



function out2 = GramHWProb()

W1 = [0,0,0,0]'; %This is the initial value for determining the Gramian

X1 = [1,0,0,1]'; %This is the initial value for determining the STM

tspan = [0,2]; %Time span for simulation

xspan = [[1;3],[-1;5]]; %Initial and final positions, respectively.

%Define A,B from eq(2.1)

A = inline('[1,t;0,2]','t');

B = inline('[1;5-t]','t');

%Set option for ode45 functions...

options = odeset('InitialStep',0.025,'RelTol', 1e-6);

%Calculate W to get Wr(t0,t1).

[TW,W] = ode45(@(t,x) odetest(t,x,1,A,B), tspan, W1, options);

%Calculate X to get X(t1,t0), and X(TX,t0) for u() calculation.

[TX,X] = ode45(@(t,x) odetest(t,x,2,A,B), tspan, X1, options);

%Calculate u(TX,X,B)

out1 = uCalc(W(end,:),TX,X,B,xspan);

%Plot u(TX) v. TX

figure;

title('\fontsize{16} Plot of Control, u(t), vs. Time');

xlabel('\fontsize{13} Time');

ylabel('\fontsize{13} u(t)');

hold on

scatter(TX,out1{1}, 'MarkerFaceColor','b', 'SizeData', 8^2);

hold off

%Calculate x values

[Tx,x] = ode45(@(t,x) xCalcODE(t,x,A,B,out1{2}), tspan, [X1;xspan(:,1)]', options);

%Plot x(2) v. x(1) trajectory

figure;

title('\fontsize{16} Trajectory Through Time');

xlabel('\fontsize{13} x(1) Component (from 1 to -1)');

ylabel('\fontsize{13} x(2) Component (from 3 to 5)');

hold on

scatter(x(:,5),x(:,6), 'MarkerFaceColor','b', 'SizeData', 8^2);

hold off

%Pass everything just incase.

out2 = {TW,W,TX,X,Tx,x,out1};

function xprime = odetest(t,x,ii,A,B)

% Easier to define A,B above so that it can be fed into u, x

% Since both are linearly dependent on t, there's no need to evaluate their derivatives within the odetest function / ode solver.

n = length(x)^.5;

xNew = [];

for jj = 1:n

xNew = [xNew,x(1+(n\*(jj-1)):n\*jj,1)];

end

switch ii

case 1 %Calculate Wr

gp = @(t,xin) A(t)\*xin + B(t)\*B(t)' + xin'\*A(t)';

case 2 %Calculate STM

gp = @(t,xin) A(t)\*xin;

end

tempgp = gp(t,xNew);

gpNew =[];

for jj = 1:n

gpNew = [gpNew;tempgp(:,jj)];

end

%output, i.e., updated xprime value

xprime = gpNew;

function out = uCalc(tempW,TX,X,B,xspan)

%Put Wr, STM in matrix form.

tempSTM = X(end,:);

tempSTM = tempSTM(:);

tempW = tempW(:);

n = (length(tempW)^.5);

Wr = [];

STM = [];

for jj = 1:n

Wr = [Wr,tempW(1+(n\*(jj-1)):n\*jj)];

STM = [STM,tempSTM(1+(n\*(jj-1)):n\*jj)];

end

K = Wr^-1\*(xspan(:,2) - STM\*xspan(:,1));

u = [];

for jk = 1:length(TX)

index = n\*(jk-1);

Xjk = [X(1+index:2+index)',X(3+index:4+index)'];

u(jk) = B(TX(jk))'\*Xjk'\*K;

end

%Pass the constant K for use in calculating the trajectory function

K = STM'\*K;

out = {u,K};

function xprime = xCalcODE(t,xin,A,B,K)

%Set current values for X and x; make sure they are correct orientation and that X is in matrix form.

Q = xin(1:4); Q=Q(:); Q = [Q(1:2),Q(3:4)];

x = xin(5:end); x = x(:);

tempQ = A(t)\*Q; %X update for calculating u(t) on fly.

tempxp = A(t)\*x + B(t)\*B(t)'\*(Q^-1)'\*K; %Ax+Bu

%Updated xprime value.

xprime = [tempQ(:,1);tempQ(:,2);tempxp(:)];