

**ECE-597: Optimal Control**  
Homework #6

Due: October 23, 2007

You will probably need to read the Appendix before trying these problems.

1. We will solve the same problem here in both continuous-time and discrete-time form.

a) Assume the continuous time equations of motion

$$\begin{aligned}\dot{u}(t) &= a \cos(\theta(t)) \\ \dot{v}(t) &= a \sin(\theta(t)) - g \\ \dot{y}(t) &= v(t) \\ \dot{x}(t) &= u(t)\end{aligned}$$

We want to find  $\theta(t)$  to maximize  $u(t_f)$  and we want  $y(t_f) = y_f$ ,  $x(t_f) = x_f$  and  $v(t_f) = 0$ , where  $a = 1$ ,  $g = 1/3$ ,  $y_f = 0.2$ ,  $x_f = 0.15$  and  $t_f = 1.0$ . Solve the problem using **fopc.m** and include your code and final plot.

b) Assume the discrete-time equations of motion

$$\begin{aligned}u(i+1) &= u(i) + a\Delta T \cos(\theta(i)) \\ v(i+1) &= v(i) + \Delta T [a \sin(\theta(i)) - g] \\ y(i+1) &= y(i) + \Delta T v(i) + \frac{1}{2} \Delta T^2 [a \sin(\theta(i)) - g] \\ x(i+1) &= x(i) + \Delta T u(i) + \frac{1}{2} \Delta T^2 a \cos(\theta(i))\end{aligned}$$

We want to find  $\theta(i)$  to maximize  $u(N)$  and we want  $y(N) = y_f$ ,  $x(N) = x_f$  and  $v(N) = 0$ , where  $a = 1$ ,  $g = 1/3$ ,  $y_f = 0.2$ ,  $x_f = 0.15$ ,  $t_f = 1.0$ , and  $\Delta T = t_f/N$ . Solve the problem using **dopc.m** and include your code and final plot.

2. We will solve the same problem here in both continuous-time and discrete-time form.

a) Assume the continuous time equations of motion

$$\begin{aligned}\dot{v}(t) &= a - g \sin(\gamma(t)) \\ \dot{y}(t) &= v(t) \sin(\gamma(t)) \\ \dot{x}(t) &= v(t) \cos(\gamma(t))\end{aligned}$$

We want to find  $\gamma(t)$  to maximize  $x(t_f)$  and we want  $y(t_f) = y_f$ , where  $a = 0.5$ ,  $g = 1$ ,  $y_f = 0.1$ , and  $t_f = 1.0$ . Solve the problem using **fopc.m** and include your code and final plot.

b) Assume the discrete-time equations of motion

$$\begin{aligned}v(i+1) &= v(i) + \Delta T [\tilde{a} - \sin(\gamma(i))] \\y(i+1) &= y(i) + \Delta l(i) \sin(\gamma(i)) \\x(i+1) &= x(i) + \Delta l(i) \cos(\gamma(i)) \\ \Delta l(i) &= v(i)\Delta T + \frac{1}{2} [\tilde{a} - \sin(\gamma(i))] \Delta T^2\end{aligned}$$

We want to find  $\theta(i)$  to maximize  $x(N)$  and we want  $y(N) = y_f$ , where  $\tilde{a} = 0.5$ ,  $y_f = 0.1$ ,  $t_f = 1.0$ , and  $\Delta T = t_f/N$ . Solve the problem using **dopc.m** and include your code and final plot.

**3.** Assume we have a model for heating a room

$$\dot{\theta}(t) = -a(\theta(t) - \theta_a) + bu(t)$$

where  $\theta(t)$  is the room temperature at time  $t$ ,  $\theta_a$  is the ambient room temperature,  $a$  and  $b$  are constants, and  $u(t)$  is the rate of heat supplied to the room. Defining the state variable

$$x(t) = \theta(t) - \theta_a$$

we have the state equations

$$\dot{x}(t) = -ax(t) + bu(t)$$

Suppose we want to heat the room to a desired final temperature using the minimum energy. We would have

$$J = \frac{1}{2} \int_0^T u^2(t) dt$$

subject to

$$\begin{aligned}\dot{x}(t) &= -ax(t) + bu(t) \\x(0) &= \theta(0) - \theta_a \\x_f &= \theta_f - \theta_a\end{aligned}$$

where  $T$  is the given final time.

a) Show that

$$\lambda_x(t) = \nu e^{-a(T-t)}$$

b) Show that

$$u(t) = -b\lambda_x(t)$$

and that

$$\frac{d}{dt} (x(t)e^{at}) = -b^2 \lambda_x(t) e^{at}$$

c) Defining

$$\gamma = -b^2 \nu e^{-aT}$$

show that

$$x(t) = x(0)e^{-at} + \frac{\gamma}{a} \sinh(at)$$

d) Equating  $x(T)$  with  $x_f$ , show that we get

$$\nu = \frac{2a}{b^2(1 - e^{-2aT})} [x_0 e^{-aT} - x_f]$$

e) Using the Matlab function **fopc.m** to simulate this system for the case  $a = 10, b = 2, x_0 = 10, T = 2, x_f = 20$  Try small values of  $k$  (near  $10^{-6}$ ), set both tolerances to about  $5 \times 10^{-3}$ , and a maximum of 25 simulations. Compare the true and estimated inputs and states.

f) As the results show, this may not be a very good way to try and heat up a room. Devise a different performance index that still penalizes the energy, but does not produce the rapid drop in temperature we got in part (e). Rerun your simulation for this performance index. (You do not need an analytical solution!) Try to be creative and don't just copy something from a friend.

4. From *Bryson*. This is a **Zermelo** problem, we have seen before. Assume a ship travels with constant velocity  $V$  with respect to the water through a region of water where the velocity of the current is parallel to the  $x$ -axis, but varies with  $y$ , so that

$$\begin{aligned} \dot{x}(t) &= V \cos(\theta(t)) + \frac{Vy(t)}{h} \\ \dot{y}(t) &= V \sin(\theta(t)) \end{aligned}$$

The discretized equations of motion are

$$\begin{aligned} y(i+1) &= y(i) + \Delta T \sin(\theta(i)) \\ x(i+1) &= x(i) + \Delta T [\cos(\theta(i)) + y(i)] + \frac{1}{2} (\Delta T)^2 \sin(\theta(i)) \end{aligned}$$

where  $\Delta T$  is in units of  $h/V$  and  $(x, y)$  are in units of  $h$ . We now want to find the sequence  $\theta(i)$  to maximize  $x(t_f)$  and include the terminal constraint  $y(t_f) = y_f$ .

a) Show that

$$\begin{aligned} \lambda_y(N) &= \Phi_y = \nu \\ \lambda_x(i) &= 1 \\ \lambda_y(i) &= \nu + (N - i)\Delta T \end{aligned}$$

b) Show that the optimal direction is given by

$$\tan[\theta(i)] = \tan[\theta(0)] - i\Delta T$$

Note that we still have not solved the problem. We have the  $N + 1$  unknowns  $\theta(0), \dots, \theta(N)$  which must be chosen to be sure we match the boundary conditions ( $x(0) = 0$  and  $y(N) = y_f$ )

5. From *Bryson*. This is a rendezvous problem. We have the equations of motion

$$\begin{aligned}\dot{u}(t) &= \cos(\theta(t)) \\ \dot{v}(t) &= \sin(\theta(t)) \\ \dot{y}(t) &= v(t) \\ \dot{x}(t) &= u(t)\end{aligned}$$

We want to find  $\theta(t)$  (the thrust direction) to maximize  $u(1)$  subject to  $v(1) = 0$ ,  $x(1) = x_f$  and  $y(1) = y_f$ . Here  $t$  is measured in units of  $t_f$ , so  $t = 1$  at time  $t_f$ . Also  $(u, v)$  are measured in units of  $at_f$  and  $(x, y)$  are measured in units of  $at_f^2$ . We assume all initial conditions are zero.

a) Show that the adjoint equations are

$$\begin{aligned}\dot{\lambda}_u &= -\lambda_x \\ \dot{\lambda}_v &= -\lambda_y \\ \dot{\lambda}_y &= 0 \\ \dot{\lambda}_x &= 0\end{aligned}$$

b) Show that the optimality condition is

$$0 = H_\theta = -\lambda_u \sin \theta + \lambda_v \cos \theta$$

c) Show that the terminal conditions are

$$\begin{aligned}\lambda_u(1) &= 1 \\ \lambda_v(1) &= \nu_v \\ \lambda_y(1) &= \nu_y \\ \lambda_x(1) &= \nu_x\end{aligned}$$

d) Show that the adjoint equations are

$$\begin{aligned}\lambda_y(t) &= \nu_y \\ \lambda_x(t) &= \nu_x \\ \lambda_u(t) &= 1 + \nu_x(1 - t) \\ \lambda_v(t) &= \nu_v + \nu_y(1 - t)\end{aligned}$$

e) Show that we can combine these to write

$$\tan \theta(t) = \frac{\nu_v + \nu_y(1 - t)}{1 + \nu_x(1 - t)}$$

where the constants  $\nu_v, \nu_y, \nu_x$  must be determined to satisfy  $v(1) = 0, y(1) = y_f, x(1) = x_f$ .  
f) Consider the trigonometric identity

$$\tan(\theta - \alpha) \equiv \frac{\tan \theta - \tan \alpha}{1 + \tan \theta \tan \alpha}$$

Substituting this into our equation from part e gives

$$\tan(\theta - \alpha) \equiv \frac{\nu_v - \tan \alpha + (\nu_y - \nu_x \tan \alpha)(1 - t)}{1 + \nu_v \tan \alpha + (\nu_x + \nu_y \tan \alpha)(1 - t)}$$

If we choose  $\tan \alpha = -\nu_x/\nu_y$ , then this becomes of the form

$$\tan(\theta - \alpha) = n - mt \Rightarrow n = \tan(\theta_0 - \alpha)$$

where the constants  $\theta_0, \alpha, m$  have replaced the constants  $\nu_v, \nu_y, \nu_x$ .

# Appendix

*Continuous-Time Problems with Hard Terminal Constraints* using **fopc.m**

We will be utilizing the routine **fopc.m** to solve continuous-time optimization problems of the form:

Find the input  $u(t)$ ,  $t_0 \leq t \leq t_f$  to minimize

$$J = \phi[x(t_f)] + \int_{t_0}^{t_f} L[x(t), u(t), t] dt$$

subject to the constraints

$$\begin{aligned}\dot{x}(t) &= f[x(t), u(t), t] \\ \psi[x(t_f)] &= 0 \\ x(0) &= x_0 \text{ (known)}\end{aligned}$$

Note that we are now adding **hard** terminal constraint, i.e., we can force  $x(t_f)$  to end at certain values.

In the *Mayer* formulation, the state vector is augmented by one state  $q(t)$  that is the integral of  $L$  from  $t_0$  to  $t$ :

$$q(t) = \int_{t_0}^t L[x(t), u(t), t] dt$$

The *Bolza* performance index

$$J = \phi[x(t_f)] + \int_{t_0}^{t_f} L[x(t), u(t), t] dt$$

becomes the *Mayer* performance index

$$J = \phi[x(t_f)] + q(t_f) = \bar{\phi}[\bar{x}(t_f)]$$

where

$$\bar{x} = \begin{bmatrix} x \\ q \end{bmatrix}$$

and  $\psi[x(t_f)]$  is another constraint (like  $\dot{x}(t) = f(x(t), u(t), t)$ ). We usually drop the bar on  $x$  and  $\phi$  and the problem is stated as finding a sequence  $u(t)$ ,  $t_0 \leq t \leq t_f$  to minimize (or maximize)

$$J = \phi[x(t_f)]$$

subject to

$$\begin{aligned}\dot{x}(t) &= f[x(t), u(t), t] \\ \psi[x(t_f)] &= 0\end{aligned}$$

and  $t_0$ ,  $x(t_0)$  and  $t_f$  are specified.

In order to use the routine **fopc.m**, you need to write a routine that returns one of three things depending on the value of the variable **flg**. The general form of your routine will be as follows:

```
function [f1,f2] = bobs_fopc(u,s,t,flg)
```

Here  $u$  is the current input,  $u(t)$ , and  $s$  ( $s(t)$ ) contains the current state (including the augmented state), so  $\dot{s}(t) = f(s(t), u(t), t)$ . Your routine should compute the following:

$$\begin{aligned} \text{if } \mathbf{flg} = 1 & \quad f1 = \dot{s}(t) = f(s(t), u(t), t) \\ \text{if } \mathbf{flg} = 2 & \quad f1 = \begin{bmatrix} \phi[s(t_f)] \\ \psi[s(t_f)] \end{bmatrix}, f2 = \begin{bmatrix} \phi_{s(t_f)}[s(t_f)] \\ \psi_{s(t_f)}[s(t_f)] \end{bmatrix} \\ \text{if } \mathbf{flg} = 3 & \quad f1 = f_s, f2 = f_u \end{aligned}$$

An example of the usage is:

```
[tu,ts,nu,la0] = fopc('bobs_fopc',tu,tf,s0,k,told,tols,mxit,eta)
```

The (input) arguments to **fopc.m** are the following:

- the function you just created (in single quotes).
- $tu$  is an initial guess of times (first column), and control values (subsequent columns) that minimizes  $J$ . If there are multiple control signals at a given time, they are all in the same row. Note that these are just the initial time and control values, the times and control values will be modified as the program runs. The initial time should start at zero.
- the initial states,  $s0$ . Note that you must include an initial guess for the “cumulative” state  $q$  also.
- the final time,  $tf$ .
- $k$ , the step size parameter,  $k > 0$  to minimize. Often you need to play around with this one.
- $told$ , the tolerance (a stopping parameter) for ode23 (differential equation solver for Matlab)
- $tols$ , the tolerance (a stopping parameter), when  $|\Delta u| < tols$  between iterations, the program stops.
- $mxit$ , the maximum number of iterations to try.
- $eta$ , where  $0 < eta \leq 1$ , and  $d(\psi) = -eta*\psi$  is the desired change in the terminal constraints on the next iteration.

**fopc.m** returns the following:

- *tu* the optimal input sequence and corresponding times. The first column is the time, the corresponding columns are the control signals. All entries in one row correspond to one time.
- *ts* the states and corresponding times. The first column is the time, the corresponding columns are the states. All entries in one row correspond to the same time. Note that the times in *tu* and the times in *ts* may not be the same, and they may not be evenly spaced.
- *nu*, the Lagrange multipliers on psi
- *la0* the Lagrange multipliers

It is usually best to start with a small number of iterations, like 5, and see what happens as you change *k*. Start with small values of *k* and gradually increase them. It can be very difficult to make this program converge, especially if your initial guess is far away from the true solution.

**Note!!** If you are using the **fopc.m** file, and you use the maximum number of allowed iterations, assume that the solution has NOT converged. You must usually change the value of *k* and/or increase the number of allowed iterations. Do not set *tol* to less than about 5e-5. Also try to make *k* as large as possible and still have convergence.

**Example A** Assume the continuous-time equations of motion

$$\begin{aligned}\dot{u}(t) &= a \cos(\theta(t)) \\ \dot{v}(t) &= a \sin(\theta(t)) \\ \dot{y}(t) &= v(t) \\ \dot{x}(t) &= u(t)\end{aligned}$$

We want to find  $\theta(t)$  to maximize  $u(t_f)$  and we want  $v(t_f) = 0$ , and  $y(t_f) = y_f$ , where  $a = 1$ ,  $y_f = 0.2$ , and  $t_f = 1.0$ .

Let's define the state vector as

$$s = \begin{bmatrix} u \\ v \\ y \\ x \end{bmatrix}$$

and let's define

$$\Phi[s(t_f)] = \begin{bmatrix} \phi[s(t_f)] \\ \psi[s(t_f)] \end{bmatrix}$$



For this problem we want to **maximize**  $u(t_f)$ , so we have

$$\phi[s(t_f)] = u(t_f), \quad L = 0$$

Hence  $q(t) = 0$  for all  $t$  and we do not need to augment the state.

We also have the hard terminal constraints

$$\psi[s(t_f)] = \begin{bmatrix} v(t_f) \\ y(t_f) - y_f \end{bmatrix}$$

so we can write

$$\Phi[s(t_f)] = \begin{bmatrix} u(t_f) \\ v(t_f) \\ y(t_f) - y_f \end{bmatrix}$$

We can then write

$$f = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} = \begin{bmatrix} a \cos(\theta(t)) \\ a \sin(\theta(t)) \\ v(t) \\ u(t) \end{bmatrix}$$

Next we need

$$\begin{aligned} \Phi_{s(t_f)}[s(t_f)] &= \begin{bmatrix} \phi_{s(t_f)}[s(t_f)] \\ \psi_{s(t_f)}[s(t_f)] \end{bmatrix} \\ &= \begin{bmatrix} \frac{\partial \phi[s(t_f)]}{\partial u(t_f)} & \frac{\partial \phi[s(t_f)]}{\partial v(t_f)} & \frac{\partial \phi[s(t_f)]}{\partial y(t_f)} & \frac{\partial \phi[s(t_f)]}{\partial x(t_f)} \\ \frac{\partial \psi[s(t_f)]}{\partial u(t_f)} & \frac{\partial \psi[s(t_f)]}{\partial v(t_f)} & \frac{\partial \psi[s(t_f)]}{\partial y(t_f)} & \frac{\partial \psi[s(t_f)]}{\partial x(t_f)} \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \end{aligned}$$

Then we need

$$\begin{aligned} f_s &= \begin{bmatrix} \frac{\partial f}{\partial u(t)} & \frac{\partial f}{\partial v(t)} & \frac{\partial f}{\partial y(t)} & \frac{\partial f}{\partial x(t)} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial u(t)} & \frac{\partial f_1}{\partial v(t)} & \frac{\partial f_1}{\partial y(t)} & \frac{\partial f_1}{\partial x(t)} \\ \frac{\partial f_2}{\partial u(t)} & \frac{\partial f_2}{\partial v(t)} & \frac{\partial f_2}{\partial y(t)} & \frac{\partial f_2}{\partial x(t)} \\ \frac{\partial f_3}{\partial u(t)} & \frac{\partial f_3}{\partial v(t)} & \frac{\partial f_3}{\partial y(t)} & \frac{\partial f_3}{\partial x(t)} \\ \frac{\partial f_4}{\partial u(t)} & \frac{\partial f_4}{\partial v(t)} & \frac{\partial f_4}{\partial y(t)} & \frac{\partial f_4}{\partial x(t)} \end{bmatrix} \\ &= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \end{aligned}$$

and finally

$$f_u = f_\theta = \begin{bmatrix} \frac{\partial f_1}{\partial \theta(t)} \\ \frac{\partial f_2}{\partial \theta(t)} \\ \frac{\partial f_3}{\partial \theta(t)} \\ \frac{\partial f_4}{\partial \theta(t)} \end{bmatrix} = \begin{bmatrix} -a \sin(\theta(t)) \\ a \cos(\theta(t)) \\ 0 \\ 0 \end{bmatrix}$$

This is implemented in the routine **bobs\_fopc\_a.m** on the class web site, and it is run using the driver file **fopc\_example\_a.m**.

*Discrete-Time Problems with Hard Terminal Constraints* using **dopc.m**

We will be utilizing the routine **dopc.m** to solve discrete-time optimization problems of the form:

Find the input sequence  $u(i)$ ,  $i = 0, \dots, N - 1$  to minimize

$$J = \phi[x(N)] + \sum_{i=0}^{N-1} L[x(i), u(i), i]$$

subject to the constraints

$$\begin{aligned} x(i+1) &= f[x(i), u(i), i] \\ \psi[x(N)] &= 0 \\ x(0) &= x_0 \text{ (known)} \end{aligned}$$

Note that we are now adding **hard** terminal constraint, i.e., we can force  $x(N)$  to end at certain values.

In the *Mayer* formulation, the state vector is augmented by one state  $q(i)$  that is the cumulative sum of  $L$  up to step  $i$ :

$$q(i+1) = q(i) + L[x(i), u(i), i]$$

The *Bolza* performance index

$$J = \phi[x(N)] + \sum_{i=1}^{N-1} L[x(i), u(i), i]$$

becomes the *Mayer* performance index

$$J = \phi[x(N)] + q(N) = \bar{\phi}[\bar{x}(N)]$$

where

$$\bar{x} = \begin{bmatrix} x \\ q \end{bmatrix}$$

and  $\psi[x(N)]$  is another constraint (like  $x(i+1) = f(x(i), u(i), i)$ ). We usually drop the bar on  $x$  and  $\phi$  and the problem is stated as finding a sequence  $u(i)$ ,  $i = 0, \dots, N-1$  to minimize (or maximize)

$$J = \phi[x(N)]$$

subject to

$$\begin{aligned} x(i+1) &= f[x(i), u(i), i] \\ \psi[x(N)] &= 0 \end{aligned}$$

and  $x(0)$  and  $N$  are specified.

In order to use the routine **dopc.m**, you need to write a routine that returns one of three things depending on the value of the variable **flg**. The general form of your routine will be as follows:

```
function [f1,f2] = bobs_dopc(u,s,dt,t,flg)
```

Here  $u$  is the current input,  $u(i)$ , and  $s$  ( $s(i)$ ) contains the current state (including the augmented state) so  $s(i+1) = f(s(i), u(i), \Delta T)$ .  $dt$  is the time increment  $\Delta T$ , and  $t$  is the current time. You can compute the current index  $i$  using the relationship  $i = \frac{t}{\Delta T} + 1$ . Your routine should compute the following:

$$\begin{aligned} \text{if } \mathbf{flg} = 1 & \quad f1 = s(i+1) = f(s(i), u(i), \Delta T) \\ \text{if } \mathbf{flg} = 2 & \quad f1 = \begin{bmatrix} \phi[s(N)] \\ \psi[s(N)] \end{bmatrix}, f2 = \begin{bmatrix} \phi_{s(N)}[s(N)] \\ \psi_{s(N)}[s(N)] \end{bmatrix} \\ \text{if } \mathbf{flg} = 3 & \quad f1 = f_s, f2 = f_u \end{aligned}$$

An example of the usage is:

```
[u,s,nu,la0] = dopc('bobs_dopc',u,s0,tf,k,tol,mxit,eta)
```

The (input) arguments to **dop0.m** are the following:

- the function you just created (in single quotes).
- the initial guess for the value of  $u$  that minimizes  $J$ . This initial size is used to determine the time step: if there are  $n$  components in  $u$ , then  $\Delta T = \frac{t_f}{n}$
- an initial guess of the states,  $s0$ . Note that you must include an initial guess for the “cumulative” state  $q$  also.
- the final time,  $tf$ .
- $k$ , the step size parameter,  $k > 0$  to minimize. Often you need to play around with this one.

- *tol*, the tolerance (a stopping parameter), when  $|\Delta u| < tol$  between iterations, the program stops.
- *mxit*, the maximum number of iterations to try.
- *eta*, where  $0 < eta \leq 1$ , and  $d(\psi) = -eta*\psi$  is the desired change in the terminal constraints on the next iteration.

**dopc.m** returns the following:

- *u* the optimal input sequence
- *s* the corresponding states
- *nu*, the Lagrange multipliers on  $\psi$
- *la0* the Lagrange multipliers

It is usually best to start with a small number of iterations, like 5, and see what happens as you change  $k$ . Start with small values of  $k$  and gradually increase them. It can be very difficult to make this program converge, especially if your initial guess is far away from the true solution.

**Note!!** If you are using the **dopc.m** file, and you use the maximum number of allowed iterations, assume that the solution has NOT converged. You must usually change the value of  $k$  and/or increase the number of allowed iterations. Do not set  $tol$  to less than about  $5e-5$ . Also try to make  $k$  as large as possible and still have convergence.

**Example B** Assume the discrete-time equations of motion (these are the discretized equations from **Example A**)

$$\begin{aligned}
 u(i+1) &= u(i) + a\Delta T \cos(\theta(i)) \\
 v(i+1) &= v(i) + a\Delta T \sin(\theta(i)) \\
 y(i+1) &= y(i) + \Delta T v(i) + \frac{a}{2}\Delta T^2 \sin(\theta(i)) \\
 x(i+1) &= x(i) + \Delta T u(i) + \frac{a}{2}\Delta T^2 \cos(\theta(i))
 \end{aligned}$$

We want to find  $\theta(i)$  to maximize  $u(N)$  and we want  $v(N) = 0$  and  $y(N) = y_f$ , where  $a = 1$ ,  $y_f = 0.2$ ,  $t_f = 1.0$ , and  $\Delta T = t_f/N$ .

Let's define the state vector as

$$s = \begin{bmatrix} u \\ v \\ y \\ x \end{bmatrix}$$

and let's define

$$\Phi[s(N)] = \begin{bmatrix} \phi[s(N)] \\ \psi[s(N)] \end{bmatrix}$$

For this problem we want to **maximize**  $u(N)$ , so we have

$$\phi[s(N)] = x(N), \quad L = 0$$

Hence  $q(i) = 0$  for all  $i$  and we do not need to augment the state.

We also have the hard terminal constraints

$$\psi[s(N)] = \begin{bmatrix} v(N) \\ y(N) - y_f \end{bmatrix}$$

so we can write

$$\Phi[s(N)] = \begin{bmatrix} u(N) \\ v(N) \\ y(N) - y_f \end{bmatrix}$$

We can then write

$$f = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} = \begin{bmatrix} u(i) + a\Delta T \cos(\theta(i)) \\ v(i) + \Delta T a \sin(\theta(i)) \\ y(i) + \Delta T v(i) + \frac{1}{2}\Delta T^2 a \sin(\theta(i)) \\ x(i) + \Delta T u(i) + \frac{1}{2}\Delta T^2 a \cos(\theta(i)) \end{bmatrix}$$

Next we need

$$\begin{aligned} \Phi_{s(N)}[s(N)] &= \begin{bmatrix} \phi_{s(N)}[s(N)] \\ \psi_{s(N)}[s(N)] \end{bmatrix} \\ &= \begin{bmatrix} \frac{\partial \phi[s(N)]}{\partial u(N)} & \frac{\partial \phi[s(N)]}{\partial v(N)} & \frac{\partial \phi[s(N)]}{\partial y(N)} & \frac{\partial \phi[s(N)]}{\partial x(N)} \\ \frac{\partial \psi[s(N)]}{\partial u(N)} & \frac{\partial \psi[s(N)]}{\partial v(N)} & \frac{\partial \psi[s(N)]}{\partial y(N)} & \frac{\partial \psi[s(N)]}{\partial x(N)} \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \end{aligned}$$

Then we need

$$\begin{aligned} f_s &= \begin{bmatrix} \frac{\partial f}{\partial u(i)} & \frac{\partial f}{\partial v(i)} & \frac{\partial f}{\partial y(i)} & \frac{\partial f}{\partial x(i)} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial u(i)} & \frac{\partial f_1}{\partial v(i)} & \frac{\partial f_1}{\partial y(i)} & \frac{\partial f_1}{\partial x(i)} \\ \frac{\partial f_2}{\partial u(i)} & \frac{\partial f_2}{\partial v(i)} & \frac{\partial f_2}{\partial y(i)} & \frac{\partial f_2}{\partial x(i)} \\ \frac{\partial f_3}{\partial u(i)} & \frac{\partial f_3}{\partial v(i)} & \frac{\partial f_3}{\partial y(i)} & \frac{\partial f_3}{\partial x(i)} \\ \frac{\partial f_4}{\partial u(i)} & \frac{\partial f_4}{\partial v(i)} & \frac{\partial f_4}{\partial y(i)} & \frac{\partial f_4}{\partial x(i)} \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & \Delta T & 1 & 0 \\ \Delta T & 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

and finally

$$f_u = f_\theta = \begin{bmatrix} \frac{\partial f_1}{\partial \theta(i)} \\ \frac{\partial f_2}{\partial \theta(i)} \\ \frac{\partial f_3}{\partial \theta(i)} \\ \frac{\partial f_4}{\partial \theta(i)} \end{bmatrix} = \begin{bmatrix} -a\Delta T \sin(\theta(i)) \\ a\Delta T \cos(\theta(i)) \\ \frac{a}{2}\Delta T^2 \cos(\theta(i)) \\ -\frac{a}{2}\Delta T^2 \sin(\theta(i)) \end{bmatrix}$$

This is implemented in the routine **bobs\_dopc\_b.m** on the class web site, and it is run using the driver file **dopc\_example\_b.m**.