Numerical Solutions of Finite–Volume Equations

Larry Caretto
Mechanical Engineering 692

Computational Fluid Dynamics

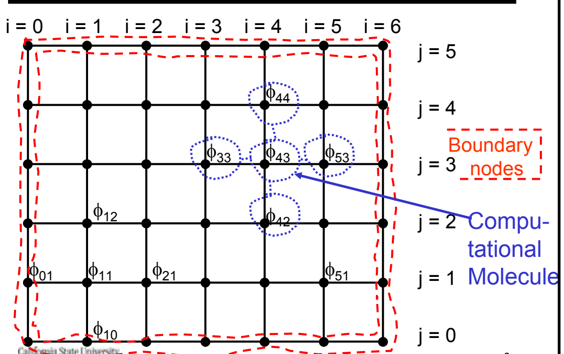
March 15–17, 2010

Equation to Be Solved

- $A_N\phi_N + A_S\phi_S + A_E\phi_E + A_P\phi_P + A_W\phi_W = Q_P$
- $A_N\phi_{iJ+1} + A_S\phi_{ij-1} + A_E\phi_{i+1j} + A_P\phi_{ij} + A_W\phi_{i-1j}$
- Have a set of simultaneous linear equation to be solved algebraically
- A_K coefficients different for u, v, p, but all equations seen here link central (P) node to 4 nearest neighbors
- Sparse matrix system, look at iterative methods for solution

 2

A Small Grid (N = 6, M = 5)



 3

Grid i,j Notation

- For system typically use this notation in combination with compass points
- Notation A_{ij}^{point} is general coefficient
  - ij refers to a particular node
  - Point = N(orth), S(outh), E(ast), W(est) refers to neighboring nodes by direction
- General equation shown below

$$A_{ij}^S\phi_{ij-1} + A_{ij}^W\phi_{i-1j} + A_{ij}^P\phi_{ij} + A_{ij}^E\phi_{i+1j} + A_{ij}^N\phi_{ij+1} = b_{ij}$$

$$A_{ij}^P = -A_{ij}^S - A_{ij}^W - A_{ij}^E - A_{ij}^N \qquad A_{ij}^S = A_{ij-1}^N \qquad A_{ij}^W = A_{i-1j}^E$$

 4

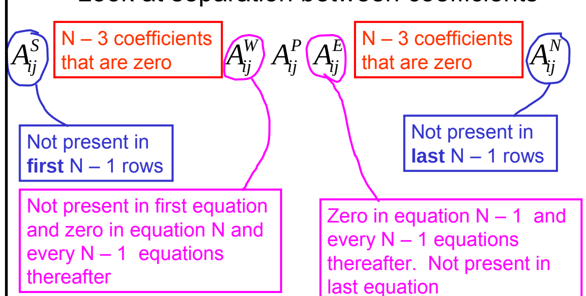
Solving the Equations

- Typically have large number of equations forming sparse matrix
  - For $\Delta x = \Delta y = .01$ have 99^2 equations so matrix has $96\text{x}10^6$ potential coefficients
  - Only 48609 (0.051%) are nonzero
- Want data structure and algorithm for handling sparse matrices
- Gauss elimination uses storage for banded matrices
- Iterative methods used for solutions

 5

General Equation in a Matrix

- Look at separation between coefficients



 6

Sparse Matrix Structure

- 20 equations can have 400 coefficients
- Here each equation has no more than five coefficients (100 possible)
- Boundaries give another 2(N + M - 2) zero coefficients (18 in this example)
- Thus, we have 82 nonzero coefficients and 400 – 82 = 318 zeros in matrix
  - Nearly 80% of coefficients are zero
  - Fraction increases for larger grids

How Sparse is the Matrix?

- The M by N grid has (M – 1)(N – 1) nodes with equations giving $(M-1)^2(N-1)^2$ possible coefficients
- Without boundaries we have only 5 (M – 1)(N – 1) nonzero coefficients
- Boundaries give 2(N + M – 2) = 2(M – 1) + 2(N – 1) additional zero coefficients

$$\begin{bmatrix}Nonzero\\Fraction\end{bmatrix} = \frac{5(N-1)(M-1) - 2(N-1) - 2(M-1)}{(N-1)^2(M-1)^2} = \frac{5}{(N-1)(M-1)} - \frac{2}{(N-1)(M-1)^2} - \frac{2}{(N-1)^2(M-1)}$$

What Makes Sparseness?

- Each node is connected only to a small number of nearest neighbors
  - Problem here has four neighbors
  - Higher order schemes and 3D finite-volume equation can have more neighbors
- Can have complex coefficients so long as number of neighbors is limited
- Convection-diffusion coefficients with uneven grid spacing are an example of complex coefficients in a sparse matrix

Iterative Solutions

- Simplest examples are Jacobi, Gauss-Seidel, and Successive Over Relaxation
- Move from iteration n to iteration n+1
- Iteration 0 is initial guess (often all zero)
- Straightforward approach: solve equation for ϕ_{ij} and use this as basis for iteration

$$\phi_{ij} = \frac{b_{ij} - A_{ij}^S \phi_{ij-1} - A_{ij}^W \phi_{i-1j} - A_{ij}^E \phi_{i+1j} - A_{ij}^N \phi_{ij+1}}{A_{ij}^P}$$

$$\phi_{ij} = b_{ij}' - A_{ij}^{S'} \phi_{ij-1} - A_{ij}^{W'} \phi_{i-1j} - A_{ij}^{E'} \phi_{i+1j} - A_{ij}^{N'} \phi_{ij+1}$$

Iterative Solutions II

- Use superscript (n) for iteration number
- Jacobi iteration uses all old values

$$\phi_{ij}^{(n+1)} = b_{ij}' - A_{ij}^{S'} \phi_{ij-1}^{(n)} - A_{ij}^{W'} \phi_{i-1j}^{(n)} - A_{ij}^{E'} \phi_{i+1j}^{(n)} - A_{ij}^{N'} \phi_{ij+1}^{(n)}$$

- Gauss–Seidel uses most-recent values

$$\phi_{ij}^{(n+1)} = b_{ij}' - A_{ij}^{S'} \phi_{ij-1}^{(n+1)} - A_{ij}^{W'} \phi_{i-1j}^{(n+1)} - A_{ij}^{E'} \phi_{i+1j}^{(n)} - A_{ij}^{N'} \phi_{ij+1}^{(n)}$$

- Relaxation basis: Gauss–Seidel provides a correction that can be adjusted

$$\phi_{ij}^{(n+1)} = \phi_{ij}^{(n)} + \omega\left[\phi_{ij}^{(n+1,GS)} - \phi_{ij}^{(n)}\right]$$

Relaxation Factor

Example Problem

- Look at simple system of equations
  - Could solve exactly to find x = 1, y = 2
  - Use to illustrate iteration

Original system: $6x + y = 8$, $2x + 5y = 12$

Iteration Form: $x = \frac{8-y}{6}$, $y = \frac{12-2x}{5}$

- Jacobi – general form and first steps

$$x^{(n+1)} = \frac{8 - y^{(n)}}{6} \qquad x^{(0)} = 0 \qquad x^{(1)} = \frac{8 - y^{(0)}}{6} = \frac{8}{6}$$

$$y^{(n+1)} = \frac{12 - 2x^{(n)}}{5} \qquad y^{(0)} = 0 \qquad y^{(1)} = \frac{12 - 2x^{(0)}}{5} = \frac{12}{5}$$

Jacobi Example Continued

$$x^{(2)} = \frac{8 - y^{(1)}}{6} = \frac{8 - 12/5}{6} = \frac{28}{30}$$
$$y^{(2)} = \frac{12 - 2x^{(1)}}{5} = \frac{12 - 2(8/6)}{5} = \frac{56}{30}$$

$$x^{(3)} = \frac{8 - y^{(2)}}{6} = \frac{8 - 56/30}{6} = \frac{184}{180}$$
$$y^{(3)} = \frac{12 - 2x^{(2)}}{5} = \frac{12 - 2(28/30)}{5} = \frac{304}{150}$$

- What is next iteration?
- How do we know we're finished?

Concluding Iterations

- In general, do not know correct answers
- Two common measures
  - Residual: $r_i = \Sigma_j a_{ij} x_j - b_i$
  - Difference in one iteration $|x_i^{(n+1)} - x_i^{(n)}|$
- Can use relative or absolute measure
- Need vector norm such as maximum absolute value or root mean square
- Look at summary of iterations for Jacobi

Jacobi Iteration History

n	x_n	y_n	x residual	y residual	x change	y change
0	0	0	8	12		
1	1.33333	2.4	-2.4	-2.66667	1.333333	2.4
2	0.93333	1.86667	0.533333	0.8	-0.4	-0.53333
3	1.02222	2.02667	-0.16	-0.17778	0.088889	0.16
4	0.99556	1.99111	0.035556	0.053333	-0.02667	-0.03556
5	1.00148	2.00178	-0.01067	-0.01185	0.005926	0.010667
6	0.9997	1.99941	0.00237	0.003556	-0.00178	-0.00237
7	1.0001	2.00012	-0.00071	-0.00079	0.000395	0.000711
8	0.99998	1.99996	0.000158	0.000237	-0.00012	-0.00016
9	1.00001	2.00001	-4.7E-05	-5.3E-05	2.63E-05	4.74E-05

Gauss-Seidel Iteration

- Apply Gauss-Seidel Iteration to same set of equations

Original system
$$6x + y = 8 \qquad 2x + 5y = 12$$

Iteration Form
$$x = \frac{8 - y}{6} \qquad y = \frac{12 - 2x}{5}$$

- Gauss-Seidel – general iteration form and first step (uses most recent values)

$$x^{(n+1)} = \frac{8 - y^{(n)}}{6} \qquad y^{(n+1)} = \frac{12 - 2x^{(n+1)}}{5}$$
$$x^{(0)} = 0 \qquad y^{(0)} = 0$$
$$x^{(1)} = \frac{8 - y^{(0)}}{6} = \frac{8}{6} \qquad y^{(1)} = \frac{12 - 2x^{(1)}}{5} = \frac{12}{5} - \frac{2}{5}\frac{8}{6} = \frac{56}{30}$$

Gauss-Seidel Iteration II

$$x^{(2)} = \frac{8 - y^{(1)}}{6} = \frac{8 - 56/30}{6} = \frac{184}{180}$$
$$y^{(2)} = \frac{12 - 2x^{(2)}}{5} = \frac{12 - 2(184/180)}{5} = \frac{1792}{900}$$

$$x^{(3)} = \frac{8 - y^{(2)}}{6} = \frac{8 - 1792/900}{6} = \frac{5408}{5400} = 1.00148\ldots$$
$$y^{(3)} = \frac{12 - 2x^{(3)}}{5} = \frac{12 - 2(5408/5400)}{5} = 1.9941\ldots$$

- Faster convergence in Gauss-Seidel

Relaxation Methods

- Relaxation factor, ω, greater than or less than 1 is over- or underrelaxation
  - Underrelaxation procures stability in problems that will not converge
  - Overrelaxation procures speed in well-behaved problems

$$\phi_{ij}^{(n+1)} = \phi_{ij}^{(n)} + \omega\left[\phi_{ij}^{(n+1,GS)} - \phi_{ij}^{(n)}\right] = (1-\omega)\phi_{ij}^{(n)} + \omega\phi_{ij}^{(n+1,GS)} = (1-\omega)\phi_{ij}^{(n)} - \omega\Big[-b_{ij}^{'} + A_{ij}^{S'}\phi_{ij-1}^{(n+1)} + A_{ij}^{W'}\phi_{i-1j}^{(n+1)} + A_{ij}^{E'}\phi_{i+1j}^{(n)} + A_{ij}^{N'}\phi_{ij+1}^{(n)}\Big]$$

Relaxation Code (f is ϕ)

```
do iter = 1, maxIter
  maxResid = 0
  do i = 1, N – 1
    do j = 1, M – 1
      old = f(i,j)
      f(i,j) = (omega – 1) * f(i,j)
        + omega * ( AN(i,j) * f(i,j+1)
        + AE(i,j) * f(i+1,j) + AS(i,j)
        * f(i,j-1) + AW(i,j) * f(i-1,j)
        - b(i,j) )
      resid = abs( ( f(i,j) – old ) /
          f(i,j) )
      if ( resid > maxResid ) then
        maxResid = resid;
      end if
```

One set of iterations (omitted on next page)

19

Relaxation Code II

```
do iter = 1, maxIter
  maxResid = 0;
  do i = 1, N - 1
    do j = 1, M - 1
       !compute new f(i,j) and maxResid
    end do
  end do
  if ( maxResid <= errTol ) exit
end do
if ( maxResid > errTol ) then
  print *, "Not converged"
else
  call doOutput( f, N, M )
end if
```

20

Converging Iterations

- Have three different "solutions"
  - Correct solution to differential equation
  - Exact solution to finite-difference equations
  - Current and previous iteration values
- Iterations should approach correct solution to finite-difference equations
- Since neither correct solution is known, we use norm of error estimates
  - Residual in finite-difference equations
  - Change in iteration value

21

Converging Iterations II

- At each grid node we can compute a relative change or a residual
  - Both are zero at convergence

$$\begin{bmatrix}\text{Relative} \\ \text{Change}\end{bmatrix}_{ij} = \frac{\phi_{ij}^{(n+1)} - \phi_{ij}^{(n)}}{\phi_{ij}^{(n+1)}}$$

$$[\text{Residual}]_{ij} = \phi_{ij}^{(n+1)} + A_{ij}^{S'}\phi_{ij-1}^{(n+1)} + A_{ij}^{W'}\phi_{i-1j}^{(n+1)} + A_{ij}^{E'}\phi_{i+1j}^{(n)} + A_{ij}^{N'}\phi_{ij+1}^{(n)} - b_{ij}'$$

22

Converging Iterations III

- Need some overall measure of convergence error
- Consider error (relative change or residual) at each point as one component of a vector
- Use vector norm for overall error
  - Maximum absolute value (zero norm)
  - Root mean squared error (two norm)

$$\varepsilon_{overall} = \sqrt{\frac{\sum_{all\ nodes} \varepsilon_{node}^2}{N_{nodes}}}$$

23

Simple Numerical Example

- Look at simple, two-dimensional case with diffusion only (velocities are zero)
- Dirichlet (fixed ϕ) boundary conditions
- Use finite-volume equation from original work on diffusion with a source term
- Set source term to zero and use constant grid sizes and Γ
- Solve finite-volume equation for this case (**v** = **0**, Δx, Δy fixed, constant Γ)

24

v = 0, Δx, Δy, Γ constant

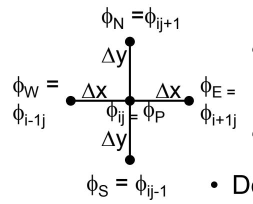
$$\left[\Gamma_e^{(\phi)}\frac{\phi_E-\phi_P}{x_E-x_P}\right]\Delta y+\left[\Gamma_n^{(\phi)}\frac{\phi_N-\phi_P}{y_N-y_P}\right]\Delta x+$$
$$\left[\Gamma_w^{(\phi)}\frac{\phi_W-\phi_P}{x_P-x_W}\right]\Delta y+\left[\Gamma_s^{(\phi)}\frac{\phi_S-\phi_P}{y_P-y_S}\right]\Delta x=0$$

- Divide by ΓΔy/Δx

$$\phi_E-\phi_P+(\phi_N-\phi_P)\left(\frac{\Delta x}{\Delta y}\right)^2+\phi_W-\phi_P+(\phi_S-\phi_P)\left(\frac{\Delta x}{\Delta y}\right)^2=0$$

Northridge 25

v = 0, Δx, Δy, Γ constant II



$$\phi_E-\phi_P+(\phi_N-\phi_P)\left(\frac{\Delta x}{\Delta y}\right)^2+$$
$$\phi_W-\phi_P+(\phi_S-\phi_P)\left(\frac{\Delta x}{\Delta y}\right)^2=0$$

- Define β = Δx/Δy and rearrange terms

$$\phi_E+\phi_W+\beta^2(\phi_N+\phi_S)-2(1+\beta^2)\phi_P=0$$
$$\phi_{i+1j}+\phi_{i-1j}+\beta^2(\phi_{ij+1}+\phi_{ij-1})-2(1+\beta^2)\phi_{ij}=0$$

Northridge 26

Finite-difference Equation

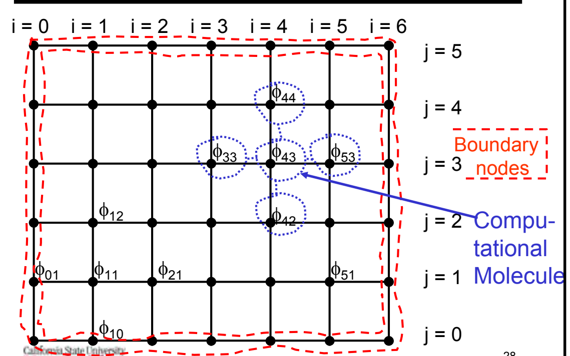
- Finite-volume form typical of two-dimensional Laplace equation
- If β = Δx/Δy = 1, ϕ_{ij} is the average of its four nearest neighbors

$$\phi_{ij-1}+\phi_{i-1j}-4\phi_{ij}+\phi_{i+1j}+\phi_{ij+1}=0$$

- Consider Dirichlet boundary conditions
  – ϕ_{ij} known at all boundary nodes
  – Need to find (N – 1)(M – 1) unknown values of ϕ_{ij} on grid

Northridge 27

A Small Grid (N = 6, M = 5)



Northridge 28

Grid Equations (β = 1)

$$\begin{array}{lllllllllll}
-4\phi_{11} & +\phi_{21} & & & & +\phi_{12} & & & & & =-\phi_{10}-\phi_{01}\\
\phi_{11} & -4\phi_{21} & +\phi_{31} & & & & +\phi_{22} & & & & =-\phi_{20}\\
& \phi_{21} & -4\phi_{31} & +\phi_{41} & & & & +\phi_{23} & & & =-\phi_{30}\\
& & \phi_{31} & -4\phi_{41} & +\phi_{51} & & & & +\phi_{24} & & =-\phi_{40}\\
& & & \phi_{41} & -4\phi_{51} & & & & & +\phi_{25} & =-\phi_{40}-\phi_{61}\\
\phi_{11} & & & & & -4\phi_{12} & +\phi_{22} & & & +\phi_{31} & =-\phi_{02}\\
& \phi_{21} & & & & +\phi_{12} & -4\phi_{22} & +\phi_{23} & & +\phi_{32} & =0\\
& & \phi_{31} & & & & +\phi_{22} & -4\phi_{23} & +\phi_{24} & +\phi_{33} & =0
\end{array}$$

- N = 6 and M = 5 gives (6 – 1)(5 – 1) = 20 equations – only eight shown
- Diagonal structure incorrect here

Northridge 29

Execution Times and Errors

- Examine square region with zero boundary conditions at x = 0, x = x_{max}, and y = 0; two cases for y = y_{max}
  – Case 1: constant value of ϕ_N(x) = 1
  – Case 2: ϕ_N(x) = sin(πx)
- First case has discontinuity for y = y_{max} at x = 0 and x = x_{max}
- Use overrelaxation (SOR) with variable relaxation factors

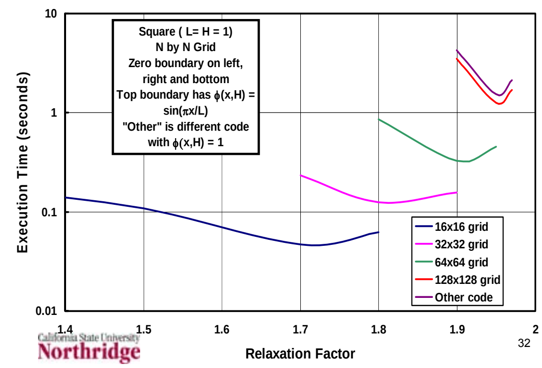
Northridge 30

Execution Times and Errors II

- Iterate until maximum iteration difference in ϕ_{ij} is about machine error
  - Case 1: constant value of $\phi_N(x) = 1$
  - Case 2: $\phi_N(x) = \sin(\pi x)$
- First case has discontinuity for $y = y_{max}$ at $x = 0$ and $x = x_{max}$
- Compare solutions to exact solution of differential equation and exact solution of finite difference equations

31

Effect of Relaxation Factor on Execution Time



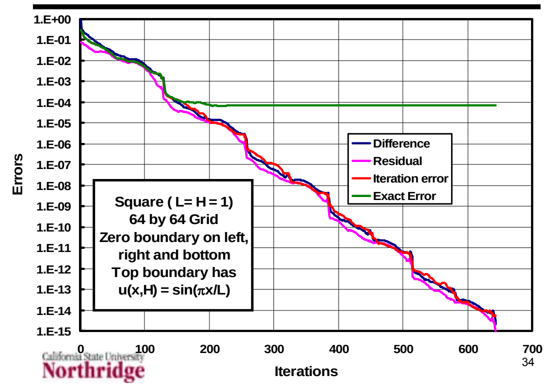
32

Effect of Iterations on Errors

- Compare three error measures using the maximum value on the grid
  - True iteration error: difference between the current value and the value found by an exact solution of the ***difference*** equations
  - Difference in ϕ_{ij} between two iterations
  - Residual = $\phi_{i+1j} + \phi_{i-1j} + \phi_{ij+1} + \phi_{ij-1} - 4\phi_{ij}$
  - Exact error is difference between iteration value and exact solution of ***differential*** equation

33

Effects of Iterations on Laplace Equation Errors



34

Will Iterations Converge?

- How do we ensure that an iterative process converges?
- Look at general example of solving a system of simultaneous equations by iteration
- Write equation in matrix form $\mathbf{A}\phi = \mathbf{b}$
- Develop general iteration algorithm in matrix form
- Look at criterion for error to decrease

35

Matrix Equation Form

- Advanced solution techniques treat matrix for finite-difference equations
- Leads to dimensional confusion
  - Start with 2D grid (x and y indices)
  - Treat as matrix equation where unknowns ϕ_{ij} form a column vector (one-dimensional)
  - The coefficients in the matrix form a two-dimensional display
  - Examine small grid example
  - Take $A_E = A_W = A_N = A_S = 1$, $A_P = -4$

36

General Matrix Structure

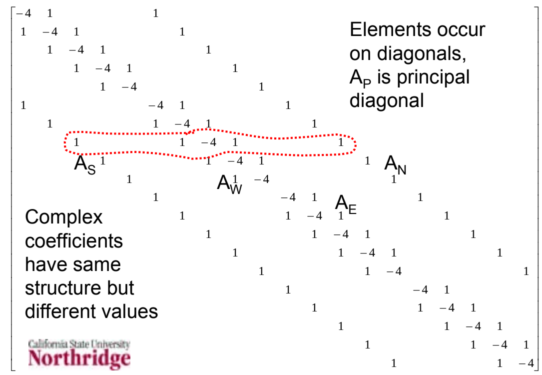
- Confusion about two two-dimensional representations
- Grid has two space dimensions with (N – 1)(M – 1) unknown nodes
- ϕ_{ij} forms a one dimensional column matrix of unknowns (at right)
- Coefficient matrix has five diagonals
- Right-hand side has boundary values

$$A^S_{ij}\phi_{ij-1} + A^W_{ij}\phi_{i-1j} + A^P_{ij}\phi_{ij} + A^E_{ij}\phi_{i+1j} + A^N_{ij}\phi_{ij+1} = b_{ij}$$

$$\begin{bmatrix} \phi_{11} \\ \phi_{12} \\ \phi_{13} \\ \phi_{14} \\ \phi_{15} \\ \phi_{21} \\ \phi_{22} \\ \phi_{23} \\ \vdots \\ \phi_{45} \end{bmatrix}$$

Northridge 37

Previous Solution Matrix



Northridge

General Solution Matrix, A

- Like the one on the previous chart
  - Has more rows for more grid points
  - Coefficients may not be the same
  - Will be generally sparse
  - Has regular structure for simple grids
  - Unstructured grids do not give simple structure, but keep a sparse matrix
  - We want to solve **A**ϕ = **b** where ϕ is a vector of all the unknowns on the grid

Northridge 39

General Iteration Approaches

- We want to solve **A**ϕ = **b** by iteration
- As the solution to the finite-difference equations, ϕ has truncation error even with perfect iteration solution
- Define iteration error as $\varepsilon^{(n)} = \phi - \phi^{(n)}$
- Define residual, $\mathbf{r}^{(n)} = \mathbf{b} - \mathbf{A}\phi^{(n)}$
- Combine equations: $\mathbf{r}^{(n)} = \mathbf{b} - \mathbf{A}\phi^{(n)} = \mathbf{A}\phi - \mathbf{A}\phi^{(n)} = \mathbf{A}(\phi - \phi^{(n)}) = \mathbf{A}\varepsilon^{(n)}$
- $\mathbf{r}^{(n)} = \mathbf{A}\varepsilon^{(n)}$ relates computable $\mathbf{r}^{(n)}$ to $\varepsilon^{(n)}$ that we want to control but can't compute

Northridge 40

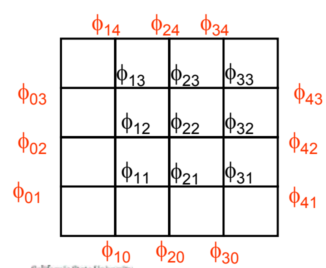
General Iteration Approaches II

- One iteration step takes the old values, $\phi^{(n)}$, to the new values, $\phi^{(n+1)}$
- General iteration: $\mathbf{M}\phi^{(n+1)} = \mathbf{N}\phi^{(n)} + \mathbf{b}$
- Methods select **M** and **N** to accelerate convergence of iterations
- At convergence, $\phi^{(n+1)} = \phi^{(n)} = \phi$, so that $\mathbf{M}\phi^{(n+1)} = \mathbf{N}\phi^{(n)} + \mathbf{b}$ is $(\mathbf{M} - \mathbf{N})\phi = \mathbf{b}$
- We are solving **A**ϕ = **b**, so we must have **M** – **N** = **A**

Northridge 41

Example of M and N Matrices

- Heat conduction with constant properties and no source term with $N_x = N_y = 4$

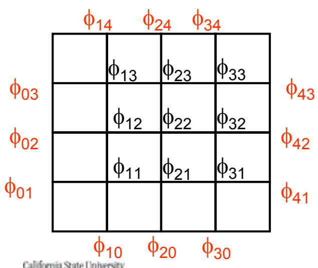


- System of equations with nine unknowns
- Boundary values known
- Solve **A**ϕ = **b**

Northridge 42

Example of **M** and **N** Matrices II

- Iterate Gauss Seidel from lower left to upper right using newest values



$$\phi_{ij-1}^{(n+1)} + \phi_{i-1j}^{(n+1)} + \phi_{ij}^{(n+1)} + \phi_{i+1j}^{(n)} + \phi_{ij+1}^{(n)} = 0$$

When solving for ϕ_{22} we will have current iteration values for ϕ_{12} and ϕ_{21}

43

Aϕ = b and M$\phi^{(n+1)}$ = N$\phi^{(n)}$ + b

$$\begin{bmatrix} -4 & 1 & & 1 & & & & & \\ 1 & -4 & 1 & & 1 & & & & \\ & 1 & -4 & & & 1 & & & \\ 1 & & & -4 & 1 & & 1 & & \\ & 1 & & 1 & -4 & 1 & & 1 & \\ & & 1 & & 1 & -4 & & & 1 \\ & & & 1 & & & -4 & 1 & \\ & & & & 1 & & 1 & -4 & 1 \\ & & & & & 1 & & 1 & -4 \end{bmatrix} \begin{bmatrix} \phi_{11} \\ \phi_{21} \\ \phi_{31} \\ \phi_{12} \\ \phi_{22} \\ \phi_{32} \\ \phi_{13} \\ \phi_{23} \\ \phi_{33} \end{bmatrix} = \begin{bmatrix} -\phi_{10}-\phi_{01} \\ -\phi_{20} \\ -\phi_{30}-\phi_{41} \\ -\phi_{02} \\ 0 \\ -\phi_{42} \\ -\phi_{03}-\phi_{14} \\ -\phi_{24} \\ -\phi_{43}-\phi_{34} \end{bmatrix}$$

GS **M** matrix

GS **–N** matrix

$$\phi_{ij-1}^{(n+1)} + \phi_{i-1j}^{(n+1)} - 4\phi_{ij}^{(n+1)} = -\phi_{i+1j}^{(n)} - \phi_{ij+1}^{(n)} + b_{ij} \qquad b_{ij} = 0$$

44

M Matrix for SOR

$$\begin{bmatrix} -4 & & & & & & & & \\ \omega & -4 & & & & & & & \\ & \omega & -4 & & & & & & \\ \omega & & & -4 & & & & & \\ & \omega & & \omega & -4 & & & & \\ & & \omega & & \omega & -4 & & & \\ & & & \omega & & & -4 & & \\ & & & & \omega & & \omega & -4 & \\ & & & & & \omega & & \omega & -4 \end{bmatrix} \begin{bmatrix} \phi_{11} \\ \phi_{21} \\ \phi_{31} \\ \phi_{12} \\ \phi_{22} \\ \phi_{32} \\ \phi_{13} \\ \phi_{23} \\ \phi_{33} \end{bmatrix}^{(n+1)} =$$

$$\omega\phi_{ij-1}^{(n+1)} + \omega\phi_{i-1j}^{(n+1)} - 4\phi_{ij}^{(n+1)} = -\omega\phi_{i+1j}^{(n)} - \omega\phi_{ij+1}^{(n)} + (4\omega - 4)\phi_{ij}^{(n)} = 0$$

45

N Matrix for SOR

$$\begin{bmatrix} d & -\omega & & -\omega & & & & & \\ & d & -\omega & & -\omega & & & & \\ & & d & & & -\omega & & & \\ & & & d & -\omega & & -\omega & & \\ & & & & d & -\omega & & -\omega & \\ & & & & & d & & & -\omega \\ & & & & & & d & -\omega & \\ & & & & & & & d & -\omega \\ & & & & & & & & d \end{bmatrix} \begin{bmatrix} \phi_{11} \\ \phi_{21} \\ \phi_{31} \\ \phi_{12} \\ \phi_{22} \\ \phi_{32} \\ \phi_{13} \\ \phi_{23} \\ \phi_{33} \end{bmatrix}^{(n)} - \omega \begin{bmatrix} \phi_{10}+\phi_{01} \\ \phi_{20} \\ \phi_{30}+\phi_{41} \\ \phi_{02} \\ 0 \\ \phi_{42} \\ \phi_{03}+\phi_{14} \\ \phi_{24} \\ \phi_{43}+\phi_{34} \end{bmatrix}$$

$$d = 4\omega - 4$$

$$\omega\phi_{ij-1}^{(n+1)} + \omega\phi_{i-1j}^{(n+1)} - 4\phi_{ij}^{(n+1)} = -\omega\phi_{i+1j}^{(n)} - \omega\phi_{ij+1}^{(n)} + (4\omega - 4)\phi_{ij}^{(n)} = 0$$

46

Next Steps

- Look at general iteration equation: **M**$\phi^{(n+1)}$ = **N**$\phi^{(n)}$ + **b**
- Get equation for evolution of error vector, $\varepsilon^{(n)}$, representing error in each unknown ϕ at step n
- How does error at new step, $\varepsilon^{(n+1)}$ depend on error at old step, $\varepsilon^{(n)}$
- How can we guarantee that error does not grow at each step?

47

Convergence

- Start with **M**$\phi^{(n+1)}$ = **N**$\phi^{(n)}$ + **b**
- Subtract **M**$\phi^{(n)}$ from each side
- Result is **M**($\phi^{(n+1)}$ – $\phi^{(n)}$) = **b** – (**M** – **N**)$\phi^{(n)}$
- But we said that **M** – **N** = **A**, so result is
- **M($\phi^{(n+1)}$ – $\phi^{(n)}$) = b – (M – N)$\phi^{(n)}$ = b – A$\phi^{(n)}$**
- We defined **b – A$\phi^{(n)}$** = **r**$^{(n)}$ = **A**$\varepsilon^{(n)}$, so
- **M($\phi^{(n+1)}$ – $\phi^{(n)}$) = b – A$\phi^{(n)}$ = r$^{(n)}$ = A$\varepsilon^{(n)}$**

48

Convergence II

- From last chart: $\mathbf{M}(\phi^{(n+1)} - \phi^{(n)}) = \mathbf{r}^{(n)} = \mathbf{A}\varepsilon^{(n)}$
- Define update = $\delta^{(n)} = \phi^{(n+1)} - \phi^{(n)}$
- $\mathbf{M}(\phi^{(n+1)} - \phi^{(n)}) = \mathbf{M}\delta^{(n)} = \mathbf{r}^{(n)} = \mathbf{A}\varepsilon^{(n)}$
- Have two computable measures of error, $\varepsilon^{(n)}$; these are $\delta^{(n)}$ and $\mathbf{r}^{(n)}$
- What makes error decrease?

California State University Northridge 49

Does the Error Decrease?

- Iteration equation: $\mathbf{M}\phi^{(n+1)} = \mathbf{N}\phi^{(n)} + \mathbf{b}$
- At convergence, $\phi^{(n+1)} = \phi^{(n)} = \phi$, so that $\mathbf{M}\phi = \mathbf{N}\phi + \mathbf{b}$
- Subtract $\mathbf{M}\phi = \mathbf{N}\phi + \mathbf{b}$ from $\mathbf{M}\phi^{(n+1)} = \mathbf{N}\phi^{(n)} + \mathbf{b}$ giving $\mathbf{M}(\phi^{(n+1)} - \phi) = \mathbf{N}(\phi^{(n)} - \phi)$, which gives $\mathbf{M}\varepsilon^{(n+1)} = \mathbf{N}\varepsilon^{(n)}$
- New error given by $\varepsilon^{(n+1)} = \mathbf{M}^{-1}\mathbf{N}\varepsilon^{(n)}$
- Does the error go to zero as we take more iterations?

California State University Northridge 50

Matrix Eigenvalues: $\mathbf{Ax} = \lambda\mathbf{x}$

- Used to determine convergence
- If a matrix, **A**, multiplies a vector **x** and produces a constant λ times **x**
  - **x** is an eigenvector of **A**
  - λ is the eigenvalue associated with **x**
  - An n by n matrix can have up to n linearly independent eigenvalues
  - If the n eigenvectors are linearly independent we can expand any n component vector in terms of the eigenvectors

California State University Northridge 51

Error Decrease Depends on $\mathbf{M}^{-1}\mathbf{N}$

- Assume that $\mathbf{M}^{-1}\mathbf{N}$ has a complete set of K eigenvalues, $\mathbf{x}_{(k)}$ so we can expand the initial error vector in terms of these eigenvectors $\varepsilon^{(0)} = \Sigma_k a_k\mathbf{x}_{(k)}$
- Iteration process gives following results where λ_k is eigenvalue ($\mathbf{M}^{-1}\mathbf{N}\mathbf{x}_{(k)} = \lambda_k\mathbf{x}_{(k)}$)

$$\boldsymbol{\varepsilon}^{(1)} = \mathbf{M}^{-1}\mathbf{N}\boldsymbol{\varepsilon}^{(0)} = \mathbf{M}^{-1}\mathbf{N}\sum_{k=1}^{K} a_k\mathbf{x}_{(k)} = \sum_{k=1}^{K} a_k\mathbf{M}^{-1}\mathbf{N}\mathbf{x}_{(k)} = \sum_{k=1}^{K} a_k\lambda_k\mathbf{x}_{(k)}$$

$$\boldsymbol{\varepsilon}^{(2)} = \mathbf{M}^{-1}\mathbf{N}\boldsymbol{\varepsilon}^{(1)} = \mathbf{M}^{-1}\mathbf{N}\sum_{k=1}^{K} a_k\lambda_k\mathbf{x}_{(k)} = \sum_{k=1}^{K} a_k\lambda_k\mathbf{M}^{-1}\mathbf{N}\mathbf{x}_{(k)} = \sum_{k=1}^{K} a_k\lambda_k^2\mathbf{x}_{(k)}$$

California State University Northridge 52

General Error Equation

- Reasoning by induction from the last two equations gives $\varepsilon^{(n)}$ as follows

$$\boldsymbol{\varepsilon}^{(n)} = \sum_{k=1}^{K} a_k\lambda_k^n\mathbf{x}_{(k)}$$

- For error to become small as iterations increase, we must have all $|\lambda_k| < 1$
  - Largest $|\lambda_k| = |\lambda_1|$, called spectral radius, will dominate sum for large n
  - $\varepsilon^{(n)} \approx a_1\lambda_1^n\mathbf{x}_{(1)}$
  - Want this to reach desired error, δ

California State University Northridge 53

General Error Equation II

- To control error in $\varepsilon^{(n)} \approx a_1\lambda_1^n\mathbf{x}_{(1)}$ require factor $a_1\lambda_1^n \approx \delta$ or $\lambda_1^n = \delta/a_1$
- Take logs of both sides and solve for n

$$n \approx \frac{\ln\left(\dfrac{\delta}{a_1}\right)}{\ln\lambda_1}$$

- Recall that ln(1 + x) ≈ x for small x
- When λ_1 is close to 1, ln λ_1 will be a small number and n will be large
- Seek iteration matrices with small λ_1

California State University Northridge 54

SOR Spectral Radius

- Use MATLAB to compute the spectral radius, λ_1 = maximum $|\lambda|$ for SOR
  - Find optimum ω (minimum λ_1) by trial and error

N = M = 11		N = M = 21		N = M = 41	
ω	λ_1	ω	λ_1	ω	λ_1
1	0.9206	1	0.9778	1	0.9941
1.56	0.5759	1.74	0.7562	1.85	0.8968
1.57	0.5700	1.75	0.7500	1.86	0.8600
1.58	0.5800	1.76	0.7600	1.87	0.8700

California State University Northridge 55

Diagonal Dominance

- The real requirement is that the largest eigenvalue be less than one in absolute value
- This is guaranteed in the solution matrix is diagonally dominant
- This means that the diagonal coefficient (in absolute value) is greater than or equal to the sum of the absolute values of all other coefficients in the equation

California State University Northridge 56

Diagonal Dominance II

- If the coefficients in the **A** matrix are a_{km}, the rules for diagonal dominance in an N x N matrix are
- $|a_{kk}| >= \Sigma_m |a_{km}|$ for $1 \le k \le N$
- $|a_{kk}| > \Sigma_m |a_{km}|$ for at least one value of k
- General finite-difference equations satisfy the >= condition and boundary conditions satisfy the > condition
- Upwind difference diagonally dominant

California State University Northridge 57

Advanced Methods

- See text for greater discussion
- Methods use different iteration matrices to get faster convergence
  - Alternating Direction Implicit (ADI)
  - Stone's Method
  - Conjugate Gradient
  - Multigrid
- Multigrid generally considered fastest method for CFD calculations

California State University Northridge 58

Multigrid Method

- Solve equations on a set of different grids
- Analysis of error shows that convergence rate depends on grid size
- Getting a solution to a coarse grid then using those results for the fine grid gives solution faster
- Use prolongation and restriction to get results between fine and coarse gride

California State University Northridge 59

Multigrid Method II

- Different patterns used; example below
  - Start with fine grid
  - After partial convergence on find grid use coarser grid and do iterations to get more convergence on that grid
  - Continue to coarsest grid and get convergence there
  - Prolong solution to finer grids and get converged solution on each grid
  - Finally get converged solution on finest grid

California State University Northridge 60

Thomas Algorithm

- Used for simple solution of one-dimensional problems
- Can be extended to improved iteration approach for two- or three-dimensional problems
- Basic problem: $a_W\phi_W + a_P\phi_P + a_E\phi_E = b_K$
- Generalized one-dimensional problem
  - $A_kf_{k-1} + B_kf_k + C_kf_{k+1} = D_k$
  - Look at matrix form

California State University Northridge 61

Thomas Algorithm II

- General format for tridiagonal equations

$$\begin{bmatrix} B_0 & C_0 & 0 & 0 & \cdots & 0 & 0 \\ A_1 & B_1 & C_1 & 0 & \cdots & 0 & 0 \\ 0 & A_2 & B_2 & C_2 & \cdots & 0 & 0 \\ 0 & 0 & A_3 & B_3 & & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & B_{N-1} & C_{N-1} \\ 0 & 0 & 0 & 0 & \cdots & A_N & B_N \end{bmatrix} \begin{bmatrix} \phi_0 \\ \phi_1 \\ \phi_2 \\ \vdots \\ \vdots \\ \phi_{N-1} \\ \phi_N \end{bmatrix} = \begin{bmatrix} D_0 \\ D_1 \\ D_2 \\ \vdots \\ \vdots \\ D_{N-1} \\ D_N \end{bmatrix}$$

California State University Northridge 62

Thomas Algorithm III

- The matrix is called a tridiagonal matrix
  - Has principal diagonal,
  - one diagonal above principal diagonal, and
  - one diagonal below principal diagonal
- Can apply traditional Gauss elimination for solution of simultaneous linear equations to get simple upper triangular form
- Simple equations to obtain this

California State University Northridge 63

Thomas Algorithm IV

- Gauss elimination upper triangular form

$$\begin{bmatrix} 1 & -E_0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & -E_1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & -E_2 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 1 & & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 & -E_{N-1} \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} \phi_0 \\ \phi_1 \\ \phi_2 \\ \vdots \\ \vdots \\ \phi_{N-1} \\ \phi_N \end{bmatrix} = \begin{bmatrix} F_0 \\ F_1 \\ F_2 \\ \vdots \\ \vdots \\ F_{N-1} \\ F_N \end{bmatrix}$$

California State University Northridge 64

Thomas Algorithm V

- Forward computations
  - Initial: $E_0 = -C_0 / B_0$ $F_0 = D_0 / B_0$
  - For i = 1,… N-1:

$$E_i = \frac{-C_i}{B_i + A_iE_{i-1}} \qquad F_i = \frac{D_i - A_iF_{i-1}}{B_i + A_iE_{i-1}}$$

- Get last x value first $x_N = F_N = \dfrac{D_N - A_NF_{N-1}}{B_N + A_NE_{N-1}}$
- Back substitute: $x_i = F_i + E_ix_{i+1}$

California State University Northridge 65

Thomas Example

$$\begin{bmatrix} 1 & 2 & 0 & 0 \\ 3 & 4 & 5 & 0 \\ 0 & 6 & 7 & 8 \\ 0 & 0 & 9 & 10 \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \end{bmatrix} = \begin{bmatrix} 10 \\ 34 \\ 40 \\ 28 \end{bmatrix} \qquad E_0 = \frac{-C_0}{B_0} = -\frac{2}{1} = -2 \qquad F_0 = \frac{D_0}{B_0} = \frac{10}{1} = 10$$

$$E_1 = \frac{-C_1}{B_1 + A_1E_0} = \frac{-5}{4+3(-2)} = \frac{5}{2}$$

$$F_1 = \frac{D_1 - A_1F_0}{B_1 + A_1E_0} = \frac{34-3(10)}{4+3(-2)} = -2$$

- Continue to find E_2, F_2, E_3, and F_3

California State University Northridge 66

Thomas Example II

- Back substitution (shows F and E results)

$$\phi_3 = F_3 = 1$$
$$\phi_2 = F_2 + E_2\phi_3 = \frac{26}{11} + \left(-\frac{4}{11}\right)1 = \frac{22}{11} = 2$$
$$\phi_1 = F_1 + E_1\phi_2 = -2 + \frac{5}{2}2 = 3$$
$$\phi_0 = F_0 + E_0\phi_1 = 10 + (-2)3 = 4$$

- Original equation set shows results are correct

$$\begin{bmatrix} 1 & 2 & 0 & 0 \\ 3 & 4 & 5 & 0 \\ 0 & 6 & 7 & 8 \\ 0 & 0 & 9 & 10 \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \end{bmatrix} = \begin{bmatrix} 10 \\ 34 \\ 40 \\ 28 \end{bmatrix}$$

California State University Northridge

67

Thomas for Two Dimensions

- Two dimensional equation: $a_N\phi_{ij+1} + a_S\phi_{ij-1} + a_E\phi_{i+1j} + a_P\phi_{ij} + a_W\phi_{i-1j} = b_{ij}$
- Look at one-dimensional approach in x direction: $a_E\phi_{i+1j} + a_P\phi_{ij} + a_W\phi_{i-1j} = b_{ij} - a_N\phi_{ij+1} - a_S\phi_{ij-1}$
- Use Thomas algorithm in x-direction

$$a_W\phi_{i-1j}^{(n+1)} + a_P\phi_{ij}^{(n+1)} + a_E\phi_{i+1j}^{(n+1)} = b_{ij} - a_N\phi_{ij+1}^{(n)} - a_S\phi_{ij-1}^{(n)}$$
$$A\phi_{i-1j}^{(n+1)} + B\phi_{ij}^{(n+1)} + C\phi_{i+1j}^{(n+1)} = D$$

- Next apply algorithm in y direction

California State University Northridge

68

Thomas for Two Dimensions II

- y-direction form

$$a_S\phi_{ij-1}^{(n+1)} + a_P\phi_{ij}^{(n+1)} + a_N\phi_{ij+1}^{(n+1)} = b_{ij} - a_E\phi_{i+1j}^{(n)} - a_W\phi_{i-1j}^{(n)}$$
$$A\phi_{ij-1}^{(n+1)} + B\phi_{ij}^{(n+1)} + C\phi_{ij+1}^{(n+1)} = D$$

- This approach involves more calculations per iteration, but it can reduce error more quickly by getting simultaneous solutions of results along one coordinate direction
- Can be extended to three dimensions

California State University Northridge

69

Unstructured Grids

- Do not have ijk indexing system that regular grids have
- Nodes numbered sequentially with single index
- Must store information on numbers of nearest neighbors for each node
- Equation matrix is still sparse, but not so well structured
  - Do not have all coefficients on 5 or 7 diagonals

California State University Northridge

70

Nonlinear Problems

- CFD equations are nonlinear system of difference equations
- Have terms like $u\phi$ and uu
- Have to solve for u, v, w, T, p, etc.
- Typically linearize problem by writing terms like $u\phi$ as $u^{(n)}\phi^{(n+1)}$ to solve for $\phi^{(n+1)}$
- Once iteration n is complete, update linearized terms
- Usually requires underrelaxation

California State University Northridge

71