**Impedance and Capacitances of Transmission lines (Short. Medium, Long)**

clear all

clc

D=input('Enter the length of the transmission line')

if(D<=50)

r=input('Enter the value of resistance')

X=input('Enter the inductive resistance')

Vr=input('Enter the receiving end voltage')

Pr=input('Enter the receiving end power')

Pfr=input('Enter the receiving end powerfactor')

Ir=Pr/(Vr\*Pfr)

a=cos(Pfr)

Vs=Vr+(Ir\*r\*Pfr)+(Ir\*X\*sin(a))

Pfs=((Vs\*Pfr)+(Ir\*r))/Vs;

reg=(((abs(Vs)-abs(Vr))\*100)/abs(Vr))

loss=(Ir^2\*r)

eff=(Pr/(Pr+loss))\*100

elseif(D>50 && D<=100)

r=input('Enter the value of resistance')

X=input('Enter the inductive resistance')

Y=input('Enter the susceptance in mho')

Vr=input('Enter the receiving end voltage')

Pr=input('Enter the receiving end power')

Pfr=input('Enter the receiving end powerfactor')

Vr1=(Vr/(3^(0.5)))

Ir=(Pr/(3^(1/2)\*Vr\*Pfr))

a=Pfr

b=sqrt(1-(Pfr^2))

Irc=(Pfr-b\*i)

Ir1=Ir\*Irc

Z=r+(i\*X)

Ic1=(i\*Vr1\*Y)/2

I1=Ir1+Ic1

Vs=Vr1+(I1\*Z)

Ic2=(i\*Vs\*Y)/2

Is=I1+Ic2

d=radtodeg(angle(Vr))-radtodeg(angle(Vs))

c=radtodeg(angle(Vr))- radtodeg(angle(Is))

Pfa=c-d

Pfs=cos(Pfa)

reg=((abs(Vr1)-abs(Vs))/abs(Vs))\*100

Ps=3\*abs(Vs)\*abs(Is)\*Pfs

eff=(Pr/Ps)\*100

elseif(D>100)

r=input('Enter the value of resistance')

X=input('Enter the inductive resistance')

Y=input('Enter the susceptance in mho')

Vr=input('Enter the receiving end voltage')

Pr=input('Enter the receiving end power')

Pfr=input('Enter the receiving end powerfactor')

Z=r+(i\*X)

Vr1=(Vr/(3^(0.5)))

Ir=(Pr/(3^(0.5)\*Vr\*Pfr))

a1=1+((Y\*Z)\*(0.5))+(Y\*Y\*Z\*Z\*(1/24))

b1=((Y\*Z)^(0.5))+((Y\*Z)^(3/2)/6)

Vs=(Vr1\*a1)+(Ir\*b1\*(Z/Y)^(0.5))

Is=(Vr1\*(Y/Z)^(0.5)\*b1)+(Ir\*a1)

Pfs=cosd((radtodeg(angle(Vs)))-radtodeg(angle(Is)))

Ps=3\*abs(Vs)\*abs(Is)\*Pfs

reg=((abs(Vs)-abs(Vr1))/abs(Vr1))\*100

eff=(Pr/Ps)\*100

end

**Impedance and Admittance Matrix**

clear all

clc

n=input('Enter the no of buses');

m=input('Enter 1 for impedance and 2 for admittance');

if m==1

for i=1:n

for j=i+1:n

fprintf('Enter the impedance value b/w bus %d and bus %d',i,j);

z(i,j)=input(' ');

if(z(i,j)~=0)

a(i,j)=1/(z(i,j));

a(j,i)=a(i,j);

else

a(i,j)=0;

end

end

end

end

if m==2

for i=1:n

for j=i+1:n

fprintf('Enter the admittance value b/w bus %d and bus %d',i,j);

a(i,j)=input(' ');

a(j,i)=a(i,j);

end

end

end

sh=input('Enter 1 for shunt admittance and 2 for ignoring shunt admittance');

if sh==1

if m==2

for i=1:n

fprintf('Enter the shunt impedance value b/w bus %d and bus %d',i,i);

z(i,i)=input(' ');

if(z(i,i)~=0)

a(i,i)=1/z(i,i);

else

a(i,i)=0;

end

end

end

else

end

for i=1:n

for j=1:n

if (i~=j)

a(i,i)=a(i,i)+a(i,j);

y(i,j)=-a(i,j);

end

end

y(i,i)=a(i,i);

end

fprintf('\n Y BUS is :\n')

disp(y)

remove=input('Enter 1 to remove a node and 2 for exit');

if remove==1

p=input('Enter the node to be eliminated:');

if p==n

for i=1:n-1

for j=1:n-1

ynew(i,j)=y(i,j)-((y(i,p)\*y(p,j))/y(p,p));

end

end

fprintf('\n The new Y BUS is \n');

disp(ynew)

else

y([p n],:)=y([n p],:);

y(:,[p n])=y(:,[n p]);

a=n;

for i=1:n-1

for j=1:n-1

ynew(i,j)=y(i,j)-((y(i,a)\*y(a,j))/y(a,a));

end

end

end

fprintf('\n The new Y BUS is \n')

disp(ynew)

else

end

**Unsymmetrical Fault**

display('1.Single line to Ground Fault');

display('2.Line to line fault');

display('3.Double line to ground fault');

display('4.Exit');

ch=input('Enter your choice');

MVA=input('Enter the value of base MVA');

KV= input('Enter the value of base KV');

vl=input('Enter the line voltage');

vph=vl/sqrt(3);

Ibb=MVA/(sqrt(3)\*KV);

E=1+0i;

display('Let E be 1 p.u');

a=-0.5+0.866i;

a1=-0.5-0.866i;

switch(ch)

case 1

z1=input('Enter the positive sequence impedance in p.u');

z2=input('Enter the negative sequence impedance in p.u');

z0=input('Enter the zero sequence impedance in p.u');

zf=input('Enter fault impedance');

Ib=0;

Ic=0;

Ia1=E/(z1+z2+z0)+(3\*zf)

Ia2=Ia1

Ia0=Ia1

va1=E-(z1\*Ia1)

va2=-(z2\*Ia2)

va0=-(z0\*Ia0)

If=3\*abs(Ia1)

puIf=If\*Ibb

va=va0+va1+va2

vb=va0+(a1\*va1)+(a\*va2)

vc=va0+(a\*va1)+(a1\*va2)

vab=va-vb

vbc=vb-vc

vca=vc-va

puvab=abs(vab\*vph)

puvbc=abs(vbc\*vph)

puvca=abs(vca\*vph)

case 2

z1=input('Enter the positive sequence impedance in p.u');

z2=input('Enter the negative sequence impedance in p.u');

z0=input('Enter the zero sequence impedance in p.u');

zf=input('Enter fault impedance');

Ia0=0;

Ia1=E/(z1+z2+zf)

Ia2=-Ia1

Ia=Ia0+Ia1+Ia2

Ib=Ia0+(Ia1\*a1)+(a\*Ia2)

Ic=-Ib

If=(-sqrt(3)\*Ia1)

va0=0;

va1=E-(z1\*Ia1)

va2=va1

va=va0+va1+va2

vb=va0+(a1\*va1)+(a\*va2)

vc=vb

vab=va-vb

vbc=vb-vc

vca=vc-va

puvab=(vab\*vph)

puvbc=(vbc\*vph)

puvca=(vca\*vph)

puIf=Ibb\*If

puIa=Ibb\*Ia

puIb=Ibb\*Ib

puIc=-puIb

pause

case 3

z1=input('Enter the positive sequence impedance in p.u');

z2=input('Enter the negative sequence impedance in p.u');

z0=input('Enter the zero sequence impedance in p.u');

zf=input('Enter fault impedance');

vb=0;

vc=0;

Ia1=E/(z1+((z2\*z0)/(z2+z0)))

va1=E-(Ia1\*z1)

va0=va1

va2=va0

va1=va2

Ia2=-va2/z2

Ia0=-va0/z0

Ia=Ia0+Ia1+Ia2

Ib=Ia0+(Ia1\*a1)+(a\*Ia2)

Ic=Ia0+(Ia1\*a)+(a1\*Ia2)

If=Ib+Ic

va=va0+va1+va2

vab=va-vb

vbc=vb-vc

vca=vc-va

puIb=abs(Ibb)\*Ib

puIf=abs(Ibb)\*Ic

puvab=vab\*vph

puvca=vca\*vph

pause

case 4

exit;

end

**Symmetrical Fault**

clc

clear all

n=input('Enter no of elements');

display('Enter the reactance in order');

z=zeros(n);

for i=1:n

z(i)=input('');

end

e=1+0i;

f=input('Enter the fault bus');

zf=input('Enter the fault impedance');

z1=0;

z2=0;

for i=1:n;

if(i<=f)

z1=z1+z(i);

else

z2=z2+z(i);

end

end

z1

z2

z3=(z1\*z2)/(z1+z2);

display(z3);

cf=(e/(z3+zf));

display(cf);

ig1=(cf\*z2)/(z1+z2);

display(ig1);

ig2=(cf\*z1)/(z1+z2);

display(ig2);

z4=0;

dv=zeros(n);

vf=zeros(n);

for i=1:n-1

z4=z(i)+z4;

dv(i)=-(z4\*ig1);

end

display(dv);

for i=1:n-1

vf(i)=1+dv(i);

end

display(vf);

display('Post fault current');

for i=2:n-1

ip=(vf(i-1)-vf(i))/z(i);

end

scc=-(1\*1)/(z3+zf);

display(ip);

display(scc);

**Newton Raphson Method**

clear all

clc

n=input('Enter the number of buses');

y=zeros(n,n);

display('Enter the admittance values');

for i=1:n

for j=1+i:n

fprintf('Enter the admittance value b/w bus %d and %d',i,j);

a(i,j)=input('');

a(j,i)=a(i,j);

end

end

for i=1:n

fprintf('Enter the half line charging admittance value for bus %d',i);

a(i,i)=input('');

end

for i=1:n

for j= 1:n

if(i~=j)

a(i,i)=a(i,i)+a(i,j);

y(i,j)=-a(i,j);

end

end

y(i,i)=a(i,i);

end

y

ymag=abs(y)

yangle=angle(y)

a=input('Enter the number of iterations');

for i=1:n

v(i)=input('Enter the input voltage');

vmag=abs(v);

vangle=angle(v);

end

vmag

vangle

for i=1:a

j=1;

for j=1:n-1

j=j+1;

p(j)=0;

for k=1:n

p(j)=p(j)+(abs(v(j))\*abs(v(k))\*abs(y(j,k))\*cos(angle(y(j,k))+angle(v(k))-angle (v(j))));

end

p(j)

q(j)=0;

for k=1:n

q(j)=q(j)+(abs(v(j))\*abs(v(k))\*abs(y(j,k))\*sin(angle(y(j,k))+angle(v(k))-angle (v(j))));

end

q(j)=-q(j);

q(j)

c=input('Enter 1 for PV bus');

if(c==1)display('Enter the input in per unit value');

pspec=input('Enter the real power');

delp=pspec-p(j);

q1=input('Enter the minimum value of reactive power');

q2=input('Enter the maximum value of reactive power');

if(q(j)<q1||q(j)>q2)

if(q(j)<q1)

qspec=q1

fprintf('Now %lf bus will act as a load bus',j);

v(j)=1+0i;

else

qspec=q2;

end

end

delq=qspec-q(j);

else

display('Enter the input in per unit value');

pspec=input('Enter the real power');

delp=pspec-p(j);

qspec=input('Enter the reactive power');

delq=qspec-q(j);

end

end

J1=0;

J2=0;

J3=0;

J4=0;

T=0;

t=0;

display('The diagonal and off diagonal elements of J1');

for k=1:n

if k~=j

J1=J1+(abs(v(j))\*abs(v(k))\*abs(y(j,k))\*sin(angle(y(j,k))+angle(v(k))-angle (v(j))));

end

end

J1

display('The diagonal and off diagonal elements of J2');

for k=1:n

if k~=j

T=T+(abs(v(j))\*abs(v(k))\*abs(y(j,k))\*cos(angle(y(j,k))+angle(v(k))-angle (v(j))));

end

J2=2\*abs(v(j))\*abs(y(j,k))\*cos(angle(y(j,j)))+T;

end

J2

display('The diagonal and off diagonal elements of J3');

for k=1:n

if k~=j

J3=J3+(abs(v(j))\*abs(v(k))\*abs(y(j,k))\*cos(angle(y(j,k))+angle(v(k))-angle (v(j))));

end

end

J3

display('The diagonal and off diagonal elements of J4');

for k=1:n

if k~=j

t=t+(abs(v(j))\*abs(v(k))\*abs(y(j,k))\*sin(angle(y(j,k))+angle(v(k))-angle (v(j))));

end

J4=(-2\*abs(v(j))\*abs(y(j,k))\*sin(angle(y(j,j))))-t;

end

J4

J=zeros(n,n)

J=[J1 J2;J3 J4]

Jinv= inv(J)

pq=[delp;delq]

delx=Jinv\*pq

x=[angle(v(j));abs(v(j))];

x

newx=x+delx;

newx

end

**Gauss Seidel Method**

clear all

clc

temp=0;

t=0;

alpha=input('Enter the acceleration factor');

err=1;

tol=5e-5;

iter=1;

n=input('Enter no of buses');

v=ones(n,1);

p=zeros(n,1);

q=zeros(n,1);

%get Ybus

for i= 1:n

for j=1:n

fprintf('Y%d%d=',i,j);

y(i,j)=input('');

end

end

y

i=1;

%get Bus details

v(i)=input('Enter slack bus voltage');

for i=2:n

fprintf('Enter bus %d details', i);

type(i)=input('Enter the type of bus: 1 for PQ 2 for PV');

p(i)=input('p in pu=');

if type(i)==1

q(i)= input('q in pu=');

else

v(i)=input('v in pu=');

end

end

while((err>tol)&&(iter<8))

display('iter no=');

iter

vold=v;

for i=2:n

if type(i)==1

for j=1:n

if j~=i

temp= temp+(v(j)\*y(i,j));

t= t+y(i,j);

end

end

s(i)=complex(p(i),q(i));

v(i)=(((-1\*(s(i)))/conj(v(i)))+temp)/t;

temp=0;

t=0;

else

for j=1:n

if j~=i

temp=temp+(v(j)\*y(i,j));

t=t+y(i,j);

end

end

q(i)=-imag(conj(v(i))\*((v(i)\*t)-temp));

temp=0;

t=0;

for j=1:n

if j~=i

temp=temp+(v(j)\*y(i,j));

t=t+y(i,j);

end

end

s(i)=complex(p(i),q(i));

cv=(-1\*s(i))/(conj(v(i)));

vang=((cv+temp)/t);

delta=angle(vang);

vreal=abs(v(i))\*cos(delta);

vimag=abs(v(i))\*sin(delta);

v(i)=complex(vreal,vimag);

temp=0;

t=0;

end

end

v=vold+(alpha\*(v-vold));

vabs=abs(v-vold);

err=max(vabs);

fprintf('Voltages after iteration %d',iter);

v

iter=iter+1;

end

fprintf('Convergence is reached in iteration %d',iter-1);

display('Voltage after convergence=');

v

for i=1:n

for j=1:n

if i~=j

I(i,j)=y(i,j)\*(v(i)-v(j));

sf(i,j)=-100\*v(i)\*conj(I(i,j));

end

end

end

for i=1:n

for j=1:n

if i~=j

fprintf('Current in line %d %d',i,j);

I(i,j)

end

end

end

for i=1:n

for j=1:n

if i~=j

fprintf('Power flow in line %d %d',i,j);

sf(i,j)

end

end

end

Fast Decoupled Method

clc

clear all

n= input('No of buses');

z=zeros(n,n);

y=zeros(n,n);

display('Admittance');

for a=1:n

for b=1:n

y(a,b)=input('');

end

end

display(y);

for i=1:n

fprintf('Enter the P & Q values of bus %d',i);

s1(i)=input('');

end

for i=1:n

fprintf('Enter the V of bus %d',i);

v(i)=input('');

end

ps=real(s1);

qs=imag(s1);

for c=1:n

for d=1:n

mag(c,d)=abs(y(c,d));

phangle(c,d)=angle(y(c,d));

end

end

display(mag);

display(phangle);

magv=abs(v);

vangle=angle(v);

a1=zeros(n,n);

for m=2:3

for n=2:3

a1(m,n)=mag(m,n)\*(sin(phangle(m,n)));

end

end

display(a1);

a2=a1(2:3,2:3);

display(a2);

bp=inv(a2);

display(bp);

BP=zeros(n);

BP=a2(2:2,2:2);

display(BP);

BP1=inv(BP);

display(BP1);

p=zeros(n);

for j=2:n

for k=1:n

p(j)=p(j)+(magv(j)\*magv(k)\*mag(j,k)\*(cos((phangle(j,k)))));

end

end

pp=zeros(n);

pp=p(2:3);

ps1=ps(1,2:3);

delp1=ps1-pp;

display(delp1);

magv1=magv(1,2:3);

delpv=zeros(n-1);

delpv(1,1)=delp1(1,1)/magv1(1,1);

delpv(2,1)=delp1(1,2)/magv1(1,2);

delpv1=delpv(1:2,1);

display(delpv1);

deldel=-(bp\*delpv1);

display(deldel);

q1=zeros(n);

j=3

for k=1:3

q1(j)=q1(j)+(magv(j)\*magv(k)\*mag(j,k)\*(sin(phangle(j,k)-vangle(j)+vangle(k))));

end

q1(1,3)=-q1(3,1);

q1(3,1)=0;

q2=q1(1,:);

delq=qs-q2;

display(delq);

conq=max(abs(delq));

display(conq);

delqyv=conq/magv(1,3);

delv=zeros(n);

delv=-BP1\*delqyv;

display(delv);

vnew=zeros(n);

vnew=magv;

vnew(3)=magv(3)+delv;

display(vnew);

vanglenew=zeros(3,1);

vanglenew=deldel;

display(vanglenew);