	ϕ_4	ε_4
0	0	1.4548	0	1.3645	0	1.2776	0	0.06354
1	1	0.4548	0.6667	0.6978	0.8333	0.4443	−0.4286	0.3650
2	1.2222	0.2326	1.1389	0.2257	0.9841	0.2935	−0.1905	0.1269
3	1.3796	7.52E−02	1.2381	0.1265	1.1812	9.64E−02	−0.1474	8.38E−02
4	1.4127	4.22E−02	1.3234	4.11E−02	1.2215	5.61E−02	−9.11E−02	2.75E−02
5	1.4411	1.37E−02	1.3411	2.34E−02	1.2593	1.83E−02	−7.96E−02	1.6E−02

The results for the first five iterations are given in Table (G-S)

Iter #	ϕ_1	ε_1	ϕ_2	ε_2	ϕ_3	ε_3	ϕ_4	ε_4
0	0	1.4548	0	1.3645	0	1.2776	0	0.06354
1	1	0.4548	1	0.3645	1.1667	0.1109	−0.09523	0.03169
2	1.3333	0.1215	1.3056	5.90E−02	1.2526	2.49E−02	−7.07E−02	7.13E−03
3	1.4352	1.97E−02	1.3538	1.07E−02	1.2728	4.76E−03	−6.49E−02	1.36E−03
4	1.4513	3.57E−03	1.3626	1.98E−03	1.2767	8.88E−04	−6.38E−02	2.54E−04
5	1.4542	6.61E−04	1.3642	3.68E−04	1.2774	1.65E−04	−6.36E−02	4.72E−05

Iterative method

Preconditioning and Iterative Methods

A "fixed-point" iteration can always be associated to the above system by decomposing matrix A as

$$\mathbf{A} = \mathbf{M} - \mathbf{N}$$

Using this decomposition, AΦ=b is rewritten as

$$(\mathbf{M} - \mathbf{N})\boldsymbol{\phi} = \mathbf{b}$$

Applying a fixed point iteration solution procedure, Eq. becomes

$$\mathbf{M}\boldsymbol{\phi}^{(n)} = \mathbf{N}\boldsymbol{\phi}^{(n-1)} + \mathbf{b}$$

which can be rewritten in the following form

$$\boldsymbol{\phi}^{(n)} = \mathbf{B}\boldsymbol{\phi}^{(n-1)} + \mathbf{C}\mathbf{b} \quad n = 1, 2, \ldots$$

$$\mathbf{B} = \mathbf{M}^{-1}\mathbf{N} \qquad \mathbf{C} = \mathbf{M}^{-1}$$

Different choices of these matrices define different iterative methods

Iterative method

- The rate of convergence of iterative methods depends on the spectral properties of the iteration matrix B
- Based on that an iterative method looks for a transformation of the system of equations into an equivalent one that has the same solution, but of better spectral properties
- Under these conditions the eigenvalues of the equivalent system are more clustered allowing the iterative solution to be obtained faster than with the original system.

A preconditioner is defined as a matrix that effects such a transformation

Iterative method

A preconditioning matrix P is defined such that the system

$$\mathbf{A\Phi=b}$$

$$\mathbf{P}^{-1}\mathbf{A}\boldsymbol{\phi} = \mathbf{P}^{-1}\mathbf{b}$$

has the same solution as the original system $A\phi = b$, but the spectral properties of its coefficient matrix $P^{-1}A$ are more conducive, recall

$$\boldsymbol{\phi}^{(n)} = \mathbf{B}\boldsymbol{\phi}^{(n-1)} + \mathbf{Cb} \quad n = 1, 2, \ldots \qquad \mathbf{B} = \mathbf{M}^{-1}\mathbf{N} \qquad \mathbf{C} = \mathbf{M}^{-1}$$

With $\mathbf{M} = \mathbf{P}$ and $\mathbf{A} = \mathbf{P} - \mathbf{N}$

$$\begin{aligned}
\boldsymbol{\phi}^{(n)} &= \mathbf{B}\boldsymbol{\phi}^{(n-1)} + \mathbf{Cb} \\
&= \mathbf{P}^{-1}\mathbf{N}\boldsymbol{\phi}^{(n-1)} + \mathbf{P}^{-1}\mathbf{b} \\
&= \mathbf{P}^{-1}(\mathbf{P} - \mathbf{A})\boldsymbol{\phi}^{(n-1)} + \mathbf{P}^{-1}\mathbf{b} \\
&= (\mathbf{I} - \mathbf{P}^{-1}\mathbf{A})\boldsymbol{\phi}^{(n-1)} + \mathbf{P}^{-1}\mathbf{b}
\end{aligned}$$

Iterative method

in residual form can be written as

$$\begin{aligned}\boldsymbol{\phi}^{(n)} &= \left(\mathbf{I} - \mathbf{P}^{-1}\mathbf{A}\right)\boldsymbol{\phi}^{(n-1)} + \mathbf{P}^{-1}\mathbf{b} \\ &= \boldsymbol{\phi}^{(n-1)} + \mathbf{P}^{-1}\left(\mathbf{b} - \mathbf{A}\boldsymbol{\phi}^{(n-1)}\right) \\ &= \boldsymbol{\phi}^{(n-1)} + \mathbf{P}^{-1}\mathbf{r}^{(n-1)}\end{aligned}$$

Jacobi: $\phi^{(n)} = -\mathbf{D}^{-1}(\mathbf{L}+\mathbf{U})\phi^{(n-1)} + \mathbf{D}^{-1}\mathbf{b}$

G-S: $\boldsymbol{\phi}^{(n)} = -(\mathbf{D}+\mathbf{L})^{-1}\mathbf{U}\boldsymbol{\phi}^{(n-1)} + (\mathbf{D}+\mathbf{L})^{-1}\mathbf{b}$

By comparison, the preconditioning matrix for the Jacobi and Gauss-Seidel methods are simply

$$\mathbf{P}_J = \mathbf{D}$$

$$\mathbf{P}_{GS} = \mathbf{D} + \mathbf{L}$$

where D and L are respectively the diagonal and lower triangular part of matrix A

Iterative method

- Preconditioning is a manipulation of the original system to improve its spectral properties with the preconditioning matrix P used in the associated iterative procedure

- It is possible to develop more advanced preconditioning matrices in which the coefficients are defined in a more complex way

- The low rate of convergence of the Gauss-Seidel and Jacobi methods was the prime motivator for the development of faster iterative techniques. One approach to accelerate the convergence rate of solvers and to develop iterative methods is through the use of more advanced preconditioners

Iterative method

Incomplete LU Factorization with no Fill-in ILU(0)

A simple, efficient, approach for that purpose is to perform an incomplete factorization of the original matrix of coefficients A

- **The stress on incomplete is essential since a complete factorization of A into a lower L and an upper triangular matrix U is equivalent to a direct solution and is very expensive in term of memory requirements (fill in and loss of sparsity) and computational cost**

- **Many variants of the ILU factorization technique exist and the simplest is the one denoted by ILU(0). In ILU(0) the pattern of zero elements in the combined L and U matrices is taken to be precisely the pattern of zero elements in the original matrix A**

Iterative method

- Using Gaussian elimination, computations are performed as in the case of a full LU factorization, but any new nonzero element arising in the process is dropped if it appears at a location where a zero element exists in the original matrix A
- Hence, the combined L and U matrices have together the same number of non zeros as the original matrix A
- In the process however, the accuracy is reduced thereby increasing the number of required iterations for convergence to be reached
- To remedy this shortcoming, more accurate ILU factorization methods, which are often more efficient and more reliable, have been developed. These methods, differing by the level of fill-in allowed, are denoted by ILU(p), where p represents the order of fill-ins. The higher the level of fill-ins, the more expensive the ILU decomposition step becomes

Iterative method

An ILU(0) factorization algorithm which assumes L to be a unit lower triangular matrix and for which the same matrix A is used to store the elements of the unit lower and upper triangular matrices L and U is as given next

ILU(0) Factorization Algorithm:

$$
\begin{aligned}
&For\ k = 1\ to\ N-1\\
&\{\\
&\quad For\ i = k+1\ to\ N\ and\ if\ a_{ik} \neq 0\ \ Do:\\
&\quad \{\\
&\qquad a_{ik} = \frac{a_{ik}}{a_{kk}}\ \ (\ell\ values)\\
&\qquad \{\\
&\qquad\quad For\ \ j = k+1\ to\ N\ and\ if\ a_{ij} \neq 0\ Do:\\
&\qquad\quad a_{ij} = a_{ij} - a_{ik} * a_{kj}\ \ (u\ values)\\
&\qquad \}\\
&\quad \}\\
&\}
\end{aligned}
$$

Iterative method

ILU Factorization Preconditioners

A very popular class of preconditioners is based on incomplete factorizations. In the discussions of direct methods it was shown that decomposing a sparse matrix A into the product of a lower and an upper triangular matrices may lead to substantial fill-in

- **Because a preconditioner is only required to be an approximation to A^{-1}, it is sufficient to look for an approximate decomposition of A such that $A \approx \bar{L}\bar{U}$. Choosing $P = \bar{L}\bar{U}$ leads also to an efficient evaluation of the inverse of the preconditioned matrix P^{-1}, since the inversion can easily be performed by the forward and backward substitution, as described above, in which the exact L and U are now replaced by the approximations $\bar{L}$ and $\bar{U}$, respectively**

Iterative method

For the ILU(0) method, the incomplete factorization mimics the nonzero elements sparsity of the original matrix such that the pre-conditioner has exactly the size of the original matrix

- In order to reduce the storage needed, Pommerell introduced a simplified version of the ILU called diagonal ILU (DILU). In the DILU the fill-in of the off-diagonal elements is eliminated (i.e., the upper and lower parts of the matrix are kept unchanged) and only the diagonal elements are modified

$$\mathbf{P} = (\mathbf{D}^* + \mathbf{L})\mathbf{D}^{*-1}(\mathbf{D}^* + \mathbf{U})$$

where L and U are the lower and upper triangular decomposition of A, and $D*$ is now a proper diagonal matrix, different from the diagonal of A. The $D*$ matrix is thus defined, as shown below, in a way that the diagonal of the product of the matrices in Eq. equals the diagonal of A

Iterative method

Algorithm for the Calculation of $D*$ in the DILU Method:

$$
\begin{aligned}
&For\ i = 1\ to\ N\ \ Do: \\
&\quad \{ \\
&\qquad d_{ii} = a_{ii} \\
&\quad \} \\
&For\ i = 1\ to\ N\ \ Do: \\
&\quad \{ \\
&\qquad For\ j = i+1\ to\ N\ and\ if\ a_{ij} \neq 0, a_{ji} \neq 0\ Do: \\
&\qquad\quad \{ \\
&\qquad\qquad d_{jj} = d_{jj} - \frac{a_{ji}}{d_{ii}} * a_{ij} \\
&\qquad\quad \} \\
&\quad \}
\end{aligned}
$$

Iterative method

Forward and Backward Solution Algorithm with the DILU Method:

$$
\begin{aligned}
&For\ i = 1\ to\ N\ \ Do: \\
&\{ \\
&\quad For\ \ j = 1\ to\ i-1\ \ Do: \\
&\quad \{ \\
&\qquad t_i = d_{ii}^{-1}(r_i - \ell_{ij} * t_j) \\
&\quad \} \\
&\} \\
&For\ i = N\ to\ 1\ \ Do: \\
&\{ \\
&\quad For\ \ j = i+1\ to\ N\ \ Do: \\
&\quad \{ \\
&\qquad \phi'_i = t_i - d_{ii}^{-1}(u_{ij} * t_j) \\
&\quad \} \\
&\}
\end{aligned}
$$

Iterative method

The Multigrid Approach

A severe limitation for iterative solvers:

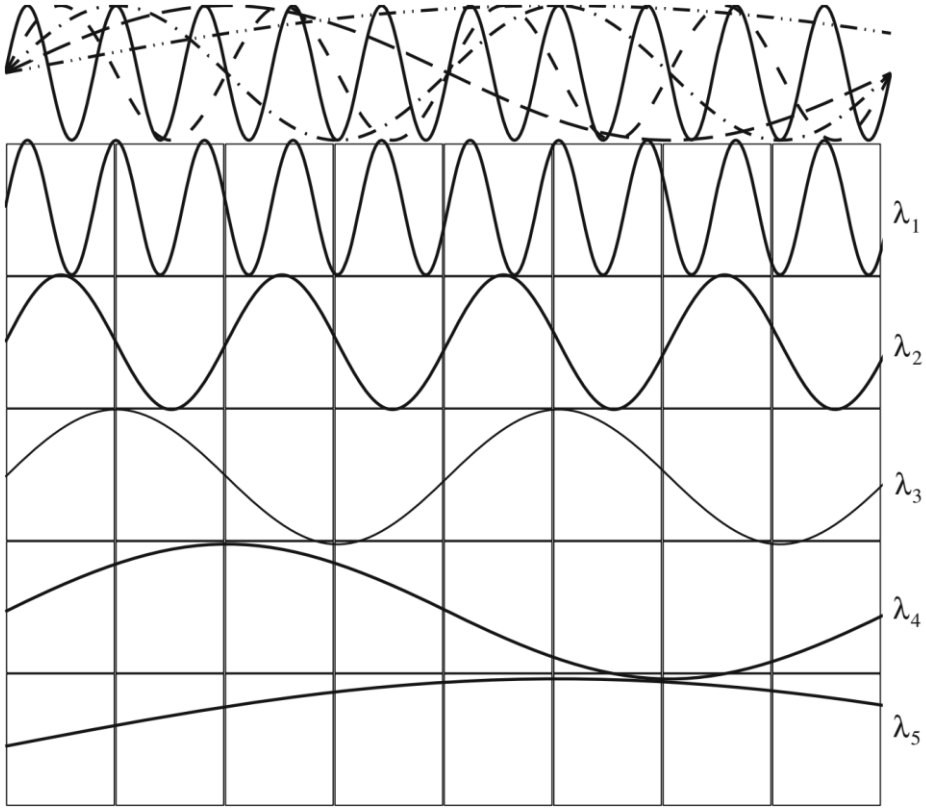
The rate of convergence of iterative methods drastically deteriorates as the size of the algebraic system increases, with the drop in convergence rate even observed in medium to large systems after the initial errors have been eliminated

It was found that the combination of multigrid and iterative methods can practically remedy this weakness

- **Multigrid methods started with the work of Fedorenko (Geometric Multigrid), Poussin (Algebraic Multigrid), and Settari and Azziz, and gained more interest with the theoretical work of Brandt**

Iterative method

- In Multigrid solver, high-frequency or oscillatory errors are easily eliminated with standard iterative solvers (Jacobi, Gauss-Seidel, ILU), these solution techniques cannot easily remove the smooth or low frequency error components

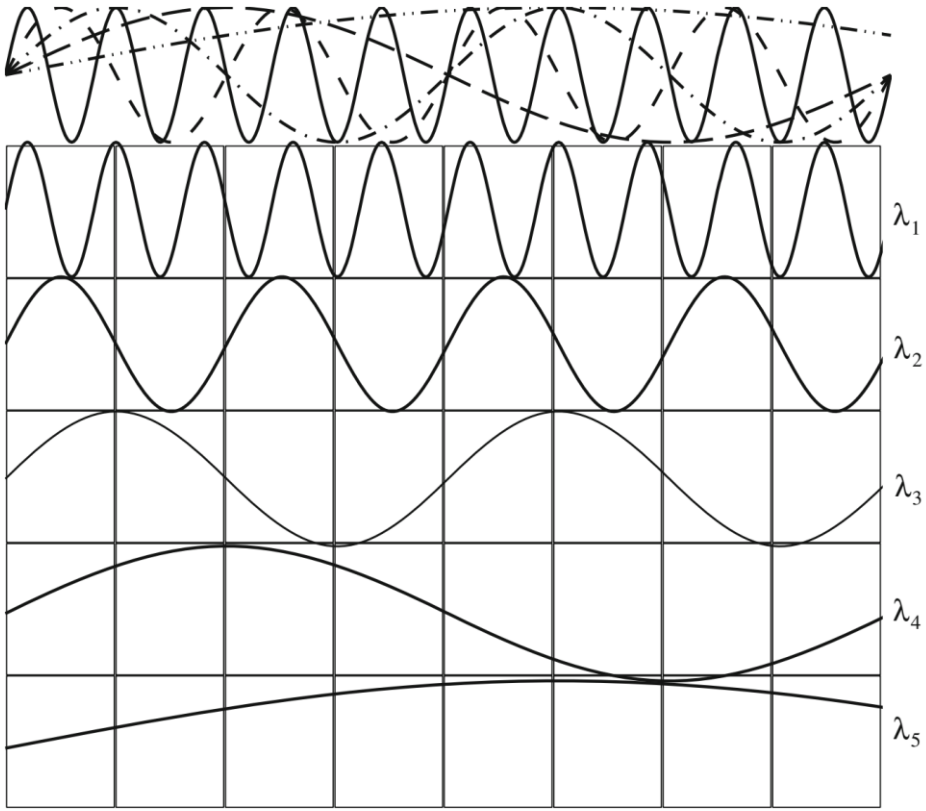


- The one dimensional domain is discretized using the one dimensional grid shown and the various modes are separately plotted over the same grid
- The error modes shown in Fig. vary from high frequency of short wavelength λ1 to low frequency of long wavelength λ5

Schematic of different error modes in a one dimensional grid

Iterative method

- The high frequency error appears oscillatory over an element and is easily sensed by the iterative method

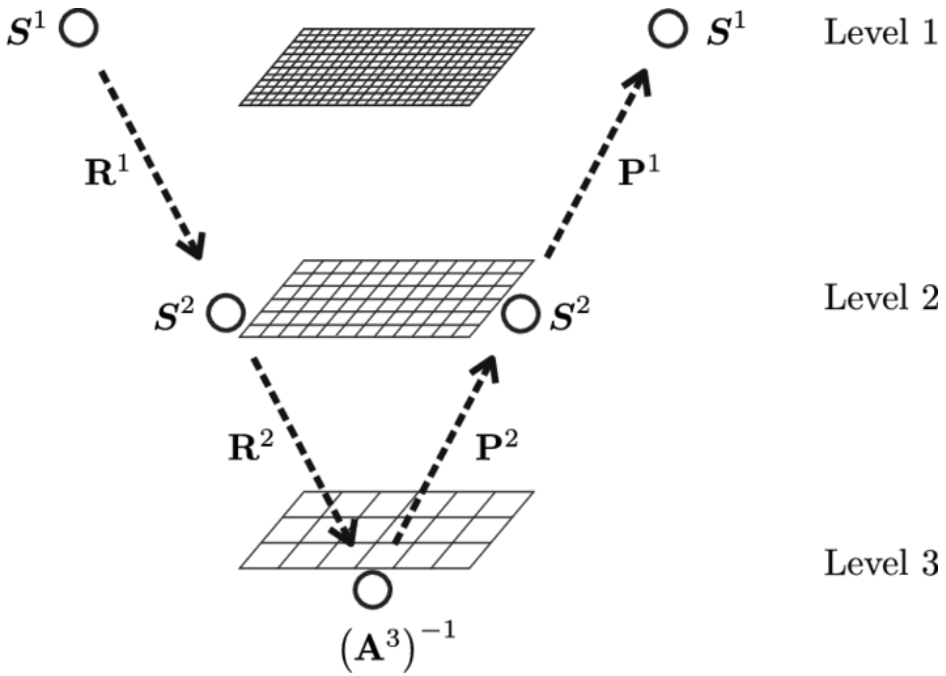


- As the frequency of the error decreases(as the wavelength λ increases), the error becomes increasingly smoother over the grid as only a small portion of the wavelength lies within any cell. This gets worse as the grid is further refined, leading to a higher number of equations as the size of the system increases

Iterative method

Multigrid methods improve the efficiency of iterative solvers by ensuring that the resulting low frequency errors that arise from the application of a smoother at any grid level are transformed into higher frequency errors at a coarser grid level

- **By using a hierarchy of coarse grids, multigrid methods are able to overcome the convergence degradation**



Iterative method

Generally the coarse mesh can be formed using

- either the topology and geometry of the finer mesh, this is akin to generating a new mesh for each coarse level on top of the finer level mesh
- or by direct agglomeration of the finer mesh elements; this approach is also known as the Algebraic MultiGrid Method (AMG)

A M G

1. In the AMG no geometric information is directly needed or used, and the agglomeration process is purely algebraic, with the equations at each coarse level reconstructed from those of the finer level, again through the agglomeration process

2. This approach can be used to build highly efficient and robust linear solvers, for both highly anisotropic grids and/or problems with large changes in the coefficients of their equations

Iterative method

In either approach (GMG, AMG), a multigrid cycling procedure is used to guide the traversal of the various grid hierarchies. Each traversal from a fine grid to a coarse one involves:

I. a restriction procedure
II. the setup or update of the system of equations for the coarse grid level
III. the application of a number of smoother iterations

A traversal from a coarse grid to a finer one requires:

I. a prolongation procedure
II. the correction of the field values at the finer level
III. the application of a number of smoother iterations on the equations constructed during restriction

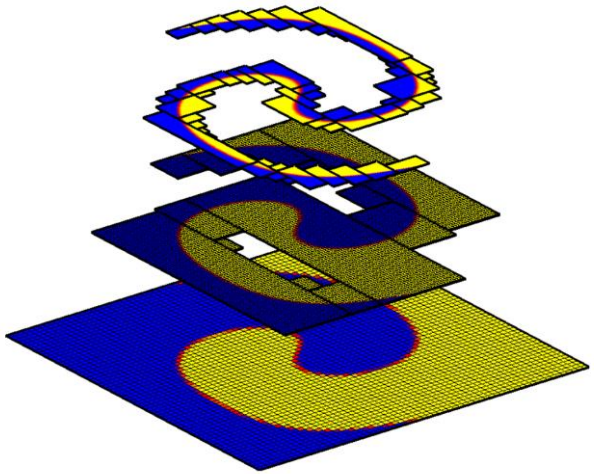
Iterative method

Element Agglomeration/Coarsening

The first step in the solution process is to generate the coarse/ fine grid levels by an agglomeration/ coarsening algorithm. Two different approaches can be adopted for that purpose

- **In the first approach, the coarse mesh is initially generated and the fine levels are obtained by refinement**

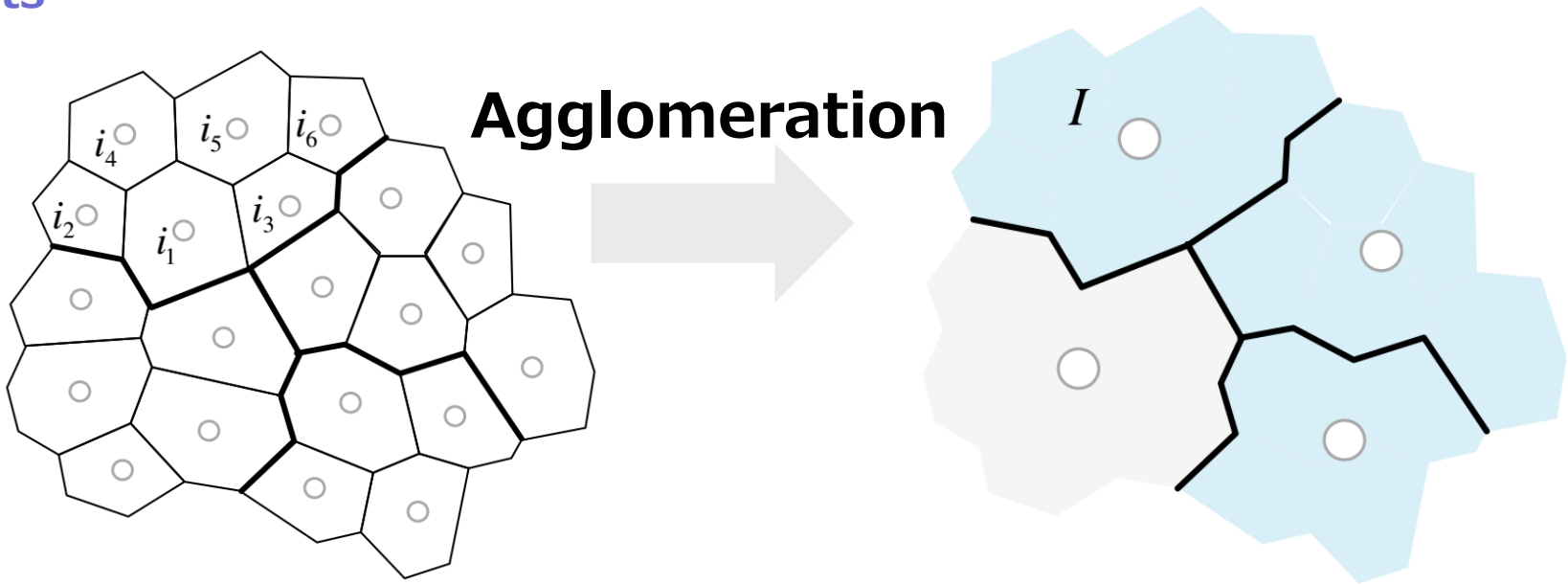
This facilitates the definitions of the coarse-fine grid relations and is attractive in an adaptive grid setup. A major drawback however, is the dependence of the fine grid distribution on the coarse grid



Iterative method

- In the second approach (recommended), the process starts with the generation of the finest mesh that will be used in solving the problem. The coarse grid levels are developed through agglomeration of the fine-grid elements, as shown in Fig.

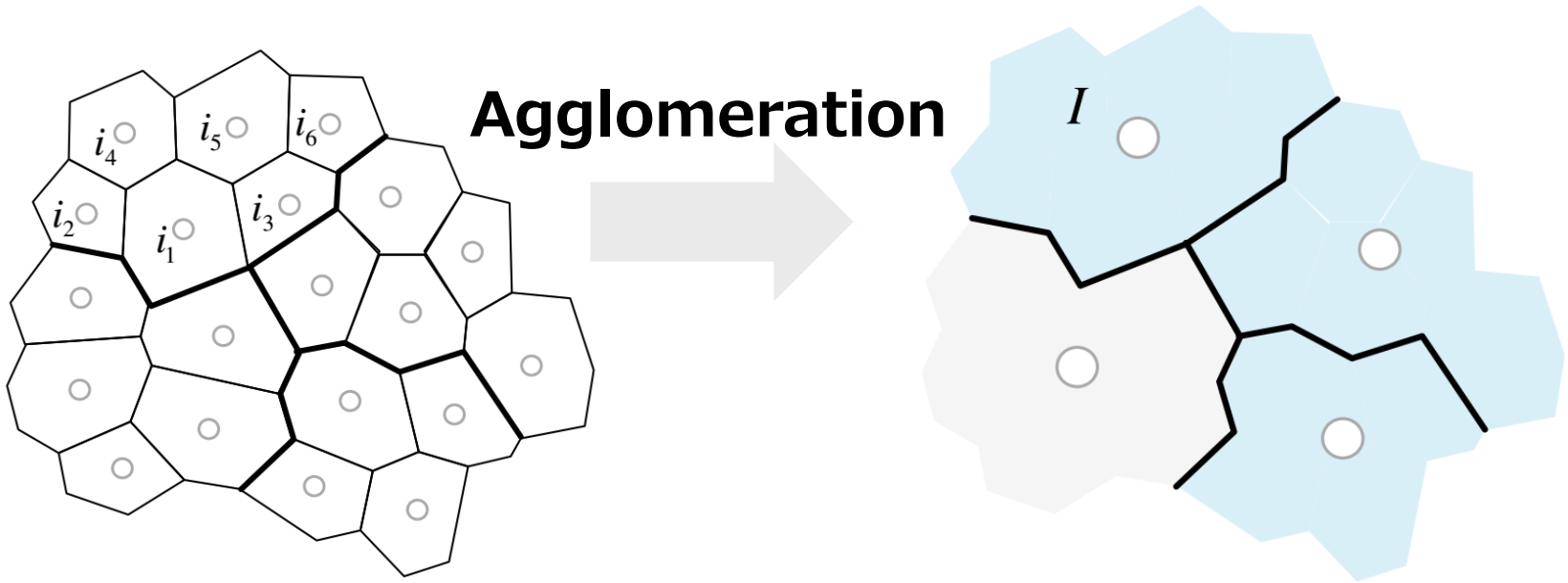
with the agglomeration process based either on the elements geometry or on a criterion to be satisfied by the coefficients of neighboring elements



Iterative method

Agglomeration of a fine grid level to form a coarse grid level:

- **Coarse grid levels are generated by fusing fine grid elements through an agglomeration algorithm**
- **For each coarse grid level, the algorithm is repeatedly applied until all grid cells of the finer level become associated with coarse grid cells**



Iterative method

The Restriction Step and Coarse Level Coefficients

Original form:

$$\mathbf{A}\boldsymbol{\phi} = \mathbf{b}$$

$$\mathbf{e}^{(n)} = \boldsymbol{\phi}^{(n)} - \boldsymbol{\phi}$$

$$\mathbf{r}^{(n)} = \mathbf{b} - \mathbf{A}\boldsymbol{\phi}^{(n)}$$

Correction form:

$$\mathbf{A}\mathbf{e}^{(n)} = \mathbf{r}^{(n)}$$

Error at n: e(n); Residual at n: r(n)

- **The solution starts at the fine grid level. After performing few iterations, the error is transferred (restricted) to a coarser grid level and the solution is found at that level.**
- **Then after performing few iterations at that level the error is restricted again to a higher level and the sequence of events repeated until the highest or coarsest grid level is reached**

Iterative method

Let (k) denotes some level at which the solution has been found by solving the following system of equations in correction form

$$\mathbf{A}^{(k)}\mathbf{e}^{(k)} = \mathbf{r}^{(k)}$$

The next coarser level is (k+1) to which the error will be restricted. Let GI represents the set of cells i on the fine grid level (k) that are agglomerated to form cell I of the coarse grid level (k+1). Then, the system to be solved on the coarse grid at level (k+1) is

$$\mathbf{A}^{(k+1)}\mathbf{e}^{(k+1)} = \mathbf{r}^{(k+1)}$$

with the residuals on the RHS of Eq. computed as

$$\mathbf{r}^{(k+1)} = I_k^{k+1}\mathbf{r}^{(k)}$$

I_k^{k+1} is the restriction operator (i.e., the interpolation matrix) from the fine grid to the coarse grid as defined by the agglomeration process

Iterative method

In AMG the restriction operator is defined in a linear manner to yield a summation of the fine grid residuals as

$$\mathbf{r}_I^{(k+1)} = \sum_{i \in G_I} \mathbf{r}_i^{(k)}$$

Moreover, the coefficients of the coarse element are constructed by adding the appropriate coefficients of the constituting fine elements. Recalling that a linear equation after discretization has the form

$$a_i^{(k)} \phi_i^{(k)} + \sum_{j=NB(i)} a_{ij}^{(k)} \phi_j^{(k)} = b_i^{(k)}$$

the coarse mesh (k+1) correction equation becomes

$$a_I^{(k+1)} \phi_I'^{(k+1)} + \sum_{J=NB(I)} a_{IJ}^{(k+1)} \phi_J'^{(k+1)} = r_I^{(k+1)}$$

Iterative method

$$a_I^{(k+1)}\phi_I'^{(k+1)} + \sum_{J=NB(I)} a_{IJ}^{(k+1)}\phi_J'^{(k+1)} = r_I^{(k+1)}$$

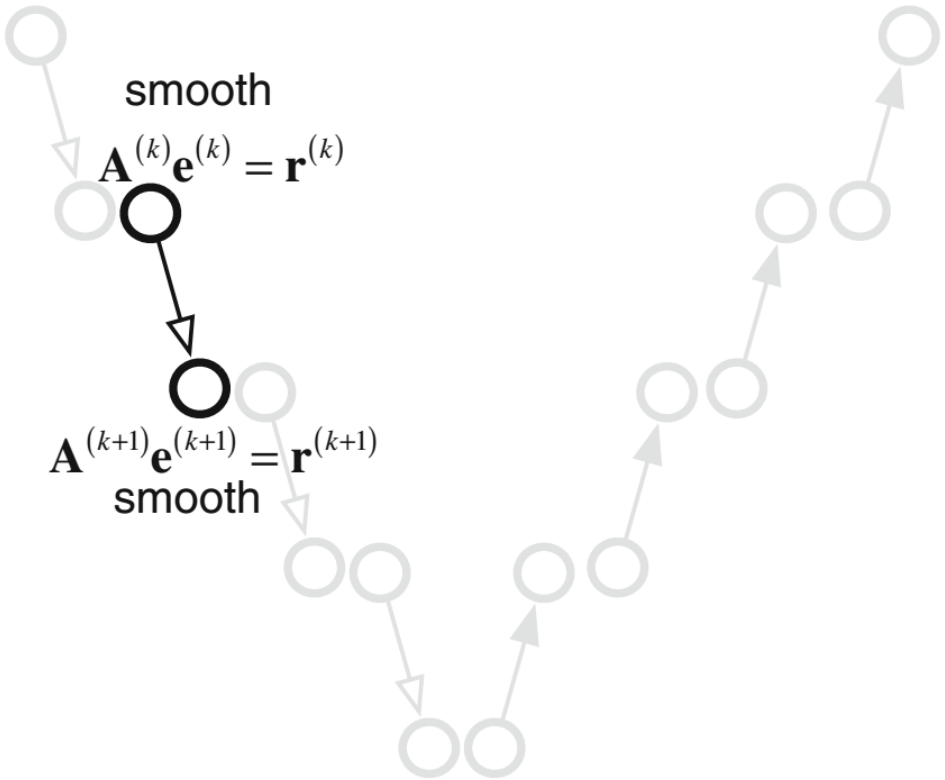
$a_I^{(k+1)}, a_{IJ}^{(k+1)},$ and $r_I^{(k+1)}$ **are derived directly from fine grid coefficients**

Restriction

$$a_I^{(k+1)} = \sum_{i\in G_I} a_i^{(k)} + \sum_{i\in G_I}\sum_{j\in G_I} a_{ij}^{(k)}$$

$$a_{IJ}^{(k+1)} = \sum_{i\in G_I}\sum_{\substack{j\notin G_I \\ j\in NB(I)}} a_{ij}^{(k)}$$

$$r_I^{(k+1)} = \sum_{i\in G_I} r_i^{(k)}$$



Iterative method

The Prolongation Step and Fine Grid Level Corrections

The prolongation operator is used to transfer the correction from a coarse to a fine grid level. i.e., the error at a coarse grid cell will be inherited by all the children of this cell on the fine grid level

The correction is basically obtained from the solution of the system of equations at the coarse grid. The interpolation or prolongation to the fine grid level is denote as

$$\mathbf{e}^{(k)} = I_{k+1}^{k}\mathbf{e}^{(k+1)}$$

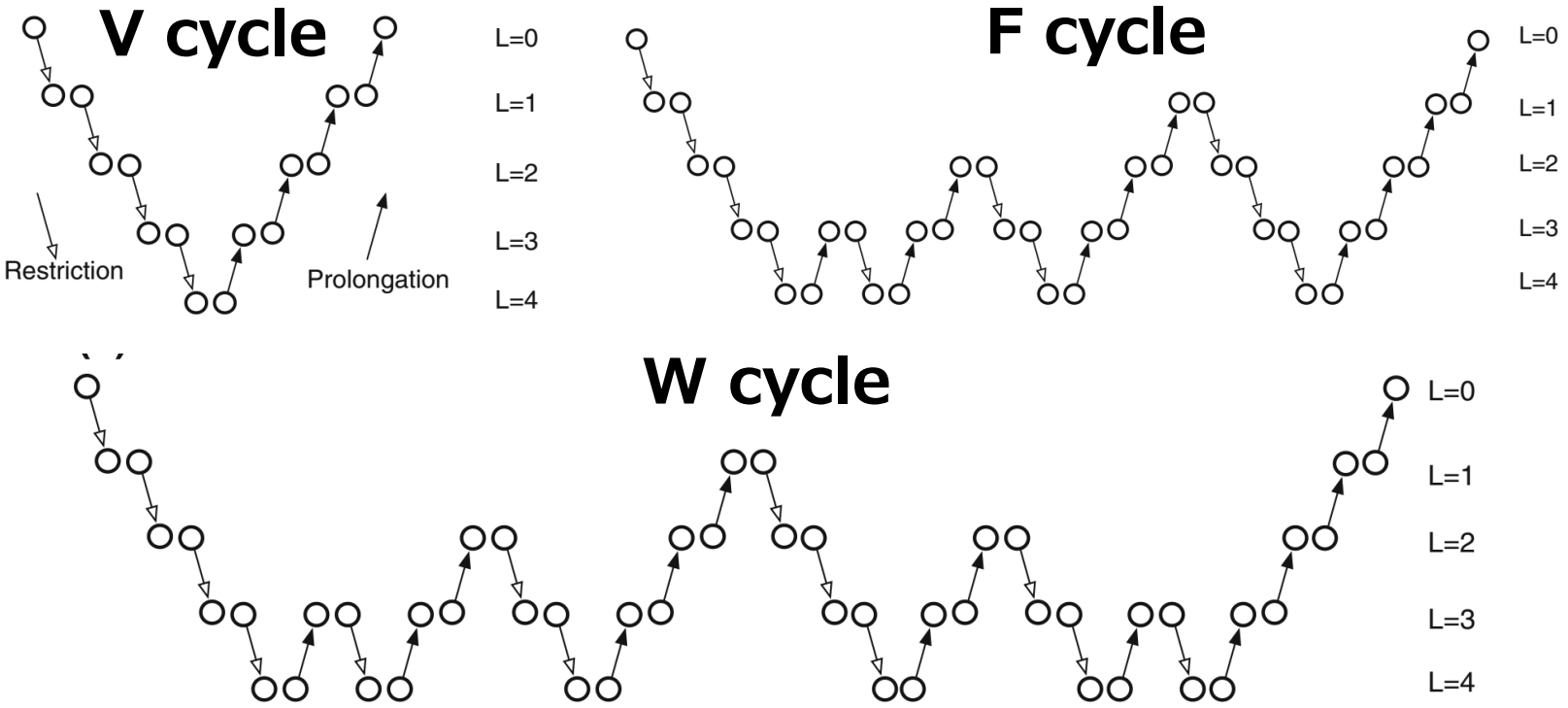
I_{k+1}^{k} **is an interpolation matrix from the coarse grid to the fine grid. Finally, the fine grid solution is corrected as**

$$\boldsymbol{\phi}^{(k)} \leftarrow \boldsymbol{\phi}^{(k)} + \mathbf{e}^{(k)}$$

Iterative method

Traversal Strategies and Algebraic Multigrid Cycles

Traversal strategies refer to the way by which coarse grids are visited during the solution process, which are also known as multigrid cycle. The usual cycles used in the AMG method are the V cycle, the W cycle, and the F cycle



Iterative method

- W cycle is based on applying smaller V cycles on each visited coarse grid level. In this manner, the W cycle consists of nested coarse and fine grid level sweeps with the complexity increasing as the number of AMG levels increases

For very stiff systems, the V cycle may not be sufficient for accelerating the solution and therefore, more iterations on the coarse level are required

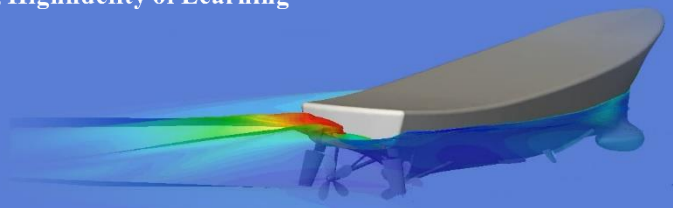
- The F cycle is a variant of the W cycle and can be thought of as splitting the W cycle in half. The F cycle requires less coarse level sweeps than the W cycle but more sweeps than the V cycle

F cycle lies in between the V and W cycling strategies

谢 谢！

CMHL SJTU COMPUTATIONAL MARINE HYDRODYNAMICS LAB
上海交大船舶与海洋工程计算水动力学研究中心

创新创智·求真求实
Creation of Mind, Highfidelity of Learning



http://dcwan.sjtu.edu.cn