APPENDIX 1

PROGRAM FOR ROUTH ARRAY FORMULATION

%Program to formulate Routh Array
function ROU=routharray(poly);
%Enter the coefficients of the considered polynomial
poly=input('Enter the given polynomial')
%Get the size of the polynomial considered
dim=size(poly);
%Obtain the number of coefficients of the given polynomial
coeffi=dim(2);
%perform symbolic array initialization of Routh array
ROU=sym(zeros(coeffi,ceil(coeffi/2)));
for m=1:coeffi,
    ROU(2-rem(m,2),ceil(m/2))=poly(m);
end
%number of rows that need determinants
rows=coeffi-2;
%initialize columns-per-row index vector
index=zeros(rows,1);
for m=1:rows,
    index(rows-m+1)=ceil(m/2);
end
%Compute the Routh Array Elements using Routh Multiplication Rule
for m=3:coeffi,
    for n=1:index(m-2),
        ROU(m,n)=-det([ROU(m-2,1) ROU(m-2,n+1);ROU(m-1,1) ROU(m-1,n+1)])/ROU(m-1,1);
    end
end
APPENDIX 2

PROGRAM TO COMPUTE OUTPUT FROM BACK PROPAGATION NETWORK

%Program to calculate the output of the Back Propagation Network
%Activation function used is Bipolar Activation function
%In the activation function, a slope parameter q is used apart from normal activation function

% Step 1 - Initialize the weights
% Step 2 - Give the inputs x(k) and x(k-1)
% Step 3 - Calculate the net input to the hidden neurons
% Step 4 - Calculate the output from hidden neurons using the activation function
f={\[2/1+e^{(-v/q)}\]-1}
% Step 5 - Using the output from hidden neurons, calculate net input into the output unit
% Step 6 - Calculate the output x(k+1)
% Step 7 - Once output is calculated, first iteration is completed, then assign x(k)=x(k+1) and x(k-1)=x(k), now repeat steps from 2 to 6.
% Step 8 - Continue the process for a specified number of iterations
% Step 9 - Plot (x(k),k), where x(k)-output values for kth iteration, k-iteration number

% Illustration 4.1 (Tanaka, 1996)
% Step 1 - Initializing the weights
w111=1.0;
w121=-0.5;
w112=-1.0;
w122=-0.5;
w211=1.0;
w212=1.0;
q=0.75;
lambda = 2;

%Step 2 - Input the given initial conditions
x1=-0.5;  \( %x(k) \)
x2=-1.48;  \( %x(k-1) \)

%Step 3 - Calculate the net input to the hidden layer neurons
v11=w111*x1+w121*x2
v12=w112*x1+w122*x2

%Step 4 - Calculate the output from the hidden layer neurons
f11=(2/(1+exp(-v11/q)))-1
f12=(2/(1+exp(-v12/q)))-1

%Step 5 - Compute the net input entering the output layer neuron
v21=w211*f11+w212*f12

%Step 6 - Obtain the output from the output layer neuron using the
% activation function
f21=(2/(1+exp(-v21/q)))-1

%Graphical plot of the output values obtained
k=0:35
x=[-0.5 0.5067 0.1949 -0.2160 -0.0845 0.0952 0.0374 -0.0422 -0.0166
    0.0187 0.0074 -0.0083 -0.0033 0.0037 -0.0015 -0.0016 0.00067
    0.00071 -0.0003 -0.0003 0.0001 0.0001 -0.00004 -0.00004
    -0.000001 0.000001 0.000004 -0.000004 -0.000001 0.000001
    0.0000004 -0.0000004 -0.0000001 0.0000001 4.44e-08 -4.44e-08]
stem(k,x)  \( % \) the plot obtained is as shown in Figure 4.4.
APPENDIX 3

PROGRAM TO COMPUTE OUTPUT RESPONSE OF A FUZZY SYSTEM

% Program to compute the output response of a fuzzy system.
% Illustration 5.1 (Kawamoto et al 1992)
% Enter the elements of system matrix A1
a11 = 1.503;
a12 = -0.588;
a21 = 1.0;
a22 = 0.0;
% Enter the elements of system matrix A2
b11 = 1.0;
b12 = -0.361;
b21 = 1.0;
b22 = 0.0;
% Input the initial conditions of the fuzzy system considered
x1 = 0.9;
x2 = -0.7;
% Computing the output response for 20 instants of time index
for iter = 1:20
  % Compute the membership values of m1 and m2 depending upon x2.
  m1 = -0.5*x2+0.5;
  if(m1 >= 1.0)
    m1 = 1.0;
  elseif(m1 < 0.0)
    m1 = 0.0;
  end
end
end
m2 = 0.5*x2+0.5;
if(m2 >= 1.0)
    m2 = 1.0;
elseif(m2 < 0.0)
    m2 = 0.0;
end

% Using the obtained membership values, elements of System matrices and 
% the given initial conditions computing the output at (K+1)th instant
xt1 = (a11*x1+a12*x2)*m1 + (b11*x1+b12*x2)*m2;
xt2 = (a21*x1+a22*x2)*m1 + (b21*x1+b22*x2)*m2;
x1 = xt1;
x2 = xt2;
disp(x1)
end

% Plot of the output response
k=0:20
xk=[0.9000 1.6726 1.3795 0.7757 0.2777 -0.0064 -0.1306 -0.1606 -0.1424 -
    0.1048 -0.0650 -0.0321 -0.0094 0.0035 0.0089 0.0094 0.0076 0.0050 0.0027
    0.0010 -0.0001]
stem(k,xk) % The plot of this is as shown in Figure 5.3
APPENDIX 4

PROGRAM FOR MAX-MIN COMPOSITION OF FUZZY SYSTEMS

% Program to perform Max-Min Composition of Fuzzy Systems given their Fuzzy Relational Matrices
% enter the two fuzzy relations for which composition is to be performed
R=input('enter the first fuzzy relational matrix')
S=input('enter the second fuzzy relational matrix')
% find the size of two matrices
[m,n]=size(R);
[a,b]=size(S);
if(n==a)
    for i=1:m
        for j=1:b
            c=R(i,:);
            d=S(:,j);
            f=d';
            e=min(c,f);
            h(i,j)=max(e);
        end
    end
else
    display('The fuzzy relation cannot be find')
end
% Output the result
display('the fuzzy compositional matrix between the two given fuzzy relational matrices is');
display(h)
APPENDIX 5

PROGRAMS FOR PROPOSED ALGORITHMS OF FUZZY SYSTEMS

% Programs for proposed algorithms of fuzzy systems using the
% compositional matrices.
% Proposed Algorithm 1 using Step responses of compositional matrices
% Illustration 5.3 (Kiszka et al 1985)
clear all;
clc;
% Input the Fuzzy relation matrix P representing the given Fuzzy system
P=[0.1 0.5;0.3 0.2]
% Step 1 - Scale the given Fuzzy relation matrix
PS=P/4
% Step 2 - Obtain the characteristic polynomial of PS
P1=poly(PS)
num=1;
den=P1;
% Step 3 - Formulate an open loop all pole fuzzy system
G=tf(num,den,0.1)
% Step 4 - Obtain the step response of the system given by G
[out,state]=dstep(num,den)
stem(out) % Observe the peak amplitude value of the step response of the
% fuzzy system

% Proposed Algorithm 2 using necessary conditions of compositional
% matrices
% Illustration 5.5 (Kiszka et al 1985)
clear all;
clc;

%Input the Fuzzy relation matrix P representing the given Fuzzy system
P=[0.1 0.5;0.3 0.2]

%Step 1 - Scale the given Fuzzy relation matrix
PS=P/4

%Step 2 - Obtain the characteristic polynomial Z of PS
Z=poly(PS)

%Step 3 - Evaluate the characteristic polynomial Z at three points
%Evaluating at Z=0
fzero=polyval(Z,0)

%Evaluating at Z=1
fone=polyval(Z,1)

%Evaluating at Z=-1
fminusone=polyval(Z,-1)

%Proposed Algorithm 3 using Trace and Determinant of compositional matrices

%Illustration 5.7 (Kiszka et al 1985)
clear all;
clc;

%Step 1 - Input the Fuzzy relation matrix P representing the given Fuzzy system
P=[0.1 0.5;0.3 0.2]

%Step 2 - Compute the trace of the given relational matrix P
T = trace(P)

%Step 3 - Compute the Determinant of the given relational matrix P
D = det(P)

% In all the above proposed algorithms the various steps involved are performed for all the compositional matrices obtained from the given fuzzy relational matrix using max-min composition approach
APPENDIX 6

PROGRAM OF MAMDANI FUZZY INFERENCE SYSTEM DEVELOPED FOR INVERTED PENDULUM MODEL

[System]
Name='rulependulum'
Type='mamdani'
Version=2.0
NumInputs=2
NumOutputs=1
NumRules=49
AndMethod='min'
OrMethod='max'
ImpMethod='min'
AggMethod='max'
DefuzzMethod='centroid'

[Input1]
Name='error'
Range=[-1.5 1.5]
NumMFs=7
MF1='NL':'trapmf',[-2 -1.5 -1 -0.8]
MF2='NM':'trimf',[-0.9 -0.7 -0.4]
MF3='NS':'trimf',[-0.5 -0.3 -0.1]
MF4='ZE':'trimf',[-0.2 0 0.2]
MF5='PS':'trimf',[0.1 0.3 0.5]
MF6='PM':'trimf',[0.4 0.7 0.9]
MF7='PL':'trapmf',[0.8 1 1.5 2]
[Input2]
Name='errorderivative'
Range=[-1.5 1.5]
NumMFs=7
MF1='NL':'trapmf',[-2 -1.5 -1 -0.8]
MF2='NM':'trimf',[-0.9 -0.7 -0.4]
MF3='NS':'trimf',[-0.5 -0.3 -0.1]
MF4='ZE':'trimf',[-0.2 0 0.2]
MF5='PS':'trimf',[0.1 0.3 0.5]
MF6='PM':'trimf',[0.4 0.7 0.9]
MF7='PL':'trapmf',[0.8 1 1.5 2]

[Output1]
Name='output'
Range=[-1.5 1.5]
NumMFs=7
MF1='NL':'trapmf',[-2 -1.5 -1 -0.8]
MF2='NM':'trimf',[-0.9 -0.7 -0.4]
MF3='NS':'trimf',[-0.5 -0.3 -0.1]
MF4='ZE':'trimf',[-0.2 0 0.2]
MF5='PS':'trimf',[0.1 0.3 0.5]
MF6='PM':'trimf',[0.4 0.7 0.9]
MF7='PL':'trapmf',[0.8 1 1.5 2]

[Rules]
1 1, 1
1 2, 1
1 3, 1
1 4, 2
1 5, 3
16, 3
17, 4
21, 1
22, 2
23, 2
24, 2
25, 3
26, 4
27, 4
31, 2
32, 2
33, 3
34, 3
35, 4
36, 4
37, 5
41, 3
42, 3
43, 4
44, 4
45, 4
46, 5
47, 5
51, 3
52, 4
53, 4
54, 5
55, 5
56, 6
57, 6
a) Program for lower order model formulation of SISO LTICS

Program to obtain lower order model for the given higher order model using Auxilary Polynomial approach and Particle Swarm optimization for SISO LTICS

Illustration 6.1 (Krishnamurthy et al 1978)

clc;
clear all;

Enter the numerator and denominator coefficients of given higher order system

num=[35 1086 13285 82402 278376 511812 482964 194480];
den=[1 33 437 3017 11870 27470 37492 28880 9600];
t=[0:0.1:20];

a=tf(num,den);
range1=2;
range2=0.5;
generations=90;
particles=80;
c1=0.5;
c2=0.5;

The s-term and constant term in the denominator of this lower order model are to be tuned using PSO process.
num1=[35 5.1887];
den1=[1 0.7703 0.2561];
ratio=20.26; %Steady state gain of the system
gbestcost=0;

%Set the range of the parameters to be tuned using PSO approach
l_range1=-range1+den1(2);
h_range1=range1+den1(2);
l_range2=-range2+den1(3);
h_range2=range2+den1(3);
mincost=0;
index=0;
b=tf(num1,den1);
[y1,t]=step(a,t);

%Initialize the particle velocities and position
for i=1:particles,
    vel1(i)=0;
    vel2(i)=0;
    pos1(i)=0;
    pos2(i)=0;
    cost(i)=0;
    pbest1(i)=0;
    pbest2(i)=0;
    bestcost(i)=0;
end

%For each particle generated obtain the step response and compute the
%integral square error
for i=1:particles,
    den1(3)=rand*(h_range2-l_range2)+l_range2;
    den1(2)=rand*(h_range1-l_range1)+l_range1;
    pos1(i)=den1(2);
pos2(i)=den1(3);
b=tf(num1,d1);
[y2,t]=step(b,t);
pbest1(i)=den1(2);
pbest2(i)=den1(3);
num1(2)=ratio*den1(3);
bestcost(i)=sum((y1-y2).*(y1-y2));
end

[mincost,gbest_pos]=min(bestcost)
gbestcost=mincost;

%Obtain the particle personal best i.e., pbest
gbest1=pbest1(gbest_pos);
gbest2=pbest2(gbest_pos);

%Perform velocity updation and position updation
%Recalculate the objective function - Integral Square Error so that the
%minimum ISE so far in the generation is observed to get gbest

for i=1:generations,
    for j=1:particles,
        p1=rand;
p2=rand;
        vel1(j)=vel1(j)+c1*p1*(pbest1(j)-pos1(j)) + c2*p2*(gbest1-pos1(j));
p1=rand;
p2=rand;
        vel2(j)=vel2(j)+c1*p1*(pbest2(j)-pos2(j)) + c2*p2*(gbest2-pos2(j));
pos1(j)=pos1(j)+vel1(j);
pos2(j)=pos2(j)+vel2(j);
den1(2)=pos1(j);
den1(3)=pos2(j);
num1(2)=den1(3)*ratio;
b=tf(num1,d1);
[y2,t]=step(b,t);
cost(j)=sum((y1-y2).*(y1-y2));
if cost(j) < bestcost(j)
    bestcost(j)=cost(j);
pbest1(j)=den1(2);
pbest2(j)=den1(3);
end
end
[mincost,best]=min(cost);
i
mincost
cost
sval=sum(cost)/particles
if mincost < gbestcost
    gbestcost=mincost;
    gbest1=pos1(best);
    gbest2=pos2(best);
end
gbestcost
gbest1
gbest2
end
gbestcost
gbest1
gbest2
den1(2)=gbest1;
den1(3)=gbest2;
um1(2)=gbest2*ratio;
b=tf(num1,den1);
[y2,t]=step(b,t)
per_index=sum((y1-y2).*(y1-y2))

% Plot the response for the minimum gbest value so that the characteristics of
% the obtained lower order model closely matches with the given higher order
% model
plot(per_index, generations)
num1
den1 % Denominator coefficients tuned using PSO with the minimum gbest
% value
step(a,'r',b,'b--',t)

b) Program for lower order model formulation of MIMO LTICS

% Program to obtain lower order model for the given higher order model using
% Auxiliary Polynomial approach and Particle Swarm Optimization for
% MIMO LTICS
% Illustration 6.3 (Prasad et al 1995)
clc;
clear all;

% Enter the given Transfer Function Matrix of the given MIMO system
g11=tf([2 10],[conv([1 1],[1 10])])
g12=tf([1 4],[conv([1 2],[1 5])])
g21=tf([1 10],[conv([1 1],[1 20])])
g22=tf([1 6],[conv([1 2],[1 3])])

% Obtain the common denominator of the given sixth order system
D=conv([1 20],[conv([1 10],[conv([1 5],[conv([1 3],[conv([1 1],[1 2])])])])])

% Now the MIMO system is split up into individual SISO systems with
% common denominator and are displayed as follows
G11=tf(conv([1 20],[conv([1 3],[conv([1 5],[conv([2 10],[1 2])])]))),D)
G12=tf([1 20],[conv([1 3],[conv([1 10],[conv([1 4],[1 1])])]))),D)
G21=tf(conv([1 3],[conv([1 5],[conv([1 2],[conv([1 10],[1 10])])]))),D)
G22=tf(conv([1 20],[conv([1 5],[conv([1 10],[conv([1 6],[1 1])])]))),D)
For each $G_{11}, G_{12}, G_{21}$ and $G_{22}$ apply the program of lower order model formulation for SISO LTICS in Appendix 7(a) and obtain the respective lower order models with minimal ISE which are as given below.

$$r_{11} = \text{tf}([2 \ 10.0648],[1 \ 11.0537 \ 10.0648])$$

$$r_{12} = \text{tf}([1 \ 2.5651],[1 \ 5.3858 \ 6.4128])$$

$$r_{21} = \text{tf}([1 \ 9.8146],[1 \ 20.6362 \ 19.6293])$$

$$r_{22} = \text{tf}([1 \ 5.9849],[1 \ 4.9928 \ 5.9849])$$

Commonize the denominators of the lower order models by taking average of the corresponding denominator coefficients in $r_{11}, r_{12}, r_{21}$ and $r_{22}$.

$$d = \{((1+1+1+1)/4),((11.0537+5.3858+20.6362+4.9928)/4),((10.0648+6.4128+19.6293+5.9849)/4)\}$$

Using the transient and steady state gains of given higher order system and reconstruct the final second order models

$$R_{11} = \text{tf}([2 \ 10.52295.*1],[1 \ d])$$

$$R_{12} = \text{tf}([1 \ 10.52295.*0.4],[1 \ d])$$

$$R_{21} = \text{tf}([1 \ 10.52295.*0.5],[1 \ d])$$

$$R_{22} = \text{tf}([1 \ 10.52295.*1],[1 \ d])$$
APPENDIX 8

PROGRAM FOR LOWER ORDER MODEL
FORMULATION OF LTIDS

a) Program for lower order model formulation of SISO LTIDS

%Program to obtain lower order model for the given higher order model using
%Auxiliary Polynomial approach and Particle Swarm optimization for SISO
%LTIDS
%Illustration 6.4 (Prasad 1993)
clc;
clear all;

%Enter the higher order polynomial
num=[1.682 1.116 -0.21 0.152 -0.516 -0.262 0.044 -0.018]
den=[8 -5.046 -3.348 0.63 -0.456 1.548 0.786 -0.132 0.018]
in=ones(1,61)
ts=0.1 %sampling time
t=0:1:60 %time instants for the plot which occurs in x-axis
a=tf(num,den,ts) % transfer function of higher order system

%obtaining the altitude values (y-axis values for particular time instants for
%higher order systems)
y1=dstep(num,den,t)
outh=filter(num,den,in);
num1=[0.21025 -0.4091]
den1=[1 -0.1679 0.0229]
y2=dstep(num1,den1,t)

%To obtain the value of 'a' at z=1
n=polyval(num,1);
d=polyval(den,1);
ho_z=n/d \% gives the value of higher order system at \( z=1 \)
\% range for the 3 parameters
range_high=2;
range_low=0;
\% initializing no of generations & particles
generations=150;
particles=1000;
mincost=0;
gbestcost=0;
c1=1;
c2=1;
diff=1;
\% initialize gbest
for i=1:4,
gbest(i)=0;
end
\% initializing the fitness, cost, bestcost variables for all the particles
for i=1:particles,
    fitness(i)=0;
    cost(i)=0;
    bestcost(i)=0;
end
\% initializing the position, velocity, pbest variables for all the particles
for i=1:particles,
    for j=1:3,
        pos(i,j)=rand*(range_high-range_low)+range_low;
        vel(i,j)=0;
        pbest(i,j)=pos(i,j);
    end
end
% calculating fitness of each particle
num1(2)=pos(i,1);
den1(2)=pos(i,2);
den1(3)=pos(i,3);
y2=dstep(num1,den1,t);
per_index=sum((y1-y2).*(y1-y2)) ;

nl=polyval(num1,1);
dl=polyval(den1,1);
lo_z=nl/dl %gives the value of higher order system at z=1

if ho_z < lo_z
    diff=lo_z-ho_z;
else
    diff=-lo_z+ho_z;
end

bestcost(i)=per_index+diff;
end

[mincost,gbest_pos]=min(bestcost)
gbestcost=mincost;

% update gbest
for i=1:3,
    gbest(i)=pbest(gbest_pos,i);
end

for i=1:generations,
    for j=1:particles,
        p1=rand;
p2=rand;
        for k=1:3,
            vel(j,k)=vel(j,k)+c1*p1*(pbest(j,k)-pos(j,k))+c2*p2*(gbest(k)-pos(j,k));
pos(j,k)=pos(j,k)+vel(j,k);
end
end
% compute fitness of each particle in current generation
num1(2)=pos(j,1);
den1(2)=pos(j,2);
den1(3)=pos(j,3);
% b=tf(num1,den1,ts) % transfer function of lower order system
y2=dstep(num1,den1,t);
% Calculation of performance index J
per_index=sum((y1-y2).*(y1-y2));
nl=polyval(num1,1);
dl=polyval(den1,1);
lo_z=nl/dl;% gives the value of lower order system at z=1
if ho_z < lo_z
    diff=lo_z-ho_z;
else
    diff=-lo_z+ho_z;
end
cost(j)=per_index+diff;
if cost(j) < bestcost(j)
    bestcost(j)=cost(j);
    for k=1:3,
        pbest(j,k)=pos(j,k); % Obtain the particle personal best i.e., pbest
    end
end
end
[mincost,best_pos]=min(cost);
i mincost
% cost
% pos
sval=sum(cost)/particles
if mincost < gbestcost
    gbestcost=mincost;
    for k=1:3,
        gbest(k)=pos(best_pos,k); %Obtain the particle global best i.e., gbest
    end
end
gbestcost
end
num1(2)=gbest(1);
den1(2)=gbest(2);
den1(3)=gbest(3);
b=tf(num1,den1,ts) % transfer function of lower order system corresponding
    % to minimum fitness value
y2=dstep(num1,den1,t);
per_index=sum((y1-y2).*(y1-y2))
outh=filter(num,den,in);
stem(t,outh,'b') %response for higher order system
hold on;
outl=filter(num1,den1,in);
stem(t,outl,'r') %response for lower order system corresponding to minimum
    %fitness value
nl=polyval(num1,1);
dl=polyval(den1,1);
lo_z=nl/dl %gives the value of higher order system at z=1
ho_z

b) Program for lower order model formulation of MIMO LTIDS
%Program to obtain lower order model for the given higher order model using
%Auxiliary Polynomial approach and Particle Swarm optimization for MIMO
%LTIDS
% Illustration 6.6 (Bistritz and Shaked 1984)
clc;
clear all;
% Enter the given Transfer Function Matrix of the given MIMO LTIDS system
g11=tf([2.25 -1.6875],[conv([1 -0.95],[1 -0.5])],0.1)
g12=tf([1.5 -1.2],[conv([1 -0.9],[1 -0.75])],0.1)
g21=tf([1.04 -0.676],[conv([1 -0.95],[1 -0.3])],0.1)
g22=tf([1 -0.7],[conv([1 -0.9],[1 -0.35])],0.1)
% Obtain the common denominator of the given sixth order system
D=conv([1 -0.35], conv([1 -0.3], conv([1 -0.75], conv([1 -0.9],conv([1 -0.95],[1 -0.5])))))
% Now the MIMO system is split up into individual SISO systems with
% common denominator and are displayed as follows
G11=tf(conv([1 -0.9],conv([1 -0.35],conv([1 -0.3],conv([2.25 -1.6875],[1 -0.75])))),D,0.1)
G12=tf(conv([1 -0.35],conv([1 -0.3],conv([1 -0.5],conv([1.5 -1.2],[1 -0.95])))),D,0.1)
G21=tf(conv([1 -0.35],conv([1 -0.75],conv([1 -0.9],conv([1.04 -0.676],[1 -0.5])))),D,0.1)
G22=tf(conv([1 -0.3],conv([1 -0.75],conv([1 -0.5],conv([1 -0.7],[1 -0.95])))),D,0.1)
% For each G11, G12, G21 and G22 apply the program of lower order
% model formulation for SISO LTIDS in Appendix 8(a) and obtain the
% respective lower order models with minimal ISE which are as given below.
  r11=tf([2.25 0.1738],[1 -0.06023 -0.8288])
  r12=tf([1.5 1.133],[1 -0.02448 -0.7549])
  r21=tf([1.04 0.1643],[1 -0.0488 -0.9267])
  r22=tf([1 -0.3318],[1 -0.6534 -0.2015])
% Commonize the denominators of the lower order models by taking average
% of the corresponding denominator coefficients in r11, r12, r21 and r22.
\[ \begin{align*}
    d & = \left( \frac{1}{4} + \frac{1}{4} + \frac{1}{4} + \frac{1}{4} \right) - \left( \frac{0.06023 + 0.02448 + 0.0488 + 0.6534}{4} \right) - \\
    & \left( \frac{0.8288 + 0.7549 + 0.9267 + 0.2015}{4} \right) \\
\end{align*} \]

With Commonized denominators the lower order MIMO LTIDS models are as given below:

\[ \begin{align*}
    R_{11} & = \text{tf}([2.25 \ 0.1738],d,0.1) \\
    R_{12} & = \text{tf}([1.5 \ 1.133],d,0.1) \\
    R_{21} & = \text{tf}([1.04 \ 0.1643],d,0.1) \\
    R_{22} & = \text{tf}([1 \ -0.3318],d,0.1) \\
\end{align*} \]

Further, the above \( R_{11}, R_{12}, R_{21} \) and \( R_{22} \) are reconstructed using the transient and steady state gains of the given higher order system and the final MIMO second order approximant is obtained as given in equation \( (6.91) \) of Chapter 6
APPENDIX 9

PROGRAM FOR LOWER ORDER MODEL
FORMULATION OF LTIDS WITH TRANSFORMATION

a) Program for lower order model formulation of SISO LTIDS with transformation

%Program to obtain lower order model for the given higher order model using
%Auxiliary Polynomial approach and Particle Swarm Optimization for SISO
%LTIDS with transformation
%Illustration 6.7 (Prasad 1993)
clc;
clear all;
%Enter the higher order polynomial
num=[1.682 1.116 -0.21 0.152 -0.516 -0.262 0.044 -0.018]
den=[8 -5.046 -3.348 0.63 -0.456 1.548 0.786 -0.132 0.018]
gd=tf(num,den,0.1) %Transfer function of given higher order LTIDS
in=ones(1,81)
ts=0.1 %sampling time
t=0:1:80 %time instants for the plot which occurs in x-axis
a=tf(num,den,ts) % transfer function of higher order system
y1=dstep(num,den,t) %obtaining the altitude values i.e., y-axis values for
%particular time instants for higher order systems)
outh=filter(num,den,in);
stem(0:1:80,outh,'b') %response for higher order system
hold on;
%Transform the given higher order polynomial in LTIDS to continuous
%domain using linear transformation z=p+1
a=[1 1]; % the coefficients of z which is to be transformed
b=conv(a,a); % Evaluation of (p+1)^2
c=conv(b,a); % Evaluation of (p+1)^3
d=conv(c,a); % Evaluation of (p+1)^4
e=conv(d,a); % Evaluation of (p+1)^5
f=conv(e,a); % Evaluation of (p+1)^6
g=conv(f,a); % Evaluation of (p+1)^7
h=conv(g,a); % Evaluation of (p+1)^8
% To obtain the numerator coefficients in continuous domain multiply the
evaluated polynomials a,b,c,d,e,f,g with the given numerator coefficients of
given higher order LTIDS.
N0=[0 0 0 0 0 0 -0.018];
N1=conv(0.044,[0 0 0 0 0 a]);
N2=conv(-0.262,[0 0 0 0 b]);
N3=conv(-0.516,[0 0 0 c]);
N4=conv(0.152,[0 0 d]);
N5=conv(-0.21,[0 0 e]);
N6=conv(1.116,[0 f]);
N7=conv(1.682,[g]);
NUMC=N7+N6+N5+N4+N3+N2+N1+N0
% To obtain the denominator coefficients in continuous domain multiply the
evaluated polynomials a,b,c,d,e,f,g,h with the given denominator
coefficients of given higher order LTIDS.
D0=[0 0 0 0 0 0 0 0.018];
D1=conv(-0.132,[0 0 0 0 0 0 a]);
D2=conv(0.786,[0 0 0 0 0 b]);
D3=conv(1.548,[0 0 0 0 c]);
D4=conv(-0.456,[0 0 0 d]);
D5=conv(0.63,[0 0 e]);
D6=conv(-3.348,[0 0 f]);
D7=conv(-5.046,[0 g]);
D8=conv(8,[h]);
DENC=D8+D7+D6+D5+D4+D3+D2+D1+D0

Thus the transfer function in p-domain (continuous domain) from z-domain
%(discrete domain) using linear transformation of given higher order system
%is:
GC=tf(NUMC,DENC)

For the transformed continuous system in p-domain apply the program for
%lower order model formulation of SISO LTICS in Appendix 7(a) to
%obtain the second order model as given below
NUMLO=[0.2103 0.07876]
DENLO=[1 0.2286 0.0792]
LC=tf(NUMLO,DENLO)

To obtain the lower order transfer function in z-domain now apply reverse
%linear transformation p=z-1
x=[1 -1];
y=conv(x,x);
n0=[0 0.07876];
n1=conv(0.2103,x);
nd=n1+n0
d0=[0 0 0.0792];
d1=conv(0.2286,[0 x]);
d2=conv(1,[y]);
dd=d2+d1+d0

Thus the transfer function in z-domain (discrete domain) from p-domain
%(continuous domain) using reverse linear transformation of lower order
%model is:
LD=tf(nd,dd,0.1)
y2=dstep(nd,dd,t)
outl=filter(nd,dd,in);
stem(0:1:80,outl,'r')

%response for lower order system

%Calculation of performance index i.e., integral square error
per_index=sum((y1-y2).*(y1-y2))

b) Program for lower order model formulation of MIMO LTIDS with transformation

%Program to obtain lower order model for the given higher order model using
%Auxiliary Polynomial approach and Particle Swarm optimization for MIMO
%LTIDS with transformation
%Illustration 6.9 (Bistritz and Shaked 1984)
clc;
clear all;
%Enter the given Transfer Function Matrix of the given MIMO LTIDS
%system

g11=tf([2.25 -1.6875],[conv([1 -0.95],[1 -0.5]),0.1])
g12=tf([1.5 -1.2],[conv([1 -0.9],[1 -0.75]),0.1])
g21=tf([1.04 -0.676],[conv([1 -0.95],[1 -0.3]),0.1])
g22=tf([1 -0.7],[conv([1 -0.9],[1 -0.35]),0.1])

%Obtain the common denominator of the given sixth order system
D=conv([1 -0.35], conv([1 -0.3], conv([1 -0.75], conv([1 -0.9],conv([1 -0.95],[1 -0.5])))))

%Now the MIMO system is split up into individual SISO systems with
%common denominator and are displayed as follows
G11=tf(conv([1 -0.9],conv([1 -0.35],conv([1 -0.3],conv([2.25 -1.6875],[1 -0.75])))),D,0.1)
G12=tf(conv([1 -0.35],conv([1 -0.3],conv([1 -0.5],conv([1.5 -1.2],[1 -0.95])))),D,0.1)
G21=tf(conv([1 -0.35],conv([1 -0.75],conv([1 -0.9],conv([1.04 -0.676],[1 -0.5])))),D,0.1)
G22=tf(conv([1 -0.3],conv([1 -0.75],conv([1 -0.5],conv([1 -0.7],[1 -0.95])))),D,0.1)

%Apply the linear transformation z=p+1 to G11, G12, G21 and G22 as in the
% program of lower order model formulation of SISO LTIDS with
% transformation in Appendix 9(a) to obtain individual transfer functions
% in continuous domain as given below:
DenC=[1 2.25 1.9125 0.759375 0.1413375 0.01108125 0.000284375]
G11C=tf([2.25 4.3875 3.099375 0.97003125 0.13415625 0.0063984375],DenC)
G12C=tf([1.5 3.15 2.40375 0.79275 0.1022625 0.0034125],DenC)
G21C=tf([1.04 1.924 1.3286 0.42211 0.06032 0.0029575],DenC)
G22C=tf([1 1.8 1.1725 0.33675 0.040375 0.0013125],DenC)

%For the transformed MIMO continuous system in p-domain apply the
% program for lower order model formulation of MIMO LTICS in Appendix
% 7(b) to obtain the second order model as given below
L11C=tf([2.25 0.9473],[1 0.6614 0.0421])
L12C=tf([1.5 0.5052],[1 0.6614 0.0421])
L21C=tf([1.04 0.4378],[1 0.6614 0.0421])
L22C=tf([1 0.1943],[1 0.6614 0.0421])

%To obtain the lower order transfer function of MIMO system in z-domain
% now apply reverse linear transformation p=z-1 given in the program of lower
% order model formulation of SISO LTIDS with transformation as in
% Appendix 9(a). Thus the lower order MIMO models in z-domain are as
% given below
L11D=tf([2.25 -1.3027],[1 -1.3386 0.3807],0.1)
L12D=tf([1.5 -0.9948],[1 -1.3386 0.3807],0.1)
L21D=tf([1.04 -0.6022],[1 -1.3386 0.3807],0.1)
L22D=tf([1 -0.8057],[1 -1.3386 0.3807],0.1)
APPENDIX 10

PROGRAM FOR COMPARISON OF VARIOUS METHODS USING STEP RESPONSES FOR LTIS

a) Program for Comparison of various methods using Step Responses for LTICS

% Program to find integral square errors and comparison of step response curves for given higher order system, proposed lower order system and with the other models from the literature
% Illustration 6.1 (Krishnamurthy and Seshadri 1978)
% Enter numerator and denominator of given higher order system
clc; clear all;
nuh=[35 1086 13285 82402 278376 511812 482964 194480];
denh=[1 33 437 3017 11870 27470 37492 28880 9600];
t=[0:0.1:10];
a=tf(nuh,denh);
% Enter numerator and denominator of proposed lower order system
num1=[35 61.9485];
den1=[1 3.2467 3.0577];
b=tf(num1,den1);
% Enter numerator and denominator of Pal Model
nump=[33.00792 145.24272];
dep=[1 5.39208 7.169529];
c=tf(nump,denp);
% Enter numerator and denominator of Gutmann Model
numg=[51.52715 145.24272];
deng=[1 5.39208 7.169521];
d=tf(numg,deng);

% Enter numerator and denominator of Krishnamurthy et al model
numk=[16.63852 9.664227];
denk=[1 0.90024 0.47705];
e=tf(numk,denk);

% Enter numerator and denominator of Chen et al model
numc=[14.12423 5.68755];
denc=[1 0.844593 0.280751];
f=tf(numc,denc);

% Obtain the response for higher and lower order system from various methods (includes proposed method)
[y1,t]=step(a,t);
y2,t]=step(b,t);
y3,t]=step(c,t);
y4,t]=step(d,t);
y5,t]=step(e,t);
y6,t]=step(f,t);

% Calculate the integral square error between higher and lower order system from various methods
per_indexproposed=sum((y1-y2).*(y1-y2))
per_indexpal=sum((y1-y3).*(y1-y3))
per_indexgut=sum((y1-y4).*(y1-y4))
per_indexkris=sum((y1-y5).*(y1-y5))
per_indexchen=sum((y1-y6).*(y1-y6))

% Step response of all lower order models and higher order model
step(a,'r',b,'b',c,'g',d,'c',e,'m',f,'y',t) % to obtain colour figures
b) Program for Comparison of various methods using Step Responses for LTIDS

%Program to find integral square errors and comparison of step response curves for given higher order system,
%proposed lower order system and with the other models from the literature
%Illustration 6.4 (Prasad 1993)
clc;
clear all;
in=ones(1,61);
t=0:1:60 %time instants for the plot which occurs in x-axis
%Enter the given higher order LTIDS
numh=[1.682 1.116 -0.21 0.152 -0.516 -0.262 0.044 -0.018]
denh=[8 -5.046 -3.348 0.63 -0.456 1.548 0.786 -0.132 0.018]
y1=dstep(numh,denh,t) %obtaining the y-axis values for particular time instants for higher order systems
outh=filter(numh,denh,in);
stem(t,outh,'ro')
hold on;
%Enter the lower order polynomial from the proposed Scheme
numl=[0.21025 -0.1486]
denl=[1 -1.7890 0.8507]
y2=dstep(numl,denl,t)
outl=filter(numl,denl,in);
stem(t,outl,'b+')
hold on;
%Enter the lower order polynomial from the Sastry Model
nums=[0.3166654281 -0.277339669]
dens=[1 -1.68317616 0.72277914]
y3=dstep(nums,dens,t)
outs=filter(nums,dens,in);
stem(t,outs,'g+')
hold on;
% Enter the lower order polynomial from the Prasad Model
nump=[0.08516 -0.03123]
denp=[1 -1.730351 0.784282]
y4=dstep(nump,denp,t)
outp=filter(nump,denp,in);
stem(t,outp,'mx')
% Calculate the integral square error between higher and lower order system
% from various methods
per_indexproposed=sum((y1-y2).*(y1-y2))
per_indexsastry=sum((y1-y3).*(y1-y3))
per_indexprasad=sum((y1-y4).*(y1-y4))