Laboratory Experiment 10

Transient Stability using MATLAB

Objectives: To simulate transient stability in a 3-bus example power system.

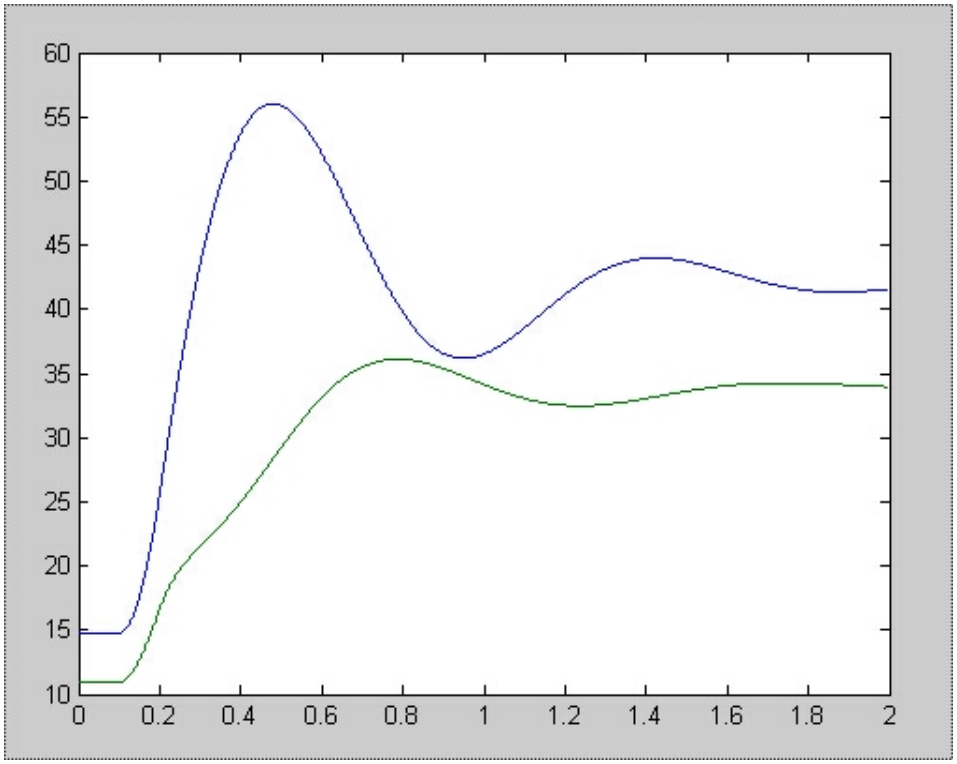
Laboratory Tasks and Report:

The MATLAB file to calculate Transient Stability in the example 3-bus power system is **TransientStability.m**, which is included in this folder. This file has a power flow to initialize the simulation and then three separate simulations for prefault, during the fault, and post fault dynamics. The program initially simulates a three phase fault on line 1-2 at 1/3 of the distance from bus 1 to bus 2. See video clip# 14.

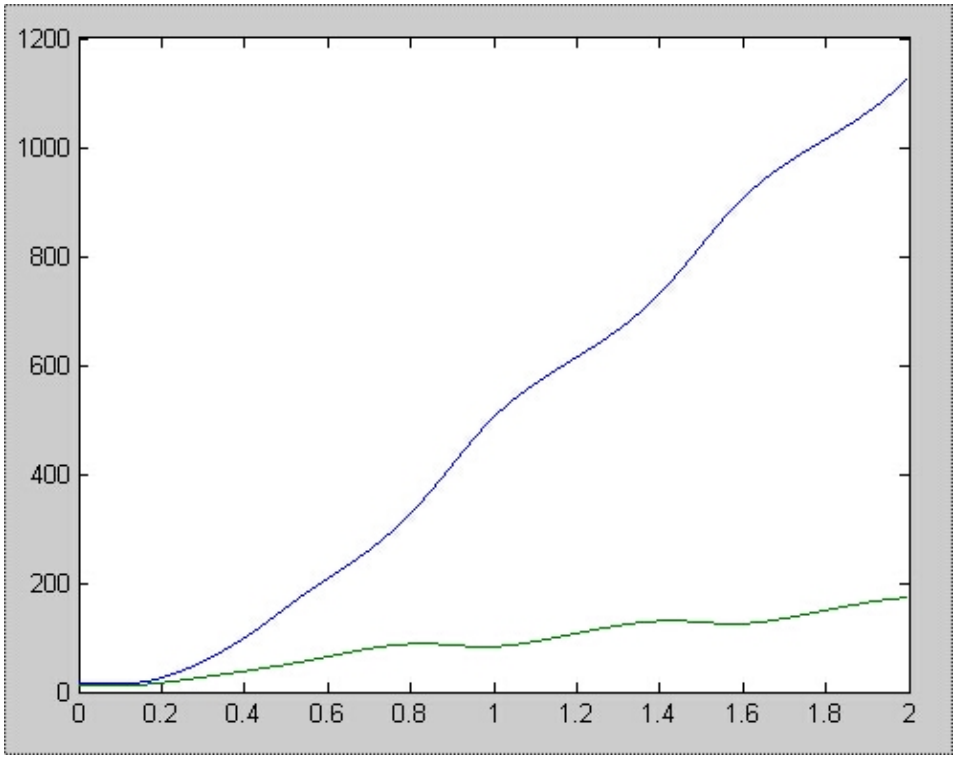
a. Annotate this file based on the material and equations of Chapter 11.
b. Execute this file get the plots of rotor angles of generators 1 and 2 (see figures below)
    i. Note that the clearing time is shown in seconds at line 125 of the program.
    ii. When you get the program the clearing time is set to 0.2 second which results in a stable system.
    iii. Start increasing the clearing time until the system goes unstable. Try to determine the "critical clearing time" which is the amount of time that can be allowed before the system goes unstable. Save the plots and note the maximum swing angle for each clearing time you try.
c. Next you are going to convert generator 2 to an "infinite" generator by setting its H constant to a very large number. You do this by putting a % sign in front of the code where H2 is calculated and then removing the % sign in front of the next line thereby making H2 = 50000.0. Now start the clearing time at 0.2 seconds and find the critical clearing time with gen 2 as an infinite generator.
d. Assume that a three-phase fault occurs on line between buses 2 and 3, one-third away from bus 2. Modify the program to represent the fault on line 2-3 instead of line 1-2. Run with generator 2 as a normal generator (not an infinite generator). Start the clearing time at 0.2 second and then find the critical clearing time for this case.

Note that the program you have for this lab has included some damping into the differential equations of the generators. The textbook does not show this damping but damping is always present in real world systems.

Here is a plot of the two generator rotor angles when the fault is cleared quickly:



Here is a plot when the clearing time is too long:



TransientStability.m

```
% Example 11-3 Swing Curves
clear all
j = sqrt(-1);

XL_km=0.367; % ohm/km at 60 Hz
RL_km= 0.1*XL_km; % Resistance in ohm/km

KV_LL= 345; MVA_Base=100; % common 3-phase base
Z_Base=KV_LL^2/MVA_Base; % common base

% YBUS Creation
Z13_ohm=(RL_km+j*XL_km)*200; B13_Micro_Mho=4.5*200; % Line 1-3 is 200 km long
Z12_ohm=(RL_km+j*XL_km)*150; B12_Micro_Mho=4.5*150; % Line 1-2 is 150 km long
Z23_ohm=(RL_km+j*XL_km)*150; B23_Micro_Mho=4.5*150; % Line 2-3 is 150 km long
Z13=Z13_ohm/Z_Base; Z12=Z12_ohm/Z_Base; Z23=Z23_ohm/Z_Base; % line impedances in per unit

Y(1,1)=1/Z12 + 1/Z13; Y(1,2)=-1/Z12; Y(1,3)=-1/Z13;
Y(2,1)=-1/Z12; Y(2,2)=1/Z12 + 1/Z23; Y(2,3)=-1/Z23;
Y(3,1)=-1/Z13; Y(3,2)=-1/Z23; Y(3,3)=1/Z13 + 1/Z23;
G(1,1)=real(Y(1,1)); B(1,1)=imag(Y(1,1)); G(1,2)=real(Y(1,2)); B(1,2)=imag(Y(1,2));
G(1,3)=real(Y(1,3)); B(1,3)=imag(Y(1,3));
G(2,1)=real(Y(2,1)); B(2,1)=imag(Y(2,1)); G(2,2)=real(Y(2,2)); B(2,2)=imag(Y(2,2));
G(2,3)=real(Y(2,3)); B(2,3)=imag(Y(2,3));
G(3,1)=real(Y(3,1)); B(3,1)=imag(Y(3,1)); G(3,2)=real(Y(3,2)); B(3,2)=imag(Y(3,2));
G(3,3)=real(Y(3,3)); B(3,3)=imag(Y(3,3));

% Given Specifications
V1MAG=1.0; ANG1=0; V2MAG=1.03; P2sp=5.0; P3sp=-9.0; Q3sp=-4.0;

% Solution Parameters
Tolerance= 0.001; Iter_Max=10;

% Initialization
Iter=0; ConvFlag=1;
ANG2=0; ANG3=0; V3MAG=1.0;
delANG2=0; delANG3=0; delMAG3=0;

% Start Iteration Process for N-R
while( ConvFlag==1 & Iter < Iter_Max)
    Iter=Iter+1;
ANG2=ANG2+delANG2;
ANG3=ANG3+delANG3;
V3MAG=V3MAG+delMAG3;

% Creation of Jacobian J
% J(1,1)=dP2/dAng2; k=2, m=1,3
J(1,1)=V2MAG*(V1MAG*(-G(2,1)*sin(ANG2-ANG1)+B(2,1)*cos(ANG2-ANG1)) + V3MAG*(-G(2,3)*sin(ANG2-
ANG3)+B(2,3)*cos(ANG2-ANG3)));
% J(1,2)=dP2/dAng3; k=2, j=3
J(1,2)=V2MAG*(V3MAG*(G(2,3)*sin(ANG2-ANG3)-B(2,3)*cos(ANG2-ANG3)));
% J(1,3)=dP2/dMAG3; k=2, j=3
J(1,3)=V2MAG*((G(2,3)*cos(ANG2-ANG3)+B(2,3)*sin(ANG2-ANG3)));
% J(2,1)=dP3/dAng2; k=3, j=2
J(2,1)=V3MAG*(V2MAG*(G(3,2)*sin(ANG3-ANG2)-B(3,2)*cos(ANG3-ANG2)));
% J(2,2)=dP3/dAng3; k=3, m=1,2
J(2,2)=V3MAG*(V1MAG*(-G(3,1)*sin(ANG3-ANG1)+B(3,1)*cos(ANG3-ANG1)) + V2MAG*(-G(3,2)*sin(ANG3-
ANG2)+B(3,2)*cos(ANG3-ANG2)));
% J(2,3)=dP3/dMAG3; k=3, m=1,2
J(2,3)=2*G(3,3)*V3MAG + V1MAG*(G(3,1)*cos(ANG3-ANG1)+B(3,1)*sin(ANG3-ANG1)) +
V2MAG*(G(3,2)*cos(ANG3-ANG2)+B(3,2)*sin(ANG3-ANG2));
% J(3,1)=dQ3/dAng2; k=3, j=2
J(3,1)=V3MAG*(V2MAG*(G(3,2)*cos(ANG3-ANG2)-B(3,2)*sin(ANG3-ANG2)));
% J(3,2)=dQ3/dAng3; k=3, m=1,2
J(3,2)=V3MAG*(V1MAG*(G(3,1)*cos(ANG3-ANG1)+B(3,1)*sin(ANG3-ANG1)) + V2MAG*(G(3,2)*cos(ANG3-
ANG2)+B(3,2)*sin(ANG3-ANG2)));
% J(3,3)=dQ3/dMAG3; k=3, m=1,2
J(3,3)=- 2*B(3,3)*V3MAG + V1MAG*(G(3,1)*sin(ANG3-ANG1)-B(3,1)*cos(ANG3-ANG1))  +
V2MAG*(G(3,2)*sin(ANG3-ANG2)-B(3,2)*cos(ANG3-ANG2));
```

```
% Voltages
V(1,1)=V1MAG*exp(j*ANG1);
V(2,1)=V2MAG*exp(j*ANG2);
V(3,1)=V3MAG*exp(j*ANG3);

% Injected currents
Iinj=Y*V;

% P and Q Injected
S(1,1)=V(1,1)*conj(Iinj(1)); S(2,1)=V(2,1)*conj(Iinj(2)); S(3,1)=V(3,1)*conj(Iinj(3));

% Mismatch at PQ and PV buses
Mismatch(1,1)=P2sp-real(S(2,1)); Mismatch(2,1)=P3sp-real(S(3,1)); Mismatch(3,1)=Q3sp-
imag(S(3,1));

% calculate new delta values for ANG2, ANG3, and MAG3
del=inv(J)*Mismatch;
delANG2=del(1); delANG3=del(2); delMAG3=del(3);
 if max(abs(Mismatch)) > Tolerance,
     ConvFlag=1;
 else
     ConvFlag=0;
 end
end
Pm1=real(S(1,1))
Pm2=real(S(2,1))
P3=-real(S(3,1))
Q3=-imag(S(3,1))
ZLoad=V(3,1)/(-Iinj(3,1));

Xtr1_PU=0.12*MVA_Base/500;
% Transformer base is 500 MVA
S_Gen1=500;

XdP1_PU=0.23*(MVA_Base/S_Gen1);
% Gen XdP is 0.23pu on the base of 500MVA and 22kVLL
S_Gen2=600;

Xtr2_PU=0.12*MVA_Base/600;
% Transformer base is 600 MVA

XdP2_PU=0.23*(MVA_Base/S_Gen2);
% Gen XdP is 0.23pu on the base of 600MVA and 22kVLL

H_Gen=3.5; wsyn=377;

H1=H_Gen*(S_Gen1/MVA_Base);
damp1 = 0.2;
X1=XdP1_PU+Xtr1_PU; % internal plus transformer impedances for gen 1

H2=H_Gen*(S_Gen2/MVA_Base);
%H2 = 50000.;
damp2 = 2.0;
X2=XdP2_PU+Xtr2_PU; % internal plus transformer impedances for gen 2

% Pre-Fault (Pre) steady state
EP1=V(1,1)+j*X1*Iinj(1,1);
EP1MAG=abs(EP1);
EP2=V(2,1)+j*X2*Iinj(2,1);
EP2MAG=abs(EP2);


DT=0.0001;              % Time step is 1*10^-4 seconds
Time_fault_start = 0.1; % start the fault at 0.1 seconds
Time_fault_clear = 0.2; % user can change the clear time, it must be greater than 0.1 sec and
less than 2.0 sec
Time_max = 2.0;         % maximum time is 2 seconds

% set up initial conditions
Pe1=Pm1; Pe2=Pm2;
```

```
Del1(1)=angle(EP1); Del2(1)=angle(EP2); w1(1)=wsyn; w2(1)=wsyn; time1(1)=0; DelREF=0;
DelDIFF1_DEG(1)=(Del1(1)-DelREF)*180/pi;
DelDIFF2_DEG(1)=(Del2(1)-DelREF)*180/pi;

% Pre-Fault Transient
% build Y matrix for all lines in, no fault
Y(1,1)=1/(j*X1)+1/Z12 + 1/Z13; Y(1,2)=-1/Z12; Y(1,3)=-1/Z13;
Y(2,1)=-1/Z12; Y(2,2)=1/(j*X2)+1/Z12 + 1/Z23; Y(2,3)=-1/Z23;
Y(3,1)=-1/Z13; Y(3,2)=-1/Z23; Y(3,3)=1/ZLoad+1/Z13 + 1/Z23;
imax = round( Time_fault_start/DT );  % imax is the number of steps to use in the pre-fault
simulation
for i=2:imax
    time1(i)=time1(i-1)+DT;
    w1(i)=w1(i-1)+(wsyn/(2*H1))*(Pm1-Pe1 - damp1*(w1(i-1) - wsyn))*DT;
    w2(i)=w2(i-1)+(wsyn/(2*H2))*(Pm2-Pe2 - damp2*(w2(i-1) - wsyn))*DT;
    Del1(i)=Del1(i-1)+w1(i-1)*DT;
    Del2(i)=Del2(i-1)+w2(i-1)*DT;
    EP1=EP1MAG*(cos(Del1(i))+j*sin(Del1(i)));
    EP2=EP2MAG*(cos(Del2(i))+j*sin(Del2(i)));
    I_Norton(1,1)=EP1/(j*X1); I_Norton(2,1)=EP2/(j*X2); I_Norton(3,1)=0;
    V=inv(Y)*I_Norton;
    Pe1=real(V(1,1)*conj(I_Norton(1,1)));
    Pe2=real(V(2,1)*conj(I_Norton(2,1)));
    DelREF=DelREF+DT*wsyn;
    DelDIFF1_DEG(i)=(Del1(i)-DelREF)*180/pi;
    DelDIFF2_DEG(i)=(Del2(i)-DelREF)*180/pi;
    DelDIFF(i)=DelDIFF1_DEG(i)-DelDIFF2_DEG(i);
end
fprintf(' %s  %5d \n',' Number of time steps in prefault = ',imax);

% During Fault Transient
% build Y matrix with fault on line 1-2
Y(1,1)=1/(j*X1)+1/(Z12/3) + 1/Z13; Y(1,2)=0; Y(1,3)=-1/Z13;
Y(2,1)=0; Y(2,2)=1/(j*X2)+1/(2*Z12/3)+ 1/Z23; Y(2,3)=-1/Z23;
Y(3,1)=-1/Z13; Y(3,2)=-1/Z23; Y(3,3)=1/ZLoad+1/Z13 + 1/Z23;
kmax = round( (Time_fault_clear - Time_fault_start)/DT );
%kmax is the number of time steps used during the fault simulation
for k=1:kmax
    i=imax+k;
    time1(i)=time1(i-1)+DT;
    w1(i)=w1(i-1)+(wsyn/(2*H1))*(Pm1-Pe1 - damp1*(w1(i-1) - wsyn))*DT;
    w2(i)=w2(i-1)+(wsyn/(2*H2))*(Pm2-Pe2 - damp2*(w2(i-1) - wsyn))*DT;
    Del1(i)=Del1(i-1)+w1(i-1)*DT;
    Del2(i)=Del2(i-1)+w2(i-1)*DT;
    EP1=EP1MAG*(cos(Del1(i))+j*sin(Del1(i)));
    EP2=EP2MAG*(cos(Del2(i))+j*sin(Del2(i)));
    I_Norton(1,1)=EP1/(j*X1); I_Norton(2,1)=EP2/(j*X2); I_Norton(3,1)=0;
    V=inv(Y)*I_Norton;
    Pe1=real(V(1,1)*conj(I_Norton(1,1)));
    Pe2=real(V(2,1)*conj(I_Norton(2,1)));
    DelREF=DelREF+DT*wsyn;
    DelDIFF1_DEG(i)=(Del1(i)-DelREF)*180/pi;
    DelDIFF2_DEG(i)=(Del2(i)-DelREF)*180/pi;
    DelDIFF(i)=DelDIFF1_DEG(i)-DelDIFF2_DEG(i);
end
fprintf(' %s  %5d \n',' Number of time steps during fault = ',kmax);

% Post Fault Transient (breakers on faulted line have opened)
% build Y matrix with line 1-2 removed
Y(1,1)=1/(j*X1)+ 1/Z13; Y(1,2)=0; Y(1,3)=-1/Z13;
Y(2,1)=0; Y(2,2)=1/(j*X2)+ 1/Z23; Y(2,3)=-1/Z23;
Y(3,1)=-1/Z13; Y(3,2)=-1/Z23; Y(3,3)=1/ZLoad+1/Z13 + 1/Z23;
nmax = round( (Time_max - Time_fault_clear)/DT );
%nmax is the number of time steps used to complete the simulation
for n=1:nmax
    i=imax+kmax+n;
    time1(i)=time1(i-1)+DT;
    w1(i)=w1(i-1)+(wsyn/(2*H1))*(Pm1-Pe1 - damp1*(w1(i-1) - wsyn))*DT;
    w2(i)=w2(i-1)+(wsyn/(2*H2))*(Pm2-Pe2 - damp2*(w2(i-1) - wsyn))*DT;
    Del1(i)=Del1(i-1)+w1(i-1)*DT;
    Del2(i)=Del2(i-1)+w2(i-1)*DT;
```

```
    EP1=EP1MAG*(cos(Del1(i))+j*sin(Del1(i)));
    EP2=EP2MAG*(cos(Del2(i))+j*sin(Del2(i)));
    I_Norton(1,1)=EP1/(j*X1); I_Norton(2,1)=EP2/(j*X2); I_Norton(3,1)=0;
    V=inv(Y)*I_Norton;
    Pe1=real(V(1,1)*conj(I_Norton(1,1)));
    Pe2=real(V(2,1)*conj(I_Norton(2,1)));
    DelREF=DelREF+DT*wsyn;
    DelDIFF1_DEG(i)=(Del1(i)-DelREF)*180/pi;
    DelDIFF2_DEG(i)=(Del2(i)-DelREF)*180/pi;
    DelDIFF(i)=DelDIFF1_DEG(i)-DelDIFF2_DEG(i);
end
fprintf(' %s  %5d \n',' Number of time steps after fault = ',nmax);

plot(time1,DelDIFF1_DEG,time1,DelDIFF2_DEG)
```