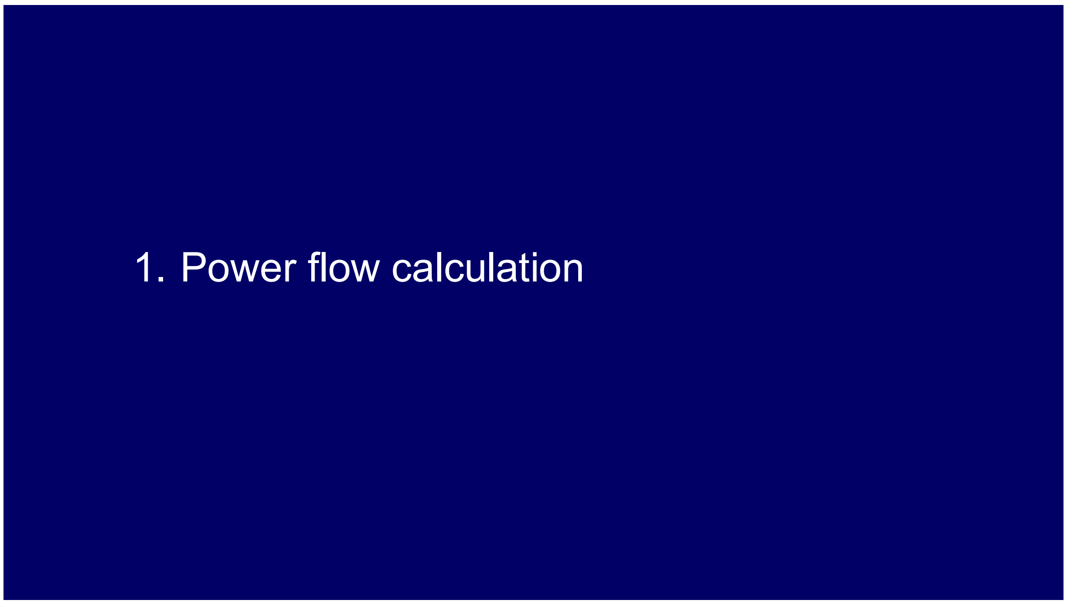
Chapter 1

Power System Analysis / Power Flow

1.1 Lecture

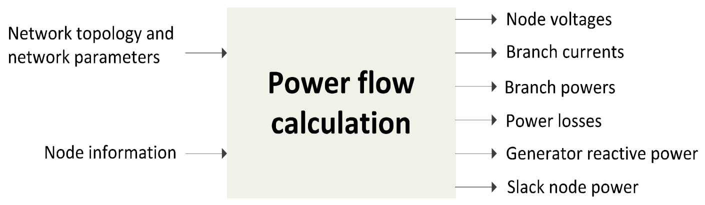


1. Power flow calculation

1.1 WHAT IS POWER FLOW CALCULATION?

Power flow calculation is an iterative procedure for determination of the power flows (active and reactive) throughout the entire network, using information available about

- ❖Nodal power injections (P,Q)
- ❖Voltages (magnitude U, angle δ)
- ❖System topology (interconnection of components), including parameters (e.g. reactances)

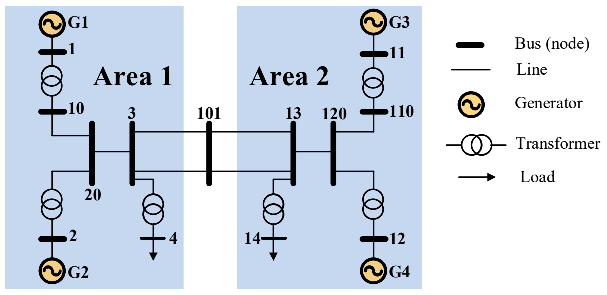


Note: The terms power flow and load flow both stand for the same analysis technique, however it is recommended to use the name power flow. P. Kundur: **"Load does not flow, but power flows."**

1.2 WHY DO WE NEED POWER FLOW CALCULATION?

Applications:

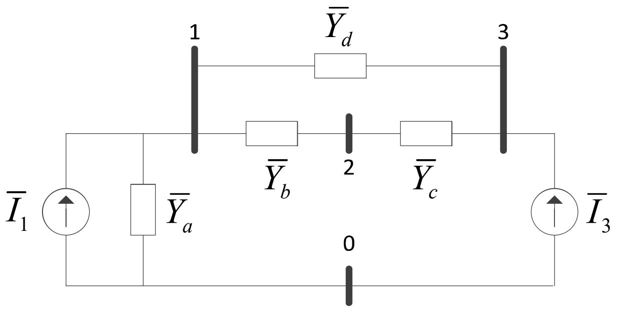
1. State estimation (power system analysis)
2. Feasibility of power dispatch
3. Contingency management, etc.



1.3 THE YBUS MATRIX

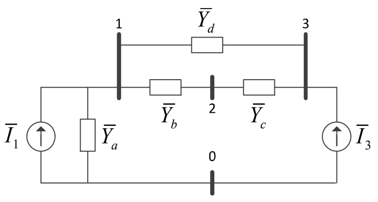
Example 1

In the system shown below, determine the relationship between currents and voltages by applying Kirchhoff current's law. Node voltages measured w.r.t. node 0 (reference)



1.3 THE YBUS MATRIX

Example 1



Currents at node 1:

$$\overline{I}_1 = \overline{U}_1\overline{Y}_a + \left(\overline{U}_1 - \overline{U}_2\right)\overline{Y}_b + \left(\overline{U}_1 - \overline{U}_3\right)\overline{Y}_d$$

Currents at node 2:

$$0 = \left(\overline{U}_2 - \overline{U}_1\right)\overline{Y}_b + \left(\overline{U}_2 - \overline{U}_3\right)\overline{Y}_c$$

Currents at node 3:

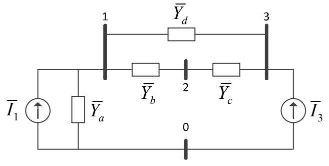
$$\overline{I}_3 = \left(\overline{U}_3 - \overline{U}_1\right)\overline{Y}_d + \left(\overline{U}_3 - \overline{U}_2\right)\overline{Y}_c$$

In matrix form:

$$\begin{bmatrix} \overline{I}_1 \\ 0 \\ \overline{I}_3 \end{bmatrix} = \begin{bmatrix} \left(\overline{Y}_a + \overline{Y}_b + \overline{Y}_c\right) & -\overline{Y}_b & -\overline{Y}_d \\ -\overline{Y}_b & \left(\overline{Y}_b + \overline{Y}_c\right) & -\overline{Y}_c \\ -\overline{Y}_d & -\overline{Y}_c & \left(\overline{Y}_c + \overline{Y}_d\right) \end{bmatrix} \begin{bmatrix} \overline{U}_1 \\ \overline{U}_2 \\ \overline{U}_3 \end{bmatrix}$$

1.3 THE YBUS MATRIX

Example 1



$$\begin{bmatrix} \overline{I}_1 \\ 0 \\ \overline{I}_3 \end{bmatrix} = \begin{bmatrix} \left(\overline{Y}_a + \overline{Y}_b + \overline{Y}_d\right) & -\overline{Y}_b & -\overline{Y}_d \\ -\overline{Y}_b & \left(\overline{Y}_b + \overline{Y}_c\right) & -\overline{Y}_c \\ -\overline{Y}_d & -\overline{Y}_c & \left(\overline{Y}_c + \overline{Y}_d\right) \end{bmatrix} \begin{bmatrix} \overline{U}_1 \\ \overline{U}_2 \\ \overline{U}_3 \end{bmatrix}$$

Generalizing for any system, the network admittance matrix, $\mathbf{Y}_{\mathbf{bus}}$, relates the nodal voltages vector, **U,** to the nodal currents vector, **I**, such that:

$$\mathbf{I}_{[N x 1]} = \mathbf{Y}_{bus_{[N x N]}} \cdot \mathbf{U}_{[N x 1]}$$ N: total number of network buses (nodes)

There are two ways to build $\mathbf{Y}_{\mathbf{bus}}$:

a) Manual Inspection.
b) Through the incidence Matrix.

1.3 THE YBUS MATRIX

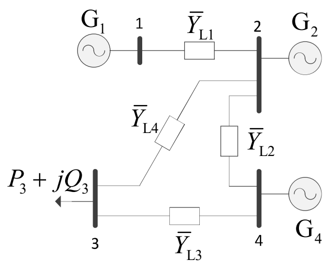
Manual Inspection
(Easy to apply in small networks)

Rule:

Diagonal Elements: $\bar{Y}_{ij}$ → Sum of all the admittances connected to node *i*.
Off-diagonal Elements: $\bar{Y}_{ij}$ → Negative of the admittance between node *i* and *j*.

Example 2

$$\begin{bmatrix} \bar{I}_1 \\ \bar{I}_2 \\ \bar{I}_3 \\ \bar{I}_4 \end{bmatrix} = \underbrace{\begin{bmatrix} \bar{Y}_{L1} & -\bar{Y}_{L1} & 0 & 0 \\ -\bar{Y}_{L1} & \left(\bar{Y}_{L1}+\bar{Y}_{L2}+\bar{Y}_{L4}\right) & -\bar{Y}_{L4} & -\bar{Y}_{L2} \\ 0 & -\bar{Y}_{L4} & \left(\bar{Y}_{L3}+\bar{Y}_{L4}\right) & -\bar{Y}_{L3} \\ 0 & -\bar{Y}_{L2} & -\bar{Y}_{L3} & \left(\bar{Y}_{L2}+\bar{Y}_{L3}\right) \end{bmatrix}}_{\mathbf{Y}_{bus}} \cdot \begin{bmatrix} \bar{U}_1 \\ \bar{U}_2 \\ \bar{U}_3 \\ \bar{U}_4 \end{bmatrix}$$



1.3 THE YBUS MATRIX

Incidence Matrix: A_M
(More convenient for automated generation of $\mathbf{Y}_{bus}$ in large networks)

$$\mathbf{Y}_{bus} = \left(\mathbf{A}_M\right)^T \cdot \mathbf{Y}_p \cdot \mathbf{A}_M$$

$\mathbf{Y}_p$: primitive admittance matrix

$$\mathbf{Y}_p = \begin{bmatrix} \bar{Y}_{L1} & 0 & \cdots & 0 \\ 0 & \bar{Y}_{L1} & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \bar{Y}_{LN_B} \end{bmatrix}$$

N_B: number of branches

Order of $\mathbf{Y}_p$: $N_B \times N_B$

1.3 THE YBUS MATRIX

Incidence Matrix: A_M
(More convenient for automated generation of $\mathbf{Y}_{bus}$ in large networks)

$$\mathbf{Y}_{bus} = (\mathbf{A}_M)^T \cdot \mathbf{Y}_p \cdot \mathbf{A}_M \Rightarrow$$

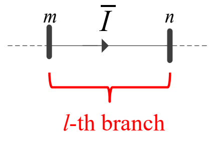
Order of $\mathbf{A}_M$: $N_B \times N$

N_B: number of branches

N: total number of network buses (nodes)

Elements of $\mathbf{A}_M$

Convention for current direction is important, e.g.



$$a_{mn} = \begin{cases} 0 & \text{If } l\text{-th branch is not connected to node } m \\ 1 & \text{If current in } l\text{-th branch is directed away from node } m \\ -1 & \text{If current in } l\text{-th branch is directed towards node } m \end{cases}$$

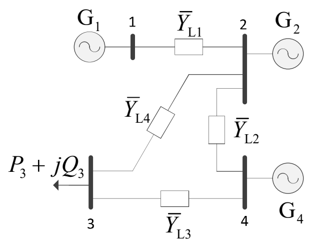
1.3 THE YBUS MATRIX

Incidence Matrix: A_M
(More convenient for automated generation of $\mathbf{Y}_{bus}$ in large networks)

Example 3

$$\mathbf{Y}_{bus} = (\mathbf{A}_M)^T \cdot \mathbf{Y}_p \cdot \mathbf{A}_M$$

$$\underbrace{\mathbf{Y}_{bus}}_{N \times N} = \underbrace{(\mathbf{A}_M)^T}_{N \times N_B} \cdot \underbrace{\begin{pmatrix} \bar{Y}_{L1} & 0 & 0 & 0 \\ 0 & \bar{Y}_{L2} & 0 & 0 \\ 0 & 0 & \bar{Y}_{L3} & 0 \\ 0 & 0 & 0 & \bar{Y}_{L4} \end{pmatrix}}_{\mathbf{Y}_p[N_B \times N_B]} \cdot \underbrace{\begin{pmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \\ 0 & 1 & -1 & 0 \end{pmatrix}}_{\mathbf{A}_M[N_B \times N]}$$



N_B: number of branches

N: total number of network buses (nodes)

1.4 THE POWER FLOW EQUATIONS

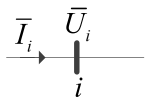
Recalling

$$\mathbf{I}_{[N x 1]} = \mathbf{Y}_{bus_{[N x N]}} \cdot \mathbf{U}_{[N x 1]}$$ N: total number of network buses (nodes)

The elements (e.g. between nodes *i* and *j*) of $\mathbf{Y}_{\mathbf{bus}}$:

$$\overline{Y}_{ij} = Y_{ij}\angle\theta_{ij} = Y_{ij}\left(\cos\left(\theta_{ij}\right) + j\sin\left(\theta_{ij}\right)\right) = G_{ij} + jB_{ij}$$

The voltage of the network nodes (e.g. at node *i*) :



$$\overline{U}_i = U_i\angle\delta_i = U_i\left(\cos\left(\delta_i\right) + j\sin\left(\delta_i\right)\right)$$

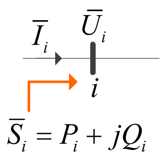
Therefore, the current injected at node *i* can be obtained as:

$$\overline{I}_i = \overline{Y}_{i1}\overline{U}_1 + \overline{Y}_{i2}\overline{U}_2 + \ldots + \overline{Y}_{iN}\overline{U}_N = \sum_{n=1}^{N}\overline{Y}_{in}\overline{U}_n$$

1.4 THE POWER FLOW FLOW EQUATIONS

Considering
$$\begin{cases} \overline{Y}_{ij} = Y_{ij}\angle\theta_{ij} = Y_{ij}\left(\cos\left(\theta_{ij}\right) + j\sin\left(\theta_{ij}\right)\right) = G_{ij} + jB_{ij} \\ \overline{U}_i = U_i\angle\delta_i = U_i\left(\cos\left(\delta_i\right) + j\sin\left(\delta_i\right)\right) \\ \overline{I}_i = \overline{Y}_{i1}\overline{U}_1 + \overline{Y}_{i2}\overline{U}_2 + \ldots + \overline{Y}_{iN}\overline{U}_N = \sum_{n=1}^{N}\overline{Y}_{in}\overline{U}_n \end{cases}$$

The active power injected into node *i*, is given by



$$\overline{S}_i = P_i + jQ_i = \overline{U}_i \cdot \overline{I}_i^*$$

$$\boxed{P_i = \sum_{n=1}^{N} U_i U_n Y_{in}\cos\left(\delta_i - \delta_n - \theta_{in}\right) \qquad Q_i = \sum_{n=1}^{N} U_i U_n Y_{ij}\sin\left(\delta_i - \delta_n - \theta_{in}\right)}$$

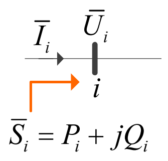
N: total number of network buses (nodes)

1.5 NETWORK NODE TYPES

$$P_i = \sum_{n=1}^{N} U_i U_n Y_{in} \cos(\delta_i - \delta_n - \theta_{in}) \qquad Q_i = \sum_{n=1}^{N} U_i U_n Y_{ij} \sin(\delta_i - \delta_n - \theta_{in})$$

For every node i, four quantities are defined:

1. Active Power: P_i
2. Reactive Power: Q_i
3. Voltage phasor amplitude: U_i
4. Voltage phasor angle: δ_i



Since

$$P_i = f_p(U_1, K, U_i K, U_N, \delta_1, K, \delta_i, K, \delta_N) \quad Q_i = f_q(U_1, K, U_i K, U_N, \delta_1, K, \delta_i, K, \delta_N)$$

$\Longrightarrow$ U_i and δ_i are state variables: $\mathbf{x} = [\boldsymbol{\delta}\ \mathbf{U}]^{\mathrm{T}}$

Thus, for every node there are, in principle, two equations:
$\rightarrow$ One for the active power, P_i, and one for the reactive power Q_i.

1.5 NETWORK NODE TYPES

However, without a reference node, there would be infinite solutions possible.

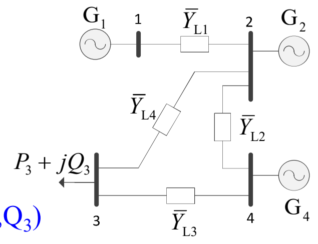
Reference = slack node → A node in which the voltage phasor amplitude and angle are known (usually the node with the largest generator.)

Node Type	Number of nodes	Known Variables	Unkown Variables	Number of equations per bus type
Slack bus (reference) $(i = 1)$	1	U_1 and $\delta_1 = 0$	P_1 and Q_1	0
PV bus (generation) $(i = 2 \ldots N_g + 1)$	N_g	P_i and U_i	δ_i and Q_i	N_g (associated to P_i)
PQ bus (load) $(i = N_g + 2 \ldots N)$	$N - N_g - 1$	P_i and Q_i	δ_i and U_i	$2(N - N_g - 1)$ (associated to P_i and Q_i)

N: total number of network nodes

$$\Rightarrow \mathbf{x} = [\boldsymbol{\delta} \quad \mathbf{U}]^T = \left[\delta_2, K, \delta_N, U_{N_g+2} K, U_N\right]^T$$

1.5 NETWORK NODE TYPES



Example 4

In the 4 node network:

1. Slack node → node 1
2. PV node → 1 active power equation (P_2)
3. PQ node → 1 active and 1 reactive power equation (P_3,Q_3)
4. PV node → 1 active power equation (P_4).

Node Type	**Number of nodes**	**Known Variables**	**Unkown Variables**	**Number of equations per bus type**
Slack bus	1	U_1 and δ_1	P_1 and Q_1	0
PV buses	2	P_2 and U_2 P_4 and U_4	δ_2 and Q_2 δ_4 and Q_4	2 (related to P_2 and P_4)
PQ bus	4-2-1	P_3 and Q_3	δ_3 and U_3	2 (related to P_3 and Q_3)

Total of 4 equations, and 4 state variables, $\mathbf{x} = [\delta_2\ \delta_3\ \delta_4\ U_3]^{\mathrm{T}}$

1.6 NEWTON–RAPHSON METHOD

- The power flow equations cannot be solved analytically due to the complexity of the non linear equations.

- Newton–Raphson is an iterative method:

$$1:\ \mathbf{J}(\mathbf{x}^k)\Delta\mathbf{x}^k = \mathbf{h}(\mathbf{x}^k)$$

$$2:\ \mathbf{x}^{k+1} = \mathbf{x}^k + \Delta\mathbf{x}^k$$

$$\mathbf{x} = [\boldsymbol{\delta}\quad \mathbf{U}]^T = \left[\delta_2, \mathrm{K}\ , \delta_N, U_{N_g+2} \mathrm{K}\ , U_N\right]^T$$

- **h** is a vector which holds the mismatch power flow equations:

$$\mathbf{h} = [\Delta\mathbf{P}(\mathbf{x})\ \Delta\mathbf{Q}(\mathbf{x})]^{\mathrm{T}}$$

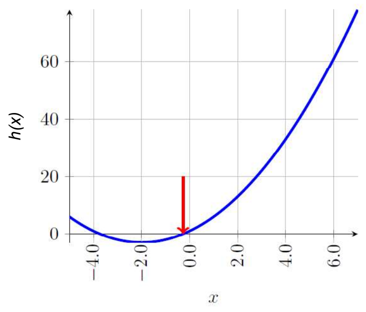
- **J** is called the power flow Jacobian matrix. **J** = -**H**, where **H** is the Jacobian of the power mismatches, i.e. **H**Δ**x**=-**h**(**x**)

1.6 NEWTON–RAPHSON METHOD

Example 5: 2nd order polynomial

1. Suppose function y=h(x)
2. y depends non-linearly on x (e.g. $y = 1 + 4x + x^2$)
3. This is a 2nd order *polynomial*
4. Solving y=0 through *abc formula*: $x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$
5. *Roots*: x1=-0.268 & x2 =-3.73
6. Similar formulas exist for 3rd and 4th order polynomials
7. Higher order polynomials cannot be solved **analytically**

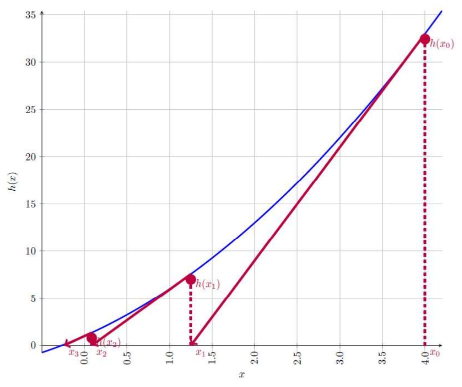
$\rightarrow$ **Numerical solution needed!**



1.6 NEWTON–RAPHSON METHOD

Example 5: 2nd order polynomial

1. Let the actual root be x_*
2. Make a first *initial* guess (e.g. $x_0 = 4$)
3. Calculate $h(x_0)$
4. Calculate the derivative of $h(x_0)$, $\frac{dh(x_0)}{dx}$
5. Then $x_1 = x_0 - \frac{h(x_0)}{dh(x_0)/dx}$
6. Continue until $|h(x_n) - h(x_{n-1})| < \varepsilon$



Coverges rapidly, but needs to determine $\frac{dh(x_0)}{dx}$ at every iteration

1.6 NEWTON–RAPHSON METHOD

The mismatch equations are nothing more than the power flow equations, cleverly re-written, so that they become zero when the method converges:

$$\Delta P_i(\mathbf{x}) = \underbrace{P_i}_{\text{Specified for node } i} - P_i(\mathbf{x}) = P_i - \underbrace{\sum_{n=1}^{N} U_i U_n Y_{in}\cos(\delta_i - \delta_n - \theta_{in})}_{\text{Iteratively computed for node } i}$$

$$\Delta Q_i(\mathbf{x}) = \underbrace{Q_i}_{\text{Specified for node } i} - Q_i(\mathbf{x}) = Q_i - \underbrace{\sum_{n=1}^{N} U_i U_n Y_{in}\sin(\delta_i - \delta_n - \theta_{in})}_{\text{Iteratively computed for node } i}$$

$$\mathbf{x} = [\boldsymbol{\delta} \quad \mathbf{U}]^T = \left[\delta_2, \ldots, \delta_N, U_{N_g+2} \ldots, U_N\right]^T$$ Updated throughout the iterations

1.6 NEWTON–RAPHSON METHOD

The Jacobian Matrix **J** contains the partial derivatives of the equations of the injected P and Q with respect to the state variables, that is

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_{11} & \mathbf{J}_{12} \\ \mathbf{J}_{21} & \mathbf{J}_{22} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial \mathbf{f_p}}{\partial \boldsymbol{\delta}} & \dfrac{\partial \mathbf{f_p}}{\partial \mathbf{U}} \\ \dfrac{\partial \mathbf{f_q}}{\partial \boldsymbol{\delta}} & \dfrac{\partial \mathbf{f_q}}{\partial \mathbf{U}} \end{bmatrix} \Rightarrow \mathbf{J} = \begin{array}{c} \mathbf{J}_{11} \\ \\ \\ \mathbf{J}_{21} \end{array} \left[\begin{array}{ccc|ccc} \dfrac{\partial P_2}{\partial \delta_2} & \cdots & \dfrac{\partial P_2}{\partial \delta_N} & \dfrac{\partial P_2}{\partial U_{N_g+2}} & \cdots & \dfrac{\partial P_2}{\partial U_N} \\ \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ \dfrac{\partial P_N}{\partial \delta_2} & \cdots & \dfrac{\partial P_N}{\partial \delta_N} & \dfrac{\partial P_N}{\partial U_{N_g+2}} & \cdots & \dfrac{\partial P_N}{\partial U_N} \\ \hline \dfrac{\partial Q_{N_g+2}}{\partial \delta_2} & \cdots & \dfrac{\partial Q_{N_g+2}}{\partial \delta_N} & \dfrac{\partial Q_{N_g+2}}{\partial U_{N_g+2}} & \cdots & \dfrac{\partial Q_{N_g+2}}{\partial U_N} \\ \vdots & \cdots & \vdots & \cdots & \cdots & \vdots \\ \dfrac{\partial Q_N}{\partial \delta_2} & \cdots & \dfrac{\partial Q_N}{\partial \delta_N} & \dfrac{\partial Q_N}{\partial U_{N_g+2}} & \cdots & \dfrac{\partial Q_N}{\partial U_N} \end{array}\right] \begin{array}{c} \mathbf{J}_{12} \\ \\ \\ \mathbf{J}_{22} \end{array}$$

J contains four submatrices, each one with a myriad of partial derivatives

1.6 NEWTON–RAPHSON METHOD

So the following equation system is solved at every iteration:

$$\underbrace{\begin{bmatrix} \frac{\partial P_2}{\partial \delta_2} & \cdots & \frac{\partial P_2}{\partial \delta_N} & \frac{\partial P_2}{\partial U_{N_g+2}} & \cdots & \frac{\partial P_2}{\partial U_N} \\ \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ \frac{\partial P_N}{\partial \delta_2} & \cdots & \frac{\partial P_N}{\partial \delta_N} & \frac{\partial P_N}{\partial U_{N_g+2}} & \cdots & \frac{\partial P_N}{\partial U_N} \\ \frac{\partial Q_{N_g+2}}{\partial \delta_2} & \cdots & \frac{\partial Q_{N_g+2}}{\partial \delta_N} & \frac{\partial Q_{N_g+2}}{\partial U_{N_g+2}} & \cdots & \frac{\partial Q_{N_g+2}}{\partial U_N} \\ \vdots & \cdots & \vdots & \cdots & \cdots & \vdots \\ \frac{\partial Q_N}{\partial \delta_2} & \cdots & \frac{\partial Q_N}{\partial \delta_N} & \frac{\partial Q_N}{\partial U_{N_g+2}} & \cdots & \frac{\partial Q_N}{\partial U_N} \end{bmatrix}^{(k)}}_{\text{Jacobian}} \underbrace{\begin{bmatrix} \Delta \delta_2^{(k)} \\ \vdots \\ \Delta \delta_N^{(k)} \\ \Delta U_{N_g+2}^{(k)} \\ \vdots \\ \Delta U_N^{(k)} \end{bmatrix}}_{\text{Corrections}} = \underbrace{\begin{bmatrix} \Delta P_2^{(k)} \\ \vdots \\ \Delta P_N^{(k)} \\ \Delta Q_{N_g+2}^{(k)} \\ \vdots \\ \Delta Q_N^{(k)} \end{bmatrix}}_{\text{Mismatches}}$$

1.6 NEWTON–RAPHSON METHOD

Example 6: Partial derivatives

Suppose a function with multiple variables:

$$f(x,y) = 1 + 2x + x^2 y + 3y$$

Partial derivative w.r.t. x: regard y terms as constants!

$$\frac{\partial f(x,y)}{\partial x} = 2 + 2xy$$

Partial derivative w.r.t. y: regard x terms as constants!

$$\frac{\partial f(x,y)}{\partial y} = x^2 + 3$$

1.6 NEWTON–RAPHSON METHOD

Calculation of the terms inside the Jacobian Matrix

➢ **J**$_{11}$ block:

Diagonal elements: $$\frac{\partial P_i}{\partial \delta_i} = \sum_{\substack{n=1 \\ n \neq i}}^{N} U_i U_n Y_{in} \sin(\theta_{in} + \delta_n - \delta_i)$$

Off-diagonal elements: $$\frac{\partial P_i}{\partial \delta_j} = -U_i U_j Y_{ij} \sin(\theta_{ij} + \delta_j - \delta_i)$$

1.6 NEWTON–RAPHSON METHOD

➢ **J**$_{12}$ block:

Diagonal elements: $$\frac{\partial P_i}{\partial U_i} = 2U_i Y_{ii} \cos(\theta_{ii}) + \sum_{\substack{n=1 \\ n \neq i}}^{N} U_n Y_{in} \cos(\theta_{in} + \delta_n - \delta_i)$$

Off-diagonal elements: $$\frac{\partial P_i}{\partial U_j} = U_i Y_{ij} \cos(\theta_{ij} + \delta_j - \delta_i)$$

➢ **J**$_{21}$ block:

Diagonal elements: $$\frac{\partial Q_i}{\partial \delta_i} = \sum_{\substack{n=1 \\ n \neq i}}^{N} U_i U_n Y_{in} \cos(\theta_{in} + \delta_n - \delta_i)$$

Off-diagonal elements: $$\frac{\partial Q_i}{\partial \delta_j} = -U_i U_j Y_{ij} \cos(\theta_{ij} + \delta_j - \delta_i)$$

1.6 NEWTON–RAPHSON METHOD

➢ $\mathbf{J}_{22}$ block:

Diagonal elements: $$\frac{\partial Q_i}{\partial U_i} = -2U_i Y_{ii} \sin(\theta_{ii}) + \sum_{\substack{n=1 \\ n \neq i}}^{N} U_n Y_{in} \sin(\theta_{in} + \delta_n - \delta_i)$$

Off-diagonal elements: $$\frac{\partial Q_i}{\partial U_j} = -U_i Y_{ij} \sin(\theta_{ij} + \delta_j - \delta_i)$$

1.7 ITERATIVE POWER FLOW CALCULATION

1. Build the network admittance matrix $\mathbf{Y}_{\mathbf{bus}}$
2. Make an initial estimation: $k = 0,\ \mathbf{x}^k = \left[\boldsymbol{\delta}^k \quad \mathbf{U}^k\right]^T$
3. Calculate the mismatches: $\mathbf{h}(\mathbf{x}^k) = \left[\Delta\mathbf{P}_i(\mathbf{x}^k) \quad \Delta\mathbf{Q}_i(\mathbf{x}^k)\right]^T$
4. Perform the stop test: $\begin{cases} \max(|\mathbf{h}(\mathbf{x}^k)|) < \varepsilon ? \Rightarrow \textit{stop}. \\ \max(|\mathbf{h}(\mathbf{x}^k)|) > \varepsilon ? \Rightarrow \textit{go to } 5. \end{cases}$
5. Construct the Jacobian Matrix: $\mathbf{J}^k = \mathbf{J}(\mathbf{x}^k)$
6. Calculate the corrections from: $\Delta\mathbf{x}^k = \left[\Delta\boldsymbol{\delta}^k \quad \Delta\mathbf{U}^k\right]^T = \mathbf{J}(\mathbf{x}^k)^{-1} \cdot \mathbf{h}(\mathbf{x}^k)$
7. Add the corrections to the initial estimation:

$$\text{New iteration } k+1:\ \mathbf{x}^{k+1} = \mathbf{x}^k + \Delta\mathbf{x}^k \Rightarrow \begin{cases} \boldsymbol{\delta}^{k+1} = \boldsymbol{\delta}^k + \Delta\boldsymbol{\delta}^k \\ \mathbf{U}^{k+1} = \mathbf{U}^k + \Delta\mathbf{U}^k \end{cases}$$

ε is a pre-specified tolerance (e.g. 0.0001)

1.8 DECOUPLED POWER FLOW (DPF)

For real electrical systems, inverting the Jacobian matrix at every iteration of the power flow calculation process increases the computational burden.

Solution: use the Jacobian matrix calculated at step k = 0, in all steps of the power flow calculation. Approach usually termed as dishonest Newton method

$$\mathbf{J}^{k} = \mathbf{J}^{0} = \mathbf{J}(\mathbf{x}^{0})$$

Advantage: Lower computational burden.

Disadvantage: More iterations are needed to reach the final results (longer convergence).

1.8 DECOUPLED POWER FLOW (DPF)

Simplifications were introduced by Stott and Alsac in 1974:

1. Resistance of the overhead lines much smaller than the reactance (X >> R)

$$\overline{Y}_{ij} = -\frac{1}{jX_{ij}} = j\frac{1}{X_{ij}} = Y_{ij}e^{j\frac{\pi}{2}}$$

2. Small differences between the voltage angles i.e. $(\delta_i - \delta_j) \to 0$

$$\sin(\delta_i - \delta_j) \approx \delta_i - \delta_j$$

$$\cos(\delta_i - \delta_j) \approx 1$$

1.8 DECOUPLED POWER FLOW (DPF)

Consequences:

- ➢ $\mathbf{J}_{12}$ and $\mathbf{J}_{21} = 0$
- ➢ Decoupling of the active power and the reactive power channels:

$$\mathbf{J}(\mathbf{x}^k)\Delta\mathbf{x}^k = \mathbf{h}(\mathbf{x}^k)$$

$$\begin{bmatrix} \mathbf{J}_{11}^0 & 0 \\ 0 & \mathbf{J}_{22}^0 \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{\delta}^k \\ \Delta\mathbf{U}^k \end{bmatrix} = \begin{bmatrix} \Delta\mathbf{P}_i\left(\mathbf{x}^k\right) \\ \Delta\mathbf{Q}_i\left(\mathbf{x}^k\right) \end{bmatrix}$$

$$\Rightarrow \begin{cases} \Delta\boldsymbol{\delta}^k = \left(\mathbf{J}_{11}^0\right)^{-1}\Delta\mathbf{P}_i\left(\mathbf{x}^k\right) \\ \Delta\mathbf{U}^k = \left(\mathbf{J}_{22}^0\right)^{-1}\Delta\mathbf{Q}_i\left(\mathbf{x}^k\right) \end{cases}$$

As long as the mismatch equations are evaluated in the exact manner, i.e. without the simplifications, the DPF method will provide accurate solutions, but it will just need more iterations to converge.

1.8 DECOUPLED POWER FLOW (DPF)

So the following equation system is solved at every iteration:

$$\mathbf{J}_{11}^0 = \mathbf{B}^{\dagger}$$

$$-\begin{bmatrix} B_{22} & \cdots & B_{2N} \\ \vdots & \cdots & \vdots \\ B_{N2} & \cdots & B_{NN} \end{bmatrix} \begin{bmatrix} \Delta\delta_2 \\ \vdots \\ \Delta\delta_N \end{bmatrix} = \begin{bmatrix} \dfrac{\Delta P_2}{U_2} \\ \vdots \\ \dfrac{\Delta P_N}{U_N} \end{bmatrix}$$

$$\mathbf{J}_{22}^0 = \mathbf{B}^{\dagger\dagger}$$

$$-\begin{bmatrix} B_{N_g+2,N_g+2} & \cdots & B_{N_g+2,N} \\ \vdots & \cdots & \vdots \\ B_{N,N_g+2} & \cdots & B_{NN} \end{bmatrix} \begin{bmatrix} \Delta U_{N_g+2} \\ \vdots \\ \Delta U_N \end{bmatrix} = \begin{bmatrix} \dfrac{\Delta Q_{N_g+2}}{U_{N_g+2}} \\ \vdots \\ \dfrac{\Delta Q_N}{U_N} \end{bmatrix}$$

- ❖ $\mathbf{B}^{\dagger}$ and $\mathbf{B}^{\dagger\dagger}$ are taken from the imaginary part of $\mathbf{Y}_{\mathbf{bus}}$.
- ❖ Row and column corresponding to the slack bus is omitted

1.8 DECOUPLED POWER FLOW (DPF)

1. Build the network admittance matrix $\mathbf{Y_{bus}}$
2. Make an initial estimation: $\mathbf{x}^k = \left[\boldsymbol{\delta}^k \quad \mathbf{U}^k\right]^T$
3. Calculate the active power mismatches: $\Delta P_i\left(U_i^k, \delta_i^k\right) / U_i^k$
4. Solve the voltage angle corrections: $\Delta\boldsymbol{\delta}^k = \left(\mathbf{J}_{11}^0\right)^{-1} \Delta\mathbf{P}_i\left(\mathbf{U}^k, \boldsymbol{\delta}^k\right)$
5. Update the voltage angles: $\boldsymbol{\delta}^{k+1} = \boldsymbol{\delta}^k + \Delta\boldsymbol{\delta}^k$
6. Use the updated voltage angles to calculate the reactive power mismatches $\Delta Q_i\left(U_i^k, \delta_i^{k+1}\right) / U_i^k$
7. Solve the voltage magnitude corrections: $\Delta\mathbf{U}^k = \left(\mathbf{J}_{22}^0\right)^{-1} \Delta\mathbf{Q}_i\left(\mathbf{U}^k, \boldsymbol{\delta}^{k+1}\right)$
8. Update the voltage magnitudes: $\mathbf{U}^{k+1} = \mathbf{U}^k + \Delta\mathbf{U}^k$
9. Perform the stop test: $\begin{cases} \max\left(\left|\mathbf{h}(\mathbf{x}^k)\right|\right) < \varepsilon ? \Rightarrow stop. \\ \max\left(\left|\mathbf{h}(\mathbf{x}^k)\right|\right) > \varepsilon ? \Rightarrow go\ to\ 3. \end{cases}$

ε is a pre-specified tolerance (e.g. 0.0001)

1.9 "DC" POWER FLOW (DCPF)

➢Based on the same hypothesis of the DPF, the non linear equations are linearized to solve only the active power channel.

➢Further speeds up the computation: only 1 matrix inversion needed.

➢Not an accurate solution.

$$\Delta P_i(\mathbf{x}) = P_i - P_i(\mathbf{x}) = P_i - \sum_{n=1}^{N} U_i U_n Y_{in} \cos\left(\delta_i - \delta_n - \theta_{in}\right) = 0$$

Original power flow equations

$$U_i = U_j = 1\ [\mathrm{pu}],\ \overline{Y}_{ij} = Y_{ij} e^{j\frac{\pi}{2}},\ \delta_i - \delta_j \to 0$$

DC power flow hypotheses

1.9 “DC” POWER FLOW (DCPF)

Resulting DCPF equation:

$$P_i = \sum_{n=1}^{N} U_i U_n Y_{in} \cos(\delta_i - \delta_n - \theta_{in}) = -\sum_{\substack{n=1 \\ n\neq 1}}^{N} Y_{in} \sin(\delta_n - \delta_i)$$

$$\Rightarrow P_i \approx \sum_{\substack{n=1 \\ n\neq 1}}^{N} Y_{ij} (\delta_i - \delta_n)$$

In Matrix form, it is possible to demonstrate that:

Row and column corresponding to the slack bus deleted

$$\mathbf{P} = -\mathbf{B}^{\dagger} \cdot \boldsymbol{\delta}$$

$$\Rightarrow \boldsymbol{\delta} = \left(-\mathbf{B}^{\dagger}\right)^{-1} \cdot \mathbf{P}$$

or

$$\begin{bmatrix} P_2 \\ \mathrm{M} \\ P_N \end{bmatrix} = - \begin{bmatrix} B_{22} & \dots & B_N \\ \mathrm{M} & \dots & \mathrm{M} \\ B_{N2} & \dots & B_{NN} \end{bmatrix} \begin{bmatrix} \delta_2 \\ \mathrm{M} \\ \delta_N \end{bmatrix}$$

1.10 SUITABILITY OF STUDIED POWER FLOW CALCULATION METHODS

Application	**Gauss-Seidel**	**Newton-Raphson**	**DPF**	**DCPF**
Small networks	X	X		
Large networks		X	X	
N-1 analysis		X	X	X
Market studies				X