Numerical Solutions of Elliptic Equations

Larry Caretto
Mechanical Engineering 501B
Seminar in Engineering Analysis

March 25, 2009

California State University Northridge

Outline

- Review last class
- Numerical solution of elliptic equations
- Laplace's equation as an example
- Finite difference forms
- Iterative solution of algebraic equations
- Sample results

Review Properties of Solutions

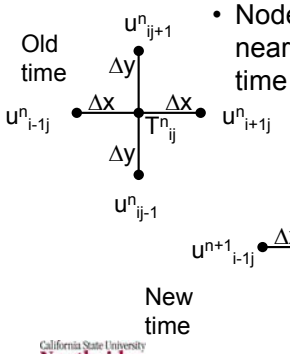
- *Consistency* – truncation error becomes zero as step sizes approach zero
- *Stability* – errors remain bounded
- *Convergent* – tends to the exact solution of the differential equation as the grid size tends towards zero
- *Physical reality* – Solutions produce physically realistic results
- *Accuracy* – Many sources of error in numerical solutions.

Review Two Space Dimensions

$$\left[\frac{\partial T}{\partial t}=\alpha\left(\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2}\right)\right]_{ij}^{(time\ step=n,n+1/2,n+1)}$$

- Extension of one space dimension
- Have grid in y_j as well as x_i and time
- Explicit method has almost no difference. but has more stringent stability limit
- Cannot apply Thomas algorithm directly to implicit method
- Define $f_x = \alpha\Delta t/(\Delta x)^2$ and $f_y = \alpha\Delta t/(\Delta y)^2$

Finite-difference Grid



- Node related to four nearest neighbors at both time steps
- CN uses all T values at old time step

Review Algorithms

- Explicit method
$$T_{ij}^{n+1}=f_x\left(T_{i+1j}^n+T_{i-1j}^n\right)+f_y\left(T_{ij+1}^n+T_{ij-1}^n\right)+\left(1-2f_x-2f_y\right)T_i^n$$
- Crank-Nicholson
$$-f_yT_{ij+1}^{n+1}-f_xT_{i+1j}^{n+1}+2\left(1+f_x+f_y\right)T_{ij}^{n+1}-f_xT_{i-1j}^{n+1}-f_yT_{ij-1}^{n+1}=f_x\left(T_{i+1j}^n+T_{i-1j}^n\right)+f_y\left(T_{ij+1}^n+T_{ij-1}^n\right)+2\left(1-f_x-f_y\right)T_{ij}^n=R_{ij}^n$$
- Fully implicit
$$-f_yT_{ij-1}^{n+1}-f_xT_{i-1j}^{n+1}+(1+2f_x+2f_y)T_{ij}^{n+1}-f_xT_{i+1j}^{n+1}-f_yT_{ij+1}^{n+1}=T_{ij}^n$$

Review Explicit Stability

- $1 - 2f_x - 2f_y$ must be positive for stability so $f_x + f_y$ must be less than 1/2

$$T_{ij}^{n+1} = f_x\left(T_{i+1j}^n + T_{i-1j}^n\right) + f_y\left(T_{ij+1}^n + T_{ij-1}^n\right) + \left(1 - 2f_x - 2f_y\right)T_i^n$$

$$f_x + f_y = \frac{\alpha\Delta t}{(\Delta x)^2} + \frac{\alpha\Delta t}{(\Delta y)^2} \le \frac{1}{2}$$

- To cut Δx and Δy by a factor of 2 we must cut Δt by a factor of 4
- Work increases by a factor of 16

Alternating Direction Implicit

- ADI is technique for allowing use of Thomas algorithm for simple solution
  - At each time step, treat one direction as explicit and one as implicit
  - Even number of time steps required

$$\frac{u_{ij}^* - u_{ij}^n}{\Delta t} = \alpha\left[\frac{u_{i+1j}^* + u_{i-1j}^* - 2u_{ij}^*}{(\Delta x)^2} + \frac{u_{ij+1}^n + u_{ij-1}^n - 2u_{ij}^n}{(\Delta y)^2}\right]$$

$$\frac{u_{ij}^{n+2} - u_{ij}^*}{\Delta t} = \alpha\left[\frac{u_{i+1j}^* + u_{i-1j}^* - 2u_{ij}^*}{(\Delta x)^2} + \frac{u_{ij+1}^{n+2} + u_{ij-1}^{n+2} - 2u_{ij}^{n+2}}{(\Delta y)^2}\right]$$

Elliptic Equations

- Solutions at any point in the region depend on conditions at all boundaries
- No open boundaries like parabolic case
- Main examples are Laplace and Poisson Equations

$$\nabla^2 u = 0 \qquad \nabla^2 u = F(space)$$

- Laplace equation in two-dimensional Cartesian coordinates

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$$

2D Cartesian Laplace

- Use finite-difference grid where $x_i = x_0 + i\Delta x$ and $y_j = y_0 + j\Delta y$
  - $x_0 \le x_i \le x_N$ and $y_0 \le y_j \le y_M$
  - $0 \le i \le N$ and $0 \le j \le M$
  - $\Delta x = (x_N - x_0)/N$ and $\Delta y = (y_M - y_0)/M$
  - Must specify boundary conditions on u at all boundaries
    - South j = 0, $0 \le i \le N$
    - North j = M, $0 \le i \le N$
    - West i = 0, $0 \le j \le M$
    - East i = N, $0 \le j \le M$
- Dependent variable, u

$$u_{ij} = u(x_i, y_j)$$

Finite Differences

- Use second-order expressions for second derivatives

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{ij} = \frac{u_{i+1j} + u_{i-1j} - 2u_{ij}}{(\Delta x)^2} + O[(\Delta x)^2]$$

$$\left.\frac{\partial^2 u}{\partial y^2}\right|_{ij} = \frac{u_{ij+1} + u_{ij-1} - 2u_{ij}}{(\Delta y)^2} + O[(\Delta y)^2]$$

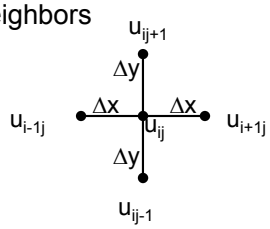
$$\beta = \frac{\Delta x}{\Delta y}$$

$$\left[\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right]_{ij} = \frac{u_{i+1j} + u_{i-1j} - 2u_{ij}}{(\Delta x)^2} + \frac{u_{ij+1} + u_{ij-1} - 2u_{ij}}{(\Delta y)^2} + O[(\Delta x)^2, (\Delta y)^2] = 0$$

- Multiply by $(\Delta x)^2$

Finite-difference Equation

- Links central node to four nearest neighbors



- Only parameter: $\beta = \Delta x/\Delta y$
- For $\beta = 1$, $2(1 + \beta^2) = 4$

$$u_{i+1j} + u_{i-1j} + \beta^2\left(u_{ij+1} + u_{ij-1}\right) - 2\left(1 + \beta^2\right)u_{ij} = 0$$

Finite-difference Equation

- For 2D Cartesian Laplace equation with $\beta = \Delta x/\Delta y = 1$, u_{ij} is the average of its four nearest neighbors

$$u_{ij} = \frac{u_{i+1\,j} + u_{i-1\,j} + u_{ij+1} + u_{ij-1}}{4}$$

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

California State University Northridge 13

General Equation

- Most general case has five diagonals, but can have different terms $\frac{\partial}{\partial x}k\frac{\partial T}{\partial x} + \frac{\partial}{\partial y}k\frac{\partial T}{\partial y} + \dot{Q} = 0$
  - Occurs with variable properties
- 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

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

California State University Northridge 14

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 96×10^6 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

California State University Northridge 15

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)
- Solve equation for u_{ij} and use this as basis for iteration

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

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

California State University Northridge 16

Iterative Solutions II

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

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

- Gauss–Seidel uses most-recent values

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

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

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

Relaxation Factor

California State University Northridge 17

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

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

California State University Northridge 18

Excel Relaxation Code I

- Gauss-Seidel solution to Poisson equation with constant source term

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\dot{Q}}{k}=0$$

$$u_{i+1j}+u_{i-1j}+\beta^2\left(u_{ij+1}+u_{ij-1}\right)-2\left(1+\beta^2\right)u_{ij}+\frac{(\Delta x)^2\dot{Q}}{k}=0$$

$$u_{ij}=\frac{u_{i+1j}+u_{i-1j}+\beta^2\left(u_{ij+1}+u_{ij-1}\right)+\frac{(\Delta x)^2\dot{Q}}{k}}{2\left(1+\beta^2\right)}=0$$

- Iterative Excel Formula for Cell F9

=(E9+G9+betaSqd*(F8+F10)+Source)/Denominator

Excel Relaxation Code II

- Modify previous solution to include relaxation factor, ω

$$u_{ij}^{(n+1)}=(1-\omega)u_{ij}^{(n)}-\omega\frac{u_{i+1j}^{(n)}+u_{i-1j}^{(n+1)}+\beta^2\left(u_{ij+1}^{(n)}+u_{ij-1}^{(n+1)}\right)+\frac{(\Delta x)^2\dot{Q}}{k}}{2\left(1+\beta^2\right)}$$

- Iterative Excel Formula for Cell F9

=oneMinusw*F9+omega*(E9+G9+betaSqd *(F8+F10)+Source)/Denominator

Visual Basic Relaxation Code

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

One set of iterations (omitted on next page)

Visual Basic Relaxation Code II

```
for iter = 1 to maxIter
  maxResid = 0;
  for i = 1 to N – 1
    for j = 1 to N – 1
      'Code in red from previous slide
    next j
  next i
  if maxResid <= errTol exit for
next iter
if maxResid > errTol then
  print #1 "Not converged"
else
  call doOutput( u, N, M )
end if
```


Converging Iterations

- Have three different results
  - Correct solution to differential equation
  - Correct solution 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

Converging Iterations II

- Take infinity norm (maximum absolute value) or L2 norm (RMS error)
  - Former easiest to code

$$\left[\begin{matrix}\text{Relative}\\ \text{Change}\end{matrix}\right]_{ij}=\left|\frac{u_{ij}^{(n+1)}-u_{ij}^{(n)}}{u_{ij}^{(n+1)}}\right|$$

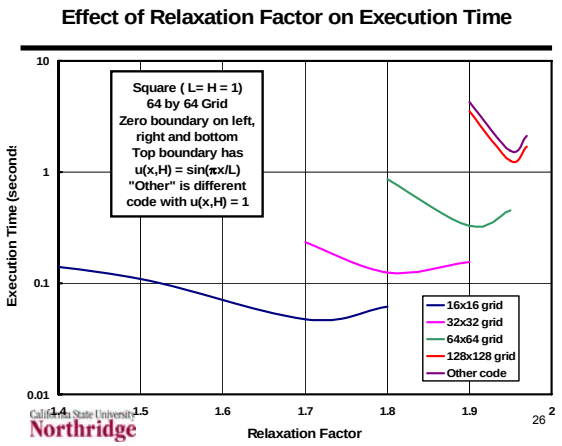
$$\left[\text{Residual}\right]_{ij}^{(n+1)}=\left|u_{ij}^{(n+1)}+A_{ij}^{S'}u_{ij-1}^{(n+1)}\right.$$

$$\left.+A_{ij}^{W'}u_{i-1j}^{(n+1)}+A_{ij}^{E'}u_{i+1j}^{(n+1)}+A_{ij}^{N'}u_{ij+1}^{(n+1)}-b_{ij}^{'}\right|$$

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 $u_N(x) = 1$
  - Case 2: $u_N(x) = \sin(\pi x)$
- First case has discontinuity for y = y_{max} at x = 0 and x = x_{max}
- Use overrelaxation (SOR) with variable relaxation factors

California State University Northridge 25



California State University Northridge 26

Execution Times and Errors II

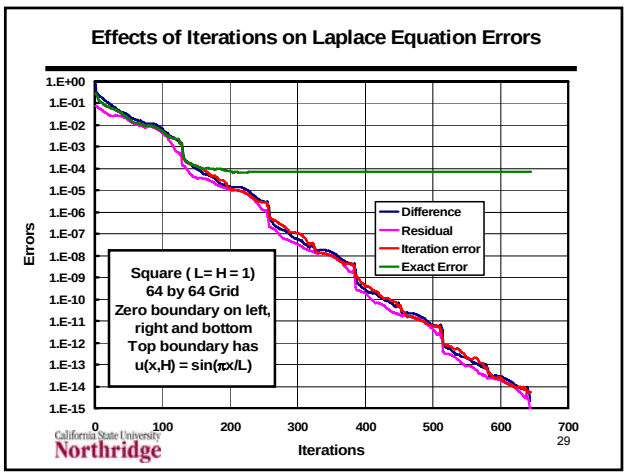
- For second order algorithm the error depends on the boundary condition
  - Case 1: constant value of $u_N(x) = 1$
  - Case 2: $u_N(x) = \sin(\pi x)$
- First case has discontinuity for y = y_{max} at x = 0 and x = x_{max}
  - First case gives first order error due to discontinuity
  - Second case gives second order error

California State University Northridge 27

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 u_{ij} between two iterations
  - Residual = $u_{i+1j} + u_{i-1j} + u_{ij+1} + u_{ij-1} - 4u_{ij}$
  - Exact error is difference between iteration value and exact solution

California State University Northridge 28



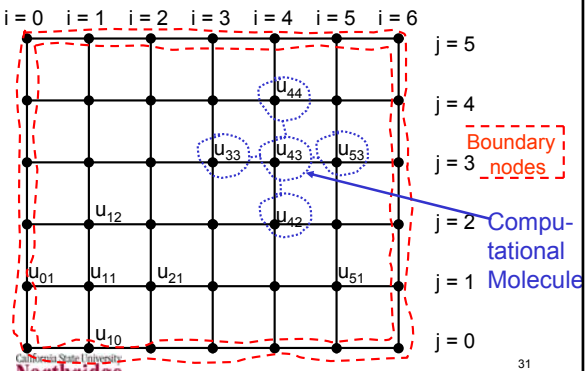
California State University Northridge 29

Advanced Solvers

- 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 u_{ij} form a column vector (one-dimensional)
  - The coefficients in the matrix form a two-dimensional display
  - Examine small grid example

California State University Northridge 30

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

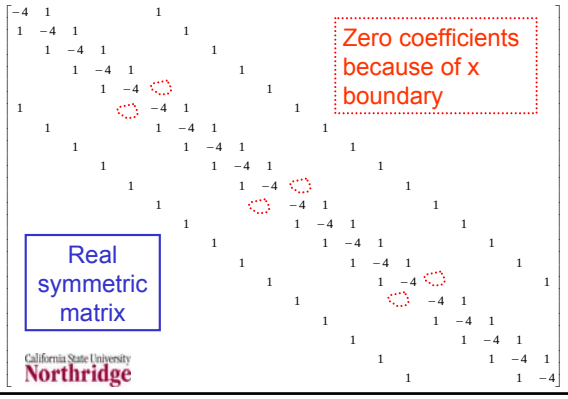


Grid Equations (β = 1)

$$
\begin{array}{cccccccc}
-4u_{11} & +u_{21} & & & +u_{12} & & & = -u_{10} - u_{01} \\
u_{11} & -4u_{21} & +u_{31} & & & +u_{22} & & = -u_{20} \\
 & u_{21} & -4u_{31} & +u_{41} & & & +u_{23} & = -u_{30} \\
 & & u_{31} & -4u_{41} & +u_{51} & & & +u_{24} = -u_{40} \\
 & & & u_{41} & -4u_{51} & & & +u_{25} = -u_{40} - u_{6} \\
u_{11} & & & & -4u_{12} & +u_{22} & & +u_{31} = -u_{02} \\
 & u_{21} & & & +u_{12} & -4u_{22} & +u_{23} & +u_{32} = 0 \\
 & & u_{31} & & & +u_{22} & -4u_{23} & +u_{24} \quad +u_{33} = 0
\end{array}
$$

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

N = 6, M = 5 Matrix Structure



General Matrix Structure

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

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

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

General Equation in a Matrix

- Look at separation between coefficients

A_{ij}^{S} N – 3 coefficients that are zero A_{ij}^{W} A_{ij}^{P} A_{ij}^{E} N – 3 coefficients that are zero A_{ij}^{N}

Not present in **first** N – 1 rows

Not present in **last** N – 1 rows

Not present in first equation and zero in equation N and every N – 1 equations thereafter

Zero in equation N – 1 and every N – 1 equations thereafter. Not present in last equation

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

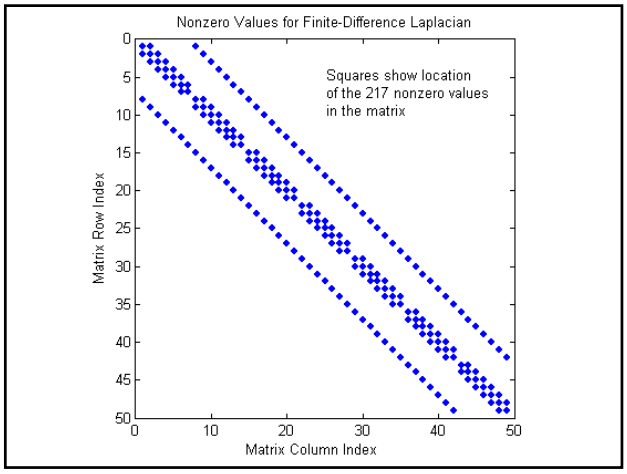
How Sparse is the Matrix?

- 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

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

$$\frac{5}{(N-1)(M-1)} - \frac{2}{(N-1)(M-1)^2} - \frac{2}{(N-1)^2(M-1)^2}$$

California State University Northridge



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 Laplace equation can have more neighbors
- Can have complex coefficients so long as number of neighbors is limited
- Look at uneven grid spacing as an example of this

California State University Northridge

38