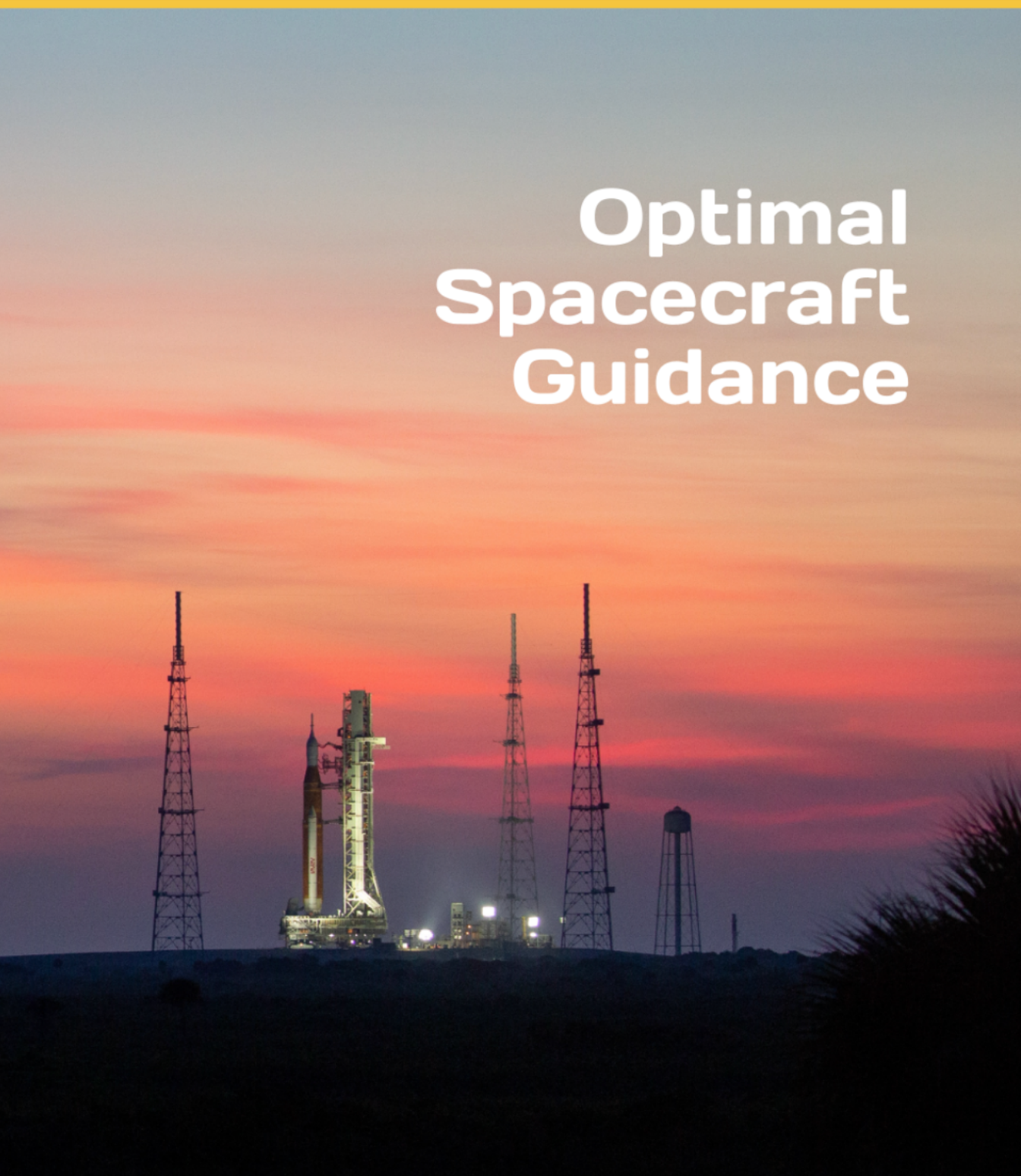
Matthew W. Harris
M. Benjamin Rose

Optimal Spacecraft Guidance



OPTIMAL SPACECRAFT GUIDANCE

OPTIMAL SPACECRAFT GUIDANCE

Matthew W. Harris
M. Benjamin Rose

The authors received financial support from the College of Engineering at Utah State University.

Published in 2023 by Utah State University, Logan, UT.

Distributed by Kindle Direct Publishing.

ISBN 979-8394013461

This book should be cited as follows.

```
@book{OptimalSpacecraftGuidance,
  author    = {Matthew W. Harris and M. Benjamin Rose},
  title     = {Optimal Spacecraft Guidance},
  publisher = {Utah State University},
  address   = {Logan, UT},
  year      = {2023}
}
```

Cover art selected from NASA's online repository. "Sunrise on the Artemis I Moon Rocket – NASA's Space Launch System (SLS) rocket with the Orion spacecraft aboard is seen at sunrise atop the mobile launcher as it arrives at Launch Pad 39B, Wednesday, Aug. 17, 2022, at NASA's Kennedy Space Center in Florida."

Contents

Preface

This book is designed for a one-semester course at Utah State University titled MAE 6570 Optimal Spacecraft Guidance. The class meets for 75 minutes, twice per week, for 14 weeks. There are no prerequisites other than graduate standing in engineering. Proficiency in calculus, differential equations, linear algebra, and computer programming is required. Students find that previous experience in space dynamics, linear multivariable control, or optimal control is helpful.

The goal of the book and course is for students to develop fundamental skills needed to do professional work in the area of spacecraft guidance. After working through the book, students should have an understanding of the linear quadratic framework, E-guidance, Q-guidance, Apollo descent guidance, and more. To this end, the book contains seven chapters. An approximate timeline for the course is the following.

- Chapter 1 | Week 1
- Chapter 2 | Weeks 2 and 3
- Chapter 3 | Weeks 4 and 5
- Chapter 4 | Weeks 6, 7, and 8
- Chapter 5 | Weeks 9 and 10
- Chapter 6 | Weeks 11 and 12
- Chapter 7 | Weeks 13 and 14

Three dynamical models are used throughout to illustrate the concepts. These models are a nonlinear two-body model, a linear flat planet model, and a linear relative orbital motion model. A key feature of the book is its integration of MATLAB implementations into the text as early as possible. For example, Chapter 1 includes a Q-guidance implementation, Chapter 2 includes a polynomial guidance implementation, and so on. Each chapter ends with a set of problems suitable for independent homework. Several of the chapter problems require modification or extension of these implementations. The final two chapters focus on descent guidance and ascent guidance. By this point, students are expected to be coding independently.

Please email errors to matthew.harris@usu.edu. Book updates, MATLAB source files, and errata are available at the following webpages.

`https://profmattharris.wordpress.com/OptimalSpacecraftGuidance`
`https://github.com/profmattharris/OptimalSpacecraftGuidance`

Matt Harris and Ben Rose

Chapter 1

Introduction to Guidance

Chapter Learning Objectives

1. Understand the basic functions of guidance, navigation, and control.
2. Analyze and solve simple one-dimensional guidance problems.
3. Implement Q-guidance for an earth ascent in a "high-fidelity" simulation.

Optimal spacecraft guidance refers to the design of optimal trajectories and to optimal control of spacecraft. The subject emanated from contemporaneous advances in computation, optimal control, and space exploration during the 1960s.[1] The limited computing power then demanded that problems, no matter how complicated, be solved with simple algorithms.[2] Some of these algorithms, such as E-guidance, Q-guidance, Apollo descent guidance, and the linear tangent law, have withstood the test of time and remain relevant today. These guidance laws and others are topics in this book.

Significant advances have been made in all elements of optimal spacecraft guidance. Advances in computing hardware are evident to all with a desktop computer or smartphone. In fact, NASA's Ingenuity Mars Helicopter uses a Samsung S5 processor. It may come as a surprise then that the speedup achieved by algorithmic advances in linear optimization has outpaced that due to hardware advances by a factor of two.[3] Improvements in convex optimization and integer optimization are equally impressive.

The ability to solve optimization problems quickly means that modern guidance algorithms can push the spacecraft to its performance limits. This is observed in the SpaceX landings. Modern guidance laws are also topics in this book. A limiting factor, however, is that flight computers are radiation-hardened and have limited computational abilities relative to a desktop computer. Customized algorithms are being developed for this purpose.[4] Though the history of optimal spacecraft guidance is rich, much work remains for the guidance engineer.

[1] Battin, *Space Guidance Evolution – A Personal Narrative*, 1982.
[2] Battin, *Some Funny Things Happened on the Way to the Moon*, 2002.
[3] Bixby, *A Brief History of Linear and Mixed-Integer Programming Computation*, 2012.
[4] Dueri, et al., *Customized Real-Time Interior-Point Methods for Onboard Powered-Descent Guidance*, 2017.

1.1 The Role of Guidance

Most vehicles moving through air, space, or water include guidance, navigation, and control (GN&C) systems that operate in real-time as the vehicle moves. The navigation system consists of sensors to measure the state of the system as well as tools for filtering, outlier detection, estimation, etc. The guidance system uses the current estimate of the state provided by the navigation system along with the mission objectives to compute state and control trajectories. The control trajectory is fed to the control system to affect actuators such as engines and wing surfaces.

A person driving a car is part of all three systems. The person's locational knowledge is navigation. The person's decision to stay straight or turn is guidance. The person's pressing of the pedals and turning of the wheel are control. Many "navigation" features of modern cars and smartphones are useful because they serve as a guidance system – "turn right in 25 feet" is guidance.

The three components of GN&C are obviously coupled. The guidance system should not rely on state estimates unavailable from the navigation system. The guidance system should not generate control commands beyond the actuator limits. And so on. With that said, it is not uncommon for the three subsystems to be designed almost independently with specification information shared between the design teams.

The guidance system may generate trajectories by using a reference trajectory (or planned path), solving an optimization problem, interpolation, approximation, or other means. In any case, the method must be quick and guaranteed to work. A good GN&C system should be 1) robust to measurement noise, disturbances, and unmodeled dynamics; 2) stable so that small errors do not cause large changes in results; and 3) simple enough so that it can run in real-time.

Such a system is demonstrated in the YouTube video "Apollo 12 landing from PDI to Touchdown".[5] The astronauts play an important role in this GN&C system. They can be heard calling out state estimates, adjusting the throttle, etc. More recent GN&C systems for planetary descent are computer-based and rely upon mathematical algorithms. With this as background, it is time for a first example.

Example 1.1. Consider a vehicle (block) that can slide in one dimension along a horizontal plane as in Figure 1.1. The vehicle can thrust left or right (but not both simultaneously). The goal is for the vehicle to pass through the target position on the right.

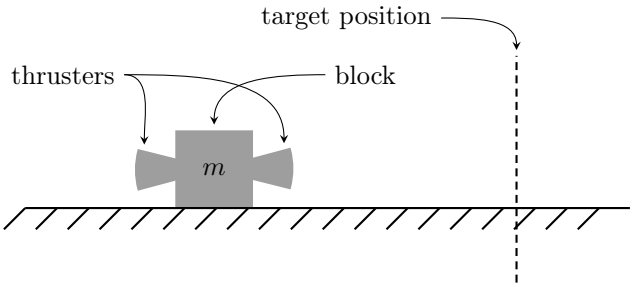


Figure 1.1: One-dimensional sliding block with target position.

[5] `https://youtu.be/kFSa6vUix70`

Because the vehicle does not have to stop at the target position, it is clear that it should take one of two actions: 1) thrust left (move right) when left of the target or 2) thrust right (move left) when right of the target. It should be clear that this guidance solution requires having a target (the mission objective), knowing the current position (the navigation system), and having thrusters that do as commanded (the control system).

ℹ How does the solution change if the vehicle is required to pass through the target at a certain time? How does the solution change if the vehicle is required to stop at the target? What additional features are required from navigation and control?

★

With the first, simple example done, a one-dimensional problem with dynamics is investigated. The problem is representative of the final vertical descent phase of a planetary landing.

Example 1.2. A lunar lander is 10 m above its landing site and has 1 m/s downward velocity. The goal is to descend to the surface and touch down with 1 m/s velocity. What thrust acceleration is needed to achieve the goal?

To analyze the problem, Newton's laws are applied. Relative to the ground, the altitude is measured by r. The vehicle mass is m. The constant gravitational acceleration is g, and the thrust force is T. The ratio T/m is the thrust acceleration u. The free-body diagram for the mass subjected to gravitational and thrust forces is in Figure 1.2.

Figure 1.2: Free-body diagram of lunar lander.

The resulting equation of motion is $m\ddot{r} = T - mg$. Because the given initial velocity equals the desired final velocity, one approach is to descend at constant velocity. This requires $0 = \ddot{r} = u - g$. Hence, one solution to this problem is to use constant thrust acceleration $u = g$. The problem of landing vertically and minimizing fuel consumption is studied in Section 6.1.

ℹ Though the thrust acceleration is constant, will the thrust force be decreasing, constant, or increasing? What role do the mission objective, navigation system, and control system play in the solution?

★

In the previous example, the required thrust acceleration was a constant function. In more challenging circumstances, the required function may vary with time. It may even be discontinuous. The guidance problem is to determine such a function so that mission objectives are satisfied. Let t_0 be the current time and t_f be the final time. The problem of finding a function

$$u : [t_0, t_f] \to \mathbb{R}^3$$

that optimizes a performance metric and satisfies some constraints is the optimal spacecraft guidance problem. The resulting formulas that dictate how to compute u are an optimal guidance law. The function u and associated position and velocity functions are the optimal spacecraft trajectories. In landing scenarios such as the one in Example 1.2, common performance metrics are fuel consumption, flight time, and risk to crew. Common constraints are thrust magnitude limits, target positions, and target velocities.

Solving the guidance problem to obtain a guidance law and compute trajectories requires knowledge of dynamical models (see Chapter 2), optimization (see Chapter 3), optimal control (see Chapters 4 and 5), and numerical methods (discussed in every chapter). Before delving into details, a simulation is coded to demonstrate guidance in action.

1.2 MATLAB Implementation of Q-Guidance

Performance of a guidance law is studied through both analysis and simulation. A simulation framework is now presented in MATLAB code that shows how a guidance law can be tested. As you work through the book, it may be useful to revisit this simulation and recall the fundamental role guidance plays. The guidance law used for illustration purposes is Q-guidance, which is developed in Section 7.3. Assuming a constant gravitational field, the problem is to ascend from a point on the surface `r0` with initial velocity `v0` to a pre-determined point in position space `rf`. The maximum allowable control acceleration is `umax`. The flight time is set at `tf`. A vector of linearly spaced times between 0 and `tf` is `t`. Standard units of meters and seconds are used.

```
% Data
r0 = [0;0];              % m
v0 = [0;0];              % m/s
rf = [90000; 45000];     % m
umax = 50;               % m/s2
tf = 100;                % s
t  = linspace(0,tf,1e3)';
```

With this data and a function `ode` to be specified shortly, a simulation can be run using MATLAB's built-in integrator `ode45`.

```
% Guidance Simulation
[~,x] = ode45(@ode,t,[r0;v0],[],tf,rf,umax);
figure, plot(x(:,1),x(:,2)), grid on
xlabel('Range (m)')
ylabel('Altitude (m)')
```

Observe that the call to `ode45` includes the time, initial state, desired final position, and maximum acceleration. Lines 10-12 simply make a plot for visualization purposes. The function to be integrated is now specified and called `ode`. Included in this function are a navigation block, guidance block, control block, and dynamics block.

```
13 function [xdot,u,u_cmd] = ode(t,x,tf,rf,umax)
14 r = x(1:2);
15 v = x(3:4);
16 % placeholder
17
18 % Navigation Block
19 rhat = r;
20 vhat = v;
21
22 % Guidance Block
23 u_cmd = [0;0];
24 tgo   = tf-t;
25 if tgo >= 1
26     vr = 1/tgo * ( rf - rhat - 1/2*tgo^2*[0;-9.81] );
27     vg = vr - vhat;
28     if norm(vg) >= 1
29         u_cmd = vg/norm(vg) * umax;
30     end
31 end
32
33 % Control Block
34 % placeholder
35 u = u_cmd;
36
37 % Dynamics Block
38 rdot = v;
39 vdot = u + [0;-9.81];
40 xdot = [rdot; vdot];
```

Lines 14 and 15 define the position and velocity state variables. The navigation block outputs estimates of the position and velocity, `rhat` and `vhat`, respectively. For the time being, the navigation estimates are equal to the true position and velocity. The estimates are passed to the guidance block, which outputs a thrust acceleration command `u_cmd`. The control block outputs the actual thrust achieved by the control system `u`. For the time being, the actual thrust equals the commanded thrust. This thrust is passed to the dynamics, which are integrated in the call to `ode45`. In this simulation, the dynamics assume a constant gravitational field. A more complicated model could be used and disturbances could also be included. As shown in Figure 1.3, the resulting state trajectory begins in the lower left at the origin and rises until it terminates at the desired position.

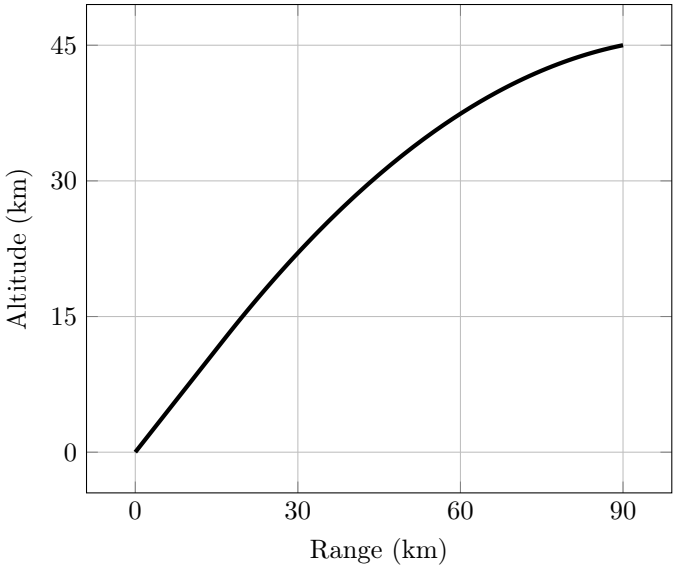


Figure 1.3: State trajectory generated by Q-guidance.

In a more realistic simulation, navigation does not output exactly the state and the commanded control is not exactly achieved. Though the intricacies of a navigation system are beyond the scope of this book, the guidance law can be tested by simply corrupting the state estimate from the navigation block. This is done by adding a time-varying signal bounded by $\pm 5\%$ of the truth.

```
% Navigation Block
rhat = r + 0.05*r*cos(3*t);
vhat = v + 0.05*v*sin(2*t);
```

Rerunning the code with this new navigation block confirms that the guidance law continues to achieve the mission objective even without perfect state information. Control actuators have inertia and cannot instantly change to achieve commanded values. For this reason, actuators are modeled as dynamical systems. Actuator dynamics are commonly modeled as first or second-order systems. The code below uses a first-order model. As such, the control states need to be added.

```
u = x(5:6);
```

In the control block, first-order dynamics now appear.

```
% Control Block
sigma = 10;
udot  = sigma*u_cmd-sigma*u;
```

The speed of the actuator response is affected by `sigma`. Larger values create a faster response. The new states also need to be added to the output of `ode`.

```
xdot = [rdot; vdot; udot];
```

With these changes, rerunning the code confirms that the guidance law continues to achieve the mission objective without perfect state information and with actuator dynamics. As a final test of the guidance system, the "real" dynamical environment is changed so that the environment assumed in the generation of the law is different from the real environment. To do so, the real gravity is 75% of that on earth and a time-varying disturbance is added.

```
% Dynamics Block
rdot = v;
vdot = u + 0.75*[0;-9.81] + 0.25*sin(5*t);
xdot = [rdot; vdot; udot];
```

Note that the guidance law still thinks the gravity is the same as that on earth and is unaware of the time-varying disturbance. Rerunning the code again confirms the guidance law continues to achieve the mission objective without perfect state information, with actuator dynamics, and without good knowledge of the dynamical environment. Actually, the final position target is missed by approximately 150 m. The percent error is 0.16%.

Though we have undertaken this exercise for demonstration purposes, a similar (and more extensive) exercise must be done before a guidance law is ever used onboard a vehicle. The real purpose is to show that the GN&C system satisfies the three aforementioned properties: 1) robust to measurement noise, disturbances, and unmodeled dynamics; 2) stable so that small errors do not cause large changes in results; and 3) simple enough so that it can run in real-time. With the code in its current form, spend some time trying to "break" the guidance law. Things to try include adding a bias in the navigation output, slowing the actuator response, making more dramatic changes to the dynamical model, restricting the flight time or max acceleration, and so on. This guidance law works well over a wide array of imperfections, but there is a point past which it will fail. As a guidance engineer, it is important to know where this point is.

1.3 Resources and Nomenclature

As may be evident from the implementation exercise, the design and analysis of guidance laws requires a healthy mix of theory, computation, and experience. To this end, the chapter problems serve as a review of "theoretical" concepts from calculus, differential equations, and linear algebra.[6] Proficiency in each area is required. Any difficulties encountered in the problem set should be resolved before moving ahead in the book. Proficiency in computer coding is also required.[7] MATLAB is not required. Free software packages such as Octave, Julia, or Python are suitable alternatives; however, all implementations in this book are written in MATLAB.

Throughout this book, "vectors" such as position and velocity are not bolded or decorated with an arrow. The dimension of the variable is specified using the notation $r \in \mathbb{R}^3$. This nomenclature is adopted because optimal spacecraft guidance problems involve states, controls, costates, and Lagrange multipliers. These commonly have different dimensions. In this context, having bolded symbols tells nothing about whether two bolded symbols can be added, dotted, or crossed. The magnitude of a vector is denoted by double bars as $||r||$ and is equal to $\sqrt{r^\top r} = \sqrt{r \cdot r}$.

[6] Kreyszig, *Advanced Engineering Mathematics*, 2020.
[7] Gilat, *MATLAB: An Introduction with Applications*, 2014.

The time derivative of a function is denoted with an overdot $\dot{x} = \frac{dx}{dt}$. A function of time $x : [t_0, t_f] \to \mathbb{R}$ is absolutely continuous if it has a derivative $\dot{x}$ almost everywhere that is Lebesgue integrable and

$$x(t) = x(t_0) + \int_{t_0}^{t} \dot{x}(\tau)\, d\tau \tag{1.1}$$

for every $t \in [t_0, t_f]$. The meaning of "almost everywhere" is explained in Chapter 5.

A function is differentiable if all of its first derivatives exist. A function is continuously differentiable if all of its first derivatives exist and are continuous. A function is twice continuously differentiable if all of its second derivatives exist are continuous. The following conventions regarding derivatives are adopted. Given a scalar-valued vector function $f : \mathbb{R}^n \to \mathbb{R}$, its gradient is

$$\frac{\partial f}{\partial x} = \nabla_x f = \begin{bmatrix} \dfrac{\partial f}{\partial x_1} \\ \vdots \\ \dfrac{\partial f}{\partial x_n} \end{bmatrix}, \tag{1.2}$$

which is $n \times 1$. Given a vector-valued vector function $f : \mathbb{R}^n \to \mathbb{R}^m$, its gradient is

$$\frac{\partial f}{\partial x} = \nabla_x f = \begin{bmatrix} \dfrac{\partial f_1}{\partial x_1} & \cdots & \dfrac{\partial f_m}{\partial x_1} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial f_1}{\partial x_n} & \cdots & \dfrac{\partial f_m}{\partial x_n} \end{bmatrix}, \tag{1.3}$$

which is $n \times m$.

1.4 Chapter Problems

Problem 1.1. Solve (by hand) the following differential equations. Simulate numerically using a variable step integrator such as MATLAB's `ode45`.

(a) $\ddot{x} + x = 1, \quad x(0) = 1, \quad \dot{x}(0) = 1$
(b) $\ddot{x} = t, \quad x(0) = 1, \quad \dot{x}(0) = 1$
(c) $\ddot{x} + \dot{x} + x = 5, \quad x(0) = 1, \quad \dot{x}(0) = 1$

Problem 1.2. The equation of motion for a mass-spring system is $\ddot{x} + x = u$, where u is a control function. From initial conditions $x(0) = 1$ and $\dot{x}(0) = 1$, compute the position and velocity when $t = 10$ for the following control functions.

(a) $u = -1$
(b) $u = 0$
(c) $u = +1$

Problem 1.3. For the mass-spring system $\ddot{x} + x = u$, find a control function that drives the state to the origin asymptotically. Find a control function that drives the state to the origin in finite time.

Problem 1.4. Let $c, x \in \mathbb{R}^n$ and $f : \mathbb{R}^n \to \mathbb{R}$ such that $f(x) = c^\top x$. Compute the gradient and Hessian of f.

Problem 1.5. Let $x \in \mathbb{R}^n$ and $f : \mathbb{R}^n \to \mathbb{R}$ such that $f(x) = x^\top x$. Compute the gradient and Hessian of f.

Problem 1.6. Consider a position vector $r(t) = [\sin(t), \cos(t), 0]^\top$. Compute the velocity $v(t)$, acceleration $a(t)$, and radial velocity $\frac{d}{dt}\|r(t)\|$.

Problem 1.7. Let $r(t) \in \mathbb{R}^3$ be a position vector. Show that $r \cdot v = \|r\| \frac{d}{dt}\|r\|$.

Problem 1.8. Define range space and null space for a matrix $A \in \mathbb{R}^{m \times n}$.

Problem 1.9. Determine the rank, range space, and null space for the following matrix.

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 5 & 7 \\ 3 & 7 & 10 \end{bmatrix}$$

Problem 1.10. For the A matrix in the previous problem, determine all solutions to the equation $Ax = b$ where $b = [6, 14, 20]^\top$.

Problem 1.11. For $x = [1, 2, 3, 4]^\top$, compute $\|x\|_1$, $\|x\|_2$, and $\|x\|_\infty$.

Problem 1.12. Compute the spectral norm of the following matrix.

$$A = \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix}$$

Problem 1.13. Replicate the Q-guidance implementation in Section 1.2. Replace the first-order actuator models with second-order actuator models of the form

$$\ddot{u} + 2\zeta\omega\dot{u} + \omega^2 u = \omega^2 u_{cmd}.$$

With the damping ratio $\zeta = 0.7$, examine the system response for values of ω between 0.1 and 20.

Chapter 2

Dynamical Models

Chapter Learning Objectives

1. Derive the two-body equation of orbital motion in Cartesian coordinates.
2. Derive the "flat planet" model describing motion in a constant gravitational field.
3. Derive the Clohessy-Wiltshire equations describing relative motion of spacecraft.
4. Implement polynomial guidance for descent and an autopilot for rendezvous.

Important in the development and testing of a guidance law are dynamical models. In developing a guidance law, simple dynamical models tend to be used because they are more amenable to analysis. Simple models are obtained through assumptions such as constant gravity and small relative motion. In simulation, higher-fidelity dynamical models tend to be used so that simulation results are representative of reality. The development of higher-fidelity models that include non-spherical earth, multi-body perturbations, wind gusts, and so on is beyond the scope of this book.[8]

The model used depends on the environment in which the vehicle is operating. In spacecraft guidance, the fundamental equation of motion is called the two-body equation of motion. From it, other models can be derived. These models include a relative motion model for spacecraft in close proximity and a flat planet model for which the gravitational field is constant in direction and magnitude. Another important part of the dynamical model is the mass equation, which relates fuel consumption to thrust used during the mission. These models are the topics of this chapter.

Physics-based models are continuous in time and described by ordinary differential equations. It is sometimes convenient, however, to have models that are discrete in time. Propagation of the state can then be done in a simple loop and any optimization problem that includes dynamics as a constraint is finite-dimensional rather than infinite-dimensional. There is a simple approach for discretizing linear systems using the state transition matrix that is exact assuming the control is constant over the sampling period. Discretization of nonlinear systems, on the other hand, is slightly more involved and deferred until Chapter 4.

[8]Bond and Allman, *Modern Astrodynamics: Fundamentals and Perturbation Methods*, 1996.

2.1 Orbital Motion

In the two-body setting, there are only two spherical masses in the universe that do not touch. This is of course a great simplification of the universe, but experiments and observation confirm this is a reasonable assumption for the purposes of developing a guidance law. It may not be a reasonable assumption for developing a simulation model, which may need to account for other massive bodies, non-spherical masses, and more.

To understand the motion of the two masses in the universe, three assumptions are made.

1. A fixed inertial frame exists.
2. The gravitational field adheres to an inverse square law.
3. Newton's laws of motion apply.

With these assumptions in place, derivation of the equation of motion can begin. The first mass is m_1 and the second mass is m_2. The position of the first mass measured in the fixed inertial frame is $R_1 \in \mathbb{R}^3$. The position of the second mass measured in the fixed inertial frame is $R_2 \in \mathbb{R}^3$. The relative position of m_2 with respect to m_1 is $r = R_2 - R_1$. This configuration is shown in Figure 2.1.

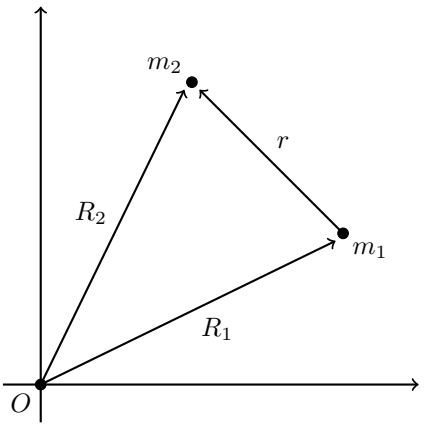


Figure 2.1: Inertial frame with two masses.

Applying the inverse square law and Newton's laws of motion indicates that

$$m_1 \ddot{R}_1 = +\frac{G m_1 m_2}{\|r\|^3} r, \tag{2.1a}$$

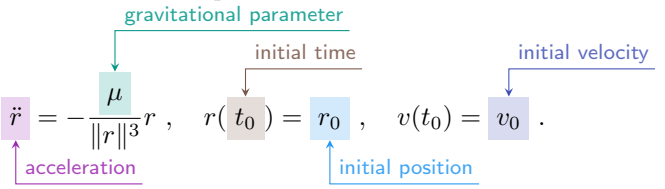
$$m_2 \ddot{R}_2 = -\frac{G m_1 m_2}{\|r\|^3} r. \tag{2.1b}$$

These are two, coupled, second-order differential equations. Given six initial conditions for each, the system can be numerically integrated to find future state values. The main challenge is that the state is measured relative to some inertial frame. Most measurements are made from earth, a satellite in orbit, or other celestial body. It should be clear that each of these objects is accelerating and cannot serve as the origin of the assumed inertial frame. Thus, this system of differential equations is not much

use in its current form. A more useful form is obtained by studying the motion of the second mass with respect to the first. Recognizing that $\ddot{r} = \ddot{R}_2 - \ddot{R}_1$ gives

$$\ddot{r} = -\frac{G(m_1 + m_2)}{\|r\|^3} r. \tag{2.2}$$

This is a single, second-order differential equation. Given an initial relative position and velocity of the second mass with respect to the first, the system can be numerically integrated to find future states. If one now thinks of the first mass as the earth and the second mass as a spacecraft, it is very reasonable to have measurements of the position and velocity required for integration. The gravitational parameter for the two-body system is $\mu = G(m_1 + m_2)$. Whenever the second mass is very small compared to the first $\mu \approx Gm_1$. The two-body equation of motion and its initial conditions are summarized below in the annotated equation.



(2.3)

Solutions to the two-body equation of motion are the standard circles, ellipses, parabolas, and hyperbolas (depending on the initial conditions). Centuries of work by bright people have led to nearly analytical solutions of the two-body problem using Kepler's equation, which reduces the problem of solving the differential equation to that of solving a single-variable algebraic equation.[9] This algebraic equation does not have a closed-form solution and must be solved numerically.

In the presence of thrust acceleration u and disturbance acceleration a_d, the equation of motion becomes

$$\ddot{r} + \frac{\mu}{\|r\|^3} r = a_d + u, \quad r(t_0) = r_0, \quad v(t_0) = v_0. \tag{2.4}$$

In first-order state-space form

$$\begin{bmatrix} \dot{r} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} v \\ -\frac{\mu}{\|r\|^3} r + a_d + u \end{bmatrix}. \tag{2.5}$$

Because u is the control variable, which is to be designed in guidance, it is common to isolate it as in the following form.

$$\begin{bmatrix} \dot{r} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} v \\ -\frac{\mu}{\|r\|^3} r + a_d \end{bmatrix} + \begin{bmatrix} 0 \\ u \end{bmatrix} \tag{2.6}$$

With given initial conditions r_0 and v_0, known a_d at each time, and known u at each time, the nonlinear dynamical system can be numerically integrated. In many guidance simulations, this model serves as the high-fidelity simulation model. Simplified models appropriate for developing guidance laws can be attained by making simplifying assumptions.

[9] Curtis, *Orbital Mechanics for Engineering Students*, 2021.

2.2 Flat Planet Model

With the two-body setting in mind, it is now assumed that the position of the second mass is near the surface of the first mass and does not move far from its initial location. In this case, the gravitational acceleration can be assumed constant in magnitude and direction, i.e., the quantity

$$g := -\frac{\mu}{||r||^3} r \approx -\frac{\mu}{||r_0||^3} r_0 \tag{2.7}$$

is constant where, again, r_0 denotes the initial position. When atmospheric forces are absent or negligible compared to gravitational and thrust forces, the flat planet model in first-order form is

$$\dot{r} = v, \qquad r(t_0) = r_0, \tag{2.8}$$
$$\dot{v} = g + u, \quad v(t_0) = v_0. \tag{2.9}$$

The obvious advantage of the flat planet model is that the equations of motion are linear with a constant gravitational disturbance. Upon defining the state vector $x = [r^\top, v^\top]^\top$, the flat planet equations of motion may be written in state-space form.

$$\dot{x} = Ax + Bu + Bg, \quad A = \begin{bmatrix} 0_3 & I_3 \\ 0_3 & 0_3 \end{bmatrix}, \quad B = \begin{bmatrix} 0_3 \\ I_3 \end{bmatrix} \tag{2.10}$$

The matrix 0_3 is the 3×3 zero matrix and I_3 is the 3×3 identity matrix.

Example 2.1. Consider the problem of guiding a vehicle to the surface of the earth. Mathematically, the guidance problem is to compute the thrust acceleration u required to drive the final position r_f to zero at the final time t_f. For the time being, assume that u is constant. Then, integrating forward from the initial time gives

$$v = g(t - t_0) + u(t - t_0) + v_0,$$
$$r = \frac{1}{2} g(t - t_0)^2 + \frac{1}{2} u(t - t_0)^2 + v_0(t - t_0) + r_0.$$

By setting $r(t_f) = 0$, one can solve for the required u. Upon defining $t_{go} = t_f - t_0$, the required thrust acceleration is

$$u = -\frac{2}{t_{go}^2} \left(\frac{1}{2} g t_{go}^2 + v_0 t_{go} + r_0 \right).$$

ℹ What happens as t_{go} (called the time-to-go) approaches zero? Within any guidance algorithm, care must be taken as $t_{go} \to 0$. This is the reason for the `if` statements within the guidance block implemented in Section 1.2.

Are there enough degrees of freedom to also hit a desired velocity target? How could degrees of freedom be added?

★

An answer to the last two questions is to assume a higher-order polynomial. This approach is called polynomial guidance, and it leads to guidance laws such as E-guidance and Apollo lunar descent guidance. These are discussed in more detail in Section 6.2. For now, assume the desired position trajectory is a cubic function of time, i.e.,

$$r = r_0 + v_0(t - t_0) + c_2(t - t_0)^2 + c_3(t - t_0)^3. \tag{2.11}$$

Differentiation gives the quadratic velocity profile

$$v = v_0 + 2c_2(t - t_0) + 3c_3(t - t_0)^2. \tag{2.12}$$

Given a predetermined flight time t_f, position target $r_f = r(t_f)$, and velocity target $v_f = v(t_f)$, the unknown coefficients $c_2 \in \mathbb{R}^3$ and $c_3 \in \mathbb{R}^3$ can be resolved. In matrix form,

$$\begin{bmatrix} r_f - r_0 - v_0 t_{go} \\ v_f - v_0 \end{bmatrix} = \begin{bmatrix} t_{go}^2 I_3 & t_{go}^3 I_3 \\ 2t_{go} I_3 & 3t_{go}^2 I_3 \end{bmatrix} \begin{bmatrix} c_2 \\ c_3 \end{bmatrix}. \tag{2.13}$$

Provided $t_{go} \neq 0$, the linear system has a unique solution. Differentiating the velocity gives the linear acceleration profile in terms of the now known coefficients.

$$a = 2c_2 + 6c_3(t - t_0) = g + u \tag{2.14}$$

Solving for the required thrust acceleration yields

$$u = 2c_2 + 6c_3(t - t_0) - g. \tag{2.15}$$

Within guidance, the time t_0 serves as the time at which the guidance system is being called. Care must be taken as t_{go} approaches zero. This phase of guidance is commonly called the close-out phase. In the implementation below, the thrust is set to zero during the close-out phase.

MATLAB Implementation of Polynomial Guidance

In this implementation, a planar landing scenario is simulated. The goal is to drive the vehicle from an initial position `r0` and velocity `v0` to a final position `rf` and velocity `vf` in a predetermined amount of time `tf`. For landing, `rf` and `vf` are zero.

```
% Data
r0 = [1000; 1000];      % m
v0 = [-25; 0];          % m/s
rf = [0;0];             % m
vf = [0;0];             % m/s
tf = 100;               % s
t  = linspace(0,tf,1e3)';
```

With this data and a function `ode` to be specified shortly, a simulation can be run using MATLAB's built-in integrator `ode45`.

```
% Guidance Simulation
[~,x] = ode45(@ode,t,[r0;v0;0;0],[],rf,vf,tf);
figure, plot(x(:,1),x(:,2)), grid on
xlabel('Range (m)'), ylabel('Altitude (m)')
```

This block of code simply completes the integration and generates a plot of the trajectory in position space. The function to be integrated is now specified and called `ode`. Included in this function are a navigation block, guidance block, control block, and dynamics block.

```
12  function [xdot,u,u_cmd] = ode(t,x,rf,vf,tf)
13  r = x(1:2);
14  v = x(3:4);
15  u = x(5:6);
16  g = [0;-9.81];
17
18  % Navigation Block
19  rhat = r + .02*r*sin(2*t);
20  vhat = v + .02*v*cos(3*t);
21
22  % Guidance Block
23  I = eye(2); tgo = tf-t;
24  if tgo >= .1
25      C = [tgo^2*I, tgo^3*I; 2*tgo*I, 3*tgo^2*I] \ ...
26          [rf-rhat-vhat*tgo; vf-vhat];
27      c2 = C(1:2); c3 = C(3:4);
28      u_cmd = 2*c2 + 6*c3*(t-t)-g;
29  else
30      u_cmd = [0;0];
31  end
32
33  % Control Block
34  sigma = 10;
35  udot = sigma*u_cmd - sigma*u;
36
37  % Dynamics Block
38  rdot = v;
39  vdot = u + g - 1/2*1*1.5*10*norm(v)*v/1000;
40  xdot = [rdot; vdot; udot];
```

Lines 13-15 define the position, velocity, and control states. Line 16 defines the gravity vector. The navigation block provides a state estimate. It is not perfect; it has been corrupted by a time-varying signal with magnitude $\pm 2\%$ of the true value. The guidance block implements the polynomial guidance law and outputs a commanded thrust acceleration `u_cmd`. When the time-to-go is less than 0.1 seconds, the engines shut off and the commanded acceleration is zero. Note in line 28 that the linear term in the control is zero because the `c3` coefficient multiplies the difference between the simulation time `t` and the time at which guidance is being called, which is also `t`. The control block generates the actual thrust acceleration assuming a first-order actuator model. Finally, the "high-fidelity" equations of motion appear in the dynamics block. See that there is a perturbing acceleration modeled as quadratic drag. The drag term appears in the dynamics block but is unknown to the guidance law. Additional gravitational, atmospheric, and stochastic perturbations can be included in line 39.

Running the code shows that the guidance law drives the state close to zero position and velocity. This is seen in Figure 2.2. The state trajectory begins in the upper right and descends until it terminates at the origin.

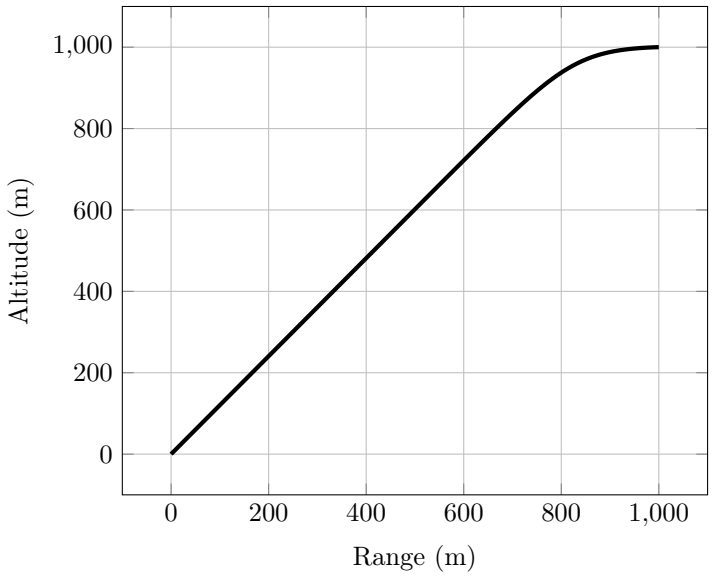


Figure 2.2: State trajectory generated by polynomial guidance.

With the code in its current form, spend some time trying to "break" the guidance law. Things to try include adding a bias in the navigation output, slowing the actuator response, making more dramatic changes to the dynamical model, restricting the flight time or max acceleration, and so on. Like Q-guidance, this guidance law works well over a wide array of imperfections, but there is a point past which it will fail. As a guidance engineer, it is important to know where this point is.

2.3 Relative Orbital Motion

The problem of two spacecraft orbiting near each other is now considered. By assuming the spacecraft are sufficiently close, the equations describing their relative motion are linear. Linearity is advantageous because it facilitates integration, analysis, and control design. A subscript 0 was used previously to denote an initial condition. In this section, a subscript 0 denotes the target spacecraft, i.e., r_0 is the position of the target spacecraft and v_0 is the velocity of the target spacecraft. These values change with time because the target spacecraft is orbiting the earth according to the two-body problem. The position and velocity of the so-called chaser spacecraft are r and v, respectively. This spacecraft is also orbiting the earth according to the two-body problem with thrust acceleration $u \in \mathbb{R}^3$. The equations of motion for each spacecraft relative to the central body are

$$\ddot{r} = -\frac{\mu}{\|r\|^3} r + u \quad \text{and} \quad \ddot{r}_0 = -\frac{\mu}{\|r_0\|^3} r_0. \tag{2.16}$$

Assuming that the masses of the spacecraft are much smaller than the mass of the earth, the gravitational effects of the two spacecraft upon each other can be neglected. The relative position of the chaser with respect to the target is $\delta = r - r_0$. The relative position vector is shown in Figure 2.3.

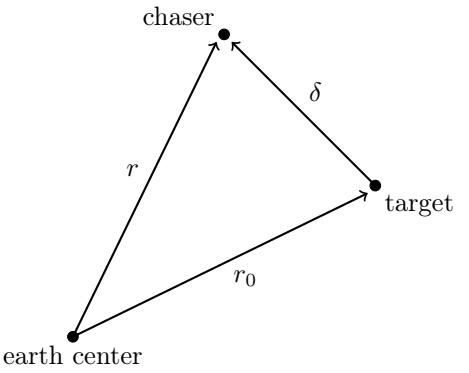


Figure 2.3: Relative orbital position diagram.

The equation of motion for the chaser relative to the target is then

$$\ddot{\delta} = -\ddot{r}_0 - \frac{\mu}{||r||^3}(r_0 + \delta) + u. \tag{2.17}$$

Observe that

$$\begin{aligned} ||r||^2 &= r \cdot r = (r_0 + \delta) \cdot (r_0 + \delta) && (2.18a)\\ &= r_0 \cdot r_0 + 2(r_0 \cdot \delta) + \delta \cdot \delta && (2.18b)\\ &= ||r_0||^2 + 2(r_0 \cdot \delta) + ||\delta||^2 && (2.18c)\\ &= ||r_0||^2 \left[1 + \frac{2(r_0 \cdot \delta)}{||r_0||^2} + \left(\frac{||\delta||}{||r_0||}\right)^2\right]. && (2.18d) \end{aligned}$$

It is assumed that $\frac{||\delta||}{||r_0||} << 1$ such that the last term can be neglected, i.e., the distance between the spacecraft is much smaller than the distance between the target and the center of the earth. Eq. (2.18) implies the following relations

$$\begin{aligned} &\implies ||r||^2 \approx ||r_0||^2 \left[1 + \frac{2(r_0 \cdot \delta)}{||r_0||^2}\right] && (2.19a)\\ &\implies ||r||^{-3} \approx ||r_0||^{-3} \left[1 + \frac{2(r_0 \cdot \delta)}{||r_0||^2}\right]^{-\frac{3}{2}}. && (2.19b) \end{aligned}$$

Expanding Eq. (2.19b) using the binomial theorem and keeping only first-order terms in δ yields

$$||r||^{-3} \approx ||r_0||^{-3} \left[1 - \frac{3}{||r_0||^2}(r_0 \cdot \delta)\right] = \frac{1}{||r_0||^3} - \frac{3}{||r_0||^5}(r_0 \cdot \delta). \tag{2.20}$$

Substituting Eq. (2.20) into the Eq. (2.17) gives

$$\ddot{\delta} \approx \ddot{r}_0 - \mu \left[\frac{1}{||r_0||^3} - \frac{3}{||r_0||^5}(r_0 \cdot \delta) \right] (r_0 + \delta) + u. \tag{2.21}$$

Expanding and keeping only first-order terms gives

$$\ddot{\delta} \approx -\ddot{r}_0 - \mu \frac{r_0}{||r_0||^3} - \frac{\mu}{||r_0||^3} \left[\delta - \frac{3}{||r_0||^2}(r_0 \cdot \delta) r_0 \right] + u. \tag{2.22}$$

Assuming that $\ddot{r}_0 = -\frac{\mu}{||r_0||^3} r_0$, it follows that

$$\ddot{\delta} \approx -\frac{\mu}{||r_0||^3} \left[\delta - \frac{3}{||r_0||^2}(r_0 \cdot \delta) r_0 \right] + u. \tag{2.23}$$

The coordinate directions associated with δ and its derivates are inherited from the inertial frame. By assuming that the target moves in a circular orbit and attaching a local vertical local horizontal (LVLH) frame to the target, these equations can be transformed into the Clohessy-Wiltshire (CW) equations of relative motion. The local vertical direction is defined as

$$\hat{\imath} = \frac{r_0}{||r_0||}, \tag{2.24}$$

and the local vertical position and velocity are $x \in \mathbb{R}$ and $\dot{x} \in \mathbb{R}$, respectively. The local horizontal direction is

$$\hat{\jmath} = \frac{v_0}{||v_0||}. \tag{2.25}$$

The local horizontal position and velocity are $y \in \mathbb{R}$ and $\dot{y} \in \mathbb{R}$. The out-of-plane direction is defined as

$$\hat{k} = \frac{r_0 \times v_0}{||r_0 \times v_0||}, \tag{2.26}$$

and the out-of-plane position and velocity are $z \in \mathbb{R}$ and $\dot{z} \in \mathbb{R}$. Given a δ whose coordinate directions are aligned with those of the inertial frame, it can be transformed to the LVLH frame using

$$x = \delta \cdot \hat{\imath}, \tag{2.27a}$$
$$y = \delta \cdot \hat{\jmath}, \tag{2.27b}$$
$$z = \delta \cdot \hat{k}. \tag{2.27c}$$

The velocity transformation must account for the fact that the LVLH frame is rotating with the target spacecraft with angular velocity $\Omega = \omega \hat{k}$ where $\omega^2 = \frac{\mu}{||r_0||^3}$.

$$\dot{x} = (\dot{\delta} - \Omega \times \delta) \cdot \hat{\imath} \tag{2.28a}$$
$$\dot{y} = (\dot{\delta} - \Omega \times \delta) \cdot \hat{\jmath} \tag{2.28b}$$
$$\dot{z} = (\dot{\delta} - \Omega \times \delta) \cdot \hat{k} \tag{2.28c}$$

The target is assumed to be in a circular orbit such that $\dot{\Omega} = 0$ and the acceleration transformation is given by the following equations.

$$\ddot{x} = (\ddot{\delta} - \Omega \times \dot{\delta}) \cdot \hat{\imath} + (\dot{\delta} - \Omega \times \delta) \cdot (\Omega \times \hat{\imath}) \tag{2.29a}$$
$$\ddot{y} = (\ddot{\delta} - \Omega \times \dot{\delta}) \cdot \hat{\jmath} + (\dot{\delta} - \Omega \times \delta) \cdot (\Omega \times \hat{\jmath}) \tag{2.29b}$$
$$\ddot{z} = (\ddot{\delta} - \Omega \times \dot{\delta}) \cdot \hat{k} + (\dot{\delta} - \Omega \times \delta) \cdot (\Omega \times \hat{k}) \tag{2.29c}$$

Plugging in Eq. (2.23) and simplifying terms yields the CW equations of relative motion

$$\ddot{x} = 3\omega^2 x + 2\omega\dot{y} + u_x \tag{2.30a}$$
$$\ddot{y} = -2\omega\dot{x} + u_y \tag{2.30b}$$
$$\ddot{z} = -\omega^2 z + u_z \tag{2.30c}$$

where u_x, u_y, and u_z are the external control, or thrust, accelerations in the $\hat{\imath}$, $\hat{\jmath}$, and $\hat{k}$ directions, respectively. Upon defining the state vector $s = [x, y, z, \dot{x}, \dot{y}, \dot{z}]^\top$, the CW equations may be written in state-space form.

$$\dot{s} = As + Bu, \quad A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 3\omega^2 & 0 & 0 & 0 & 2\omega & 0 \\ 0 & 0 & 0 & -2\omega & 0 & 0 \\ 0 & 0 & -\omega^2 & 0 & 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.31}$$

The CW equations are linear and time-invariant. The out-of-plane z motion is decoupled and a harmonic oscillator. The in-plane x-y motion is coupled. The control u appears linearly. These equations have been derived by linearization.

A natural problem arises in physics as to whether or not there exists a function L so that the Euler-Lagrange equations generate the CW equations. Such a problem is referred to as the inverse problem of Lagrange. Choosing

$$U = -\frac{3}{2}\omega^2 x^2 - 2\omega\dot{y}x + \frac{1}{2}\omega^2 z^2 \quad \text{and} \quad T = \frac{1}{2}(\dot{x}^2 + \dot{y}^2 + \dot{z}^2), \tag{2.32}$$

it is a simple matter to show that $L = T - U$ in the Euler-Lagrange equations generates the correct equations of motion. Moreover, it can be shown that the CW equations with zero control admit five constants of motion.[10]

$$\psi_1 = -6\omega x - 3\dot{y} \tag{2.33a}$$
$$\psi_2 = 4\dot{x}^2 + (6\omega x + 4\dot{y})^2 \tag{2.33b}$$
$$\psi_3 = \omega^2 z^2 + \dot{z}^2 \tag{2.33c}$$
$$\psi_4 = \tan\left(\omega \frac{y - \frac{2}{\omega}\dot{x}}{-6\omega x - 3\dot{y}} - \tan^{-1}\frac{6\omega x + 4\dot{y}}{-2\dot{x}}\right) \tag{2.33d}$$
$$\psi_5 = \tan\left(\tan^{-1}\frac{6\omega x + 4\dot{y}}{-2\dot{x}} - \tan^{-1}\frac{\dot{z}}{-\omega z}\right) \tag{2.33e}$$

[10] Sinclair and Hurtado, *The Motion Constants of Linear Autonomous Dynamical Systems*, 2013.

Also in the zero control case, the CW equations can be integrated analytically to arrive at the following formulas.

$$x = (4 - 3\cos\omega t)\, x_0 + \tfrac{\dot{x}_0}{\omega} \sin\omega t + \tfrac{2}{\omega}\,(1 - \cos\omega t)\, \dot{y}_0 \tag{2.34a}$$

$$y = 6\,(\sin\omega t - \omega t)\, x_0 + y_0 + \tfrac{2}{\omega}\,(\cos\omega t - 1)\, \dot{x}_0 + \tfrac{1}{\omega}\,(4\sin\omega t - 3\omega t)\, \dot{y}_0 \tag{2.34b}$$

$$z = z_0 \cos\omega t + \tfrac{\dot{z}_0}{\omega} \sin\omega t \tag{2.34c}$$

$$\dot{x} = 3\omega x_0 \sin\omega t + \dot{x}_0 \cos\omega t + 2\dot{y}_0 \sin\omega t \tag{2.34d}$$

$$\dot{y} = 6\omega\,(\cos\omega t - 1)\, x_0 - 2\dot{x}_0 \sin\omega t + (4\cos\omega t - 3)\, \dot{y}_0 \tag{2.34e}$$

$$\dot{z} = -\omega z_0 \sin\omega t + \dot{z}_0 \cos\omega t \tag{2.34f}$$

Observe that most of the terms are periodic (constant, sin, or cos). However, the y equation has two coefficients that drift with time. Purposeful selection of initial conditions cancels the drift terms so that the resulting motion is periodic. By factoring out the initial conditions in Eq. (2.34), the equations can be written in matrix form.

$$\left[\begin{array}{c} x \\ y \\ z \\ \hline \dot{x} \\ \dot{y} \\ \dot{z} \end{array}\right] = \left[\begin{array}{c|c} & \\ \Phi_{rr}(t) & \Phi_{rv}(t) \\ & \\ \hline & \\ \Phi_{vr}(t) & \Phi_{vv}(t) \\ & \end{array}\right] \left[\begin{array}{c} x_0 \\ y_0 \\ z_0 \\ \hline \dot{x}_0 \\ \dot{y}_0 \\ \dot{z}_0 \end{array}\right] \tag{2.35}$$

Each of the sub-matrices Φ_{**} is 3×3. The 6×6 matrix

$$\Phi(t) = \left[\begin{array}{c|c} \Phi_{rr}(t) & \Phi_{rv}(t) \\ \hline \Phi_{vr}(t) & \Phi_{vv}(t) \end{array}\right] \tag{2.36}$$

is called the state transition matrix for the CW system. The state transition matrix concept of mapping the state at one time to the state at another time through a linear matrix multiplication is true more generally. Consider the linear system $\dot{x} = A(t)x$ where A is a continuous function of time. For any t_0, x_0 there is a unique continuously differentiable solution

$$x = \Phi(t, t_0)\, x_0. \tag{2.37}$$

Again, the matrix Φ is called the state transition matrix (STM) for A. Some texts refer to this as the "fundamental" matrix.

There are many methods for computing the state transition matrix that can be found in most books on linear system theory.[11] It is commonly defined in terms of the

[11] Rugh, *Linear System Theory*, 1995.

Peano-Baker series. When the system matrix A is constant, the series simplifies so that the state transition matrix is simply the matrix exponential $\Phi(t, t_0) = e^{A(t-t_0)}$. The general state transition matrix also satisfies some interesting properties.

- $\frac{d}{dt}\Phi(t, t_0) = A(t)\,\Phi(t, t_0)$ (2.38a)
- $\Phi(t_0, t_0) = I$ (2.38b)
- $\Phi(t, t_0) = \Phi^{-1}(t_0, t)$ (2.38c)
- $\Phi_{-A^\top}(t, t_0) = \Phi_A^{-\top}(t, t_0) = \Phi_A^{\top}(t_0, t)$ (2.38d)
- $\Phi(t_2, t_0) = \Phi(t_2, t_1)\,\Phi(t_1, t_0)$ (2.38e)

A controlled linear system may be written in the following standard form

$$\dot{x} = A(t)x + B(t)u \tag{2.39}$$

where A is the system matrix, B is the control influence matrix, x is the state, and u is the control. The general solution to the forced (controlled) linear system is given by

$$x = \Phi(t, t_0)x_0 + \int_{t_0}^{t} \Phi(t, \sigma)\, B(\sigma)\, u(\sigma) d\sigma. \tag{2.40}$$

To this point, only continuous-time systems have been discussed because these naturally arise in physics. However, the nature of guidance is discrete because the guidance system is called at some frequency. To discretize continuous-time systems, discretize time.

$$t_0 < \cdots < t_k < t_{k+1} < \cdots < t_f \tag{2.41}$$

Assuming that on every interval $[t_k, t_{k+1}]$ the control is held constant at the initial value $u(t_k)$, it follows that

$$x(t_{k+1}) = \Phi(t_{k+1}, t_k)x(t_k) + \int_{t_k}^{t_{k+1}} \Phi(t_{k+1}, \sigma)B(\sigma)\, d\sigma\; u(t_k). \tag{2.42}$$

The state transition matrix $\Phi(t_{k+1}, t_k)$ is the discrete-time system matrix A_k. The integral pre-multiplying $u(t_k)$ is the discrete-time control influence matrix B_k. With $x_k = x(t_k)$ and $u_k = u(t_k)$, the discrete-time system is

$$x_{k+1} = A_k x_k + B_k u_k. \tag{2.43}$$

By using the state transition matrix, it is possible to convert continuous-time systems to discrete-time systems. This discretization is exact under the assumption of piecewise constant control functions. Discretization of nonlinear systems is discussed in Chapter 4.

MATLAB Implementation of LQR Guidance

To explore the discrete nature of guidance and the role of the discretized dynamics, an orbital rendezvous problem is now considered. Relative motion in low earth orbit (LEO) is unintuitive. With the chaser a short distance behind the target, thrusting to accelerate in the forward ($+y$) direction increases energy, raises the semi-major axis of the orbit, increases the orbital period, and has the effect of moving the chaser up

and back relative to the target. For this reason, early attempts during the Gemini era to pilot in a relative motion setting were unsuccessful. To illustrate this difficulty, a simulation is built requiring manual (pilot) inputs. The initial state vector is `x0`. The first three elements are relative positions (m) and the last three elements are relative velocities (m/hr) in the LVLH frame.

```
% Data
x0 = [0;-10;0;0;0;0];
```

The simulation loop is the following. Lines 6-10 are initialization lines. Line 11 begins the `for` loop, which then prints position and time information to the screen in lines 12 and 13, prompts the user for a thrust acceleration command in line 14, and integrates in line 15. Lines 16-21 store information and generate an animated plot to see the result of the user's command.

```
%-------------------------%
% Loop with 'pilot' input %
%-------------------------%
X = [];
T = [];
x(1,:) = x0.';
fprintf('\nHuman Pilot Scenario \n\n')
figure
for i = 1:10
 fprintf('Current pos. is [%4.2f,%4.2f] m. ',x(end,1),x(end,2));
 fprintf('Time is %4.2f s. ',i-1);
    u = input('u = '); % Input as column [ux;uy;uz]
    [t,x] = ode45(@ode,[i-1,i],x(end,:),[],u);
    X = [X; x];
    T = [T; t];
    for j = 1:length(t)
        plot(x(j,2),x(j,1),'ko','MarkerSize',5), hold on, grid on
        axis([-20 20 -20 20])
        pause(0.01)
    end
end
```

The simulation is simply an integration of the CW equations in Eq. (2.31). The `ode` function is given below with a mean motion of $\omega = 4$ rad/hour corresponding to a circular orbit with period $\pi/2$ hours suitable for LEO.

```
function xdot = ode(t,x,u)
w = 4; % rad/hr
A = [0, 0, 0, 1, 0, 0;
    0, 0, 0, 0, 1, 0;
    0, 0, 0, 0, 0, 1;
    3*w^2, 0, 0, 0, 2*w, 0;
    0, 0, 0, -2*w, 0, 0;
    0, 0, -w^2, 0, 0, 0];
B = [zeros(3,3); eye(3,3)];
```

```
57  xdot = A*x+B*u;
58  end
```

Running the code prompts the user for a control input of the form `[ux;uy;uz]`. Try several times to pilot the state to the origin for rendezvous. If successful, change the initial state and try again. The simulation terminates after ten inputs.

To run the simulation with a guidance system or "autopilot", add the following lines of code. Line 24 simply pauses the run requiring the user to hit enter to proceed. Lines 28-33 are initialization lines. The loop begins on line 34. The user is no longer prompted for a thrust acceleration command. A function is used on line 35 to generate the command. The loop then proceeds as before by completing the integration, storing information, and animating a plot.

```
24  keyboard
25  %-----------------------%
26  % Loop with 'autopilot' %
27  %-----------------------%
28  clear x
29  X = [];
30  T = [];
31  x(1,:) = x0.';
32  fprintf('\n\nAuto Pilot Scenario \n\n')
33  figure
34  for i = 1:10
35    u = getControl( x(end,:).' );
36    fprintf('Current pos. is [%4.2f,%4.2f] m. ',x(end,1),x(end,2));
37    fprintf('Time is %4.2f s. ',i-1);
38    fprintf('u = [%4.2f,%4.2f,%4.2f].\n',u(1),u(2),u(3)); pause(2)
39    [t,x] = ode45(@ode,[i-1,i],x(end,:),[],u);
40    X = [X; x];
41    T = [T; t];
42    for j = 1:length(t)
43        plot(x(j,2),x(j,1),'ko','MarkerSize',5), hold on, grid on
44        axis([-20 20 -20 20])
45        pause(0.01)
46    end
47  end
```

How exactly the `getControl` function works is left for later discussion on discrete optimal control in Chapter 4. Notwithstanding, the function is below.

```
59  function u = getControl(x)
60  w = 4; % rad/hr
61  A = [0, 0, 0, 1, 0, 0;
62       0, 0, 0, 0, 1, 0;
63       0, 0, 0, 0, 0, 1;
64       3*w^2, 0, 0, 0, 2*w, 0;
65       0, 0, 0, -2*w, 0, 0;
66       0, 0, -w^2, 0, 0, 0];
67  B = [zeros(3,3); eye(3,3)];
```

```
68  Q = diag([10;1;1;10;1;1]);
69  K = lqrd(A,B,Q,eye(3),1);
70  u = -K*x;
71  end
```

Included in the function are the mean motion `w`, system matrix `A`, control influence matrix `B`, weight matrix `Q`, gain matrix `K`, and commanded thrust acceleration `u`. The built-in MATLAB function `lqrd` generates the linear quadratic regulator (LQR) gain matrix for the discrete-time system assuming a one-second discretization and constant hold on the control. Running the code shows that the autopilot has no trouble achieving rendezvous for various initial conditions as seen in Figure 2.4. The trajectory starts on the left and moves to the right terminating at the origin. The vertical position never deviates beyond ± 1 m.

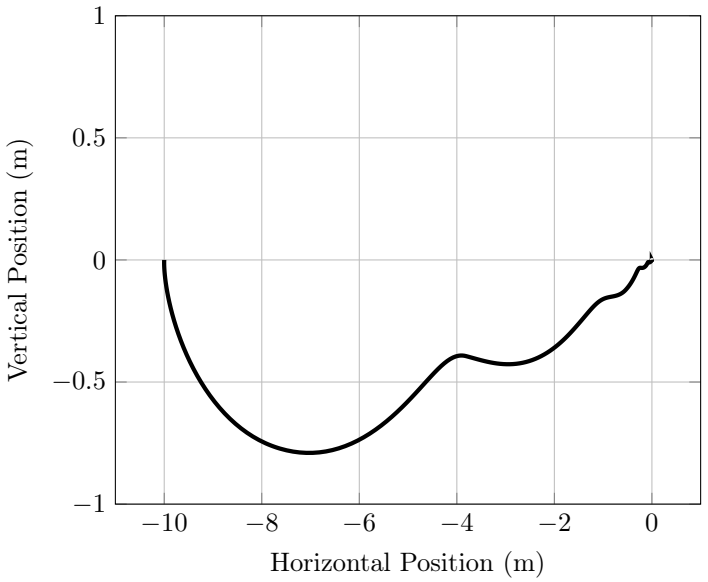


Figure 2.4: State trajectory generated by LQR guidance.

2.4 Mass Dynamics

Thus far, state variables have been positions and velocities. Another key state is mass, m. Throughout the book, mass dynamics are modeled with the equation

$$\dot{m} = -\frac{\|T\|}{I_{sp}\, g_0}. \tag{2.44}$$

thrust force magnitude

engine's specific impulse (sec)

sea-level acceleration of gravity on Earth

For any of the previous examples and guidance implementations, this differential equation can be appended to the others to compute the fuel mass consumed using the guidance law. The specific impulse depends upon the propellant type. Typical values

are about 50 seconds for cold gas, 300 seconds for solid and monoprop, and 3000 seconds for ion propulsion.

ℹ Note that the equation is nonlinear in the thrust vector because

$$\|T\| = \left[T_x^2 + T_y^2 + T_z^2\right]^{1/2}. \tag{2.45}$$

Moreover, because

$$u = \frac{T}{m}, \tag{2.46}$$

problems with mass dynamics are nonlinear.

Solid rockets provide a constant thrust magnitude. In this case, the mass varies linearly with time

$$m = m_0 - \frac{\|T\|}{I_{sp}\, g_0} t. \tag{2.47}$$

This is useful because the $\dot{m}$ equation can be eliminated and the $\frac{T}{m}$ terms now appear linearly.

2.5 Chapter Problems

Problem 2.1. The initial state of the International Space Station (ISS) is given below.

```
[1622.39; 5305.10; 3717.44; -7.29977; 0.492357; 2.48318];
```

The first three elements are positions (in km). The last three elements are velocities (in km/s). Simulate the uncontrolled, undisturbed trajectory for five orbital periods by integrating Eq. (2.6) using a fixed-step RK4 integrator with 10-second step. Provide a three-dimensional plot of the ISS position to see that the solution is an elliptical orbit.

Problem 2.2. The initial state of a spacecraft in proximity to the ISS is given below.

```
[1612.75; 5310.19; 3750.33; -7.35321; 0.463856; 2.46920];
```

As in the previous problem, units are km and km/s. Using the same parameters from Problem 2.1, simulate the uncontrolled, undisturbed two-body motion of the spacecraft. Provide a three-dimensional plot of the spacecraft position and the ISS position on the same figure.

Problem 2.3. Building upon the previous two problems, consider the ISS as the target and the spacecraft as the chaser.

a) Compute the relative position and velocity in the inertial frame. Provide three-dimensional plots of the relative position and velocity.
b) Using Eq. (2.23), integrate the linearized differential equations for relative position and velocity. Overlay the linearized plots on those from part a).
c) Transform the linearized relative position and velocity to the LVLH frame using Eqs. (2.27) and (2.28). Note that the angular velocity vector is given by $\omega = 0.00115697\ \hat{k}$ rad/s. Provide three-dimensional plots of the relative position and relative velocity in the LVLH frame.
d) Use the analytical solutions to the CW equations in Eq. (2.34) to calculate the linearized relative position and velocity in the LVLH frame. Overlay the CW solutions on those from part c).

Problem 2.4. Consider two spacecraft in low earth orbit (LEO) in close proximity such that a linearized relative motion model is appropriate. The mean motion of the target orbit is $\omega = 4$ rad/hr. The chaser spacecraft is 1 km behind the target and has no relative velocity.

a) Show that the chaser spacecraft's initial state is a stationary point in the CW equations given by Eq. (2.31).
b) For a time step of one hour, compute the state transition matrix.
c) Compute a two-impulse maneuver so that the chaser spacecraft rendezvous with the target spacecraft at one hour. That is, at the initial and final times determine the instantaneous change in velocity required to drive the state to the origin.

Problem 2.5. Compute the discrete-time system matrices for the CW equations in Eq. (2.31) with $\omega = 4$ rad/hr. Use a step size of one second.

Problem 2.6. Compute the discrete-time system matrices for the flat planet model in Eq. (2.10). Use a step size of one second.

Problem 2.7. A lunar lander is at 200 m altitude and descending at 1 m/s. The flat planet model given by Eq. (2.10) is applicable. The goal is to impact the lunar surface with downward velocity of 1 m/s.

a) Assuming no thrust, at what speed will the lander impact the lunar surface?
b) What thrust acceleration can be applied to hit the surface with downward velocity of 1 m/s? How much time is required to achieve touchdown?
c) Simulate the descent using i) continuous-time dynamics and ii) discrete-time dynamics with one-second step. For each case, provide two-dimensional plots of the velocity as a function of altitude.

Problem 2.8. By differentiation, show that the general solution to the forced linear system given by Eq. (2.40) satisfies the differential equation $\dot{x} = Ax + Bu$.

Problem 2.9. For what matrix $A(t)$ is the state transition matrix given by

$$\Phi(t,\tau) = e^{-(t^2-\tau^2)} \begin{bmatrix} \cos(t-\tau) & -\sin(t-\tau) \\ \sin(t-\tau) & \cos(t-\tau) \end{bmatrix}.$$

Problem 2.10. Show that the linear time-varying system $\dot{x} = A(t)x$ can be transformed to a time-invariant system if and only if the state transition matrix for $A(t)$ can be written as $\Phi(t,0) = T(t)e^{Rt}$ where R is a constant matrix and $T(t)$ is an invertible matrix.

Problem 2.11. Verify the state transition matrix properties specified in Eq. (2.38) for the following matrix.

$$A = \begin{bmatrix} 0 & 1 \\ -1 & -2 \end{bmatrix}$$

Problem 2.12. Replicate the polynomial guidance implementation beginning on page 15. Replace the first-order actuator models with second-order actuator models of the form

$$\ddot{u} + 2\zeta\omega\dot{u} + \omega^2 u = \omega^2 u_{cmd}.$$

With the damping ratio $\zeta = 0.7$, examine the system response for values of ω between 0.1 and 20.

Problem 2.13. Replicate the LQR guidance implementation beginning on page 22.

(a) Practice your piloting skills in the manual entry loop.
(b) Change the positive values used on code line 68 and observe how the system response changes.

Chapter 3

Optimization

Chapter Learning Objectives

1. Motivate the study of optimization with a discrete-time control problem.
2. Understand optimality conditions for unconstrained and constrained optimization.
3. Understand the role convexity plays in strengthening optimality conditions.
4. Apply optimality conditions to solve optimization problems.

Having now an appreciation for dynamical systems in space applications, the topic of optimization is introduced so that the problems of designing optimal spacecraft trajectories and performing optimal spacecraft control can be solved. This chapter serves primarily as introductory material so the reader becomes familiar with the elements of an optimization problem, how to analyze it, and how to solve it. These carry over to discrete optimal control and continuous optimal control discussed in Chapters 4 and 5, respectively. The mathematical presentation is rigorous, but the theory of optimization is beyond the scope of this book.[12]

The elements of an optimization problem are an objective function, optimization (or decision) variable, inequality constraints, and equality constraints. Problems in this chapter are finite-dimensional meaning the decision variable is in $\mathbb{R}^n$. First, problems without constraints are considered. The reader may recall from calculus how to solve such problems: set the derivative equal to zero to find candidate points and pick the best candidate. Problems with inequality and equality constraints are then considered.

After discussing general problems and theories, convexity and convex optimization are introduced. It turns out that convex optimization problems are the ones "easily" solved. The optimality conditions are necessary and sufficient, and efficient algorithms exist for solving convex problems numerically. Convexity in optimal spacecraft guidance is playing an ever-increasing role for these reasons. It is the engineer's responsibility to formulate the spacecraft guidance problem intelligently so that it can be solved efficiently. This often means, at the very least, recognizing convexity so a convex solver is used and, more ambitiously, transforming a nonconvex problem into a convex one.

[12] Berkovitz, *Convexity and Optimization in* $\mathbb{R}^n$, 2002.

3.1 Motivating Problem

In Section 2.3 on linearized relative orbital motion, it was shown how to discretize a continuous-time linear system using the state transition matrix and assuming piecewise constant controls. The result is a discrete-time linear system of the form

$$x_{k+1} = Ax_k + Bu_k, \quad k = 0, ..., N-1. \tag{3.1}$$

The state at time index k is $x_k \in \mathbb{R}^n$. The control at time index k is $u_k \in \mathbb{R}^m$. By writing out a few terms, it can be seen how the final state at time index N depends on the initial state and control inputs.

$$x_1 = Ax_0 + Bu_0, \tag{3.2a}$$

$$\begin{aligned} x_2 &= Ax_1 + Bu_1 \\ &= A^2x_0 + ABu_0 + Bu_1, \end{aligned} \tag{3.2b}$$

$$\begin{aligned} x_3 &= Ax_2 + Bu_2 \\ &= A^3x_0 + A^2Bu_0 + ABu_1 + Bu_2, \end{aligned} \tag{3.2c}$$

$$\vdots$$

$$x_N = A^Nx_0 + \sum_{k=0}^{N-1} A^{(N-1-k)}B\,u_k. \tag{3.2d}$$

By stacking all of the controls into a tall $(Nm \times 1)$ vector

$$U = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}, \tag{3.3}$$

and defining the $n \times Nm$ matrix

$$C = \begin{bmatrix} A^{N-1}B, & A^{N-2}B, & \cdots, & AB, & B \end{bmatrix}, \tag{3.4}$$

Eq. (3.2d) can be written as

$$x_N - A^Nx_0 = CU. \tag{3.5}$$

For this linear algebraic equation to be solvable, it is required that the point $X = x_N - A^Nx_0$ be in the range or image space of C, i.e., $X \in \text{im}(C)$. Let U_p be some particular solution to the equation. Then the controls in U_p will drive the discrete-time system from its initial condition x_0 to the final condition x_N in N time steps. Suppose further that the matrix C has a non-trivial null space. If there is one solution to the equation, then there are infinitely many solutions. Let L be a matrix such that

$$\text{im}\,(L) = \text{null}\,(C)\,. \tag{3.6}$$

Then for any vector V with dimension equal to that of the null space, the vector

$$U = U_p + LV, \tag{3.7}$$

solves the system $X = CU$. That is, any such U drives the discrete-time system from x_0 to x_N in N steps.

If there are infinitely many control trajectories to drive x_0 to x_N, how should one be chosen? The answer, given away by the title of this chapter, is optimization. Common quantities to optimize in spacecraft guidance are fuel consumed, control effort, and flight time. In discrete-time settings, the flight time cannot be minimized directly because that time is associated with the fixed value of N.

3.2 Unconstrained Optimization

To optimize means to minimize or maximize. Graphically, the concepts of minima and maxima are intuitive and self-evident as seen in Figure 3.1.

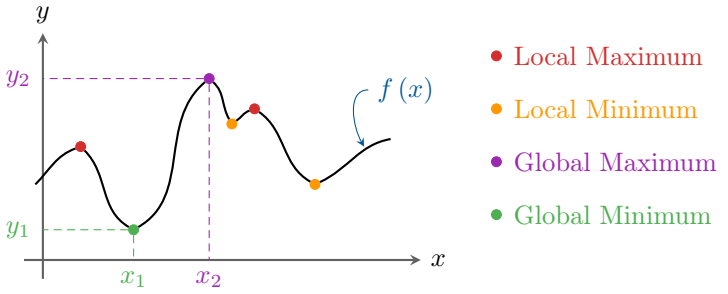
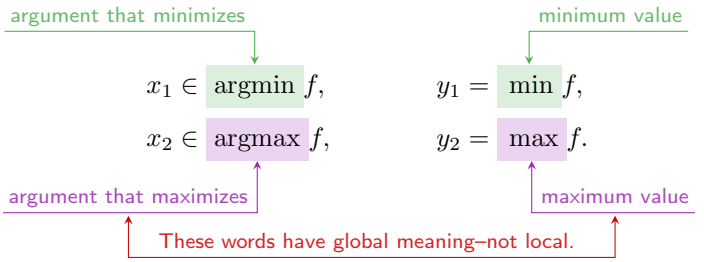


Figure 3.1: Graphical example of local and global maxima and minima.

Mathematically, the following nomenclature is used.



(3.8)

(3.9)

The argmin and argmax operators return sets, hence the $\in$ symbol. Generally speaking, it is global optima that are of interest. However, local optima also satisfy the optimality conditions presented herein. Care must be taken to correctly identify global optima. By drawing a few pictures, the following facts are evident.

$$\operatorname{argmin} f = \operatorname{argmax} -f\,, \tag{3.10}$$

$$\min f = -\max -f\,. \tag{3.11}$$

That is, points that minimize a function are the same points that maximize the negated function, and the minimum value of the function is related to the maximum value of the negated function. Therefore, only a theory for minimization problems is needed. From calculus, recall the following theorem.

Theorem 3.1. *Let $f : \mathbb{R} \to \mathbb{R}$ be differentiable. If $x \in \operatorname{argmin} f$, then $f'(x) = 0$.*

ℹ Any points that satisfy $f'(x) = 0$ are called candidates. All optimal points are candidates, but not all candidates are optimal points.

The theorem is a necessary condition that transforms the optimization problem into a root-finding problem. The following examples, all of which involve a single variable and an analytic function, show the many pitfalls one may encounter in trying to solve optimization problems. Think carefully about each example and draw a picture of the function to help.

Example 3.1.

$$\text{minimize} \qquad f(x) = x^2$$
$$f'(x) = 2x = 0 \quad \implies \quad x = 0$$

There is one candidate that globally minimizes the function. ★

Example 3.2.

$$\text{minimize} \qquad f(x) = x^3$$
$$f'(x) = 3x^2 = 0 \quad \implies \quad x = 0$$

There is one candidate that does not minimize (locally or globally) the function. ★

Example 3.3.

$$\text{minimize} \qquad f(x) = e^{-x}$$
$$f'(x) = -e^{-x} \neq 0$$

There are no candidates and hence no minima for the function. ★

Example 3.4.

$$\text{minimize} \qquad f(x) = (x-2)^2 (x+2)^2$$

Setting $f'(x) = 0$ implies that there are 3 candidates $-2,\ 0,\ +2$. Two of them are global minima. ★

Example 3.5.

$$\text{minimize} \qquad f(x) = \sin(x)$$
$$f'(x) = \cos(x) = 0 \quad \implies \quad x = \tfrac{\pi}{2} + \pi k$$

There are countably infinite candidates. The points $\frac{\pi}{2} + 2\pi k$ are global maxima. The points $\frac{3\pi}{2} + 2\pi k$ are global minima. There are countably infinite minima. ★

Example 3.6.

$$\text{minimize} \qquad f(x) = 1$$
$$f'(x) = 0 \text{ for all } x \in \mathbb{R}$$

There are uncountably many candidates that are global minima and global maxima. ★

The examples demonstrate cases in which the theory does not tell everything one would want to know. This fact is exacerbated in higher-dimensional problems and problems with constraints. There is a class of optimization problems for which the theory does identify global minima. These are convex problems, which are discussed in Section 3.5.

3.3 Constrained Optimization

Most of the problems arising in optimal spacecraft guidance have constraints arising from physics, thrust limits, and boundary conditions. Therefore, general nonlinear constrained optimization problems are now introduced.

$$\begin{array}{lllll} \text{minimize} & f(x) & \text{(objective)} & f:\mathbb{R}^n \to \mathbb{R} & \\ \text{subject to} & g(x) \leq 0 & \text{(inequality constraints)} & g:\mathbb{R}^n \to \mathbb{R}^p & (3.12) \\ & h(x) = 0 & \text{(equality contraints)} & h:\mathbb{R}^n \to \mathbb{R}^q & \end{array}$$

The problem is to find an $x \in \mathbb{R}^n$ that satisfies the p inequality constraints, q equality constraints, and minimizes the objective function f. The constraint set is defined to be

$$\mathcal{X} = \{x \in \mathbb{R}^n : g(x) \leq 0,\ h(x) = 0\}. \tag{3.13}$$

The optimization problem may now be written in compact forms.

$$\min_{x \in \mathcal{X}} f \quad \text{and} \quad x \in \operatorname*{argmin}_{x \in \mathcal{X}} f \tag{3.14}$$

It is evident that the constraint set is a subset of $\mathbb{R}^n$ and any optimal point must be a feasible point. As with unconstrained problems, there are necessary conditions to help identify candidate points, and these candidates may or may not correspond to minima. The Lagrangian for the problem is defined to be

$$L(x, \lambda_0, \lambda, \nu) = \lambda_0 f(x) + \lambda^\top g(x) + \nu^\top h(x). \tag{3.15}$$

Now the optimality conditions for the problem in Eq. (3.12) can be stated.

Theorem 3.2 (Optimality Conditions for Nonlinear Optimization)**.** *Let f, g, and h be continuously differentiable. If the problem attains a minimum at x, then there exist multipliers $\lambda \in \mathbb{R}^p$ and $\nu \in \mathbb{R}^q$ such that the following equations are satisfied.*

$$g(x) \leq 0 \tag{3.16a}$$

$$h(x) = 0 \tag{3.16b}$$

non-triviality condition

$$(\lambda_0, \lambda, \nu) \neq 0, \quad \lambda_0 \in \{0, 1\}, \quad \lambda \geq 0 \tag{3.16c}$$

abnormal multiplier

$$\lambda^\top g(x) = 0 \tag{3.16d}$$

complementarity condition

$$\nabla_x L(x, \lambda_0, \lambda, \nu) = 0 \tag{3.16e}$$

Solutions with $\lambda_0 = 0$ are "abnormal" solutions and those with $\lambda_0 = 1$ are "normal" solutions. Abnormality implies that the neighborhood of points surrounding the candidate is degenerate. The complementarity condition implies that either $\lambda_i = 0$ or $g_i(x) = 0$ for every $i = 1, ..., p$. What looks like one equation is actually p equations. This is a useful fact when solving problems by hand. The general strategy for solving problems is to set $\lambda_0 = 1$ and solve the remaining equations to arrive at normal candidates, set $\lambda_0 = 0$ and solve the remaining equations to arrive at abnormal candidates, and then determine minima by comparing objective values of all candidates. Several examples are now worked to illustrate the process.

Example 3.7.

$$\begin{aligned} \text{minimize} \quad & x^2 \\ \text{subject to} \quad & (x-1)^2 = 0 \end{aligned}$$

It is obvious that $x = 1$ is the answer since it is the only feasible point. To use the optimality conditions, the solution procedure is to first write down the Lagrangian

$$L = \lambda_0 x^2 + \nu (x-1)^2,$$

and compute its gradient

$$\nabla_x L = 2\lambda_0 x + 2\nu (x-1) = 0.$$

Suppose $\lambda_0 = 1$. From the gradient condition,

$$2x(1+\nu) = 2\nu \quad \Longrightarrow \quad x = \frac{\nu}{\nu+1}.$$

To be feasible, x must be equal to 1. Therefore,

$$\nu = \nu + 1 \quad \Longrightarrow \quad 0 = 1.$$

The obvious contradiction implies that there are no normal solutions. Suppose $\lambda_0 = 0$. From the gradient condition,

$$2\nu(x-1) = 0.$$

The non-triviality condition requires $\nu \neq 0$. Thus, $x = 1$. The global minimum is an abnormal solution. ★

Example 3.8.

$$\begin{aligned} \text{minimize} \quad & x_1^2 + x_2^2 + x_1 x_2 - 3x_1 \\ \text{subject to} \quad & x_1 \geq 0 \\ & x_2 \geq 0 \end{aligned}$$

The Lagrangian is

$$L = \lambda_0 \left(x_1^2 + x_2^2 + x_1 x_2 - 3x_1\right) - \lambda_1 x_1 - \lambda_2 x_2.$$

Components of the gradient are

$$\frac{\partial L}{\partial x_1} = 2\lambda_0 x_1 + \lambda_0 x_2 - 3\lambda_0 - \lambda_1 = 0, \tag{a}$$

$$\frac{\partial L}{\partial x_2} = 2\lambda_0 x_2 + \lambda_0 x_1 - \lambda_2 = 0. \tag{b}$$

The complementarity conditions are

$$\lambda_1 x_1 = 0, \quad \lambda_2 x_2 = 0.$$

From these complementarity conditions, there are four cases to investigate corresponding to the four ways in which they are satisfiable.

① Suppose $x_1 = x_2 = 0$.

$$\begin{aligned} \text{(a)} \quad &\Longrightarrow \quad 3\lambda_0 = -\lambda_1. \\ \text{(b)} \quad &\Longrightarrow \quad \lambda_2 = 0. \end{aligned}$$

If $\lambda_0 = 0$, then $\lambda_1 = 0$ violating the non-triviality condition. If $\lambda_0 = 1$, then $\lambda_1 = -3 \not\geq 0$. Case ① is ruled out.

② Suppose $\lambda_1 = \lambda_2 = 0$. The non-triviality condition tells us that $\lambda_0 = 1$.

$$\begin{aligned} \text{(a)} \quad &\Longrightarrow \quad 2x_1 + x_2 = 3. \\ \text{(b)} \quad &\Longrightarrow \quad x_1 + 2x_2 = 0. \end{aligned}$$

Solving this linear system gives $x_1 = 2$, $x_2 = -1$. This point is not feasible. Case ② is ruled out.

③ Suppose $x_1 = 0$ and $\lambda_2 = 0$.

$$\begin{aligned} \text{(a)} \quad &\Longrightarrow \quad \lambda_0 x_2 - 3\,\lambda_0 - \lambda_1 = 0. \\ \text{(b)} \quad &\Longrightarrow \quad 2\lambda_0 x_2 = 0 \\ &\Longrightarrow \quad \lambda_0 = 0 \quad \text{or} \quad x_2 = 0. \end{aligned}$$

If $\lambda_0 = 0$, then (a) gives $\lambda_1 = 0$ violating the non-triviality condition. If $x_2 = 0$, then $\lambda_1 = -3 \not\geq 0$. Case ③ is ruled out.

④ Suppose $\lambda_1 = 0$ and $x_2 = 0$.

$$\begin{aligned} \text{(a)} \quad &\Longrightarrow \quad 2\lambda_0 x_1 - 3\,\lambda_0 = 0. \\ \text{(b)} \quad &\Longrightarrow \quad \lambda_0 x_1 - \lambda_2 = 0. \end{aligned}$$

If $\lambda_0 = 0$, (b) gives $\lambda_2 = 0$ violating the non-triviality condition. If $\lambda_0 = 1$, then $x_1 = \frac{3}{2} = \lambda_2 \geq 0$. Case ④ yields a candidate. The optimality conditions are necessary. They are not sufficient. If there is a solution, this is it.

★

After all that work, it would be nice to have a conclusive answer. A useful tool in this regard is the Weierstrass Theorem.

Theorem 3.3 (Weierstrass Theorem)**.** *If f is continuous and $\mathcal{X}$ is closed and bounded, then f attains a minimum and maximum on $\mathcal{X}$.*

In the previous example, the domain is not closed and bounded. It can be made so by adding the artificial constraints

$$x_1 \leq \sigma, \quad x_2 \leq \sigma, \quad \sigma > \tfrac{3}{2}. \tag{3.17}$$

These constraints do not change the candidate points. The Weierstrass Theorem ensures that the candidate point $(\frac{3}{2}, 0)$ is optimal for any $\sigma \in (\frac{3}{2}, \infty)$. By placing the problem in a large, but finite box, a definite conclusion can be drawn. This artificial construction can be useful theoretically and numerically.

Example 3.9.

$$\begin{array}{lcccl} \text{maximize} & (x-1)^2 & \longrightarrow & \text{minimize} & -(x-1)^2 \\ \text{subject to} & x \leq 2 & & \text{subject to} & x \leq 2 \end{array}$$

The Lagrangian and its gradient are

$$L = -\lambda_0 (x-1)^2 + \lambda (x-2), \tag{3.18}$$
$$\nabla_x L = -2\lambda_0 (x-1) + \lambda = 0. \tag{3.19}$$

If $\lambda_0 = 0$, the gradient condition implies $\lambda = 0$ violating the non-triviality condition. Thus, $\lambda_0 = 1$. If $\lambda = 0$, then $x = 1 \leq 2$. If $x = 2$, then $\lambda = 2 \geq 0$. There are two candidates.

ℹ Neither candidate is optimal because $f(x) \to -\infty$ as $x \to -\infty$. This example illustrates difficulties that can arise when the hypotheses of the Weierstrass Theorem fail to hold. It is important to always keep in mind the following. If a solution exists, the solution is a candidate. If a solution does not exist, no candidate is a solution.

★

Example 3.10.

$$\begin{array}{ll} \text{minimize} & (x_1-1)^2 + x_2 - 2 \\ \text{subject to} & x_1 + x_2 - 2 \leq 0 \\ & x_2 - x_1 - 1 = 0 \end{array}$$

The Lagrangian is

$$L = \lambda_0 (x_1-1)^2 + \lambda_0 x_2 - 2\lambda_0 + \lambda (x_1 + x_2 - 2) + \nu (x_2 - x_1 - 1).$$

Components of the gradient are

$$\frac{\partial L}{\partial x_1} = 2\lambda_0 (x_1 - 1) + \lambda - \nu = 0, \tag{a}$$
$$\frac{\partial L}{\partial x_2} = \lambda_0 + \lambda + \nu = 0. \tag{b}$$

The complementarity condition is

$$\lambda (x_1 + x_2 - 2) = 0.$$

Suppose that $\lambda_0 = 0$.

$$\begin{array}{lcl} \text{(a)} & \Longrightarrow & \lambda = \nu. \\ \text{(b)} & \Longrightarrow & \lambda = -\nu. \end{array}$$

Thus, $\lambda = \nu = 0$. This violates the non-triviality condition. Therefore, $\lambda_0 = 1$.

Suppose $\lambda = 0$.

$$\begin{array}{lcl} \text{(a)} & \Longrightarrow & 2(x_1 - 1) = -1 \quad \Longrightarrow \quad x_1 = \dfrac{1}{2}. \\ \text{(b)} & \Longrightarrow & \nu = -1. \end{array}$$

The equality constraint gives $x_2 = \frac{3}{2}$. Substituting into the inequality constraint gives $\frac{1}{2} + \frac{3}{2} - 2 = 0 \leq 0$. Therefore, $(x_1, x_2) = \left(\frac{1}{2}, \frac{3}{2}\right)$ is a candidate.

Suppose that $x_1 + x_2 - 2 = 0$. Together with the equality constraint,

$$\begin{aligned} x_1 + x_2 &= 2, \\ x_2 - x_1 &= 1. \end{aligned}$$

The solution to this system is again $(x_1, x_2) = \left(\frac{1}{2}, \frac{3}{2}\right)$. There is one candidate. ★

Example 3.11.

$$\begin{aligned} \text{minimize} \quad & x_1^2 + 4x_2^2 \\ \text{subject to} \quad & x_1^2 + 2x_2^2 \geq 4 \end{aligned}$$

The Lagrangian is

$$L = \lambda_0 x_1^2 + 4\lambda_0 x_2^2 + \lambda\left(-x_1^2 - 2x_2^2 + 4\right).$$

Components of the gradient are

$$\frac{\partial L}{\partial x_1} = 2\lambda_0 x_1 - 2\lambda x_1 = 0, \tag{a}$$

$$\frac{\partial L}{\partial x_2} = 8\lambda_0 x_2 - 4\lambda x_2 = 0. \tag{b}$$

The complementarity condition is

$$\lambda\left(-x_1^2 - 2x_2^2 + 4\right) = 0.$$

Suppose that $\lambda_0 = 0$. Then $\lambda \neq 0$, or else it would violate the non-triviality condition. Therefore, $x_1 = x_2 = 0$. But this does not satisfy the inequality constraint. Therefore, $\lambda_0 = 1$.

Suppose $\lambda = 0$. Then again, $x_1 = x_2 = 0$ which is impossible. Therefore, $x_1^2 + 2x_2^2 = 4$.

$$\begin{aligned} \text{(a)} \quad &\Longrightarrow \quad 2x_1(1 - \lambda) = 0. \\ \text{(b)} \quad &\Longrightarrow \quad 4x_2(2 - \lambda) = 0. \end{aligned}$$

If $\lambda = 1$, then $x_2 = 0$ and $x_1 = \pm 2$. The objective value at these points is $f = 4$. If $\lambda = 2$, then $x_1 = 0$ and $x_2 = \pm\sqrt{2}$. The objective value at this point is $f = 8$. There are two candidates: $(x_1, x_2) = (\pm 2, 0)$. ★

Example 3.12.

$$\begin{aligned} \text{minimize} \quad & x_2 - (x_1 - 2)^3 + 3 \\ \text{subject to} \quad & x_2 \geq 1 \end{aligned}$$

The Lagrangian is

$$L = \lambda_0 x_2 - \lambda_0 (x_1 - 2)^3 + 3\lambda_0 + \lambda(1 - x_2).$$

Components of the gradient are

$$\frac{\partial L}{\partial x_1} = -3\lambda_0 (x_1 - 2)^2 = 0, \tag{a}$$
$$\frac{\partial L}{\partial x_2} = \lambda_0 - \lambda = 0. \tag{b}$$

It is evident from Eq. (b) that $\lambda_0 = \lambda = 1$, or else the non-triviality condition would be violated. Eq. (a) then gives $x_1 = 2$. The complementarity condition $\lambda (1 - x_2) = 0$ gives $x_2 = 1$. The candidate is $(x_1, x_2) = (2, 1)$ giving an objective value of 4.

For fixed x_2, however, the objective is unbounded in x_1. The problem does not have a minimum. ★

Example 3.13.

$$\begin{aligned} \text{minimize} \quad & (x_1 - 2)^2 + (x_2 - 1)^2 \\ \text{subject to} \quad & x_2 - x_1^2 \geq 0 \\ & 2 - x_1 - x_2 \geq 0 \\ & x_1 \geq 0 \end{aligned}$$

The Lagrangian is

$$\begin{aligned} L = {} & \lambda_0 (x_1 - 2)^2 + \lambda_0 (x_2 - 1)^2 \\ & + \lambda_1 \left(x_1^2 - x_2\right) + \lambda_2 (x_1 + x_2 - 2) + \lambda_3 (-x_1) . \end{aligned}$$

Components of the gradient are

$$\frac{\partial L}{\partial x_1} = 2\lambda_0 (x_1 - 2) + 2\lambda_1 x_1 + \lambda_2 - \lambda_3 = 0, \tag{a}$$
$$\frac{\partial L}{\partial x_2} = 2\lambda_0 (x_2 - 1) - \lambda_1 + \lambda_2 = 0. \tag{b}$$

Suppose the first two constraints are active. Then

$$\begin{aligned} x_1^2 = x_2 \quad & \Longrightarrow \quad x_1 = 1, \\ x_1 + x_2 = 2 \quad & \qquad\quad x_2 = 1. \end{aligned}$$

Since $x_1 = 1 > 0$, $\lambda_3 = 0$.

$$\begin{aligned} \text{(a)} \quad & \Longrightarrow \quad 2\lambda_0 (-1) + 2\lambda_1 + \lambda_2 = 0. \\ \text{(b)} \quad & \Longrightarrow \quad \lambda_1 = \lambda_2. \end{aligned}$$

Thus,

$$-2\lambda_0 + 3\lambda_1 = 0.$$

If $\lambda_0 = 0$, then $\lambda_1 = \lambda_2 = \lambda_3 = 0$, which violates the non-triviality condition. Therefore, $\lambda_0 = 1$, $\lambda_1 = \lambda_2 = \frac{2}{3}$, $\lambda_3 = 0$. The candidate is $(x_1, x_2) = (1, 1)$. ★

3.4 Convex Functions and Sets

The challenge of finding candidate solutions and determining which, if any, of those are minima is exacerbated in higher dimensions. The problems described in Chapters 4, 5, 6, and 7 can easily have hundreds or thousands of variables. Numerical methods for general nonlinear optimization problems are sensitive to initial guesses, and providing good initial guesses is yet another challenge. All of these challenges are avoidable when the optimization problem is convex. A convex optimization problem is one with a convex objective function and a convex constraint set.

Convex functions are linear or bowl-shaped. Examples appear in Figure 3.2.

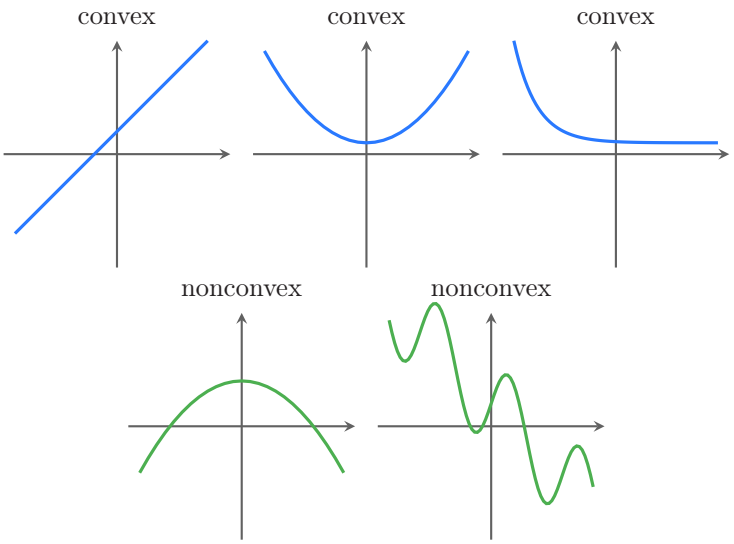


Figure 3.2: Examples of convex and nonconvex functions.

Convex sets have no holes or indentations. Examples appear in Figure 3.3.

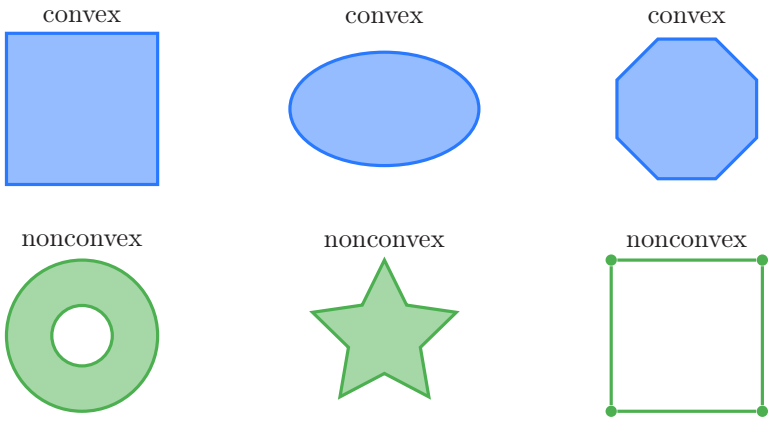


Figure 3.3: Examples of convex and nonconvex sets.

Definition 3.1. A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if for every $x, y \in \mathbb{R}^n$ and $\theta \in [0, 1]$

$$f(\theta x + (1 - \theta)y) \leq \theta f(x) + (1 - \theta)f(y). \tag{3.20}$$

That is to say, the line connecting any two points on the function is not below the function between those two points. This "line above the curve" feature is illustrated in Figure 3.4.

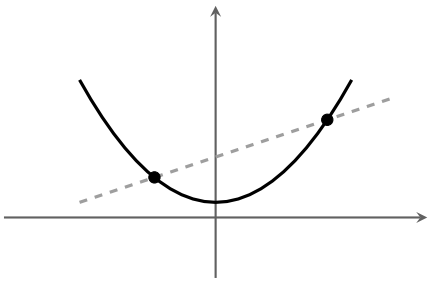


Figure 3.4: Illustration of the definition of a convex function.

Linear functions are convex because linearity requires that for every $x, y \in \mathbb{R}^n$ and every $\alpha, \beta \in \mathbb{R}$

$$f(\alpha x + \beta y) = \alpha f(x) + \beta f(y), \tag{3.21}$$

which is the definition of convexity restricted to equality.

Example 3.14. Show that the function $f(x) = x_1 x_2$ is nonconvex. Take two points in $\mathbb{R}^2$ as

$$x = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \quad \text{and} \quad y = \begin{bmatrix} 2 \\ 1 \end{bmatrix}.$$

Then

$$\theta x + (1 - \theta)\, y = \theta \begin{bmatrix} 1 \\ 2 \end{bmatrix} + (1 - \theta) \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 - \theta \\ 1 + \theta \end{bmatrix}.$$

Evaluating the function at this point gives

$$f(\theta x + (1 - \theta)\, y) = (2 - \theta)\,(1 + \theta) = 2 + \theta - \theta^2.$$

Evaluating the linear approximation gives

$$\theta f(x) + (1 - \theta)\, f(y) = 2\theta + 2\,(1 - \theta) = 2.$$

The required inequality $2 + \theta - \theta^2 \leq 2$ is not satisfied for all $\theta \in [0, 1]$. With $\theta = \frac{1}{2}$

$$2 + \frac{1}{2} - \frac{1}{4} = \frac{9}{4} > 2.$$

Therefore, the function is nonconvex. ★

Example 3.15. Show that $f(x) = x^2$ is convex. To begin, the left and right-hand sides of Eq. (3.20) are computed.

$$\begin{aligned} f(\theta x + (1-\theta)\, y) &= (\theta x + (1-\theta)\, y)^2 \\ &= \theta^2 x^2 + 2\theta\,(1-\theta)\, xy + (1-\theta)^2\, y^2. \\ \theta f(x) + (1-\theta)\, f(y) &= \theta x^2 + (1-\theta)\, y^2. \end{aligned}$$

Applying the inequality in the definition gives the following.

$$\begin{aligned} & \theta x^2 + (1-\theta)\, y^2 - \theta^2 x^2 - 2\theta\,(1-\theta)\, xy - (1-\theta)^2\, y^2 \geq 0 \\ \implies \quad & \theta x^2 + y^2 - \theta y^2 - \theta^2 x^2 - 2\theta\,(1-\theta)\, xy - \left(1 - 2\theta + \theta^2\right) y^2 \geq 0 \\ \implies \quad & \theta\,(1-\theta)\, x^2 - 2\theta\,(1-\theta)\, xy + \theta\,(1-\theta)\, y^2 \geq 0 \\ \implies \quad & \theta\,(1-\theta)\,(x-y)^2 \geq 0 \end{aligned}$$

The colored lines highlight terms that are grouped together. The last inequality is true for every $x, y \in \mathbb{R}$ and every $\theta \in [0, 1]$. Therefore, $f(x) = x^2$ is convex. ★

As experienced in the previous examples, determining convexity using the definition can be cumbersome for even simple functions. Under some smoothness requirements, convexity can be tested based on derivatives.

Theorem 3.4. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be twice continuously differentiable. The function f is convex if and only if for every $x \in \mathbb{R}^n$ the Hessian $\nabla^2 f(x)$ is positive semi-definite.*

Example 3.16. Determine if $f(x) = x^2$ is convex.

$$\begin{aligned} \nabla f &= 2x, \\ \nabla^2 f &= 2 \geq 0 \quad \implies \quad \text{convexity.} \end{aligned}$$

★

Example 3.17. Determine if $f(x) = x^3$ on $[0, \infty)$ is convex.

$$\begin{aligned} \nabla f &= 3x^2, \\ \nabla^2 f &= 6x \leq 0 \quad \text{on} \quad (-\infty, 0] \quad \implies \quad \text{nonconvexity.} \end{aligned}$$

★

Example 3.18. Determine if $f(x) = x_1^2 - x_2^2$ is convex.

$$\nabla f = [2\, x_1,\ -2\, x_2]$$

$$\nabla^2 f = \begin{bmatrix} 2 & 0 \\ 0 & -2 \end{bmatrix}$$

The eigenvalues of the Hessian are ± 2. The Hessian is not positive definite and the function is not convex. ★

Definition 3.2. A set S is convex if the line segment between any two points in S lies in S, i.e., if for every x_1, $x_2 \in S$ and $\theta \in [0, 1]$

$$\theta x_1 + (1 - \theta)\, x_2 \in S. \tag{3.22}$$

Some examples of nonconvex sets and the violating line segment are shown in Figure 3.5.

Figure 3.5: Nonconvex sets and the violating line segments.

Example 3.19. Show that the line segment $S = \{x \in \mathbb{R} : 0 \leq x \leq 1\}$ is convex. Let x_1 and x_2 be any two points in S. Without loss of generality, let $x_1 \leq x_2$. Then, for any $\theta \in [0, 1]$, the point $\theta x_1 + (1 - \theta)\, x_2$ is bounded below by x_1 and above by x_2. These are, in turn, bounded below and above by 0 and 1, respectively. Therefore, $\theta x_1 + (1 - \theta)\, x_2 \in S$ and the set is convex. ★

Example 3.20. Show that the boundary of a square given by $S = \{x \in \mathbb{R}^2 : ||x||_\infty = 1\}$ is nonconvex. Let $x_1 = (1, 0)$, $x_2 = (-1, 0)$, and $\theta = \frac{1}{2}$. Then $\theta x_1 + (1 - \theta)\, x_2 = (0, 0)$, which is not in the set. The set is nonconvex. ★

There are many fascinating properties about convex functions and sets that are beyond the scope of this book.[13]

3.5 Convex Optimization

With knowledge about convex functions, it is time to return to the optimization problem in Eq. (3.12). This problem is convex if the objective function is convex and the constraint set in Eq. (3.13) is convex. Sufficient conditions for the constraint set to be convex are that the inequality functions g be convex and the equality functions h be linear meaning $h(x) = Ax - b$. Because most of the constraint functions arising in optimal spacecraft guidance applications are twice continuously differentiable, testing the functions for convexity using Theorem 3.4 is easier than testing the constraint set for convexity.

Before stating the optimality conditions for convex optimization problems, a definition is given.

Definition 3.3. A point $\hat{x}$ is strictly feasible if $h(\hat{x}) = 0$ and $g(\hat{x}) < 0$.

Any problem that has a feasible point but no strictly feasible point can be changed by an arbitrarily small $\epsilon > 0$ so that it does by rewriting $g(x) - \epsilon \leq 0$. As a result, every feasible problem has or *almost* has a strictly feasible point.

[13] Boyd and Vandenberghe, *Convex Optimization*, 2004.

The Lagrangian for convex optimization problems is

$$L(x, \lambda, \nu) = f(x) + \lambda^\top g(x) + \nu^\top h(x), \tag{3.23}$$

which is the same as before except there is no λ_0 pre-multiplying $f(x)$. In other words, $\lambda_0 = 1$. The necessary and sufficient conditions for convex optimization problems are now stated.

Theorem 3.5 (Optimality Conditions for Convex Optimization)**.** *Let f and g be convex and differentiable and $h(x) = Ax - b$. Suppose a strictly feasible point exists. The problem attains a minimum at x if and only if there exist multipliers $\lambda \in \mathbb{R}^p$ and $\nu \in \mathbb{R}^q$ such that the following equations are satisfied.*

$$g(x) \leq 0 \tag{3.24a}$$
$$Ax = b \tag{3.24b}$$
$$\lambda \geq 0 \tag{3.24c}$$
$$\lambda^\top g(x) = 0 \tag{3.24d}$$
$$\nabla_x L(x, \lambda, \nu) = 0 \tag{3.24e}$$

The equations of the theorem are the same as those in Theorem 3.2 with $\lambda_0 = 1$. For this reason, the non-triviality condition is always satisfied and removed as a condition. Most importantly, Theorem 3.5 is necessary and sufficient. The "if" of Theorem 3.2 has been strengthened to "if and only if". Even so, interesting cases still occur in convex optimization.

🛈 The following convex problems have no solution, one solution, and infinitely many solutions.

$$\begin{array}{rl} \text{minimize} & e^{-x} \\ \text{subject to} & 0 \leq x < \infty \end{array} \qquad \begin{array}{rl} \text{minimize} & x^2 \\ \text{subject to} & -1 \leq x \leq 1 \end{array}$$

$$\begin{array}{rl} \text{minimize} & 0x \\ \text{subject to} & -1 \leq x \leq 1 \end{array}$$

The steps for solving convex optimization problems are the same as those used in previous problems. The first example of convex optimization illustrates the importance of having a strictly feasible point.

Example 3.21.

$$\begin{array}{rl} \text{minimize} & x^2 \\ \text{subject to} & (x-1)^2 \leq 0 \end{array}$$

The objective function and inequality constraint function are convex. Hence, the problem is convex. The constraint set contains only one point, $x = 1$, which is therefore the global minimizer. No strictly feasible point exists, and the optimality conditions should *not* be applied. Ignoring this fact momentarily, the Lagrangian is

$$L = x^2 + \lambda(x-1)^2.$$

The gradient of the Lagrangian is

$$\nabla_x L = 2x + 2\lambda(x - 1) = 0.$$

Substituting $x = 1$ implies that $2 = 0$, an obvious contradiction. All hypotheses of Theorem 3.5 must be checked before using it. ★

Example 3.22.

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}x^\top Px + q^\top x + r, \quad P = P^\top \geq 0 \\ \text{subject to} & Ax = b \end{array}$$

The objective function is convex because of the positive semi-definiteness of P. The equality constraint is linear. Hence, the problem is convex. The requirement for strict feasibility is trivially satisfied because there are no inequality constraints. The Lagrangian is

$$L = \frac{1}{2}x^\top Px + q^\top x + r + \nu^\top (Ax - b).$$

The gradient of the Lagrangian is

$$\nabla_x L = Px + q + A^\top \nu = 0.$$

Therefore, the solution to this problem is obtained by solving the linear system

$$\begin{bmatrix} P & A^\top \\ A & 0 \end{bmatrix} \begin{bmatrix} x \\ \nu \end{bmatrix} = \begin{bmatrix} -q \\ b \end{bmatrix}.$$

Any solution of this matrix equation is a global minimizer. ★

3.6 Solution to the Motivating Problem

We are now prepared to consider again the motivating problem of Section 3.1. The problem of driving the system from its initial state x_0 to a final state x_N in N steps was reduced to the linear algebra problem

$$X = CU, \tag{3.25}$$

which has infinitely many solutions when it is feasible and the null space of C is nontrivial. Consider the following quadratic objective given by

$$\frac{1}{2}\sum_k \|u_k\|_2^2 = \frac{1}{2}\sum_k u_k^\top u_k = \frac{1}{2}U^\top U = \frac{1}{2}\|U\|_2^2. \tag{3.26}$$

The resulting constrained optimization problem is the following.

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}U^\top U \\ \text{subject to} & X = CU \end{array} \tag{3.27}$$

The optimization variable here is U, the objective function is convex, the equality constraint is linear, and the requirement for strict feasibility is trivially satisfied because there are no inequality constraints. The Lagrangian is

$$L = \frac{1}{2}U^\top U + \nu^\top (CU - X). \tag{3.28}$$

The gradient of the Lagrangian is

$$\nabla_u L = U + C^\top \nu = 0. \tag{3.29}$$

Assuming that C is full rank, one can solve for ν by pre-multiplying with C

$$CU + CC^\top \nu = 0 \quad \Longrightarrow \quad \nu = -\left(CC^\top\right)^{-1} CU, \tag{3.30a}$$

$$\Longrightarrow \quad \nu = -\left(CC^\top\right)^{-1} X. \tag{3.30b}$$

from the equality constraint

Substituting this back into the gradient condition gives the candidate solution

$$U = C^\top (CC^\top)^{-1} X. \tag{3.31}$$

This is the global minimizer. By definition of optimality, no other feasible control can have lower objective while driving the state of the system from x_0 to x_N in N steps. The controls at each discrete time can be extracted from U and substituted into the discrete-time system to generate the associated states. The sequence of controls u_k is the optimal control trajectory. The sequence of states x_k is the optimal state trajectory.

MATLAB Implementations including Direct Transcription

There are multiple ways of solving the problem in MATLAB for a specific data instance. These ways include typing in the analytic solution, using `pinv`, using `quadprog`, and using YALMIP to parse the problem. The use of YALMIP is inordinate for this small example, but as will be demonstrated, doing so allows one to easily solve more challenging problems.

To demonstrate the numerical implementations, the CW equations 2.31 are used. For a low earth orbit, the mean motion is set to 4 rad/hour and converted to rad/sec. The continuous-time system matrices are as follows.

```
% Continuous-time CW System
w = 4 / 3600;
A = [0, 0, 0, 1, 0, 0;
     0, 0, 0, 0, 1, 0;
     0, 0, 0, 0, 0, 1;
     3*w^2, 0, 0, 0, 2*w, 0;
     0, 0, 0, -2*w, 0, 0;
     0, 0, -w^2, 0, 0, 0];
B = [zeros(3,3); eye(3,3)];
```

Conversion to a discrete-time system is done using MATLAB's built-in `c2d` command with a time step `dt` of one second. An alternative approach is to use the state transition matrix for the CW system to compute the discrete-time matrices.

```
10  % Discretization
11  dt = 1;
12  sysc = ss(A,B,[],[]);
13  sysd = c2d(sysc,dt);
14  A = sysd.A;
15  B = sysd.B;
```

The analytical approach outlined above is now implemented. The initial and final conditions are specified in `x0` and `xF`, respectively. The number of steps is `N`. The optimal solution is `U`.

```
16  % Analytical Implementation
17  x0 = [0;-1;0;-.1;.1;0];
18  xF = [0;0;0;0;0;0];
19  N  = 5*60;
20  X  = xF - A^N*x0;
21  C  = [];
22  for k = 0:N-1
23      C = [C, A^(N-1-i)*B];
24  end
25  U = C.'*inv(C*C.')*X;
```

To simulate the system, the tall `U` is reshaped into `u` where each column of `u` represents a control. The discrete-time system is simulated using a `for` loop.

```
26  % Simulation
27  u = reshape(U,3,N);
28  x(:,1) = x0;
29  for k = 1:N
30      x(:,k+1) = A*x(:,k) + B*u(:,k);
31  end
32
33  figure, plot(x(2,:),x(1,:)), grid on
```

Running the code at this point solves the optimization using the analytical solution, simulates the discrete-time system, and generates a plot of the vertical and horizontal positions. The solution in line 25 can also be calculated using the pseudoinverse.

```
25  U = pinv(C)*X;
```

Yet another approach is to solve a quadratic program.

```
25  U = quadprog(eye(3*N),zeros(3*N,1),[],[],C,X);
```

A different approach is to use YALMIP to parse the problem and Gurobi to solve the problem. In Chapter 4, such an approach is called direct transcription. If the Gurobi solver is not available, the SDPT3 solver also works on this problem. To get started with YALMIP, some options and optimization variables are defined. The command `sdpsettings` requires YALMIP to be installed.[14]

[14] Löfberg, *YALMIP : A Toolbox for Modeling and Optimization in MATLAB*, 2004.

```
33  % YALMIP
34  opts        = sdpsettings;
35  opts.solver = 'gurobi';
36
37  yalmip('clear')
38  x    = sdpvar(6,N+1);
39  u    = sdpvar(3,N);
```

Note that in code, the time index starts at 1 rather than 0. The constraints of the problem are accumulated in `con` and the objective is `obj`. In direct transcription, the problem is input in its discrete-time form. Never are the C and X matrices used.

```
40  con = [x(:,1) == x0];
41  obj = 0;
42  for k = 1:N
43      con = [con, x(:,k+1) == A*x(:,k) + B*u(:,k)];
44      obj = obj + u(:,k).'*u(:,k);
45  end
46  con = [con, x(:,N+1) == xF];
47  % placeholder
48  % placeholder
49  sol = solvesdp(con,obj,opts);
```

Running the YALMIP portion of the code generates optimal controls and states consistent with those calculated from the analytical solution.

A key benefit of this approach is that it is now easy to impose other constraints on the problem. For example, if it is desired to restrict the state trajectory to ± 3.5 km, constraints can be added to lines 47 and 48.

```
47  con = [con, -3.5 <= x(1,:) <= 3.5];
48  con = [con, -3.5 <= x(2,:) <= 3.5];
```

Running the code now generates an optimal control that transfers the state from its initial condition to the desired final condition in N steps and keeps the state in its 3.5 km box. The state trajectories for the unconstrained and constrained cases are shown in Figure 3.6.

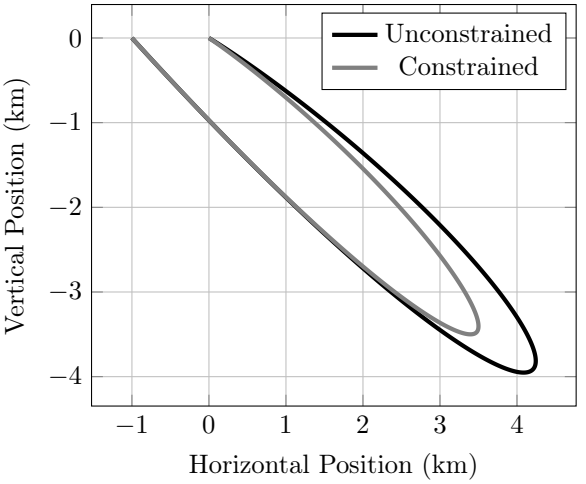


Figure 3.6: Unconstrained and constrained state trajectories.

The remaining chapters investigate discrete-time and continuous-time optimal control problems. While the theory for continuous-time problems is significantly more advanced, a sound numerical approach for solving them is often the same: discretize, parse using YALMIP, and solve using Gurobi. A MATLAB implementation of this procedure for an orbit transfer problem begins on page 77.

3.7 Chapter Problems

Problem 3.1. Solve the following problem analytically and numerically using `fmincon`.

$$\begin{aligned} \text{minimize} \quad & x_1^2 + x_2^2 \\ \text{subject to} \quad & x_2 + x_1 - 2 \leq 0 \\ & x_1^2 - x_2 - 4 \leq 0 \end{aligned}$$

Problem 3.2. Solve the following problem analytically and numerically using `fmincon`.

$$\begin{aligned} \text{minimize} \quad & x_1^2 + x_2^2 \\ \text{subject to} \quad & x_1 - 10 \leq 0 \\ & x_1 - x_2^2 - 4 \geq 0 \end{aligned}$$

Problem 3.3. Solve the following problem analytically and numerically using `fmincon`.

$$\begin{aligned} \text{minimize} \quad & x_1^2 + x_2^2 \\ \text{subject to} \quad & 4 - x_1 - x_2^2 \leq 0 \\ & 3x_2 - x_1 \leq 0 \\ & -3x_2 - x_1 \leq 0 \end{aligned}$$

Problem 3.4. Solve the following problem analytically and numerically using `fmincon`.

$$\begin{aligned} \text{minimize} \quad & x_1 x_2 \\ \text{subj. to} \quad & x_1 + x_2 \geq 2 \\ & x_2 \geq x_1 \end{aligned}$$

Problem 3.5. Solve the following problem analytically and numerically using `fmincon`.

$$\begin{aligned} \text{minimize} \quad & -x_1 \\ \text{subject to} \quad & -x_1 \leq 0 \\ & -x_2 \leq 0 \\ & x_2 + (x_1 - 1)^3 \leq 0 \end{aligned}$$

Problem 3.6. Solve the following problem analytically and numerically using `fmincon`.

$$\begin{aligned} \text{minimize} \quad & 2x_1^2 - x_2^2 \\ \text{subject to} \quad & x_1^2 x_2 - x_2^3 = 0 \end{aligned}$$

Problem 3.7. Why is convexity important in optimization?

Problem 3.8. Prove that the following set is convex.

$$S = \{u \in \mathbb{R}^m : \|u\|_2 \leq 2\}$$

Problem 3.9. Prove that the following set is nonconvex.

$$S = \{u \in \mathbb{R}^m : 1 \leq \|u\|_2 \leq 2\}$$

Problem 3.10. Consider a lunar lander in close proximity to the lunar surface such that the flat planet model in Eq. (2.10) is valid. The initial downrange, crossrange, altitude, and associated rates are given below in meters and meters per second.

```
r0 = [1000; 50; 1000]; % m
v0 = [-25; 0; 0]; % m/s
```

The goal is to land at the origin of the coordinate system with downward velocity of 1 m/s. The flight time is two minutes. Design a nominal continuous-time descent trajectory using polynomial guidance as described in Section 2.2.

Problem 3.11. Building upon the previous problem, design a nominal discrete-time descent trajectory using direct transcription. An implementation of direct transcription for the motivating problem begins on page 46. Recognize that the dynamics in that implementation are *not* based on the flat planet model. Discretize the system using a time step of one second. Use the same objective function given in code line 44.

Chapter 4

Discrete Optimal Control

Chapter Learning Objectives

1. Derive optimality conditions for discrete optimal control problems.
2. Develop and implement optimal control, regulation, and tracking algorithms.
3. Apply these algorithms to relative orbital motion and descent problems.
4. Develop and implement Chebyshev discretization of nonlinear systems.
5. Solve an optimal orbit transfer problem using direct transcription.

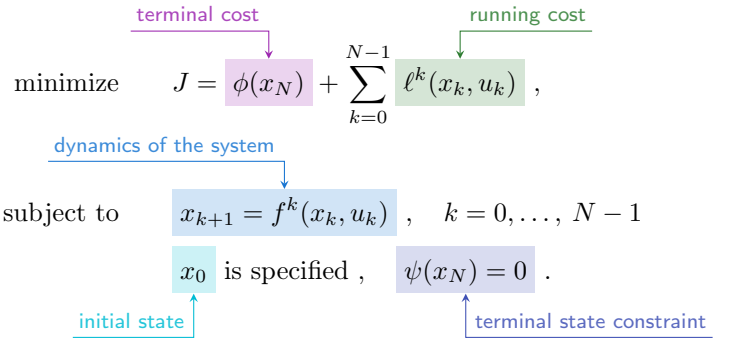
The motivating problem of Chapter 3 was to drive a discrete-time system from an initial condition to a final condition in N steps. By recursively writing out the state equation, the problem was "reduced" to one with a quadratic objective function and linear constraint. The problem was solved numerically several ways at the end of the chapter. Three of those ways used the "reduced" form. The final way, using YALMIP and Gurobi, left the problem in its original form with discrete-time dynamics. The original form of the problem is called a discrete optimal control problem.

The first objective is to derive optimality conditions that apply directly to problems written in a form with discrete-time dynamics. One reason for doing so is that the presence of nonlinear dynamics severely complicates the "reduction" process. Another is that the natural structure of discrete-time dynamics imposes a dual structure on the multipliers; they now also evolve according to discrete-time dynamics. Lastly, future chapters consider continuous-time optimal control problems in which the dynamics are described by ordinary differential equations. The resulting optimization conditions mirror their discrete-time counterparts in this chapter.

A linear quadratic framework is explored extensively. Algorithms for control, regulation, and tracking are applied to the relative orbital motion model to achieve rendezvous and circumnavigation. The regulation idea is applied to follow a nominal descent trajectory based on polynomial guidance. The chapter concludes by investigating nonlinear systems, presenting a discretization technique using Chebyshev polynomials, and implementing direct transcription to solve an optimal orbit transfer problem.

4.1 A General Discrete Optimal Control Problem

A general form for a discrete optimal control problem is given below. Like all previous optimization problems, it includes an objective function that is scalar-valued and constraint functions.



(4.1)

The terminal cost function $\phi(x_N)$ is a penalty on the states only at the final time. For example, if it is desired to drive the system close to the origin, one may choose

$$\phi(x_N) = x_N^\top x_N. \tag{4.2}$$

The running cost function $\ell^k(x_k, u_k)$ penalizes states and controls all along the trajectory (except at the final time). For example, if it is desired to keep the states and controls close to zero, one may choose

$$\ell^k(x_k, u_k) = \frac{1}{2} x_k^\top x_k + \frac{1}{2} u_k^\top u_k. \tag{4.3}$$

The dynamics of the system are $x_{k+1} = f^k(x_k, u_k)$. The initial condition of the system is fixed at x_0, and the final state of the system x_N is not fixed, but constrained by $\psi(x_N) = 0$. At present, the controls are unconstrained, i.e., $u_k \in \mathbb{R}^m$.

Though it may look more complicated than previous optimization problems, it is not. What is different is that the objective and constraint functions have specific forms. The optimality conditions of either Theorem 3.2 or Theorem 3.5 can be specialized to arrive at conditions specific to this problem. The multiplier λ was previously associated with inequality constraints and ν with equality constraints. The standard usage in optimal control is to associate λ with the dynamics and ν with the terminal constraint.

> ℹ To not confuse the abnormal multiplier (previously λ_0) with a value of λ at time index 0, the abnormal multiplier is now denoted λ^0.

Consistent with its definition for the nonlinear optimization problem in Eq. 3.12, the Lagrangian is

$$L = \lambda^0 \phi(x_N) + \lambda^0 \sum_{k=0}^{N-1} \ell^k(x_k, u_k) + \sum_{k=0}^{N-1} \lambda_{k+1}^\top \Big(f^k(x_k, u_k) - x_{k+1} \Big) + \nu^\top \psi(x_N). \tag{4.4}$$

see that there are N λ's: $\lambda_1, \ldots, \lambda_N$

Before moving on to computing components of the gradient, it is convenient to define the Hamiltonian function

$$H^k\big(x_k, u_k, \lambda^0, \lambda_{k+1}\big) = \lambda^0 \ell^k(x_k, u_k) + \lambda_{k+1}^\top f^k(x_k, u_k), \tag{4.5}$$

for $k = 0, \ldots, N-1$. The Lagrangian may then be rewritten in terms of the Hamiltonian as

$$L = \lambda^0 \phi(x_N) + \nu^\top \psi(x_N) + \sum_{k=0}^{N-1} H^k\big(x_k, u_k, \lambda^0, \lambda_{k+1}\big) - \sum_{k=1}^{N} \lambda_k^\top x_k \tag{4.6a}$$

$$= \lambda^0 \phi(x_N) + \nu^\top \psi(x_N) + H^0\big(x_0, u_0, \lambda^0, \lambda_1\big) - \lambda_N^\top x_N + \sum_{k=1}^{N-1} \Big(H^k\big(x_k, u_k, \lambda^0, \lambda_{k+1}\big) - \lambda_k^\top x_k \Big). \tag{4.6b}$$

Computing components of the gradient with respect to all the u's and all the x's (except x_0 because it is fixed) and setting them equal to zero gives rise to the specialized form of optimality conditions.

$$\frac{\partial L}{\partial x_N} = \lambda^0 \frac{\partial \phi}{\partial x_N} + \frac{\partial \psi}{\partial x_N} \nu - \lambda_N = 0 \qquad \text{(transversality condition)} \tag{4.7}$$

$$\frac{\partial L}{\partial u_k} = \frac{\partial H^k}{\partial u_k} = 0 \qquad \text{(stationarity condition)}_{k=0,\ldots,N-1} \tag{4.8}$$

$$\frac{\partial L}{\partial x_k} = \frac{\partial H^k}{\partial x_k} - \lambda_k = 0 \qquad \text{(costate equation)}_{k=1,\ldots,N-1} \tag{4.9}$$

$$x_{k+1} = \frac{\partial H^k}{\partial \lambda_{k+1}} = f^k\left(x_k, u_k\right) \qquad \text{(state equation)}_{k=0,\ldots,N-1} \tag{4.10}$$

The state equation is not one generated from the gradient condition. By writing it, however, it is seen that the state equation and costate equation share a similar Hamiltonian form. For cases where the final state is fixed, computing the gradient with respect to x_N is not required, and the transversality condition is removed from the set of optimality conditions. Lastly, the non-triviality condition stating that not all multipliers be zero remains one of the optimality conditions and is useful for sometimes ruling out the abnormal case.

When the problem is convex and differentiable with a strictly feasible point, $\lambda^0 = 1$ and the optimality conditions are necessary and sufficient as in Theorem 3.5. Because the discrete-time dynamics and terminal state constraint are equality constraints, convex problems have linear dynamics and linear terminal constraint.

Connecting Optimization and Discrete Optimal Control

To make the connection between optimization and discrete optimal control clearer, consider the following one-dimensional problem. The goal is to move an object from its starting position $x_0 = 2$ to a final position $x_2 = 0$ in two steps. This means that at time 0 the object can be moved and at time 1 the object can be moved. After this second move, the object should be at zero as illustrated in Figure 4.1.

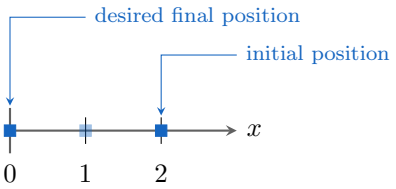


Figure 4.1: Illustration of the one-dimensional discrete optimal control problem.

The amounts to be moved are denoted by u_0 and u_1. Thus, the object moves according to

$$x_1 = x_0 + u_0, \tag{4.11}$$
$$x_2 = x_1 + u_1. \tag{4.12}$$

Combining the equations gives

$$x_2 = x_0 + u_0 + u_1. \tag{4.13}$$

Since x_2 and x_0 are known, they are grouped together

$$x_2 - x_0 = -2 = u_0 + u_1. \tag{4.14}$$

Any movements u_0, u_1 that add to -2 are feasible movements. Of all the feasible movements, the objective is to minimize the quadratic function

$$J = \frac{1}{2}\left(u_0^2 + u_1^2\right). \tag{4.15}$$

The problem is solved in three ways. Though the problem is convex, the abnormal multiplier is kept for the sake of practice.

In the first approach, the x variables are eliminated and Theorem 3.2 is used.

$$\begin{array}{ll} \underset{u_0,u_1}{\text{minimize}} & \dfrac{1}{2}\left(u_0^2 + u_1^2\right) \\ \text{subject to} & u_0 + u_1 + 2 = 0 \end{array} \tag{4.16}$$

The Lagrangian is

$$L = \frac{\lambda_0}{2}\left(u_0^2 + u_1^2\right) + \nu\left(u_0 + u_1 + 2\right). \tag{4.17}$$

Components of the gradient are

$$\frac{\partial L}{\partial u_0} = \lambda_0\, u_0 + \nu = 0, \tag{4.18}$$
$$\frac{\partial L}{\partial u_1} = \lambda_0\, u_1 + \nu = 0. \tag{4.19}$$

If $\lambda_0 = 0$, then $\nu = 0$ violating non-triviality. Thus, $\lambda_0 = 1$ and $u_0 = u_1 = -\nu$. Since $u_0 + u_1 = -2$ and $u_0 = u_1$, it is concluded that $u_0 = u_1 = -1$.

In the second approach, the x_1 variable is kept and Theorem 3.2 is used.

$$\begin{array}{ll} \underset{u_0,u_1,x_1}{\text{minimize}} & \frac{1}{2}\left(u_0^2+u_1^2\right) \\ \text{subject to} & x_1 = 2+u_0 \\ & 0 = x_1+u_1 \end{array} \tag{4.20}$$

The Lagrangian is

$$L = \frac{\lambda_0}{2}\left(u_0^2+u_1^2\right) + \lambda_1\left(2+u_0-x_1\right) + \lambda_2\left(x_1+u_1-0\right). \tag{4.21}$$

Components of the gradient are

$$\frac{\partial L}{\partial u_0} = \lambda_0\, u_0 + \lambda_1 = 0, \tag{4.22}$$

$$\frac{\partial L}{\partial u_1} = \lambda_0\, u_1 + \lambda_2 = 0, \tag{4.23}$$

$$\frac{\partial L}{\partial x_1} = -\lambda_1 + \lambda_2 = 0. \tag{4.24}$$

If $\lambda_0 = 0$, then $\lambda_1 = \lambda_2 = 0$ violating non-triviality. Thus, $\lambda_0 = 1$. It is again seen that $u_0 = u_1$ and $u_0 + u_1 = -2$. Thus, $u_0 = u_1 = -1$.

In the final approach, conditions from discrete optimal control are used. Write the Hamiltonian

$$H^k = \frac{1}{2}\lambda^0\, u_k^2 + \lambda_{k+1}\, f^k(x_k,\, u_k), \tag{4.25a}$$

$$\Downarrow$$

$$H^0 = \frac{1}{2}\lambda^0\, u_0^2 + \lambda_1\left(x_0+u_0\right), \tag{4.25b}$$

$$H^1 = \frac{1}{2}\lambda^0\, u_1^2 + \lambda_2\left(x_1+u_1\right). \tag{4.25c}$$

The stationarity conditions are

$$\frac{\partial H^0}{\partial u_0} = \lambda^0 u_0 + \lambda_1 = 0, \tag{4.26}$$

$$\frac{\partial H^1}{\partial u_1} = \lambda^0 u_1 + \lambda_2 = 0. \tag{4.27}$$

The costate equation is

$$\lambda_1 = \frac{\partial H^1}{\partial x_1} = \lambda_2. \tag{4.28}$$

If $\lambda^0 = 0$, then $\lambda_1 = \lambda_2 = 0$ violating non-triviality. Thus, $\lambda^0 = 1$. The optimality conditions again require $u_0 = u_1$ and $u_0 + u_1 = -2$. Thus, $u_0 = u_1 = -1$. The three formulations give the same answer. This is expected because the three approaches are equivalent ways of enforcing the optimality conditions in Theorems 3.2 and 3.5

4.2 Scalar Linear Quadratic Control

The linear quadratic framework is now introduced. The problem is to minimize a quadratic control objective while driving a scalar linear system from its given initial condition to given final condition.

$$\begin{aligned} &\text{minimize} && J = \frac{1}{2}\sum_{k=0}^{N-1} u_k^2 \\ &\text{subject to} && x_{k+1} = ax_k + bu_k, \quad x_0, x_N \text{ given} \end{aligned} \tag{4.29}$$

It is assumed that $b \neq 0$ so that the control affects the state. As in the motivating example from Section 3.1, the goal is to find the control sequence u_0, u_1, ..., u_{N-1} to drive the state from x_0 to x_N in N steps. Unlike regular optimization problems for which the first step is to write the Lagrangian, the first step is to write the Hamiltonian.

$$H^k = \frac{1}{2}\lambda^0 u_k^2 + \lambda_{k+1}\left(ax_k + bu_k\right) \qquad \text{(Hamiltonian)} \tag{4.30}$$

From the Hamiltonian, the costate equation and stationarity condition may be formed.

$$\lambda_k = a\lambda_{k+1} \qquad \text{(costate equation)} \tag{4.31}$$

$$0 = \lambda^0 u_k + b\lambda_{k+1} \qquad \text{(stationarity condition)} \tag{4.32}$$

Suppose λ^0 is zero. The stationarity condition implies that all λ's are zero. Because the final state is fixed, the transversality condition and its multiplier ν can be done away with. Hence, all the multipliers are zero violating the non-triviality condition.

> If you want to write the fixed final point using the constraint $\psi(x) = 0$, the multiplier ν remains. However, the gradient of ψ is full rank and the transversality condition implies ν is zero, which again leads to a violation of the non-triviality condition.

Yet another realization is that the problem is convex, lacks inequality constraints, and trivially satisfies the requirement for a strictly feasible point. Hence, the optimality conditions for convex optimization should be used. In any case, setting $\lambda^0 = 1$ and solving for u_k in the stationarity condition gives

$$u_k = -b\lambda_{k+1}. \tag{4.33}$$

Once λ_{k+1} is known, the optimal control is known. To find λ_{k+1}, eliminate u_k in the state equation resulting in

$$x_{k+1} = ax_k - b^2\lambda_{k+1}. \tag{4.34}$$

The costate equation is a simple recursion with the solution

$$\lambda_k = a^{N-k}\lambda_N, \tag{4.35}$$

such that

$$x_{k+1} = ax_k - b^2 a^{N-k-1}\lambda_N. \tag{4.36}$$

The problem has now been reduced to finding λ_N. Written in terms of the initial state x_0, the solution to the above equation is

$$x_k = a^k x_0 - b^2 a^{N+k-2} \lambda_N \sum_{j=0}^{k-1} a^{-2j}. \tag{4.37}$$

The summation is a geometric series whose sum can be written explicitly. Another approach will be taken later.

$$x_k = a^k x_0 - b^2 a^{N+k-2} \lambda_N \frac{\left(1 - a^{-2k}\right)}{\left(1 - a^{-2}\right)} \tag{4.38a}$$

$$= a^k x_0 - b^2 a^{N-k} \lambda_N \frac{\left(1 - a^{2k}\right)}{\left(1 - a^2\right)} \tag{4.38b}$$

Evaluating at the final time yields

$$x_N = a^N x_0 - b^2 \lambda_N \frac{\left(1 - a^{2N}\right)}{\left(1 - a^2\right)} \tag{4.39a}$$

$$= a^N x_0 - \Lambda \lambda_N \tag{4.39b}$$

where

$$\Lambda = b^2 \frac{\left(1 - a^{2N}\right)}{\left(1 - a^2\right)}. \tag{4.40}$$

Provided $\Lambda \neq 0$, solving for λ_N gives

$$\lambda_N = \frac{1}{\Lambda} \left(a^N x_0 - x_N\right). \tag{4.41}$$

Substituting this back into the costate equation gives

$$\lambda_k = \frac{1}{\Lambda} \left(a^N x_0 - x_N\right) a^{N-k}. \tag{4.42}$$

At last, the optimal control is

$$u_k = -b\lambda_{k+1} = -\frac{b}{\Lambda} \left(a^N x_0 - x_N\right) a^{N-k-1}. \tag{4.43}$$

This control drives the (a, b) system from x_0 to x_N in N steps and minimizes the quadratic control objective.

To obtain the above expression, a formula for a geometric series was used. This was not required. Starting back at

$$x_k = a^k x_0 - b^2 a^{N+k-2} \lambda_N \sum_{j=0}^{k-1} a^{-2j} \tag{4.44}$$

and replacing k with N, then

$$x_N = a^N x_0 - b^2 a^{2N-2} \lambda_N \sum_{j=0}^{N-1} a^{-2j}. \tag{4.45}$$

By redefining

$$\Lambda = b^2 a^{2N-2} \sum_{j=0}^{N-1} a^{-2j}, \tag{4.46}$$

the final state can be written as

$$x_N = a^N x_0 - \Lambda \lambda_N. \tag{4.47}$$

Solving for λ_N gives

$$\lambda_N = \frac{1}{\Lambda} \left(a^N x_0 - x_N \right). \tag{4.48}$$

And from here, the analysis is the same. The geometric formula was used only because it provides a clean formula for Λ that is easier to implement. Note that when $|a| > 1$, the numerical values explode quickly leading to numerical issues.

MATLAB Implementation

For an implementation in MATLAB, define the system `a` and `b`, the boundary conditions `x0` and `xN`, and the number of steps `N`. The problem is to drive the state from 1 to 0 in 10 steps.

```
% Data
a  = 1.1;
b  = 1;
x0 = 1;
xN = 0;
N  = 10;
```

Compute Λ and the optimal control at each time index. Indices in MATLAB start at one rather than zero.

```
% Optimal Control
L = b^2*( 1-a^(2*N) ) / (1-a^2);
for k = 0:N-1
    u(k+1) = -b/L * (a^N*x0-xN)*a^(N-k-1);
end
```

With the optimal control known, the system can be simulated and the result plotted.

```
% Simulation
x(1) = x0;
for k = 1:N
    x(k+1) = a*x(k) + b*u(k);
end
figure, plot(0:N,x), grid on
xlabel('k'), ylabel('x')
```

Figure 4.2 shows the state evolving in time from 1 to 0 as required.

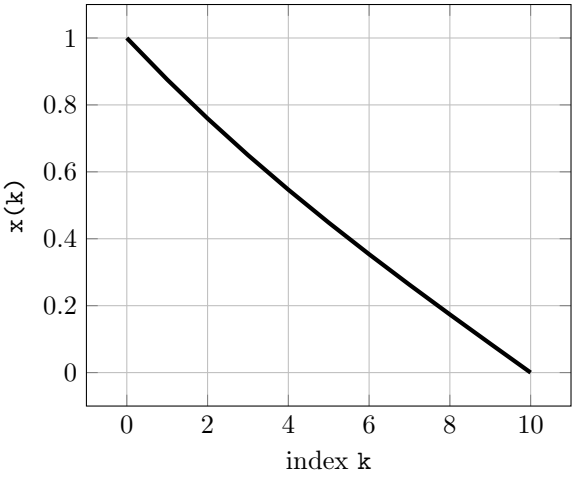


Figure 4.2: State trajectory for scalar discrete optimal control.

4.3 Linear Quadratic Control

The matrix-vector version of the problem in Section 4.2 is called the linear quadratic control (LQC) problem. The state is $x_k \in \mathbb{R}^n$ and the control is $u_k \in \mathbb{R}^m$.

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\sum_{k=0}^{N-1} u_k^\top R u_k, \qquad R > 0 \\ \text{subject to} & x_{k+1} = Ax_k + Bu_k, \quad x_0, x_N \text{ given} \end{array} \tag{4.49}$$

Because R is positive definite and the dynamics are linear, the problem is convex. Without inequality constraints, the requirement for a strictly feasible point is satisfied and $\lambda^0 = 1$. Begin by writing the Hamiltonian and optimality conditions.

$$H^k = \frac{1}{2}u_k^\top R\, u_k + \lambda_{k+1}^\top (Ax_k + Bu_k) \qquad \text{(Hamiltonian)} \tag{4.50}$$

$$\lambda_k = A^\top \lambda_{k+1} \qquad \text{(costate equation)} \tag{4.51}$$

$$0 = Ru_k + B^\top \lambda_{k+1} \qquad \text{(stationarity condition)} \tag{4.52}$$

Solving for the control in the stationarity condition gives

$$u_k = -R^{-1}B^\top \lambda_{k+1}. \tag{4.53}$$

Substitute this into the state dynamics to get

$$x_{k+1} = Ax_k - BR^{-1}B^\top \lambda_{k+1}. \tag{4.54}$$

The costate equation can be rewritten in terms of λ_N as

$$\lambda_k = A^{\top(N-k)}\lambda_N, \tag{4.55}$$

such that the state equation can be written in terms of λ_N also.

$$x_{k+1} = Ax_k - BR^{-1}B^\top A^{\top(N-k-1)}\lambda_N \tag{4.56}$$

Writing out a few terms in the sequence gives

$$x_1 = Ax_0 - BR^{-1}B^\top A^{\top(N-1)}\lambda_N, \tag{4.57a}$$

$$x_2 = Ax_1 - BR^{-1}B^\top A^{\top(N-2)}\lambda_N \tag{4.57b}$$

$$= A^2x_0 - ABR^{-1}B^\top A^{\top(N-1)}\lambda_N - B\,R^{-1}B^\top A^{\top(N-2)}\lambda_N. \tag{4.57c}$$

From here, the following form is deduced.

$$x_k = A^kx_0 - \sum_{i=0}^{k-1} A^{k-i-1}BR^{-1}B^\top A^{\top(N-i-1)}\lambda_N \tag{4.58}$$

Set $k = N$ to find an expression for x_N.

$$x_N = A^Nx_0 - \sum_{i=0}^{N-1} A^{N-i-1}BR^{-1}B^\top A^{\top(N-i-1)}\lambda_N \tag{4.59}$$

Defining the summation to be Λ gives

$$x_N = A^Nx_0 - \Lambda\lambda_N. \tag{4.60}$$

Solving for λ_N (provided Λ^{-1} exists) gives

$$\lambda_N = \Lambda^{-1}\left(A^Nx_0 - x_N\right). \tag{4.61}$$

Thus,

$$\lambda_k = A^{\top(N-k)}\Lambda^{-1}\left(A^Nx_0 - x_N\right), \tag{4.62}$$

and the optimal control is

$$u_k = -R^{-1}B^\top\lambda_{k+1} \tag{4.63a}$$

$$= -R^{-1}B^\top A^{\top(N-k-1)}\Lambda^{-1}\left(A^Nx_0 - x_N\right). \tag{4.63b}$$

The above solution requires that Λ^{-1} exists. The meaning of this requirement is clear after writing Λ in matrix form.

$$\Lambda = A^{N-1}BR^{-1}B^\top A^{\top(N-1)} + A^{N-2}BR^{-1}B^\top A^{\top(N-2)} + \cdots + A^0BR^{-1}B^\top A^{\top(0)}, \tag{4.64a}$$

$$= \left[B,\, AB,\, \cdots,\, A^{N-1}B\right]\begin{bmatrix} R^{-1} & & 0 \\ & \ddots & \\ 0 & & R^{-1} \end{bmatrix}\left[B,\, AB,\, \cdots,\, A^{N-1}B\right]^\top. \tag{4.64b}$$

The matrix

$$C = \begin{bmatrix} B, & AB, & \cdots, & A^{N-1}B \end{bmatrix} \tag{4.65}$$

is called the controllability matrix. The matrix Λ is invertible when the controllability matrix has rank n. A linear time-invariant system is controllable if and only if the controllability matrix has rank n. Thus, the (A, B) system can be driven from x_0 to x_N in N steps with the above optimal control provided the system is controllable. Most engineered systems are designed to be controllable.

ℹ What is a weaker condition than having Λ^{-1} exist? Recall conditions for a linear system being solvable in terms of range space.

A variation of the problem removes the fixed final point x_N and instead adds it to the objective. The optimal control now seeks some balance between minimizing control usage and minimizing the magnitude of the final state.

$$\begin{aligned} &\text{minimize} && \frac{1}{2}x_N^\top S_N x_N + \frac{1}{2}\sum_{k=0}^{N-1} u_k^\top R u_k, \qquad R > 0, \quad S \geq 0 \\ &\text{subject to} && x_{k+1} = Ax_k + Bu_k, \quad x_0 \text{ given} \end{aligned} \tag{4.66}$$

The problem remains convex. The Hamiltonian and optimality conditions are the following.

$$H^k = \frac{1}{2}u_k^\top R u_k + \lambda_{k+1}^\top (Ax_k + Bu_k) \quad \text{(Hamiltonian)} \tag{4.67}$$

$$\lambda_k = A^\top \lambda_{k+1} \quad \text{(costate equation)} \tag{4.68}$$

$$\lambda_N = S_N x_N \quad \text{(transversality condition)} \tag{4.69}$$

$$0 = Ru_k + B^\top \lambda_{k+1} \quad \text{(stationarity condition)} \tag{4.70}$$

Solving for the control in the stationarity condition gives

$$u_k = -R^{-1}B^\top \lambda_{k+1}. \tag{4.71}$$

Substituting this into the state equation gives

$$x_{k+1} = Ax_k - BR^{-1}B^\top \lambda_{k+1}. \tag{4.72}$$

Following the same logic as before, we arrive at

$$x_N = A^N x_0 - \Lambda \lambda_N. \tag{4.73}$$

The final point x_N is unknown in this problem. Impose the transversality condition in Eq. (4.69) to get

$$x_N = A^N x_0 - \Lambda S_N x_N \quad \implies \quad (I + \Lambda S_N)\, x_N = A^N x_0. \tag{4.74}$$

Provided the inverse exists, one can solve for the final state.

$$x_N = (I + \Lambda S_N)^{-1} A^N x_0 \tag{4.75}$$

With the final state known, the optimal control can be written in terms of the problem data.

$$u_k = -R^{-1}B^\top A^{\top(N-k-1)} \Lambda^{-1} \left(I - (I + \Lambda S_N)^{-1}\right) A^N x_0 \tag{4.76}$$

When $S_N = 0$, the final state becomes unimportant and $x_N = A^N x_0$, which arises from uncontrolled motion. The system simply drifts for N steps.

Control laws that depend on x_0 and x_N are called open-loop; they lack independence on the current state of the system. Control laws that depend on the current state of the system are called closed-loop control laws. Both have their place in guidance. Open-loop solutions are commonly used as nominal trajectories. Closed-loop systems are commonly used for tracking nominal trajectories.

MATLAB Implementation of Relative Orbit Control

To demonstrate linear quadratic control, the linearized CW equations (2.31) are used. The problem is to perform a rendezvous, i.e., drive a given initial state to the origin in N steps. The mean motion is 4 rad/hr and the time step for discretization is one second. Units used in MATLAB are meters and seconds. In code, the first step is to write the continuous-time system and discretize it. MATLAB's `c2d` command is used for discretization.

```
1  % Continuous-time CW matrices
2  w = 4 / 3600;
3  A = [0, 0, 0, 1, 0, 0;
4      0, 0, 0, 0, 1, 0;
5      0, 0, 0, 0, 0, 1;
6      3*w^2, 0, 0, 0, 2*w, 0;
7      0, 0, 0, -2*w, 0, 0;
8      0, 0, -w^2, 0, 0, 0];
9  B = [zeros(3,3); eye(3,3)];
10
11 % Discretization
12 dt = 1;
13 t  = 0:dt:600;
14 N  = length(t)-1;
15 sysc = ss(A,B,[],[]);
16 sysd = c2d(sysc,dt);
17 A = sysd.A;
18 B = sysd.B;
```

For a given final state and control weighting matrix, the matrix Λ can be computed as `L` in code lines 21-24. Then for a given initial state, the optimal control and state trajectories are computed on lines 27-31.

```
19 xN = [0;0;0;0;0;0];
20 R = eye(3);
21 L = 0;
22 for i = 0:N-1
23     L = L + A^(N-i-1)*B*inv(R)*B.'*A.'^(N-i-1);
24 end
25
26 x0 = [100;0;0;0;5;0];
27 x(:,1) = x0;
28 for k = 1:N
29     u(:,k) = -inv(R)*B.'*A.'^(N-k)*inv(L)*(A^N*x0-xN);
30     x(:,k+1) = A*x(:,k) + B*u(:,k);
31 end
```

The resulting trajectory in x-y space and control time histories are shown in Figure 4.3.

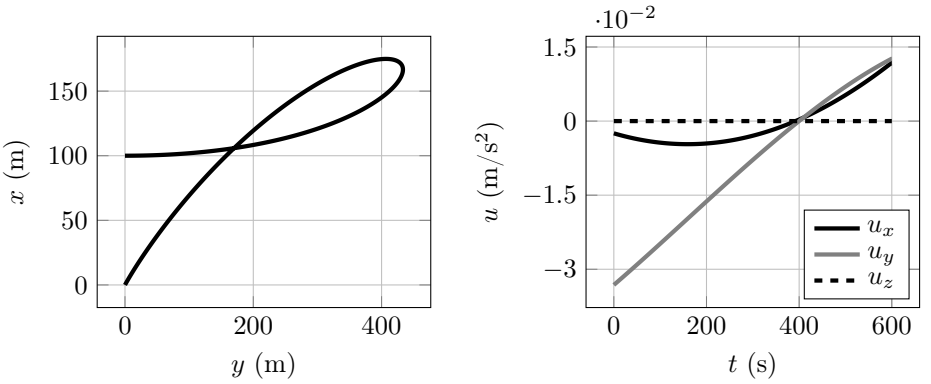


Figure 4.3: Relative orbit control trajectories.

4.4 Linear Quadratic Regulation

The linear quadratic regulation (LQR) problem has a running cost that penalizes state and control deviations from zero. A terminal cost penalizes state deviations at the final point.

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}x_N^\top S_N x_N + \frac{1}{2}\sum_{k=0}^{N-1} x_k^\top Q x_k + u_k^\top R u_k \\ \text{subject to} \quad & x_{k+1} = Ax_k + Bu_k \end{aligned} \tag{4.77}$$

It is assumed that $S_N = S_N^\top \geq 0$, $Q = Q^\top \geq 0$, $R = R^\top > 0$. The matrices A, B, Q, and R could be time-varying. Not much changes in the analysis and they are assumed constant only for notational simplicity. As before, the problem is convex and trivially satisfies the requirement for strict feasibility such that $\lambda^0 = 1$.

To analyze this problem, write the Hamiltonian.

$$H^k = \frac{1}{2}\left(x_k^\top Q x_k + u_k^\top R u_k\right) + \lambda_{k+1}^\top \left(Ax_k + Bu_k\right) \tag{4.78}$$

The optimality conditions are

$$\lambda_k = Qx_k + A^\top \lambda_{k+1}, \tag{4.79}$$

$$\lambda_N = S_N x_N, \tag{4.80}$$

$$0 = Ru_k + B^\top \lambda_{k+1}. \tag{4.81}$$

Solving for the control gives

$$u_k = -R^{-1}B^\top \lambda_{k+1}. \tag{4.82}$$

The techniques used before no longer work because the recursion for λ_k is no longer homogeneous. The technique to use is called the sweep method in which it is assumed, based on the form in Eq. (4.80), that there are matrices S_k such that

$$\lambda_k = S_k x_k. \tag{4.83}$$

Formulas for S_k are needed. Substituting into the state equation gives

$$x_{k+1} = Ax_k - BR^{-1}B^\top S_{k+1}x_{k+1}. \tag{4.84}$$

Solving for x_{k+1} gives

$$x_{k+1} = \left(I + BR^{-1}B^\top S_{k+1}\right)^{-1} Ax_k, \tag{4.85}$$

which is a forward, homogeneous recursion for the state. Substituting Eq. (4.83) into Eq. (4.79) gives

$$S_k x_k = Qx_k + A^\top S_{k+1} x_{k+1} \tag{4.86a}$$

$$= Qx_k + A^\top S_{k+1} \left(I + BR^{-1}B^\top S_{k+1}\right)^{-1} Ax_k. \tag{4.86b}$$

Since this must hold for all x_k, it is concluded that

$$S_k = Q + A^\top S_{k+1} \left(I + BR^{-1}B^\top S_{k+1}\right)^{-1} A. \tag{4.87a}$$

Another way to write this (using the matrix inversion lemma) is

$$S_k = Q + A^\top \Big(S_{k+1} - S_{k+1}\, B \left(B^\top S_{k+1}\, B + R\right)^{-1} B^\top S_{k+1}\Big) A. \tag{4.87b}$$

The above equation is known as the Riccati equation. Since S_N is known, all S_k can be found. The optimal control is

$$u_k = -R^{-1}B^\top S_{k+1} x_{k+1}, \tag{4.88}$$

which depends upon a future state. Eliminate x_{k+1} by substituting in the state equation.

$$u_k = -R^{-1}B^\top S_{k+1} \left(Ax_k + Bu_k\right) \tag{4.89}$$

$$\implies \quad \left(I + R^{-1}B^\top S_{k+1}B\right) u_k = -R^{-1}B^\top S_{k+1}Ax_k \tag{4.90}$$

Pre-multiplying by R and inverting gives

$$u_k = -\left(R + B^\top S_{k+1}B\right)^{-1} B^\top S_{k+1}Ax_k. \tag{4.91}$$

Define the Kalman gain as

$$K_k = \left(R + B^\top S_{k+1}B\right)^{-1} B^\top S_{k+1}A \tag{4.92}$$

so that the control is simply

$$u_k = -K_k x_k. \tag{4.93}$$

The Kalman gain is time-varying even though A, B, Q, and R are time-invariant. This is a feedback, or closed-loop, control law because it depends on the current state x_k – not the initial state x_0.

While simple to implement, a sequence of gain matrices must be stored. This storage begs the question: Is it possible to come up with a single matrix K to control the system? One approach to answering the question is to consider very long time horizons where $N - k \to \infty$. If the S_k recursion reaches steady state, then $S_k = S_{k+1} \equiv S$. The Riccati equation (4.87) becomes the Algebraic Riccati Equation (ARE)

$$S = Q + A^\top \Big(S - SB \left(B^\top SB + R\right)^{-1} B^\top S\Big) A. \tag{4.94}$$

The Kalman gain is then constant. There are of course serious questions such as convergence of the limit, dependence of S on S_N, and performance of the resulting system. The control is no longer optimal but it may still regulate the system. For this all to work, there are control-centric requirements related to stabilizability and observability of the system. Details are omitted here. Assuming the limit converges, a technique for finding the steady state value is to pick an S_N and iterate backward until steady state is reached. The feedback control now uses a constant gain and no storage is required.

MATLAB Implementation of Relative Orbit Regulation

To demonstrate LQR, the linearized CW equations (2.31) are used. The problem is to regulate the state and control close to zero. The mean motion is 4 rad/hr and the time step for discretization is one second. As such, lines 1-18 of the previous implementation remain the same here.

Lines 19-21 fix the objective matrices. Lines 23-27 back propagate to store the feedback gains. Lines 29-34 compute the optimal control and state trajectories.

```
19  SN = eye(6);
20  Q  = 1e-1*eye(6);
21  R  = 1e+6*eye(3);
22
23  S = SN;
24  for k = N:-1:1
25      K(:,:,k) = inv(R+B'*S*B)*B'*S*A;
26      S = Q + A'*S*inv(eye(6)+B*inv(R)*B'*S)*A;
27  end
28
29  x0 = [100;0;0;0;5;0];
30  x(:,1) = x0;
31  for k = 1:N
32      u(:,k) = -K(:,:,k)*x(:,k);
33      x(:,k+1) = A*x(:,k) + B*u(:,k);
34  end
```

The resulting trajectory in x-y space and control time histories are shown in Figure 4.4. Because the LQR objective penalizes state deviations from zero, the state trajectory remains closer to zero than it does in LQC. For this reason, the scale of the left graph in Figure 4.4 is smaller than that in Figure 4.3.

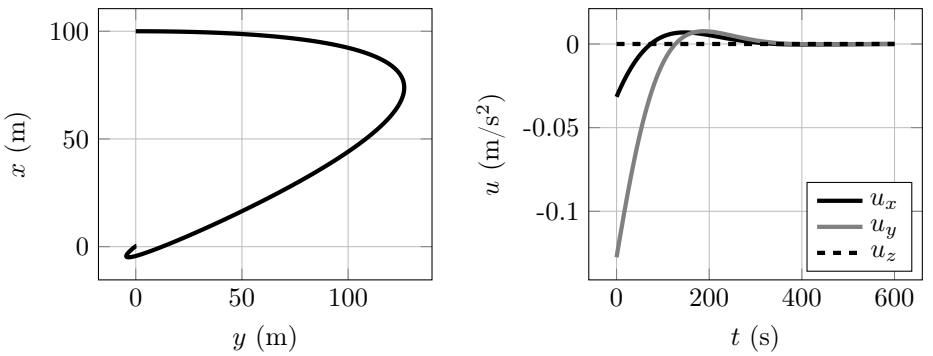


Figure 4.4: Relative orbit regulation trajectories.

A constant gain solution can be tried by replacing the gain `K(:,:,k)` in line 32 with its value at the initial time `K(:,:,1)`.

4.5 Linear Quadratic Tracking

Yet another variation of the problem is the linear quadratic tracking (LQT) problem. The goal is to now have the state track a reference trajectory. The results of this section reduce to those of the previous section when the reference trajectory is zero. The given reference trajectory r_k is one that may not depend on all states such that the goal is to balance the use of control and having

$$Cx_k \cong r_k. \tag{4.95}$$

The linear quadratic tracking problem is the following.

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\left(Cx_N - r_N\right)^\top P\left(Cx_N - r_n\right) \\ & + \frac{1}{2}\sum_{k=0}^{N-1}\left[\left(Cx_k - r_k\right)^\top Q\left(Cx_k - r_k\right) + u_k^\top R u_k\right] \\ \text{subject to} \quad & x_{k+1} = Ax_k + Bu_k \end{aligned} \tag{4.96}$$

To begin analyzing the problem, write the Hamiltonian.

$$H^k = \frac{1}{2}\left(Cx_k - r_k\right)^\top Q\left(Cx_k - r_k\right) + \frac{1}{2}u_k^\top R u_k + \lambda_{k+1}^\top\left(Ax_k + Bu_k\right) \tag{4.97}$$

The abnormal multiplier has again been set to one because of convexity and satisfaction of the strictly feasible requirement. The costate equation is

$$\lambda_k = A^\top\lambda_{k+1} + C^\top QCx_k - C^\top Qr_k. \tag{4.98}$$

The stationarity condition is

$$0 = Ru_k + B^\top\lambda_{k+1}, \tag{4.99}$$

$$\implies \quad u_k = -R^{-1}B^\top\lambda_{k+1}. \tag{4.100}$$

The transversality condition is

$$\begin{aligned} \lambda_N &= C^\top P\left(C\,x_N - r_N\right) \\ &= \underbrace{C^\top P\,C}_{S_N}\,x_N - \underbrace{C^\top P\,r_N}_{v_N}. \end{aligned} \tag{4.101}$$

As done before, use the sweep method whereby it is assumed that

$$\lambda_k = S_k x_k - v_k. \tag{4.102}$$

The control equation is then

$$u_k = -R^{-1}B^\top\left(S_{k+1}x_{k+1} - v_{k+1}\right), \tag{4.103}$$

$$\implies \quad x_{k+1} = Ax_k - B\,R^{-1}B^\top S_{k+1}x_{k+1} + BR^{-1}B^\top v_{k+1}. \tag{4.104}$$

Solving for x_{k+1} gives

$$x_{k+1} = \left(I + BR^{-1}B^\top S_{k+1}\right)^{-1}\left(Ax_k + BR^{-1}B^\top v_{k+1}\right). \tag{4.105}$$

Using this in the costate equation (4.98) gives

$$\begin{aligned} S_k\, x_k - v_k &= C^\top Q\, C\, x_k - C^\top Q\, r_k + A^\top \left(S_{k+1}\, x_{k+1} - v_{k+1}\right) \\ &= C^\top Q\, C\, x_k - C^\top Q\, r_k - A^\top v_{k+1} \\ &\quad + A^\top S_{k+1} \left(I + B\, R^{-1} B^\top S_{k+1}\right)^{-1} \left(A\, x_k + B\, R^{-1} B^\top v_{k+1}\right). \end{aligned} \tag{4.106}$$

Grouping all of the x_k terms and non-x_k terms gives

$$\begin{aligned} &\left[-S_k + C^\top Q C + A^\top S_{k+1} \left(I + B R^{-1} B^\top S_{k+1}\right)^{-1} A\right] x_k \\ &+ \left[v_k - C^\top Q r_k - A^\top v_{k+1} + A^\top S_{k+1} \left(I + B R^{-1} B^\top S_{k+1}\right)^{-1} B R^{-1} B^\top v_{k+1}\right] = 0. \end{aligned} \tag{4.107}$$

Since this must hold for all x_k, both terms need to be zero. The first term determines S_k as a function of S_{k+1}. The second term determines v_k as a function of S_{k+1} and v_{k+1}. The optimal control is then

$$u_k = -R^{-1} B^\top \lambda_{k+1} \tag{4.108a}$$

$$= -R^{-1} B^\top \left(S_{k+1} x_{k+1} - v_{k+1}\right) \tag{4.108b}$$

$$= -R^{-1} B^\top S_{k+1} \left(A x_k + B u_k\right) + R^{-1} B^\top v_{k+1}. \tag{4.108c}$$

Pre-multiplying by R and solving for u_k gives

$$u_k = \left(R + B^\top S_{k+1} B\right)^{-1} B^\top \left(-S_{k+1} A x_k + v_{k+1}\right). \tag{4.109}$$

Two gain matrices are now defined

$$\text{Feedback Gain :} \qquad K_k = \left(R + B^\top S_{k+1} B\right)^{-1} B^\top S_{k+1} A \tag{4.110}$$

$$\text{Feedforward Gain :} \qquad K_k^v = \left(R + B^\top S_{k+1} B\right)^{-1} B^\top \tag{4.111}$$

such that

$$u_k = -K_k x_k + K_k^v v_{k+1}. \tag{4.112}$$

One can again look for sub-optimal constant feedback gains. Satisfaction of control-centric requirements implies that the recursions for K_k and K_k^v reach steady state as $N - k \to \infty$. The constant gains and control are

$$K = \left(B^\top S_\infty B + R\right)^{-1} B^\top S_\infty A, \tag{4.113}$$

$$K^v = \left(B^\top S_\infty B + R\right)^{-1} B^\top, \tag{4.114}$$

$$u_k = -K x_k + K^v v_{k+1}. \tag{4.115}$$

It appears that storing the v_k sequence is required, but it is not. Instead, store v_0 and propagate forward using

$$v_{k+1} = (A - BK)^{-\top} v_k - (A - BK)^{-\top} C^\top Q r_k. \tag{4.116}$$

MATLAB Implementation of Relative Orbit Tracking

To demonstrate LQT, the linearized CW equations (2.31) are used and lines 1-18 of the previous implementations remain the same. The problem is to traverse two circular relative orbits with radius of 100 meters and period of 5 minutes. Such a reference trajectory is created in lines 19-23. The objective matrices are specified in lines 25-29. The feedback and feedforward recursions are computed by backward propagation in lines 31-45. Finally, in lines 47-52, the optimal control and state trajectories are computed.

```
19 % Create a reference trajectory to follow...
20 W  = 2*pi/300;
21 rx = 100*cos(W*t);
22 ry = 100*sin(W*t);
23 r = [rx; ry; 0*t];
24
25 % Define the data for the LQT problem...
26 C = [eye(3) zeros(3)];
27 P = eye(3);
28 Q = 1e2*eye(3);
29 R = eye(3);
30
31 % Compute the feedback and feedforward gains...
32 S = C'*P*C;
33 V = C'*P*r(:,N+1);
34
35 Ss(:,:,N+1) = S;
36 Vs(:,:,N+1) = V;
37 for k = N:-1:1
38     K(:,:,k)  = inv(R+B'*S*B)*B'*S*A;
39     Kv(:,:,k) = inv(R+B'*S*B)*B';
40     V = C'*Q*r(:,k) + A'*V ...
41       - A'*S*inv(eye(6)+B*inv(R)*B'*S)*B*inv(R)*B'*V;
42     S = C'*Q*C + A'*S*inv(eye(6)+B*inv(R)*B'*S)*A;
43     Ss(:,:,k) = S;
44     Vs(:,:,k) = V;
45 end
46
47 % Implement the control and simulate the system...
48 x0 = [100;0;0;0;5;0];
49 x(:,1) = x0;
50 for k = 1:N
51     u(:,k)   = -K(:,:,k)*x(:,k) + Kv(:,:,k)*Vs(:,:,k+1);
52     x(:,k+1) = A*x(:,k) + B*u(:,k);
53 end
```

The resulting trajectory in x-y space and control time histories are shown in Figure 4.5.

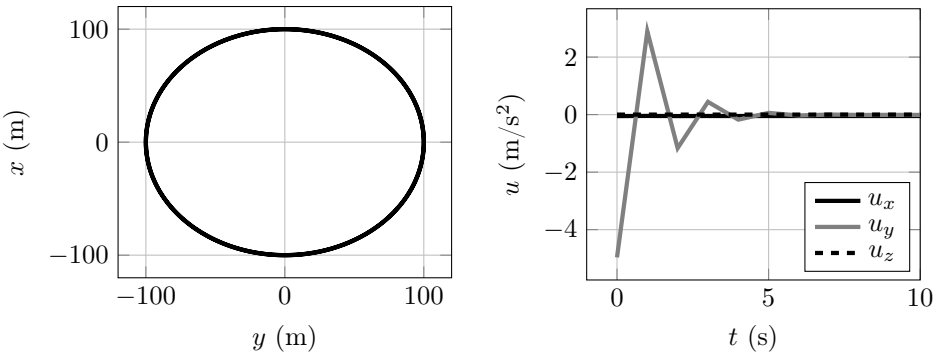


Figure 4.5: Relative orbit tracking trajectories.

The figures do not show well the time element of the tracking performance. To better visualize the simulation, add the following code to your script. In the generated animation, the blue dot is the reference position and the red dot is the actual position.

```
colorMap = [0,0,1; 1 0 0];
xylim = [-100 100 -100 100];
t_now = linspace(t(1),t(N+1),100);
figure
for i = 1:length(t_now)
    rx_now = interp1(t,rx,t_now(i));
    ry_now = interp1(t,ry,t_now(i));
    x_now  = interp1(t,x(1,:),t_now(i));
    y_now  = interp1(t,x(2,:),t_now(i));
    scatter([ry_now; y_now], [rx_now; x_now], ...
            [200;100], colorMap, 'filled'), grid on
    axis(xylim)
    title('Blue = Reference, Red = Tracker')
    pause(.1)
end
```

4.6 Trajectory Following using LQR

The linear quadratic framework requires dynamics of the form $x_{k+1} = A_k x_k + B_k u_k$, which excludes nonlinear systems of the form $x_{k+1} = f^k(x_k, u_k)$ and even linear systems with a disturbance of the form $x_{k+1} = A_k x_k + B_k u_k + E_k w_k$. Even so, the LQR problem remains relevant. To see this, consider the flat planet model in discrete-time

$$x_{k+1} = Ax_k + Bu_k + Bg, \tag{4.117}$$

where A and B are the discrete-time versions of those presented in Eq. (2.10). Let u_k^* be a nominal control generating the nominal state trajectory x_k^*. This nominal trajectory can be designed by any means, for example, polynomial guidance discussed in Section 2.2. An off-nominal control u_k generates an off-nominal state trajectory x_k. The control and state perturbations are defined to be $\delta u_k = u_k - u_k^*$ and $\delta x_k = x_k - x_k^*$,

respectively. The evolution of δx_k is given by

$$\delta x_{k+1} = A\delta x_k + B\delta u_k. \tag{4.118}$$

Obviously, dynamics of the perturbed trajectory fit in the linear quadratic framework. The problem of following the nominal trajectory, or regulating the perturbed state close to zero, is an LQR problem in the δ variables. A MATLAB implementation for planetary descent is given next.

MATLAB Implementation of Descent Trajectory Following

To begin the implementation, data must be specified and the nominal trajectories computed. The nominal initial position and velocity are `r0` and `v0`. The desired final position and velocity are `r1` and `v1`. Units are m and m/s, respectively. The final time is `t1`, and for discretization purposes `N` is set to 1000. Other constants used in subsequent analysis are the gravitational acceleration vector `g`, the identity matrix `I`, and zero matrix `Z`.

```
% Nominal boundary conditions and time
r0 = [1000; 1000];
v0 = [-25; 0];
r1 = [0;0];
v1 = [0;0];

t1 = 100;
N  = 1e3;
t  = linspace(0,t1,N);

% Other constants
g  = [0;-9.81];
I  = eye(2);
Z  = zeros(2);
```

Nominal control and state trajectories are computed using polynomial guidance as described in Section 2.2. The nominal control is stored in `u_nom` and the nominal state trajectories are stored in `x_nom`. The first row is range, the second is altitude, the third is range rate, and the fourth is altitude rate.

```
% Nominal trajectory
C = [t1^2*I, t1^3*I; 2*t1*I, 3*t1^2*I] \ [r1-r0-v0*t1; v1-v0];
c2 = C(1:2); c3 = C(3:4);
u_nom = 2*c2 + 6*c3*t-g;
v_nom = v0 + 2*c2*t + 3*c3*t.^2;
r_nom = r0 + v0*t + c2*t.^2 + c3*t.^3;
x_nom = [r_nom; v_nom];
```

The flat planet model equations of motion given in Eq. (2.10) are discretized.

```
% Discretization
A = [Z, I; Z Z];
B = [Z; I];
```

```
25  sysc = ss(A,B,[],[]);
26  sysd = c2d(sysc,t(2)-t(1));
27  A = sysd.A;
28  B = sysd.B;
```

The discrete-time LQR problem is set up by defining the weighting matrices `SN`, `Q`, and `R`. This is followed in lines 34-38 by a backward recursion for the feedback gain matrix `K`.

```
29  % Compute the feedback gain
30  SN = eye(4);
31  Q  = eye(4);
32  R  = 1e3*eye(2);
33
34  S = SN;
35  for k = N:-1:1
36      K(:,:,k) = inv(R+B'*S*B)*B'*S*A;
37      S = Q + A'*S*inv(eye(4)+B*inv(R)*B'*S)*A;
38  end
```

With these elements in place, the simulation can be conducted. Line 40 specifies the perturbation in the state at the initial time. The actual state of the system at the initial time is specified in line 41 as `x`. Once in the simulation loop, line 43 calculates the state perturbation, the control perturbation on line 44, the actual control on line 45, and the next state on line 46. For simplicity, the constant, steady-state gain matrix `K(:,:,1)` is used at all times.

```
39  % Simulation
40  dx0 = [50; 75; -20; -5];
41  x(:,1) = x_nom(:,1)+dx0;
42  for k = 1:N-1
43      dx(:,k)  = x(:,k) - x_nom(:,k);
44      du(:,k)  = -K(:,:,1)*dx(:,k);
45      u(:,k)   = u_nom(:,k)+du(:,k);
46      x(:,k+1) = A*x(:,k) + B*u(:,k) + B*g;
47  end
```

A plot of the nominal and actual position trajectories is shown in Figure 4.6. The shape of the actual trajectory and amount of control used are influenced by the choice of weighting matrices. The actual state trajectory can be driven back to the nominal more quickly by reducing `R`. The amount of control usage can be driven down by increasing `R`.

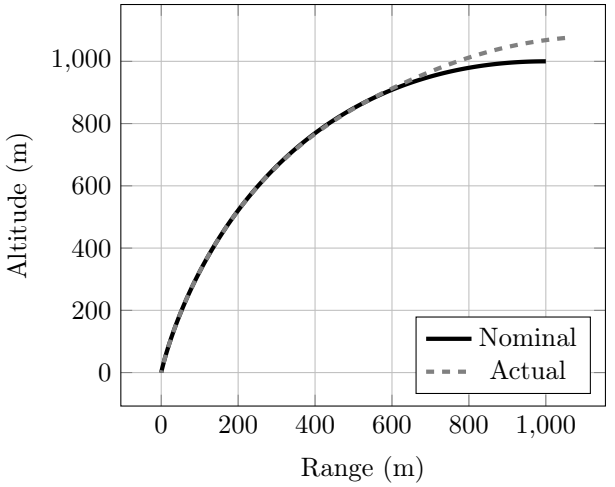


Figure 4.6: Nominal and actual planetary descent trajectories.

4.7 Discretization of Nonlinear Systems

Having given a thorough treatment of discrete optimal control problems with linear dynamics, attention is now turned to nonlinear systems. Discretization of linear systems was described in Chapter 2 by assuming the control is piecewise constant and using the state transition matrix. A different idea is needed for nonlinear systems.

Given a nonlinear function $f : \mathbb{R} \to \mathbb{R}$, it may be possible to decompose it into linear combinations of basis functions T_i, i.e.,

$$f(t) = \sum_{i=0}^{\infty} a_i T_i(t). \tag{4.119}$$

The idea of decomposing a quantity into a weighted sum of basis components is familiar from linear algebra. For example, the vector

$$\begin{bmatrix} 2 \\ 3 \end{bmatrix} = 2 \underbrace{\begin{bmatrix} 1 \\ 0 \end{bmatrix}}_{\text{basis vectors}} + 3 \underbrace{\begin{bmatrix} 0 \\ 1 \end{bmatrix}}_{\text{basis vectors}}. \tag{4.120}$$

Just as there are many bases for $\mathbb{R}^2$, and hence many ways of decomposing the vector, there are many bases for a function space. An important difference, however, is that function spaces are infinite-dimensional. One such basis is the Chebyshev polynomials. These polynomials are defined on the domain $[-1, 1]$ and given by the formulas

$$T_0(t) = 1, \tag{4.121a}$$
$$T_1(t) = t, \tag{4.121b}$$
$$T_{n+1}(t) = 2tT_n(t) - T_{n-1}(t). \tag{4.121c}$$

Example 4.1. Compute $T_1(t)$ and $T_2(t)$.

$$\begin{aligned} T_2(t) &= 2tT_1(t) - T_0(t) \\ &= 2t\,(t) - 1 \\ &= 2t^2 - 1, \end{aligned}$$

$$\begin{aligned} T_3(t) &= 2tT_2(t) - T_1(t) \\ &= 2t\left(2t^2 - 1\right) - t \\ &= 4t^3 - 2t - t \\ &= 4t^3 - 3t. \end{aligned}$$

★

A related basis uses Chebyshev polynomials of the second kind given by the formulas

$$U_0(t) = 1, \tag{4.122a}$$
$$U_1(t) = 2t, \tag{4.122b}$$
$$U_{n+1}(t) = 2tU_n(t) - U_{n-1}(t). \tag{4.122c}$$

Both kinds of Chebyshev polynomials form an orthogonal basis. The two kinds of polynomials are related by

$$2T_n(t) = U_n(t) - U_{n-2}(t), \tag{4.123}$$
$$T_n(t) = U_n(t) - tU_{n-1}(t). \tag{4.124}$$

They satisfy a number of interesting properties.

$$T_n(1) = 1 \tag{4.125a}$$
$$T_n(-1) = (-1)^n \tag{4.125b}$$
$$U_n(1) = n + 1 \tag{4.125c}$$
$$U_n(-1) = (-1)^n\,(n+1) \tag{4.125d}$$

Their derivatives are also related.

$$\dot{T}_i(t) = iU_{i-1}(t) \tag{4.126}$$
$$\dot{U}_i(t) = \frac{(i+1)\,T_{i+1}(t) - tU_n(t)}{t^2 - 1} \tag{4.127}$$

Suppose the time interval $[-1, 1]$ is discretized into $N + 1$ nodes t_0, t_1, …, t_N. The function values at the nodes are $f(t_k)$. One can then use the first $N + 1$ polynomials to approximate the function.

$$f(t) \cong \sum_{k=0}^{N} a_k T_k(t)\ . \tag{4.128}$$

This requires first solving for the a_k values.

$$\underbrace{\begin{bmatrix} f(t_0) \\ f(t_1) \\ \vdots \\ f(t_N) \end{bmatrix}}_{f} = \underbrace{\begin{bmatrix} T_0(t_0) & T_1(t_0) & \cdots \\ T_0(t_1) & T_1(t_1) & \cdots \\ \vdots & \vdots & \\ T_0(t_N) & T_1(t_N) & \cdots \end{bmatrix}}_{\mathcal{T}} \underbrace{\begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_N \end{bmatrix}}_{a} \tag{4.129}$$

$$\implies \quad a = \mathcal{T}^{-1} f. \tag{4.130}$$

After computing each $\dot{T}_k(t)$, one can also approximate the derivative of f using

$$\dot{f}(t) \cong \sum_{k=0}^{N} a_k \dot{T}_k(t). \tag{4.131}$$

With the same matrix notation as above, evaluation at the nodes gives

$$\begin{aligned} \dot{f} &= \dot{\mathcal{T}}\, a \\ &= \underbrace{\dot{\mathcal{T}}\, \mathcal{T}^{-1}}_{D} f \\ &= Df. \end{aligned} \tag{4.132}$$

That is, there is a matrix D that maps the function values at nodes to derivative values at nodes. This matrix is called the differentiation matrix.

Placement of the nodes is done to minimize the approximation error. The optimally placed nodes are called the Chebyshev nodes.

$$t_k = -\cos\left(\frac{\pi}{2}\frac{(2k+1)}{N+1}\right), \quad k = 0, \ldots, N \tag{4.133}$$

It is common for these polynomials to be used in optimization and boundary value problems. However, the Chebyshev nodes do not occupy the interval endpoints -1 and 1. Therefore, they are sometimes approximated as

$$t_k = -\cos\left(\frac{\pi k}{N}\right), \quad k = 0, \ldots, N \tag{4.134}$$

so that $t_0 = -1$ and $t_N = 1$. The differentiation matrix depends on the node selection.

Example 4.2. Using the approximate Chebyshev nodes, compute $\mathcal{T}$, $\dot{\mathcal{T}}$, and D with $N = 2$. Eq. (4.134) gives nodes $t_0 = -1$, $t_1 = 0$, and $t_2 = +1$. The $\mathcal{T}$ matrix is

$$\mathcal{T} = \begin{bmatrix} T_0(t_0) & T_1(t_0) & T_2(t_0) \\ T_0(t_1) & T_1(t_1) & T_2(t_1) \\ T_0(t_2) & T_1(t_2) & T_2(t_2) \end{bmatrix} = \begin{bmatrix} 1 & -1 & 1 \\ 1 & 0 & -1 \\ 1 & 1 & 1 \end{bmatrix}.$$

Its derivative is

$$\dot{\mathcal{T}} = \begin{bmatrix} \dot{T}_0(t_0) & \dot{T}_1(t_0) & \dot{T}_2(t_0) \\ \dot{T}_0(t_1) & \dot{T}_1(t_1) & \dot{T}_2(t_1) \\ \dot{T}_0(t_2) & \dot{T}_1(t_2) & \dot{T}_2(t_2) \end{bmatrix} = \begin{bmatrix} 0 & 1 & -4 \\ 0 & 1 & 0 \\ 0 & 1 & 4 \end{bmatrix}.$$

The differentiation matrix is

$$D = \dot{\mathcal{T}}\mathcal{T}^{-1} = \frac{1}{2}\begin{bmatrix} -3 & 4 & -1 \\ -1 & 0 & 1 \\ 1 & -4 & 3 \end{bmatrix}.$$

★

MATLAB Implementation of Chebyshev Discretization

Numerical implementations are straightforward because MATLAB has built-in functions `chebyshevT` and `chebyshevU`. Lines 1-4 define the number of time steps N and associated node times t_k denoted `t`. Lines 6-10 calculate $\mathcal{T}$. Lines 11-15 calculate $\dot{\mathcal{T}}$. Finally, the differentiation matrix is computed in line 18.

```
1  % N and node times
2  N = 10;
3  k = 0:N;
4  t = -cos(pi*k/N)';
5
6  % Calculate T
7  for k = 1:N+1
8      T(:,k)  = chebyshevT(k-1,t);
9  end
10
11 % Calculate Tdot
12 Tdot(:,1) = zeros(N+1,1);
13 for k = 1:N
14     Tdot(:,k+1) = k*chebyshevU(k-1,t);
15 end
16
17 % Calculate the differentiation matrix
18 D = Tdot*inv(T);
```

Consider now the function $f(t) = \sin(t) + \frac{1}{2}t^2$. The goal is to use the differentiation matrix to approximate its derivative at the node times. Because the function is simple, the analytical derivative is $f'(t) = \cos(t) + t$. In MATLAB, we evaluate the function at the node times and approximate the derivative.

```
19 % Function evaluation and derivative approximation
20 f  = sin(t)+.5*t.^2;
21 df = D*f;
```

The error in the approximation is the difference between the analytical solution and the approximation.

```
% Error computation
fp = cos(t)+1.0*t;
error = fp-df;
```

For the data given, the error is on the order `1e-9`. If the domain of interest is not $[-1, 1]$, the node times and formulas need to be adjusted. Suppose the interval of interest is $[0, t_f]$. The original domain needs to be translated one unit to the right and stretched by $t_f/2$. New time nodes `s` are related to the old time nodes `t` using `s = tf/2*(t+1)`. The new differentiation matrix `E` is related to the old differentiation matrix `D` using `E = (2/tf)*D`.

Lastly, the differentiation matrix can be used to discretize a nonlinear dynamical system. Consider the following nonlinear differential equations.

$$\dot{x}_1(t) = f_1(t, x_1(t), \ldots, x_n(t)) \tag{4.135a}$$

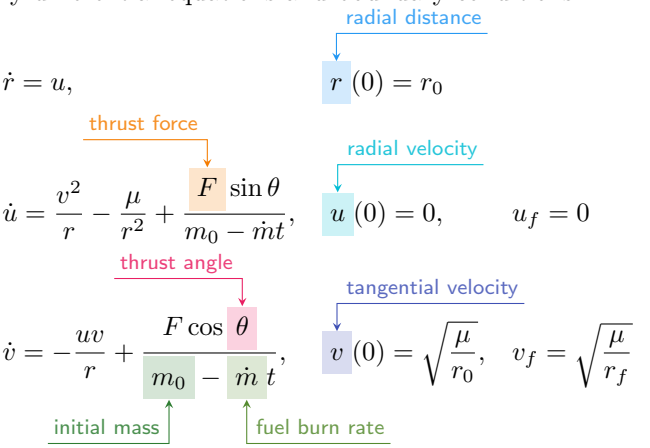
$$\vdots$$

$$\dot{x}_n(t) = f_n(t, x_1(t), \ldots, x_n(t)) \tag{4.135b}$$

In code, stack the node times in a vector `t` as done before. The vector containing values of the state x_i evaluated at node times is denoted `Xi`. The vector containing values of the function f_i evaluated at node times is denoted `Fi`. As such, the objects `t`, `Xi`, and `Fi` are all column vectors of dimension $N + 1$. Enforcement of the i^{th} differential equation is approximated through the algebraic equation `D*Xi = Fi`. When the function f_i is nonlinear in the states, this equality constraint is nonlinear in the states. This process is demonstrated in Section 4.8.

4.8 Maximal Orbit Raise

In this section, the problem of transferring from a sun-centered circular orbit to the largest possible circular orbit with fixed thrust magnitude and fixed flight time is considered. The control variable is the thrust angle. The controlled two-body equation of motion in radial and tangential coordinates is used. The dynamics are described by the following ordinary differential equations and boundary conditions.



$$\begin{aligned}
&\dot{r} = u, && r(0) = r_0 \\
&\dot{u} = \frac{v^2}{r} - \frac{\mu}{r^2} + \frac{F \sin\theta}{m_0 - \dot{m}t}, && u(0) = 0, \quad u_f = 0 \\
&\dot{v} = -\frac{uv}{r} + \frac{F\cos\theta}{m_0 - \dot{m}\,t}, && v(0) = \sqrt{\frac{\mu}{r_0}}, \quad v_f = \sqrt{\frac{\mu}{r_f}}
\end{aligned} \tag{4.136}$$

The objective is to maximize the final radius r_f. To be solved as a discrete optimal control problem, the nonlinear dynamics need to be discretized. Because of the numerous nonlinearities, analytical analysis is difficult. Solution of the problem is more easily attained using numerical optimization and a technique known as direct transcription. In direct transcription, the problem is discretized and both states and controls are considered optimization variables. These topics are covered in the following subsection.

MATLAB Implementation of Direct Transcription

In numerical optimization, it is important to use units so that the problem data are similar in magnitude. Units used in MATLAB are astronomical units (AU), days, and kilograms. The scaling information, problem data, and initial conditions are specified below.

```
% Unit scalings
MU = 1 / 1;
DU = 1 / 149.6e9;
TU = 1 / 86400;

% Data
F    = 4        * MU*DU/TU^2;
m0   = 4500     * MU;
mdot = 7e-5     * MU/TU;
mu   = 1.327e20 * DU^3/TU^2;
tf   = 193;

% Initial conditions
r0 = 1;
u0 = 0;
v0 = sqrt(mu/r0);
x0 = [r0;u0;v0];
```

With this information, the node times and differentiation matrix can be computed using Chebyshev polynomials.

```
% Node times and differentiation matrix
N = 50;
k = 0:N;
t = -cos(pi*k/N)';

% Calculate T
for k = 1:N+1
    T(:,k)  = chebyshevT(k-1,t);
end

% Calculate Tdot
Tdot(:,1) = zeros(N+1,1);
for k = 1:N
    Tdot(:,k+1) = k*chebyshevU(k-1,t);
end
```

```
33
34  % Calculate the differentiation matrix
35  D = Tdot*inv(T);
36
37  % Scale the nodes and differentiation matrix
38  t = tf/2*(t+1);
39  D = 2/tf*D;
```

To solve the problem using MATLAB's `fmincon`, an initial guess must be provided. The initial guess is important and can have dramatic effects on the solver performance. Our initial guess is simply a linear interpolation between the initial and final points. The final radius is not known but guessed to be 2 AUs. Neither the initial nor final control value is known. They are guessed to be 30 and 300 degrees, respectively. The guessed states and control are then stacked into a long vector to be passed to the solver.

```
40  % Initial guess
41  rguess  = linspace(1,2,N+1)';
42  uguess  = linspace(0,0,N+1)';
43  vguess  = linspace(sqrt(mu/r0),sqrt(mu/2),N+1)';
44  thguess = linspace(30,300,N+1)' * pi/180;
45  xguess  = [rguess; uguess; vguess; thguess];
```

For `fmincon`, objective and constraint functions must be specified. On line 66, the optimization variable is reshaped into four columns. The first column contains r values, the second contains u values, the third contains v values, and the fourth contains θ values. This is consistent with the construction of our initial guess. On line 67, the objective function is the negative of the final radius because the problem is to maximize.

```
64  % Objective function
65  function J = obj(x,N)
66  x = reshape(x,N+1,4);
67  J = -x(N+1,1);
68  end
```

The constraint function enforces the discretized differential equations and boundary conditions. As on line 66 of the objective function, line 71 reshapes the optimization variable into four columns and the variables are extracted on line 72. The right-hand sides of the differential equations are coded on lines 73-75. Initial conditions are enforced on line 77. The discretized differential equations are enforced on line 78. The terminal constraints are enforced on line 79. There are no inequality constraints such that `cin = []` on line 76.

```
69  % Constraint function
70  function [cin,ceq] = con(x,r0,u0,v0,m0,mdot,F,mu,t,D,N)
71  x = reshape(x,N+1,4);
72  r = x(:,1); u = x(:,2); v = x(:,3); theta = x(:,4);
73  rdot = u;
74  udot = v.^2./r - mu./r.^2 + F*sin(theta) ./ (m0-mdot*t);
75  vdot = -u.*v./r + F*cos(theta) ./ (m0-mdot*t);
76  cin = [];
77  ceq = [r(1)-r0; u(1)-u0; v(1)-v0];
```

```
ceq = [ceq; D*r-rdot; D*u-udot; D*v-vdot];
ceq = [ceq; u(N+1)-0; v(N+1)-sqrt(mu/r(N+1))];
end
```

Finally, the optimization problem can be solved by calling `fmincon`.

```
% Optimize
ops = optimoptions('fmincon','Display','iter',...
                   'EnableFeasibilityMode',true,...
                   'SubproblemAlgorithm','cg',...
                   'MaxFunEvals',5e4);
pobj = @(x) obj(x,N);
pcon = @(x) con(x,r0,u0,v0,m0,mdot,F,mu,t,D,N);
x = fmincon(pobj,xguess,[],[],[],[],[],[],pcon,ops);
```

The solver converges to a solution and the maximized final radius is approximately 1.56 AU. MATLAB code to generate plots is below.

```
% Extract the solution
x = reshape(x,N+1,4);
r = x(:,1); u = x(:,2); v = x(:,3); theta = x(:,4);

% Plot the solution
figure, plot(t,r), grid on
xlabel('t (days)'), ylabel('r (AU)')

figure, plot(t,theta*180/pi), grid on
xlabel('t (days)'), ylabel('theta (deg)')
```

A plot of the radius as a function of time is shown in Figure 4.7. The trajectory begins in the lower left and rises to a maximum radius in the top right.

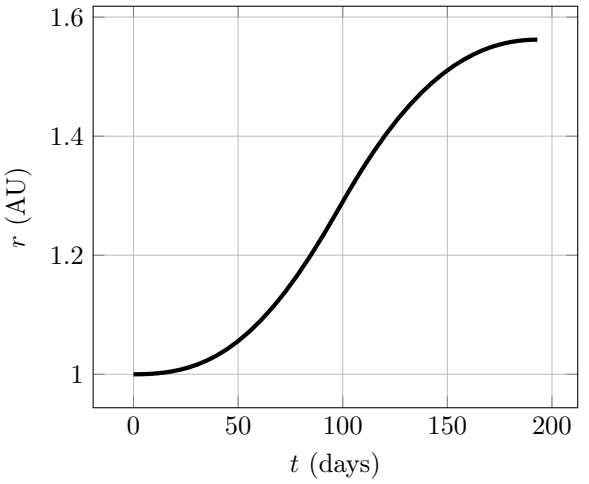


Figure 4.7: Orbit radius as a function of time.

The thrust angle, intuitively, starts by pointing in an outward direction to increase the radius and terminates pointing inward to circularize. This is shown in Figure 4.8.

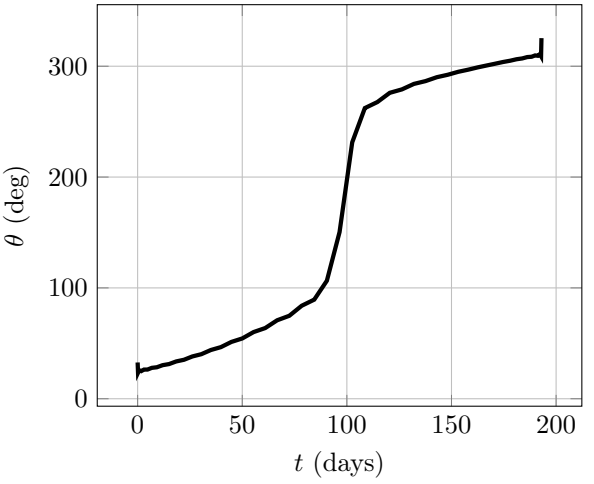


Figure 4.8: Thrust angle as a function of time using indirect shooting.

4.9 Chapter Problems

Problem 4.1. Consider the generic discrete optimal control problem in Eq. (4.1). All other things the same, include in the problem a control constraint of the form

$$\forall k = 0, \dots, N-1, \quad u_k \in \Omega = \{\omega \in \mathbb{R}^m : g(\omega) \leq 0\},$$

where $g : \mathbb{R}^m \to \mathbb{R}^p$ is continuously differentiable. Derive the optimality conditions for this problem. The optimality conditions include the transversality condition, stationarity condition, and costate equation.

Problem 4.2. Consider the discrete scalar problem with objective

$$J = \frac{1}{2}(x_N - x^*)^2 + \frac{r}{2}\sum_{k=0}^{N-1} u_k^2$$

and linear dynamics

$$x_{k+1} = ax_k + bu_k, \quad k = 0, \dots, N-1.$$

The system starts at a fixed point x_0. It does not have any terminal boundary conditions. The objective is, however, trying to push the final state close to x^*. Solve for the optimal control u_k as a function of the problem data (a, b, x_0, r, x^*, N). What happens as $r \to \infty$?

Problem 4.3. Consider the following linear quadratic problem with mixed cost.

$$\begin{aligned} \text{minimize} \quad & \tfrac{1}{2}x_N^\top S_N x_N + \frac{1}{2}\sum_{k=0}^{N-1} \begin{bmatrix} x_k \\ u_k \end{bmatrix}^\top \begin{bmatrix} Q_k & T_k \\ T_k^\top & R_k \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} \\ \text{subject to} \quad & x_{k+1} = A_k x_k + B_k u_k, \quad x_0 \text{ given} \end{aligned}$$

Each of the matrices Q_k, R_k, S_N is symmetric and positive definite. This is a matrix-vector problem with $x_k \in \mathbb{R}^n$ and $u_k \in \mathbb{R}^m$. Solve the problem by developing a feedback control law of the form $u_k = -K_k x_k$.

Problem 4.4. A chaser spacecraft is in close proximity to a target spacecraft in circular orbit with mean motion $\omega = 4/3600$ rad/s. Use LQT as described in Section 4.5 to circumnavigate the target on the following reference trajectory. Position is given in meters and time is given in seconds.

```
W  = 2*pi/300;
rx = 100*cos(W*t).*cos(W*t);
ry = 100*cos(W*t).*sin(W*t);
rz = 100*sin(W*t);
```

Problem 4.5. Recall Problems 3.10 and 3.11. A lunar lander is in close proximity to the lunar surface with the following initial position (m) and velocity (m/s).

```
r0 = [1000; 50; 1000]; % m
v0 = [-25; 0; 0]; % m/s
```

The goal is to land at the origin of the coordinate system with downward velocity of 1 m/s. The flight time is two minutes.

a) Use direct transcription to design an optimal trajectory with objective $J = \sum ||u_k||^2$ and control constraints $||u_k|| \leq 3.5$ m/s^2.
b) Use direct transcription to design an optimal trajectory with objective $J = \sum ||u_k||$ and control constraints $||u_k|| \leq 3.5$ m/s^2.

Compare and contrast the solutions.

Problem 4.6. Building upon the previous problem, solve the free final time versions of a) and b) by conducting a line search for the N that minimizes the objectives. Compare and contrast the solutions.

Problem 4.7. Building upon the previous problem, use discrete LQR to follow the nominal lunar descent trajectories. Incorporate into the simulation navigation errors and disturbances. Recall that trajectory following using LQR was implemented in Section 4.6.

Problem 4.8. Use direct transcription to solve the maximal orbit raise problem described in Section 4.8 with a flight time of 365 days.

Chapter 5

Optimal Control

Chapter Learning Objectives

1. Understand optimality conditions for optimal control problems.
2. Develop optimal feedback laws for linear quadratic and minimum time problems.
3. Develop and implement a constant tangent law for ascent.
4. Implement indirect shooting, direct shooting, and global control parameterization to solve an optimal orbit transfer problem.

The performance of continuous-time systems can be optimized using optimal control. An optimal control problem (typically) has as objective an integral and as constraints differential equations, control limits, and boundary conditions. The optimization variables are control functions. Again, the mathematical presentation herein is rigorous, but the theory of optimal control is beyond the scope of this book.[15,16]

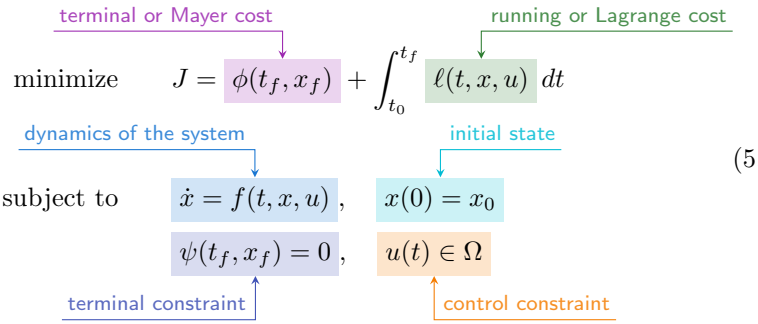
After stating the general optimal control problem and associated optimality conditions, several small problems are solved to cement a solution procedure and get familiarized with the "almost everywhere" concept. Applied to a linear quadratic control problem, the solution procedure follows very closely that from discrete optimal control. A minimum time problem is analyzed and solved in state feedback form. These linear quadratic and minimum time problems set up for descent guidance presented in Chapter 6.

A minimum time pursuit/ascent problem is solved leading to the constant tangent law. This sets up the linear tangent law for ascent guidance in Chapter 7. The optimal orbit transfer problem is revisited and solved using indirect shooting, direct shooting, and global control parameterization. The indirect shooting method is simply a technique for satisfying the optimality conditions. The method is sensitive to the initial guess, and a continuation procedure is introduced to facilitate making a good initial guess. Direct shooting and global control parameterization, on the other hand, are approximate techniques that do not use the optimality conditions.

[15] Berkovitz and Medhin, *Nonlinear Optimal Control Theory*, 2012.
[16] Liberzon, *Calculus of Variations and Optimal Control Theory*, 2012.

5.1 A General Optimal Control Problem

A standard optimal control problem is



(5.1)

As in discrete optimal control: ϕ is the terminal or Mayer cost, ℓ is the running or Lagrange cost, f is the system dynamics with initial condition x_0, and ψ is the terminal constraint. The key differences are that in optimal control the running cost is measured by an integral (not a sum) and the dynamics are specified as ordinary differential equations (not discrete-time equations). The set Ω is the control constraint. The control function must take values in the set. For example, if the magnitude of the control is bounded by ρ then $\Omega = \{u \in \mathbb{R}^m : ||u|| \leq \rho\}$. Control constraints are important in optimal control, especially minimum time problems, because the control may become unbounded otherwise.

The optimality conditions are conveniently stated in terms of the Hamiltonian and endpoint functions.

$$H(t, x, u, \lambda_0, \lambda) = \lambda_0 \ell(t, x, u) + \lambda^\top f(t, x, u) \quad \text{(Hamiltonian)} \tag{5.2}$$

$$G(t_f, x_f, \lambda_0, \nu) = \lambda_0 \phi(t_f, x_f) + \nu^\top \psi(t_f, x_f) \quad \text{(endpoint function)} \tag{5.3}$$

If x and u are minimizing functions, then there exist a scalar $\lambda_0 \in \{0, 1\}$, an absolutely continuous function λ, and constant ν such that the following hold.

$$\dot{\lambda} = -\frac{\partial H}{\partial x} \quad \text{(costate equation)} \tag{5.4}$$

$$\dot{H} = \frac{\partial H}{\partial t} \quad \text{(Hamiltonian equation)} \tag{5.5}$$

$$\lambda_f = \frac{\partial G}{\partial x_f} \quad \text{(transversality condition)} \tag{5.6}$$

$$H_f = -\frac{\partial G}{\partial t_f} \quad \text{(transversality condition)} \tag{5.7}$$

$$u \overset{\text{ae}}{\in} \underset{\omega \in \Omega}{\operatorname{argmin}}\ H(t, x, \omega, \lambda_0, \lambda) \quad \text{(pointwise minimum condition)} \tag{5.8}$$

$$(\lambda_0, \lambda) \neq 0 \quad \text{(non-triviality condition)} \tag{5.9}$$

Of course, the state equation, initial condition, and terminal constraint must also be satisfied. The pointwise minimum condition is the analog of the stationarity condition. These necessary conditions are the optimality conditions to be satisfied when searching for candidate solutions. This particular statement of optimality conditions assumes that the gradient of ψ has linearly independent columns.

In other texts, the abnormal multiplier λ_0 is required to be in $\{0, -1\}$ and the pointwise condition is changed to a maximum. Either approach is fine and none of the other conditions changes. Unlike in the discrete optimal control setting, the authors are unaware of any a priori conditions to ensure $\lambda_0 \neq 0$. The strategy is to explore both abnormal and normal cases.

It is preferable to use a letter to denote a function, e.g., u is a function, and to denote the value of the function at time t as $u(t)$. Doing so consistently results in long, difficult-to-read equations. For this reason alone, the (t) is typically dropped. To be specific, it would be better to write the non-triviality condition as $\forall t \in [0, t_f]$, $(\lambda_0, \lambda(t)) \neq 0$. Similarly, in the pointwise minimum condition, the quantity on the left side of the inclusion is the value of the control at time t and the quantity on the right side of the argmin is the value of the Hamiltonian with all arguments evaluated at time t.

The costate and Hamiltonian differential equations indicate that λ and H are absolutely continuous functions with respect to time. Thus, these quantities do not exhibit discontinuities in time. The transversality conditions specify the respective boundary conditions.

Our goal in solving the optimal control problem is to find the optimal control function u. Amazingly, the pointwise minimum condition says that this can be done pointwise in time. The notation a.e. means almost everywhere, which is a precise measure-theoretic term. Most importantly here, two functions u_1 and u_2 that are not equal but equal almost everywhere generate the same state equation and objective function because they appear only in integration.

Within this book, readers need know only six facts from measure theory.

1. A condition holds almost everywhere if the set on which it fails has zero measure.
2. A countable set has zero measure.
3. An analytic function has at most countable zeros or is identically zero.
4. A point $p \in \mathbb{R}$ is isolated in I if there exists an $\epsilon > 0$ such that $(p-\epsilon, p+\epsilon) \cap I = \{p\}$.
5. A set of isolated points is countable and has zero measure.
6. There are sets of positive measure that do not contain an interval.

Example 5.1. Consider a simple car on a straight track trying to reach the finish line as quickly as possible.

$$\begin{aligned} \text{minimize} \quad & t_f \\ \text{subject to} \quad & \dot{x} = u, \quad x_0 = 0, \quad x_f = 1 \\ & -1 \leq u \leq 1 \end{aligned}$$

The solution procedure is to first form the Hamiltonian and endpoint functions.

$$\begin{aligned} H &= \lambda u \\ G &= \lambda_0 t_f + \nu(x_f - 1) \end{aligned}$$

The costate, Hamiltonian, and transversality conditions generate the following equations.

$$\begin{aligned} \dot{\lambda} &= 0, \qquad \lambda_f = \nu \\ \dot{H} &= 0, \qquad H_f = -\lambda_0 = \lambda_f \, u_f \end{aligned}$$

It is immediately deduced that λ and H are constants, from which it follows that u is constant. The pointwise minimum condition yields

$$u \overset{\text{ae}}{\in} \underset{-1 \le \omega \le 1}{\operatorname{argmin}} \; \lambda \omega \qquad \Longrightarrow \qquad u \overset{\text{ae}}{=} \begin{cases} -1, & \lambda > 0 \\ +1, & \lambda < 0 \\ \text{singular}, & \lambda = 0 \end{cases}.$$

If $\lambda = 0$, then $\lambda_0 = 0$ violating non-triviality. Thus, "singular" solutions cannot occur. The optimal control is either always -1 or always $+1$. If $u = -1$, then $x = -t$. Since $t \ge 0$, the terminal constraint $x_f = 1$ cannot be satisfied. If $u = +1$, then $x = t$. The terminal constraint is satisfied when $t_f = 1$. The candidate optimal control is $u(t) = 1$ on the time domain $[0, 1]$. ★

Example 5.2. Consider the related problem of moving the car from start to finish with a quadratic control objective and no control constraint. The final time is fixed at $\bar{t}_f$.

$$\begin{aligned} \text{minimize} \qquad & \frac{1}{2} \int_0^{t_f} u^2 \, dt \\ \text{subject to} \qquad & \dot{x} = u, \quad x_0 = 0, \quad x_f = 1, \quad t_f = \bar{t}_f \end{aligned}$$

The solution procedure is to first form the Hamiltonian and endpoint functions.

$$\begin{aligned} H &= \frac{1}{2} \lambda_0 u^2 + \lambda u \\ G &= \nu_1 \, (x_f - 1) + \nu_2 \, (t_f - \bar{t}_f) \end{aligned}$$

The costate, Hamiltonian, and transversality conditions generate the following equations.

$$\begin{aligned} \dot{\lambda} &= 0, \qquad \lambda_f = \nu_1 \\ \dot{H} &= 0, \qquad H_f = -\nu_2 = \frac{1}{2} \lambda_0 u_f^2 + \lambda_f u_f \end{aligned}$$

It is immediately deduced that λ and H are constants, from which it follows that u is constant. The pointwise minimum condition yields

$$u \overset{\text{ae}}{\in} \underset{\omega}{\operatorname{argmin}} \; \frac{1}{2} \lambda_0 \omega^2 + \lambda \omega \qquad \Longrightarrow \qquad \lambda_0 u + \lambda \overset{\text{ae}}{=} 0.$$

Having $\lambda_0 = 0$ implies $\lambda = 0$, which violates non-triviality. With $\lambda_0 = 1$, $u = -\lambda$. A constant control that drives the system state from 0 to 1 in time $\bar{t}_f$ is $u = 1/\bar{t}_f$. All the optimality conditions are satisfied, and this is a candidate optimal control.

The objective value is

$$\frac{1}{2} \int_0^{\bar{t}_f} \frac{1}{\bar{t}_f^2} \, dt = \frac{1}{2} \frac{1}{\bar{t}_f}.$$

By extending the final time, the objective can be driven to zero in the limit. However, the control becomes zero in the limit. Zero control does not drive the state from 0 to 1. Hence, one should expect that the free final time version of the problem lacks a solution.

ℹ The infimum of the free final time objective is zero but a minimum does not exist. This is the equivalent of trying to minimize e^{-x}. Although motivated by a real problem, the mathematical problem is ill-posed.

★

Example 5.3. Consider now the problem with an absolute value control objective.

$$\begin{aligned} \text{minimize} \quad & \int_0^{t_f} |u| \, dt \\ \text{subject to} \quad & \dot{x} = u, \quad x_0 = 0, \quad x_f = 1, \quad t_f = \bar{t}_f > 1 \\ & -1 \leq u \leq 1 \end{aligned}$$

The solution procedure is to first form the Hamiltonian and endpoint functions.

$$\begin{aligned} H &= \lambda_0 \, |u| + \lambda u \\ G &= \nu_1 \, (x_f - 1) + \nu_2 \, (t_f - \bar{t}_f) \end{aligned}$$

The costate, Hamiltonian, and transversality conditions generate the following equations.

$$\begin{aligned} \dot{\lambda} &= 0, \qquad \lambda_f = \nu_1 \\ \dot{H} &= 0, \qquad H_f = -\nu_2 \end{aligned}$$

It is immediately deduced that λ and H are constants. Constancy of u cannot be deduced in this problem. If $\lambda_0 = 0$, then the pointwise minimum condition yields

$$u \overset{\text{ae}}{\in} \underset{-1 \leq \omega \leq 1}{\operatorname{argmin}} \; \lambda \omega \quad \Longrightarrow \quad u \overset{\text{ae}}{=} \begin{cases} -1, & \lambda > 0 \\ +1, & \lambda < 0 \\ \text{singular}, & \lambda = 0 \end{cases} .$$

The abnormal singular case cannot occur because it violates non-triviality. Also, $u \overset{\text{ae}}{=} -1$ and $u \overset{\text{ae}}{=} +1$ do not steer to the final conditions. Thus, $\lambda_0 = 1$. To satisfy the pointwise minimum condition in the normal case, one needs to solve the following problem almost everywhere.

$$\begin{aligned} \text{minimize} \quad & |u| + \lambda u \\ \text{subject to} \quad & -1 \leq u \leq 1 \end{aligned}$$

The absolute value prevents application of the theories presented in Chapter 3. Breaking the problem into subcases resolves the issue.

- If $\lambda > 1$, then $u \overset{\text{ae}}{=} -1$. This control does not steer to the final conditions.
- If $-1 < \lambda < 1$, then $u \overset{\text{ae}}{=} 0$. This control does not steer to the final conditions.
- If $\lambda < -1$, then $u \overset{\text{ae}}{=} 1$. This control does not steer to the final conditions.
- If $\lambda = \pm 1$, the minimizing control is not determined by the condition.

This is the singular case meaning the pointwise minimum condition does not uniquely specify the control. In this problem, the normal singular case is the only feasible option. It must be that $\lambda = -1$.

Any feasible control taking values in the interval $[0, 1]$ is a candidate optimal control. Two options are:

$$u = \frac{1}{\bar{t}_f}, \qquad \longrightarrow \quad J = 1,$$
$$u = \begin{cases} 1 & t \in [0, 1) \\ 0 & t \in [1, \bar{t}_f] \end{cases} \qquad \longrightarrow \quad J = 1.$$

There are many more candidates. Like regular optimization problems, optimal control problems may have no solutions, one solution, or infinitely many solutions. In the event that the optimality conditions generate multiple candidates, the optimal solution can be determined by computing the objective of each candidate and picking the best one(s).

ℹ Optimal controls need not be differentiable or even continuous. It is not uncommon in optimal spacecraft guidance for the control to be bang-bang (switching between extreme limits) or bang-off-bang (switching between extreme limits and zero). A general theory of optimal control permits a large class of control functions called measurable functions or even generalized measure-driven functions.

★

5.2 Scalar Linear Quadratic Control

The continuous-time scalar linear quadratic control problem is now investigated.

$$\begin{aligned} &\text{minimize} && \frac{1}{2}\int_0^{t_f} u^2\, dt \\ &\text{subject to} && \dot{x} = ax + bu, \quad x_0 = \bar{x}_0, \quad x_f = \bar{x}_f, \quad t_f = \bar{t}_f \end{aligned} \tag{5.10}$$

The barred quantities are given constants. The solution procedure is to first form the Hamiltonian and endpoint functions.

$$H = \frac{1}{2}\lambda_0 u^2 + \lambda\,(ax + bu) \tag{5.11}$$
$$G = \nu_1\,(x_f - \bar{x}_f) + \nu_2\,(t_f - \bar{t}_f) \tag{5.12}$$

The costate, Hamiltonian, and transversality conditions generate the following equations.

$$\dot{\lambda} = -a\lambda, \qquad \lambda_f = \nu_1 \tag{5.13}$$
$$\dot{H} = 0, \qquad H_f = -\nu_2 \tag{5.14}$$

It is immediately deduced that H is constant. Also, from the homogeneity of the costate equation, λ being zero somewhere implies it is zero everywhere. The pointwise minimum condition yields

$$u \overset{\text{ae}}{\in} \underset{\omega}{\operatorname{argmin}}\ \frac{1}{2}\lambda_0\omega^2 + b\lambda\omega \quad \Longrightarrow \quad \lambda_0 u + b\lambda \overset{\text{ae}}{=} 0. \tag{5.15}$$

Having $\lambda_0 = 0$ implies $\lambda = 0$, which violates non-triviality. With $\lambda_0 = 1$, the optimal control is $u \overset{\text{ae}}{=} -b\lambda$. Though not required to do so, we choose $u = -b\lambda$. Substituting into the state equation gives

$$\dot{x} = ax - b^2\lambda. \tag{5.16}$$

Since the costate is homogenous, its solution is given by

$$\lambda = e^{a(t_f - t)}\lambda_f, \tag{5.17}$$

making the state equation

$$\dot{x} = ax - b^2 e^{a(t_f - t)}\lambda_f. \tag{5.18}$$

The solution to this equation is

$$x = e^{a(t-0)}x_0 - \int_0^t e^{a(t-\tau)}b^2 e^{a(t_f-\tau)}\lambda_f \, dt. \tag{5.19}$$

Evaluating at the final time gives

$$x_f = e^{at_f}x_0 - \int_0^{t_f} e^{a(t_f-\tau)}b^2 e^{a(t_f-\tau)}\lambda_f \, d\tau \tag{5.20a}$$

$$= e^{at_f}x_0 - \Lambda\lambda_f. \tag{5.20b}$$

Provided $\Lambda \neq 0$, which is again related to controllability of the system, we can now solve for λ_f as

$$\lambda_f = \frac{1}{\Lambda}\left(e^{at_f}x_0 - x_f\right). \tag{5.21}$$

Substituting this back into the costate equation gives

$$\lambda = \frac{1}{\Lambda}e^{a(t_f-t)}\left(e^{at_f}x_0 - x_f\right). \tag{5.22}$$

The candidate optimal control is

$$u = -\frac{b}{\Lambda}e^{a(t_f-t)}\left(e^{at_f}x_0 - x_f\right). \tag{5.23}$$

Using an existence theorem, it can be shown that an optimal solution exists making this candidate the optimal solution. This control drives the (a, b) system from x_0 to x_f in t_f time and minimizes the quadratic control objective. This optimal control is open-loop because it depends on the initial state rather than the current state.

The solution process for this continuous-time problem follows very closely the process for the discrete-time problem. After writing the optimality conditions, solve for the optimal control in terms of the costate, back propagate the costate, forward propagate the state, and satisfy the boundary conditions. A sufficient condition for the optimality conditions to be solvable is that the system is controllable, which guarantees that Λ is non-zero. Analysis of all the linear quadratic variants LQC, LQR, and LQT is similar. Analysis of each is left as a problem for the reader.

MATLAB Implementation

For an implementation in MATLAB, define the system `a` and `b`, the boundary conditions `x0` and `xf`, and the final time `tf`. The problem is to drive the state from 1 to 0 in 1 second.

```
% Data
a  = 1.1;
b  = 1;
x0 = 1;
xf = 0;
tf = 1;
```

Compute Λ, which can then be used to compute the optimal control.

```
L = b^2/(2*a)*( exp(2*a*tf)-1 )
```

The function to be integrated is then coded. Note the extra parameters being passed into the integration.

```
function xdot = ode(t,x,a,b,x0,xf,tf,L)
u = -b/L*exp( a*(tf-t) )* (exp( a*(tf) )*x0 - xf);
xdot = a*x+b*u;
end
```

The system can be simulated and the result plotted.

```
% Simulation
[t,x] = ode45(@ode,[0,tf],x0,[],a,b,x0,xf,tf,L);
figure, plot(t,x), grid on
xlabel('t'), ylabel('x')
```

Figure 5.1 shows the state evolving in time from 1 to 0 as required.

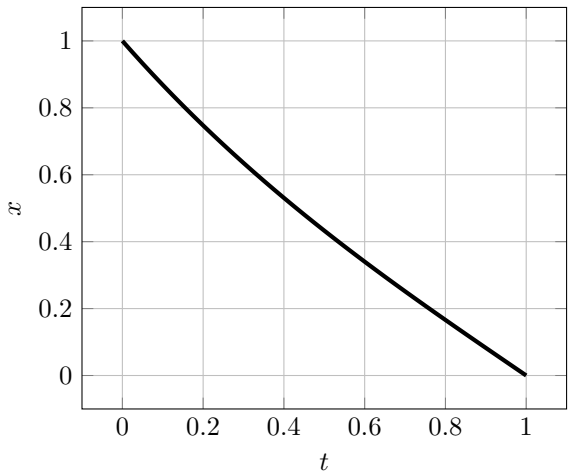


Figure 5.1: State trajectory for scalar linear quadratic optimal control.

5.3 Optimal Thrust Angle Control

Vehicle p is pursuing vehicle e, which is evading by moving to the right with constant velocity v_e. This is illustrated in Figure 5.2

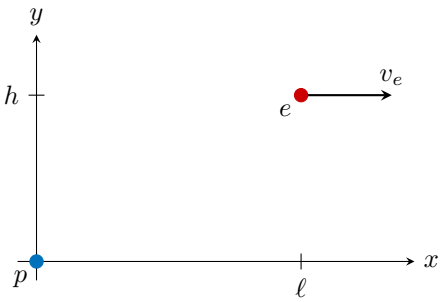


Figure 5.2: Illustration of vehicle p pursuing vehicle e.

The pursuer starts at rest at the origin. The evader starts at the point (ℓ, h). The dynamics of the pursuer are

$$\dot{x} = u, \quad \dot{u} = \tau \cos\theta, \quad x(0) = u(0) = 0, \qquad \text{(horizontal motion)} \tag{5.24a}$$

$$\dot{y} = v, \quad \dot{v} = \tau \sin\theta, \quad y(0) = v(0) = 0. \qquad \text{(vertical motion)} \tag{5.24b}$$

The known constant thrust acceleration magnitude is τ. The control variable is the thrust angle θ. The pursuer wants to intercept the evader as quickly as possible.

$$\begin{aligned} &\text{minimize} && t_f \\ &\text{subject to} && (5.24a) - (5.24b), \\ & && y_f = h, \quad x_f = \ell + v_e t_f \end{aligned} \tag{5.25}$$

The solution procedure is to first form the Hamiltonian and endpoint functions.

$$H = \lambda_x u + \lambda_y v + \lambda_u \tau \cos\theta + \lambda_v \tau \sin\theta \tag{5.26}$$

$$G = \lambda_0 t_f + \nu_x (x_f - \ell - v_e t_f) + \nu_y (y_f - h) \tag{5.27}$$

The costate and Hamiltonian differential equations are the following.

$$\dot{\lambda}_x = 0 \quad \Longrightarrow \quad \lambda_x \text{ constant} \tag{5.28}$$

$$\dot{\lambda}_y = 0 \quad \Longrightarrow \quad \lambda_y \text{ constant} \tag{5.29}$$

$$\dot{\lambda}_u = -\lambda_x \quad \Longrightarrow \quad \lambda_u = -\lambda_x (t - t_f) + \lambda_{uf} \tag{5.30}$$

$$\dot{\lambda}_v = -\lambda_y \quad \Longrightarrow \quad \lambda_v = -\lambda_y (t - t_f) + \lambda_{vf} \tag{5.31}$$

$$\dot{H} = 0 \quad \Longrightarrow \quad H \text{ constant} \tag{5.32}$$

The transversality conditions specify terminal boundary conditions on the costates and Hamiltonian.

$$\lambda_{xf} = \nu_x, \quad \lambda_{uf} = 0 \tag{5.33}$$

$$\lambda_{yf} = \nu_y, \quad \lambda_{vf} = 0 \tag{5.34}$$

$$H_f = -\lambda_0 + \nu_x v_e = \nu_x u_f + \nu_y v_f \tag{5.35}$$

Using the costate and transversality conditions together gives

$$\lambda_u = -\nu_x (t - t_f), \tag{5.36}$$
$$\lambda_v = -\nu_y (t - t_f). \tag{5.37}$$

The pointwise minimum condition yields

$$\theta \overset{\text{ae}}{\in} \underset{\omega}{\operatorname{argmin}}\ \lambda_u \tau \cos\omega + \lambda_v \tau \sin\omega \quad \Longrightarrow \quad -\lambda_u \sin\theta + \lambda_v \cos\theta \overset{\text{ae}}{=} 0, \tag{5.38}$$

which determines the thrust angle θ except when λ_u and λ_v are both zero. This singular case is negligible if it occurs on a set of zero measure. Suppose, on the other hand, it occurs on a set of positive measure, i.e., λ_u and λ_v are both zero on a set of positive measure. Because they are linear (analytic) functions of time, they are zero everywhere. This implies that $\lambda_x = \nu_x$ and $\lambda_y = \nu_y$ are also zero. The Hamiltonian is zero and the H_f condition implies $\lambda_0 = 0$, which violates non-triviality. The singular case cannot occur on sets of positive measure. It is deduced that

$$\tan\theta \overset{\text{ae}}{=} \frac{\lambda_v}{\lambda_u}. \tag{5.39}$$

Though not required, the $\overset{\text{ae}}{=}$ is replaced with $=$. Substituting in the costate expressions gives

$$\tan\theta = \frac{-\nu_y (t - t_f)}{-\nu_x (t - t_f)} = \frac{\nu_y}{\nu_x}. \tag{5.40}$$

That is, the optimal thrust angle is constant. The state equations can be integrated.

$$u = \tau t \cos\theta, \qquad x = \frac{1}{2}\tau t^2 \cos\theta \tag{5.41}$$
$$v = \tau t \sin\theta, \qquad y = \frac{1}{2}\tau t^2 \sin\theta \tag{5.42}$$

At the final time, the terminal constraints yield

$$\left.\begin{aligned} \frac{1}{2}\tau t_f^2 \cos\theta &= \ell + v_e t_f \\ \frac{1}{2}\tau t_f^2 \sin\theta &= h \end{aligned}\right\} \quad \Longrightarrow \quad \tan\theta = \frac{h}{\ell + v_e t_f}. \tag{5.43}$$

The only thing remaining is to find the optimal final time. One way to do this is to square both sides in the above equations and add.

$$\frac{1}{4}\tau^2 t_f^4 \left(\sin^2\theta + \cos^2\theta\right) = h^2 + (\ell + v_e t_f)^2 \tag{5.44}$$
$$\Longrightarrow \quad \frac{1}{4}\tau^2 t_f^4 = h^2 + \ell^2 + 2\ell v_e t_f + v_e^2 t_f^2\ . \tag{5.45}$$

Solving this quartic equation gives four possible values of t_f. Choose the least real value from the four. This is the minimum intercept time.

MATLAB Implementation

For MATLAB implementation, specify the initial altitude `h` and range `L` of the evader, its speed `ve`, and the thrust acceleration of the pursuer `a`.

```
% Data
h  = 50;    % m
L  = 100;   % m
ve = 25;    % m/s
a  = 10;    % m/s2
```

Compute the final time by solving the quartic polynomial and selecting the real positive root.

```
% Solve for the final time
tf = roots([-1/4*a^2,0,ve^2,2*L*ve,h^2+L^2]);
tf = tf(1);
```

Compute the optimal thrust angle.

```
% Solve for the thrust angle
theta = atan( h/(L+ve*tf) );
```

Lastly, verify that the final positions of the pursuer and evader are equal.

```
% Final position of the evader
xe = L+ve*tf;
ye = h;

% Final position of the pursuer
xp = 1/2*a*tf^2*cos(theta);
yp = 1/2*a*tf^2*sin(theta);

% Position error
error = norm([xe-xp,ye-yp])
```

A simulation involving integration could also be created. For this problem, however, the differential equations can be analytically integrated to compute the final positions.

5.4 Minimum Time Control

Drive the double integrator system to the origin in minimum time with bounded control. All quantities are scalars.

$$\begin{aligned} \text{minimize} \quad & \int_0^{t_f} 1 \, dt \\ \text{subject to} \quad & \dot{x}_1 = x_2, \quad x_1(0) = x_{10}, \quad x_{1f} = 0 \\ & \dot{x}_2 = u, \quad x_2(0) = x_{20}, \quad x_{2f} = 0 \\ & |u| \leq 1 \end{aligned} \tag{5.46}$$

The solution procedure is to first form the Hamiltonian and endpoint functions.

$$H = \lambda_0 + \lambda_1 x_2 + \lambda_2 u \tag{5.47}$$
$$G = \nu_1 (x_{1f} - 0) + \nu_2 (x_{2f} - 0) \tag{5.48}$$

The costate, Hamiltonian, and transversality conditions generate the following equations.

$$\dot{\lambda}_1 = 0, \qquad \lambda_{1f} = \nu_1 \tag{5.49}$$
$$\dot{\lambda}_2 = -\lambda_1, \qquad \lambda_{2f} = \nu_2 \tag{5.50}$$
$$\dot{H} = 0, \qquad H_f = 0 \tag{5.51}$$

It is immediately deduced that H is constant and zero. The costate λ_1 is constant. The costate λ_2 is constant or varying linearly in time. The pointwise minimum condition yields

$$u \overset{\text{ae}}{\in} \underset{|\omega| \leq 1}{\operatorname{argmin}} \ \lambda_2 \omega \quad \Longrightarrow \quad u \overset{\text{ae}}{=} \begin{cases} -1 \ , & \lambda_2 > 0 \\ +1 \ , & \lambda_2 < 0 \ . \\ \text{singular}, & \lambda_2 = 0 \end{cases} \tag{5.52}$$

The singular case is negligible if it occurs on a set of zero measure. Suppose, on the other hand, it occurs on a set of positive measure, i.e., λ_2 is zero on a set of positive measure. Because it is a linear (analytic) function of time, it is zero everywhere. Having $\lambda_2 = 0$ everywhere implies $\lambda_1 = 0$ everywhere. The Hamiltonian being zero implies that $\lambda_0 = 0$, which violates non-triviality. The singular case cannot occur on sets of positive measure.

As a result, the control can be chosen to take only values of ± 1. Because λ_2 is a linear function of time, it can switch signs at most one time. Denote such a switch time as t_1. There are four possible control solutions.

$$u = \begin{cases} +1 & \forall t \in [t_0, t_f] \\ -1 & \forall t \in [t_0, t_f] \\ +1 & \forall t \in [t_0, t_1), \quad -1 \quad \forall t \in [t_1, t_f] \\ -1 & \forall t \in [t_0, t_1), \quad +1 \quad \forall t \in [t_1, t_f] \end{cases} \tag{5.53}$$

Integrating the state equations with $u = \pm 1$ gives

$$x_2 = \pm t + a, \tag{5.54}$$
$$x_1 = \pm \frac{1}{2} t^2 + at + b, \tag{5.55}$$

where $a = x_{20}$ and $b = x_{10}$. Eliminating t gives

$$x_1 = +\frac{1}{2} x_2^2 + c \quad \text{for} \quad u = +1, \tag{5.56}$$
$$x_1 = -\frac{1}{2} x_2^2 + d \quad \text{for} \quad u = -1. \tag{5.57}$$

The constants c and d are functions of a and b. One can then plot the parabolas for various values of c and d as seen in Figure 5.3.

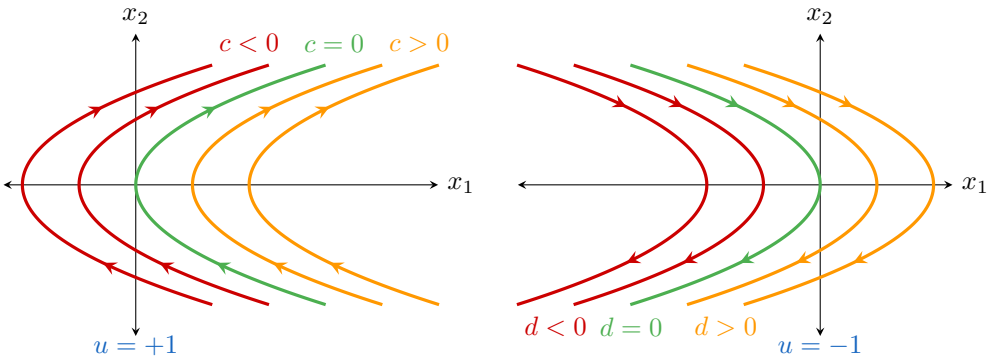


Figure 5.3: Plot of the parabolas for various values of c and d.

These graphs provide a map of how to move throughout the state space. The parabolas are one-way roads upon which the system can travel. If starting in the first quadrant, applying $u = +1$ moves the state farther from the origin. Applying $u = -1$ moves the state into the fourth quadrant. As soon as the green ($u = +1$) curve is hit in the fourth quadrant, switching to $u = +1$ drives the state to the origin. This motivates the following switching curve $x_1 = -\frac{1}{2}x_2\,|x_2|$, which is shown in Figure 5.4.

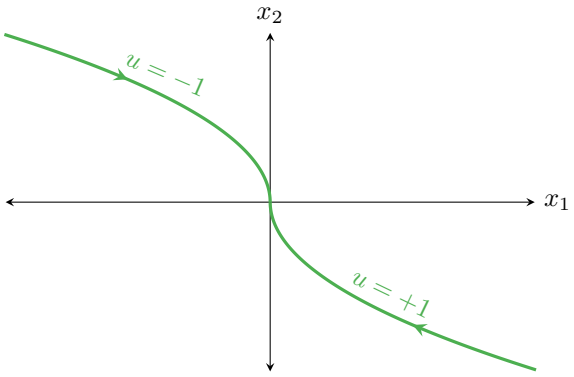


Figure 5.4: Plot of the switching curve.

If the current state is above the switching curve, apply $u = -1$. If the current state is below the switching curve, apply $u = +1$. If the current state is on the switching curve and x_2 is positive, apply $u = -1$. If the current state is on the switching curve and x_2 is negative, apply $u = +1$.

This optimal strategy is in closed-loop form because it depends on the current state. It does, however, require perfect knowledge of the state. In the presence of any error in state knowledge or integration, the control will chatter as it switches between ± 1. Care must be taken to prevent this in practice.

5.5 Maximal Orbit Raise

Consider a spacecraft in a circular orbit and the problem of using continuous thrust to transfer to the largest possible circular orbit in a specified time. The thrust force is a known constant. The control variable is the thrust angle θ. The optimal control problem is the following.

$$
\begin{aligned}
&\text{maximize} && r_f, && t_f = \bar{t}_f \\
&&&&& \text{radial distance} \\
&\text{subject to} && \dot{r} = u, && r\,(0) = r_0 \\
&&&&& \text{thrust force} \qquad \text{radial velocity} \\
&&& \dot{u} = \frac{v^2}{r} - \frac{\mu}{r^2} + \frac{T \sin\theta}{m_0 - \dot{m}t}, && u\,(0) = 0, \qquad u_f = 0 \\
&&&&& \text{thrust angle} \qquad \text{tangential velocity} \\
&&& \dot{v} = -\frac{u\,v}{r} + \frac{T\cos\theta}{m_0 - \dot{m}\,t}, && v\,(0) = \sqrt{\frac{\mu}{r_0}}, \quad v_f = \sqrt{\frac{\mu}{r_f}} \\
&&&&& \text{initial mass} \qquad \text{fuel burn rate}
\end{aligned}
\tag{5.58}
$$

The solution procedure is to first form the Hamiltonian and endpoint functions.

$$H = \lambda_r u + \lambda_u \left(\frac{v^2}{r} - \frac{\mu}{r^2} + \frac{T\sin\theta}{m_0 - \dot{m}t} \right) + \lambda_v \left(-\frac{uv}{r} + \frac{T\cos\theta}{m_0 - \dot{m}t} \right) \tag{5.59}$$

$$G = \lambda_0 r_f + \nu_1 u_f + \nu_2 \left(v_f - \sqrt{\frac{\mu}{r_f}} \right) + \nu_3 (t_f - \bar{t}_f) \tag{5.60}$$

The costate equations are

$$\dot{\lambda}_r = -\frac{\partial H}{\partial r} = -\lambda_u \left(-\frac{v^2}{r} + \frac{2\mu}{r^3} \right) - \lambda_v \left(\frac{uv}{r^2} \right), \tag{5.61}$$

$$\dot{\lambda}_u = -\frac{\partial H}{\partial u} = -\lambda_r + \lambda_v \frac{v}{r}, \tag{5.62}$$

$$\dot{\lambda}_v = -\frac{\partial H}{\partial v} = -\lambda_u \frac{2v}{r} + \lambda_v \frac{u}{r}. \tag{5.63}$$

The transversality conditions are

$$\lambda_{rf} = \frac{\partial G}{\partial r_f} = \lambda_0 + \frac{\nu_2 \sqrt{\mu}}{2 r_f^{3/2}}, \tag{5.64}$$

$$\lambda_{uf} = \frac{\partial G}{\partial u_f} = \nu_1, \tag{5.65}$$

$$\lambda_{vf} = \frac{\partial G}{\partial v_f} = \nu_2. \tag{5.66}$$

The pointwise maximum (because the problem is to maximize) condition yields

$$\theta \overset{\text{ae}}{\in} \underset{\omega}{\operatorname{argmax}}\ \lambda_u \sin\omega + \lambda_v \cos\omega \quad \Longrightarrow \quad \lambda_u \cos\theta - \lambda_v \sin\theta \overset{\text{ae}}{=} 0, \tag{5.67}$$

which determines the thrust angle θ except when λ_u and λ_v are both zero. This singular case is negligible if it occurs on a set of zero measure. Suppose, on the other hand, it occurs on a set of positive measure, i.e., λ_u and λ_v are both zero on a set of positive measure. Note that λ_u and λ_v satisfy differential equations that depend on the states and the states depend on the control. Hence, they may not be analytic functions of time. For non-analytic functions, being zero on a set of positive measure does not imply being zero everywhere. A deeper analysis is required.

Let s be a point in the supposed set such that $\lambda_u(s) = \lambda_v(s) = 0$. Observe that the costate equations are homogeneous. If $\lambda_r(s) = 0$, then all three functions are zero everywhere and the λ_{rf} condition then requires $\lambda_0 = 0$, violating non-triviality. Hence, $\lambda_r(s) \neq 0$. A formula for λ_u is obtained by integrating its derivative.

$$\lambda_u(t) = \lambda_u(s) + \int_s^t -\lambda_r(\tau) + \lambda_v(\tau)\frac{v(\tau)}{r(\tau)}d\tau \tag{5.68}$$

The $\lambda_u(s)$ term is zero. The integrand has a value of $-\lambda_r(s)$ at s. Because the integrand is absolutely continuous, the integrand has the sign of $-\lambda_r(s)$ on $[s, t]$ for t sufficiently close to s. That is, there exists a neighborhood around s on which the only zero of λ_u is at s. This implies that all zeros of λ_u are isolated. A set of isolated points has zero measure. The singular case cannot occur on sets of positive measure. The non-singular optimal control satisfies the equation

$$\tan\theta = \frac{\lambda_u}{\lambda_v}. \tag{5.69}$$

Note that some papers, especially those in the engineering literature, investigate singular cases by supposing an interval exists rather than a set of positive measure. This simplifies the analysis, and one could argue it is the "practical" thing to do. It leaves open, however, the possibility that the optimal control is singular on a set of positive measure that contains no intervals.

With the optimal control given above, it is impossible to integrate the resulting equations analytically. The indirect shooting method can be used to numerically solve the problem.

There are six differential equations (three states and three costates). The three initial conditions for the states are known. The strategy in indirect shooting is to guess the three initial conditions for the costates. Integrate all six differential equations to the final time. Check if the three terminal constraints are satisfied.

$$u_f = 0, \quad v_f = \sqrt{\frac{\mu}{r_f}}, \quad \lambda_{rf} = \lambda_0 + \frac{\lambda_{vf}\sqrt{\mu}}{2\,r_f^{3/2}} \tag{5.70}$$

If so, terminate. Otherwise, iterate using Newton's Method. MATLAB's built-in Newton solver is `fsolve`. This procedure reduces the optimal control problem to a root-finding problem.

MATLAB Implementation of Indirect Shooting

For numerical implementation, the initial radius around the sun is $149.6{\cdot}10^9$ m or 1 AU. The initial mass is 4500 kg. The constant thrust force is 4 N. The mass burn rate is $7{\cdot}10^{-5}$ kg/s. The final time is 193 days. Units used in MATLAB are kg, days, and AU.

```
% Unit scalings
MU = 1 / 1;
DU = 1 / 149.6e9;
TU = 1 / 86400;

% Data
T    = 4        * MU*DU/TU^2;
m0   = 4500     * MU;
mdot = 7e-5     * MU/TU;
mu   = 1.327e20 * DU^3/TU^2;
tf   = 193;

% Initial conditions
r0 = 1;
u0 = 0;
v0 = sqrt(mu/r0);
x0 = [r0;u0;v0];
```

A guess for the initial costates is made. How to make such a good guess is discussed shortly.

```
% Initial guess
% placeholder
% placeholder
s=1; unk=[2.1455405424997; 61.7825876741792; 136.3528303294001];
```

A function to integrate the state/costate system is specified.

```
function xdot = ode(t,x,m0,mdot,T,mu)

% States and costates
r  = x(1); u  = x(2); v  = x(3);
lr = x(4); lu = x(5); lv = x(6);

% Optimal thrust angle
theta = atan2(lu,lv);

% State equations
rdot = u;
udot = v^2/r - mu/r^2 + T*sin(theta)/(m0 - mdot*t);
vdot = -u*v/r + T*cos(theta)/(m0 - mdot*t);

% Costate equations
lrdot = -lu*(-v^2/r + 2*mu/r^3) - lv*(u*v/r^2);
ludot = -lr + lv*v/r;
lvdot = -lu*2*v/r + lv*u/r;

xdot = [rdot; udot; vdot; lrdot; ludot; lvdot];
end
```

MATLAB's built-in function `fsolve` is used to perform Newton's method and converge to an initial costate such that all terminal constraints are satisfied. In this light, a function is created to specify the three terminal constraints. Note that integration is performed in this function so it can be evaluated in Newton's method.

```
55  function F = shooting(unk,x0,m0,mdot,T,mu,tf,s)
56
57  lambda0 = 1;
58
59  % Initial costates
60  lr = unk(1);
61  lu = unk(2);
62  lv = unk(3);
63
64  % Integrate state/costate system
65  [~,x] = ode45(@ode,[0,tf],[x0;lr;lu;lv],[],m0,mdot,T,mu);
66
67  % Terminal states and costates
68  rf  = x(end,1); uf  = x(end,2); vf  = x(end,3);
69  lrf = x(end,4); luf = x(end,5); lvf = x(end,6);
70
71  % Terminal constraints
72  F(1) = uf;
73  F(2) = vf - sqrt(mu/rf);
74  F(3) = ( lrf - lambda0 - lvf*sqrt(mu)/(2*rf^(3/2)) )*s;
75  end
```

Newton's method may now be run via MATLAB's `fsolve`.

```
22  % Shooting method
23  ops = optimoptions('fsolve','Display','iter-detailed');
24  sol = fsolve(@shooting,unk,ops,x0,m0,mdot,T,mu,tf,s);
```

Doing so generates the following initial costates.

```
>> sol = [2.1455405424997; 61.7825876741792; 136.3528303294001]
```

We now return to lines 18-21 and the problem of starting with a good guess. A reasonable idea might be to guess random numbers, e.g., `unk = rand(3,1)` or `unk = randn(3,1)`. Doing so does not lead to convergence in `fsolve`. One could also try scaling these random numbers. To simplify this guessing process, recognize that it is the terminal costate constraint that is most complicated. By setting the scaling factor `s = 1e-5`, which in code multiplies the `F(3)` equation, the difficulty can be eliminated. On line 19, implement the following.

```
19  s=1e-5; unk=1e-3*[1;1;1];
```

Running the code leads to a solution (to the scaled problem). Increase the scaling factor to `s = 0.1` and use the previous solution as the initial guess. To do so, implement the following on line 20.

```
20  s=0.1; unk=[2.1454783539651; 61.7820129652734; 136.3627108440081];
```

Running the code leads to a solution (to the scaled problem). Repeat this process by increasing the scaling factor to one and using the previous solution as the initial guess. To do so, implement the following on line 21.

```
21  s=1; unk=[2.1455405424997; 61.7825876741792; 136.3528303294001];
```

With the scaling factor equal to one, the problem of interest is obtained. Running the code leads to a solution of the problem. The final radius is approximately 1.56 AU. This is consistent with the result obtained using direct transcription in Section 4.8. A plot of the radius as a function of time is shown in Figure 5.5.

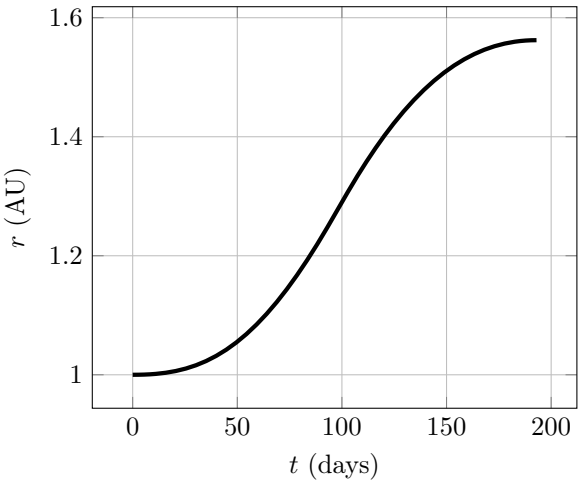


Figure 5.5: Orbit radius as a function of time.

The optimal thrust angle is shown in Figure 5.6.

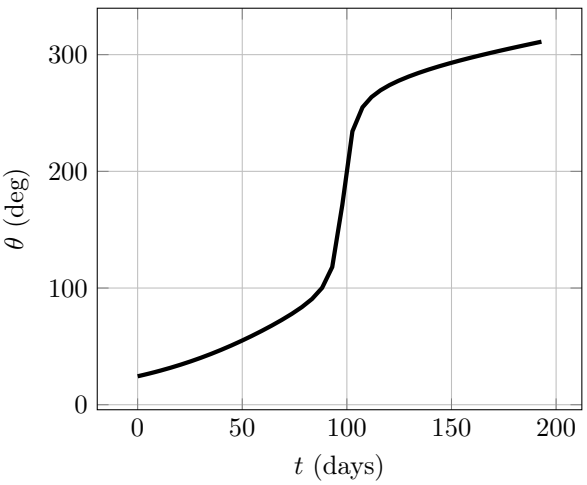


Figure 5.6: Thrust angle as a function of time using indirect shooting.

MATLAB code to generate the plots is below.

```
25 % Simulation and plots
26 [t,x] = ode45(@ode,[0,tf],[x0;sol],[],m0,mdot,T,mu);
27 figure, plot(t,x(:,1)), grid on
28 xlabel('t'), ylabel('r (AU)')
29
30 theta = atan2(x(:,5),x(:,6));
31 theta = 180/pi*wrapTo2Pi(theta);
32 figure, plot(t,theta), grid on
33 xlabel('t'), ylabel('theta (deg)')
```

ℹ This numerical solution is one candidate solution. Other solutions may be generated by a different initialization process. Furthermore, the search for abnormal solutions has not been done. These challenges are left for the reader to explore.

MATLAB Implementation of Direct Shooting

The indirect shooting method requires the guess of costates, which are non-physical. An alternative is direct shooting. The controls are guessed at node times, an interpolation scheme is chosen, and the state equations are integrated. The benefit of direct shooting is that costates are completely eliminated and physical quantities are guessed. The downside of direct shooting is that the number of quantities to be guessed increases. Furthermore, parameterization (or discretization and interpolation) of the controls results in a sub-optimal (or approximate) control.

Numerically, the solution process begins the same and lines 1-17 are unchanged. A selection of time nodes and guess of control values at those nodes must be made. As before, a good guess leads to a quick numerical solution and a poor guess leads to convergence problems. Based on the indirect shooting solution, it is seen that the thrust angle time history looks like a scaled and shifted hyperbolic tangent function. For this reason, the following guess is made.

```
18 % Control guess
19 N = 50;
20 tspan = linspace(0,tf,N);
21 theta = 150*tanh( (tspan-100)/30 )+170;
22 theta = theta*pi/180;
```

In between node times, linear interpolation is done. Before using the guess in optimization, the state differential equations, objective function, and constraint function must be coded for compatibility with MATLAB's constrained solver `fmincon`. The differential equations are in the function `ode`. The guessed control values are entering the function as `thvec` and the associated node times are `tvec`. Interpolation occurs on line 45.

```
42 % State differential equation
43 function xdot = ode(t,x,m0,mdot,T,mu,tvec,thvec)
44 r = x(1); u = x(2); v = x(3);
45 theta = interp1(tvec,thvec,t);
46 rdot  = u;
```

```
47  udot  = v^2/r - mu/r^2 + T*sin(theta)/(m0-mdot*t);
48  vdot  = -u*v/r + T*cos(theta)/(m0-mdot*t);
49  xdot  = [rdot;udot;vdot];
50  end
```

The objective function `obj` integrates so that the final radial position is known. It is negated because the problem is to maximize.

```
51  % Objective function
52  function J = obj(theta,x0,m0,mdot,T,mu,tspan)
53  [~,x] = ode45(@ode,tspan,x0,[],m0,mdot,T,mu,tspan,theta);
54  J = -x(end,1);
55  end
```

The constraint function `con` integrates so that all final states are known. The terminal state constraints are enforced as equality constraints in the variable `ceq`. There are no inequality constraints in `cin`.

```
56  % Constraint function
57  function [cin,ceq] = con(theta,x0,m0,mdot,T,mu,tspan)
58  [~,x] = ode45(@ode,tspan,x0,[],m0,mdot,T,mu,tspan,theta);
59  rf = x(end,1);
60  uf = x(end,2);
61  vf = x(end,3);
62  cin = [];
63  ceq(1,1) = uf - 0;
64  ceq(2,1) = vf - sqrt(mu/rf);
65  end
```

With these functions in place, MATLAB's `fmincon` is used to optimize the control values at the node times.

```
23  % Optimize
24  ops = optimoptions('fmincon','Display','iter',...
25                     'EnableFeasibilityMode',true,...
26                     'SubproblemAlgorithm','cg',...
27                     'MaxFunEvals',5e3);
28  LB = linspace(0,0,N);
29  UB = linspace(2*pi,2*pi,N);
30  pobj = @(theta) obj(theta,x0,m0,mdot,T,mu,tspan);
31  pcon = @(theta) con(theta,x0,m0,mdot,T,mu,tspan);
32  theta = fmincon(pobj,theta,[],[],[],[],LB,UB,pcon,ops);
```

Running the code leads to a solution of the problem. The final radius is approximately 1.56 AU as in the indirect shooting method. A plot of the thrust angle is shown in Figure 5.7. Observe that it is not smooth because the optimizer is simply adjusting the control values at the node times.

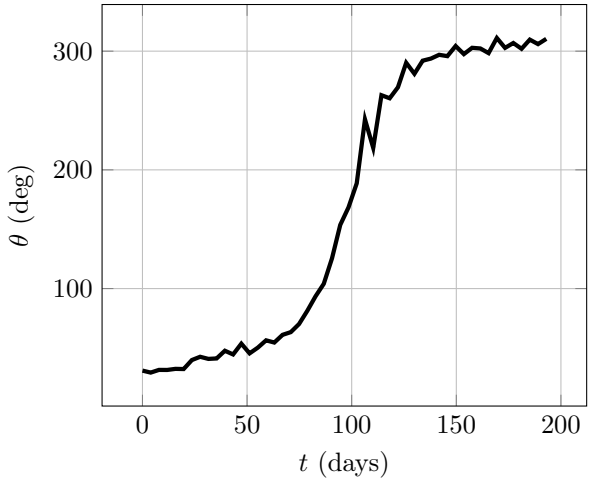


Figure 5.7: Thrust angle as a function of time using direct shooting.

The thrust angle time history can be made monotonic (and smoother) by enforcing an inequality constraint in line 62.

```
62  cin = -diff(theta);
```

In any case, MATLAB code to generate the plots is below.

```
33  % Integrate the solution
34  [~,x] = ode45(@ode,tspan,x0,[],m0,mdot,T,mu,tspan,theta);
35
36  % Plot the solution
37  figure, plot(tspan,x(:,1)), grid on
38  xlabel('t (days)'), ylabel('r (AU)')
39
40  figure, plot(tspan,theta*180/pi), grid on
41  xlabel('t (days)'), ylabel('theta (deg)')
```

MATLAB Implementation of Global Control Parameterization

The hyperbolic tangent initial guess was sufficient to achieve convergence in direct shooting. A more extreme approach, which dramatically reduces the problem size, is to parameterize the control function over the entire time domain. In this example, it is assumed that the thrust angle has the form

$$\theta = c_1 \tanh\left((t - c_2)/c_3\right) + c_4, \tag{5.71}$$

and the optimization variables are now the coefficients c_1, c_2, c_3, and c_4. The resulting control is sub-optimal unless the optimality conditions dictate the optimal control has this form. For numerical implementation, lines 1-17 remain the same. Consistent with the coefficients specified in the direction shooting method, the guess now appears as the following.

```
% Control guess
N = 50;
tspan = linspace(0,tf,N);
coeff = [150;100;30;170];
```

The state differential equations, objective function, and constraint function undergo minor modification. Most importantly, the thrust angle is computed using the hyperbolic tangent function on line 37.

```
% State differential equation
function xdot = ode(t,x,m0,mdot,T,mu,tvec,coeff)
r = x(1); u = x(2); v = x(3);
theta = coeff(1)*tanh( (t-coeff(2))/coeff(3) )+coeff(4);
theta = theta*pi/180;
rdot = u;
udot = v^2/r - mu/r^2 + T*sin(theta)/(m0-mdot*t);
vdot = -u*v/r + T*cos(theta)/(m0-mdot*t);
xdot = [rdot;udot;vdot];
end

% Objective function
function J = obj(coeff,x0,m0,mdot,T,mu,tspan)
[~,x] = ode45(@ode,tspan,x0,[],m0,mdot,T,mu,tspan,coeff);
J = -x(end,1);
end

% Constraint function
function [cin,ceq] = con(coeff,x0,m0,mdot,T,mu,tspan)
[~,x] = ode45(@ode,tspan,x0,[],m0,mdot,T,mu,tspan,coeff);
rf = x(end,1);
uf = x(end,2);
vf = x(end,3);
cin = [];
ceq(1,1) = uf - 0;
ceq(2,1) = vf - sqrt(mu/rf);
end
```

With these functions in place, MATLAB's `fmincon` is used to optimize the coefficient values.

```
% Optimize
ops = optimoptions('fmincon','Display','iter',...
                   'EnableFeasibilityMode',true,...
                   'SubproblemAlgorithm','cg',...
                   'MaxFunEvals',5e3);
LB = [100;50;5;100];
UB = [200;200;50;200];
pobj = @(coeff) obj(coeff,x0,m0,mdot,T,mu,tspan);
pcon = @(coeff) con(coeff,x0,m0,mdot,T,mu,tspan);
coeff = fmincon(pobj,coeff,[],[],[],[],LB,UB,pcon,ops);
```

```
32  theta = coeff(1)*tanh( (tspan-coeff(2))/coeff(3) )+coeff(4);
33  theta = theta*pi/180;
```

Running the code leads to a solution of the problem. The final radius is approximately 1.56 AU as with both previous methods. A plot of the thrust angle is shown in Figure 5.8.

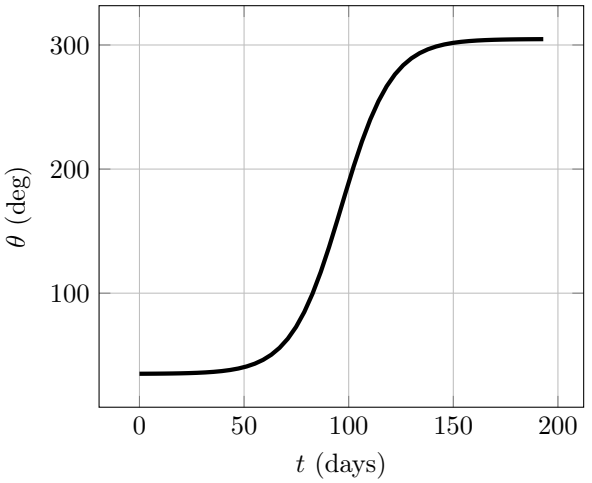


Figure 5.8: Thrust angle as a function of time using global control parameterization.

This problem has now been solved four ways. Direct transcription was used in Section 4.8. A Chebyshev discretization was used and both states and controls were optimization variables. In this chapter, the indirect shooting, direct shooting, and global control parameterization methods were used. Indirect shooting requires guessing costates and integrating the state/costate system. Direct shooting requires guessing the control at node times and integrating the state system. Global parameterization requires guessing coefficients in an assumed functional form and integrating the state system. Though there are many variants of these four methods, they form the foundation of numerical optimal control. The names of these numerical methods vary over specialized disciplines. The names used in this book are standard in optimal spacecraft guidance.[17]

In each method, the solution is sensitive to the initial guess. This is because the problem is not convex. For onboard implementations, this sensitivity is problematic. For onboard implementations, it is common to reduce the problem to a simpler form or transform it to a convex form (if it is not naturally convex). This ensures that convergence is achieved in an appropriate amount of time.

[17] Hull, *Conversion of Optimal Control Problems into Parameter Optimization Problems*, 1997.

5.6 Chapter Problems

Problem 5.1. Let I be an interval of time and $f : I \to \mathbb{R}$ be an analytic function. Is the following statement true or false?

$$f \overset{\text{ae}}{=} 0 \quad \Longrightarrow \quad f = 0$$

Problem 5.2. Let I be an interval of time and $f : I \to \mathbb{R}$ be an absolutely continuous function. Is the following statement true or false?

$$f \overset{\text{ae}}{=} 0 \quad \Longrightarrow \quad f = 0$$

Problem 5.3. Let $f(t) = \sin t$. What is a function g such that $f \overset{\text{ae}}{=} g$ and $f \neq g$?

Problem 5.4. What is the definition of an absolutely continuous function (in terms of derivatives and integrals) and how has this been used in the analysis of the maximal orbit raise problem in Section 5.5?

Problem 5.5. Derive the open-loop solution for the continuous-time LQC problem analogous to that in Section 4.3.

Problem 5.6. Derive the closed-loop solution for the continuous-time LQR problem analogous to that in Section 4.4.

Problem 5.7. Derive the closed-loop solution for the continuous-time LQT problem analogous to that in Section 4.5

Problem 5.8. Solve Problem 4.4 using a continuous-time formulation.

Problem 5.9. Solve the minimum time problem in Eq. (5.46) by implementing the state feedback solution. Test the implementation by starting at different initial conditions.

Problem 5.10. Find the globally optimal control for the following minimum time problem.

$$\begin{aligned}
&\text{minimize} && t_f \\
&\text{subject to} && \dot{x}_1 = x_2, && x_1(0) = 2, \quad x_{1f} = 0 \\
&&& \dot{x}_2 = -x_1 + u, && x_2(0) = 0, \quad x_{2f} = 0 \\
&&& |u| \leq 1
\end{aligned}$$

The variables $x_1, x_2, u \in \mathbb{R}$. Note that the dynamics represent a harmonic oscillator (a mass attached to a spring without damping). State whether the globally optimal solution is normal or abnormal.

Problem 5.11. Similar to the setting of Section 5.3, an evading vehicle is flying at a constant altitude h, is initially downrange ℓ from the launch site, and is moving downrange at constant speed V. At this moment, the pursuing vehicle launches and applies a constant thrust acceleration τ so that its only control is the thrust angle θ. The goal is to achieve a minimum time intercept.

(a) Write down the equations of motion assuming a flat planet model.
(b) Using optimal control, derive the equations to solve for the optimal thrust angle.

(c) Using reasonable numerical values, simulate the trajectory to confirm interception occurs.

Problem 5.12. Solve the maximal orbit raise problem described in Section 5.5 with a flight time of 365 days. Keep all other problem data the same.

a) Use direct transcription.
b) Use indirect shooting.
c) Use direct shooting.
d) Use global control parameterization.

Chapter 6

Descent Guidance

Chapter Learning Objectives

1. Develop and implement the optimal state feedback law for terminal descent.
2. Develop Apollo-era guidance laws from optimization and polynomial perspectives.
3. Implement Apollo-era guidance laws.
4. Develop and implement computational guidance laws using convex optimization.

There are several phases to a powered descent trajectory: deorbit, braking and approach, pitchover, and vertical descent to touchdown. In the vertical descent phase, the vehicle is already positioned above the landing site with downward velocity. The optimal guidance problem is to determine the fuel optimal thrust profile to achieve a soft (zero terminal speed) landing. In this phase, the minimum fuel problem is equivalent to the minimum time problem. Singular solutions cannot occur and it is shown that the optimal thrust profile contains a coasting phase followed by a maximum thrust phase. Because of this, the optimal control can be written in state feedback form.

The braking and approach phase is three-dimensional and designed to transfer a vehicle from periapse of an approach orbit to the landing site (or near it). An optimal control problem that penalizes flight time and thrust acceleration admits an analytical solution suitable for implementation on Apollo-era flight computers. It is then shown that this optimal guidance law is obtainable by special choice of polynomial basis functions. In this setting, the historically significant E-guidance and Apollo lunar descent guidance laws are derived. These are similar to the polynomial guidance developed and implemented in Section 2.2.

Perhaps the most important feature of the Apollo-era guidance laws is their analytic nature. They are not, however, fuel optimal. They solve for thrust acceleration rather than thrust force, and they do not incorporate thrust magnitude constraints. To remedy these deficiencies, a computational approach for computing optimal trajectories is developed. In this approach, direct transcription is used to solve a convex, discrete-time version of an originally nonconvex, continuous-time problem. Then, LQR can be used to follow this fuel optimal trajectory.

6.1 Terminal Descent Guidance

Consider the one-dimensional terminal descent phase of a planetary landing[18]. At the initial time, the vehicle is directly above the landing site with downward velocity. The thrust is bounded in magnitude. The goal is to reach the surface with zero speed and minimize the fuel consumed. In this one-dimensional setting, the equations of motion are consistent with the flat planet model in Eq. (2.10) and mass dynamics in Eq. (2.44).

$$\ddot{r} = g + T/m, \quad \dot{m} = -\alpha T \tag{6.1}$$

Observe that $\dot{m}/m = \frac{d}{dt}\ln m$ so that the dynamics may be rewritten as

$$\ddot{r} = g - \frac{1}{\alpha}\frac{d}{dt}\ln m. \tag{6.2}$$

Integration from 0 to t gives

$$\dot{r} - \dot{r}_0 = gt - \frac{1}{\alpha}\ln\frac{m}{m_0}. \tag{6.3}$$

To land with zero speed, $\dot{r}_f = 0$, such that

$$\frac{1}{\alpha}\ln\frac{m_f}{m_0} = \dot{r}_0 + gt_f. \tag{6.4}$$

The mass at the final time is

$$m_f = m_0 \exp\left[\alpha(\dot{r}_0 + gt_f)\right]. \tag{6.5}$$

The fuel consumed is

$$m_0 - m_f = m_0 - m_0 \exp\left[\alpha(\dot{r}_0 + gt_f)\right]. \tag{6.6}$$

It is evident from Eq. (6.6) that for given constants m_0, $\dot{r}_0$, g, and α, the fuel required to bring the vehicle to rest is strictly monotonic with flight time t_f. For this reason, the minimum fuel problem can be solved as the following minimum time problem.

$$\begin{array}{rlll} \text{minimize} & t_f & & \\ \text{subject to} & \dot{r} = v, & r(0) = r_0, & r_f = 0 \\ & \dot{v} = g + T/m, & v(0) = v_0, & v_f = 0 \\ & \dot{m} = -\alpha T, & m(0) = m_0 & \\ & 0 \le T \le T_{max} & & \end{array} \tag{6.7}$$

The solution procedure is to first form the Hamiltonian and endpoint functions.

$$H = \lambda_r v + \lambda_v (g + T/m) + \lambda_m (-\alpha T) \tag{6.8}$$

$$G = \lambda_0 t_f + \nu_r r_f + \nu_v v_f \tag{6.9}$$

[18] Meditch, *On the Problem of Optimal Thrust Programming for a Lunar Soft Landing*, 1964.

The costate, Hamiltonian, and transversality conditions generate the following equations.

$$\dot{\lambda}_r = 0, \qquad \lambda_{rf} = \nu_r \tag{6.10}$$

$$\dot{\lambda}_v = -\lambda_r, \qquad \lambda_{vf} = \nu_v \tag{6.11}$$

$$\dot{\lambda}_m = \lambda_v T/m^2, \quad \lambda_{mf} = 0 \tag{6.12}$$

$$\dot{H} = 0, \qquad H_f = -\lambda_0 \tag{6.13}$$

It is immediately deduced that λ_r and H are constants. The costate λ_v is constant or varying linearly in time. The pointwise minimum condition yields

$$T \overset{\text{ae}}{\in} \underset{0\leq\omega\leq T_{max}}{\operatorname{argmin}} \left(\frac{\lambda_v}{m} - \alpha\lambda_m\right)\omega \quad \Longrightarrow \quad T \overset{\text{ae}}{=} \begin{cases} 0, & \lambda_v - m\alpha\lambda_m > 0 \\ T_{max}, & \lambda_v - m\alpha\lambda_m < 0 \\ \text{singular}, & \lambda_v - m\alpha\lambda_m = 0 \end{cases}. \tag{6.14}$$

The singular case occurs when the switching function $f = \lambda_v - m\alpha\lambda_m$ is zero. The singular case is negligible if it occurs on a set of zero measure. Suppose, on the other hand, it occurs on a set of positive measure. Because of m and λ_m, the function f is absolutely continuous. Being zero on a set of positive measure does not imply being zero everywhere. A deeper analysis is required.

Let s be a point in the supposed set such that $f(s) = 0$. A formula for f is obtained by integrating its derivative.

$$f(t) = f(s) + \int_s^t -\lambda_r - \alpha f(\tau)\frac{T(\tau)}{m(\tau)}\, d\tau \tag{6.15}$$

The $f(s)$ term is zero. The integrand has a value of $-\lambda_r$ at s. If λ_r is non-zero, the integrand has the sign of $-\lambda_r$ on $[s, t]$ for t sufficiently close to s. This is true because f and m are absolutely continuous and T is bounded. This implies that the zero s of the switching function is isolated. If all zeros are isolated, then the set of zeros has zero measure. Therefore, having f be zero on a set of positive measure implies $\lambda_r = 0$. It follows that λ_v is constant. There are three cases to investigate.

① Suppose $\lambda_v = 0$. Then λ_m is constant and equal to zero. At the final time, the Hamiltonian is zero such that $\lambda_0 = 0$. This violates non-triviality. Thus, $\lambda_v \neq 0$.

② Suppose $\lambda_v > 0$. At s, $\lambda_m(s) > 0$ since $\lambda_v(s) - m(s)\alpha\lambda_m(s) = 0$. Also, $\dot{\lambda}_m \geq 0$ almost everywhere. A function that is positive and never decreasing cannot satisfy the transversality condition $\lambda_{mf} = 0$. Thus, $\lambda_v \not> 0$.

③ Suppose $\lambda_v < 0$. At s, $\lambda_m(s) < 0$ since $\lambda_v(s) - m(s)\alpha\lambda_m(s) = 0$. Also, $\dot{\lambda}_m \leq 0$ almost everywhere. A function that is negative and never increasing cannot satisfy the transversality condition $\lambda_{mf} = 0$. Thus, $\lambda_v \not< 0$.

The singular case cannot occur on sets of positive measure. The optimal control can be chosen to take only two values: 0 and T_{max}. It is now shown that the control switches at most one time. It has already been shown that zeros of f are isolated. Let t_1 and t_2 be consecutive zeros of f. It follows from continuity and non-singularity that f has constant non-zero sign on (t_1, t_2).

① Suppose $f > 0$ and $T = 0$ on (t_1, t_2). It follows that $\dot{f} = -\lambda_r$ and f is monotonic on (t_1, t_2). This is impossible. Thus, $f \not> 0$ on (t_1, t_2).

② Suppose $f < 0$ and $T = T_{max}$ on (t_1, t_2). It follows that $\dot{f} = -\lambda_r + \alpha|f|T_{max}/m$ and is continuous on (t_1, t_2). The second term is strictly positive. For f to not be monotonic, $\lambda_r > 0$. In a sufficiently small neighborhood of t_2, $\dot{f}$ is negative. The function f is negative and decreasing near t_2 such that t_2 cannot be a zero of f. Thus $f \not< 0$ on (t_1, t_2).

It is concluded that the switching function cannot have two zeros, i.e., there is at most one control switch. Lastly, the thruster must be on at the final time to meet the zero velocity constraint. This means there are two possible optimal thrust sequences: $\{0, T_{max}\}$ and $\{T_{max}\}$. Designing a state feedback law to initiate the thrust is left as a problem for the reader.

6.2 Apollo-era Descent Guidance

Returning to the three-dimensional flat planet model, an optimal control problem that penalizes flight time and thrust acceleration is now considered.[19] An advantage of the formulation is that it yields an analytical solution for the optimal controls. Disadvantages are that it does not minimize fuel consumption, it solves for thrust acceleration rather than thrust, and it does not incorporate thrust magnitude constraints. If thrust magnitude constraints are violated, the optimal controls must be recomputed with a new final time weighting Γ in hopes that the new Γ lowers the required thrust magnitude a sufficient amount. The optimal control problem is the following.

$$\begin{aligned} \text{minimize} \quad & \Gamma t_f + \frac{1}{2}\int_0^{t_f} u^\top u \, dt \\ \text{subject to} \quad & \dot{r} = v, \qquad r(0) = r_0, \quad r_f = 0 \\ & \dot{v} = g + u, \quad v(0) = v_0, \quad v_f = 0 \end{aligned} \tag{6.16}$$

The position r, velocity v, thrust acceleration u, and gravitational acceleration g are all three-dimensional. For small values of Γ, longer flight times and smaller control magnitudes are expected. For large values of Γ, shorter flight times and larger control magnitudes are expected. The solution procedure is to first form the Hamiltonian and endpoint functions.

$$H = \frac{\lambda_0}{2} u^\top u + \lambda_r^\top v + \lambda_v^\top (g + u) \tag{6.17}$$

$$G = \lambda_0 \Gamma t_f + \nu_r^\top r_f + \nu_v^\top v_f \tag{6.18}$$

The costate and Hamiltonian equations are the following.

$$\dot{\lambda}_r = 0 \tag{6.19}$$

$$\dot{\lambda}_v = -\lambda_r \tag{6.20}$$

$$\dot{H} = 0 \tag{6.21}$$

It is immediately deduced that λ_r and H are constants. The costate λ_v is constant or varying linearly with time.

[19] D'Souza, *An Optimal Guidance Law for Planetary Landing*, 1997.

The transversality conditions generate the following equations.

$$\lambda_{rf} = \nu_r \tag{6.22}$$
$$\lambda_{vf} = \nu_v \tag{6.23}$$
$$H_f = -\lambda_0 \Gamma \tag{6.24}$$

Integrating the costate equations in terms of $t_{go} = t_f - t$ and substituting the boundary conditions gives the following.

$$\lambda_r = \nu_r, \quad \lambda_v = \nu_r t_{go} + \nu_v \tag{6.25}$$

The pointwise minimum condition yields

$$u \overset{\text{ae}}{\in} \underset{\omega}{\operatorname{argmin}} \frac{\lambda_0}{2}\omega^\top \omega + \lambda_v^\top \omega \quad \implies \quad \lambda_0 u + \lambda_v \overset{\text{ae}}{=} 0. \tag{6.26}$$

Having $\lambda_0 = 0$ implies $\lambda_v = 0$ everywhere. The costate equations imply $\lambda_r = 0$. This violates non-triviality. With $\lambda_0 = 1$, the optimal controls are

$$u = -\lambda_v = -\nu_r t_{go} - \nu_v. \tag{6.27}$$

The optimal controls are linear functions of time, which is why the simple idea of polynomial guidance discussed in Section 2.2 is a good one. These functions can be substituted into the state equations and integrated to yield the following state equations.

$$v = \frac{1}{2}\nu_r t_{go}^2 + \nu_v t_{go} - g t_{go}, \quad r = -\frac{1}{6}\nu_r t_{go}^3 - \frac{1}{2}\nu_v t_{go}^2 + \frac{1}{2} g t_{go} \tag{6.28}$$

These six equations contain the undetermined constants ν_r and ν_v. Solving for these in terms of the state values and substituting into the control equations yields the optimal control.

$$u = -\frac{4v}{t_{go}} - \frac{6r}{t_{go}^2} - g \tag{6.29}$$

> ℹ As time approaches t_f, t_{go} approaches zero. In practice, this causes the control accelerations to explode. Simple close-out strategies are to turn control off beyond a threshold value of t_{go} or hold t_{go} constant beyond a threshold value.

The optimal control problem is not yet solved because the final time has not been determined. Substituting the state, costate, and control functions into the Hamiltonian results in a quartic function of t_{go}.

$$(\Gamma + \frac{1}{2} g^\top g) t_{go}^4 - 2v^\top v t_{go}^2 - 12 r^\top v t_{go} - 18 r^\top r = 0. \tag{6.30}$$

This equation admits four solutions. The strategy from here is to numerically solve the equation and pick the real, positive root associated with the least objective value.

There are historically significant variations of the above guidance law, and these can be achieved by selection of basis functions rather than optimization.[20] The desired

[20] Lu, *The Theory of Fractional-Polynomial Powered Descent Guidance*, 2020.

thrust acceleration u_d is written as a linear combination of two basis functions ϕ_1 and ϕ_2.

$$u_d = c_1\phi_1(t_{go}) + c_2\phi_2(t_{go}) \tag{6.31}$$

First and second integrals of the basis functions are

$$\bar{\phi}_i(t_{go}) = \int_{t_{go}}^{0} \phi_i(\tau)\, d\tau, \tag{6.32}$$

$$\hat{\phi}_i(t_{go}) = \int_{t_{go}}^{0} \bar{\phi}_i(\tau)\, d\tau. \tag{6.33}$$

Integrating the state equations backward from t_f gives the desired velocity and position.

$$v_d = c_1\bar{\phi}_1(t_{go}) + c_2\bar{\phi}_2(t_{go}) - gt_{go} \tag{6.34}$$

$$r_d = c_1\hat{\phi}_1(t_{go}) + c_2\hat{\phi}_2(t_{go}) + \frac{1}{2}gt_{go}^2 \tag{6.35}$$

To track this trajectory, consider the feedback form

$$u = u_d - \beta_v(t_{go})\left(v - v_d\right) - \beta_r(t_{go})\left(r - r_d\right), \tag{6.36}$$

where β_v and β_r are feedback gains that must be determined. Henceforth, the (t_{go}) argument is dropped for brevity. Substituting in for a_d, v_d, and r_d gives

$$\begin{aligned} u = {} & c_1\left(\phi_1 + \beta_v\bar{\phi}_1 + \beta_r\hat{\phi}_1\right) + c_2\left(\phi_2 + \beta_v\bar{\phi}_2 + \beta_r\hat{\phi}_2\right) \\ & + gt_{go}\left(\frac{1}{2}\beta_r t_{go} - \beta_v\right) - \beta_v v - \beta_r r. \end{aligned} \tag{6.37}$$

The feedback gains β_v and β_r are chosen, for simplicity, such that the coefficients of c_1 and c_2 are zero.

$$\beta_r = \frac{\phi_1\bar{\phi}_2 - \phi_2\bar{\phi}_1}{\Delta}, \quad \beta_v = \frac{\phi_2\hat{\phi}_1 - \phi_1\hat{\phi}_2}{\Delta}, \quad \Delta = \bar{\phi}_1\hat{\phi}_2 - \hat{\phi}_1\bar{\phi}_2 \neq 0 \tag{6.38}$$

With this selection, the thrust acceleration becomes

$$u = gt_{go}\left(\frac{1}{2}\beta_r t_{go} - \beta_v\right) - \beta_v v - \beta_r r. \tag{6.39}$$

This thrust acceleration guides the vehicle from its current state to the origin. Basis functions are now chosen to be

$$\phi_1 = 1, \quad \phi_2 = t_{go}. \tag{6.40}$$

First integrals are

$$\bar{\phi}_1 = -t_{go}, \quad \bar{\phi}_2 = -\frac{1}{2}t_{go}^2. \tag{6.41}$$

Second integrals are

$$\hat{\phi}_1 = \frac{1}{2}t_{go}^2, \quad \hat{\phi}_2 = \frac{1}{6}t_{go}^3. \tag{6.42}$$

The resulting feedback gains are

$$\beta_r = \frac{6}{t_{go}^2}, \quad \beta_v = \frac{4}{t_{go}}. \tag{6.43}$$

The guidance law, after some simplification, is

$$u = -\frac{4v}{t_{go}} - \frac{6r}{t_{go}^2} - g. \tag{6.44}$$

This is exactly the same result obtained by solving the optimal control problem. This guidance law is called E-guidance and was first derived in 1964.[21]

The two-term parameterization in Eq. (6.31) can be generalized by increasing to a three-term parameterization. The third term is the desired final thrust acceleration u_f supposing $\phi_1(0) = \phi_2(0) = 0$. The desired thrust acceleration is

$$u_d = u_f + c_1\phi_1(t_{go}) + c_2\phi_2(t_{go}). \tag{6.45}$$

Integrating the state equations backward from t_f gives the desired velocity and position.

$$v_d = c_1\bar{\phi}_1 + c_2\bar{\phi}_2 - (g + u_f)t_{go} \tag{6.46}$$

$$r_d = c_1\hat{\phi}_1 + c_2\hat{\phi}_2 + \frac{1}{2}(g + u_f)t_{go}^2 \tag{6.47}$$

The feedback form for tracking in Eq. (6.36) becomes, after substitution,

$$\begin{aligned} u = c_1\left(\phi_1 + \beta_v\bar{\phi}_1 + \beta_r\hat{\phi}_1\right) + c_2\left(\phi_2 + \beta_v\bar{\phi}_2 + \beta_r\hat{\phi}_2\right) \\ + u_f + (g + u_f)t_{go}\left(\frac{1}{2}\beta_r t_{go} - \beta_v\right) - \beta_v v - \beta_r r. \end{aligned} \tag{6.48}$$

The same choice of β_r and β_v as before renders the coefficients of c_1 and c_2 zero. With this selection, the thrust acceleration becomes

$$u = u_f + (g + u_f)t_{go}\left(\frac{1}{2}\beta_r t_{go} - \beta_v\right) - \beta_v v - \beta_r r. \tag{6.49}$$

Basis functions are now chosen to be

$$\phi_1 = t_{go}, \quad \phi_2 = t_{go}^2, \tag{6.50}$$

which results in the feedback gains

$$\beta_r = \frac{12}{t_{go}^2}, \quad \beta_v = \frac{6}{t_{go}}. \tag{6.51}$$

The guidance law, after some simplification, is

$$u = u_f - \frac{6v}{t_{go}} - \frac{12r}{t_{go}^2}. \tag{6.52}$$

This is exactly the Apollo lunar descent guidance law.[22] Implementation of these Apollo-era guidance laws is left as a problem for the reader.

[21] Cherry, *A General, Explicit, Optimizing Guidance Law for Rocket-Propelled Spaceflight*, 1964.
[22] Klumpp, *Apollo Lunar Descent Guidance*, 1974.

6.3 Computational Descent Guidance

The Apollo-era guidance laws are analytical. This was necessary in the 1960s. However, they do not minimize fuel consumption, they solve for thrust acceleration rather than thrust force, and they do not incorporate thrust magnitude constraints. Thus, there is room for improvement. Modern flight computers, while still nowhere near as powerful as desktop computers, are capable of non-trivial online computation, and this can be exploited in guidance.[23]

Consider the following minimum fuel consumption optimal control problem.

$$\begin{aligned}
&\text{maximize} && m_f \\
&\text{subject to} && \dot{r} = v, && r(0) = r_0, && r_f = 0 \\
& && \dot{v} = g + T/m, && v(0) = v_0, && v_f = 0 \\
& && \dot{m} = -\alpha||T||, && m(0) = m_0 \\
& && 0 < \rho_1 \leq ||T|| \leq \rho_2
\end{aligned} \tag{6.53}$$

The dynamics are again consistent with the flat planet model in Eq. (2.10) and mass dynamics in Eq. (2.44). The thrust magnitude is upper bounded by ρ_2. This constraint arises because thrusters cannot produce infinite force. The thrust magnitude is lower bounded by ρ_1. This constraint arises because thrusters cannot produce thrust reliably below a certain level. Also, it is unwise to turn an engine off during descent. If the engine fails to restart, the mission terminates with a crash. In two dimensions, the thrust constraint appears as the annulus shown in Figure 6.1.

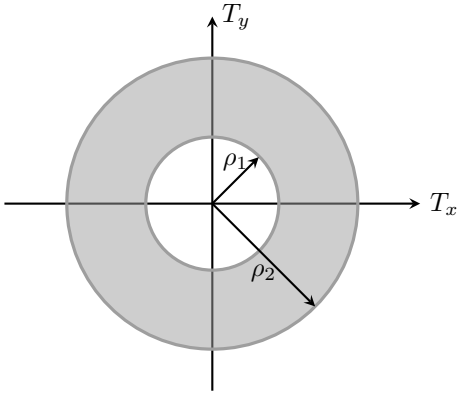


Figure 6.1: Nonconvex annular thrust constraint.

The solution procedure is to first form the Hamiltonian and endpoint functions.

$$H = \lambda_r^\top v + \lambda_v^\top (g + T/m) + \lambda_m (-\alpha||T||) \tag{6.54}$$

$$G = \lambda_0 m_f + \nu_r r_f + \nu_v v_f \tag{6.55}$$

[23] Blackmore, *Autonomous Precision Landing of Space Rockets*, 2016.

The costate, Hamiltonian, and transversality conditions generate the following equations.

$$\dot{\lambda}_r = 0, \qquad \lambda_{rf} = \nu_r \tag{6.56}$$

$$\dot{\lambda}_v = -\lambda_r, \qquad \lambda_{vf} = \nu_v \tag{6.57}$$

$$\dot{\lambda}_m = \lambda_v^\top T/m^2, \quad \lambda_{mf} = \lambda_0 \tag{6.58}$$

$$\dot{H} = 0, \qquad H_f = 0 \tag{6.59}$$

It is immediately deduced that H is constant and zero. The costate λ_r is constant. The costate λ_v is constant or varying linearly in time. The pointwise maximum condition yields

$$T \overset{\text{ae}}{\in} \underset{\rho_1 \leq ||\omega|| \leq \rho_2}{\operatorname{argmax}} \frac{\lambda_v^\top \omega}{m} - \alpha \lambda_m ||\omega||. \tag{6.60}$$

The first term implies that the thrust direction should be aligned with λ_v. The directionally singular case occurs when λ_v is zero. The directionally singular case is negligible if it occurs on a set of zero measure. Suppose, on the other hand, it occurs on a set of positive measure. Because λ_v is a linear (analytic) function of time, it is zero everywhere. This implies $\lambda_r = 0$ everywhere. The Hamiltonian being zero implies that $\lambda_m = 0$ everywhere. The transversality condition on λ_{mf} implies $\lambda_0 = 0$, which violates non-triviality. The directionally singular case cannot occur on sets of positive measure. The optimal thrust direction $\hat{T}$ satisfies, everywhere except possibly a single point where $\lambda_v = 0$,

$$\hat{T} = \frac{\lambda_v}{||\lambda_v||}. \tag{6.61}$$

The pointwise maximum condition is rewritten as follows.

$$||T|| \overset{\text{ae}}{\in} \underset{\rho_1 \leq ||\omega|| \leq \rho_2}{\operatorname{argmax}} \left(\frac{||\lambda_v||}{m} - \alpha \lambda_m \right) ||\omega|| \implies ||T|| \overset{\text{ae}}{=} \begin{cases} \rho_2, & ||\lambda_v|| - m\alpha\lambda_m > 0 \\ \rho_1, & ||\lambda_v|| - m\alpha\lambda_m < 0 \\ \text{singular}, & ||\lambda_v|| - m\alpha\lambda_m = 0 \end{cases} \tag{6.62}$$

The magnitudinally singular case occurs when the switching function $f = ||\lambda_v|| - m\alpha\lambda_m$ is zero. The magnitudinally singular case is negligible if it occurs on a set of zero measure. Suppose, on the other hand, it occurs on a set of positive measure. Because of m and λ_m, the function f is absolutely continuous. Being zero on a set of positive measure does not imply being zero everywhere. A deeper analysis is required.

Let s be a point in the supposed set such that $f(s) = 0$ and $\lambda_v(s) \neq 0$. A formula for f is obtained by integrating its derivative.

$$f(t) = f(s) + \int_s^t -\frac{\lambda_v(\tau)^\top \lambda_r}{||\lambda_v(\tau)||} - \alpha f(\tau) \frac{||T(\tau)||}{m(\tau)} \, d\tau \tag{6.63}$$

The $f(s)$ term is zero. Define the linear function $h = \lambda_v^\top \lambda_r$. The integrand has a value of $-h(s)/||\lambda_v(s)||$ at s. If $h(s)$ is non-zero, the integrand has the sign of $-h(s)$ on $[s, t]$ for t sufficiently close to s. This is true because f and m are absolutely continuous and T is bounded. This implies that the zero s of the switching function is isolated. If all zeros are isolated, then the set of zeros has zero measure. Having f be zero on a set of positive measure implies it has uncountably many non-isolated zeros. Let s and s'

be two such points, i.e., $f(s) = f(s') = h(s) = h(s') = 0$. It follows from linearity of h that it is zero everywhere. Differentiating h and setting to zero implies $\lambda_r = 0$ and λ_v is constant.

Rewriting the Hamiltonian with $\lambda_r = 0$ and $T = \hat{T}||T||$ implies

$$\lambda_v^\top g + f\frac{||T||}{m} = 0 \tag{6.64}$$

everywhere. In the supposed case, the second term is zero on a set of positive measure. Therefore, the first term is zero on a set of positive measure. Being analytic, this implies that the magnitudinally singular case requires $\lambda_v^\top g = 0$ everywhere. This means that the thrust vector is always perpendicular to the gravity vector. This is impossible in a soft, zero velocity landing. The magnitudinally singular case cannot occur on sets of positive measure. The magnitude of the of optimal control can be chosen to take only two values: ρ_1 and ρ_2. Though not shown, typical sequences are $\{\rho_2, \rho_1, \rho_2\}$, $\{\rho_1, \rho_2\}$, and $\{\rho_2\}$ depending on the initial conditions.

The numerical methods indirect shooting, direct shooting, and direct transcription can be used to solve the problem completely. The problem is nonconvex, however, and each of these methods is sensitive to the initial guess. Indirect shooting is implemented in the following subsection. A convex reformulation is then proposed suitable for direct transcription in the subsequent subsection.

MATLAB Implementation of Indirect Shooting

An instance of this problem is solved using indirect shooting and data specified in Problem 6.7. Data corresponds to that of a martian descent.

```
1  % Data
2  r0 = [2000; 100; 1500]; % m
3  v0 = [-100; -10; 0];    % m/s
4  m0 = 2100;              % kg
5  x0 = [r0; v0; m0];
6  g  = [0; 0; -3.71];     % m/s2
7  Isp  = 225;             % s
8  rho1 = 13e3;            % N
9  rho2 = 20e3;            % N
10 alpha = 1/(9.81*Isp);
```

The state and costate equations must be coded for integration in `ode45`. The states and costates are extracted on lines 29 and 30. The optimal thrust is computed by calling a subfunction `getControl` on line 33. The state equations are specified in lines 36-38 and the costate equations are specified in lines 41-43.

```
26 function xdot = ode(~,x,g,rho1,rho2,alpha)
27
28 % State/costate extraction
29 r  = x(1:3);  v  = x(4:6);   m  = x(7);
30 Lr = x(8:10); Lv = x(11:13); Lm = x(14);
31
32 % Optimal control
33 T = getControl(Lv,Lm,m,rho1,rho2,alpha);
```

```
34
35  % State equations
36  rdot = v;
37  vdot = g + T/m;
38  mdot = -alpha*norm(T);
39
40  % Costate equations
41  Lrdot = zeros(3,1);
42  Lvdot = -Lr;
43  Lmdot = Lv.'*T/m^2;
44
45  xdot = [rdot; vdot; mdot; Lrdot; Lvdot; Lmdot];
46  end
```

This function uses a subfunction `getControl` to calculate the thrust vector. This calculation, like the analysis, is decomposed into a directional part and magnitudinal part.

```
47  function [T,Tmag] = getControl(Lv,Lm,m,rho1,rho2,alpha)
48  That = Lv/norm(Lv);
49  f = norm(Lv) - m*Lm*alpha;
50  if f >= 0
51      Tmag = rho2;
52  else
53      Tmag = rho1;
54  end
55  T = That*Tmag;
56  end
```

With this in place, integration can be done provided the seven initial costates and final time are known. Indirect shooting is used to solve for these unknowns. The following function enforces the six terminal state constraints, one terminal costate constraint, and terminal Hamiltonian constraint. There are eight unknowns and eight boundary conditions. It is assumed in code that $\lambda_0 = 1$. It is left for the reader to explore the case with $\lambda_0 = 0$.

```
57  function F = shooting(unk,x0,g,rho1,rho2,alpha)
58
59  L = unk(1:7); tf = unk(8);
60  [~,x] = ode45(@ode,[0,tf],[x0;L],[],g,rho1,rho2,alpha);
61
62  rf  = x(end,1:3).';  vf  = x(end,4:6).';   mf  = x(end,7);
63  Lrf = x(end,8:10).'; Lvf = x(end,11:13).'; Lmf = x(end,14);
64  Tf  = getControl(Lvf,Lmf,mf,rho1,rho2,alpha);
65  Hf  = Lrf.'*vf + Lvf.'*(g+Tf/mf) - Lmf*alpha*norm(Tf);
66  F   = [rf; vf; Lmf-1; Hf];
```

MATLAB's built-in Newton solver `fsolve` is now used to find initial costates and final time so that all terminal boundary conditions are satisfied.

```
11  % Shooting method
12  ops = optimoptions('fsolve','Display','iter-detailed');
```

```
Lguess = [-0.0167;0.0002;-0.0503;-0.0537;0.0149;-0.3644;0.9011];
tguess = 35;
guess  = [Lguess; tguess];
sol = fsolve(@shooting,guess,ops,x0,g,rho1,rho2,alpha);

% Solution Extraction
L = sol(1:7); tf = sol(8);
[t,x] = ode45(@ode,[0,tf],[x0;L],[],g,rho1,rho2,alpha);
r  = x(:,1:3);  v  = x(:,4:6);   m  = x(:,7);
Lr = x(:,8:10); Lv = x(:,11:13); Lm = x(:,14);
for i = 1:length(t)
  [T(i,:),Tmag(i)]=getControl(Lv(i,:),Lm(i,:),m(i),rho1,rho2,alpha);
end
```

Running the code solves the problem giving a final mass of 1,883.7 kg and final time of 32.4 seconds. Figure 6.2 shows the thrust magnitude as a function of time. The magnitude is bounded by ρ_1 and ρ_2 and in sequence $\{\rho_1, \rho_2\}$.

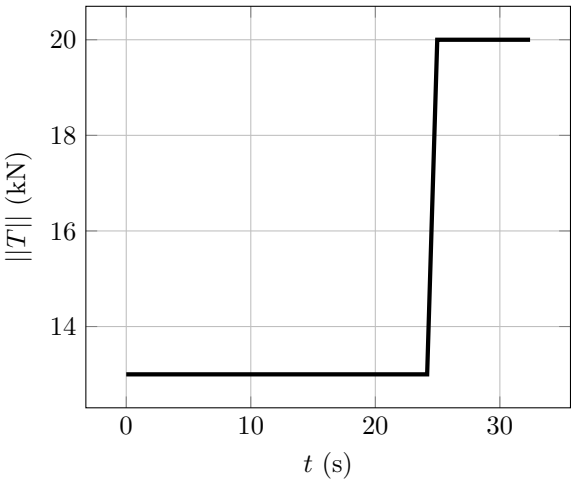


Figure 6.2: Thrust magnitude as a function of time.

The indirect shooting method is sensitive to the initial guess and requires a good initial guess. The method is not well-suited for an onboard implementation. For this reason, a convex relaxation and approximation are proposed so that the problem can be discretized and solved via direct transcription using efficient convex solvers. This is the topic of the following subsection.

Convex Relaxation and Approximation

It is now shown that there is a convex problem that closely approximates the problem of interest. First, fuel consumption is equal to the integral of thrust magnitude because

$$m = m_0 - \alpha \int_0^t ||T||\, d\tau. \tag{6.65}$$

The objective function can be changed to minimize the integral. Consider the relaxed problem with two controls T and Γ.[24]

$$\begin{aligned}
\text{minimize} \quad & \int_0^{t_f} \Gamma \, dt \\
\text{subject to} \quad & \dot{r} = v, && r(0) = r_0, \quad r_f = 0 \\
& \dot{v} = g + T/m, && v(0) = v_0, \quad v_f = 0 \\
& \dot{m} = -\alpha\Gamma, && m(0) = m_0 \\
& \|T\| \leq \Gamma, && 0 < \rho_1 \leq \Gamma \leq \rho_2
\end{aligned} \tag{6.66}$$

This problem is nonconvex because the dynamics remain nonlinear. The control constraints, however, are convex. Any feasible solution of the original problem is feasible in this problem but the converse is not true. The introduction of Γ lifts the control space into a larger dimension as illustrated in Figure 6.3.

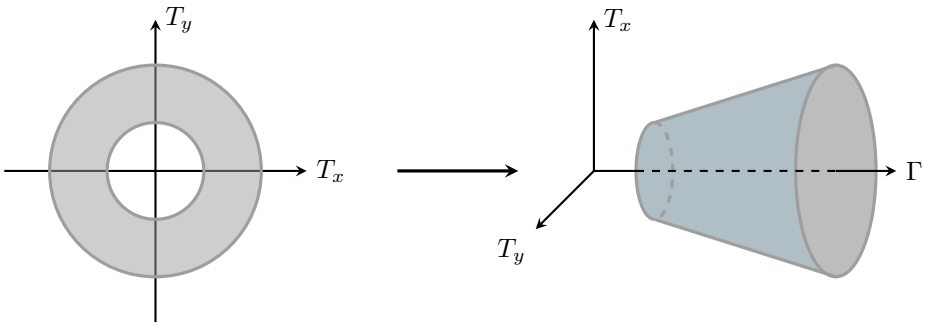


Figure 6.3: Control set lifting.

The interior of the cone on the right side contains feasible points in the relaxed problem, but these interior points do not map to feasible points on the left side. It is now shown that these problematic points cannot be part of the optimal solution.

The solution procedure is to first form the Hamiltonian and endpoint functions.

$$H = \lambda_0\Gamma + \lambda_r^\top v + \lambda_v^\top (g + T/m) + \lambda_m(-\alpha\Gamma) \tag{6.67}$$

$$G = \nu_r^\top r_f + \nu_v^\top v_f \tag{6.68}$$

The costate, Hamiltonian, and transversality conditions generate the following equations.

$$\dot{\lambda}_r = 0, \qquad \lambda_{rf} = \nu_r \tag{6.69}$$

$$\dot{\lambda}_v = -\lambda_r, \qquad \lambda_{vf} = \nu_v \tag{6.70}$$

$$\dot{\lambda}_m = \lambda_v^\top T/m^2, \quad \lambda_{mf} = 0 \tag{6.71}$$

$$\dot{H} = 0, \qquad H_f = 0 \tag{6.72}$$

[24] Açıkmeşe and Ploen, *Convex Programming Approach to Powered Descent Guidance for Mars Landing*, 2007.

It is immediately deduced that λ_r and H are constants. The costate λ_v is constant or varying linearly with time. The pointwise minimum condition yields, in part,

$$T \overset{\text{ae}}{\in} \underset{\|\omega\| \leq \Gamma}{\operatorname{argmin}} \lambda_v^\top \omega \quad \Longrightarrow \quad T \overset{\text{ae}}{=} \begin{cases} -\dfrac{\lambda_v}{||\lambda_v||}\Gamma, & \lambda_v \neq 0 \\ \text{singular}, & \lambda_v = 0 \end{cases}. \tag{6.73}$$

The singular case is negligible if it occurs on a set of zero measure. Suppose, on the other hand, it occurs on a set of positive measure. Because λ_v is a linear (analytic) function of time, it is zero everywhere. Having $\lambda_v = 0$ everywhere implies $\lambda_r = 0$ everywhere. Inspection of the costate and transversality conditions reveals that $\lambda_m = 0$ everywhere. The Hamiltonian being zero implies that $\lambda_0 = 0$, which violates non-triviality. The singular case cannot occur on sets of positive measure. Consequently, $||T|| = \Gamma$. It has been proven that optimal solutions of the relaxed problem are indeed feasible in the original problem. Moreover, because it is a relaxation, optimal solutions of the relaxed problem are optimal in the original problem.

Nonetheless, the relaxed problem is still nonconvex because of the nonlinear dynamics. Introduce the following variable transformations, which define two new control variables u and σ.

$$u = \frac{T}{m}, \quad \sigma = \frac{\Gamma}{m} \tag{6.74}$$

The equations of motion are then

$$\dot{r} = v, \tag{6.75}$$

$$\dot{v} = g + u, \tag{6.76}$$

$$\dot{m} = -\alpha m \sigma. \tag{6.77}$$

The first two equations are linear in the states and new controls. The third remains nonlinear. Rearranging the third gives

$$\frac{\dot{m}}{m} = -\alpha\sigma. \tag{6.78}$$

Integrating yields the mass as a function of time.

$$m = m_0 \exp\left[-\alpha \int_0^t \sigma(\tau)\, d\tau\right] \tag{6.79}$$

It is evident that minimizing fuel consumption is equivalent to minimizing

$$\int_0^{t_f} \sigma\, dt. \tag{6.80}$$

Furthermore, in terms of the new control variables, the relaxed constraint is $||u|| \leq \sigma$. The third state equation can be rigorously linearized through another variable transformation. Let $z = \ln m$ such that

$$\dot{z} = \frac{\dot{m}}{m} = -\alpha\sigma. \tag{6.81}$$

All state equations are now linear.

Variable substitutions in the upper thrust constraint yield

$$\Gamma \leq \rho_2 \quad \longrightarrow \quad m\sigma \leq \rho_2 \quad \longrightarrow \quad \sigma \leq \rho_2 e^{-z}. \tag{6.82}$$

The transformed constraint is nonconvex. An approximation is now introduced by replacing e^{-z} locally by a linear approximation. It is then shown that this approximation is conservative thereby ensuring that the original constraint $||T|| \leq \rho_2$ is in fact satisfied. A linear approximation is given by

$$\rho_2 e^{-z} \approx \rho_2 e^{-\tilde{z}} - \rho_2 e^{-\tilde{z}} (z - \tilde{z}), \tag{6.83}$$

where $\tilde{z}$ is to be determined. The conservative nature of the linear approximation is easily shown using the mean value theorem, which states there is a $\hat{z}$ such that equality holds, i.e.,

$$\rho_2 e^{-z} = \rho_2 e^{-\tilde{z}} - \rho_2 e^{-\tilde{z}}(z - \tilde{z}) + \frac{1}{2}\rho_2 e^{-\hat{z}}(z - \tilde{z})^2. \tag{6.84}$$

Because the last term is non-negative, it is concluded that

$$\rho_2 e^{-\tilde{z}} - \rho_2 e^{-\tilde{z}}(z - \tilde{z}) \leq \rho_2 e^{-z}. \tag{6.85}$$

As for $\tilde{z}$, a good selection is

$$\tilde{z} = \begin{cases} \ln(m_0 - \alpha\rho_2 t), & m_0 - \alpha\rho_2 t \geq m_{dry} \\ \ln(m_{dry}), & \text{otherwise} \end{cases} \tag{6.86}$$

where m_{dry} is the dry (structural) mass of the spacecraft. With this definition of $\tilde{z}$, it is known that $\tilde{z} \leq z$ at any time because $\tilde{z}$ is being calculated assuming maximum thrust force.

Variable substitutions in the lower thrust constraint yield

$$\rho_1 \leq \Gamma \quad \longrightarrow \quad \rho_1 \leq m\sigma \quad \longrightarrow \quad \rho_1 e^{-z} \leq \sigma. \tag{6.87}$$

The transformed constraint is convex. It is not, however, a second-order cone constraint suitable for an efficient numerical solver. To make it one, a second-order Taylor approximation about $\tilde{z}$ is used.

$$\rho_1 e^{-\tilde{z}} \left[1 - (z - \tilde{z}) + \frac{1}{2}(z - \tilde{z})^2\right] \leq \sigma \tag{6.88}$$

Using the mean value theorem, it can again be shown that this is a conservative approximation. To summarize, the final approximate convex problem is the following.

$$\begin{aligned}
\text{minimize} \quad & \int_0^{t_f} \sigma \, dt \\
\text{subject to} \quad & \dot{r} = v, \qquad r(0) = r_0, \quad r_f = 0 \\
& \dot{v} = g + u, \quad v(0) = v_0, \quad v_f = 0 \\
& \dot{z} = -\alpha\sigma, \quad z(0) = z_0 \\
& \|u\| \leq \sigma \\
& \sigma \leq \rho_2 e^{-\tilde{z}} - \rho_2 e^{-\tilde{z}}(z - \tilde{z}) \\
& \sigma \geq \rho_1 e^{-\tilde{z}} \left[1 - (z - \tilde{z}) + \tfrac{1}{2}(z - \tilde{z})^2\right] \\
& \tilde{z} = \begin{cases} \ln(m_0 - \alpha\rho_2 t), & m_0 - \alpha\rho_2 t \geq m_{dry} \\ \ln(m_{dry}), & \text{otherwise} \end{cases}
\end{aligned} \tag{6.89}$$

This problem is convex for fixed final times. The discretized, fixed final time problem is a second-order cone program suitable for efficient solvers. Through direct transcription and a one-dimensional search for the optimal time, the problem can be solved efficiently. This is left as a problem for the reader. The use of convex optimization and other computational techniques for optimal spacecraft guidance is an active research area.[25]

[25] Lu, *Introducing Computational Guidance and Control*, 2017.

6.4 Chapter Problems

Problem 6.1. In the setting of Section 6.1, solve for the optimal control as a function of the current state and other problem data. Simulate the optimal earth descent trajectory using the following data.

```
r0 = 500;       % m
v0 = -10;       % m/s
m0 = 1000;      % kg
Tmax = 20e3;    % N
Isp  = 300;     % s
```

Problem 6.2. Consider a variation of the optimal control problem in Eq. (6.7) with control constraint $0 < T_{min} \leq T \leq T_{max}$. Determine all possible optimal thrust sequences.

Problem 6.3. Simulate a nominal lunar descent trajectory using E-guidance and the following data. States are ordered as downrange, crossrange, and altitude.

```
r0 = [500e3; 100e3; 50e3];  % ft
v0 = [-3e3; 0; 0];          % ft/s
tf = 350;                   % s
```

Problem 6.4. Building upon the previous problem, incorporate into the simulation navigation errors, actuator dynamics, and disturbances.

Problem 6.5. Simulate a nominal lunar descent trajectory using Apollo descent guidance and the following data. States are ordered as downrange, crossrange, and altitude.

```
r0 = [500e3; 100e3; 50e3];  % ft
v0 = [-3e3; 0; 0];          % ft/s
tf = 350;                   % s
```

Problem 6.6. Building upon the previous problem, incorporate into the simulation navigation errors, actuator dynamics, and disturbances.

Problem 6.7. Simulate a nominal martian descent trajectory by solving the optimal control problem in Eq. (6.89). Solve the problem using direct transcription and a one-dimensional search for the optimal time. Use the following data. States are ordered as downrange, crossrange, and altitude.

```
r0 = [2000; 100; 1500]; % m
v0 = [-100; -10; 0];    % m/s
m0 = 2100;              % kg
mdry = 1500;            % kg
Isp  = 225;             % s
rho1 = 13e3;            % N
rho2 = 20e3;            % N
```

Recall that direct transcription was implemented using a standard discretization in Section 3.6 and a Chebyshev discretization in Section 4.8.

Problem 6.8. Building upon the previous problem, use discrete LQR to follow the nominal martian descent trajectory. Incorporate into the simulation navigation errors and disturbances. Recall that trajectory following using LQR was implemented in Section 4.6.

Chapter 7

Ascent Guidance

Chapter Learning Objectives

1. Develop and implement singular solutions to Goddard's vertical launch problem.
2. Develop and implement a guidance law based on minimum time orbit injection.
3. Develop and implement Q-guidance using a flat planet model.
4. Develop Q-guidance using a spherical planet model.

One of the earliest problems in ascent guidance was posed by Goddard in 1919. The problem was to determine the thrust profile to achieve maximum altitude. Optimal control theory, which was developed 40 years later, provides a conclusive answer. The optimal thrust profile is a sequence of maximum thrust, singular thrust, and zero thrust. Surprisingly, when using the standard quadratic drag model, the solution is not to apply maximum thrust until burnout. From a mathematical perspective, it is interesting that singular solutions are part of the optimal solution, and Goddard's problem is the only applied problem in this book for which the singular case can occur.

The minimum time orbit injection problem is then studied. This problem and its variations have led to well-known ascent guidance laws including the iterative guidance mode (IGM) for Saturn V and powered explicit guidance (PEG) for Shuttle. Under a constant thrust acceleration assumption, it is shown that the optimal thrust angle behaves according to a linear tangent law. A change of variables facilitates analytic integration of the state equations so that the problem is reduced to an algebraic problem in three physical variables.

Finally, Q-guidance is developed – first using a flat planet model and then using a spherical planet model. The development is dynamics-based rather than optimization-based. Nonetheless, Q-guidance is fuel optimal in some settings. Q-guidance is also called cross-product steering because the control acceleration is chosen so that a certain cross-product is zero. Originally developed for missile guidance applications, Q-guidance provides a flexible framework suitable for ascent applications. Because of its military applications and the intellectual challenge of computing Q, Q-guidance was originally classified. It is now in the public domain.

7.1 Goddard's Problem

Consider the one-dimensional vertical launch of a rocket.[26] At the initial time, the vehicle is on the ground with zero velocity. The thrust is bounded in magnitude. The goal is to reach maximum altitude. In this one-dimensional setting, the equations of motion are consistent with the flat planet model in Eq. (2.10), though a state-dependent drag function $D(r, v)$ is also included, and the mass dynamics in Eq. (2.44). Surprisingly, the optimal solution is not to use maximum thrust until burnout. The optimal solution includes a period of maximum thrust followed by a period of intermediate thrust. The intermediate thrust corresponds to a singular solution. The optimal control problem is the following.

$$\begin{aligned}
&\text{maximize} && r_f && \\
&\text{subject to} && \dot{r} = v, && r(0) = 0 \\
&&& \dot{v} = g + T/m - D(r,v)/m, && v(0) = 0 \\
&&& \dot{m} = -\alpha T, && m(0) = m_0, \quad m_f = \bar{m}_f \\
&&& 0 \leq T \leq T_{max} &&
\end{aligned} \tag{7.1}$$

The solution procedure is to first form the Hamiltonian and endpoint functions.

$$H = \lambda_r v + \lambda_v \left(g + T/m - D/m\right) + \lambda_m(-\alpha T) \tag{7.2}$$

$$G = \lambda_0 r_f + \nu \left(m_f - \bar{m}_f\right) \tag{7.3}$$

The costate, Hamiltonian, and transversality conditions generate the following equations.

$$\dot{\lambda}_r = \frac{\lambda_v}{m}\frac{\partial D}{\partial r}, \qquad \lambda_{rf} = \lambda_0 \tag{7.4}$$

$$\dot{\lambda}_v = -\lambda_r + \frac{\lambda_v}{m}\frac{\partial D}{\partial v}, \quad \lambda_{vf} = 0 \tag{7.5}$$

$$\dot{\lambda}_m = \frac{\lambda_v T}{m^2} - \frac{\lambda_v D}{m^2}, \quad \lambda_{mf} = \nu \tag{7.6}$$

$$\dot{H} = 0, \qquad H_f = 0 \tag{7.7}$$

It is immediately deduced that H is constant and zero. The pointwise maximum condition yields

$$T \overset{\text{ae}}{\in} \underset{0 \leq \omega \leq T_{max}}{\operatorname{argmax}} \left(\frac{\lambda_v}{m} - \alpha\lambda_m\right)\omega \quad \Longrightarrow \quad T \overset{\text{ae}}{=} \begin{cases} T_{max}, & \lambda_v - m\alpha\lambda_m > 0 \\ 0, & \lambda_v - m\alpha\lambda_m < 0 \\ \text{singular}, & \lambda_v - m\alpha\lambda_m = 0 \end{cases}. \tag{7.8}$$

The singular case occurs when the switching function $f = \lambda_v - m\alpha\lambda_m$ is zero. The singular case is negligible if it occurs on a set of zero measure. Suppose, on the other hand, it occurs on a set of positive measure. The function f is absolutely continuous. Being zero on a set of positive measure does not imply being zero everywhere. A deeper analysis is required.

[26] Garfinkel, *A Solution of the Goddard Problem*, 1963.

Let s be a point in the supposed set such that $f(s) = 0$. A formula for f is obtained by integrating its derivative.

$$f(t) = f(s) + \int_s^t h(\tau) - \alpha f(\tau)\frac{T(\tau)}{m(\tau)}\,d\tau \tag{7.9}$$

The function h is

$$h = -\lambda_r + \frac{\lambda_v}{m}\frac{\partial D}{\partial v} + \alpha\frac{\lambda_v}{m}D. \tag{7.10}$$

The $f(s)$ term is zero. The integrand has a value of $h(s)$ at s. If $h(s)$ is non-zero, the integrand has the sign of $h(s)$ on $[s, t]$ for t sufficiently close to s. This is true because f, h, and m are absolutely continuous and T is bounded. This implies that the zeros of the switching function are isolated and the set of zeros has zero measure. Therefore, having f be zero on a set of positive measure implies $h(s) = 0$. Furthermore, $H(s) = 0$. In matrix form, having $f(s) = h(s) = H(s) = 0$ means

$$\left[\begin{array}{c|c|c} \frac{1}{m}(T-D)+g & v & -\alpha T \\ \frac{1}{\alpha} & 0 & -m \\ \frac{\partial D}{\partial v}+\alpha D & -m & 0 \end{array}\right]\left[\begin{array}{c} \lambda_v \\ \lambda_r \\ \lambda_m \end{array}\right](s) = \left[\begin{array}{c} 0 \\ 0 \\ 0 \end{array}\right]. \tag{7.11}$$

If the matrix is full rank at s, then $\lambda_r(s) = \lambda_v(s) = \lambda_m(s) = 0$. The costate equations are homogenous. If the costates are all zero somewhere, then they are all zero everywhere. At the final time $\lambda_{rf} = \lambda_0$. This violates non-triviality. The matrix must have rank less than three at s. Equivalently, the absolutely continuous determinant function Δ given by

$$\Delta = D - mg - \alpha v D - v\frac{\partial D}{\partial v} \tag{7.12}$$

must equal zero at s. A formula for Δ is obtained by integrating its derivative.

$$\Delta(t) = \Delta(s) + \int_s^t \dot{\Delta}(\tau)\,d\tau \tag{7.13}$$

The $\Delta(s)$ term is zero. The integrand has a value of $\dot{\Delta}(s)$ at s. If $\dot{\Delta}(s)$ is non-zero, the integrand has the sign of $\dot{\Delta}(s)$ on $[s, t]$ for t sufficiently close to s. This is true again because of continuity and boundedness arguments. This implies that the zeros of Δ are isolated and the set of zeros has zero measure. Therefore, having Δ be zero on a set of positive measure implies $\dot{\Delta}(s) = 0$. Upon defining the quantities

$$\phi = \alpha D + \frac{\partial^2 D}{\partial v^2}v + \alpha\frac{\partial D}{\partial v}v, \tag{7.14}$$

$$\theta = \frac{\partial D}{\partial r}v - \alpha\frac{\partial D}{\partial r}v^2 - \frac{\partial^2 D}{\partial v \partial r}v^2 + \left(\frac{D}{m} - g\right)\phi, \tag{7.15}$$

$$\psi = \frac{\phi}{m} - \alpha g, \tag{7.16}$$

it can be shown that $\dot{\Delta}$ depends linearly on T as $\dot{\Delta} = \theta - \psi T$. By choosing $T = \theta/\psi$ at s, the value of $\dot{\Delta}$ at s is zero. In conclusion, at every point s in the supposed set

of positive measure, this choice of T guarantees that all of the conditions required for singularity hold. This is the singular control.

In practice, solutions to the problem typically have the sequence max thrust, singular thrust, and finally zero thrust. To implement this, apply maximum thrust at the initial time. Monitor the determinant and switch to singular thrust when it is zero. Continue applying singular thrust until the terminal mass constraint is satisfied. At this point, coast to maximum altitude. Implementation is left as a problem for the reader.

7.2 Minimum Time Orbit Injection

A planar ascent problem is now considered assuming constant thrust acceleration $\tau = T/m$. The control variable is the thrust angle θ, and the dynamics are consistent with the flat planet model in Eq. (2.10). The horizontal range x is free. The range rate u and altitude rate v at the final time are fixed. The problem is to reach orbit in minimum time. This formulation is consistent with that used to develop the iterative guidance mode (IGM) for Saturn V ascent guidance.[27]

$$\begin{array}{lllll} \text{minimize} & t_f & & & \\ \text{subject to} & \dot{x} = u & x(0) = 0 & & \\ & \dot{y} = v, & y(0) = 0, & y_f = \bar{y}_f & \\ & \dot{u} = \tau\cos\theta, & u(0) = 0, & u_f = \bar{u}_f & \\ & \dot{v} = g + \tau\sin\theta, & v(0) = 0, & v_f = 0 & \end{array} \tag{7.17}$$

The solution procedure is to first form the Hamiltonian and endpoint functions.

$$H = \lambda_x u + \lambda_y v + \lambda_u \tau\cos\theta + \lambda_v\left(g + \tau\sin\theta\right) \tag{7.18}$$

$$G = \lambda_0 t_f + \nu_y(y_f - \bar{y}_f) + \nu_u(u_f - \bar{u}_f) + \nu_v v_f \tag{7.19}$$

The costate and Hamiltonian differential equations are the following.

$$\dot{\lambda}_x = 0 \quad \Longrightarrow \quad \lambda_x \text{ constant} \tag{7.20}$$

$$\dot{\lambda}_y = 0 \quad \Longrightarrow \quad \lambda_y \text{ constant} \tag{7.21}$$

$$\dot{\lambda}_u = -\lambda_x \quad \Longrightarrow \quad \lambda_u = -\lambda_x\left(t - t_f\right) + \lambda_{uf} \tag{7.22}$$

$$\dot{\lambda}_v = -\lambda_y \quad \Longrightarrow \quad \lambda_v = -\lambda_y\left(t - t_f\right) + \lambda_{vf} \tag{7.23}$$

$$\dot{H} = 0 \quad \Longrightarrow \quad H \text{ constant} \tag{7.24}$$

The transversality conditions specify terminal boundary conditions on the costates and Hamiltonian.

$$\lambda_{xf} = 0, \quad \lambda_{uf} = \nu_u \tag{7.25}$$

$$\lambda_{yf} = \nu_y, \quad \lambda_{vf} = \nu_v \tag{7.26}$$

$$H_f = -\lambda_0 \tag{7.27}$$

[27] Chandler and Smith, *Development of the Iterative Guidance Mode with its Application to Various Vehicles and Missions*, 1967.

Using the costate and transversality conditions together gives

$$\lambda_x = 0, \tag{7.28}$$
$$\lambda_y = \nu_y, \tag{7.29}$$
$$\lambda_u = \nu_u, \tag{7.30}$$
$$\lambda_v = -\nu_y\,(t - t_f) + \nu_v. \tag{7.31}$$

That is, λ_x is constant and zero, λ_y is constant, λ_u is constant, and λ_v is constant or varying linearly with time. The pointwise minimum condition yields

$$\theta \overset{\text{ae}}{\in} \underset{\omega}{\operatorname{argmin}}\ \lambda_u \tau \cos\omega + \lambda_v \tau \sin\omega \quad \Longrightarrow \quad -\lambda_u \sin\theta + \lambda_v \cos\theta \overset{\text{ae}}{=} 0, \tag{7.32}$$

which determines the thrust angle θ except when λ_u and λ_v are both zero. This singular case is negligible if it occurs on a set of zero measure. Suppose, on the other hand, it occurs on a set of positive measure, i.e., λ_u and λ_v are both zero on a set of positive measure. Because they are linear (analytic) functions of time, they are zero everywhere. This implies that all the costates are zero. The Hamiltonian is zero and the H_f condition implies $\lambda_0 = 0$, which violates non-triviality. The singular case cannot occur on sets of positive measure. It is deduced that

$$\tan\theta = \frac{\lambda_v}{\lambda_u} = \frac{-\nu_y(t - t_f) + \nu_v}{\nu_u}. \tag{7.33}$$

Observe that $\tan\theta$ is a linear function of time such that it can be written differently in terms of unknowns θ_0 and c.

$$\tan\theta = \tan\theta_0 - ct \tag{7.34}$$

This is called the linear tangent law. Using θ as the independent variable, the state equations can be integrated to the final point in an analytic fashion. Differentiating the linear tangent law with respect to time gives

$$\sec^2\theta \frac{d\theta}{dt} = -c \quad \Longrightarrow \quad \frac{dt}{d\theta} = -\frac{\sec^2\theta}{c}. \tag{7.35}$$

It follows that

$$\frac{du}{d\theta} = \frac{du}{dt}\frac{dt}{d\theta} = -\frac{\tau}{c}\sec\theta. \tag{7.36}$$

Integrating both sides with $u_0 = 0$ gives

$$u_f = -\frac{\tau}{c}\int_{\theta_0}^{\theta_f} \sec\phi\, d\phi \tag{7.37a}$$
$$= -\frac{\tau}{c}\ln|\tan\theta + \sec\theta|_{\theta_0}^{\theta_f} \tag{7.37b}$$
$$= -\frac{\tau}{c}\ln|\tan\theta_f + \sec\theta_f| + \frac{\tau}{c}\ln|\tan\theta_0 + \sec\theta_0| \tag{7.37c}$$
$$= +\frac{\tau}{c}\log\left(\frac{\tan\theta_0 + \sec\theta_0}{\tan\theta_f + \sec\theta_f}\right). \tag{7.37d}$$

Absolute values in the last equation are removed recognizing that θ will be in the first quadrant for the ascent problem.

Repeating this process for each state yields the following integrated state equations.

$$u_f = \frac{\tau}{c} \log\left(\frac{\tan\theta_0 + \sec\theta_0}{\tan\theta_f + \sec\theta_f}\right) \tag{7.38a}$$

$$v_f = \frac{\tau}{c}(\sec\theta_0 - \sec\theta_f) + gt_f \tag{7.38b}$$

$$x_f = \frac{\tau}{c^2}\left[\sec\theta_0 - \sec\theta_f - \tan\theta_f \log\left(\frac{\tan\theta_0 + \sec\theta_0}{\tan\theta_f + \sec\theta_f}\right)\right] \tag{7.38c}$$

$$y_f = \frac{\tau}{2c^2}\left[(\tan\theta_0 - \tan\theta_f)\sec\theta_0 - (\sec\theta_0 - \sec\theta_f)\tan\theta_f \right. \tag{7.38d}$$

$$\left. - \log\left(\frac{\tan\theta_0 + \sec\theta_0}{\tan\theta_f + \sec\theta_f}\right)\right] + \frac{1}{2}gt_f^2$$

At the final time, $\tan\theta_f = \tan\theta_0 - ct_f$. As such, there are three unknowns in the above equations. These are θ_0, θ_f, and t_f. There are also three boundary conditions: y_f, u_f, and v_f. The three equations can be solved iteratively. Numerical implementation is left as a problem for the reader.

Another popular ascent guidance law is powered explicit guidance (PEG).[28] Though the reader has by now developed the skills needed to derive PEG, it is lengthy and beyond the scope of this book. A modern exposition of PEG-like guidance is available.[29]

7.3 Q-Guidance

Another guidance technique from the 1950s called Q-guidance is now discussed. Q-guidance is also called cross-product steering. The development is dynamics-based rather than optimization-based; however, Q-guidance is fuel optimal in some settings.[30] Q-guidance was first developed for missiles but is applicable in various settings including ascent.

To begin, the flat planet model in Eq. (2.10) is assumed. At the current time t, the position is r. The goal is to reach a position r_f at some later time t_f. Provided the velocity associated with r at time t, denoted v_r, satisfies

$$r_f = r + (t_f - t)v_r + \tfrac{1}{2}(t_f - t)^2 g, \tag{7.39}$$

then the vehicle can coast to the desired final point. This can be verified by simply integrating the flat planet equations of motion with zero control. Solving for v_r gives

$$v_r = \frac{1}{t_f - t}\left[r_f - r - \frac{1}{2}(t_f - t)^2 g\right]. \tag{7.40}$$

If, at time t, the vehicle's actual velocity v is not equal to v_r, i.e., the vehicle is not on a trajectory that coasts to the desired final point, then control is required. In yet another form, v_r must satisfy the equation

$$(t_f - t)v_r = r_f - r - \tfrac{1}{2}(t_f - t)^2 g. \tag{7.41}$$

[28] Jaggers, *An Explicit Solution to the Exoatmospheric Powered Flight Guidance and Trajectory Optimization Problem for Rocket Propelled Vehicles*, 1977.

[29] Hull and Harris, *Optimal Solutions for Quasiplanar Ascent over a Spherical Moon*, 2012.

[30] Battin, *An Introduction to the Mathematics and Methods of Astrodynamics*, 1999.

Differentiating and solving for $\dot{v}_r$ gives

$$\dot{v}_r = \frac{v_r - v}{t_f - t} + g. \tag{7.42}$$

The difference between the needed velocity v_r and the actual velocity v is the velocity-to-be-gained v_g.

$$v_g = v_r - v \tag{7.43}$$

Differentiating with respect to time and recognizing that $\dot{v} = g + u$ gives

$$\dot{v}_g = \frac{1}{t_f - t} v_g - u. \tag{7.44}$$

Historically, the quantity pre-multiplying v_g is called Q, hence the name Q-guidance. To reach the desired trajectory, the control acceleration should be chosen to drive v_g to zero. To explore this further, see that

$$\frac{d}{dt}(v_g \cdot v_g) = \frac{d}{dt}\left(\|v_g\|^2\right) = 2\dot{v}_g \cdot v_g \tag{7.45a}$$

$$= \frac{2}{t_f - t}\|v_g\|^2 - 2u \cdot v_g. \tag{7.45b}$$

Since $\|v_g\|^2$ is non-negative, its derivative must be made negative to drive it to zero. To do this, choose u parallel to v_g and as large as possible in magnitude. If u_{max} is the upper bound on acceleration at time t,

$$u = \frac{v_g}{\|v_g\|} u_{max}. \tag{7.46}$$

It may not be possible to make the derivative negative if u_{max} is not sufficiently large. Also, Q is simply a scaled identity matrix in the current setup, which is a very special situation. This algorithm was implemented in Section 1.2.

ℹ Observe that this choice of u causes $\dot{v}_g \times v_g$ to be zero since

$$\dot{v}_g \times v_g = \left(\frac{1}{t_1 - t} v_g - u\right) \times v_g \tag{7.47a}$$

$$= \frac{1}{t_1 - t} v_g \times v_g - u \times v_g \tag{7.47b}$$

$$= 0. \tag{7.47c}$$

It is this cross-product property that is important and generalizes to spherical bodies. This is why Q-guidance is also called cross-product steering.

The more general case in which the gravitational acceleration is a function of r is now considered. The equation of motion is

$$\dot{v} = g(r) + u. \tag{7.48}$$

The velocity-to-be-gained is again

$$v_g = v_r - v, \tag{7.49}$$

and its time derivative is

$$\dot{v}_g = \dot{v}_r - g(r) - u. \tag{7.50}$$

Because v_r depends on t and r, the chain rule gives

$$\begin{aligned}
\frac{dv_r}{dt} &= \frac{\partial v_r}{\partial t} + \frac{\partial v_r}{\partial r}\frac{dr}{dt} && (7.51a)\\
&= \frac{\partial v_r}{\partial t} + \frac{\partial v_r}{\partial r} v && (7.51b)\\
&= \frac{\partial v_r}{\partial t} + \frac{\partial v_r}{\partial r}(v_r - v_g) && (7.51c)\\
&= \underbrace{\frac{\partial v_r}{\partial t} + \frac{\partial v_r}{\partial r} v_r}_{= \frac{dv_r}{dt} = g(r)} - \frac{\partial v_r}{\partial r} v_g && (7.51d)\\
&= g(r) - \frac{\partial v_r}{\partial r} v_g. && (7.51e)
\end{aligned}$$

since it is a coasting trajectory

Substituting this back into the $\dot{v}_g$ equation with $Q = \frac{\partial v_r}{\partial r}$ gives

$$\dot{v}_g = -Q v_g - u. \tag{7.52}$$

The control acceleration should now be chosen so that $\dot{v}_g \times v_g = 0$. To facilitate the process of doing so, define $p = -Q v_g$ which means $\dot{v}_g = p - u$. Cross-product steering is then to choose u such that

$$\begin{aligned}
& (p - u) \times v_g = p \times v_g - u \times v_g = 0, && (7.53a)\\
\implies \quad & u \times v_g = p \times v_g. && (7.53b)
\end{aligned}$$

Again, the vehicle must have sufficient control acceleration to achieve this equality. Vector post-multiplication by v_g and using the identity $(a \times b) \times c = (a \cdot c)b - (b \cdot c)a$ yields

$$(u \cdot v_g)\, v_g - \|v_g\|^2 u = (p \cdot v_g)\, v_g - \|v_g\|^2 p. \tag{7.54}$$

Dividing by $\|v_g\|^2$ and rearranging,

$$u = p + \frac{1}{\|v_g\|^2}(u \cdot v_g - p \cdot v_g) v_g. \tag{7.55}$$

Defining $\hat{\imath}_g = \frac{v_g}{\|v_g\|}$ and $q = u \cdot \hat{\imath}_g$,

$$u = p + (q - p \cdot \hat{\imath}_g)\hat{\imath}_g. \tag{7.56}$$

Squaring both sides gives

$$\begin{aligned}
& u^\top u = p^\top p + 2\,(q - p \cdot \hat{\imath}_g)\, p^\top \hat{\imath}_g + (q - p \cdot \hat{\imath}_g)^2 && (7.57a)\\
\implies \quad & \|u\|^2 = \|p\|^2 + 2qp \cdot \hat{\imath}_g - 2\,(p \cdot \hat{\imath}_g)^2 + q^2 - 2qp \cdot \hat{\imath}_g + (p \cdot \hat{\imath}_g)^2 && (7.57b)\\
& \quad = \|p\|^2 + q^2 - (p \cdot \hat{\imath}_g)^2. && (7.57c)
\end{aligned}$$

Using the above to solve for q gives

$$q = \left[\|u\|^2 - \|p\|^2 + (p \cdot \hat{\imath}_g)^2\right]^{\frac{1}{2}}. \tag{7.58}$$

From here, it is again evident that $\|u\|$ must be sufficiently large for q to be real. It is common to know the magnitude of control acceleration available at a given time. Thus, Eq. (7.58) is used to calculate q and then Eq. (7.56) to calculate u. By doing this continuously, or periodically in guidance, v_g is driven to zero.

ℹ When the available control acceleration is not sufficiently large, it is chosen parallel to v_g and as large as possible in magnitude, i.e.,

$$u = \frac{v_g}{\|v_g\|} u_{max}. \tag{7.59}$$

The calculation of v_r and Q depends on the target orbit and can be quite involved. Those quantities for a circular target orbit and elliptical target orbit are presented now. In both, the matrix $S(x)$ is the cross-product matrix mapping elements of the three-dimensional x to

$$S(x) = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix}. \tag{7.60}$$

To target a circular orbit with an orbit plane defined by $\hat{\imath}_h$,

$$v_r = S(\hat{\imath}_h) r \sqrt{\frac{\mu}{\|r\|^3}}, \tag{7.61}$$

$$Q = \sqrt{\frac{\mu}{\|r\|^3}} S(\hat{\imath}_h) \left(I - \frac{3}{2} \hat{\imath}_r \hat{\imath}_r^\top \right). \tag{7.62}$$

To target an elliptical orbit with parameter ℓ, eccentricity e, and orbit plane $\hat{\imath}_h$,

$$v_r = \pm \left\{ \frac{\mu}{\ell} \left[e^2 - \left(\frac{\ell}{\|r\|} - 1 \right)^2 \right] \right\}^{\frac{1}{2}} \hat{\imath}_r + \frac{\sqrt{\mu\ell}}{\|r\|} \hat{\imath}_h \times \hat{\imath}_r, \tag{7.63}$$

$$\begin{aligned} Q = {} & \pm \frac{\sqrt{\mu\ell}}{\|r\|^2} \left[\left(\frac{||r||e}{\ell - ||r||} \right)^2 - 1 \right]^{-\frac{1}{2}} \hat{\imath}_r \hat{\imath}_r^\top \\ & \pm \frac{1}{||r||} \left\{ \frac{\mu}{\ell} \left[e^2 - \left(\frac{\ell}{\|r\|} - 1 \right)^2 \right] \right\}^{\frac{1}{2}} \left(I - \hat{\imath}_r \hat{\imath}_r^\top \right) \\ & - \frac{\sqrt{\mu\ell}}{\|r\|^2} S(\hat{\imath}_h) \left(I - 2 \hat{\imath}_r \hat{\imath}_r^\top \right). \end{aligned} \tag{7.64}$$

7.4 Chapter Problems

Problem 7.1. Consider the one-dimensional vertical launch of a rocket from earth's surface as in Section 7.1 with the following data and aerodynamic drag $D(v) = v^2/2$.

```
m0 = 17500;      % kg
mf = 10000;      % kg
r0 = 0;          % m
v0 = 0;          % m/s
Tmax = m0*20;    % N
Isp = 330;       % s
```

(a) Using maximum thrust at all times, compute the burnout time and maximum altitude.
(b) Using optimal thrust, compute the time at which singular thrust initiates, burnout time, and maximum altitude.
(c) Plot altitude as a function of time for the maximum thrust and optimal thrust cases.
(d) Plot thrust as a function of time for the maximum thrust and optimal thrust cases.

Problem 7.2. Derive the integrated state equations in Eq. (7.38).

Problem 7.3. Simulate a nominal lunar ascent trajectory using the techniques of Section 7.2 and the following data.

```
yf  = 50000;     % ft
uf  = 5330;      % ft/s
tau = 3*abs(g); % ft/s2
```

Problem 7.4. Building upon the previous problem, incorporate into the simulation navigation errors, actuator dynamics, and disturbances.

Problem 7.5. Simulate a nominal lunar ascent trajectory by solving Eq. (7.17) using direct transcription and a one-dimensional search for the optimal time. Use the following data.

```
yf  = 50000;     % ft
uf  = 5330;      % ft/s
tau = 3*abs(g); % ft/s2
```

Recall that direct transcription was implemented using a standard discretization in Section 3.6 and a Chebyshev discretization in Section 4.8.

Problem 7.6. Building upon the previous problem, use discrete LQR to follow the nominal lunar ascent trajectory. Incorporate into the simulation navigation errors and disturbances. Recall that trajectory following using LQR was implemented in Section 4.6.

Problem 7.7. Simulate a nominal earth ascent trajectory using Q-guidance and the following data. States are ordered as downrange and altitude.

```
rf   = [100e3; 50e3];   % m
umax = 50;              % m/s2
```

Recall that Q-guidance was implemented in Section 1.2.

Problem 7.8. Building upon the previous problem, incorporate into the simulation navigation errors, actuator dynamics, and disturbances. Recall that Q-guidance was implemented in Section 1.2.