# LOAD FLOW ANALYSIS

Soheil Shirvani

810195416

# Contents

# Introduction

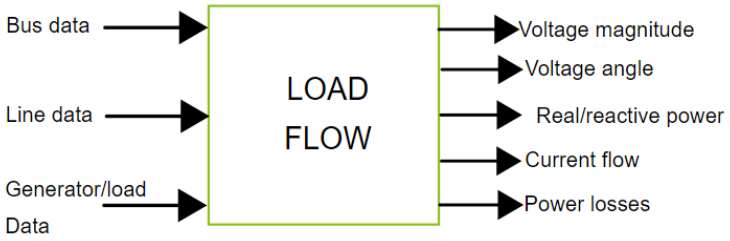
In this project we want to code load flow solvers to analysis 30 bus IEEE standard. To do so, we are going to implement three different ways including Gauss-Seidel, Newton-Raphson, and Decouple methods. 30 bus IEEE bus data and line flows are given for further use and methods are applied to the same data. For the same purpose, we first choose the method out of three then run that method on the data for a fixed number of iterations, then, losses, Y bus, and load flows between all the power lines will be printed. All codes are implemented in MATLAB. At the end of each method, we will see the results of that method along with the duration it took for it to complete.

# Load Flow

## Load flow objective

The objective of load flow calculations is to determine the steady-state operating characteristics of the power system for a given load and generator real power and voltage conditions. Once we have this information, we can calculate easily real and reactive power flow in all branches together with power losses.

## What are the inputs and outputs of load flow analysis?




Minimum input data for power flow analysis:

- Bus data (types of buses explained in bus types):
  - For PV buses:
    - Real power (generation and demand),
    - Reactive power (demand),
    - Voltage magnitude.
  - For PQ buses:
    - Real power (generation and demand),
    - Reactive power (generation and demand).
  - For slack bus:
    - Voltage magnitude (usually 1 per unit),
    - Voltage angle (specified to be zero),
    - Real power (demand),
    - Reactive power (demand).
- Line data:
  - Transmission lines:
    - Resistance,
    - Reactance,
    - Capacitance (can be negligible).
  - Transformers:
    - Winding resistances on low and high voltage side,
    - Leakage reactance on low and high voltage side,
    - Magnetization reactance,
    - Iron loss admittance.

Power flow analysis provides following output data for each node/branch:

- Voltage magnitude,
- Voltage angle,
- Real and reactive power,
- Power losses.

## Bus types

Depending, upon which two variables you specify, the buses (nodes) can be categorized into three categories:

- Slack bus (swing or reference bus),
- PQ bus (sometimes called as a load bus),
- PV bus.

| Bus types | Quantities specified | Unknown variables |
|---|---|---|
| Slack | $\|U\|, \delta, P_D, Q_D$ | $P_G, Q_G$ |
| PQ | $P_G, Q_G, P_D, Q_D$ | $\|U\|, \delta$ |
| PV | $\|U\|, P_G, P_D, Q_D$ | $\delta, Q_G$ |

## Slack bus

There is only one slack bus in system under consideration. Slack bus always has a generator attached to it, with no exception. Normally this generator is biggest in the system. Its two main tasks are to:

- Serve as the reference for voltage angle,
- Balance generation, load and losses, because the power losses are not known until end of load flow calculation. Slack bus needs to supply complex losses.

The rest of buses swing with the reference to this particular bus.

Whatever is extra left that will come from this slack bus remaining, anything which we could not fulfill from the rest of the buses will come from it.

## PQ bus

Load buses may contain generators with specified real and reactive power outputs.

## PV bus

Have generator connected to them. The PV buses can have voltage control capabilities and uses a tap-adjustable transformer and and/or VAR compensator instead of generator.

## Load flow calculation – step by step

We can set following general steps in order to describe load flow calculation:

1. Specify values of elements for network components
2. Specify place, values and constraints for loads in the power system
3. Define specifications and constraints for generators in the power system
4. Establish a math model describing load flow in the power system
5. Solve the model equations for the voltage profile of the power system
6. Solve the model equations for the power flows and losses in the power system
7. Verify if there are some constraint violations

## Main File

To see the results of any of the three methods we should run main.m file. So, first, we are going to see our main file. It can be run by “run main.m”.

Main.m code in like below:

```
met = input('Enter the method for load flow (1 - GS, 2 - NR, 3 - Decouple): ');
while met ~= 1 && met ~= 2 && met ~= 3
    fprintf('Invalid Input try again\n');
    met = input('Enter the method for load flow (1 - GS, 2 - NR, 3 - Fast Decouple): ');
end
switch met
    case 1
        maingauss
    case 2
        mainnewton
    case 3
        maindecouple
end
```

Here first of all, an input is requested that will choose the method which is going to run. Then the selected method will run. By choosing a number between 1 to 3 a main file including the method will run. Codes in main files are shown below:

### Gauss-Seidel main file

```
basemva = 100;  accuracy = 0.001; accel = 1.2; maxiter = 100;

lfybus                          % form the bus admittance matrix
Ybus
tic
lfgauss               % Load flow solution by Gauss-Seidel method
toc
busout              % Prints the power flow solution on the screen
lineflow          % Computes and displays the line flow and losses
```

### Newton-Raphson main file

```
basemva = 100;  accuracy = 0.001; maxiter = 10;

lfybus                          % form the bus admittance matrix
Ybus
tic
lfnewton             % Load flow solution by Newton-Raphson method
toc
busout % Prints the power flow solution on the screen
lineflow          % computes and displays the line flow and losses
```

### Decouple main file

```
basemva = 100;  accuracy = 0.001; maxiter = 50;

lfybus                           % form the bus admittance matrix
Ybus
tic
decouple1              % Load flow solution by fast decoupled method
toc
busout              % Prints the power flow solution on the screen
lineflow          % Computes and displays the line flow and losses
```

As it can be seen, all the files are the same except a function named the method. So, we first show what is **lfybus** which is the same for all of them and then talk about each method individually.

## Lfybus: Load Flow Y bus: Calculating Y Bus

Our first file which was going to interduce was **lfybus.** For load flow analysis, we first should calculate our Y[1] bus.

```
j=sqrt(-1); i = sqrt(-1);
nl = linedata(:,1); nr = linedata(:,2); R = linedata(:,3);
X = linedata(:,4); Bc = j*linedata(:,5); a = linedata(:, 6);
nbr=length(linedata(:,1)); nbus = max(max(nl), max(nr));
Z = R + j*X; y= ones(nbr,1)./Z;        %branch admittance
for n = 1:nbr
if a(n) <= 0  a(n) = 1; else end
Ybus=zeros(nbus,nbus);     % initialize Ybus to zero
                % formation of the off diagonal elements
    for k=1:nbr;
       Ybus(nl(k),nr(k))=Ybus(nl(k),nr(k))-y(k)/a(k);
       Ybus(nr(k),nl(k))=Ybus(nl(k),nr(k));
    end
end
              % formation of the diagonal elements
for  n=1:nbus
     for k=1:nbr
         if nl(k)==n
         Ybus(n,n) = Ybus(n,n)+y(k)/(a(k)^2) + Bc(k);
         elseif nr(k)==n
         Ybus(n,n) = Ybus(n,n)+y(k) +Bc(k);
         else, end
     end
end
```

Here we first real all the line flow file, in this file, we have information about loads on line nl to nr (bus numbers) along with its relative R, X and bus impedance. Last column represents tap information which can be different from 1 and we should apply it to the impedance of our load.

To calculate the admittance first we find impedance from the information of line flow, inverse it, and then for each load first we calculate the off diagonal elements meaning the admittance from bus I to J where $I \neq J$ and then we calculate the admittance of each bus means I to I. The difference between them is the effect of tap ratio and the bus load it self which has effect on the admittance related to each bus. So far we have calculated the Y bus needed for load flow analysis in each of the three methods. After this we are going to interduce each method individually and talk about the ways they calculate the voltage of each line bus.

[1] Admittance Matrix

# Methods

So far, we have calculated the admittance matrix between each of the lines. Y bus is a 30*30 matrix (we have 30 bus), which indicates the admittance between each of two lines.

This matrix was needed in any of the three methods since to find the voltage we need to predict and update based on this matrix. Our updating rule is different in each of the methods but all of them should result in voltage bus and losses of each line. After this point we are going to determine each method individually and see the implementation. We have three files including “lfgauss, lfnewton, decouple1” indicating the method for updating and finding voltage. These files are going to represent in the following chapters.

## Gauss-Seidel

First, we are going to determine the gauss seidel method and demonstrate how it updates the voltages.

Gauss-Seidel method for power flow studies, let it be assumed that all buses other than the slack bus are P-Q or load buses. At slack bus both V and δ are specified and they remain fixed throughout. There are (n – 1) buses where P and Q are given. Initially we assume the magnitudes and angles at these (n – 1) buses and update these voltages at every step of iteration.

$$\mathbf{I}_i = \frac{P_i - jQ_i}{V_i^*} \quad \textbf{(1)}$$

$$\mathbf{V}_i = \frac{1}{\mathbf{Y}_n}\left[\mathbf{I}_i - \sum_{\substack{k=1 \\ k=i}}^{n} \mathbf{Y}_{ik}\mathbf{V}_k\right] \quad (2)$$

$$\mathbf{V}_i = \frac{1}{\mathbf{Y}_{ii}}\left[\frac{\mathbf{P}_i - j\mathbf{Q}_i}{\mathbf{V}_i^*} - \sum_{\substack{\mathbf{k=1} \\ k=i}}^{n} \mathbf{Y}_{ik}\mathbf{V}_{\mathrm{k}}\right]; \quad (3)$$

$$i = 2,3, \ldots n$$

Thus Eq. (3) represents a set of (n – 1) equations for i = 2, 3, …, n which are to be solved simultaneously for V2, V3, V4 … Vn.

In the Gauss method, we assume the voltage for all the buses except the slack bus where the voltage magnitude and phase angle are specified and remain fixed. Normally, we set i.e., assume the voltage magnitude and phase angle of these buses equal to that of the slack bus and work in per unit system.

The assumed bus voltage and the slack bus voltage along with P and Q are substituted in RHS of the Eq. (3) to obtain new set of bus voltages. After the entire iteration is complete, the new set of bus voltages is again substituted along with the specified slack bus voltage in the RHS of Eq. (3) to obtain a new set of bus voltages.

The process is continued till:

$$\mathrm{V}_i^{r+1} - \mathrm{V}_i^r \leq \varepsilon \text{ for } i = 2,3 \ldots n \quad (4)$$

where r is an iteration count and ε is a very small number which depends upon the system accuracy and is normally equal to 0.0001 etc.

Thus, the process is continued till the mod of the bus voltage obtained at the current iteration less the value of bus voltage at the previous iteration is smaller than a chosen very small number and, in this way, we obtain the solution, i.e., magnitude and phase angle of voltage.

Gauss iterative method, explained above, is much slower to converge and may sometimes fail to do so.

In case of Gauss-Seidel method, the value of bus voltages calculated for any bus immediately replace the previous values in the next step while in case of Gauss method, as stated earlier, the calculated bus voltages replace the earlier value only at the end of the iteration. Due to this Gauss- Seidel method converges much faster than that of Gauss method compared to Gauss method.

It has the following advantages and disadvantages:
Advantages:

1. Simplicity of technique.

2. Small computer memory requirement.

3. Less computational time per iteration.

Disadvantages:

1. Slow rate of convergence resulting in larger number of iterations.

2. Increase in number of iterations directly with the increase in the number of buses.

3. Effect on convergence due to choose of slack bus.

Because of the above drawbacks, use of Gauss-Seidel method is limited only to systems with smaller number of buses.

## Computations of Line Flows and Line Losses:

Current flowing from bus i towards bus k,

$$I_{ik} = [\mathbf{V}_i - \mathbf{V}_k]\mathbf{y}_{ik} + \mathbf{v}_i\mathbf{y}_{ik0} \qquad (5)$$

where Vi and Vk are the bus voltages at the buses i and k respectively which are already calculated from the power flow studies.
The power flow in the line i-k at the bus i is given as –

$$\mathbf{S}_{ik} = \mathrm{P}_{ik} + j\mathrm{Q}_{ik} = \mathbf{V}_i\mathbf{r}_{ik}^* = \mathbf{V}_i(\mathbf{v}_i^* - \mathbf{V}_k^*)\mathbf{y}_{ik}^* + \mathbf{V}_i\mathbf{v}_i^*\mathbf{y}_{ik0} \qquad \mathbf{(6)}$$

Similarly, the power flow in the line i-k at the bus k is given as –

$$\mathbf{S}_{ki} = \mathbf{v}_k(\mathbf{v}_k^* - \mathbf{v}_i^*)\mathbf{y}_{ik}^* + \mathbf{v}_k\mathbf{v}_k^*\mathbf{y}_{ki}^* \qquad (7)$$

Thus, power flows over all the lines can be computed.

The power losses in the (I – k) th line are given by sum of the power flows determined from above Eqs. (6) and (7) i.e.
power losses in the (I – k) th line = Sik + Ski.
Total transmission losses can be computed by summing all the line flows (i.e., Sik + Ski. for all i, k).
It may be noted that the slack bus power can also be determined by summing the power flows on the lines terminating at the slack bus. This concludes the power flow study for the case of P-Q buses only.

## Treatment to Voltage Controlled Buses in Gauss-Seidel Method:

In a power system, some of the buses are voltage-controlled buses where P and V are specified while Q and 8 are unknowns and are to be determined. However, usually limit of reactive power, i.e., Qmax and Qmin to hold the generation voltage within limits are also given.

So far power flow study in a system with generation or voltage-controlled buses with G-S method the values of Q and δ are to be updated in every GS iteration through appropriate bus equations and therefore computational procedure needs some modifications.

Let the buses be numbered as –

i = 1 slack or swing bus

i = 2, 3 … m voltage controlled or P-V buses

i = m + 1, m + 2, m + 3, …, n load or P-Q buses

For the voltage-controlled buses, the bus voltage Vi must be equal to the specified voltage Vi specified in magnitude and the value of reactive power Qi (for i = 2, 3, …, m) must lie between the limits.

Thus, the conditions to be satisfied are –

$$V_i = V_{i\,\text{specified}} \text{ for } i = 2,3,\dots,m$$

$$Q_{i\min} < Q_i < Q_{i\max} \text{ for } i = 2,3,\dots,m$$

I. Calculate reactive power generation which is reproduced below:

$$\mathrm{Q}_i = -\mathrm{V}_i \sum_{\mathrm{k}=1}^{n} \mathrm{V}_\mathrm{k} \mathrm{Y}_{i\mathrm{k}} \sin\left(\theta_{i\mathrm{k}} + \delta_\mathrm{k} - \delta_i\right) \qquad \textbf{(8)}$$

At the beginning of (r + 1) th iteration, Vi(r), i.e., magnitude of Vi obtained during rth iteration may not necessarily be equal to Vi specified. We have this value of Vi(r)

$$\mathbf{v}_i^{(r)} = \mathbf{v}_i^{(r)} \angle \delta_i^m \qquad \textbf{(9)}$$

The values of Vk and δk to be used in Eq. (9) are those obtained during (r + 1) th iteration for (k = 1 to i – 1) and those obtained during rth iteration (for k = i to n).

Moreover, for every iteration Vi must be equal to Vi specified. In fact, for the (r + 1) th iteration one can write from Eq. (8) –

$$Q_i^{r+1} = -V_{i\ \textbf{specified}} \sum_{k=1}^{n-1} Y_{ik} V_k^{(r+1)} \sin\left(\theta_\lambda + \delta_k^{(r+1)} - \delta_i^{(r)}\right) \qquad \textbf{(10)}$$

$$= -V_{i\ \textbf{specified}} \sum_{k=1}^{n} Y_{ik} V_k^{r} \sin\left(\theta_{ik} + \delta_k^{r} - \delta_i^{r}\right)$$

(ii) If Qi(r + 1) is found to lie within limits, Qi min < Qi(r + 1) < Qi max calculate new value of Vi(r + 1) using Vi specified and δi(r) for the magnitude and phase angle of Vi(r + 1). Reset Vi(r + 1) to Vi specified but retain the phase angle δi(r + 1) and continue to the next bus.

If Qi(r + 1) < Qi min, set Qi(r + 1) = Qi min and treat bus i as a P-Q bus.

If Qi(r + 1) > Qi max set Qi(r + 1) = Qi max and treat bus i as P-Q bus.

# Implementation

## Load Bus data:

```
for k=1:nbus
n=busdata(k,1);
kb(n)=busdata(k,2); Vm(n)=busdata(k,3); delta(n)=busdata(k, 4);
Pd(n)=busdata(k,5); Qd(n)=busdata(k,6); Pg(n)=busdata(k,7); Qg(n) = busdata(k,8);
Qmin(n)=busdata(k, 9); Qmax(n)=busdata(k, 10);
Qsh(n)=busdata(k, 11);
    if Vm(n) <= 0  Vm(n) = 1.0; V(n) = 1 + j*0;
    else delta(n) = pi/180*delta(n);
         V(n) = Vm(n)*(cos(delta(n)) + j*sin(delta(n)));
         P(n)=(Pg(n)-Pd(n))/basemva;
         Q(n)=(Qg(n)-Qd(n)+ Qsh(n))/basemva;
         S(n) = P(n) + j*Q(n);
    end
DV(n)=0;
end
```

## Find Y * V and calculate power for buses:

```
for n = 1:nbus;
YV = 0+j*0;
  for L = 1:nbr;
          if nl(L) == n, k=nr(L);
          YV = YV + Ybus(n,k)*V(k);
          elseif nr(L) == n, k=nl(L);
          YV = YV + Ybus(n,k)*V(k);
          end
  end
     Sc = conj(V(n))*(Ybus(n,n)*V(n) + YV) ;
     Sc = conj(Sc);
     DP(n) = P(n) - real(Sc);
     DQ(n) = Q(n) - imag(Sc);
```

## Treat each type of bus and handle Q conditions:

```
   if kb(n) == 1
   S(n) =Sc; P(n) = real(Sc); Q(n) = imag(Sc); DP(n) =0; DQ(n)=0;
   Vc(n) = V(n);
   elseif kb(n) == 2
   Q(n) = imag(Sc); S(n) = P(n) + j*Q(n);

     if Qmax(n) ~= 0
       Qgc = Q(n)*basemva + Qd(n) - Qsh(n);
       if abs(DQ(n)) <= .005 & iter >= 10 % After 10 iterations
         if DV(n) <= 0.045                % the Mvar of generator buses are
            if Qgc < Qmin(n),             % tested. If not within limits Vm(n)
            Vm(n) = Vm(n) + 0.005;        % is changed in steps of 0.005 pu
            DV(n) = DV(n)+.005;           % up to .05  pu in order to bring
            elseif Qgc > Qmax(n),         % the generator Mvar within the
            Vm(n) = Vm(n) - 0.005;        % specified limits.
            DV(n)=DV(n)+.005; end
         else, end
       else,end
     else,end
   end
if kb(n) ~= 1
Vc(n) = (conj(S(n))/conj(V(n)) - YV )/ Ybus(n,n);
else, end
   if kb(n) == 0
   V(n) = V(n) + accel*(Vc(n)-V(n));
   elseif kb(n) == 2
   VcI = imag(Vc(n));
   VcR = sqrt(Vm(n)^2 - VcI^2);
   Vc(n) = VcR + j*VcI;
    V(n) = V(n) + accel*(Vc(n) -V(n));
```

Find Results:

```
for n = 1:nbus
   Vm(n) = abs(V(n)); deltad(n) = angle(V(n))*180/pi;
      if kb(n) == 1
      S(n)=P(n)+j*Q(n);
      Pg(n) = P(n)*basemva + Pd(n);
      Qg(n) = Q(n)*basemva + Qd(n) - Qsh(n);
      k=k+1;
      Pgg(k)=Pg(n);
      elseif  kb(n) ==2
      k=k+1;
      Pgg(k)=Pg(n);
      S(n)=P(n)+j*Q(n);
      Qg(n) = Q(n)*basemva + Qd(n) - Qsh(n);
      end
 yload(n) = (Pd(n)- j*Qd(n)+j*Qsh(n))/(basemva*Vm(n)^2);
end
 Pgt = sum(Pg);  Qgt = sum(Qg); Pdt = sum(Pd); Qdt = sum(Qd); Qsht = sum(Qsh);
 busdata(:,3)=Vm'; busdata(:,4)=deltad';
```

# Newton-Raphson

The power flow problem can also be solved by using Newton-Raphson method. In fact, among the numerous solution methods available for power flow analysis, the Newton-Raphson method is considered to be the most sophisticated and important. Many advantages are attributed to the Newton-Raphson (N-R) approach.

Gauss-Seidel (G-S) is a simple iterative method of solving n number load flow equations by iterative method. It does not require partial derivatives. Newton-Raphson method is based on Taylor's series and partial derivatives.

The N-R method is recent, needs less number of iterations to reach convergence, takes less computer time hence computation cost is less and the convergence is certain. The N-R method is more accurate, and is insensitive to factors like slack bus selection, regulating transformers etc. and the number of iterations required in this method is almost independent of the system size.

The drawbacks of this method are difficult solution technique, more calculations involved in each iteration resulting in large computer time per iteration and the large requirement of computer memory but the last drawback has been overcome through a compact storage scheme.

Convergence can be considerably speeded up by performing the first iteration through the G-S method and using the values of voltages so obtained for starting the N-R iterations. These voltages are used to compute active power P at every bus except the swing bus and also reactive power Q wherever reactive power is specified.

The difference between the specified and calculated values is used to determine the correction of bus voltages. The process of iteration is continued till the difference in the specified and calculated values of P, Q and V are within the given permissible limit.

N-R method can be applied to power flow problems in a number of ways, here we are going to interduce and use rectangular coordinates.

In this formulation the quantities are expressed in rectangular form.

The general expression for power is given a –

$$\mathrm{P}_i - j\mathrm{Q}_i = \mathbf{V}_i^* \mathrm{I}_i = \mathbf{V}_i^* \sum_{\mathrm{k}=1}^{n} \mathbf{Y}_{i\mathrm{k}} \mathbf{V}_{\mathrm{k}} \qquad \textbf{(11)}$$

**Let** $\mathbf{V}_i = e_i + jf_i$

Where ei and fi are the real and imaginary components of the bus voltage vi, and therefore –

$$\mathbf{V}_i^* = e_i - jf_i \qquad \textbf{(12)}$$

$$\mathbf{V}_{\mathbf{k}} = e_{\mathbf{k}} + jf_{\mathbf{k}} \qquad \textbf{(13)}$$

$$\mathbf{Y}_{ik} = \mathrm{G}_{ik} - j\ \mathrm{B}_{ik} \qquad \textbf{(14)}$$

Where Gik and Bik are conductance and susceptance respectively.
Substituting Eqs. (12), (13) and (14) in Eq.(11) for power, we have:

$$\mathrm{P}_i - j\mathrm{Q}_i = (e_i - jf_i) \sum_{\mathrm{k}=1}^{n} (\mathrm{G}_{\mathrm{ik}} - j\ \mathrm{B}_{\mathrm{hk}})(e_{\mathrm{k}} + jf_{\mathrm{k}}) \qquad \textbf{(15)}$$

Separating the real and imaginary parts, we have-

$$P_i = \sum_{k=1}^{n} e_i(e_k G_{ik} + f_k B_{ik}) + f_i(f_k G_{ik} - e_k B_{ik}) \qquad \textbf{(16)}$$

$$Q_i = \sum_{k=1}^{n} f_i(e_k G_{ik} + f_k B_{ik}) - e_i(f_k G_{ik} - e_k B_{ik}) \qquad \textbf{(17)}$$

$$\mathrm{V}_i^2 = e_i^2 + f_i^2 \qquad \textbf{(18)}$$

Separating for ith bus, the power Eqs. (16) and (17) become –

$$\mathrm{P}_i = e_i \quad (e_i \mathrm{G}_{ii} + f_i \, \mathrm{B}_{ii}) + f_i (f_i \mathrm{G}_{ii} - e_i \, \mathrm{B}_{ii}) + \sum_{\substack{\mathrm{k}=1 \\ \mathrm{k} * i}}^{n} e_i (e_\mathrm{k} \mathrm{G}_{ik} + f_\mathrm{k} \mathrm{B}_{ik}) + f_i (f_\mathrm{k} \mathrm{G}_\mathrm{k} - e_\mathrm{k} \mathrm{B}_{ik}) \quad \textbf{(19)}$$

$$Q_i = f_i (e_i G_{ii} + f_i B_{ii}) - e_i (f_i G_{ii} - e_i B_{ii}) + \sum_{\substack{k=1 \\ k \neq i}}^{n} f_i (e_k G_{ik} + f_k B_{ik}) - e_i (f_k G_{ik} - e_k B_{ik}) \quad \textbf{(20)}$$

Thus the above formulation results in a system of nonlinear algebraic equations, two equations (one for Pi and the other for Qi) at each bus. So excluding the slack bus (bus 1) where V and δ are specified and remains fixed throughout, the total number of equations to be solved for n bus system will be (2n – 1) equations. With the help of the Newton-Raphson method, the above nonlinear algebraic equations of power is transferred into a set of linear algebraic equations inter-relating the changes in power (i.e., error in power) with the change in real and reactive components of bus voltages with the help of jacobian matrix. Thus we have from Eqs. (16) and (17).

$$\begin{bmatrix} \Delta \mathrm{P}_2 \\ \Delta \mathrm{P}_3 \\ | \\ \Delta \mathrm{P}_n \\ \Delta \mathrm{Q}_2 \\ \Delta \mathrm{Q}_3 \\ | \\ \Delta Q_n \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathrm{P}_2}{\partial e_2} & \frac{\partial \mathrm{P}_2}{\partial e_3} & \frac{\partial \mathrm{P}_2}{\partial e_n} & \frac{\partial \mathrm{P}_2}{\partial f_2} & \frac{\partial \mathrm{P}_2}{\partial f_3} & \frac{\partial \mathrm{P}_2}{\partial f_n} & \Delta e_2 \\ \frac{\partial \mathrm{P}_3}{\partial e_2} & \frac{\partial \mathrm{P}_3}{\partial e_3} & \frac{\partial \mathrm{P}_3}{\partial e_n} & \frac{\partial \mathrm{P}_3}{\partial f_2} & \frac{\partial \mathrm{P}_3}{\partial f_3} & \frac{\partial \mathrm{P}_3}{\partial f_n} & \Delta e_3 \\ \frac{\partial \mathrm{P}_n}{\partial e_2} & \frac{\partial \mathrm{P}_n}{\partial e_3} & \frac{\partial \mathrm{P}_n}{\partial e_n} & \frac{\partial \mathrm{P}_n}{\partial f_2} & \frac{\partial \mathrm{P}_n}{\partial f_3} & \frac{\partial \mathrm{P}_n}{\partial f_n} & \Delta e_n \\ \frac{\partial \mathrm{Q}_2}{\partial e_2} & \frac{\partial Q_2}{\partial e_3} & \frac{\partial Q_2}{\partial e_n} & \frac{\partial \mathrm{Q}_2}{\partial f_2} & \frac{\partial Q_2}{\partial f_3} & \frac{\partial Q_2}{\partial f_n} & \Delta f_2 \\ \frac{\partial \mathrm{Q}_3}{\partial e_2} & \frac{\partial Q_3}{\partial e_3} & \frac{\partial Q_3}{\partial e_n} & \frac{\partial Q_3}{\partial f_2} & \frac{\partial \mathbf{Q}_3}{\partial f_3} & \frac{\partial Q_3}{\partial f_n} & \Delta f_3 \\ \frac{\partial Q_n}{\partial e_2} & \frac{\partial Q_n}{\partial e_3} & \frac{\partial Q_n}{\partial e_n} & \frac{\partial Q_n}{\partial f_2} & \frac{\partial Q_n}{\partial f_3} & \frac{\partial Q_n}{\partial f_n} & \end{bmatrix} \quad \textbf{(21)}$$

In short form Eq. (21) can be written as –

$$\begin{bmatrix} \frac{\Delta \mathrm{P}}{\Delta \mathrm{Q}} \end{bmatrix} = \begin{bmatrix} \mathrm{J}_1 & \mathrm{J}_2 \\ \mathrm{J}_3 & \mathrm{J}_4 \end{bmatrix} \begin{bmatrix} \frac{\Delta e}{\Delta f} \end{bmatrix} \quad \textbf{(22)}$$

In case the system contains all types of buses, the set of equations can be written as

$$\begin{bmatrix}\Delta \mathrm{P} \\ \Delta \mathrm{Q}\end{bmatrix} = \begin{bmatrix}\mathrm{J}_1 & \mathrm{J}_2 \\ \mathrm{J}_3 & \mathrm{J}_4 \\ \mathrm{J}_5 & \mathrm{J}_6\end{bmatrix}\begin{bmatrix}\Delta e \\ \dots \dots \\ \Delta f\end{bmatrix} \quad \textbf{(23)}$$

The elements of the jacobian matrix can be derived from the three-power flow Eqs. (19), (20) and (18).

The off-diagonal elements of J1 are –

$$\frac{\partial \mathrm{P}_i}{\partial e_\mathrm{k}} = e_i \mathrm{G}_{i\mathrm{k}} - f_i \, \mathrm{B}_{ik} \textbf{ for } \mathrm{k} \neq i \quad \textbf{(24)}$$

And diagonal elements of J1 are –

$$\begin{aligned}\frac{\partial P_i}{\partial e_i} &= 2e_i G_{ii} + f_i B_{ii} - f_i B_i + \sum_{\substack{k=1 \\ k=1}}^{n} (e_k G_{ik} + f_k B_{ik}) \\ &= 2e_i G_{ii} + \sum_{\substack{k=1 \\ k=1}}^{n} (e_k G_{ik} + f_k B_{ik})\end{aligned} \quad \textbf{(25)}$$

The off-diagonal elements of J2 are –

$$\frac{\partial P_i}{\partial f_k} = e_i B_{ik} + f_i G_{ik} \textbf{ for } k \neq i \quad \textbf{(26)}$$

And the diagonal elements of J2 are –

$$\frac{\partial P_i}{\partial f_i} = e_i B_i + 2f_i G_{ii} - e_i B_{ij} + \sum_{\substack{k=1 \\ k=1}}^{n} (f_k G_{ik} - e_k B_{ik}) = 2f_i G_{ii} + \sum_{\substack{k=1 \\ k=i}}^{n} (f_k G_{ik} - e_k B_{ik}) \quad \textbf{(27)}$$

The off-diagonal elements of j3 are –

$$\frac{\partial Q_i}{\partial e_k} = e_i B_{ik} + f_i G_{ik} \textbf{ for } k \neq i \quad \textbf{(28)}$$

And the diagonal elements of j3 are –

$$\frac{\partial Q_i}{\partial e_i} = f_i G_{ij} - f_i G_{ii} + 2e_i B_{ii} - \sum_{\substack{k=1 \\ k=1}}^{n} (f_k G_{ik} - e_k B_{ik}) = 2e_i B_{ij} - \sum_{\substack{k=1 \\ k=i}}^{n} (f_k G_{ik} - e_k B_k) \quad \textbf{(29)}$$

The off-diagonal elements of j4 are –

$$\frac{\partial Q_i}{\partial f_k} = -e_i G_{ik} + f_i B_{ii} \textbf{ for } k \neq i \quad \textbf{(30)}$$

And the diagonal elements of j4 are –

$$\frac{\partial Q_i}{\partial f_i} = e_i G_{ii} + 2f_i B_{ii} - e_i G_{ij} + \sum_{k=1}^{n} (e_k G_{ik} + f_k B_{ik}) = 2f_i B_u + \sum_{\substack{k=1 \\ k=1}}^{n} (e_k G_{ik} + f_k B_{ik}) \quad \textbf{(31)}$$

The off-diagonal and diagonal elements of j5 are –

$$\frac{\partial \mathrm{V}_i^2}{\partial e_{\mathrm{k}}} = 0 \textbf{ for } \mathrm{k} \neq \textbf{i} \quad \textbf{(32)}$$

$$\frac{\partial \mathrm{V}_i^2}{\partial e_i} = 2e_i \quad \textbf{(33)}$$

The off-diagonal and diagonal elements of j6 are –

$$\frac{\partial \mathrm{V}_i^2}{\partial f_{\mathrm{k}}} = 0 \textbf{ for } \mathrm{k} \neq \mathrm{i} \quad \textbf{(34)}$$

$$\frac{\partial V_i^2}{\partial f_i} = 2f_i \quad \textbf{(35)}$$

## Algorithm

**The steps for solving power flow problem by the N-R method are given below:**

1. So, for the load buses where P and Q are given, we assume the bus voltages magnitude and phase angle for all the buses except the slack bus where V and δ are specified. Normally we have the flat voltage start, i.e., we set the assumed bus voltage magnitude and its phase angle (i.e., the real and imaginary components e and f of the bus voltages) equal to the slack bus quantities.
2. Substituting this assumed bus voltages (i.e., e and f) in Eqs. (19) and (20), we calculate the real and reactive components of power, i.e., Pi and Qi for all the buses i = 2, 3, 4, …, n except the slack bus (bus no. 1).
3. Since Pi and Qi for any bus i is given, i.e., specified, the error in power will be –

$$\Delta P_i^r = P_{i\ \text{specified}} - P_i^r \quad (36)$$

$$\Delta Q_i^r = Q_{i\ \text{specified}} - Q_i^r \quad (37)$$

where r is an iteration count.

Here Rri and Qri are the power calculated with the latest value of bus voltages at any iteration r.
4. Then the elements of Jacobian matrix (J1, J2, J3 and J4) are determined with the latest bus voltages and calculated power Eqs. (19) and (20).
5. After this the linear set of Eq. (21) is solved by iterative technique or by the method of elimination (normally by Gaussian elimination method) to determine the voltage correction, i.e., Δei and Δ*f*i at any bus i.
6. This value of voltage correction is used to determine the new estimate of bus voltages as follows:

$$e_i^{r+1} = e_i^r + \Delta e_i^r \quad (38)$$

$$f_i^{r+1} = f_i^r + \Delta f_i^r \quad (39)$$

Where r is an iteration count.

7. Now this new estimate of the bus voltage, i.e. eir + 1 and *f*ir + 1 is used in Eqs. (19) and (20) for power to re-compute the error in power and thus entire algorithm starting from step 3 as listed above is repeated.
Here in each iteration, the elements of Jacobian are computed as these depend upon the latest voltage estimate and calculated power. The process is continued till the error in power becomes very small.

i.e., $\Delta P < \varepsilon$ and $\Delta Q < \varepsilon$ (40)

where ϵ is very small number.

## Voltage-Controlled or Generation Bus:

Here P and the magnitude of voltage V are given.

Now the real power P for any bus i is given as:

$$P_i = \textbf{Real}\, V_i^* \sum_{k=1}^{n} V_k Y_{ik} \quad \textbf{(41)}$$

And also for ith bus, we have –

$$V_i^2 = e_i^2 + f_i^2 \quad \textbf{(42)}$$

Where Vi is the voltage magnitude and ei and *f*i are its real and imaginary components.
The matrix equations inter-relating the changes in bus powers and square of the bus voltage magnitude to the changes in the real and imaginary components of voltage are given by Eq.(6.98 b).

Where $\Delta(V^r i)^2 = \left(V_{i\ \text{specified}}\right)^2 - (V_i^r)^2$ and Vri is the calculated bus voltage after the rth iteration.
Elements of jacobian matrix are calculated using Eqs. (32), (35).

Here Vri is the bus voltage computed at the rth iteration and Vi specified is the voltage specified at any bus i as it is the generation bus.
After obtaining bus voltages, power flow and lines losses are calculated using Eqs. (6) and (7).

## Implementation

### Load Data:

```
for k=1:nbus
n=busdata(k,1);
kb(n)=busdata(k,2); Vm(n)=busdata(k,3); delta(n)=busdata(k, 4);
Pd(n)=busdata(k,5); Qd(n)=busdata(k,6); Pg(n)=busdata(k,7); Qg(n) = busdata(k,8);
Qmin(n)=busdata(k, 9); Qmax(n)=busdata(k, 10);
Qsh(n)=busdata(k, 11);
    if Vm(n) <= 0  Vm(n) = 1.0; V(n) = 1 + j*0;
    else delta(n) = pi/180*delta(n);
         V(n) = Vm(n)*(cos(delta(n)) + j*sin(delta(n)));
         P(n)=(Pg(n)-Pd(n))/basemva;
         Q(n)=(Qg(n)-Qd(n)+ Qsh(n))/basemva;
         S(n) = P(n) + j*Q(n);
    end
end
```

### Calculate Jacobian:

```
 for i=1:nbr
   if nl(i) == n | nr(i) == n
      if nl(i) == n,  l = nr(i); end
      if nr(i) == n,  l = nl(i); end
      J11=J11+ Vm(n)*Vm(l)*Ym(n,l)*sin(t(n,l)- delta(n) + delta(l));
      J33=J33+ Vm(n)*Vm(l)*Ym(n,l)*cos(t(n,l)- delta(n) + delta(l));
      if kb(n)~=1
      J22=J22+ Vm(l)*Ym(n,l)*cos(t(n,l)- delta(n) + delta(l));
      J44=J44+ Vm(l)*Ym(n,l)*sin(t(n,l)- delta(n) + delta(l));
      else, end
      if kb(n) ~= 1  & kb(l) ~=1
      lk = nbus+l-ngs(l)-nss(l)-ns;
      ll = l -nss(l);
    % off diagonalelements of J1
      A(nn, ll) =-Vm(n)*Vm(l)*Ym(n,l)*sin(t(n,l)- delta(n) + delta(l));
            if kb(l) == 0  % off diagonal elements of J2
            A(nn, lk) =Vm(n)*Ym(n,l)*cos(t(n,l)- delta(n) + delta(l));end
            if kb(n) == 0  % off diagonal elements of J3
            A(lm, ll) =-Vm(n)*Vm(l)*Ym(n,l)*cos(t(n,l)- delta(n)+delta(l)); end
            if kb(n) == 0 & kb(l) == 0  % off diagonal elements of  J4
            A(lm, lk) =-Vm(n)*Ym(n,l)*sin(t(n,l)- delta(n) + delta(l));end
      else end
   else , end
 end
```

Calculate Result:

```
for n=1:nbus
   nn=n-nss(n);
   lm=nbus+n-ngs(n)-nss(n)-ns;
     if kb(n) ~= 1
     delta(n) = delta(n)+DX(nn); end
     if kb(n) == 0
     Vm(n)=Vm(n)+DX(lm); end
 end
```

Final Results:

```
for n = 1:nbus
      if kb(n) == 1
      k=k+1;
      S(n)= P(n)+j*Q(n);
      Pg(n) = P(n)*basemva + Pd(n);
      Qg(n) = Q(n)*basemva + Qd(n) - Qsh(n);
      Pgg(k)=Pg(n);
      Qgg(k)=Qg(n);      %june 97
      elseif  kb(n) ==2
      k=k+1;
      S(n)=P(n)+j*Q(n);
      Qg(n) = Q(n)*basemva + Qd(n) - Qsh(n);
      Pgg(k)=Pg(n);
      Qgg(k)=Qg(n);  % June 1997
   end
 yload(n) = (Pd(n)- j*Qd(n)+j*Qsh(n))/(basemva*Vm(n)^2);
end
busdata(:,3)=Vm'; busdata(:,4)=deltad';
Pgt = sum(Pg);  Qgt = sum(Qg); Pdt = sum(Pd); Qdt = sum(Qd); Qsht = sum(Qsh);
```

# Decoupled

The fast decoupled power flow method is a very fast and efficient method of obtaining power flow problem solution. In this method, both, the speeds as well as the sparsity are exploited. This is actually an extension of Newton-Raphson method formulated in polar coordinates with certain approximations which result into a fast algorithm for power flow solution.

This method exploits the property of the power system where in MW flow-voltage angle and MVAR flow-voltage magnitude are loosely coupled. In other words a small change in the magnitude of the bus voltage does not affect the real power flow at the bus and similarly a small change in phase angle of the bus voltage has hardly any effect on reactive power flow.

Because of this loose physical interaction between MW and MVAR flows in a power system, the MW- δ and MVAR-V calculations can be decoupled. This decoupling results in a very simple, fast and reliable algorithm. As we know, the sparsity feature of admittance matrix minimizes the computer memory requirements and results in faster computations. The accuracy is comparable to that of the N-R method.

The earlier equation of power flow studies using N-R method can be written in as

$$\begin{bmatrix}\Delta P\\ \Delta Q\end{bmatrix} = \begin{bmatrix}H & N\\ M & L\end{bmatrix}\begin{bmatrix}\Delta\delta\\ \frac{\Delta V}{V}\end{bmatrix} \qquad \textbf{(43)}$$

where H, N, M and L are the elements (viz., J1, J2, J3 and J4) of the Jacobian matrix.
Since changes in real power (i.e., ΔP) are less sensitive to the changes in voltage magnitude (i.e., ΔV) and changes in reactive power (i.e., ΔQ) are less sensitive to the changes in phase angle of voltage (i.e., Δδ), Eq. (43) can be reduced to –

$$\begin{bmatrix}\Delta P\\ \Delta Q\end{bmatrix} = \begin{bmatrix}H & 0\\ 0 & L\end{bmatrix}\begin{bmatrix}\Delta 8\\ \frac{\Delta V}{V}\end{bmatrix} \qquad \textbf{(44)}$$

The Eq. (44) is decoupled equation and can be expanded as –

$$\dot{\Delta P} = H\Delta\delta \quad \textbf{(45)}$$

$$\Delta Q = L\frac{\Delta V}{V} \quad \textbf{(46)}$$

The next step in deriving the algorithm is to make suitable assumptions in deriving the expressions for H and L.

Off-diagonal element of H is –

$$\begin{aligned} \mathrm{H}_{ik} &= \frac{\partial \mathrm{P}_i}{\partial \mathrm{Q}_\mathrm{k}} = \mathrm{V}_i\ \mathrm{V}_\mathrm{k}\mathrm{Y}_{i\mathrm{k}}\sin\left(\theta_{i\mathrm{k}} + \delta_i - \delta_\mathrm{k}\right) \\ &= \mathrm{V}_i\ \mathrm{V}_\mathrm{k}\mathrm{Y}_{ik}[\sin\ \theta_{ik}\cos\left(\delta_i - \delta_\mathrm{k}\right) + \cos\ \theta_{ik}\sin\left(\delta_i - \delta_\mathrm{k}\right)] \\ &= \mathrm{V}_i\ \mathrm{V}_k[\mathrm{Y}_{ik}\sin\ \theta_{ik}\cos\left(\delta_i - \delta_k\right) + \mathrm{Y}_{ik}\cos\ \theta_{ik}\sin\left(\delta_i - \delta_\mathrm{k}\right)] \\ &= \mathrm{V}_i\ \mathrm{V}_\mathrm{k}[-\mathrm{B}_{i\mathrm{k}}\cos\left(\delta_i - \delta_\mathrm{k}\right) + \mathrm{G}_{ik}\sin\left(\delta_i - \delta_\mathrm{k}\right)] \end{aligned} \quad \textbf{(47)}$$

Similarly, off-diagonal element of L is –

$$\mathrm{L}_{ik} = \frac{\partial \mathrm{Q}_i\ \mathrm{V}_\mathrm{k}}{\partial \mathrm{V}_\mathrm{k}} = \mathrm{V}_j\ \mathrm{V}_\mathrm{k}\mathrm{Y}_{ik}\sin\left(\theta_{ik} + \delta_i - \delta_\mathrm{k}\right) = \mathrm{V}_i\ \mathrm{V}_\mathrm{k}[\mathrm{G}_{ik}\sin\left(\delta_i - \delta_\mathrm{k}\right) - \mathrm{B}_{ik}\cos\left(\delta_i - \delta_\mathrm{k}\right)] \quad \textbf{(48)}$$

From Eqs. (47) and (48), we have –

$$H_{ik} = L_{ik} = V_iV_k[G_{ik}\sin\left(\delta_i - \delta_k\right) - B_{ik}\cos\left(\delta_i - \delta_k\right)] \quad \textbf{(49)}$$

The diagonal elements of H are given as –

$$\begin{aligned} \mathrm{H}_{il} = \frac{\partial \mathrm{P}_i}{\partial \delta_j} &= -\sum_{\substack{k=1 \\ k=i}}^{n} \mathrm{V}_i\ \mathrm{V}_\mathrm{k}\mathrm{Y}_{2\mathrm{k}}\sin\left(\theta_{i\mathrm{k}} + \delta_i - \delta_\mathrm{k}\right) \\ &= -\mathrm{Q}_i + \mathrm{V}_i\ \mathrm{V}_j\mathrm{Y}_{ii}\sin\ \theta_{ij} = -\mathrm{Q}_i - \mathrm{V}_i^2\ \mathrm{B}_{ii} \end{aligned} \quad \textbf{(50)}$$

Similarly, diagonal elements for the matrix are given as –

$$\begin{aligned} L_{il} &= \frac{\partial Q_i V_i}{\partial V_i} = 2V_i^2 Y_{ii}\sin\,\theta_{ii} + \sum_{\substack{k=1 \\ k\neq i}}^{n} V_i V_k Y_{ik}\sin\,(\theta_{ik} + \delta_i - \delta_k) \\ &= 2V_i^2 Y_{ii}\sin\,\theta_{ii} + Q_i - V_i^2 Y_{ii}\sin\,\theta_{ii} = Q_t + V_i^2 Y_{ii}\sin\,\theta_{ii} = Q_i - V_i^2 B_{ii} \end{aligned} \quad \textbf{(51)}$$

In the case of fast decoupled power flow method of power flow studies, the following approximations are made for evaluating Jacobian elements.

$$\begin{aligned} \cos\,(\delta_i - \delta_k) &= 1 \\ G_{ik}\sin\,(\delta_i - \delta_k) &\leq B_{ik} \\ Q_i &< B_{ii} v_i^2 \end{aligned} \quad \textbf{(52)}$$

With the above assumptions the Jacobian elements become –

$$\mathrm{H}_{i\mathrm{k}} = \mathrm{L}_{i\mathrm{k}} = -\mathrm{V}_i\,\mathrm{V}_{\mathrm{k}}\mathrm{B}_{2\mathrm{k}} \textbf{ for } \mathrm{k} \neq i$$

$$\textbf{and } \mathrm{H}_{ii} = \mathrm{L}_{ii} = -\mathrm{V}_i^2\,\mathrm{B}_i \quad \textbf{(53)}$$

With these Jacobian elements Eqs. (45) and (46) become –

$$\Delta\mathrm{P}_i = \mathrm{H}\Delta\delta = \mathrm{V}_i\,\mathrm{V}_{\mathrm{k}}\mathrm{B}'_{i\mathrm{k}}\Delta\delta_{\mathrm{k}} \quad \textbf{(54)}$$

$$\Delta\mathrm{Q}_i = \mathrm{L}\frac{\Delta\mathrm{V}}{\mathrm{V}} = \mathrm{V}_i\,\mathrm{V}_{\mathrm{k}}\mathrm{B}''_{\mathbf{ik}}\,\frac{\Delta\mathrm{V}_{\mathrm{k}}}{\mathrm{V}_{\mathrm{k}}} \quad \textbf{(55)}$$

Where Bik and Bik are elements of- Bik matrix.

## Algorithm

I. Omitting from B', the representation of those network elements that affect MVAR flows, i.e., shunt reactance and off-nominal in phase transformer taps.

II. Omitting from B", the angle shifting effects of phase shifters.

III. Dividing Eqs. (54) and (55) by Vi and assuming Vk = 1.0 pu and also neglecting series resistance in calculating the elements of B'.

With the above assumptions, Eqs. (54) and (55) for the power flow studies become:

$$\frac{\Delta P_i}{V_t} = B'\Delta\delta \qquad \textbf{(56)}$$

$$\frac{\Delta Q_i}{V_i} = B''\Delta V \qquad \textbf{(57)}$$

In the above equations B' and B" are real and sparse and have similar structures as those of H and L respectively. Since, they contain only network admittances, they are constant and do not change during successive iterations for solution of power flow problem, they need to be evaluated only once and inverted once during the first iteration and then used in all successive iterations.

It is due to the nature of Jacobian matrices B' and B" and the sparsity of these matrices that the method is fast. In this method of power flow studies, each cycle of iteration consists of one solution for $\Delta\delta$ to update $\delta$ and one solution for $\Delta V$ to update V. The iterations are continued till $\Delta P$ and $\Delta Q$ at all load buses and $\Delta P$ at all generation buses are within prescribed (or assumed) tolerances.

# Implementation

## Load and initialize data:

```
ns=0; Vm=0; delta=0; yload=0; deltad=0;

nbus = length(busdata(:,1));
for k=1:nbus
n=busdata(k,1);
kb(n)=busdata(k,2); Vm(n)=busdata(k,3); delta(n)=busdata(k, 4);
Pd(n)=busdata(k,5); Qd(n)=busdata(k,6); Pg(n)=busdata(k,7); Qg(n) = busdata(k,8);
Qmin(n)=busdata(k, 9); Qmax(n)=busdata(k, 10);
Qsh(n)=busdata(k, 11);
    if Vm(n) <= 0  Vm(n) = 1.0; V(n) = 1 + j*0;
    else delta(n) = pi/180*delta(n);
         V(n) = Vm(n)*(cos(delta(n)) + j*sin(delta(n)));
         P(n)=(Pg(n)-Pd(n))/basemva;
         Q(n)=(Qg(n)-Qd(n)+ Qsh(n))/basemva;
         S(n) = P(n) + j*Q(n);
    end
if kb(n) == 1, ns = ns+1; else, end
nss(n) = ns;
end
```

## Find B1 Matrix:

```
for ib=1:nbus
     if kb(ib) == 0 | kb(ib) == 2
     ii = ii+1;
      jj=0;
         for jb=1:nbus
             if kb(jb) == 0 | kb(jb) == 2
             jj = jj+1;
             B1(ii,jj)=imag(Ybus(ib,jb));
             else,end
         end
     else, end
end
```

## Find B2 Matrix:

```
for ib=1:nbus
     if kb(ib) == 0
     ii = ii+1;
      jj=0;
         for jb=1:nbus
             if kb(jb) == 0
             jj = jj+1;
             B2(ii,jj)=imag(Ybus(ib,jb));
             else,end
         end
     else, end
end
B1inv=inv(B1); B2inv = inv(B2);
```

## Create Decouple Algorithm

```
for n=1:nbus
nn=n-nss(n);
J11=0;  J33=0;
   for i=1:nbr
     if nl(i) == n | nr(i) == n
        if nl(i) == n,  l = nr(i); end
        if nr(i) == n,  l = nl(i); end
        J11=J11+ Vm(n)*Vm(l)*Ym(n,l)*sin(t(n,l)- delta(n) + delta(l));
        J33=J33+ Vm(n)*Vm(l)*Ym(n,l)*cos(t(n,l)- delta(n) + delta(l));
     else , end
   end
   Pk = Vm(n)^2*Ym(n,n)*cos(t(n,n))+J33;
   Qk = -Vm(n)^2*Ym(n,n)*sin(t(n,n))-J11;
   if kb(n) == 1 P(n)=Pk; Q(n) = Qk; end   % Swing bus P
     if kb(n) == 2  Q(n)=Qk;
         Qgc = Q(n)*basemva + Qd(n) - Qsh(n);
         if Qmax(n) ~= 0
           if iter <= 20                % Between the 1th & 6th iterations
              if iter >= 10             % the Mvar of generator buses are
                if Qgc  < Qmin(n),      % tested. If not within limits Vm(n)
                Vm(n) = Vm(n) + 0.005;  % is changed in steps of 0.05 pu to
                elseif Qgc > Qmax(n),   % bring the generator Mvar within
                Vm(n) = Vm(n) - 0.005;end % the specified limits.
              else, end
           else,end
         else,end
     end
```

## Find V:

```
   if kb(n) ~= 1
   id = id+1;
     DP(id) = P(n)-Pk;
     DPV(id) = (P(n)-Pk)/Vm(n);
   end
   if kb(n) == 0
   iv=iv+1;
     DQ(iv) = Q(n)-Qk;
     DQV(iv) = (Q(n)-Qk)/Vm(n);
   end
end
Dd=-B1\DPV';
DV=-B2\DQV';
id=0;iv=0;
  for n=1:nbus
    if kb(n) ~= 1
    id = id+1;
    delta(n) = delta(n)+Dd(id); end
    if kb(n) == 0
    iv = iv+1;
    Vm(n)=Vm(n)+DV(iv); end
```

Calculate Results:

```
k=0;
V = Vm.*cos(delta)+j*Vm.*sin(delta);
deltad=180/pi*delta;
clear A; clear DC; clear DX
i=sqrt(-1);
for n = 1:nbus
     if kb(n) == 1
     S(n)=P(n)+j*Q(n);
     Pg(n) = P(n)*basemva + Pd(n);
     Qg(n) = Q(n)*basemva + Qd(n) - Qsh(n);
     k=k+1;
     Pgg(k)=Pg(n);
     elseif  kb(n) ==2
     S(n)=P(n)+j*Q(n);
     Qg(n) = Q(n)*basemva + Qd(n) - Qsh(n);
     k=k+1;
     Pgg(k)=Pg(n);
     end
yload(n) = (Pd(n)- j*Qd(n)+j*Qsh(n))/(basemva*Vm(n)^2);
end
busdata(:,3)=Vm'; busdata(:,4)=deltad';
Pgt = sum(Pg);  Qgt = sum(Qg); Pdt = sum(Pd); Qdt = sum(Qd); Qsht = sum(Qsh);
```

## Bus Out prints

In “busout” file, we just implement some print to print the outputs of each bus line and total power in bus lines so there not much to talk about here. All the outputs are calculated through the run of one of the three methods which were explained.

### Implementation

```
disp(tech)
fprintf('                     Maximum Power Mismatch = %g \n', maxerror)
fprintf('                           No. of Iterations = %g \n\n', iter)
head =['    Bus  Voltage  Angle    ------Load------    ---Generation---   Injected'
       '    No.  Mag.     Degree     MW       Mvar       MW       Mvar       Mvar '
       '                                                                          '];
disp(head)
for n=1:nbus
     fprintf(' %5g', n), fprintf(' %7.3f', Vm(n)),
     fprintf(' %8.3f', deltad(n)), fprintf(' %9.3f', Pd(n)),
     fprintf(' %9.3f', Qd(n)),  fprintf(' %9.3f', Pg(n)),
     fprintf(' %9.3f ', Qg(n)), fprintf(' %8.3f\n', Qsh(n))
end
    fprintf('      \n'), fprintf('    Total              ')
    fprintf(' %9.3f', Pdt), fprintf(' %9.3f', Qdt),
    fprintf(' %9.3f', Pgt), fprintf(' %9.3f', Qgt), fprintf(' %9.3f\n\n', Qsht)
```

## Line Flow

Current flowing from bus i towards bus k,

$$I_{ik} = [\mathbf{V}_i - \mathbf{V}_k]\mathbf{y}_{ik} + \mathbf{v}_i\mathbf{y}_{ik0} \qquad (5)$$

where Vi and Vk are the bus voltages at the buses i and k respectively which are already calculated from the power flow studies.
The power flow in the line i-k at the bus i is given as –

$$\mathbf{S}_{ik} = \mathrm{P}_{ik} + j\mathrm{Q}_{ik} = \mathbf{V}_i\mathbf{r}_{ik}^* = \mathbf{V}_i(\mathbf{v}_i^* - \mathbf{V}_k^*)\mathbf{y}_{ik}^* + \mathbf{V}_i\mathbf{v}_i^*\mathbf{y}_{ik0} \qquad \mathbf{(6)}$$

Similarly, the power flow in the line i-k at the bus k is given as –

$$\mathbf{S}_{ki} = \mathbf{v}_k(\mathbf{v}_k^* - \mathbf{v}_i^*)\mathbf{y}_{ik}^* + \mathbf{v}_k\mathbf{v}_k^*\mathbf{y}_{ki}^* \qquad (7)$$

Thus, power flows over all the lines can be computed.

The power losses in the (I – k) th line are given by sum of the power flows determined from above Eqs. (6) and (7) i.e.
power losses in the (I – k) th line = Sik + Ski.
Total transmission losses can be computed by summing all the line flows (i.e., Sik + Ski. for all i, k).
It may be noted that the slack bus power can also be determined by summing the power flows on the lines terminating at the slack bus. This concludes the power flow study for the case of P-Q buses only.

## Implementation

Calculate loss power and sum of powers for each line:

```
busprt = 1;
else, end
if nl(L)==n      k = nr(L);
In = (V(n) - a(L)*V(k))*y(L)/a(L)^2 + Bc(L)/a(L)^2*V(n);
Ik = (V(k) - V(n)/a(L))*y(L) + Bc(L)*V(k);
Snk = V(n)*conj(In)*basemva;
Skn = V(k)*conj(Ik)*basemva;
SL  = Snk + Skn;
SLT = SLT + SL;
elseif nr(L)==n  k = nl(L);
In = (V(n) - V(k)/a(L))*y(L) + Bc(L)*V(n);
Ik = (V(k) - a(L)*V(n))*y(L)/a(L)^2 + Bc(L)/a(L)^2*V(k);
Snk = V(n)*conj(In)*basemva;
Skn = V(k)*conj(Ik)*basemva;
SL  = Snk + Skn;
SLT = SLT + SL;
else, end
  if nl(L)==n | nr(L)==n
  fprintf('%12g', k),
  fprintf('%9.3f', real(Snk)), fprintf('%9.3f', imag(Snk))
  fprintf('%9.3f', abs(Snk)),
  fprintf('%9.3f', real(SL)),
      if nl(L) ==n & a(L) ~= 1
      fprintf('%9.3f', imag(SL)), fprintf('%9.3f\n', a(L))
      else, fprintf('%9.3f\n', imag(SL))
      end
  else, end
```

Print the results which is half of calculated loss:

```
SLT = SLT/2;
fprintf('   \n'), fprintf('    Total loss                          ')
fprintf('%9.3f', real(SLT)), fprintf('%9.3f\n', imag(SLT))
```

# Results

Here we are going to load the data and run each method and see the results:

## Data

First, we load our 30 bus and line data to a matlab file named 30 bus:

30 bus data:

```
busdata=[1   1    1.06    0.0     0.0   0.0    0.0  0.0   0   0       0
         2   2    1.043   0.0    21.70  12.7   40.0  50.0 -40  50      0
         3   0    1.0     0.0     2.4   1.2    0.0  0.0   0   0       0
         4   0    1.06    0.0     7.6   1.6    0.0  0.0   0   0       0
         5   2    1.01    0.0    94.2   19.0   0.0  37.0 -40  40      0
         6   0    1.0     0.0     0.0   0.0    0.0  0.0   0   0       0
         7   0    1.0     0.0    22.8   10.9   0.0  0.0   0   0       0
         8   2    1.01    0.0    30.0   30.0   0.0  37.3 -10  40      0
         9   0    1.0     0.0     0.0   0.0    0.0  0.0   0   0       0
        10   0    1.0     0.0     5.8   2.0    0.0  19.0  0   0       0
        11   2    1.082   0.0     0.0   0.0    0.0  16.2 -6  24      0
        12   0    1.0     0        11.2  7.5   0    0     0   0       0
        13   2    1.071   0        0    0.0    0    10.6 -6  24      0
        14   0    1       0       6.2   1.6    0    0     0   0       0
        15   0    1       0       8.2   2.5    0    0     0   0       0
        16   0    1       0       3.5   1.8    0    0     0   0       0
        17   0    1       0       9.0   5.8    0    0     0   0       0
        18   0    1       0       3.2   0.9    0    0     0   0       0
        19   0    1       0       9.5   3.4    0    0     0   0       0
        20   0    1       0       2.2   0.7    0    0     0   0       0
        21   0    1       0      17.5   11.2   0    0     0   0       0
        22   0    1       0       0     0.0    0    0     0   0       0
        23   0    1       0       3.2   1.6    0    0     0   0       0
        24   0    1       0       8.7   6.7    0    4.3   0   0       0
        25   0    1       0       0     0.0    0    0     0   0       0
        26   0    1       0       3.5   2.3    0    0     0   0       0
        27   0    1       0       0     0.0    0    0     0   0       0
        28   0    1       0       0     0.0    0    0     0   0       0
        29   0    1       0       2.4   0.9    0    0     0   0       0
        30   0    1       0      10.6   1.9    0    0     0   0       0];
```

Line Data:

```
linedata=[1   2   0.0192   0.0575   0.02640    1
          1   3   0.0452   0.1852   0.02040    1
          2   4   0.0570   0.1737   0.01840    1
          3   4   0.0132   0.0379   0.00420    1
          2   5   0.0472   0.1983   0.02090    1
          2   6   0.0581   0.1763   0.01870    1
          4   6   0.0119   0.0414   0.00450    1
          5   7   0.0460   0.1160   0.01020    1
          6   7   0.0267   0.0820   0.00850    1
          6   8   0.0120   0.0420   0.00450    1
          6   9   0.0      0.2080   0.0      0.978
          6  10   0         .5560   0        0.969
          9  11   0         .2080   0           1
          9  10   0         .1100   0           1
          4  12   0         .2560   0        0.932
         12  13   0         .1400   0           1
         12  14    .1231    .2559   0           1
         12  15    .0662    .1304   0           1
         12  16    .0945    .1987   0           1
         14  15    .2210    .1997   0           1
         16  17    .0824    .1923   0           1
         15  18    .1073    .2185   0           1
         18  19    .0639    .1292   0           1
         19  20    .0340    .0680   0           1
         10  20    .0936    .2090   0           1
         10  17    .0324    .0845   0           1
         10  21    .0348    .0749   0           1
         10  22    .0727    .1499   0           1
         21  22    .0116    .0236   0           1
         15  23    .1000    .2020   0           1

       22  24    .1150    .1790   0          1
       23  24    .1320    .2700   0          1
       24  25    .1885    .3292   0          1
       25  26    .2544    .3800   0          1
       25  27    .1093    .2087   0          1
       28  27     0       .3960   0       0.968
       27  29    .2198    .4153   0          1
       27  30    .3202    .6027   0          1
       29  30    .2399    .4533   0          1
        8  28    .0636    .2000   0.0214     1
        6  28    .0169    .0599   0.065      1];
```

## Methods:

Here we run the main file with “run main.m” and run each method to see the results

run main.m

```
Enter the method for load flow (1 - GS, 2 - NR, 3 - Decouple):
```

```
                                        --Line loss--
                                         MW      Mvar
```

## Gauss-Seidel

Results are attached as file Gauss-Seidel Result in a separate file.

Total loss is:

```
Total loss                              17.207   20.626
```

## Newton-Raphson

Results are attached as file Newton-Raphson Result in a separate file.

Total loss is:

```
 Total loss                              17.599   22.244
```

## Decouple

Results are attached as file Decouple Result in a separate file.

Total loss is:

```
Total loss                           17.598   22.245
```

## Conclusion

It can be seen that Losses are very similar. This is because number of busses are not really big so the drawbacks can not be seen in each method. This result could be predicted since the drawback of each method will be revealed when we have a large number of buses but on 30 bus they are almost the same.

Summery of result along with the number of iteration and elapsed times are:

```
Elapsed time is 0.087647 seconds.
                    Power Flow Solution by Fast Decoupled Method
                       Maximum Power Mismatch = 0.000919582
                             No. of Iterations = 15

Elapsed time is 0.134764 seconds.
                    Power Flow Solution by Newton-Raphson Method
                       Maximum Power Mismatch = 7.54898e-07
                             No. of Iterations = 4

Press Enter to terminate the iterations and print the results
Elapsed time is 1.563863 seconds.
                       ITERATIVE SOLUTION DID NOT CONVERGE
                       Maximum Power Mismatch = 0.00862049
                             No. of Iterations = 101
```

It can be seen that the fastest result was for Fast decupled and slowest was Gauss-Seidel.

All the results are attached in different pdf files. All the codes and report are attached along the results.

The End.

Soheil Shirvani

Power System Analysis 1

Final Project: Load Flow Analysis