Análise de Sistemas de Potência

Aula 08: Fluxo de Carga

Prof. Lucas Melo

Universidade Federal do Ceará

11 de setembro de 2024

Este material é inteiramente baseado no livro “Power Systems Design and Analisys” (6th Edition) dos autores J. Duncan Glover, Thomas J. Overbye e Mulukutlas S. Sarma.

Sumário I

Sumário II

Introduction

Successful power system operation under normal balanced three-phase steadystate conditions requires the following:

- Generation supplies the demand (load) plus losses.
- Bus voltage magnitudes remain close to rated values.
- Generators operate within specified real and reactive power limits.
- Transmission lines and transformers are not overloaded.

The power flow (sometimes also called the load flow) is the basic tool for investigating these requirements. The power flow determines the voltage magnitude and angle at each bus in a power system under balanced three-phase steady-state conditions.

Introduction

It also computes real and reactive power flows for all equipment interconnecting the buses, as well as equipment losses.

Both existing power systems and proposed changes, including new generation and transmission, are of interest.

Conventional nodal or loop analysis is not suitable for power flow studies because the input data for loads are normally given in terms of power, not impedance.

Also, generators are considered to be power sources, not voltage or current sources.

Introduction

The power flow problem is therefore formulated as a set of nonlinear algebraic equations suitable for computer solution.

Since balanced three-phase steady-state conditions are assumed, this chapter uses only positive-sequence networks. Also, all power flow equations and input/output data are given in per unit (p.u.).

The Power Flow Problem

The power flow problem is the computation of voltage magnitude and phase angle at each bus in a power system under balanced three-phase steady-state conditions.

As a by-product of this calculation, real and reactive power flows in equipment such as transmission lines and transformers, as well as equipment losses, can be computed.

The starting point for a power flow problem is a single-line (oneline) diagram of the power system, from which the input data can be obtained. Input data consist of bus data, transmission line data, and transformer data.

As shown in Figure below, the following four variables are associated with each bus k :

- **voltage magnitude V_k,**
- **phase angle δ_k,**
- **net real power P_k, and**
- **reactive power Q_k supplied to the bus.**

At each bus, two of these variables are specified as input data, and the other two are unknowns to be computed by the power flow program.

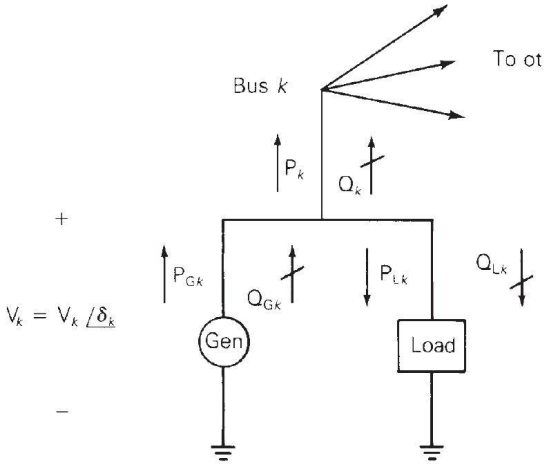


FIGURE 6.1

Bus variables V_k, δ_k, P_k, and Q_k

For convenience, the power delivered to bus k is separated into generator and load terms. That is:

$$P_k = P_{Gk} - P_{Lk}$$
$$Q_k = Q_{Gk} - Q_{Lk}$$

Each bus k is categorized into one of the following three bus types:

1. Swing bus (or slack bus)-There is only one swing bus, which for convenience is numbered bus 1 in this text. The swing bus is a reference bus for which V_1/δ_1, is input data with the angle typically zero degrees and the voltage magnitude close to 1.0 per unit. The power flow computes P_1 and Q_1.

2. **Load (PQ) bus- P_k and Q_k are input data. The power flow computes V_k and δ_k. Most buses in typical power flows are load buses.**
3. **Voltage controlled (PV) bus –P_k and V_k are input data. The power flow program computes Q_k and δ_k. Examples are buses to which generators, switched shunt capacitors, or static var systems are connected. Maximum and minimum reactive power (var) limits $Q_{Gk\,max}$ and $Q_{Gk\,min}$ that this equipment can supply are also input data. If an upper or lower reactive power limit is reached, then the reactive power output of the generator is held at the limit, and the bus is modeled as a PQ bus. Another example is a bus to which a tap-changing transformer is connected; the power flow then computes the tap setting.**

Note that when bus k is a load bus with no generation, $\mathbf{P}_k = -\mathbf{P}_{\mathbf{L}k}$ is negative; that is, the real power supplied to bus k in Figure 6.1 is negative. If the load is inductive, $\mathbf{Q}_k = -\mathbf{Q}_{\mathbf{L}k}$ is negative.

Transmission lines are represented by the equivalent π circuit. Transformers are also represented by equivalent circuits:

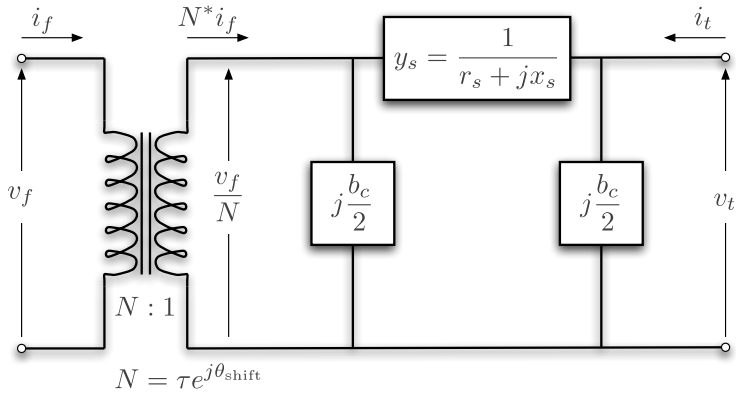
- Two-winding transformer;
- Three-winding transformer; or
- Tap-changing transformer.

Input data for each transmission line include the per-unit equivalent π circuit series impedance Z' and shunt admittance Y', the two buses to which the line is connected, and maximum MVA rating.

Similarly, input data for each transformer include per-unit winding impedances Z, the per-unit exciting branch admittance Y, the buses to which the windings are connected, and maximum MVA ratings.

Input data for tap-changing transformers also include maximum tap settings.

Both elements can be combined, as shown in figure below:



O modelo matemático da matriz de admitâncias nodais para esse ramo fica sendo então:

$$\begin{bmatrix} i_f \\ i_t \end{bmatrix} = Y_{br} \begin{bmatrix} v_f \\ v_t \end{bmatrix}$$

Com:

$$Y_{br} = \begin{bmatrix} \left(y_s + j\frac{b_c}{2}\right)\frac{1}{\tau^2} & -y_s\frac{1}{\tau e^{-j\theta_{\mathbf{shift}}}} \\ -y_s\frac{1}{\tau e^{j\theta_{\mathbf{shift}}}} & y_s + j\frac{b_c}{2} \end{bmatrix}$$

The bus admittance matrix Y_{bus} can be constructed from the line and transformer input data.

The elements of Y_{bus} are:

Diagonal elements: Y_{kk} = sum of admittances connected to bus k

Off-diagonal elements: $Y_{kn} = -($ sum of admittances connected between buses k and $n) k \neq n$.

Using Y_{bus} , the nodal equations for a power system network are written as:

$$\mathbf{I} = Y_{\text{bus}}\,\mathbf{V}$$

Where I is the N vector of source currents injected into each bus and V is the N vector of bus voltages. For bus k, the k th equation in equation above is:

$$I_k = \sum_{n=1}^{N} Y_{kn} V_n$$

The complex power delivered to bus k is:

$$S_k = \mathbf{P}_k + j\mathbf{Q}_k = V_k I_k^*$$

Power flow solutions by Gauss-Seidel are based on nodal equations, with each current source I_k:

$$I_k = \sum_{n=1}^{N} Y_{kn} V_n$$

Inserted into power equations:

$$I_k = \sum_{n=1}^{N} Y_{kn} V_n = \frac{P_k - jQ_k}{V_k^*}$$

This give the closed form of Gauss-Seidel equations are:

$$V_k(i+1) = \frac{1}{Y_{kk}}\left[\frac{\mathbf{P}_k - j\mathbf{Q}_k}{V_k^*(i)} - \sum_{n=1}^{k-1} Y_{kn}V_n(i+1) - \sum_{n=k+1}^{N} Y_{kn}V_n(i)\right]$$

Now, using the current I_k definition and applying in complex power equation delivered to bus k:

$$I_k = \sum_{n=1}^{N} Y_{kn} V_n$$

$$S_k = \mathbf{P}_k + j\mathbf{Q}_k = V_k I_k^* = V_k \left[\sum_{n=1}^{N} Y_{kn} V_n \right]^* \qquad k = 1, 2, \ldots, N$$

With the following notation,

$$V_n = \mathbf{V}_n e^{j\delta_n}$$
$$Y_{kn} = \mathbf{Y}_{kn} e^{j\theta_{kn}} = G_{kn} + jB_{kn} \quad k, n = 1, 2, \ldots, N$$

This equation becomes:

$$\mathbf{P}_k + j\mathbf{Q}_k = \mathbf{V}_k \sum_{n=1}^{N} \mathbf{Y}_{kn}\, \mathbf{V}_n e^{j\left(\delta_k - \delta_n - \theta_{kn}\right)}$$

Taking the real and imaginary parts of above equation, the power balance equations are written as either:

$$\mathbf{P}_k = \mathbf{V}_k \sum_{n=1}^{N} \mathbf{Y}_{kn}\, \mathbf{V}_n \cos\left(\delta_k - \delta_n - \theta_{kn}\right)$$

$$\mathbf{Q}_k = \mathbf{V}_k \sum_{n=1}^{N} \mathbf{Y}_{kn}\, \mathbf{V}_n \sin\left(\delta_k - \delta_n - \theta_{kn}\right) \qquad k = 1, 2, \ldots, N$$

Or when the Y_{kn} is expressed in rectangular coordinates as:

$$\mathbf{P}_k = \mathbf{V}_k \sum_{n=1}^{N} V_n \left[G_{kn} \cos\left(\delta_k - \delta_n\right) + B_{kn} \sin\left(\delta_k - \delta_n\right) \right]$$

$$\mathbf{Q}_k = \mathbf{V}_k \sum_{n=1}^{N} V_n \left[G_{kn} \sin\left(\delta_k - \delta_n\right) - B_{kn} \cos\left(\delta_k - \delta_n\right) \right] \qquad k = 1, 2, \ldots, N$$

Power flow solutions by Newton-Raphson are based on these nonlinear power flow equations, in polar or cartesian forms.

Iterative Solutions To Linear Algebraic Equations: Jacobi And Gauss-Seidel

A general iterative solution to $\mathbf{Ax} = \mathbf{y}$ proceeds as follows.

First select an initial guess $\mathbf{x}(0)$. Then use:

$$\mathbf{x}(i+1) = \mathbf{g}[\mathbf{x}(i)] \quad i = 0, 1, 2, \ldots$$

Where $\mathbf{x}(i)$ is the i th guess and $\mathbf{g}$ is an N vector of functions that specify the iteration method.

Continue this procedure until the following stopping condition is satisfied, as:

$$\left| \frac{x_k(i+1) - x_k(i)}{x_k(i)} \right| < \varepsilon \quad \textbf{for all } k = 1, 2, \ldots, N$$

where $x_k(i)$ is the k th component of $\mathbf{x}(i)$ and ε is a specified tolerance level.

The following questions are pertinent:

1. **Will the iteration procedure converge to the unique solution?**
2. **What is the convergence rate (how many iterations are required)?**
3. **When using a digital computer, what are the computer storage and time requirements?**

These questions are addressed for two specific iteration methods: Jacobi and Gauss-Seidel.

The Jacobi method is obtained by considering the k th equation of $\mathbf{Ax} = \mathbf{y}$, as follows:

$$y_k = \mathbf{A}_{k1}x_1 + \mathbf{A}_{k2}x_2 + \cdots + \mathbf{A}_{kk}x_k + \cdots + \mathbf{A}_{kN}x_N$$

Solving for x_k:

$$\begin{aligned} x_k &= \frac{1}{\mathbf{A}_{kk}}\left[y_k - \left(\mathbf{A}_{k1}x_1 + \cdots + \mathbf{A}_{k,k-1}x_{k-1} + \mathbf{A}_{k,k+1}x_{k+1} + \cdots + \mathbf{A}_{kN}x_N\right)\right] \\ &= \frac{1}{\mathbf{A}_{kk}}\left[y_k - \sum_{n=1}^{k-1} \mathbf{A}_{kn}x_n - \sum_{n=k+1}^{N} \mathbf{A}_{kn}x_n\right] \end{aligned}$$

The Jacobi method uses the "old" values of $\mathbf{x}(i)$ at iteration i on the right side of last equation to generate the "new" value $x_k(i+1)$ on the left side.

That is:

$$x_k(i+1) = \frac{1}{\mathbf{A}_{kk}} \left[y_k - \sum_{n=1}^{k-1} \mathbf{A}_{kn} x_n(i) - \sum_{n=k+1}^{N} \mathbf{A}_{kn} x_n(i) \right] k = 1, 2, \ldots, N$$

The Jacobi method also can be written in the following matrix format:

$$\mathbf{x}(i+1) = \mathbf{M}\mathbf{x}(i) + \mathbf{D}^{-1}\mathbf{y}$$

where

$$\mathbf{M} = \mathbf{D}^{-1}(\mathbf{D} - \mathbf{A})$$

and

$$\mathbf{D} = \begin{bmatrix} \mathbf{A}_{11} & 0 & 0 & \cdots & 0 \\ 0 & \mathbf{A}_{22} & 0 & \cdots & 0 \\ 0 & \vdots & \vdots & & \vdots \\ \vdots & & & & 0 \\ 0 & 0 & 0 & \cdots & \mathbf{A}_{NN} \end{bmatrix}$$

For Jacobi, D consists of the diagonal elements of the A matrix.

The Gauss-Seidel method is given by:

$$x_k(i+1) = \frac{1}{A_{kk}} \left[y_k - \sum_{n=1}^{k-1} \mathbf{A}_{kn} x_n(i+1) - \sum_{n=k+1}^{N} \mathbf{A}_{kn} x_n(i) \right]$$

The Jacobi method is also called the Gauss method.

Comparing equations, note that Gauss-Seidel is similar to Jacobi except that during each iteration, the 'new' values, $x_n(i+1)$, for $n < k$ are used on the right side of equation to generate the 'new' value $x_k(i+1)$ on the left side.

The Gauss-Seidel method also can be written in the matrix format, where:

$$\mathbf{D} = \begin{bmatrix} \mathbf{A}_{11} & 0 & 0 & \cdots & 0 \\ \mathbf{A}_{21} & \mathbf{A}_{22} & 0 & \cdots & 0 \\ \vdots & \vdots & & & \vdots \\ \mathbf{A}_{N1} & \mathbf{A}_{N2} & \cdots & & \mathbf{A}_{NN} \end{bmatrix}$$

For Gauss-Seidel, D is the lower triangular portion of A, whereas for Jacobi, D in is the diagonal portion of A.

The convergence rate is faster with Gauss-Seidel for some A matrices, but faster with Jacobi for other A matrices.

In some cases, one method diverges while the other converges.

In other cases both methods diverge, as illustrated by examples.

If any diagonal element A_{kk} equals zero, then Jacobi and Gauss-Seidel are undefined, because the right-hand sides of (6.2.5) and (6.2.9) are divided by A_{kk}.

Also, if any one diagonal element has too small a magnitude, these methods will diverge.

In Examples 6.3 and 6.4, Jacobi and Gauss-Seidel converge, since the diagonals (10 and 9) are both large; in Example 6.5, however, the diagonals (5 and 2) are small compared to the off-diagonals, and the methods diverge.

Power Flow Solution By Gauss-Seidel

Nodal equations $\mathbf{I} = Y_{\text{bus}}\,\mathbf{V}$ are a set of linear equations analogous to $\mathbf{y} = \mathbf{Ax}$, solved in a later section using Gauss-Seidel.

Since power flow bus data consists of P_k and Q_k for load buses or P_k and V_k for voltage-controlled buses, nodal equations do not directly fit the linear equation format;

The current source vector I is unknown and the equations are actually nonlinear.

For each load bus, I_k can be calculated, giving:

$$I_k = \sum_{n=1}^{N} Y_{kn} V_n = \frac{P_k - jQ_k}{V_k^*}$$

Applying the Gauss-Seidel method to the nodal equations with I_k given above, we can obtain for V_k:

$$V_k(i+1) = \frac{1}{Y_{kk}} \left[\frac{\mathbf{P}_k - j\mathbf{Q}_k}{V_k^*(i)} - \sum_{n=1}^{k-1} Y_{kn} V_n(i+1) - \sum_{n=k+1}^{N} Y_{kn} V_n(i) \right]$$

Equation above can be applied twice during each iteration for load buses, first using $V_k^*(i)$, then replacing $V_k^*(i)$, by $V_k^*(i+1)$ on the right side of voltages equations above.

For a voltage-controlled bus, $\mathbf{Q}_k$ is unknown but can be calculated, giving:

$$\mathbf{Q}_k = \mathbf{V}_k(i) \sum_{n=1}^{N} \mathbf{Y}_{kn}\, \mathbf{V}_n(i)\, \mathbf{sin}\left[\delta_k(i) - \delta_n(i) - \theta_{kn}\right]$$

Also,

$$\mathbf{Q}_{\mathbf{G}k} = \mathbf{Q}_k + \mathbf{Q}_{\mathbf{L}k}$$

If the calculated value of Q_{Gk} does not exceed its limits, then Q_k is used in $V_k(i+1)$ equation to calculate:

$$\mathbf{V_k}(i+1) = \mathbf{V_k}(i+1)\underline{/\delta_k(i+1)}$$

Then the magnitude $\mathbf{V_k}(i+1)$ is changed to V_k, which is input data for the voltage-controlled bus.

Thus, use

$$V_k(i+1) = \frac{1}{Y_{kk}}\left[\frac{\mathbf{P}_k - j\mathbf{Q}_k}{V_k^*(i)} - \sum_{n=1}^{k-1} Y_{kn}V_n(i+1) - \sum_{n=k+1}^{N} Y_{kn}V_n(i)\right]$$

To compute only the angle $\delta_k(i+1)$ for voltage-controlled buses.

If the calculated value exceeds its limit $Q_{Gk\,max}$ or $Q_{Gk\,min}$ during any iteration, then the bus type is changed from a voltage-controlled bus to a load bus, with Q_{Gk} set to its limit value.

Under this condition, the voltage-controlling device (e.g., generator, capacitor bank, static var compensator) is not capable of maintaining V_k as specified by the input data. The power flow then calculates a new value of V_k.

For the swing bus, denoted bus $1, V_1$ and δ_1 are input data. As such, no iterations are required for the swing bus.

After the iteration process has converged, one pass through:

$$\mathbf{P}_k = \mathbf{V}_k \sum_{n=1}^{N} \mathbf{Y}_{kn} \mathbf{V}_n \cos\left(\delta_k - \delta_n - \theta_{kn}\right)$$

$$\mathbf{Q}_k = \mathbf{V}_k \sum_{n=1}^{N} \mathbf{Y}_{kn} \mathbf{V}_n \sin\left(\delta_k - \delta_n - \theta_{kn}\right)$$

Can be made to compute $\mathbf{P}_1$ and $\mathbf{Q}_1$.

Iterative Solutions To Nonlinear Algebraic Equations: Newton-Raphson

A set of nonlinear algebraic equations in matrix format is given by

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ \vdots \\ f_N(\mathbf{x}) \end{bmatrix} = \mathbf{y}$$

Where y and x are N vectors and f(x) is an N vector of functions. Given y and f(x), the objective is to solve for x.

The iterative methods described in last Section can be extended to nonlinear equations as follows. Rewriting last equation:

$$0 = \mathbf{y} - \mathbf{f}(\mathbf{x})$$

Adding Dx to both sides of equation above, where D is a square $N \times N$ invertible matrix.

$$\mathbf{Dx} = \mathbf{Dx} + \mathbf{y} - \mathbf{f}(\mathbf{x})$$

Premultiplying by $\mathbf{D}^{-1}$.

$$\mathbf{x} = \mathbf{x} + \mathbf{D}^{-1}[\mathbf{y} - \mathbf{f}(\mathbf{x})]$$

The old values $\mathbf{x}(i)$ are used on the right side of equation to generate the new values $\mathbf{x}(i+1)$ on the left side. That is:

$$\mathbf{x}(i+1) = \mathbf{x}(i) + \mathbf{D}^{-1}\{\mathbf{y} - \mathbf{f}[\mathbf{x}(i)]\}$$

For linear equations, $\mathbf{f}(\mathbf{x}) = \mathbf{A}\mathbf{x}$ and equation reduces to:

$$\mathbf{x}(i+1) = \mathbf{x}(i) + \mathbf{D}^{-1}[\mathbf{y} - \mathbf{A}\mathbf{x}(i)] = \mathbf{D}^{-1}(\mathbf{D} - \mathbf{A})\mathbf{x}(i) + \mathbf{D}^{-1}\mathbf{y}$$

Which is identical to the Jacobi and Gauss-Seidel methods. For nonlinear equations, the matrix D must be specified.

One method for specifying D, called Newton-Raphson, is based on the following Taylor series expansion of f(x) about an operating point $\mathbf{x}_0$:

$$\mathbf{y} = \mathbf{f}(\mathbf{x}_0) + \left.\frac{d\mathbf{f}}{d\mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_0} (\mathbf{x} - \mathbf{x}_0) \cdots$$

Neglecting the higher order terms and solving for x:

$$\mathbf{x} = \mathbf{x}_0 + \left[\left.\frac{d\mathbf{f}}{d\mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_0}\right]^{-1} [\mathbf{y} - \mathbf{f}(\mathbf{x}_0)]$$

The Newton-Raphson method replaces $\mathbf{x}_0$ by the old value $\mathbf{x}(i)$ and $\mathbf{x}$ by the new value $\mathbf{x}(i+1)$. Thus:

$$\mathbf{x}(i+1) = \mathbf{x}(i) + \mathbf{J}^{-1}(i)\{\mathbf{y} - \mathbf{f}[\mathbf{x}(i)]\}$$

Where

$$\mathbf{J}(i) = \left.\frac{d\mathbf{f}}{d\mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}(i)} = \left[\begin{array}{cccc} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_N} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_N} \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_N}{\partial x_1} & \frac{\partial f_N}{\partial x_2} & \cdots & \frac{\partial f_N}{\partial x_N} \end{array}\right]_{\mathbf{x}=\mathbf{x}(i)}$$

The $N \times N$ matrix $\mathbf{J}(i)$, whose elements are the partial derivatives is called the Jacobian matrix.

The Newton-Raphson method is similar to extended Gauss-Seidel, except that D is replaced by $\mathbf{J}(i)$.

Equation $\mathbf{x}(i+1) = \mathbf{x}(i) + \mathbf{J}^{-1}(i)\{\mathbf{y} - \mathbf{f}[\mathbf{x}(i)]\}$ contains the matrix inverse $\mathbf{J}^{-1}$.

Instead of computing $\mathbf{J}^{-1}$, this equation can be rewritten as follows:

$$\mathbf{J}(i)\Delta\mathbf{x}(i) = \Delta\mathbf{y}(i)$$

Where

$$\Delta\mathbf{x}(i) = \mathbf{x}(i+1) - \mathbf{x}(i)$$

and

$$\Delta\mathbf{y}(i) = \mathbf{y} - \mathbf{f}[\mathbf{x}(\mathbf{i})]$$

Then, during each iteration, the following four steps are completed:

STEP 1: Compute $\Delta\mathbf{y}(i)$ from $\Delta\mathbf{y}(i) = \mathbf{y} - \mathbf{f}[\mathbf{x}(\mathbf{i})]$.

STEP 2: Compute $\mathbf{J}(i)$.

STEP 3: Using Gauss elimination and back substitution, solve $\mathbf{J}(i)\Delta\mathbf{x}(i) = \Delta\mathbf{y}(i)$ for $\Delta\mathbf{x}(i)$.

STEP 4: Compute $\mathbf{x}(i + 1)$ from $\Delta\mathbf{x}(i) = \mathbf{x}(i + 1) - \mathbf{x}(i)$.

Experience from power flow studies has shown that Newton-Raphson converges in many cases where Jacobi and Gauss-Seidel diverge. Furthermore, the number of iterations required for convergence is independent of the dimension

Power Flow Solution By Newton-Raphson

Equations:

$$\mathbf{P}_k = \mathbf{V}_k \sum_{n=1}^{N} \mathbf{Y}_{kn}\, \mathbf{V}_n \cos\left(\delta_k - \delta_n - \theta_{kn}\right)$$

$$\mathbf{Q}_k = \mathbf{V}_k \sum_{n=1}^{N} \mathbf{Y}_{kn}\, \mathbf{V}_n \sin\left(\delta_k - \delta_n - \theta_{kn}\right) \qquad k = 1, 2, \ldots, N$$

Are analogous to the nonlinear equation y = f(x), solved by Newton-Raphson.

The x, y, and f vectors for the power flow problem are defined as:

$$\mathbf{x} = \begin{bmatrix} \delta \\ \mathbf{V} \end{bmatrix} = \begin{bmatrix} \delta_2 \\ \vdots \\ \delta_N \\ \mathbf{V}_2 \\ \vdots \\ \mathbf{V}_N \end{bmatrix}; \quad \mathbf{y} = \begin{bmatrix} \mathbf{P} \\ \mathbf{Q} \end{bmatrix} = \begin{bmatrix} \mathbf{P}_2 \\ \vdots \\ \mathbf{P}_N \\ \mathbf{Q}_2 \\ \vdots \\ \mathbf{Q}_N \end{bmatrix}$$

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} \mathbf{P}(\mathbf{x}) \\ \mathbf{Q}(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} \mathbf{P}_2(\mathbf{x}) \\ \vdots \\ \mathbf{P}_N(\mathbf{x}) \\ \mathbf{Q}_2(\mathbf{x}) \\ \vdots \\ \mathbf{Q}_N(\mathbf{x}) \end{bmatrix}$$

All V, P, and Q terms are in per-unit and δ terms are in radians.

The swing bus variables δ_1 and V_1 are omitted in equations, since they are already known.

Nonlinear Power Flow Equations in Newton Raphson then have the following form:

$$y_k = \mathbf{P}_k = \mathbf{P}_k(\mathbf{x}) = \mathbf{V}_k \sum_{n=1}^{N} \mathbf{Y}_{kn}\, \mathbf{V}_n \cos\left(\delta_k - \delta_n - \theta_{kn}\right)$$

$$y_{k+N} = \mathbf{Q_k} = \mathbf{Q}_k(\mathbf{x}) = \mathbf{V_k} \sum_{n=1}^{N} \mathbf{Y_{kn}} \mathbf{V}_n \sin\left(\delta_k - \delta_n - \theta_{kn}\right)$$

$$k = 2, 3, \ldots, N$$

The Jacobian matrix has the form:

$$\mathbf{J} = \left[\begin{array}{c|c} \mathbf{J1} & \mathbf{J2} \\ \hline \mathbf{J3} & \mathbf{J4} \end{array}\right] = \left[\begin{array}{ccc|ccc} \frac{\partial \mathbf{P}_2}{\partial \delta_2} & \cdots & \frac{\partial \mathbf{P}_2}{\partial \delta_N} & \frac{\partial \mathbf{P}_2}{\partial \mathbf{V}_2} & \cdots & \frac{\partial \mathbf{P}_2}{\partial \mathbf{V}_N} \\ \vdots & & \vdots & \vdots & & \vdots \\ \frac{\partial \mathbf{P}_N}{\partial \delta_2} & \cdots & \frac{\partial \mathbf{P}_N}{\partial \delta_N} & \frac{\partial \mathbf{P}_N}{\partial \mathbf{V}_2} & \cdots & \frac{\partial \mathbf{P}_N}{\partial \mathbf{V}_N} \\ \hline \frac{\partial \mathbf{Q}_2}{\partial \delta_2} & \cdots & \frac{\partial \mathbf{Q}_2}{\partial \delta_N} & \frac{\partial \mathbf{Q}_2}{\partial \mathbf{V}_2} & \cdots & \frac{\partial \mathbf{Q}_2}{\partial \mathbf{V}_N} \\ \vdots & & \vdots & \vdots & & \vdots \\ \frac{\partial \mathbf{Q}_N}{\partial \delta_2} & \cdots & \frac{\partial \mathbf{Q}_N}{\partial \delta_N} & \frac{\partial \mathbf{Q}_N}{\partial \mathbf{V}_2} & \cdots & \frac{\partial \mathbf{Q}_N}{\partial \mathbf{V}_N} \end{array}\right]$$

Equation J is partitioned into four blocks. The partial derivatives in each block, are given below.

For $n \neq k$:

$$\mathbf{J1}_{kn} = \frac{\partial \mathbf{P}_k}{\partial \delta_n} = \mathbf{V}_k \mathbf{Y_{kn}} \mathbf{V}_n \sin\left(\delta_k - \delta_n - \theta_{kn}\right)$$

$$\mathbf{J2}_{kn} = \frac{\partial \mathbf{P}_k}{\partial \mathbf{V}_n} = \mathbf{V}_k \mathbf{Y_{kn}} \cos\left(\delta_k - \delta_n - \theta_{kn}\right)$$

$$\mathbf{J3}_{kn} = \frac{\partial \mathbf{Q}_k}{\partial \delta_n} = -\mathbf{V}_k \mathbf{Y_{kn}} \mathbf{V}_n \cos\left(\delta_k - \delta_n - \theta_{kn}\right)$$

$$\mathbf{J4}_{kn} = \frac{\partial \mathbf{Q}_k}{\partial \mathbf{V}_n} = \mathbf{V}_k \mathbf{Y_{kn}} \sin\left(\delta_k - \delta_n - \theta_{kn}\right)$$

For $n = k$:

$$\mathbf{J1}_{kk} = \frac{\partial \mathbf{P}_k}{\partial \delta_k} = -\mathbf{V}_k \sum_{\substack{n=1 \\ n \neq k}}^{N} \mathbf{Y_{kn}} \mathbf{V}_n \sin\left(\delta_k - \delta_n - \theta_{kn}\right)$$

$$\mathbf{J2}_{kk} = \frac{\partial \mathbf{P}_k}{\partial \mathbf{V}_k} = \mathbf{V}_k \mathbf{Y_{kk}} \cos\theta_{kk} + \sum_{n=1}^{N} \mathbf{Y_{kn}} \mathbf{V}_n \cos\left(\delta_k - \delta_n - \theta_{kn}\right)$$

$$\mathbf{J3}_{kk} = \frac{\partial \mathbf{Q}_k}{\partial \delta_k} = \mathbf{V}_k \sum_{\substack{n=1 \\ n \neq k}}^{N} \mathbf{Y_{kn}} \mathbf{V}_n \cos\left(\delta_k - \delta_n - \theta_{kn}\right)$$

$$\mathbf{J4}_{kk} = \frac{\partial \mathbf{Q}_k}{\partial \mathbf{V}_k} = -\mathbf{V}_k \mathbf{Y_{kk}} \sin\theta_{kk} + \sum_{n=1}^{N} \mathbf{Y_{kn}} \mathbf{V}_n \sin\left(\delta_k - \delta_n - \theta_{kn}\right)$$

With $k, n = 2, 3, \ldots, N$.

Now apply to the power flow problem the four Newton-Raphson steps outlined in later section, starting with $\mathbf{x}(i) = \begin{bmatrix} \delta(i) \\ \mathbf{V}(i) \end{bmatrix}$ at the i th iteration.

STEP 1: Compute:

$$\Delta\mathbf{y}(i) = \begin{bmatrix} \Delta\mathbf{P}(i) \\ \Delta\mathbf{Q}(i) \end{bmatrix} = \begin{bmatrix} \mathbf{P} - \mathbf{P}[\mathbf{x}(i)] \\ \mathbf{Q} - \mathbf{Q}[\mathbf{x}(i)] \end{bmatrix}$$

STEP 2: Calculate the Jacobian matrix.

STEP 3: Use Gauss elimination and back substitution to solve:

$$\left[\begin{array}{c|c} \mathbf{J}1(i) & \mathbf{J}2(i) \\ \hline \mathbf{J}3(i) & \mathbf{J}4(i) \end{array}\right] \begin{bmatrix} \Delta\delta(i) \\ \Delta\mathbf{V}(i) \end{bmatrix} = \begin{bmatrix} \Delta\mathbf{P}(i) \\ \Delta\mathbf{Q}(i) \end{bmatrix}$$

STEP 4: Compute:

$$\mathbf{x}(i+1) = \begin{bmatrix} \delta(i+1) \\ \mathbf{V}(i+1) \end{bmatrix} = \begin{bmatrix} \delta(i) \\ \mathbf{V}(i) \end{bmatrix} + \begin{bmatrix} \Delta\delta(i) \\ \Delta\mathbf{V}(i) \end{bmatrix}$$

Starting with initial value $\mathbf{x}(0)$, the procedure continues until convergence is obtained or until the number of iterations exceeds a specified maximum.

Convergence criteria are often based on $\Delta\mathbf{y}(i)$ (called power mismatches) rather than on $\Delta\mathbf{x}(i)$ (phase angle and voltage magnitude mismatches).

For each voltage-controlled bus, the magnitude V_k is already known, and the function $Q_k(x)$ is not needed.

Therefore, V_k from the x vector can be omitted, as well as Q_k from the y vector. The column corresponding to partial derivatives with respect to V_k and the row corresponding to partial derivatives of $Q_k(x)$ can also be omitted from the Jacobian matrix.

Alternatively, rows and corresponding columns for voltage-controlled buses can be retained in the Jacobian matrix. Then during each iteration, the voltage magnitude $V_k(i+1)$ of each voltage-controlled bus is reset to V_k which is input data for that bus.

At the end of each iteration, $Q_k(x)$ is computed using (6.6.3) and $Q_{Gk} = Q_k(x) + Q_{Lk}$ for each voltage-controlled bus.

If the computed value of Q_{Gk} exceeds its limits, then the bus type is changed to a load bus with Q_{Gk} set to its limit value.

The power flow program also computes a new value for V_k.

Voltage Controlled Buses in Newton-Raphson Algorithm

Voltage-controlled buses to which tap-changing or voltage-regulating transformers are connected can be handled by various methods.

One method is to treat each of these buses as a load bus. The equivalent π circuit parameters are first calculated with tap setting $c = 1.0$ for starting.

During each iteration, the computed bus voltage magnitude is compared with the desired value specified by the input data.

If the computed voltage is low (or high), c is increased (or decreased) to its next setting, and the parameters of the equivalent π circuit as well as Y_{bus} are recalculated.

The procedure continues until the computed bus voltage magnitude equals the desired value within a specified tolerance or until the high or low tap-setting limit is reached.

Phase-shifting transformers can be handled in a similar way by using a complex turns ratio $c = 1.0\angle\underline{\alpha}$ **and by varying the phaseshift angle** α**.**

A method with faster convergence makes c **a variable and includes it in the x vector. An equation is then derived to enter into the Jacobian matrix [4].**

Comparing Gauss-Seidel and Newton-Raphson algorithms

In comparing the Gauss-Seidel and Newton-Raphson algorithms, experience from power flow studies has shown that Newton-Raphson converges in many cases where Jacobi and Gauss-Seidel diverge.

Furthermore, the number of iterations required for convergence is independent of the number of buses N for Newton-Raphson, but increases with N for Jacobi and Gauss-Seidel.

The principal advantage of the Jacobi and Gauss-Seidel methods had been their more modest memory storage requirements and their lower computational requirements per iteration.

However, with the vast increases in low-cost computer memory over the last several decades, coupled with the need to solve power flow problems with tens of thousands of buses, these advantages have been essentially eliminated.

Therefore the Newton-Raphson, or one of the derivative methods discussed in next Sections, are the preferred power flow solution approaches.

Fast Decoupled Power Flow

Contingencies are a major concern in power system operations. For example, operating personnel need to know what power flow changes will occur due to a particular generator outage or transmission-line outage.

Contingency information, when obtained in real time, can be used to anticipate problems caused by such outages and can be used to develop operating strategies to overcome the problems.

Fast power flow algorithms have been developed to give power flow solutions in seconds or less [8].

These algorithms are based on the following simplification of the Jacobian matrix.

Neglecting $\mathbf{J}_2(i)$ and $\mathbf{J}_3(i)$:

$$\left[\begin{array}{c|c} \mathbf{J}1(i) & \mathbf{J}2(i) \\ \hline \mathbf{J}3(i) & \mathbf{J}4(i) \end{array}\right] \left[\begin{array}{c} \Delta\delta(i) \\ \Delta\mathbf{V}(i) \end{array}\right] = \left[\begin{array}{c} \Delta\mathbf{P}(i) \\ \Delta\mathbf{Q}(i) \end{array}\right]$$

Reduces to two sets of decoupled equations:

$$\mathbf{J}_1(i)\Delta\delta(i) = \Delta\mathbf{P}(i)$$
$$\mathbf{J}_4(i)\Delta\mathbf{V}(i) = \Delta\mathbf{Q}(i)$$

The computer time required to solve these equations is significantly less than that required to solve complete equations.

Further reduction in computer time can be obtained from additional simplification of the Jacobian matrix.

For example, assume $V_k \approx V_n \approx 1.0$ per unit and that the angle differences are small so the sin terms can be ignored.

Then J_1 and J_4 are constant matrices whose elements are the negative of the imaginary components of Y_{bus}.

As such, J_1 and J_4 do not have to be recalculated during successive iterations.

The above simplifications can result in rapid power flow solutions for most systems.

While the fast decoupled power flow usually takes more iterations to converge, it is usually significantly faster than the Newton-Raphson algorithm since the Jacobian does not need to be recomputed each iteration.

And since the mismatch equations themselves have not been modified, the solution obtained by the fast decoupled algorithm is the same as that found with the Newton-Raphson algorithm.

However, in some situations in which only an approximate power flow solution is needed, the fast decoupled approach can be used with a fixed number of iterations (typically one) to give an extremely fast, albeit approximate solution.

The “DC” Power Flow

The power flow problem can be further simplified by extending the fast decoupled power flow to completely neglect the Q-V equations by assuming that the voltage magnitudes are constant at 1.0 per unit.

With these simplifications, the power flow on the line from bus j to bus k with reactance X_{jk} becomes:

$$P_{jk} = \frac{\delta_j - \delta_k}{X_{jk}}$$

And the real power balance equations reduce to a completely linear problem:

$$-\mathbf{B}\partial = \mathbf{P}$$

Where B is the imaginary component of the of Y_{bus} calculated neglecting line resistance and excepting the slack bus row and column and P is the vector of real power injections (with generation assumed positive).

Because this is a linear equation with a form similar to that found in solving dc resistive circuits, this technique is referred to as the dc power flow.

However, in contrast to the previous power flow algorithms, the dc power flow only gives an approximate solution with the degree of approximation system dependent.

Nevertheless, with the advent of power system restructuring, the dc power flow has become a commonly used analysis technique.

Control Of Power Flow

The following means are used to control system power flows:

1. **Prime mover and excitation control of generators**
2. **Switching of shunt capacitor banks, shunt reactors, and static var systems**
3. **Control of tap-changing and regulating transformers**

A simple model of a generator operating under balanced steady-state conditions is the Thévenin equivalent shown in Figure below.

V_t is the generator terminal voltage, E_g is the excitation voltage, δ is the power angle, and X_g is the positive-sequence synchronous reactance.

Generator Thevenin equivalent:

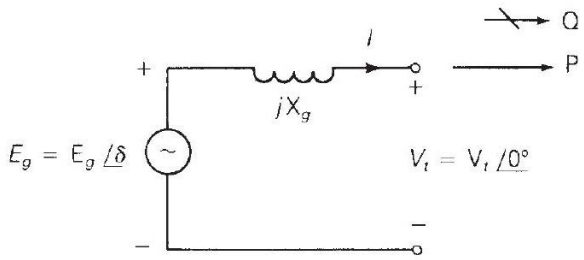


FIGURE 6.7

Generator Thévenin equivalent

From the figure, the generator current is:

$$I = \frac{\mathbf{E}_g e^{j\delta} - \mathbf{V}_t}{j\mathbf{X}_g}$$

and the complex power delivered by the generator is:

$$S = \mathbf{P} + j\mathbf{Q} = V_t I^* = \mathbf{V}_t \left(\frac{\mathbf{E}_g e^{-j\delta} - \mathbf{V}_t}{-j\mathbf{X}_g} \right)$$

$$= \frac{\mathbf{V}_t \mathbf{E}_g (j \cos\delta + \sin\delta) - j\,\mathbf{V}_t^2}{\mathbf{X}_g}$$

The real and reactive powers delivered are then

$$\mathbf{P} = \mathbf{Re}\, S = \frac{\mathbf{V}_t \mathbf{E}_g}{\mathbf{X}_g} \sin \delta$$

$$\mathbf{Q} = \mathbf{Im}\, S = \frac{\mathbf{V}_t}{\mathbf{X}_g} \left(\mathbf{E}_g \cos \delta - \mathbf{V}_t \right)$$

Equation above shows that the real power P increases when the power angle δ increases.

From an operational standpoint, when the prime mover increases the power input to the generator while the excitation voltage is held constant, the rotor speed increases.

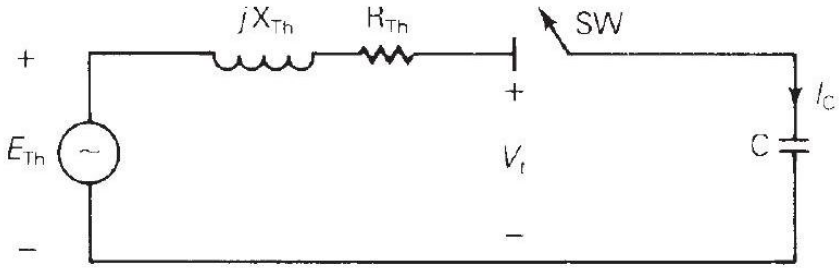
As the rotor speed increases, the power angle δ also increases, causing an increase in generator real power output P.

There is also a decrease in reactive power output Q.

However, when δ is less than 15°, the increase in P is much larger than the decrease in Q. From the power flow standpoint, an increase in prime-move power corresponds to an increase in P at the constant-voltage bus to which the generator is connected.

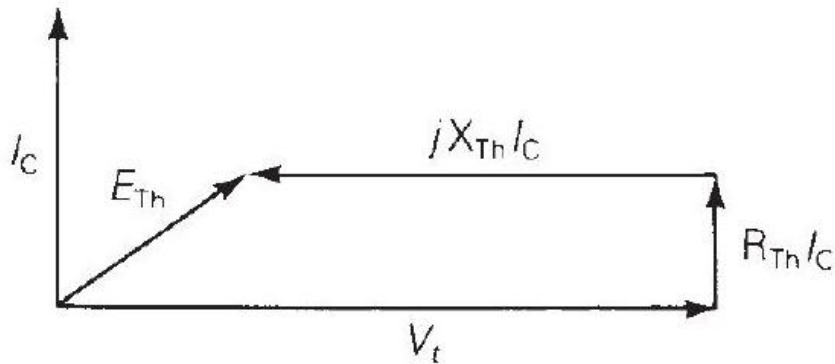
The power flow program computes the increase in δ along with the small change in Q.

Effect of adding a shunt capacitor bank to a power system bus:



(a) Equivalent circuit

Effect of adding a shunt capacitor bank to a power system bus:



(b) Phasor diagram with switch SW closed

Equation $Q = \mathrm{Im}\, S = \frac{V_t}{X_g}\left(E_g \cos\delta - V_t\right)$ **shows that reactive power output Q increases when the excitation voltage** E_g **increases.**

From the operational standpoint, when the generator exciter output increases while holding the prime-mover power constant, the rotor current increases.

As the rotor current increases, the excitation voltage E_g **also increases, causing an increase in generator reactive power output Q.**

There is also a small decrease in δ required to hold P constant in $P = \mathrm{Re}\, S = \frac{V_t E_g}{X_g} \sin \delta$.

From the power flow standpoint, an increase in generator excitation corresponds to an increase in voltage magnitude at the constant-voltage bus to which the generator is connected.

The power flow program computes the increase in reactive power Q supplied by the generator along with the small change in δ.

Figures above shows the effect of adding a shunt capacitor bank to a power system bus.

The system is modeled by its Thévenin equivalent. Before the capacitor bank is connected, the switch SW is open and the bus voltage equals E_{Th}.

After the bank is connected, SW is closed, and the capacitor current I_C leads the bus voltage V_t by 90°.

The phasor diagram shows that V_t is larger than E_{Th} when SW is closed.

From the power flow standpoint, the addition of a shunt capacitor bank to a load bus corresponds to the addition of a negative reactive load, since a capacitor absorbs negative reactive power.

The power flow program computes the increase in bus voltage magnitude along with the small change in δ.

Similarly, the addition of a shunt reactor corresponds to the addition of a positive reactive load, wherein the power flow program computes the decrease in voltage magnitude.

Tap-changing and voltage-magnitude-regulating transformers are used to control bus voltages as well as reactive power flows on lines to which they are connected.

Similarly, phase-angle regulating transformers are used to control bus angles as well as real power flows on lines to which they are connected.

Both tap-changing and regulating transformers are modeled by a transformer with an off-nominal turns ratio c (See the Tap Change Model).

From the power flow standpoint, a change in tap setting or voltage regulation corresponds to a change in c.

The power flow program computes the changes in Y_{bus}, bus voltage magnitudes and angles, and branch flows.

Besides the above controls, the power flow program can be used to investigate the effect of switching in or out lines, transformers, loads, and generators.

Proposed system changes to meet future load growth, including new transmission, new transformers, and new generation can also be investigated.

Power flow design studies are normally conducted by trial and error.

Using engineering judgment, adjustments in generation levels and controls are made until the desired equipment loadings and voltage profile are obtained.