LECTURE 17

DIRECT SOLUTIONS TO LINEAR SYSTEMS OF ALGEBRAIC EQUATIONS

- Solve the system of equations

$$\mathbf{A}\boldsymbol{X} = \boldsymbol{B}$$

$$\Rightarrow$$

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}$$

- The solution is formally expressed as:

$$\boldsymbol{X} = \mathbf{A}^{-1}\boldsymbol{B}$$

- Typically it is more efficient to solve for $\boldsymbol{X}$ directly without solving for $\mathbf{A}^{-1}$ since finding the inverse is an expensive (and less accurate) procedure

- Types of solution procedures
    - Direct Procedures
        - Exact procedures which have infinite precision (excluding roundoff error)
        - Suitable when $\mathbf{A}$ is relatively fully populated/dense or well banded
        - A predictable number of operations is required
    - Indirect Procedures
        - Iterative procedures
        - Are appropriate when $\mathbf{A}$ is
            - Large and sparse but not tightly banded
            - Very large (since roundoff accumulates more slowly)
        - Accuracy of the solution improves as the number of iterations increases

Cramer's Rule - A Direct Procedure

- The components of the solution $\boldsymbol{X}$ are computed as:

$$x_k = \frac{|\mathbf{A}_k|}{|\mathbf{A}|}$$

where

$\mathbf{A}_k$ is the matrix $\mathbf{A}$ with its k^{th} column replaced by vector $\boldsymbol{B}$

$|\mathbf{A}|$ is the determinant of matrix $\mathbf{A}$

- For each $\boldsymbol{B}$ vector, we must evaluate $N+1$ determinants of size N where N defines the size of the matrix $\mathbf{A}$

- Evaluate a determinant as follows using the method of expansion by cofactors

$$|\mathbf{A}| = \sum_{i=1}^{N} a_{I,j}[cof(a_{I,j})] = \sum_{i=1}^{N} a_{i,J}[cof(a_{i,J})]$$

where

I = specified value of i

J = specified value of j

$$cof(a_{i,j}) = (-1)^{i+j}(minor(a_{i,j}))$$

minor $(a_{i,j})$ = determinant of the sub-matrix obtained by deleting the i^{th} row and the j^{th} column

- Procedure is repeated until 2×2 matrices are established (which has a determinant by definition):

$$|\mathbf{A}| = \begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix} = a_{1,1}a_{2,2} - a_{2,1}a_{1,2}$$

Example

- Evaluate the determinant of $\mathbf{A}$

$$det[\mathbf{A}] = |\mathbf{A}| = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix} \Rightarrow$$

$$det[\mathbf{A}] = a_{1,1}(-1)^{(1+1)}\begin{bmatrix} a_{2,2} & a_{2,3} \\ a_{3,2} & a_{3,3} \end{bmatrix} + a_{1,2}(-1)^{(1+2)}\begin{bmatrix} a_{2,1} & a_{2,3} \\ a_{3,1} & a_{3,3} \end{bmatrix}$$

$$+ a_{1,3}(-1)^{(1+3)}\begin{bmatrix} a_{2,1} & a_{2,2} \\ a_{3,1} & a_{3,2} \end{bmatrix} \Rightarrow$$

$$det[\mathbf{A}] = a_{1,1}(+1)(a_{2,2}a_{3,3} - a_{3,2}a_{2,3}) + a_{1,2}(-1)(a_{2,1}a_{3,3} - a_{3,1}a_{2,3})$$

$$+ a_{1,3}(+1)(a_{2,1}a_{3,2} - a_{3,1}a_{2,2})$$

- Note that more efficient methods are available to compute the determinant of a matrix. These methods are associated with alternative direct procedures.

- This evaluation of the determinant involves $O(N)^3$ operations

- Number of operations for Cramers' Rule $O(N)^4$

$$2 \times 2 \text{ system} \Rightarrow O(2^4) = O(16)$$

$$4 \times 4 \text{ system} \Rightarrow O(4^4) = O(256)$$

$$8 \times 8 \text{ system} \Rightarrow O(8^4) = O(4096)$$

- ***Cramer's rule is not a good method for very large systems!***

- If $|\mathbf{A}| = 0$ and $|\mathbf{A}_k| \neq 0 \quad \Rightarrow$ no solution! The matrix $\mathbf{A}$ is singular

- If $|\mathbf{A}| = 0$ and $|\mathbf{A}_k| = 0 \quad \Rightarrow$ infinite number of solutions!

Gauss Elimination - A Direct Procedure

- ***Basic concept is to produce an upper or lower triangular matrix and to then use backward or forward substitution to solve for the unknowns.***

Example application

- Solve the system of equations

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

- Divide the first row of $\mathbf{A}$ and $\boldsymbol{B}$ by $a_{1,1}$ (***pivot element***) to get

$$\begin{bmatrix} 1 & a'_{1,2} & a'_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b'_1 \\ b_2 \\ b_3 \end{bmatrix}$$

- Now multiply row 1 by $a_{2,1}$ and subtract from row 2

  and then multiply row 1 by $a_{3,1}$ and subtract from row 3

$$\begin{bmatrix} 1 & a'_{1,2} & a'_{1,3} \\ 0 & a'_{2,2} & a'_{2,3} \\ 0 & a'_{3,2} & a'_{3,3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b'_1 \\ b'_2 \\ b'_3 \end{bmatrix}$$

- Now divide row 2 by $a'_{2,2}$ (***pivot element***)

$$\begin{bmatrix} 1 & a'_{1,2} & a'_{1,3} \\ 0 & 1 & a''_{2,3} \\ 0 & a'_{3,2} & a'_{3,3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b'_1 \\ b''_2 \\ b'_3 \end{bmatrix}$$

- Now multiply row 2 by $a'_{3,2}$ and subtract from row 3 to get

$$\begin{bmatrix} 1 & a'_{1,2} & a'_{1,3} \\ 0 & 1 & a''_{2,3} \\ 0 & 0 & a''_{3,3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b'_1 \\ b''_2 \\ b''_3 \end{bmatrix}$$

- Finally divide row 3 by $a''_{3,3}$ (***pivot element***) to complete the triangulation procedure and results in the ***upper triangular matrix***

$$\begin{bmatrix} 1 & a'_{1,2} & a'_{1,3} \\ 0 & 1 & a''_{2,3} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b'_1 \\ b''_2 \\ b'''_3 \end{bmatrix}$$

- We have triangularized the coefficient matrix simply by taking linear combinations of the equations

- We can very conveniently solve the ***upper triangularized*** system of equations

$$\begin{bmatrix} 1 & a'_{1,2} & a'_{1,3} \\ 0 & 1 & a''_{2,3} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b'_1 \\ b''_2 \\ b'''_3 \end{bmatrix}$$

- We apply a ***backward substitution*** procedure to solve for the components of ***X***

$$x_3 = b'''_3$$

$$x_2 + a''_{2,3}x_3 = b''_2 \quad \Rightarrow \quad x_2 = b''_2 - a''_{2,3}x_3$$

$$x_1 + a'_{1,2}x_2 + a'_{1,3}x_3 = b'_1 \quad \Rightarrow \quad x_1 = b'_1 - a'_{1,2}x_2 - a'_{1,3}x_3$$

- We can also produce a lower triangular matrix and use a forward substitution procedure

- Number of operations required for Gauss elimination
  - Triangularization $\frac{1}{3}N^3$
  - Backward substitution $\frac{1}{2}N^2$
  - Total number of operations for Gauss elimination equals $O(N)^3$ versus $O(N)^4$ for Cramer's rule
  - Therefore we save $O(N)$ operations as compared to Cramer's rule

Gauss-Jordan Elimination - A Direct Procedure

- Gauss Jordan elimination is an adaptation of Gauss elimination in which both elements above and below the pivot element are cleared to zero $\rightarrow$ the entire column except the pivot element become zeroes

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1'''' \\ b_2'''' \\ b_3'''' \\ b_4'''' \end{bmatrix}$$

- No backward/forward substitution is necessary

Matrix Inversion by Gauss-Jordan Elimination

- Given $\mathbf{A}$, find $\mathbf{A}^{-1}$ such that

$$\mathbf{A}\mathbf{A}^{-1} \equiv \mathbf{I}$$

where $\mathbf{I}$ = identity matrix = $\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$

- Procedure is similar to finding the solution of $\mathbf{A}\boldsymbol{X} = \boldsymbol{B}$ except that the matrix $\mathbf{A}^{-1}$ assumes the role of vector $\boldsymbol{X}$ and matrix $\mathbf{I}$ serves as vector $\boldsymbol{B}$
  - Therefore we perform the same operations on $\mathbf{A}$ and $\mathbf{I}$

- Convert $\mathbf{A} \rightarrow \mathbf{I}$ through Gauss-Jordan elimination

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$$

$$\Rightarrow$$

$$\mathbf{A}'\mathbf{A}^{-1} = \mathbf{I}'$$

- However through the manipulations $\mathbf{A} \rightarrow \mathbf{A}' = \mathbf{I}$ and therefore

$$\mathbf{I}\mathbf{A}^{-1} = \mathbf{I}'$$

$$\Rightarrow$$

$$\mathbf{A}^{-1} = \mathbf{I}'$$

- The right hand side matrix, $\mathbf{I}'$, has been transformed into the inverted matrix

- Notes:
  - Inverting a diagonal matrix simply involves computing reciprocals

$$\mathbf{A} = \begin{bmatrix} a_{11} & 0 & 0 \\ 0 & a_{22} & 0 \\ 0 & 0 & a_{33} \end{bmatrix}$$

$$\mathbf{A}^{-1} = \begin{bmatrix} 1/a_{11} & 0 & 0 \\ 0 & 1/a_{22} & 0 \\ 0 & 0 & 1/a_{33} \end{bmatrix}$$

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$$

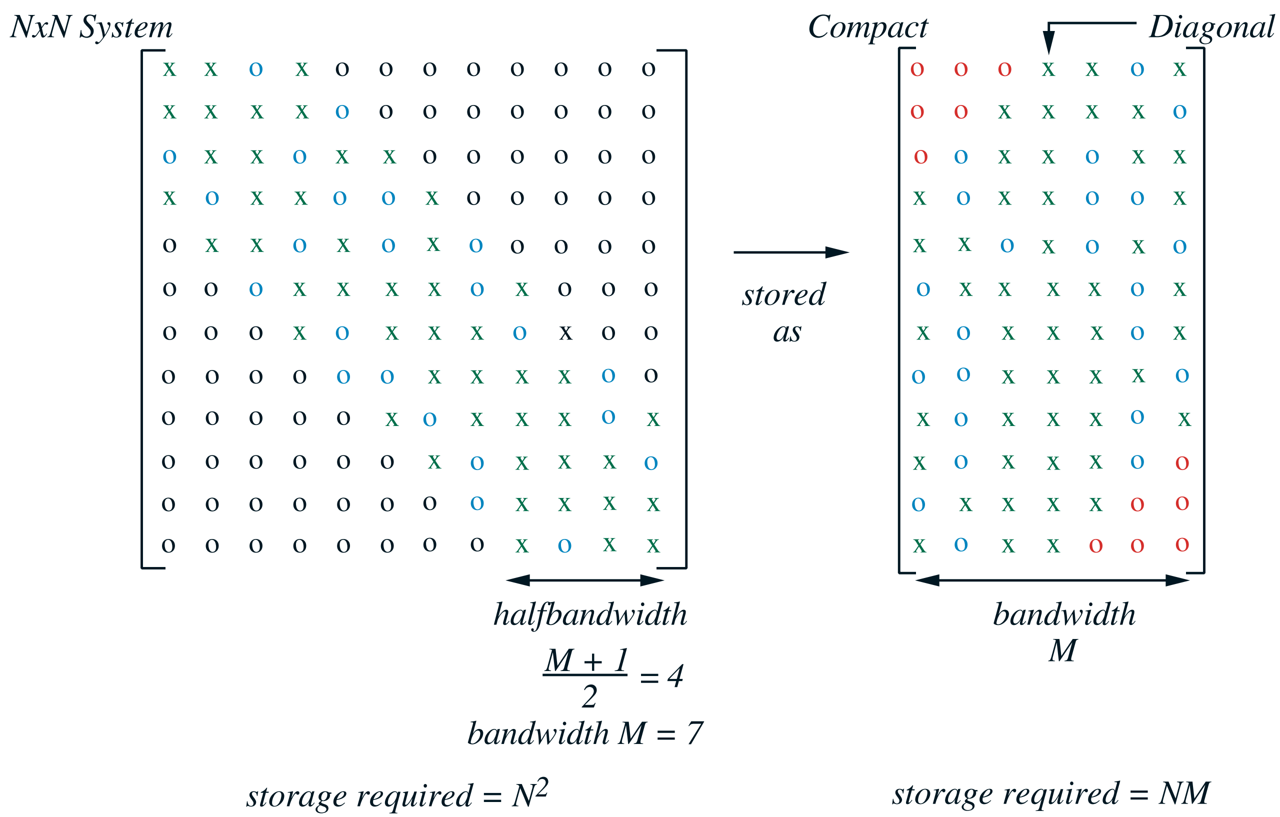
  - Inverse of the product relationship

$$[\mathbf{A}_1\mathbf{A}_2\mathbf{A}_3]^{-1} = \mathbf{A}_3^{-1}\mathbf{A}_2^{-1}\mathbf{A}_1^{-1}$$

Gauss Elimination Type Solutions to Banded Matrices

Banded matrices

- Have non-zero entries contained within a defined number of positions to the left and right of the diagonal (bandwidth)



- Notes on banded matrices
  - The advantage of banded storage mode is that we avoid storing and manipulating zero entries ***outside*** of the defined bandwidth
  - Banded matrices typically result from finite difference and finite element methods (conversion from p.d.e. → algebraic equations)
  - Compact banded storage mode can still be sparse (this is particularly true for *large* finite difference and finite element problems)

Savings on storage for banded matrices

- N^2 for full storage versus NM for banded storage

  where N = the size of the matrix and M = the bandwidth

- Examples:

N	M	full	banded	ratio
400	20	160,000	8,000	20
10^6	10^3	10^{12}	10^9	1000

Savings on computations for banded matrices

- Assuming a Gauss elimination procedure

$$O(N^3) \text{ versus } O(NM^2)$$
(full) (banded)

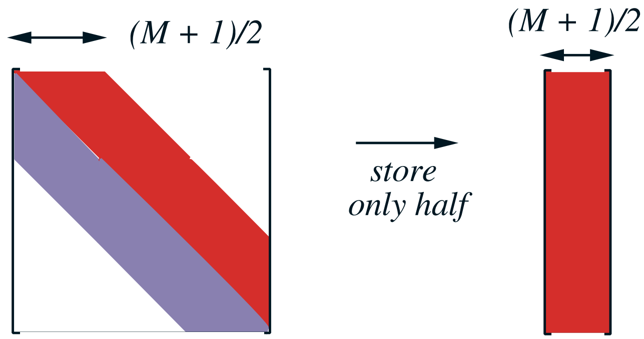
- Therefore save $O(N^2/M^2)$ operations since we are not manipulating all the zeros outside of the bands!

- Examples:

N	M	full	banded	ratio
400	20	$O(6.4\text{x}10^7)$	$O(1.6\text{x}10^5)$	$O(400)$
10^6	10^3	$O(10^{18})$	$O(10^{12})$	$O(10^6)$

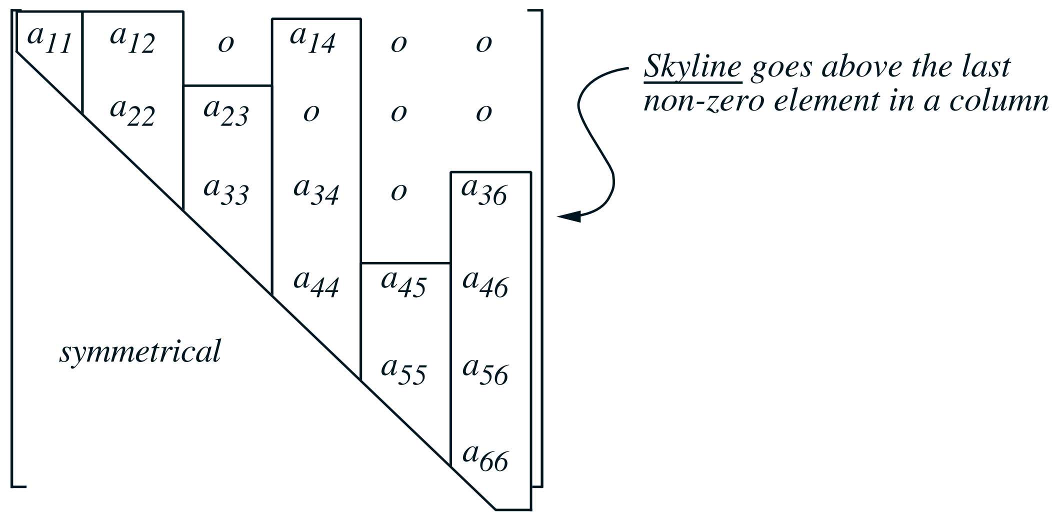
Symmetrical banded matrices

- Substantial savings on both storage and computations if we use a banded storage mode
- Even greater savings (both storage and computations) are possible if the matrix **A** is symmetrical
  - Therefore if $a_{ij} = a_{ji}$ we need only store and operate on half the bandwidth in a banded matrix (half the matrix in a full storage mode matrix)



Alternative Compact Storage Modes for Direct Methods

- Skyline method defines an alternative compact storage procedure for symmetrical matrices
- The skyline goes below the last non-zero element in a column



- Store *all* entries between skyline and diagonal into a vector as follows:

$$\begin{bmatrix} A(1) & A(3) & o & A(9) & o & o \\ & A(2) & A(5) & A(8) & o & o \\ & & A(4) & A(7) & o & A(15) \\ & & & A(6) & A(11) & A(14) \\ & \textit{symmetrical} & & & A(10) & A(13) \\ & & & & & A(12) \end{bmatrix}$$

- Accounting procedure must be able to identify the location within the matrix of elements stored in vector mode in $A(i)$ $\Rightarrow$ Store locations of diagonal terms in $A(i)$

$$MaxA = \begin{bmatrix} 1 \\ 2 \\ 4 \\ 6 \\ 10 \\ 12 \end{bmatrix}$$

- Savings in storage and computation time due to the elimination of the additional zeroes

  e.g. storage savings:

full	symmetrical banded	skyline
$N^2 = 36$	$\left(\frac{M+1}{2}\right)N = \left(\frac{7+1}{2}\right)6 = 24$	15

- Program COLSOL (Bathe and Wilson) available for skyline storage solution

Problems with Gauss Elimination Procedures

Inaccuracies originating from the pivot elements

- **The *pivot element*** is the diagonal element which divides the associated row
- As more pivot rows are processed, the number of times a pivot element has been modified increases.
- Sometimes a pivot element can become very small compared to the rest of the elements in the pivot row
    - Pivot element will be inaccurate due to roundoff
    - When the pivot element divides the rest of the pivot row, large inaccurate numbers result across the pivot row
    - Pivot row now subtracts (after being multiplied) from all rows below the pivot row, resulting in propagation of large errors throughout the matrix!

Partial pivoting

- Always look below the pivot element and pick the row with the largest value and switch rows

Complete pivoting

- Look at all columns and all rows to the right/below the pivot element and switch so that the largest element possible is in the pivot position.
- For complete pivoting, you must change the order of the variable array
- Pivoting procedures give large diagonal elements
    - minimize roundoff error
    - increase accuracy
- Pivoting is not required when the matrix is diagonally dominant
    - A matrix is diagonally dominant when the absolute values of the diagonal terms is greater than the sum of the absolute values of the off diagonal terms for each row