MADHA ENGINEERING COLLEGE

(A Christian Minority Institution)

KUNDRATHUR, CHENNAI – 600 069

Power System Simulation Lab Manual

Name	**:**	
Subject	**:**	
Roll No.	**:**	
Semester	**:**	**Year:**

I N D E X

NAME : CLASS : **IV YEAR EEE**

REG. NO. : SEMESTER : **VII SEMESTER**

LIST OF EXPERIMENTS

S. No.	Date	Name of the Experiments	Page No.	Sign.
1		Computation of Parameters and Modeling of Transmission Lines		
2		Formation of Bus Admittance and Impedance Matrices and Solutionof Networks		
3		Load Flow Analysis - I: Solution of Load Flow and RelatedProblems using Gauss-Seidel Method		
4		Load Flow Analysis - II: Solution of Load Flow and Related Problems Using Newton-Raphson Method		
5		Symmetric and Unsymmetrical Fault Analysis		
6		Transient and Small Signal Stability Analysis: Single-Machine Infinite Bus System		
7		Transient Stability Analysis of Multi machine Power Systems		
8		Economic Dispatch in Power Systems		
9		Load – Frequency Dynamics of Single- Area and Two-Area PowerSystems		
10		Electromagnetic Transients in Power Systems		

Lab Incharge

EXP.NO:1(a)
DATE:

COMPUTATION OF PARAMETERS

AIM:

To determine the positive sequence line parameters L and C per phase per kilometer of a single phase, three phase single and double circuit transmission lines for different conductor arrangements.

SOFTWARE REQUIRED:

MATLAB

THEORY:

Transmission line has four parameters – resistance, inductance, capacitance and conductance. The inductance and capacitance are due to the effect of magnetic and electric fields around the conductor. The resistance of the conductor is best determined from the manufactures data, the inductances and capacitances can be evaluated using the formula.

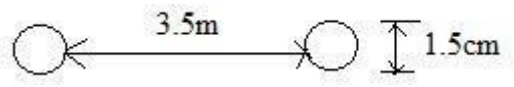
ARRANGEMENT	INDUCTANCE	CAPACITANCE
SINGLE PHASE SYSTEM	$L_c=2*10^{-7}\ln(D/r')$ $r'=0.7788r.$ $L_{loop}=2L_c$	$C_c=2\pi\xi_0/\ln(D/r)$ $C_{loop}=C_c/2.$
3PHASE SYMMETRICAL SYSTEM	$L_c=L_{ph}=2*10^{-7}\ln(D/r')$	$C_c=C_{ph}=2\pi\xi_0/\ln(D/r)$
3PHASE UNSYMMETRICAL TRANSPOSED SYSTEM	$L_c=L_{ph}=2*10^{-7}\ln(D_{eq}/r')$ $D_{eq}=(D_{AB}*D_{BC}*D_{CA})^{(1/3)}$	$C_c=C_{ph}=2\pi\xi_0/\ln(D_{eq}/r)$
3PHASE UNSYMMETRICAL UNTRANSPOSED SYSTEM	$L_a=2*10^{-7}[\ln\sqrt{(D_{ab}*D_{ca})}/r'$ $+j\sqrt{3}*\ln\sqrt{(D_{ab}/D_{ca})}$ $L_b=2*10^{-7}[\ln\sqrt{(D_{bc}*D_{ab})}/r'$ $+j\sqrt{3}*\ln\sqrt{(D_{bc}/D_{ab})}$ $L_c=2*10^{-7}[\ln\sqrt{(D_{ca}*D_{bc})}/r'$ $+j\sqrt{3}*\ln\sqrt{(D_{ca}/D_{bc})}$	----------

3PHASE SYMMETRICAL DOUBLECIRCUIT SYSTEM	$L_c=2*10^{-7}\ln(\sqrt{3}D/2r')$ $L_{ph}=L_c/2$	$C_c=2\pi\xi_0/\ln(\sqrt{3}D/2r)$ $C_{ph}=C_c*2.$
3PHASE UNSYMMETRICAL TRANSPOSED SYSTEM WITH VERTICAL PROFILE	$L_c=2*10^{-7}\ln[2^{(1/3)}(D/r')(m/n)^{(2/3)}]$ $L_{ph}=L_c/2$	$C_c=2\pi\xi_0/\ln(2^{(1/3)}(D/r)(m/n)^{(2/3)}]$ $C_{ph}=C_c*2.$
3PHASE UNSYMMETRICAL TRANSPOSED DOUBLE CIRCUIT	$L_c=2*10^{-7}\ln(D_m/D_s)$ $L_{ph}=L_c/2$	$C_c=2\pi\xi_0/\ln[i^2m^2jh/r^3n^3d]^{(1/3)}$ $C_{ph}=C_c*2.$
3PHASE LINE WITH BUNDELED CONDUCTORS	$L_c=2*10^{-7}\ln(D_m/D_s)$ $L_{ph}=L_c/2$ $D_m=(D_{AB}*D_{BC}*D_{CA})^{(1/3)}$ $D_s=(D_{SA}*D_{SB}*D_{SC})^{(1/3)}$	--------

PROCEDURE:

1. Enter the command window of the MATLAB.
2. Create a new M – file by selecting File - New – M – File
3. Type and save the program in the editor window.
4. Execute the program by either pressing Tools – Run.
5. View the results.

1. (a) Calculate the loop inductance and capacitance of a 1 phase line with two parallel conductors spaced 3.5m apart. The diameter of each conductor is 1.5 cm.



Manual Calculation:

1(a) CALCULATION OF INDUCTANCE AND CAPACITANCE OF SINGLE PHASE LINE

PROGRAM:

```
clc;
clear all;
disp('CALCULATION OF INDUCTANCE AND CAPACITANCE OF 1 PHASE LINE');
d=input('Enter diameter in cm:');
r=d/2;
rad=r*10^(-2);
D=input('Enter distance between conductors in m:');
r1=rad*0.7788; L=4*10^(-
7)*log(D/r1); C=(pi*8.854*10^(-
12))/(log(D/rad));
disp('INDUCTANCE(in H/m):');
disp(L);
disp('CAPACITANCE(in F/m):');
disp(C);
```

OUTPUT:

EXP.NO:1 (b)
DATE:

MODELLING OF TRANSMISSION LINES

AIM:

To understand modeling and performance of short, medium and long transmission lines.

SOFTWARE REQUIRED:

MATLAB

FORM ULAE:

$Vs=AV_R+BI_R$

$Is=CV_R+DI_R$

TYPE	METHOD	ABCD PARAMETERS
Short	----------	A=D=1; B=Z; C=0.
Medium	Nominal T Method	A=D=1+YZ/2; B=Z(1+YZ/4); C=Y;
	Nominal π Method	A=D=1+YZ/2; B=Z; C=Y(1+YZ/4);
Long	Rigorous Method	A=D=cos h(γℓ); B=Z_c sin h(γℓ); C=1/Z_c sin h(γℓ); γ=√(ZY); Z_c=√(Z/Y);
	Equivalent π Method	A=D=1+YZ/2; B=Z; C=Y(1+YZ/A); Z=Z sin h(γℓ)/ γℓ; Y=Y tan h(γℓ/2)/ (γℓ/2); γ=√(ZY); Z_c=√(Z/Y);
	Equivalent T Method	A=D=1+YZ/2; B=Z; C=Y(1+YZ/A); Z=Z tan h(γℓ/2)/ (γℓ/2); Y=Y sin h(γℓ)/ γℓ; γ=√(ZY); Z_c=√(Z/Y);

PROCEDURE:

1. Enter the command window of the MATLAB.
2. Create a new M – file by selecting File - New – M – File
3. Type and save the program in the editor window.
4. Execute the program by either pressing Tools – Run.
5. View the results.

1. An overhead 3 phase transmission line delivers 4000KW at 11 KV at 0.8 pf lagging. The resistance and reactance of each conductor are 1.5Ω and 4Ω per phase. Determine the line performance.

Manual Calculation:

SHORT TRANSMISSION LINE

PROGRAM:

```
clc;
clear all;
R=input('Resistance                    :');
XL=input('Inductive Reactance                :');
XC=input('Capacitive Reactance                :');
G=input('Conductance                    :');
length=input('Length of Transmission Line          :');
f=input('Frequency                    :');
Z1=   (R+j*XL)*length;
Y1=  (G+j*XC)*length;
A = 1;
B = Z1;
C = 0;
D =1;
TM = [ A B; C D ];
VRL=input('ENTER RECEIVEING END VOLTAGE            :');
VRP=VRL/(sqrt(3));
PR = input('ENTER RECEIVING END LOAD IN MW          :');
Pf=input('ENTER THE RECEIVING END LOAD POWER FACTOR    :');
h=acos(Pf);
SR=PR/Pf;
SR=SR*(cos(h)+j*sin(h));
QR=imag(SR);
IR=conj(SR)/(3*conj(VRP));
SM=TM*[VRP;IR];
VS=SM(1,1);
IS=SM(2,1);
Pfs=cos(angle(VS)-angle(IS));
SS=3*VS*conj(IS);
VSA=angle(VS)*(180/pi);
ISA=angle(IS)*(180/pi);
VS=sqrt(3)*abs(VS);
IS=abs(IS)*1000;
VREG=((VS/(abs(TM(1,1)))-VRL)/VRL)*100;
PS=real(SS);
QS=imag(SS);
eff=PR/PS*100;
PL=PS-PR;
```

```
QL=QS-QR;
Z1
Y1
TM
fprintf('SENDING END LINE VOLTAGE %g at %g degrees \n',VS,VSA);
fprintf('SENDING END LINE CURRENT %g at %g degrees \n',IS,ISA);
fprintf('SENDING END POWER FACTOR %g\n',Pfs);
fprintf('SENDING END REAL POWER %g\n',PS);
fprintf('SENDING END REACTIVE POWER %g\n',QS);
fprintf('PERCENTAGE VOLTAGE REGULATION %g\n',VREG);
fprintf('REAL POWER LOSS %g\n',PL);
fprintf('REACTIVE POWER LOSS %g\n',QL);
fprintf('EFFICIENCY %G', eff);
```

OUTPUT:

RESULT

EXP NO: 2

DATE:

FORMATION OF BUS ADMITTANCE AND IMPEDANCE MATRICES AND SOLUTION OF NETWORKS

AIM:

To develop a program to obtain Y_{bus} matrix for the given networks by the method of inspection.

FORMATION OF Y-BUS MATRIX

Generalized [Y-Bus] = $\begin{bmatrix} Y_{ii} & Y_{ij} \\ Y_{ji} & Y_{jj} \end{bmatrix}$

Each admittance Y_{ii} (i =1,2,……n) is called the self admittance or driving point admittance of bus I and equals the sum of all admittances terminating on the particular bus.

Each off-diagonal term Y_{ij} (i,j = 1,2…n; j≠i) is the transfer admittance between buses I and j, n=total number of buses. Further, $Y_{ij} = Y_{ji}$

SIMULATION

In this exercise matrix, Z-Bus for the system is developed by first forming the Y_{bus} and then inverting it to get the Z-Bus matrix. The generator and transformer impedances are taken into account.

Y_{bus} is a sparse matrix, Z-Bus is a full matrix, i.e., zero elements of Y_{bus} become non-zero values in the corresponding Z-Bus elements. The bus impedance matrix is most useful for short circuit studies.

ALGORITHM

Step (1): Initialize [Y-Bus] matrix that is replace all entries by zero.

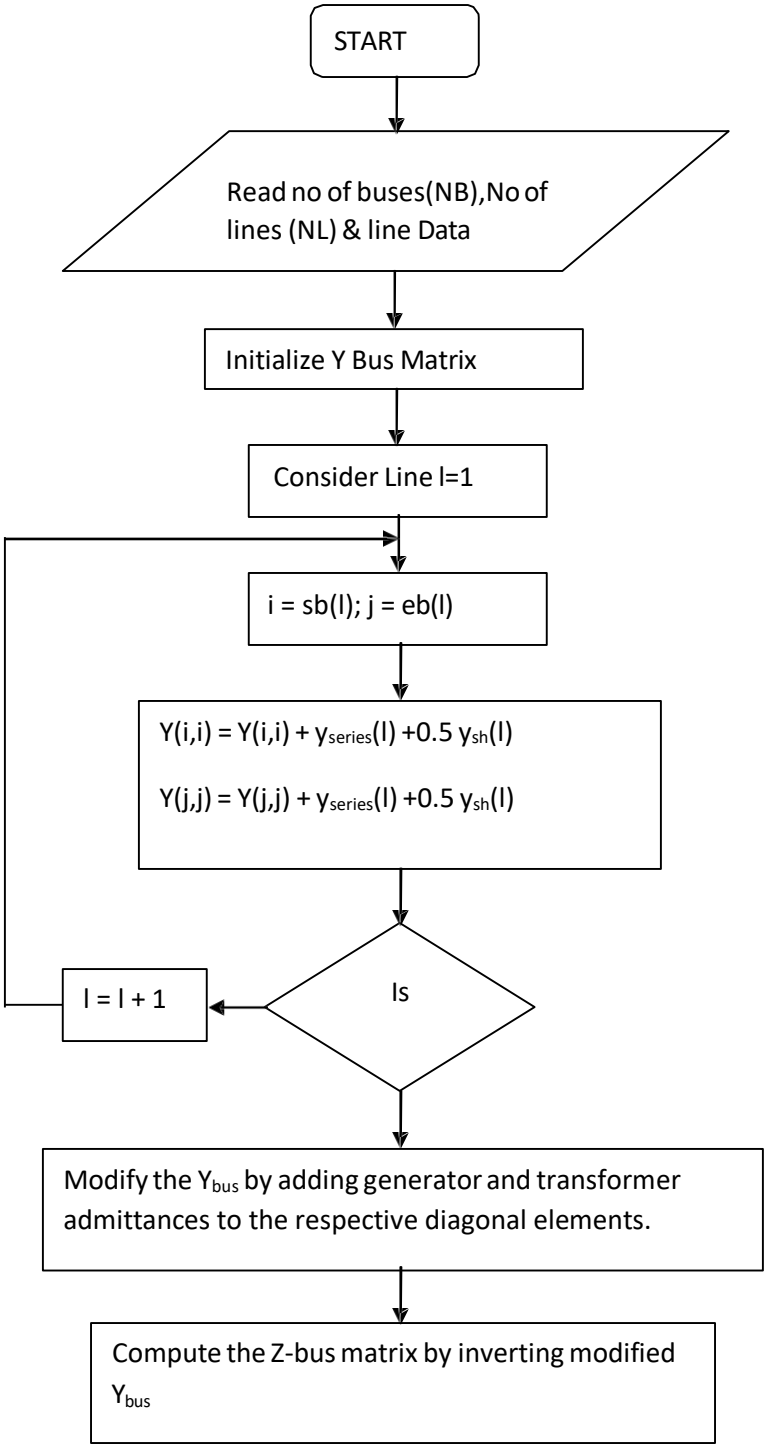
$Y_{ij} = Y_{ij}-y_{ij} = Y_{ji}$ = off diagonal element

Step (2): Compute $Y_{ii} = \sum_{j=1}^{n} y_{ij}$ = diagonal element.

Step (3) : Modify the Y_{bus} matrix by adding the transformer and the generator admittances

to the respective diagonal elements of Y- bus matrix.

Step (4) : Compute the Z-Bus matrix by inverting the modified Y_{bus} matrix.



1. The [Y-Bus] matrix is formed by inspection method for a four bus system. The line data and is given below.

LINE DATA

Line Number	SB	EB	Series Impedance (p.u)	Line charging Admittance (p.u)
1	1	2	0.10 + j0.40	j0.015
2	2	3	0.15 + j0.60	j0.020
3	3	4	0.18 + j0.55	j0.018
4	4	1	0.10 + j0.35	j0.012
5	4	2	0.25 + j0.20	j0.030

FORMATION OF Y-BUS BY THE METHOD OF INSPECTION

PROGRAM:

```
clc;
clear all;
n=input('Enter number of buses');
l=input('Number of lines');
s=input('1.Impedance or 2:Admittance');
ybus=zeros(n,n);
lc=zeros(n,n);

for i=1:l
    a=input('Starting bus:');
    b=input('Ending bus:');
    t=input('Admittance or Impedance of line:');
    lca=input('Line charging admittance:');
    if(s==1)
        y(a,b)=1/t;
    else
        y(a,b)=t;
    end
    y(b,a)=y(a,b);
    lc(a,b)=lca;
    lc(b,a)=lc(a,b);
end

for i=1:n
    for j=1:n
        if i==j
            for k=1:n
                ybus(i,j)=ybus(i,j)+y(i,k)+lc(i,k)/2;
            end
            else
            ybus(i,j)=-y(i,j);
        end
        ybus(j,i)=ybus(i,j);
    end
end
ybus
```

OUTPUT:

ybus =

FORMATION OF BUS IMPEDANCE MATRICES

ALGORITHM

Step (1): Initialize [Y-Bus] matrix, that is replace all entries by zero

$Y_{ij} = Y_{ij} - y_{ij} = Y_{ji}$ = off diagonal element

Step (2): Compute $Y_{ii} = \sum_{j=1}^{n} y_{ij}$ = diagonal element

Step (3): Modify the Y_{bus} matrix by adding the transformer and the generator admittancesto the respective diagonal elements of Y- bus matrix.

Step (4): Compute the Z-Bus matrix by inverting the modified Y_{bus} matrix.

1. Determine Z bus for the given network using direct inspection method.

LINE DATA

Line Number	SB	EB	Series Impedance (p.u)	Line charging Admittance (p.u)
1	1	2	0.10 + j0.40	j0.015
2	2	3	0.15 + j0.60	j0.020
3	3	4	0.18 + j0.55	j0.018
4	4	1	0.10 + j0.35	j0.012
5	4	2	0.25 + j0.20	j0.030

FORMATION OF Z-BUS BY THE METHOD OF INSPECTION

PROGRAM:

```
clc;
clear all;
n=input('Enter number of buses');
l=input('Number of lines');
s=input('1.Impedance or 2:Admittance');

for i=1:l
    a=input('Starting bus:');
    b=input('Ending bus:');
    t=input('Admittance or Impedance of line:');
    lca=input('Line charging admittance:');
    if(s==1)
        y(a,b)=1/t;
    else
        y(a,b)=t;
    end
    y(b,a)=y(a,b);
    lc(a,b)=lca;
    lc(b,a)=lc(a,b);
end

ybus=zeros(n,n);
for i=1:n
    for j=1:n
        if i==j
            for k=1:n
                ybus(i,j)=ybus(i,j)+y(i,k)+lc(i,k)/2;
            end
            else
            ybus(i,j)=-y(i,j);
        end
        ybus(j,i)=ybus(i,j);
    end
end
zbus=(ybus)^(-1);
zbus
```

OUTPUT:

ybus =

zbus =

RESULT:

EXP NO: 3

DATE:

LOAD FLOW ANALYSIS BY GAUSS – SEIDAL METHOD

AIM:

To carryout load flow analysis of the given power system by Gauss – Seidal method.

ALGORITHM:

Step 1: Assume a flat voltage profile of 1+j0 for all buses except the slack bus. The voltage of slack bus is the specified voltage and it is not modified in any iteration.

Step 2: Assume a suitable value of ε called convergence criterion. Here ε is a specified change in bus voltage that is used to compare the actual change in bus voltage k^{th} and $(k+1)^{th}$ iteration.

Step 3: Set iteration count, k=0 and assumed voltage profile of the buses is denoted as $V_0^1, V_0^2, V_0^3, \ldots V_0^n$ except slack bus.

Step 4: Set bus count, p=1

Step 5: Check for slack bus. If it is a slack bus then go to step-12,otherwise go to next step.

Step 6: Check for generator bus. If it is a generator bus go to next step, otherwise (i.e., if it is load bus) go to step=9.

Step 7: Temporarily set $\left|V_p^k\right| = \left|V_p\right|_{spec}$ and phase of V_p^k as the k^{th} iteration value if the bus-p is a generator bus where $\left|V_p\right|_{spec}$ is the specified magnitude of voltage for bus – p. Then calculate the reactive power of the generator bus using the following equation.

$$Q_{p,cal}^{k+1} = (-1) * \left(V^k\right) \left[\sum Y \quad V^{k+1} + \sum \; Y \quad V^k \right]$$

$$\mathrm{Im}\left\{ \left(\right)^* \left[\sum_{q=1}^{p-1} Y_{pq} \; V_q^{k+1} + \sum_{q=p}^{n} Y_{pq} \; V_q^k \right] \right\}$$

The calculated reactive power may be within specified limits or it may violate the limits.

If the calculated reactive power is within the specified limits then consider this bus as generator bus and set $Q_p = Q^{k+1}$ for this iteration and go to step-8.

If the calculated reactive power violates the specified limit for reactive power then treat this bus as a load bus.The magnitude of the reactive power at this bus will correspond to the limit it has violated.

i.e., if $Q_{p,cal}^{k+1} < Q_{p,\min}$ then $Q_p = Q_{p,\min}$

(or) $Q_{p,cal}^{k+1} > Q_{p,\min}$ then $Q_p = Q_{p,\max}$

Since the bus is treated as load bus, take actual value of V_p^k for $(k+1)^{th}$ iteration.i.e. V_p^k Need not be replaced by $|V_p|_{spec}$ when the generator bus is treated as load bus. Go to step9.

Step 8: For generator bus the magnitude of voltage does not change and so for all iteration the magnitude of bus voltage is the specified value. The phase of the bus voltage can be as shown below.

$$V_{p,temp}^{k+1} = \frac{1}{Y_{pp}}\left\{\frac{P_p - jQ_p}{\left(V_p^k\right)^*} - \sum_{q=1}^{p-1} Y_{pq} V_q^{k+1} - \sum_{q=p+1}^{n} Y_{pq} V_q^k\right\}$$

$$\delta_p^{k+1} = \tan^{-1}\left[\frac{\mathrm{Im}\left(V_{p,temp}^{k+1}\right)}{\mathrm{Real}\left(V_{p,temp}^{k+1}\right)}\right]$$

Now that $(k+1)^{th}$ iteration voltage of the generator bus is given by

$$V_p^{k+1} = |V_p|_{spec} \angle \delta_p^{k+1}$$

After calculating V_p^{k+1} for generator buses, go to step -11.

Step 9: For the load bus the $(k+1)^{th}$ iteration value of load bus –p voltage, V_p^{k+1} can be calculated using the following equation.

$$V_p^{k+1} = \frac{1}{Y_{pp}}\left\{\frac{P_p - jQ_p}{\left(V_p^k\right)^*} - \sum_{q=1}^{p-1} Y_{pq} V_q^{k+1} - \sum_{q=p+1}^{n} Y_{pq} V_q^k\right\}$$

Step 10: An acceleration factor, α can be used for faster convergence. If the acceleration factor is specified then modify the $(k+1)^{th}$ iteration value of bus-p voltage using the following equation.

$$V_{p,acc}^{k+1} = V_p^k + \alpha(V_p^{k+1} - V_p^k)$$

Then set , $V_p^{k+1} = V_{p,acc}^{k+1}$

Step 11: Calculate the change in the bus-p voltage, using the relation,

$$\Delta V_p^{k+1} = V_p^{k+1} - V_p^k$$

Step 12: Repeat the steps 5 to 11 until all the bus voltages have been calculated. For this increment the bus count by1 and go to step-5, until the bus count is n.

Step 13: Find out the largest of the absolute value of the change in voltage.

i.e., Find the largest among $\left|\Delta V_1^{k+1}\right|$, $\left|\Delta V^{k+1}\right|$, ….., $\left|\Delta V_{2^n}^{k+1}\right|$ Let this be the largest change $\left|\Delta V_{\max}\right|$ .Check whether this largest change $\left|\Delta V_{\max}\right|$ is less than the prescribed tolerance ε .If $\left|\Delta V_{\max}\right|$ is less than ε then move to the next step. Otherwise increment the iteration count and go to step-4.

Step 14: Calculate the line flows and slack bus power using the bus voltages.

PROBLEM:

The system data for a load flow solution are given below. Determine the voltages by Gauss – Seidal method

Line admittances

Bus code	Admittance
1 – 2	2 – j8
1 – 3	1 – j4
2 – 3	0.666 – j2.664
2 – 4	1 – j4
3 – 4	2 – j8

Bus Specifications

Bus code	P	Q	V	Remarks
1	-	-	1.06	Slack
2	0.5	0.2	-	PQ
3	0.4	0.3	-	PQ
4	0.3	0.1	-	PQ

PROGRAM

```
clc;
clear all;
n=input ('no of buses');
l=input ('no of lines');
s=input ('impedance 1 or admittance 2');
for i=1:l
   a=input ('starting bus');
   b=input ('ending bus');
   t=input ('admittance or impedance value');
   if s==1
      y(a,b)=1/t;
   else
      y(a,b)=t;
   end
   y(b,a)=y(a,b);
end
ybus=zeros(n,n);
for i=1:n
   for j=1:n
   if i==j
      for k=1:n
         ybus(i,j)=ybus(i,j)+y(i,k);
      end
   else
      ybus(i,j)=-y(i,j);
   end
   ybus(j,i)=ybus(i,j);
end
end

ybus
p=zeros(1,n);
q=zeros(1,n);
v=zeros(1,n);
pv=input ('no of pv buses');
pq=input('no of pq buses');
s=input ('slack bus number');
v(s)=input ('slack bus voltage');
acc=input ('accleration factor');
```

```
for i=1:pv
   b(i)=input('pv bus number');
   p(b(i))=input('real power');
   v(b(i))=input ('voltage value');
   qmin(b(i))=input ('min value of q');
   qmax(b(i))=input ('max value of q');
end
for i=1:pq
   c(i)=input('pq bus number');
   p(c(i))=input('real power');
   p(c(i))=-p(c(i));
   q(c(i))=input ('reactive power');
   q(c(i))=-q(c(i));
   v(c(i))=1+0i;
end
e=v;
e
enew(s)=v(s);
it=0;
yy=zeros(1,n);
for ii=1:n
   ypq(ii)=0;
   if ii~=s
      flag=0;
      gen=0;
      for j=1:pv
         if ii==b(j)
            flag=1;
         end
      end
      if flag==1
         for k=1:n
            yy(ii)=yy(ii)+ybus(ii,k)*v(k);
         end
         qcal(ii)=-imag(conj(v(ii))*yy(ii));
         if qcal(ii)<qmin(ii)
qcal(ii)=qmin(ii);
elseif qcal(ii)>qmax(ii)
   qcal(ii)=qmax(ii);
   else
      qcal(ii)=qcal(ii);
      gen=1;
      end
      else
         qcal(ii)=q(ii);
         end
```

```
qcal(ii)=qcal(ii)*sqrt(-1);
for k=1:n
   if k~=ii
      ypq(ii)=ypq(ii)+ybus(ii,k)*e(k);
   end
end
enew(ii)=(((p(ii)-qcal(ii))/conj(e(ii)))-ypq(ii))/ybus(ii,ii);
dele(ii)=enew(ii)-e(ii);
enew(ii)=e(ii)+acc*dele(ii);

if gen==1 ang=angle(enew(ii));
   enew(ii)=v(ii)*cos(ang)+v(ii)*sin(ang)*sqrt(-1);
end
  e(ii)=enew(ii);
  end
  end
  disp('voltages');
  enew
```

OUTPUT:

ybus =

RESULT:

EXP NO: 4

DATE:

LOAD FLOW ANALYSIS BY NEWTON - RAPHSON METHOD

AIM:

To carryout load flow analysis of the given power system by Newton raphson method.

ALGORITHM:

Step-1: Assume a flat voltage profile 1 + j0 for all buses (nodes) except the slack bus. The voltage of the slack bus is the specified voltage and it is not modified in any iteration.

Step-2: Assume a suitable value of ε called convergence criterion. Hence ε is a specified change in the residue that is used to compare the critical residues (Δ P and Δ Q or Δ V) at the end of each iteration.

Step-3: Set iteration count k = 0, and assumed voltage profile of the buses are denoted as V10, V20 …Vn0 except slack bus.

Step-4: Set bus count p = 1.

Step-5: Check for slack bus. If it is a slack bus then go to Step 13, otherwise go to next step.

Step-6: Calculate the real and reactive power of bus-p using the following equation.

$$P_i = \sum_{k=1}^{n} |V_i||V_k||Y_{ik}|\cos(\theta_{ik} - \delta_i + \delta_k)$$

$$Q_i = -\sum_{k=1}^{n} |V_i||V_k||Y_{ik}|\sin(\theta_{ik} - \delta_i + \delta_k)$$

Step-7: Calculate the change in real power, change in real power, $\Delta P_k = Pp_{,spec} - P_{pk}$; where $Pp_{,spec}$ = Specified real power for bus-p.

Step-8: Check for Generator bus. If it is a Generator bus go to next step, otherwise go to Step 12.

Step-9: Check for reactive power limit violation of Generator buses. For this compare the calculated reactive power Qpk with specified limits. If the limit is violated go to Step 11, otherwise go to next step.

Step-10: If the calculated reactive power is within the specified limits then consider this bus as Generator bus. Now calculate the voltage residue (change in voltage) using the following equation.

$|\Delta V_{pk}|^2 = |V_p|_{spec}^2 - |V_{pk}|^2$ where $|V_p|_{spec}$ = specified voltage.

Step-11: If the reactive power limit is violated then treat this bus as a load bus. Now the specified reactive power for this bus will correspond to the limit violated.

i.e., if $Q_{pk} < Q_p$, min then $Qp_{, spec} = Qp_{, min}$

(Or) if $Q_{pk} > Q_p$, min then $Qp_{, spec} = Qp_{, max}$

Step-12: Calculate the change in reactive power for load bus (or for the Generator bus treated as load bus). Change in reactive power, $\Delta Qp_k = |Qp_{, spec}| - Qp_k$

Step-13: Repeat steps 5 to12 until all residues (change in P and Q or V) are calculated. For this increment the bus count by 1 and go to Step 5, until the bus count is n.

Step-14: Determine the largest of the absolute value of the residue (i.e., find the largest among ΔP_k,

ΔQ_k or $|\Delta Vp_k|^2$. Let this largest change be ΔE.

Step-15: Compare ΔE and ε. If $\Delta E < \varepsilon$ then to Step 20, If $\Delta E > \varepsilon$ go to next step.

Step-16: Determine the elements of Jacobian matrix (**J**) by partially differentiating the load flow equations and evaluating the equation using Kth iteration values.

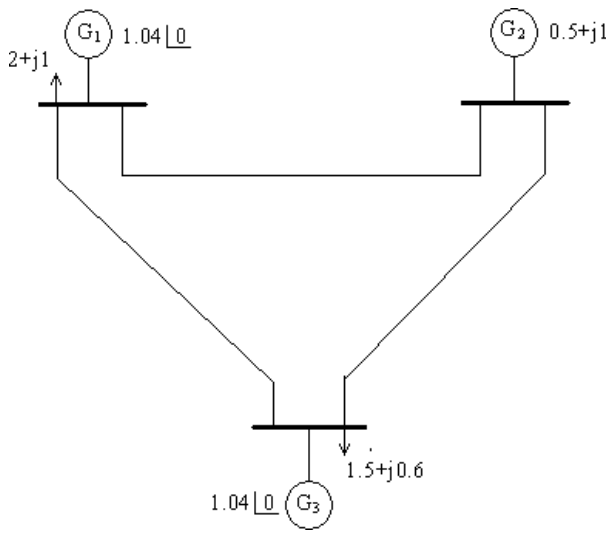
Step-17: Calculate the increments in real and reactive part of voltages.

Step-18: Calculate the new bus voltage.

Step-19: Advance the iteration count, i.e., $k = k + 1$ and go to Step 4.

Step-20: Calculate the line flows.

PROBLEM:



Consider the 3 bus system each of the 3 line bus a series impedance of 0.02 + j0.08 p.u and a total shunt admittance of j0.02 pu.The specified quantities at the buses are given below: Find the voltages in each bus for the given system using Newton-Raphson Method

Bus	Real load demand, P_D	Reactive Load demand, Q_D	Real power generation,P_G	Reactive Power Generation,Q_G	Voltage Specified
1	2	1	-	-	V_1=1.04
2	0	0	0.5	1	Unspecified
3	1.5	0.6	0	Q_{G3} = ?	V_3 = 1.04

PROGRAM:

```
clc;
basemva=100;
accuracy=0.001;
accel=1.8;
maxiter=100;
busdata=[1  1  1.04  0     0      0    0    0    0   0   0
         2  0  1       0    0.5    1    0    0    0   0   0
         3  2  1.04    0    0      0    1.5  0.6  0   0   0   ];
linedata=[ 1    2  0.02    0.08    0.01    1
           1    3  0.02    0.08    0.01    1
           2    3  0.02    0.08    0.01    1];
lfybus
lfnewton
busout
lineflow
```

Load Flow Y 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
clear Pgg
```

Load Flow Newton

```
ns=0; ng=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
end
for k=1:nbus
if kb(k) == 1, ns = ns+1; else, end
if kb(k) == 2 ng = ng+1; else, end
ngs(k) = ng;
nss(k) = ns;
end
Ym=abs(Ybus); t = angle(Ybus);
m=2*nbus-ng-2*ns;
maxerror = 1; converge=1;
iter = 0;
% Start of iterations
clear A  DC   J  DX
while maxerror >= accuracy & iter <= maxiter % Test for max. power mismatch
for i=1:m
for k=1:m
   A(i,k)=0;       %Initializing Jacobian matrix
end, end
iter = iter+1;
for n=1:nbus
nn=n-nss(n);
lm=nbus+n-ngs(n)-nss(n)-ns;
J11=0; J22=0; J33=0; J44=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));
        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
    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;
          if Qmax(n) ~= 0
            Qgc = Q(n)*basemva + Qd(n) - Qsh(n);
            if iter <= 7                  % Between the 2th & 6th iterations
               if iter > 2                % the Mvar of generator buses are
                 if Qgc  < Qmin(n),       % tested. If not within limits Vm(n)
                 Vm(n) = Vm(n) + 0.01;    % is changed in steps of 0.01 pu to
                 elseif Qgc  > Qmax(n),   % bring the generator Mvar within
                 Vm(n) = Vm(n) - 0.01;end % the specified limits.
               else, end
            else,end
          else,end
      end
    if kb(n) ~= 1
      A(nn,nn) = J11;  %diagonal elements of J1
      DC(nn) = P(n)-Pk;
    end
    if kb(n) == 0
      A(nn,lm) = 2*Vm(n)*Ym(n,n)*cos(t(n,n))+J22;  %diagonal elements of J2
      A(lm,nn)= J33;        %diagonal elements of J3
      A(lm,lm) =-2*Vm(n)*Ym(n,n)*sin(t(n,n))-J44;  %diagonal of elements of J4
      DC(lm) = Q(n)-Qk;
    end
end
DX=A\DC';
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
  maxerror=max(abs(DC));
     if iter == maxiter & maxerror > accuracy
   fprintf('\nWARNING: Iterative solution did not converged after ')
   fprintf('%g', iter), fprintf(' iterations.\n\n')
   fprintf('Press Enter to terminate the iterations and print the results \n')
   converge = 0; pause, else, end

end

if converge ~= 1
   tech= ('                         ITERATIVE SOLUTION DID NOT CONVERGE'); else,
   tech=('                     Power Flow Solution by Newton-Raphson Method');
```

```
end
V = Vm.*cos(delta)+j*Vm.*sin(delta);
deltad=180/pi*delta;
i=sqrt(-1);
k=0;
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);
     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);
  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);
```

busout

```
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)
```

lineflow

```
SLT = 0;
fprintf('\n')
fprintf('                              Line Flow and Losses \n\n')
```

```
fprintf('      --Line--  Power at bus & line flow     --Line loss--  Transformer\n')
fprintf('      from to     MW      Mvar     MVA        MW      Mvar       tap\n')

for n = 1:nbus
busprt = 0;
   for L = 1:nbr;
       if busprt == 0
       fprintf('   \n'), fprintf('%6g', n), fprintf('      %9.3f', P(n)*basemva)
       fprintf('%9.3f', Q(n)*basemva), fprintf('%9.3f\n', abs(S(n)*basemva))

       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
  end
end
SLT = SLT/2;
fprintf('   \n'), fprintf('    Total loss                          ')
fprintf('%9.3f', real(SLT)), fprintf('%9.3f\n', imag(SLT))
clear Ik In SL SLT Skn Snk
```

OUTPUT:

RESULT:

Thus the load flow analysis of the given power system by Newton – Raphson method was performedfor the given problem using Matlab-Power Tool Software.

EXP NO: 5

DATE:

SHORT CIRCUIT ANALYSIS

AIM:

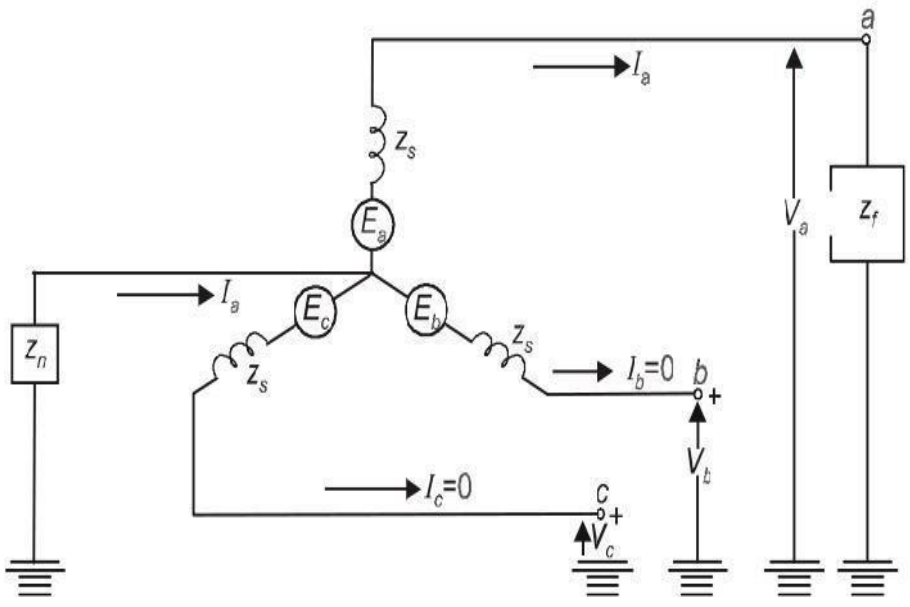
To become familiar with modeling and analysis of power systems under faulted condition and to compare the fault level, post-fault voltages and currents for different types of faults, both symmetric and unsymmetric.

OBJECTIVES

To conduct fault analysis on a given system using software available and obtain fault analysis report with fault level and current at the faulted point and post-fault voltages and currents in the network for the following faults.

1. Line-to-Ground
2. Line-to-Line
3. Double Line-to-Ground

SINGLE LINE-TO-GROUND FAULT

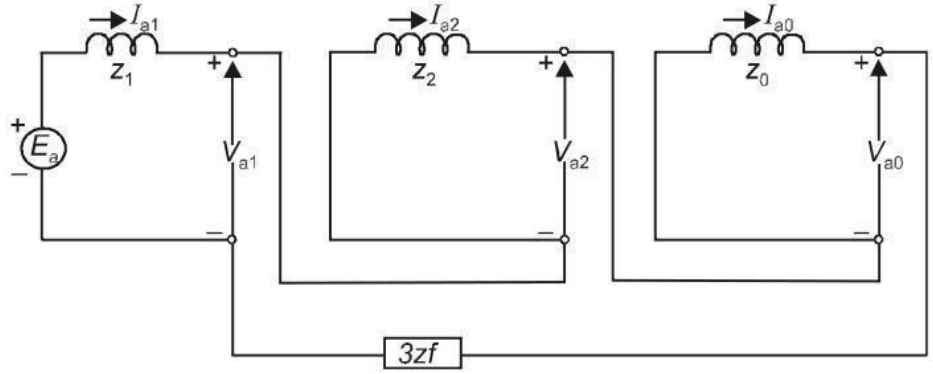


$I_b = 0$

$I_c = 0$

$V_a = Z_f I_a$

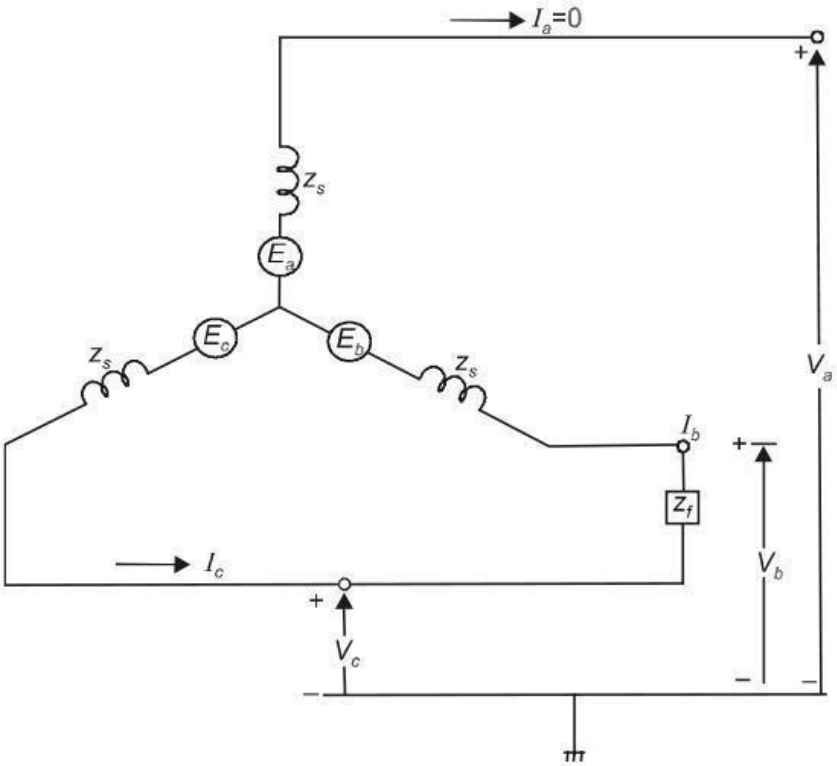
Sequence Network of Single line-to-ground-fault



$$I_a = 3I_{a1} = \frac{3E_a}{(Z_1 + Z_2 + Z_0) + 3Z_f}$$

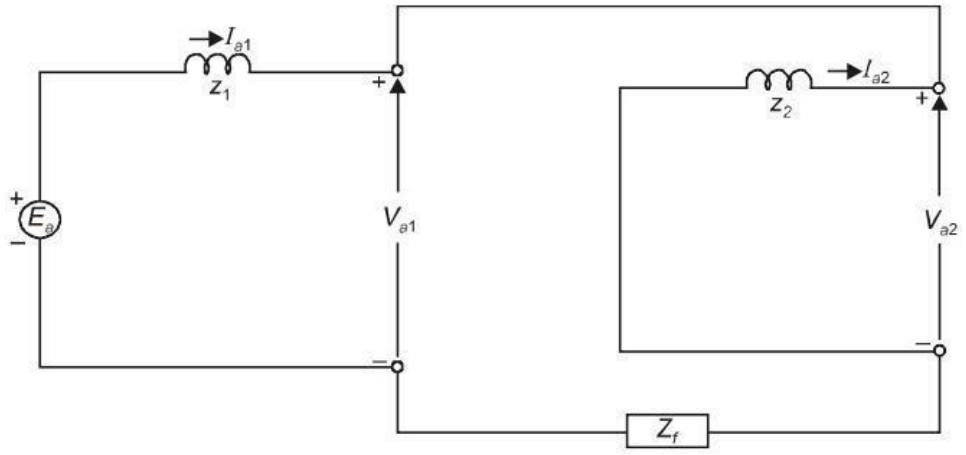
Fault Current

LINE-TO-LINE FAULT



$$V_b - V_c = Z_f . I_b$$
$$I_b + I_c = 0$$
$$I_a = 0$$

Sequence Network of Line-to-Line Fault



$$I_{a1} = \frac{E_a}{(Z_1 + Z_2 + Z_f)}$$

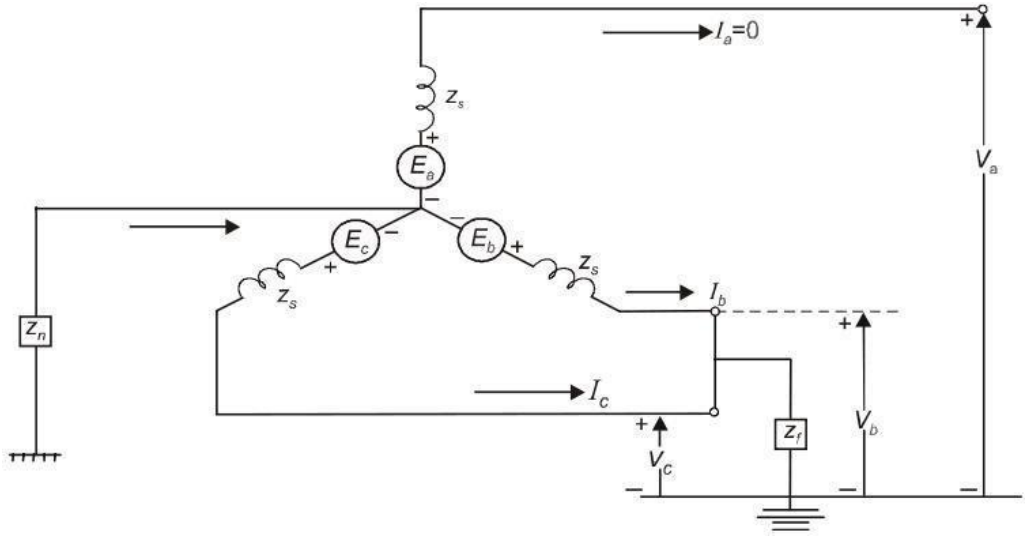
$$I_b = -I_c = \frac{-j\sqrt{3}\,E_a}{(Z_1 + Z_2 + Z_f)}$$

DOUBLE LINE-TO-GROUND FAULT

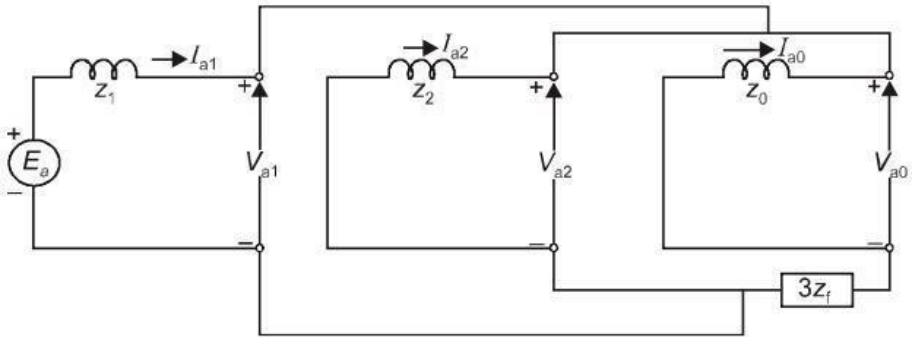
$$I_a = 0$$

$$I_{a1} + I_{a2} + I_{a0} = 0$$

$$V_b = V_c = (I_b + I_c)\,Z_f = 3Z_f I_{a0}$$



Sequence of Double line-to-ground fault



$$I_{a0} = \frac{-(E_a - Z_1 I_{a1})}{(Z_0 + 3Z_f)}$$

$$I_{a1} = \frac{E_a}{Z_1 + \dfrac{Z_2(Z_0 + 3Z_f)}{(Z_2 + Z_0 + 3Z_f)}}$$

$$I_{a2} = \frac{-(E_a - Z_1 I_{a1})}{Z_2}$$

PROBLEM:

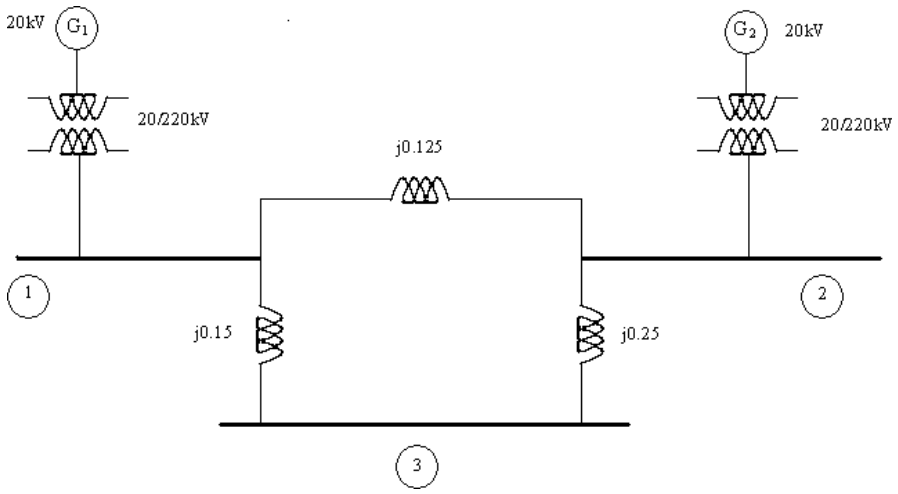
The one line diagram of a simple power system is shown in the figure. The neutral of each generator is grounded through a current limiting reactor of 0.25/3 per unit on a 100MVA base. The system data expressed in per unit on a common 100MVA base is tabulated below. The generators are running on no load at their rated voltage and rated frequency with their emf's in phase.

Determine the fault current for the following faults.

a) A balanced three phase fault at bus 3 through a fault impedance Z_f=j0.1 per unit.
b) A single line to ground fault at bus 3 through a fault impedance Z_f=j0.1 per unit.
c) A line to line fault at bus 3 through a fault impedance Z_f=j0.1 per unit.
d) A double line to ground fault at bus 3 through a fault impedance Z_f=j0.1 per unit.

Item	Base MVA	Voltage Rating	X^1	X^2	X^0
G1	100	20kV	0.15	0.15	0.05
G2	100	20kV	0.15	0.15	0.05
T1	100	20kV/220kV	0.1	0.1	0.1
T2	100	20kV/220kV	0.1	0.1	0.1
L12	100	220kV	0.125	0.125	0.3
L13	100	220kV	0.15	0.15	0.35
L23	100	220kV	0.25	0.25	0.7125

Manual Calculation:



PROGRAM:

zdata1 = [0 1 0 0.25

0 2 0 0.25

1 2 0 0.125

1 3 0 0.15

2 3 0 0.25];

zdata0 = [0 1 0 0.40

0 2 0 0.10

1 2 0 0.30

1 3 0 0.35

2 3 0 0.7125];

zdata2 = zdata1;

Zbus1 = zbuild(zdata1)

Zbus0 = zbuild(zdata0)

Zbus2 = Zbus1;

symfault(zdata1,Zbus1)

lgfault(zdata0, Zbus0, zdata1, Zbus1, zdata2, Zbus2)

llfault(zdata1, Zbus1, zdata2, Zbus2)

dlgfault(zdata0, Zbus0, zdata1, Zbus1, zdata2, Zbus2)

SYMMETRICAL FAULT

```
function symfaul(zdata, Zbus, V)

nl = zdata(:,1); nr = zdata(:,2); R = zdata(:,3);
X = zdata(:,4);
nc = length(zdata(1,:));
  if nc > 4
  BC = zdata(:,5);
  elseif nc ==4, BC = zeros(length(zdata(:,1)), 1);
  end
ZB = R + j*X;
nbr=length(zdata(:,1)); nbus = max(max(nl), max(nr));
if exist('V') == 1
    if length(V) == nbus
```

```
    V0 = V;
    else, end
else, V0 = ones(nbus, 1) + j*zeros(nbus, 1);
end
fprintf('\Three-phase balanced fault analysis \n')
ff = 999;
while ff > 0
nf = input('Enter Faulted Bus No. -> ');
     while nf <= 0 | nf > nbus
     fprintf('Faulted bus No. must be between 1 & %g \n', nbus)
     nf = input('Enter Faulted Bus No. -> ');
     end
fprintf('\nEnter Fault Impedance Zf = R + j*X in ')
Zf = input('complex form (for bolted fault enter 0). Zf = ');
fprintf('  \n')
fprintf('Balanced three-phase fault at bus No. %g\n', nf)

If = V0(nf)/(Zf + Zbus(nf, nf));
Ifm = abs(If); Ifmang=angle(If)*180/pi;
fprintf('Total fault current = %8.4f per unit \n\n', Ifm)
%fprintf(' p.u. \n\n', Ifm)
fprintf('Bus Voltages during fault in per unit \n\n')
fprintf('     Bus     Voltage       Angle\n')
fprintf('     No.     Magnitude     degrees\n')

for n = 1:nbus
    if n==nf
    Vf(nf) = V0(nf)*Zf/(Zf + Zbus(nf,nf)); Vfm = abs(Vf(nf));
angv=angle(Vf(nf))*180/pi;
    else, Vf(n) = V0(n) - V0(n)*Zbus(n,nf)/(Zf + Zbus(nf,nf));
    Vfm = abs(Vf(n)); angv=angle(Vf(n))*180/pi;
    end
    fprintf('   %4g',   n),  fprintf('%13.4f',  Vfm),fprintf('%13.4f\n',  angv)

end

fprintf('  \n')

fprintf('Line currents for fault at bus No. %g\n\n', nf)
fprintf('     From      To     Current     Angle\n')
fprintf('     Bus       Bus    Magnitude   degrees\n')

for n= 1:nbus
   %Ign=0;
    for I = 1:nbr
      if nl(I) == n | nr(I) == n
         if  nl(I)  ==n   k  =  nr(I);
         elseif nr(I) == n k = nl(I);
         end
           if k==0
            Ink = (V0(n) - Vf(n))/ZB(I);
            Inkm = abs(Ink); th=angle(Ink);
               %if th <= 0
                if real(Ink) > 0
                fprintf('  G  '),  fprintf('%7g',n),  fprintf('%12.4f',  Inkm)
                fprintf('%12.4f\n', th*180/pi)
```

```
                elseif real(Ink) ==0 & imag(Ink) < 0
                fprintf('      G    '), fprintf('%7g',n), fprintf('%12.4f', Inkm)
                fprintf('%12.4f\n', th*180/pi)
               else, end
            Ign=Ink;
            elseif k ~= 0
             Ink = (Vf(n) - Vf(k))/ZB(I)+BC(I)*Vf(n);
             %Ink = (Vf(n) - Vf(k))/ZB(I);
             Inkm = abs(Ink); th=angle(Ink);
             %Ign=Ign+Ink;
             %if th <= 0
              if real(Ink) > 0
                   fprintf('%7g', n), fprintf('%10g', k),
                   fprintf('%12.4f', Inkm), fprintf('%12.4f\n', th*180/pi)
                   elseif real(Ink) ==0 & imag(Ink) < 0
                   fprintf('%7g', n), fprintf('%10g', k),
                   fprintf('%12.4f', Inkm), fprintf('%12.4f\n', th*180/pi)
                   else, end
               else, end
      else, end
   end

 if n==nf
 fprintf('%7g',n), fprintf('         F'), fprintf('%12.4f', Ifm)
 fprintf('%12.4f\n', Ifmang)
 else, end
end
   resp=0;
   while strcmp(resp, 'n')~=1 & strcmp(resp, 'N')~=1 & strcmp(resp, 'y')~=1 &
strcmp(resp, 'Y')~=1
   resp = input('Another fault location? Enter ''y'' or ''n'' within single quote
-> ');
   if strcmp(resp, 'n')~=1 & strcmp(resp, 'N')~=1 & strcmp(resp, 'y')~=1 &
strcmp(resp, 'Y')~=1
   fprintf('\n Incorrect reply, try again \n\n'), end
   end
   if resp == 'y' | resp == 'Y'
   nf = 999;
   else ff = 0; end
end   % end for while
```

Line to ground fault

```
function lgfault(zdata0, Zbus0, zdata1, Zbus1, zdata2, Zbus2, V)
if exist('zdata2') ~= 1
zdata2=zdata1;
else, end
if exist('Zbus2') ~= 1
Zbus2=Zbus1;
else, end
nl = zdata1(:,1); nr = zdata1(:,2); nl0 = zdata0(:,1);
nr0 = zdata0(:,2); nbr=length(zdata1(:,1)); nbus =
max(max(nl), max(nr)); nbr0=length(zdata0(:,1));
```

```
R0 = zdata0(:,3); X0 = zdata0(:,4);
R1 = zdata1(:,3); X1 = zdata1(:,4);
R2 = zdata1(:,3); X2 = zdata1(:,4);

for k=1:nbr0
    if R0(k)==inf | X0(k) ==inf
    R0(k) = 99999999;  X0(k) = 99999999;
    else, end
end
ZB1 = R1 + j*X1;  ZB0 = R0 + j*X0;
ZB2 = R2 + j*X2;

if exist('V') == 1
    if length(V) == nbus
    V0 = V;
    else, end
else, V0 = ones(nbus, 1) + j*zeros(nbus, 1);
end
fprintf('\nLine-to-ground fault analysis \n')
ff = 999;
while ff > 0
nf = input('Enter Faulted Bus No. -> ');
while nf <= 0 | nf > nbus
     fprintf('Faulted bus No. must be between 1 & %g \n', nbus)
     nf = input('Enter Faulted Bus No. -> ');
     end
fprintf('\nEnter Fault Impedance Zf = R + j*X in ')
Zf = input('complex form (for bolted fault enter 0). Zf = ');
fprintf(' \n')
fprintf('Single line to-ground fault at bus No. %g\n', nf)
a =cos(2*pi/3)+j*sin(2*pi/3);
sctm = [1   1   1; 1 a^2  a; 1 a  a^2];
Ia0 = V0(nf)/(Zbus1(nf,nf)+Zbus2(nf, nf)+ Zbus0(nf, nf)+3*Zf); Ia1=Ia0; Ia2=Ia0;
I012=[Ia0; Ia1; Ia2];
Ifabc = sctm*I012;
Ifabcm = abs(Ifabc);
fprintf('Total fault current = %9.4f per unit\n\n', Ifabcm(1))
fprintf('Bus Voltages during the fault in per unit \n\n')
fprintf('     Bus     -------Voltage Magnitude ------- \n')
fprintf('     No.     Phase a      Phase b      Phase c  \n')

for n = 1:nbus
Vf0(n)= 0 - Zbus0(n, nf)*Ia0;
Vf1(n)= V0(n) - Zbus1(n, nf)*Ia1;
Vf2(n)= 0 - Zbus2(n, nf)*Ia2;
Vabc = sctm*[Vf0(n); Vf1(n); Vf2(n)];
Va(n)=Vabc(1); Vb(n)=Vabc(2); Vc(n)=Vabc(3);
fprintf(' %5g',n)
fprintf(' %11.4f', abs(Va(n))),fprintf(' %11.4f', abs(Vb(n)))
fprintf(' %11.4f\n', abs(Vc(n)))
end
fprintf('  \n')
fprintf('Line currents for fault at bus No.  %g\n\n', nf)
fprintf('     From       To       -----Line Current Magnitude ----- \n')
fprintf('     Bus        Bus      Phase a      Phase b      Phase c  \n')
for n= 1:nbus
```

```
   for I = 1:nbr
      if nl(I) == n | nr(I) == n
         if  nl(I)  ==n   k  =  nr(I);
         elseif nr(I) == n k = nl(I);
         end
            if k ~= 0
            Ink1(n, k) = (Vf1(n) - Vf1(k))/ZB1(I);
            Ink2(n, k) =  (Vf2(n) - Vf2(k))/ZB2(I);
            else, end
      else, end
   end
   for I = 1:nbr0
      if nl0(I) == n | nr0(I) == n
         if  nl0(I)  ==n   k  =  nr0(I);
         elseif nr0(I) == n k = nl0(I);
         end
            if k ~= 0
            Ink0(n, k) =  (Vf0(n) - Vf0(k))/ZB0(I);
            else, end
      else, end
   end
   for I = 1:nbr
      if nl(I) == n | nr(I) == n
         if  nl(I)  ==n   k  =  nr(I);
         elseif nr(I) == n k = nl(I);
         end
            if k ~= 0
            Inkabc = sctm*[Ink0(n, k); Ink1(n, k); Ink2(n, k)];
            Inkabcm = abs(Inkabc); th=angle(Inkabc);
                if real(Inkabc(1)) > 0
                fprintf('%7g', n), fprintf('%10g', k),
                fprintf(' %11.4f', abs(Inkabc(1))),fprintf(' %11.4f',
abs(Inkabc(2)))
                fprintf(' %11.4f\n', abs(Inkabc(3)))
                elseif real(Inkabc(1)) ==0 & imag(Inkabc(1)) < 0
                fprintf('%7g', n), fprintf('%10g', k),
                fprintf(' %11.4f', abs(Inkabc(1))),fprintf(' %11.4f',
abs(Inkabc(2)))
                fprintf(' %11.4f\n', abs(Inkabc(3)))
                else, end
             else, end
      else, end
   end
   if n==nf
   fprintf('%7g',n),  fprintf('          F'),
   fprintf(' %11.4f', Ifabcm(1)),fprintf(' %11.4f', Ifabcm(2))
   fprintf(' %11.4f\n', Ifabcm(3))
   else, end
end
resp=0;
   while strcmp(resp, 'n')~=1 & strcmp(resp, 'N')~=1 & strcmp(resp, 'y')~=1 &
strcmp(resp, 'Y')~=1
   resp = input('Another fault location? Enter ''y'' or ''n'' within single quote
-> ');
   if strcmp(resp, 'n')~=1 & strcmp(resp, 'N')~=1 & strcmp(resp, 'y')~=1 &
strcmp(resp, 'Y')~=1
   fprintf('\n Incorrect reply, try again \n\n'), end
```

```
   end
   if resp == 'y' | resp == 'Y'
   nf = 999;
   else ff = 0; end
end   % end for while
```

Line to Line fault

```
function llfault(zdata1, Zbus1, zdata2, Zbus2, V)
if exist('zdata2') ~= 1
zdata2=zdata1;
else, end
if exist('Zbus2') ~= 1
Zbus2=Zbus1;
else, end

nl = zdata1(:,1); nr = zdata1(:,2);
R1 = zdata1(:,3); X1 = zdata1(:,4);
R2 = zdata2(:,3); X2 = zdata2(:,4);
ZB1 = R1 + j*X1;  ZB2 = R2 + j*X2;
nbr=length(zdata1(:,1)); nbus = max(max(nl), max(nr));
if exist('V') == 1
    if length(V) == nbus
    V0 = V;
    else, end
else, V0 = ones(nbus, 1) + j*zeros(nbus, 1);
end
fprintf('\nLine-to-line fault analysis \n')
ff = 999;
while ff > 0
nf = input('Enter Faulted Bus No. -> ');
while nf <= 0 | nf > nbus
     fprintf('Faulted bus No. must be between 1 & %g \n', nbus)
     nf = input('Enter Faulted Bus No. -> ');
     end
fprintf('\nEnter Fault Impedance Zf = R + j*X in ')
Zf = input('complex form (for bolted fault enter 0). Zf = ');
fprintf(' \n')
fprintf('Line-to-line fault at bus No. %g\n', nf)
a =cos(2*pi/3)+j*sin(2*pi/3);
sctm = [1   1   1; 1 a^2  a; 1 a  a^2];
Ia0=0;
Ia1 = V0(nf)/(Zbus1(nf,nf)+Zbus2(nf, nf)+Zf); Ia2=-Ia1;
I012=[Ia0; Ia1; Ia2];
Ifabc = sctm*I012;
Ifabcm = abs(Ifabc);
fprintf('Total fault current = %9.4f per unit\n\n', Ifabcm(2))
fprintf('Bus Voltages during the fault in per unit \n\n')
fprintf('     Bus     -------Voltage  Magnitude----------\n')
fprintf('     No.     Phase a     Phase b     Phase c  \n')

for n = 1:nbus
Vf0(n)= 0;
Vf1(n)= V0(n) - Zbus1(n, nf)*Ia1;
Vf2(n)= 0 - Zbus2(n, nf)*Ia2;
```

```
Vabc = sctm*[Vf0(n); Vf1(n); Vf2(n)];
Va(n)=Vabc(1); Vb(n)=Vabc(2); Vc(n)=Vabc(3);
fprintf(' %5g',n)
fprintf(' %11.4f', abs(Va(n))),fprintf(' %11.4f', abs(Vb(n)))
fprintf(' %11.4f\n', abs(Vc(n)))
end
fprintf('  \n')
fprintf('Line currents for fault at bus No.  %g\n\n', nf)
fprintf('     From      To      -----Line Current Magnitude----- \n')
fprintf('     Bus       Bus     Phase a     Phase b     Phase c  \n')

for   n=  1:nbus
   for I = 1:nbr
      if nl(I) == n | nr(I) == n
         if  nl(I)  ==n  k  =  nr(I);
         elseif nr(I) == n  k = nl(I);
         end
            if k ~= 0
            Ink0(n, k) = 0;
            Ink1(n, k) = (Vf1(n) - Vf1(k))/ZB1(I);
            Ink2(n, k) =  (Vf2(n) - Vf2(k))/ZB2(I);

            Inkabc = sctm*[Ink0(n, k); Ink1(n, k); Ink2(n, k)];
            Inkabcm = abs(Inkabc); th=angle(Inkabc);
                 if real(Inkabc(2)) < 0

               fprintf('%7g', n), fprintf('%10g', k),
                fprintf(' %11.4f', abs(Inkabc(1))),fprintf(' %11.4f',
abs(Inkabc(2)))
                fprintf(' %11.4f\n', abs(Inkabc(3)))
                elseif real(Inkabc(2)) ==0 & imag(Inkabc(2)) > 0
                fprintf('%7g', n), fprintf('%10g', k),
                fprintf(' %11.4f', abs(Inkabc(1))),fprintf(' %11.4f',
abs(Inkabc(2)))
                fprintf(' %11.4f\n', abs(Inkabc(3)))
                else, end
             else, end
      else, end
   end
   if n==nf
   fprintf('%7g',n),  fprintf('          F'),
   fprintf(' %11.4f', Ifabcm(1)),fprintf(' %11.4f', Ifabcm(2))
   fprintf(' %11.4f\n', Ifabcm(3))
   else, end
end
resp=0;
   while strcmp(resp, 'n')~=1 & strcmp(resp, 'N')~=1 & strcmp(resp, 'y')~=1 &
strcmp(resp, 'Y')~=1
   resp = input('Another fault location? Enter ''y'' or ''n'' within single quote
-> ');
   if strcmp(resp, 'n')~=1 & strcmp(resp, 'N')~=1 & strcmp(resp, 'y')~=1 &
strcmp(resp, 'Y')~=1
   fprintf('\n Incorrect reply, try again \n\n'), end
   end
   if resp == 'y' | resp == 'Y'
   nf = 999;
```

```
   else ff = 0; end
end   % end for while
```

Double Line to Ground fault

```
function dlgfault(zdata0, Zbus0, zdata1, Zbus1, zdata2, Zbus2, V)

if exist('zdata2') ~= 1
zdata2=zdata1;
else, end
if exist('Zbus2') ~= 1
Zbus2=Zbus1;
else, end

nl = zdata1(:,1); nr = zdata1(:,2); nl0 = zdata0(:,1);
nr0 = zdata0(:,2); nbr=length(zdata1(:,1)); nbus =
max(max(nl), max(nr)); nbr0=length(zdata0(:,1));
R0 = zdata0(:,3); X0 = zdata0(:,4);
R1 = zdata1(:,3); X1 = zdata1(:,4);
R2 = zdata2(:,3); X2 = zdata2(:,4);

for k = 1:nbr0
    if R0(k) == inf | X0(k) == inf
    R0(k) = 99999999; X0(k) = 999999999;
    else, end
end
ZB1 = R1 + j*X1;  ZB0 = R0 + j*X0;
ZB2 = R2 + j*X2;

if exist('V') == 1
    if length(V) == nbus
    V0 = V;
    else, end
else, V0 = ones(nbus, 1) + j*zeros(nbus, 1);
end

fprintf('\nDouble line-to-ground fault analysis \n')
ff = 999;
while ff > 0
nf = input('Enter Faulted Bus No. -> ');
while nf <= 0 | nf > nbus
     fprintf('Faulted bus No. must be between 1 & %g \n', nbus)
     nf = input('Enter Faulted Bus No. -> ');
     end
fprintf('\nEnter Fault Impedance Zf = R + j*X in ')
Zf = input('complex form (for bolted fault enter 0). Zf = ');
fprintf(' \n')
fprintf('Double line-to-ground fault at bus No. %g\n', nf)
a =cos(2*pi/3)+j*sin(2*pi/3);
sctm = [1   1   1; 1 a^2  a; 1 a  a^2];

Z11 = Zbus2(nf, nf)*(Zbus0(nf, nf)+ 3*Zf)/(Zbus2(nf, nf)+Zbus0(nf, nf)+3*Zf);
Ia1 = V0(nf)/(Zbus1(nf,nf)+Z11);
Ia2 =-(V0(nf) - Zbus1(nf, nf)*Ia1)/Zbus2(nf,nf);
```

```
Ia0 =-(V0(nf) - Zbus1(nf, nf)*Ia1)/(Zbus0(nf,nf)+3*Zf);
I012=[Ia0; Ia1; Ia2];
Ifabc = sctm*I012; Ifabcm=abs(Ifabc);
Ift = Ifabc(2)+Ifabc(3);
Iftm = abs(Ift);

fprintf('Total fault current = %9.4f per unit\n\n', Iftm)
fprintf('Bus Voltages during the fault in per unit \n\n')
fprintf('     Bus     -------Voltage Magnitude ------- \n')
fprintf('     No.     Phase a      Phase b      Phase c  \n')

for n = 1:nbus
Vf0(n)= 0 - Zbus0(n, nf)*Ia0;
Vf1(n)= V0(n) - Zbus1(n, nf)*Ia1;
Vf2(n)= 0 - Zbus2(n, nf)*Ia2;
Vabc = sctm*[Vf0(n); Vf1(n); Vf2(n)];
Va(n)=Vabc(1); Vb(n)=Vabc(2); Vc(n)=Vabc(3);
fprintf(' %5g',n)
fprintf(' %11.4f', abs(Va(n))),fprintf(' %11.4f', abs(Vb(n)))
fprintf(' %11.4f\n', abs(Vc(n)))
end
fprintf('  \n')
fprintf('Line currents for fault at bus No.  %g\n\n', nf)
fprintf('     From       To       -----Line Current Magnitude ----- \n')
fprintf('     Bus        Bus      Phase a      Phase b      Phase c  \n')
for n= 1:nbus
   for I = 1:nbr
      if nl(I) == n | nr(I) == n
         if  nl(I)  ==n   k  =  nr(I);
         elseif nr(I) == n  k = nl(I);
         end
            if k ~= 0
            Ink1(n, k) = (Vf1(n) - Vf1(k))/ZB1(I);
            Ink2(n, k) =  (Vf2(n) - Vf2(k))/ZB2(I);
            else, end
      else, end
   end
   for I = 1:nbr0
      if nl0(I) == n | nr0(I) == n
         if  nl0(I)  ==n   k  =  nr0(I);
         elseif nr0(I) == n  k = nl0(I);
         end
            if k ~= 0
            Ink0(n, k) =  (Vf0(n) - Vf0(k))/ZB0(I);
            else, end
      else, end
   end
   for I = 1:nbr
      if nl(I) == n | nr(I) == n
         if  nl(I)  ==n   k  =  nr(I);
         elseif nr(I) == n  k = nl(I);
         end
            if k ~= 0
            Inkabc = sctm*[Ink0(n, k); Ink1(n, k); Ink2(n, k)];
            Inkabcm = abs(Inkabc); th=angle(Inkabc);
                if real(Inkabc(2)) < 0
```

```
                fprintf('%7g', n),  fprintf('%10g', k),
                fprintf(' %11.4f', abs(Inkabc(1))),fprintf(' %11.4f',
abs(Inkabc(2)))
                fprintf(' %11.4f\n', abs(Inkabc(3)))
                elseif real(Inkabc(2)) ==0 & imag(Inkabc(2)) > 0
                fprintf('%7g', n), fprintf('%10g', k),
                fprintf(' %11.4f', abs(Inkabc(1))),fprintf(' %11.4f',
abs(Inkabc(2)))
                fprintf(' %11.4f\n', abs(Inkabc(3)))
                else, end
             else, end
      else, end
   end
   if n==nf
   fprintf('%7g',n),  fprintf('           F'),
   fprintf(' %11.4f', Ifabcm(1)),fprintf(' %11.4f', Ifabcm(2))
   fprintf(' %11.4f\n', Ifabcm(3))
   else, end
end
```

```
resp=0;
   while strcmp(resp, 'n')~=1 & strcmp(resp, 'N')~=1 & strcmp(resp, 'y')~=1 &
strcmp(resp, 'Y')~=1
   resp = input('Another fault location? Enter ''y'' or ''n'' within single quote
-> ');
   if strcmp(resp, 'n')~=1 & strcmp(resp, 'N')~=1 & strcmp(resp, 'y')~=1 &
strcmp(resp, 'Y')~=1
   fprintf('\n Incorrect reply, try again \n\n'), end
   end
   if resp == 'y' | resp == 'Y'
   nf = 999;
   else ff = 0; end
end   % end for while
```

OUTPUT:

RESULT:

Thus the modeling and analysis of power system under faulted condition was made familiar and the fault level, post fault voltage and currents for different types of fault both symmetric and unsymmetrical was computed.

EXP NO: 6

DATE:

TRANSIENT AND SMALL SIGNAL STABILITY ANALYSIS – SINGLE MACHINE INFINITE BUS SYSTEM

AIM:

To become familiar with various aspects of the transient and small signal stability analysis of Single-Machine-Infinite Bus (SMIB) system.

OBJECTIVES

- To understand modeling and analysis of transient and small signal stability of a SMIB power system.
- To examine the transient stability of a SMIB and determine the critical clearing time of the system, through stimulation by trial and error method and by direct method.
- To assess the transient stability of a multi- machine power system when subjected to a common disturbance sequence: fault application on a transmission line followed by fault removal and line opening.
- To determine the critical clearing time.

THEORY :

Stability : Stability problem is concerned with the behaviour of power system when it is subjected to disturbance and is classified into small signal stability problem if the disturbances are small and transient stability problem when the disturbances are large.

Transient stability: When a power system is under steady state, the load plus transmission loss equals to the generation in the system. The generating units run a synchronous speed and system frequency, voltage, current and power flows are steady. When a large disturbance such as three phase fault, loss of load, loss of generation etc., occurs the power balance is upset and the generating units rotors experience either acceleration or deceleration. The system may come back to a steady state condition

maintaining synchronism or it may break into subsystems or one or more machines may pull out of synchronism. In the former case the system is said to be stable and in the later case it is said to be unstable.

Small signal stability: When a power system is under steady state, normal operating condition, the system may be subjected to small disturbances such as variation in load and generation, change in field voltage, change in mechanical toque etc., The nature of system response to small disturbance depends on the operating conditions, the transmission system strength, types of controllers etc. Instability that may result from small disturbance may be of two forms,

(i) Steady increase in rotor angle due to lack of synchronising torque.

(ii) Rotor oscillations of increasing magnitude due to lack of sufficient damping torque.

FORMULA :

Reactive power $Q_e = \sin(\cos^{-1}(p.f))$

$$\text{Stator Current } I_t = \frac{S^*}{E_t^*}$$

$$= \frac{P_e - jQ_e}{E_t^*}$$

Voltage behind transient condition

$$E^1 = E_t + j X_d^1 I_t$$

Voltage of infinite bus

$$E_B = E_t - j(X_3 + X_{tr})I_t$$

$$\text{where, } X_3 = \frac{X_1 X_2}{X_1 + X_2}$$

Angular separation between E^1 and E_B

$$\delta o = \angle E^1 - \angle E_B$$

Prefault Operation:

$$X = j X_d^1 + jX_{tr} + \frac{X_1 X_2}{X_1 + X_2}$$

$$\text{Power } P_e = \frac{E^1 \times E_B}{X} \sin\delta_o$$

$$\delta_o = \sin^{-1} \frac{P_e * X}{E^1 * E_B}$$

During Fault Condition:

$P_e = P_{Eii} = 0$

Find out X from the equivalent circuit during fault condition

Post fault Condition:

Find out X from the equivalent circuit during post fault condition

$$\text{Power } P_e = \left\{ \frac{E^1 \times E_B}{X} \right\} \sin\delta_o$$

$\delta_{max} = \pi - \delta_o$

$$P_e = \frac{P_m}{\sin\delta_{max}}$$

Critical Clearing Angle:

$$\text{Cos}\delta_{cr} = \frac{P_m(\delta_{\mathbf{max}} - \delta_o) + P_{3max}\text{cos}\delta_{\mathbf{max}} - P_{2max}\text{cos}\delta_{\mathbf{o}}}{\mathbf{P_{3max} - P_{2max}}}$$

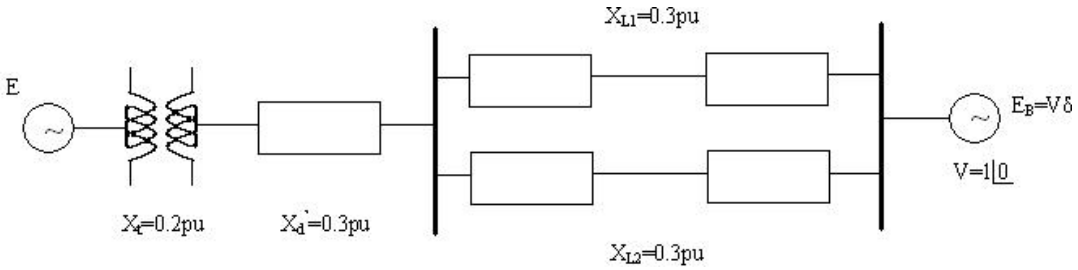
Critical Clearing Time:

$$t_{cr} = \frac{\sqrt{2H\,(\delta_{cr} - \delta_o)}}{\sqrt{\pi f_o\, P_m}} \quad \text{Secs}$$

PROBLEM:

Transient stability of SMIB system

A 60Hz synchronous generator having inertia constant H = 5 MJ/MVA and a direct axis transient reactance X_d^1 = 0.3 per unit is connected to an infinite bus through a purely reactive circuit as shown in figure. Reactances are marked on the diagram on a common system base. The generator is delivering real power P_e = 0.8 per unit and Q = 0.074 per unit to the infinite bus at a voltage of V = 1 per unit.



a) A temporary three-phase fault occurs at the sending end of the line at point F.When the fault is cleared, both lines are intact. Determine the critical clearing angle and the critical fault clearing time.

b) A three phase fault occurs at the middle of one of the lines, the fault is cleared and teh faulted line is isolated. Determine the critical clearing angle.

PROGRAM :

a) Pm = 0.8; E = 1.17; V = 1.0;

X1 = 0.65; X2 = inf; X3 = 0.65;

eacfault(Pm, E, V, X1, X2, X3)

b) Pm = 0.8; E = 1.17; V = 1.0;

X1 = 0.65; X2 = 1.8; X3 = 0.8;

eacfault(Pm, E, V, X1, X2, X3)

eacfault

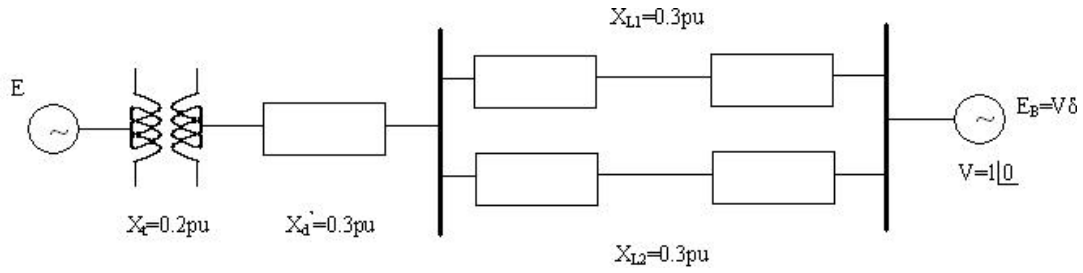
```
function eacfault(Pm, E, V, X1, X2, X3)
if exist('Pm')~=1
Pm = input('Generator output power in p.u. Pm = '); else, end
if exist('E')~=1
E = input('Generator e.m.f. in p.u. E = '); else, end
if exist('V')~=1
V = input('Infinite bus-bar voltage in p.u. V = '); else, end
if exist('X1')~=1
X1 = input('Reactance before Fault in p.u. X1 = '); else, end
if exist('X2')~=1
X2 = input('Reactance during Fault in p.u. X2 = '); else, end
if exist('X3')~=1
X3 = input('Reactance aftere Fault in p.u. X3 = '); else, end
Pe1max = E*V/X1; Pe2max=E*V/X2; Pe3max=E*V/X3;
delta = 0:.01:pi;
Pe1 = Pe1max*sin(delta); Pe2 = Pe2max*sin(delta); Pe3 = Pe3max*sin(delta);
d0 =asin(Pm/Pe1max); dmax = pi-asin(Pm/Pe3max);
cosdc = (Pm*(dmax-d0)+Pe3max*cos(dmax)-Pe2max*cos(d0))/(Pe3max-Pe2max);
  if abs(cosdc) > 1
  fprintf('No critical clearing angle could be found.\n')
  fprintf('system can remain stable during this disturbance.\n\n')
  return
  else, end
dc=acos(cosdc);
if dc > dmax
fprintf('No critical clearing angle could be found.\n')
  fprintf('System can remain stable during this disturbance.\n\n')
  return
  else, end
```

```
Pmx=[0   pi-d0]*180/pi; Pmy=[Pm Pm];
x0=[d0 d0]*180/pi; y0=[0 Pm]; xc=[dc dc]*180/pi; yc=[0 Pe3max*sin(dc)];
xm=[dmax dmax]*180/pi; ym=[0 Pe3max*sin(dmax)];
d0=d0*180/pi; dmax=dmax*180/pi; dc=dc*180/pi;
x=(d0:.1:dc);
y=Pe2max*sin(x*pi/180);
y1=Pe2max*sin(d0*pi/180);
y2=Pe2max*sin(dc*pi/180);
x=[d0 x dc];
y=[Pm y Pm];
xx=dc:.1:dmax;
h=Pe3max*sin(xx*pi/180);
xx=[dc xx  dmax];
hh=[Pm h Pm];
delta=delta*180/pi;
if X2 == inf
fprintf('\nFor this case tc can be found from analytical formula. \n')
H=input('To find tc enter Inertia Constant H, (or 0 to skip) H = ');
   if H ~= 0
   d0r=d0*pi/180; dcr=dc*pi/180;
   tc = sqrt(2*H*(dcr-d0r)/(pi*60*Pm));
   else, end
else, end
%clc
fprintf('\nInitial  power  angle  =  %7.3f  \n',  d0)
fprintf('Maximum  angle  swing  =  %7.3f  \n',  dmax)
fprintf('Critical clearing angle = %7.3f \n\n', dc)
if X2==inf & H~=0
fprintf('Critical clearing time = %7.3f sec. \n\n', tc)
else, end
h = figure; figure(h);
fill(x,y,'m')
hold;
fill(xx,hh,'c')
plot(delta, Pe1,'-', delta, Pe2,'r-',  delta, Pe3,'g-', Pmx, Pmy,'b-', x0,y0,
xc,yc, xm,ym), grid
Title('Application of equal area criterion to a critically cleared system')
xlabel('Power angle, degree'), ylabel(' Power, per unit')
text(5, 1.07*Pm, 'Pm')
text(50, 1.05*Pe1max,['Critical clearing angle = ',num2str(dc)])
axis([0 180  0 1.1*Pe1max])
hold off;
```

OUTPUT:

Small signal stability of SMIB system

2) A 60Hz synchronous generator having inertia constant H = 9.94 MJ/MVA and a direct axis transient reactance X_d^1 = 0.3 per unit is connected to an infinite bus through a purely reactive circuit as shown in figure. Reactances are marked on the diagram on a common system base. The generator is delivering real power P_e = 0.6 per unit and 0.8 power factor lagging to the infinite bus at a voltage of V = 1 per unit.



Assume the per unit damping power co-efficient is D=0.138. Consider a small disturbance of Δδ=10°=0.1745 radian. Obtain equations describing the motion of the rotor angle and the generator frequency.

PROGRAM:

```
E=1.35; V=1.0; H=9.94; X=0.65; Pm=0.6; D=0.138; f0=60;
Pmax=E*V/X, d0=asin(Pm/Pmax)
Ps=Pmax*cos(d0)
wn=sqrt(pi*60/H*Ps)
z=D/2*sqrt(pi*60/(H*Ps))
wd=wn*sqrt(1-z^2),fd=wd/(2*pi)
tau=1/(z*wn)
th=acos(z)
Dd0=10*pi/180;
t=0:.01:3;
Dd=Dd0/sqrt(1-z^2)*exp(-z*wn*t).*sin(wd*t+th);
d=(d0+Dd)*180/pi;
```

```
Dw=-wn*Dd0/sqrt(1-z^2)*exp(-z*wn*t).*sin(wd*t);
f=f0+Dw/(2*pi);
subplot(2,1,1),plot(t,d),grid
xlabel('t sec'),ylabel('Delta degree')
subplot(2,1,2),plot(t,f),grid
xlabel('t sec'),ylabel('frquency hertz')
subplot(111)
```

OUTPUT:

RESULT :

Thus the various aspects of transient and small signal stability analysis of single machine infinite bus system were made familiar.

EXP. NO: 7

DATE:

TRANSIENT STABILITY ANALYSIS OF MULTIMACHINE POWER SYSTEMS

AIM:

(i) To become familiar with modeling aspects of synchronous machines and network for transient stability analysis of multi-machine power systems.

(ii) To become familiar with the state-of-the-art algorithm for simplified transient stability simulation involving only classical machine models for synchronous machines.

(iii) To understand system behavior when subjected to large disturbances in the presence of synchronous machine controllers.

(iv) To become proficient in the usage of the software to tackle real life problems encountered in the areas of power system planning and operation.

OBJECTIVES

(i) To assess the transient stability of a multi machine power system when subjected to a common disturbance sequence: fault application on a transmission line followed by fault removal and line opening.

(ii) To determine the critical clearing time for the above sequence.

(iii) To observe system response and understand its behavior during a full load rejection at a substation with and without controllers.

(iv) To observe system response and understand its behavior during loss of a major generating station.

(v) To understand machine and system behavior during loss of excitation.

(vi) To study the effect of load relief provided by under frequency load shedding scheme.

THEORY:

The classical transient stability study is based on the application of a three – phase fault. A solid three – phase fault at bus k in the network results in $V_k = 0$. This is simulated by removing the k^{th} row and column from the pre-fault bus admittance matrix. The new bus admittance matrix is reduced by eliminating all nodes except the internal generator nodes. The generator excitation voltages during the fault and post – fault modes are assumed to remain constant.

The electrical power of the i^{th} generator in terms of the new reduced bus admittance matrices are obtained from

$$P_{ei} = \sum_{j=1}^{m} |E_i'||E_j'||Y_{ij}| \cos(\theta_{ij} - \delta_i + \delta_j) \quad \text{----------(1)}$$

The swing equation with damping neglected, for machine i becomes, is given by

$$\frac{H_i}{\pi f_0}\frac{d^2\delta_i}{dt^2} = P_{mi} - \sum_{j=1}^{m} |E_i'||E_j'||Y_{ij}| \cos(\theta_{ij} - \delta_i + \delta_j) \quad \text{---------- (2)}$$

where Y_{ij} are the elements of the faulted reduced bus admittance matrix, and H_i is the inertia constant of machine i expressed on the common MVA base S_B. If H_{Gi} is the inertia constant of machine i expressed on the machine rated MVA S_{Gi}, then H_i is given by

$$H_i = (S_{Gi} / S_B) H_{Gi} \quad \text{----------(3)}$$

Showing the electrical power of the i^{th} generator by P_e^f and transforming equation (2) into state variable model yields

$$\frac{d\delta_i}{dt} = \Delta\omega_i, \quad i = 1,2, \ldots\ldots m \quad \text{----------(4)}$$

$$\frac{d\Delta\omega_i}{dt} = (\pi f_0 / H_i)(P_m - P_e^f) \quad \text{----------(5)}$$

For multi-machine transient stability analysis of an interconnected power system, it is necessary to solve two state equations for each generator, with initial power angles δ_0 and $\Delta\omega_{0i} = 0$.

When the fault is cleared, which may involve the removal of the faulty line, the bus admittance matrix is recomputed to reflect the change in the network. Next the post- fault reduced bus admittance matrix is evaluated and the post- fault electrical power of the i^{th} generator shown by P_i^{pf} is readily determined from (1).

Using the post-fault power P_i^{pf}, the simulation is continued to determine the system stability, until the plots reveal a definite trend as to stability or instability.

Usually the slack generator is selected as the reference, and the phase angle difference of all other generators with respect to the reference machine are plotted.

Usually, the solution is carried out for two swings to show that the second swing is not greater than the first one. If the angle differences do not increase, the system is stable. If any of the angle differences increase indefinitely, the system is unstable.

PROBLEM:

For bus 1, the voltage is given as $V_1=1.06\angle 0$ and it is taken as slack bus. The base value is 100MVA.

LOAD DATA		
BUS NO	LOAD	
	MW	Mvar
1	0	0
2	0	0
3	0	0
4	100	70
5	90	30
6	160	110

GENERATION SCHEDULE				
BUS NO	VOLTAGE MAG	GENERATION MW	Mvar LIMITS	
			Min	Max
1	1.06	------	-----	------
2	1.04	150	0	140
3	1.03	100	0	90

LINE DATA				
LINE NO (START)	LINE NO(END)	R(PU)	X(PU)	1/2B(PU)
1	4	0.035	0.225	0.0065
1	5	0.025	0.105	0.0045
1	6	0.040	0.215	0.0055
2	4	0.000	0.035	0.0000
3	5	0.000	0.042	0.0000
4	6	0.028	0.125	0.0035
5	6	0.026	0.175	0.0300

MACHINE DATA			
GEN	Ra	Xd’	H
1	0	0.20	20
2	0	0.15	4
3	0	0.25	5

A three phase occurs on line 5-6 near bus 6 and is cleared by the simultaneous opening of breakers at both ends of the line. Perform the transient stability analysis and determine the system stability for a) when the fault is cleared in 0.4 second b) when the fault is cleared in 0.5 second c) Repeat the simulation to determine the critical clearing angle

Manual Calculation:

PROGRAM:

```
basemva = 100; accuracy = 0.0001;  maxiter = 10;
busdata=[1  1  1.06   0.0  00.00    00.00    0.00    00.00     0     0  0
         2  2  1.04   0.0  00.00    00.00  150.00    00.00     0   140  0
         3  2  1.03   0.0  00.00    00.00  100.00    00.00     0    90  0
         4  0  1.0    0.0 100.00    70.00   00.00    00.00     0    0   0
         5  0  1.0    0.0  90.00    30.00   00.00    00.00     0    0   0
         6  0  1.0    0.0 160.00   110.00   00.00    00.00     0    0   0];

linedata=[1  4  0.035   0.225   0.0065   1  0
          1  5  0.025   0.105   0.0045   1  0
          1  6  0.040   0.215   0.0055   1  0
          2  4  0.000   0.035   0.0000   1  0
          3  5  0.000   0.042   0.0000   1  0
          4  6  0.028   0.125   0.0035   1  0
          5  6  0.026   0.175   0.0300   1  0];

lfybus                % form the bus admittance matrix
lfnewton     % Power flow solution by Newton-Raphson method
busout       % Prints the power flow solution on the screen

%        Gen.  Ra   Xd'    H
gendata=[ 1     0   0.20   20
          2     0   0.15    4
          3     0   0.25    5];
trstab
```

trstab

```
global Pm f H E  Y th ngg
f=60;
%zdd=gendata(:,2)+j*gendata(:,3);
ngr=gendata(:,1);
```

```
%H=gendata(:,4);
ngg=length(gendata(:,1));
%%
for k=1:ngg
zdd(ngr(k))=gendata(k,   2)+j*gendata(k,3);
%H(ngr(k))=gendata(k, 4);
H(k)=gendata(k,4);    % new
end
%%
for k=1:ngg
I=conj(S(ngr(k)))/conj(V(ngr(k)));
%Ep(ngr(k)) = V(ngr(k))+zdd(ngr(k))*I;
%Pm(ngr(k))=real(S(ngr(k)));
Ep(k) = V(ngr(k))+zdd(ngr(k))*I;  % new
Pm(k)=real(S(ngr(k)));              % new

end
E=abs(Ep); d0=angle(Ep);
for k=1:ngg
nl(nbr+k) = nbus+k;

nr(nbr+k) = gendata(k, 1);

%R(nbr+k)  = gendata(k, 2);
%X(nbr+k)  = gendata(k, 3);

R(nbr+k)  = real(zdd(ngr(k)));
X(nbr+k)  = imag(zdd(ngr(k)));

Bc(nbr+k)  = 0;
a(nbr+k) = 1.0;
yload(nbus+k)=0;
end
nbr1=nbr; nbus1=nbus;
nbrt=nbr+ngg;
nbust=nbus+ngg;
linedata=[nl, nr, R, X, -j*Bc, a];
[Ybus, Ybf]=ybusbf(linedata, yload, nbus1,nbust);
fprintf('\nPrefault reduced bus admittance matrix \n')
Ybf
Y=abs(Ybf); th=angle(Ybf);
Pm=zeros(1, ngg);
disp(['      G(i)     E''(i)      d0(i)       Pm(i)'])
for ii = 1:ngg
for jj = 1:ngg
Pm(ii) = Pm(ii) + E(ii)*E(jj)*Y(ii, jj)*cos(th(ii, jj)-d0(ii)+d0(jj));

end,
fprintf('       %g', ngr(ii)), fprintf('   %8.4f',E(ii)), fprintf('   %8.4f',
180/pi*d0(ii))
fprintf('  %8.4f \n',Pm(ii))
end
respfl='y';
while respfl =='y' | respfl=='Y' nf=input('Enter
faulted bus No. -> '); fprintf('\nFaulted reduced
bus admittance matrix\n')
```

```
Ydf=ybusdf(Ybus, nbus1, nbust, nf)
%Fault cleared
[Yaf]=ybusaf(linedata, yload, nbus1,nbust, nbrt);
fprintf('\nPostfault reduced bus admittance matrix\n')
Yaf
resptc='y';
while resptc =='y' | resptc=='Y'
tc=input('Enter clearing time of fault in sec. tc = ');
tf=input('Enter final simulation time in sec.  tf = ');
clear t  x  del
t0 = 0;
w0=zeros(1, length(d0));
x0 = [d0, w0];
tol=0.0001;
Y=abs(Ydf); th=angle(Ydf);
%[t1, xf] =ode23('dfpek', t0, tc, x0, tol); % Solution during fault (use with
MATLAB 4)
tspan=[t0, tc];
[t1, xf] =ode23('dfpek', tspan, x0); % Solution during fault (use with MATLAB 5)
x0c =xf(length(xf), :);
Y=abs(Yaf); th=angle(Yaf);
%[t2,xc] =ode23('afpek', tc, tf, x0c, tol); %  Postfault solution (use with MATLAB
4)
tspan = [tc, tf];                                % use with MATLAB 5
[t2,xc] =ode23('afpek', tspan, x0c);        % Postfault solution (use with MATLAB
5)
t =[t1; t2]; x = [xf; xc];
fprintf('\nFault is cleared at %4.3f Sec. \n', tc)
for k=1:nbus
    if kb(k)==1
    ms=k; else, end
end
fprintf('\nPhase angle difference of each machine \n')
fprintf('with respect to the slack in degree.\n')
fprintf('   t - sec')
kk=0;
for k=1:ngg
    if k~=ms
    kk=kk+1;
    del(:,kk)=180/pi*(x(:,k)-x(:,ms));
    fprintf('    d(%g,',ngr(k)), fprintf('%g)', ngr(ms))
    else, end
end
fprintf(' \n')
disp([t, del])
h=figure; figure(h)
plot(t, del)
title(['Phase angle difference (fault cleared at ', num2str(tc),'s)'])
xlabel('t, sec'), ylabel('Delta, degree'), grid
   resp=0;
   while strcmp(resp, 'n')~=1 & strcmp(resp, 'N')~=1 & strcmp(resp, 'y')~=1 &
strcmp(resp, 'Y')~=1
   resp=input('Another clearing time of fault? Enter ''y'' or ''n'' within quotes
-> ');

   if strcmp(resp, 'n')~=1 & strcmp(resp, 'N')~=1 & strcmp(resp, 'y')~=1 &
strcmp(resp, 'Y')~=1
```

```
   fprintf('\n Incorrect reply, try again \n\n'), end
   end
resptc=resp;

end
    resp2=0;
    while strcmp(resp2, 'n')~=1 & strcmp(resp2, 'N')~=1 & strcmp(resp2, 'y')~=1 &
```

```
strcmp(resp2, 'Y')~=1
    resp2=input('Another fault location: Enter ''y'' or ''n'' within quotes -> ');
    if strcmp(resp2, 'n')~=1 & strcmp(resp2, 'N')~=1 & strcmp(resp2, 'y')~=1 &
strcmp(resp2, 'Y')~=1
    fprintf('\n Incorrect reply, try again \n\n'), end
    respf1=resp2;
    end
    if respf1=='n' | respf1=='N', return, else, end
end
```

OUTPUT:

RESULT:

Thus the multi-machine transient stability analysis is simulated on a given power system network.

EXP. NO: 8

DATE:

ECONOMIC DISPATCH IN POWER SYSTEMS

AIM :

To understand the fundamentals of economic dispatch and solve the problem using classical method with and without line losses.

Mathematical Model for Economic Dispatch of Thermal Units

Statement of Economic Dispatch Problem

In a power system, with negligible transmission loss and with N number of spinning thermal generating units the total system load PD at a particular interval can be met by different sets of generation schedules

$\{PG_1^{(k)}, PG_2^{(k)}, \ldots\ldots\ldots\ldots\ldots PG_N^{(K)}\}$; $k = 1,2,.....NS$

Out of these NS set of generation schedules, the system operator has to choose the set of schedules, which minimize the system operating cost, which is essentially the sum of the production cost of all the generating units. This economic dispatch problem is mathematically stated as an optimization problem.

Given : The number of available generating units N, their production cost functions, their operating limits and the system load PD,

To determine : The set of generation schedules,

$$PG_i ; \qquad i = 1,2\ldots\ldots\ldots N \qquad \text{——} \quad (\mathbf{1})$$

Which minimize the total production cost,

$$\text{Min} ; F_T = \sum_{i=1}^{N} F_i(PG_i) \qquad \text{——} \quad (\mathbf{2})$$

and satisfies the power balance constraint

$$\phi = \sum_{i=1}^{N} PG_i - PD = 0 \qquad \text{——} \qquad (3)$$

and the operating limits

$$PG_{i,min} \le PG_i \le PG_{i,,max} \qquad \text{——} \qquad (4)$$

The units production cost function is usually approximated by quadratic function

$$F_i(PG_i) = a_i PG^2_i + b_i PG_i + c_i \;;\quad i = 1,2,\ldots\ldots N \qquad \text{——} \qquad (5)$$

where a_i , b_i and c_i are constants

Necessary conditions for the existence of solution to ED problem

The ED problem given by the equations (1) to (4). By omitting the inequality constraints (4) tentatively, the reduce ED problem (1),(2) and (3) may be restated as an unconstrained optimization problem by augmenting the objective function (1) with the constraint ϕ multiplied by LaGrange multiplier, λ to obtained the LaGrange function, L as

$$\text{Min : } L(PG_1 \ldots\ldots PG_N, \lambda) = \sum_{i=1}^{N} F_i(PG_i) - \lambda \left[\sum_{i=1}^{N} PG_i - PD\right] \qquad \text{——} \qquad (6)$$

The necessary conditions for the existence of solution to (6) are given by

$$\partial L / \partial PG_i = 0 = dF_i(PG_i) / dPG_i - \lambda \;;\quad i = 1, 2,\ldots\ldots N \qquad \text{——} \qquad (7)$$

$$\partial L / \partial \lambda = 0 = \sum_{i=1}^{N} PG_i - PD \qquad \text{——} \qquad (8)$$

The solution to ED problem can be obtained by solving simultaneously the necessary conditions (7) and (8) which state that the economic generation schedules not only satisfy the system power balance equation (8) but also demand that the incremental cost rates of all the units be equal be equal to λ which can be interpreted as “incremental cost of received power”.

When the inequality constraints (4) are included in the ED problem the necessary condition (7) gets modified as

$$dF_i(PG_i) / dPG_i = \lambda \quad \text{for} \quad PG_{i,min} \le PG_i \le PG_{i,,max}$$

$$\le \lambda \quad \text{for} \quad PG_i = PG_{i,,max}$$

$$\ge \lambda \quad \text{for} \quad PG_i = PG_{i,,mi} \qquad \text{____(9)}$$

Economic Schedule

$$PG_i = (\lambda - b_i)/ 2a_i ; \quad i=1,2\ldots\ldots\ldots\ldots N \qquad \text{——} \quad \textbf{(10)}$$

Incremental fuel cost

$$\lambda = PD + \Sigma(b_i/2a_i) / \Sigma (1/2a_i) \qquad \text{——} \quad \textbf{(11)}$$

($i=1$ to N for the first sum; $i=1$ to N for the second sum)

ALGORITHM:

1. Read the total number of generating units, power demand, fuel cost and B_{mn}, co-efficient.
2. Find the initial value of lambda by using the given formula

$$\lambda = \frac{PD + \sum_{i=1}^{N} \frac{b_i}{2a_i}}{\sum_{i=1}^{N} \frac{1}{2a_i}}$$

3. Calculate the value of

$$P_{Gi} = (\lambda - b_i)/ 2a_i$$

4. If $P_{Gi} > P_{\max}$, then $P_{Gi} = P_{\max}$
   If $P_{Gi} < P_{\min}$, then $P_{Gi} = P\min$

5. Calculate the change in power,

$$\Delta p = \left| \Sigma P_{Gi} - P_D \right| .$$

6. If $\Delta P < 0.0001$, then stop. Otherwise go to next step.
7. Calculate change in lamb(d*N*a.

$$\Delta\lambda = \Delta P / \left(\sum_{i=1} (1/2a_i) \right)$$

8. If $\sum P_{Gi} < P_D$, $\lambda = \lambda + \Delta\lambda$
   Otherwise, $\lambda = \lambda - \Delta\lambda$

9. Read the total number of generating units, power demand, fuel cost and B_{mn} co-efficient.
10. Find the initial value of lambda by using the given formula

$$\lambda = \frac{P_D + \sum_{i=1}^{N} \frac{b_i}{2a_i}}{\sum_{i=1}^{N} \frac{1}{2a_i}}$$

11. Calculate the value of

$$P_{Gi} = \left(1 - (b_i/\lambda) - \sum_{j=1}^{N} \left(2 * B_{ij} * P_{Gi}\right)/\left((2a_i/\lambda) + (2 * B_{ii})\right)\right)$$

12. If $P_{Gi} > P_{\max}$, then $P_{Gi} = P_{\max}$
    If $P_{Gi} < P_{\min}$, then $P_{Gi} = P\min$

13. Calculate the transmission loss

$$P_L = \sum_{i=1}^{N} \sum_{j=1}^{N} \left(P_{Gi} * B_{ij} * P_{Gj}\right.$$

14. Calculate the change in power,

$$\Delta p = \left|\sum P_{Gi} - P_L - P_D\right|$$

15. If $\Delta P < 0.0001$, then stop. Otherwise go to next step.
16. Calculate change in lambda

$$\Delta\lambda = \Delta P / \left(\sum_{i=1}^{N} (1/2a_i)\right)$$

17. If $\sum P_{Gi} < P_D$, $\lambda = \lambda + \Delta\lambda$
    Otherwise, $\lambda = \lambda - \Delta\lambda$

18. Stop the program.

PROCEDURE :

1. Enter the command window of the MATLAB.
2. Create a new M – file by selecting File - New – M – File
3. Type and save the program.
4. Execute the program by either pressing Tools – Run.
5. View the results.

PROBLEM1a: (WITHOUT LOSS AND GENERATING LIMITS)

A power plant has three units with the following cost characteristics:

$$C_1 = 0.004P_{G1}^2 + 5.3P_{G1} + 500Rs/hr$$
$$C_2 = 0.006P_{G2}^2 + 5.5P_{G2} + 400Rs/hr$$
$$C_3 = 0.009P_{G3}^2 + 5.8P_{G3} + 200Rs/hr$$

where P_{Gi}'s are in MW. Find the scheduling for a load of 975 MW.

Manual Calculation:

PROGRAM:

```
clc;
clear all;
n=input('Enter the number of units:');
a=zeros(n);
b=zeros(n);
c=zeros(n);
for i=1:n
    fprintf('Enter the unit %g Data \n',i);
    a(i)=input('Enter the value of a:');
    b(i)=input('Enter the value of b:');
    c(i)=input('Enter the value of c:');
end
pd=input('Enter the value of load demand:');
P=zeros(n);
sum=0; den=0;
for i=1:n
    sum=sum+(b(i)/(2*a(i)));
end
for i=1:n
    den=den+(1/(2*a(i)));
end
num=pd+sum;
lamda=num/den;
for i=1:n
    P(i)=(lamda-b(i))/(2*a(i));
end
for i=1:n
    fprintf('Optimal Generation of unit %g: %g MW\n',i,P(i));
end
fprintf('Lamda: %g \n',lamda);
for i=1:n
    unitcost=a(i)*P(i)^2+b(i)*P(i)+c(i);
    fprintf('Generation cost of unit  %g   : %g\n',i,unitcost);
end
    totalcost=0;
    for i=1:n
        totalcost=totalcost+a(i)*P(i)^2+b(i)*P(i)+c(i);
    end
    fprintf('Total generation cost : %g\n', totalcost);
```

OUTPUT:

PROBLEM1b: (WITHOUT LOSS AND WITH GENERATING LIMITS))

A power plant has three units with the following cost characteristics:

$$C_1 = 0.004P_{G1}^2 + 5.3P_{G1} + 500 Rs/hr \qquad 100 \le P_{G1} \le 450$$

$$C_2 = 0.006P_{G2}^2 + 5.5P_{G2} + 400 Rs/hr \qquad 100 \le P_{G2} \le 350$$

$$C_3 = 0.009P_{G3}^2 + 5.8P_{G3} + 200 Rs/hr \qquad 100 \le P_{G3} \le 225$$

where P_{Gi}'s are in MW. Find the scheduling for a load of 975 MW.

PROGRAM:

```
clc;
clear all;
n=input('Enter the number of units:');
a=zeros(n);
b=zeros(n);
c=zeros(n);
pmin=zeros(n);
pmax=zeros(n);
P=zeros(n);
for i=1:n
    fprintf('Enter the unit %g Data \n',i);
    a(i)=input('Enter the value of a:');
    b(i)=input('Enter the value of b:');
    c(i)=input('Enter the value of c:');
    pmin(i)=input('Enter the minimum value of generation:');
    pmax(i)=input('Enter the maximum value of generation:');
end
pd=input('Enter the value of load demand:');
sum=0; den=0;
for i=1:n
    sum=sum+(b(i)/(2*a(i)));
    den=den+(1/(2*a(i)));
end
num=pd+sum;
lamda=num/den;
t=1;
while t>0
    for i=1:n
        P(i)=0;
    end
    PG=0;
for i=1:n
    P(i)=(lamda-b(i))/(2*a(i));
end
for i=1:n
    if(P(i)>pmax(i))
        P(i)=pmax(i);
    elseif P(i)<pmin(i)
        p(i)=pmin(i);
    else
        P(i)=P(i);
    end
    PG=PG+P(i);
end
delp=pd-PG;
dellamda=delp/den;
if PG<pd
    lamda=lamda+dellamda;
else
    lamda=lamda-dellamda;
end
if delp<0.0001
    t=0;
end
```

```
end
for i=1:n
    fprintf('Optimal Generation of unit %g: %g MW\n',i,P(i));
end
fprintf('Lamda: %g \n',lamda);
for i=1:n
    unitcost=a(i)*P(i)^2+b(i)*P(i)+c(i);
    fprintf('Generation cost of unit  %g   : %g\n',i,unitcost);
end
    totalcost=0;
    for i=1:n
        totalcost=totalcost+a(i)*P(i)^2+b(i)*P(i)+c(i);
    end
    fprintf('Total generation cost : %g\n', totalcost);
```

OUTPUT

PROBLEM 1c: (WITH LOSS AND GENERATING LIMITS))

A power plant has three units with the following cost characteristics:

$$C_1 = 0.008P_{G1}^2 + 7P_{G1} + 200 Rs/hr \qquad 10 \le P_{G1} \le 85$$

$$C_2 = 0.009P_{G2}^2 + 6.3P_{G2} + 180 Rs/hr \qquad 10 \le P_{G2} \le 80$$

$$C_3 = 0.007P_{G3}^2 + 6.8P_{G3} + 140 Rs/hr \qquad 10 \le P_{G3} \le 70$$

$PL = 0.0218\, P_{G1}^2 + 0.0228\, P_{G2}^2 + 0.0179\, P_{G3}^2$.

where P_{Gi}'s are in MW. Find the scheduling for a load of 150 MW.

PROGRAM:

```
clc;
clear all;
n = input('Enter the no. of Units : ');
a = zeros(n);
b = zeros(n);
c = zeros(n);
pmin=zeros(n);
pmax=zeros(n);
p = zeros(n);
sum = 0;
den = 0;
bm = zeros(n,n);
for i = 1:n
    fprintf('Enter the unit %g data \n',i);
    a(i) = input('Enter the value of a : ');
    b(i) = input('Enter the value of b : ');
    c(i) = input('Enter the value of c : ');
    pmin(i)=input('Enter the MIN value of Generation : ');
    pmax(i)=input('Enter the MAX value of Generation : ');
end
for i = 1:n
    for j = 1:n
        bm(i,j) = input('Enter B Coefficient : ');
    end
end
pd = input('Enter the value of load demand : ');
for i = 1:n
    sum = sum + (b(i)/(2*a(i)));
    den = den + (1/(2*a(i)));
end
lambda = (pd + sum)/den;
t = 1;
while t >0
    fprintf('\nLambda is %g\n',lambda);
    for i = 1:n
        p(i) = 0;
    end
    pg = 0; pl
    = 0; deldde
    = 0; nb =
    0;
    for i = 1:n
        for j = 1:n
            if i~=j
                nb = nb + bm(i,j)*p(j);
            end
        end
    end
    for i = 1:n
        p(i) = (1 - b(i)/lambda - nb)/(2*(a(i)/lambda + bm(i,i)));
        fprintf('Optimal generation of unit %g is %g MW\n',i,p(i));
    end
    for i = 1:n
        for j = 1:n
```

```
            pl = pl + p(i)*bm(i,j)*p(j);
        end
    end
    fprintf('Power Loss is %g\n', pl);
    for i=1:n
        if (p(i)>pmax(i))
            p(i) = pmax(i);
        elseif (p(i)<pmin(i))
            p(i) = pmin(i);
        else
            p(i) = p(i);
        end
    end
    for i = 1:n
        pg = pg + p(i);
    end
    delp = pd + pl - pg;
    fprintf('Change in load is %g\n', delp);
    for i = 1:n
        deldde = deldde + (a(i) + b(i)*bm(i,i))/(2*((a(i) + lambda*bm(i,i))^2));
    end
    dellambda = delp/deldde;
    fprintf('Change in Lambda is %g\n',dellambda);
    lambda = lambda + dellambda;
    if delp<1
        t = 0;
    end
end
```

OUTPUT:

RESULT:

Thus the economic dispatch problem with and without loss has been written and executed.

EXP NO: 9

DATE:

LOAD – FREQUENCY DYNAMICS OF SINGLE- AREA AND TWO-AREA POWER SYSTEMS

AIM:

To become familiar with the modeling and analysis of load frequency and tie line flow dynamics of a power systems with load frequency controller (LFC) under different control modes and to design improved controllers to obtain the best system response.

OBJECTIVES:

i. To study the time response(both steady state and transient) of area frequency deviation and transient power output change of regulating generator following a small load change in a single-area power system with the regulating generator under "free governor action" for different operating conditions and different system parameters.

ii. To study the time response (both steady state and transient) of area frequency deviation and the turbine output change of regulating generator following a small load change in a single area system provided with an integral frequency controller, to study the effect of changing the gain of the controller and to select the best gain for the controller to obtain the best response.

iii. To analyze the time response of area frequency deviation and net interchange deviation following a small load change in one of the areas in an inter connected two area power system under different control modes, to study the effect of changes in controller parameters on the response and to select the optimal set of parameters for the controllers to obtain the best response under different operating conditions.

LOAD FREQUENCY CONTROL:

Primary control:

The speed change from the synchronous speed initiates the governor control action resulting in all the participating generator-turbine units taking up the change in load, and stabilizes the system frequency.

Secondary control:

It adjusts the load reference set points of selected turbine-generator units so as to give nominal value of frequency. The frequency control is a matter of speed control of the machines in the generating stations. The frequency of a power system is dependent entirely upon the speed in which the generators are rotated by their prime movers. All prime movers, whether they are steam or hydraulic turbines, are equipped with speed governors which are purely mechanical speed sensitive devices, to adjust the gate or control valve opening for the constant speed.

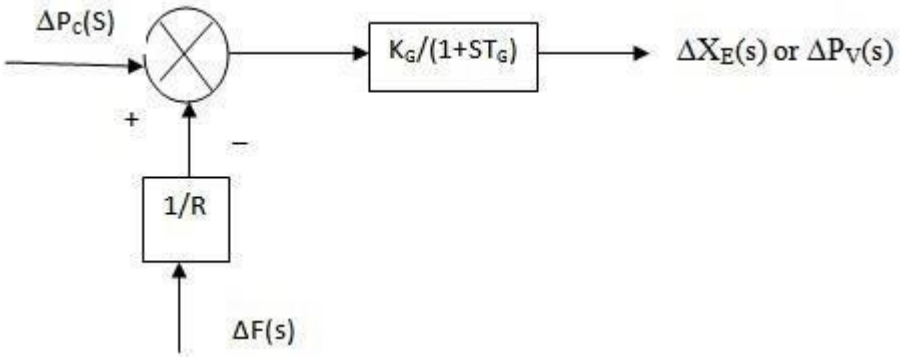
$$N = 120\, f / P$$

Therefore $N \alpha f$

where, N = Speed in rpm

f = Frequency in Hz

P = Number of poles.



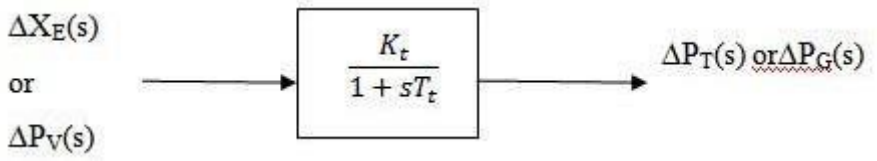
$$\Delta X_E(s) = [\Delta Pc\,(s) - (1/R)\,\Delta F(s)\,] \times \frac{k_G}{1+sT_G}$$

where $R = \frac{k_s k_c}{k_1}$ = speed regulation of the governor in Hz/MW

$k_G = \frac{k_g k_s k_c}{k_4}$ = Gain of speed governor

$T_G = \frac{1}{k_4 k_5}$ = Time constant of speed governor

Turbine model:

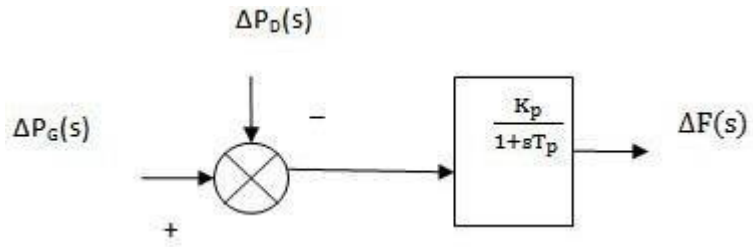


T_t = Time constant of turbine

k_t = Gain constant

$\Delta P_V(s)$ = per unit change in valve position from nominal value

Generator Load model:

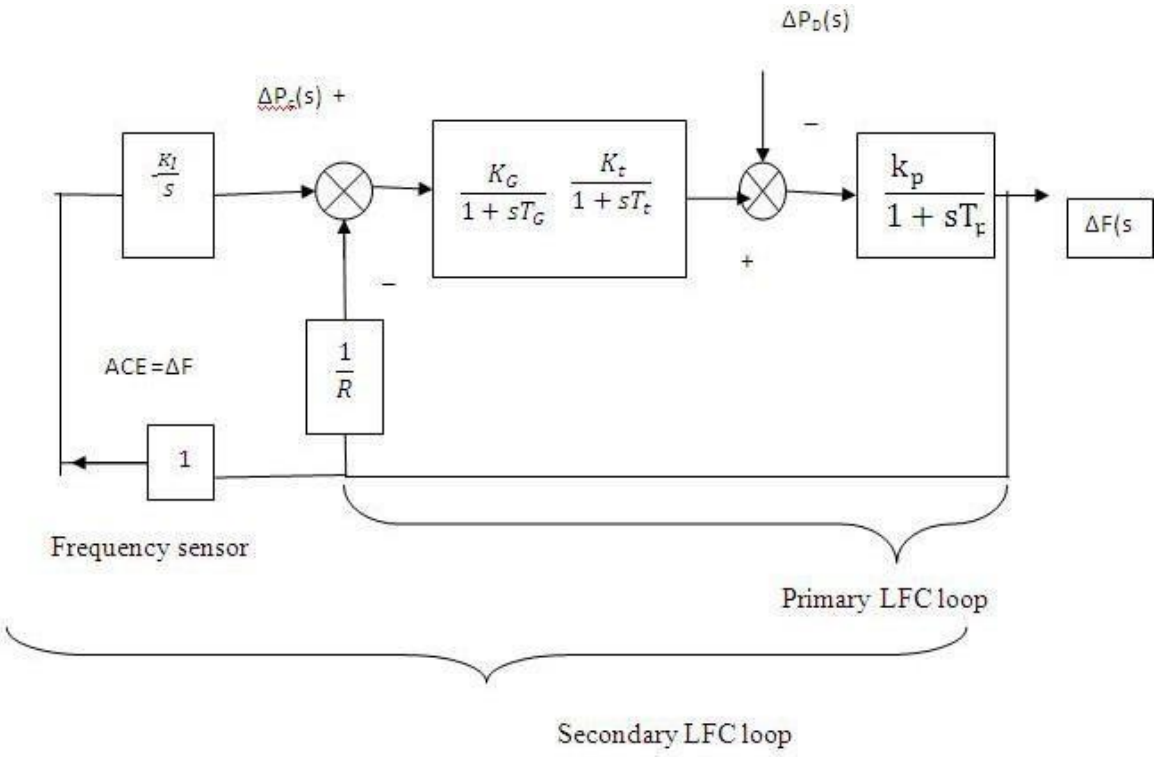


$\Delta P_D(s)$ = Real load change

$k_p = 1/B$ = Power system gain

$T_p = 2H/Bf_o$ = Power system time constant

Model of load frequency control with integral control of single area system:



PROBLEM:

1. The load dynamics of a single area system are P_r=2000 MW; N_{OL}=1000MW;H=5s;f=50Hz; R=4%; T_G=0.08s;T_T=0.3s; Assume linear characteristics . The area has governor but not frequency control. It is subjected to an increase of 20MW. Construct simulink diagram and hence i) determine steady state frequency. ii) If speed governor loop was open, what would be the frequency drop? iii) Prove frequency is zero if secondary controller is included.

PROGRAM:

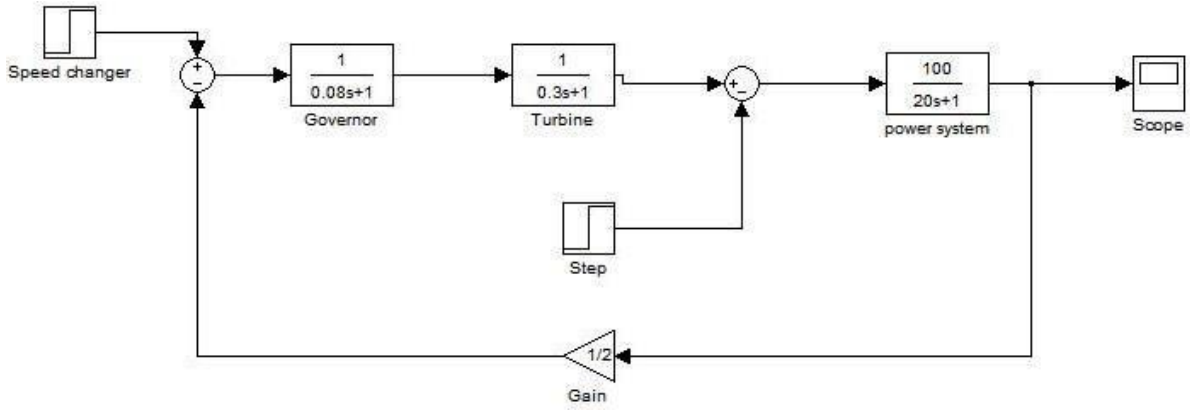
```
clear all;
clc;
rac=input('Enter the value of Related Area
capcity:'); nol=input('Enter the value of nominal
operating load:'); f=input('Enter the value of
frequency:'); cil=input('Enter the value of change
in load:');

dpd=input('Enter the value of deviation in load in percentage:');
df=input('Enter the value of deviation in frequency in percentage:');
r=input('Enter the value  of  regulation in  Hz/pu  MW:');
H=input('Enter the value of Inertia Time constant:');
D=((dpd*(nol))/(df*f));
Dpu=(D/rac);
Kp=(1/Dpu);
Tp=((2*H)/(f*Dpu));
delpd=cil/rac;
if r~= inf
    B=(Dpu+(1/r));
else
    B=(Dpu);
end
Fs=-(delpd/B);
disp('The static gain of the power system in  Hz/pu MW' );
Kp
disp('The Time constant of the power system in  seconds');
Tp
disp('Steady state frequency deviation in Hz');
Fs
```

OUTPUT:

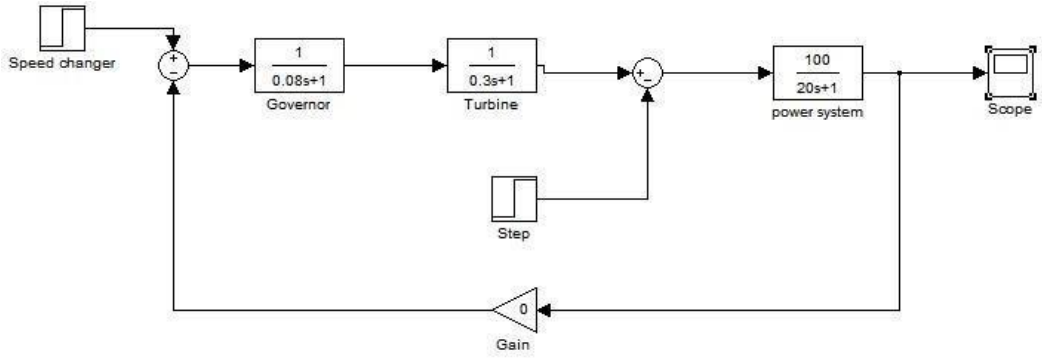
OUTPUT:

WITH GAIN



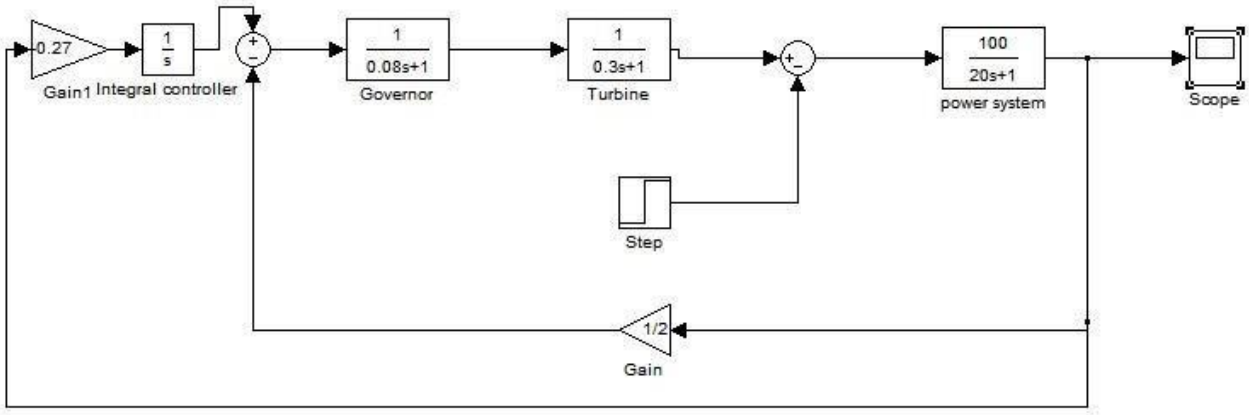
Frequency response

WITHOUT GAIN



Frequency response

WITH INTEGRAL CONTROLLER



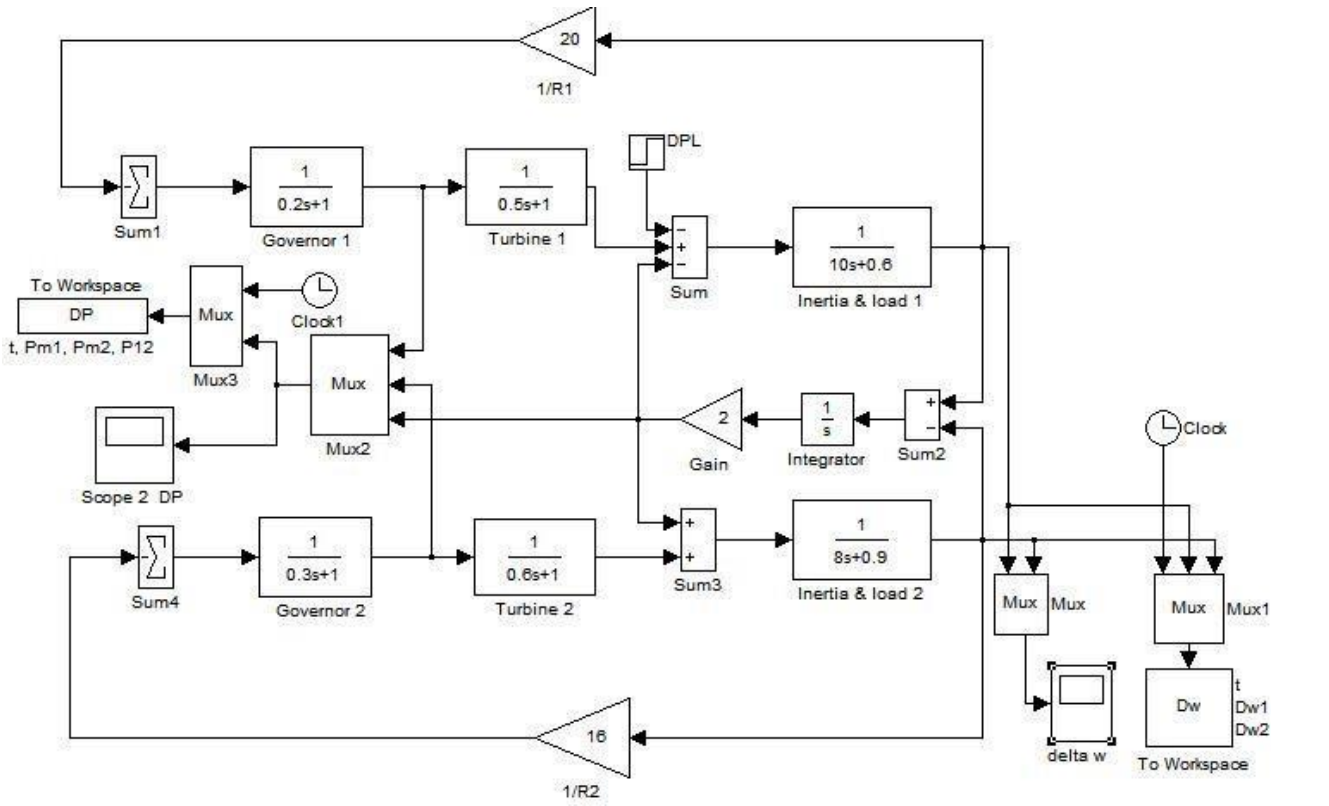
FOR TWO AREA SYSTEMS

1. A two area system connected by a tie line has the following parameters on a 1000 MVA base. R_1=0.05pu, R_2=0.0625pu,D_1=0.6, D_2=0.9,H_1=5,H_2=4; Base power$_1$=Base power$_2$=1000MVA, T_{G1}=0.2s, T_{G2}=0.3s, T_{T1}=0.5s,T_{T2}=0.6s. The units are operating in parallel at the nominal frequency of 50Hz. The synchronizing power coefficient is 2pu. A load change of 200MW occurs in area1. Find the new steady state frequency and change in the tie line flow. Construct simulink block diagram and find deviation in frequency response for the condition mentioned.
   PROGRAM:

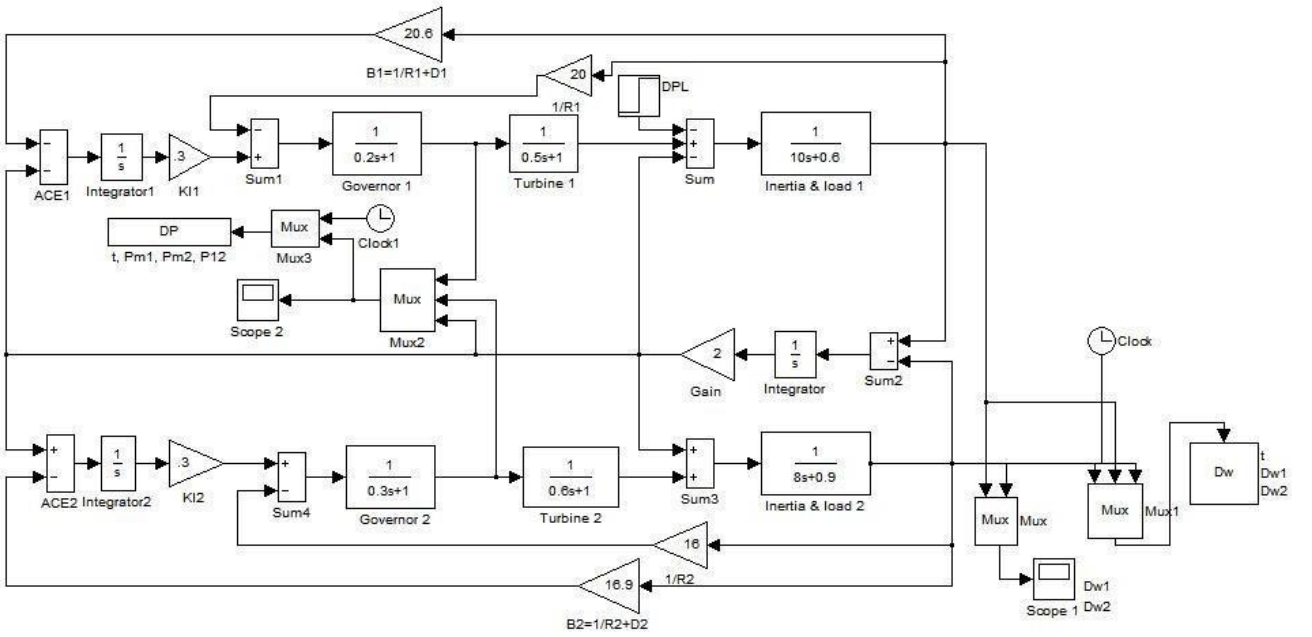
```
clear all;
clc;
rac1=input('Enter the value of Related area capacity-1 in MW:');
cil1=input('Enter the value of change in load-1 in MW:');
D1=input('Enter the value of damping coefficient-1 in pu:');
r1=input('Enter teh value of regulation-1 in pu:');
rac2=input('Enter the value of Related area capacity-2 in MW:');
cil2=input('Enter the value of change in load-2 in MW:');
D2=input('Enter the value of damping coefficient-2 in pu:');
r2=input('Enter teh value of regulation-2 in pu:');
f0=input('Enter the value of frequency:');
B1=(D1+(1/r1));
B1=B1/f0;
B2=(D1+(1/r2));
B2=B2/f0;
delpd1=cil1/rac1;
delpd2=cil2/rac2;
a12=-(rac1/rac2);
delf=((a12*delpd1)-delpd2)/(B2-(a12*B1));
f=f0+delf;
delptie=((B1*delpd2)-(B2*delpd1))/(B2-(a12*B1));
delptie=delptie*rac1;
fprintf('Steady state frequency deviation is           : %g    Hz\n',delf);
fprintf('System frequency                  :%g   Hz\n',f);
```

```
fprintf('Change in tie line flow from area 1 to area 2      :%g      MW\n',
delptie);
```

OUTPUT:



2. A two area system connected by a tie line has the following parameters on a 1000 MVA base. R_1=0.05pu, R_2=0.0625pu, D_1=0.6, D_2=0.9, H_1=5, H_2=4; Base power$_1$=Base power$_2$=1000MVA, T_{G1}=0.2s, T_{G2}=0.3s, T_{T1}=0.5s, T_{T2}=0.6s. The units are operating in parallel at the nominal frequency of 50Hz. The synchronizing power coefficient is 2pu. A load change of 200MW occurs in area1. Find the new steady state frequency and change in the tie line flow. Construct simulink block diagram **with the inclusion of the ACE's** and find deviation in frequency response for the condition mentioned.



RESULT:

Thus the modeling and analysis of load frequency and tie line flow dynamics of a power systems with load frequency controller (LFC) under different control modes and to design improved controllers to obtain the best system response was done using Matlab simulink.

EXP NO:10

DATE:

ELECTROMAGNETIC TRANSIENTS IN POWER SYSTEMS

AIM:

(i) To study and understand the electromagnetic transient phenomena in power systems caused due to switching and fault by PSCAD software.

ii) To become proficient in the usage of PSCAD software to address problems in the areas of overvoltage protection and mitigation and insulation coordination of EHV systems.

OBJECTIVES:

a) To study the transients due to energization of a single-phase and three-phase load from a non-ideal source with line represented by π model.

b) To study the transients due to energization of a single-phase and three-phase load from a non-ideal source and line represented by distributed parameters.

c) To study the transient over voltages due to faults for a SLG fault at far end of a line.

d) To study the Transient Recovery Voltage (TRV) associated with a breaker for a

Solution Method for Electromagnetic Transients Analysis

Intentional and inadvertent switching operations in EHV systems initiate over voltages, which might attain dangerous values resulting in destruction of apparatus. Accurate computation of these over voltages is essential for proper sizing, coordination of insulation of various equipments and specification of protective devices. Meaningful design of EHV systems is dependent on modeling philosophy built into a computer program. The models of equipment's must be detailed enough to reproduce actual conditions successfully – an important aspect where a general purpose digital computer program scores over transient network analyzers. The program employs a direct integration time-domain technique evolved by Dommel. The essence of this method is discretization of differential equations associated with network elements using trapezoidal rule of integration and solution of the resulting difference equations for the unknown voltages. Any network which consists of

interconnections of resistances, inductances, capacitances, single and multiphase π circuits, distributed parameter lines, and certain other elements can be solved. To keep the explanations simple, however single phase network elements will be used rather than the more complex multiphase network elements.

SOLUTION METHOD FOR ELECTROMAGNETIC TRANSIENTS ANALYSIS:

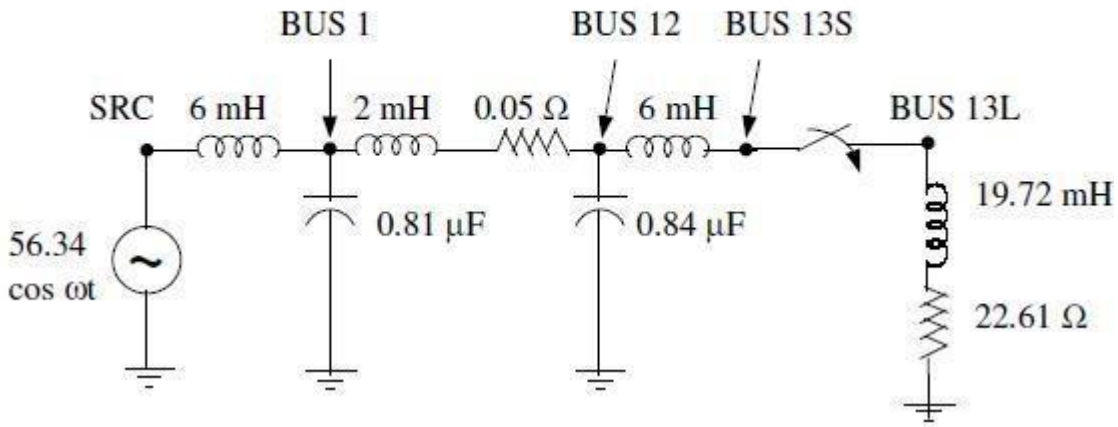
Intentional and inadvertent switching operations in EHV systems initiate over voltages, which might attain dangerous values resulting in destruction of apparatus. Accurate computation of these voltages is essential for proper sizing, co-ordination of insulation of various equipments and specification of protective devices. Meaningful use of EHV is dependent on modeling philosophy built into a computer program. The models of equipments must be detailed enough to reproduce actual conditions successfully –an important aspect where a general purpose digital computer program scores over transient network analyzers.

The program employs a direct integration time-domain technique evolved by Dommel. The essence of this method is discretization of differential equations associated with network elements using trapezoidal rule of integration and solution of the resulting equations for the unknown voltages. Any network which consists of interconnections of resistances, inductances, capacitances, single and multiphase pi circuits, distributed parameter lines, and certain other elements can be solved. To keep the explanations simple, however, single phase network elements will be used, rather than the complex multiphase network elements.

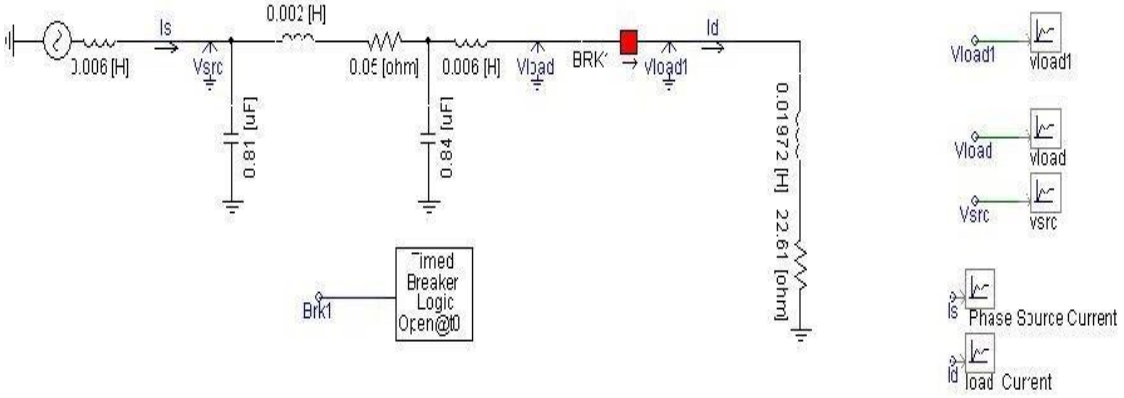
PROBLEM

1. Prepare the data for the network given in the given figure and run PSCAD software .Obtain the plots of source voltage, load bus voltage and load current following the energization of a single-phase load. Comment on the results. Double the source inductance and obtain the plots of the variables mentioned earlier. Comment on the effect of doubling the source inductance. Energization of a single phase 0.95 pf load from a non ideal source and a more realistic line representation (lumped R,L,C) using PSCAD software.

(i) Circuit Diagram



PSCAD MODEL



OUTPUT:

RESULT:

Thus the electromagnetic transient phenomena in power systems caused due to switching and fault by using PSCAD software are analyzed and results were obtained.