Simultaneous Linear Algebraic Equations Concluded

Larry Caretto
Mechanical Engineering 309

Numerical Analysis of Engineering Systems

March 3, 2014

California State University Northridge

Outline

- Review last lecture
- Iterative methods for solution of simultaneous linear algebraic equations
  - Jacobi
  - Gauss-Seidel
  - Overrelaxation
  - Advanced methods
- Fourth programming assignment
  - MATLAB
  - Excel

From Equations to **Ax** = **b**

- Usual form for N = 3 equations

$$\begin{aligned} 3x + 7y - 3z &= 8 \\ 2x - 4y + z &= -3 \\ 8x + 6y - 2z &= 14 \end{aligned}$$

$$\mathbf{A}\quad\mathbf{x} = \mathbf{Ax} = \mathbf{b}$$

$$\begin{bmatrix} 3 & 7 & -3 \\ 2 & -4 & 1 \\ 8 & 6 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 3x_1 & 7x_2 & -3x_3 \\ 2x_1 & -4x_2 & 1x_3 \\ 8x_1 & 6x_2 & -2x_3 \end{bmatrix} = \begin{bmatrix} 8 \\ -3 \\ 14 \end{bmatrix}$$

- An equation is a row in the **Ax** = **b** format

Review Upper Triangular Form

- Convert original set of equations to

$$\begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \cdots & \cdots & \alpha_{1n-1} & \alpha_{1n} \\ 0 & \alpha_{22} & \alpha_{23} & \cdots & \cdots & \alpha_{2n-1} & \alpha_{2n} \\ 0 & 0 & \alpha_{33} & \cdots & \cdots & \alpha_{3n-1} & \alpha_{3n} \\ \vdots & \vdots & \vdots & \ddots & & & \\ \vdots & \vdots & \vdots & & \ddots & & \\ 0 & 0 & 0 & \cdots & \cdots & \alpha_{n-1n-1} & \alpha_{n-1n} \\ 0 & 0 & 0 & \cdots & \cdots & 0 & \alpha_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \vdots \\ \vdots \\ \beta_{n-1} \\ \beta_n \end{bmatrix}$$

Gauss Pseudocode

For pivot = 1 to n − 1

Find maximum element in pivot column, test for singular matrix (exit if singular) and swap rows to put maximum on pivot if not singular

For row = pivot + 1 to n

For column = pivot + 1 to n + nRHS

$$a_{row,column} \leftarrow a_{row,column} - \frac{a_{row,pivot}}{a_{pivot,pivot}} a_{pivot,column}$$

Augment **A** matrix with multiple **b** vectors

Test for "zero" (singular matrix) in a_{nn}

Return error message if singular matrix found; skip back substitution

Multiple **b** Back Substitution

$$x_i = \frac{b_i - \sum_{j=i+1}^{n} a_{ij} x_j}{a_{ii}} \quad \Rightarrow \quad x_{row,bCol} = \frac{a_{row,n+bCol} - \sum_{col=row+1}^{n} a_{row,col}\, x_{col,bCol}}{a_{row,row}}$$

- Loop over all **b** columns, **bCol** = 1 to nRHS
  - Loop over all rows in reverse order: ***row*** = n, n – 1, n – 2, … , 2, 1
    - Set x(***row***, **bCol**) equal to a(***row***, n+**bCol**)
    - Loop over all columns, ***col*** = ***row***+1 to n
      - In column loop set x(***row***, **bCol**) = x(***row***, **bCol**) – a(***row***,***col***) * x(***col***,**bCol**)
    - At end of loop set x(***row***, **bCol**) = x(***row***, **bCol**) / a(***row***, ***row***)

Two Tasks in One

- When we search for the maximum element in the pivot column
  - We improve the accuracy of the solution by swapping the row with the maximum element with the original pivot row data
  - If we find that we do not have a nonzero element in the pivot column (to machine accuracy) in the pivot row and below we know that we do not have a unique solution
    - In this case we stop the solution process and exit with a "singular matrix" error message

California State University Northridge 7

Summary of Methods

- Gaussian – basic approach, useful for one or a few **b** columns, smaller operation count
- Gauss-Jordan – creates unit matrix from original A matrix instead of upper triangular matrix
  - No back substitution required
  - Higher operation count, simpler to perform?
  - Used to get inverse matrix by having n **b** columns that form a diagonal matrix

California State University Northridge 8

Summary of Methods II

- LU method creates upper triangular matrix, **U**, and lower triangular matrix, **L**, such that **LU** = **A**
- Easy to find solution from **LU** when **b** is not know until later
  - Real-time data processing of new **b** data
- Powerful but code more difficult to understand – often used because it has operation count similar to Gaussian

California State University Northridge 9

Gauss-Jordan Pseudo Code

- Loop over all rows,1 to n–1, as pivot rows
  - Search for largest element in pivot column below pivot row and swap rows if necessary
    - If you cannot find a nonzero value in the pivot column return error message "singular matrix"
    - Divide pivot row by $a_{pivot,pivot}$, col = pivot to n+nRHS

$$a_{pivot,column} \leftarrow \frac{a_{pivot,column}}{a_{pivot,pivot}}$$

    - Note that this makes $a_{pivot,pivot} = 1$ so that we can set $a_{pivot,pivot} = 1$ in the usual updating equation

$$a_{row,column} \leftarrow a_{row,column} - \frac{a_{row,pivot}}{a_{pivot,pivot}} a_{pivot,column}$$

$a_{pivot,pivot} = 1$

California State University Northridge 10

Gauss-Jordan Pseudo Code

- Loop over **all** rows **except** the pivot row
  - Loop over all columns to the right of the pivot column (col = 1 to n + nRHS to include s the right-side b values)
    - Replace each column element, $a_{row,column}$, by the formula

$$a_{row,column} \leftarrow a_{row,column} - a_{row,pivot} a_{pivot,column}$$

- At end of pivot loop check for $a_{nn} \approx 0$
- Back substitution: loop over all **b** columns, bcol = 1 to nRHS
  - Loop over all rows, 1 to n
    - Set $x_{row,bCol} = a_{row,n+bCol}$

California State University Northridge 11

MATLAB Solvers

- To solve **Ax** = **b**, where **b** (and x) may have more than one column use x = A\b
- `linsolve` function with options to take advantage of special matrix structures
- `[L U] = lu(A)` generates **L** and **U**
- For infinite solutions `x = pinv(A)*b` gives solution with minimum $||\mathbf{x}||_2$ and `rref(A)` gives equation coefficients
- Use y = A\b for least squares solution

California State University Northridge 12

Iterative Solvers

- Specialized applications: solutions of large matrices that arise in numerical solutions of partial differential equations
  - Such matrices, which are mostly zeros, are called sparse matrices
  - May have 10^4 equations in 10^4 unknonws which could possibly have 10^8 coefficients
  - Typical sparse matrix will only have 5×10^4 or 7×10^4 coefficients
  - Current codes may have 10^6 or more equations

California State University Northridge 13

Simple Iterative Solvers

- Start with equation in usual form

$$\sum_{j=1}^{N} a_{kj} x_j = b_k \qquad k = 1,\ldots,N$$

- Solve equation k for x_k

$$x_k = \frac{b_k - \sum_{j=1,\, j\neq k}^{N} a_{kj} x_j}{a_{kk}} \qquad k = 1,\ldots,N$$

- Use superscript (m) for iteration

$$x_k^{(m)} = \frac{b_k - \sum_{j=1,\, j\neq k}^{N} a_{kj} x_j^{(m-1)}}{a_{kk}} \qquad k = 1,\ldots,N$$

California State University Northridge 14

Simple Iterative Solvers II

- Make initial guesses for all the $x_k^{(0)}$ (typically use $x_k^{(0)} = 0$, for k = 1, N
- Use equations from previous slide, shown below, to get iterations 1, 2, 3, *etc*.
  - First approach uses values from previous iteration for all calculations in new iteration
    - This is called Jacobi Iteration

$$x_k^{(m+1)} = \frac{b_k - \sum_{j=1,\, j\neq k}^{N} a_{kj} x_j^{(m)}}{a_{kk}} \qquad k = 1,\ldots,N$$

California State University Northridge 15

Jacobi Example

- Solve 4x + y = 6 and x + 2y = 5
- Iteration form: x = (6 – y)/4, y = (5 – x)/2
- Initial guesses x = y = 0
- First iteration x = (6 – y)/4 = (6 – 0)/4 = 3/2 and y = (5 – x)/2 = (5 – 0)/2 = 5/2
- Second iteration x = (6 – y)/4 = (6 – 5/2)/4 = 7/8 and y = (5 – x)/2 = (5 – 3/2)/2 = 7/4
- Third iteration x = (6 – y)/4 = (6 – 7/4)/4 = 17/16 and y = (5 – x)/2 = (5 – 7/8)/2 = 33/16

California State University Northridge ***Continue until successive iterations close enough*** 16

Simple Iterative Solvers III

- To improve Jacobi method use new iterations as they become available
  - When we solve for x_k we will have new iteration values for x_1 to x_{k-1}

$$x_k^{(m+1)} = \frac{b_k - \sum_{j=1}^{k-1} a_{kj} x_j^{(m+1)} - \sum_{j=k+1}^{N} a_{kj} x_j^{(m)}}{a_{kk}} \qquad k = 1,\ldots,N$$

- Called Gauss-Seidel iteration
- Repeat solution of previous equations: x = (6 – y)/4, y = (5 – x)/2, with x = y = 0 initially

California State University Northridge 17

Gauss-Seidel Example

- Solve 4x + y = 6 and x + 2y = 5
- Iteration form: x = (6 – y)/4, y = (5 – x)/2
- Initial guesses x = y = 0
- First iteration x = (6 – y)/4 = (6 – 0)/4 = 3/2 and y = (5 – x)/2 = (5 – 3/2)/2 = 7/4
- Second: x = (6 – y)/4 = (6 – 7/4)/4 = 17/16 and y = (5 – x)/2 = (5 – 17/16)/2 = 63/32
- Third: x = (6 – y)/4 = (6 – 63/32)/4 = 129/128; y = (5 – x)/2 = (5 – 129/128)/2 = 511/256

California State University Northridge ***Continue until successive iterations close enough*** 18

Successive Overrelaxation

- View iteration method as giving a correction: $x_k^{(m+1)} = x_k^{(m)} + \Delta x_k$
- If Δx_k is good, maybe more Δx_k is better
  - Define a relaxation factor, ω, such that $x_k^{(m+1)} = x_k^{(m)} + \omega \Delta x_k$
    - Underrelaxation ($\omega<1$) can cure divergence and overrelaxation ($1<\omega<2$) accelerates convergence
    - Choose $\Delta x_k = x_k^{(m+1,GS)} - x_k^{(m)}$ where

GS superscript indicates conventional Gauss-Seidel iteration value

$$x_k^{(m+1,GS)} = \frac{b_k - \sum_{j=1}^{k-1} a_{kj} x_j^{(m+1)} - \sum_{j=k+1}^{N} a_{kj} x_j^{(m)}}{a_{kk}}$$

Successive Overrelaxation II

- Combine equations on previous slide
  - $x_k^{(m+1)} = x_k^{(m)} + \omega \Delta x_k$
    $= x_k^{(m)} + \omega[x_k^{(m+1,GS)} - x_k^{(m)}]$
    $= (1-\omega)x_k^{(m)} + \omega x_k^{(m+1,GS)}$

$$x_k^{(m+1)} = (1-\omega)x_k^{(m)} + \omega \frac{b_k - \sum_{j=1}^{k-1} a_{kj} x_j^{(m+1)} - \sum_{j=k+1}^{N} a_{kj} x_j^{(m)}}{a_{kk}}$$

- Usual application is to equation where most of the a_{kj} are zeros and $x_k^{(m+1)}$ is written in special form for application

Advanced Methods

- Methods currently used in numerical solution of partial differential equations
- Stone's method or strongly-implicit method uses more values from next iteration step
- Conjugate gradient is optimization approach equivalent to solving **Ax** = **b**
- Multigrid methods use different size grids to optimize solution convergence

Limitations on Iterative Solvers

- System of equations must be diagonally dominant

$$|a_{kk}| \geq \sum_{j=1,\, j\neq k}^{N} |a_{kj}| \qquad all\ k$$

$$|a_{kk}| > \sum_{j=1,\, j\neq k}^{N} |a_{kj}| \qquad at\ least\ one\ k$$

- A stringent requirement for most problems, but is usually met for numerical analysis of partial differential equations

Programming Assignment 4

- Look at MATLAB and Excel solvers using Gaussian elimination
- Set of 100 equations/known solution
- Compare errors in solution by various methods starting with MATLAB
  - VBA code with single and double data types with and without pivoting

$$E_{RMS} = \sqrt{\frac{\sum_{k=1}^{N} \left(x_{Numerical,k} - x_{Exact,k}\right)^2}{N}}$$

MATLAB Exercise
x_N = [1.1 1.8 3.3]
x_E = [1 2 3]
What is E_{RMS}?

Programming Assignment 4 II

- MATLAB Operations
  - Import data from Excel
  - Solve problem and compute RMS error for ***each*** MATLAB solution
    - Compare x and x_{Exact} for each of the twelve solutions given as well as for the combination all 1,200 values of $x_{k,solution}$
  - Compute Det(**A**) and error in $\mathbf{AA}^{-1}$
  - Use MATLAB condition number and norm
  - Examine infinite solutions problem with four equations using `pinv` and `rref`

Programming Assignment 4 III

- Determine RMS errors in Excel
  - Use minverse
  - Use VBA code provided using double data type and pivoting
  - Modify previous code and use
    - Type double and no pivoting
    - Type single and no pivoting
    - Type single and pivoting
- Modify VBA code in Workbook to find determinant of a matrix

$$E_{RMS} = \sqrt{\frac{\sum_{k=1}^{N}\left(x_{Excel,k} - x_{Exact,k}\right)^2}{N}}$$

California State University Northridge 25

Programming Assignment 4 V

- Modify Gaussian code to compute determinant instead of **x** solution
  - If **A** is converted to upper triangular form, Det(**A**) is product of diagonal elements a_{kk}
  - The sign of a determinant changes if rows are interchanged
    - Must modify code to keep track of each time a row is swapped
- See written assignment for discussion questions and files to submit

California State University Northridge 26

Excel Exercise Tonight

- Download Assignment four Excel workbook from web site
- Confirm that names A, b, and x, refer to the matrix, **A**, the set of 12 right-hand sides, **b**, and the 12 exact solutions, **x**
- Create two new worksheets
  - On one use mmult(minverse(A),b) for x_{Num}
  - On the other use Gaussian(A,b) for x_{Num}
  - Find RMS errors in each solution, $\{[\Sigma(x_{correct,i} - x_{Excel,i})^2]/N\}^{1/2}$ using SUMSQ

California State University Northridge 27

Fourth Quiz Comments

- Matrix multiplication generally OK
- For inverse, basic idea is that product of **A** and $\mathbf{A}^{-1}$ = **I**, a diagonal matrix with ones one the diagonal
- Solution of system of equations by Gauss-Seidel on quiz this Wesnesday
  - Solutions for two different sets of right hand sides
  - Three possibilities: unique solution, no solution, infinite solution

California State University Northridge 29

Remaining Quizzes/Exams

- Schedule (all Wednesdays, except final)
  - March 5: Quiz 5: Use Gaussian elimination to solve a system of linear equations
  - March 12: Quiz 6
  - March 26: Midterm Exam
  - April 16: Quiz 7
  - April 23: Quiz 8
  - May 7: Programming Exam
  - Monday, May 12: Final Exam, 8-10 pm

California State University Northridge 30