|  |  |
| --- | --- |
| **1.IMPULSE SIGNAL**  clc;  clear all;  close all;  n=input('Enter the value for time'); %20  a=input('Enter the value for Amplitude') %5  t=-n:1:n;  y=[zeros(1,n),a\*ones(1,1),zeros(1,n)];  stem(t,y);  ylabel('Amplitude');  xlabel('Time');  title('Impulse Signal'); | **2.STEP SIGNAL**  clc;  clear all;  close all;  n=input('Enter the value for time'); %20  a=input('Enter the value for Amplitude') %5  t=-n:1:n;  y=[zeros(1,n+1),a\*ones(1,n)];  plot(t,y);  ylabel('Amplitude');  xlabel('Time');  title('Step Signal'); |
| **3.RAMP SIGNAL**  clc;  clear all;  close all;  n=input('Enter the value for time'); %20  a=input('Enter the value for Amplitude') %5  t=-n:1:n;  y=a\*t;  plot(t,y);  ylabel('Amplitude');  xlabel('Time');  title('Ramp Signal'); | **4.PARABOLIC SIGNAL**  clc;  clear all;  close all;  n=input('Enter the value for time'); %20  a=input('Enter the value for Amplitude') %5  t=-n:1:n;  y=a\*(t.\*t); plot(t,y);  ylabel('Amplitude');  xlabel('Time');  title('Parabolic Signal'); |
| **5.EXPONENTIAL SIGNAL**  clc;  clear all;  close all;  n=input('Enter the value for time'); %10  a=input('Enter the value for Amplitude') %5  t=0:0.5:n;  y=a\*exp(t);  plot(t,y);  ylabel('Amplitude');  xlabel('Time');  title('Exponential Signal'); | **6.SINE SIGNAL**  clc;  clear all;  close all;  n=input('Enter the value for time'); %20  a=input('Enter the value for Amplitude') %5  t=-n:0.005:n;  y=a\*sin(t);  plot(t,y);  ylabel('Amplitude');  xlabel('Time');  title('Sine Signal'); |
| **7.COSINE SIGNAL**  clc;  clear all;  close all;  n=input('Enter the value for time'); %50  a=input('Enter the value for Amplitude') %5  t=-n:0.005:n;  y=a\*cos(t);  plot(t,y);  ylabel('Amplitude');  xlabel('Time');  title('Cosine Signal'); |  |

|  |  |
| --- | --- |
| **1. Mass-damper-spring (Mechanical System)**  M=input('Enter Mass: ');  B=input('Enter Damping Constant: ');  K=input('Enter Spring Constant: ');  F=input('Enter Force: ');  n=[F]  d=[M B K]  sys1=tf(n,d);  %impluse  subplot(2,1,1);  impulse(sys1);  title("Impluse");  ylabel('Displacement');  %Step  subplot(2,1,2);  step(sys1);  title("Step");  ylabel("Displacemnt")  %sine  t = linspace(0,10,100);  u = sin(t);  y = lsim(sys1, u, t);  figure(2)  plot(t, y)  grid  ylabel('Displacement');  xlabel('Time');  title('Sine'); | **2. Electrical System**  clc;  close all;  clear all;  V=input('Enter input voltage V=');  R=input('Enter resistance R=');  C=input('Enter capacitance C=');  s=tf('s');  sys=V/((s^2\*R^2\*C^2)+(3\*s\*R\*C)+1)  subplot (2,1,1);  impulse(sys);  title('Electrical system impulse response');  ylabel('output voltage');  subplot (2,1,2);  step(sys);  title('Electrical system step response');  ylabel('output voltage'); |
| **3. Armature controlled DC motor (Electromechanical System)**  clc;  clear all;  close all;  V=input('Enter the value for input Voltage ');  kb=input('Enter the value for back emf constant kb= ');  kt=input('Enter the value for torque constant kt= ');  r=input('Enter the value for Resistance R= ');  b=input('Enter the value for Friction Torque B= ');  j=input('Enter the value for J= ');  l=input('Enter the value for inductance L= ');  num=[V\*kt];  den=[l\*j (r\*j)+(b\*l) (r\*b)+(kb\*kt) 0];  sys=tf(num,den)  subplot (2,1,1);  impulse(sys);  title('Electromechanical system impulse response');  ylabel('angular displacement');  subplot (2,1,2);  step(sys);  title('Electromechanical system step response');  ylabel('angular displacement'); | **4. Field Controlled DC Motor**  clc;  clear all;  close all;  kt=input('Enter the value for torque constant kt= ');  r=input('Enter the value for Resistance R= ');  b=input('Enter the value for Friction Torque B= ');  j=input('Enter the value for J= ');  l=input('Enter the value for inductance L= ');  num=[kt];  den=[l\*j (r\*j)+(b\*l) (r\*b) 0];  sys=tf(num,den)  subplot (2,1,1);  impulse(sys);  title('Electromechanical system impulse response');  ylabel('angular displacement');  subplot (2,1,2);  step(sys);  title('Electromechanical system step response');  ylabel('angular displacement'); |

|  |  |  |
| --- | --- | --- |
| 1. **First Order Mechanical System**   clc;  clear all;  close all;  F=input('Enter the value for input force F= ');  M=input('Enter the value for Mass M=');  B=input('Enter the value for Damping coefficient B=');  num=[F];  den=[M B];  sys=tf(num,den)  ltiview(sys); | 1. **First Order Electrical System**   clc;  clear all;  close all;  V=input('Enter the value for input voltage V=');  R=input('Enter the value for resistance R= ');  C=input('Enter the value for capacitance C=');  s=tf('s');  sys=V/((s\*R\*C)+1)  ltiview(sys); | |
| 1. **Second Order Mechanical System**   clc;  clear all;  close all;  F=input('Enter the value for input force F= ');  M=input('Enter the value for Mass M=');  B=input('Enter the value for Damper B=');  K=input('Enter the value for Spring constant K=');  num=[F];  den=[M B K];  sys=tf(num,den)  ltiview(sys); | 1. **Second Order Electrical System**   clc;  clear all;  close all;  V=input('Enter the value for input voltage V=');  R=input('Enter the value for resistance R= ');  C=input('Enter the value for capacitance C=');  L=input('Enter the value for inductance L=');  s=tf('s');  sys=V/(((s^2)\*L\*C)+(s\*R\*C)+1)  subplot(2,1,1);  impulse(sys);  title('Impulse response for RLC circuit');  ylabel('output voltage');  subplot(2,1,2);  step(sys);  title('Step response for RLC circuit');  ylabel('output voltage');  ltiview(sys); | |
| **PID Controller**  clc;  clear all;  close all;  w=logspace(0.1,2,400); %creates logarithmic freq scale  s=tf('s');  sys=100/((s+1)\*(s+2)\*(s+10)); %OLTF with controller  sys\_cl=feedback(sys,1) %CLTF without controller  figure;  bode(sys,w) %bode plot of OLTF without controller  figure;  step(sys\_cl) %step response of CLTF without controller    kp=input("Enter kp value:"); %kp=1.8  ki=input("Enter ki value:"); %ki=2  kd=input("Enter kd value:"); %kd=0.344  cntr=pid(kp,ki,kd); %creates the TF of PID controller  sys1=cntr\*sys; %OLTF with controller  sys1\_cl=feedback(sys1,1) %CLTF with controller  figure;  bode(sys1,w) %bode plot of OLTF with controller  figure;  step(sys1\_cl) %step response of CLTF with controller | |

|  |  |
| --- | --- |
| **Root Locus:**  clc;  clear all;  close all;  num=[1];  den=[1 3 2 0];  sys=tf(num,den) %open loop transfer function  cl\_sys=feedback(sys,1) %closed loop transfer function  step(cl\_sys); %step response of closed loop system  figure;  rlocus(sys); %root locus of open loop transfer function  z=input('Enter the value of zeta:'); %For 20 % overshoot, z=0.456  w=input('Enter the value of omega n:'); %w=1  sgrid(z,w); %plots the line with z and w given  [k,poles]=rlocfind(sys); %returns the value of gain and poles at a selected point on the root locus  new\_clsys=feedback(k\*sys,1) %closed loop transfer function of new system with modified gain  figure  step(new\_clsys); %step response of new closed loop system | **Bode Plot:**  clc;  clear all;  close all;  num=[1];  den=[1 6 5 0];  sys=tf(num,den) %open loop transfer function  w=logspace(-1,2,400); %define freq in log scale  for k=1:40  [mag,phase,w]=bode(k\*sys,w); %returns mag, phase of the open loop system for the given freq  [gm,pm,wgc,wpc]=margin(mag,phase,w); %returns gain margin,phase margin, gain and phase crossover freq  fprintf('Gain:'); disp(k);  fprintf('Gain Cross over frequency:'); disp(wgc);  fprintf('Phase Cross over frequency:'); disp(wpc);  end  (OPTIONAL)  %Bode plot and step response of stable system  k1=input('Enter Gain value for stable system:'); %wgc>wpc, k1=29  figure;  bode(k1\*sys,w);  sys1=feedback(k1\*sys,1);  figure;  step(sys1);  %Bode plot and step response of marginally stable system  k2=input('Enter Gain value for marginally stable system:'); %wgc=wpc, k2=30  figure;  bode(k2\*sys,w);  sys2=feedback(k2\*sys,1);  figure;  step(sys2);  %Bode plot and step response of unstable system  k3=input('Enter Gain value for unstable system:'); %wgc<wpc, k3=31  figure;  bode(k3\*sys,w);  sys3=feedback(k3\*sys,1);  figure;  step(sys3); |

|  |  |
| --- | --- |
| **LEAD COMPENSATOR BODE PLOT:**  clc;  clear all;  close all;  ess=input("Enter required steady state error:"); %ess=1/15  Kv=1/ess %computes velocity error constant  K=input("Enter the value of gain for required steady state error:");%K=15  %in this case K = Kv --> compute manually and enter  e=5; %additional phase angle  s=tf('s');  G=K/(s\*(s+1)) %OLTF of uncompensated system  [GM,PM,wgc,wpc]=margin(G); %returns the freq domain specs  %PM is the phase margin of uncompensated system  PM\_des=input("Enter desired phase margin:"); %PM\_des=45  phi\_req=PM\_des-PM+e %compute required phase margin from lead compensator  alpha=(1-sind(phi\_req))/(1+sind(phi\_req)) %computes alpha of lead compensator  %sind command computes sine value of arguments in degree  mag=-20\*log10(1/sqrt(alpha)) %computes dB magnitude corresponding to 1/sqrt(alpha)  figure;  bode(G); %Bode plot of uncompensated system  %right click on bode plot and select "all stability margins"  %then run the following code step by step  hold on  yline(mag,'r-') %draw line in mag plot corresponding to value of 'mag'  %read the freq corresponding to 'mag' and enter below  wm=input("Enter the freq of mag:"); %wm=5.34  T=1/(wm\*sqrt(alpha)) %computes time constant of lead compensator  zc=1/T %computes zero of lead compensator  pc=1/(alpha\*T) %computes pole of lead compensator  Gc=(s+zc)/(s+pc) %TF of lead compensator  sys=(1/alpha)\*Gc\*G %OLTF of compensated system  figure;  margin(sys) %find the PM of compensated system - will be around 45 degree | **LAG COMPENSATOR BODE PLOT:**  clc;  clear all;  close all;  e=5;  K=input("Enter the gain of OL system:"); %Gain computed from velocity error constant  den=[2 1 0]; %Coefficients of denominator polynomial  G=tf(K,den); %OLTF  figure;  margin(G) %Plots bode graph with GM and PM  PM\_des=input("Enter the required phase margin:");  PM\_req=PM\_des+e; %Additional phase angle  phi\_req=PM\_req-180  wgc\_new=input("Enter the new gain crossover freq:");  [beta,phi]=bode(G,wgc\_new)%returns magnitude and phase at wgc\_new  T=10/wgc\_new  Zc=1/T %Zero of the lag compensator  Pc=1/(beta\*T) % Pole of the lag compensator  Gc=tf([1 Zc],[1 Pc]) %TF of lag compensator  sys=(1/beta)\*Gc\*G % Multiply by (1/beta) to OLTF of compensated system  figure;  margin(sys) %bode plot and GM, PM of compensated system. |
| **LEAD COMPENSATOR ROOT LOCUS:**  clear all  close all  n=input('Enter the Coefficients of Numerator')  d=input('Enter the Coefficients of Denominator') %enter as array [1 11 28 0]  'G1(s)'  G1=tf(n,d)  G2=tf(conv([1 0],n),d); % this is for evaluating the Kv we multiply numerator with s.  M=input('Enter peak overshoot: ')  z=sqrt(log(M)^2 /(pi^2 + log(M)^2)) % calulate damping ratio  wn=input('Enter wn: ')  Kv=input('Enter velocity constant: ')  Sd=complex((-z\*wn),(wn\*sqrt(1-z^2))) % forming the dominant pole  rlocus(G1) % plotting uncompensated root locus  hold on  plot(Sd,'\*') % mark the dominant pole  p = pzmap(G1) % make pole zero plot.  hold on  plot(Sd,'\*') % mark the dominant pole  plot(conj(Sd),'\*')  Po=roots(d)  pang=zeros(1,size(Po,1));  % Finding the angle contributions of poles  for i=1:size(Po,1)  pang(i)=abs(atand((abs(imag(Sd))-abs(imag(Po(i,1))))/(abs(real(Sd))-abs(real(Po(i,1))))));  if(real(Po(i,1))>real(Sd))  pang(i)=180-pang(i);  end  end  Zo=roots(n);  zang=zeros(1,size(Zo,1));  % Finding the angle contributions of zeroes  for i=1:size(Zo,1)  zang(i)=abs(atand((abs(imag(Sd))-abs(imag(Zo(i,1))))/(abs(real(Sd))-abs(real(Zo(i,1))))));  if(real(Zo(i,1))>real(Sd))  zang(i)=180-zang(i);  end  end  thetap=sum(pang);  thetaz=sum(zang);  'Required Phase Lead'  phm=thetap-thetaz-180  phmh=phm;  t=1;  while phmh>60  phmh=phm/(2\*t)  t=t+1;  end  t=2\*(t-1);  ph1=180-atand(abs(imag(Sd))/abs(real(Sd)));  ph1=ph1/2;  c=(real(Sd))-((imag(Sd))/tand(ph1));  ph=ph1-(phmh);  pol=(real(Sd))-((imag(Sd))/tand(ph))  ph=(ph+phmh);  zer=(real(Sd))-((imag(Sd))/tand(ph))  Gc=tf([1 -zer],[1,-pol]) % 'The lead Compensator'  % 'Open Loop Transfer Function K=1'  if phmh==phm  Go=Gc\*G1;  sG=Gc\*G2; % For evaluating error constant  else  Go=(Gc^t)\*G1  sG=(Gc^t)\*G2 % For evaluating error constant  end  sG=minreal(sG); % For evaluating error constant  rlocus(Go)  plot(Sd,'\*')  [K a]=rlocfind(Go)  Kvn=dcgain(K\*sG); % For evaluating error constant  if Kvn>=Kv  'Condition is satisfied';  Kvn  else  'Improve design';  Kvn  end  Go=Go\*K %'The Complete transfer open loop function'  rlocus(Go)  hold on  plot(Sd,'\*') | **LAG COMPENSATOR ROOT LOCUS:**  clc;  clear all;  close all;  K=1; %Assign K=1  den=[1 10 16 0];%coefficients of denominator polynomial  G=tf(K,den) %OLTF  figure;  rlocus(G) %Root locus of OLTF  M=input("Enter Percentage Overshoot:"); %M=0.16  zeta=sqrt(log(M)^2/(pi^2 +log(M)^2)) %compute damping ratio  sgrid(zeta,0) %select the dominant pole at line cos^(-1) of zeta  [K p]=rlocfind(G) %returns gain and poles at the selected point on root locus  Kvu=input("Enter velocity error constant of uncompensated system:"); %Kvu=1.25  ess=input("Enter the value of required steady state error:"); %ess=0.125  Kvd=1/ess %velocity error constant of desired system  A=Kvd/Kvu %compute the improvement factor in velocity error constants  beta=1.2\*A  poles=pole(G) %returns poles of OLTF  poles=sort(poles,'descend') %sort the poles in descending order  T=1/(-0.1\*poles(2)) %takes the values of second pole of the OL system  Zc=1/T %Zero of the lag compensator  Pc=1/(beta\*T) % Pole of the lag compensator  Gc=tf([1 Zc],[1 Pc]) %TF of lag compensator  sys=Gc\*K\*G %OLTF of compensated system  figure;  rlocus(sys) %root locus of compensated system |

|  |  |
| --- | --- |
| **State Space To Transfer Function**  A= input ('Enter the System Matrix A:')  B= input ('Enter the Matrix B:')  C= input ('Enter the Output C:')  D= input ('Enter the Transition D:')  [num,den] = ss2tf(A,B,C,D)  g = tf(num,den)  step(g)  eig(A)  if (eig(A)< 0)  system = 1 %Stable  else  system = 0 %Unstable  end | **Transfer Function to State Space**  num = input ('Enter the Numerator value: ')  den = input ('Enter the Denominator value: ')  disp ('Transfer Function is: ')  t = tf (num,den)  [A,B,C,D] = tf2ss (num,den)  step(A,B,C,D)  eig (A)  if (eig(A) < 0)  system = 1  else  system = 0  end |