WASHINGTON STATE UNIVERSITY
TRI-CITIES
World Class. Face to Face.

Instructor: Zhenyu (Henry) Huang
(509) 372-6781, zhenyu.huang@pnl.gov

EE 521: Analysis of Power Systems

Lecture 4
Matrix Operations

Fall 2009

Mondays & Wednesdays 5:45-7:00

August 24 – December 18

Test 216

Topics

- Needs for Matrix Operations
- Sparsity
  - Storage Format
- LU Decomposition
  - Crout’s Method
  - DooLittle’s Method
  - Matrix Permutations

We need matrix operations all the time

- Matrix Construct and Update
  - Y Matrix
  - Contingency Analysis
- Matrix Inversion
  - Newton-Raphson
  - Decoupled Power Flow
  - DC Power Flow

$$Ax = b$$

$$Y_{new} = Y_0 + \Delta Y$$

$$[J(x)]\begin{bmatrix}\Delta\theta \\ \Delta V\end{bmatrix} = \begin{bmatrix}\Delta P(x) \\ \Delta Q(x)\end{bmatrix}$$

$$\begin{cases} [-B][\Delta\theta] = \left[\dfrac{\Delta P(\theta^n)}{|V|}\right] \\ [-B][\Delta V] = \left[\dfrac{\Delta Q(V^n)}{|V|}\right] \end{cases}$$

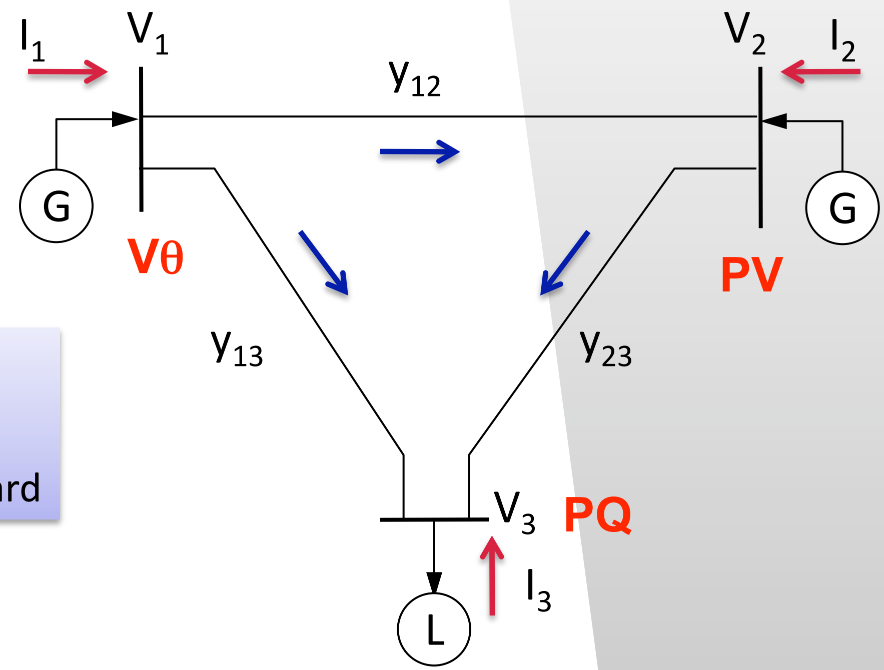
$$[-B][\theta] = [P]$$

Network Incidence Matrix

$$Y = \begin{bmatrix} y_{12} + y_{13} & -y_{12} & -y_{13} \\ -y_{12} & y_{12} + y_{23} & -y_{23} \\ -y_{13} & -y_{23} & y_{13} + y_{23} \end{bmatrix}$$

$$A = \begin{array}{c} \\ 12 \\ 23 \\ 13 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ 1 & 0 & -1 \end{bmatrix} \end{array}$$

0 – no connection
1 – current flowing away
-1 – Current flowing toward



$$Y = A^T \begin{bmatrix} y_{12} & 0 & 0 \\ 0 & y_{23} & 0 \\ 0 & 0 & y_{13} \end{bmatrix} A$$

Matrix Update for Contingency Analysis

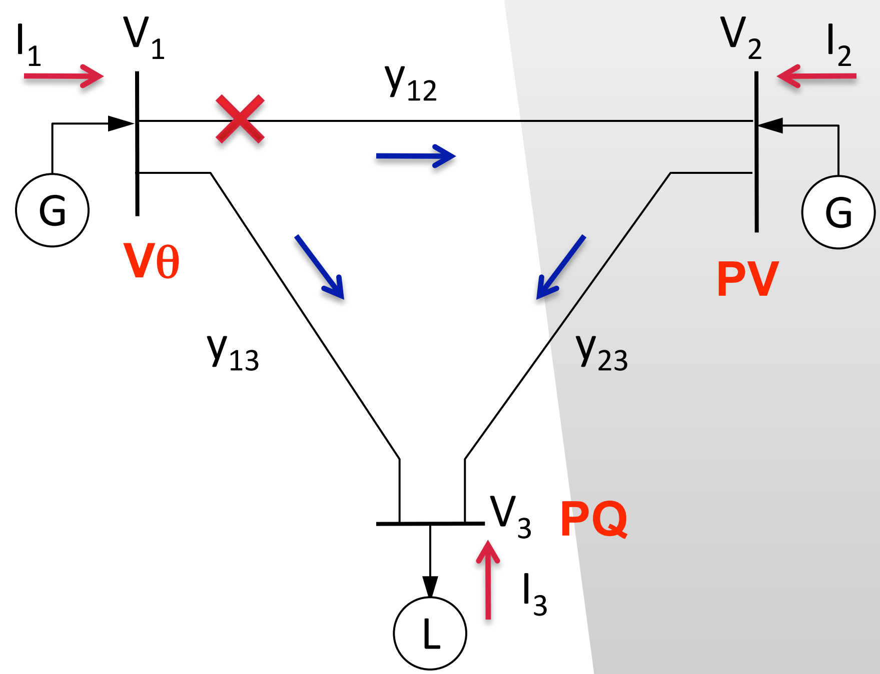
Method 1 – retain branch admittance

$$A = \begin{array}{c} \\ 12 \\ 23 \\ 13 \end{array} \begin{array}{c} \begin{matrix} 1 & 2 & 3 \end{matrix} \\ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & -1 \\ 1 & 0 & -1 \end{bmatrix} \end{array}$$

Method 2 – retain topology

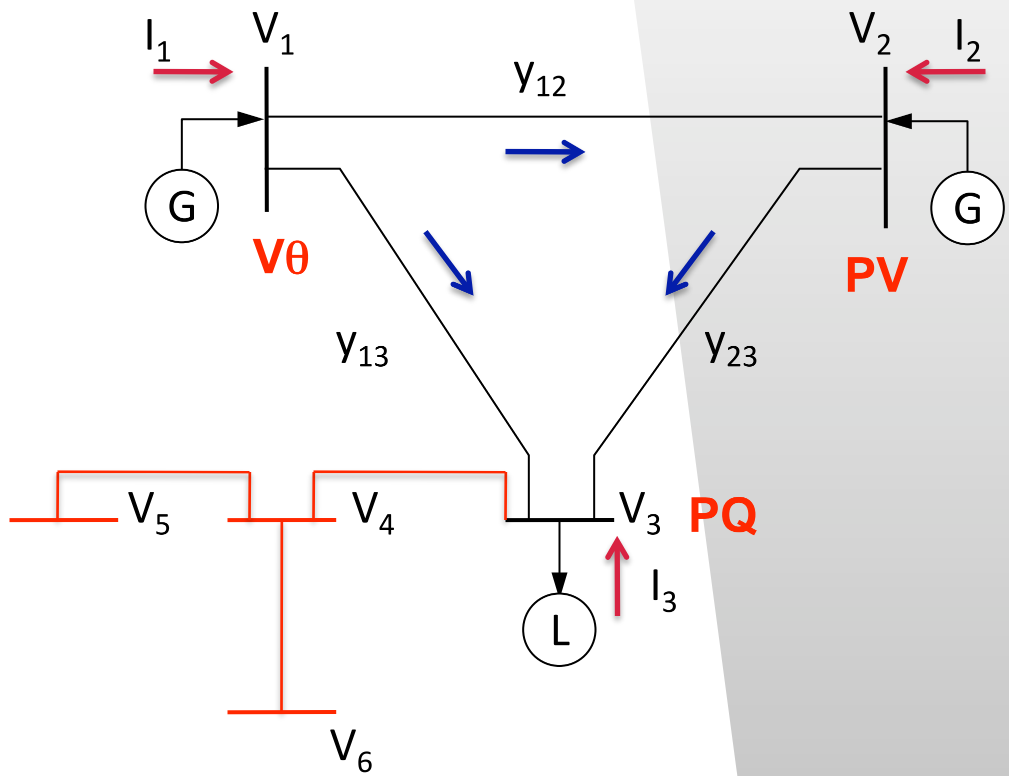
$$Y = A^T \begin{bmatrix} y_{12} - y_{12} & 0 & 0 \\ 0 & y_{23} & 0 \\ 0 & 0 & y_{13} \end{bmatrix} A$$

$$= A^T \begin{bmatrix} y_{12} & 0 & 0 \\ 0 & y_{23} & 0 \\ 0 & 0 & y_{13} \end{bmatrix} A + A^T \begin{bmatrix} -y_{12} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} A = Y + \begin{bmatrix} -y_{12} & y_{12} & 0 \\ y_{12} & -y_{12} & 0 \\ 0 & 0 & 0 \end{bmatrix}$$



Large-scale Systems

$$Y = \begin{bmatrix} * & * & * & 0 & 0 & 0 \\ * & * & * & 0 & 0 & 0 \\ * & * & * & * & 0 & 0 \\ 0 & 0 & * & * & * & * \\ 0 & 0 & 0 & * & * & 0 \\ 0 & 0 & 0 & * & 0 & * \end{bmatrix}$$



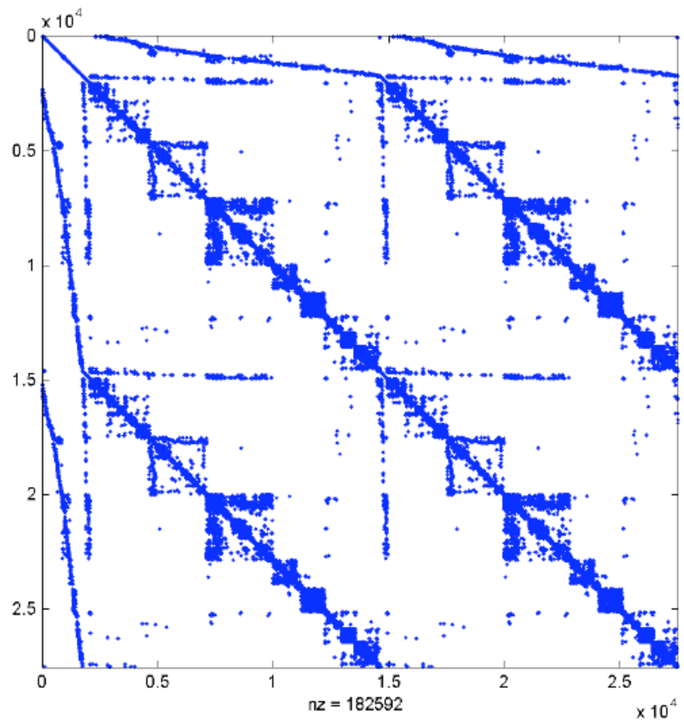
Sparsity Increases…

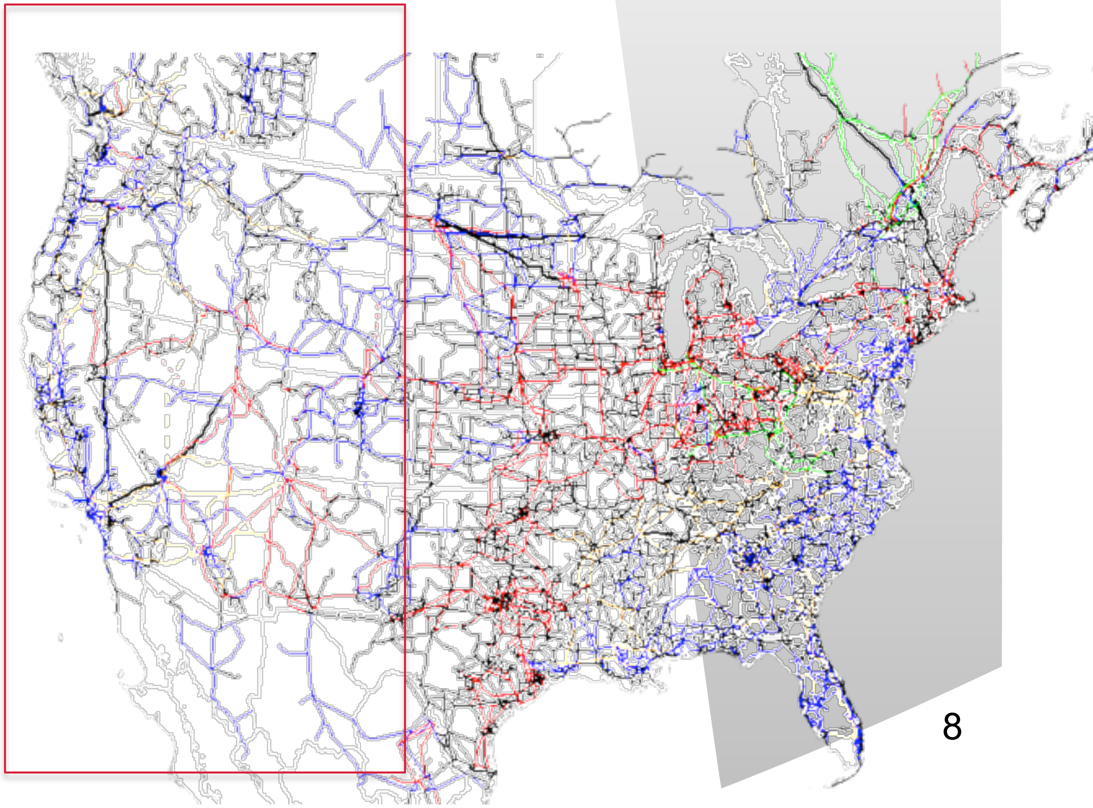
Sparsity

- Many of the matrices in power systems analysis are sparse
- In general, an inverse of a matrix is a computationally intensive process
- Taking advantage of sparse matrix techniques reduces data storage requirements and improves computational performance

Sparse Matrix Example

- WECC model: 14,000 bus, 17,000 lines. Its Jacobian has only 0.024% of the entries as non-zero values





Sparse Matrix Storage Format

- Because power system matrices are so sparse, storing them as full matrices is inefficient
  - WECC Y Matrix dimension: 28,000 x 28,000
  - If each element takes 2 bytes, 28,000 x 28,000 x 2 bytes = 1.57 GB
  - 1.57 GB x 0.024% = 0.38 MB
- There are numerous methods that exist which only store the non-zero values and their positions in the matrix
- One such example, which is used by SuperLU, is the Compressed Row Sparse Matrix format

Example –
Sparse Matrix Storage Format

Problem: Store the given sparse matrix in Compressed Row Format

$$\begin{bmatrix} 5 & 7 & 0 & 0 \\ 7 & 1 & 3 & 0 \\ 0 & 3 & 4 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix}$$

Solution:

$$V = \begin{bmatrix} 5 & 7 & 7 & 1 & 3 & 3 & 4 & 2 \end{bmatrix}$$
$$C = \begin{bmatrix} 1 & 2 & 1 & 2 & 3 & 2 & 3 & 4 \end{bmatrix}$$
$$R = \begin{bmatrix} 1 & 1 & 2 & 2 & 2 & 3 & 3 & 4 \end{bmatrix}$$

$$V = \begin{bmatrix} 5 & 7 & 7 & 1 & 3 & 3 & 4 & 2 \end{bmatrix}$$
$$C = \begin{bmatrix} 1 & 2 & 1 & 2 & 3 & 2 & 3 & 4 \end{bmatrix}$$
$$R = \begin{bmatrix} 1 & 3 & 6 & 8 & 9 \end{bmatrix}$$

Compressed Row Sparse Matrix Format

Compressed Row Sparse Matrix Format

- Instead of storing all the values of a matrix, including the zero values, three dense matrices are constructed that represent the original
- Matrix 1: stores all of the non-zero values
- Matrix 2: stores the column position
- Matrix 3: stores the row position, last value =nnz+1

- Instead of storing n^2 elements, we need only 2nnz+n+1 storage locations.

Inverse of Sparse Matrix

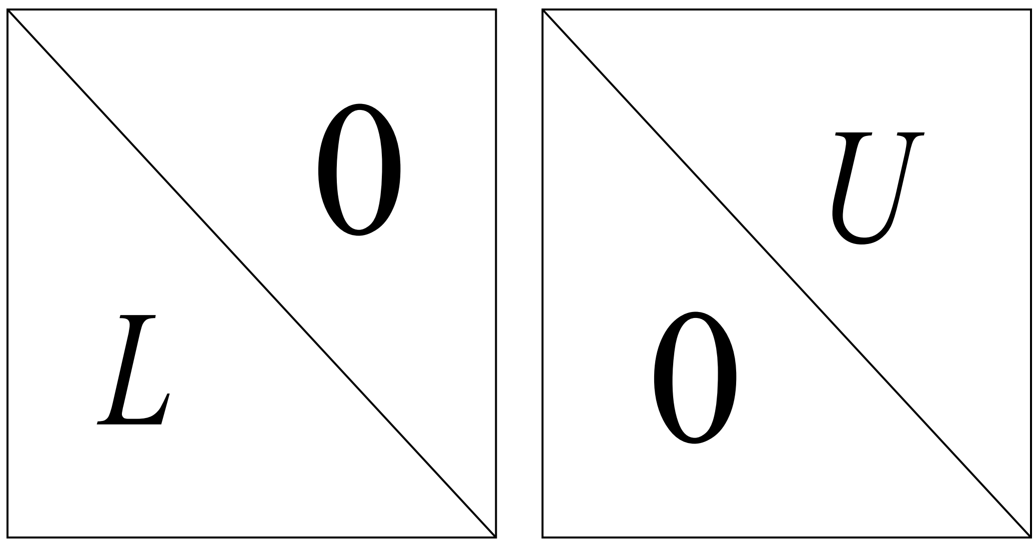
$$Ax = b$$

$$x = A^{-1}b$$

- A matrix is sparse
- Inverse A may not be sparse
- Especially true for large sparse matrices
- Therefore, we try not to inverse matrices

Triangular Factorization – LU Decomposition

- The A matrix is decomposed into 2 matrices, a lower triangular matrix (L), and an upper triangular matrix (U)
- Once the A matrix has been decomposed, simple Gaussian elimination can be used to solved for x



Triangular Factorization – LU Decomposition

- Step 1:
  - Decompose the A matrix into *L* and *U*
- Step 2:
  - Solve $Lz = b$ for z
  - Solve $Ux = z$ for x

$$Ax = b$$

$$LUx = b$$

$$\begin{cases} Lz = b \\ Ux = z \end{cases}$$

Crout's Method of LU Factorization

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} 1 & u_{12} & u_{13} \\ 0 & 1 & u_{23} \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} l_{11} & l_{11}u_{12} & l_{11}u_{13} \\ l_{21} & l_{21}u_{12} + l_{22} & l_{21}u_{13} + l_{22}u_{23} \\ l_{31} & l_{31}u_{12} + l_{32} & l_{31}u_{13} + l_{32}u_{23} + l_{33} \end{bmatrix}$$

$$\boxed{\begin{aligned} l_{ij} &= a_{ij}^{\ j} \\ u_{ij} &= \frac{a_{ij}^{\ i}}{a_{ii}^{\ i}} \\ a_{ij}^{\ k+1} &= a_{ij}^{\ k} - l_{ik}u_{kj} \end{aligned}}$$

$$L = \begin{bmatrix} a_{11} & 0 & 0 \\ a_{21} & a_{22} - a_{21}\,a_{12}/a_{11} & 0 \\ a_{31} & a_{32} - a_{31}\,a_{12}/a_{11} & a_{33} - a_{31}\,a_{13}/a_{11} - (a_{32} - a_{31}\,a_{12}/a_{11})\left[(a_{23} - a_{21}\,a_{13}/a_{11})/(a_{22} - a_{21}\,a_{12}/a_{11})\right] \end{bmatrix}$$

$$U = \begin{bmatrix} 1 & a_{12}/a_{11} & a_{13}/a_{11} \\ 0 & 1 & (a_{23} - a_{21}\,a_{13}/a_{11})/(a_{22} - a_{21}\,a_{12}/a_{11}) \\ 0 & 0 & 1 \end{bmatrix}$$

Example of LU Decomposition

Problem: Find the LU decomposition of the given matrix using Crout's Method

$$A = \begin{bmatrix} -2 & 1 & 0 \\ 1 & -2 & 1 \\ 0 & 1 & -4 \end{bmatrix}$$

Solution:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{32} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} 1 & u_{12} & u_{13} \\ 0 & 1 & u_{23} \\ 0 & 0 & 1 \end{bmatrix}$$

Example of LU Decomposition

Solution (*cont'd*):

1

$$l_{11} = a_{11}^1 = -2$$
$$l_{21} = a_{21}^1 = 1$$
$$l_{31} = a_{31}^1 = 0$$

$$u_{12} = \frac{a_{12}^1}{a_{11}^1} = \frac{1}{-2} = -0.5$$
$$u_{13} = \frac{a_{13}^1}{a_{11}^1} = \frac{0}{-2} = 0$$

$$a_{22}^2 = a_{22}^1 - l_{21}u_{12} = -2 - 1\times(-0.5) = -1.5$$
$$a_{23}^2 = a_{23}^1 - l_{21}u_{13} = 1 - 1\times 0 = 1$$
$$a_{32}^2 = a_{32}^1 - l_{31}u_{12} = 1 - 0\times(-0.5) = 1$$
$$a_{33}^2 = a_{33}^1 - l_{31}u_{13} = -4 - 0\times(0) = -4$$

2

$$l_{22} = a_{22}^2 = -1.5$$
$$l_{32} = a_{32}^2 = 1$$

$$u_{23} = \frac{a_{23}^2}{a_{22}^2} = \frac{1}{-1.5} = -0.667$$

$$a_{33}^3 = a_{33}^2 - l_{32}u_{23} = -4 - 1\times(-0.667) = -3.333$$

3

$$l_{33} = a_{33}^3 = -3.333$$

$$L = \begin{bmatrix} -2 & 0 & 0 \\ 1 & -1.5 & 0 \\ 0 & 1 & -3.33 \end{bmatrix} \qquad U = \begin{bmatrix} 1 & -.5 & 0 \\ 0 & 1 & -.667 \\ 0 & 0 & 1 \end{bmatrix}$$

$$A = \begin{bmatrix} -2 & 1 & 0 \\ 1 & -2 & 1 \\ 0 & 1 & -4 \end{bmatrix} = \begin{bmatrix} -2 & 0 & 0 \\ 1 & -1.5 & 0 \\ 0 & 1 & -3.33 \end{bmatrix} \begin{bmatrix} 1 & -.5 & 0 \\ 0 & 1 & -.667 \\ 0 & 0 & 1 \end{bmatrix}$$

Example – Solution of Linear Equations

Problem: Solve for *x* in the following equations

$$\begin{bmatrix} -2 & 1 & 0 \\ 1 & -2 & 1 \\ 0 & 1 & -4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

Solution:

$$\begin{bmatrix} -2 & 0 & 0 \\ 1 & -1.5 & 0 \\ 0 & 1 & -3.33 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

$$\begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} -.5 \\ -1.667 \\ -1.401 \end{bmatrix}$$

$$\begin{bmatrix} 1 & .5 & 0 \\ 0 & 1 & .667 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -.5 \\ -1.667 \\ -1.401 \end{bmatrix}$$

$$\boxed{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -1.8 \\ -2.6 \\ -1.4 \end{bmatrix}}$$

Dolittle's Method of LU Factorization

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}$$

- Assignment (due: Sept 14):
  - 1. Derive the formula for Dolittle's Method
  - 2. Find the LU decomposition of the matrix in the previous example using Dolittle's Method
  - 3. Solve the linear equations using Dolittle's Method

Existence of Triangular Factorization

- If the leading principal minors of the matrix A are all nonzero, then the matrix A is non-singular and the triangular factorization exists
- This condition is satisfied if A is symmetric positive definite, or strictly diagonally dominant, or irreducibly diagonal dominant
- If we allow row or column permutations, then for any non-singular matrix A there exists permutation matrices P and Q such that AP and QA have triangular factorizations

Example of Non-existence of Triangular Factorization

Problem: Solve for *x* in the following equations

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

By inspection, the solution is $x_1=1$ & $x_2=1$

- A solution via the Crout's LU decomposition is not possible because of the zero value of a_{11}

$$l_{11} = a_{11} = 0$$
$$l_{21} = a_{21} = 1$$
$$u_{12} = \frac{a_{12}}{l_{11}} = undefined$$

$$\boxed{\begin{aligned} l_{ij} &= a_{ij}^{\ j} \\ u_{ij} &= \frac{a_{ij}^{\ i}}{a_{ii}^{\ i}} \\ a_{ij}^{\ k+1} &= a_{ij}^{\ k} - l_{ik}u_{kj} \end{aligned}}$$

Alternative Approach

- An alternative approach might be to use a value for a_{11} that is very small but greater than zero, e.g. 10^{-10}

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \approx \begin{bmatrix} 10^{-10} & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

$$l_{11} = a_{11} = 10^{-10}$$

$$l_{21} = a_{21} = 1$$

$$u_{12} = \frac{a_{12}}{l_{11}} = 10^{10}$$

$$l_{22} = a_{22} - l_{21}u_{12} = 1 - 1 \cdot 10^{10}$$

$$\begin{bmatrix} 10^{-10} & 0 \\ 1 & 1 - 10^{10} \end{bmatrix} \begin{bmatrix} 1 & 10^{10} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Alternative Approach *cont'd*

$$l_{11} z_1 = 1$$

$$l_{21} z_1 + l_{22} z_2 = 2$$

$$z_1 = \frac{1}{l_{11}} = \frac{1}{10^{-10}} = 10^{10}$$

$$z_2 = \frac{2 - l_{21} z_1}{l_{22}} = \frac{2 - 10^{10}}{1 - 10^{10}}$$

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} 10^{10} \\ \approx 1 \end{bmatrix}$$

$$x_2 = z_2 = 1$$

$$x_1 = z_1 - u_{12} z_2 = 10^{10} - 10^{10} \cdot 1 = 0$$

- This illustrates a classic problem in numeric computation that occurs when small values are used that are orders of magnitude smaller than the other values

Matrix Permutations

- One possible solution is to use a matrix permutation to ensure that the principle minors contain non-zero values

$$Ax = b$$

$$PAx = Pb$$

$$P = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

Matrix Permutations *Cont'd*

- Now LU decomposition is possible.
- After decomposition, with the proper form the forward and back substitution will yield the proper value of x_1 & x_2
- Most software packages will include pivoting as a standard part of the solution engine

Questions?