UNIVERSITY OF MANITOBA

Matrix Solution Methods and Sparsity Implications

Hui Ding
Shengtao Fan

Advisor: Dr. Ani Gole

Department of Electrical and Computer Engineering
University of Manitoba

Outline

1 Introduction (EMT)

Parse the System Information
Parse the Devices

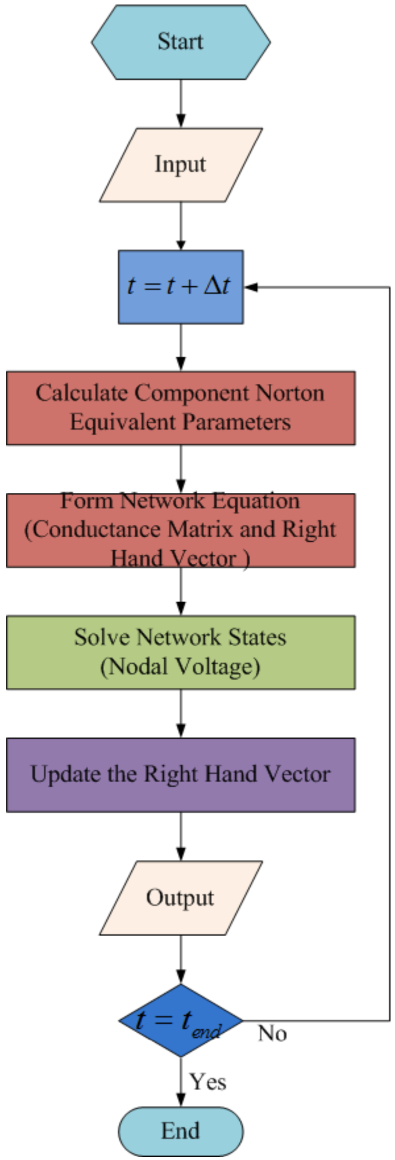
Norton Equivalent Calculation

Form Y Matrix and J Vector

Solve the System

Update Dynamic Devices

Save the Outputs to File



1 Introduction (Reference Books)

EMT Reference:

- 1) H.W. Dommel, EMTP Theory Book (2nd edition) , Microtran Power System Analysis Corporation, Vancouver, BC. 1992
- 2) N. Watson and J. Arrillaga, Power Systems Electromagnetic Transients Simulation, IEE Power and Energy Series, No. 39, IEE Press, 2003, UK,

1 Introduction (Matlab Solver)

INV is slow and inaccurate. Use A\b for INV(A)*b, and b/A for b*inv(A).

Explanation

M-Lint has detected a call to `inv` in a multiplication operation.

The inverse of a matrix is primarily of theoretical value, and rarely finds any use in practical computations. Never use the inverse of a matrix to solve a linear system `Ax=b` with `x=inv(A)*b`, because it is slow and inaccurate.

Suggested Action

Instead of multiplying by the inverse, use matrix right division (/) or matrix left division (\). That is:

- Replace `inv(A)*b` with `A\b`
- Replace `b*inv(A)` with `b/A`

Frequently, an application needs to solve a series of related linear systems `Ax=b`, where `A` does not change, but `b` does. In this case, use `lu`, `chol`, or `qr` instead of `inv`, depending on the matrix type.

1 Introduction (Matrix Solution Methods)

- Nowadays, easy access to computers makes the solution of large sets of linear algebraic equations possible and practical.
  - Direct Method

    By taking the advantage of **“sparsity”**
  - Iterative Method

    often the only choice for nonlinear equations, also useful even for linear problems involving a large number of variables
- Direct solution methods were often preferred to iterative methods in real applications because of their robustness and predictable behavior.

1 Introduction (Reference Books)

Direct Methods Reference:

- 1) Davis, T.A., **Direct methods for sparse linear systems**. Vol. 2. 2006: Society for Industrial Mathematics.
- 2) George, A., J. Liu, and E. Ng, Computer Solution of Sparse Linear Systems. 1994.
- 3) Duff, I., A. Erisman, and J. Reid, Direct Methods for Sparse Matrices, 1986, Oxford: Clarendon Press.
- 4) Østerby, O. and Z. Zlatev, Direct methods for sparse matrices. DAIMI PB, 1980. 9(123).

Iterative Methods Reference:

- 5) Saad, Y., Iterative methods for sparse linear systems. 2003: Society for Industrial and Applied Mathematics.

Outline

1 Introduction

2 Matrix Solution Methods

2.1 Graphical method

2.2 Cramer's rule

2.3 Gaussian Elimination

2.4 LU Factorization

3 Sparsity Implication

2 Matrix Solution Methods

- For small number of equations ($n \leq 3$) can be solved readily by simple techniques.
  - Graphical method
  - Cramer's rule

- For large number of equations ($n > 3$) can be solved readily by other techniques.
  - Elimination method
  - LU method

2.1 Graphical method

- For two equations:

$$a_{11}x_1 + a_{12}x_2 = b_1$$

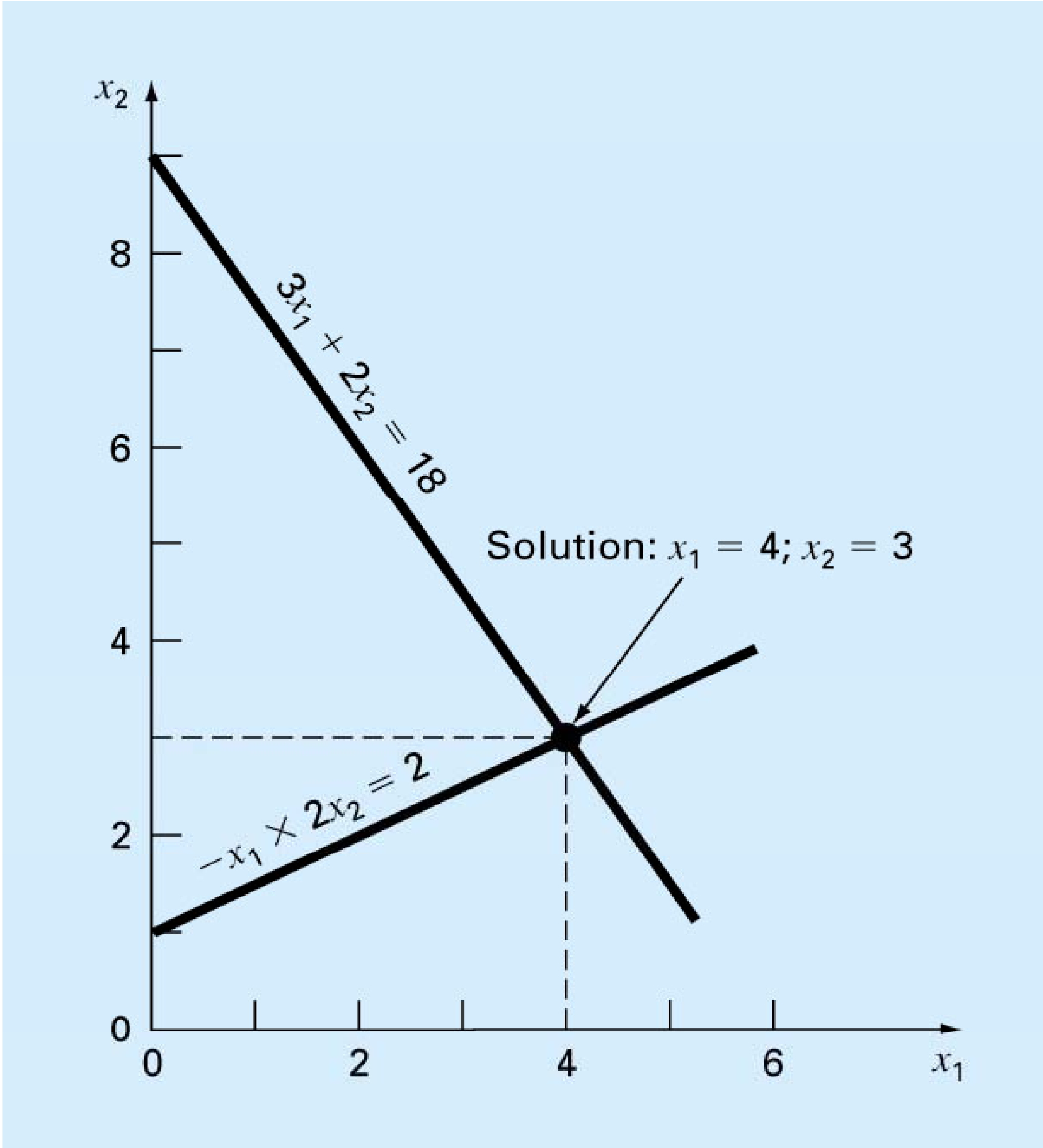
$$a_{21}x_1 + a_{22}x_2 = b_2$$

- Solve both equations for x_2:

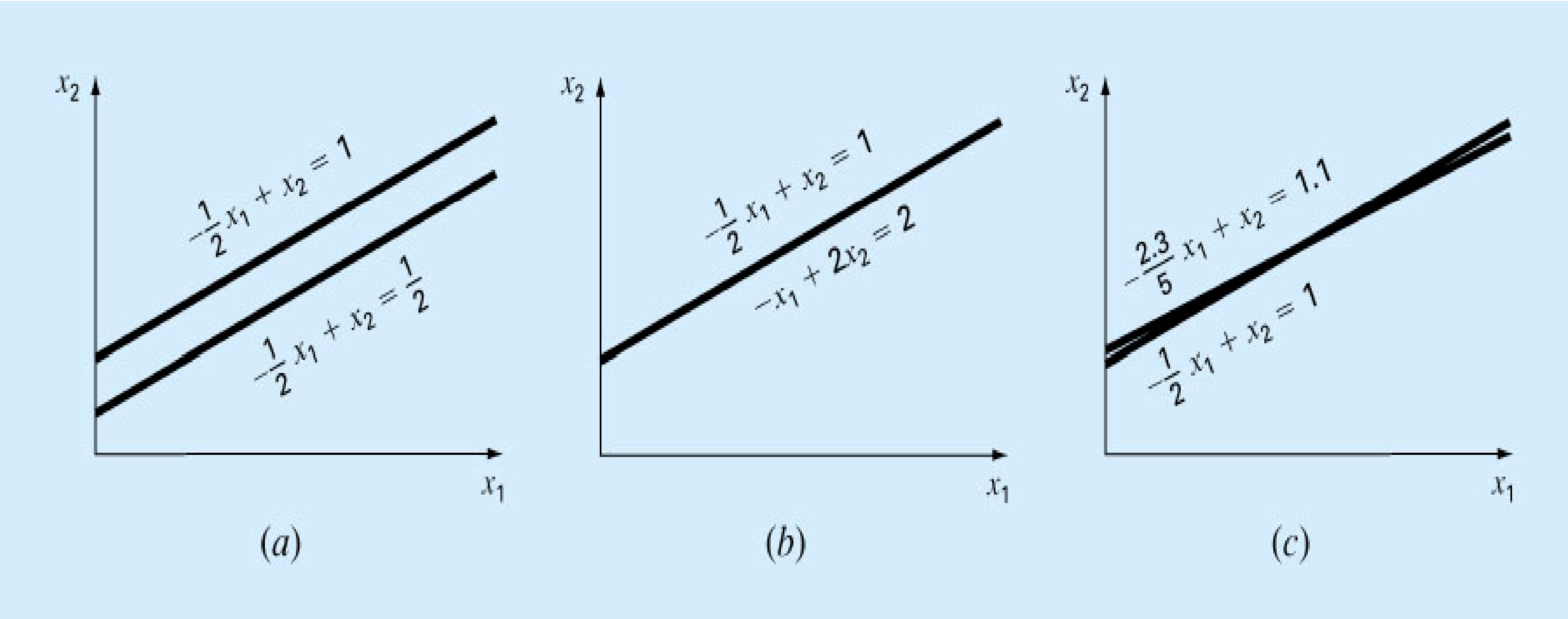
$$\left.\begin{aligned} x_2 &= -\left(\frac{a_{11}}{a_{12}}\right)x_1 + \frac{b_1}{a_{12}} \\ x_2 &= -\left(\frac{a_{21}}{a_{22}}\right)x_1 + \frac{b_2}{a_{22}} \end{aligned}\right\} \Rightarrow x_2 = (\text{slope})x_1 + \text{intercept}$$

2.1 Graphical method

- Plot x_2 vs. x_1 on rectilinear paper, the intersection of the lines present the solution.



2.1 Graphical method



(a) No solution
(b) Infinite solutions
(c) Ill-conditioned (Slopes are too close)

2.2 Cramer's rule

- If [A] is order 1, then [A] has one element:

  $A=[a_{11}]$

  the determinant is $D=a_{11}$

- For a square matrix of order 2

$$A=\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

the determinant is $D= a_{11} a_{22}-a_{21} a_{12}$

2.2 Cramer's rule

- For a square matrix of order 3

$$D = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = a_{11}D_{11} - a_{12}D_{12} + a_{13}D_{13}$$

$$D_{11} = \begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} = a_{22}\,a_{33} - a_{32}\,a_{23}$$

$$D_{12} = \begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} = a_{21}\,a_{33} - a_{31}\,a_{23}$$

$$D_{13} = \begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix} = a_{21}\,a_{32} - a_{31}\,a_{22}$$

2.2 Cramer's rule

- For a square matrix of order 3

$$x_1 = \frac{\begin{vmatrix} b_1 & a_{12} & a_{13} \\ b_2 & a_{22} & a_{23} \\ b_3 & a_{32} & a_{33} \end{vmatrix}}{\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix}} = \frac{\begin{vmatrix} b_1 & a_{12} & a_{13} \\ b_2 & a_{22} & a_{23} \\ b_3 & a_{32} & a_{33} \end{vmatrix}}{D}$$

$$x_2 = \dots$$

$$x_3 = \dots$$

2.3 Gaussian Elimination

- the technique for *n* equations consists of two phases:
  - Forward elimination of unknowns
  - Back substitution

Forward elimination

$$\left[\begin{array}{ccc|c} a_{11} & a_{12} & a_{13} & c_1 \\ a_{21} & a_{22} & a_{23} & c_2 \\ a_{31} & a_{32} & a_{33} & c_3 \end{array}\right]$$

$$\Downarrow$$

$$\left[\begin{array}{ccc|c} a_{11} & a_{12} & a_{13} & c_1 \\ & a'_{22} & a'_{23} & c'_2 \\ & & a''_{33} & c''_3 \end{array}\right]$$

$$\Downarrow$$

Back substitution

$$x_3 = c''_3 / a''_{33}$$

$$x_2 = (c'_2 - a'_{23}x_3)/a'_{22}$$

$$x_1 = (c_1 - a_{12}x_2 - a_{13}x_3)/a_{11}$$

2.3 Gaussian Elimination

$$\begin{cases} 2x + y - z = 8 & \text{(L1)} \\ -3x - y + 2z = -11 & \text{(L2)} \\ -2x + y + 2z = -3 & \text{(L3)} \end{cases}$$

$$(1)\begin{cases} L_2 + \frac{3}{2}L_1 \to L_2 \\ L_3 + L_1 \to L3 \end{cases} \to \begin{cases} 2x + y - z = 8 & \text{(L1)} \\ 0 + 1/2\,y + 1/2z = 1 & \text{(L2)} \\ 0 + 2\,y + z = 5 & \text{(L3)} \end{cases}$$

$$(2)\, L_3 + (-4)L_2 \to L3 \to \begin{cases} 2x + y - z = 8 & \text{(L1)} \\ 0 + 1/2\,y + 1/2z = 1 & \text{(L2)} \\ 0 + 0 + -z = 1 & \text{(L3)} \end{cases}$$

2.4 LU Factorization

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

$Case:$

$$\begin{bmatrix} 4 & 3 \\ 6 & 3 \end{bmatrix}=\begin{bmatrix} l_{11} & 0 \\ l_{21} & l_{22} \end{bmatrix}\begin{bmatrix} u_{11} & u_{12} \\ 0 & u_{22} \end{bmatrix}$$

$$\begin{cases} l_{11}\cdot u_{11}+0\cdot 0 &=4 \\ l_{11}\cdot u_{12}+0\cdot u_{22} &=3 \\ l_{21}\cdot u_{11}+l_{22}\cdot 0 &=6 \\ l_{21}\cdot u_{12}+l_{22}\cdot u_{22} &=3 \end{cases}$$

2.4 LU Factorization

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

$Case:$

$$\begin{bmatrix} 4 & 3 \\ 6 & 3 \end{bmatrix}=\begin{bmatrix} l_{11} & 0 \\ l_{21} & l_{22} \end{bmatrix}\begin{bmatrix} u_{11} & u_{12} \\ 0 & u_{22} \end{bmatrix}$$

$$\begin{cases} l_{11}\cdot u_{11}+0\cdot 0 = 4 \\ l_{11}\cdot u_{12}+0\cdot u_{22} = 3 \\ l_{21}\cdot u_{11}+l_{22}\cdot 0 = 6 \\ l_{21}\cdot u_{12}+l_{22}\cdot u_{22} = 3 \end{cases} if \begin{cases} l_{11}=1 \\ l_{22}=1 \end{cases} \rightarrow \begin{cases} u_{11}+0\cdot 0 = 4 \\ u_{12}+0\cdot u_{22} = 3 \\ l_{21}\cdot u_{11}+0 = 6 \\ l_{21}\cdot u_{12}+u_{22} = 3 \end{cases}$$

2.4 LU Solver

$$\mathbf{Ax} = \mathbf{b} \rightarrow \mathbf{LUx} = \mathbf{b} \rightarrow \begin{cases} \mathbf{Ly} = \mathbf{b} \\ \mathbf{Ux} = \mathbf{y} \end{cases}$$

2.4 Gaussian Elimination & LU Factorization

$$\mathbf{A}=\begin{bmatrix}\times&\times&\times&\times\\\times&\times&\times&\times\\\times&\times&\times&\times\\\times&\times&\times&\times\end{bmatrix}\xrightarrow{\mathbf{E}_1}\begin{bmatrix}\times&\times&\times&\times\\0&\times&\times&\times\\0&\times&\times&\times\\0&\times&\times&\times\end{bmatrix}\xrightarrow{\mathbf{E}_2}\begin{bmatrix}\times&\times&\times&\times\\0&\times&\times&\times\\0&0&\times&\times\\0&0&\times&\times\end{bmatrix}\xrightarrow{\mathbf{E}_3}\begin{bmatrix}\times&\times&\times&\times\\0&\times&\times&\times\\0&0&\times&\times\\0&0&0&\times\end{bmatrix}$$

$$\mathbf{A} \qquad \mathbf{E}_1\mathbf{A} \qquad \mathbf{E}_2\mathbf{E}_1\mathbf{A} \qquad \mathbf{E}_3\mathbf{E}_2\mathbf{E}_1\mathbf{A}$$

Each $\mathbf{E}_i$ introduces zeros below diagonal of column i:

$$\mathbf{E}_3\mathbf{E}_2\mathbf{E}_1\mathbf{A}=\mathbf{U}\longrightarrow\mathbf{A}=\mathbf{L}\mathbf{U}\quad\text{where}\quad\mathbf{L}=(\mathbf{E}_3\mathbf{E}_2\mathbf{E}_1)^{-1}=\mathbf{E}_1^{-1}\mathbf{E}_2^{-1}\mathbf{E}_3^{-1}$$

2.4 Gaussian Elimination & LU Factorization

$$\begin{cases} \mathbf{E}_n\mathbf{E}_{n-1}\cdots\mathbf{E}_2\mathbf{E}_1\mathbf{A}\,\mathbf{x} = \mathbf{E}_n\mathbf{E}_{n-1}\cdots\mathbf{E}_2\mathbf{E}_1\mathbf{b} \longleftrightarrow \quad \mathbf{Ux} = \mathbf{E}_n\mathbf{E}_{n-1}\cdots\mathbf{E}_2\mathbf{E}_1\mathbf{b} \\ \mathbf{Ax} = \mathbf{b} \qquad\qquad\qquad\qquad \longleftrightarrow \mathbf{LUx} = \mathbf{b} \end{cases}$$

$$\begin{cases} \textit{For}\ \text{Gaussian Elimination:}\ \ \mathbf{Ux} = \mathbf{E}_n\mathbf{E}_{n-1}\cdots\mathbf{E}_2\mathbf{E}_1\mathbf{b} = \mathbf{L}^{-1}\mathbf{b} \\ \textit{For LU}: \qquad\qquad\qquad\qquad \mathbf{LUx} = \mathbf{b} \end{cases}$$

if vector **b** varies each simulation step,

For Gaussian Elimination we should store **U** and $\mathbf{L}^{-1}$ and calculate $\mathbf{L}^{-1}\mathbf{b}$.

For LU *method*, we should store **L** and **U**. As $\mathbf{L}^{-1}$ is a dense matrix, so Gaussian Elimination method is less efficient than LU method.

2.4 Arithmetic Complexity Analysis

- Gaussian elimination to solve a system of n equations for n unknowns requires
  - Divisions: $n(n-1)/2$
  - multiplications: $(2n^3 + 3n^2 - 5n)/6$
  - Subtractions: $(2n^3 + 3n^2 - 5n)/6$
  - Total of approximately: $2n^3 / 3$ operations.
  - Thus it has **arithmetic complexity** of $O(n^3)$.

- LU decomposition requires $2n^3 / 3$ floating point operations, if neglecting lower order terms

Outline

3.1 contents

- Sparse linear equation system
- Storage of sparse linear equation
- LU factorization considering sparsity
- Typical sparse solver

3.2 Sparse linear equation system

- A sparse matrix is a matrix populated primarily with zeros.
- Why the coefficient matrix of linear equation system is sparse?
- Sparsity comes from the loose coupling of systems.

3.2 How to utilize the sparsity

- The required memory will be greatly reduced, if proper data structure is used to avoid storing of the zero elements.

- The number of operations is also greatly reduced, if the operations involved zero elements are avoided.

- Without sparse techniques, it is impractical to solve some very large systems with direct method.

3.3 Storage of sparse linear equation

- Static or dynamic structures can be used to store sparse matrix.
- The triplet form and compressed-column form / compressed-row form are widely used.

$$\mathbf{A} = \begin{bmatrix} 1.0 & 0 & 5.0 & 7.0 \\ 0 & 3.0 & 0 & 0 \\ 2.0 & 0 & 6.0 & 8.0 \\ 0 & 4.0 & 0 & 9.0 \end{bmatrix}$$

$$\begin{aligned}
\text{int } n &= 4; \\
\text{int } nnz &= 9; \\
\text{int } i[] &= \{\ 0,\ 2,\ 1,\ 3,\ 0,\ 2,\ 0,\ 2,\ 3\ \}; \\
\text{int } j[] &= \{\ 0,\ 0,\ 1,\ 1,\ 2,\ 2,\ 3,\ 3,\ 3\ \}; \\
\text{int } x[] &= \{\ 1.0,\ 2.0,\ 3.0,\ 4.0,\ 5.0,\ 6.0,\ 7.0,\ 8.0,\ 9.0\ \};
\end{aligned}$$

$$\begin{aligned}
\text{int n} &= 4; \\
\text{int p} &= \{\ 0,\ 2,\ 4,\ 6,\ 9\}; \\
\text{int i[]} &= \{\ 0,\ 2,\ 1,\ 3,\ 0,\ 2,\ 0,\ 2,\ 3\ \}; \\
\text{double x[]} &= \{\ 1.0,\ 2.0,\ 3.0,\ 4.0,\ 5.0,\ 6.0,\ 7.0,\ 8.0,\ 9.0\ \};
\end{aligned}$$

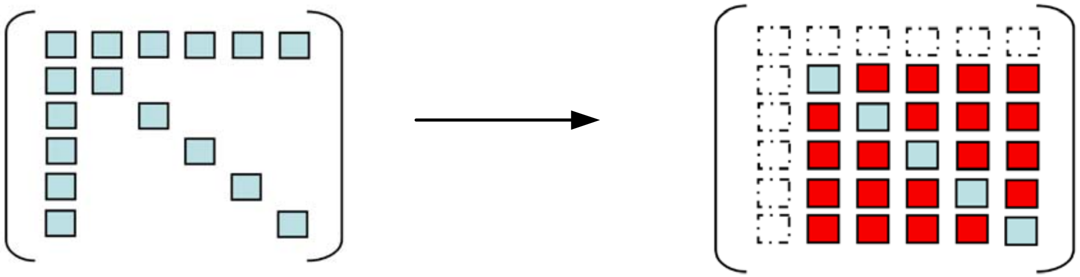
3.4 LU factorization considering sparsity

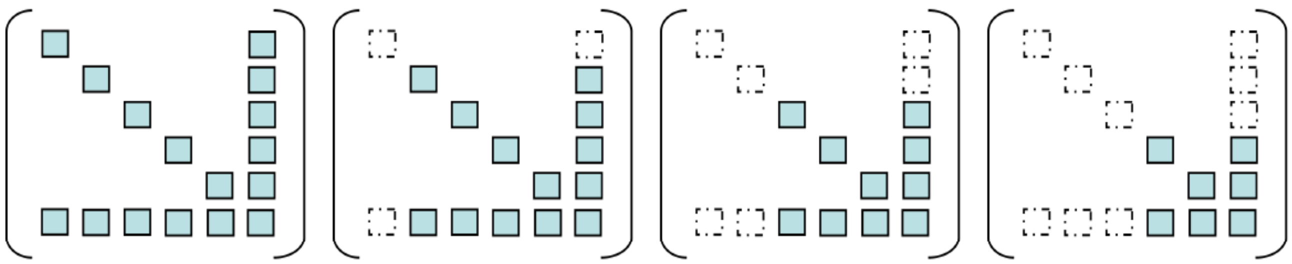
- Apply the gauss elimination process to matrix represented with sparse storage structure.
- The main step of Gauss Elimination or LU factorization is multiplying one row with a number and then adding it to another row.

non-zero		non-zero	
		↓	
non-zero	→	fill-in	

3.4 LU factorization

- Investigation shows that the number of fill-ins in Gauss Elimination process is greatly affected by ordering of the matrix.

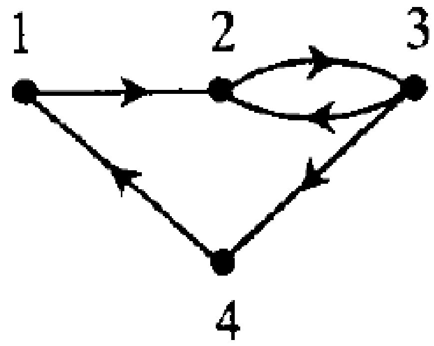




3.4 LU factorization

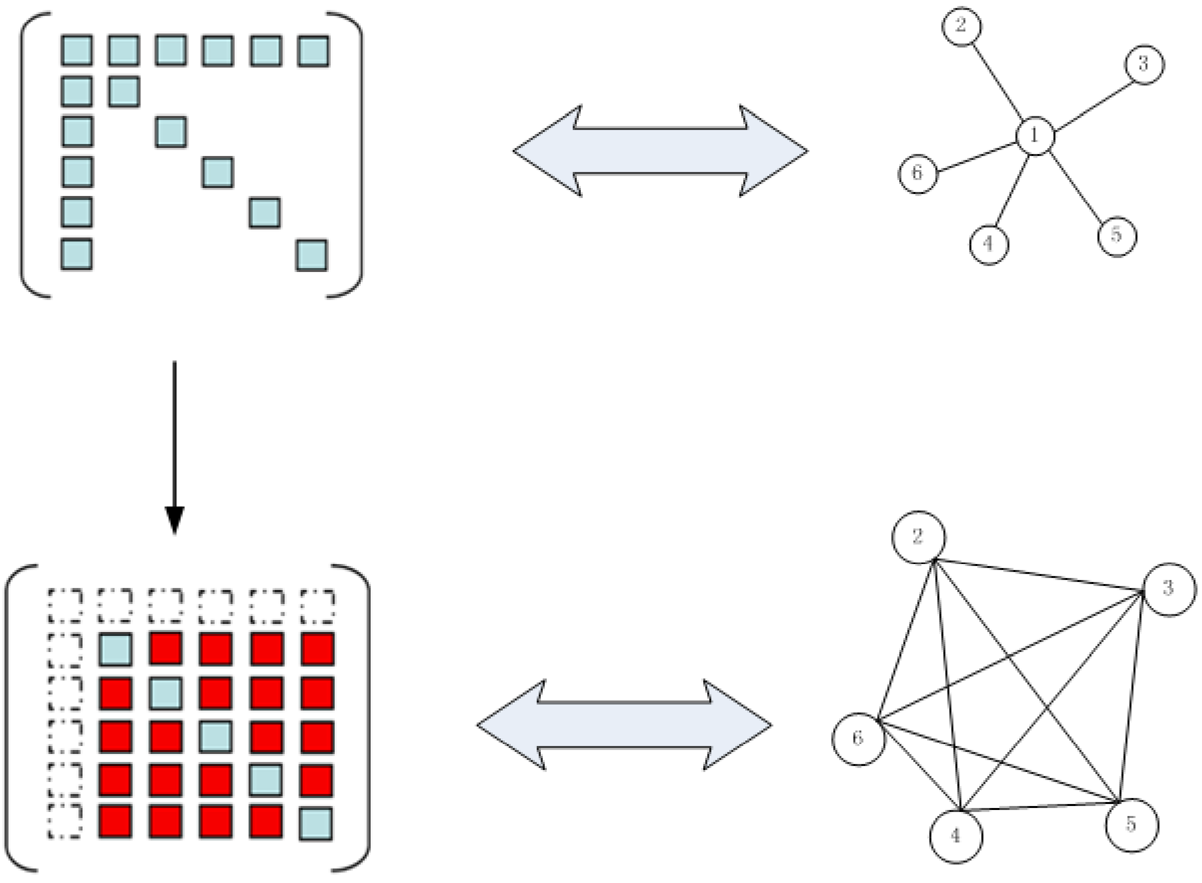
- The non-zero pattern of a square sparse matrix can be represented by a graph.
- For any square sparse matrix, the number of vertices in the graph equals to the order of the matrix.
- If aij is a none-zero entry, there is an edge from node i to node j in the directed graph.

$$\begin{pmatrix} \times & \times & & \\ & \times & \times & \\ & \times & \times & \times \\ \times & & & \times \end{pmatrix}$$



3.4 LU factorization

- For a symmetric matrix, a connection from node i to node j implies there must be a connection from node j to node i. So, the arrows may be dropped.



3.4 LU factorization

- The fill-in will greatly affect the operation numbers needed to perform Guass Elimination and Forward/Backward substitution.

- Therefore, we need to find the best ordering to generate the least fill-ins.

- However, the bad news is that: Determining the best ordering in the elimination process which results in the minimum number of fill-ins is NP-Hard .

3.4 LU factorization

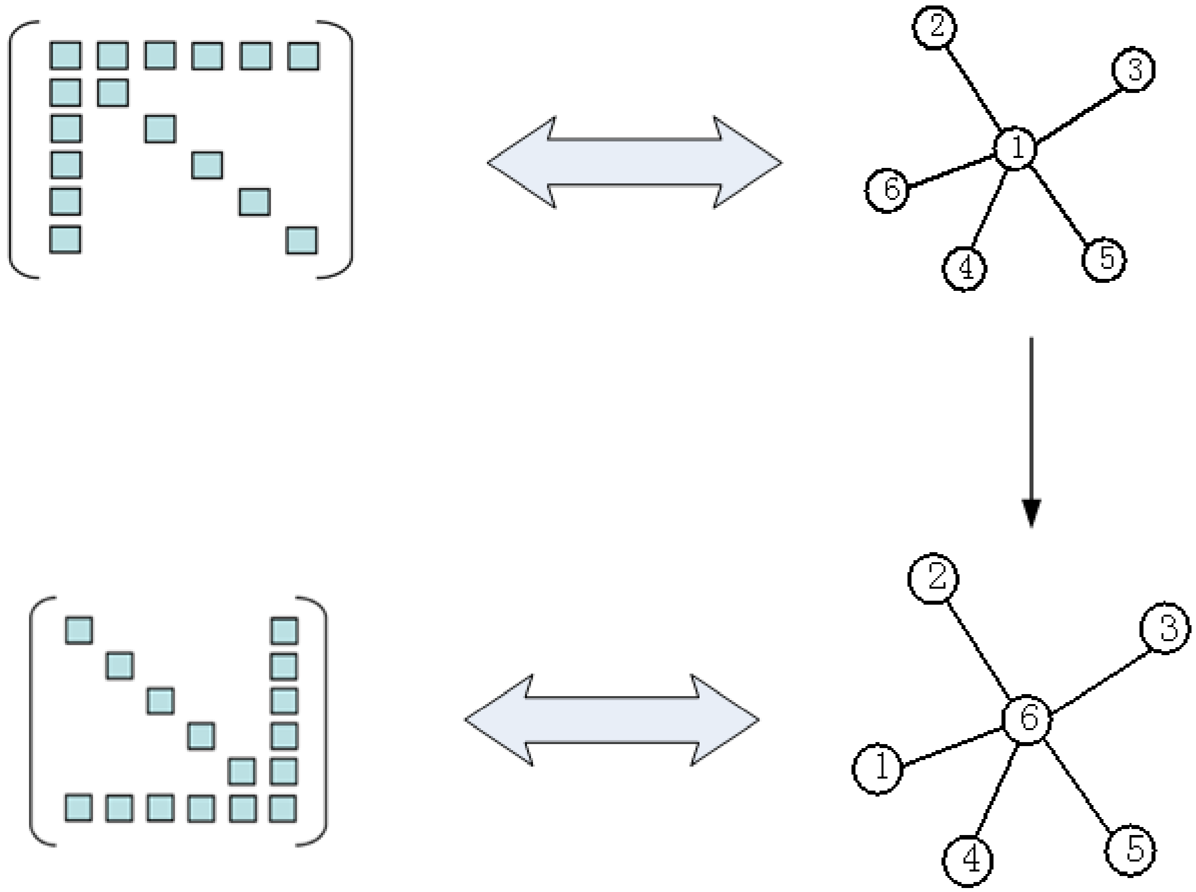
- This implies that minimizing the work of performing Gauss Elimination is more costly than itself, which is a P-Hard problem.

- The good news is that we can approximate this minimum using graph-based heuristics.

- One such heuristic is to always select the vertex with minimum degree.

3.4 LU factorization

- For example:

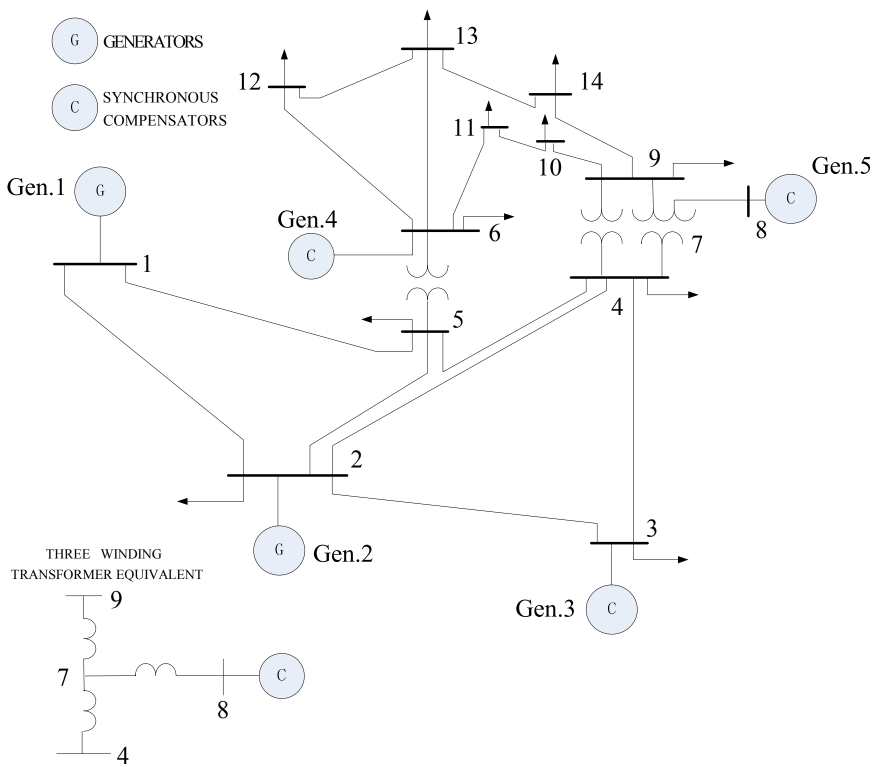


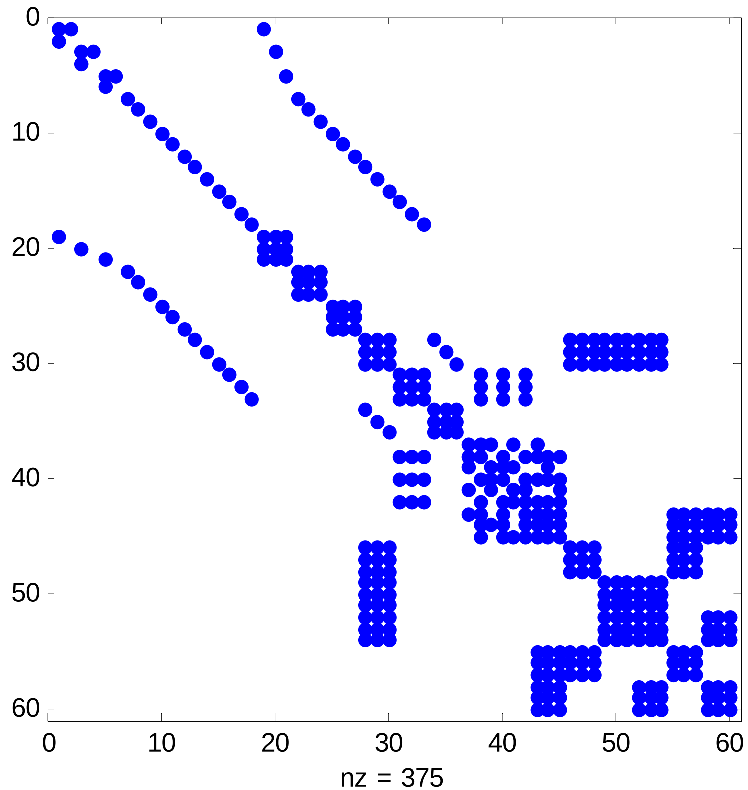
3.5 Typical sparse solver

- One typical solver consists of the following steps:
- Symbolic analysis
- Numerical factorization
- Forward and backward substitution

3.6 IEEE-14 system case

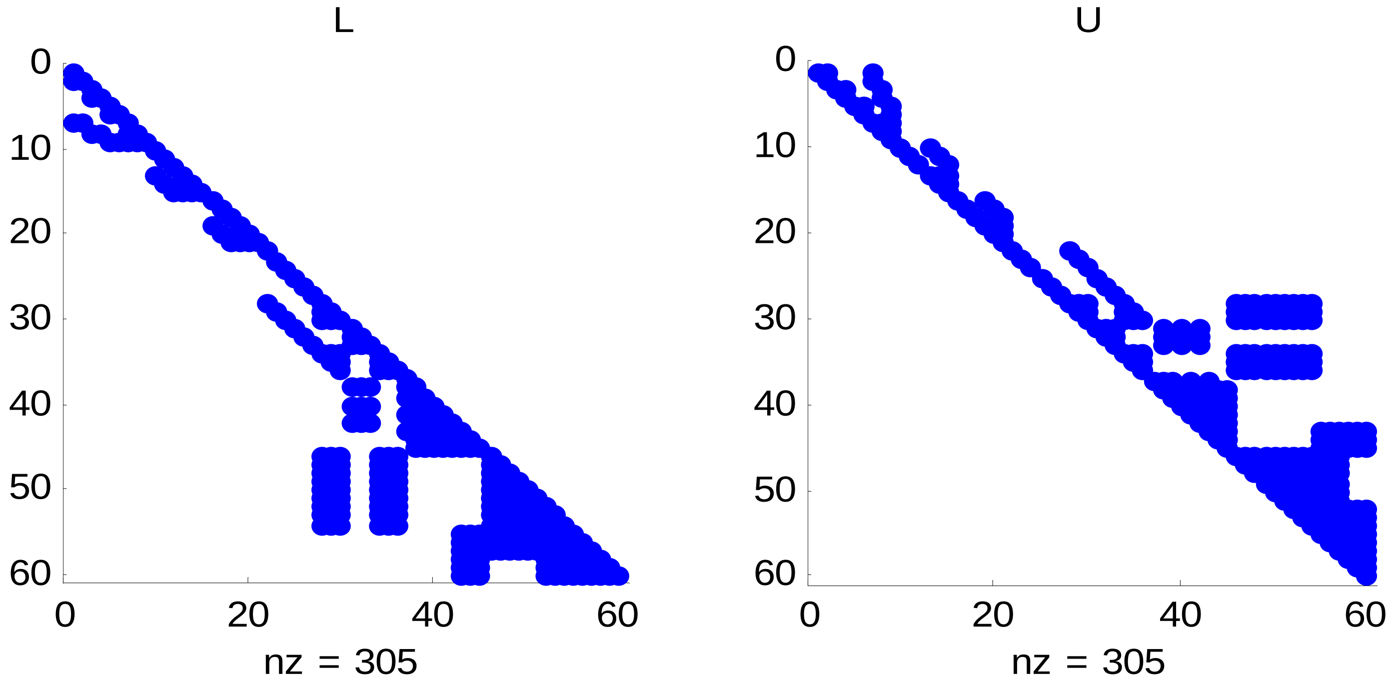
- The linear equation of the IEEE-14 system has 60 unknown variables.





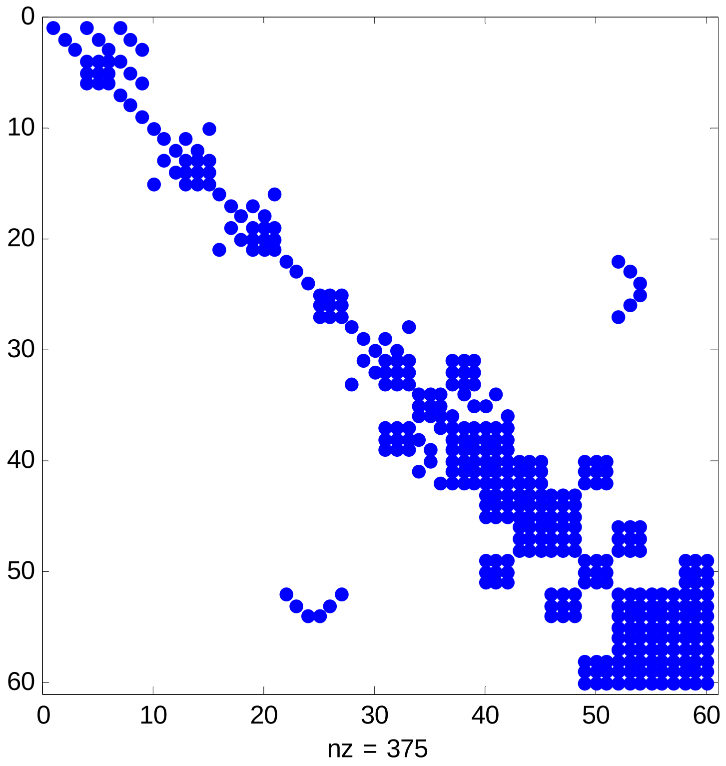
3.6 IEEE-14 system case

- LU factorization with natural ordering



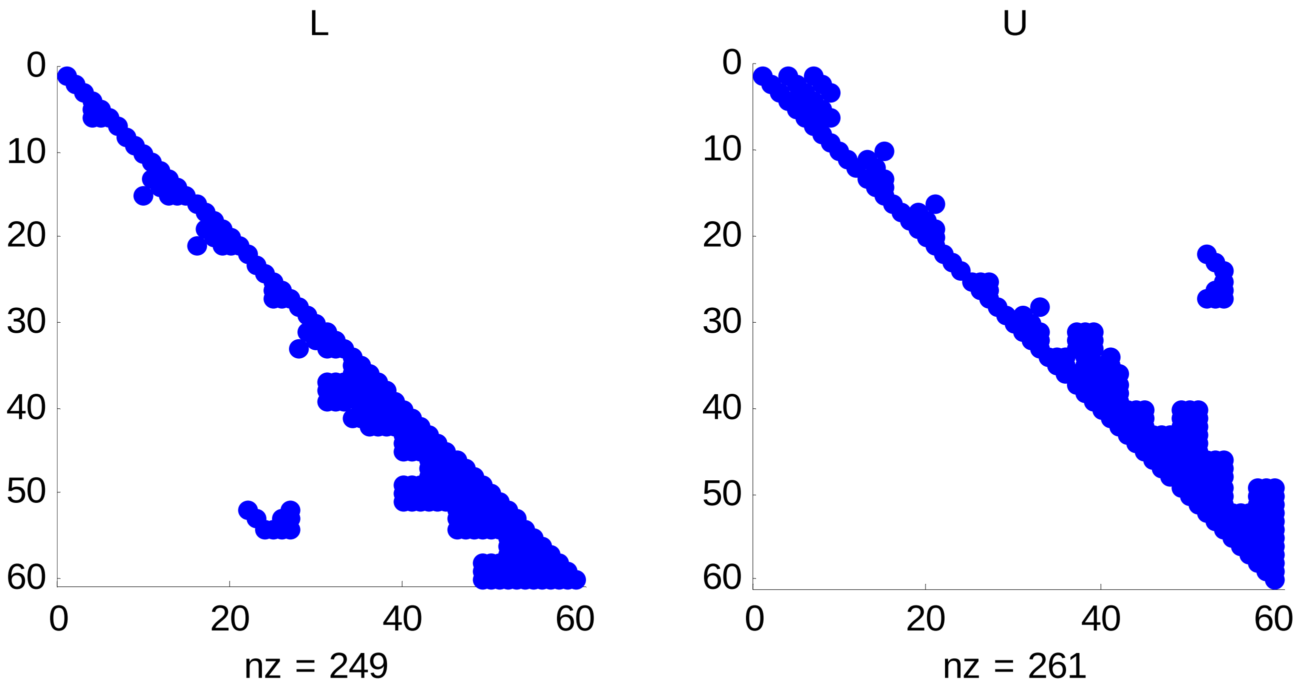
3.6 IEEE-14 system case

- Non-zero pattern after Approximate Minimum Degree ordering.



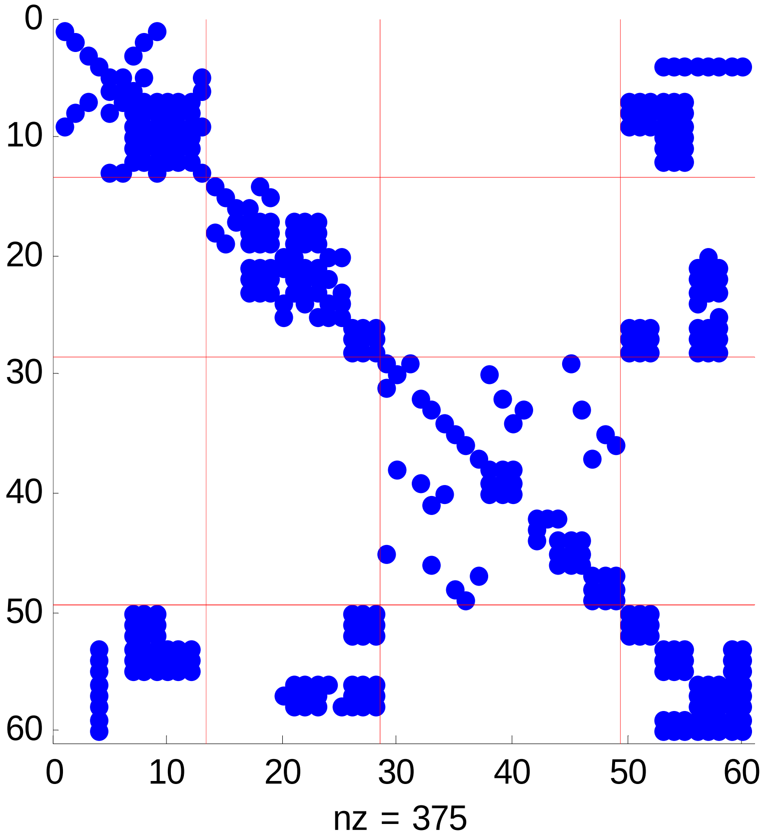
3.6 IEEE-14 system case

- LU factorization after AMD



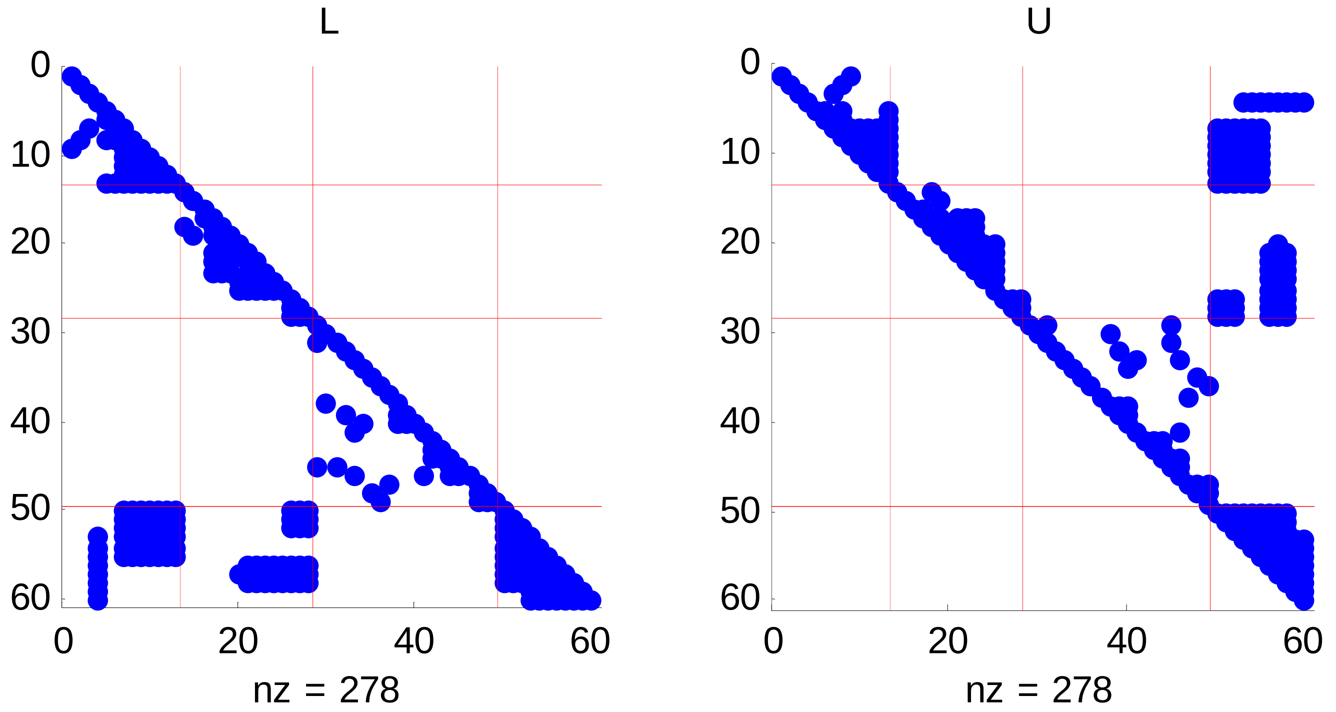
3.6 IEEE-14 system case

- Non-zero pattern of Bordered Block Diagonal (BBD) form.



3.6 IEEE-14 system case

- LU factorization of the BBD form



Discussion and Suggestion