

# Numerical Methods for Loadflow calculation



- Review: Loadflow calculation
- Solution of Nonlinear Algebraic Equations
  - Newton Raphson Method
  - One dimensional case
  - Multi dimensional case
- Sparse Matrices
  - Characteristics
  - Storage Methods
- Solution of large scale linear Algebraic Equations
  - Gauss Method
  - LDR Factorization
  - Methods for increasing efficiency

- Review: Loadflow calculation
- Solution of Nonlinear Algebraic Equations
  - Newton Raphson Method
  - One dimensional case
  - Multi dimensional case
- Sparse Matrices
  - Characteristics
  - Storage Methods
- Solution of large scale linear Algebraic Equations
  - Gauss Method
  - LDR Factorization
  - Methods for increasing efficiency

- Voltage Magnitude  $U_k$
- Voltage Angle  $\theta_k$
- Net Active Power  $P_k$  (Sum of total generation and loads)
- Net Reactive Power  $Q_k$  (Sum of total generation and loads)

$$\underline{S}_k = \underline{E}_k \cdot \underline{I}_k^* = \underline{E}_k \cdot \left( \sum_{m=1}^5 \underline{Y}_{km} \cdot \underline{E}_m \right)^*$$

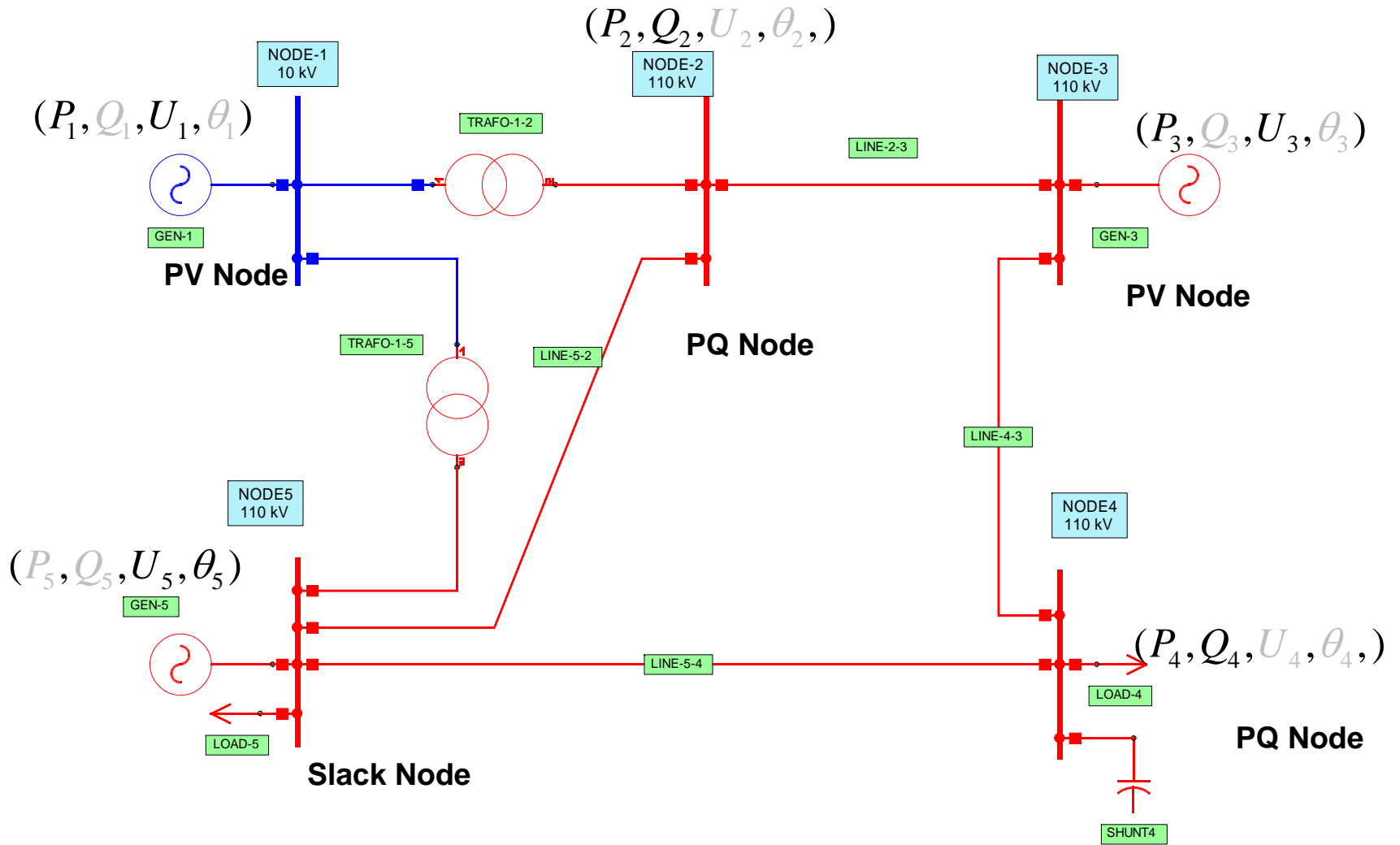
$$\underline{E}_k = U_k \cdot e^{j\theta_k}$$

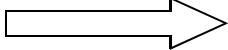
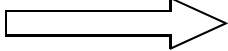
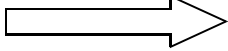
$$\underline{Y}_{km} = G_{km} + jB_{km}$$

$$\underline{S}_k = U_k \cdot e^{j\theta_k} \cdot \left( \sum_{m=1}^5 (G_{km} - jB_{km}) \cdot (U_m \cdot e^{-j\theta_m}) \right)$$

$$P_k = \text{real}(\underline{S}_k) = U_k \cdot \left( \sum_{m=1}^5 U_m \cdot (G_{km} \cos(\theta_k - \theta_m) + B_{km} \sin(\theta_k - \theta_m)) \right)$$

$$Q_k = \text{imag}(\underline{S}_k) = U_k \cdot \left( \sum_{m=1}^5 U_m \cdot (G_{km} \sin(\theta_k - \theta_m) - B_{km} \cos(\theta_k - \theta_m)) \right)$$



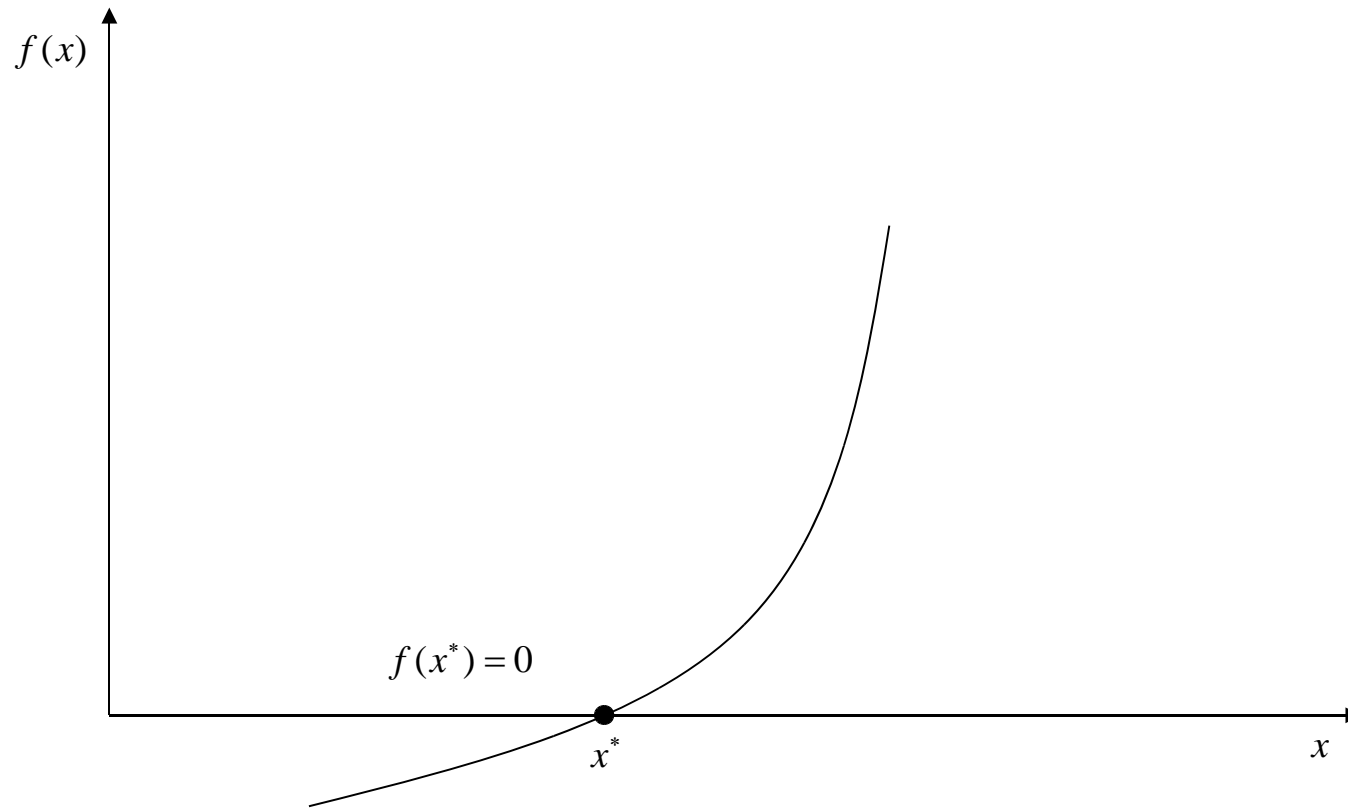
		Equations	Unknown
<b>PQ Node</b>		$P_k = P(U_1, \dots, U_N, \theta_1, \dots, \theta_N)$ $Q_k = Q(U_1, \dots, U_N, \theta_1, \dots, \theta_N)$	$U_k, \theta_k$
<b>PV Node</b>		$P_k = P(U_1, \dots, U_N, \theta_1, \dots, \theta_N)$	$\theta_k$
<b>Slack Node</b>		no equations needed	none

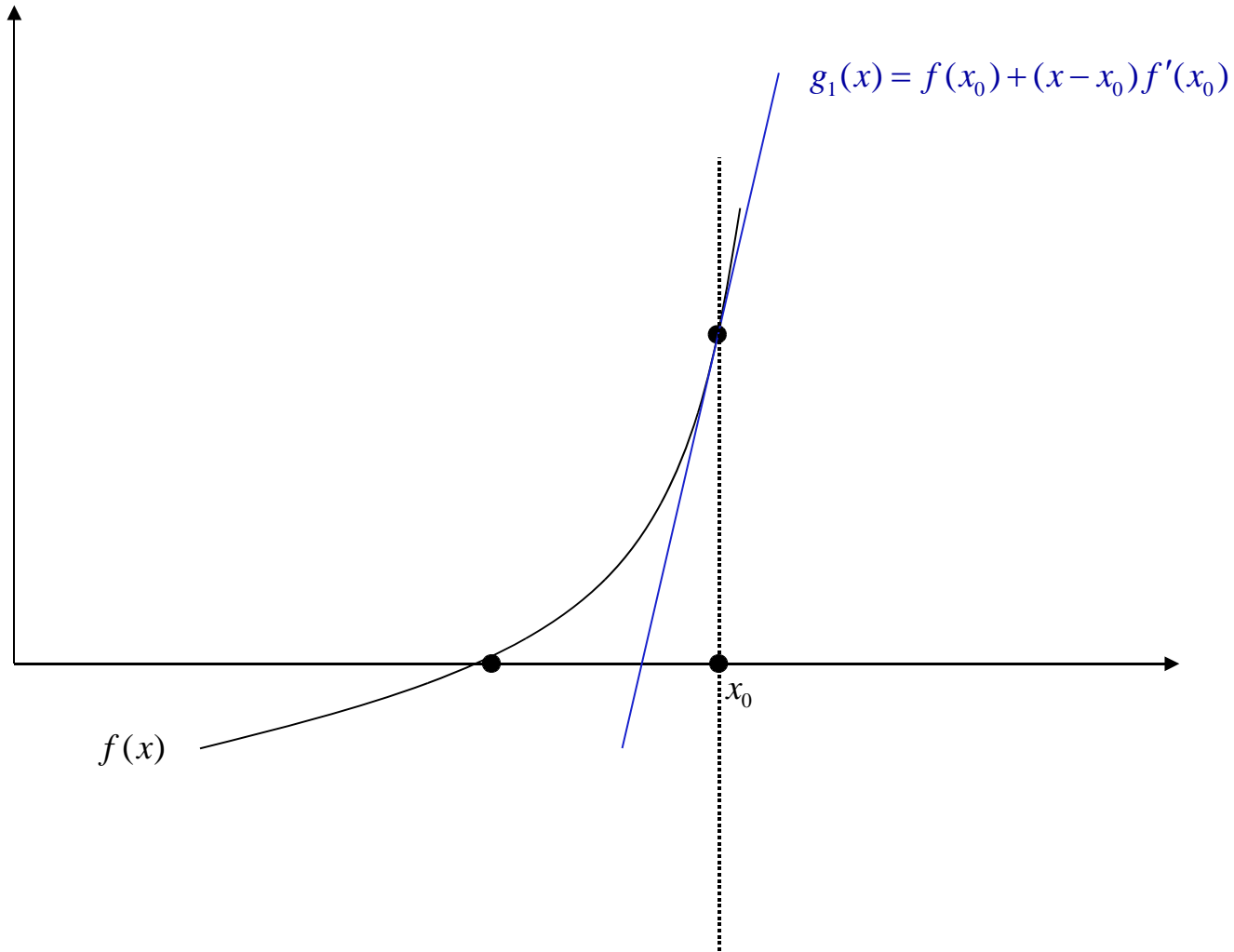
- We have  $2N_{PQ} + N_{PV}$  unknowns and equations

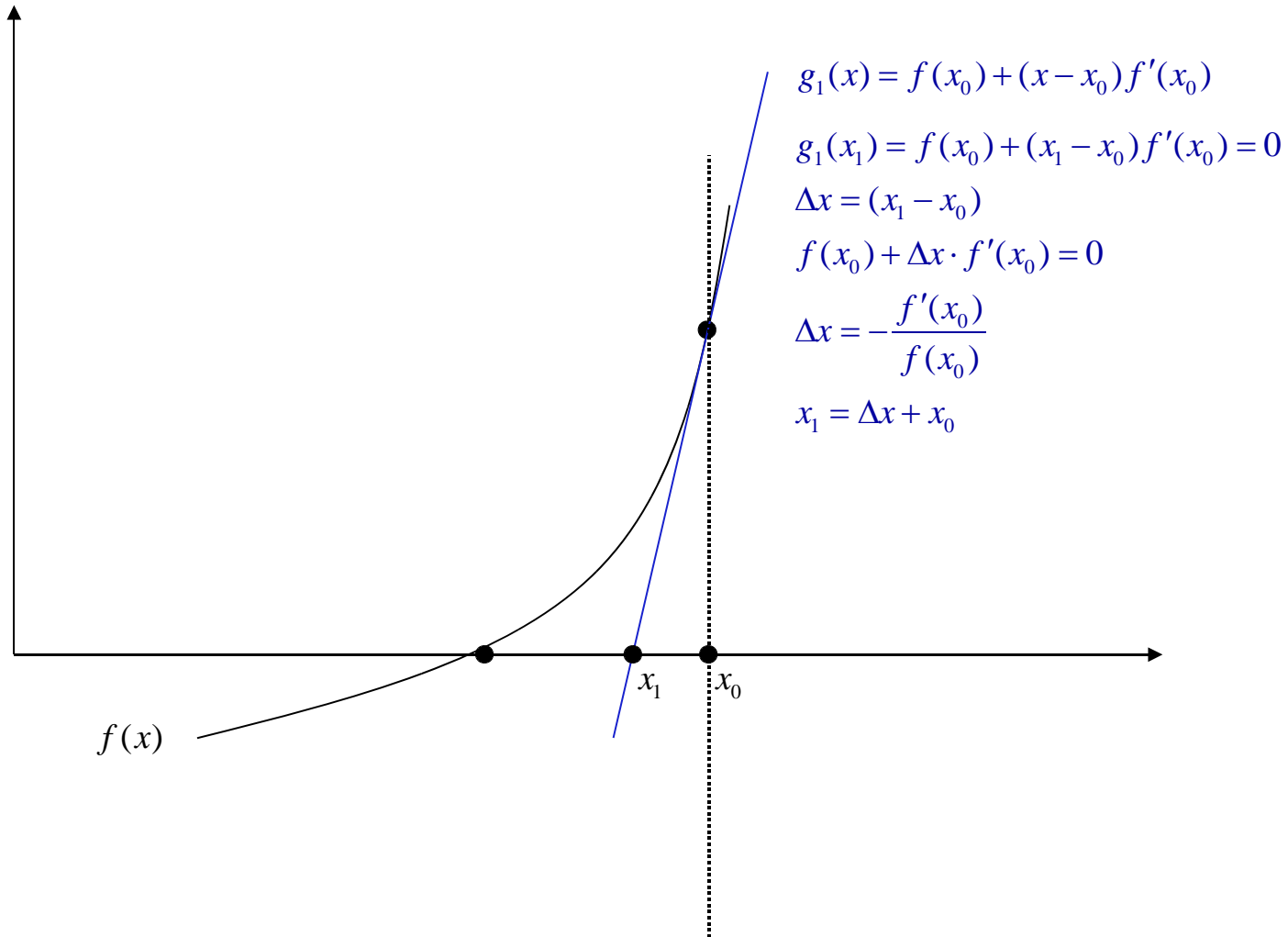
$$\begin{bmatrix} P_1 - P(U_1^v \dots U_N^v, \theta_1^v \dots \theta_N^v) \\ \dots \\ \dots \\ P_N - P(U_1^v \dots U_N^v, \theta_1^v \dots \theta_N^v) \\ Q_1 - Q(U_1^v \dots U_N^v, \theta_1^v \dots \theta_N^v) \\ \dots \\ \dots \\ Q_N - Q(U_1^v \dots U_N^v, \theta_1^v \dots \theta_N^v) \end{bmatrix} = 0 \quad \Rightarrow \quad f(x) = 0$$

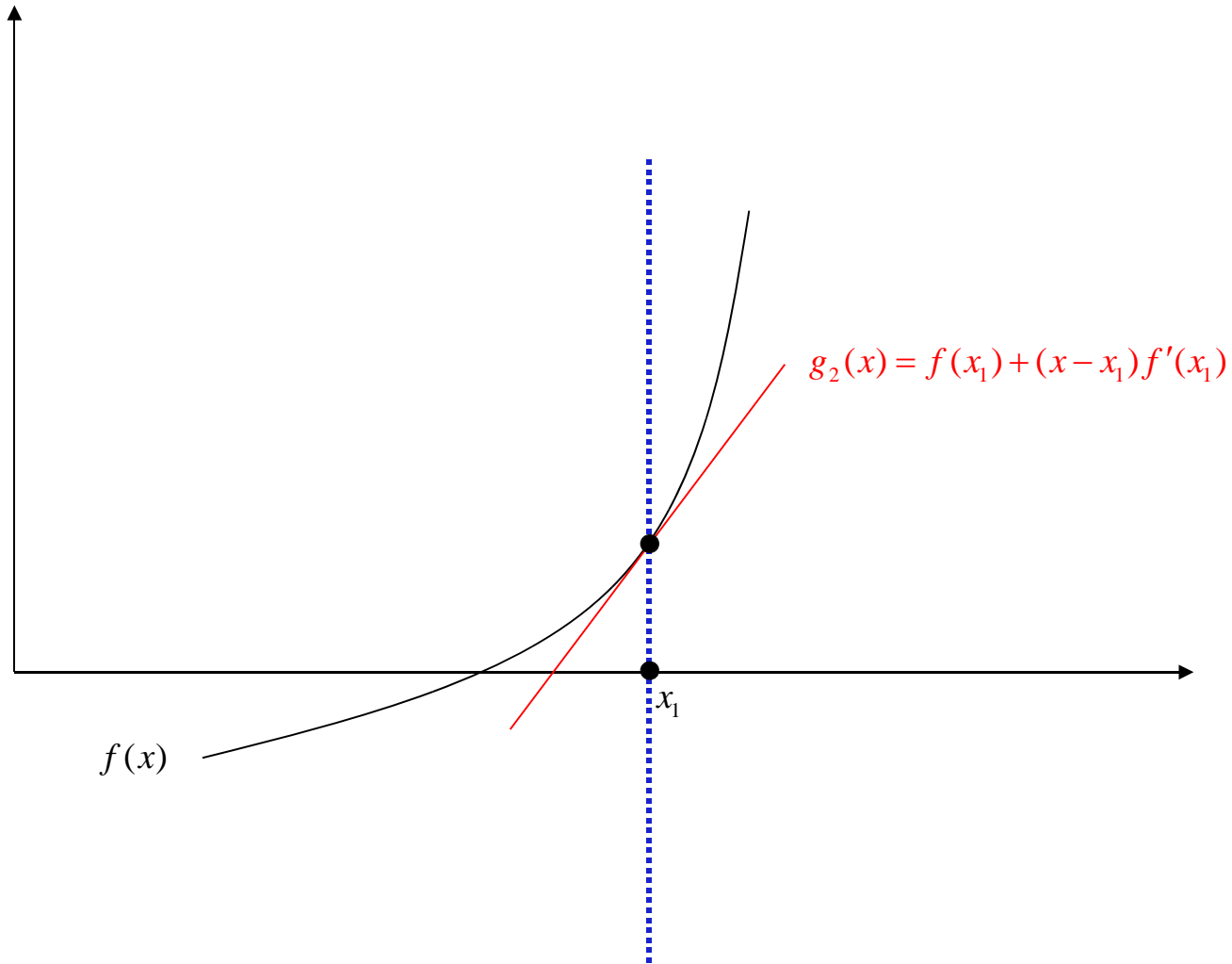
- Large scale, nonlinear, algebraic equation system
- Analytically not solvable
- Solution with numerical methods

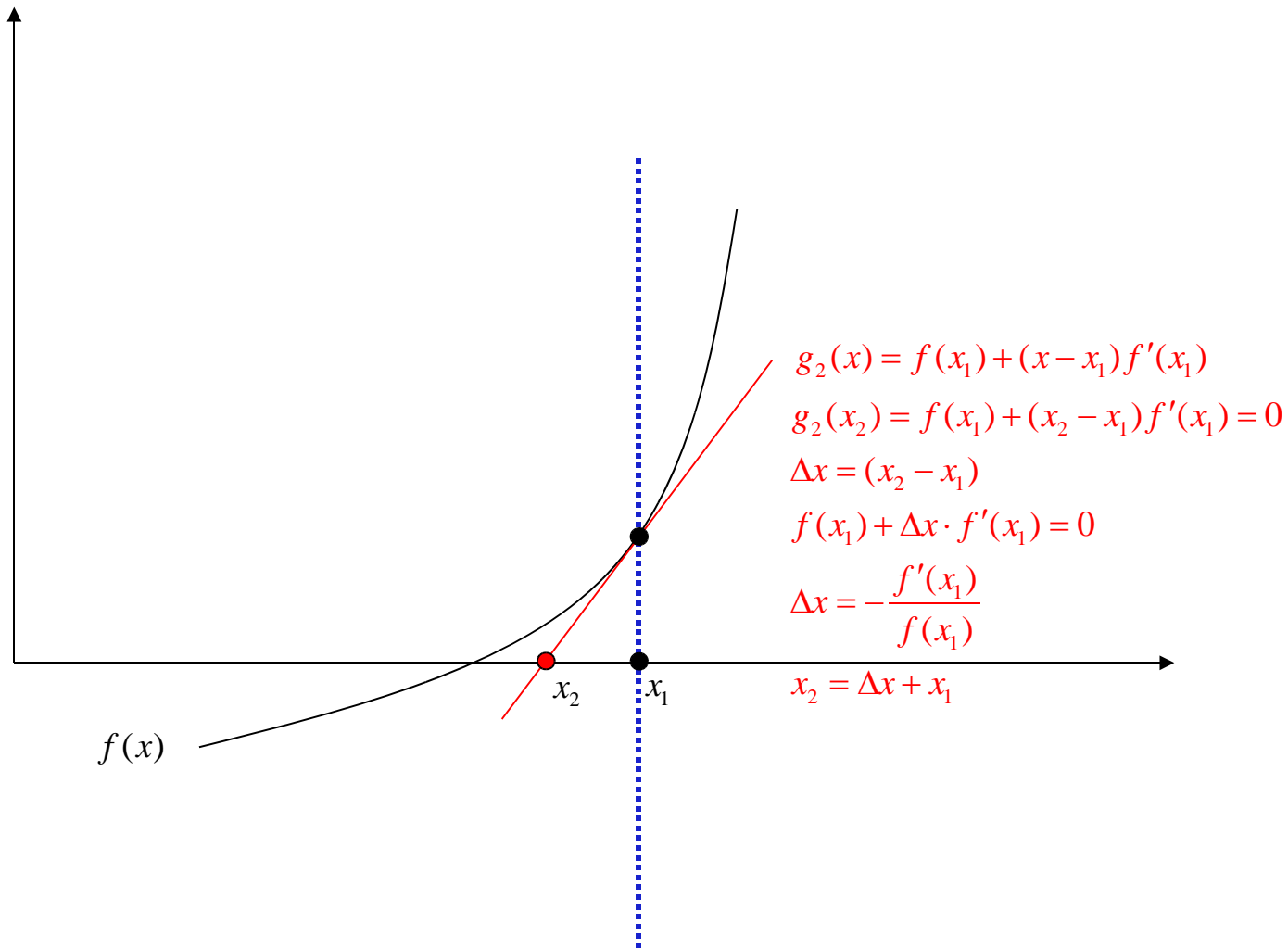
- Review: Loadflow calculation
- Solution of Nonlinear Algebraic Equations
  - Newton Raphson Method
  - One dimensional case
  - Multi dimensional case
- Sparse Matrices
  - Characteristics
  - Storage Methods
- Solution of large scale linear Algebraic Equations
  - Gauss Method
  - LDR Factorization
  - Methods for increasing efficiency

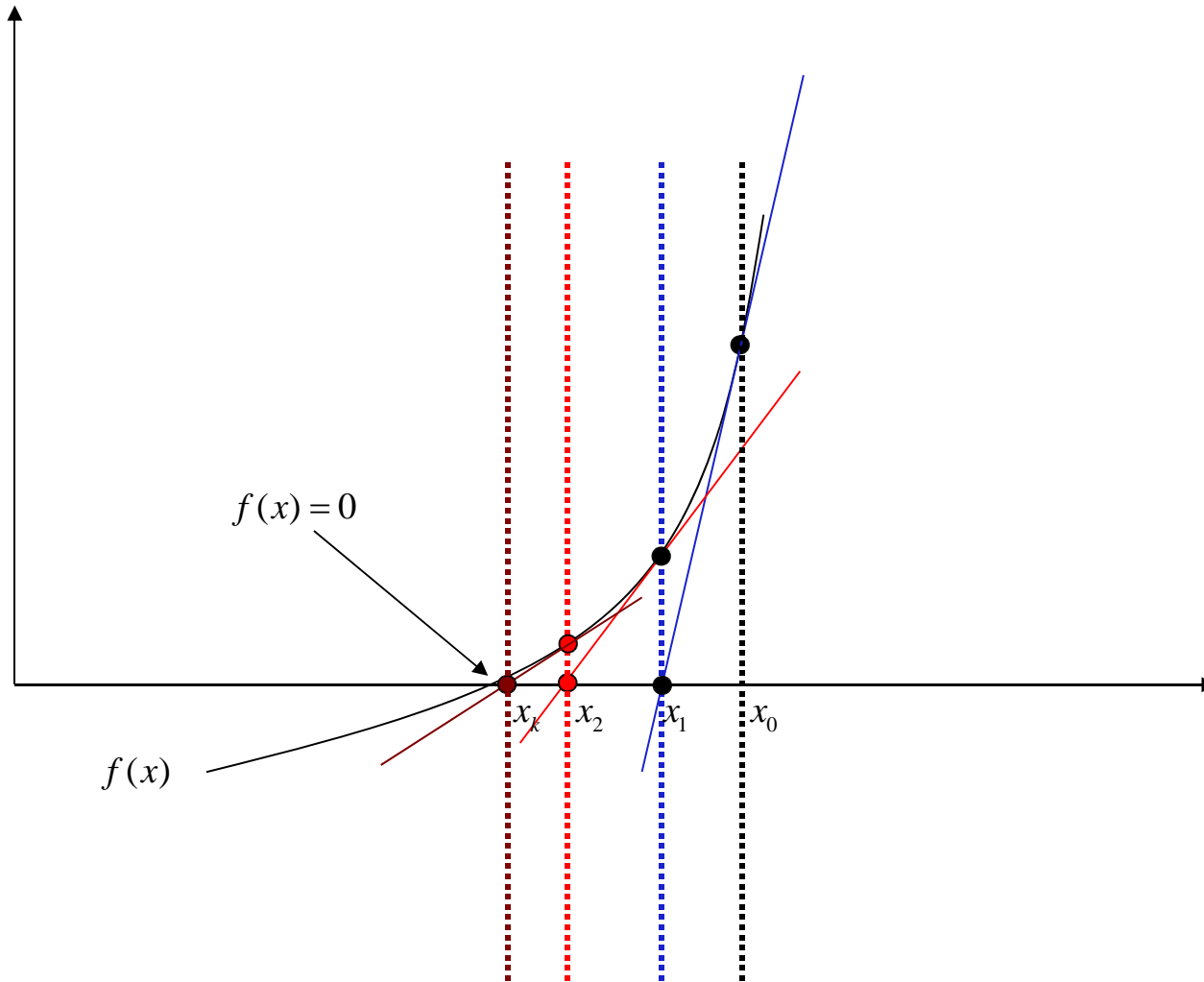




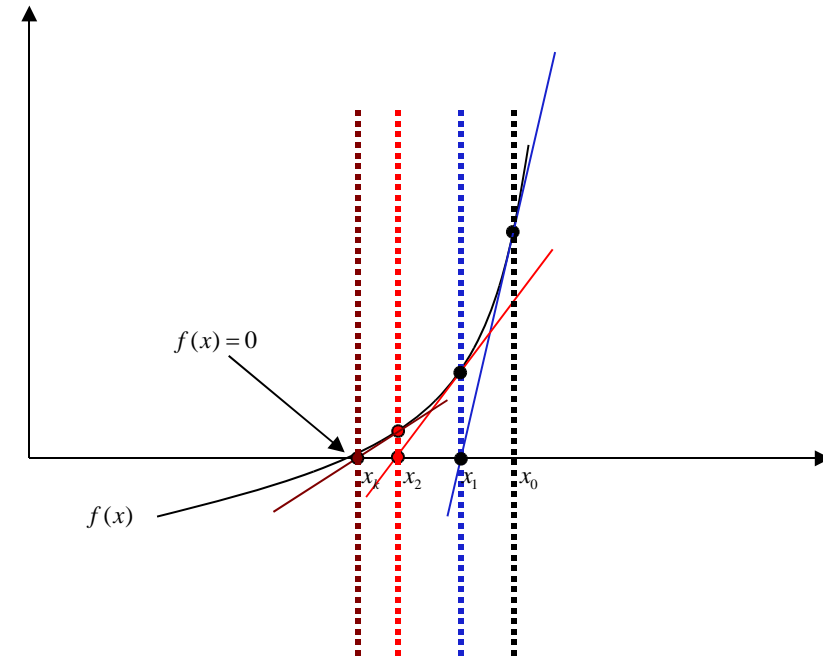








- Set  $k = 0$  , choose  $x_0$
- while  $|f(x_k)| > \varepsilon$ 
  - Calculate  $\Delta x_k$  with
$$f'(x_k) \cdot \Delta x_k = -f(x_k)$$
linear algebraic equation system
$$A \cdot x = b$$
  - calculate  $x_{k+1} = x_k + \Delta x_k$
  - Set  $k = k + 1$
- set  $x_{\text{lösung}} = x_k$



$$\underbrace{\begin{bmatrix} \frac{\partial P_1}{\partial \theta_1} & \dots & \frac{\partial P_1}{\partial \theta_N} & \frac{\partial P_1}{\partial V_1} & \dots & \frac{\partial P_1}{\partial V_N} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \frac{\partial P_N}{\partial \theta_1} & \dots & \frac{\partial P_N}{\partial \theta_N} & \frac{\partial P_N}{\partial V_1} & \dots & \frac{\partial P_N}{\partial V_N} \\ \frac{\partial Q_1}{\partial \theta_1} & \dots & \frac{\partial Q_1}{\partial \theta_N} & \frac{\partial Q_1}{\partial V_1} & \dots & \frac{\partial Q_1}{\partial V_N} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \frac{\partial Q_N}{\partial \theta_1} & \dots & \frac{\partial Q_N}{\partial \theta_N} & \frac{\partial Q_N}{\partial V_1} & \dots & \frac{\partial Q_N}{\partial V_N} \\ \frac{\partial Q_N}{\partial \theta_1} & \dots & \frac{\partial Q_N}{\partial \theta_N} & \frac{\partial Q_N}{\partial V_1} & \dots & \frac{\partial Q_N}{\partial V_N} \end{bmatrix}}_{J(x^k)} \cdot \underbrace{\begin{bmatrix} \Delta \theta_1 \\ \dots \\ \dots \\ \Delta \theta_N \\ \Delta V_1 \\ \dots \\ \dots \\ \Delta V_N \end{bmatrix}}_{\Delta x^k} = \begin{bmatrix} P_1 - P(U_1^k, \dots, U_N^k, \theta_1^k, \dots, \theta_N^k) \\ \dots \\ \dots \\ P_N - P(U_1^k, \dots, U_N^k, \theta_1^k, \dots, \theta_N^k) \\ Q_1 - Q(U_1^k, \dots, U_N^k, \theta_1^k, \dots, \theta_N^k) \\ \dots \\ \dots \\ Q_N - Q(U_1^k, \dots, U_N^k, \theta_1^k, \dots, \theta_N^k) \end{bmatrix}$$

$$A \cdot x = b$$

- Efficient solution of  $A \cdot x = b$  as it is solved at each iteration.
- Characteristics of  $A$  ?

- Review: Loadflow calculation
- Solution of Nonlinear Algebraic Equations
  - Newton Raphson Method
  - One dimensional case
  - Multi dimensional case
- **Sparse Matrices**
  - Characteristics
  - Storage Methods
- Solution of large scale linear Algebraic Equations
  - Gauss Method
  - LDR Factorization
  - Methods for increasing efficiency

- Matrices with large dimensions (e.g. 1000 x 1000)
- Not dense matrices, they are sparse with only a few non-zero elements (NZ)
- Degree of sparsity 0.3 – 0.4 % ( $\mu$ )

$$(\mu) = \frac{\text{Number of NZ}}{\text{Number of rows} \times \text{Number of columns}}$$

- Let us assume we have an A-Matrix with dimension (1000 x 1000) and degree of sparsity 0.4%
- With normal storage methods, we would need to store 1 000 000 elements, although we have only 4000 non-zero elements in our A matrix.
- Storage of only nonzero elements.

$$A = \begin{bmatrix} 10 & 0 & 0 & 0 & -2 & 0 \\ 3 & 9 & 0 & 0 & 0 & 3 \\ 0 & 0 & 8 & 0 & 2 & 0 \\ 3 & 0 & 0 & 7 & 5 & 0 \\ 0 & 1 & 0 & 0 & 9 & 13 \\ 0 & 0 & 6 & 0 & 0 & -1 \end{bmatrix}$$

$$\begin{array}{l}
 R1 \\
 R2 \\
 R3 \\
 R4 \\
 R5 \\
 R6
 \end{array}
 \begin{bmatrix}
 C1 & C2 & C3 & C4 & C5 & C6 \\
 10 & 0 & 0 & 0 & -2 & 0 \\
 3 & 9 & 0 & 0 & 0 & 3 \\
 0 & 0 & 8 & 0 & 2 & 0 \\
 3 & 0 & 0 & 7 & 5 & 0 \\
 0 & 1 & 0 & 0 & 9 & 13 \\
 0 & 0 & 6 & 0 & 0 & -1
 \end{bmatrix}$$

I	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
<b>V</b>	<b>10</b>	<b>-2</b>	<b>3</b>	<b>9</b>	<b>3</b>	<b>8</b>	<b>2</b>	<b>3</b>	<b>7</b>	<b>5</b>	<b>1</b>	<b>9</b>	<b>13</b>	<b>6</b>	<b>-1</b>
<b>C</b>	<b>1</b>	<b>5</b>	<b>1</b>	<b>2</b>	<b>6</b>	<b>3</b>	<b>5</b>	<b>1</b>	<b>4</b>	<b>5</b>	<b>2</b>	<b>5</b>	<b>6</b>	<b>3</b>	<b>6</b>
<b>R</b>	<b>1</b>	<b>1</b>	<b>2</b>	<b>2</b>	<b>2</b>	<b>3</b>	<b>3</b>	<b>4</b>	<b>4</b>	<b>4</b>	<b>5</b>	<b>5</b>	<b>5</b>	<b>6</b>	<b>6</b>

- Total memory neede ( 3 x NNZ) = 3 x 15 = 45
- Disadvantage, finding a specific entry  $A(3,3)=?$





- Review: Loadflow calculation
- Solution of Nonlinear Algebraic Equations
  - Newton Raphson Method
  - One dimensional case
  - Multi dimensional case
- Sparse Matrices
  - Characteristics
  - Storage Methods
- Solution of large scale linear Algebraic Equations
  - Gauss Method
  - LDR Factorization
  - Methods for increasing efficiency

## linear algebraic equation systems

$$A \cdot x = b$$

### ■ Direct methods

- calculation of the inverse of

$$A \cdot x = b$$

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

$2(n+1)!$  multiplications needed

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

- Gauss method

$$A \cdot x = b \quad \Longrightarrow \quad R \cdot x = b' \quad \Longrightarrow \quad \text{back substitution}$$

### ■ Indirect (iterativ) Methods

- Relaxations Method
- Conjugate Gradient Method

$$\begin{cases} x_1 + 3x_2 + 1x_3 = 3 \\ 6x_2 + 2x_3 = 1 \\ 3x_1 + 4x_2 + 1x_3 = 4 \end{cases}$$

$$\begin{bmatrix} 1 & 3 & 1 \\ 0 & 6 & 2 \\ 3 & 4 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \\ 4 \end{bmatrix}$$

$$\left[ \begin{array}{ccc|c} 1 & 3 & 1 & 3 \\ 0 & 6 & 2 & 1 \\ 3 & 4 & 1 & 4 \end{array} \right] \begin{array}{l} -3 \\ \\ 1 \end{array}$$

$$\left[ \begin{array}{ccc|c} 1 & 3 & 1 & 3 \\ 0 & 6 & 2 & 1 \\ 0 & -5 & -2 & -5 \end{array} \right] \begin{array}{l} \\ 5/6 \\ 1 \end{array}$$

$$\left[ \begin{array}{ccc|c} 1 & 3 & 1 & 3 \\ 0 & 6 & 2 & 1 \\ 0 & 0 & -1/3 & -25/6 \end{array} \right] \cdot$$

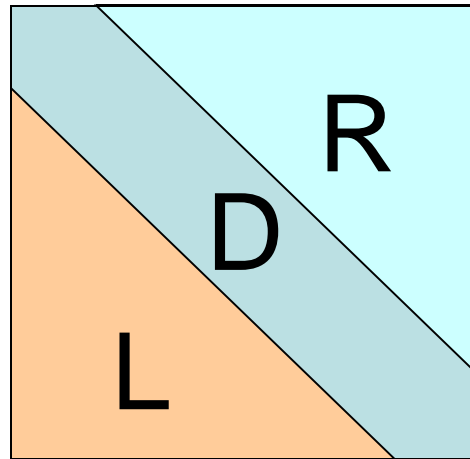
$$\begin{bmatrix} 1 & 3 & 1 \\ 0 & 6 & 2 \\ 0 & 0 & -1/3 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \\ -25/6 \end{bmatrix}$$

$$A \cdot x = b$$



Triangular form

$$R \cdot x = b'$$

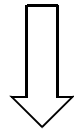


$$A = L \cdot D \cdot R$$

$$A \cdot x = b \quad \Longrightarrow \quad L \cdot \underbrace{(D \cdot R \cdot x)}_{\omega} = b \quad \Longrightarrow \quad D \cdot \underbrace{R \cdot x}_v = \omega \quad \Longrightarrow \quad R \cdot x = v$$

$$A \cdot x = b \quad \Longrightarrow \quad L \cdot \omega = b \quad \Longrightarrow \quad D \cdot v = \omega \quad \Longrightarrow \quad R \cdot x = v$$

$$\begin{aligned} A &= \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = L \cdot D \cdot R \\ &= \begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix} \cdot \begin{pmatrix} d_{11} & 0 & 0 \\ 0 & d_{22} & 0 \\ 0 & 0 & d_{33} \end{pmatrix} \cdot \begin{pmatrix} 1 & r_{12} & r_{13} \\ 0 & 1 & r_{23} \\ 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} d_{11} & d_{11}r_{12} & d_{11}r_{13} \\ l_{21}d_{11} & l_{21}d_{11}r_{12} + d_{22} & l_{21}d_{11}r_{13} + d_{22}r_{23} \\ l_{31}d_{11} & l_{31}d_{11}r_{12} + l_{32}d_{22} & l_{31}d_{11}r_{13} + l_{32}d_{22}r_{23} + d_{33} \end{pmatrix} \end{aligned}$$



$$\begin{pmatrix} d_{11} & r_{12} & r_{13} \\ l_{21} & d_{22} & r_{23} \\ l_{31} & l_{32} & d_{33} \end{pmatrix}$$

Run Matlab Example

1. Set  $k=1$

2. choose  $a_{kk}$

3. 
$$a_{ij}^{(neu)} = a_{ij}^{(alt)} - \frac{a_{ik}^{(alt)} a_{kj}^{(alt)}}{a_{kk}^{(alt)}} \quad \forall i, j \quad i > k, j > k$$

4. 
$$a_{ik}^{(neu)} = \frac{a_{ik}^{(alt)}}{a_{kk}^{(alt)}} \quad \forall i > k$$

5. 
$$a_{kj}^{(neu)} = \frac{a_{kj}^{(alt)}}{a_{kk}^{(alt)}} \quad \forall j > k$$

6. set  $k=k+1$  und go to step 2

$$\begin{pmatrix} d_{11} & d_{11}r_{12} & d_{11}r_{13} \\ l_{21}d_{11} & l_{21}d_{11}r_{12}+d_{22} & l_{21}d_{11}r_{13}+d_{22}r_{23} \\ l_{31}d_{11} & l_{31}d_{11}r_{12}+l_{32}d_{22} & l_{31}d_{11}r_{13}+l_{32}d_{22}r_{23}+d_{33} \end{pmatrix}$$

„Fill ins“ (NNE)

- LDR – factorisation  $\alpha$
- Back substitution  $\beta$

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \Rightarrow \begin{pmatrix} d_{11} & r_{12} & r_{13} \\ l_{21} & d_{22} & r_{23} \\ l_{31} & l_{32} & d_{33} \end{pmatrix} = Q$$

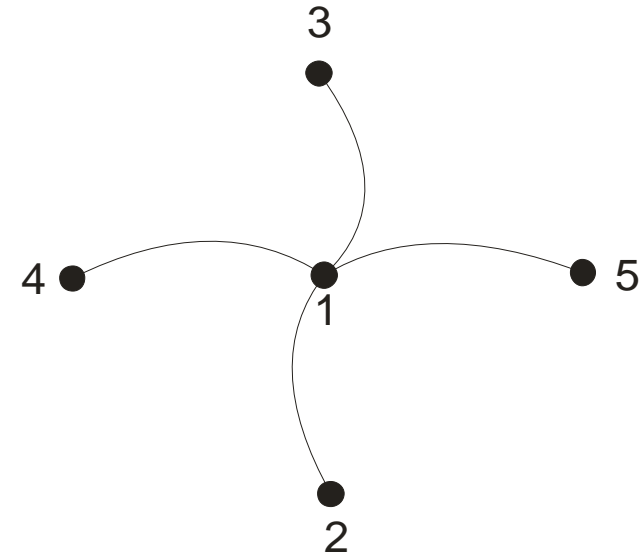
$$\alpha = \sum_{i=1}^N ((NNE \text{ in Spalte } i \text{ unter } q_{ii} + 1) \times (NNE \text{ in Zeile } i \text{ rechts von } q_{ii}))$$

$$\beta = NNE \text{ von } Q$$

- Number of Non-zero elements play a significant role in the total number of needed operations

Does the node ordering play a role in the number of new non-zero elements

	1	2	3	4	5	Degree
1						4
2						1
3						1
4						1
5						1

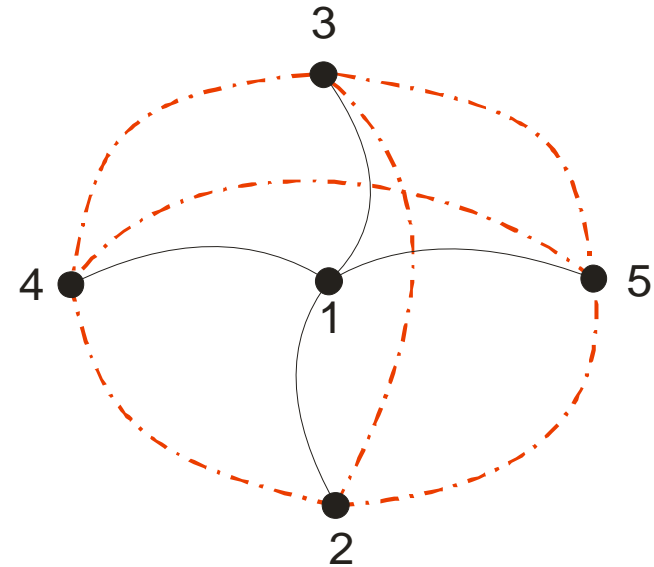


$$a_{ij}^{(neu)} = a_{ij}^{(alt)} - \frac{a_{ik}^{(alt)} a_{kj}^{(alt)}}{a_{kk}^{(alt)}} \quad \forall i, j \quad i > k, j > k$$

Degree = Number of connected nodes

Does the node ordering play a role in the number of new non-zero elements

	1	2	3	4	5	Degree
1	1	1	1	1	1	4
2	1	1	1	1	1	1
3	1	1	1	1	1	1
4	1	1	1	1	1	1
5	1	1	1	1	1	1



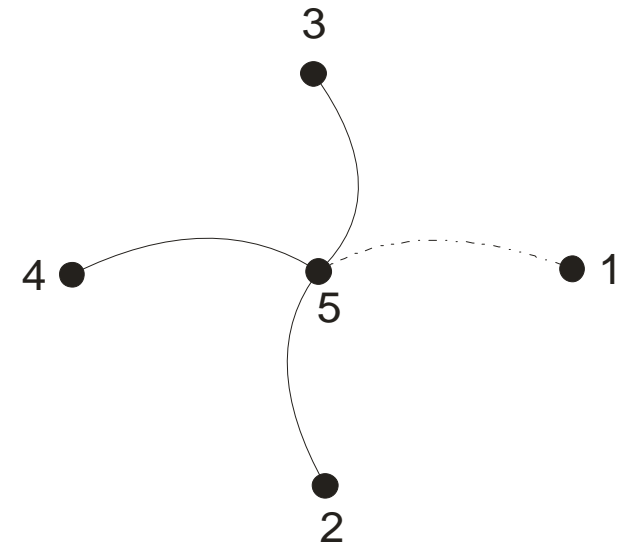
$$a_{ij}^{(neu)} = a_{ij}^{(alt)} - \frac{a_{ik}^{(alt)} a_{kj}^{(alt)}}{a_{kk}^{(alt)}} \quad \forall i, j \quad i > k, j > k$$

$$a_{23}^{(neu)} = a_{23}^{(alt)} - \frac{a_{21}^{(alt)} a_{13}^{(alt)}}{a_{11}^{(alt)}}$$

Degree = Number of connected nodes

Does the node ordering play a role in the number of new non-zero elements

	1	2	3	4	5	Degree
1						1
2						1
3						1
4						1
5						4



$$a_{ij}^{(neu)} = a_{ij}^{(alt)} - \frac{a_{ik}^{(alt)} a_{kj}^{(alt)}}{a_{kk}^{(alt)}} \quad \forall i, j \quad i > k, j > k$$

$$a_{23}^{(neu)} = a_{23}^{(alt)} - \frac{a_{21}^{(alt)} a_{13}^{(alt)}}{a_{11}^{(alt)}}$$

Degree = Number of connected nodes

	1	2	3	4	5	6	7	8	9	10
1	■	■		■				■		
2	■	■					■			■
3			■	■	■		■	■		■
4	■		■	■	■					
5			■	■	■	■	■			■
6					■	■	■			
7		■	■		■	■	■	■	■	
8	■		■				■	■		
9							■		■	
10		■	■		■					■

Degree

3
3
5
4
5
2
6
3
1
3

## Tinney Schema 0

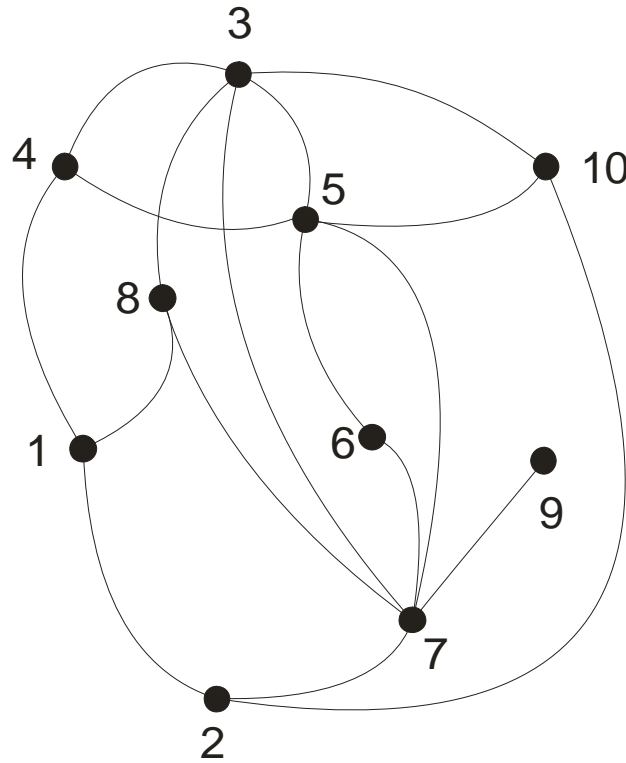
1. Calculate the degree of each node
2. Choose the node with the minimum degree.
3. If draw, choose the one with the lower node ID
4. Go to step 2

9	6	1	2	8	10	4	3	5	7
---	---	---	---	---	----	---	---	---	---

- + Easy to implement
- New constellation not considered

Matlab Example

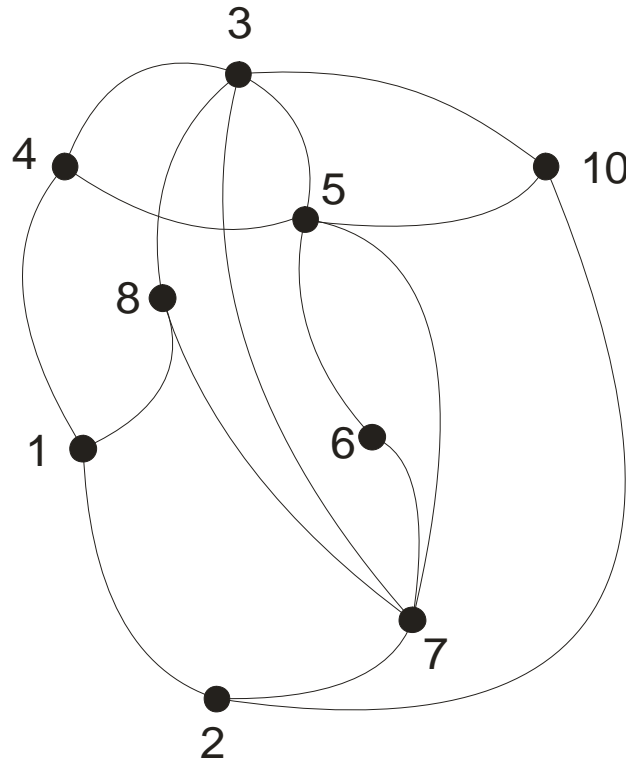
# Step - 1



ID	Degree
9	1
6	2
1	3
2	3
8	3
10	3
4	4
3	5
5	5
7	6

9									
---	--	--	--	--	--	--	--	--	--

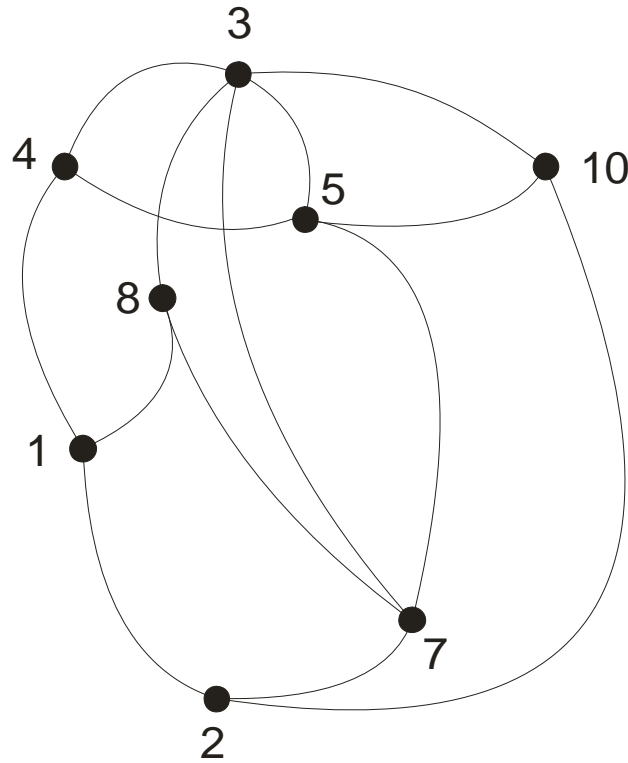
# Step - 2



ID	Degree
6	2
1	3
2	3
8	3
10	3
4	4
3	5
5	5
7	5

9	6								
---	---	--	--	--	--	--	--	--	--

# Step - 3



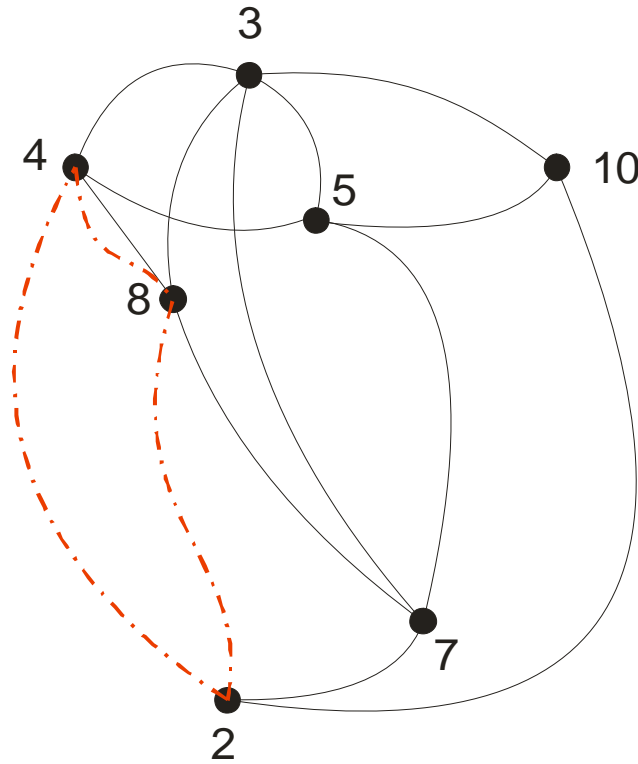
ID Degree

1
2
8
10
4
5
7
3

3
3
3
3
4
4
4
4
5

9	6	1							
---	---	---	--	--	--	--	--	--	--

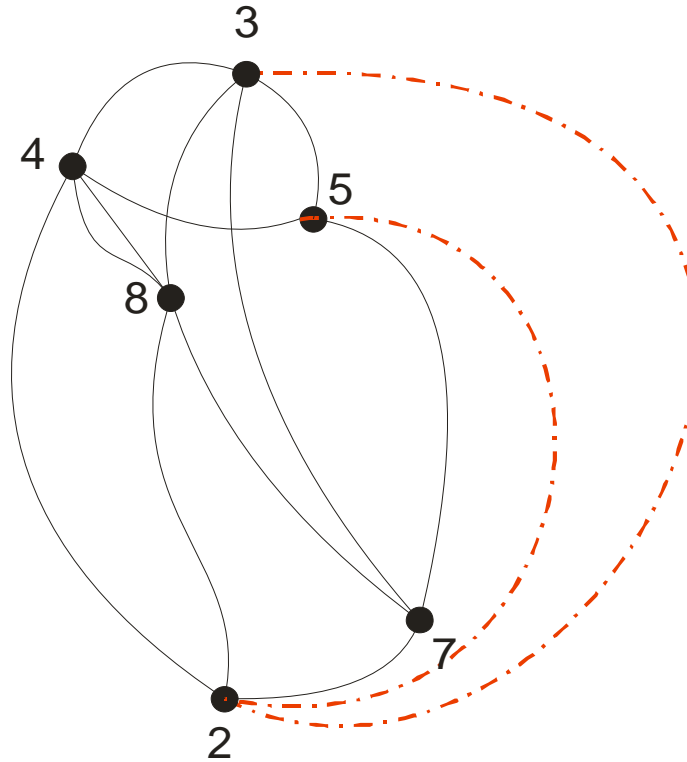
# Step - 4



ID	Degree
10	3
2	4
4	4
5	4
7	4
8	4
3	5

9	6	1	10						
---	---	---	----	--	--	--	--	--	--

# Step - 5



ID Degree

4
5
7
8
2
3

4
4
4
4
5
5

9	6	1	10	4					
---	---	---	----	---	--	--	--	--	--

	1	2	3	4	5	6	7	8	9	10
1	█	█		█				█		
2	█	█					█			█
3			█	█	█		█	█		█
4	█		█	█	█					
5			█	█	█	█	█			█
6					█	█	█			
7		█	█		█	█	█	█	█	
8	█		█				█	█		
9							█		█	
10		█	█		█					█

Degree

3
3
5
4
5
2
6
3
1
3

## Tinney Schema 1

1. Calculate the degree of each node
2. Choose the node with the minimum degree,
3. If draw, choose the one with the lower node ID
4. Remove it and **Calculate the degree of each node.**
5. Go to Step 1

9	6	1	10	4	2	3	5	7	8
---	---	---	----	---	---	---	---	---	---

- + Also easy to implement
- Considers new constellation, but not the number of „Fill ins“

Matlab Example

	1	2	3	4	5	6	7	8	9	10
1	■	■		■				■		
2	■	■					■			■
3			■	■	■		■	■		■
4	■		■	■	■					
5			■	■	■	■	■			■
6					■	■	■			
7		■	■		■	■	■	■	■	
8	■		■				■	■		
9							■		■	
10		■	■		■					■

Degree

3
3
5
4
5
2
6
3
1
3

## Tinney Schema 2

1. Calculate for each node the number of „Fill ins“, if the node is removed
2. Choose the node with the minimum number of „Fill ins“
3. If draw, take the one with the lowest degree
4. If again draw, take the one with the lowest node ID
5. Place it into the new node ordering and update matrix with new „Fill ins“
6. Go to Step 1

9	6	4	8	2	1	3	5	7	10
---	---	---	---	---	---	---	---	---	----

Matlab Example

	NNZ	NNZ(LDR)	„Fill-Ins“	$\alpha$	$\beta$	$\alpha + \beta$
Org	44	68	24	134	68	202
Tinney 0	44	60	16	110	60	170
Tinney 1	44	56	12	92	56	148
Tinney 2	44	54	10	84	54	138

- Loadflow Calculation – Power balance

$$P_k = P(U_1, \dots, U_N, \theta_1, \dots, \theta_N)$$

$$Q_k = Q(U_1, \dots, U_N, \theta_1, \dots, \theta_N)$$

- Solution of nonlinear algebraic equation system
  - Newton Raphson Method
- Solution of large scale sparse linear equation system.
  - Sparse Matrices
  - Solution methods for  $Ax=b$
- LDR Factorisation and Back substitution
  - Node ordering methods

- Mariesa Crow  
Computational Methods for Electric Power Systems