CODE.M

```
% [r,p,k] = residue(P1, P2) % Numerator, Poles of denominator and remainder polynomial of partial fractions
% F = tf(P1, P2) % P1(s)/P2(s),
% G = zpk([1 2],[3 4],10)  % 10(s-1)(s-2)/(s-3)(s-4)
% [numerator, denominator] = tfdata(F, 'v');
% Ftf1 = tf(G);%Convert Fzpk1 (s) to coeff  form.
% Fzpk2 = zpk(F); % Convert Ftf2 (s) to factored form
% [numzeroes, denpoles] = tf2zp(P1, P2) %zeroes and poles of TF
% [numgtf, dengtf] = zp2tf([1 2]',[3 4],10)%TFfromZPK
% s=tf ('s'); % Define 's' as an LTI object in polynomial
% F=150*(s^2+2*s+7)/[s*(s^2+5*s+4)]; % Form F (s)
% s=zpk('s'); % Define 's' as an LTI object in factored
% F=150*(s^2+2*s+7)/[s*(s^2+5*s+4)]; % Form F (s)
% plot(t,f1,'r',t,f2,'g')
% F=ss(A,B,C,D) % Create an LTI object and display
% [A,B,C,D]=tf2ss(P1, P2)
% P=[0 0 1;0 1 0;1 0 0];
% Ap=inv(P)*A*P
% Bp=inv(P)*B
% Cp=C*P
% Dp=D
% Tss = ss(F)
% F = tf(Tss)
% [num,den]=ss2tf(A,B,C,D,1)
% G = zpk(Tss)
% f=ilaplace(3/(s*(s^2+2*s+5)))
% f=2*exp(-t)-2*t*exp(-2*t)-2*exp(-2*t);
% F=laplace(f);
%  F=factor(F); % Factorize
%G=54*(s+27)*(s^3+52*s^2+37*s+73)/(s*(s^4+872*s^3+437*s^2+89*s+65)*(s^2+79*s+36));
% [numg,deng]=numden(A); % Extract symbolic numerator and denominator.
% numg=sym2poly(numg) % Form vector for numerator of A(s).
% deng=sym2poly(deng) % Form vector for denominator of A(s).
```

Ss_to_tf_symbolic

```
T=C*((s*I-A)^-1)* B+D; % Find transfer function.
```

A_second_order_characteristics

```
omegan=sqrt (deng (3) /deng (1)) % Calculate the natural frequency,sqrt(c/a).
zeta=(deng(2)/deng(1))/(2*omegan)
Ts=4/(zeta*omegan) % Calculate settling time,
x=1-(zeta^2);
Tp=pi/(omegan*sqrt(x)) % Calculate peak time, pi/wn*sqrt(l -z^2).
pos=100*exp( -zeta*pi/sqrt(1-zeta^2)) % Calculate percent overshoot (100*e^(-z*pi/sqrt(l-z^2)).
```

B_plotting_step_responses

```
clf % Clear graph.
T1=tf(numt1,dent1) % Create and display T1(s).
step(T1) % Run a demonstration step response and plot
pause
T=ss(A,B,C,D) % Generate LTI object, T, in state space and display.
step(T,t) % Plot step response for given
```

c_laplace_ABCD_solution_and_Eigenvalues_vectors

```
X0=[2;1];
U=1/(s+1);
X=((s*I-A)^-1)*(X0+B*U)
x1=ilaplace(X(1))
Y=C*X+D*U
Y=simplify(Y)
y=ilaplace(Y)
```

d_STM_from_ABCD

```
X0=[1;0]; % Create initial condition vector, X(0).
U=1/s; % Create U(s).
E=((s*I-A)^-1) % Find Laplace transform of state-transition matrix,(sI-A)^-1.
Fi=ilaplace(E)
Fitmtau=subs(Fi,t,t-tau); % Form Fi(t-tau).
u = ilaplace(U);
utau=subs(u,t,tau);
X=Fi*X0+int(Fitmtau*B*utau,tau,0,t); % Solve for X(t).
X=expand(X); % Expand X for clearer display.
```

F_diagonalization

```
[P,d]=eig(A) % Generate transformation matrix, P, and eigenvalues, d.
Adt=inv(P)*A*P % Calculate diagonal system A.
Bdt=inv(P)*B % Calculate diagonal system B.
Cdt=C*P % Calculate diagonal system C.
S=ss (A,B,C,0) % Create state-space LTI object.
Sp=canon(S, 'modal') % Sp is an ss object with diagonal A
```

H_block_reduction

```
G1=tf(1,[1 1]); G2=G1;G3=G1;
H1=tf(1,[1 0]); H2=H1;H3=H1;
System=append(G1,G2,G3,H1,H2,H3);
input=1;output=3;
Q= [1 -4 0 0 0;
2 1 -5 0 0];;
T=connect(System,Q,input,output);
T=tf(T)
T=minreal(T)
```