function [xSol, uSol] = HW05OpenCont(Start, Stop, Guess)

%Part (i): Find optimal control for original system

solinit = bvpinit(linspace(Start,Stop,Stop-Start+1),Guess);

uSol = bvp4c(@OrigOdeFunc, @OrigBCFunc, solinit);

%Part (ii): simulate the system using u\*

%Simulate x trajectory with the Theory Control

xSol = ode45(@(t,y)SimxWCont(t,y,uSol), [0,10], 4);

%Part (iii): Calculate the Kalman Gain, get S

%Should be able to use BVP solver to get S, since this works when provided with a final value for the function.

solinit = bvpinit(linspace(Start,Stop,Stop-Start+1),Guess);

SSol = bvp4c(@SOdeFunc, @SBCFunc, solinit);

%Part (v):

Delta = inline('4\*(t>2).\*((3-2)/2-abs(t-2.5)).\*(t<3)', 't');

xSolDel = ode45(@(t,y)SimxWContIn(t,y,uSol,Delta), [0,10], 4);

%Calculate cost of solution.

t = linspace(0,10);

y(:,1) = deval(xSol,t);

y(:,2) = -0.5\*deval(uSol,t,2);

CostFunc = inline('.5\*z(:,1).^2 + 3\*z(:,2).^2', 'z');

Cost = sum(Stop / length(t) \* CostFunc(y)) + 5\*y(1,end)^2

%Functions for orig system solving

function dxdy = OrigOdeFunc(~,y,~)

dxdy = [.5\*y(1) - 3/2\*y(2);...

-y(1) + .5\*y(2)];

function res = OrigBCFunc(ya, yb)

res = [ya(1) - 4;...

10\*yb(1) - yb(2)];

%Function to feed into ode45 for simulating 'x'

function dy = SimxWCont(t,y,uSol)

u = -0.5\*deval(uSol,t,2);

dy = .5\*y + 3\*u;

%Function with inline input to feed into ode45 for simulating 'x'

function dy = SimxWContIn(t,y,uSol,F)

u = -0.5\*deval(uSol,t,2);

dy = .5\*y + 3\*u + F(t);

%Function for system with delta solving

function dxdy = OrigOdeFuncIn(t,y,F)

dxdy = [.5\*y(1) - 3/2\*y(2) + F(t);...

-y(1) + .5\*y(2)];

%Functions for finding sol for S

function dy = SOdeFunc(~,y,~)

dy = -y(1) + y(1)^2\*(9/6) - 1;

function res = SBCFunc(~, yb)

res = yb(1) - 10;