THE POWER FLOW PROBLEM

CHRIS MUTZEL

DECEMBER 7TH, 2011

EE193 – POWER SYSTEMS ANALYSIS – A. STANKOVIC

PRESENTATION OUTLINE

1) Background
2) Problem Formulation
3) Solution: Newton-Raphson
4) Approach and Program Flow
    1) Finding Ybus, Data Format, Current Injections
    2) Determining the Jacobian
5) Results
6) Code Excerpt

BACKGROUND

- **One of most common tools in power systems analysis**
- **Used in system control and planning to minimize costs and increase stability.**
- **Analysis is done for balanced, single phase network**

Relevant Details:

- Thousands of Nodes (large amounts of data)
- Nodes have only a few connections (lots of zeros)
- Each node has 4 state variables (P,Q,V,θ)
- 3 Types: PV, PQ, slack

PROBLEM FORMULATION

Each bus has a current injection:

$$I_i = \sum_{j=1}^{n} Y_{ij} V_j = \sum_{j=1}^{n} |Y_{ij}||V_j| \angle \theta_{ij} + \delta_j$$

$$P_i^{[k]} = \sum_{j=1}^{n} \left|V_i^{[k]}\right| \left|V_j^{[k]}\right| \left|Y_{ij}\right| \cos\left(\theta_{ij} - \delta_i^{[k]} + \delta_j^{[k]}\right)$$

$$Q_i^{[k]} = -\sum_{j=1}^{n} \left|V_i^{[k]}\right| \left|V_j^{[k]}\right| \left|Y_{ij}\right| \sin\left(\theta_{ij} - \delta_i^{[k]} + \delta_j^{[k]}\right)$$

From these we get real and reactive power flow equations.

Then we can make the problem of the form f(x) = b

$$x^{[k]} = \begin{bmatrix} \delta^{[k]} \\ V^{[k]} \end{bmatrix} \qquad f\left(x^{[k]}\right) = \begin{bmatrix} P_{inj}\left(x^{[k]}\right) \\ Q_{inj}\left(x^{[k]}\right) \end{bmatrix}$$

SOLUTION: NEWTON-RAPHSON

-Iterative Method for Solution of non-linear equations

-An initial guess is made:

$$c = f\left(x_{solution}\right) \qquad x^{[0]} = \text{initial estimate of } x_{solution}$$

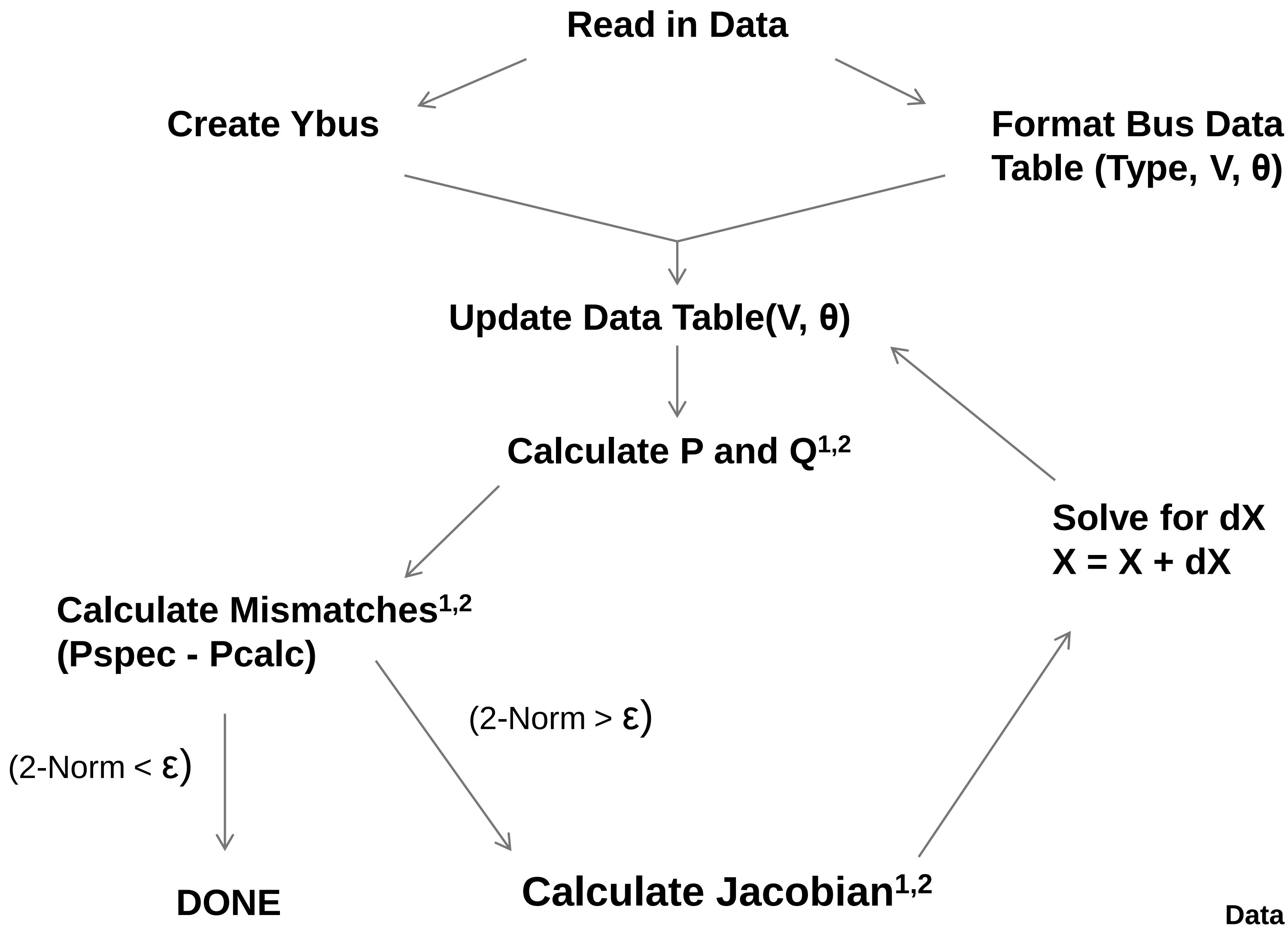
-From this initial guess, we can use the error to calculate a new guess:

$$x^{[k+1]} = x^{[k]} + \frac{c - f\left(x^{[k]}\right)}{\left(df\left(x^{[k]}\right)/dx\right)}$$

Bottom term is the Jacobian

$$df(x)/dx \Rightarrow \begin{bmatrix} \Delta P \\ \Delta Q \end{bmatrix} = \begin{bmatrix} \partial P/\partial\delta & \partial P/\partial|V| \\ \partial Q/\partial\delta & \partial Q/\partial|V| \end{bmatrix} \begin{bmatrix} \Delta\delta \\ \Delta|V| \end{bmatrix}$$

APPROACH AND PROGRAM FLOW



Data Sources:
1 – Data Table
2 – Ybus

YBUS, CURRENT INJECTIONS, DATA STORAGE

<u>CreateYbus.m</u> (from previous assignment)

Data is stored in sparse matrix, expanded at runtime

Row	Col	Value
1	1	2.6-3j
1	5	1.3
2	2	7.7j
…	…	…

<u>busData Array</u> (update each iter.)

-bus type,

-P, Q

-V, θ

Mismatches, Jacobian calculations

use bus type to evaluate only

correct elements

```
EDU>> busData

busData =
```

Type	P	Q	V	θ
3.0000	2.9750	0.1420	1.0000	0
2.0000	0.1830	0.3730	1.0431	-0.1348
0	-0.0240	-0.0120	0.9976	-0.1746
0	-0.0760	-0.0160	0.9985	-0.2144
2.0000	-0.9420	0.2100	1.0099	-0.2982
0	0	0	1.0011	-0.2505
0	-0.2280	-0.1090	0.9969	-0.2793
2.0000	-0.3000	0.1000	1.0099	-0.2667
0	0	0	1.0069	-0.3296
0	-0.0580	-0.0200	0.9727	-0.3727
2.0000	0	0.2400	1.0821	-0.3296
0	-0.1120	-0.0750	1.0094	-0.3555
2.0000	0	0.2400	1.0709	-0.3555
0	-0.0620	-0.0160	0.9858	-0.3757
0	-0.0820	-0.0250	0.9722	-0.3773
0	-0.0350	-0.0180	0.9854	-0.3670
0	-0.0900	-0.0580	0.9714	-0.3755
0	-0.0320	-0.0090	0.9594	-0.3895
0	-0.0950	-0.0340	0.9551	-0.3930

<u>FindInjections.m</u>

References data table and Ybus.

DETERMINING JACOBIAN

Jacobian is hard because of dimensions and bus data references. It must have same dimensions and order as X.

$$x^{[k]} = \begin{bmatrix} \delta^{[k]} \\ V^{[k]} \end{bmatrix}$$

First N-1 entries correspond to PQ and PV buses.

N-m-1 entries for PQ buses only

For simplicity, Jacobian is calculated for all buses (N=30).

Needed entries are then copied to a new Jacobian, which is used to solve power flow.

Size (old) =

2N x 2N

Throw away irrelevant data

Size (new) =

2N – M – 2
X
2N – m – 2

RESULTS

Solution for N = 30, (Ave. time = .00320 sec)

Iteration	Error (2-Norm)
1	1.403115
2	0.198388
3	0.002192
4	0.0000007

Solution for N = 30, (Ave. time = .00304 sec)
Freeze Jacobian after 1st iteration

Iteration	Error (2-Norm)
1	1.403115
2	0.019838
3	0.002192
4	.0000719

JACOBIAN CODE

```
for Jr=1:2
    for Jc=1:2
        %J11
        if (Jr == 1 && Jc == 1)
            for SJr=1:N

                for SJc=1:N

                    if (SJr == SJc) % diagonal elements
                        Jacob(SJr,SJr) = -Qcalc(SJc,1) - imag(Ybus(SJc,SJc))*abs(busData(SJc,4))^2;

                    else            % off-diagonal elements
                        Jacob(SJr, SJc) = abs(busData(SJr,4))*abs(busData(SJc,4))*(...
                            real(Ybus(SJr,SJc))*sin(busData(SJr,5)-busData(SJc,5))-...
                            imag(Ybus(SJr,SJc))*cos(busData(SJr,5)-busData(SJc,5)));
                    end
                end
            end
        end
        %J12
        if (Jr == 1 && Jc == 2)
            for SJr=1:N
                for SJc=1:N
                    if SJr == SJc % diagonal elements
                        Jacob(SJr, N + SJc) = Pcalc(SJc)/abs(busData(SJc,4)) +...
                            real(Ybus(SJc,SJc))*abs(busData(SJc,4));
                    else        % off-diagaonal elements
                        Jacob(SJr, N + SJc) = abs(busData(SJr,4))*(...
                            real(Ybus(SJr, SJc))*cos(busData(SJr,5)-busData(SJc, 5))+...
                            imag(Ybus(SJr, SJc))*sin(busData(SJr,5)-busData(SJc, 5)));
                    end
                end
            end
        end
        %J21
        if (Jr == 2 && Jc == 1)
            for SJr=1:N
                for SJc=1:N
                    if SJr == SJc % diagonal elements
                        Jacob(N + SJr, SJc) = Pcalc(SJc) - real(Ybus(SJc, SJc))*abs(busData(SJc, 4))^2;
                    else        % off-diagonal elements
                        Jacob(N + SJr, SJc) = -abs(busData(SJr,4))*abs(busData(SJc,4))*(...
                            real(Ybus(SJr, SJc))*cos(busData(SJr,5) - busData(SJc,5)) +...
                            imag(Ybus(SJr, SJc))*sin(busData(SJr,5) - busData(SJc,5)));
```

THANKS!