EML 6934 - Optimal Control

Bonus Assignment #2

Arif Mohammed

Objective

The objective of this assignment is to solve the Hyper-Sensitive optimal control problem and Brachistochrone Optimal Control Problem using Indirect multiple shooting method.

Problem #1:

Hyper-Sensitivity Optimal Control Problem

First order optimality conditions formulated in the bonus assignment #1 were used and are solved by using the below attached code.

First Order optimality conditions are

$$\frac{dx}{dt} = -x(t) - \lambda(t);$$
$$\frac{d\lambda}{dt} = -x(t) + \lambda(t);$$

Optimal Control: -

$$u(t) = -\lambda(t);$$

Boundary Conditions: -

$$x(0) = 1;$$
$$x(tf) = 1;$$
$$tf = fixed;$$
$$t0 = 0;$$
$$tf = [10, 20, 30, 40, 50];$$

MATLAB Code for Solving Hyper-Sensitivity Problem Using the Indirect multiple shooting method is attached in next page.

```
clear all; close all; clc
%% HS Main function bonus 2
% Given
x0 = 1;
xf=1;
t0 = 0;
tf = 50;
nx = 1; %(only lambda0 is variable)
k = 10; %(No. of intervals)
tau = linspace(-1,+1,k+1);
lambda0guess = 0;
pRemint0guess = zeros(2*nx,k-1);
zguess = [lambda0guess;pRemint0guess(:)];
options = optimset('TolX',1e-8,'TolFun',1e-8,'display','Iter','MaxFunEvals',↙
10000,'MaxIter',10000);
z = fsolve(@HSError1,zguess,options,x0,t0,xf,tf,nx,k,tau);
[E,t,y] = HSError1(z,x0,t0,xf,tf,nx,k,tau);

figure(1)
plot(t,y(:,1),'r--');
xlabel('tau');
ylabel('x(tau)');

figure(2)
plot(t,y(:,2),'b--');
xlabel('tau');
ylabel('lambda(tau)')
set(gca,'YDir','reverse');

%% Error Function
function [E,t,y] = HSError1(z,x0,t0,xf,tf,nx,k,tau)
lambda0 = z(1);
pRemint = z(2:end);
pRemint = reshape(pRemint,2*nx,k-1);% Reshaping      2 X k-1 matrix.
options = odeset('RelTol',1e-8);
E = [];% alocating memory for E
t = [];% alocating memory for t
y = [];% alocating memory for y
% assigning the guess values for x0 and lambda0guess at all the intervals
for i = 1:k
    if i == 1
        y0 = [x0;lambda0];
    else
        y0 = pRemint(:,i-1);
    end
    tauspan = [tau(i),tau(i+1)];
    [tout,yout] = ode113(@HSmyode1,tauspan,y0,options,t0,tf);
    ytf = yout(end,:)';


    if i < k
        E = [E;ytf-pRemint(:,i)];
    end
    t = [t;tout];
```

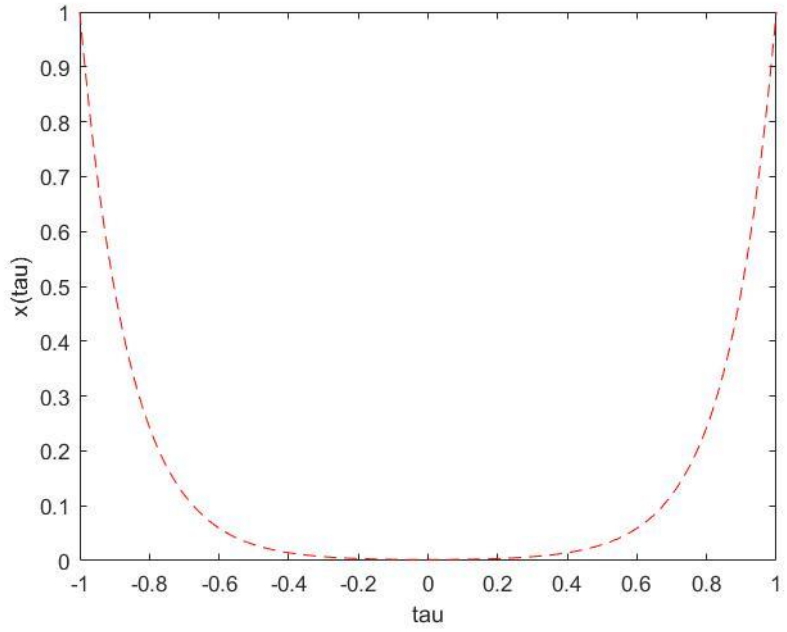
```
    y = [y ;yout];
end
E = [E;ytf(1) - xf];% Computing final error.
end

%% Dynamics Function
function dydt = HSmyode1(t,y,t0,tf)
dydt = zeros(2,1);
dydt(1) = -y(1)-y(2);
dydt(2) = -y(1)+y(2);
dydt = (tf-t0)/2 * dydt;%% For scaling it between -1 and +1 timespan
end
```

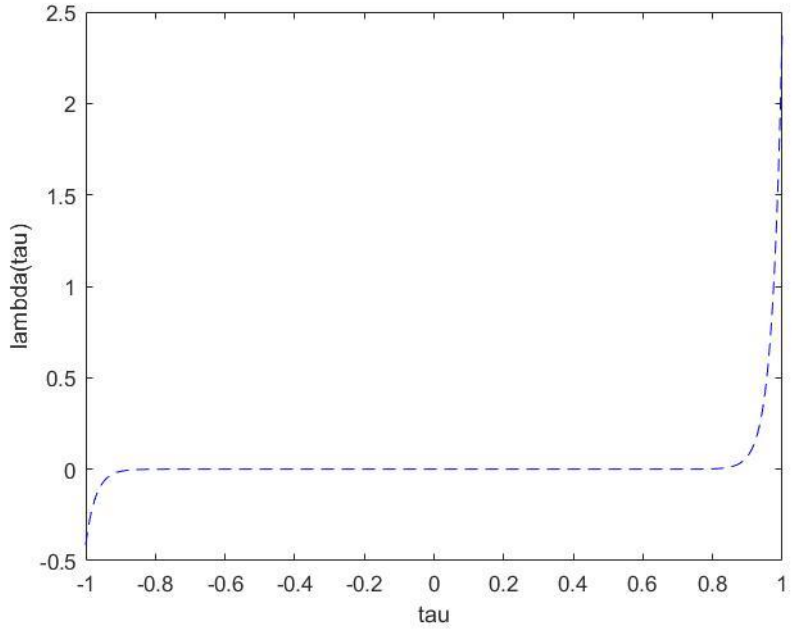
Results of Hyper-Sensitivity problem using Indirect Multiple Shooting Method: -

For tf = 10.

Trajectory Plot on τ grid:

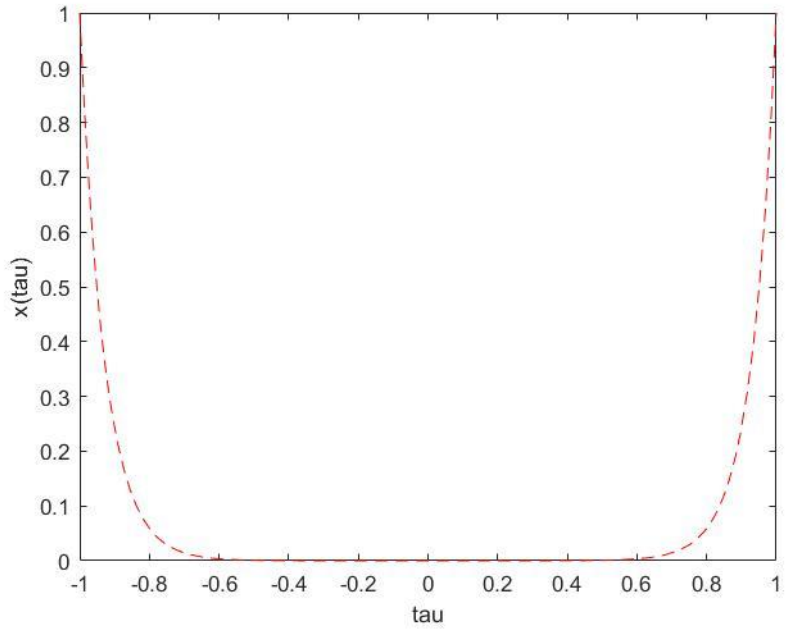


Control Plot on τ grid:

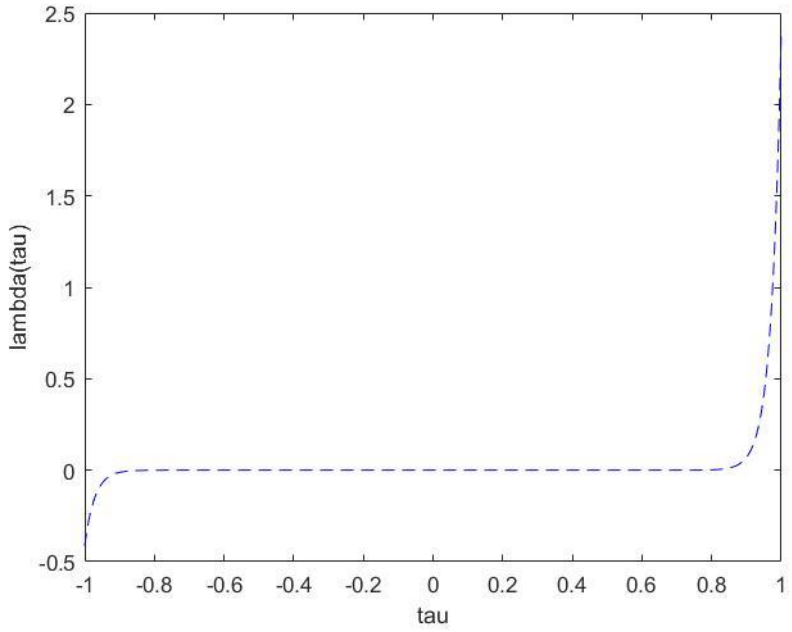


For tf = 20.

Trajectory Plot on τ grid:

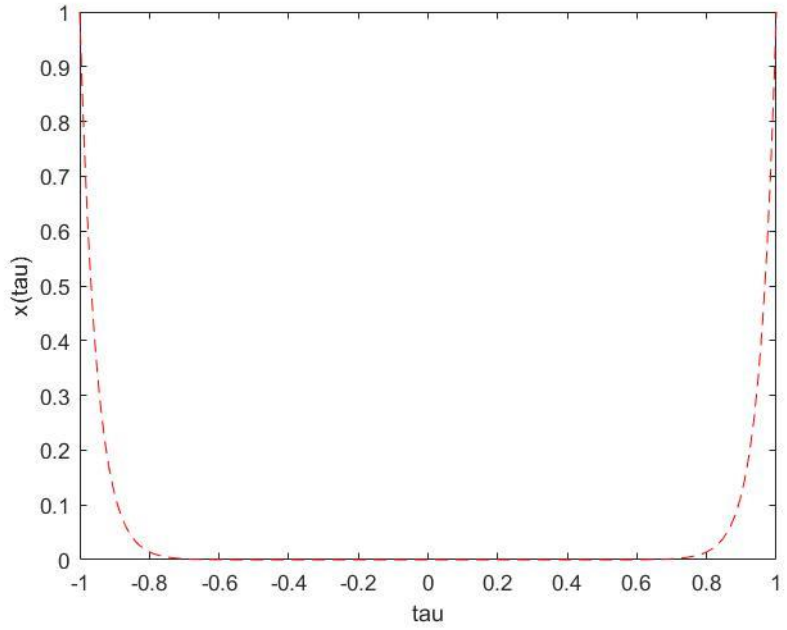


Control Plot on τ grid:

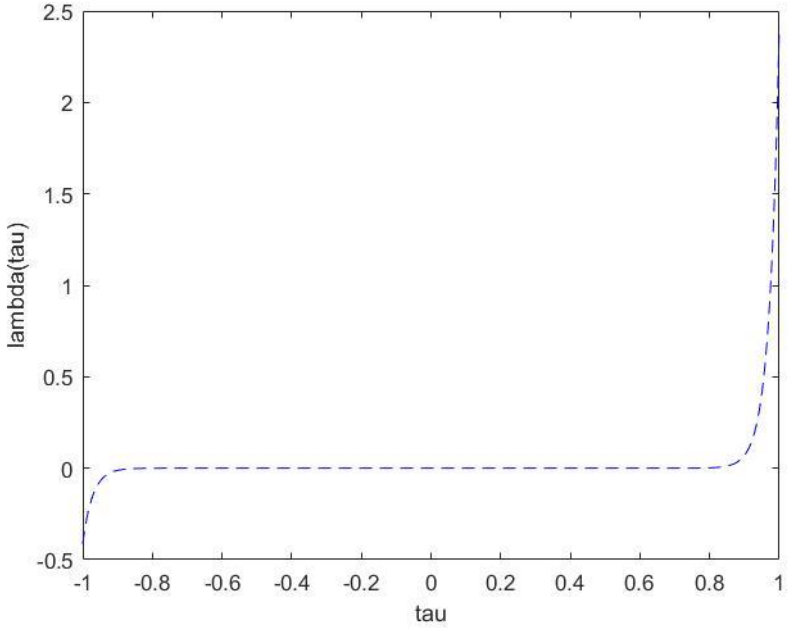


For tf = 30.

Trajectory Plot on τ grid:

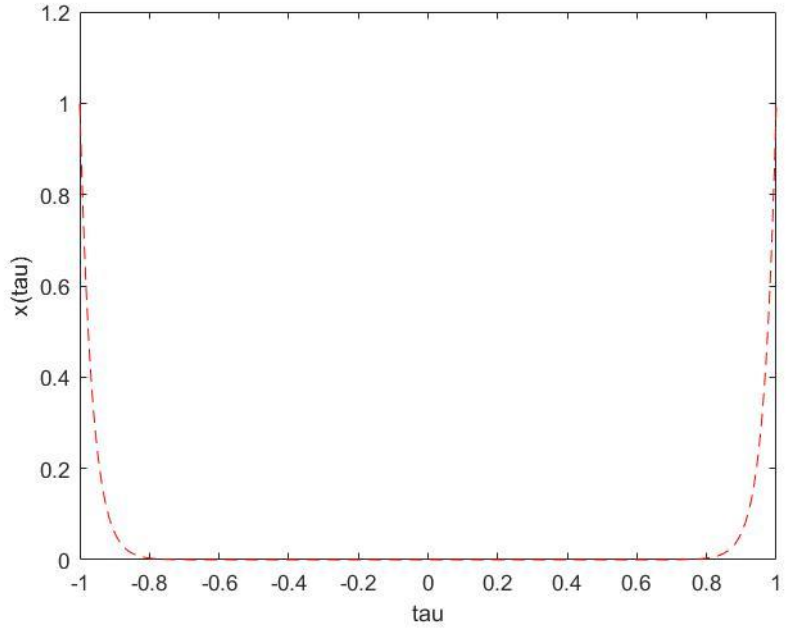


Control Plot on τ grid:

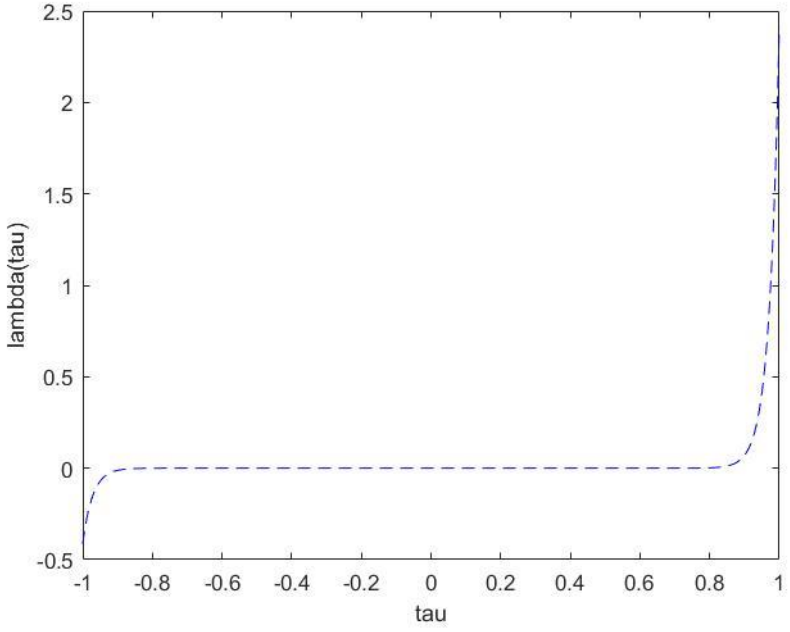


For tf = 40.

Trajectory Plot on τ grid:

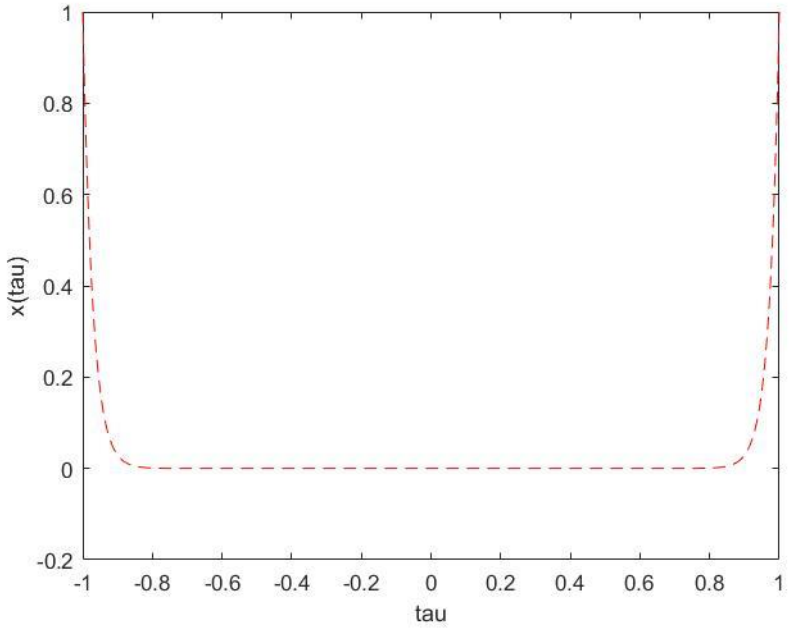


Control Plot on τ grid:

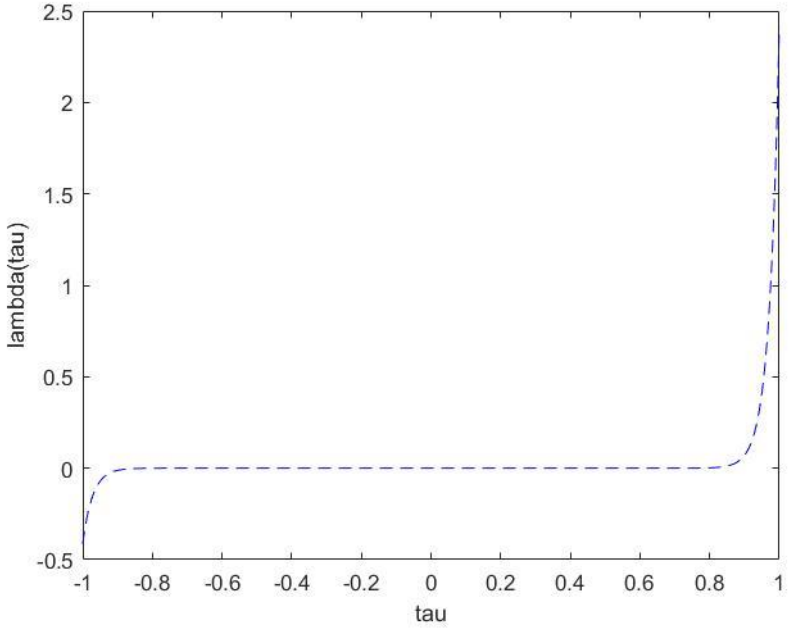


For tf = 50.

Trajectory Plot on τ grid:



Control Plot on τ grid:



By checking the Y array generated by solving the problem ("Y(2,1)")

It was found that initial value of lambda was 0.4142.

Conclusion:

As the name of the technique describes, Hyper-Sensitive problem was solved by shooting initial values of the state and co-state by dividing the tau span into 10 intervals between [-1, +1]. It was observed that Hyper-Sensitive problem was solved for longer duration of time even up to 50 secs which was not solved using the single shooting method in previous assignment. By increasing the value of tf it was observed that the number of iterations for solving the dynamic state and co-state equations was increasing and the accuracy of the solution was decreasing. We can observe that the trajectory and the control plots remain same regard less of time span as they are plot by transforming it in the range [-1, +1].

Problem #2

Brachistochrone Optimal Control Problem

First order Optimality Conditions from Bonus Assignment #1

$$-\frac{\partial H}{\partial x}=\frac{\partial \lambda_x}{\partial t}=0;$$

$$-\frac{\partial H}{\partial y}=\frac{\partial \lambda_y}{\partial t}=0;.$$

$$-\frac{\partial H}{\partial v}=-\lambda_x\cdot\sin(\theta)-\lambda_y\cdot\cos(\theta)\ =0;$$

$$\frac{\partial H}{\partial \theta}=\lambda_x\cdot v\cdot\cos(\theta)\ -\left(\lambda_y\cdot v+\lambda_v\cdot g\right)\cdot\sin(\theta)\ =0;$$

Constraints:

$$\frac{dx}{dt}=v(t)\cdot\sin(\theta);$$

$$\frac{dy}{dt}=v(t)\cdot cos(\theta);$$

$$\frac{dv}{dt}=g\cdot\cos(\theta);$$

Boundary Conditions:

$$x(0)\ =0;$$

$$y(0)\ =0;$$

$$v(0)\ =0;$$

$$let\ final\ time\ be\ tf\ .$$

$$x(tf)\ =2;$$

$$y(tf)\ =2;$$

$$v(tf)\ =tf\ =free;$$

MATLAB Code for Solving Brachistochrone optimal control Problem Using the Indirect multiple shooting method is attached in next page.

```
%% Brochistochrone Bonus 2
clear all;close all;clc

% Given
x0 = 0;
y0 = 0;
v0 = 0;
t0 = 0;
xf = 2;
yf = 2;
n = 3;% No. of states
lambdaxguess = 0;
lambdayguess = 0;
lambdavguess = 0;
tfguess = 10;
g = 10;
k =2; % 2 stage shooting
tau = linspace(-1,+1,k+1);
p0guess = zeros(2*n,k-1);
z0g = [lambdaxguess;lambdayguess;lambdavguess;tfguess;];
z0 = [z0g;p0guess(:)]; % Initial variables for shooting
options = optimset('Display','Iter','TolX',1e-8,'TolFun',1e-8);
z = fsolve(@BSError,z0,options,t0,x0,y0,v0,xf,yf,g,n,k,tau);
[E,t,p] = BSError(z,t0,x0,y0,v0,xf,yf,g,n,k,tau);

%%Plots

figure(1)
plot(t,p(:,1),'r*-',t,p(:,2),'b*-',t,p(:,3),'b--')
title('Trajectory and velocity components plot in tau grid');
legend('X(tau)','Y(tau)','V(tau)');
grid on;

% Evaluating the control plot

for j = 1: length(t)
vtj = p(j,3);
lambdaxtj = p(j,4);
lambdaytj = p(j,5);
lambdavtj = p(j,6);
thetatj(j) = fsolve(@solvecontrol,0,options,lambdaxtj,lambdaytj,lambdavtj,vtj,g);
end

% control plot
figure(2)
plot(t,thetatj)
hold on
title('Control vs Tau')
xlabel('Tau')
ylabel('u(tau)')
legend('Control u(tau)');

%% Optimal trajectory plot
figure(3)
```

```
plot(p(:,1),p(:,2))
set(gca,'YDir','reverse')
hold on
title('Optimal trajectory')
xlabel('X(tau)')
ylabel('Y(tau)')
xlim([0 2.00])
ylim([0 2.00])
legend('optimal Trajectory');
%% Error function
function [E,t,p] = BSError(z0,t0,x0,y0,v0,xf,yf,g,n,k,tau)
% p0 = [x0;y0;v0;z0(1);z0(2);z0(3)];
tfguess = z0(4);
premint = z0(5:end);
premint = reshape(premint,2*n,k-1);
options = odeset('RelTol',1e-8);
options1 = optimset('Display','Iter','TolX',1e-8,'TolFun',1e-8);
E = [];
p = [];
t = [];
for i = 1:k
    if i == 1
        p0 = [x0;y0;v0;z0(1);z0(2);z0(3)];
    else
        p0 = premint(:,k-1);
    end

    tauspan = [tau(i),tau(i+1)];
    [tout,pout] = ode113(@BSODE,tauspan,p0,options,t0,tfguess,g);
    ptf = pout(end,:)';

    if i < k
        E = [E;ptf - premint(:,i)];
    end
    t = [t; tout];
    p = [p; pout];
end
xtf = ptf(1);
ytf = ptf(2);
vtf = ptf(3);
lambdaxtf = ptf(4);
lambdaytf = ptf(5);
lambdavtf = ptf(6);
theta0guess = 0;
thetaf = fsolve(@solvecontrol,theta0guess,options1,lambdaxtf,lambdaytf,lambdavtf,vtf, ↙
g);
htfout = lambdaxtf*vtf*sin(thetaf)+(lambdaytf*vtf+lambdavtf*g)*cos(thetaf);
E = [E;ptf(1)-xf;ptf(2)-yf;lambdavtf;htfout+1];
end

%% ODE Function
function pdot = BSODE(t,p,t0,tf,g)
thetaGuess = 0;
v = p(3);
```

```
lambdax = p(4);
lambday = p(5);
lambdav = p(6);
options = optimset('Display','Iter','TolX',1e-8,'TolFun',1e-8);
theta = fsolve(@solvecontrol,thetaGuess,options,lambdax,lambday,lambdav,v,g);
xdot = v*sin(theta);
ydot = v*cos(theta);
vdot = g*cos(theta);
lambdaxdot = 0;
lambdaydot = 0;
lambdavdot = -lambdax*sin(theta)-lambday*cos(theta);
pdot = [xdot;ydot;vdot;lambdaxdot;lambdaydot;lambdavdot];
pdot = ((tf-t0)/2)*pdot;
end

%% Solve control function
function htheta = solvecontrol(theta,lambdax,lambday,lambdav,v,g)
htheta = lambdax*v*cos(theta) - (lambday*v + lambdav*g)*sin(theta);
end
```

Result for Brachistochrone Problem Using Indirect Multiple Shooting Method:

Initial Values of Co-state were found to be as follows:

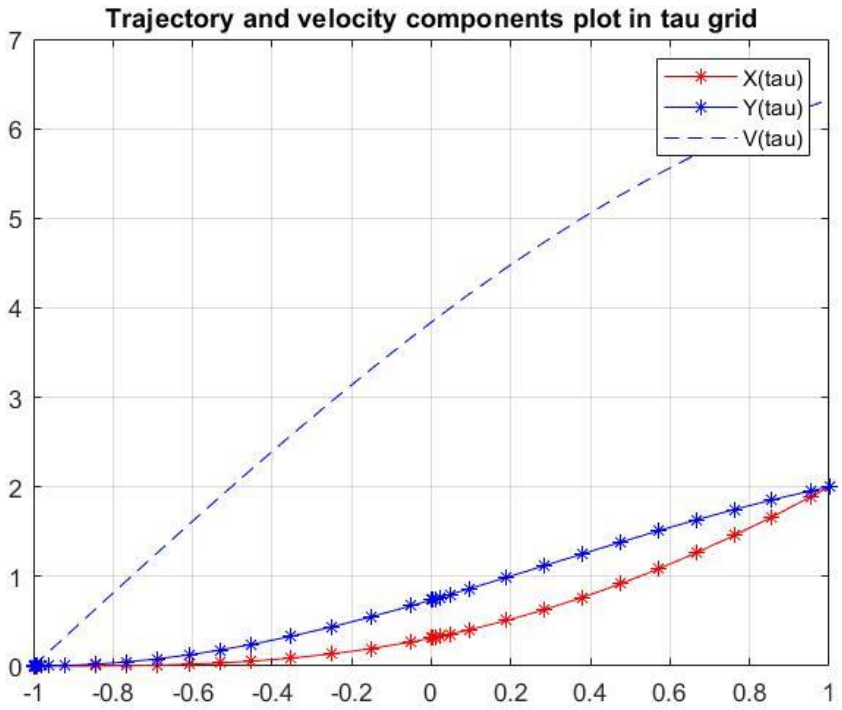
$$\lambda_x = -0.147709740249883;$$

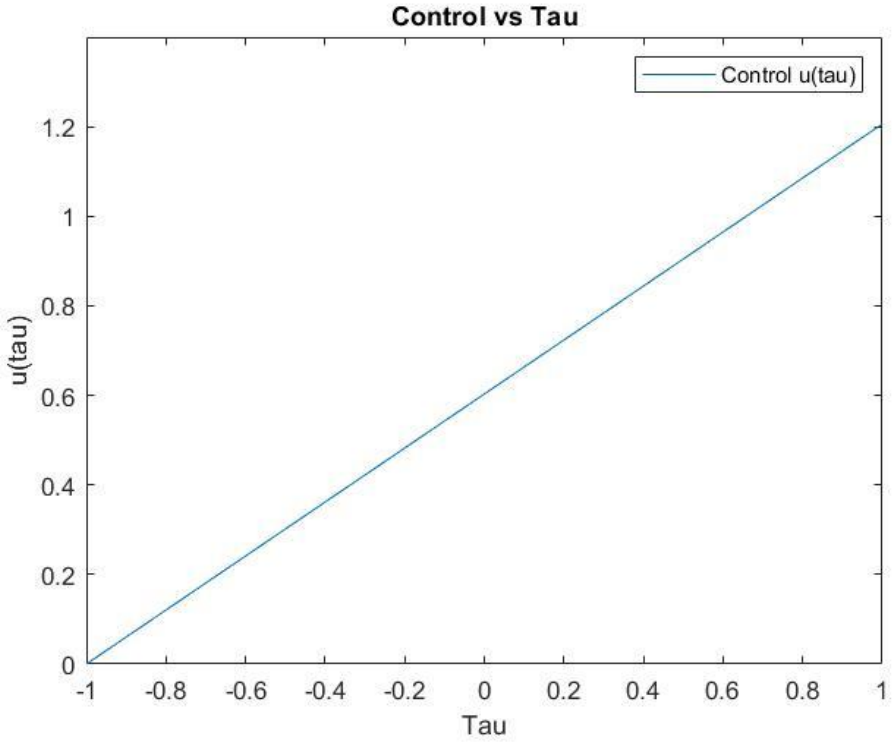
$$\lambda_y = -0.0564077337504497;$$

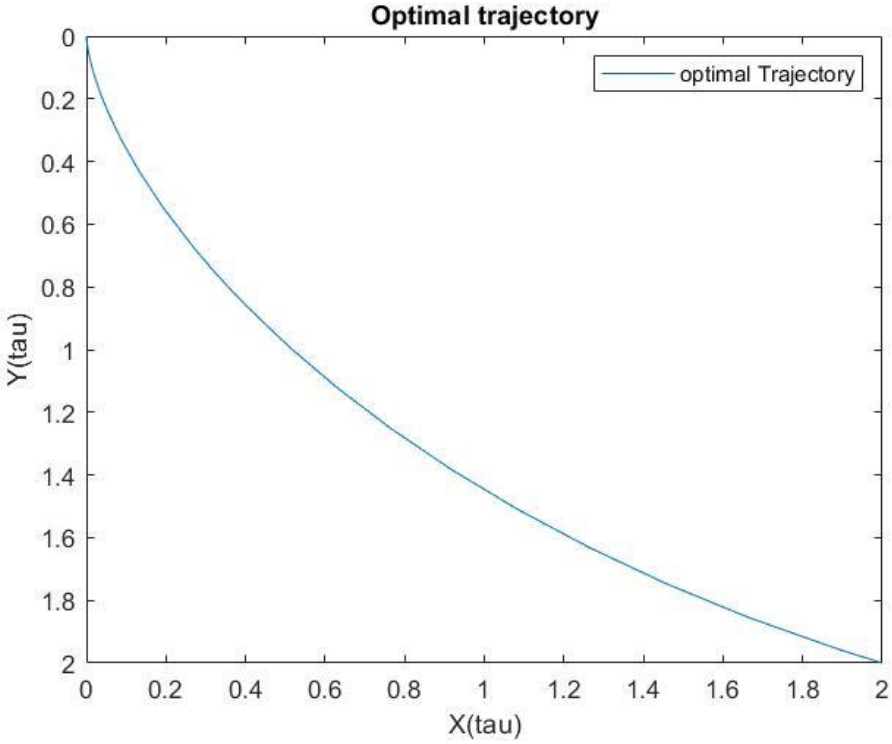
$$\lambda_v = -0.100000001719718;$$

$$V_{tf} = 6.32455534677782;$$

Trajectory and Velocity Plot on τ Grid:







- **Need to compute final time. Haven't done yet. Will submit it sooner professor.**