Exp - 13	Transient Analysis of SMIB

Aim:

To analyse the transient performance of single machine infinite bus system.

Apparatus Required:

SI.No	Apparatus	Specification
1	PC	Dual core, RAM 512 MB 1.2 GHz speed, 80 GB
2	MATLAB	7.5

Algorithm:

Step1: Start the program

Step2: Determine the external reactance using the formula

$$X_E = X_T + \frac{X_1 X_2}{X_1 + X_2}$$

Step3: Calculate the total reactance of the circuit

$$X = X_E + X_D^{'}$$

Step4: Find the transient internal voltage by using the formula $E' = V + jX_1(I)$

Step5: Calculate the power angle

P_{max} sin δ= E_V/X sin δ

Cos δ_C =P_m/P_{max} (δ_{max}-δ_0)+cos δ_{max}

Step6: find the critical clearing time

$$\rho_C = \sqrt{\frac{2H(\delta_C - \delta_o)}{H + P_m}}$$

Step7:Fault occurs at the middle of the line is calculated by using the formula

$$Cos\,\delta_C = \frac{P_M(\delta_{\max} - \delta_o) + P_{\max} \times \cos\delta_{\max} - P_{2\max}\cos\delta 0}{P_{3\max} - P_{2\max}}$$

Step8: Display the program

Step9: Stop the program

Program:

```
clc;
clear all;
pm=input('generator output power in pu MW:');
E=input('generator emf in pu v:');
v=input('infinite bus bar voltage in pu V:');
x1=input('reactance before fault in pu x1:');
x2=input('reactance during fault in pu x2:');
x3=input('reactance after fault is cleared in pu x3:');
pe1max=E*v/x1;
pe2max=E*v/x2;
pe3max=E*v/x3;
delta=0:0.01: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 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 could be found\n');
   fprintf('system can remain stable during this disturbance \n\n');
   return
else
end
d0=d0*180/pi;
dmax=dmax*180/pi;
dc=dc*180/pi;
if x2==inf
   fprintf('For 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
fprintf('\n Initial power angle=%7.3f\n',d0);
fprintf('\n maximum angle swing=%7.3f\n',dmax);
fprintf('\n Critical clearing angle=%7.3f\n\n',dc);
if x2==inf&H~=0
   fprintf('\n Critical clearing time=%7.3fsec \n\n',tc);
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(dmax)];
xm=[dmax dmax]*180/pi;
ym=[0 pe3max*sin(dmax)];
x=(d0:0.1:dc);
y=pe2max*sin(x*pi/180);
y1=pe2max*sin(d0*pi/180);
y2=pe2max*sin(d0*pi/180);
x=[d0 x dc];
y=[pm y pm];
xx=dc:0.1:dmax;
h=pe3max*sin(xx*pi/180);
xx=[dc xx dmax];
hh=[pm h pm];
delta=delta*180/pi;
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,perunit')
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;
```

Result: