16.323 Optimal Control

PROBLEM SET 4

Due: Thursday, April 24, 2014

Author:
Daniel WIESE

Instructor:
Prof. Steven HALL

1 Optimal Control with BVP4C

The problem is to find the optimal controller for the inverted pendulum, with dynamics given by

$$\ddot{\theta}(t) = \sin(\theta(t)) + u$$

$\theta = 0$ corresponds to vertical. The cost functional to minimize is

$$J = \int_0^{t_f} (\theta^2 + \rho u^2) dt$$

For the purposes of this problem, we will take $t_f = 10$, $\rho = 10$.

Solution

This problem is described by the following equations

$$\begin{aligned} \dot{X} &= f(X, u) \\ J &= h + \int g dt \end{aligned}$$

where the state vector $X = [\ x_1\ x_2\ ]^\top$ where $x_1 = \theta$ and $x_2 = \dot{\theta}$. In our problem the function h for the terminal condition is

$$\boxed{h = 0}$$

The dynamics are given by

$$\begin{aligned} \dot{x}_1 &= f_1 \\ \dot{x}_2 &= f_2 \end{aligned}$$

where the function f is written in component form as

$$\boxed{\begin{aligned} f_1 &= x_2 \\ f_2 &= \sin x_1 + u \end{aligned}}$$

From the cost function we have

$$\boxed{g = x_1^2 + \rho u^2}$$

Forming the Hamiltonian

$$\begin{aligned}H &= g + p^\top f \\ &= g + p_1 f_1 + p_2 f_2 \\ &= x_1^2 + \rho u^2 + p_1 x_2 + p_2(\sin x_1 + u)\end{aligned}$$

Taking derivatives

$$\begin{aligned}H_x &= \begin{bmatrix}2x_1 + p_2 \cos x_1 & p_1\end{bmatrix} \\ H_u &= 2\rho u + p_2\end{aligned}$$

And since $\dot{p} = -H_x^\top$ and $H_u = 0$ we get the control is

$$u = -\frac{p_2}{2\rho}$$

Putting everything together, the differential equations (state and costate) which describe the system are

$$\boxed{\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= \sin x_1 - \frac{p_2}{2\rho} \\ \dot{p}_1 &= -2x_1 - p_2 \cos x_1 \\ \dot{p}_2 &= -p_1\end{aligned}}$$

We have no terminal constraint, so the function m is zero

$$\boxed{m = 0}$$

and we augment the terminal cost with terminal constraint as

$$w = h + \nu m$$

and so

$$\boxed{w = 0}$$

And to find the boundary conditions we are given initial conditions on the state, and no final conditions. So we need to use final conditions on the costate equations which come from

$$p(t_f) = \left(\frac{\partial w}{\partial x}\bigg|_{t_f} \right)^\top$$

So this with the above that $w = 0$ we get the final costate is zero. That is, the summary of all the boundary conditions is

$$\boxed{\begin{aligned} x_1(0) &= \pi \\ x_2(0) &= 0 \\ p_1(t_f) &= 0 \\ p_2(t_f) &= 0 \end{aligned}}$$

The scripts and functions which were used to solve this problem in MATLAB using `bvp4c.m` can be found in the appendix. A plot of the state and control histories is shown in Figure 1 below.

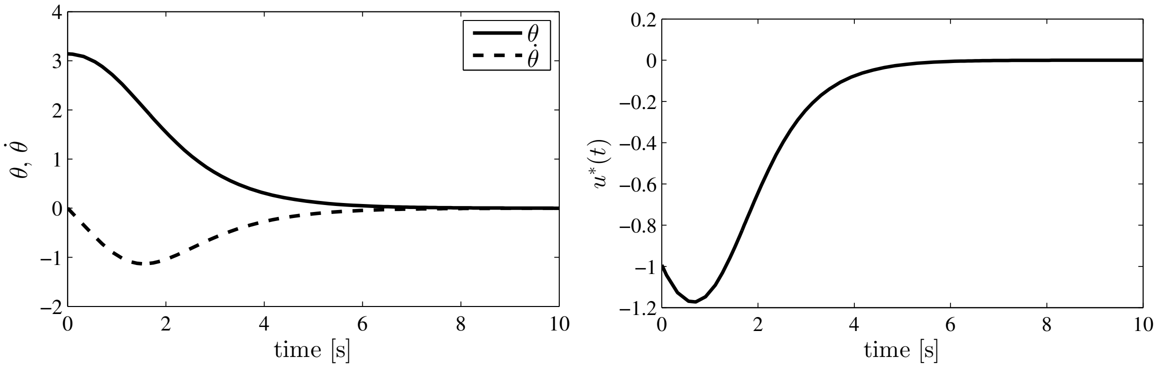

Figure 1: State and control history plots for the pendulum system

2 Paper Summary: Survey of Numerical Methods for Trajectory Optimization

In the paper “Survey of Numerical Methods for Trajectory Optimization”, the author, John T. Betts, reviews different approaches to solving the trajectory optimization problem, and summarizes the advantages and disadvantages of several methods. He starts with an overview of trajectory optimization and nonlinear programming (NLP) and discusses how the optimal control problem can be viewed as an extension of NLP.He provides an overview of numerical schemes and discusses five widely used methods for solving the trajectory optimization problem.

The general trajectory optimization problem is to determine the control input and parameters which will minimize a given performance index over a set of phases. The most general form is that where the objective function contains terms which depend on interval endpoints, as well as integrals over the time interval, and is known as the problem of Bolza.

Nonlinear programming is introduced by considering the unconstrained minimization problem. This problem is extended to handle equality constraints using Lagrange multipliers, and finally to inequality constraints. The NLP problem is summarized simply as minimizing a given function $F(x)$ while imposing some inequality constraints on a function $c(x)$. The author gives a brief overview of approaches to solve this problem, including the use of penalty/barrier functions as mentioned in class.

The optimal control problem is introduced as an extension of the nonlinear programming problem to an infinite number of variables. Using calculus of variations and augmented Lagrangians, this leads to a performance index which has continuous Lagrange multipliers, or costate variables. When optimality is sought by setting the first variation of this performance index to zero, the result is a set of necessary conditions which must be satisfied for optimality. This results in the Euler-Lagrange equations, including adjoint equations, control equations, and transversality conditions, which is a set of differential and algebraic equations, or DAE system. Solving such a system is a two-point boundary value problem. This problem is compared with the NLP problem, where the NLP Lagrange multipliers approximate the costate, and by using enough variables, the NLP solution approaches the optimal solution.

Considering the equations developed using variational methods, the author describes several different methods to solve these equations numerically. These methods are: direct and indirect shooting method, multiple shooting method, and both direct and indirect transcription. The direct shooting method is widely used, and works well for launch vehicle and orbit transfer applications, due to the small number of NLP variables, but is difficult to apply when the number becomes large. The indirect shooting method is less applicable, as it is more prone to errors in the initial guess, and requires the derivation of the necessary conditions. The multiple shooting method seeks to improve the other shooting methods by splitting the trajectory into small segments and shooting over each segment. This results in increased robustness, and the ability to leverage parallel process for improved computation time, although this method is still somewhat sensitive to the initial guess. Indirect transcription, or collocation methods involve using a piecewise continuous polynomial to approximate the given function. As with other indirect methods, indirect transcription requires determining the costate equations, and can be difficult to use with path inequality constraints. Direct transcription does’t require the costate equations or direct specification of path inequalities like in the indirect case. However, derivative information is required which is similar to the costate equations, and the direct method only approximates the solution of inequality constraints, often resulting in jump discontinuities in the control.

In summary, given a trajectory optimization problem, the control designer should consider things such as the number of NLP variables required, how difficult it is to derive the costate equations, and whether path inequalities are present. Using this information will allow the control designer to pick the solution method which will likely work the best to solve the given problem.

3 Optimal Control of Launch Vehicle

Consider the optimal control of thrust vector direction that maximizes the terminal velocity of a space launch vehicle in the rectilinear frame. The vehicle dynamics are represented as

$$\begin{aligned}\dot{u} &= a\cos\beta \\ \dot{v} &= a\sin\beta \\ \dot{x} &= u \\ \dot{y} &= v\end{aligned}$$

where (x, y) and (u, v) denote the position and the velocity of the rocket with respect to a rectilinear inertial frame. The thrust acceleration a is assumed to be a known function of time. The boundary conditions are given as:

$$\begin{aligned}u(t_0) = v(t_0) = x(t_0) = y(t_0) &= 0 \\ v(t_f) &= 0 \\ y(t_f) &= z\end{aligned}$$

Also, the terminal time t_f is given and z is the altitude of the target orbit.

1. Show that the optimal control law takes the form of

$$\tan\beta = \tan\beta_0 - ct$$

   where β_0 and c are constant. This law is referred to as the *linear tangent law*.

2. What happens if there exists the gravity described as

$$\dot{v} = a\sin\theta - G$$

   with g being constant?

3. Use the numerical methods discussed in class to find the optimal control inputs and resulting path when $G = 0$, $z = 5000$ m, $a = 2$ m/s^2, and $t_f = 200$ sec. Compare the numerical and analytical solutions.

Solution 1.

As in the first problem, this problem is described by the following equations

$$\begin{aligned}\dot{X} &= f(X, u) \\ J &= h + \int g dt\end{aligned}$$

where the state vector $X = [\ x_1\ x_2\ x_3\ x_4\ ]^\top$ where $x_1 = u$, $x_2 = v$, $x_3 = x$, and $x_4 = y$, and the input is $\beta = u$. In our problem the function h for the terminal cost is to maximize the terminal velocity

$$\boxed{h = -x_1(t_f)}$$

The dynamics are given by

$$\begin{aligned}\dot{x}_1 &= f_1\\ \dot{x}_2 &= f_2\\ \dot{x}_3 &= f_3\\ \dot{x}_4 &= f_4\end{aligned}$$

where a is a constant, the function f is written in component form as

$$\boxed{\begin{aligned}f_1 &= a\cos u\\ f_2 &= a\sin u\\ f_3 &= x_1\\ f_4 &= x_2\end{aligned}}$$

From the cost function we have

$$\boxed{g = 0}$$

Forming the Hamiltonian

$$\begin{aligned}H &= g + p^\top f\\ &= g + p_1 f_1 + p_2 f_2 + p_3 f_3 + p_4 f_4\\ &= p_1(a\cos u) + p_2(a\sin u) + p_3 x_1 + p_4 x_2\end{aligned}$$

Taking derivatives

$$\begin{aligned}H_x &= \begin{bmatrix} p_3 & p_4 & 0 & 0\end{bmatrix}\\ H_u &= -p_1 a\sin u + p_2 a\cos u\end{aligned}$$

Now we have the necessary conditions that $\dot{p} = -H_x^\top$ and $H_u = 0$. This gives

$$\begin{aligned}\dot{p}_1 &= -p_3\\ \dot{p}_2 &= -p_4\\ \dot{p}_3 &= 0\\ \dot{p}_4 &= 0\end{aligned}$$

and

$$p_1 a \sin u = p_2 a \cos u$$

Putting everything together, the differential equations (state and costate) which describe the system when $g = 0$ are

$$\boxed{\begin{aligned}\dot{x}_1 &= a\cos u\\ \dot{x}_2 &= a\sin u\\ \dot{x}_3 &= x_1\\ \dot{x}_4 &= x_2\\ \dot{p}_1 &= -p_3\\ \dot{p}_2 &= -p_4\\ \dot{p}_3 &= 0\\ \dot{p}_4 &= 0\end{aligned}}$$

We have the terminal constraint

$$\boxed{m = x_4(t_f) - z}$$

and we augment the terminal cost with terminal constraint as

$$w = h + \nu m$$

and so

$$\boxed{w = -x_1(t_f) + \nu(x_4(t_f) - z)}$$

And to find the boundary conditions we are given initial conditions on the state, and final conditions on two of the states. So we need to use final conditions on the costate equations which come from the following, since t_f is fixed

$$p(t_f) = \left(\frac{\partial w}{\partial x}\bigg|_{t_f}\right)^{\top}$$

So this with the above that $w = 0$ we get the final costate is zero. That is, the summary of all the boundary conditions is

$$\boxed{\begin{aligned} x_1(0) &= 0 \\ x_2(0) &= 0 \\ x_3(0) &= 0 \\ x_4(0) &= 0 \\ x_2(t_f) &= 0 \\ x_4(t_f) &= z \\ p_1(t_f) &= -1 \\ p_3(t_f) &= 0 \end{aligned}}$$

From the costate equations we know that p_3 and p_4 are constants. However, from the terminal boundary condition, we have that $p_3(t_f) = 0$ which means p_3 must be zero.

$$\begin{aligned} \dot{p}_1 &= 0 \\ \dot{p}_2 &= -k \end{aligned}$$

So this gives that p_1 is constant and p_2 is a linear function of time. That is

$$\begin{aligned} p_1 &= c_1 \\ p_2 &= b_2 t + c_2 \end{aligned}$$

From above we have

$$\tan\beta = \frac{p_2}{p_1}$$

and now we substitute in our known expressions for p_1 and p_2 to get the desired result. In particular, since p_1 is a constant, from the final condition we have $p_1 = -1$ giving

$$\tan\beta = -b_2 t - c_2$$

And note to solve for $u = \beta$ from $\tan\beta = \frac{p_2}{p_1}$ since $p_1 = -1$ when we take the inverse tangent, we need to consider 4-quadrant inverse tangent.

Solution 2.

If the gravity term is included the equations are modified as

$$\boxed{h = -x_1(t_f)}$$

$$\boxed{\begin{aligned} f_1 &= a\cos u \\ f_2 &= a\sin u - G \\ f_3 &= x_1 \\ f_4 &= x_2 \end{aligned}}$$

$$\boxed{g = 0}$$

Forming the Hamiltonian

$$H = p_1(a\cos u) + p_2(a\sin u - G) + p_3x_1 + p_4x_2$$

Taking derivatives

$$\begin{aligned} H_x &= \begin{bmatrix} p_3 & p_4 & 0 & 0 \end{bmatrix} \\ H_u &= -p_1 a\sin u + p_2 a\cos u \end{aligned}$$

Applying the necessary conditions we can see that the form of the solution is still the same. However, the presence of the gravity term G will change the numerical values of the solution form. We had from before $p_3 = 0$, $p_1 = -1$, and p_4 is constant, and p_2 depends on p_4. So the numerical value of p_4 due to gravity is different, and so the coefficients describing the linear function p_2 are different, resulting in different control history. The form will be the same though.

Solution 3.

The equations and boundary conditions of part 1 were implemented in MATLAB and simulated. The following plots show the trajectory of the spacecraft in x and y, as well as the control history β which is linear, as expected.

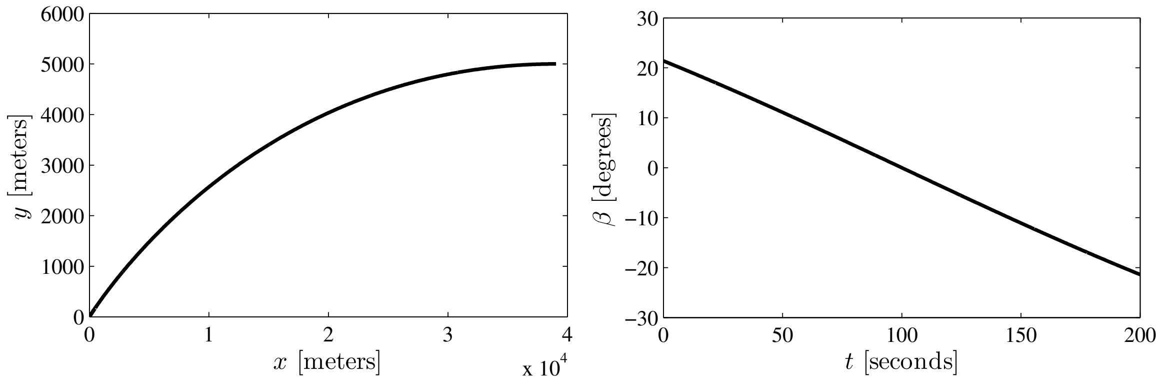

Figure 2: State and control history plots for the pendulum system

Code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Dan Wiese
% 16.323 - HW4
% Problem_1.m
%-------------------------------------------------------------------------------
% This is the master script which runs BVP4C to solve the state and costate
% equations and determine the optimal control for a pendulum.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all;
close all;
clc;

global rho
rho=10;

options = bvpset('Stats','on','RelTol',1e-1);
solinit=bvpinit(linspace(0,10),@p1_initfun);

sol1=bvp4c(@p1_odefun,@p1_bcfun,solinit);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

sol=struct;
sol.t=sol1.x;
sol.x=sol1.y(1:2,:);
sol.p=sol1.y(3:4,:);
sol.u=-sol.p(2,:)/(2*rho);

%-------------------------------------------------------------------------------
%% Plot Results

sizefont=14;
ticklabelfontsize=12;

figure(1)
plot(sol.t,sol.u,'color',[0 0 0],'linewidth',2)
set(gcf,'Units','pixels');
set(gcf,'PaperUnits','inches','PaperPosition',[0 0 5 3]);
set(gcf,'PaperPositionMode','manual')
set(gcf,'InvertHardCopy','off');
set(gcf,'color',[1 1 1])
xlabel('time [s]','interpreter','latex','FontSize',sizefont)
ylabel('$u^{*}(t)$','interpreter','latex','FontSize',sizefont)
set(gca,'fontsize',ticklabelfontsize);
set(gca,'fontname','times');
print('-depsc','../Figures/prob1_control.eps');

figure(2)
plot(sol.t,sol.x(1,:),'color',[0 0 0],'linewidth',2)
hold on
plot(sol.t,sol.x(2,:),'color',[0 0 0],'linewidth',2,'linestyle','--')
legend('$\theta$','$\dot{\theta}$','Location','NorthEast')
temphandle=legend;
```

```
set(temphandle,'interpreter','latex','FontSize',sizefont,'box','on')
set(gcf,'Units','pixels');
set(gcf,'PaperUnits','inches','PaperPosition',[0 0 5 3]);
set(gcf,'PaperPositionMode','manual')
set(gcf,'InvertHardCopy','off');
set(gcf,'color',[1 1 1])
xlabel('time [s]','interpreter','latex','FontSize',sizefont)
ylabel('$\theta$, $\dot{\theta}$','interpreter','latex','FontSize',sizefont)
set(gca,'fontsize',ticklabelfontsize);
set(gca,'fontname','times');
print('-depsc','../Figures/prob1_state.eps');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Dan Wiese
% 16.323 - HW#1
% p1_initfun.m
%-------------------------------------------------------------------------------
% This is the initial guess for the BVP4C solver for problem 1: pendulum.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function y=p1_initfun(t)

y=[1;2;3;4];

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Dan Wiese
% 16.323 - HW#1
% p1_bcfun.m
%-------------------------------------------------------------------------------
% This is the boundary conditions for BVP4C solver for pendulum problem.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function res=p1_bcfun(Y0,Yf)

%Initial Conditions
x10=Y0(1);
x20=Y0(2);
p10=Y0(3);
p20=Y0(4);

%Final Conditions
x1f=Yf(1);
x2f=Yf(2);
p1f=Yf(3);
p2f=Yf(4);

%Impose the necessary initial and final conditions
res = [ x10 - pi
x20 - 0
p1f - 0
p2f - 0 ];

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Dan Wiese
% 16.323 - HW#1
% p1_odefun.m
%-------------------------------------------------------------------------------
% These are the differential equations for state and costate for pendulum.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function Ydot=p1_odefun(t,Y)

global rho

x1=Y(1);
x2=Y(2);
p1=Y(3);
p2=Y(4);

xdot1=x2;
xdot2=sin(x1)-p2/(2*rho);
pdot1=-2*x1-p2*cos(x1);
pdot2=-p1;

Ydot=[xdot1, xdot2, pdot1, pdot2]';

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Dan Wiese
% 16.323 - HW4
% Problem_1.m
%-------------------------------------------------------------------------------
% This is the master script which runs BVP4C to solve the state and costate
% equations and determine the optimal control for a pendulum.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all;
close all;
clc;

global a g h
a=2;
g=0;
h=5000;

options = bvpset('Stats','on','RelTol',1e-1);
solinit=bvpinit(linspace(0,200),@p3_initfun);

sol1=bvp4c(@p3_odefun,@p3_bcfun,solinit);

```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
sol=struct;
sol.t=sol1.x;
sol.x=sol1.y(1:4,:);
sol.p=sol1.y(5:8,:);
sol.u=atan2(-sol.p(2,:),-sol.p(1,:));

%--------------------------------------------------------------------------------
%% Plot Results

sizefont=14;
ticklabelfontsize=12;

figure(1)
plot(sol.x(3,:),sol.x(4,:),'color',[0 0 0],'linewidth',2,'linestyle','-')
set(gcf,'Units','pixels');
set(gcf,'PaperUnits','inches','PaperPosition',[0 0 5 3]);
set(gcf,'PaperPositionMode','manual')
set(gcf,'InvertHardCopy','off');
set(gcf,'color',[1 1 1])
ylim([0 6000]);
xlabel('$x$ [meters]','interpreter','latex','FontSize',sizefont)
ylabel('$y$ [meters]','interpreter','latex','FontSize',sizefont)
set(gca,'fontsize',ticklabelfontsize);
set(gca,'fontname','times');
print('-depsc','../Figures/prob3_traj.eps');

figure(2)
plot(sol.t,sol.u*180/pi,'color',[0 0 0],'linewidth',2,'linestyle','-')
set(gcf,'Units','pixels');
set(gcf,'PaperUnits','inches','PaperPosition',[0 0 5 3]);
set(gcf,'PaperPositionMode','manual')
set(gcf,'InvertHardCopy','off');
set(gcf,'color',[1 1 1])
xlabel('$t$ [seconds]','interpreter','latex','FontSize',sizefont)
ylabel('$\beta$ [degrees]','interpreter','latex','FontSize',sizefont)
set(gca,'fontsize',ticklabelfontsize);
set(gca,'fontname','times');
print('-depsc','../Figures/prob3_control.eps');

% figure(3)
% plot(sol.t,sol.x(1,:),'color',[0 0 0],'linewidth',2)
% hold on
% plot(sol.t,sol.x(2,:),'color',[0 0 0],'linewidth',2,'linestyle',':')
% plot(sol.t,sol.x(3,:),'color',[0.6 0.6 0.6],'linewidth',2,'linestyle','-')
% plot(sol.t,sol.x(4,:),'color',[0 0 0],'linewidth',2,'linestyle','--')
% legend('$u$','$v$','$x$','$y$','Location','NorthEast')
% temphandle=legend;
% set(temphandle,'interpreter','latex','FontSize',sizefont,'box','on')
% set(gcf,'Units','pixels');
% set(gcf,'PaperUnits','inches','PaperPosition',[0 0 5 3]);
% set(gcf,'PaperPositionMode','manual')
% set(gcf,'InvertHardCopy','off');
% set(gcf,'color',[1 1 1])
% xlabel('time [s]','interpreter','latex','FontSize',sizefont)
% ylabel('$\theta$, $\dot{\theta}$','interpreter','latex','FontSize',sizefont)
% set(gca,'fontsize',ticklabelfontsize);
% set(gca,'fontname','times');
% % print('-depsc','../Figures/prob3_state.eps');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Dan Wiese
% 16.323 - HW#1
% p1_initfun.m
%-------------------------------------------------------------------------------
% This is the initial guess for the BVP4C solver for problem 3: space launch
% vehicle
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function y=p3_initfun(t)

y=[1;2;3;4;5;6;7;8];

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Dan Wiese
% 16.323 - HW#1
% p1_bcfun.m
%-------------------------------------------------------------------------------
% This is the boundary conditions for BVP4C solver for problem 3: space launch
% vehicle.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function res=p3_bcfun(Y0,Yf)

global h

%Initial Conditions
x10=Y0(1);
x20=Y0(2);
x30=Y0(3);
x40=Y0(4);
p10=Y0(5);
p20=Y0(6);
p30=Y0(7);
p40=Y0(8);

%Final Conditions
x1f=Yf(1);
x2f=Yf(2);
x3f=Yf(3);
x4f=Yf(4);
p1f=Yf(5);
p2f=Yf(6);
p3f=Yf(7);
p4f=Yf(8);
```

```

%Impose the necessary initial and final conditions
res = [ x10 - 0
x20 - 0
x30 - 0
x40 - 0
x2f - 0
x4f - h
p1f + 1
p3f - 0];

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Dan Wiese
% 16.323 - HW#1
% p1_odefun.m
%-------------------------------------------------------------------------------
% These are the differential equations for state and costate for problem 3:
% space launch vehicle.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function Ydot=p3_odefun(t,Y)

global a g

%State input
x1=Y(1);
x2=Y(2);
x3=Y(3);
x4=Y(4);

%Costate input
p1=Y(5);
p2=Y(6);
p3=Y(7);
p4=Y(8);

beta=atan2(-p2,-p1);

%State equations
xdot1=a*cos(beta);
xdot2=a*sin(beta)-g;
xdot3=x1;
xdot4=x2;

%Costate equations
pdot1=-p3;
pdot2=-p4;
pdot3=0;
pdot4=0;

Ydot=[xdot1, xdot2, xdot3, xdot4, pdot1, pdot2, pdot3, pdot4]';

```

```
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```