Preface

Acknowledgment

The exercises are modified problems from different sources:

- Exercises 2.1, 2.2, 3.5, 4.1, 4.2, 4.3, 4.5, 5.1a, and 5.3 are from
  

- Exercises 1.1, 1.2, 1.3, and 1.4 are from
  

- Exercises 8.3, 8.4, 8.6, and 8.8 are based on


Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DARE</td>
<td>discrete-time algebraic Riccati equation</td>
</tr>
<tr>
<td>DRE</td>
<td>discrete-time Riccati equation</td>
</tr>
<tr>
<td>DP</td>
<td>dynamic programming</td>
</tr>
<tr>
<td>HJBE</td>
<td>Hamilton-Jacobi-Bellman equation</td>
</tr>
<tr>
<td>LQC</td>
<td>linear quadratic control</td>
</tr>
<tr>
<td>MIL</td>
<td>matrix inversion lemma</td>
</tr>
<tr>
<td>ODE</td>
<td>ordinary differential equation</td>
</tr>
<tr>
<td>PDE</td>
<td>partial differential equation</td>
</tr>
<tr>
<td>PMP</td>
<td>Pontryagin minimization principle</td>
</tr>
<tr>
<td>TPBVP</td>
<td>two-point boundary value problem</td>
</tr>
<tr>
<td>ARE</td>
<td>algebraic Riccati equation</td>
</tr>
</tbody>
</table>
Exercises
This version: September 2015
1 Discrete Optimization

In this section, we will consider solving optimal control problems on the form

\[
\begin{align*}
\text{minimize} & \quad \phi(x_N) + \sum_{k=0}^{N-1} f_0(k, x_k, u_k) \\
\text{subject to} & \quad x_{k+1} = f(k, x_k, u_k), \\
& \quad x_0 \text{ given}, \ x_k \in X_k, \\
& \quad u_k \in U(k, x_k),
\end{align*}
\]

using the *dynamic programming* (dp) algorithm. This algorithm finds the optimal feedback control \(u_k\), for \(k = 0, 1, \ldots, N - 1\), via the backwards recursion

\[
J(N, x) = \phi(x) \\
J(n, x) = \min_{u \in U(n, x)} \{f_0(n, x, u) + J(n + 1, f(n, x, u))\}.
\]

The optimal cost is \(J(0, x_0)\) and the optimal feedback control at stage \(n\) is

\[
u^*_n = \arg \min_{u \in U(n, x)} \{f_0(n, x, u) + J(n + 1, f(n, x, u))\}.
\]

1.1 A certain material is passed through a sequence of two ovens. Denote by

\(x_0\): the initial temperature of the material,

\(x_k, k = 1, 2\): the temperature of the material at the exit of oven \(k\),

\(u_k, k = 1, 2\): the prevailing temperature in oven \(k\).

The ovens are modeled as

\[
x_{k+1} = (1 - a)x_k + au_k, \quad k = 0, 1,
\]

where \(a\) is a known scalar from the interval \((0, 1)\). The objective is to get the final temperature \(x_2\) close to a given target \(T\), while expending relatively little energy. This is expressed by a cost function of the form

\[
r(x_2 - T)^2 + u_0^2 + u_1^2,
\]

where \(r > 0\) is a given scalar. For simplicity, we assume no constraints on \(u_k\).

(a) Formulate the problem as an optimization problem.

(b) Solve the problem using the dp algorithm when \(a = 1/2, T = 0\) and \(r = 1\).

(c) Solve the problem for all feasible \(a, T\) and \(r\).

1.2 Consider the system

\[
x_{k+1} = x_k + u_k, \quad k = 0, 1, 2, 3,
\]

with the initial state \(x_0 = 5\) and the cost function

\[
\sum_{k=0}^{3} (x_k^2 + u_k^2).
\]

Apply the dp algorithm to the problem when the control constraints are

\[
U(k, x_k) = \{u \mid 0 \leq x_k + u \leq 5, \ u \in \mathbb{Z}\}.
\]

1.3 Consider the cost function

\[
\alpha^N \phi(x_N) + \sum_{k=0}^{N-1} \alpha^k f_0(k, x_k, u_k),
\]
1.4 A decision maker must continually choose between two activities over a time interval $[0, T]$. Choosing activity $i$ at time $t$, where $i = 1, 2$, earns a reward at a rate $g_i(t)$, and every switch between the two activity costs $c > 0$. Thus, for example, the reward for starting with activity 1, switching to 2 at time $t_1$, and switching back to 1 at time $t_2 > t_1$ earns total reward

$$\int_0^{t_1} g_1(t) \, dt + \int_{t_1}^{t_2} g_2(t) \, dt + \int_{t_2}^T g_1(t) \, dt - 2c.$$  

We want to find a set of switching times that maximize the total reward. Assume that the function $g_2(t) - g_1(t)$ changes sign a finite number of times in the interval $[0, T]$. Formulate the problem as an optimization problem and write the corresponding DP algorithm.

1.5 Consider the infinite time horizon linear quadratic control (LQC) problem

$$\min_{u(t)} \sum_{k=0}^{\infty} (x_k^T Q x_k + u_k^T R u_k)$$

subject to

$$x_{k+1} = Ax_k + Bu_k,$$

$$x_0 \text{ given},$$

where $Q$ is a symmetric positive semidefinite matrix, $R$ is a symmetric positive definite matrix, and the pair $(A, B)$ is controllable. Show by using the Bellman equation that the optimal state feedback controller is given by

$$u^* = \mu(x) = -(B^T PB + R)^{-1} B^T PA x,$$

where $P$ is the positive definite solution to the discrete-time algebraic Riccati equation (DARE)

$$P = A^T PA - A^T PB (B^T PB + R)^{-1} B^T PA + Q.$$

1.6 Consider the finite horizon LQC problem

$$\min_{u(.)} x_N^T Q_N x_N + \sum_{k=0}^{N-1} (x_k^T Q x_k + u_k^T R u_k)$$

subject to

$$x_{k+1} = Ax_k + Bu_k,$$

$$x_0 \text{ given},$$

where $Q_N, Q$ and $R$ are symmetric positive semidefinite matrices.

(a) Apply the discrete version of the Pontryagin minimization principle (PMP) and show that the solution to the two-point boundary value problem (TPBVP)

$$x_{k+1} = Ax_k - \frac{1}{2} BR^{-1} B^T \lambda_{k+1}, \quad x_0 \text{ given},$$

$$\lambda_k = 2Q x_k + A^T \lambda_{k+1},$$

$$\lambda_N = 2Q_N x_N,$$

yields the minimizing sequence

$$u_k = - \frac{1}{2} R^{-1} B^T \lambda_{k+1}, \quad k = 0, \ldots, N - 1.$$

(b) Show that the above TPBVP can be solved using the Lagrange multiplier

$$\lambda_k = 2 S_k x_k,$$

where $S_k$ is given by the backward recursion

$$S_N = Q_N,$$

$$S_k = A^T (S_{k+1}^{-1} + BR^{-1} B^T)^{-1} A + Q.$$

(c) Use the matrix inversion lemma (MIL) to show that the above recursion for $S_k$ is equivalent to the discrete-time Riccati equation (DRE)

$$S_k = A^T S_{k+1} A - A^T S_{k+1} (B^T S_{k+1} B + R)^{-1} B^T S_{k+1} A + Q.$$

The MIL can be expressed as

$$(A + UBV)^{-1} = A^{-1} - A^{-1} U (B^{-1} + VA^{-1} U)^{-1} VA^{-1}$$

for arbitrarily matrices $A$, $B$, and $C$ such that required inverses exists.

(d) Use your results to calculate a feedback from the state $x_k$

$$u_k = \mu(k, x_k).$$
2 Dynamic Programming

In this chapter, we will consider solving optimal control problems on the form

\[
\minimize_{u(\cdot)} \phi(x(t_f)) + \int_{t_i}^{t_f} f_0(t, x(t), u(t)) \, dt
\]

subject to

\[
\dot{x}(t) = f(t, x(t), u(t)), \\
x(t_i) = x_i, \\
u(t) \in U, \forall t \in [t_i, t_f]
\]

using dynamic programming via the Hamilton-Jacobi-Bellman equation (HJBE):

1. Define Hamiltonian as

\[
H(t, x, u, \lambda) \triangleq f_0(t, x, u) + \lambda^T f(t, x, u).
\]

2. Pointwise minimization of the Hamiltonian yields

\[
\tilde{\mu}(t, x, \lambda) \triangleq \arg \min_{u \in U} H(t, x, u, \lambda),
\]

and the optimal control is given by

\[
u^*(t) \triangleq \tilde{\mu}(t, x, V_x(t, x)),
\]

where \(V(t, x)\) is obtained by solving the HJBE in Step 3.

3. Solve the HJBE with boundary conditions:

\[
-V_t = H(t, x, \tilde{\mu}(t, x, V_x), V_x), \quad V(t_f, x) = \phi(x).
\]

2.1 Find the optimal solution to the problem

\[
\minimize_{u(\cdot)} \int_{0}^{t_f} \left((x(t) - \cos t)^2 + u^2(t)\right) \, dt
\]

subject to

\[
\dot{x}(t) = u(t), \\
x(0) = 0,
\]

expressed as a system of ODEs.

2.2 A producer with production rate \(x(t)\) at time \(t\) may allocate a portion \(u(t)\) of his/her production rate to reinvestments in the factory (thus increasing the production rate) and use the rest \((1 - u(t))\) to store goods in a warehouse. Thus \(x(t)\) evolves according to

\[
\dot{x}(t) = \alpha u(t) x(t),
\]

where \(\alpha\) is a given constant. The producer wants to maximize the total amount of goods stored summed with the capacity of the factory at final time.

(a) Formulate the continuous-time optimal control problem.
(b) Solve the problem analytically.
3 PMP: A Special Case

In this chapter, we will start solving optimal control problems on the form

\[
\begin{align*}
\text{minimize} & \quad \phi(x(t_f)) + \int_{t_i}^{t_f} f_0(t, x(t), u(t)) \, dt \\
\text{subject to} & \quad \dot{x}(t) = f(t, x(t), u(t)), \\
& \quad x(t_i) = x_i.
\end{align*}
\]

using a special case of the PMP: Assuming that the final time is fixed and that there are no terminal constraints, the problem may be solved using the following sequence of steps

1. Define the Hamiltonian as

\[
H(t, x, u, \lambda) \triangleq f_0(t, x, u) + \lambda^T f(t, x, u).
\]

2. Pointwise minimization of the Hamiltonian yields

\[
\tilde{\mu}(t, x, \lambda) \triangleq \arg \min_u H(t, x, u, \lambda),
\]

and the optimal control candidate is given by

\[
u^*(t) \triangleq \tilde{\mu}(t, x(t), \lambda(t)).\]

3. Solve the adjoint equations

\[
\dot{\lambda}(t) = -\frac{\partial H}{\partial x}(t, x(t), \tilde{\mu}(t, x(t), \lambda(t)), \lambda(t)),
\lambda(t_f) = \frac{\partial \phi}{\partial x}(x^*(t_f))
\]

4. Compare the candidate solutions obtained using PMP

3.1 Solve the optimal control problem

\[
\begin{align*}
\text{minimize} & \quad \int_{t_i}^{t_f} (x(t) + u^2(t)) \, dt \\
\text{subject to} & \quad \dot{x}(t) = x(t) + u(t) + 1, \\
& \quad x(0) = 0.
\end{align*}
\]

3.2 Find the extremals of the functionals

(a) \( J = \int_{0}^{1} \dot{y} \, dt, \)

(b) \( J = \int_{0}^{1} y \dot{y} \, dt, \)

given that \( y(0) = 0 \) and \( y(1) = 1. \)

3.3 Find the extremals of the following functionals

(a) \( J = \int_{0}^{1} (y^2 + \dot{y}^2 - 2y \sin t) \, dt, \)

(b) \( J = \int_{0}^{1} \frac{y^2}{t^3} \, dt, \)

(c) \( J = \int_{0}^{1} (y^2 + \dot{y}^2 + 2ye^t) \, dt. \)

where \( y(0) = 0. \)

3.4 Among all curves of length \( l \) in the upper half plane passing through the points \((-a, 0)\) and \((a, 0)\) find the one which encloses the largest area in the interval \([-a, a]\), i.e., solve

\[
\begin{align*}
\text{maximize} & \quad \int_{-a}^{a} x(t) \, dt \\
\text{subject to} & \quad x(-a) = 0, \\
& \quad x(a) = 0,
\end{align*}
\]

\[
K(x) = \int_{-a}^{a} \sqrt{1 + \dot{x}(t)^2} \, dt = l.
\]
3.5 From the point \((0,0)\) on the bank of a wide river, a boat starts with relative speed to the water equal \(\nu\). The stream of the river becomes faster as it departs from the bank, and the speed is \(g(y)\) parallel to the bank, see Figure 3.5a.

![Figure 3.5a. The so called Zermelo problem in Exercise 3.5.](image)

The movement of the boat is described by

\[
\begin{align*}
\dot{x}(t) &= \nu \cos(\phi(t)) + g(y(t)) \\
\dot{y}(t) &= \nu \sin(\phi(t))
\end{align*}
\]

where \(\phi\) is the angle between the boat direction and bank. We want to determine the movement angle \(\phi(t)\) so that \(x(T)\) is maximized, where \(T\) is a fixed transfer time. Show that the optimal control must satisfy the relation

\[
\frac{1}{\cos(\phi^*(t))} + \frac{g(y(t))}{\nu} = \text{constant}.
\]

In addition, determine the direction of the boat at the final time.

3.6 Consider the electromechanical system in Figure 3.6a. The electrical subsystem contains a voltage source \(u(t)\) which passes current \(i(t)\) through a resistor \(R\) and a solenoid of inductance \(l(z)\) where \(z\) is the position of the solenoid core. The mechanical subsystem is the core of mass \(m\) which is constrained to move horizontally and is retrained by a linear spring and a damper.

(a) Use the following principle:

\[
J(u(\cdot)) = \delta \int_{t_i}^{t_f} L(q(t), u(t)) \, dt + \int_{t_i}^{t_f} \tau(q(t))^T \delta q \, dt = 0,
\]

where \(q\) are generalized coordinates, \(\tau\) are generalized forces and \(\delta q = \delta u\), to derive the generalization of the Euler-Lagrange equations for nonconservative “mechanical” systems without energy dissipation (no damping):

\[
\frac{d}{dt} \frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} = \tau,
\]

Also, apply the resulting equation to the electromechanical system above.

(b) For a system with energy dissipation the Euler-Lagrange equations are given by

\[
\frac{d}{dt} \frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} = \tau - \frac{\partial F}{\partial q}
\]

where \(q\) are generalized coordinates, \(\tau\) are generalized forces, and \(F\) is the Rayleigh’s dissipation energy defined as

\[
F(q, \dot{q}) = \frac{1}{2} \sum_j k_j \dot{q}^2_j
\]

where \(k_j\) are damping constants. \(2F\) is the rate of energy dissipation due to the damping. Derive the system equations with the damping included.

3.7 Consider the optimal control problem

\[
\text{minimize} \quad \gamma \frac{1}{2} x(t_f)^2 + \frac{1}{2} \int_{t_i}^{t_f} u^2(t) \, dt
\]

subject to

\[
\begin{align*}
\dot{x}(t) &= u(t) \\
x(t_i) &= x_i.
\end{align*}
\]

Derive an optimal feedback policy using PMP.
3.8 Consider the dynamical system

\[ \dot{x}(t) = u(t) + w(t) \]

where \( x(t) \) is the state, \( u(t) \) is the control signal, \( w(t) \) is a disturbance signal, and where \( x(0) = 0 \). In so-called \( H_{\infty} \)-control the objective is to find a control signal that solves

\[ \min_{u(\cdot)} \max_{w(\cdot)} J(u(\cdot), w(\cdot)) \]

where

\[ J(u(\cdot), w(\cdot)) = \int_0^T \left( \rho^2 x^2(t) + u^2(t) - \gamma^2 w^2(t) \right) \]

where \( \rho \) is a given parameter and where \( \gamma < 1 \).

(a) Express the solution as a TBPV problem using the PMP. Note that you do not need to solve the equations yet.

(b) Solve the TBPV and express the solution in feedback form, i.e., \( u(t) = \mu(t, x(t)) \) for some \( \mu \).

(c) For what values of \( \gamma \) does the control signal exist for all \( t \in [0, T] \)
4 PMP: General Results

In this chapter, we will consider solving optimal control problems on the form

\[
\begin{align*}
\text{minimize } & \quad \phi(x(t_f)) + \int_{t_i}^{t_f} f_0(t, x(t), u(t)) \, dt \\
\text{subject to } & \quad \dot{x}(t) = f(t, x(t), u(t)), \\
& \quad x(t_i) = x_i, \quad x(t_f) \in S_f(t_f), \\
& \quad u(t) \in U, \quad t \in [t_i, t_f],
\end{align*}
\]

using the general case of the PMP:

1. Define the Hamiltonian as

\[
H(t, x, u, \lambda) \triangleq f_0(t, x, u) + \lambda^T f(t, x, u).
\]

2. Pointwise minimization of the Hamiltonian yields

\[
\bar{\mu}(t, x, \lambda) \triangleq \arg\min_{u \in U} H(t, x, u, \lambda),
\]

and the optimal control candidate is given by

\[
u^*(t) \triangleq \bar{\mu}(t, x(t), \lambda(t)).
\]

3. Solve the TPBVP

\[
\begin{align*}
\dot{x}(t) & = \frac{\partial H}{\partial \lambda}(t, x(t), \bar{\mu}(t, x(t), \lambda(t)), \lambda(t)), \quad x(t_i) = x_i, \quad x(t_f) \in S_f(t_f), \\
\dot{\lambda}(t) & = -\frac{\partial H}{\partial x}(t, x(t), \bar{\mu}(t, x(t), \lambda(t)), \lambda(t)).
\end{align*}
\]

4. Compare the candidate solutions obtained using PMP.

4.1 A producer with production rate \( x(t) \) at time \( t \) may allocate a portion \( u(t) \) of his/her production rate to reinvestments in a factory (thus increasing the production rate) and use the rest \((1 - u(t))\) to store goods in a warehouse. Thus \( x(t) \) evolves according to

\[
\dot{x}(t) = \alpha u(t)x(t),
\]

where \( \alpha \) is a given constant. The producer wants to maximize the total amount of goods stored summed with the capacity of the factory at final time. This gives us the following problem:

\[
\begin{align*}
\text{maximize } & \quad x(t_f) + \int_0^{t_f} (1 - u(t))x(t) \, dt \\
\text{subject to } & \quad \dot{x}(t) = \alpha u(t)x(t), \quad 0 < \alpha < 1 \\
& \quad x(0) = x_0 > 0, \\
& \quad 0 \leq u(t) \leq 1, \quad \forall t \in [0, t_f].
\end{align*}
\]

Find an analytical solution to the problem above using the PMP.

4.2 A spacecraft approaching the face of the moon can be described by the following equations

\[
\begin{align*}
\dot{x}_1(t) & = x_2(t), \\
\dot{x}_2(t) & = \frac{cu(t)}{x_3(t)} - g(1 - kx_1(t)), \\
\dot{x}_3(t) & = -u(t),
\end{align*}
\]

where \( u(t) \in [0, M] \) for all \( t \), and the initial conditions are given by

\[
x(0) = (h, \nu, m)^T
\]

with the positive constants \( c, g, k, M, h, \nu \) and \( m \). The state \( x_1 \) is the altitude above the surface, \( x_2 \) the velocity and \( x_3 \) the mass of the spacecraft. Calculate the structure of the fuel minimizing control that brings the craft to rest on the surface of the moon. The fuel consumption is

\[
\int_0^{t_f} u(t) \, dt,
\]
and the transfer time \( t_f \) is free. Show that the optimal control law is bang-bang with at most two switches.

4.3 A wasp community can in the time interval \([0, t_f]\) be described by

\[
\dot{x}(t) = (au(t) - b)x(t), \quad x(0) = 1, \\
\dot{y}(t) = c(1 - u(t))x(t), \quad y(0) = 0,
\]

where \( x \) is the number of worker bees and \( y \) the number of queens. The constants \( a, b \) and \( c \) are given positive numbers, where \( b \) denotes the death rate of the workers and \( a, c \) are constants depending on the environment.

The control \( u(t) \in [0, 1], \) for all \( t \in [0, t_f], \) is the proportion of the community resources dedicated to increase the number of worker bees. Calculate the control \( u \) that maximizes the number of queens at time \( t_f, \) i.e., the number of new communities in the next season.

4.4 Consider a DC motor with input signal \( u(t) \), output signal \( y(t) \) and the transfer function

\[ G(s) = \frac{1}{s^2}. \]

Introduce the states \( x_1(t) \triangleq y(t) \) and \( x_2(t) \triangleq \dot{y}(t) \) and compute a control signal that takes the system from an arbitrary initial state to the origin in minimal time when the control signal is bounded by \( |u(t)| \leq 1 \) for all \( t \).

4.5 Consider a rocket, modeled as a particle of constant mass \( m \) moving in zero gravity empty space, see Figure 4.5a. Let \( u \) be the mass flow, assumed to be a known function of time, and let \( c \) be the constant thrust velocity. The equations of motion are given by

\[
\dot{x}_1(t) = x_3(t), \\
\dot{x}_2(t) = x_4(t), \\
\dot{x}_3(t) = \frac{c}{m} u(t) \cos v(t), \\
\dot{x}_4(t) = \frac{c}{m} u(t) \sin v(t).
\]

(a) Assume that \( u(t) > 0 \) for all \( t \in \mathbb{R} \). Show that cost functionals of the class

\[ \text{minimize } \int_0^{t_f} dt \quad \text{or} \quad \text{minimize } \phi(x(t_f)) \]

gives the optimal control

\[ \tan v^*(t) = \frac{a + b \cdot t}{c + d \cdot t} \]

(the bilinear tangent law).

(b) Assume that the rocket starts at rest at the origin and that we want to drive it to a given height \( x_2f \) in a given time \( t_f \) such that the final velocity in the horizontal direction \( x_3(t_f) \) is maximized while \( x_4f = 0 \). Show that the optimal control is reduced to a linear tangent law,

\[ \tan v^*(t) = a + b \cdot t. \]

(c) Let the rocket in the example above represent a missile whose target is at rest. A natural objective is then to minimize the transfer time \( t_f \) from the state \((0, 0, x_3i, x_4i)\) to the state \((x_1f, x_2f, \text{free, free})\). Solve the problem under assumption that \( u \) is constant.

(d) To increase the realism we now assume that the motion is under a constant gravitational force \( g \). The only difference compared to the system above is that equation for the fourth state is replaced by

\[ \dot{x}_4(t) = \frac{c}{m} u(t) \sin v(t) - g. \]
Show that the bilinear tangent law still is optimal for the cost functional

\[ \min_{\bar{v}(\cdot)} \phi(x(t_f)) + \int_0^{t_f} dt \]

(e) In most cases the assumption of constant mass is inappropriate, at least if longer periods of time are considered. To remedy this flaw we add the mass of the rocket as a fifth state. The equations of motion becomes

\[
\begin{align*}
\dot{x}_1(t) &= x_3(t), \\
\dot{x}_2(t) &= x_4(t), \\
\dot{x}_3(t) &= \frac{c}{x_5(t)} u(t) \cos v(t), \\
\dot{x}_4(t) &= \frac{c}{x_5(t)} u(t) \sin v(t) - g, \\
\dot{x}_5(t) &= -u(t),
\end{align*}
\]

where \( u(t) \in [0, u_{\text{max}}], \forall t \in [0, t_f] \). Show that the optimal solution to transferring the rocket from a state of given position, velocity and mass to a given altitude \( x_{2f} \) using a given amount of fuel, such that the distance \( x_1(t_f) - x_1(0) \) is maximized, is

\[
\begin{align*}
v^*(t) &= \text{constant}, \\
u^*(t) &= \{u_{\text{max}}, 0\}.
\end{align*}
\]
5 Infinite Horizon Optimal Control

In this chapter, we will consider solving optimal control problems on the form

$$\min_{u(t)} \int_0^\infty f_0(x(t), u(t)) \, dt$$

subject to

$$\dot{x}(t) = f(x(t), u(t)), \quad x(0) = x_0, \quad u(t) \in U(x)$$

using dynamic programming via the infinite horizon Hamilton-Jacobi-Bellman equation (HJBE):

1. Define Hamiltonian as

$$H(x, u, \lambda) \triangleq f_0(x, u) + \lambda^T f(x, u).$$

2. Pointwise minimization of the Hamiltonian yields

$$\tilde{\mu}(x, \lambda) \triangleq \arg \min_{u \in U} H(x, u, \lambda),$$

and the optimal control is given by

$$u^*(t) \equiv \tilde{\mu}(x, V_x(x)).$$

where $V(x)$ is obtained by solving the infinite horizon HJBE in Step 3.

3. Solve the infinite horizon HJBE with boundary conditions:

$$0 = H(x, \tilde{\mu}(x, V_x), V_x).$$

5.1 Consider the following infinite horizon optimal control problem

$$\min_{u(t)} \frac{1}{2} \int_0^\infty \left( u^2(t) + x^4(t) \right) \, dt$$

subject to

$$\dot{x}(t) = u(t), \quad x(0) = x_0,$$

(a) Show that the solution is given by $u^*(t) = -x^2(t) \text{sgn} x(t)$.

(b) Add the constraint $|u| \leq 1$. What is now the optimal cost-to-go function $J^*(x)$ and the optimal feedback?

5.2 Solve the optimal control problem

$$\min_{u(t)} \frac{1}{2} \int_0^\infty \left( u^2(t) + x^2(t) + 2x^4(t) \right) \, dt$$

subject to

$$\dot{x}(t) = x^3(t) + u(t), \quad x(0) = x_0,$$

5.3 Consider the problem

$$\min_{u(t)} J = \int_0^\infty \left( \alpha^2 u^2(t) + \beta^2 y^2(t) \right) \, dt$$

subject to

$$\ddot{y}(t) = u(t), \quad u(t) \in R, \forall t \geq 0$$

where $\alpha > 0$ and $\beta > 0$.

(a) Write the system in a state space form with a state vector $x$.

(b) Write the cost function $J$ on the form $\int_0^\infty \left( u(t)^T R u(t) + x(t)^T Q x(t) \right) \, dt$.

(c) What is the optimal feedback $u(t)$?

(d) To obtain the optimal feedback, the algebraic Riccati equation (ARE) must be solved. Compute the optimal feedback by using the Matlab command are for the case when $\alpha = 1$ and $\beta = 2$. In addition, compute the eigenvalues of the system with optimal feedback.
(e) In this case the ARE can be solved by hand since the dimension of the state space is low. Solve the ARE by hand or by using a symbolic tool for arbitrarily $\alpha$ and $\beta$. Verify that your solution is the same as in the previous case when $\alpha = 1$ and $\beta = 2$.

(f) (Optional) Compute an analytical expression for the eigenvalues of the system with optimal feedback.

5.4 Assume that the lateral equations of motions (i.e. the horizontal dynamics) for a Boeing 747 can be expressed as

\[ \dot{x}(t) = Ax(t) + Bu(t) \]  

(5.1)

where $x = (v \ r \ p \ \phi \ \psi \ y)^T$, $u = (\delta a \ \delta r)^T$,

- $v$: lateral velocity
- $r$: yaw rate
- $p$: roll rate
- $\phi$: roll angle
- $\psi$: yaw angle
- $y$: lateral distance from a straight reference path
- $\delta a$: aileron angle
- $\delta r$: rudder angle

and

\[
A = \begin{pmatrix}
-0.089 & -2.19 & 0 & 0.319 & 0 & 0 \\
0.076 & -0.217 & -0.166 & 0 & 0 & 0 \\
-0.602 & 0.327 & -0.975 & 0 & 0 & 0 \\
0 & 0.15 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 2.19 & 0 \\
\end{pmatrix}, \quad B = \begin{pmatrix}
0 & 0.0327 \\
0 & 0.0264 & -0.151 \\
0.227 & 0.0636 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
\end{pmatrix}
\]

\[
C = \begin{pmatrix}
0 & 0 & 0 & 0 & 1
\end{pmatrix}
\]

(5.2)

The task is now to design an autopilot that keeps the aircraft flying along a straight line during a long period of time. First an intuitive approach will be used and then a optimal control approach.
6 Model Predictive Control

In this exercise session we will focus on computing the explicit solution to the model predictive control (MPC) problem (see [3]) using the multi-parametric toolbox (MPT) for MATLAB.

The toolbox is added to the MATLAB path via the command

\[ \text{initcourse('tsr08')} \]

and the MPT manual can be downloaded from the course homepage.

6.1 Here we are going to consider the second order system

\[ y(t) = \frac{2}{s^2 + 3s + 2} u(t), \quad (6.1) \]

sampled with \( T_s = 0.1 \) s to obtain the discrete-time state-space representation

\[
\begin{align*}
x_{k+1} &= \begin{pmatrix} 0.7326 & -0.1722 \\ 0.08611 & 0.9909 \end{pmatrix} x_k + \begin{pmatrix} 0.1722 \\ 0.009056 \end{pmatrix} u_k \\
y_k &= \begin{pmatrix} 0 & 1 \end{pmatrix} x_k.
\end{align*}
\]

(6.2)

(a) Use the MPT toolbox to find the MPC law for (6.2) using the 2-norm with \( N = 2 \), \( Q = I_{2 \times 2} \), \( R = 0.01 \), and the input satisfying

\[-2 \leq u \leq 2 \quad \text{and} \quad \begin{pmatrix} -10 \\ -10 \end{pmatrix} \leq x \leq \begin{pmatrix} 10 \\ 10 \end{pmatrix} \]

How may regions are there? Present plots on the control partition and the time trajectory for the initial condition \( x_0 = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \)T.

(b) Find the number of regions in the control law for \( N = 1, 2, \ldots, 14 \). How does the number of regions seem to depend on \( N \), e.g., is the complexity polynomial or exponential? Estimate the order of the complexity via

\[
\arg \min_{\alpha, \beta} \| \alpha N^\beta - n_r \|_2^2 \quad \text{and} \quad \arg \min_{\gamma, \delta} \| \gamma (\delta N - 1) - n_r \|_2^2,
\]

where \( n_r \) denotes the number of regions. Discuss the result.
7 Numerical Algorithms

All Matlab-files that are needed are available to download from the course pages, but some lines of code are missing and the task is to complement the files to be able to solve the given problems. All places where you have to write some code are indicated by three question marks (???).

A basic second-order system will be used in the exercises below and the model and the problem are presented here. Consider a 1D motion model of a particle with position \( z(t) \) and speed \( v(t) \). Define the state vector \( x = (z\ v)^T \) and the continuous time model

\[
\dot{x}(t) = f(x(t), u(t)) = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} x(t) + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u(t). \tag{7.1}
\]

The problem is to go from the state \( x_i = x(0) = (1\ 1)^T \) to \( x_f = (0\ 0)^T \), where \( t_f = 2 \), but such that the control input energy \( \int_0^{t_f} u^2(t)dt \) is minimized. Thus, the optimization problem is

\[
\min_u \ J = \int_0^{t_f} f_0(x, u)dt \\
\text{s.t.} \ \dot{x}(t) = f(x(t), u(t)) \\
x(0) = x_i \\
x(t_f) = x_f \tag{7.2}
\]

where \( f_0(t) = u^2(t) \) and \( f(\cdot) \) is given in (7.1).

7.1 Discretization Method 1. In this approach we will use the discrete time model

\[
x[k+1] = \bar{f}(x[k], u[k]) = \begin{pmatrix} 1 & h \\ 0 & 1 \end{pmatrix} x[k] + \begin{pmatrix} 0 \\ h \end{pmatrix} u[k] \tag{7.3}
\]

to represent the system where \( t = kh \), \( h = t_f/N \) and \( x[k] = x(kh) \). The discretized version of the problem (7.2) is then

\[
\min_{u[n], n=0, \ldots, N-1} h \sum_{k=0}^{N-1} u^2[k] \\
\text{s.t.} \ x[k+1] = \bar{f}(x[k], u[k]) \\
x[0] = x_i \\
x[N] = x_f. \tag{7.4}
\]

Define the optimization parameter vector as

\[
\]

Note that \( u[N] \) is superfluous, but is included to make the presentation and the code more convenient. Furthermore, define

\[
F(y) = h \sum_{k=0}^{N-1} u^2[k] \tag{7.6}
\]

and

\[
G(y) = \begin{pmatrix} g_1(y) \\ g_2(y) \\ g_3(y) \\ \vdots \\ g_{N+2}(y) \end{pmatrix} = \begin{pmatrix} x[0] - x_i \\ x[N] - x_f \\ x[1] - \bar{f}(x[0], u[0]) \\ x[2] - \bar{f}(x[1], u[1]) \\ \vdots \\ x[N] - \bar{f}(x[N-1], u[N-1]) \end{pmatrix} \tag{7.7}
\]

Then the optimization problem in (7.4) can be expressed as the constrained problem

\[
\min_y \ F(y) \\
\text{s.t.} \ G(y) = 0. \tag{7.8}
\]
(a) Show that the discrete time model in (7.3) can be obtained from (7.1) by using the Euler-approximation

\[ x[k+1] = x[k] + hf(x[k], u[k]). \]  

(7.10)

and complement the file secOrderSysEqDisc.m with this model.

(b) Complement the file secOrderSysCostCon.m with a discrete time version of the cost function.

(c) Note that the constraints are linear for this system model and this is very important to exploit in the optimization routine. In fact the problem is a quadratic programming problem, and since it only contain equality constraints it can be solved as a linear system of equations.

Complete the file mainDiscLinconSecOrderSys.m by creating the linear constraint matrix \( A_{eq} \) and the vector \( eq \) that contains the system model recursion (7.3). The optimization problem is solved by using the Matlab function fminunc.

(d) (Optional) Implement an unconstrained gradient method that can replace the file secOrderSysNonlinconSecOrderSys.m by computing

\[ c_{eq} = (g_3(y) \ g_4(y) \ \ldots \ g_{N+2}(y)) \]  

(7.11)

and the Jacobian

\[ \left( \frac{\partial g_3}{\partial x[0]} \ \frac{\partial g_3}{\partial x[1]} \ \frac{\partial g_3}{\partial x[1]} \ \frac{\partial g_3}{\partial x[1]} \ \ldots \right)^T \]  

(7.12)

Each row in (7.12) is computed in each loop in the for-statement in secOrderSysNonlincon.m. Note that the length of the state vector is 2. Also note that Matlab wants the Jacobian to be transposed, but that operation is already added last in the file. Finally, run the Matlab script mainDiscNonlinconSecOrderSys.m to solve the problem again.

7.2 Discretization Method 2. In this exercise we will also use the discrete time model in (7.3). However, the terminal constraints are removed by including it in the objective function. If the terminal constraints are not fulfilled they will penalize the objective function. Now the optimization problem can be defined as

\[ \min_{u[n], n=0, \ldots, N-1} c||x[N] - x_f||^2 + h \sum_{k=0}^{N-1} u^2[k] \]  

(7.13)

s.t. \( x[k+1] = \bar{f}(x[k], u[k]) \)

\( x[0] = x_i \)

where \( \bar{f}(\cdot) \) is given in (7.3), \( N = t_f/h \) and \( c \) is a predefined constant. This problem can be rewritten as an unconstrained nonlinear problem

\[ \min_{u[n], n=0, \ldots, N-1} J = \mathcal{F}(x_1, x_f, u[0], u[1], \ldots, u[N-1]) \]  

(7.14)

where \( w(\cdot) \) contains the system model recursion (7.3). The optimization problem will be solved by using the Matlab function fminunc.

(a) Write down an explicit algorithm for the computation of \( \mathcal{F}(\cdot) \).

(b) Complement the file secOrderSysCostUnc.m with the cost function \( \mathcal{F}(\cdot) \).

Solve the problem by running the script mainDiscUncSecOrderSys.m.

(c) Compare method 1 and 2. What are the advantages and disadvantages? Which method handles constraints best? Suppose the task was also to implement the optimization algorithm itself, which method would be easiest to implement an algorithm for (assuming that the state dynamics is non-linear)?

(d) (Optional) Implement an unconstrained gradient method that can replace the fminunc function in mainDiscUncSecOrderSys.m.

7.3 Gradient Method. As above the terminal constraint is added as a penalty in the objective function in order to introduce the PMP gradient method here in a more straightforward way. Thus, the optimization problem is

\[ \min_{u} J = c||x(t_f) - x_f||^2 + \int_{0}^{t_f} u^2(t)dt \]  

(7.15)

s.t. \( \dot{x}(t) = f(x(t), u(t)) \)

\( x(0) = x_i \).

(7.16)

(7.17)

(a) Write down the Hamiltonian and show that the Hamiltonian partial derivatives w.r.t. \( x \) and \( u \) are

\[ H_x = (0 \ \lambda_1) \]

\[ H_u = 2u(t) + \lambda_2, \]  

(7.18)
respectively.

(b) What is the adjoint equation and its terminal constraint?

(c) Complement the files secOrderSysEq.m with the system model, the file secOrderSysAdjointEq.m with the adjoint equations, the file secOrderSysFinalLambda.m with the terminal values of $\lambda$, and the file secOrderSysGradient.m with the control signal gradient. Finally, complement the script mainGradientSecOrderSys.m and solve the problem.

(d) Try some different values of the penalty constant $c$. What happens if $c$ is “small”? What happens if $c$ is “large”?

7.4 Shooting Method. Shooting methods are based on successive improvements of the unspecified terminal conditions of the two point boundary value problem. In this case we can use the original problem formulation in (7.2).

Complement the files secOrderSysEqAndAdjointEq.m with the combined system and adjoint equation, and the file theta.m with the final constraints. The main script is mainShootingSecOrderSys.m.

7.5 Reflections.

(a) What are the advantages and disadvantages of discretization methods?

(b) Discuss the advantages and disadvantages of the problem formulation in 7.1 compared to the formulation in 7.2.

(c) What are the advantages and disadvantages of gradient methods?

(d) Compare the methods in 7.1, 7.2, and 7.3 in terms of accuracy and complexity/speed. Also compare the results for different algorithms used by fmincon in Exercises 7.1. Can all algorithms handle the optimization problem?

(e) Assume that there are constraints on the control signal and you can use either the discretization method in 7.1 or the gradient method in 7.3. Which methods would you use?

(f) What are the advantages and disadvantages of shooting methods?
8 The PROPT toolbox

PROPT is a Matlab toolbox for solving optimal control problems. See the PROPT manual for more information.

PROPT is using a so called Gauss pseudospectral method (GPM) to solve optimal control problems. A GPM is a direct method for discretizing a continuous optimal control problem into a nonlinear program (NLP) that can be solved by well-developed numerical algorithms that attempt to satisfy the Karush-Kuhn-Tucker (KKT) conditions associated with the NLP. In pseudospectral methods the state and control trajectories are parameterized using global and orthogonal polynomials, see Chapter 10.5 in the lecture notes. In GPM the points at which the optimal control problem is discretized (the collocation points) are the Legendre-Gauss (LG) points. See for details.

PROPT initialization; PROPT is available in the ISY computer rooms with Linux, e.g. Egypten. Follow these steps to initialize PROPT:

1. In a terminal window; run module add matlab/tomlab
2. Start Matlab, e.g. by running matlab & in a terminal window
3. At the Matlab prompt; run initcourse tsrt08
4. At the Matlab prompt; run initPropt
5. Check that the printed information looks OK.

8.1 Minimum Curve Length; Example 1. Consider the problem of finding the curve with the minimum length from a point $(0,0)$ to $(x_f,y_f)$. The solution is of course obvious, but we will use this example as a starting point.

The problem can be formulated by using an expression of the length of the curve from $(0,0)$ to $(x_f,y_f)$:

$$s = \int_{0}^{x_f} \sqrt{1 + y'(x)^2} \, dx.$$  \hfill (8.1)

Note that $x$ is the “time” variable. The optimal control problem is solved in minCurveLength.m by using PROPT/Matlab.

(a) Derive the expression of the length of the curve in (8.1).
(b) Write problem on a standard optimal control form. Thus, what are $\psi(\cdot)$, $f(\cdot)$, $f_0(\cdot)$, etc.
(c) Use Pontryagins minimum principle to show that the solution is a straight line.
(d) Examine and run the script minCurveLength.m

8.2 Minimum Curve Length; Example 2. Consider the minimum length curve problem again. The problem can be re-formulated by using a constant speed model where the control signal is the heading angle. This problem is solved in the PROPT/Matlab file minCurveLengthHeadingCtrl.m.

(a) Examine the PROPT/Matlab file and write down the optimal control problem that is solved in this example, i.e., what are $f$, $f_0$ and $\phi$ in the standard optimal control formulation.
(b) Run the script and compare with the result from the previous exercise.

8.3 The Brachistochrone problem; Find the curve between two points, $A$ and $B$, that is covered in the least time by a body that starts in $A$ with zero speed and is constrained to move along the curve to point $B$, under the action of gravity and assuming no friction. The Brachistochrone problem (Gr. brachistos - the shortest, chronos - time) was posed by Johann Bernoulli in Acta Eruditorum in 1696 and the history about this problem is indeed very interesting and
The Brachistochrone problem; Example 1. Let the motion of the particle, under the influence of gravity $g$, be defined by a time continuous state space model $\dot{X} = f(X, \theta)$ where the state vector is defined as $X = (x \ y \ v)^T$ and $(x \ y)^T$ is the Cartesian position of the particle in a vertical plane and $v$ is the speed, i.e.,

$$\dot{x} = v \sin(\theta)$$
$$\dot{y} = -v \cos(\theta)$$

(8.2)

The motion of the particle is constrained by a path that is defined by the angle $\theta(t)$, see Figure 8.3a.

(a) Give an explicit expression of $f(X, \theta)$ (only the expression for $\dot{v}$ is missing).

(b) Define the Brachistochrone problem as an optimal control problem based on this state space model. Assume that the initial position of the particle is in the origin and the initial speed is zero. The final position of the particle is $(x_f \ y_f)^T = (10 \ -3)^T$.

(c) Modify the script `minCurveLengthHeadingCtrl.m` of the minimum curve length example [8.2] above and solve this optimal control problem with PROPT.

8.4 The Brachistochrone problem; Example 2. The time for a particle to travel on a curve between the points $p_0 = (0,0)$ and $p_f = (x_f, y_f)$ is

$$t_f = \int_{p_0}^{p_f} \frac{1}{v} ds$$

(8.3)

where $ds$ is an element of arc length and $v$ is the speed.

(a) Show that the travel time $t_f$ is

$$t_f = \int_0^{x_f} \sqrt{1 + y'(x)^2} \frac{1}{\sqrt{-2gy(x)}} dx$$

(8.4)

in the Brachistochrone problem where the speed of the particle is due to gravity and its initial speed is zero.

(b) Define the Brachistochrone problem as an optimal control problem based on the expression in (8.4). Note that the time variable is eliminated and that $y$ is a function $x$ now.

(c) Use the PMP to show (or derive) that the optimal trajectory is a cycloid

$$y'(x) = \sqrt{\frac{C - y(x)}{y(x)}}.$$  

(8.5)

with the solution

$$x = \frac{C}{2} (\phi - \sin(\phi))$$
$$y = \frac{C}{2} (1 - \cos(\phi)).$$

(8.6)

(d) Try to solve the this optimal control problem with PROPT/Matlab, by modifying the script `minCurveLength.m` from the minimum curve length example. Compare the result with the solution in Examples 1 above. Try to explain why we have problems here.

8.5 The Brachistochrone problem; Example 3. To avoid the problem when solving the previous problem in PROPT/Matlab we can use the results in the exercise [8.4]. The cycloid equation (8.5) can be rewritten as

$$y(x)(y'(x)^2 + 1) = C.$$  

(8.7)
PROPT can now be used to solve the Brachistochrone problem by defining a feasibility problem

\[
\begin{align*}
\min \quad & C \\
\text{s.t.} \quad & y(x)(y'(x)^2 + 1) = C \\
& y(0) = 0 \\
& y(x_f) = y_f
\end{align*}
\] (8.8)

Solve this feasibility problem with PROPT/Matlab. Note that the system dynamics in example 3 (and 4) are not modeled on the standard form as an ODE. Instead, the differential equations on the form \(F(x, x) = 0\) is called differential algebraic equations (DAE) and this is a more general form of system of differential equations.

8.6 The Brachistochrone problem; Example 4. An alternative formulation of the Brachistochrone problem can be obtained by considering the “law of conservation of energy” (which is related to the “principle of least action”, see Example 17 in the lecture notes). Consider the position \((x, y)^T\) of the particle and its velocity \((\dot{x}, \dot{y})^T\).

(a) Write the kinetic energy \(E_k\) and the potential energy \(E_p\) as functions of \(x, y, \dot{x}, \dot{y}\), the mass \(m\), and the gravity constant \(g\).

(b) Define the Brachistochrone problem as an optimal control problem based on the “law of conservation of energy”.

(c) Solve this optimal control problem with PROPT. Assume that the mass of the particle is \(m = 1\).

8.7 The Brachistochrone problem; Reflections.

(a) Is the problem in Exercise 8.4 due to the problem’s formulation or the solution method or both?

(b) Discuss why not only the solution method (i.e., the optimization algorithm) is important, but also the problem formulation.

(c) Compare the different approaches to the Brachistochrone problem in Exercises 3.3, 8.5, and 8.6. Try to explain the advantages and disadvantages of the different formulations of the Brachistochrone problem. Which approach can handle additional constraints best?

8.8 The Zermelo problem. Consider a special case of Exercise 3.3 where \(g(y) = y\). Thus, from the point \((0, 0)\) on the bank of a wide river, a boat starts with relative speed to the water equal \(\nu\). The stream of the river becomes faster as it departs from the bank, and the speed is \(g(y) = y\) parallel to the bank. The movement of the boat is described by

\[
\begin{align*}
\dot{x}(t) &= \nu \cos(\phi(t)) + y(t) \\
\dot{y}(t) &= \nu \sin(\phi(t))
\end{align*}
\]

where \(\phi\) is the angle between the boat direction and bank. We want to determine the movement angle \(\phi(t)\) so that \(x(T)\) is maximized, where \(T\) is a fixed transfer time. Use PROPT to solve this optimal control problem by complementing the file `mainZermeloPropt.m`, i.e., replace all question marks ??? with some proper code.

8.9 Second-order system. Solve the problem in (7.2) by using PROPT.
Hints
This version: September 2015
1 Discrete Optimization

1.2 Note that the control constraint set and the system equation give lower and upper bounds on $x_k$. Also note that both $u$ and $x$ are integers, and because of that, you can use tables to examine different cases, where each row corresponds to one valid value of $x$, i.e.,

<table>
<thead>
<tr>
<th>$x$</th>
<th>$J(k, x)$</th>
<th>$u^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>1</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
</tbody>
</table>

1.3 Define $V(n, x) := \alpha^{-n} J(n, x)$ where $J$ is the cost to go function.

1.4 Let $t_1 < t_2 < \ldots < t_{N-1}$ denote the times where $g_1(t) = g_2(t)$. It is never optimal to switch activity at any other times! We can therefore divide the problem into $N-1$ stages, where we want to determine for each stage whether or not to switch.

1.5 Consider a quadratic cost

$$V(x) = x^TPx,$$

where $P$ is a symmetric positive definite matrix.
2 Dynamic Programming

2.1 $V(t, x)$ quadratic in $x$.

2.2 What is the sign of $x(t)$? In the HJBE, how does $V(t, x)$ seem to depend on $x$?
3 PMP: A Special Case

3.4 Rewrite the equation $K(x) = l$ as a differential equation $\dot{z} = f(z, x, \dot{x})$, plus constraints, and then introduce two adjoint variables corresponding to $x$ and $z$, respectively.

3.6 a) The generalized force is $u(t)$.

3.8 a) You can consider $\begin{bmatrix} u \\ w \end{bmatrix}$ as a “control signal” in the PMP framework.
5 Infinite Horizon Optimal Control

5.2 Make the ansatz \( V(x) = \alpha x^2 + \beta x^4 \)

5.3 Use Theorem 10 in the lecture notes.
   Note that the solution matrix \( P \) is symmetric and positive definite, thus \( P = P^T > 0 \).
6 Model Predictive Control

6.1 Useful commands: mpt_control, mpt_simplify, mpt_plotPartition, mpt_plotTimeTrajectory and lsqnonlin.
7 Numerical Algorithms

7.1 The structure of the matrix $A_{eq}$ is

$$A_{eq} = \begin{pmatrix}
    E & 0 & 0 & 0 & \ldots & 0 & 0 \\
    0 & 0 & 0 & 0 & \ldots & 0 & E \\
    -F & E & 0 & 0 & \ldots & 0 & 0 \\
    0 & -F & E & 0 & \ldots & 0 & 0 \\
    0 & 0 & -F & E & \ldots & 0 & 0 \\
    \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
    0 & 0 & 0 & 0 & \ldots & -F & E
\end{pmatrix}$$

where the size of $F$ and $E$ is $(2 \times 3)$.

You can use the function checkCeqNonlcon to check that the Jacobian of the equality constraints are similar to the numerical Jacobian. You can also compare the Jacobian to the matrix $A_{eq}$ in this example.
8 The PROPT toolbox

8.4 Use the fact that $H(y^*, u^*, \lambda) = \text{const.}$ in this case (explain why!).
Solutions
This version: September 2015
1 Discrete Optimization

1.1 (a) The discrete optimization problem is given by

\[
\begin{align*}
\text{minimize} & \quad r(x_2 - T)^2 + \sum_{k=0}^{1} u_k^2 \\
\text{subject to} & \quad x_{k+1} = (1-a)x_k + au_k, \ k = 0, 1, \\
& \quad x_0 \text{ given,} \\
& \quad u_k \in \mathbb{R}, \ k = 0, 1.
\end{align*}
\]

(b) With \( a = 1/2, T = 0 \) and \( r = 1 \) this can be cast on standard form with \( N = 2, \phi(x) = x^2, f_0(k,x,u) = u^2, \) and \( f(k,x,u) = \frac{1}{2}x + \frac{1}{2}u. \) The DP algorithm gives us:

**Stage \( k = N = 2: \)**

\[
J(2,x) = x^2.
\]

**Stage \( k = 1: \)**

\[
J(1,x) = \min_u \{u^2 + J(2,f(1,x,u))\} = \min_u \{u^2 + (\frac{1}{2}x + \frac{1}{2}u)^2\}.
\]

The minimization is done by setting the derivative, w.r.t. \( u, \) to zero, since the function is strictly convex in \( u. \) Thus, we have

\[
2u + (\frac{1}{2}x + \frac{1}{2}u) = 0
\]

which gives the control function

\[
u_1^* = \mu(1,x) = -\frac{1}{5}x.
\]

Note that we now have computed the optimal control for each possible state \( x. \) By substituting the optimal \( u_1^* \) into (1.1) we obtain

\[
J(1,x) = \frac{1}{5}x^2.
\]

(c) This can be cast on standard form with \( N = 2, \phi(x) = r(x-T)^2, f_0(k,x,u) = u^2, \) and \( f(k,x,u) = (1-a)x + au. \) The DP algorithm gives us:

**Stage \( k = N = 2: \)**

\[
J(2,x) = r(x - T)^2.
\]

**Stage \( k = 1: \)**

\[
J(1,x) = \min_u \{u^2 + J(2,f(1,x,u))\} = \min_u \{u^2 + r((1-a)x + au - T)^2\}.
\]

The minimization is done by setting the derivative, w.r.t. \( u, \) to zero, since the function is strictly convex in \( u. \) Thus, we have

\[
2u + 2ra((1-a)x + au - T) = 0
\]

which gives the control function

\[
u_1^* = \mu(1,x) = ra(T - (1-a)x) \frac{1}{1 + ra^2}.
\]

1
Note that we now have computed the optimal control for each possible state $x$. By substituting the optimal $u^*_k$ into $J(1, x)$, we obtain (after some work)

$$J(1, x) = \frac{r((1-a)x - T)^2}{1 + ra^2}. \quad (1.6)$$

Stage $k = 0$:

$$J(0, x) = \min_u \{u^2 + J(1, f(0, x, u))\} = \min_u \{u^2 + J(1, (1-a)x + au)\}. \quad (1.7)$$

Substituting $J(1, x)$ by $\phi(x)$ and minimizing by setting the derivative, w.r.t. $u$, to zero gives (after some calculations) the optimal control

$$u^*_0 = \mu(0, x) = \frac{r(1-a)a(T - (1-a)^2x)}{1 + ra^2(1 + (1-a)^2)}. \quad (1.8)$$

and the optimal cost

$$J(0, x) = \frac{r((1-a)^2x - T)^2}{1 + ra^2(1 + (1-a)^2)}. \quad (1.9)$$

Stage $k = 1$:

$$J(1, x) = \min_{u \in U(1, x)} \{f_0(1, x, u) + J(2, f(1, x, u))\} = x^2 + \min_{-x \leq u \leq x} \{u^2 + J(2, x + u)\} \quad (1.10)$$

Stage $k = N = 4$: $J(4, x) = 0$.

Stage $k = 0$:

$$J(0, x) = \min_u \{u^2 + J(1, f(0, x, u))\} = \min_u \{u^2 + J(1, (1-a)x + au)\}.$$
Stage $k = 0$: Note that $x_0 = 5!$

$$J(0, x) = \min_{u \in U(0, x)} \{ f(0, x, u) + J(1, f(0, x, u)) \} = (x = 5) = 25 + \min_{-5 \leq u \leq 0} \{ u^2 + J(1, 5 + u) \} = 41 \quad (1.12)$$

given by $u^* = -3$ since the expression inside the brackets $u^2 + J(1, 5 + u)$ are, for $x = 5$ and different $u$,

<table>
<thead>
<tr>
<th>$x$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2+1</td>
<td>4+6</td>
<td>9+14</td>
<td>16+24</td>
<td>25+38</td>
</tr>
<tr>
<td>1</td>
<td>1+6</td>
<td>4+14</td>
<td>9+24</td>
<td>16+38</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1+14</td>
<td>4+24</td>
<td>9+38</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1+24</td>
<td>4+38</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1+38</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

To summarize; the system evolves as follows:

<table>
<thead>
<tr>
<th>$k$</th>
<th>$x_k$</th>
<th>$u_k$</th>
<th>$J_k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5</td>
<td>-3</td>
<td>41</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>-1</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>-1 or 0</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>0 or 1</td>
<td>0</td>
<td>0 or 1</td>
</tr>
</tbody>
</table>

1.3 Apply the general DP algorithm to the given problem:

$$J(N, x) = \alpha^N \Phi(x) \Rightarrow \alpha^{-N} J(N, x) = \Phi(x)$$

$$J(n, x) = \min_{u \in U(n, x)} \{ \alpha^n f_0(n, x, u) + J(n + 1, f(n, x, u)) \} \Rightarrow \alpha^{-n} J(n, x) = \min_{u \in U(n, x)} \{ f_0(n, x, u) + \alpha^{-n} J(n + 1, f(n, x, u)) \} \Rightarrow \alpha^{-n} J(n, x) = \min_{u \in U(n, x)} \{ f_0(n, x, u) + \alpha \cdot \alpha^{-(n+1)} J(n + 1, f(n, x, u)) \} \Rightarrow$$

In general, defining $V(n, x) := \alpha^{-n} J(n, x)$ yields

$$V(N, x) = \Phi(x),$$

$$V(n, x) = \min_{u \in U(n, x)} \{ f_0(n, x, u) + \alpha V(n + 1, f(n, x, u)) \},$$

1.4 Define

$$x_k = \begin{cases} 0 & \text{if on activity } g_1 \text{ just before time } t_k \\ 1 & \text{if on activity } g_2 \text{ just before time } t_k \end{cases}$$

$$u_k = \begin{cases} 0 & \text{continue current activity} \\ 1 & \text{switch between activities} \end{cases}$$

The state transition is

$$x_{k+1} = f(x_k, u_k) = (x_k + u_k) \mod 2$$

and the profit for stage $k$ is

$$f_0(x, u) = \int_{t_k}^{t_{k+1}} g_1 + f(x, u)(t) dt - u_k c.$$
1.5 The Bellman equation is given by
\[ V(x) = \min_u \{ f_0(x, u) + V(f(x, u)) \} \quad (1.13) \]
where
\[ f_0(x, u) = x^T Q x + u^T R u, \]
\[ f(x, u) = A x + B u. \]
Assume that the cost is on the form \( V = x^T P x, \ P > 0 \). Then the optimal control law is obtained by minimizing \( f(x, u) \) and setting the derivative to zero, thus
\[ \mu(x) = \arg \min_u \{ f_0(x, u) + V(f(x, u)) \} = \arg \min_u \{ x^T Q x + u^T R u + (A x + B u)^T P (A x + B u) \} = \arg \min_u \{ x^T Q x + u^T R u + x^T A^T P A x + u^T B^T P B u + 2 x^T A^T P B u \}. \]
The stationary point of the expression inside the brackets are obtained by
setting the derivative to zero, thus
\[ 2 R u^* + 2 B^T P B u^* + 2 A^T P A x = 0 \iff u^* = -(B^T P B + R)^{-1} B^T P A x, \]
which is a minimum since \( R > 0 \) and \( P > 0 \). Now the Bellman equation \( (1.13) \) can be expressed as
\[ x^T P x = x^T Q x + u^T (R u^* + B^T P B u^* + B^T P A x) + x^T A^T P A x + x^T A^T P B u^* \]
\[ = x^T Q x + x^T A^T P A x - x^T A^T P B (B^T P B + R)^{-1} B^T P A x \]
which holds for all \( x \) if and only if the DARE
\[ P = A^T P A - A^T P B (B^T P B + R)^{-1} B^T P A + Q, \quad (1.14) \]
holds. Optimal feedback control and globally convergent closed loop system are guaranteed by Theorem 2, assuming that there is a positive definite solution \( P \) to \( (1.14) \).

1.6 (a) 1. The Hamiltonian is
\[ H(k, x, u, \lambda) = f_0(k, x, u) + \lambda^T f(k, x, u) \]
\[ = x^T Q x + u^T R u + \lambda^T (A x + B u). \]

2. Pointwise minimization yields
\[ 0 = \frac{\partial H}{\partial u}(k, x_k, u_k, \lambda_{k+1}) = 2 R u_k + B^T \lambda_{k+1} = 0 \implies u_k^* = -\frac{1}{2} R^{-1} B^T \lambda_{k+1}. \]

3. The adjoint equations are
\[ \lambda_k = \frac{\partial H}{\partial x_k}(x_k^*, u_k^*, \lambda_{k+1}) = 2 Q x_k^* + A^T \lambda_{k+1}, \ k = 1, \ldots, N - 1 \]
\[ \lambda_N = \frac{\partial \phi}{\partial x}(x_N) = 2 Q_N x_N. \]
Hence, we obtain
\[ x_{k+1}^* = A x_k^* - 1 \frac{1}{2} B R^{-1} B^T \lambda_{k+1}. \]
Thus, the minimizing control sequence is \( (1.15) \) where \( \lambda_{k+1} \) is the solution to the TPBVP in \( (1.16) \) and \( (1.17) \).

(b) Assume that \( S_k \) is invertible for all \( k \). Using
\[ \lambda_k = 2 S_k x_k \iff x_k = \frac{1}{2} S_k^{-1} \lambda_k \]
in \( (1.17) \) yields
\[ \frac{1}{2} S_{k+1}^{-1} \lambda_{k+1} = A x_k - \frac{1}{2} B R^{-1} B^T \lambda_{k+1} \iff \lambda_{k+1} = 2 (S_{k+1}^{-1} + B R^{-1} B^T)^{-1} A x_k. \]
Insert this result in the adjoint equation \( (1.16) \)
\[ \lambda_k = 2 S_k x_k = 2 Q x_k + 2 A^T (S_{k+1}^{-1} + B R^{-1} B^T)^{-1} A x_k, \]
\[ S_N x_N = Q_N x_N \]
and since these equations must hold for all \( x_k \) we see that the backwards recursion for \( S_k \) is
\[ S_N = Q_N \]
\[ S_k = A^T (S_{k+1}^{-1} + B R^{-1} B^T)^{-1} A + Q \]
(c) Now, the MIL yields

\[
S_k = A^T(S_{k+1}^{-1} + BR^{-1}B^T)^{-1}A + Q
\]

\[
= A^T(S_{k+1} - S_{k+1}B(R + B^T S_{k+1}B)^{-1}B^T S_{k+1})A + Q
\]

\[
= A^T S_{k+1} A - A^T S_{k+1} B(R + B^T S_{k+1}B)^{-1}B^T S_{k+1} A + Q
\]

This recursion converges to the DARE in (1.14) as \(k \to \infty\) if \((A,B)\) is controllable and \((A,C)\) is observable where \(Q = C^T C\).

(d) By combining (1.15) and (1.18) we get

\[
u_k^* = -R^{-1}B^T(S_{k+1}^{-1} + BR^{-1}B^T)^{-1}Ax_k
\]

(1.19)

Another matrix identity also referred to as the MIL is

\[
BV(A + UBV)^{-1} = (B^{-1} + VA^{-1}U)VA^{-1}
\]

with this version of the MIL we get

\[
u_k^* = -(R + B^T S_{k+1}B)^{-1}B^T S_{k+1} A x_k
\]

(1.20)

which converges to the optimal feedback for infinite time horizon LQ as \(k \to \infty\).
2 Dynamic Programming

2.1 1. The Hamiltonian is given by
\[ H(t, x, u, \lambda) \triangleq f_0(t, x, u) + \lambda^T f(t, x, u) \]
\[ = (x - \cos t)^2 + u^2 + \lambda u. \]

2. Point-wise optimization yields
\[ \tilde{\mu}(t, x, \lambda) \triangleq \arg \min_{u \in \mathbb{R}} H(t, x, u, \lambda) \]
\[ = \arg \min_{u \in \mathbb{R}} \{ (x - \cos t)^2 + u^2 + \lambda u \} = -\frac{\lambda}{2}, \]
and the optimal control is
\[ u^*(t) \triangleq \tilde{\mu}(t, x(t), V_z(t, x(t))) = -\frac{1}{2}V_x(t, x(t)), \]
where \( V_z(t, x(t)) \) is obtained by solving the HJBE.

3. The HJBE is given by
\[ -V_t = H(t, x, \tilde{\mu}(t, x, V_z), V_z), \quad V(t_f, x) = \phi(x), \]
where \( V_t \triangleq \partial V/\partial t \) and \( V_z \triangleq \partial V/\partial x \). In our case, this is equivalent to
\[ V_t + (x - \cos t)^2 - V_z^2/4 = 0, \quad V(t_f, x) = 0, \]
which is a partial differential equation (PDE) in \( t \) and \( x \). These equations are quite difficult to solve directly and one often needs to utilize the structure of the problem. Since the original cost-function
\[ f_0(x(t), u(t)) \triangleq (x(t) - \cos t)^2 + u^2(t), \]
is quadratic in \( x(t) \), we make the guess
\[ V(t, x) \triangleq P(t)x^2 + 2q(t)x + c(t). \]
This yields
\[ V_t = \dot{P}(t)x^2 + 2\dot{q}(t)x + \dot{c}(t), \]
\[ V_z = 2P(t)x + 2q(t). \]
Now, substituting these expressions into the HJBE, we get
\[ \dot{p}(t)x^2 + 2\dot{q}(t)x + \dot{c}(t) + (x - \cos t)^2 - \frac{1}{4}P(t)x^2 + 2q(t)x = 0, \]
which can be rewritten as
\[ (\dot{P}(t) + 1 - P(t)^2)x^2 + 2(\dot{q}(t) - \cos t - P(t)q(t))x + \dot{c}(t) + \cos^2 t - q^2(t) = 0. \]
Since this equation must hold for all \( x \), the coefficients needs to be zero, i.e.,
\[ \dot{P}(t) = -1 + P^2(t), \]
\[ \dot{q}(t) = \cos t + P(t)q(t), \]
\[ \dot{c}(t) = -\cos^2 t + q^2(t), \]
which is the sought system of ODEs. The boundary conditions are given by
\[ P(t_f) = q(t_f) = c(t_f) = 0, \]
which come from \( V(t_f, x) = 0 \).

2.2 (a) The problem is a maximization problem and can be formulated as
\[
\begin{align*}
\text{minimize} & \quad x(t_f) - \int_0^{t_f} (1 - u(t))x(t) \, dt \\
\text{subject to} & \quad \dot{x}(t) = \alpha u(t)x(t), \\
& \quad x(0) = x_0 > 0, \\
& \quad u(t) \in [0, 1], \forall t
\end{align*}
\]
(b) 1. The Hamiltonian is given by
\[ H(t, x, u, \lambda) \triangleq f_0(t, x, u) + \lambda^T f(t, x, u) \]
\[ = -\lambda(x - \cos t) + \lambda \alpha u x. \]
2. Pointwise optimization yields

$$\bar{\mu}(t, x, \lambda) \triangleq \arg \min_{u \in [0,1]} H(t, x, u, \lambda)$$

$$= \arg \min_{u \in [0,1]} \{- (1 - u)x + \lambda \alpha u x\}$$

$$= \arg \min_{u \in [0,1]} \{(1 + \lambda)ux\}$$

$$= \begin{cases} 
1, & (1 + \lambda)x < 0 \\
0, & (1 + \lambda)x > 0 \\
\bar{u}, & (1 + \lambda)x = 0 
\end{cases}$$

where $\bar{u}$ is arbitrary in $[0, 1]$. Noting that $x(t) > 0$ for all $t \geq 0$, we find that

$$\bar{\mu}(t, x, \lambda) = \begin{cases} 
1, & \alpha \lambda < -1 \\
0, & \alpha \lambda > -1 \\
\bar{u}, & \alpha \lambda = -1
\end{cases} \quad (2.1)$$

3. The HJB is given by

$$-V_t = H(t, x, \bar{\mu}(t, x, V_x), V_x), \quad V(t_f, x) = \phi(x),$$

where $V_t \triangleq \partial V / \partial t$ and $V_x \triangleq \partial V / \partial x$. In our case, this is may be written as

$$V_t + -(1 - \bar{\mu}(t, x, V_x))x + V_x \alpha \bar{\mu}(t, x, V_x)x = 0.$$ 

By collecting $\bar{\mu}(t, x, V_x)$ this may be written as

$$V_t + (1 + \alpha V_x)x \bar{\mu}(t, x, V_x) - x = 0,$$

and inserting (2.1) yields

$$\begin{cases} 
V_t + \alpha V_x x = 0, & \alpha V_x x < -1 \\
V_t - x = 0, & \alpha V_x x > -1
\end{cases} \quad (2.2)$$

with the boundary condition $V(t_f, x) = \phi(x) = -x$. Both the equations above are linear in $x$, which suggests that $V(t, x) = g(t)x$ for some function $g(t)$. Substituting this into (2.2) yields

$$\begin{cases} 
\dot{g}(t)x + \alpha g(t)x = 0, & g(t) < -1/\alpha \\
\dot{g}(t)x - x = 0, & g(t) > -1/\alpha
\end{cases}$$

These equations are valid for every $x > 0$ and can thus cancel the $x$ term, which yields

$$\begin{cases} 
\dot{g}(t) + \alpha g(t) = 0, & g(t) < -1/\alpha \\
\dot{g}(t) - 1 = 0, & g(t) > -1/\alpha
\end{cases}$$

The solutions to the equations above are

$$\begin{cases} 
g_1(t) = c_1 e^{-\alpha(t-t')}, & g(t) < -1/\alpha \\
g_2(t) = t + c_2, & g(t) > -1/\alpha
\end{cases}$$

for some constants $c_1, c_2$ and $t'$. The boundary value is

$$V(t_f, x) = -x \quad \Rightarrow \quad g(t_f) = -1 > -1/\alpha \quad (\text{since } \alpha \in (0, 1)).$$

This gives $g_2(t_f) = t + c_2$ in the end. From $g_2(t_f) = -1$, we obtain $c_2 = -1 - t_f$ and

$$g(t) = t - 1 - t_f,$$

in some interval $(t', t_f)$. Here, $t'$ fulfills $g_2(t) = -1/\alpha$, and hence

$$t' = 1 - 1/\alpha + t_f,$$

which is the same time at which $g$ switches. Now, use this as a boundary value for the solution to the other PDE $g_1(t) = c_1 e^{-\alpha(t-t')}$:

$$g_1(t') = -1/\alpha \quad \Rightarrow \quad c_1 = -1/\alpha.$$ 

Thus, we have

$$g(t) = \begin{cases} 
-\frac{1}{\alpha} e^{-\alpha(t-t')}, & t < t' \\
(t - 1 - t_f) & t > t'
\end{cases}$$

where $t' = 1 - 1/\alpha + t_f$. The optimal cost function is given by

$$V(t, x) = \begin{cases} 
-\frac{1}{\alpha} e^{-\alpha(t-t')}x, & t < t' \quad (u = 0) \\
(t - 1 - t_f) x & t > t' \quad (u = 1)
\end{cases}$$

where $t' = 1 - 1/\alpha + t_f$. Thus, the optimal control strategy is

$$u'(t) = \bar{\mu}(t, x(t), V_x(t, x(t))) = \begin{cases} 
1, & t < t' \\
0, & t > t'
\end{cases}$$

and we are done. Not also that $V_x$ and $V_t$ are continuous at the switch boundary, thus fulfilling the $C^1$ condition on $V(t, x)$ of the verification theorem.
3 PMP: A Special Case

3.1 The Hamiltonian is given by

\[
H(t, x, u, \lambda) = f_0(t, x, u) + \lambda^T f(t, x, u) = x + u^2 + \lambda x + \lambda u + \lambda.
\]

Pointwise minimization yields

\[
\tilde{\mu}(t, x, \lambda) = \arg \min_u H(t, x, u, \lambda) = -\frac{1}{2} \lambda,
\]

and the optimal control can be written as

\[
u^*(t) = \tilde{\mu}(t, x(t), \lambda(t)) = -\frac{1}{2} \lambda(t).
\]

The adjoint equation is given by

\[
\dot{\lambda}(t) = -\lambda(t) - 1,
\]

which is a first order linear ODE and can thus be solved easily. The standard integrating factor method yields

\[
\lambda(t) = e^{C-t} - 1,
\]

for some constant \( C \). The boundary condition is given by

\[
\lambda(t_f) = \frac{\partial \phi}{\partial x}(t_f, x(t_f)) \iff \lambda(t_f) = 0,
\]

Thus,

\[
\lambda(t) = e^{t-r-t} - 1,
\]

and the optimal control is found by

\[
u^*(t) = -\frac{1}{2} \lambda(t) = \frac{1 - e^{t-r-t}}{2}.
\]

3.2 (a) Since

\[
J = \int_0^1 \dot{y} dt = y(1) - y(0) = 1,
\]

it holds that all \( y(\cdot) \in C^1[0, 1] \) such that \( y(0) = 0 \) and \( y(1) = 1 \) are minimal.

(b) Since

\[
J = \int_0^1 \dot{y} dt = \frac{1}{2} \int_0^1 \frac{d}{dt}(y^2) dt = \frac{1}{2} (y(1)^2 - y(0)^2) = \frac{1}{2},
\]

it holds that all \( y(\cdot) \in C^1[0, 1] \) such that \( y(0) = 0 \) and \( y(1) = 1 \) are minimal.

3.3 (a) Define \( x = y, u = \dot{y} \) and \( f_0(t, x, u) = x^2 + u^2 - 2x \sin t \), then the Hamiltonian is given by

\[
H(t, x, u, \lambda) = x^2 + u^2 - 2x \sin t + \lambda u
\]

The following equations must hold

\[
y(0) = 0 \quad (3.1a)
\]

\[
0 = \frac{\partial H}{\partial u}(t, x, u, \lambda) = 2u + \lambda \quad (3.1b)
\]

\[
\dot{\lambda} = -\frac{\partial H}{\partial x}(t, x, u, \lambda) = -2x + 2 \sin t \quad (3.1c)
\]

\[
\lambda(1) = 0 \quad (3.1d)
\]

Equation (3.1b) gives \( \dot{\lambda} = -2\dot{u} = -2\ddot{y} \) and with (3.1c) we have

\[
\ddot{y} - y = -\sin t
\]

with the solution

\[
y(t) = c_1 e^t + c_2 e^{-t} + \frac{1}{2} \sin t, \quad c_1, c_2 \in \mathbb{R}
\]

where (3.1a) gives the relationship of \( c_1 \) and \( c_2 \) as

\[
c_2 = -c_1
\]

and where (3.1d) gives

\[
c_2 = c_1 e^2 + \frac{e}{2} \cos t.
\]

Consequently, we have

\[
y(t) = \frac{e \cos \frac{1}{2}(e^t - e^{-t}) + \frac{1}{2} \sin t,}{2(e^2 + 1)}
\]
Define $x = y$, $u = y$ and $f_0(t, x, u) = u^2/t^3$, then the Hamiltonian is given by

$$H(t, x, u, \lambda) = u^2 t^{-3} + \lambda u.$$ 

The following equations must hold

1. $y(0) = 0$  
2. $0 = \frac{\partial H}{\partial u}(t, x, u, \lambda) = 2u t^{-3} + \lambda$  
3. $\dot{\lambda} = -\frac{\partial H}{\partial x}(t, x, u, \lambda) = 0$  
4. $\lambda(1) = 0$

Equations (3.2c) gives $\lambda = c_1, c_1 \in \mathbb{R}$, and (3.2d) that $c_1 = 0$. Then (3.2b) gives $u(t) = 0$ which implies $y(t) = c_2, c_2 \in \mathbb{R}$. Finally, with (3.2a) this gives $y(t) = 0$

(c) Analogous to (a). The system equation is

$$\ddot{y} - y = e^t$$

with the solution

$$y(t) = c_1 e^t + c_2 e^{-t} + \frac{1}{2} t e^t$$

where

$$c_2 = -c_1$$

and where

$$c_2 = e^2(c_1 + 1)$$

which gives

$$y(t) = \frac{-e^2}{c^2 + 1} (e^t - e^{-t}) + \frac{1}{2} t e^t$$

3.4 Define

$$z(t) = \int_{-a}^t \sqrt{1 + \dot{x}(s)^2} \, ds.$$
where \( c_3 \) is constant, and this can be rewritten as the equation of a circle with the radius \( c_2 \) and the center in \((t, x(t)) = (c_1, c_3)\), i.e.,
\[
(x(t) - c_3)^2 + (t + c_1)^2 = c_2^2.
\]

The requirements \( x(-a) = 0 \) and \( x(a) = 0 \) gives
\[
c_3^2 + (a - c_1)^2 = c_2^2
\]
\[
c_3^2 + (a + c_1)^2 = c_2^2,
\]
respectively. Thus, \( c_1 = 0 \) and \( c_3^2 + a^2 = c_2^2 \). We have three different cases

- \( l < 2a \): No solution since the distance between \((-a, 0)\) and \((a, 0)\) is longer than the length of the curve \( l \).
- \( 2a \leq l \leq \pi a \): Let \( \phi \) define the angle from the \( x(t) \)-axis to the vector from the circle center to the point \((a, 0)\), i.e, \( a = c_2 \sin \phi \), see Figure 3.4a(a). Furthermore, the length of the circle segment curve is \( l = 2\phi c_2 \). Thus
\[
\sin \frac{l}{2c_2} = \frac{a}{c_2}
\]
which can be used to determine \( c_2 \).
- \( \pi a < l \): as the previous case, but the length of the circle segment curve is now \( l = 2(\pi - \phi)c_2 \), see Figure 3.4a(b).

3.5 By introducing
\[
x_1 = x, \quad x_2 = y, \quad u = \phi \quad \text{and redefining} \quad x \triangleq (x_1, x_2)^T\]
\[
\phi(T, x(T)) = -x_1(T),
\]
the problem at hand can be written on standard form as

\[
\begin{align*}
\text{minimize} & \quad -x_1(T) \\
\text{subject to} & \quad \dot{x}_1(t) = \nu \cos (u(t)) + g(x_2(t)), \\
& \quad \dot{x}_2(t) = \nu \sin(u(t)), \\
& \quad x_1(0) = 0, \quad x_2(0) = 0, \\
& \quad u(t) \in \mathbb{R}, \quad \forall t \in [0, T].
\end{align*}
\]

The Hamiltonian is given by
\[
H(t, x, u, \lambda) \triangleq f_0(t, x, u) + \lambda^T f(t, x, u) \\
= \lambda_1(\nu \cos (u) + g(x_2)) + \lambda_2 \nu \sin(u) \\
= \lambda_1 g(x_2) + \lambda_1 \nu \cos(u) + \lambda_2 \nu \sin(u).
\]

Pointwise optimization yields
\[
\tilde{\mu}(t, x, \lambda) \triangleq \arg \min_u H(t, x, u, \lambda) \\
= \arg \min_u \{ \lambda_1 \nu \cos (u) + \lambda_2 \nu \sin(u) \} \\
= \arg \min_u \sqrt{\lambda_1^2 + \lambda_2^2} \sin (u + \theta) \\
\left( \sin \theta = \frac{\lambda_1}{\sqrt{\lambda_1^2 + \lambda_2^2}}, \quad \cos \theta = \frac{\lambda_2}{\sqrt{\lambda_1^2 + \lambda_2^2}} \right) \\
= -\theta - \frac{\pi}{2}.
\]
Hence,
\[\tan (u^*(t)) = \frac{\sin (-\theta(t) - \frac{\pi}{2})}{\cos (-\theta(t) - \frac{\pi}{2})} = -\cos \theta(t) = \lambda_2(t)/\lambda_1(t).\] (3.5)

Now, the adjoint equations are
\[\dot{\lambda}_1(t) = -\frac{\partial H}{\partial x_1}(x(t), u^*(t), \lambda(t)) = 0\]
\[\dot{\lambda}_2(t) = -\frac{\partial H}{\partial x_2}(x(t), u^*(t), \lambda(t)) = -g'(x_2(t))\lambda_1(t).\]

The first equation yields \(\lambda_1(t) = c_1\) for some constant \(c_1\). The boundary constraints are given by
\[\lambda(T) = \frac{\partial \phi}{\partial x}(T, x(T)) \iff \begin{pmatrix} \lambda_1(T) \\ \lambda_2(T) \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \end{pmatrix},\]
and hence \(\lambda_1(t) = -1\). Substituting this into the expression for the optimal control law (3.5) yields
\[\tan (u^*(t)) = -\lambda_2(t).\]

What remains is to take care of \(\lambda_2(t)\), for which no explicit solution can be found. To this end, we need the following general result:
A system is said to be autonomous if it does not have a direct dependence on the time variable \(t\), i.e., \(\dot{x}(t) = f(x(t), u(t))\). On the optimal trajectory, the Hamiltonian for autonomous system has the property
\[H(x(t), u^*(t), \lambda(t)) = \begin{cases} 0, & \text{if the final time is free,} \\ \text{constant} & \forall t, \text{if the final time is fixed,} \end{cases}\]
see Chapter 6 in the course compendium for details.

In this example the final time is fixed. Thus, the constant \(= H(x(t), u^*(t), \lambda(t))\)
\[= \lambda_1(t) \left( \nu \cos u^*(t) + g(x_2(t)) \right) + \lambda_2(t) \nu \sin u^*(t)\]
\[= -\nu \cos u^*(t) - g(x_2(t)) - \tan u^*(t)\]
\[= -g(x_2(t)) - \nu (\cos u^*(t) + \tan u^*(t) \sin u^*(t))\]
\[= -g(x_2(t)) - \frac{\nu}{\cos u^*(t)},\]
and by dividing the above relation with \(-\nu\) the desired result is attained.

3.6 (a) 1. Firstly, let us derive the Euler-Lagrange equations. Adjoining the constraint \(\delta q = \delta t\) to the cost function yields
\[\int_{t_i}^{t_f} \left( \frac{\partial L}{\partial u} \delta u + \frac{\partial L}{\partial q} + \tau^T \delta q + \lambda^T (\delta u - \delta q) \right) dt = 0.\]

Integrating the last term by parts yields
\[\int_{t_i}^{t_f} \left( \frac{\partial L}{\partial u} + \lambda^T \right) \delta u + \left( \frac{\partial L}{\partial q} + \tau^T + \dot{\lambda}^T \right) \delta q \right) dt = 0.\]

To make the entire integral vanish, we choose
\[\dot{\lambda}^T = -\frac{\partial L}{\partial q} - \tau^T,\]
\[\lambda^T = -\frac{\partial L}{\partial u}.\]

Combining the above, we have
\[\frac{d}{dt} \left( \frac{\partial L}{\partial q} \right) - \frac{\partial L}{\partial q} = \tau^T.\]

2. Define the generalized coordinates as \(q_1 = z\) and \(q_2 = i\), and the generalized force \(\tau_2 = u(t)\). The Lagrangian is \(L = T - V\) where the kinetic energy is the sum of the mechanical and the electrical kinetic energies
\[T = \frac{1}{2} m q_1^2 + \frac{1}{2} l (q_1)^2\]
and the potential energy is the spring energy
\[V = \frac{1}{2} k q_2^2.\]

Euler-Lagrange’s equations are
\[\frac{d}{dt} \frac{\partial L}{\partial q_1} - \frac{\partial L}{\partial q_1} = 0\]
\[\frac{d}{dt} \frac{\partial L}{\partial q_2} - \frac{\partial L}{\partial q_2} = u(t)\]
and this gives the system equations
\[ m \ddot{q}_1 + k q_1 - \frac{1}{2} \frac{\partial l(q_1)}{\partial q_1} = 0 \]
\[ l(q_1) \dot{q}_2 = u(t). \]

(b) The Rayleigh dissipative energy is
\[ F(q, \dot{q}) = \frac{1}{2} d \dot{q}_1^2 + \frac{1}{2} R \dot{q}_2^2. \]

Euler-Lagrange's equations of motion is
\[
\frac{d}{dt} \frac{\partial L}{\partial \dot{q}_1} - \frac{\partial L}{\partial q_1} = 0 - \frac{\partial F}{\partial q_1}, \\
\frac{d}{dt} \frac{\partial L}{\partial \dot{q}_2} - \frac{\partial L}{\partial q_2} = u(t) - \frac{\partial F}{\partial \dot{q}_2}
\]

and this gives the system equations
\[ m \ddot{q}_1 + d \dot{q}_1 + k q_1 - \frac{1}{2} \frac{\partial l(q_1)}{\partial q_1} = 0 \]
\[ l(q_1) \dot{q}_2 + R \dot{q}_2 = u(t). \]

3.7 The Hamiltonian is given by
\[ H(t, x, u, \lambda) \triangleq f_0(t, x, u) + \lambda^T f(t, x, u) \]
\[ = \frac{1}{2} u^2 + \lambda u \]

Pointwise minimization yields
\[ \bar{u}(t, x, \lambda) \triangleq \arg \min_u H(t, x, u, \lambda) = -\lambda, \]

and the optimal control can be written as
\[ u^*(t) \triangleq \bar{u}(t, x(t), \lambda(t)) = -\lambda(t). \]

The adjoint equation is given by
\[ \dot{\lambda}(t) = -\frac{\partial H}{\partial x_1}(x(t), u^*(t), \lambda(t)) = 0, \]

which means that \( \lambda \) is constant. The boundary condition is given by
\[ \lambda(t_f) = \frac{\partial \phi}{\partial x}(x(t_f)) = \gamma x(t_f) \Rightarrow \lambda(t) = \gamma x(t_f). \]

The final state \( x(t_f) \) is not given, however it can be calculated as a function of \( x(t) \) assuming the optimal control signal \( u^*(t) = -\lambda(t) = -\gamma x(t_f) \) is applied from \( t \) to \( t_f \). This will also give us a feedback solution.
\[ x(t_f) = x(t) + \int_t^{t_f} \dot{x}(t) dt = x(t) - \int_t^{t_f} \gamma x(t_f) dt = x(t) - \gamma x(t_f)(t_f - t). \]

Thus,
\[ x(t_f) = \frac{1}{1 + \gamma(t_f - t)} x(t), \]

and the optimal control is found by
\[ u^*(t) = -\lambda(t) = -\gamma x(t) = -\frac{1}{\gamma^{-1} + (t_f - t)} x(t). \]

3.8 (a) Define
\[ u \triangleq \left( \begin{array}{c} u \\ w \end{array} \right). \]

With
\[ f_0(x, u) \triangleq \rho^2 x^2 + u^2 - \gamma^2 w \quad \text{and} \quad f(x, u) \triangleq u + w, \]

the Hamiltonian is given by
\[ H(x, u, \lambda) \triangleq f_0(x, u) + \lambda^T f(x, u) = \rho^2 x^2 + u^2 - \gamma^2 w^2 + \lambda(u + w). \]

Pointwise minimization yields
\[ 0 = \frac{\partial H}{\partial u}(x, u, \lambda) = \left( 2u + \lambda \right) = 2u + \lambda \Rightarrow \mu(x, t), \lambda(t) = \lambda(t). \]

The TPBVP is given by
\[ \dot{\lambda}(t) = -\frac{\partial H}{\partial x}(x(t), \mu(t, x(t), \lambda(t)), \lambda(t)) = -2\rho^2 x(t), \quad \lambda(T) = 0 \quad (3.6) \]
\[ \dot{x}(t) = \frac{\partial H}{\partial \lambda}(x(t), \mu(t, x(t), \lambda(t)), \lambda(t)) = \left( \frac{1}{\gamma^2 - 1} \right) \lambda(t), \quad x(0) = 0. \quad (3.7) \]
(b) Differentiation of (3.7) yields

\[ \ddot{x}(t) = (\gamma^2 - 1) \dot{\lambda}(t) - 2 \rho^2 (\gamma^2 - 1) x(t). \]

which is a second order homogeneous ODE

\[ \ddot{x}(t) + 2 \rho^2 (\gamma^2 - 1) x(t) = 0. \]

The characteristic polynomial

\[ h^2 + 2 \rho^2 (\gamma^2 - 1) = 0, \]

has solutions on the imaginary axis (since \( 0 < \gamma < 1 \)) given by

\[ h = \pm i \rho \sqrt{2(\gamma^2 - 1)}. \]

Thus, the solution is

\[ x(t) = A \sin(rt) + B \cos(rt), \]

for some constants \( A \) and \( B \), where \( r = \rho \sqrt{2(\gamma^2 - 1)} \). Now, the boundary condition \( \lambda(T) = 0 \) implies that \( \dot{x}(T) = 0 \) (see (3.7)) and thus

\[ \begin{pmatrix} \sin(rt) & \cos(rt) \\ r \cos (rT) & -r \sin (rT) \end{pmatrix} \begin{pmatrix} A \\ B \end{pmatrix} = \begin{pmatrix} x(t) \\ 0 \end{pmatrix}, \]

which has the solution

\[ \begin{pmatrix} A \\ B \end{pmatrix} = \frac{1}{\sin(rt) \sin(rT) - r \cos(rt) \cos(rT)} \begin{pmatrix} -r \sin(rt) & - \cos(rt) \\ -r \cos(rt) & -r \sin(rt) \end{pmatrix} \begin{pmatrix} x(t) \\ 0 \end{pmatrix} \]

\[ = \frac{x(t)}{\sin(rt) \sin(rT) + \cos(rt) \cos(rT)} \begin{pmatrix} \sin(rt) \\ \cos(rt) \end{pmatrix} \]

\[ = \frac{x(t)}{\cos(r(T-t))} \begin{pmatrix} \sin(rt) \\ \cos(rt) \end{pmatrix} . \]

Finally, the optimal control is given by

\[ u^*(t) = - \frac{1}{2} \lambda(t) \left[ \frac{1}{2} \frac{2}{\gamma^2 - 1} \ddot{x}(t) \right] \]

\[ = - \frac{1}{\gamma^2 - 1} \left( rA \cos(rt) - rB \sin(rt) \right) \]

\[ = - \frac{1}{\gamma^2 - 1} \left( \frac{r}{\cos(r(T-t))} \cos(rt) - \frac{\cos(rT)}{\cos(r(T-t))} \sin(rt) \right) x(t) \]

\[ = - \frac{\rho \sqrt{2}}{\gamma^2 - 1} \left( \frac{\sin(rt) \cos(rt) - \cos(rt) \sin(rt)}{\cos(r(T-t))} \right) x(t) \]

\[ = - \frac{\rho \sqrt{2}}{\gamma^2 - 1} \tan(r(T-t)) x(t). \]

(c) Since \( \tan(\cdot) \) is defined everywhere on \([0, \pi/2]\) and \( r(T-t) \leq rT \) for all \( t = \in [0, \pi/2] \), it must hold that \( rT < \pi/2 \). Thus, for

\[ \rho \sqrt{2(\gamma^2 - 1)} T < \frac{\pi}{2} \quad \Rightarrow \quad \gamma > \frac{1}{\sqrt{1 + 2 \left( \frac{\pi}{2\rho T} \right)^2}} , \]

the control signal is defined for all \( t \in [0, T] \).
4 PMP: General Results

4.1 The problem can be rewritten to standard form as

\[
\begin{align*}
& \text{minimize} \quad -x(t_f) - \int_0^T (1 - u(t))x(t) \, dt \\
& \text{subject to} \quad \dot{x}(t) = \alpha u(t)x(t), \quad 0 < \alpha < 1 \\
& \quad \quad x(0) = x_0 > 0, \\
& \quad \quad 0 \leq u(t) \leq 1, \quad \forall t \in [0, t_f].
\end{align*}
\]

The Hamiltonian is given by

\[
H(t, x, u, \lambda) \triangleq f_0(t, x, u) + \lambda^T f(t, x, u) = -(1 - u)x + \lambda \alpha x,
\]

Pointwise optimization of the Hamiltonian yields

\[
\mu(t, x, \lambda) \triangleq \arg \min_{u \in [0,1]} H(t, x, u, \lambda)
= \arg \min_{u \in [0,1]} \left\{ - (1 - u)x + \lambda \alpha u x \right\}
= \begin{cases} 
1, & (1 + \lambda \alpha)x < 0 \\
0, & (1 + \lambda \alpha)x > 0 \\
\tilde{u}, & (1 + \lambda \alpha)x = 0
\end{cases}
\]

where \(\tilde{u}\) is an arbitrary value in \([0, 1]\). To be able to find an analytical solution, it is important to remove the variable \(x\) from the pointwise optimal solution above. Otherwise we are going to end up with a PDE, which is difficult to solve. In this particular case it is simple. Since \(x_0 > 0\), \(\alpha > 0\) and \(u > 0\) in (4.1), it follows that \(x(t) > 0\) for all \(t \in [0, t_f]\). Hence, the optimal control is given by

\[
u^*(t) \triangleq \mu(t, x(t), \lambda(t)) = \begin{cases} 
1, & (1 + \lambda \alpha) < 0 \\
0, & (1 + \lambda \alpha) > 0 \\
\tilde{u}, & (1 + \lambda \alpha) = 0
\end{cases}
\]

and we define the switching function as

\[
\sigma(t) \triangleq 1 + \lambda(t) \alpha.
\]

The adjoint equations are now given by

\[
\dot{\lambda}(t) \triangleq -\frac{\partial H}{\partial x}(t, x(t), u^*(t), \lambda(t))
= (1 - u^*(t)) - \alpha \lambda(t) u^*(t)
= \begin{cases} 
-\alpha \lambda(t), & (1 + \lambda \alpha) < 0, \quad (u^*(t) = 1) \\
0, & (1 + \lambda \alpha) > 0, \quad (u^*(t) = 0) \\
1, & (1 + \lambda \alpha) = 0
\end{cases}
\]

(4.1)

The boundary constraints are

\[
\lambda(t_f) = \frac{\partial \phi}{\partial x}(t_f, x(t_f)) \perp S_f(t_f) \iff \lambda(t_f) + 1 = 0,
\]

which implies that \(\lambda(t_f) = -1\). Thus,

\[
\sigma(t_f) = 1 + \lambda(t_f) \alpha = 1 - \alpha > 0,
\]

so that \(u^*(t_f) = 0\). What remains to be determined is how many switches occurs. A hint of the number of switches can often be found by considering the value of \(\dot{\sigma}(t)|_{\sigma(t) = 0}\). From (4.1), it follows that

\[
\dot{\sigma}(t)|_{\sigma(t) = 0} = \dot{\lambda}(t) \alpha|_{1 + \lambda(t) \alpha = 0} = \alpha > 0.
\]

Hence, there can only be at most one switch, since we can pass \(\sigma(t) = 0\) only once. Since \(u^*(t_f) = 0\), it is not possible that \(u^*(t) = 1\) for all \(t \in [0, t_f]\). Thus,

\[
u^*(t) = \begin{cases} 
1, & 0 \leq t \leq t' \\
0, & t' \leq t \leq t_f
\end{cases}
\]

for some unknown switching time \(t' \in [0, t_f]\). The switching occurs when

\[
0 = \sigma(t') = 1 + \lambda(t') \alpha,
\]

and to find the value of \(t'\) we need to determine the value of \(\lambda(t')\). From (4.1), it holds that during the period \(t' \leq t \leq t_f\) where \(u^*(t) = 0\) we have that

\[
\dot{\lambda}(t) = 1, \quad \lambda(t_f) = -1
\]
4.2 The problem to be solved is

\[
\begin{align*}
\text{minimize} & \quad \int_0^T u(t) \, dt \\
\text{subject to} & \quad \dot{x}_1(t) = x_2(t), \\
& \quad \dot{x}_2(t) = \frac{c u(t)}{x_3(t)} - g(1 - k x_1(t)), \\
& \quad \dot{x}_3(t) = -u(t), \\
& \quad x_1(0) = h, \quad x_2(0) = \nu, \quad x_3(0) = m, \\
& \quad x_1(t_f) = 0, \quad x_2(t_f) = 0, \quad 0 \leq u(t) \leq M.
\end{align*}
\]

The Hamiltonian is given by

\[
H(t, x, u, \lambda) \triangleq f_0(t, x, u) + \lambda^T f(t, x, u) \\
= u + \lambda_1 x_2 + \frac{c u}{x_3 - g(1 - k x_1)} - \lambda_3 u \\
= \lambda_1 x_2 - \lambda_2 g(1 - k x_1) + \left(1 + \frac{c \lambda_2}{x_3} - \lambda_3\right) u.
\]

Pointwise minimization yields

\[
\tilde{\nu}(t, x, \lambda) \triangleq \arg \min_{u \in [0, M]} H(t, x, u, \lambda) \\
= \arg \min_{0 \leq u \leq M} \left(1 + \frac{c \lambda_2}{x_3} - \lambda_3\right) u \\
= \begin{cases} 
M, & \sigma < 0 \\
0, & \sigma > 0 \ , \\
\tilde{u}, & \sigma = 0
\end{cases}
\]

where \(\tilde{u} \in [0, M]\) is arbitrary. Thus, the optimal control is expressed by

\[
u^*(t) \triangleq \tilde{\nu}(t, x(t), \lambda(t)) = \begin{cases} 
M, & \sigma(t) < 0 \\
0, & \sigma(t) > 0 \ , \\
\tilde{u}, & \sigma(t) = 0
\end{cases}
\]

where the switching function is given by

\[
\sigma(t) \triangleq 1 + \frac{c \lambda_2(t)}{x_3(t)} - \lambda_3(t).
\]

The adjoint equations are

\[
\dot{\lambda}_1(t) = -\frac{\partial H}{\partial x_1}(x(t), u^*(t), \lambda(t)) = -g k \lambda_2(t), \\
\dot{\lambda}_2(t) = -\frac{\partial H}{\partial x_2}(x(t), u^*(t), \lambda(t)) = -\lambda_1(t), \\
\dot{\lambda}_3(t) = -\frac{\partial H}{\partial x_3}(x(t), u^*(t), \lambda(t)) = \frac{c \lambda_2(t) u^*(t)}{x_3^2(t)}.
\]

The boundary conditions are given by

\[
\lambda(t_f) - \frac{\partial \phi}{\partial x}(t_f, x(t_f)) \perp S_f(t_f) \iff \begin{pmatrix} \lambda_1(t_f) \\ \lambda_2(t_f) \\ \lambda_3(t_f) \end{pmatrix} = \nu_1 \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + \nu_2 \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix},
\]

which says that \(\lambda_3(t_f) = 0\), while \(\lambda_1(t_f)\) and \(\lambda_2(t_f)\) are free. This, yields no particular information about the solution to the adjoint equations.
Now, let us try to determine the number of switches. Using the state dynamics and the adjoint equations, it follows that
\[
\hat{\sigma}(t) = \frac{c\lambda_2(t)x_3(t) - c\lambda_2(t)x_3(t)}{x_3^2(t)} - \lambda_3(t)
\]
\[
= -c\lambda_1(t)x_3(t) + c\lambda_2(t)u^*(t) - \frac{c\lambda_2(t)u^*(t)}{x_3^2(t)}
\]
\[
= -c\frac{\lambda_1(t)}{x_3(t)}.
\]
Since \(c > 0\) and \(x_3(t) \geq 0\), it holds that \(\text{sign} \hat{\sigma}(t) = -\text{sign} \lambda_1(t)\) and we need to know which values \(\lambda_1(t)\) can take. From the adjoint equations, we have
\[
\lambda_1(t) = -gk\lambda_2(t) = gk\lambda_1(t),
\]
which has the solution
\[
\lambda_1(t) = Ae^\sqrt{gkt} + Be^-\sqrt{gkt}.
\]
Now, if \(A\) and \(B\) have the same signs, \(\lambda_1(t)\) will never reach zero. If \(A\) and \(B\) have opposite signs, \(\lambda_1(t)\) has one isolated zero. This implies that \(\hat{\sigma}(t)\) has at most one isolated zero. Thus, \(\sigma(t)\) can only, at the most, pass zero two times and the optimal control is bang-bang. The only possible sequences are thus \(\{M, 0, M\}\), \(\{0, M\}\) and \(\{M\}\) (since the spacecraft should be brought to rest, all sequences ending with a 0 are ruled out).

4.3 By introducing \(x_1 = x, x_2 = y\) and redefining \(x \triangleq (x_1, x_2)^T\), the problem at hand can be written on standard form as
\[
\text{minimize } u(\cdot)
\]
\[
\text{subject to } \begin{align*}
\dot{x}_1(t) &= (au(t) - b)x_1(t), \\
\dot{x}_2(t) &= c(1 - u(t))x_1(t), \\
x_1(0) &= 1, \ x_2(0) = 0, \\
u(t) &\in [0, 1], \ \forall t \in [0, t_f].
\end{align*}
\]
The Hamiltonian is given by
\[
H(t, x, u, \lambda) \triangleq f_0(t, x, u) + \lambda^T f(t, x, u)
\]
\[
= \lambda_1(au - b)x_1 + \lambda_2c(1 - u)x_1
\]
\[
= -\lambda_1bx_1 + \lambda_2c + (a\lambda_1 - c\lambda_2)x_1u.
\]
Pointwise minimization yields
\[
\tilde{\mu}(t, x, \lambda) \triangleq \arg \min H(t, x, u, \lambda)
\]
\[
= \begin{cases}
1, & \text{if } (a\lambda_1 - c\lambda_2)x_1 < 0 \\
0, & \text{if } (a\lambda_1 - c\lambda_2)x_1 > 0, \\
\tilde{u}, & \text{otherwise}
\end{cases}
\]
where \(\tilde{u} \in [0, M]\) is arbitrary. Now, since the number of worker bees \(x_1(t)\) must be positive, we define the switching function as
\[
\sigma(t) \triangleq a\lambda_1(t) - c\lambda_2(t).
\]
Thus, the optimal control is expressed by
\[
u^*(t) \triangleq \tilde{\mu}(t, x(t), \lambda(t)) = \begin{cases}
1, & \sigma(t) < 0 \\
0, & \sigma(t) > 0, \\
\tilde{u}, & \sigma(t) = 0
\end{cases}
\]
The adjoint equations are
\[
\dot{\lambda}_1(t) = -\frac{\partial H}{\partial x_1}(x(t), u^*(t), \lambda(t))
\]
\[
= -\lambda_1(t)(au^*(t) - b) - \lambda_2(t)c(1 - u^*(t)),
\]
\[
\dot{\lambda}_2(t) = -\frac{\partial H}{\partial x_2}(x(t), u^*(t), \lambda(t)) = 0.
\]
The boundary conditions are given by
\[
\lambda(t_f) - \frac{\partial \phi}{\partial x}(t_f, x(t_f)) \perp S_f(t_f) \iff \begin{pmatrix} \lambda_1(t_f) \\ \lambda_2(t_f) + 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix},
\]
which yields \(\lambda_1(t_f) = 0\) and \(\lambda_2(t_f) = -1\). Substituting this into the adjoint equations implies that \(\lambda_2(t) = -1\) and thus
\[
\sigma(t) = a\lambda_1(t) + c.
\]
This yields
\[
\sigma(t_f) = a\lambda_1(t_f) + c = c > 0,
\]
where \(c > 0\).
so that $u^*(t_f) = 0$. To get the number of switches, consider
\[
\dot{\sigma}(t) = a\dot{\lambda}_1(t) = a\left( -\lambda_1(t)(au^*(t) - b) - \lambda_2(t) c(1 - u^*(t)) \right)
\]
\[
= a\left( -\lambda_1(t)(au^*(t) - b) + c(1 - u^*(t)) \right)
\]
which implies that
\[
\dot{\sigma}(t)|_{\sigma(t)=0} = \dot{\sigma}(t)|_{\lambda_1(t)=-c/a} = a\left( \frac{c}{a}(au^*(t) - b) + c(1 - u^*(t)) \right) = c(a - b).
\]
We have three cases:

i) $a < b : \sigma(t)$ crosses zero from positive to negative at most once. Since we have already shown that $\sigma(t_f) > 0$, it follows that no switch is present and thus $u(t) = 0$ for all $t \in [0, t_f]$ (since $u(t_f) = 0$).

ii) $a > b : \sigma(t)$ crosses zero from negative to positive at most once. Thus
\[
u^*(t) = \begin{cases} 1, & 0 \leq t \leq t' \\ 0, & t' < t \leq t_f \end{cases}
\]
for some unknown switching time $t'$. To find $t'$, we assume that $\sigma(t) > 0$. Then $t \in [t', t_f], u^*(t) = 0$ and the adjoint equation can be simplified to
\[
\dot{\lambda}_1(t) = -\lambda_1(t)(au^*(t) - b) - \lambda_2(t) c(1 - u^*(t)) = b\lambda_1(t) + c,
\]
with the boundary constraint $\lambda_1(t_f) = 0$. This first order boundary value problem has the solution
\[
\lambda_1(t) = \frac{c}{b}(e^{b(t-t_f)} - 1).
\]
For $t = t'$, we have $\sigma(t') = 0$ and thus
\[
\frac{c}{a} = \lambda_1(t') = \frac{c}{b}(e^{b(t'-t_f)} - 1),
\]
which can be rewritten as
\[
t' = t_f + \frac{1}{b} \log \left( 1 - \frac{b}{a} \right).
\]

iii) $a = b :$ We cannot say anything about the number of switches. Solve
\[
\begin{align*}
\dot{\lambda}_1(t) &= a\lambda_1(t) + c, \\
\lambda_1(t_f) &= 0 \\
\Rightarrow \quad \lambda_1(t) &= \frac{c}{a} \left( -1 + e^{a(t-t_f)} \right) \neq -\frac{c}{a}
\end{align*}
\]
Hence, there is no switch, and $u^*(t) = 0, \forall t \in [0, t_f]$.

4.4 The problem can be written on standard form as
\[
\begin{align*}
\text{minimize} & \quad \int_0^{t_f} 1 \, dt \\
\text{subject to} & \quad \dot{x}_1(t) = x_2(t), \\
& \quad \dot{x}_2(t) = u(t), \\
& \quad x_1(0) = x_{1,i}, \quad x_2(0) = x_{2,i}, \\
& \quad x_1(t_f) = 0, \quad x_2(t_f) = 0, \\
& \quad u(t) \in [-1, 1], \quad \forall t \in [0, t_f].
\end{align*}
\]
The Hamiltonian is given by
\[
H(t, x, u, \lambda) \triangleq f_0(t, x, u) + \lambda^T f(t, x, u) = 1 + \lambda_1 x_2 + \lambda_2 u.
\]
Pointwise optimization of the Hamiltonian yields
\[
\tilde{\mu}(t, x, \lambda) \triangleq \underset{u \in [-1, 1]}{\arg \min} H(t, x, u, \lambda)
\]
\[
= \underset{u \in [-1, 1]}{\arg \min} \lambda_2 u = \begin{cases} 1, & \lambda_2 < 0 \\
-1, & \lambda_2 > 0 \\
\tilde{u}, & \lambda_2 = 0
\end{cases}
\]
where $\tilde{u}$ is an arbitrary value in $[-1, 1]$. Hence, the optimal control candidate is given by
\[
u^*(t) \triangleq \tilde{\mu}(t, x(t), \lambda(t)) = \begin{cases} 1, & \lambda_2 < 0 \\
-1, & \lambda_2 > 0 \\
\tilde{u}, & \lambda_2 = 0
\end{cases}
\]
and we define the switching function as
\[
\sigma(t) \triangleq \lambda_2(t).
\]
The adjoined equations are now given by
\[
\dot{\lambda}_1(t) = -\frac{\partial H}{\partial x_1}(x(t), u^*(t), \lambda(t)) = 0,
\]
\[
\dot{\lambda}_2(t) = -\frac{\partial H}{\partial x_2}(x(t), u^*(t), \lambda(t)) = -\lambda_1(t).
\]
The first equation yields \(\lambda_1(t) = c_1\) for some constant \(c_1\). Substituting this into the second equation yields \(\lambda_2(t) = -c_1 t + c_2\) for some constant \(c_2\). The boundary conditions are given by
\[
\lambda(t_f) - \frac{\partial \phi}{\partial x}(t_f, x(t_f)) \perp S_f(t_f) \Rightarrow \left(\frac{\lambda_1(t_f)}{\lambda_2(t_f)}\right) = \nu_1 \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \nu_2 \begin{pmatrix} 0 \\ 1 \end{pmatrix},
\]
which does not add any information about the constants \(c_1\) and \(c_2\). So, how many switches occurs? Since \(\dot{\sigma}(t) = -c_1\) it follows that at most one switch can occur and therefore
\[
u(t) = \begin{cases} 1, & 0 \leq t < t' \\ -1, & t' \leq t \leq t_f \end{cases}
\]
for some \(t' \in [0, t_f]\). We have two cases:

i) When \(u(t) = 1\), we get \(\dot{x}_1 = x_2\) and \(\dot{x}_2 = 1\), so that
\[
\frac{\dot{x}_1}{x_2} = \frac{dx_1}{dx_2} = x_2 \iff x_1 + c_3 = \frac{1}{2} x_2^2,
\]
for some constant \(c_3\). The contour curves for some different values of \(c_3\) are depicted in Figure 4.4a(a).

ii) When \(u(t) = -1\) we get \(\dot{x}_1 = x_2\) and \(\dot{x}_2 = -1\)
\[
\frac{\dot{x}_1}{x_2} = \frac{dx_1}{dx_2} = -x_2 \iff x_1 + c_4 = -\frac{1}{2} x_2^2,
\]
for some constant \(c_4\). The contour curves for some different values of \(c_4\) are depicted in Figure 4.4a(b).

We know that \(u(t) = \pm 1\) with at most one switch. Hence, we must approach the origin along the trajectories depicted in Figure 4.4a(c) for which the control is defined by
\[
u(t) = -\text{sgn} \left\{ x_1(t) - \frac{1}{2} \text{sgn} \{ x_1(t) \} x_2^2(t) \right\}.
\]

4.5 (a) First consider the time optimal case, i.e., the problem is given by
\[
\min_{\nu(t)} \int_0^{t_f} dt.
\]
The Hamiltonian is
\[
H(t, x, v, \lambda) \triangleq f_0(t, x, v) + \lambda^T f(t, x, v)
= 1 + \lambda_1 x_3 + \lambda_2 x_4 + \lambda_3 \frac{c}{m} u \cos v + \lambda_4 \frac{c}{m} u \sin v.
\]
Pointwise minimization yields (observe that the method of setting the first partial derivative to zero and solve the equation does not work, since one cannot guarantee that the second partial derivative is positive)

\[ \hat{u}(t, x, \lambda) \triangleq \arg \min_v H(t, x, u, \lambda) \]

\[ = \arg \min_v (\lambda_3 \cos v + \lambda_4 \sin v) \]

\[ = \arg \min_v \sqrt{\lambda_3^2 + \lambda_4^2} \sin(v + \alpha), \]

where \( \alpha \) satisfies

\[ \cos \alpha = \lambda_4 / \sqrt{\lambda_3^2 + \lambda_4^2}, \quad \sin \alpha = \lambda_3 / \sqrt{\lambda_3^2 + \lambda_4^2}. \]

Thus, the optimal control is

\[ v^*(t) \triangleq \hat{u}(t, x(t), \lambda(t)) = -\alpha(\lambda(t)) - \frac{\pi}{2}. \] (4.2)

Thus, the optimal control satisfies

\[ \tan \left( v^*(t) \right) = \frac{\lambda_4(t)}{\lambda_3(t)} \] (4.3)

The adjoint equations are given by

\[ \dot{\lambda}_1(t) = -\frac{\partial H}{\partial x_1}(t, x(t), v^*(t), \lambda(t)) = 0, \]

\[ \dot{\lambda}_2(t) = -\frac{\partial H}{\partial x_2}(t, x(t), v^*(t), \lambda(t)) = 0, \] (4.4)

\[ \dot{\lambda}_3(t) = -\frac{\partial H}{\partial x_3}(t, x(t), v^*(t), \lambda(t)) = -\lambda_1(t), \]

\[ \dot{\lambda}_4(t) = -\frac{\partial H}{\partial x_4}(t, x(t), v^*(t), \lambda(t)) = -\lambda_2(t). \]

The first two equations yields

\[ \lambda_1(t) = c_1 \quad \text{and} \quad \lambda_2(t) = c_2, \]

for some constants \( c_1 \) and \( c_2 \). Substituting this into the last two equations yields

\[ \lambda_3(t) = -c_1 t + c_3 \]

\[ \lambda_4(t) = -c_2 t + c_4, \]

for some constants \( c_3 \) and \( c_4 \). Finally, the optimal control law (4.3) satisfies

\[ \frac{\tan v^*(t)}{c_3(t)} = \frac{\lambda_4(t)}{\lambda_3(t)} = \frac{c_4 - c_2 t}{c_3 - c_1 t}, \]

which is the desired result.

Now, consider the problem given by

\[ \min_{v: v(0) = 0} \phi(x(t_f)). \]

The solution to this problem is quite similar to the one already solved and the only differences is that the Hamiltonian does not contain the constant 1, and that the boundary value conditions for \( \lambda(t) \), when solving the adjoint equations, are different. This does not affect the optimal solution \( v^*(t) \) more than changing the values of the constants \( c_1, \ldots, c_4 \). Thus, the same expression is attained.

(b) The optimal control problem can be stated as

\[ \min_{v(\cdot)} -x_3(t_f) \]

subject to

\[ \dot{x}_1(t) = x_3(t), \]

\[ \dot{x}_2(t) = x_4(t), \]

\[ \dot{x}_3(t) = \frac{e}{m} u(t) \cos v(t), \]

\[ \dot{x}_4(t) = \frac{e}{m} u(t) \sin v(t), \]

\[ x_2(t_f) = x_{2f}, \quad x_4(t_f) = 0. \]

The first steps of the solution in (a) are still valid, and the solution to the adjoint equations, are different. This does not affect the optimal solution \( v^*(t) \) more than changing the values of the constants \( c_1, \ldots, c_4 \). Thus, the same expression is attained.
(c) The optimal control problem can be stated as

\[\text{minimize } \int_0^{t_f} dt \text{ subject to } \dot{x}_1(t) = x_3(t), \quad \dot{x}_2(t) = x_4(t), \quad \dot{x}_3(t) = \frac{c}{m} u(t) \cos v(t), \quad \dot{x}_4(t) = \frac{c}{m} u(t) \sin v(t), \quad x_1(0) = 0, \quad x_2(0) = 0, \quad x_3(0) = x_{3i}, \quad x_4(0) = x_{4i}, \quad x_1(t_f) = x_{1f}, \quad x_2(t_f) = x_{2f}.\]

This problem is similar to (a) and the only difference is the new boundary conditions, which are now given by

\[\lambda(t_f) - \frac{\partial\phi}{\partial x}(t_f, x(t_f)) \perp S_f \implies \begin{pmatrix} \lambda_1(t_f) \\ \lambda_2(t_f) \\ \lambda_3(t_f) \\ \lambda_4(t_f) \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} \text{free} \\ \text{free} \\ 0 \\ 0 \end{pmatrix} .\]

Therefore, we get \(\lambda_3(t) = -c_1(t - t_f), \lambda_4(t) = -c_2(t - t_f),\) and

\[\tan(v^*(t)) = \frac{\lambda_4(t)}{\lambda_3(t)} = \frac{c_2}{c_1} \text{ (constant)}.\]

Since \(v^*(t)\) and \(u\) are constant, we can solve the system of equations

\[\begin{align*}
\dot{x}_3(t) &= \frac{cu}{m} \cos v, \quad x_3(0) = x_{3i} \implies x_3(t) = \frac{cu}{m} t \cos v + x_{3i} \\
\dot{x}_4(t) &= \frac{cu}{m} \sin v, \quad x_4(0) = x_{4i} \implies x_4(t) = \frac{cu}{m} t \sin v + x_{4i} \\
\dot{x}_1(t) &= x_3(t), \quad x_1(0) = 0 \implies x_1(t) = \frac{cu}{m} \frac{t^2}{2} \cos v + x_{3i} t \\
\dot{x}_2(t) &= x_4(t), \quad x_2(0) = 0 \implies x_2(t) = \frac{cu}{m} \frac{t^2}{2} \sin v + x_{4i} t.
\end{align*}\]

Using the terminal conditions \(x_1(t_f) = x_{1f}, x_2(t_f) = x_{2f}\), we can calculate \(v\) and \(t_f\).

(d) The new Hamiltonian is given by

\[H_{\text{new}} = H - \lambda_4 g,\]

and the rest is the same.

(e) The problem is

\[\text{minimize } \int_0^{t_f} dt \text{ subject to } \dot{x}_1(t) = x_3(t), \quad \dot{x}_2(t) = x_4(t), \quad \dot{x}_3(t) = \frac{c}{x_5(t)} u(t) \cos v(t), \quad \dot{x}_4(t) = \frac{c}{x_5(t)} u(t) \sin v(t) - g, \quad x(0) \text{ given}, \quad x_2(t_f) = x_{2f}, \quad x_5(t_f) = x_{5f}, \quad u(t) \in [0, u_{\text{max}}], \forall t \in [0, t_f].\]

The Hamiltonian is

\[H(t, x, [u, v]^T, \lambda) \triangleq \lambda_1 x_3 + \lambda_2 x_4 + \lambda_3 \frac{c}{x_5} u \cos v + \lambda_4 \left( \frac{c}{x_5} u \sin v - g \right) - \lambda_5 u.\]

Pointwise minimization

\[\tilde{\lambda}(t, x, \lambda) \triangleq \arg\min_{u \in [0, u_{\text{max}}], v} H(t, x, [u, v]^T, \lambda) = \arg\min_{u \in [0, u_{\text{max}}], v} \left( \lambda_3 \frac{c}{x_5} \cos v + \lambda_4 \frac{c}{x_5} \sin v - \lambda_5 \right) u.\]

If \(u = 0\), then \(v\) can be arbitrary. If \(u \neq 0\), then \(\tan v = \lambda_4/\lambda_3\) as earlier. In any case, the optimal \(v^*\) can be taken as

\[\tan v^*(t) = \frac{\lambda_4(t)}{\lambda_3(t)}.\]
The adjoint equations are
\[
\begin{align*}
\dot{\lambda}_1(t) &= 0 \\
\dot{\lambda}_2(t) &= 0 \\
\dot{\lambda}_3(t) &= -\lambda_1(t) \\
\dot{\lambda}_4(t) &= -\lambda_2(t) \\
\dot{\lambda}_5(t) &= \frac{c}{x_5(t)^2} u^*(t) (\lambda_3(t) \cos v^*(t) + \lambda_4(t) \sin v^*(t))
\end{align*}
\]

Solving the four first yields (as before)
\[
\lambda_1(t) = c_1, \quad \lambda_2(t) = c_2, \quad \lambda_3(t) = -c_1 t + c_3, \quad \lambda_4(t) = -c_2 t + c_4.
\]
The boundary conditions are
\[
\lambda(t_f) - \frac{\partial \phi}{\partial x} (t_f, x(t_f)) \perp S_f \implies \begin{pmatrix} \lambda_1(t_f) + 1 \\ \lambda_2(t_f) \\ \lambda_3(t_f) \\ \lambda_4(t_f) \\ \lambda_5(t_f) \end{pmatrix} = \begin{pmatrix} 0 \\ \nu_1 \\ 0 \\ 0 \\ \nu_2 \end{pmatrix},
\]
and thus \(\lambda_1(t_f) = -1\), \(\lambda_3(t_f) = \lambda_4(t_f) = 0\) while the remaining are free.

Thus, we can conclude that \(c_1 = -1\) and
\[
\begin{align*}
\lambda_3(t) &= t - t_f \\
\lambda_4(t) &= -c_2(t - t_f)
\end{align*}
\]
\(\Rightarrow\) \(\tan v^*(t) = \frac{\lambda_4(t)}{\lambda_3(t)} = -c_2 = \text{constant}\)

Hence, the steering angle \(v^*(t)\) is constant. This gives
\[
u^*(t) = \begin{cases} 0, & \text{if } \sigma(t) > 0 \\ u_{\text{max}}, & \text{if } \sigma(t) < 0 \\ \text{arbitrary}, & \text{if } \sigma = 0 \end{cases}
\]
where the switching function is
\[
\sigma(t) \triangleq \lambda_3(t) \frac{c}{x_5(t)} \cos v + \lambda_4(t) \frac{c}{x_5(t)} \sin v - \lambda_5(t)
\]
Now, let us look at the switching functions. The standard trick for deter-
5 Infinite Horizon Optimal Control

5.1 (a) 1. The Hamiltonian is given by
\[ H(x, u, \lambda) \triangleq f_0(x, u) + \lambda T f(x, u) \]
\[ = \frac{1}{2} u^2 + \frac{1}{2} x^4 + \lambda u \]

2. Point-wise optimization yields
\[ \tilde{\mu}(x, \lambda) \triangleq \arg \min_{u \in \mathbb{R}} H(x, u, \lambda) \]
\[ = \arg \min_{u \in \mathbb{R}} \left\{ \frac{1}{2} u^2 + \frac{1}{2} x^4 + \lambda u \right\} = -\lambda, \]
and the optimal control is
\[ u^*(t) \triangleq \tilde{\mu}(x(t), V_x(x(t))) = -V_x(x(t)), \]
where \( V_x(x(t)) \) is obtained by solving the infinite time horizon HJB.

3. The HJB is given by
\[ 0 = H(x, \tilde{\mu}(x, V_x), V_x). \]
In our case this is equivalent to
\[ 0 = -\frac{1}{2} V_x^2 + \frac{1}{2} x^4, \]
which implies that \( V_x = \pm x^2 \) and thus \( V(x) = \pm \frac{1}{2} x^3 + C \), for some constant \( C \). Since \( V(x) > 0 \) for all \( x \neq 0 \) and \( V(0) = 0 \), we have
\[ V(x) = \frac{1}{3} |x^3|. \]
Finally, the optimal control is given by
\[ u^*(t) = -V_x(x(t)) = -x^2(t) \sgn(x(t)), \]
which is reasonable if one looks at the optimal control problem we are solving.

(b) 1. The Hamiltonian will be the same
2. Point-wise optimization yields
\[ \tilde{\mu}(x, \lambda) \triangleq \arg \min_{|u| \leq 1} H(x, u, \lambda) \]
\[ = \arg \min_{|u| \leq 1} \left\{ \frac{1}{2} u^2 + \frac{1}{2} x^4 + \lambda u \right\} = \begin{cases} -\lambda, & \text{if } |\lambda| \leq 1 \\ -\sgn(\lambda), & \text{if } |\lambda| > 1 \end{cases} \]
and the optimal control is
\[ u^*(t) \triangleq \tilde{\mu}(x(t), V_x(x(t))) = \begin{cases} -V_x(x(t)), & \text{if } |V_x(x(t))| \leq 1 \\ -\sgn(V_x(x(t))), & \text{if } |V_x(x(t))| > 1 \end{cases}, \]
where \( V_x(x(t)) \) is obtained by solving the infinite time horizon HJB.
3. The HJB is given by
\[ 0 = H(x, \tilde{\mu}(x, V_x), V_x). \]
In our case this is equivalent to
\[ 0 = \begin{cases} -\frac{1}{2} V_x^2 + \frac{1}{2} x^4, & \text{if } |V_x| \leq 1 \\ \frac{1}{2} + \frac{1}{2} x^4 - V_x \sgn(V_x), & \text{if } |V_x| > 1 \end{cases} \]
\[ V_x = \begin{cases} \pm x^2, & \text{if } |V_x| \leq 1 \\ \frac{1}{2}(1 + x^4) \sgn(V_x), & \text{if } |V_x| > 1 \end{cases}. \]
Since \( |\pm x^2| \leq 1 \Rightarrow |x| \leq 1 \) and \( \left| \frac{1}{2}(1 + x^4) \sgn(V_x) \right| > 1 \Rightarrow |x| > 1 \) we have
\[ V_x = \begin{cases} \pm x^2, & \text{if } |x| \leq 1 \\ \frac{1}{2}(1 + x^4) \sgn(V_x), & \text{if } |x| > 1 \end{cases}, \]
which implies
\[ V(x) = \begin{cases} \frac{1}{12} |x|^3, & \text{if } |x| \leq 1 \\ \frac{1}{2} |x| + \frac{1}{10} |x|^5 + C, & \text{if } |x| > 1 \end{cases}. \]
Since $V(x)$ must be continuous we have
$$\lim_{x \to \pm 1^\pm} V(x) = V(1) \Rightarrow \frac{1}{2} + \frac{1}{10} + C = \frac{1}{3} \Rightarrow C = -\frac{4}{15}$$

We also notice that $V_x$ is continuous $\Rightarrow V \in C^1$. Thus, the optimal cost-to-go function is
$$J^*(x) = V(x) = \begin{cases} \frac{1}{3}|x|^3, & \text{if } |x| \leq 1 \\ \frac{1}{2}|x| + \frac{1}{10}|x|^5 - \frac{4}{15}, & \text{if } |x| > 1 \end{cases}$$

Finally, the optimal control is given by
$$u^*(t) = \begin{cases} -V_x(x(t)), & \text{if } |x(t)| \leq 1 \\ -\text{sgn}(V_x), & \text{if } |x(t)| > 1 \end{cases} = \begin{cases} -x^2(t) \text{sgn}(x(t)), & \text{if } |x(t)| \leq 1 \\ -\text{sgn}(x(t)), & \text{if } |x(t)| > 1, \end{cases}$$

which is reasonable if we combine the control signal constraint with the unconstrained optimal control (5.1).

5.2. The Hamiltonian is given by
$$H(x, u, \lambda) \triangleq f_0(x, u) + \lambda^T f(x, u)$$
$$= \frac{1}{2} u^2 + \frac{1}{2} x^2 + x^4 + \lambda(x^3 + u)$$

2. Point-wise optimization yields
$$\tilde{u}(x, \lambda) \triangleq \arg \min_{u \in \mathbb{R}} H(x, u, \lambda)$$
$$= \arg \min_{u \in \mathbb{R}} \left\{ \frac{1}{2} u^2 + \frac{1}{2} x^2 + x^4 + \lambda(x^3 + u) \right\} = -\lambda,$$

and the optimal control is
$$u^*(t) \triangleq \tilde{u} (x(t), V_x(x(t))) = -V_x(x(t)),$$

where $V_x(x(t))$ is obtained by solving the infinite time horizon HJB equation.

3. The HJB equation is given by
$$0 = H(x, \tilde{u}(x, V_x), V_x).$$

In our case this is equivalent to
$$0 = -\frac{1}{2} V^2 + \frac{1}{2} x^2 + x^4 + V_x x^3,$$

We make the ansatz $V(x) = \alpha x^2 + \beta x^4$ which gives $V_x = 2\alpha x + 4\beta x^3$ and we have
$$0 = -\frac{1}{2} (2\alpha x + 4\beta x^3)^2 + \frac{1}{2} x^2 + x^4 + (2\alpha x + 4\beta x^3)x^3$$
$$-2\alpha^2 x^2 - 8\alpha \beta x^4 - 8\beta^2 x^6 + \frac{1}{2} x^2 + x^4 + 2\alpha x^4 + 4\beta x^6$$
$$= (-2\alpha^2 + \frac{1}{2}) x^2 + (-8\alpha \beta + 1 + 2\alpha) x^4 + (-8\beta^2 + 4\beta) x^6,$$

which is true for all $x$ if $\alpha = \frac{1}{2}$ and $\beta = \frac{1}{2}$ and we have
$$V(x) = \frac{1}{2} (x^2 + x^4).$$

Finally, the optimal control is given by
$$u^*(t) = -V_x(x(t)) = -x(t) - 2x^3(t).$$

5.3 (a)
$$\dot{x}(t) = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix} x(t) + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u(t)$$
$$y(t) = \begin{pmatrix} 1 & 0 \end{pmatrix} x(t)$$

(b)
$$R = \alpha^2$$
$$Q = \beta^2 C^T C$$

(c) Optimal feedback is $u^*(t) = -R^{-1} B^T P x(t)$, where $P$ is the solution to the ARE
$$PA + A^T P - PB R^{-1} B^T P + Q = 0$$

(See Theorem 10 in the lecture notes.)
(d) The closed loop system is
\[ \dot{x}(t) = Ax(t) + B(-R^{-1}B^TPx(t)) = (A - BR^{-1}B^TP)x(t). \] (5.5)

Read about `are` by running `help are` in Matlab!

```matlab
>> A = [0 1; 0 0]; B = [0; 1]; C = [1 0];
>> alpha = 1; beta = 2;
>> R = alpha^2; Q = beta^2*C'*C;
>> P = are(A, B*inv(R)*B', Q)
P =
 4.0000  2.0000
 2.0000  2.0000
>> eig(A - B*inv(R)*B'*P)
an =
 -1.0000 + 1.0000i
 -1.0000 - 1.0000i
```

(e) Let
\[ P = \begin{pmatrix} p_1 & p_2 \\ p_2 & p_3 \end{pmatrix}. \] (5.6)

and the ARE can be expressed as
\[ \begin{pmatrix} \beta^2 - \alpha^2 p_2^3 & p_1 - \alpha^2 p_2 p_3 \\ p_1 - \alpha^2 p_2 p_3 & 2p_2 - \alpha^2 p_3^2 \end{pmatrix} = 0. \] (5.7)

or equivalently as
\[ \begin{align*}
p_1 &= \alpha^2 p_2 p_3 \\
p_2^2 &= \alpha^2 \beta^2 \\
p_3^2 &= 2\alpha^2 p_2.
\end{align*} \] (5.8)

Note that the last equation gives that \( p_2 > 0 \), and furthermore we have that \( p_1 > 0 \) and \( p_3 > 0 \) since \( P \) is positive definite. Thus, the solution to the ARE is
\[ P = \begin{pmatrix} \beta \sqrt{2\alpha \beta} & \alpha \beta \\ \alpha \beta & \sqrt{2\alpha \beta} \end{pmatrix}. \] (5.9)

and the optimal control is
\[ u^*(t) = -\frac{\beta}{\alpha} x_1(t) - \sqrt{\frac{2\beta}{\alpha}} x_2(t) = -\frac{\beta}{\alpha} y(t) - \sqrt{\frac{2\beta}{\alpha}} \hat{y}(t). \] (5.10)

Note that this is a PD-controller dependent on the fraction \( \beta/\alpha \).

(f) The system matrix of the closed loop system is
\[ A - BR^{-1}B^TP = \begin{pmatrix} 0 & -\beta/\alpha \\ -\beta/\alpha & -\sqrt{2\beta/\alpha} \end{pmatrix}. \] (5.11)

with the eigenvalues
\[ \lambda = \sqrt{\frac{\beta}{2\alpha}}(-1 \pm i). \] (5.12)

5.4 (a) >> A = [-0.089 -2.19 0 0.319 0 0
0.076 -0.217 -0.166 0 0 0
-0.602 0.327 -0.975 0 0 0
0 0.15 1 0 0 0
0 1 0 0 0 0
1 0 0 0 2.19 0];
>> B = [0 0.0327
0.0264 -0.151
0.227 0.0636
0 0
0 0
0 0];
>> C = [0 0 0 0 0 1];
>> K1 = [0 0 0 1 0 0; 0 0 0 0 -1 0]
K1 =
 0 0 0 1 0 0
 0 0 0 0 -1 0
>> E1 = eig(A-B*K1)
E1 =
 0
-1.0591
-0.0235 + 0.7953i
-0.0235 - 0.7953i
-0.0874 + 0.1910i
-0.0874 - 0.1910i

Note that the first eigenvalue is in the origin, and hence the system is not asymptotically stable. This eigenvalue correspond to the lateral position, and this state will not be driven to zero by this controller. Thus, it is not straightforward to design a controller to this system by intuition.
(b) Consider the following optimal control problem

\[
\begin{align*}
\text{minimize } u & \quad J = \int_0^\infty (x(t)^T Q x(t) + u(t)^T R u(t)) \, dt \\
\text{subject to } \quad _x(x(t)) &= Ax(t) + Bu(t), \\
& \quad u(t) \in \mathbb{R}^2, \quad \forall t \geq 0.
\end{align*}
\]

The Matlab command `lqr` is used to solve this problem, read the information about `lqr` by running `help lqr` in Matlab.

\[
\begin{align*}
\text{>> } Q &= \text{eye}(6); \quad R = \text{eye}(2); \\
\text{>> } [K2, S2, E2] &= \text{lqr}(A, B, Q, R); \\
\text{>> } K2, E2 \\
K2 &=
\begin{bmatrix}
2.7926 & 4.2189 & 2.0211 & 3.2044 & 9.7671 & 0.8821 \\
-1.2595 & -6.8818 & -1.0375 & -2.0986 & -6.9901 & -0.4711
\end{bmatrix}
\end{align*}
\]

\[
\begin{align*}
E2 &=
\begin{bmatrix}
-1.2275 \\
-0.2911 + 0.7495i \\
-0.2911 - 0.7495i \\
-0.2559 + 0.3553i \\
-0.2559 - 0.3553i \\
-0.4618
\end{bmatrix}
\end{align*}
\]

All eigenvalues are in the left half plane, and the feedback system is asymptotically stable.

(c) Note that it is not the absolute values of the elements of \( Q \) and \( R \) that are important, but their ratios. If \( Q \) is “large” and \( R \) is “small”, then the states are forced to zero quickly and the magnitude of the control signal will be large, and vice versa.
6 Model Predictive Control

6.1 (a) The following script solves the problem

```matlab
mptSolutionA.m

% the zoh system
close all
clear all
s=tf('s');
G=2/(s^2+3*s+2);
Ts = 0.05;
S=c2d(ss(G),Ts)
Z.A = S.A;
Z.B = S.B;
Z.C = S.C;
Z.D = S.D;
Z.umin = -2;
Z.umax = 2;
Z.xmin = [-10;10];
Z.xmax = [10;10];
prob.norm = 2;
prob.Q = eye(2);
prob.R = 1e-2;
prob.N = 2;
prob.Tconstraint = 0;
prob.subopt_lev = 0;
ctrl = mpt_control(Z,prob);
figure
mpt_plotPartition(ctrl)
figure
mpt_plotU(ctrl)
figure
mpt_plotTimeTrajectory(ctrl, [1;1])
```

(b) MATLAB script based on the results in (a)

```matlab
mptSolutionB.m

nr = [];
n = 1:14;
for i = n
    prob.N = i;
    ctrl = mpt_control(Z,prob,[]);
    nr = [nr,length(ctrl.dynamics)];
end
figure
plot(nr)
hold on
x = lsqnonlin(@(x)monomial(x,[n.';nr.'],[1,2]));
alpha = x(1)
beta = x(2)
plot(alpha*(n').^beta,'r--')
x = lsqnonlin(@(x)exponent(x,[n.';nr.'],[1,2]))
gamma = x(1)
delta = x(2)
plot(gamma*delta.^(n'),'go-')
legend('n_r','\alpha*N^{\beta}','\gamma*\delta^{N}')
```

where we defined the functions

```matlab
monomial.m

function f = monomial(x,c)
N = length(c);
f = x(1)*c(1:N/2).^x(2)-c(N/2+1:end);
end

exponent.m

function f = exponent(x,c)
N = length(c);
f = x(1)*(x(2).^c(1:N/2)-1)-c(N/2+1:end);
end
```

The values of the optimized parameters are given by

\[ \alpha = 1.7914, \beta = 1.9739, \gamma = 44.1304, \delta = 1.1665, \]
and the complexity, at least in this case, seems to be quadratic and not exponential (since the monomial has good adaptation and that the estimated base for the exponential is close to 1). This shows that the complexity in practice may be better than the theoretical complexity which is exponential in $N$. 


7 Numerical Algorithms

7.1 (a) \[ x[k+1] = x[k] + hf(x[k], u[k]) \]
\[ = x[k] + h \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} x[k] + h \begin{pmatrix} 0 \\ 1 \end{pmatrix} u[k] \]
\[ = \begin{pmatrix} 1 & h \\ 0 & 1 \end{pmatrix} x[k] + \begin{pmatrix} 0 \\ h \end{pmatrix} u[k] \] (7.1)

secOrderSysEqDisc.m

function xk1 = secOrderSysEqDisc(xk, uk, h)
% secOrderSysEqDisc - Discrete-time dynamic equation
% xk - State at time index k
% uk - Input at time index k
% h - Sampeling time
% xk1 - State at time index k+1
xk1 = zeros(2,1);
%#x(1) = ???;
%#x(2) = ???;
  xk1(1) = xk(1) + h*xk(2);
  xk1(2) = xk(2) + h*uk;

secOrderSysCostCon.m

function J = secOrderSysCostCon(y, h)
% secOrderSysCostCon - Cost function J = F(y)
% y - Optimization vector: y = [x[0]^T u[0],...,x[N]^T,u[N]]
% h - Sampeling time
% J = Value of cost function
%#F = ???;
 J = h*sum(y(3:3:end).^2);

mainDiscLinconSecOrderSys.m

% Second order optimal control problem solved by using fmincon on % a discretized model.
% The constraints due to the system model are handled as nonlinear % constraints to illustrate the handling of a general system model.
N = 50; % number of sample points
tf = 2; % final time
h = tf/N; % sample time
%%% Constraints
xi = [1,1]'; % initial constraint
xf = [0,0]'; % final constraint
n = 3*(N+1);
%#E = [???];
%#F = [???];
E = [1, 0, 0; 0, 1, 0];
F = [1, h, 0; 0, 1, h];
Aeq = zeros(4 + 2*N, n); % linear constrints
%#Aeq(1:2,1:2) = ???;
%#Aeq(3:4,end-2:end-1) = ???;
  Aeq(1:2,1:2) = eye(2);
  Aeq(3:4,end-2:end-1) = eye(2);
for i = 1:N
    ii = 4 + 2*(i-1) + (1:2);
    jj = 3*(i-1) + (1:6);
    Aeq(ii,jj) = [-F, E];
ext = 
end%
Aeq = [x; x; zeros(2*N,1)];
%% Start guess
y0 = [x; 0]*ones(1,N+1);
y0 = y0(:);
%% Optimization options
options = optimset('fmincon');
options = optimset(options, 'Algorithm','interior-point');
options = optimset(options, 'GradConstr','on');
%%% Solve the problem
[y,fval,exitflag,output] = fmincon(@secOrderSysCostCon, y0, [], [], Aeq, Beq, ... [] , [], @secOrderSysNonlcon, options, h);%
%%% Show the result
fprintf('Final value of the objective function: %0.6f 
', fval)
tt = h*(0:N);
z = y(1:3:end)';
v = y(2:3:end)';
u = y(3:3:end)';
figure
subplot(2,1,1), plot( tt, z, 'b.-', tt, v, 'g+-'), xlabel('t'), legend('z','v'),
title('Second Order System state variables');
subplot(2,1,2), plot( tt(1:end-1), u(1:end-1), 'b+-'), xlabel('t'), ylabel('u'),
title('Second Order System control');

mainDiscNonlinconSecOrderSys.m

% Second order optimal control problem solved by using fmincon on
% a discretizised model.
% The constraints due to the system model are handled as nonlinear
% constraints to illustrate the handling of a general system model.
N = 50; % number of sample points
tf = 2; % final time
h = tf/N; % sample time
%% Constraints
xi = [1,1]'; % initial constraint
xf = [0,0]'; % final constraint
n = 3*(N+1);
Aeq = zeros(4, n);

Aeq(1:2,1:2) = eye(2);
Aeq(3:4,end-2:end-1) = eye(2);
Beq = [x; x; zeros(2*N,1)];
y0 = [x; 0]*ones(1,N+1);
y0 = y0(:);
%% Optimization options
options = optimset('fmincon');
options = optimset(options, 'Algorithm','interior-point');
options = optimset(options, 'GradConstr','on');
%%% Solve the problem
[y,fval,exitflag,output] = fmincon(@secOrderSysCostCon, y0, [], [], Aeq, Beq, ... [] , [], @secOrderSysNonlcon, options, h);
%%% Show the result
fprintf('Final value of the objective function: %0.6f 
', fval)
tt = h*(0:N);
z = y(1:3:end)';
v = y(2:3:end)';
u = y(3:3:end)';
function \[c,ceq,cJac, ceqJac\] = secOrderSysNonlcon(y, h)
% \[c,ceq,cJac, ceqJac\] = secOrderSysCostCon(y, h) - Cost function
% y - Optimization vector: y = \[x[0]^T u[0],...,x[N]^T,u[N]\]
% h - Sampeing time
% c - Nonlinear inequality constraint c(y) <= 0
% ceq - Nonlinear equality constraint ceq(y) <= 0
% cJac - Jacobian of nonlinear inequality constraint
% ceqJac - Jacobian of nonlinear equality constraint

N = length(y)/3-1; % Compute N
%c = []; cJac =[]; % No inequality constrints
% ceq = zeros(N*2,1); % (N+1)*2 equality constrints
% ceqJac = zeros(N*2, 3*N);
for k = 0:(N-1)
    xk = y((1:2) + k*3); %Extract x[k]
    uk = y(3 + k*3); %Extract u[k]
    xk1 = y((1:2) + (k+1)*3); %Extract x[k+1]
    xk1b = secOrderSysEqDisc(xk, uk, h); %Compute f(x[k],u[k])
    %#ceq((1:2) +k*2) = ???; %Compute ceq_k = x[k+1] - f(x[k],u[k])
    ceq((1:2) + k*2) = xk1 - xk1b; %Compute ceq_k = x[k+1] - f(x[k],u[k])
    % Compute \[d(ceq_k)/dx_k, d(ceq_k)/du_k, d(ceq_k)/dx_{k+1}, \]
    % d(ceq_k)/du_{k+1}\]
    %#ceqJac(2+k*2, 3+k*3) = ???;
    %#ceqJac(2+k*2, 4+k*3) = ???;
    %#ceqJac(2+k*2, 5+k*3) = ???;
    %#ceqJac(2+k*2, 6+k*3) = ???;
    %#ceqJac(2+k*2, 1+k*3) = 0;
    %#ceqJac(2+k*2, 2+k*3) = -1;
    %#ceqJac(2+k*2, 3+k*3) = -h;
    %#ceqJac(2+k*2, 4+k*3) = 0;
    %#ceqJac(2+k*2, 5+k*3) = 1;
    %#ceqJac(2+k*2, 6+k*3) = 0;
end
ceq = ceq';
ceqJac = ceqJac';

7.2 (a) (i) \(x := x_i\)
(ii) for \(k = 0\) to \(N-1\) do \(x := \bar{f}(x, u[k])\)
(iii) \(w := c||x-x_f||^2 + h \sum_{k=0}^{N-1} u^2[k]\)

(b)

function %secOrderSysNonlcon(y, h)
% secOrderSysNonlcon - Nonlinear constraints
% y - Optimization vector: y = \[x[0]^T u[0],...,x[N]^T,u[N]\]
% h - Sampling time
% c - Nonlinear inequality constraint c(y) <= 0
% ceq - Nonlinear equality constraint ceq(y) <= 0
% cJac - Jacobian of nonlinear inequality constraint
% ceqJac - Jacobian of nonlinear equality constraint

N = length(y)/3-1; % Compute N
%c = []; cJac =[]; % No inequality constrints
% ceq = zeros(N*2,1); % (N+1)*2 equality constrints
% ceqJac = zeros(N*2, 3*N);
for k = 0:(N-1)
    xk = y((1:2) + k*3); %Extract x[k]
    uk = y(3 + k*3); %Extract u[k]
    xk1 = y((1:2) + (k+1)*3); %Extract x[k+1]
    xk1b = secOrderSysEqDisc(xk, uk, h); %Compute f(x[k],u[k])
    %#ceq((1:2) +k*2) = ???; %Compute ceq_k = x[k+1] - f(x[k],u[k])
    ceq((1:2) + k*2) = xk1 - xk1b; %Compute ceq_k = x[k+1] - f(x[k],u[k])
    % Compute \[d(ceq_k)/dx_k, d(ceq_k)/du_k, d(ceq_k)/dx_{k+1}, \]
    % d(ceq_k)/du_{k+1}\]
    %#ceqJac(2+k*2, 3+k*3) = ???;
    %#ceqJac(2+k*2, 4+k*3) = ???;
    %#ceqJac(2+k*2, 5+k*3) = ???;
    %#ceqJac(2+k*2, 6+k*3) = ???;
    %#ceqJac(2+k*2, 1+k*3) = 0;
    %#ceqJac(2+k*2, 2+k*3) = -1;
    %#ceqJac(2+k*2, 3+k*3) = -h;
    %#ceqJac(2+k*2, 4+k*3) = 0;
    %#ceqJac(2+k*2, 5+k*3) = 1;
    %#ceqJac(2+k*2, 6+k*3) = 0;
end
ceq = ceq';
ceqJac = ceqJac';

mainDiscUncSecOrderSys.m
% Second order optimal control problem solved by using fmincon on % a discretized model.
% The final state constraint is added to the cost function as a penalty.
N = 50; % number of sample points
N = 50; % number of sample points
tf = 2; % final time
h = tf/N; % sample time
c = 1e3; % cost on the terminal constraint
%%% Constraints
xi = [1,1]'; % initial constraint
xf = [0,0]'; % final constraint
n = 3*(N+1);
%%% Start guess
u0 = 0*ones(1,N);
%%% Solve the problem
options = optimset('fminunc');
options = optimset('fminunc', 'LargeScale','off');
%[u,fval,exitflag,output] = fminunc(@secOrderSysCostUnc,u0,options,h,c,xi,xf);
%%% Show the result
fprintf('Final value of the objective function: %0.6f \n', fval)
X = zeros(2,N+1); X(:,1) = xi;

30
for $k = 1:N$, $X(:,k+1) = \text{secOrderSysEqDisc}(X(:,k), u(k), h)$; end

tt = h*(0:N);
$z = X(1,:)$;
$v = X(2,:)$;

figure
subplot(2,1,1), plot( tt, z, 'b.-', tt, v, 'g+-'), xlabel('t'), legend('z','v')
title('Second Order System state variables');
subplot(2,1,2), plot( tt(1:end-1), u, 'b+-'), xlabel('t'), ylabel('u')
title('Second Order System control');

secOrderSysCostUnc.m

function J = secOrderSysCostUnc(u, h, c, xi, xf)
% secOrderSysCostUnc - Cost function $J = F(u)$ - unconstrained version
% u - Optimization vector with input values: $u = [u[0],...,u[N-1]]$
% h - Sampling time
% c - Penalty constant
% xi - Initial constraint
% xf - Final constraint
% J = Value of cost function
N = length(u);
x = xi;
for $k = 1:N$ % Compute $x[k]$ recursively
    x = secOrderSysEqDisc(x, u(k), h);
end
%#J = ???;
J = h*sum(u.^2) + c * norm(x-xf)^2;

(c) Method 1 can handle path constraints better. Unconstrained optimization algorithms are in general much easier to implement than algorithms for constrained problems.

7.3 (a) Hamiltonian:

$$H = f_0(x,u) + \lambda^T f(x,u) = u(t)^2 + \lambda_1 v(t) + \lambda_2 u(t)$$

(7.6)

$$H_x = (0 \quad \lambda_1)$$

$$H_u = 2u(t) + \lambda_2$$

(7.7)

(b) Adjoint equation:

$$\dot\lambda(t) = -H_x(x(t),u(t),\lambda(t))$$

Thus

$$\dot\lambda_1(t) = 0$$

$$\dot\lambda_2(t) = -\lambda_1$$

(7.9)

where

$$\lambda(t_f) = \phi_x(x(t_f)) = 2c(x(t_f) - x_{t_f})$$

(7.10)

secOrderSysEq.m

function dx = secOrderSysEq(t, x, tu, u)
% secOrderSysEq - Continuous-time dynamic equation
% t - Time
% x - State at time t
% tu - Discrete-time time vector
% u - Discrete-time input vector
% dx - Time derivative of state at time t
% Compute the input at time t by interpolating the discrete-time input vector
u = interp1(tu, u, t);
%#dx = [???; ???];
dx = [x(2); u];

secOrderSysAdjointEq.m

function dlambda = secOrderSysAdjointEq(t, lambda, tu, u, tx, x)
% secOrderSysAdjointEq - Continuous-time adjoint equation
% t - Time
% lambda - Adjoint variable at time t
% tu - Discrete-time time vector for input vector
% u - Discrete-time input vector
% tx - Discrete-time time vector for state vector
% x - Discrete-state state vector
% dx - Time derivative of state at time t

% Compute the input and state at time t by interpolating the discrete-time
% input and state vectors
u = interp1(tu, u, t);
x = interp1(tx, x, t);

%dlambda = [???; ???];
dlambda = [0; -lambda(1)];

secOrderSysFinalLambda.m

function Laf = secOrderSysFinalLambda(x, xf, c)
% secOrderSysFinalLambda - Final constraint on lambda
% x - State
% xf - Final constraint on x(tf)
% Laf - Final constraint on lambda(tf)

%#Laf = ???
Laf = 2*c*(x(end,:)'-xf);

secOrderSysGradient.m

function Hu = secOrderSysGradient(tLa, La, tu, u, tx, x)
% secOrderSysGradient - Computes the gradient of the Hamiltonian with
% respect to u
% tLa - Discrete-time time vector for input vector
% La - Discrete-time input vector
% tu - Discrete-time time vector for input vector
% u - Discrete-time input vector
% tx - Discrete-time time vector for state vector
% x - Discrete-time state vector
% Hu - Derivative of Hamiltonian with respect to u

% Compute the input and state at time t by interpolating the discrete-time
% input and state vectors
% States at time tu
x1 = interp1(tx, x(:,1), tu);
x2 = interp1(tx, x(:,2), tu);
% Adjoint variables at time tu
la1 = interp1(tLa, La(:,1), tu);

% Hu = ???
Hu = 2*u + la2;

mainGradientSecOrderSys.m

% Second order optimal control problem solved with a gradient method.
% The final state constraint is added to the cost function as a penalty.
N = 2000; % number of sample points
tf = 2; % final time
h = tf/N; % sample time
x = 300; % cost on the terminal constraint
%#Hu = ???
Hu = 2*u + la2;

la2 = interp1(tLa, La(:,2), tu);

%#Hu = ???
Hu = 2*u + la2;

mainGradientSecOrderSys.m
\texttt{fval} = c*\text{norm}(\text{end,:})'-\text{xf}'^2 + h*\text{sum}(u.^2);\\
\texttt{fprintf('Final value of the objective function: \%0.6f 
', fval)\\
figure}\\
\texttt{subplot(2,1,1), plot( tx, z, 'b+-', tx, v, 'g+-'), xlabel('t')}\\
\texttt{title('Second Order System state variables');}\\
\texttt{subplot(2,1,2), plot( tu, u, 'b-'), xlabel('t'), ylabel('u')}\\
\texttt{title('Second Order System control');}\\
\texttt{(d) Small } c \text{ will cause that the final state is far from the desired final state.}\\
\texttt{For large } c \text{ the solution will not converge.}\\

7.4

\texttt{mainShootingSecOrderSys.m}\\
\texttt{\% Second order optimal control problem solved with a shooting method.}\\
\texttt{\% The final state constraint is added to the cost function as a penalty.}\\
\texttt{tf = 2; \% final time}\\
\texttt{x1 = [1,1]'; \% initial constraint}\\
\texttt{x2 = [0,0]'; \% final constraint}\\
\texttt{%% Optimization options}\\
\texttt{optim = optimset('fsolve');}\\
\texttt{optim.Display = 'off';}\\
\texttt{optim.Algorithm = 'levenberg-marquardt';}\\
\texttt{%% Start guess}\\
\texttt{lambda0 = [0;0];}\\
\texttt{%% Solve the problem}\\
\texttt{lambda0 = fsolve(@theta, lambda0, optim, x1, x2, tf); \% solve mu(lambda0)=0}\\
\texttt{%% Check the terminal constraints are satisfied}\\
\texttt{maxConditionError=1e-3; }\\
\texttt{if norm( theta(lambda0, x1, x2, tf) ) > maxConditionError}\\
\texttt{\hspace{1cm} disp('Warning! Terminal constraints not satisfied!');}\\
\texttt{end }\\
\texttt{%% Show the result}\\
\texttt{[ts,s] = ode23(@secOrderSysEqAndAdjointEq, [0 tf], [x1;lambda0]);}\\
\texttt{x1 = s(:,1)';}\\
\texttt{x2 = s(:,2)';}\\
\texttt{lambda1 = s(:,3)';}\\
\texttt{lambda2 = s(:,4)';}\\
\texttt{u = -lambda2/2;}\\
\texttt{fval = 0.5*sum(diff(ts).^2.*u(1:end-1).^2) + ...}\\
\texttt{0.5*sum(diff(ts).^2.*u(2:end).^2);}\\
\texttt{\% sloppy approximation!!!}\\
\texttt{fprintf('Final value of the objective function: \%0.6f 
', fval)\\
figure}\\
\texttt{subplot(2,1,1), plot( ts, x1, 'b+-', ts, x2, 'g+-'), xlabel('t')}\\
\texttt{title('Second Order System state variables');}\\
\texttt{subplot(2,1,2), plot( ts, u, 'b-'), xlabel('t'), ylabel('u')}\\
\texttt{title('Second Order System control');}\\

\texttt{secOrderSysEqAndAdjointEq.m}\\
\texttt{function ds = secOrderSysEqAndAdjointEq(t, s)}\\
\texttt{\% secOrderSysEqAndAdjointEq - Continuous-time joint system}\\
\texttt{\%}\\
\texttt{\% t - Time}\\
\texttt{\% s - State and adjoint variable at time t}\\
\texttt{x = s(1:2); }\\
\texttt{lambda = s(3:4); }\\
\texttt{%#u = ??; % Choose u such as Hu = 2*u + la2 = 0}\\
\texttt{u = -lambda2/2;}\\
\texttt{ds = zeros(4,1); }\\
\texttt{%#ds(1) = ??; }\\
\texttt{ds(1) = x(2); }\\
\texttt{%#ds(2) = ??; }\\
\texttt{ds(2) = u;}\\
\texttt{%#ds(3) = ??; }\\
\texttt{ds(3) = 0;}\\
\texttt{%#ds(4) = ??; }\\
\texttt{ds(4) = -lambda1;}\\

\texttt{theta.m}\\
\texttt{function mu = theta(lambda0, x0, xf, tf)}\\
\texttt{\% theta - Computes deviation from the final constrints for the joint system, (10.4) in \%}\\
\texttt{\% lambda0 - Adjoint variable at time t=0}\\
\texttt{\% x0 - State at time t=0}\\
\texttt{\% xf - State at time t=tf}\\
\texttt{\% tf - Final time}\\
\texttt{\% Integrate the system and adjoint equations forward in time}
\[ ts, s = \text{ode23}(@\text{secOrderSysEqAndAdjointEq}, [0 \ tf], [x_0; \lambda_0]); \]
\[ x_{tf} = s(\text{end}, 1:2); \]
\[ \lambda_{tf} = s(\text{end}, 3:4); \]
\[ \mu = \text{zeros}(2, 1); \]
\[
\theta(1) = \text{???}; \%
\text{final state constraint 1}
\mu(1) = x_f(1) - x_{tf}(1);
\]
\[
\theta(2) = \text{???}; \%
\text{final state constraint 2}
\mu(2) = x_f(2) - x_{tf}(2);
\]

7.5 (a) See Section 10.1 of the course compendium \cite{4}.

(b) The formulation in 7.2 is more general which makes it more flexible if the original problem must be changed. On the other hand, the objective function of 7.2 is a “black-box”, and the formulation 7.1 may be easier to develop tailor-made and faster optimization algorithms for. This is not a complete answer, try to come up with additional comments!

(c) See Section 10.3 of the course compendium \cite{4}.

(d) -

(e) In general, it is easier to add additional constraints to the discretization approach.

(f) See Section 10.2 of the course compendium \cite{4}.
8 The PROPT toolbox

8.1 (a) An element of arc length is
\[ ds = \sqrt{dx^2 + dy^2} = \sqrt{1 + \left( \frac{dy}{dx} \right)^2} \, dx \] (8.1)
and the total length of the curve between the points \( p_0 = (0,0) \) and \( p_f = (x_f, y_f) \) is
\[ s = \int_{p_0}^{p_f} ds = \int_0^{x_f} \sqrt{1 + y'(x)^2} \, dx. \] (8.2)

(b) The problem can be written on the form
\[ \min_{u(x)} \int_{x_0}^{x_f} \sqrt{1 + y'(x)^2} \, dx \]
subject to
\[ y'(x) = u(x) \]
\[ y(0) = 0 \]
\[ y(x_f) = y_f \] (8.3)

(c) The Hamiltonian can be expressed as
\[ H(y, u) = \sqrt{1 + u^2} + \lambda u \] (8.4)
where \( u = y' \). Pointwise minimization of \( H \) w.r.t. \( u \) gives
\[ \frac{\partial H}{\partial u}(y^*, u^*, \lambda) = \frac{u^*}{\sqrt{1 + u^*^2}} + \lambda = 0. \] (8.5)
(the second derivative is positive for all \( u^* \)). The adjoint equation is
\[ \dot{\lambda} = -\frac{\partial H}{\partial y}(y^*, u^*, \lambda) = 0. \] (8.6)

Note that there is no constraint on \( \lambda(t_f) \) since there is a constraint on the final point (see special case 5 on p.93 in the lecture notes). Hence \( \lambda \) is constant and
\[ \frac{u^*}{\sqrt{1 + u^*^2}} = c \] (8.7)
where \( c \) is constant. This can be rewritten as
\[ u^* = y^* = a \] (8.8)
where \( a = c/\sqrt{1 - c^2} \) is constant. The solution is a straight line
\[ y = ax + b \] (8.9)
where \( b \) is constant. This is an extremum path, but for this problem it is obviously also the minimum length path.

(d) \minCurveLength.m

%%% Problem setup
\xf = 10; % end point
\yf = -3;
toms \x
\p = tomPhase(‘p’, \x, 0, \xf, 128);
setPhase(p);
tomStates \y
tomControls \u
% Initial guess
\x0 = {icollocate(y == \yf/\xf * \x)
collocate(u == -1)};
% Box constraints
cbox = {};
% Boundary constraints
cbnd = {initial({y == 0})
final({y == \yf})};
% ODEs and path constraints
ceq = collocate({
dot(y) == u
y <= 0
});
% Objective
objective = integrate(sqrt((1 + dot(y).^2)));

%%% Solve the problem
options = struct;
options.name = ‘minCurveLength’;
solution = ezsolve(objective, (cbox, cbnd, ceq), \x0, options);

%%% Plot the result
\[
\begin{align*}
\text{(a)} & \quad \min_{\theta(t)} \int_0^{t_f} dt \\
\text{s.t.} & \quad \dot{x} = v \sin(\theta) \\
& \quad \dot{y} = -v \cos(\theta) \\
& \quad x(0) = 0 \\
& \quad y(0) = 0 \\
& \quad x(t_f) = x_f \\
& \quad y(t_f) = y_f
\end{align*}
\] (8.10)

\[
\begin{align*}
\text{(b)} & \quad \text{minCurveLengthHeadingCtrl.m} \\
& \quad \%\% \text{Problem setup} \\
& \quad x_f = 10; \ % \text{end point} \\
& \quad y_f = -3; \ % \text{speed} \\
& \quad \text{toms } t \\
& \quad \text{toms tf} \\
& \quad p = \text{tomPhase('p', t, 0, tf, 50)}; \\
& \quad \text{setPhase(p);} \\
& \quad \text{tomStates x y} \\
& \quad \text{tomControls theta} \\
& \quad \% \text{Initial guess} \\
& \quad x0 = \{t_f == 10 \} \\
& \quad \text{icollocate({} \\
& \quad \text{x == v*t/2} \\
& \quad \text{y == -1} \\
& \quad \})} \\
& \quad \text{collocate(theta==0));} \\
& \quad \% \text{Box constraints} \\
& \quad \text{cbox = {};} \\
& \quad \% \text{Boundary constraints} \\
& \quad \text{cbnd = {initial({x == 0; y == 0})} \\
& \quad \text{final({x == xf; y == yf}))} \\
& \quad \% \text{ODEs and path constraints}
\end{align*}
\] 

\[
\begin{align*}
\text{(c)} & \quad \text{brachistochroneHeadingCtrl.m} \\
& \quad \%\% \text{Problem setup} \\
& \quad x_f = 10; \ % \text{end point} \\
& \quad y_f = -3; \\
& \quad g = 9.81; \ % \text{gravity constant} \\
& \quad \text{toms('t')} \\
\end{align*}
\]
toms('tf')
p = tomPhase('p', t, 0, tf, 20);
setPhase(p);
tomStates x y v
tomControls theta

% Initial guess
x0 = {tf == 10
  icollocate(
    v == t
    x == v*t/2
    y == -1
  )
  collocate(theta==0)};

% Box constraints
cbox = {0.1 <= tf <= 100
  0 <= icollocate(v)
  0 <= collocate(theta) <= pi};

% Boundary constraints
cbnd = {initial({x == 0; y == 0; v == 0})
  final({x == xf; y == yf})};

% ODEs and path constraints
ceq = collocate(
  dot(x) == v.*sin(theta)
  dot(y) == -v.*cos(theta)
  dot(v) == g*cos(theta)
);

% Objective
objective = tf;

%%% Solve the problem
options = struct;
options.name = 'brachistochroneHeadingCtrl';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);

%%% Plot the result
x = subs(collocate(x),solution);
y = subs(collocate(y),solution);
v = subs(collocate(v),solution);
theta = subs(collocate(theta),solution);
t = subs(collocate(t),solution);
figure
plot(x, y); axis equal, axis([0,10,-6,0]), ylabel('y'), xlabel('x')
figure
subplot(2,1,1), plot(t, v); xlabel('t'), ylabel('v')
subplot(2,1,2), plot(t, theta); xlabel('t'), ylabel('\theta')

No. 37

Note that v == t must be placed before x == v*t/2 (in the initializing of x0) to avoid problems.

8.4 (a) We have

\[ t_f = \int_0^{x_f} \frac{\sqrt{1 + y'(x)^2}}{v} \, dx \quad (8.13) \]

in a similar way as in Exercise [8.1]. The law of energy conservation says that

\[ \frac{1}{2}mv^2 = -mg y \quad (8.14) \]

which gives \( v = \sqrt{-2gy} \).

(b)

\[
\min_{u(x)} \int_0^{x_f} \sqrt{1 + y'(x)^2} \, dx \\
\text{s.t. } y'(x) = u(x) \\
y(0) = 0 \\
y(x_f) = y_f
\]
(c) The Hamiltonian can be expressed as
\[
H(y, u) = \frac{\sqrt{1 + u^2}}{\sqrt{-2gy}} + \lambda u \tag{8.16}
\]
where \( u = y' \). Pointwise minimization of \( H \) w.r.t. \( u \) gives
\[
\frac{\partial H}{\partial u}(y^*, u^*, \lambda) = \frac{u^*}{\sqrt{-2gy^*\sqrt{1 + u^{*2}}} + \lambda = 0, \tag{8.17}
\]
(the second derivative is positive for all \( u^* \)). Note that there is no constraint on \( \lambda(t_f) \) (see special case 5 on p.93). For an autonomous systems we have
\[
H(y^*, u^*, \lambda) = \frac{\sqrt{1 + u^{*2}}}{\sqrt{-2gy^*}} + \lambda u^* = c \tag{8.18}
\]
where \( c \) is a constant. Eliminating \( \lambda \) by using (8.17) in (8.18) gives the cycloid equation
\[
y'(x) = u(x) = \sqrt{C - y(x)} \tag{8.19}
\]
where
\[
C = -\frac{1}{2ge^2}. \tag{8.20}
\]

(d) The method used in PROPT has problem when both \( y \) and \( y' \) are near zero, see Figure 8.4. The collocation points (i.e. the discretization points) are chosen to give a good result for integration of functions similar to a polynom of degree \( 2N \). Evidently, this is not the case here.

brachistochroneMinCurveLength.m

%%% Problem setup
xf = 10; % end point
yf = -3;
g = 9.81; % gravity constant
toms x
p = tomPhase('p', x, 0, xf, 20);
setPhase(p);
tomStates y
tomControls u
% Initial guess
x0 = {icollocate(y == yf/xf * x)
collocate(u == -1)};
% Box constraints
cbox = {};
% Boundary constraints
cbnd = {initial({y == 0})
final({y == yf})};
% ODEs and path constraints
ceq = collocate({ dot(y) == u
y <= 0 });
% Objective
objective = integrate(sqrt( (1 + dot(y).^2) ./ (-2*g*y)));
%%% Solve the problem
options = struct;
options.name = 'brachistochroneMinCurveLength';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);
%%% Plot the result
x = subs(collocate(x),solution);
y = subs(collocate(y),solution);
u = subs(collocate(u),solution);
figure, plot(x, y); axis equal, axis([0,10,-6,0]), xlabel('x'), ylabel('y')
8.5 (a) 

brachistochroneCykloid.m

```matlab
%% Problem setup
xf = 10; % end point
yf = -3;
toms x C
p = tomPhase('p', x, 0, xf, 20);
setPhase(p);
tomStates y
% Initial guess
x0 = {icollocate(y == yf/xf * x)};
% Boundary constraints
cbnd = {initial({y == 0})
  final({y == yf})};
% ODEs and path constraints
cpath = collocate( (1+dot(y)^2) == -C/y );
%%% Solve the problem
options = struct;
options.name = 'brachistochroneCykloid';
solution = ezsolve(0, {cbnd, cpath}, x0, options);
%%% Plot the result
x = subs(collocate(x),solution);
y = subs(collocate(y),solution);
figure, plot(x, y); axis equal, axis([0,10,-6,0]), ylabel('y'), xlabel('x')
```

8.6 (a)

(a) 

\[ E_k = \frac{m}{2} (x'^2 + y'^2) \]  \hspace{1cm} (8.21) 

\[ E_p = mgy \]

(b) 

\[
\begin{align*}
\min_{t_f} & \quad t_f \\
\text{s.t.} & \quad E_k + E_p = 0 \\
& \quad x(0) = 0 \\
& \quad y(0) = 0 \\
& \quad x(t_f) = x_f \\
& \quad y(t_f) = y_f
\end{align*}
\]  \hspace{1cm} (8.22)

(c) 

brachistochroneEnergyConservation.m

```matlab
%% Problem setup
xf = 10; % end point
yf = -3;
m = 1; % mass
g = 9.81; % gravity constant
toms t
toms tf
p = tomPhase('p', t, 0, tf, 20);
setPhase(p);
tomStates x y
% Initial guess
x0 = {tf == 10};
% Box constraints
cbox = {0.1 <= tf <= 100};
% Boundary constraints
cbnd = {initial({x == 0; y == 0})
  final({x == xf; y == yf})};
% Expressions for kinetic and potential energy
Ek = 0.5*m*(dot(x).^2+dot(y).^2);  
Ep = m*g*y;
v = sqrt(2/m*Ek);  
% ODEs and path constraints
ceq = collocate(Ek + Ep == 0);
% Objective
objective = tf;
%%% Solve the problem
options = struct;
options.name = 'brachistochroneEnergyConservation';
solution = ezsolve(objective, {cbox, cbnd, ceq}, x0, options);
%%% Plot the result
x = subs(collocate(x),solution);
y = subs(collocate(y),solution);
figure, plot(x, y); axis equal, axis([0,10,-6,0]), ylabel('y'), xlabel('x')
```

8.7 (a) The problem is due to numerical issues caused by the problem formulation. Note that (8.5) is not well defined for \( x = 0 \).

(b) Optimization algorithms are often based on assumptions and designed to make use of special structures to give accurate results in a quick and robust manner. Thus, it is important that the problem formulation suits the
algorithm. This is not a complete answer, try to come up with additional comments!

(c) The approach in exercise 8.3 is in standard form and it is quite straightforward to add additional constraints. As we have seen, the formulation of the problem in 8.4 is not suitable for this optimization algorithm. The feasibility problem in 8.5 requires some work to obtain the cycloid equation and the optimization algorithm must be able to handle DAEs. The problems formulation in 8.6 is elegant, but it might not be straightforward to include additional constraints. This is not a complete answer, try to come up with additional comments!

8.8 Note the loop with an increasing number of points. It is often a good idea to solve the problem with a low number of points first and then solve the problem again with an increasing number of points with the last solution as the start guess.

---

### mainZermeloPropt.m

```matlab
... %%% Constraints
cbox = {0 <= mcollocate(x1) <= 10
0 <= mcollocate(x2) <= 10
0 <= mcollocate(u1) <= 2*pi};
%cobnd = {initial({x1 == ???; x2 == ???})};
cobnd = {initial({x1 == xi(1); x2 == xi(2)});
... w = 1;
```
Figure 8.8a. The result of the Zermelo problem in Exercise 8.8. Upper plot shows the path. The lower figure shows the state trajectories and the control signal.
Bibliography
This version: September 2015
Bibliography


