ELE B7 Power Systems Engineering

Newton-Raphson Method

Newton-Raphson Algorithm

- The second major power flow solution method is the Newton-Raphson algorithm
- Key idea behind Newton-Raphson is to use sequential linearization

General form of problem: Find an x such that

$$f(\hat{x}) = 0$$

Newton-Raphson Method (scalar)

1. For each guess of $\hat{x}$, $x^{(v)}$, define

$$\Delta x^{(v)} = \hat{x} - x^{(v)}$$

2. Represent $f(\hat{x})$ by a Taylor series about $f(x)$

$$f(\hat{x}) = f(x^{(v)}) + \frac{df(x^{(v)})}{dx}\Delta x^{(v)} + $$

$$+ \frac{d^2 f(x^{(v)})}{dx^2}\left(\Delta x^{(v)}\right)^2 + \text{higher order terms}$$

Newton-Raphson Method, cont'd

3. Approximate $f(\hat{x})$ by neglecting all terms except the first two

$$f(\hat{x}) = 0 \approx f(x^{(v)}) + \frac{df(x^{(v)})}{dx} \Delta x^{(v)}$$

4. Use this linear approximation to solve for $\Delta x^{(v)}$

$$\Delta x^{(v)} = -\left[\frac{df(x^{(v)})}{dx}\right]^{-1} f(x^{(v)})$$

5. Solve for a new estimate of x̂

$$x^{(v+1)} = x^{(v)} + \Delta x^{(v)}$$

Newton-Raphson Example

Use Newton-Raphson to solve $f(x) = x^2 - 2 = 0$

The equation we must iteratively solve is

$$\Delta x^{(v)} = -\left[\frac{df(x^{(v)})}{dx}\right]^{-1} f(x^{(v)})$$

$$\Delta x^{(v)} = -\left[\frac{1}{2x^{(v)}}\right]((x^{(v)})^2 - 2)$$

$$x^{(v+1)} = x^{(v)} + \Delta x^{(v)}$$

$$x^{(v+1)} = x^{(v)} - \left[\frac{1}{2x^{(v)}}\right]((x^{(v)})^2 - 2)$$

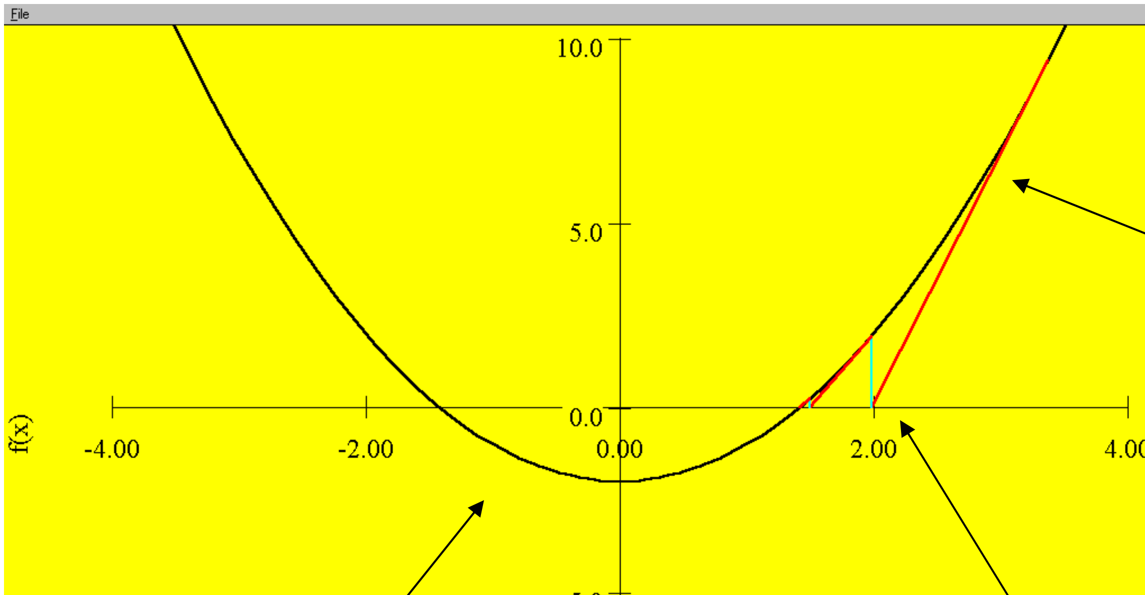
Newton-Raphson Example, cont'd

$$x^{(v+1)} = x^{(v)} - \left[\frac{1}{2x^{(v)}}\right]((x^{(v)})^2 - 2)$$

Guess $x^{(0)} = 1$. Iteratively solving we get

v	$x^{(v)}$	$f(x^{(v)})$	$\Delta x^{(v)}$
0	1	-1	0.5
1	1.5	0.25	-0.08333
2	1.41667	6.953×10^{-3}	-2.454×10^{-3}
3	1.41422	6.024×10^{-6}	

Sequential Linear Approximations



At each iteration the N-R method uses a linear approximation to determine the next value for x

Function is $f(x) = x^2 - 2 = 0$. Solutions are points where f(x) intersects f(x) = 0 axis

Newton-Raphson Comments

- When close to the solution the error decreases quite quickly -- method has quadratic convergence
- $f(x^{(v)})$ is known as the mismatch, which we would like to drive to zero
- Stopping criteria is when $|f(x^{(v)})| < \varepsilon$
- Results are dependent upon the initial guess. What if we had guessed $x^{(0)} = 0$, or $x^{(0)} = -1$?
- A solution's region of attraction (ROA) is the set of initial guesses that converge to the particular solution. The ROA is often hard to determine

Multi-Variable Newton-Raphson

Next we generalize to the case where $\mathbf{x}$ is an n-dimension vector, and $\mathbf{f}(\mathbf{x})$ is an n-dimension function

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad \mathbf{f}(\mathbf{x}) = \begin{bmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ \vdots \\ f_n(\mathbf{x}) \end{bmatrix}$$

Again define the solution $\hat{\mathbf{x}}$ so $\mathbf{f}(\hat{\mathbf{x}}) = 0$ and

$$\Delta\mathbf{x} = \hat{\mathbf{x}} - \mathbf{x}$$

Multi-Variable Case, cont'd

The Taylor series expansion is written for each $f_i(\mathbf{x})$

$$f_1(\hat{\mathbf{x}}) = f_1(\mathbf{x}) + \frac{\partial f_1(\mathbf{x})}{\partial x_1}\Delta x_1 + \frac{\partial f_1(\mathbf{x})}{\partial x_2}\Delta x_2 + \ldots$$

$$\frac{\partial f_1(\mathbf{x})}{\partial x_n}\Delta x_n + \text{higher order terms}$$

$$\vdots$$

$$f_n(\hat{\mathbf{x}}) = f_n(\mathbf{x}) + \frac{\partial f_n(\mathbf{x})}{\partial x_1}\Delta x_1 + \frac{\partial f_n(\mathbf{x})}{\partial x_2}\Delta x_2 + \ldots$$

$$\frac{\partial f_n(\mathbf{x})}{\partial x_n}\Delta x_n + \text{higher order terms}$$

Multi-Variable Case, cont'd

This can be written more compactly in matrix form

$$\mathbf{f}(\hat{\mathbf{x}}) = \begin{bmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ \vdots \\ f_n(\mathbf{x}) \end{bmatrix} + \begin{bmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \frac{\partial f_1(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_1(\mathbf{x})}{\partial x_n} \\ \frac{\partial f_2(\mathbf{x})}{\partial x_1} & \frac{\partial f_2(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_2(\mathbf{x})}{\partial x_n} \\ \vdots & \ddots & \ddots & \vdots \\ \frac{\partial f_n(\mathbf{x})}{\partial x_1} & \frac{\partial f_n(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_n(\mathbf{x})}{\partial x_n} \end{bmatrix} \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \\ \vdots \\ \Delta x_n \end{bmatrix}$$

$$+ \quad \text{higher order terms}$$

Jacobian Matrix

The n by n matrix of partial derivatives is known as the Jacobian matrix, $\mathbf{J}(\mathbf{x})$

$$\mathbf{J}(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \frac{\partial f_1(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_1(\mathbf{x})}{\partial x_n} \\ \frac{\partial f_2(\mathbf{x})}{\partial x_1} & \frac{\partial f_2(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_2(\mathbf{x})}{\partial x_n} \\ \vdots & \ddots & \ddots & \vdots \\ \frac{\partial f_n(\mathbf{x})}{\partial x_1} & \frac{\partial f_n(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_n(\mathbf{x})}{\partial x_n} \end{bmatrix}$$

Multi-Variable N-R Procedure

Derivation of N-R method is similar to the scalar case

$$\mathbf{f}(\hat{\mathbf{x}}) = \mathbf{f}(\mathbf{x}) + \mathbf{J}(\mathbf{x})\Delta\mathbf{x} + \text{higher order terms}$$

$$\mathbf{f}(\hat{\mathbf{x}}) = 0 \approx \mathbf{f}(\mathbf{x}) + \mathbf{J}(\mathbf{x})\Delta\mathbf{x}$$

$$\Delta\mathbf{x} \approx -\mathbf{J}(\mathbf{x})^{-1}\mathbf{f}(\mathbf{x})$$

$$\mathbf{x}^{(v+1)} = \mathbf{x}^{(v)} + \Delta\mathbf{x}^{(v)}$$

$$\mathbf{x}^{(v+1)} = \mathbf{x}^{(v)} - \mathbf{J}(\mathbf{x}^{(v)})^{-1}\mathbf{f}(\mathbf{x}^{(v)})$$

Iterate until $\left\| \mathbf{f}(\mathbf{x}^{(v)}) \right\| < \varepsilon$

Multi-Variable Example

Solve for $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ such that $\mathbf{f}(\mathbf{x}) = 0$ where

$$f_1(\mathbf{x}) = 2x_1^2 + x_2^2 - 8 = 0$$

$$f_2(\mathbf{x}) = x_1^2 - x_2^2 + x_1 x_2 - 4 = 0$$

First symbolically determine the Jacobian

$$\mathbf{J}(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \frac{\partial f_1(\mathbf{x})}{\partial x_2} \\ \frac{\partial f_2(\mathbf{x})}{\partial x_1} & \frac{\partial f_2(\mathbf{x})}{\partial x_2} \end{bmatrix}$$

Multi-variable Example, cont'd

$$\mathbf{J}(\mathbf{x}) = \begin{bmatrix} 4x_1 & 2x_2 \\ 2x_1 + x_2 & x_1 - 2x_2 \end{bmatrix}$$

Then

$$\begin{bmatrix} \Delta x_1 \\ \Delta x_2 \end{bmatrix} = -\begin{bmatrix} 4x_1 & 2x_2 \\ 2x_1 + x_2 & x_1 - 2x_2 \end{bmatrix}^{-1} \begin{bmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \end{bmatrix}$$

Arbitrarily guess $\mathbf{x}^{(0)} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

$$\mathbf{x}^{(1)} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 4 & 2 \\ 3 & -1 \end{bmatrix}^{-1} \begin{bmatrix} -5 \\ -3 \end{bmatrix} = \begin{bmatrix} 2.1 \\ 1.3 \end{bmatrix}$$

Multi-variable Example, cont'd

$$\mathbf{x}^{(2)} = \begin{bmatrix} 2.1 \\ 1.3 \end{bmatrix} - \begin{bmatrix} 8.40 & 2.60 \\ 5.50 & -0.50 \end{bmatrix}^{-1} \begin{bmatrix} 2.51 \\ 1.45 \end{bmatrix} = \begin{bmatrix} 1.8284 \\ 1.2122 \end{bmatrix}$$

Each iteration we check $\|\mathbf{f}(\mathbf{x})\|$ to see if it is below our specified tolerance ε

$$\mathbf{f}(\mathbf{x}^{(2)}) = \begin{bmatrix} 0.1556 \\ 0.0900 \end{bmatrix}$$

If $\varepsilon = 0.2$ then we would be done. Otherwise we'd continue iterating.

NR Application to Power Flow

We first need to rewrite complex power equations as equations with real coefficients

$$S_i = V_i I_i^* = V_i \left(\sum_{k=1}^{n} Y_{ik} V_k \right)^* = V_i \sum_{k=1}^{n} Y_{ik}^* V_k^*$$

These can be derived by defining

$$Y_{ik} = G_{ik} + jB_{ik}$$

$$V_i = |V_i| e^{j\theta_i} = |V_i| \angle \theta_i$$

$$\theta_{ik} = \theta_i - \theta_k$$

Recall $e^{j\theta} = \cos\theta + j\sin\theta$

Real Power Balance Equations

$$S_i = P_i + jQ_i = V_i \sum_{k=1}^{n} Y_{ik}^* V_k^* = \sum_{k=1}^{n} |V_i||V_k| e^{j\theta_{ik}} (G_{ik} - jB_{ik})$$

$$= \sum_{k=1}^{n} |V_i||V_k| (\cos\theta_{ik} + j\sin\theta_{ik})(G_{ik} - jB_{ik})$$

Resolving into the real and imaginary parts

$$P_i = \sum_{k=1}^{n} |V_i||V_k| (G_{ik}\cos\theta_{ik} + B_{ik}\sin\theta_{ik}) = P_{Gi} - P_{Di}$$

$$Q_i = \sum_{k=1}^{n} |V_i||V_k| (G_{ik}\sin\theta_{ik} - B_{ik}\cos\theta_{ik}) = Q_{Gi} - Q_{Di}$$

Newton-Raphson Power Flow

In the Newton-Raphson power flow we use Newton's method to determine the voltage magnitude and angle at each bus in the power system.

We need to solve the power balance equations

$$\mathrm{P_i} = \sum_{k=1}^{n} |V_i||V_k|(G_{ik}\cos\theta_{ik} + B_{ik}\sin\theta_{ik}) = P_{Gi} - P_{Di}$$

$$\mathrm{Q_i} = \sum_{k=1}^{n} |V_i||V_k|(G_{ik}\sin\theta_{ik} - B_{ik}\cos\theta_{ik}) = Q_{Gi} - Q_{Di}$$

Power Flow Variables

Assume the slack bus is the first bus (with a fixed voltage angle/magnitude). We then need to determine the voltage angle/magnitude at the other buses.

$$\mathbf{X} = \begin{bmatrix} \theta_2 \\ \vdots \\ \theta_n \\ |V_2| \\ \vdots \\ |V_n| \end{bmatrix}, \quad \mathbf{f}(\mathbf{x}) = \begin{bmatrix} \Delta\mathbf{P}(\mathbf{x}) \\ \Delta\mathbf{Q}(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} P_2(\mathbf{x}) - P_{G2} + P_{D2} \\ \vdots \\ P_n(\mathbf{x}) - P_{Gn} + P_{Dn} \\ Q_2(\mathbf{x}) - Q_{G2} + Q_{D2} \\ \vdots \\ Q_n(\mathbf{x}) - Q_{Gn} + Q_{Dn} \end{bmatrix}$$

N-R Power Flow Solution

The power flow is solved using the same procedure discussed last time:

Set $v = 0$; make an initial guess of $\mathbf{x}$, $\mathbf{x}^{(v)}$

While $\left\| \mathbf{f}(\mathbf{x}^{(v)}) \right\| > \varepsilon$ Do

$$\mathbf{x}^{(v+1)} = \mathbf{x}^{(v)} - \mathbf{J}(\mathbf{x}^{(v)})^{-1}\mathbf{f}(\mathbf{x}^{(v)})$$

$$v = v + 1$$

End While

Power Flow Jacobian Matrix

The most difficult part of the algorithm is determining and inverting the n by n Jacobian matrix, $\mathbf{J}(\mathbf{x})$

$$\mathbf{J}(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \frac{\partial f_1(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_1(\mathbf{x})}{\partial x_n} \\ \frac{\partial f_2(\mathbf{x})}{\partial x_1} & \frac{\partial f_2(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_2(\mathbf{x})}{\partial x_n} \\ \vdots & \ddots & \ddots & \vdots \\ \frac{\partial f_n(\mathbf{x})}{\partial x_1} & \frac{\partial f_n(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_n(\mathbf{x})}{\partial x_n} \end{bmatrix}$$

Power Flow Jacobian Matrix, cont'd

Jacobian elements are calculated by differentiating each function, $\mathrm{f_i}(\mathbf{x})$, with respect to each variable.

For example, if $\mathrm{f_i}(\mathbf{x})$ is the bus i real power equation

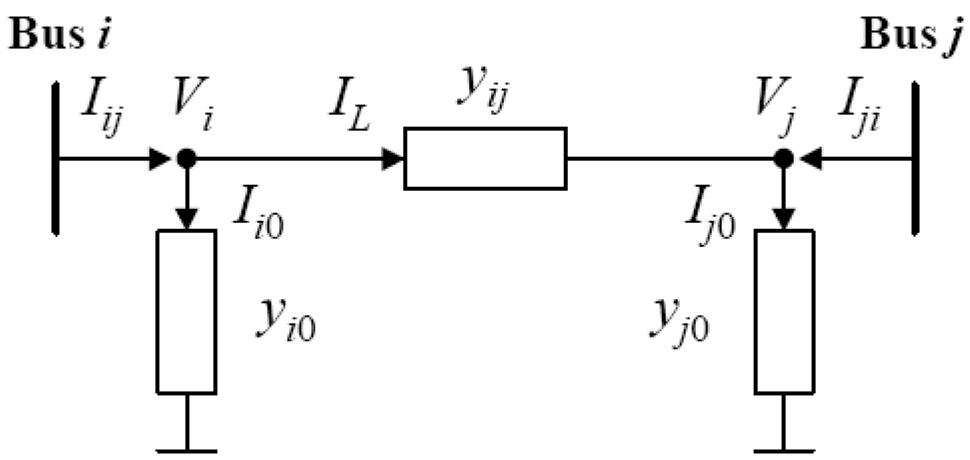
$$\mathrm{f_i}(x) = \sum_{k=1}^{n} |V_i||V_k|(G_{ik}\cos\theta_{ik} + B_{ik}\sin\theta_{ik}) - P_{Gi} + P_{Di}$$

$$\frac{\partial \mathrm{f_i}(x)}{\partial \theta_i} = \sum_{\substack{k=1 \\ k\neq i}}^{n} |V_i||V_k|(-G_{ik}\sin\theta_{ik} + B_{ik}\cos\theta_{ik})$$

$$\frac{\partial \mathrm{f_i}(x)}{\partial \theta_j} = |V_i||V_j|(G_{ik}\sin\theta_{ik} - B_{ik}\cos\theta_{ik}) \quad (j \neq i)$$

Line Flows and Losses

- **After solving for bus voltages and angles, power flows and losses on the network branches are calculated**
  - Transmission lines and transformers are network branches
  - The direction of positive current flow are defined as follows for a branch element (demonstrated on a medium length line)
  - Power flow is defined for each end of the branch
    - Example: the power leaving bus i and flowing to bus j



Line Flows and Losses

- **current and power flows:**

$i \rightarrow j$

$$I_{ij} = I_L + I_{i0} = y_{ij}\left(V_i - V_j\right) + y_{i0}\,V_i$$

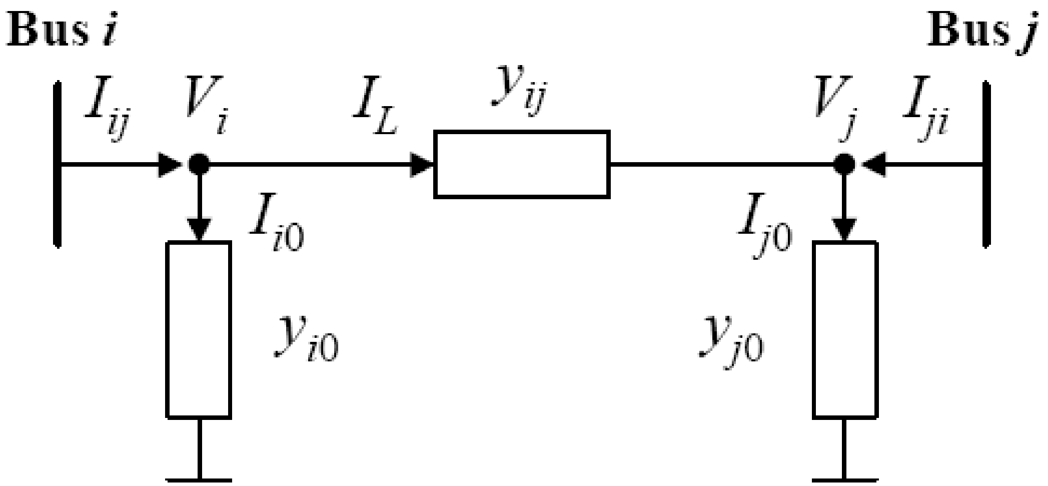
$$S_{ij} = V_i\,I_{ij}^* = V_i^2\left(y_{ij} + y_{i0}\right)^* - V_i\,y_{ij}^*\,V_j^*$$

$j \rightarrow i$

$$I_{ji} = -I_L + I_{j0} = y_{ij}\left(V_j - V_i\right) + y_{j0}\,V_j$$

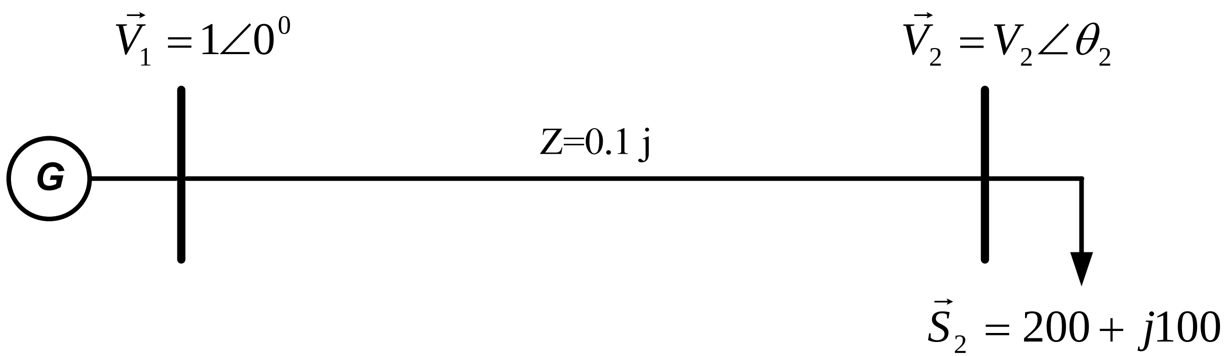
$$S_{ji} = V_j\,I_{ji}^* = V_j^2\left(y_{ij} + y_{j0}\right)^* - V_j\,y_{ij}^*\,V_i^*$$

- **power loss:** $S_{Loss\,ij} = S_{ij} + S_{ji}$



Two Bus Newton-Raphson Example

For the two bus power system shown below, use the Newton-Raphson power flow to determine the voltage magnitude and angle at bus two. Assume that bus one is the slack and $S_{Base} = 100$ MVA.



$$\mathbf{x} = \begin{bmatrix} \theta_2 \\ |V_2| \end{bmatrix} \qquad \mathbf{Y}_{bus} = \begin{bmatrix} -j10 & j10 \\ j10 & -j10 \end{bmatrix}$$

Two Bus Example, cont'd

General power balance equations

$$\mathrm{P_i} = \sum_{k=1}^{n} |V_i||V_k|(G_{ik}\cos\theta_{ik} + B_{ik}\sin\theta_{ik}) = P_{Gi} - P_{Di}$$

$$\mathrm{Q_i} = \sum_{k=1}^{n} |V_i||V_k|(G_{ik}\sin\theta_{ik} - B_{ik}\cos\theta_{ik}) = Q_{Gi} - Q_{Di}$$

Bus two power balance equations

$$\mathrm{P_2} = |V_2||V_1|(10\sin\theta_2) + 2.0 = 0$$

$$Q_2 = |V_2||V_1|(-10\cos\theta_2) + |V_2|^2(10) + 1.0 = 0$$

Two Bus Example, cont'd

$$P_2(\mathbf{x}) = |V_2|(10\sin\theta_2) + 2.0 = 0$$

$$Q_2(\mathbf{x}) = |V_2|(-10\cos\theta_2) + |V_2|^2(10) + 1.0 = 0$$

Now calculate the power flow Jacobian

$$J(\mathbf{x}) = \begin{bmatrix} \dfrac{\partial P_2(\mathbf{x})}{\partial \theta_2} & \dfrac{\partial P_2(\mathbf{x})}{\partial |V|_2} \\ \dfrac{\partial Q_2(\mathbf{x})}{\partial \theta_2} & \dfrac{\partial Q_2(\mathbf{x})}{\partial |V|_2} \end{bmatrix}$$

$$= \begin{bmatrix} 10|V_2|\cos\theta_2 & 10\sin\theta_2 \\ 10|V_2|\sin\theta_2 & -10\cos\theta_2 + 20|V_2| \end{bmatrix}$$

Two Bus Example, First Iteration

Set $v = 0$, guess $\mathbf{x}^{(0)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

Calculate

$$f(\mathbf{x}^{(0)}) = \begin{bmatrix} |V_2|(10\sin\theta_2) + 2.0 \\ |V_2|(-10\cos\theta_2) + |V_2|^2(10) + 1.0 \end{bmatrix} = \begin{bmatrix} 2.0 \\ 1.0 \end{bmatrix}$$

$$\mathbf{J}(\mathbf{x}^{(0)}) = \begin{bmatrix} 10|V_2|\cos\theta_2 & 10\sin\theta_2 \\ 10|V_2|\sin\theta_2 & -10\cos\theta_2 + 20|V_2| \end{bmatrix} = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}$$

Solve $\mathbf{x}^{(1)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} - \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}^{-1} \begin{bmatrix} 2.0 \\ 1.0 \end{bmatrix} = \begin{bmatrix} -0.2 \\ 0.9 \end{bmatrix}$

Two Bus Example, Next Iterations

$$f(\mathbf{x}^{(1)}) = \begin{bmatrix} 0.9(10\sin(-0.2)) + 2.0 \\ 0.9(-10\cos(-0.2)) + 0.9^2 \times 10 + 1.0 \end{bmatrix} = \begin{bmatrix} 0.212 \\ 0.279 \end{bmatrix}$$

$$\mathbf{J}(\mathbf{x}^{(1)}) = \begin{bmatrix} 8.82 & -1.986 \\ -1.788 & 8.199 \end{bmatrix}$$

$$\mathbf{x}^{(2)} = \begin{bmatrix} -0.2 \\ 0.9 \end{bmatrix} - \begin{bmatrix} 8.82 & -1.986 \\ -1.788 & 8.199 \end{bmatrix}^{-1} \begin{bmatrix} 0.212 \\ 0.279 \end{bmatrix} = \begin{bmatrix} -0.233 \\ 0.8586 \end{bmatrix}$$

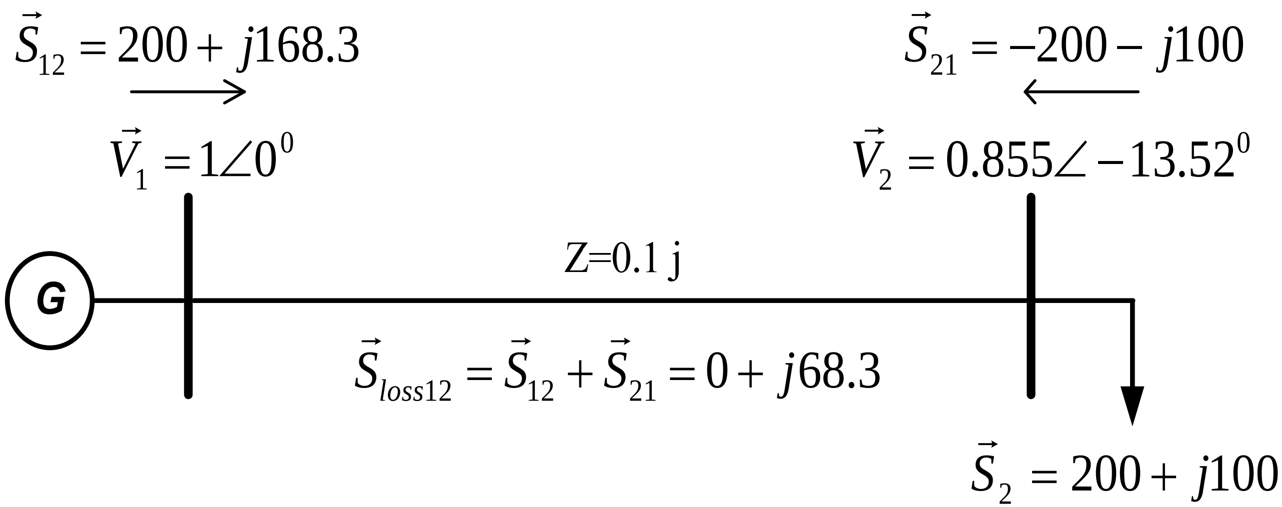
$$f(\mathbf{x}^{(2)}) = \begin{bmatrix} 0.0145 \\ 0.0190 \end{bmatrix} \qquad \mathbf{x}^{(3)} = \begin{bmatrix} -0.236 \\ 0.8554 \end{bmatrix}$$

$$f(\mathbf{x}^{(3)}) = \begin{bmatrix} 0.0000906 \\ 0.0001175 \end{bmatrix}$$

Done! $V_2 = 0.8554\angle -13.52°$

Two Bus Solved Values

Once the voltage angle and magnitude at bus 2 are known we can calculate all the other system values, such as the line flows and the generator reactive power output



PV Buses

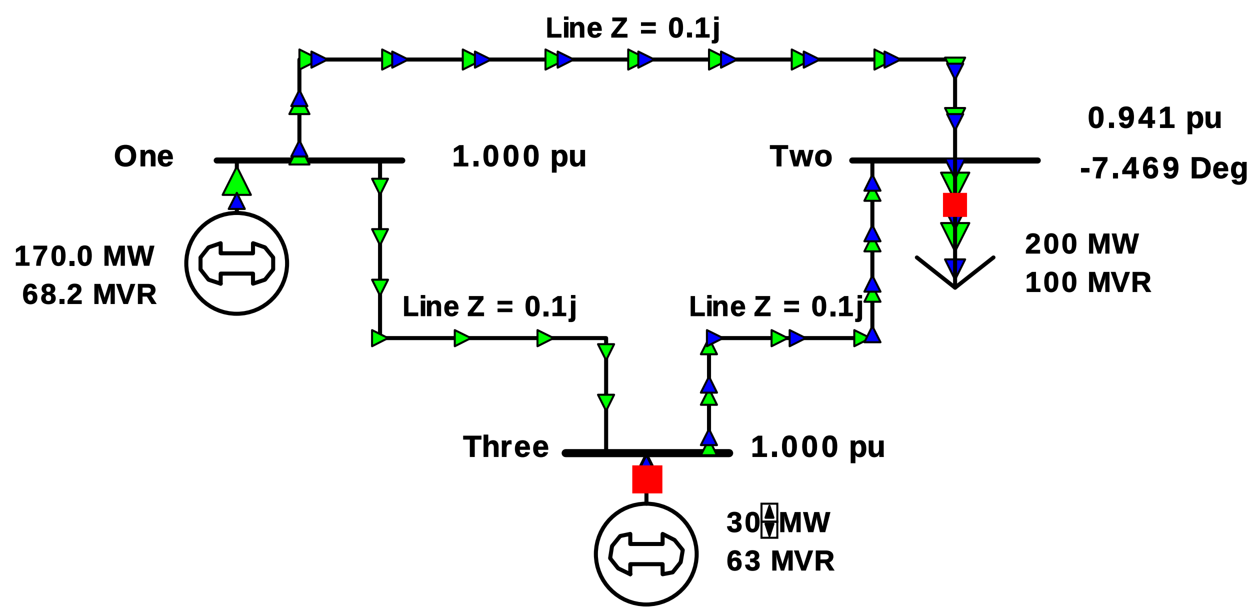
- Since the voltage magnitude at PV buses is fixed there is no need to explicitly include these voltages in **x** or write the reactive power balance equations
  - the reactive power output of the generator varies to maintain the fixed terminal voltage (within limits)
  - optionally these variations/equations can be included by just writing the explicit voltage constraint for the generator bus

$$|V_i| - V_{i\ setpoint} = 0$$

Three Bus PV Case Example

For this three bus case we have

$$\mathbf{x} = \begin{bmatrix} \theta_2 \\ \theta_3 \\ |V_2| \end{bmatrix} \qquad \mathbf{f}(\mathbf{x}) = \begin{bmatrix} P_2(\mathbf{x}) - P_{G2} + P_{D2} \\ P_3(\mathbf{x}) - P_{G3} + P_{D3} \\ Q_2(\mathbf{x}) + Q_{D2} \end{bmatrix} = 0$$



Solving Large Power Systems

- The most difficult computational task is inverting the Jacobian matrix
  - inverting a full matrix is an order n^3 operation, meaning the amount of computation increases with the cube of the size size
  - this amount of computation can be decreased substantially by recognizing that since the Y_{bus} is a sparse matrix, the Jacobian is also a sparse matrix
  - using sparse matrix methods results in a computational order of about $n^{1.5}$.
  - this is a substantial savings when solving systems with tens of thousands of buses

Newton-Raphson Power Flow

- Advantages
  - fast convergence as long as initial guess is close to solution
  - large region of convergence
- Disadvantages
  - each iteration takes much longer than a Gauss-Seidel iteration
  - more complicated to code, particularly when implementing sparse matrix algorithms
- Newton-Raphson algorithm is very common in power flow analysis