Lecture 5 - The Finite Volume Method in 2D

Sections 4.4, 7.2, 7.6 (Versteeg)

ME 2256/MEMS 1256 - Applications of Computational Heat and Mass Transfer

Mechanical Engineering and Materials Science Department
University of Pittsburgh

Student Learning Objectives

At the end of the lecture, students should be able to:

- Understand iterative solution methods;
- Construct the 2D FVM formulation.

Generalized Transport Equation

- Recall the transport equation in vector form:

$$\underbrace{\frac{\partial}{\partial t}(\rho\phi)}_{\text{Accumulation}} + \underbrace{\nabla\cdot(\rho\vec{V}\phi)}_{\text{Convection}} = \underbrace{\nabla\cdot(\Gamma\nabla\phi)}_{\text{Diffusion}} + \underbrace{S}_{\text{Generation}}$$

- $\phi \rightarrow$ conserved quantity
- $\Gamma \rightarrow$ diffusion coefficient
- $\rho \rightarrow$ density

Algorithms for Solution

- To solve 1D and 2D steady-state diffusion problem, we need to select a proper algorithm to solve our system of equation
- There exists two main methods:
  1. Direct methods
     1.1. Matrix Inversion
     1.2. Gaussian Elimination
     1.3. Tri-Diagonal Matrix Algorithm (TDMA)
  2. Iterative methods
     2.1. Jacobi Iteration
     2.2. Gauss-Seidel
- Let us look at each of these methods, in no particular order, for an arbitrary system of equations

Gaussian Elimination

- Consider the arbitrary system of equation:

$$\begin{aligned} 10\phi_1 + \phi_2 + 2\phi_3 &= 44 \\ 2\phi_1 + 10\phi_2 + \phi_3 &= 51 \\ \phi_1 + 2\phi_2 + 10\phi_3 &= 61 \end{aligned} \implies \begin{bmatrix} 10 & 1 & 2 \\ 2 & 10 & 1 \\ 1 & 2 & 10 \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{bmatrix} = \begin{bmatrix} 44 \\ 51 \\ 61 \end{bmatrix}$$

- Gaussian Elimination requires getting the systems of equations in the following form via row operations:

$$\implies \begin{bmatrix} \# & \# & \# \\ 0 & \# & \# \\ 0 & 0 & \# \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{bmatrix} = \begin{bmatrix} B_1^* \\ B_2^* \\ B_3^* \end{bmatrix}$$

Gaussian Elimination

- To achieve the form of an upper triangular, there are two steps:
  1. Forward elimination
     - elimination of variables from subsequent rows to create final row with one number (i.e. LU factorization)
  2. Backward substitution
     - evaluate $\phi_3 \rightarrow \phi_2 \rightarrow \phi_1$ in sequence taking advantage of form following previous form, i.e. 1.

Gaussian Elimination

- Proceeding with forward elimination:

$$\begin{bmatrix} 10 & 1 & 2 \\ 2 & 10 & 1 \\ 1 & 2 & 10 \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{bmatrix} = \begin{bmatrix} 44 \\ 51 \\ 61 \end{bmatrix}$$

- Divide R1 by 10

$$\begin{bmatrix} 1 & 0.1 & 0.2 \\ 2 & 10 & 1 \\ 1 & 2 & 10 \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{bmatrix} = \begin{bmatrix} 4.4 \\ 51 \\ 61 \end{bmatrix}$$

- R2 = R2 − 2R1

$$\begin{bmatrix} 1 & 0.1 & 0.2 \\ 0 & 9.8 & 0.6 \\ 1 & 2 & 10 \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{bmatrix} = \begin{bmatrix} 4.4 \\ 42.2 \\ 61 \end{bmatrix}$$

Gaussian Elimination

- $R3 = R3 - R1$

$$\begin{bmatrix} 1 & 0.1 & 0.2 \\ 0 & 9.8 & 0.6 \\ 0 & 1.9 & 9.8 \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{bmatrix} = \begin{bmatrix} 4.4 \\ 42.2 \\ 56.6 \end{bmatrix}$$

- Divide R2 by 9.8

$$\begin{bmatrix} 1 & 0.1 & 0.2 \\ 0 & 1 & 0.0612 \\ 0 & 1.9 & 9.8 \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{bmatrix} = \begin{bmatrix} 4.4 \\ 4.3061 \\ 56.6 \end{bmatrix}$$

- $R3 = R3 - 1.9R2$

$$\begin{bmatrix} 1 & 0.1 & 0.2 \\ 0 & 1 & 0.0612 \\ 0 & 0 & 9.6837 \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{bmatrix} = \begin{bmatrix} 4.4 \\ 4.3061 \\ 48.4184 \end{bmatrix}$$

Gaussian Elimination

- Now we proceed with back-substitution

$$9.6837\phi_3 = 48.4184 \implies \phi_3 = 5$$

$$\phi_2 + 0.0612\phi_3 = 4.3061 \implies \phi_2 = 4$$

$$\phi_1 + 0.1\phi_2 + 0.2\phi_3 = 4.4 \implies \phi_1 = 3$$

Example #1

- Let us reconsider Example #1 from Lecture 2, where we have the following system of equations for 5 C.∀.:

$$\begin{bmatrix} 15 & -5 & 0 & 0 & 0 \\ -5 & 10 & -5 & 0 & 0 \\ 0 & -5 & 10 & -5 & 0 \\ 0 & 0 & -5 & 10 & -5 \\ 0 & 0 & 0 & -5 & 15 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \end{bmatrix} = \begin{bmatrix} 1,000 \\ 0 \\ 0 \\ 0 \\ 2,000 \end{bmatrix}$$

- Construct a code for Gaussian Elimination and compare the results to matrix inversion. See "L5Ex1.m" for the solution. Note the difference of runtimes.

Iterative Methods

- Iterative methods require conditions for stability
- Consider a 3x3 system

$$a_{11}\phi_1 + a_{12}\phi_2 + a_{13}\phi_3 = B_1$$
$$a_{21}\phi_1 + a_{22}\phi_2 + a_{23}\phi_3 = B_2$$
$$a_{31}\phi_1 + a_{32}\phi_2 + a_{33}\phi_3 = B_3$$

- We define diagonal dominance (i.e. non-singular) if the following condition is met:

$$|a_{ii}| > \sum_{\substack{j=1 \\ j\neq i}}^{N} |a_{ij}| \implies \begin{array}{l} |a_{11}| > |a_{12}| + |a_{13}| \\ |a_{22}| > |a_{21}| + |a_{23}| \\ |a_{33}| > |a_{31}| + |a_{32}| \end{array}$$

Iterative Methods

- This manifests into the Scarborough Criterion

$$\frac{\sum |a_{nb}|}{|a_P|} \begin{cases} \leq 1 \text{ for all cells} \\ <1 \text{ for one cell at least} \end{cases}$$

- This provides a sufficient condition, although not necessary one, for convergence of iterative methods
- Satisfaction of this criterion ensures equations will be converged by at least one iterative method

Example #2

- Let us reconsider Example #1 from Lecture 2, where we have the following system of equations for 5 C.∀.:

$$\begin{bmatrix} 15 & -5 & 0 & 0 & 0 \\ -5 & 10 & -5 & 0 & 0 \\ 0 & -5 & 10 & -5 & 0 \\ 0 & 0 & -5 & 10 & -5 \\ 0 & 0 & 0 & -5 & 15 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \end{bmatrix} = \begin{bmatrix} 1,000 \\ 0 \\ 0 \\ 0 \\ 2,000 \end{bmatrix}$$

- Is the Scarborough Criterion met? See "L5Ex2.m" for the solution.

Jacobi Iteration

- Considering the 3x3 system of equations previously defined in terms of variables a, ϕ and B, we will define k as the current iteration level and $k+1$ as the calculation level (i.e. next iteration)
- The basic equation for the Jacobi iterative method is given as:

$$\phi_i^{k+1} = \frac{1}{a_{ii}} \left(B_i - \sum_{\substack{j=1 \\ j \neq i}}^{N} a_{ij} \phi_j^k \right)$$

Jacobi Iteration

- Indexing through the domain (for i=1:n):

$$\phi_1^{k+1} = \frac{1}{a_{11}}\left(B_1 - a_{12}\phi_2^k - a_{13}\phi_3^k\right)$$

$$\phi_2^{k+1} = \frac{1}{a_{22}}\left(B_2 - a_{21}\phi_1^k - a_{23}\phi_3^k\right)$$

$$\phi_3^{k+1} = \frac{1}{a_{33}}\left(B_3 - a_{31}\phi_1^k - a_{32}\phi_2^k\right)$$

- Repeat this loop until there is convergence:

$$\text{Absolute error: } max|\phi_i^{k+1} - \phi_i^k| < \epsilon$$

$$\text{Relative error: } max\left|\frac{\phi_i^{k+1} - \phi_i^k}{\phi_i^k}\right| < \epsilon$$

- ϵ is the iterative threshold (i.e. 1e-5, 1e-10)

Residual

- We could also look at the residual of the solution:

$$R_1^k = a_{11}\phi_1^k + a_{12}\phi_2^k + a_{13}\phi_3^k - B_1^k$$

- As $\underline{\phi} \to \underline{\phi}_{\text{solution}}$, $R \to 0$
- The max residual is defined as:

$$max|R_i^{k+1} - R_i^k|<\epsilon$$

- The relative residual is defined as:

$$max\left|\frac{R_i^{k+1} - R_i^k}{R_i^k}\right|<\epsilon$$

Gauss-Seidel

- Instead of using the previous iterative level values, the Gauss-Seidel method uses the latest available values (i.e. all $n-1$ ϕ values)
- Indexing through the domain
- for i=1:n:

$$\phi_1^{k+1} = \frac{1}{a_{11}}\left(B_1 - a_{12}\phi_2^k - a_{13}\phi_3^k\right)$$

$$\phi_2^{k+1} = \frac{1}{a_{22}}\left(B_2 - a_{21}\phi_1^{k+1} - a_{23}\phi_3^k\right)$$

$$\phi_3^{k+1} = \frac{1}{a_{33}}\left(B_3 - a_{31}\phi_1^{k+1} - a_{32}\phi_2^{k+1}\right)$$

- end
- Iterate until there is convergence

Application to Transport Equation

- For our discretized system of equations (in 1D):

$$a_p \phi_P = \sum_{nb} a_{nb}\phi_{nb} + b = a_e \phi_E + a_w \phi_W + a_b \phi_b + B$$

- The Jacobi method would look like:

$$\phi_P^{k+1} = \frac{1}{a_p}\left(a_e \phi_E^k + a_w \phi_W^k + a_b \phi_b^k + B\right)$$

- The Gauss-Seidel method would look like:

$$\phi_P^{k+1} = \frac{1}{a_p}\left(a_e \phi_E^k + a_w \phi_W^{k+1} + a_b \phi_b^k + B\right)$$

- Which method should we use?

Application to Transport Equation

- The Gauss-Seidel method will converge faster because we are using information from the current iteration level to calculate that of the neighbors, i.e. the information propagates through the domain faster
- The Gauss-Seidel method is also more efficient - we only need one vector to store the solution (i.e. do not need $\underline{\phi}^k$ and $\underline{\phi}^{k+1}$ as separate vectors)

Relaxation Factor

- Consider a 1D system with $n = 5$ C.∀. ϕ is a 7x1 vector
- for i=2:n-1

$$\phi_P^{k+1} = \frac{1}{a_p}\left(a_w\phi_W^{k+1} + a_e\phi_E^k + B\right)\Big|_i$$

- end
- If we rearrange the equation:

$$\phi_P^{k+1} = \phi_P^k + \frac{1}{a_p}\left(a_w\phi_W^{k+1} + a_e\phi_E^k + B - a_p\phi_P^k\right)$$

- we can introduce the relaxation parameter ω

$$\phi_P^{k+1} = \phi_P^k + \omega\Delta\phi_P$$

$$\implies \phi_P^{k+1} = \phi_P^k + \frac{\omega}{a_p}\left(a_w\phi_W^{k+1} + a_e\phi_E^k + B - a_p\phi_P^k\right)$$

Relaxation Factor

- There are an infinite number of ω we can choose, but they are broken down into three regimes;
  1. $\omega = 1 \rightarrow$ Gauss-Seidel
  2. $\omega>1 \rightarrow$ successive over-relaxation
  3. $\omega<1 \rightarrow$ under-relaxation
- There is an optimum ω, typically greater than 1 and less than 2, that minimizes the number of iterations needed for convergence
- As $\omega<1$, the solution is more stable, but comes with a higher number of iterations

Direct Methods

- Direct matrix inversion is computationally expensive
- If we have 100 cells in 1D, we have a 100x100 matrix
- Using double precision, 100x100→80 kB for A matrix
- 2D 100x100 cells→10,000x10,000 matrix→800 MB for A matrix
- Most memory requirement is to store useless “0” values in the off-diagonals
- Additionally, iterative methods take longer with increasing element count
- Can we take advantage of the matrix structure (tri-diagonal system for 1D) to make an efficient solver?

TDMA

- The Tri-Diagonal Matrix Algorithm (TDMA) is cast based upon the three row entries:

$$a\phi_{i-1} + b\phi_i + c\phi_{i+1} = d_i$$

- This is visualized from the following system of equations:

$$\begin{bmatrix} b_1 & c_1 & 0 & 0 & \dots & 0 \\ a_2 & b_2 & c_2 & 0 & 0 & 0 \\ 0 & a_3 & b_3 & c_3 & 0 & \vdots \\ 0 & 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & 0 & 0 & a_{n-1} & b_{n-1} & c_{n-1} \\ 0 & \dots & 0 & 0 & a_n & b_n \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \vdots \\ \phi_{n-1} \\ \phi_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix}$$

TDMA

- Starting with a generalized forward elimination:

$$m(i) = \begin{cases} \dfrac{c_i}{b_i} & i = 1 \\[2em] \dfrac{c_i}{a_i m_{i-1}} & i = 2:N \end{cases}$$

$$d'_i = \begin{cases} \dfrac{d_i}{b_i} & i = 1 \\[2em] \dfrac{d_i - a_i d'_{i-1}}{b_i - a_i m_{i-1}} & i = 2:N \end{cases}$$

- This is followed by back-substitution:

$$\phi_N = d'_N \qquad i = N$$

$$\phi_i = d'_i - m_i \phi_{i+1} \qquad i = N-1:1$$

Example #3

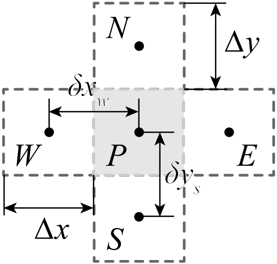
- Let us reconsider Example #1 from Lecture 2, where we have the following system of equations for 5 C.∀.:

$$\begin{bmatrix} 15 & -5 & 0 & 0 & 0 \\ -5 & 10 & -5 & 0 & 0 \\ 0 & -5 & 10 & -5 & 0 \\ 0 & 0 & -5 & 10 & -5 \\ 0 & 0 & 0 & -5 & 15 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \end{bmatrix} = \begin{bmatrix} 1,000 \\ 0 \\ 0 \\ 0 \\ 2,000 \end{bmatrix}$$

- Is there any computational benefit to using TDMA? See “L5Ex3.m” for the solution.

2D FVM

- Let us extend our analysis to two dimensions, but including north and south neighbors to P



- We note the unit depth is taken as unity

2D FVM - Steady Diffusion

- Considering the steady diffusion equation, including source terms:

$$\frac{\partial}{\partial x}\left(\Gamma\frac{\partial\phi}{\partial x}\right)+\frac{\partial}{\partial y}\left(\Gamma\frac{\partial\phi}{\partial y}\right)+S=0$$

- Let us integrate over the C.∀. 1:

$$\int_{C.\forall.}\left\{\frac{\partial}{\partial x}\left(\Gamma\frac{\partial\phi}{\partial x}\right)+\frac{\partial}{\partial y}\left(\Gamma\frac{\partial\phi}{\partial y}\right)+S=0\right\}$$

- Thus, integrating with respect to x, y and z:

$$\implies\left\{\left(\Gamma\frac{\partial\phi}{\partial x}\right)_e-\left(\Gamma\frac{\partial\phi}{\partial x}\right)_w\right\}\Delta y\Delta z+\left\{\left(\Gamma\frac{\partial\phi}{\partial y}\right)_n-\ldots\right.$$

$$\left.\ldots-\left(\Gamma\frac{\partial\phi}{\partial y}\right)_s\right\}\Delta x\Delta z+\bar{S}\Delta x\Delta y\Delta z=0$$

2D FVM - Steady Diffusion

- Assuming uniformly spaced cell-centers:

$$\frac{\Gamma_e \Delta y \Delta z}{\delta x_e}(\phi_E - \phi_P) - \frac{\Gamma_w \Delta y \Delta z}{\delta x_w}(\phi_P - \phi_W) + \ldots$$

$$\ldots + \frac{\Gamma_n \Delta x \Delta z}{\delta y_n}(\phi_N - \phi_P) - \frac{\Gamma_s \Delta x \Delta z}{\delta y_s}(\phi_P - \phi_S) + \bar{S}\Delta x \Delta y \Delta z = 0$$

- Expanding out the terms:

$$\frac{\Gamma_e \Delta y \Delta z}{\delta x_e}\phi_E - \frac{\Gamma_e \Delta y \Delta z}{\delta x_e}\phi_P - \frac{\Gamma_w \Delta y \Delta z}{\delta x_w}\phi_P + \frac{\Gamma_w \Delta y \Delta z}{\delta x_w}\phi_W + \ldots$$

$$\ldots + \frac{\Gamma_n \Delta x \Delta z}{\delta y_n}\phi_N - \frac{\Gamma_n \Delta x \Delta z}{\delta y_n}\phi_P - \frac{\Gamma_s \Delta x \Delta z}{\delta y_s}\phi_P + \frac{\Gamma_s \Delta x \Delta z}{\delta y_s}\phi_S + \ldots$$

$$\ldots + \bar{S}\Delta x \Delta y \Delta z = 0$$

2D FVM - Steady Diffusion

- Grouping like conserved quantities:

$$\phi_E\left(\frac{\Gamma_e \Delta y \Delta z}{\delta x_e}\right)+\phi_W\left(\frac{\Gamma_w \Delta y \Delta z}{\delta x_w}\right)+\phi_N\left(\frac{\Gamma_n \Delta x \Delta z}{\delta y_n}\right)+\ldots$$

$$\ldots+\phi_S\left(\frac{\Gamma_s \Delta x \Delta z}{\delta y_s}\right)+\phi_P\left(-\frac{\Gamma_e \Delta y \Delta z}{\delta x_e}-\frac{\Gamma_w \Delta y \Delta z}{\delta x_w}-\ldots\right.$$

$$\left.\ldots-\frac{\Gamma_n \Delta x \Delta z}{\delta y_n}-\frac{\Gamma_s \Delta x \Delta z}{\delta y_s}\right)+\bar{S} \Delta x \Delta y \Delta z=0$$

- We define the following coefficients:

$$a_e=\frac{\Gamma_e \Delta y \Delta z}{\delta x_e};\ a_w=\frac{\Gamma_w \Delta y \Delta z}{\delta x_w};\ a_n=\frac{\Gamma_n \Delta x \Delta z}{\delta y_n}$$

$$a_s=\frac{\Gamma_s \Delta x \Delta z}{\delta y_s};\ \bar{S}=S_c+S_P \phi_P;\ B=S_c \Delta x \Delta y \Delta z$$

2D FVM - Steady Diffusion

- Thus, the coefficient for C.∀. 1 is expressed as:

$$a_p = a_e + a_w + a_n + a_s - S_P \Delta x \Delta y \Delta z$$

- The transport equation is then expressed as:

$$a_p \phi_P = a_e \phi_E + a_w \phi_W + a_n \phi_N + a_s \phi_S + B$$

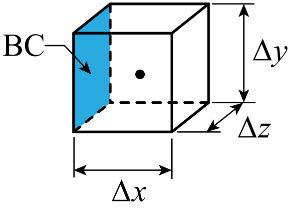
- Alternatively:

$$a_p \phi_P = \sum_{nb} a_{nb} \phi_{nb} + B$$

Boundary Conditions in 2D

- Boundary conditions are handled the same in 2D as in 1D:
- If ϕ=constant

$$a_b = \frac{\Gamma_b \Delta y \Delta z}{\frac{\delta x_b}{2}} = \frac{2\Gamma_b \Delta y \Delta z}{\delta x_b}$$

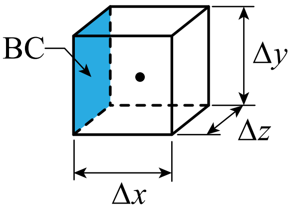


- The steps to include this into the solution are:
  1. $a_b\phi_b$ gets added to B
  2. a_b gets added to a_p
  3. Remove the coefficient in that direction (i.e. $a_w = 0$)

Boundary Conditions in 2D

- If J=constant

$$J = -\Gamma_b \frac{\partial \phi}{\partial x}\bigg|_b = \frac{2\Gamma_b \Delta y \Delta z}{\delta x_b}$$



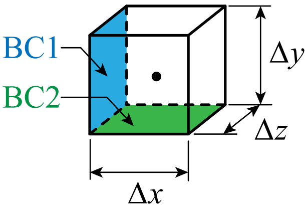
- The steps to include this into the solution are:
  1. $a_b = 0$
  2. $a_w = 0$
  3. $J\Delta y\Delta z$ get added to B

Boundary Conditions in 2D

- What is there are two boundary conditions on the C.∀. (i.e. corner)?
- We define two boundary coefficients:

$$a_{b_1} = \frac{2\Gamma_{B1}\Delta y \Delta z}{\delta x}$$

$$a_{b_2} = \frac{2\Gamma_{B2}\Delta x \Delta z}{\delta y}$$



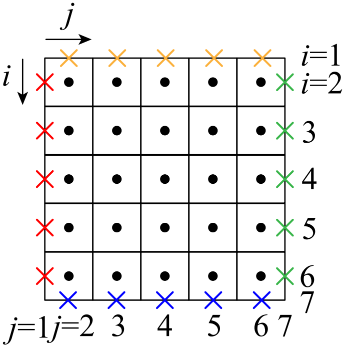
- Then we modify B such that:

$$B^* = B + a_{b_1}\phi_{B1} + a_{b_2}\phi_{B2}$$

- The same approach is taken for Neumann conditions

Example #1

- Say we have a 5x5 grid ($n = 5$), with cell-centers denoted by $\bullet$
- And there are 4 unique boundary conditions (constant ϕ), denoted by the colored $\times$, $\implies \phi(i, j)$=[7, 7]



Example #1

- Employing Jacobi iteration:
- for j=2:n+1
  - for i=2:n+1

$$\phi_P^{k+1} = \frac{1}{a_p}\left(a_w\phi_W^k + a_e\phi_E^k + a_s\phi_S^k + a_n\phi_N^k + B\right)$$

  - end
- end
- Employing Gauss-Seidel:
- for j=2:n+1
  - for i=2:n+1

$$\phi_P^{k+1} = \frac{1}{a_p}\left(a_w\phi_W^{k+1} + a_e\phi_E^k + a_{s+1}\phi_S^k + a_n\phi_N^k + B\right)$$

  - end
- end