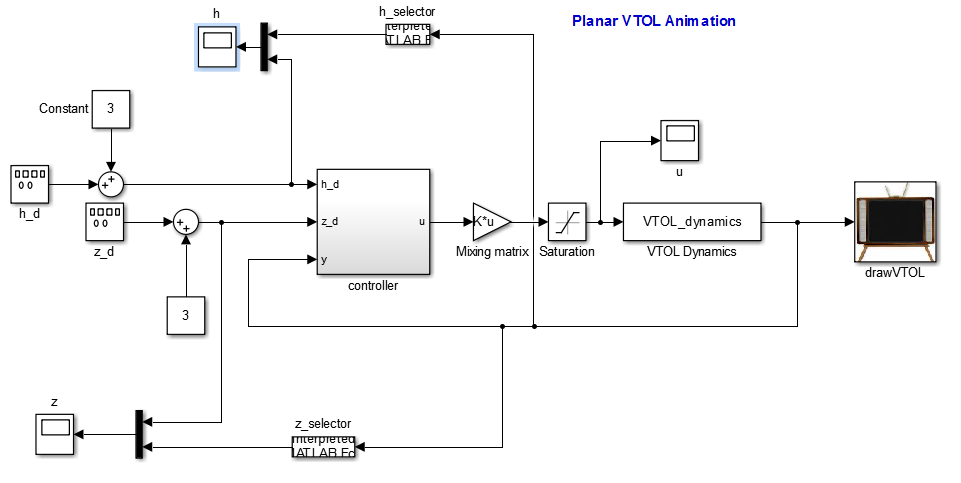
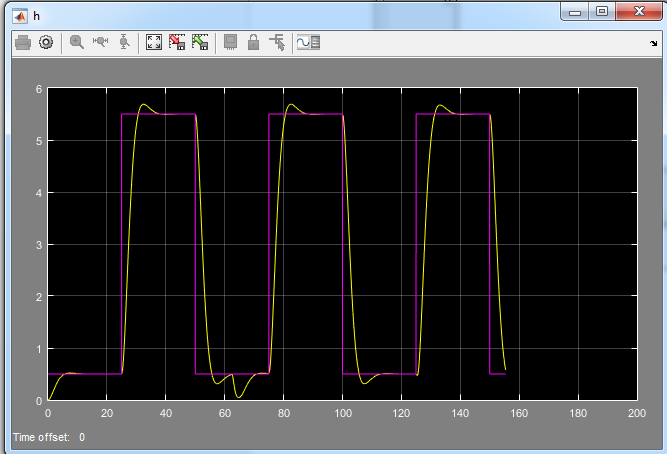
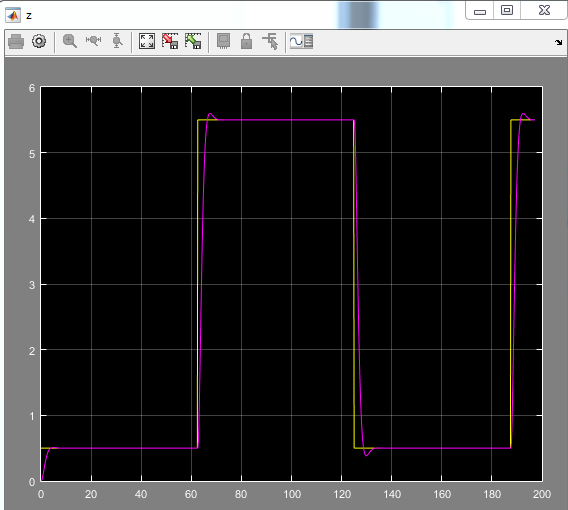
**Planar Quad Code, Plots, & Figures**



h response plot

z response plot

STATE SPACE PARAMS

%%%%%%%%%%%%%%%%%%%%%\_\_State Space Controller Params\_\_%%%%%%%%%%%%%%%%%%%%

%Desired performance characteristics

wn\_h = 1\*sqrt(P.kp\_h/(P.mc+2\*P.mr));

wn\_th = 1\*sqrt(b0\*P.kp\_th);

wn\_z = 1\*sqrt(b1\*P.kp\_z);

zeta\_h = 1\*0.707;

zeta\_th = 1\*0.707;

zeta\_z = 1\*0.707;

char\_poly\_h = [1, 2\*zeta\_h\*wn\_h, wn\_h^2];

des\_poles\_h = roots(char\_poly\_h);

char\_poly\_z\_th = conv([1, 2\*zeta\_z\*wn\_z, wn\_z^2],...

[1, 2\*zeta\_th\*wn\_th, wn\_th^2]);

des\_poles\_z\_th = roots(char\_poly\_z\_th);

%State space matrices

%%%%%%%%%%Up and Down%%%%%%%%%%%%%

A\_h = [0 1;0 0];

B\_h = [0;(1/(P.mc+2\*P.mr))];

C\_h = [1 0];

D\_h = [0];

%Check to see if system is controlable

if rank(ctrb(A\_h,B\_h)) ~= 2, disp('System Not Controllable');end

P.K\_h = place(A\_h,B\_h,des\_poles\_h);

P.kr\_h = -1/(C\_h(1,:)\*inv(A\_h-B\_h\*P.K\_h)\*B\_h);

%%%%%%%%%%%Side to Side%%%%%%%%%%%%%%%%

A = [0 0 1 0;0 0 0 1;0 -P.Fe/(P.mc+2\*P.mr) P.mu/(P.mc+2\*P.mr) 0;...

0 0 0 0];

B = [0; 0; 0; 1/(P.Jc+2\*P.mr\*P.d^2)];

C = [1 0 0 0;0 1 0 0];

D = [0; 0];

%Check to see if system is controlable

if rank(ctrb(A,B)) ~= 4, disp('System Not Controllable');end

P.K = place(A,B,des\_poles\_z\_th);

P.kr = -1/(C(1,:)\*inv(A-B\*P.K)\*B);

**VTOL STATE SPACE CONTROLLER FUNCTION**

function output = Planar\_VTOL\_SSctrl(in,P)

h\_d = in(1);

z\_d = in(2);

z = in(3);

h = in(4);

theta = in(5);

t = in(6);

%Digitally compute the derivatives zdot and thetadot

persistent zdot

persistent z\_dl

persistent thetadot

persistent theta\_dl

persistent hdot

persistent h\_dl

%Initialize persistent variables at simulation start

if t<P.Ts,

zdot = 0;

z\_dl = 0;

thetadot = 0;

theta\_dl = 0;

hdot = 0;

h\_dl = 0;

end

zdot = (2\*P.tau - P.Ts)/(2\*P.tau + P.Ts)\*zdot...

+ 2/(2\*P.tau + P.Ts)\*(z - z\_dl);

thetadot = (2\*P.tau - P.Ts)/(2\*P.tau + P.Ts)\*thetadot...

+ 2/(2\*P.tau + P.Ts)\*(theta - theta\_dl);

hdot = (2\*P.tau - P.Ts)/(2\*P.tau + P.Ts)\*hdot...

+ 2/(2\*P.tau + P.Ts)\*(h - h\_dl);

z\_dl = z;

theta\_dl = theta;

h\_dl = h;

%Construct the states

x\_h = [h; hdot];

x = [z; theta; zdot; thetadot];

%Compute the state feedback controller

F = sat(-P.K\_h\*x\_h + P.kr\_h\*h\_d + P.Fe, 2\*P.fmax);

T = sat(-P.K\*x + P.kr\*z\_d, P.taumax);

output = [F; T];

end

% saturation function

function out = sat(in,limit)

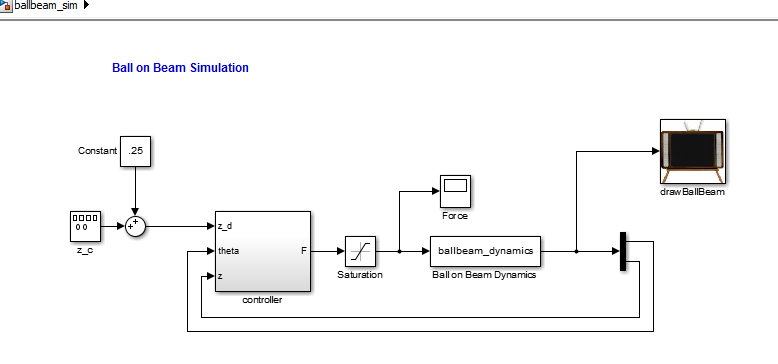
if in > limit, out = limit;

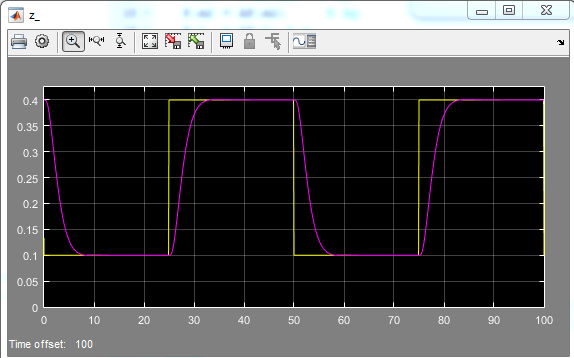
elseif in < -limit, out = -limit;

else out = in;

end

end

**Ball and Beam Code, Plots, & Figures**



z response plot

STATE SPACE PARAMS

%%%%%%%%%%%%%%%%%%%\_\_Sate Space Model Params\_\_%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%desired response parameters

wn\_th = 1.5\*2.652;

wn\_z = 1\*.6991;

zeta\_th = 0.7;

zeta\_z = 0.9;

%state space matrices

A = [0 1 0 0;0 0 -P.g 0;0 0 0 1;...

(-P.m1\*P.g)/((P.m2\*(P.L^2)/3)+P.m1\*ye^2) 0 0 0];

B = [0; 0; 0; P.L/((P.m2\*(P.L^2)/3)+P.m1\*ye^2)];

C = [1 0 0 0; 0 0 1 0];

D = [0; 0];

%gains for pole locations

charpoly = conv([1, 2\*zeta\_z\*wn\_z, wn\_z^2],...

[1, 2\*zeta\_th\*wn\_th, wn\_th^2]);

des\_poles = roots(charpoly);

%Check to see if system is controllable

if rank(ctrb(A,B)) ~= 4,

disp('System Not Controlable');

else

P.K = place(A,B,des\_poles);

P.kr = -1/(C(1,:)\*inv(A-B\*P.K)\*B);

end

**BALL AND BEAM STATE SPACE CONTROLLER FUNCTION**

function F = ballbeam\_SSctrl(in,P)

z\_d = in(1);

z = in(2);

theta = in(3);

t = in(4);

%use digital differentiator to find zdot and thetadot

persistent zdot

persistent z\_d1

persistent thetadot

persistent theta\_d1

%reset persistent variables at simulation start

if t < P.Ts

zdot = 0;

z\_d1 = 0;

thetadot = 0;

theta\_d1 = 0;

end

%compute the digital derivatives

zdot = (2\*P.tau-P.Ts)/(2\*P.tau+P.Ts)\*zdot...

+ 2/(2\*P.tau+P.Ts)\*(z-z\_d1);

thetadot = (2\*P.tau-P.Ts)/(2\*P.tau+P.Ts)\*thetadot...

+ 2/(2\*P.tau+P.Ts)\*(theta-theta\_d1);

z\_d1 = z;

theta\_d1 = theta;

%Construct the State

x = [z; zdot; theta; thetadot;];

Fe = P.m2\*P.g/2; %+ P.m1\*P.g\*z/P.L;

%Compute the State Feedback Controller

F = sat(-P.K\*x + P.kr\*z\_d + Fe, P.Fmax);

end

% saturation function

function out = sat(in,limit)

if in > limit, out = limit;

elseif in < -limit, out = -limit;

else out = in;

end

end