Linear quadratic and model predictive control

Lecture notes for EL2700

Mikael Johansson

Typeset using an adapted version of the the Legrand Orange Book.

Preface

During the last decade, model predictive control (MPC) has emerged as preferred control technology for many multi-variable and constrained control problems in a range of industries. The first heuristic applications of model-predictive control ideas were reported in the 70's, and much of the theory to guarantee stability and performance of MPC controllers was worked out already 20 years ago. Still, one can argue that it is the advances in computing hardware and efficient optimization algorithms that has enabled the broader use of the technology. Early applications of MPC considered systems with sampling intervals of several minutes and required a specialized computer for solving the associated planning problems. Today, model-predictive controllers are routinely implemented on inexpensive embedded hardware with sampling intervals on the millisecond scale.

These lecture notes were written for EL2700 – Model Predictive Control, given for the first time at KTH during the fall of 2017. One may ask if it is worth the while to write lecture notes when there are already many good books on model predictive control. For example, the ones by Maciejowski [26], Borrelli, Bemporand and Morari [9], and by Rawlings and Mayne [32] are excellent texts that we have often used as supplementary references in EL2700. Still, these books do not contain all the material that we want to teach, and they do not always present the concepts in the way or on the level that we would like. EL2700 attracts strong and ambitious students, but they come from diverse engineering programs. In addition, the course runs over seven weeks and 28 lecture hours, which is a short time for digesting all the essential aspects of MPC. We have also made an effort to develop a comprehensive set of exercises, and to promote open source software. All code listings in these lecture notes use open source packages in Python, and all exercises can be solved fully with free software.

Courses are almost never developed from scratch. Instead, they are often based on knowledge acquired from others, and the way we present different ideas is inspired by the way others have presented similar material to us. In this spirit, these notes draw on excellent lecture slides on linear dynamical systems (ee363) by Prof. Stephen Boyd at Stanford University, on discrete-time linear systems by Prof. Alberto Bemporad at IMT Lucca, on model predictive control by Prof. Mark Cannon at Oxford University, and by control systems design by Profs. Bo Bernhardsson and Karl-Johan Åström at Lund University. Dr. Stefano di Cairano and Prof. Vladimir Havlena generously shared ideas and insight into the topic ahead of the first edition of the course. Last, but not least, a continuous stream of outstanding teaching assistants, including Pedro Lima, Valerio Turri, Martin Biel, Jezdimir Milosevic, Linnea Persson, Abhishek Maji, Pedro Roque, Hamed Taghavian and Erik Berglund have all been essential to the development of EL2700.

Finally, this is still a living and partly incomplete document, but it has now reached a level of maturity that could make it helpful also outside of the context of EL2700. If you find the document useful, please acknowledge it, and if you have suggestions or ideas of how it can be improved, please do not hesitate to reach out to me directly.

A few words about EL2700

At KTH, a typical seven-week course teaches linear systems (Chapter 1) in week 1, finite-horizon optimal control (Chapter 3) in week 2, and linear-quadratic control (selected parts of Chapter 4) in week 3. To be able to prove stability of the linear-quadratic control law, we also teach Lyapunov theory (part of Chapter 2). The rest of the course is devoted to MPC (Chapter 5, and the remaining parts of Chapter 2). In week 4, we develop the basic model predictive control policy and prove its stability. Advanced linear MPC concepts, such as reference tracking, offset-free MPC and constraint softening, are discussed in week 5, and practical aspects of MPC in week 6. Week 7 is devoted to project guest lectures and a course summary.

In parallel to lectures and exercise sessions, a sequence of home works develops increasingly advanced solution to a design problem. Four basic hand-ins cover classical output feedback control based on pole placement design of state feedbacks and observers, finite-horizon optimal control, linear quadratic control and linear MPC, while a final hand in is formulated as a competition on a nonlinear model using nonlinear MPC techniques (not covered in these notes).

Contents

Introduction

Figure 1: Race driver approaching Silverstone's Maggots–Becketts–Chapel complex.

Picture a Formula 1 driver approaching Silverstone's Maggots–Becketts–Chapel sequence. Prior to the race, the team's engineers have determined the optimal racing line and speed profile: the precise trajectory through each apex that represents the fastest possible path. They have done so by solving a complex optimization problem involving vehicle dynamics, tire grip, and aerodynamic forces. If the world would match the model, the driver could apply the precomputed steering, throttle, and brake schedule and obtain the predicted outcome. On track, however, reality intervenes: light rain speckles the visor, a slower car ahead alters the feasible line, and tire pressures drift, reducing grip. The planned trajectory is valuable, but no longer optimal; it must be updated.

Model Predictive Control (MPC) formalizes this update process. Every few tenths of a second, it uses a model to predict the system state over the next few seconds, solves a constrained optimization that trades tracking the nominal plan against current limits and objectives (e.g., exit speed), applies only the first input, and then shifts the horizon and repeats as new measurements arrive. This continuous cycle of prediction, optimization, and re-optimization is the essence of MPC.

Beyond traditional feedback control

Traditional control strategies are mathematically elegant and computationally efficient, but face limitations in complex, constrained systems. Consider, for example, the classical PID controller. It computes control actions based on current error (proportional), accumulated past errors (integral), and error rate (derivative). Each component responds to information about what has already happened or is happening now, making PID inherently reactive. Imagine a driver who could only react to current conditions—maintaining constant throttle until the car began sliding, then reducing power; holding the steering wheel steady until drifting off-line, then correcting. This purely reactive approach would produce poor performance on a complex linked sequence like Maggots-Becketts-Chapel, where early decisions critically affect later corner exits. To get a strong performance, you need to "look around the corner" and plan ahead.

Linear Quadratic Regulators (LQR) represent a significant step in the right direction. LQR uses a linear model of the system dynamics to determine the feedback controller that minimizes the quadratic cost over an infinite-horizon. It enables more sophisticated control than PID, but still faces many important limitations. It does not account for constraints, such as control limitations or safety limits, and more advanced predictive capabilities must be incorporated through separate reference planning and feed-forward components. While these extensions work well within their design assumptions, they result in complex, multi-layered control architectures.

Model Predictive Control (MPC) takes a fundamentally different approach: it integrates prediction, optimization, and constraint handling into a single framework. At each time step, MPC uses a model to predict future system behavior, solves an optimization problem to find the best sequence of control actions while respecting all constraints, then applies only the first action and repeats the process with updated information. This mirrors exactly how experienced drivers navigate complex sequences like Maggots-Becketts-Chapel. They don't just react to current conditions—they mentally simulate multiple trajectories, weighing trade-offs between competing objectives while respecting physical limits, then execute the first part of their plan before re-evaluating based on new conditions. MPC formalizes this natural decision-making process, enabling control systems to "look around the corner" with mathematical rigor.

The three pillars of model predictive control

Model Predictive Control rests on three fundamental pillars that work together to enable an efficient and anticipatory control. Understanding these pillars—and how they interact—provides the conceptual foundation for everything that follows in this course.

Pillar 1: A model for predicting system behavior

The first pillar recognizes that effective control requires the ability to predict future system behavior. Just as our Formula 1 driver mentally models how steering inputs will affect vehicle trajectory through the upcoming corner sequence, MPC employs mathematical models to forecast system evolution over a finite time horizon. This predictive capability transforms control from reactive response to proactive planning. The model captures essential system dynamics—how current control actions influence future outputs—enabling the controller to "think before it acts." Whether predicting distillation column temperatures, spacecraft trajectories, or building thermal response, the internal model provides the foresight necessary for intelligent decision-making.

Pillar 2: A cost function that captures optimal behavior

The second pillar formalizes what constitutes "good control" through mathematical optimization. Our racing driver instinctively weighs competing objectives—minimize sector time while staying within track limits and preserving tire life. MPC makes these trade-offs explicit through carefully designed cost functions that quantify control objectives. The optimization process systematically

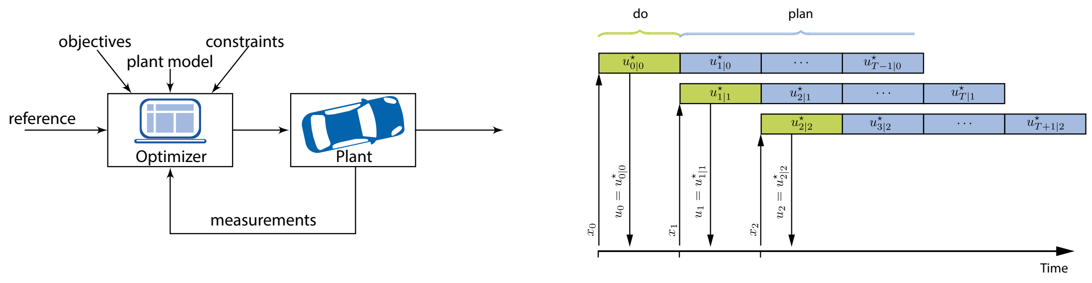


Figure 2: Essential components of a Model Predictive Control system (left). The optimizer uses the plant model to predict future behavior, then solves for optimal control actions while respecting constraints. Real-time measurements provide feedback for continuous re-planning. The right figure illustrates the receding horizon principle.

evaluates potential control sequences, selecting the one that best achieves desired performance while respecting system limitations. Whether minimizing energy consumption while maintaining product quality, or reducing tracking error while limiting actuator wear, the cost function enables controllers to make optimal decisions even when facing conflicting requirements.

Pillar 3: A receding-horizon approach

The third pillar addresses uncertainty through continuous replanning. Like our racing driver who constantly updates their racing line as conditions change, MPC implements only the first portion of each optimized control sequence, then repeats the entire prediction-optimization process with updated state information. This receding horizon approach transforms open-loop planning into closed-loop control. The prediction horizon moves forward through time, maintaining a constant forward-looking perspective while incorporating real-time feedback. Each optimization cycle benefits from the most current system information, ensuring that control actions remain relevant despite an uncertain future.

Together, these three pillars enable MPC's defining feature: the ability to optimize performance while rigorously respecting constraints. Unlike traditional control methods that treat constraints as afterthoughts, MPC elevates physical and operational limits to first-class status within the optimization framework. This paradigm shift from constraint avoidance to constraint utilization unlocks performance gains impossible with conservative control approaches.

Real-world impact and validation

Model Predictive Control emerged from the practical demands of 1970s chemical processing, where traditional control methods proved inadequate for managing large-scale, highly interconnected industrial systems. The breakthrough came in 1973 when Charles Cutler and Brian Ramaker at Shell Oil developed Dynamic Matrix Control (DMC) to address challenging distillation column control problems [12, 13]. Their approach demonstrated notable success in improving process performance and reducing variability. Parallel development occurred in France, where Jacques Richalet and colleagues introduced Model Predictive Heuristic Control in 1978, further establishing the viability of predictive control strategies [33]. What began as a specialized solution for chemical plants gained momentum throughout the process industries. By 1999, over 4,500 MPC applications had been documented, establishing predictive control as the standard approach for advanced process control in refineries, petrochemical plants, and other continuous manufacturing operations [31].

The quantified benefits explain this rapid acceptance. Implementations often achieve process variability reductions exceeding 50%, enabling plants to operate closer to optimal conditions while

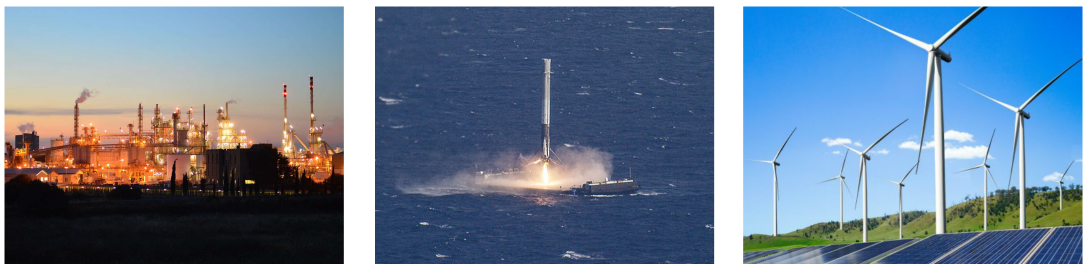


Figure 3: Modern MPC applications: from factory-wide control, through rocket landings, to renewable energy integration.

maintaining safety margins [2]. Production yield improvements of 1-5% may appear modest, but translate to millions of dollars annually for large-scale chemical operations [22, 34]. In building control systems, energy efficiency gains of 10-20% are typical, with exceptional cases reaching 30% savings [1]. The Advanced Process Control market is valued at $5-6 billion globally, with Model Predictive Control technologies comprising approximately 46% of this market [28].

Modern applications

Modern MPC implementations handle significantly more complex systems than the original applications. In process industries, MPC systems now manage dozens of controlled and manipulated variables simultaneously to optimize complex reaction networks for both product quality and energy efficiency [34]. These systems demonstrate capabilities that would be difficult to achieve with conventional control approaches. In aerospace, optimization-based guidance systems manage complex reentry and precision landing maneuvers for reusable launch vehicles [8], demonstrating MPC's ability to handle significant nonlinearities, hard constraints, and time-critical optimization under uncertainty. Smart building applications represent another growing domain, with MPC-based HVAC systems achieving 12-16% energy savings while maintaining superior comfort levels compared to conventional building automation [30]. Power grid integration of renewable energy sources increasingly relies on MPC algorithms to manage supply variability and demand fluctuations [21].

These diverse applications demonstrate that MPC has evolved into a powerful technology that supports critical industrial operations, from the refineries that produce our fuels to the spacecraft that advance human exploration, with a high practical value.

Course preview

In this course, you will develop the complete mathematical framework underlying Model Predictive Control, from fundamental prediction models to advanced optimization algorithms. You will gain fluency with stability theory for constrained systems, understanding not just how to design MPC controllers, but why they remain stable and perform reliably under uncertainty.

Beyond theoretical foundations, you'll master practical design tools that translate MPC concepts into working control systems—learning to tune prediction horizons, weight matrices, and constraint formulations for real applications. You will also develop engineering judgment about when MPC represents the right solution and when simpler approaches suffice.

We begin with linear systems and quadratic cost functions, where elegant analytical solutions illustrate core MPC principles without overwhelming complexity. We then develop deep understanding of prediction, optimization, and receding horizon concepts before confronting computational challenges of constrained systems. Finally, we tackle nonlinear systems where the full power of MPC emerges, but where numerical methods and approximation techniques become essential.

Part One: System Theory

1. Discrete-time linear systems

This first chapter summarizes basic theory for discrete-time linear systems that will be essential for our later developments. Particular attention is given to understanding how the state equations can be used to predict the future evolution of the system, and under what conditions we can find a control signal that transfers the state from any initial value to any future target. We also discuss stability concepts, how pole and zero locations affect the transient response, and how to design linear controllers using pole placement. In parallel, we explore how we can estimate the complete state vector from system output measurements. Finally, we elaborate on how the theory for discrete-time linear systems can be used to design controllers for continuous-time systems.

1.1 State-space equations, system response, and stability

We consider discrete-time linear systems on the form

$$\begin{array}{rcl} x_{t+1} & = & Ax_t + Bu_t \\ y_t & = & Cx_t + Du_t \end{array} \qquad t = 0, 1, \dots \tag{1.1}$$

Here, A, B, C and D are constant matrices, $x_t \in \mathbb{R}^n$ is the *state vector* at time t, $u_t \in \mathbb{R}^m$ is the *control vector*, and $y_t \in \mathbb{R}^p$ is the *output vector*. The time index $t \in \mathbb{Z}_{\geq 0}$ is integer valued, and the model typically describes how an underlying physical system evolves between sampling instances. Specifically, for a given initial state x_0, the model describes how an input sequence $\{u_0, u_1, \dots,\}$ affects the sequence of states $\{x_1, x_2, \dots\}$ and the corresponding outputs $\{y_0, y_1, \dots,\}$. We use the shorthand notation (A, B, C, D) for a system whose state space representation is given by (1.1).

The free system response

Let us first consider the response of the system when $u_t \equiv 0$ for all t. This free (or unforced) system response is simply given by

$$x_{t+1} = Ax_t. \tag{1.2}$$

If we start from an initial state x_0 at $t = 0$, then $x_1 = Ax_0$, $x_2 = Ax_1 = A^2x_0$, and more generally,

$$x_t = A^t x_0$$

This means that the future states are completely determined by the initial state and the system matrix A. This behavior is easiest to understand for a scalar system

$$x_{t+1} = ax_t$$

It is then clear that

$$x_t = a^t x_0$$

can have three distinct behaviors. If $|a| < 1$, the state will converge to zero; if $|a| > 1$ then it will grow exponentially, and if $|a| = 1$, it will stay bounded (either at x_0 or alternating between x_0 and $-x_0$ every sample interval) but neither converge nor diverge.

Definition 1.1.1 — Stability properties. The autonomous linear system (1.2) is *asymptotically stable* if its solution $\{x_t\}$ satisfies $x_t \to \mathbf{0}$ as $t \to \infty$ for all $x_0 \in \mathbb{R}^n$; it is *stable* if there exists $\varepsilon > 0$, dependent on x_0, such that $\|x_t\| \leq \varepsilon$ for all t; and it is *unstable* if $\|x_t\| \to \infty$ as $t \to \infty$.

When the system has multiple states, the stability properties are still determined by the system matrix A. In the special case that A is diagonal,

$$x_{t+1} = \begin{pmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{pmatrix} x_t$$

each component of the state vector evolves as an independent scalar linear system. Therefore, if $|a_{ii}| < 1$ for all i, the system is asymptotically stable; if $|a_{ii}| \leq 1$ for all i, it is stable; and if $|a_{ii}| > 1$ for some i, the system is unstable.

A similar result holds when A is a full $n \times n$ matrix with n linearly independent eigenvectors. We denote the eigenvectors by v_i and the corresponding eigenvalues by λ, so that $Av_i = \lambda_i v_i$ for all $i = 1, \ldots, n$. By introducing the matrices $V = \begin{pmatrix} v_1 & \cdots & v_n \end{pmatrix}$ and $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n)$, we can write these conditions compactly as $AV = V\Lambda$. Now, since the eigenvectors are assumed to be linearly independent, V is invertible, so $A = V\Lambda V^{-1}$ and

$$x_t = A^t x_0 = (V\Lambda V^{-1})^t x_0 = V\Lambda^t V^{-1} x_0.$$

Since V is a constant matrix, A^t vanishes as $t \to \infty$ if Λ^t vanishes, *i.e.* if $|\lambda_i| < 1$ for all $i = 1, \ldots, n$. This suggests that the eigenvalues of A determine the stability properties of the system. As the next result shows, this is indeed the case, no matter if A is diagonalizable or not.

Theorem 1.1.1 — Stability of linear systems. The discrete-time linear system $x_{t+1} = Ax_t$ with $A \in \mathbb{R}^{n \times n}$ is asymptotically stable if and only if all eigenvalues of A have magnitude strictly less than one, *i.e.* if

$$|\lambda_i(A)| < 1, \qquad \forall i = 1, \ldots, n. \tag{1.3}$$

The system is *stable* if $|\lambda_i(A)| \leq 1$ and the eigenvalues that satisfy $|\lambda_i(A)| = 1$ have as many linearly independent eigenvectors as their multiplicity; and it is unstable otherwise.

Matrices A whose eigenvalues satisfy (1.3) are called *Schur stable*. Although it is sometimes possible to compute the eigenvalues of the matrix A analytically, *e.g.* as solutions to the characteristic equation $p(\lambda) = \det(\lambda I - A) = 0$, we will typically rely on numerical calculations to assess stability.

Geometrically, the condition (1.3) requires that all eigenvalues lie inside the unit circle in the complex plane. Hence, the unit circle defines the *stability boundary* for discrete-time linear systems. A few examples of eigenvalue locations and the corresponding free system responses are shown in Figure 1.1. A system with multiple eigenvalues on the unit circle can be either stable or unstable. For example, $x_{t+1} = Ix_t$ has n eigenvalues at $\lambda = 1$, but also n independent eigenvectors (the Euclidean basis vectors). Hence, it is stable. However, there are also systems with multiple equal eigenvalues on the unit circle that are unstable; see Exercise 1.2.

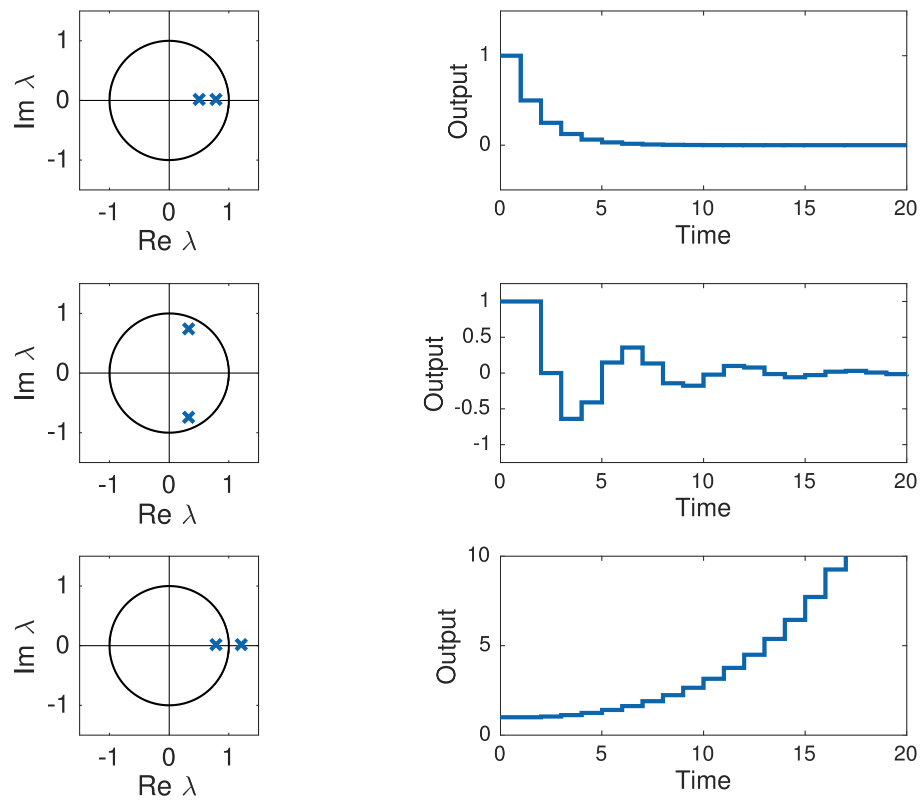


Figure 1.1: Eigenvalue locations (left) and associated unforced responses (right): eigenvalues on the positive real axis inside the unit circle give a well-damped response; eigenvalues close to the stability boundary results in an oscillatory response; one or more eigenvalues outside the unit circle results in an unstable system.

In contrast to continuous-time linear systems, where the free response can only tend to zero asymptotically but never reaches the origin in finite time, there are discrete-time linear systems whose state converges to zero in a finite number of steps. One such example is $x_{t+1} = Ax_t$ with

$$A = \begin{pmatrix} 0 & 1/2 \\ 0 & 0 \end{pmatrix}$$

It is easy to verify that $A^2 = \mathbf{0}$, so that $x_2 = \mathbf{0}$ for all initial values x_0. Matrices A such that $A^k = \mathbf{0}$ for some finite value of k are called *nilpotent*, and have all their eigenvalues at origin.

The driven response

Let us now consider the response of the system driven by the input u. We can proceed in the same way as we did for the free response, i.e. by simply iterating the system equations forward:

$$\begin{aligned}
x_1 &= Ax_0 + Bu_0 \\
x_2 &= Ax_1 + Bu_1 = A(Ax_0 + Bu_0) + Bu_1 = A^2x_0 + ABu_0 + Bu_1 \\
&\vdots \\
x_t &= A^t x_0 + \sum_{k=0}^{t-1} A^k B u_{t-1-k}
\end{aligned}$$

We note that the response is divided into two terms: the *free response* $A^t x_0$ (accounting for the effect of the initial value) and the *driven response* $\sum_{k=0}^{t-1} A^k B u_{t-1-k}$ (accounting for the effect of the system input). Since the output is just a linear combination of the state and the input, we have

$$y_t = Cx_t + Du_t = CA^t x_0 + \sum_{k=0}^{t-1} CA^k B u_{t-1-k} + Du_t \tag{1.4}$$

One important property of the driven response is that it is linear in x_0 and the input sequence $\{u_0, u_1, \dots\}$. It is convenient to represent this relation in vector form

$$\begin{aligned}
x_t &= A^t x_0 + \begin{pmatrix} A^{t-1}B & A^{t-2}B & \dots & AB & B \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_{t-1} \end{pmatrix} \\
&:= A^t x_0 + \mathcal{C}_t \mathcal{U}_t
\end{aligned} \tag{1.5}$$

The matrix $\mathcal{C}_t$ is called the *controllability matrix* over horizon t and will play an important role in our developments.

State transformations

It is sometimes useful to represent the state vector in another basis, *i.e.* to make a coordinate change

$$z = T^{-1}x$$

for some full rank matrix $T \in \mathbb{R}^{n\times n}$. In the new coordinates, the system state evolves according to

$$\begin{aligned}
z_{t+1} &= T^{-1}x_{t+1} = T^{-1}(Ax_t + Bu_t) = T^{-1}ATz_t + T^{-1}Bu_t \\
y_t &= CTz_t + Du_t
\end{aligned}$$

so it can be represented by the discrete-time linear system $(T^{-1}AT, T^{-1}B, CT, D)$. When A is diagonalizable, we have already seen how $T = V$, where V is a matrix of the eigenvectors of A, makes $T^{-1}AT$ diagonal and simplifies the analysis. However, as we will see later, coordinate changes are useful in many other contexts.

Intuitively, a proper transformation of the state vector should not alter the input-output behavior, which the following result asserts.

Theorem 1.1.2 Let $T \in \mathbb{R}^{n\times n}$ be a full rank matrix. The two discrete-time linear systems

$$\begin{aligned}
x_{t+1} &= Ax_t + Bu_t \\
y_t &= Cx_t + Du_t
\end{aligned}$$

and

$$\begin{aligned} z_{t+1} &= T^{-1}ATz_t + T^{-1}Bu_t \\ y_t &= CTz_t + Du_t \end{aligned}$$

realize the same input-output behavior, given that $z_0 = T^{-1}x_0$.

Proof. The input-output behavior for (A,B,C,D) is given by (1.4). For $(T^{-1}AT, T^{-1}B, CT, D)$,

$$\begin{aligned} y_t &= CT(T^{-1}AT)^t z_0 + \sum_{k=0}^{t-1} CT(T^{-1}AT)^k T^{-1}Bu_{t-1-k} + Du_t = \\ &= CA^t x_0 + \sum_{k=0}^{t-1} CA^k Bu_{t-1-k} + Du_t \end{aligned}$$

which proves the equivalence. ■

1.2 Reachability and state transfer

Many control problems concern *state transfer*: given an initial state x_0, we would like to find a control sequence $\{u_0, u_1, \ldots, u_{t-1}\}$ that brings the system state to a target x_{tgt} at time t.

For now, we will only consider the basic question of *if* a given discrete-time system admits a control sequence that performs the requested state transfer. Later, we will try to find control sequences that perform the state transfer in an optimal way (for example, using minimal energy or in minimal time) while respecting various constraints. We begin with the following definition.

Definition 1.2.1 — Reachability. Consider the discrete-time linear system (1.1). We call a specific target state $x_{\text{tgt}} \in \mathbb{R}^n$ reachable in k steps if there is an input sequence $\{u_0, u_1, \ldots, u_{k-1}\}$ that drives the system from initial state $x_0 = 0$ to $x_k = x_{\text{tgt}}$. We say that the system (1.1) is reachable if there exists a finite value of k such that all $x_{\text{tgt}} \in \mathbb{R}^n$ are reachable in k steps.

By Equation 1.5, a control sequence that drives $x_0 = 0$ to $x_t = x_{\text{tgt}}$ must satisfy

$$x_{\text{tgt}} = \mathcal{C}_t \mathcal{U}_t \tag{1.6}$$

This equation has a solution if and only if x_{tgt} lies in the range of $\mathcal{C}_t$. In particular, we can reach any x_{tgt} if and only if $\mathcal{C}_t$ has n linearly independent columns. This observation leads to the next result.

Theorem 1.2.1 — Reachability. The linear system (1.1) is reachable if and only if

$$\operatorname{rank}(\mathcal{C}_n) = n$$

Proof. If the controllability matrix over horizon n has rank n, then the system of linear equations

$$x_{\text{tgt}} = \mathcal{C}_n \mathcal{U}_n \tag{1.7}$$

admits a solution for every x_{tgt}. In other words, there exists a control sequence $\mathcal{U}_n$ that drives the system from $x_0 = 0$ to $x_n = x_{\text{tgt}}$ for every x_{tgt}, so the system is reachable.

To show that reachability of a system implies reachability in n steps, note that the definition of the controllability matrix implies that

$$\mathcal{C}_t = \begin{pmatrix} A^{t-1}B & A^{t-2}B & \cdots & AB & B \end{pmatrix} = \begin{pmatrix} A^{t-1}B & \mathcal{C}_{t-1} \end{pmatrix}$$

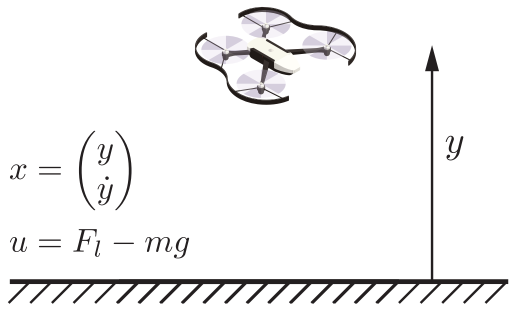


Figure 1.2: Vertical dynamics of a quadcopter: we can control the lift force F_l generated by the rotors to counteract the downwards force due to gravity, mg. The model uses the vertical position and velocity as states, and considers the net lift $F_l - mg$ as control signal.

since we add new columns to the reachability matrix when we increase t, the range of $\mathcal{C}_{t+1}$ is greater or equal to the range of $\mathcal{C}_t$. However, by the Cayley-Hamilton theorem (see Appendix A)

$$A^n = \alpha_{n-1}A^{n-1} + \alpha_{n-2}A^{n-1} + \cdots + \alpha_1 A + \alpha_0 I \tag{1.8}$$

for some scalars $\alpha_0, \alpha_1, \ldots \alpha_{n-1}$. Hence,

$$\text{range}(\mathcal{C}_{n+1}) = \text{range}(\begin{pmatrix} A^n B & \mathcal{C}_n] \end{pmatrix})$$

since the new columns added to $\mathcal{C}_{n+1}$ are linear combinations of the columns already present in $\mathcal{C}_n$. Repeated use of the Cayley-Hamilton theorem reveals that $\text{range}(\mathcal{C}_t) = \text{range}(\mathcal{C}_n)$ for all $t \geq n$. ■

An important consequence of Theorem 1.2.1 is that if a discrete-time system is reachable then, for every target state, there exists a control sequence that drives the system state from the origin to that target in exactly n steps. The next example elaborates on some of these ideas.

■ **Example 1.1** The vertical dynamics of a quadcopter is given by

$$m\ddot{y}(t) = F_l - mg$$

where y denotes the vertical position, $F_l(t)$ is the lift force generated by the rotors, and g is acceleration due to gravity; see Figure 1.2 for an illustration. With $u(t) = F_l(t) - mg$ and $m = 1$, a corresponding discrete-time model is given by

$$\begin{aligned} x_{t+1} &= \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} x_t + \begin{pmatrix} 0.5 \\ 1 \end{pmatrix} u_t \\ y_t &= \begin{pmatrix} 1 & 0 \end{pmatrix} x_t \end{aligned}$$

We can verify that the model is reachable, since

$$\mathcal{C}_2 = \begin{pmatrix} B & AB \end{pmatrix} = \begin{pmatrix} 0.5 & 1.5 \\ 1 & 1 \end{pmatrix}$$

has full rank. In addition to guaranteeing that it is possible to reach every target state, the reachability argument also provides an open-loop control strategy for doing so. Assume that we want to drive the quadcopter from rest at time zero, $x_0 = \begin{pmatrix} 0 & 0 \end{pmatrix}$, to rest at the height of one meter, $x_{\text{tgr}} = \begin{pmatrix} 1 & 0 \end{pmatrix}$. Then, we can do so using

$$\begin{pmatrix} u_1 \\ u_0 \end{pmatrix} = \mathcal{C}_2^{-1} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = -\begin{pmatrix} 1 & -1.5 \\ -1 & 0.5 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$$

We thus apply $u_0 = 1$, followed by $u_1 = -1$. If we want to move to rest at an arbitrary height h, i.e., let $x_{\text{tgt}} = \begin{pmatrix} h & 0 \end{pmatrix}$, the reachability result states that this can also be done in $n = 2$ steps. The associated control is $u_0 = h$ followed by $u_1 = -h$. In any practical system, the control signal is limited in magnitude, and we cannot hope to steer the state arbitrarily far in n steps. ■

In the reachability definition, we have assumed that $x_0 = 0$. However, it is clear that if the system is reachable then we can find a sequence $\mathcal{U}_n$ that transfers the system state from any initial state to any final state, since

$$x_{\text{tgt}} - A^n x_0 = \mathcal{C}_n \mathcal{U}_n$$

admits a solution for any left-hand side argument.

When a system is not reachable, there are states, or linear combinations of states, which we cannot influence with the controls. The following example illustrates this fact.

■ **Example 1.2** Consider the system

$$x_{t+1} = \begin{pmatrix} a_{11} & a_{12} \\ 0 & a_{22} \end{pmatrix} x_t + \begin{pmatrix} b_1 \\ 0 \end{pmatrix} u_t \tag{1.9}$$

Clearly, u_t cannot affect the second state, neither directly nor indirectly through the first state. We can come to the same conclusion by computing the controllability matrix

$$\mathcal{C}_2 = \begin{pmatrix} AB & B \end{pmatrix} = \begin{pmatrix} a_{11}b_1 & b_1 \\ 0 & 0 \end{pmatrix}.$$

With this controllability matrix, any solution to (1.7) must have the second component of x_{tgt} equal to zero. Hence, we cannot steer the second state away from its inital value. ■

It is not always this straightforward to determine reachable and unreachable states by visual inspection. However, as the next result shows, it is possible to construct a state transformation that reveals the reachable and unreachable subsystems.

Theorem 1.2.2 Let $\text{rank}(\mathcal{C}_n) = n_r < n$. Then there exists a state transformation $z = T^{-1}x$ such that the system in the transformed coordinates has the form

$$\begin{aligned} z_{t+1} &= \begin{pmatrix} A_r & A_{12} \\ 0 & A_{\bar{r}} \end{pmatrix} z_t + \begin{pmatrix} B_r \\ 0 \end{pmatrix} u_t \\ y_t &= \begin{pmatrix} C_r & C_{\bar{r}} \end{pmatrix} z_t + D u_t \end{aligned} \tag{1.10}$$

where (A_r, B_r) is reachable.

Clearly, in the transformed coordinates, the first n_r states are reachable, and the remaining ones are not (since they cannot be directly affected by the control, nor indirectly through the first states).

The following theorem, known as the Popov-Belevitch-Hautus test gives an alternative way to verify reachability. It is a convenient theoretical tool that adds some geometrical insight.

Theorem 1.2.3 — PBH test for reachability. The system (1.1) is *unreachable* if and only if there exists $\lambda \in \mathbb{C}$ and $w \in \mathbb{C}^n$ with $w \neq 0$ such that

$$w^\top A = \lambda w^\top \qquad w^\top B = 0 \tag{1.11}$$

i.e. if one of the left eigenvectors of A is orthogonal to the columns of B.

Proof. If $w \neq 0$ satisfies (1.11), then $w^\top B = 0$, $w^\top AB = \lambda w^\top B = 0$ and similarly $w^\top A^t B = \lambda^t w^\top B = 0$ for all t. Hence, $w^\top \mathcal{C}_n = w^\top \begin{pmatrix} A^{n-1}B & \cdots & AB & B \end{pmatrix} = 0$, i.e. the controllability matrix does not have full rank, and the system is therefore unreachable. Conversely, if the system is unreachable, then there is a state transformation $z = T^{-1}x$ which brings the system to the form (1.10). Now, let $w_{\bar{r}}$ be a left eigenvector of $A_{\bar{r}}$ with eigenvalue λ, i.e. $w_{\bar{r}}^\top A_{\bar{r}} = \lambda w_{\bar{r}}^\top$. Then $w^\top = \begin{pmatrix} 0 & w_{\bar{r}}^\top \end{pmatrix}$ satisfies (1.11) for the transformed system, and $w' = T^{-\top} w$ for the original one. ■

Note that the first inequality in (1.11) can also be written as $A^\top w = \lambda w$, so λ is an eigenvalue of $A^\top$ (and therefore of A; cf. Proposition A.3.2). If B has many columns, the second equality states that $w^\top B$ must evaluate to a zero matrix. When w in Theorem 1.2.3 is complex valued, $w^\top$ denotes the complex conjugate transpose of w (which is often written as w^* rather than $w^\top$).

This discussion reveals one approach for applying the PBH test. Begin by computing the eigenvalues λ_i and eigenvectors w_i of $A^\top$. If any of the eigenvectors satisfy $w_i^\top B$, the system is not reachable. In many cases, however, it is easier to simply try to solve the PBH conditions directly for w and λ. To demonstrate this use of the PBH test, we return to the system in Example 1.2.

■ **Example 1.3** To show that (1.9) is unreachable using the PBH test, we need to find λ and

$$w = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$$

that satisfy (1.11). First note that $w^\top B = w_1 b_1 = 0$ implies that $w_1 = 0$. Next, the condition

$$w^\top A = \begin{pmatrix} 0 & w_2 a_{22} \end{pmatrix} = \lambda w^\top = \lambda \begin{pmatrix} 0 & w_2 \end{pmatrix}$$

is satisfied with $w_2 = 1$ and $\lambda = a_{22}$. Hence, we have found a valid solution to (1.11) and the system is unreachable. In the proof of the PBH test, we noted that any w that satisfies the conditions (1.11) also satisfies $w^\top \mathcal{C}_2 = 0$. Hence, any solution to (1.7) must fulfill

$$w^\top x_{\text{tgt}} = w^\top \mathcal{C}_2 = 0$$

The w vector that we have determined above, therefore, implies that the second component of x_{tgt} must be zero. This is consistent with our observations in Example 1.2. ■

The solution to the PBH condition has an intesting and useful interpretation: $z_t = w^\top x_t$ defines a linear combination of the states that remains unaffected by the control. In particular, $z_{t+1} = w^\top(Ax_t + Bu_t) = \lambda w^\top x_t = \lambda z_t$. Hence, initial values with $z_0 = 0$ must satisfy $w^\top x_t$ for all $t \geq 0$.

It may appear strange that we call $\mathcal{C}_n$ the controllability matrix and not the reachability matrix. After all, we use it to determine the reachability properties of the underlying system. The reason for this naming convention is historical: for continuous-time linear systems, reachability coincides with the following concept of controllability:

Definition 1.2.2 — Controllability. Consider the discrete-time linear system (1.1). We say that the state $x_{\text{tgt}} \in \mathbb{R}^n$ is *controllable in k steps* if there exists an input sequence $\{u_0, u_1, \ldots, u_{k-1}\}$ that drives the system from initial state $x_0 = x_{\text{tgt}}$ to $x_k = 0$. If there exists a finite time k so that all $x_{\text{tgt}} \in \mathbb{R}^n$ are reachable in k steps, we say that the system (1.1) is controllable.

Controllability implies the existence of a k and a $\mathcal{U}_k \in \mathbb{R}^{m \times k}$ such that

$$-A^k x_{\text{tgt}} = \mathcal{C}_k \mathcal{U}_k$$

Clearly, any reachable system will also be controllable, since $\mathcal{C}_n$ is guaranteed to be of full rank. However, since A may be nilpotent (i.e., there is a finite t_0 such that $A^t = 0$ for all $t \geq t_0$), there are systems that are controllable but not reachable. One such system is

$$x_{t+1} = \begin{pmatrix} 0 & 1/2 \\ 0 & 0 \end{pmatrix} x_t + \begin{pmatrix} 1 \\ 0 \end{pmatrix} u_t$$

This system is a special instance of the systems studied in Example 1.2, and hence not reachable. However, it is controllable since $u_t \equiv 0$ ensures that $x_t = \mathbf{0}$ for all $t \geq 2$. In fact, one can show that a system is controllable if and only if all eigenvalues of its unreachable subsystem are equal to zero. The concept of controllability can be weakened further by the notion of stabilizability:

Definition 1.2.3 The discrete-time linear system (1.1) is *stabilizable* if there exists an input sequence $\{u_k\}_{k=0}^{\infty}$ such that $\lim_{t\to\infty} x_t = \mathbf{0}$.

The fact that we no longer require the state to reach the origin in finite time means that we can allow for systems whose unreachable modes are asymptotically stable. We summarize the observation in the following theorem, left without proof.

Theorem 1.2.4 The system (1.1) is stabilizable if all eigenvalues λ_i of its unreachable subsystem satisfy $|\lambda_i| < 1$, *i.e.* if any $w \in \mathbb{C}^n$ with $w \neq 0$ which satisfies

$$w^\top A = \lambda_i w^\top, \qquad w^\top B = 0$$

does so for $|\lambda_i| < 1$.

1.3 Observability and state reconstruction

When we cannot measure the complete state vector x_t but only observe the output y_t, it is useful to be able to estimate the current state from the observed input and output sequences, $\{y_0, y_1, \dots, y_t\}$ and $\{u_0, u_1, \dots, u_{t-1}\}$. We will address this problem in more detail later in these notes, but will now consider the simpler problem of reconstructing the initial state x_0 from measured input and output sequences. At least conceptually, once we know the initial state, we can use the model (1.1) and the applied input sequence to compute the present state.

To understand when we can estimate the initial state, we re-write (1.4) as

$$CA^t x_0 = y_t - \left(\sum_{k=0}^{t-1} CA^k B u_{t-1-k} + D u_t \right)$$

and note that $\varepsilon_t = CA^t x_0$ can be computed from the measured output y_t and the past inputs $u_0, u_1, \dots, u_t$. To be able to reconstruct the initial state, the system of equations

$$\begin{pmatrix} \varepsilon_0 \\ \varepsilon_1 \\ \vdots \\ \varepsilon_{t-1} \end{pmatrix} = \begin{pmatrix} C \\ CA \\ \cdots \\ CA^{t-1} \end{pmatrix} x_0 := \mathcal{O}_t x_0$$

must admit a unique solution x_0 for all values of $\{\varepsilon_0, \dots, \varepsilon_{t-1}\}$, *i.e.* $\mathcal{O}_t$ must have full rank. We refer to $\mathcal{O}_t$ as *observability matrix* over horizon t.

Definition 1.3.1 — Observability. A state $x_{\text{init}} \neq 0$ is said to be *unobservable* over k steps if, $x_0 = x_{\text{init}}$ and $u_t = 0$ for $t = 0, \dots, k-1$ imply that $y_t = \mathbf{0}$ for $t = 0, \dots, k-1$. The system (1.1) is called *unobservable* if it has some unobservable state, and called *observable* otherwise.

By this definition, if x_0 is an observable state and $u_t = 0$ for all t, then $\varepsilon_t = \mathbf{0}$ for all t. But the same holds for $x_0 = \mathbf{0}$, so an unobservable state cannot be distinguished from $x_0 = \mathbf{0}$ based on the observed output sequence $\{y_t\}$. The development of observability criteria parallels the one on controllability. Specifically, the rank test for observability takes the following form.

Theorem 1.3.1 — Observability. The linear system (1.1) is observable if and only if

$$\operatorname{rank}(\mathcal{O}_n) = n$$

The proof uses the Cayley-Hamilton theorem in an analogous manner to the proof of Theorem 1.2.1 and is left as an exercise. Any pair of matrices (A,C) that yield an observability matrix of full rank is called an observable pair. In this way, (A,B,C,D) is observable if and only if (A,C) is an observable pair. We also make the following observation: if, for an observable linear system, u_t and y_t are both zero over n consecutive samples $t = 0,1,\dots,n-1$, then x_0 must be identical to zero.

The PBH test for observability takes the following form.

Theorem 1.3.2 The linear system (1.1) is unobservable if and only if there exists $v \in \mathbb{C}^n$ with $v \neq 0$ and $\lambda \in \mathbb{C}$ such that

$$Av = \lambda v \qquad Cv = 0$$

i.e., if one if the right eigenvectors of A is orthogonal to the rows of C.

The next example demonstrates how the PBH test can be used to verify observability.

■ **Example 1.4** To verify observability of the quadcopter model from Example 1.1, we form

$$\mathcal{O} = \begin{pmatrix} C \\ CA \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$$

Since the observability matrix has full rank, the system is observable. Let us instead assume that we can only measure the vertical velocity, *i.e.* that $C = \begin{pmatrix} 0 & 1 \end{pmatrix}$. Then,

$$\mathcal{O} = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}$$

which implies that this system is not observable. This conclusion is intuitive: if we only measure the velocity of the quadrotor, there is no way that we can figure out its initial position. ■

Similarly to the decomposition into reachable and unreachable subsystems, a linear system can also be decomposed into an observable and an unobservable subsystem.

Theorem 1.3.3 Let $\operatorname{rank}(\mathcal{O}_n) = n_o < n$. Then there exists a state transformation $z = T^{-1}x$ such that the system in the transformed coordinates has the form

$$z_{t+1} = \begin{pmatrix} A_{\bar{o}} & A_{12} \\ 0 & A_o \end{pmatrix} z_t + \begin{pmatrix} B_{\bar{o}} \\ B_o \end{pmatrix} u_t$$
$$y_t = \begin{pmatrix} 0 & C_o \end{pmatrix} z_t + Du_t$$

where (A_o, B_o, C_o, D_o) is observable.

Analogously to the discussion about reachability, controllability and stabilizability, we can weaken the notion of observability to systems that are reconstructable and detectable.

Definition 1.3.2 The discrete-time linear system (1.1) is *reconstructable in k steps* if, for every initial condition x_0, it holds that x_k is uniquely determined by $\{u_0, u_1, \dots, u_{k-1}\}$ and $\{y_0, y_1, \dots, y_{k-1}\}$.

One can prove that a system is reconstructable if and only if the eigenvalues of its unobservable subsystem matrix are null. The concept of detectability allows for systems where the unobservable modes are asymptotically stable:

Definition 1.3.3 The discrete-time linear system (1.1) is *detectable* if it is asymptotically reconstructable in k steps as $k \to \infty$.

As the next PBH test states, a system is detectable if and only if its unobservable subspace is asymptotically stable.

Theorem 1.3.4 The system (1.1) is detectable if all eigenvalues λ_i of its unobservable subsystem satisfy $|\lambda_i| < 1$, *i.e.* any solution $v \in \mathbb{C}^n$ with $v \neq 0$ to

$$Av = \lambda_i v, \qquad Cv = 0$$

has $|\lambda_i| < 1$.

■ **Example 1.5** Let us return to the quadrotor dynamics, but instead assume that we can only measure the vertical velocity, i.e. let $C = \begin{pmatrix} 0 & 1 \end{pmatrix}$. To determine if the system is detectable, we use the PBH test and look for $v \neq 0$ such that

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} v = \lambda v, \qquad \begin{pmatrix} 0 & 1 \end{pmatrix} v = 0$$

From the second of these conditions, we see that $[v]_1 = 0$. Any such v satisfies the first equality, provided that $\lambda = 1$. Hence, the system is not detectable. ■

1.4 State feedback and observers

Most control design techniques for state-space models are based on *state feedback*. The control signal is then simply a linear combination of the system states, $u_t = -Lx_t$. The classical design technique is to choose the gain matrix L to assign desired eigenvalues to the closed-loop system matrix, but many modern design techniques find the optimal L with respect to a precise performance criterion, such as energy-efficiency, robustness to model uncertainty, or disturbance suppression.

If the full state vector cannot be measured, then one can use the model and the measured system inputs and outputs to compute an estimate $\hat{x}_t$ of the state vector. An *observer* uses the model to predict the state vector and the associated output $\hat{y}_t$. If the measured output y_t disagrees with the prediction, the state estimate is corrected in proportion to the error $K(y_t - \hat{y}_t)$. The observer gain matrix K is designed in a similar manner as the state feedback gains.

Combining the state estimator with feedback from estimated states results in an *output feedback* strategy, *i.e.* a controller that measures the system output y_t and produces a control signal u_t.

In this section, we will review a design approach for state feedback and estimator gains based on eigenvalue assignment. Later in these notes, we will derive optimal gains with respect to a quadratic cost function that accounts for both transient performance and the required control effort.

State feedback

Let us begin by studying the linear state feedback

$$u_t = -Lx_t. \tag{1.12}$$

If we insert this expression into the state-space model (1.1) we find the *closed-loop* dynamics

$$\begin{aligned} x_{t+1} &= (A - BL)x_t \\ y_t &= (C - DL)x_t \end{aligned}$$

Thus, the state evolution of the closed-loop system is characterized by the matrix $(A - BL)$. As the next result shows, the eigenvalues of this closed-loop matrix can be assigned to arbitrary locations as long as the open-loop system is reachable.

Theorem 1.4.1 The state feedback (1.12) allows to assign the eigenvalues of $A-BL$ to arbitrary complex conjugate locations if and only if the open-loop system is reachable.

The feedback gains can thus be designed by equating the closed-loop characteristic polynomial

$$\lambda(s) = \det(sI - (A - BL))$$

with a polynomial whose roots are the desired eigenvalue locations. The next example demonstrates the approach on the simple quadcopter model.

■ **Example 1.6** Let us design a state feedback controller $u_t = -Lx_t = -\begin{pmatrix} l_1 & l_2 \end{pmatrix} x_t$ for the quadcopter model considered in earlier examples. For simplicity, we assume that we would like the closed-loop eigenvalues to be located at $\lambda = 0.5$. The closed-loop system matrix is

$$A - BL = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} - \begin{pmatrix} 0.5 \\ 1 \end{pmatrix} \begin{pmatrix} l_1 & l_2 \end{pmatrix} = \begin{pmatrix} 1-0.5l_1 & 1-0.5l_2 \\ -1_1 & 1-l_2 \end{pmatrix}$$

with characteristic polynomial

$$\lambda(\lambda) = \det \begin{pmatrix} \lambda - 1 + 0.5l_1 & -1 + 0.5l_2 \\ l_1 & \lambda - 1 + l_2 \end{pmatrix} = \lambda^2 + \lambda(l_2 + 0.5l_1 - 2) + 0.5l_1 - l_2 + 1$$

Equating it with the desired $p_{\text{des}}(\lambda) = (\lambda - 0.5)^2 = \lambda^2 - \lambda + 0.25$ gives

$$l_1 = 1/4, \quad l_2 = 7/8.$$

■

The approach that we have used in the example works well for systems of low dimension and allows us to get analytical expressions for the feedback gains. However, since the resulting system of equations may be tedious to solve, we will often resort to numerical computations.

A limitation of pole placement design is that it does not generalize well to systems with multiple inputs. If the system has n states and m control signals, L has $m \times n$ entries, while the closed-loop system still only has n eigenvalues. In contrast, the design methods that we will develop in Chapter 4 extend seamlessly to systems with multiple inputs and outputs.

State estimation

When the full state vector is not measurable, it is natural to try to find an estimate $\hat{x}_t$ of the state vector and use the control strategy $u_t = -L\hat{x}_t$ in place of the nominal state feedback law. State estimates can be computed in several ways. If we were to have a perfect model of the system and knew the initial state x_0, we could set $\hat{x}_0 = x_0$ and estimate future states by simply simulating the system under the applied control sequence $\{u_t\}$

$$\begin{aligned} \hat{x}_{t+1} &= A\hat{x}_t + Bu_t \\ \hat{y}_t &= C\hat{x}_t + Du_t \end{aligned}$$

In practice, x_0 is rarely known and our model is only an approximation of the true system dynamics. It is then more reliable to use a state estimator which corrects the predicted state $\hat{x}_{t+1}$ based on discrepancies between the estimated output $\hat{y}_t$ and the measured y_t:

$$\begin{aligned} \hat{x}_{t+1} &= A\hat{x}_t + Bu_t + K(y_t - \hat{y}_t) \\ \hat{y}_t &= C\hat{x}_t + Du_t \end{aligned}$$

Here, $K \in \mathbb{R}^{n\times p}$ is a gain matrix to be determined. We refer to this particular estimator as a *Luenberger observer*. Note that it uses the input and output sequences, $\{u_0, u_1, \dots, u_{t-1}\}$ and $\{y_0, y_1, \dots, y_{t-1}\}$, together with the system model to predict the state at time t.

The aim of the estimator is to drive the estimation error $e_t = x_t - \hat{x}_t$ to zero. Since e_t evolves as

$$\begin{aligned} e_{t+1} &= x_{t+1} - \hat{x}_{t+1} = \\ &= Ax_t + Bu_t - A\hat{x}_t - Bu_t - KC(x_t - \hat{x}_t) = \\ &= (A-KC)e_t, \end{aligned}$$

the error dynamics of the observer are characterized by the eigenvalues of the matrix $A-KC$, which we can alter using the observer gain K. In particular, if we choose K so that $A-KC$ is Schur stable, then the estimation errors will decay to zero. The next result asserts that we can assign arbitrary error dynamics to the observer if (A,C) is an observable pair.

Theorem 1.4.2 For the linear system (1.1), there exists $K \in \mathbb{R}^{n\times p}$ so that the n eigenvalues of $A-KC$ can be assigned to arbitrary real or complex conjugate locations if and only if the system is observable.

Similarly to the state feedback case, we can design the observer gains by comparing the characteristic equation of $A-KC$ with a polynomial whose roots are the desired eigenvalue locations for the estimation error dynamics.

A limitation of the Luenberger observer is that $\hat{x}_t$ is only based on measurements up until time $t-1$. In fact, the observer is a one-step-ahead predictor rather than an estimator of the current state. When $D=0$, a more efficient state estimate is obtained by correcting $\hat{x}_t$ rather than $\hat{x}_{t+1}$, *i.e.* using

$$\begin{aligned} \hat{x}_{t|t-1} &= A\hat{x}_{t-1|t-1} + Bu_{t-1} \\ \hat{y}_{t|t-1} &= C\hat{x}_{t|t-1} \\ \hat{x}_{t|t} &= \hat{x}_{t|t-1} + K(y_t - \hat{y}_{t|t-1}) \end{aligned}$$

In these expressions, $\hat{x}_{t|s}$ denotes the state estimate at time t based on information until time s. Note that $\hat{x}_{t|t}$ now makes use of the most recent measurement of the system output. A state estimator that uses information up until the present time to estimate the present state is called a *filter*. Note that $D=0$ is needed for us to be able to make a one-step prediction of the output when u_t is unknown. By a similar calculation as above, the estimation error $e_t = x_t - \hat{x}_{t|t}$ evolves according to

$$e_t = (I-KC)(x_t - \hat{x}_{t|t-1}) = (I-KC)Ae_{t-1} = (A-K\tilde{C})e_{t-1}$$

where $\tilde{C} = CA$. We can thus design K just as we did for the Luenberger observer, *i.e.* by assigning eigenvalue locations for $A-K\tilde{C}$. One can also verify that $(A,\tilde{C})$ is observable whenever (A,C) is.

■ **Example 1.7** Let us design a state estimator for the quadcopter model from earlier examples, and design the estimator gain so that the estimation error dynamics have eigenvalues in $\lambda = 0.5$. For the Luenberger observer we determine K such that

$$A - KC = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} - \begin{pmatrix} k_1 \\ k_2 \end{pmatrix} \begin{pmatrix} 1 & 0 \end{pmatrix} = \begin{pmatrix} 1-k_1 & 1 \\ -k_2 & 1 \end{pmatrix}$$

has desired eigenvalues. Since its characteristic polynomial is

$$p(\lambda) = \det \begin{pmatrix} \lambda - 1 + k_1 & -1 \\ k_2 & \lambda - 1 \end{pmatrix} = \lambda^2 + (k_1 - 2)\lambda + 1 - k_1 + k_2$$

Equating it with the desired $p_{\text{des}}(\lambda) = \lambda^2 - \lambda + 0.25$ gives the observer gains

$$k_1 = 1, \quad k_2 = 1/4$$

If we, instead, want to design a filter, then we should place the eigenvalues of

$$A - KCA \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} - \begin{pmatrix} k_1 \\ k_2 \end{pmatrix} \begin{pmatrix} 1 & 1 \end{pmatrix} = \begin{pmatrix} 1-k_1 & 1-k_1 \\ -k_2 & 1-k_2 \end{pmatrix}$$

The characteristic polynomial is

$$\det \begin{pmatrix} \lambda + k_1 - 1 & k_1 - 1 \\ k_2 & \lambda + k_2 - 1 \end{pmatrix} + 1 - k_1\lambda^2 + \lambda(k_1 + k_2 - 2) + 1 - k_1$$

For the desired characteristic equation, we find the gains

$$k_1 = 3/4, \quad k_2 = 1/4.$$

It is instructive to look at the velocity estimates produced for the two observers. Recall that in the absence of inputs, the system dynamics describes a double-integrator system moving up or down with a constant velocity. We can measure the position but not the velocity, so the main task for the observer is to produce a velocity estimate. Straight-forward but tedious computations show that the velocity estimate of the Luenberger observer is on the form

$$\hat{v}_{t+1} = \hat{v}_t + \frac{1}{4}(y_t - \hat{y}_t)$$

Roughly speaking, if the present velocity estimate is too small, $\hat{y}_t$ will be smaller than y_t, and the estimate will be increased. The filter, on the other hand, uses

$$\hat{v}_{t|t} = 0.75\hat{v}_{t-1|t-1} + 0.25(y_t - \hat{y}_{t-1|t-1})$$

and hence maintains a moving average of the velocity estimates obtained by "differentiating" the position signal. This allows the filter to react faster to velocity changes than the one-step ahead predictor. ■

Output feedback

The combination of a state estimator and static linear feedback from the estimated states results in the following output feedback controller:

$$\begin{aligned} \hat{x}_{t+1} &= A\hat{x}_t + Bu_t + K(y_t - C\hat{x}_t) \\ u_t &= -L\hat{x}_t \end{aligned}$$

To emphasize that the controller is an output feedback strategy, i.e. it takes y_t as input and produces u_t as output, we re-write the equations as

$$\begin{aligned} \hat{x}_{t+1} &= (A - BL - KC)\hat{x}_t + Ky_t \\ u_t &= -L\hat{x}_t. \end{aligned}$$

The closed-loop dynamics of (1.1) under this control law is given by

$$\begin{aligned} \begin{pmatrix} x_{t+1} \\ \hat{x}_{t+1} \end{pmatrix} &= \begin{pmatrix} A & -BL \\ KC & A - BL - KC \end{pmatrix} \begin{pmatrix} x_t \\ \hat{x}_t \end{pmatrix} \\ y_t &= \begin{pmatrix} C & -DL \end{pmatrix} \begin{pmatrix} x(t) \\ \hat{x}_t \end{pmatrix} \end{aligned}$$

These dynamics are easier to analyze in terms of the system state and the estimation error:

$$\begin{pmatrix} x_{t+1} \\ e_{t+1} \end{pmatrix} = \begin{pmatrix} A-BL & BL \\ 0 & A-KC \end{pmatrix} \begin{pmatrix} x_t \\ e_t \end{pmatrix}$$
$$y(t) = \begin{pmatrix} C-DL & DL \end{pmatrix} \begin{pmatrix} x_t \\ e_t \end{pmatrix}$$

Since the system matrix is block-diagonal, its eigenvalues equal those of its diagonal blocks. Hence, if we design L so that $A-BL$ is Schur stable, and K so that $A-KC$ is Schur stable, then the closed-loop system will be asymptotically stable. Of course, these conclusions are drawn under the assumption that the model used in the estimator design is a perfect description of the true system.

Coping with disturbances: integral action and feed-forward

Many control systems are affected by disturbances. In some cases, we can measure them and adjust the control signal to counteract their effect. In other cases, we have to rely on feedback to compensate for the disturbance. Indeed, one of the key purposes of feedback control is to ensure a well-defined closed-loop behavior in the presence of disturbances and process variations.

The basic approach to compensate for constant disturbances is to use *integral action*. In discrete time, integration corresponds to summing up the error between the desired and the actual system output. It can be implemented using a controller state i_t that is updated according to

$$i_{t+1} = i_t + (r_t - y_t). \tag{1.13}$$

where r_t is the reference input, *i.e.* the desired target value for the system output y_t. Clearly, if the integral state converges to a steady state where $i_{t+1} = i_t$, we must also have $r_t = y_t$. To make sure that the integral state converges, we use a feedback law on the form

$$u_t = -Lx_t - l_i i_t \tag{1.14}$$

where L and l_i are chosen so that the extended system

$$\bar{x}_{t+1} = \begin{pmatrix} x_{t+1} \\ i_{t+1} \end{pmatrix} = \begin{pmatrix} A & 0 \\ -C & I \end{pmatrix} \begin{pmatrix} x_t \\ i_t \end{pmatrix} + \begin{pmatrix} B \\ -D \end{pmatrix} u_t + \begin{pmatrix} 0 \\ I \end{pmatrix} r_t := \bar{A}\bar{x}_t + \bar{B}u_t + \bar{B}_r r_t$$

is asymptotically stable. We can design the controller gains using pole placement, assigning the desired closed-loop poles to the matrix $\bar{A} - \bar{B}\begin{pmatrix} L & l_i \end{pmatrix} := \bar{A} - \bar{B}\bar{L}$. The actual control signal is then determined using the dynamic controller (1.13) and (1.14).

When the disturbances can be measured, they can be compensated for quicker using feed-forward. To illustrate the ideas, let us first consider the case of reference tracking of a constant output reference r and assume that we use the control signal

$$u_t = -Lx_t + u_{\text{ref}}.$$

We would like to determine u_{ref} so that the system output agrees with the reference in stationarity. In other words, we would like that $y_t \to r$ as $t \to \infty$. Assume that $A-BL$ is Schur stable. Then, $(I - (A-BL))$ is invertible and x_t and y_t will converge to constant vectors x^{ref} and y^{ref} that satisfy

$$x^{\text{ref}} = (A-BL)x^{\text{ref}} + Bu^{\text{ref}}$$
$$y^{\text{ref}} = Cx^{\text{ref}} = C(I-A+BL)^{-1}Bu^{\text{ref}}$$

If also $C(I-A+BL)^{-1}B$ is invertible, then we observe that

$$u^{\text{ref}} = \left[C(I-A+BL)^{-1}B\right]^{-1} r := L_r r \tag{1.15}$$

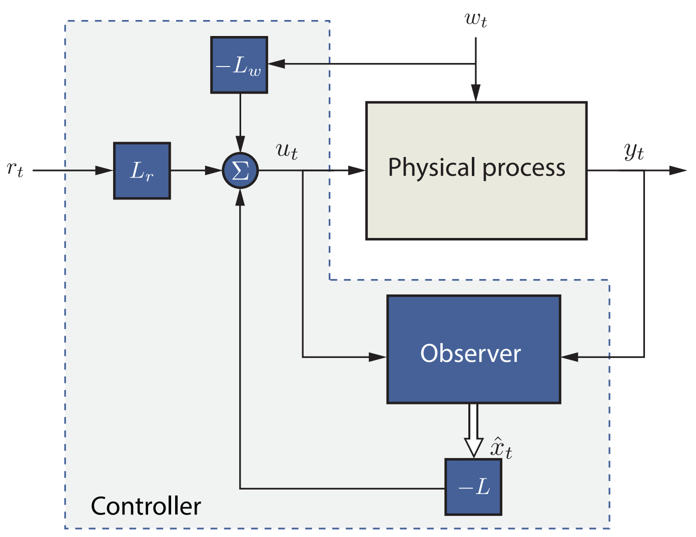


Figure 1.3: Estimator-based output feedback combined with feedforward from exogenous signals.

results in $y^{\text{ref}} = r$, *i.e.* error-free tracking in stationarity.

Another scenario occurs when the system is subject to a disturbance w_t:

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t + B_w w_t \\ y_t &= Cx_t \end{aligned}$$

Proceeding the same way that we did for the reference signal, we note that

$$u_t = -Lx_t + L_w w_t$$

with

$$L_w = -\left[C(I-A+BL)^{-1}B\right]^{-1}\left[C(I-A+BL)^{-1}B_w\right] \tag{1.16}$$

eliminates the effect of w_t on y_t in stationarity.

In many cases, we have both disturbances to suppress and references to track. Since the system is linear, we can design the compensation for each (reference or disturbance) signal in isolation and then combine them to reach the desired effect. Thus, if we have a system that needs state feedback (e.g. to stabilize an unstable physical process) as well as feedforward to track a reference and suppress a measurable disturbance, we will design the components separately and use the control

$$u_t = -Lx_t + u_t^w + u_t^r = -Lx_t - L_w w_t + L_r r_t.$$

A block diagram over the resulting closed-loop system is shown in Figure 1.3.

1.5 Discrete-time descriptions of continuous-time systems

Many models of the physical world are based on ordinary differential equations whose behavior is linear close to a fixed operating point. This makes for a compelling argument to study the control of continuous-time linear systems. However, most modern control systems are realized in digital hardware that operates in discrete time. In this section, we will demonstrate that when the control signal is held constant between sampling instances, there is a discrete-time linear system that describes exactly how the state of an underlying continuous-time linear system evolves from sample to sample. This allows us to use discrete-time linear system theory to analyze and control the continuous-time linear system at sampling instances.

Sampling and reconstruction of continuous-time signals

To understand the limitations in selecting the sampling frequency, consider a continuous-time cosine signal with frequency f Hz

$$y(t) = \cos(2\pi f t).$$

Sampling this signal every h seconds results in the discrete-time representation

$$y_k = y(kh) = \cos(2\pi f kh) \;\text{ for }\; k = 0, 1, \ldots$$

We refer to $f_s = 1/h$ as the *sampling frequency*. Since k is integer-valued, we note that

$$y_k = \cos(2\pi f kh + 2\pi nk) = \cos(2\pi(f + nh^{-1})hk) = \cos(2\pi(f + nf_s)hk)$$

for any positive integer n. In addition, since $\cos(x) = \cos(-x)$, we also have that $\cos(2\pi f hk) = \cos(-2\pi f hk) = \cos(2\pi(-f + nf_s)hk)$. Hence, from the sampled signal, it is impossible to know if the continuous signal has frequency f or $\pm f + nf_s$. This effect is called *aliasing*.

Aliasing can be avoided by sampling sufficiently fast compared to the frequency content in the underlying analog signal. To see how this can be accomplished, assume that the analog signal contains frequencies in the interval $[0, f_{\max}]$. In practice, this assumption is enforced by adding an analog low-pass filter before the analog-to-digital converter that performs the sampling. If we use sampling frequency f_s, then aliasing will appear at frequencies $[-f_{\max}, f_{\max}] + nf_s$. To ensure that we can separate these from the original signal, we have to ensure that $f_{\max} < -f_{\max} + f_s$, *i.e.* that $2f_{\max} < f_s$. The critical frequency $f_s = 2f_{\max}$ is called the *Nyquist frequency* and provides a fundamental lower bound on the sampling rate. This limitation is quite natural: it means that we have to take at least two samples per period of a sinusoidal signal with frequency $f_{\max}$.

The creation of a continuous-time signal (defined for all times) from a discrete-time sequence (defined only at sampling instances) is referred to as *reconstruction*. Most control systems use *zero-order hold* reconstruction, *i.e.* they create a continuous-time signal which is held constant between sampling instances:

$$u(t) = u_k \qquad \text{for } t \in [kh, kh + h).$$

Equivalent discrete-time system under periodic sampling and zero-order hold

We will now show how discrete-time linear systems arise naturally when we want to use a computer to control a physical system. Let the system be described by the continuous-time linear dynamics

$$\dot{x}(t) = A_c x(t) + B_c u(t) \tag{1.17}$$

The solution to these ODEs are given by

$$x(t+h) = e^{A_c h} x(t) + \int_{s=t}^{t+h} e^{A_c(t+h-s)} B u(s)\, ds$$

In particular, if the control input is held constant at some $u(s) = \bar{u}$, then

$$x(t+h) = e^{A_c h} x(t) + \int_{s=t}^{t+h} e^{A_c(t+h-s)} B_c u(s)\, ds = e^{A_c h} x(t) + \left(\int_{v=0}^{h} e^{A_c v} B_c\, dv \right) \bar{u}$$

If we use uniform sampling with interval h, sample k is taken at time $t = kh$, and with zero-order hold, the control is held constant at $\bar{u} = u(t) = u(kh)$. Hence, the state vector at the next sampling instant $kh + h = t + h$ is given by

$$x(kh+h) = Ax(kh) + Bu(kh)$$

where

$$A = e^{A_c h}, \qquad B = \int_{s=0}^{h} e^{A_c s} B_c \, ds.$$

If we drop the reference to physical time and only count sampling instances, we get the model

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k + Du_k \end{aligned}$$

which has the precise form of the discrete-time linear systems that we introduced earlier. This system is an *exact* description of how the state vector of the underlying continuous-time system evolves between sampling instances.

To compute the discrete-time system matrices from the continuous-time ones, we need to compute the matrix exponential. In practice, this is often done numerically. If we want to compute the matrix exponential analytically, we can for example use its power-series definition

$$e^{A_c h} = I + hA_c + \frac{h^2}{2!}A_c^2 + \frac{h^3}{3!}A_c^3 + \cdots$$

Appendix A discusses an alternative technique for computing the matrix exponential based on the Laplace transform, that is typically easier to use than the direct definition.

■ **Example 1.8** In Example 1.1, we mentioned that the continuous-time model for the vertical dynamics of a quadrotor with unit mass was given by

$$\ddot{y}(t) = F(t) - g$$

With $u(t) = F(t) - g$, a state-space description is given by

$$\begin{pmatrix} \dot{x}(t) \\ \dot{v}(t) \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x(t) \\ v(t) \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u(t) := A_c \begin{pmatrix} x(t) \\ v(t) \end{pmatrix} + B_c u(t)$$

Since $A_c^2 = 0$, the discrete-time system under zero-order hold is given by

$$\begin{aligned} A &= e^{A_c h} = I + hA_c + \frac{h^2}{2}A_c^2 + \cdots = I + hA_c = \begin{pmatrix} 1 & h \\ 0 & 1 \end{pmatrix} \\ B &= \int_{s=0}^{h} e^{A_c s} B_c \, ds = \int_{s=0}^{h} \begin{pmatrix} 1 & s \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} ds = \begin{pmatrix} h^2/2 \\ h \end{pmatrix} \end{aligned}$$

■

Guidelines for choosing sampling time

The Nyquist frequency provides a lower bound on the sampling frequency. However, this sampling rate is typically not enough to get a reasonable quality in the closed-loop signals when we use a simple zero-order hold reconstruction. In practice, significantly higher sampling rates are used.

Sample time selection for control involves a compromise between the computational load on the controller (since a new control signal should be computed at every sample interval) and effectiveness in tracking references and rejecting disturbances (since the controller cannot react until the next sampling instance after a disturbance hits the system). Note that even if the computational load is not an issue, too fast sampling may lead to numerical difficulties since $A = e^{A_c h} \to I$ as $h \to 0$. As the off-diagonal elements become increasingly small, it gets difficult to maintain numerical accuracy in operations involving the system matrices. The standard rules-of-thumb instead propose to use a sampling time that results in $4 - 10$ samples per rise-time of the closed-loop system [4], or a sampling frequency $2\pi/h$ of $20 - 40$ times the desired closed-loop bandwidth [16].

Using continuous-time insight to understand transient properties of discrete-time systems

One of the most popular control design techniques for linear systems is based on pole placement. To choose the appropriate closed-loop poles, it is essential to have a good understanding of how the pole locations affect the transient response of the system. In traditional control courses, significant attention is therefore given to developing such an understanding for simple prototype systems. We will leverage this understanding to develop a similar insight for discrete-time systems.

Consider the continuous-time linear system $\dot{x}(t) = A_c x(t)$ and assume that A_c has an eigenvalue λ_c with eigenvector v_c, *i.e.* that $A_c v_c = \lambda_c v_c$. Then the corresponding eigenvalue for the discrete-time system can be computed via the power-series definition of the matrix exponential:

$$Av_c = e^{A_c h} v_c = \left(I + hA_c + \frac{h^2}{2} A_c^2 + \cdots \right) v_c = \left(1 + h\lambda_c + \frac{h^2}{2} \lambda_c^2 + \cdots \right) v_c = e^{\lambda_c h} v_c$$

This means that the discrete-time system matrix A has an eigenvalue at $\lambda = e^{\lambda_c h}$ with eigenvector v_c. In particular, we notice the following

- a real eigenvalue $\lambda_c = -\omega_0$ for A_c maps to a real eigenvalue $\lambda = \exp(-\omega_0 h)$ for A
- a purely imaginary eigenvalue $\lambda_c = \pm i\omega_0$ of A_c maps into an eigenvalue on the unit circle

  $$\lambda = e^{\pm i\omega_0 h} = \cos(\omega_0 h) \pm i \sin(\omega_0 h)$$

  for A. Note that due to aliasing, we should keep $h \leq \pi/\omega_0$.
- a complex-conjugate eigenvalue on the form $\lambda_c = \omega_0(-\cos\theta + i\sin\theta)$ maps onto

  $$\lambda = e^{-\omega_0 h \cos\theta} e^{i\omega_0 h \sin\theta}$$

In other words, λ lies on a logarithmic spiral determined by θ and ω_0 when we change h. The relationship between eigenvalue locations for A_c and A are shown in Figure 1.4.

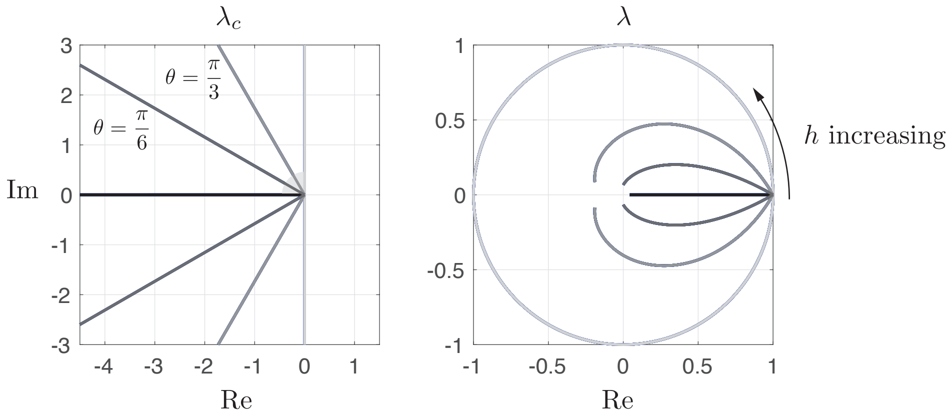


Figure 1.4: Mapping of eigenvalues λ_c for A_c into corresponding eigenvalues λ for A.

Sampling of systems with time delays

In digital control systems, it will always take some time from when the sensor signals are read until a new control action is computed and actuated. One advantage of discrete-time systems is that delays (which are infinite-dimensional in continuous-time) can be described by finite-dimensional state-space models.

Let us model the computational delay as an input delay of τ seconds and consider the system

$$\begin{cases} \dot{x}(t) &= A_c x(t) + B_c u(t-\tau) \\ y(t) &= Cx(t) + Du(t) \end{cases} \tag{1.18}$$

Assume that the computational delay is shorter than the sampling time, i.e. that $\tau \in [0,h]$. Under zero-order hold sampling, synchronized with the sensor readings, this means that the previous control signal $u(kh-h)$ is applied for the first τ seconds until $u(kh)$ begins to affect the system. Similarly to the delay-free case, integration of the solution to (1.18) yields

$$x(kh+h) = e^{A_c h}x(kh) + \int_{s=0}^{\tau} e^{A_c s}B_c u(kh-h)\,ds + \int_{s=\tau}^{h} e^{A_c s}B_c u(kh)\,ds$$

By dropping reference to physical time, and defining $x_k = x(kh)$, $u_{k-1} = u(kh-h)$ and $u_k = u(kh)$, we can represent the dynamics as

$$\begin{pmatrix} x_{k+1} \\ u_k \end{pmatrix} = \begin{pmatrix} A & B_1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x_k \\ u_{k-1} \end{pmatrix} + \begin{pmatrix} B_0 \\ I \end{pmatrix} u_k$$

where

$$A = e^{A_c h}, \qquad B_0 = \int_{s=0}^{\tau} e^{A_c s}B_c\,ds, \qquad B_1 = \int_{s=\tau}^{h} e^{A_c s}B_c\,ds.$$

Hence, the zero-order hold equivalent of the input delayed system (1.18) can be represented by an $n+m$-dimensional discrete-time linear system with state vector (x_k, u_{k-1}). By designing the controller for this dynamics, we are able to compensate for the computational delays.

If the input delay exceeds the sampling time, *i.e.* $\dot{x}(t) = A_c x(t) + B_c u(t-\tau')$ with $\tau' = lh + \tau$ for some positive integer l and some $\tau \in [0,h)$, then one can first sample the system as above, noting that it is only $u_{k-l-1} = u(t-(l+1)h)$ and $u_{k-l} = u(t-lh)$ that affect the state during sample interval k. The sampled system can then be described by the state-space model

$$\begin{pmatrix} x_{k+1} \\ u_{k-l} \\ u_{k-l+1} \\ \vdots \\ u_{k-1} \\ u_k \end{pmatrix} = \begin{pmatrix} A & B_1 & B_0 & 0 & \cdots & 0 \\ 0 & 0 & I & 0 & \ddots & 0 \\ 0 & 0 & 0 & I & 0 & 0 \\ \vdots & \ddots & \ddots & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 & I \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_k \\ u_{k-l-1} \\ u_{k-l} \\ \vdots \\ u_{k-2} \\ u_{k-1} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ I \end{pmatrix} u_k$$

This state-space model has dimension $n+(l+1)m$, but is still of finite dimension.

1.6 Design example: level control of a double tank

To get some experience of discrete-time control design, we will consider the two-tank system in Figure 1.6. We would like to design a controller that adjusts the inflow to the first tank in order to maintain a desired level of the second tank despite the presence of an exogenous disturbance flow $w(t)$. The controller should have a rise time of around 40 seconds.

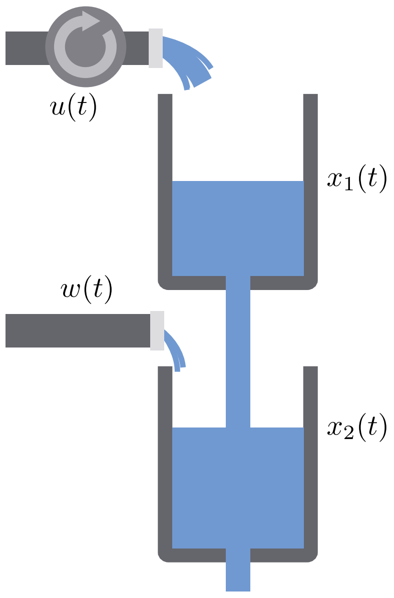


Figure 1.5: Double tank.

After linearization, the tank dynamics can be described by the second-order system

$$\dot{x}(t) = \begin{pmatrix} -0.07 & 0 \\ 0.07 & -0.07 \end{pmatrix} x(t) + \begin{pmatrix} 0.18 \\ 0 \end{pmatrix} u(t) + \begin{pmatrix} 0 \\ 0.07 \end{pmatrix} w(t)$$

Here, the first state variable is the level in the upper tank, the second state is the level in the lower tank, $u(t)$ is the voltage applied to the pump that generates the inflow in the upper tank, and $w(t)$ is the disturbance inflow into the lower tank. We use uniform sampling of the sensor signals and zero-order hold to reconstruct a continuous control signal.

With the desired rise time of 40 seconds, the rules-of-thumb for sample time selection suggest to use $h = 4 - 10$ seconds; we will settle for $h = 5$. Using the formulas derived in Section 1.5, we find that the corresponding discrete-time system is described by

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t + B_w w_t \\ y_t &= Cx_t \end{aligned}$$

where

$$A = \begin{pmatrix} 0.7047 & 0 \\ 0.2466 & 0.7047 \end{pmatrix}, \qquad B = \begin{pmatrix} 0.7594 \\ 0.1252 \end{pmatrix}, \qquad B_w = \begin{pmatrix} 0 \\ 0.2953 \end{pmatrix}, \qquad C = \begin{pmatrix} 0 & 1 \end{pmatrix}$$

We begin by disregarding the presence of the disturbance and design a state feedback

$$u_t = -Lx_t = -\begin{pmatrix} l_1 & l_2 \end{pmatrix} x_t$$

to make the closed-loop dynamics slightly faster, *i.e.* move the closed-loop poles a little closer to the origin. We will aim at placing both system poles at $\lambda = 0.5$. To find the corresponding feedback gains, we will have to ensure that the characteristic equation

$$\begin{aligned} p(\lambda) &= \det(\lambda I - (A - BL)) = \\ &= \lambda^2 + \lambda(-1.4094 + 0.7594 l_1 + 0.1252 l_2) + (0.4966 - 0.5351 l_1 + 0.0990 l_2) \end{aligned}$$

is equal to the desired $p_{\text{des}}(\lambda) = (\lambda - 0.5)^2 = \lambda^2 - \lambda + 0.25$. Equating the coefficients of the two polynomials leads to the following system of linear equations

$$\begin{cases} -1.4094 + 0.7594 l_1 + 0.1252 l_2 & = & -1 \\ 0.4966 - 0.5351 l_1 + 0.0990 l_2 & = & 0.25 \end{cases}$$

with solution $l_1 = 0.5022$ and $l_2 = 0.2237$. We also compute a reference feed-forward gain $L_r = 1.1148$ using (1.15). Figure 1.6 shows a simulation of the closed-loop system under a reference change from at time $t = 0$ and the addition of a constant input disturbance w at time $t = 50$. Note how the rise-time of the lower tank is around the desired 40 seconds and that the feed-forward ensures that the reference is followed without error. However, the controller is not very effective in dealing with the disturbance.

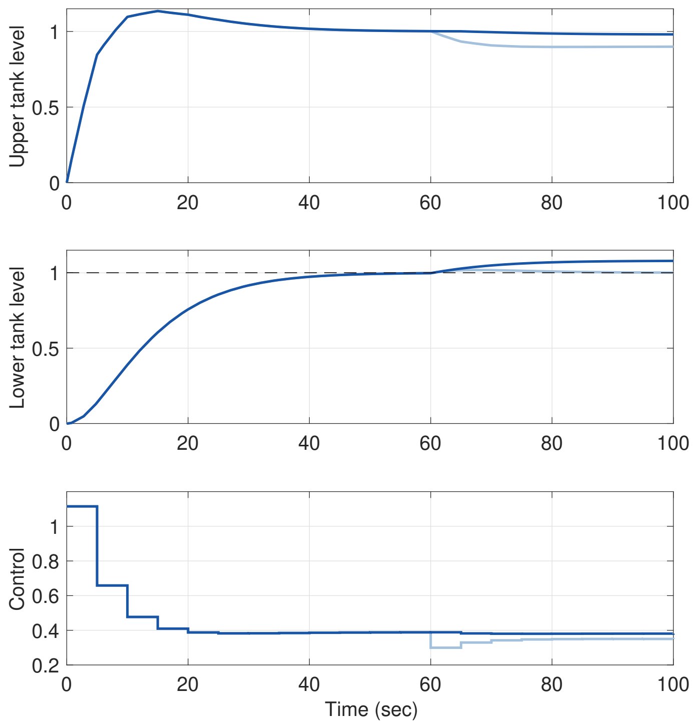


Figure 1.6: The state feedback and reference feedforward succeeds in shaping the closed-loop response (middle axes, dark color): the rise-time is around 40 seconds and the reference is followed without stationary error. However, without feedforward from the disturbance, the additional inflow to the lower tank causes a remaining error (same figure, dark color after 60 seconds). Adding the feedforward from the disturbance elimitates the effect (light color, same plot).

Assuming that we can add a sensor which measures the disturbance inflow, we can use feed-forward to compensate for it. Using Equation (1.16), we compute the feed-forward gain

$$L_w = -0.8911$$

As seen in Figure 1.6, the feedforward compensation does eliminate the effect of the disturbance.

Since we only want to measure the level of the lower tank, we construct a state observer whose poles are slightly faster than the ones used in our state feedback design. We thus consider the output

$$y_t = Cx_t = \begin{pmatrix} 0 & 1 \end{pmatrix} x(t)$$

For purposes of illustration, we choose to place the observer poles (the eigenvalues of $A - KC$) at $\lambda = 0.4$. A similar calculation as for the state feedback design results in the estimator gain matrix

$$K = \begin{pmatrix} 0.3765 \\ 0.6095 \end{pmatrix}$$

By combining the state estimator and feedback from the estimated states, we have designed an output feedback controller on the form

$$\begin{aligned} \hat{x}_{t+1} &= A\hat{x}_t + Bu_t + B_w w_t + K(y_t - C\hat{x}_t). \\ u_t &= -L\hat{x}_t + L_w w_t + L_r r_t \end{aligned}$$

Figure 1.7 shows the closed-loop response of the output feedback controller to a step-change in the reference, followed by a step change in the disturbance inflow to the second tank. After 100 seconds, we add a measurement noise (modeled as a Gaussian random variable with zero mean and variance of 0.1) to the measured output signal. The initial response is just as before, but we can see that the controller reacts to the measurement noise, causing slight ripples in the level of the second tank. If we move the observer poles closer to the origin, placing both at $\lambda = 0.2$, there is no visible change in the initial response, but the controller becomes more sensitive to the noise, leading to larger variations in the control input and the resulting tank level (note that the figure shows the actual tank level and not the noise-corrupted measurement signal).

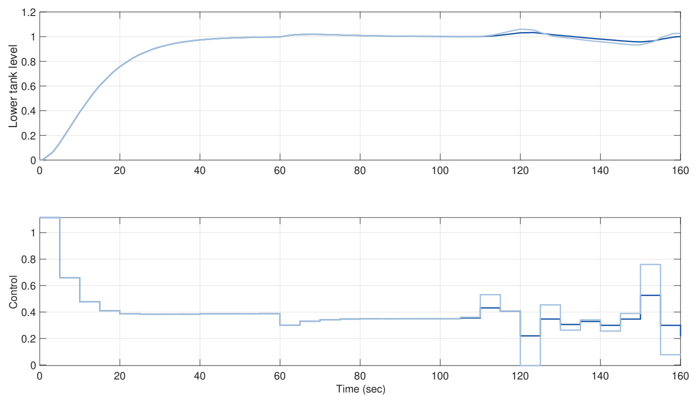


Figure 1.7: Output feedback response to set-point change at time 0, inflow disturbance at time 60, and a measurement noise signal from time 100 and on. The reference and disturbance response is almost identical to that of the state-feedback (light and dark colors), but the controller becomes more sensitive to the measurement noise when the observer poles are made faster (light color).

1.7 Input-output properties of discrete-time linear systems*

Although most of this course considers state-space models, one can develop considerable insight into the properties of discrete-time linear systems using input-output models. To this end, this section contains a brief introduction to the Z-transform, transfer functions for discrete-time linear systems, and a few words about their frequency responses.

The Z-transform

A discrete-time signal $s(k)$ can be represented by a list of real numbers (or vectors), $\{s_0, s_1, \dots\}$. The analysis of discrete-time signals and systems is sometimes simplified by using the *Z-transform*:

Definition 1.7.1 The *z-transform* $\mathcal{Z}(s)$ of the discrete-time signal $s(k) = \{s_0, s_1, \dots\}$ is

$$\mathcal{Z}(s) = \sum_{k=0}^{\infty} s_k z^{-k}$$

Note that the Z-transform maps the space of discrete-time signals to the space of functions over the complex plane (or, rather, over a subset of the complex plane for which the summation converges). To emphasize this, we will write the Z-transform of a signal $s(k)$ as $S(z)$. The next example derives the Z-transform for some common signals.

■ **Example 1.9** The unit step

$$\mathbf{1}(k) = \begin{cases} 1 & \text{if } k \geq 0 \\ 0 & \text{if } k < 0 \end{cases}$$

has Z-transform

$$\mathcal{Z}(\mathbf{1}) = \sum_{k=0}^{\infty} z^{-k} = \frac{1}{1-z}$$

with region of convergence $|z| < 1$. The geometric sequence

$$s(k) = a^k \mathbf{1}(k)$$

has Z-transform

$$\mathcal{Z}[s] = \frac{z}{z-a}$$

with region of convergence $|z| > |a|$. ■

From the definition, one can also derive many important properties of the Z-transform, some of which are summarized in Table 1.1.

Transfer functions of discrete-time linear systems

We can now apply the Z-transform to the state-space model

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k + Du_k \end{aligned}$$

with initial state x_0. For convenient notation, we define $X(z) = \mathcal{Z}[x(k)]$, $U(z) = \mathcal{Z}[u(k)]$ and $Y(z) = \mathcal{Z}[y(k)]$. Then, by the linearity and forward-shift property of the Z-transform

$$\begin{aligned} zX(z) - zx_0 &= AX(z) + BU(z) \\ Y(z) &= CX(z) + DU(z) \end{aligned}$$

Signal	Transform	Name
$\alpha s_1(k)+\beta s_2(k)$	$\alpha S_1(z)+\beta S_2(z)$	Linearity
$s(k+1)$	$zS(z)-zs(0)$	Forward shift
$s(k-1)$	$z^{-1}S(z)+s(-1)$	Backward shift
$s(k)=\sum_{l=0}^{k} x_l$	$S(z)=\frac{z}{z-1}X(z)$	Accumulation
$\lim_{k\to\infty} s(k)$	$\lim_{z\to 1}(z-1)Y(z)$	Final-value theorem
$\lim_{k\to 0} s(k)$	$\lim_{z\to\infty} S(z)$	Initial-value theorem
$s(k)=\sum_{l=0}^{k} x[l]y[k-l]$	$S(z)=X(z)Y(z)$	Convolution

Table 1.1: Key properties of the Z-transform.

from which we find

$$Y(z)=C(zI-A)^{-1}zx_0+(C(zI-A)^{-1}B+D)U(z):=C(zI-A)^{-1}zx_0+G(z)U(z)$$

The first term in this expression is the Z-transform of the free system response, while the second one is the Z-transform of the driven response. If $x_0=0$, then $Y(z)=G(z)U(z)$ where $G(z)$ is called the pulse-transfer function of the system. We define it below for easy reference

Definition 1.7.2 The *pulse-transfer function* of the discrete-time linear system (A,B,C,D) is

$$G(z)=C(zI-A)^{-1}B+D.$$

Recall that the inverse of a matrix is the ratio between its adjugate matrix and its determinant, *i.e.* $M^{-1}=\mathrm{adj}(M)/\det(M)$. This allows us to re-write the transfer function expression as

$$G(z)=\frac{C\mathrm{adj}(zI-A)B}{\det(zI-A)}+D$$

Hence, the characteristic polynomial of the system matrix A appears in the denominator of the transfer function (or transfer matrix elements, in case there are many inputs or outputs). When the linear system has a single input and a single output, then

$$G(z)=\frac{B(z)}{A(z)}=\frac{b_0z^m+\cdots+b_{m-1}z+b_m}{z^n+a_1z^{n-1}+\cdots+a_n}$$

The roots of $B(z)$ are called *zeros* of the transfer function, while the roots of $A(z)$ are known as *poles*. In the absence of cancellations between numerator and denominator, the poles of the transfer functions are exactly the eigenvalues of the system matrix A in the state-space description. While the relationship between continuous-time and discrete-time poles is well-defined, the relationship between continuous-time and discrete-time zeroes is more involved. In particular, as the next example illustrates, a discrete-time pulse transfer function can have zeroes, even if the underlying continuous-time transfer function does not.

■ **Example 1.10** With $u(t)=F(t)-mg$ and $m=1$, the continuous-time dynamics of the vertical dynamics of the quadrotor is $\ddot{y}(t)=u(t)$ with transfer function

$$G(s)=\frac{1}{s^2}$$

We can recognize this as a double integrator (two poles at $s=0$) without any zeros. The pulse-transfer function for the corresponding discrete-time system (cf. Example 1.8) with $h=1$ is

$$G(z)=C(zI-A)^1B=\frac{1}{2}\frac{z+1}{(z-1)^2}.$$

As can be expected, the system has a double pole at $z = e^0 = 1$, but it also has a zero at $z = -1$. This *sampling zero* can appear counter-intuitive at first, but it is a consequence of the sample-and-hold operation. In particular, recall that a system zero implies that there is an initial state and a specific input sequence such that the output remains at zero. For this system, it is easy to verify that $x_0 = \begin{pmatrix} 0 & -1/2 \end{pmatrix}$ and $u_t = (-1)^t$ results in $y_t \equiv 0$ for all t. ■

The frequency response of a discrete-time linear system

As we saw in the discussion about Nyquist sampling, a discrete-time sinusoid takes the form

$$u(k) = \cos(\omega_s k + \theta_0)$$

where $\omega_s = 2\pi f h$ is the angular frequency measured in radians per sample. Note that the lowest possible rate of variation for this signal is $\omega_s = 0$, which corresponds to a constant. The highest rate of variation happens for $\omega_s = \pm\pi$ and $\theta_0 = 0$, when the signal alternates sign at each time step, *i.e.* $u(k) = (-1)^k$. Thus, the interesting range for discrete-time sinusoidals is $[-\pi, \pi]$ radians per sample, which corresponds to $[-\pi/h, \pi/h]$ radians per second; outside this frequency range, the spectrum repeats periodically in ω_s.

By a similar calculation as for continuous-time systems, one can show that the output of an asymptotically stable linear systems driven by an input $u_k = \cos(\omega_s k)$ satisfies

$$y_k = |G(e^{i\omega_s})| \cos(\omega_s k + \arg G(e^{i\omega_s}))$$

once the transient has died out. Hence, one refers to $G(e^{i\omega_s})$ for $\omega_s \in [0, \pi]$ as the *frequency response* of G. As in the continuous-time case, one typically visualizes the frequency response using Bode or Nyquist diagrams. However, in contrast to the continuous-time case, there are no simple rules for drawing these diagrams by hand since $G(e^{i\omega_s})$ is irrational in ω_s, but one has to resort to numerical computations and visualization.

■ **Example 1.11** The pulse transfer function from pump voltage to lower tank level of the double tank system studied in Chapter 1.6 is

$$H(z) = \begin{pmatrix} 0 & 1 \end{pmatrix} \left(zI - \begin{pmatrix} 0.7047 & 0 \\ 0.2466 & 0.7047 \end{pmatrix} \right)^{-1} \begin{pmatrix} 0.7594 \\ 0.1252 \end{pmatrix} = \frac{0.1252z + 0.0991}{z^2 - 1.409z + 0.4966}$$

Figure 1.8 shows the frequency response $H(e^{i\omega_s})$ as a Bode diagram (left) and a Nyquist diagram (right). For comparison, we have also illustrated the frequency response of the continuous-time dynamics of the tank. ■

Just as for continuous-time systems, the frequency response can also be used to study stability and robustness of discrete-time systems. Consider the standard feedback loop in Figure 1.9 (left). Stability of the closed-loop pulse-transfer function $H_{\text{cl}}(z) = L(z)/(1 + L(z))$ can be established from the frequency response of $L(z)$ using the following simplified Nyquist criterion.

Theorem 1.7.1 If $L(z)$ and its inverse are stable, then $L(z)/(1 + L(z))$ is stable if the point $(-1, 0)$ in the complex plane is left of the Nyquist curve $(\operatorname{Re} L(e^{i\omega}), \operatorname{Im} L(e^{i\omega}))$ for all $\omega \in [0, \pi]$.

To ensure stability also in the face of modelling errors, it is advisable that the Nyquist curve avoids the critical -1 point with some margin. Two classical measures of the distance to the critical point are the *amplitude* and *phase margins*. Let ω_0 be such that $\arg(L(e^{i\omega_0})) = -\pi$, then we define the amplitude margin as $A_m = 1/|L(e^{i\omega_0})|$; it measures how much the gain of L can be increased before the system goes unstable. Similarly, with ω_c such that $|L(e^{i\omega_c})| = 1$ we can define the phase margin as $\varphi_m = \pi + \arg(L(e^{i\omega_c}))$. This quantity characterizes how much extra phase lag the system

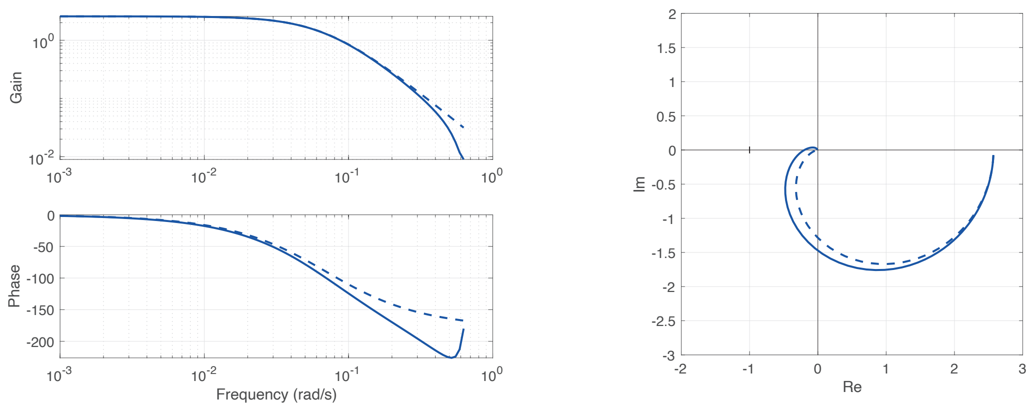
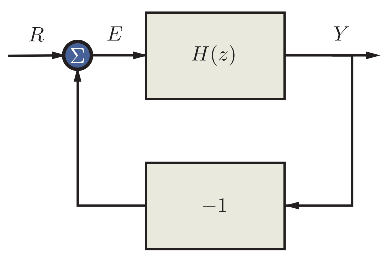


Figure 1.8: The frequency response of the continuous-time tank model (dashed lines) and the discrete-time model obtained under sample-and-hold (full). Note that the discrete-time model loses phase and high frequency gain, and that the relevant frequency range is $[0, \pi/5]$ radians/sec.

can tolerate before it goes unstable, see Figure 1.9 (right). A more comprehensive measure is the inverse of the minimal distance from the Nyquist curve to the -1 point, *i.e.*

$$S_{\max} = \frac{1}{\min_{\omega \in [0,\pi]} |1 + L(e^{i\omega})|} = \max_{\omega \in [0,\pi]} \frac{1}{|1 + L(e^{i\omega})|}$$

You may recognize this as the maximum value of the sensitivity function $S(z) = 1/(1 + L(z))$.



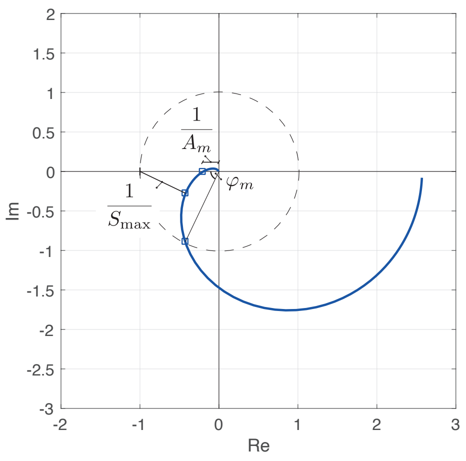
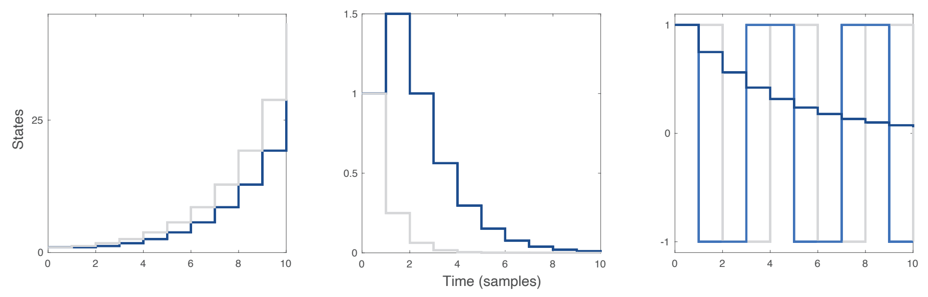


Figure 1.9: The simple feedback loop (left) and its robustness measures amplitude margin (A_m), phase margin φ_m and maximum sensitivity $S_{\max}$.

1.8 Exercises

Problem 1.1 For each of the system matrices below, compute their eigenvalues and determine if the matrix is Schur stable. Figure 1.1 shows the state evolution of $x_{t+1} = Ax_t$ from $x_0 = \mathbf{1}$ for the three different systems. Pair each of the system matrices with one of the initial value responses.

$$\text{(i)} \quad A = \begin{pmatrix} 0.5 & 1 \\ 0 & 0.25 \end{pmatrix} \quad \text{(ii)} \quad A = \begin{pmatrix} 0 & 1 \\ -0.75 & 2 \end{pmatrix} \quad \text{(iii)} \quad A = \begin{pmatrix} 0.9 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix}$$



‘

Problem 1.2 The free dynamics of the quadcopter dynamics used in many of the examples is

$$x_{t+1} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} x_t$$

(a) Compute the eigenvalues of the system matrix. Is the system asymptotically stable?
(b) Compute the eigenvectors of the system matrix. Is the system stable?
(c) In this model, the first state is the vertical position y_t of the quadcopter, while the second state is its vertical velocity v_t. Determine explicit expressions of how the system states evolve from an initial (y_0, v_0). Is the system stable?

Problem 1.3 Consider the discrete-time linear system.

$$x_{t+1} = Ax_t + Bu_t$$

(a) Express the state vector at time $t = 2$ on the form

$$x_2 = H \begin{pmatrix} u_0 \\ u_1 \end{pmatrix} + h$$

for some constant matrix H and some vector h which depends on the initial state x_0.
(b) Let

$$A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

and set $x_0 = \begin{pmatrix} 0 & 0 \end{pmatrix}$. Find inputs u_0 and u_1 that make $x_2 = \begin{pmatrix} 1 & 0 \end{pmatrix}$.
(c) Let

$$A = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, \quad B = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

and set $x_0 = \begin{pmatrix} 0 & 0 \end{pmatrix}$. Can you find inputs u_0 and u_1 that make $x_2 = \begin{pmatrix} 0 & 1 \end{pmatrix}$? Redo the calculations with $x_0 = \begin{pmatrix} 0 & 1 \end{pmatrix}$ and $x_2 = \begin{pmatrix} 0 & 0 \end{pmatrix}$.

Problem 1.4 Consider the system

$$x_{t+1} = \begin{pmatrix} a & b \\ 0 & c \end{pmatrix} x_t + \begin{pmatrix} d \\ e \end{pmatrix} u_t$$

where a, b, c, d and e are scalars.

(a) Determine the controllability matrix of the system and derive conditions on a, b, c, d and e which render the system *unreachable*.

(b) Discuss qualitatively the reason for loss of reachability when: *(i)* $e = 0$; *(ii)* $b = 0$ and $d = 0$; *(iii)* $b = 0$ and $c = a$. Reason in terms of which states the control signal can affect, or which state changes that the control signal can give rise to.

Problem 1.5 For each of the linear systems defined by the matrices below, determine if they are reachable, controllable, or stabilizable.

$$\begin{aligned}
&\text{(a)} \quad A = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \qquad && B = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \\
&\text{(b)} \quad A = \begin{pmatrix} 3 & 1 \\ -2.5 & -0.5 \end{pmatrix}, && B = \begin{pmatrix} 1 \\ -1 \end{pmatrix} \\
&\text{(c)} \quad A = \begin{pmatrix} 2 & 0 \\ -0.5 & 0.5 \end{pmatrix}, && B = \begin{pmatrix} 0 \\ 1 \end{pmatrix}
\end{aligned}$$

Problem 1.6 Use the PBH conditions for reachability to prove the following claim:

> "Let A, B and L be matrices of compatible dimensions. Then, $(A - BL, B)$ is reachable if and only if (A, B) is reachable."

This result implies that a state feedback $u_t = -Lx_t$ does not alter reachability properties.

Problem 1.7 Consider the system

$$x_{t+1} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} x_t + \begin{pmatrix} 1 \\ 0 \end{pmatrix} u_t, \qquad y_t = \begin{pmatrix} 0 & 1 \end{pmatrix} x_t.$$

You apply the input sequence $u_0 = 1, u_1 = -2$ and observe $y_0 = 1$ and $y_1 = 2$.

(a) Determine the initial state x_0.

(b) Determine y_2, the output value at time $t = 2$.

(c) Use the observability matrix rank test to determine if the system is observable.

Problem 1.8 In analogue to Problem 1.4 determine conditions on (a, b, c, d, e) under which the following system is observable

$$x_{t+1} = \begin{pmatrix} a & 0 \\ b & c \end{pmatrix} x_t, \qquad y_t = \begin{pmatrix} d & e \end{pmatrix} x_t$$

Discuss qualitatively the reason for why observability is lost in following cases: *(i)* $e = 0$; *(ii)* $b = 0$ and $d = 0$; *(iii)* $b = 0$ and $c = a$. Reason in terms of states, or combinations of states, which cannot be seen in the output.

Problem 1.9 Let the n-the order system

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t \\ y_t &= Cx_t + Du_t \end{aligned}$$

be observable. Show that we can determine the initial state x_0 from measurements alone (*i.e.* without knowing the applied input) if

$$D = CB = CAB = \cdots = CA^{n-2}B = 0$$

Explain why this condition is also necessary when there is only one output (y_t is scalar).

Problem 1.10 Consider the discrete-time linear system

$$x_{t+1} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} x_t + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u_t$$

Determine a state feedback $u_t = -Lx_t$ that places both eigenvalues of $A - BL$ at 0.5.

Problem 1.11 The discrete-time dynamics of a double integrator is

$$\begin{aligned} x_{t+1} &= \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} x_t + \begin{pmatrix} 1/2 \\ 1 \end{pmatrix} u_t \\ y_t &= \begin{pmatrix} 1 & 0 \end{pmatrix} x_t \end{aligned}$$

(a) Design a state feedback law $u_t = -Lx_t$ that places the closed-loop poles at $\lambda = 0.5$.
(b) Compute a feed-forward gain l_r such that the control law $u_t = -Lx_t + l_r r$ renders the system output y_t to the reference value r in stationarity.
(c) Determine the gain K of the state observer

$$\begin{aligned} \hat{x}_{t+1} &= A\hat{x}_t + Bu_t + K(y_t - \hat{y}_t), \\ \hat{y}_t &= C\hat{x}_t \end{aligned}$$

so that the eigenvalues of $A - KC$ are located at $\lambda = 0.2$.
(d) Write down the state-space equations for an output feedback controller that measures r and y_t, estimates the states of the system using the observer in (c) and determines $u_t = -L\hat{x}_t + l_r r$. Draw a block-diagram of the resulting closed-loop system.
(e) Simulate the system under full state feedback and under output feedback. Apply a step-change in the reference at time $t = 0$ and measurement noise (generated as a sequence of Gaussian variables with zero mean and variance 0.1) to the output measured by the observer.

Problem 1.12 The system

$$\begin{aligned} x_{t+1} &= \begin{pmatrix} 0.878 & 0.478 \\ -0.478 & 0.878 \end{pmatrix} x_t + \begin{pmatrix} 0.122 \\ 0.479 \end{pmatrix} w_t, \\ y_t &= \begin{pmatrix} 1 & 0 \end{pmatrix} x_t + v_t \end{aligned}$$

represents a harmonic oscillator whose state evolution is driven by a noise process $\{w_t\}$ and whose output measurements are corrupted by a noise sequence $\{v_t\}$. In this exercise we model each w_t as an independent zero-mean Gaussian random variable with variance 0.4 and each v_t as an independent zero-mean Gaussian random variable with variance 0.1.

Design the gain vector K for a one-step ahead predictor

$$\hat{x}_{t+1} = A\hat{x}_t + K(y_t - \hat{y}_t), \qquad \hat{y}_t = C\hat{x}_t$$

that places the eigenvalues of the error dynamics in 0.6. Then design a filter

$$\begin{aligned} \hat{x}_{t|t-1} &= A\hat{x}_{t-1|t-1} \\ \hat{x}_{t|t} &= \hat{x}_{t|t-1} + K(y_t - \hat{y}_{t|t-1}) \end{aligned}$$

so that its error dynamics has both eigenvalues in 0.6.

Simulate the two observers for the same realizations of the noise sequences. Compare their performance both visually, and in terms of the quantity $\sum_t (y_t - \hat{y}_t)^2$. What do you observe?

Problem 1.13 Use the sample and hold method to compute the discrete-time equivalents of the following continuous-time linear systems.

(a) The integrator

$$\dot{x}(t) = u(t)$$

(b) A first-order system with a pole in $s = -\alpha$ and stationary gain β:

$$\dot{x}(t) = -\alpha x(t) + \alpha\beta u(t)$$

(c) The double integrator

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

(d) The harmonic oscillator

$$\ddot{y}(t) = -\omega^2 y(t) + u(t)$$

Problem 1.14 We have shown that if the input to a continuous-time linear system

$$\dot{x}(t) = A_c x(t) + B_c u(t)$$

is kept constant between sampling instances, $t = kh$ for $k = 0, 1, \dots$, then the state vector $x_t = x(kh)$ evolves according to

$$x_{t+1} = Ax_t + Bu_t$$

where $A = e^{A_c h}$ and $B = \int_{s=0}^{h} e^{As} B\, ds$. Hence, A has a simple expression in terms of the matrix exponential, but B looks more complicated to evaluate numerically. Show that both A and B can be computed by evaluating a single matrix exponential.

Hint: Recall that the constant $u(t) = u$ can be modelled by the ODE $\dot{u}(t) = 0$ with $u(0) = u$.

Problem 1.15 Yet another way to compute the zero-order-hold equivalent of

$$\dot{x}(t) = A_c x(t) + B_c u(t)$$

can be developed for systems where A_c is invertible. In particular, show that if A_c is invertible, then

$$B = \int_{s=0}^{h} e^{A_c s} B_c\, ds = (e^{A_c h} - I) A_c^{-1} B_c$$

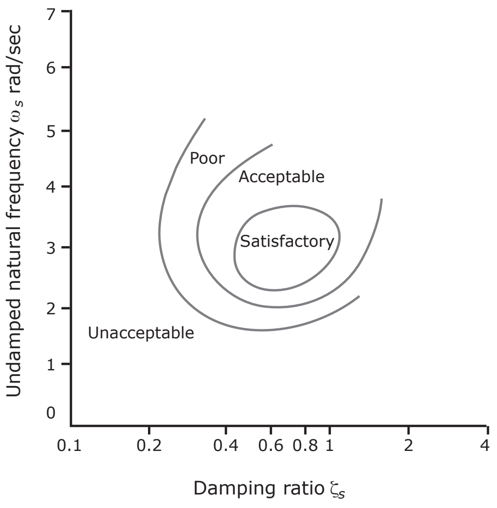
Hint. Use the definition of the matrix exponential to show that $(\int_{s=0}^{h} e^{A_c s}\, ds) A_c + I = e^{A_c h}$.

Problem 1.16 The short-period dynamics of an aircraft describes rapid changes in the angle of attack, triggered by sudden up-gusts and abrupt elevator movements. The following model describes the short-period dynamics of a commercial aircraft flying at 40,000 ft and Mach 0.8, obtained by zero-order hold sampling of the continuous-time model with sampling time $h = 1$ second:

$$\begin{pmatrix} \alpha_{t+1} \\ q_{t+1} \end{pmatrix} = \begin{pmatrix} 1 & 0.1 \\ -0.1 & 0.95 \end{pmatrix} \begin{pmatrix} \alpha_t \\ q_t \end{pmatrix} + \begin{pmatrix} 0 \\ -0.2 \end{pmatrix} \delta_t,$$

$$y_t = \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} \alpha_t \\ q_t \end{pmatrix}$$

Here, α is the angle of attack, q is the pitch rate and δ is the elevator deflection.



(a) Is the system reachable?

(b) Data collected since the 1950's have shown that pilots prefer short-period dynamics which satisfy the "thumbprint criterion" illustrated in Figure 1.16. The desired continuous-time dynamics should thus have a characteristic polynomial $\lambda(p) = p^2 + 4.2p + 9$. Use the relationship between continuous-time and discrete-time eigenvalue locations to determine the corresponding desired discrete-time characteristic polynomial.

(c) Design a state feedback

$$\delta_t = -L\begin{pmatrix}\alpha_t \\ q_t\end{pmatrix} \tag{1.19}$$

that places the closed-loop poles in the locations determined in (b).
Please proceed in two steps: first derive a set of equations in the feedback gains L which ensure that the closed-loop has a characteristic polynomial $p^2 + ap + b$. Then use your results from (b) to obtain numerical values for L.

(d) Compute a feed-forward gain such that the steady-state gain l_r such that the closed-loop dynamics under

$$\delta_t = -L\begin{pmatrix}\alpha_t \\ q_t\end{pmatrix} + l_r r_t$$

achieves $\alpha_t = r_t$ in stationarity (assuming that r_t is constant)

Problem 1.17 In this problem, we will design a control system for the elevation θ of an antenna used to track a satellite in the sky. Its dynamics is given by

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

Here u is the torque that we can generate using the motors, and w is a disturbance torque.

(a) Discretize the system using zero-order hold sampling with a sampling period of h seconds. Validate that the discrete-time system is on the form

$$x_{t+1} = \begin{pmatrix}1 & a_{12} \\ 0 & a_{22}\end{pmatrix} x_t + \begin{pmatrix}b_1 \\ b_2\end{pmatrix} u_t.$$

Derive explicit expressions of how a_{12}, a_{22}, b_1 and b_2 depend on a, b and h.

(b) From now on, we will consider a specific system given by $a = 1.5$ and $b = 3$, sampled with $h = 0.2$ seconds. To simplify calculations, we will set $b_1 = 0$ and polish the other entries to $a_{12} = 0.2$, $a_{22} = 0.75$ and $b_2 = 0.35$. In other words, we will consider the discrete-time system

$$\begin{aligned} x_{t+1} &= \begin{pmatrix} 1 & 0.2 \\ 0 & 0.75 \end{pmatrix} x_t + \begin{pmatrix} 0 \\ 0.35 \end{pmatrix} (u_t + w_t) \\ y_2 &= \begin{pmatrix} 1 & 0 \end{pmatrix} x_t \end{aligned}$$

Your task will now be to design a linear control law on the form

$$u_t = -Lx_t + L^{\mathrm{ref}} r_t.$$

such that the closed-loop poles are located at $z = 0.5$ and $z = 0.6$, while y_t should be equal to r_t in stationarity when there is no disturbance ($w_t = 0$)

(c) Determine the stationary error due to a constant disturbance $w_t = w_0$. Will the stationary error for $w_0 = 1$ be below 0.01?

(d) How can you modify your controller to reduce (or even eliminate) the stationary error? Suggest solutions for both when you can measure w and when you cannot.

2. Stability and invariance of nonlinear systems

In the previous chapter, we discussed the stability of discrete-time linear systems. We noted that the state trajectory of an autonomous linear system may essentially exhibit three qualitatively different behaviors: it may converge to zero, it may diverge, or it may stay bounded and neither converge nor diverge. We also demonstrated that the stability properties of a linear system are completely characterized by the eigenvalue structure of its system matrix.

Assessing stability of nonlinear and constrained systems is much more involved. There is no simple and conclusive algebraic test for asymptotic stability, and the local stability properties may be different from the global ones. In this chapter, we will review some basic aspects of Lyapunov theory, a flexible and powerful framework based on energy considerations that provides useful insight into the behaviour of general dynamical systems. When applied to linear systems, Lyapunov theory results in an alternative necessary and sufficient stability condition that will be essential in our analysis and design of linear-quadratic and model-predictive controllers.

We will also explore the concept of invariant sets. An invariant set of a dynamical system is a subset of the state space with the property that if the initial state belongs to this set, it is guaranteed to remain within the set for all future times. This concept is particularly useful for analyzing constrained systems, as it helps to identify boundaries that the state cannot cross and provides a means to guarantee that the system always operates within safe and acceptable limits.

2.1 Stability concepts

Consider a nonlinear system

$$x_{t+1} = f(x_t) \tag{2.1}$$

where $f : \mathbb{R}^n \mapsto \mathbb{R}^n$. We say that $x^{\mathrm{eq}} \in \mathbb{R}^n$ is an *equilibrium point* of the system if $x^{\mathrm{eq}} = f(x^{\mathrm{eq}})$. The following stability concepts characterize the system behaviour around an equilibrium point.

Definition 2.1.1 — Stability concepts. An equilibrium x^{eq} of (2.1) is

- *stable* if for every $\varepsilon \in (0, \varepsilon_{\max}]$, there exists $\delta > 0$ (possibly dependent on ε) such that

$$\|x_0 - x^{\text{eq}}\| \leq \delta \Rightarrow \|x_t - x^{\text{eq}}\| \leq \varepsilon \text{ for all } t \geq 0.$$

- *unstable*, if it is not stable
- *asymptotically stable*, if it is stable and δ can be chosen such that

$$\|x_0 - x^{\text{eq}}\| < \delta \Rightarrow \lim_{t\to\infty} x_t = x^{\text{eq}}$$

Note that a given system may have multiple equilibrium points and that the stability concepts describe the local behaviour around such an equilibrium. The stability definition says that no matter how close ε to the equilibrium point we want the system state to stay, there should be a $\delta > 0$ such that this bound is satisfied for all trajectories whose initial values lie in a δ-ball around the equilibrium. Otherwise, the system is said to be unstable. Asymptotic stability implies that trajectories that start sufficiently close to the equilibrium point also converge to it. The set of initial values x_0 for which $\lim_{t\to\infty} x_t = x^{\text{eq}}$ is known as the *region of attraction* of x^{eq}; see Figure 2.1.

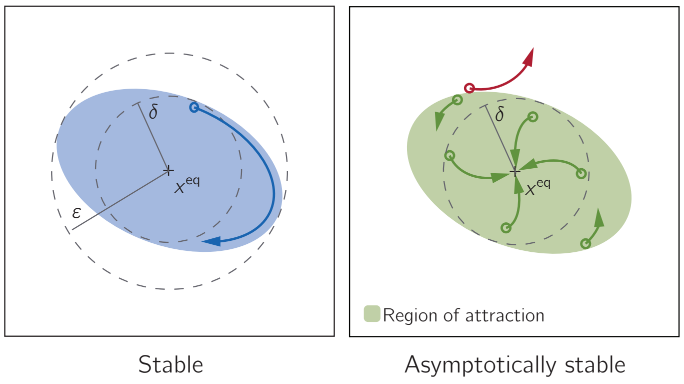


Figure 2.1: The left picture illustrates the concept of stability. For every ε, we can find a δ such that $\|x_0 - x^{\text{eq}}\| \leq \delta$ guarantees that $\|x_t - x^{\text{eq}}\| \leq \varepsilon$ for all t. The right picture illustrates the concepts of local asymptotic stability: there is a δ such that all trajectories with initial value that satisfies $\|x_0 - x^{\text{eq}}\| \leq \delta$ converge to the equilibrium point. The set of initial values that lead to trajectories that converge to the equilibrium is called the region of attraction (here marked in light green).

The next example illustrates the stability concepts on a simple system.

■ **Example 2.1** Introduce the unit saturation function

$$\text{sat}(u) = \begin{cases} -1 & \text{if } u < -1 \\ u & \text{if } -1 \leq u \leq 1 \\ +1 & \text{if } u > 1 \end{cases}$$

and consider the following unstable system under saturated feedback

$$x_{t+1} = 2x_t - \text{sat}(1.5x_t)$$

This system has three equilibria: $x^{\text{eq}} = 0$ and $x^{\text{eq}} = \pm 1$. Its dynamics can be analyzed by considering the three regimes where the feedback is linear, in negative saturation, and in positive saturation, respectively. When $|x_0| \leq 2/3$, the control will not saturate and the state will evolve as

$$x_{t+1} = 0.5x_t.$$

Hence, the origin is locally asymptotically stable. If $x_0 \geq 2/3$, the system evolves as

$$x_{t+1} = 2x_t - 1.$$

Trajectories of this system decay and move into the area of linear operation if $x_0 \in [2/3, 1)$, stay constant if $x_0 = 1$ and diverge if $x_0 > 1$. By a similar argument for $x_0 \leq -2/3$, closed-loop trajectories converge to the origin if $x_0 \geq -1$, stay constant if $x_0 = -1$ and diverge if $x_0 < -1$.

These observations imply that the origin is a locally asymptotically stable equilibrium, attracting trajectories with initial values in the region of attraction $\{x_0 \mid |x_0| < 1\}$. The other two equilibria are unstable: no matter how you choose $\varepsilon > 0$, trajectories starting from any $x_0 = 1 + \delta$ with $0 < \delta \leq \varepsilon$ will diverge and therefore grow larger than $1 + \varepsilon$ (a similar argument holds for $x_0 = -1 - \delta$). ■

2.2 Lyapunov stability

The basic idea of Lyapunov stability is to introduce a positive energy measure $V_t = V(x_t)$ that decreases along system trajectories. Specifically, if $V_{t+1} \leq V_t$ for all t, then $V_t \leq V_0$ and if V_t decreases at a sufficient rate (to be made precise), then we can guarantee that $V_t \to 0$. By selecting the Lyapunov function carefully, we will be able to relate the magnitude of V_t to the size of x_t so that $V_t \leq V_0$ implies that x_t is bounded, and $\lim_{t\to\infty} V_t = 0$ necessitates that $\lim_{t\to\infty} x_t = 0$. In this way, we will be able to use Lyapunov functions to prove both stability and asymptotic stability. The next theorem is a first result in this direction. It guarantees that if energy measures with certain properties are decreasing along all system trajectories, then the system state will be bounded.

Theorem 2.2.1 If there exists a continuous function $V(x)$ whose sublevel sets

$$\mathcal{L}_V(\alpha) = \{x \mid V(x) \leq \alpha\}$$

are bounded for every value of α and

$$V(f(x)) \leq V(x)$$

for all x, then every trajectory of (2.1) is bounded.

Proof. We re-write $V(x_t)$ as the sum of $V(x_0)$ and changes in V along the system trajectory

$$\begin{aligned} V(x_t) &= V(x_0) - V(x_0) + V(x_1) - V(x_1) + \cdots + V(x_{t-1}) - V(x_{t-1}) + V(x_t) = \\ &= V(x_0) + \sum_{k=0}^{t-1} V(f(x_k)) - V(x_k) \leq V(x_0) \end{aligned}$$

since $V(f(x)) - V(x) \leq 0$ for all x. Hence, $V(x_t) \leq V(x_0)$ and every trajectory lies in the set

$$\mathcal{L}_V(V(x_0)) = \{x \mid V(x) \leq V(x_0)\}$$

which is bounded by assumption. ■

The proof demonstrates that if V is decreasing along system trajectories, then once the state enters a level set of V it never leaves this set. The assumption of bounded level sets ensures that the state remains bounded, and hence that the system is stable; see Figure 2.2.

By imposing a few additional conditions on V, we will be able to obtain conditions that guarantee asymptotic stability. To this end, we introduce the following definitions.

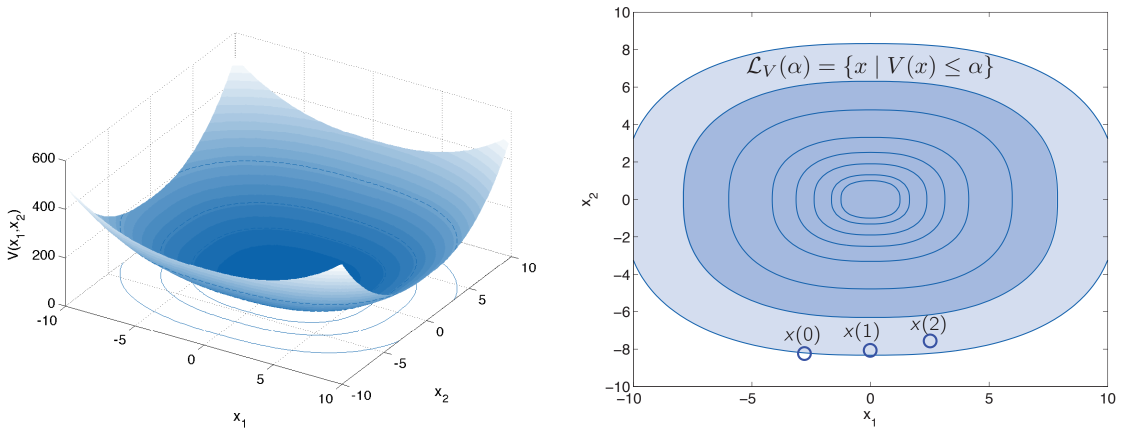


Figure 2.2: The left figure shows a Lyapunov function and some of its level sets; the right figure illustrates the property that once the state vector enters a level set, it will not leave. If the level set is bounded, this implies that the state will also be bounded.

Definition 2.2.1 A continuous function $V : \mathbb{R}^n \mapsto \mathbb{R}$ is *positive semidefinite* if
(a) $V(x) \geq 0 \quad \forall x.$
V is *positive definite* if it satisfies the conditions
(a) $V(x) \geq 0 \quad \forall x,$
(b) $V(0) = 0$ if and only if $x = 0$, and
(c) $V(x) \to \infty$ as $\|x\| \to \infty$.

These definitions are based on the corresponding notions for quadratic forms reviewed in Appendix A. Specifically, if P is a positive semidefinite matrix, then $V(x) = x^T Px$ is a positive semidefinite function, and if P is positive definite, then so is V. An important property of positive definite functions is that their level sets are bounded and closed (i.e. compact).

We can now state a first Lyapunov theorem for asymptotic stability.

Theorem 2.2.2 If there exists a continuous function $V : \mathbb{R}^n \mapsto \mathbb{R}$ such that
(a) $V(x)$ is positive definite, and
(b) $V(f(x)) - V(x) \leq -l(x)$ for some positive semidefinite function $l(x)$,
then along all trajectories $\{x_t\}$ of (2.1), it holds that $l(x_t) \to 0$ as $t \to 0$. If, in addition, $l(x)$ is positive definite, then $x_t \to 0$ as $t \to \infty$.

Proof. Condition (b) and the system dynamics (2.1) imply that $V(x_{t+1}) - V(x_t) \leq -l(x_t)$ for all x_t. Summing these inequalities across time yields

$$\sum_{t=0}^{T} V(x_{t+1}) - V(x_t) \leq -\sum_{t=0}^{T} l(x_t).$$

Hence

$$\lim_{T\to\infty} \sum_{t=0}^{T} l(x_t) \leq V(x_0) - \lim_{T\to\infty} V(x_T)$$

Since $V(x_t) \geq 0$ by condition (a), and $\{V(x_t)\}$ is a decreasing sequence by condition (b), the right-hand side will converge to a finite limit. By Cauchy's convergence criterion (see Appendix A), convergence of the infinite sum implies that $l(x_t) \to 0$ as $t \to \infty$. Finally, if $l(x)$ is positive definite, then $l(x_t) \to 0$ implies that $x_t \to 0$ and asymptotic stability follows. The proof is complete. ■

With a few additional assumptions on V and l, it is also possible to derive bounds on how quickly the state converges.

Corollary 2.2.3 If there is a continuous function $V : \mathbb{R}^n \mapsto \mathbb{R}$ such that
(a) $\alpha_1 \|x\|_2^2 \le V(x) \le \alpha_2 \|x\|_2^2$ for all x, and
(b) $V(f(x)) - V(x) \le -\beta \|x\|_2^2$,
for some positive scalars α_1, α_2 and β, then $\|x_t\|_2^2 \le \frac{\alpha_2}{\alpha_1}\left(1 - \frac{\beta}{\alpha_1}\right)^t \|x_0\|_2^2$.

Proof. With $x = x_t$, we can use the lower bound in (a) to re-write condition (b) as

$$V(x_{t+1}) \le V(x_t) - \beta \|x_t\|_2^2 \le V(x_t) - \frac{\beta}{\alpha_1} V(x_t) = \left(1 - \frac{\beta}{\alpha_1}\right) V(x_t)$$

By repeated application of this inequality, it holds that

$$V(x_t) \le \left(1 - \frac{\beta}{\alpha_1}\right)^t V(x_0)$$

Finally, we use (a) to convert this to the desired inequality for $\|x_t\|_2^2$. ■

Lyapunov stability of linear systems

We now specialize the results to linear systems. Remarkably, the asymptotic stability of a linear system is *equivalent* to the existence of a quadratic Lyapunov function. Such Lyapunov functions play a central role in many of the analysis and control design procedures introduced in later chapters.

Theorem 2.2.4 The autonomous linear system

$$x_{t+1} = Ax_t \tag{2.2}$$

is asymptotically stable if and only if, for any positive definite matrix Q, the *Lyapunov equation*

$$A^\top PA - P + Q = 0 \tag{2.3}$$

admits a positive definite solution P. In addition, for any given $Q \succ 0$, the solution P is unique.

Proof. Let Q be an arbitrary positive definite matrix and assume that (2.3) admits a positive definite solution P . The Lyapunov function candidate $V(x) = x^\top Px$ then satisfies

$$\begin{aligned} V(x_{t+1}) - V(x_t) &= x_{t+1}^\top Px_{t+1} - x_t^\top Px_t = \\ &= x_t^\top \left(A^\top PA - P\right) x_t = -x_t^\top Qx_t \end{aligned}$$

Since $l(x) = x^\top Qx$ is a positive definite function, Theorem 2.2.2 guarantees asymptotic stability.

If, on the other hand, the system (2.2) is asymptotically stable, then $|\lambda_i(A)| < 1$ for all i and

$$P = \sum_{k=0}^{\infty} \left(A^\top\right)^k QA^k$$

exists and satisfies the Lyapunov equation (2.3).

Finally we demonstrate that P is unique. To this end, let (2.2) be asymptotically stable and assume that both P and P' satisfy the Lyapunov equation. Then

$$A^\top(P - P')A - (P - P') = 0$$

Repeated application of this relationship yields

$$P - P' = A^\top (P - P')A = \cdots = \lim_{k\to\infty} (A^\top)^k (P - P')A^k = 0$$

where the last equality follows from stability of A. Thus, $P' = P$, i.e. P is unique. □.

If one is only concerned about asymptotic stability, then Theorem 2.2.4 is very simple to use: just pick any positive definite matrix Q and solve the Lyapunov equation. The system (2.2) is asymptotically stable if and only if the solution P is positive definite. Conversely, if we have a candidate Lyapunov function defined by some positive definite matrix P, we can validate it by computing the associated Q matrix and verify that it is positive definite.

Since the Lyapunov equation is linear in the elements of the matrix P and since P is symmetric, the Lyapunov equation yields a system of $n(n+1)/2$ linear equations. Although this means that one could solve the Lyapunov equation using a standard linear equation solver, there are more efficient numerical routines for solving Lyapunov equations.

Note that the theorem does *not* state that asymptotically stable linear systems admit unique Lyapunov functions. The solution P to the Lyapunov equation is unique *for a given* Q. Changing Q results in a different solution P and hence a different Lyapunov function. As the next result shows, different choices for Q allow for different bounds on how quickly the state converges to zero.

Corollary 2.2.5 Consider the linear system $x_{t+1} = Ax_t$ and assume that the Lyapunov equation (2.3) admits a positive definite solution P for a given positive definite Q. Then

$$\|x_t\|_2^2 \leq \frac{\lambda_{\max}(P)}{\lambda_{\min}(P)} \left(1 - \frac{\lambda_{\max}(Q)}{\lambda_{\min}(P)}\right)^t \|x_0\|_2^2$$

Proof. A solution to the Lyapunov equation implies that $V(x) = x^\top Px$ satisfies $\lambda_{\min}(P)\|x\|_2^2 \leq V(x) \leq \lambda_{\max}(P)\|x\|_2^2$; and (b) $V(Ax) - V(x) \leq -\lambda_{\max}(Q)\|x\|_2^2$, where $\lambda_{\min}(P), \lambda_{\max}(P)$ and $\lambda_{\max}(Q)$ are all positive. Hence, the result follows directly from Corollary 2.2.3. ■

A limitation of Theorem 2.2.4 is that it requires Q to be positive definite. This is not an issue when assessing asymptotic stability of a given system, since we can then choose Q freely. However, it can be restrictive if Q is fixed and we search for a solution to the associated Lyapunov equation. The next result shows that we can allow Q to be positive semidefinite if a certain detectability condition is met. This condition ensures that $x_t^\top Qx_t$ can only remain zero if x_t tends to zero.

Theorem 2.2.6 Let (A,C) be detectable. Then

$$x_{t+1} = Ax_t$$

is asymptotically stable if and only if the Lyapunov equation

$$A^\top PA - P + C^\top C = 0 \tag{2.4}$$

admits a unique positive semidefinite solution. If (A,C) is observable, then P is positive definite.

Proof. As in the proof of the basic Lyapunov theorem, if the system is asymptotically stable, then

$$P = \sum_{k=0}^{\infty} (A^\top)^k C^\top CA^k = \lim_{l\to\infty} \mathcal{O}_l^\top \mathcal{O}_l$$

exists and satisfies the Lyapunov equation. Uniqueness follows by the same arguments as in Theorem 2.2.4. If the system is observable, the observability matrix has rank equal to the system order n, which implies that P is positive definite.

Conversely, assume that $P \succeq 0$ satisfies the Lyapunov equation (2.4). Let λ be en eigenvalue of A with associated eigenvector v. Multiplying (2.4) with v^* from the left and v from the right yields

$$(|\lambda|^2 - 1)v^* P v = -\|Cv\|_2^2 \tag{2.5}$$

Since (A, C) is detectable, Theorem 1.3.4 implies that any v that satisfies $Cv = 0$ and $Av = \lambda v$ must have $|\lambda| < 1$. Note that any such v must also have $v^* P v = 0$. If $Cv \neq 0$, on the other hand, then the right-hand side of (2.5) is strictly negative. As $v^* P v \geq 0$, we must then have $v^* P v > 0$ and can conclude that also in this case, it holds that $|\lambda| < 1$. Since λ was chosen arbitrarily, all eigenvalues of A are inside the unit circle and the system is asymptotically stable. The proof is complete. ■

We will make use of this theorem to prove stability of optimal control laws later in these notes. Next, we will demonstrate how it can be used to evaluate quadratic performance indices.

Evaluating quadratic costs via Lyapunov functions

One way to assess the quality of a feedback control system

$$x_{t+1} = Ax_t + Bu_t, \qquad u_t = -Lx_t, \qquad y_t = Cx_t$$

is through performance indices such as the output energy,

$$J_{\text{oe}} = \sum_{t=0}^{\infty} y_t^\top y_t = \sum_{t=0}^{\infty} x_t^\top (C^\top C) x_t$$

and the input energy

$$J_{\text{ie}} = \sum_{t=0}^{\infty} u_t^\top u_t = \sum_{t=0}^{\infty} x_t^\top (L^\top L) x_t.$$

Both of these are quadratic functions of the system state trajectory $\{x_t\}$. As the next result shows, if the underlying system is asymptotically stable, then these costs are finite and quadratic functions of the initial state that can be computed by solving a Lyapunov equation.

Proposition 2.2.7 Consider the linear system $x_{t+1} = Ax_t$ and the quadratic performance index $J = \sum_{t=0}^{\infty} x_t^\top Q x_t$ with Q positive semidefinite. If $(A, Q^{1/2})$ is detectable and the Lyapunov equation

$$A^\top P A - P + Q = 0$$

admits a positive semidefinite solution P, then $J = x_0^\top P x_0$.

Proof. Pre- and post-multiply the Lyapunov equation by x_t and sum over time, we get

$$\sum_{t=0}^{T} x_t^\top (A^\top P A - P + Q) x_t = \sum_{t=0}^{T} (x_{t+1}^\top P x_{t+1} - x_t^\top P x_t + x_t^\top Q x_t) = 0$$

By re-arranging the terms, this means that

$$\sum_{t=0}^{T} x_t^\top Q x_t = \sum_{t=0}^{T} \left(x_{t+1}^\top P x_{t+1} - x_t^\top P x_t \right) = x_0^\top P x_0 - x_T^\top P x_T$$

If the solution P to the Lyapunov equation is positive semidefinite then, by Theorem 2.2.6, the system is asymptotically stable and the final term in this expression will vanish as $T \to \infty$. Hence.

$$\lim_{T \to \infty} \sum_{t=0}^{T} x_t^\top Q x_t = x_0^\top P x_0 \tag{2.6}$$

which is the desired result. ■

The next example demonstrates how the result can be used to assess control energy cost of the stabilizing state feedback for the quadcopter dynamics that we have derived earlier.

■ **Example 2.2** Let us consider the quadcopter dynamics $x_{t+1} = Ax_t + Bu_t$ with

$$A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} 0.5 \\ 1 \end{pmatrix}$$

We already know that this dynamics is not asymptotically stable, so it should come as no surprise that the Lyapunov equation with $Q = I$ does not admit any solution. In fact, it is easy to verify that the $(1,1)$ element of the matrix $P - A^\top PA$ is always zero, so there can be no solution with $Q = C^\top C$ where (A,C) is observable either; cf. Example 1.4.

Let us now consider the closed-loop dynamics under the state feedback

$$u_t = -Lx_t = -\begin{pmatrix} 1/4 & 7/8 \end{pmatrix} x_t$$

computed in Example 1.6. We know that the associated closed-loop dynamics, $x_{t+1} = (A - BL)x_t := A_{cl}x_t$ is asymptotically stable, so the Lyapunov function should have a solution. Since (A_{cl}, L) is observable, we can use $Q = L^\top L$ and solve $A_{cl}^\top PA_{cl} - P + Q = 0$ to find

$$P = \frac{1}{54}\begin{pmatrix} 4 & 12 \\ 12 & 45 \end{pmatrix}$$

Finally, note that $x^\top Qx = x^\top L^\top Lx = u^\top u$, so by (2.6) the total energy used by the controller to control the system from an initial state x_0 is

$$\lim_{T\to\infty} \sum_{t=0}^{\infty} u_t^\top u = \lim_{T\to\infty} \sum_{t=0}^{T} x_t^\top L^\top Lx_t = x_0^\top Px_0$$

where P is the matrix that we have just computed. ■

2.3 Positively invariant sets

In addition to asymptotic stability, we are often interested in guaranteeing that the state vector does not violate critical constraints. Given a set $\mathcal{A}$ of admissible (allowed) states, we would like to make sure that the state vector never leaves $\mathcal{A}$. Such guarantees can be obtained using positively invariant sets. Before introducing formal definitions, we illustrate the key idea with an example.

■ **Example 2.3** Consider the autonomous linear system

$$x_{t+1} = \begin{pmatrix} 0.8 & 0.4 \\ 0.2 & 0.4 \end{pmatrix} x_t,$$

with the admissible set $\mathcal{A} = \{x \,|\, \|x\|_\infty \leq 1\}$, shown as the white square in Figure 2.3 (left). A few simulations reveal that even if the initial state is admissible, future states may leave $\mathcal{A}$. To ensure that the state remains admissible over time, we will reason using positive invariant sets $\mathcal{I}$, i.e. sets with the property that if x_0 lies in $\mathcal{I}$, then x_t stays in this set for all $t \geq 0$. If we can find an invariant set $\mathcal{I}$ that is fully contained in $\mathcal{A}$, we can guarantee that if x_0 lies in $\mathcal{I}$, then x_t will remain in $\mathcal{I}$ for $t \geq 0$ and therefore in $\mathcal{A}$; see Figure 2.3 (right). ■

Positively invariant sets for autonomous systems

We begin by studying invariant sets for autonomous systems.

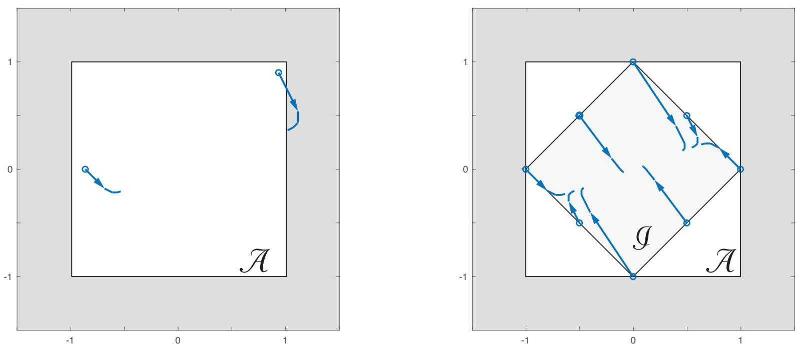


Figure 2.3: Even if the initial state is admissible, future states may not be (left). If the initial state belongs to a positive invariant set $\mathcal{I} \subseteq \mathcal{A}$, then it is also guaranteed to be admissible in the future.

Definition 2.3.1 The set $\mathcal{I} \subseteq \mathbb{R}^n$ is *positively invariant* for $x_{t+1} = f(x_t)$ if

$$x_t \in \mathcal{I} \Rightarrow x_k \in \mathcal{I} \text{ for all } k \geq t$$

The definition simply states that if the state vector belongs to a positively invariant set at time t, then it will remain in this set for all future times. Clearly, all equilibrium points of a system are positively invariant, and so is the full space $\mathbb{R}^n$. Most systems also have non-trivial invariant sets. For example, we have shown that level sets of Lyapunov functions are invariant.

Let $\mathcal{A} \subseteq \mathbb{R}^n$ be the set of admissible states, i.e. the states that satisfy all constraints. As we have argued above, the system state is guaranteed to remain admissible for all future times if we can verify that x_0 belongs to a positively invariant set $\mathcal{I}$ that is contained in $\mathcal{A}$. Of particular value is the largest positively invariant set contained in $\mathcal{A}$.

Definition 2.3.2 $\mathcal{I}^\infty(\mathcal{A})$, the *the maximal positively invariant set contained in* $\mathcal{A}$ for $x_{t+1} = f(x_t)$, is the set of all x_0 such that $x_t \in \mathcal{A}$ for all $t \geq 0$.

Our ability to compute invariant sets depends on the system dynamics and the properties of the admissible set $\mathcal{A}$. We will focus on linear systems and polyhedral constraints. Specifically, assume that $x_{t+1} = Ax_t$ and that the set of admissible states is $\mathcal{A} = \{x \mid Mx \leq \mathbf{1}\}$ (as discussed in Appendix B, every polyhedron that contains the origin in its interior can be represented on this form). The maximal invariant set in $\mathcal{A}$ is the set of initial states from which the system stays in $\mathcal{A}$:

$$\begin{aligned}\mathcal{I}^\infty(\mathcal{A}) &= \{x_0 \mid (x_0 \in \mathcal{A}) \wedge (x_1 \in \mathcal{A}) \wedge (x_2 \in \mathcal{A}) \wedge \dots\} = \\ &= \{x_0 \mid (x_0 \in \mathcal{A}) \wedge (Ax_0 \in \mathcal{A}) \wedge (A^2x_0 \in \mathcal{A}) \wedge \dots\} = \\ &= \{x_0 \mid (Mx_0 \leq 1) \wedge (MAx_0 \leq 1) \wedge (MA^2x_0 \leq 1) \wedge \dots\}\end{aligned}$$

Since the set is defined by the intersection of halfspaces, it is convex. As the next result shows, it is often defined by a finite number of halfspaces, and therefore a polyhedron.

Theorem 2.3.1 The maximal invariant set of $x_{t+1} = Ax_t$ contained in $\mathcal{A} = \{x \mid Mx \leq 1\}$ is

$$\mathcal{I}^\infty(\mathcal{A}) = \mathcal{I}^\nu(\mathcal{A}) := \left\{x \mid MA^kx \leq 1, \quad k = 0, 1, \dots, \nu\right\} \tag{2.7}$$

where the determinedness index ν is the smallest positive integer such that $\mathcal{I}^{\nu+1}(\mathcal{A}) = \mathcal{I}^\nu(\mathcal{A})$. If A is Schur stable and (A, M) is observable, then ν is finite.

The full proof is given in the appendix. Note that $\mathcal{I}^{\nu+1}(\mathcal{A}) = \mathcal{I}^\nu(\mathcal{A})$ means that the two sets should be the same. A simple technique for verifying if two set of linear inequalities define the same polyhedron is described in Appendix B.

We can develop some additional intuition into invariant sets by enforcing the constraints over a finite number of steps into the future. To this end, we make the following definition:

Definition 2.3.3 The predecessor set of $\mathcal{S} \subseteq \mathbb{R}^n$ for the dynamics $x_{t+1} = f(x_t)$ is the set

$$\text{pre}(\mathcal{S}) = \{x \mid f(x) \in \mathcal{S}\}$$

In words, $\text{pre}(\mathcal{S})$ is the set of states that will evolve into $\mathcal{S}$ in one step. For a linear system $x_{t+1} = Ax_t$ and $\mathcal{A} = \{x \mid Mx \leq \mathbf{1}\}$, we have $\text{pre}(\mathcal{A}) = \{x \mid MAx \leq \mathbf{1}\}$. With this definition, we can now construct the set of states that are guaranteed to stay in $\mathcal{A}$ for at least k time steps, denoted $\mathcal{I}^k(\mathcal{A})$. Clearly, $\mathcal{I}^0(\mathcal{A}) = \mathcal{A}$ and

$$\mathcal{I}^k(\mathcal{A}) = \mathcal{A} \cap \text{pre}(\mathcal{I}^{k-1}(\mathcal{A})) \tag{2.8}$$

i.e. the states that satisfy the constraints over k steps are the states that satisfy the constraint at the first step and evolve into states that satisfy the constraints for the remaining $k-1$ time steps. Applying this formula recursively for $k = 1, 2, \dots$ generates a sequence of sets $\mathcal{A} \supseteq \mathcal{I}^1(\mathcal{A}) \supseteq \cdots \supseteq \mathcal{I}^k(\mathcal{A})$. If we detect that $\mathcal{I}^{\nu+1}(\mathcal{A}) = \mathcal{I}^{\nu}(\mathcal{A})$, then the iteration has converged and we have found $\mathcal{I}^{\infty}(\mathcal{A})$.

The next example demonstrates the iterative procedure for finding a maximal invariant set.

■ **Example 2.4** Let us return to the set-up that we used to motivate invariant sets in Example 2.3. Figure 2.4 illustrates the recursive construction (2.8) of the maximal invariant set. In the construction of $\mathcal{I}^3(\mathcal{A})$, all new inequalities are redundant to those in $\mathcal{I}^2(\mathcal{A})$ and we conclude that the two-step invariant set is equal to the maximal invariant set for this system and this set of admissible states. ■

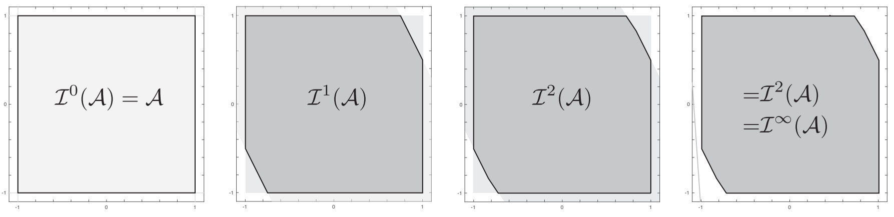

$\mathcal{I}^0(\mathcal{A}) = \mathcal{A}$
$\mathcal{I}^1(\mathcal{A})$
$\mathcal{I}^2(\mathcal{A})$
$=\mathcal{I}^2(\mathcal{A})$
$=\mathcal{I}^\infty(\mathcal{A})$

Figure 2.4: The set of admissible states $\mathcal{A} = \mathcal{I}_0(\mathcal{A})$ to the left, followed by $\mathcal{I}^1(\mathcal{A}) = \mathcal{A} \cap \text{pre}(\mathcal{I}^0(\mathcal{A}))$, and $\mathcal{I}^2(\mathcal{A})$. Since $\mathcal{I}^3(\mathcal{A}) = \mathcal{I}^2(\mathcal{A})$, the two-step invariant set is maximal.

Although Theorem 2.3.1 establishes that asymptotically stable linear systems have invariant sets represented by a finite number of linear inequalities, the complexity of the representation depends on both the dynamics and the constraints. The next example illustrates this fact.

■ **Example 2.5** Let us consider the linear system

$$\dot{x}(t) = \begin{pmatrix} 0 & 1 \\ -\omega_0^2 & -2\zeta\omega_0 \end{pmatrix} x(t)$$

with $\omega_0 = 1$ and sampling time $h = 0.25$. The dynamics of the system is increasingly oscillatory as the relative damping ζ decreases. We define the set of admissible states as $\mathcal{A} = \{x \mid \|x\|_\infty \leq 10\}$. If $\zeta = 1$, the dynamics is well-damped and $\mathcal{I}^\infty(\mathcal{A}) = \mathcal{I}^3(\mathcal{A})$; see Figure 2.5 (left). When $\zeta = 0.1$, the dynamics becomes more oscillatory, and the determinedness index increases to 5, Figure 2.5 (middle). If we would continue to decrease ζ, the index would increase even further.

To illustrate the impact of the constraints on $\mathcal{I}^\infty(\mathcal{A})$, we keep $\zeta = 0.1$ but shift the admissible states to $\mathcal{A} = \{x \mid -5 \leq [x]_i \leq 15\}$. This situation is similar to shifting the equilibrium point of the system to $(-5, -5)$. In this case, the detminedness index jumps to 16, see Figure 2.5 (right). ■

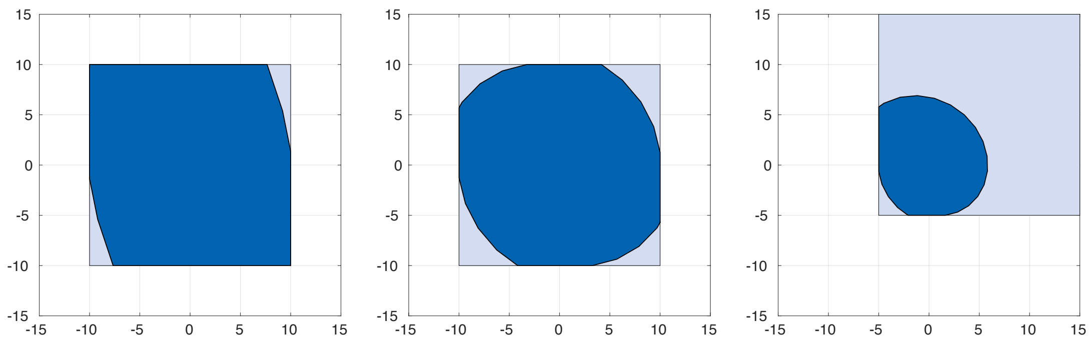


Figure 2.5: The determinedness index depends on both the system dynamics and the constraints. The constraint set $\mathcal{A}$ is in light blue and $\mathcal{I}^\infty(\mathcal{A})$ in dark blue. Decreasing the damping of the system increases the determinedness index (middle), while shifting the constraint set may alter both size, shape and determinedness index of the invariant set (right).

While finding an invariant set requires relatively sophisticated computations, it is easy to verify that a given polytope is invariant. The following result provides one useful technique.

Proposition 2.3.2 A convex polyhedron $\mathcal{P}$ with vertices $\{v_1, v_2, \dots, v_m\}$ is invariant under the linear dynamics $x_{t+1} = Ax_t$ if and only if $Av_i \in \mathcal{P}$ for all $i = 1, \dots, m$.

The proof is given in appendix. A similar invariance test for polyhedra represented as the intersection of half-spaces is explored in the exercises.

Control invariant sets

Positively invariant sets are defined for autonomous systems, *i.e.* systems without any external inputs. In control systems, we have the extra ability to adjust the system input so that the state vector remains admissible. It will then be more natural to reason using control invariant sets.

Definition 2.3.4 The set $\mathcal{C} \subseteq \mathbb{R}^n$ is *positively control invariant* for the dynamics $x_{t+1} = f(x_t, u_t)$ and the control constraint $u_t \in \mathcal{U}$ if

$$x_t \in \mathcal{C} \Rightarrow \exists \{u_t, u_{t+1}, \dots\} \text{ with } u_k \in \mathcal{U} \text{ such that } x_k \in \mathcal{C} \; \forall k \geq t$$

This definition states that if x_t belongs to a control invariant set, then there exists a control sequence satisfying the control constraints that makes the state stay in the control invariant set for all future times. Since not every control invariant set will guarantee constraint satisfaction, we will look for the maximal control invariant set that is contained in the set of admissible states $\mathcal{A}$:

Definition 2.3.5 $\mathcal{C}^\infty(\mathcal{A}, \mathcal{U})$, the *maximal positively control invariant set* contained in $\mathcal{A}$ for the dynamics $x_{t+1} = f(x_t, u_t)$ and the control constraint $u_t \in \mathcal{U}$ is the set of initial states x_0 for which there is an admissible control sequence $\{u_t\}$ that ensures that $x_t \in \mathcal{A}$ for all $t \geq 0$.

The set $\mathcal{C}^\infty(\mathcal{A}, \mathcal{U})$ is also known as the *maximal output admissible set*. By a slight redefinition of predecessor sets, we can structure the controlled invariant set computations as we did for the invariant sets. Specifically, we use the following definition

Definition 2.3.6 The predecessor set of $\mathcal{S} \subseteq \mathbb{R}^n$ for the dynamics $x_{t+1} = f(x_t, u_t)$ and control constraints $u_t \in \mathcal{U}$ is

$$\text{pre}(\mathcal{S}, \mathcal{U}) = \{x \mid \exists u \in \mathcal{U} \text{ such that } f(x, u) \in \mathcal{S}\}$$

The predecessor set is also known as the *one-step controllable set*, since it is the set of states for which there is an admissible control that drives the next state into $\mathcal{S}$. If the dynamics is linear and

Algorithm 1 Algorithm for computing maximal control invariant set contained in $\mathcal{S}$.

```
1: $\mathcal{C}^{(0)} \leftarrow \mathcal{A}$
2: for $k = 0, 1, \ldots$ do
3:     $\mathcal{C}^{(k+1)} \leftarrow \mathcal{A} \cap \text{pre}(\mathcal{C}^{(k)}, \mathcal{U})$
4:     if $\mathcal{C}^{(k+1)} == \mathcal{C}^{(k)}$ then
5:         return $\mathcal{C}^{(k)}$
```

the constraints are polyhedral, then the predecessor is itself a polyhedron that can be computed. Specifically, with $\mathcal{A} = \{x \mid M_x x \leq \mathbf{1}\}$, $\mathcal{U} = \{u \mid M_u u \leq \mathbf{1}\}$ and $x_{t+1} = Ax_t + Bu_t$, the predecessor set of $\mathcal{A}$ is the set of x for which we find a solution to the inequalities

$$\begin{pmatrix} M_x A & M_x B \\ 0 & M_u \end{pmatrix} \begin{pmatrix} x \\ u \end{pmatrix} \leq \begin{pmatrix} \mathbf{1} \\ \mathbf{1} \end{pmatrix}.$$

If we view the solution set of these inequalities as a polyhedron, the predecessor set can be seen as a projection of this polyhedron onto its first n coordinates. This projection can be computed using a procedure called Fourier-Motzkin elimination, which eliminates u from the inequalities and returns the hyperplanes in $\mathbb{R}^n$ that define the projected set (see Appendix B for details).

We can compute $\mathcal{C}^\infty(\mathcal{A}, \mathcal{U})$ recursively, analogously to how we computed $\mathcal{I}^\infty(\mathcal{A})$. To this end, let $\mathcal{C}^0(\mathcal{A}, \mathcal{U}) = \mathcal{A}$ and proceed with the iteration

$$\mathcal{C}^k(\mathcal{A}, \mathcal{U}) = \mathcal{A} \cap \text{pre}(\mathcal{C}^{k-1}(\mathcal{A}, \mathcal{U}), \mathcal{U}) \tag{2.9}$$

The recursion generates a sequence of sets which converges to $\mathcal{C}^\infty(\mathcal{A}, \mathcal{U}) = \lim_{k\to\infty} \mathcal{C}^k(\mathcal{A}, \mathcal{U})$. The maximal control invariant set is finitely generated if $\mathcal{C}^{\nu+1}(\mathcal{A}, \mathcal{U}) = \mathcal{C}^\nu(\mathcal{A}, \mathcal{U})$ for some finite $\nu \geq 0$. See Algorithm 1 for a simple pseudo-code listing.

■ **Example 2.6** Let us begin with the simple scalar system

$$x_{t+1} = 2.5x_t + u_t, \qquad \mathcal{A} = \{x \mid |x| \leq 1\}, \qquad \mathcal{U} = \{u \mid |u| \leq 1\}$$

Of course, it is easy to determine the maximal control invariant set analytically. If $x_t \geq 0$, then $u_t = -1$ makes $x_{t+1} = 2.5x_t - 1 \leq x_t$ as long as $x_t \leq 2/3$. A similar argument for negative states reveals that the maximal control invariant set is $\{x \mid |x| \leq 2/3\}$. Figure 2.6 demonstrates a single step of the recursive procedure (2.9) for the same task. ■

The next example illustrates the control invariant set for a slightly more complex system. In this case, it is no longer easy to determine the maximal control invariant set by hand.

■ **Example 2.7** Let us consider the discrete-time system

$$x_{t+1} = \begin{pmatrix} 1.005 & 0.1 \\ 0.1 & 1.005 \end{pmatrix} x_t + \begin{pmatrix} 0.005 \\ 0.1 \end{pmatrix} u_t.$$

This system describes the dynamics of an "inverted pendulum" after zero-order hold sampling. Note that the system matrix has one eigenvalue strictly outside the unit disc, and therefore is unstable.

We set $\mathcal{A} = \{x \mid \|x\|_\infty \leq 0.5\}$ and $\mathcal{U} = \{u \mid |u| \leq 1\}$. The maximal control invariant set computations converge after $\nu = 10$ iterations, returning $\mathcal{C}^\infty(\mathcal{A}, \mathcal{U})$ shown in Figure 2.7 (left).

As an alternative to the maximal control invariant set, we can consider the invariant set resulting from a *specific* admissible control input. For example, we can consider a state feedback law $u_t = -Lx_t$ and compute the positively invariant set for the closed-loop dynamics

$$x_{t+1} = Ax_t + B(-Lx_t) = (A - BL)x_t. \tag{2.10}$$

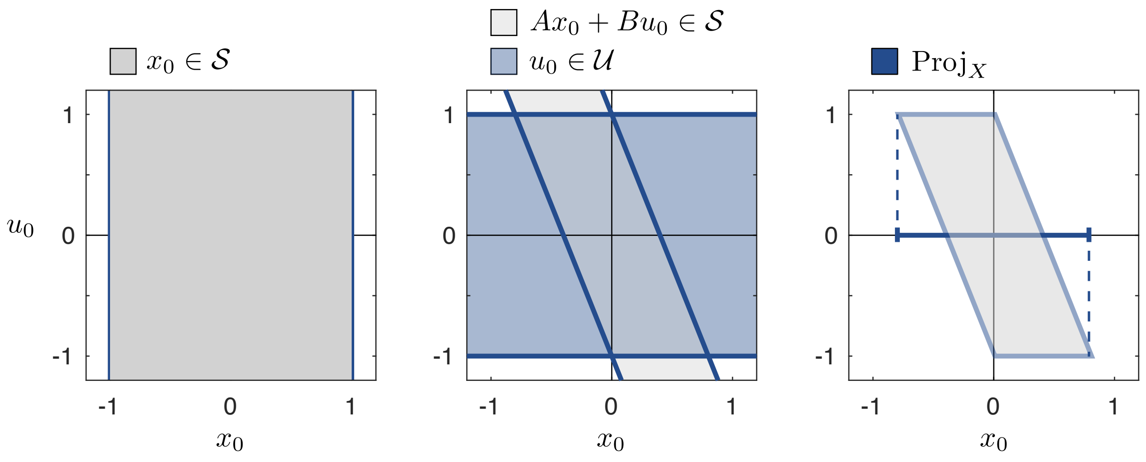


Figure 2.6: The set of admissible states $\mathcal{A}$ (left); the set of (x_0, u_0) that ensure that $x_1 \in \mathcal{A}$ and $u_0 \in \mathcal{U}$ (middle); the resulting polyhedron and its projection used in the preset computation (right). Note that this is just the first step of the recursive procedure.

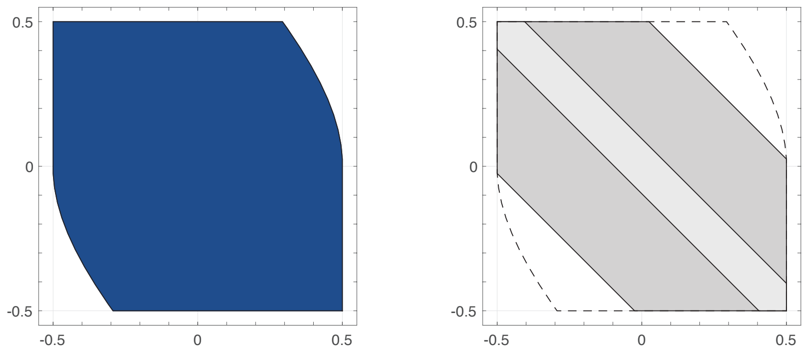


Figure 2.7: The left subfigure shows the maximally control invariant set for the linear system (2.10). The right subfigure shows the control invariant set guaranteed by two different state feedback laws (in gray). Clearly, none of the two sets are maximal.

To make sure that the state feedback is admissible for all states in the computed set, we define

$$\mathcal{A}' = \mathcal{A} \cap \{x \mid -Lx \in \mathcal{U}\}$$

and compute $\mathcal{I}^{\infty}(\mathcal{A}')$. This set will be control invariant for (2.10), since there is an admissible control law (specifically, $u_t = -Lx_t$) that keeps the state in the set for all future times. However, sets defined in this way are typically not maximal. For example, control invariant sets for two different state feedback laws, both significantly smaller than the maximal control invariant set, are shown in Figure 2.7 (right). ∎

As for invariant sets, it can sometimes be useful to have a simple technique for verifying that a given set is control invariant. The next proposition is one such result.

Proposition 2.3.3 A convex polyhedron $\mathcal{P}$ with vertices $\{v_1, v_2, \ldots, v_m\}$ is control invariant under the linear dynamics $x_{t+1} = Ax_t + Bu_t$ and the control constraints $u_t \in \mathcal{U}$ if and only if there is an $u_i \in \mathcal{U}$ such that $(Av_i + Bu_i) \in \mathcal{P}$ for all $i = 1, \ldots, m$.

It is sometimes of interest to be able to compute the set of initial states that can reach a set $\mathcal{T} \subseteq \mathcal{A}$ of target states in at most T steps. Such T*-step controllable sets* can be computed by a slight variation of Algorithm 1 where $\mathcal{C}^{(0)}$ is initialized to $\mathcal{T}$. In this way, $\mathcal{C}^{(k)}$ in the algorithm represents the set of admissible states that can be steered to $\mathcal{T}$ in k steps.

Local stability analysis using Lyapunov functions and invariance arguments

The stability theorems that we have stated earlier rely on the Lyapunov function to be positive definite and decreasing for all $x \in \mathbb{R}^n$. These theorems will not hold for systems that are only locally asymptotically stable. Nevertheless, one can still use Lyapunov arguments to establish local asymptotic stability and compute guaranteed regions of attraction.

Proposition 2.3.4 Consider the nonlinear system $x_{t+1} = f(x_t)$ with $f(0) = 0$. Let $X \subseteq \mathbb{R}^n$ be positively invariant for $x_{t+1} = f(x_t)$. If

(a) there exists a continuous function $V : \mathbb{R}^n \mapsto \mathbb{R}$ with $V(0) = 0$ and $V(x) \geq \alpha \|x\|_2^2$ for all $x \in X$,
(b) there exists a continuous function $l : \mathbb{R}^n \mapsto \mathbb{R}$ with $l(0) = 0$ and $l(x) \geq \beta \|x\|_2^2$ for all $x \in X$,
(c) it holds that

$$V(f(x)) - V(x) \leq -l(x) \qquad \text{for all } x \in X$$

Then, X is a region of attraction for $x = 0$, i.e. if $x_0 \in X$ it holds that $x_t \to 0$ as $t \to \infty$.

There are two common ways to apply this proposition. The first is to actually determine an invariant set X (possibly a maximal invariant set) and then verify the conditions (a)-(c) on this set. The other one is to verify the conditions (a)-(c) on some set X' without first proving that it is invariant. Then, since we know that level sets of Lyapunov functions are invariant, we can take X to be the largest level of V contained in X'.

2.4 Exercises

Problem 2.1 Consider the nonlinear system

$$x_{t+1} = \frac{x_t}{1+x_t^2}$$

(a) Verify that $x = 0$ is the system's unique equilibrium point.
(b) Use the Lyapunov function $V(x) = x^2$ to prove global asymptotic stability of the origin.

Problem 2.2 Consider the nonlinear system

$$x_{t+1} = g(x_t)$$

where g has the property that $g(0) = 0$ and

$$|g(x)| < |x|, \qquad \forall x \neq 0.$$

Show that the origin is globally asymptotically stable.

Problem 2.3 Consider the nonlinear system

$$x_{t+1} = \frac{az_t}{1+x_t^2}$$
$$z_{t+1} = \frac{bx_t}{1+z_t^2}$$

with state vector $(x_t, z_t) \in \mathbb{R}^2$.
(a) Prove that if $|a| < 1$ and $|b| < 1$, then the origin is globally asymptotically stable.
(b) What can you say about the case $|a| \leq 1$, $|b| \leq 1$ but $|a| + |b| \neq 2$?
(c) What can you say about the case $|a| = |b| = 1$?
Hint. Consider the Lyapunov function $V(x,y) = x^2 + y^2$.

Problem 2.4 Consider the nonlinear system

$$x_{t+1} = \frac{2x_t}{1+x_t^2}$$

(a) Verify that $x = 0$ is an *unstable* equilibrium point for the system.
(b) Use the Lyapunov function $V(x) = x^2$ to prove that all trajectories remain bounded.

Problem 2.5 In this problem, we will consider the nonlinear system

$$x_{t+1} = f(x_t) := \frac{1}{2}x_t + x_t^2.$$

and use the Lyapunov function $V(x) = x^2$ to prove local asymptotic stability.
(a) Determine the equilibrium points of the system. Justify why the system cannot be globally asymptotically stable.
(b) Show that the interval $\mathcal{X} = (-1, 1/2)$ is invariant under the given dynamics. Use $V(x)$ to prove that the origin is locally asymptotically stable and that $\mathcal{X}$ is a region of attraction.
(c) Determine the set $\mathcal{V} = \{x \mid V(x) > 0 \wedge V(f(x)) - V(x) < 0\}$. What is the largest level set of V contained in $\mathcal{V}$? How does this set compare to the set $\mathcal{X}$ considered in (b)?

Problem 2.6 Consider the linear system $x_{t+1} = Ax_t$ with

$$A = \begin{pmatrix} 1/4 & 1 \\ 1/2 & 0 \end{pmatrix}$$

Solve the Lyapunov inequality $A^T PA - P + Q = 0$ analytically for $Q = I$. What can you conclude about the stability of the linear system?

Problem 2.7 Consider a linear system $x_{t+1} = Ax_t$. For each system matrix below, solve the Lyapunov equation numerically with $Q = I$ and assess stability of the systems. Compute the eigenvalues of A and validate your findings.

$$\text{(a)} \quad A = \begin{pmatrix} 0.4 & 1 \\ 0 & 0.8 \end{pmatrix} \quad \text{(b)} \quad A = \begin{pmatrix} 1.2 & 0 \\ 0 & -0.4 \end{pmatrix} \quad \text{(c)} \quad A = \begin{pmatrix} 0.8 & -0.1 \\ 2 & 0.8 \end{pmatrix}$$

Problem 2.8 Prove that the system

$$x_{t+1} = Ax_t$$

is not asymptotically stable if, for some positive definite matrix Q, the Lyapunov equation

$$A^\top PA - P + Q = 0$$

has a solution P with one or more negative eigenvalues.

Problem 2.9 Prove that if there exists a continuous function V that satisfies

$$\begin{aligned} &\alpha_1 \|x\|_2^2 \leq V(x) \leq \alpha_2 \|x\|_2^2 \\ &V(f(x)) - V(x) \leq -\beta \|x\|_2^2 + m \end{aligned}$$

for some positive scalars α_1, α_2, m, and β. Then,

$$\lim_{t \to \infty} \|x_t\|^2 \leq \frac{\alpha_2}{\alpha_1 \beta} m$$

along every trajectory of the system $x_{t+1} = f(x_t)$. This result is useful to prove stability of systems in the presence of disturbances and uncertainties that refrain the state from converging to zero.

Hint. Start by showing that the inequalities imply $V(x_{t+1}) \leq qV(x_t) + r$ for some $q \in (0,1)$ and some positive scalar r.

Problem 2.10 Consider the following linear system

$$x_{t+1} = \begin{pmatrix} 0.5 & 0 \\ 1 & -0.5 \end{pmatrix} x_t$$

For each of the following polyhedral sets $\mathcal{X}$, determine $\text{pre}(\mathcal{X})$ under the given dynamics.

(a)

$$\mathcal{X} = \{x \in \mathbb{R}^2 \mid 2x_1 + 3x_2 \leq 5\}$$

(b)

$$\mathcal{X} = \left\{x \in \mathbb{R}^2 \mid \begin{pmatrix} -1 \\ -2 \end{pmatrix} \leq x \leq \begin{pmatrix} 2 \\ 3 \end{pmatrix}\right\}$$

(c)

$$\mathcal{X} = \{x \in \mathbb{R}^2 \mid x \geq 0\}$$

Problem 2.11 Show that $\mathcal{I} \subseteq \mathcal{S}$ is positive invariant under $x_{t+1} = Ax_t$ if and only if

$$\mathcal{I} \subseteq \text{pre}(\mathcal{I})$$

Problem 2.12 For each of the sets $\mathcal{X}$ in Exercise 2.10, determine if $\mathcal{X}$ is positive invariant under

$$x_{t+1} = \begin{pmatrix} 0.5 & 0 \\ 1 & -0.5 \end{pmatrix} x_t$$

Hint: Combine the results from Exercises 2.10 and 2.11.

Problem 2.13 Consider the double integrator

$$x_{t+1} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} x_t + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u_t$$

under the state constraints

$$u_t \in \mathcal{U} = \{u \in \mathbb{R} \mid -1 \leq u \leq 1\}$$

For each of the following polyhedral sets $\mathcal{X}$, determine $\text{pre}(\mathcal{X};\mathcal{U})$ under the given dynamics.

(a)

$$\mathcal{X} = \left\{ x \in \mathbb{R}^2 \mid \begin{pmatrix} -5 \\ -5 \end{pmatrix} \leq x \leq \begin{pmatrix} 5 \\ 5 \end{pmatrix} \right\}$$

(b)

$$\mathcal{X} = \left\{ x \in \mathbb{R}^2 \mid x_1 \geq -3, x_1 + 2.5x_2 \geq -3, x_1 \leq 3, x_1 + 2.5x_2 \leq 3 \right\}$$

Problem 2.14 Show that $\mathcal{C} \subseteq \mathcal{X}$ is control invariant under $x_{t+1} = Ax_t + Bu_t$, $u_t \in \mathcal{U}$ if and only if

$$\mathcal{C} \subseteq \text{pre}(\mathcal{C};\mathcal{U})$$

Problem 2.15 For each of the sets $\mathcal{X}$ in Exercise 2.13, determine if $\mathcal{X}$ is control invariant under

$$x_{t+1} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} x_t + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u_t, \quad u_t \in \mathcal{U} = \{u \in \mathbb{R} \mid -1 \leq u \leq 1\}$$

Hint: Use the results from 2.14.

Problem 2.16 Assume that there are no input constraints ($\mathcal{U} = \mathbb{R}^m$) and show that a set $\mathcal{C} \subseteq \mathcal{X}$ is controlled invariant under $x_{t+1} = Ax_t + Bu_t$ if and only if there exists an $m \times n$ matrix L such that $\mathcal{C}$ is positive invariant under $x_{t+1} = (A - BL)x_t$

Problem 2.17 Consider the linear system

$$x_{t+1} = \frac{1}{4} \cdot \begin{pmatrix} 5 & 3 \\ 3 & 5 \end{pmatrix} x_t + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u_t$$

subject to the input and state constraints

$$x_t \in X = \left\{ x \in \mathbb{R}^2 \mid \begin{pmatrix} -10 \\ -10 \end{pmatrix} \leq x \leq \begin{pmatrix} 10 \\ 10 \end{pmatrix} \right\}$$
$$u_t \in U = \{u \mid -10 \leq u \leq 10\}$$

(a) Is X control invariant?
(b) Consider the state feedback law

$$u_t = -Lx_t = -\begin{pmatrix} 3/2 & 3/2 \end{pmatrix} x_t$$

Draw the set $\mathcal{A}$ of $x \in X$ such that the state feedback does not saturate. Is $\mathcal{A}$ control invariant?

Problem 2.18 Consider the double integrator

$$x_{t+1} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} x_t + \begin{pmatrix} 1/2 \\ 1 \end{pmatrix} u_t$$

(a) Define the state feedback

$$u_t = -\begin{pmatrix} 1/2 & 1 \end{pmatrix} x_t. \tag{2.11}$$

and the constraint set

$$X = \{x \in \mathbb{R}^2 \mid |x_1| \le 1 \wedge |x_2| \le 1\}$$

Is X positively invariant under the control law (2.11)?

(b) Is the control law in (a) admissible for all $x \in X$ under the control constraint

$$|u_t| \le 1$$

(c) Is the set X control invariant under the control constraint in (b)?

(d) Is there an admissible state-feedback control law $u_t = -Lx_t$ which makes X positively invariant under the constraint in (b)?

Problem 2.19 Consider the double integrator

$$x_{t+1} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} x_t + \begin{pmatrix} 0.5 \\ 1 \end{pmatrix} u_t$$

under the control constraints $u \in U = \{u \mid |u| \le 1\}$ and the state constraints $x \in \{x \mid |x_1| \le 25 \text{ and } |x_2| \le 5\}$. Validate that the set in Figure 2.8. is control invariant.

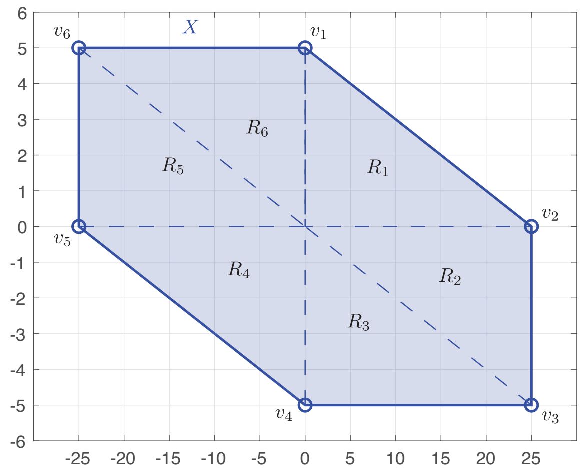


Figure 2.8: Control invariant set X with vertices v_i, subdivided into regions R_i.

Show that it is possible to derive an explicit control law that renders X control invariant under the double integrator dynamics. Specifically, show that in each region R_i in Figure 2.8, there is a linear control law $u_t = -L_i x_t$ that maps its vertices back into X.

II Part Two: Control Design

3. Finite-time optimal control

This chapter studies optimal control problems over a finite time horizon. Such problems arise in a variety of contexts, also outside of the traditional application areas of automatic control. For example, the planning financial investments and routing of packets in a data network can both be posed as optimal control problems. The theory that we present allows us to derive optimal solutions to some important control problems, but it will also be essential for justifying model predictive control strategies and for reasoning about their properties. Special attention is given to optimal control of linear systems with constraints that can be formulated as linear inequalities in the state vector and control input. We will show that such problems can be posed as convex optimization problems and solved quickly and reliably using a variety of numerical solvers. In fact, finite-horizon optimal control problems of the type that we study here are solved in every sampling instant in the model predictive control strategies that we will develop later.

3.1 A standard form for finite-time optimal control problems

In this chapter, we are given a dynamical system $x_{t+1} = f_t(x_t, u_t)$ and want to find the optimal input sequence over T steps, $\{u_0, u_1, \dots, u_{T-1}\}$. We write these problem as

$$\begin{array}{lll} \text{minimize} & \sum_{t=0}^{T-1} g_t(x_t,u_t) + g_T(x_T) & \\ \text{subject to} & x_{t+1} = f_t(x_t,u_t) & t = 0,1,\dots,T-1 \\ & (x_t,u_t) \in \mathcal{C}_t & t = 0,1,\dots,T-1 \\ & x_T \in \mathcal{X}_T & \end{array} \tag{3.1}$$

We refer to T as the *horizon* of the optimal control problem, and each time instant t as a *stage*. The system state in stage t is denoted x_t, and we can affect its evolution using the control action u_t. In particular, the state in the next stage is $x_{t+1} = f_t(x_t, u_t)$. The *transition function* f_t is indexed by t to indicate that it can vary over stages. Applying the control action u_t in state x_t at stage t incurs a *stage cost* $g_t(x_t, u_t)$. Our aim is to find a control sequence $\{u_0, u_1, \dots u_{T-1}\}$ that minimizes the accumulated cost over the horizon. Since the control action in stage t affects the state at time $t+1$, there is no u_T and the *terminal cost* g_T depends only on the terminal state x_T. The states and control actions can also be subject to constraints, defined by $\mathcal{C}_t$ and $\mathcal{X}_T$; see Figure 3.1 for an illustration.

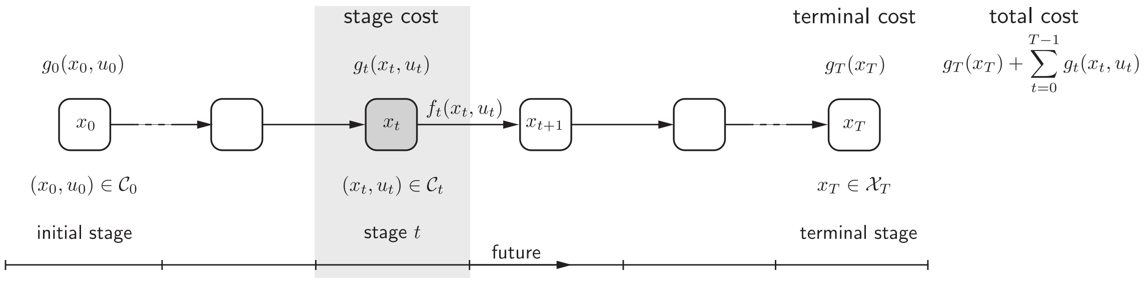


Figure 3.1: Finite-time optimal control set-up. At each stage t, the control action u_t incurs a cost $g_t(x_t,u_t)$ and affects the next state $f_t(x_t,u_t)$. The aim is to select the control sequence $\{u_0,u_1,\dots,u_{T-1}\}$ that satisfies the constraints and minimizes the total cost across the stages.

■ **Example 3.1** To illustrate the notation, consider the problem of driving the quadcopter studied in earlier examples from rest at $y_0 = 0$ to rest at $y_T = 10$ using minimal energy while respecting the control constraint $-g \le u_t \le g$ for all times. We can write this problem as

$$\begin{array}{lll}
\text{minimize} & \sum_{t=0}^{T-1} u_t^2 & \\
\text{subject to} & x_{t+1} = \begin{pmatrix}1 & 1\\ 0 & 1\end{pmatrix} x_t + \begin{pmatrix}1/2\\ 1\end{pmatrix} u_t & t=0,1,\dots,9\\
& -g \le u_t \le g & t=0,1,\dots,9\\
& x_{10} = \begin{pmatrix}10\\ 0\end{pmatrix} &
\end{array}$$

In terms of our standard notation, we use stage costs $g_t(x_t,u_t) = u_t^2$ but no terminal cost, that is, we let $g_T(x_t) \equiv 0$. The dynamics $f_t(x_t,u_t)$ is the linear system derived in Example 1.8. The constraint on the lift force is encoded by $\mathcal{C}_t = \{(x,u) \mid -g \le u \le g\}$. Finally, the drone is forced to be at rest at its target at $T = 10$ by letting $\mathcal{X}_T = \{x \mid x = x_{\text{tgt}}\}$. ■

There are several different variations of the optimal control problem (3.1). In one of them, we are given an initial state x_0 and look for the control actions $\{u_0,u_1,\dots,u_{T-1}\}$ that minimize the accumulated cost under the given dynamics and constraints. This *open-loop* optimal control is often fragile in practice, since the complete control sequence is optimized based only on the initial state and the system model. If the model is inaccurate or if the system is affected by unforeseen disturbances, the open loop sequence can be far from optimal. In another variation of (3.1), we are not given a specific initial value, but look for a *closed-loop* policy, *i.e.* a sequence of functions $\{\mu_0(x),\mu_1(x),\dots,\mu_{T-1}(x)\}$ such that $u_t = \mu_t(x_t)$ defines the optimal action at time t. Optimal feedback policies are more robust, since they can use the actual state x_t at time t, but they also more difficult to compute. In this chapter, we will explore both these options.

3.2 Open-loop optimal control via convex optimization

We can view (3.6) as a standard optimization problem in $\{x_t\}$ and $\{u_t\}$, and attempt to solve a given instance numerically using a variety of solvers. However, without additional assumptions about the cost functions, dynamics, and constraints, it can be computationally difficult to find the optimal solution. In fact, it may be difficult to even validate that a given solution candidate is globally optimal. As discussed in Appendix C, it is advantageous if a mathematical programming problem is convex, since the optimal solution can then be found quickly and reliably, both in theory and in practice. In this section, we limit ourselves to linear systems with quadratic costs and linear constraints. This allows us to formulate and solve the optimal control problem as quadratic programs (QPs). Recall that a QP is an optimization problem with a quadratic objective function and linear constraints:

$$\begin{array}{lll} \text{minimize} & \sum_{i=1}^{n_z}\sum_{j=1}^{n_z} p_{ij}z_iz_j + \sum_{j=1}^{n_z} 2q_jz_j + r & \\ \text{subject to} & \sum_{j=1}^{n_z} a_{ij}z_j \le b_i & i=1,\dots,m_z \\ & \sum_{j=1}^{n_z} g_{ij}z_j = h_i & i=1,\dots,p_z \end{array}$$

It is convenient to group the decision variables z_j into a vector $z \in \mathbb{R}^{n_z}$ and express the problem as

$$\begin{array}{ll} \text{minimize} & z^\top Pz + 2q^\top z + r \\ \text{subject to} & Az \le b \\ & Gz = h \end{array} \tag{3.2}$$

The problem is convex when P is positive semi-definite; cf. Appendix C. Although we will generally have to solve these problems numerically, we will show that some useful classes of quadratic programs even admit closed-form solutions.

Simple quadratic problems with explicit solutions

We begin by studying two important problems that admit analytical solutions. They are based on basic results for the minimization of convex quadratic functions with and without linear constraints. The first one is a vector extension of the well-known completion-of-squares, $x^2+2qx+r=(x+q)^2+r-q^2$ used to find the minimizer and the minimal value of quadratic functions.

Lemma 3.1 — Completion-of-squares. Consider the quadratic function $f:\mathbb{R}^n \mapsto \mathbb{R}$ given by

$$f(z) = z^\top Pz + 2q^\top z + r$$

where P is a positive semidefinite matrix. All minimizers $z^\star$ of f satisfy the normal equations

$$Pz^\star + q = 0.$$

If P is positive definite, then the minimizer is unique and given by

$$z^\star = -P^{-1}q$$

with corresponding minimal value

$$f^\star = r - q^\top P^{-1}q = r - (z^\star)^\top Pz^\star.$$

Moreover, $f(z)$ can then be re-written as a completion-of-squares

$$f(z) = (z-z^\star)^\top P(z-z^\star) + r - (z^\star)^\top Pz^\star.$$

The result follows immediately from the first-order optimality conditions for unconstrained convex optimization in Appendix C and the differentiation rules for quadratic forms in Appendix A. The expression for $f^\star$ follows from $f(z^\star) = (-P^{-1}q)^\top P(-P^{-1}q) + 2q^\top(-P^{-1}q) + r = r - q^\top P^{-1}q$, and the final claim from simplifying $f(z) = z^\top Pz - 2z^\top Pz^\star + r = z^\top Pz + 2q^\top z + r$.

As an example of how we can use this result, let us consider a classical least-squares problem.

■ **Example 3.2 — Least-squares state estimation.** We are interested in estimating the initial state $x_0 \in \mathbb{R}^n$ of a linear system

$$\begin{aligned} x_{t+1} &= Ax_t, \\ y_t &= Cx_t + v_t \end{aligned}$$

based on measurements $\{y_0, y_1, \dots, y_N\}$. If $v_t \equiv 0$ and (A,C) is observable, then we have already shown that we can compute x_0 exactly from n consecutive output measurements. In the presence of the disturbance sequence $\{v_t\}$, however, it is unlikely that there would be any solution to the system of linear equations

$$\begin{aligned} y_0 &= Cx_0 + v_0 \\ y_1 &= Cx_1 + v_1 = CAx_0 + v_1 \\ \vdots &= \vdots \\ y_N &= CA^N x_0 + v_N \end{aligned}$$

In absence of any other information, it is natural to try to find the “smallest” disturbance sequence that allows to explain the observations. Let us introduce

$$Y_N = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_N \end{pmatrix}, \qquad \mathbb{O}_N = \begin{pmatrix} C \\ CA \\ \vdots \\ CA^N \end{pmatrix}, \qquad V_N = \begin{pmatrix} v_0 \\ v_1 \\ \vdots \\ v_N \end{pmatrix}$$

and re-write the observation equations as

$$Y_N = \mathbb{O}_N x_0 + V_N$$

One possible measure of the size of the disturbance sequence $\{v_t\}$ is its total energy

$$\sum_{t=0}^{N} \|v_t\|^2 = \sum_{t=0}^{N} v_t^\top v_t = V_N^\top V_N.$$

The disturbance sequence of smallest energy that explains the observations can then be found by solving the optimization problem

$$\begin{array}{ll} \underset{V_N, x_0}{\text{minimize}} & V_N^\top V_N \\ \text{subject to} & V_N = Y_N - \mathbb{O}_N x_0 \end{array}$$

We can use the equality constraint to eliminate V_N, leading to the equivalent problem

$$\underset{x_0}{\text{minimize}} \quad (Y_N - \mathbb{O}_N x_0)^\top (Y_N - \mathbb{O}_N x_0) = x_0^\top (\mathbb{O}_N^\top \mathbb{O}_N) x_0 - 2Y_N^\top \mathbb{O}_N x_0 + Y_N^\top Y_N$$

If (A,C) is observable and $N \geq n$, then $\mathbb{O}_N$ has rank n and the matrix $\mathbb{O}_N^\top \mathbb{O}_N$ is positive definite. By the least-squares lemma, the optimal solution is then given by

$$x_0 = (\mathbb{O}_N^\top \mathbb{O}_N)^{-1} \mathbb{O}_N^T Y_N$$

■

The least-squares problem typically appears when we have more observations than parameters to estimate in our model. Let us now instead consider the opposite situation, *i.e.*, when we need to solve a system of $m < n$ linear equations in $z \in \mathbb{R}^n$

$$Gz = h$$

Since there may then many x that satisfy the constraints, it is natural to look for the smallest parameter vector that explains the observations. If we measure size of z in terms of its Euclidean norm $\|z\|_2^2 = z^\top z$, we would like to find the z that solves the optimization problem

$$\begin{array}{ll} \underset{z}{\text{minimize}} & z^\top z \\ \text{subject to} & Gz = h \end{array}$$

As the next result shows, also this problem admits a closed-form solution.

Proposition 3.2.1 Consider the linearly constrained quadratic program

$$\begin{array}{ll}\underset{z}{\text{minimize}} & z^\top z \\ \text{subject to} & Gz = h\end{array}$$

where $G \in \mathbb{R}^{m\times n}$, $h \in \mathbb{R}^m$ and $m < n$. If $\text{rank}(G) = m$, then the optimal solution is

$$z^\star = G^\top (GG^\top)^{-1} h.$$

Proof. See Appendix C. ■

We will use the result to study energy-optimal state transfer for linear systems.

■ **Example 3.3** Let us return to the optimal control problem for the quadcopter defined in Example 3.1, but disregard the control constraints. We are thus interested in finding the input sequence of minimum energy that ensures that $x_T = x_{\text{tgt}}$. In Chapter 1, we demonstrated that one can use the prediction equations (1.5) to express the state at time T of a linear system $x_{t+1} = Ax_t + Bu_t$ as

$$x_T = c_T + \mathcal{C}_T \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_{T-1} \end{pmatrix} = c_T + \mathcal{C}_T U_T$$

where $c_T = A^\top x_0$ and $\mathcal{C}_T$ is the controllability matrix over T steps. Since $x_0 = 0$, the free response vanishes, i.e. $c_T = 0$. The total energy of the control sequence $\{u_0, u_1, \dots, u_{T-1}\}$ is

$$\mathcal{E}(x_{\text{tgt}}, T) = \sum_{k=0}^{T-1} \|u_k\|_2^2 = \sum_{k=0}^{T-1} u_k^\top u_k = U_T^\top U_T$$

so the minimum-energy state transfer problem then reads

$$\begin{array}{ll}\text{minimize} & U_T^\top U_T \\ \text{subject to} & \mathcal{C}_T U_T = x_{\text{tgt}}.\end{array}$$

If (A,B) is reachable and $T \geq n$, the controllability matrix has full rank, and we can use Proposition 3.2.1 to determine the optimal solution

$$U_T^\star = \mathcal{C}_T^\top (\mathcal{C}_T \mathcal{C}_T^\top)^{-1} x_{\text{tgt}}.$$

We can also compute the optimal value of this problem, *i.e.*, the minimal energy cost

$$\mathcal{E}^\star(x_{\text{tgt}}, T) = (U_T^\star)^\top (U_T^\star) = x_{\text{tgt}}^\top (\mathcal{C}_T \mathcal{C}_T^\top)^{-1} x_{\text{tgt}} = x_{\text{tgt}}^\top \left(\sum_{k=0}^{T-1} A^k BB^\top (A^k)^\top \right)^{-1} x_{\text{tgt}}.$$

This expression has an interesting interpretation. The set of target states which are reachable in T steps with unit energy

$$\{x_{\text{tgt}} \mid \mathcal{E}^\star(x_{\text{tgt}}, T) \leq 1\} = \left\{ x_{\text{tgt}} \mid x_{\text{tgt}}^\top (\mathcal{C}_T \mathcal{C}_T^\top)^{-1} x_{\text{tgt}} \leq 1 \right\}$$

form ellipsoids centered at the origin, whose size increase with increasing horizon T. Figure 3.2 shows the set of states that are reachable with unit energy for various horizon lengths T. The results correspond well with physical intuition: we can displace the drone further if we do not need to drive it to rest (i.e. can allow a non-zero second state), and the unit energy budget makes it difficult to drive the system very far if the terminal velocity has to be zero. ■

In practice, one may want to consider state-transfer problems with more general objective functions and impose constraints on the states and controls. However, as constraints are added, it becomes increasingly difficult to find analytical solutions to the associated optimization problems. Instead, we typically have to resort to numerical computations.

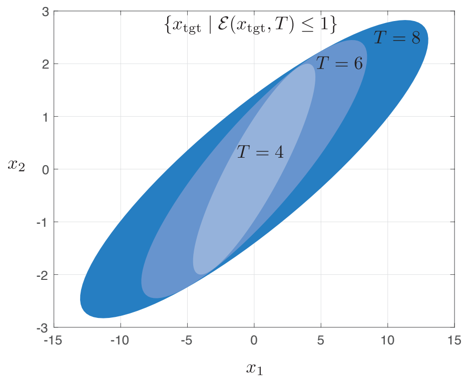


Figure 3.2: Unit-energy reachable sets for the vertical quadrotor dynamics. Recall that x_1 is the vertical position, while x_2 is the velocity.

Optimal control of linear systems with linear constraints and quadratic costs

Finite-horizon optimal control problems with more general constraints can be posed and solved as convex QPs if we restrict the costs to be convex and quadratic, and the dynamics and constraints to be linear. In particular, we assume that both the stage costs and the terminal cost are quadratic

$$q_t(x_t, u_t) = x_t^\top Q x_t + u_t^\top R u_t, \qquad q_T(x_T) = x_T^\top Q_T x_T. \tag{3.3}$$

for some positive semidefinite matrices Q, R and Q_T, while the dynamics is linear

$$f_t(x_t, u_t) = A x_t + B u_t.$$

We also assume that the constraints on u_t and x_T are defined by linear inequalities,

$$\mathcal{C}_t(x_t, u_t) = \{(x,u) \mid Mx + Nu_t \leq m\}, \qquad \mathcal{X}_T = \{x \mid M_T x \leq m_T\}.$$

Altogether, this leads to the following finite-horizon optimal control problem

$$\begin{array}{lll} \text{minimize} & \sum_{t=0}^{T-1} x_t^\top Q x_t + u_t^\top R u_t + x_T^\top Q_T x_T & \\ \text{subject to} & x_{t+1} = A x_t + B u_t & t = 0, \ldots, T-1 \\ & M x_t + N u_t \leq m & t = 0, \ldots, T-1 \\ & M_T x_T \leq m_T & \end{array} \tag{3.4}$$

We will show how these problems can be formulated as convex quadratic programs on the form (3.2), and therefore efficiently solved using standard numerical routines. Before doing so, however, we will show how some common constraints can be described in the proposed format.

■ **Example 3.4** A magnitude limitation on a scalar control signal u_t,

$$|u_t| \leq u_{\max}.$$

can be expressed by two linear inequalities $u \leq u_{\max}$ and $-u \leq u_{\max}$, *i.e.* by

$$M = 0, \qquad N = \begin{pmatrix} 1 \\ -1 \end{pmatrix}, \qquad m = \begin{pmatrix} u_{\max} \\ u_{\max} \end{pmatrix}.$$

Similarly, the constraint that a signal $y_t = Cx_t + Du_t$ should lie between upper and lower bounds

$$y_{\min} \leq Cx_t + Du_t \leq y_{\max}$$

can be written on the proposed form using

$$M = \begin{pmatrix} C \\ -C \end{pmatrix}, \qquad N = \begin{pmatrix} D \\ -D \end{pmatrix}, \qquad m = \begin{pmatrix} y_{\max} \\ -y_{\min}. \end{pmatrix}.$$

As a final example, the requirement that x_T should equal a fixed target x_{tgt} can be encoded by

$$M_T = \begin{pmatrix} I \\ -I \end{pmatrix}, \qquad N_T = 0, \qquad m_T = \begin{pmatrix} x_{\mathrm{tgt}} \\ -x_{\mathrm{tgt}} \end{pmatrix}.$$

Note that any combination of the above constraints can also be expressed on the standard form (by simply including all relevant inequalities in the definition of M, N and m). ■

We also take the opportunity to return to Example 3.1 and put it on this standard form.

■ **Example 3.5** To minimize the total input energy, we let $R = I$ while $Q = Q_T = 0$. The system is linear, with A and B matrices already given in Example 3.1. The magnitude constraints on the input are encoded as in the previous example with $u_{\max} = g$, *i.e.* by letting

$$M = 0, \qquad N = \begin{pmatrix} 1 \\ -1 \end{pmatrix}, \qquad m = \begin{pmatrix} g \\ g \end{pmatrix}.$$

and the terminal constraint can be encoded using

$$M_T = \begin{pmatrix} I \\ -I \end{pmatrix}, \qquad N_T = 0, \qquad m_T = \begin{pmatrix} x_{\mathrm{tgt}} \\ -x_{\mathrm{tgt}} \end{pmatrix}, \quad \text{where} \quad x_{\mathrm{tgt}} = \begin{pmatrix} 10 \\ 0 \end{pmatrix}$$

Finally, providing a numerical value for the horizon length T completes the specification. ■

We will now show that the finite-time optimal control problem (3.4) can be transformed to a convex QP on the standard form (3.2) with a decision vector comprised of $\{x_1, x_2, \dots, x_T\}$ and $\{u_0, u_1, \dots, u_{T-1}\}$. For convenient notation, we introduce vectors $X_T \in \mathbb{R}^{Tn}$ and $U_T \in \mathbb{R}^{Tm}$:

$$X_T = \begin{pmatrix} x_0 \\ \vdots \\ x_T \end{pmatrix}, \qquad U_T = \begin{pmatrix} u_0 \\ \vdots \\ u_{T-1} \end{pmatrix}.$$

The total cost of the optimal control problem can then be expressed as

$$X_T^\top \underbrace{\begin{pmatrix} Q & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & Q & 0 \\ 0 & \dots & 0 & Q_T \end{pmatrix}}_{\mathcal{Q}} X_T + U_T^\top \underbrace{\begin{pmatrix} R & 0 & \dots & 0 \\ 0 & R & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & R \end{pmatrix}}_{\mathcal{R}} U_T$$

Note that since Q, Q_T and R are positive semidefinite, $\mathcal{Q}$ and $\mathcal{R}$ will also be positive semidefinite. Next, the dynamics can be written as the linear equality constraints

$$\underbrace{\begin{pmatrix} I & 0 & \cdots & 0 \\ -A & I & 0 & 0 \\ 0 & \ddots & \ddots & 0 \\ 0 & 0 & -A & I \end{pmatrix}}_{\mathcal{E}} X_T + \underbrace{\begin{pmatrix} 0 & 0 & \cdots & 0 \\ 0 & -B & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -B \end{pmatrix}}_{\mathcal{F}} U_T = \underbrace{\begin{pmatrix} x_0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}}_{e}$$

while the state and control constraints can be collected into constraints on X_T and U_T:

$$\underbrace{\begin{pmatrix} M & 0 & \dots & 0 \\ 0 & \ddots & 0 & 0 \\ \vdots & \ddots & M & 0 \\ 0 & \dots & 0 & M_T \end{pmatrix}}_{\mathcal{M}} X_T + \underbrace{\begin{pmatrix} N & 0 & \dots \\ 0 & \ddots & 0 \\ \vdots & \ddots & N \\ 0 & \dots & 0 \end{pmatrix}}_{\mathcal{N}} U_T \leq \underbrace{\begin{pmatrix} m \\ \vdots \\ m \\ m_T \end{pmatrix}}_{\mathscr{m}}.$$

With this notation, the optimal control problem amounts to solving the quadratic program

$$\begin{array}{ll} \text{minimize} & X_T^\top \mathcal{Q} X_T^\top + U_T^\top \mathcal{R} U_T \\ \text{subject to} & \mathcal{M} X_T + \mathcal{N} U_T \leq \mathscr{m} \\ & \mathcal{E} X_T + \mathcal{F} U_T = e \end{array}$$

We can put this problem on the standard form for QPs (3.2) by letting

$$\begin{array}{llll} z = \begin{pmatrix} X_T \\ U_T \end{pmatrix}, & P = \begin{pmatrix} \mathcal{Q} & 0 \\ 0 & \mathcal{R} \end{pmatrix}, & q = 0, & r = 0 \\ A = \begin{pmatrix} \mathcal{M} & \mathcal{N} \end{pmatrix}, & b = \mathscr{m}, & G = \begin{pmatrix} \mathcal{E} & \mathcal{F} \end{pmatrix}, & h = e. \end{array}$$

Since both $\mathcal{Q}$ and $\mathcal{R}$ are positive semidefinite, P is too, and this is a convex QP. This formulation is known as the *extensive form* of the finite-horizon optimal control problem. It has $(T+1)n + Tm$ variables, but the matrices that describe the problem are sparse (many of their entries are zero).

It is possible to use eliminate X_T from the optimization problem and express the objective and constraints only in terms of U_T. This *condensed form* only has Tm variables, but it is described by dense matrices. To derive this formulation, note that

$$X_T = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_T \end{pmatrix} = \begin{pmatrix} Ax_0 \\ A^2x_0 \\ \vdots \\ A^Tx_0 \end{pmatrix} + \begin{pmatrix} B & 0 & \dots & 0 \\ AB & B & 0 & 0 \\ \vdots & \ddots & \ddots & 0 \\ A^{T-1}B & \dots & AB & B \end{pmatrix} U_T := \mathscr{h}_T + \mathcal{H}_T U_T. \tag{3.5}$$

We can use this relationship to eliminate X_T, first from the objective function via

$$\begin{aligned} X_T^\top \mathcal{Q} X_T + U_T^\top \mathcal{R} U_T &= (\mathcal{H}_T U_T + \mathscr{h}_T)^\top \mathcal{Q} (\mathcal{H}_T U_T + \mathscr{h}_T) + U_T^\top \mathcal{R} U_T = \\ &= U_T^\top (\mathcal{H}_T^\top \mathcal{Q}_T \mathcal{H}_T + \mathcal{R}) U_T + 2(\mathcal{H}_T^\top \mathcal{Q} \mathscr{h}_T)^\top U_T + \mathscr{h}_T^\top \mathcal{Q} \mathscr{h}_T \end{aligned}$$

and then from the state constraints using

$$\mathcal{M}(\mathcal{H}_T U_T + \mathscr{h}_T) + \mathcal{N} U_T = (\mathcal{M}\mathcal{H}_T + \mathcal{N}) U_T + \mathcal{M}\mathscr{h}_T \leq \mathscr{m}$$

The condensed formulation is thus a QP on the standard form with

$$\begin{array}{llll} z = U_T, & P = \mathcal{H}_T^\top \mathcal{Q} \mathcal{H}_T + \mathcal{R}, & q = \mathcal{H}_T^\top \mathcal{Q} \mathscr{h}_T, & r = \mathscr{h}_T^\top \mathcal{Q} \mathscr{h}_T \\ A = \mathcal{M}\mathcal{H}_T + \mathcal{N}, & b = \mathscr{m} - \mathcal{M}\mathscr{h}_T, & G = 0, & h = 0. \end{array}$$

In this formulation, it only $\mathscr{h}_T$ that depends on x_0. Hence, if we want to re-solve the same optimal control problem with a new initial state, we only need to recompute the vectors q, r and b. Since r does not affect the optimal solution (only the optimal value), it is often disregarded. In the extensive formulation, one only needs to update e when resolving the problem with a new x_0.

Extensions of the standard form

The main purpose of this section has been to show that it is possible to formulate finite time optimal control problems for constrained linear systems as convex QPs on standard form. However, to keep the notation simple, we have imposed more restrictions than strictly necessary.

Clearly, we could allow dynamics, stage costs and constraints to be time-varying, and still formulate and solve the associated optimal control problem as a QP. For example, replacing $x_{t+1} = Ax_t + Bu_t$ by $x_{t+1} = A_t x_t + B_t u_t$ would only affect the definition of the matrix $\mathcal{H}_T$, and similar observations hold for time-varying costs and constraints.

We could also couple constraints over time and allow for, *e.g.*, rate-of-change constraints

$$|u_t - u_{t-1}| \leq \delta_{\max}$$

These constraints can be dealt with by augmenting the system model with an additional state, say $z_t = u_{t-1}$ and then constrain $u_t - z_t$. But they could also be expressed more compactly as two linear inequalities in (u_t, u_{t-1}) and would therefore lead to a QP formulation where the constraints are coupled over time. Similarly, we could also allow for cost functions that couple the states and inputs over time, *e.g.* penalizing the rate-of-change $\|u_{t+1} - u_t\|_2^2$ of the control signal.

Finally, some convex costs and constraints can be expressed in the QP formalism by the introduction of auxiliary variables. One such example is the ℓ_1-norm of a vector; see Appendix C.

An important special case: finite-time linear-quadratic control

The problem (3.4) simplifies considerably if we drop the state and control constraints. The corresponding formulation

$$\begin{array}{lll} \text{minimize} & \sum_{t=0}^{T} x_t^\top Q x_t + u_t^\top R u_t + x_T^\top Q_T x_T & \\ \text{subject to} & x_{t+1} = Ax_t + Bu_t & t = 0, \ldots, T-1 \end{array}$$

is known as the finite-horizon linear-quadratic optimal control problem. In the notation introduced above, the condensed form of this problem is simply

$$\text{minimize} \quad U_T^\top \left(\mathcal{H}_T^\top \mathcal{Q} \mathcal{H}_T + \mathcal{R}\right) U_T + 2\left(\mathcal{H}_T^\top \mathcal{Q} \hbar_T\right)^\top U_T + \hbar_T^\top \mathcal{Q} \hbar_T$$

Using the least-squares lemma, this problem has the explicit solution

$$U_T^\star = -\left(\mathcal{H}_T^\top \mathcal{Q} \mathcal{H}_T + \mathcal{R}\right)^{-1} \mathcal{H}_T^\top \mathcal{Q} \hbar_T.$$

This solution is sometimes called the *batch LQR* solution. Note that it is only $\hbar_T$ that depends on x_0, and that $\hbar_T$ is a linear function of the initial state. Hence, the optimal control sequence $\{u_0^\star, u_1^\star, \ldots, u_{T-1}^\star\}$ encoded in $U_T^\star$ is also a linear function of the initial state. Moreover, since the completion-of-squares lemma yields that the optimal cost is given by

$$\hbar_T^\top \left(\mathcal{Q} - \mathcal{Q}\mathcal{H}_T(\mathcal{H}_T^\top \mathcal{Q} \mathcal{H}_T + \mathcal{R})^{-1} \mathcal{H}_T^\top \mathcal{Q}\right) \hbar_T$$

we can also conclude that the optimal cost is a quadratic function of the initial state.

Solving optimal control problems using algebraic modeling languages

The transformation of (3.4) into a convex QP on standard form is instructive, in the sense that it reveals the size and structure of the matrices that describe the QP. However, one can argue that performing such a transformation manually is tedious and error-prone. There are several domain-specific languages for optimization, e.g. [19, 25] that perform this transformation automatically. Next, we will use the modeling language cvxpy [19] to solve a real-world optimal control problem.

Example: time-optimal movement of liquid containers

In many industrial packaging processes, one needs to move a package filled with liquid from one position to another (for example, from the filling station to the sealing station). To maintain a high production rate, it is essential to carry out this movement as quickly as possible. However, the acceleration of a package induces a motion of the liquid known as slosh. If left uncontrolled, this sloshing could result in liquid spill, causing product loss and contamination of machinery and packages, as well as a possible inability to seal the package properly.

We will consider finite-time optimal control of the containers in the packaging machine illustrated in Figure 3.3. The models and parameters are taken from [20], which contains many more details about modeling and control of liquid slosh. Our focus will be on moving the open containers from the filling station to the sealing station in minimal time without creating any liquid spill.

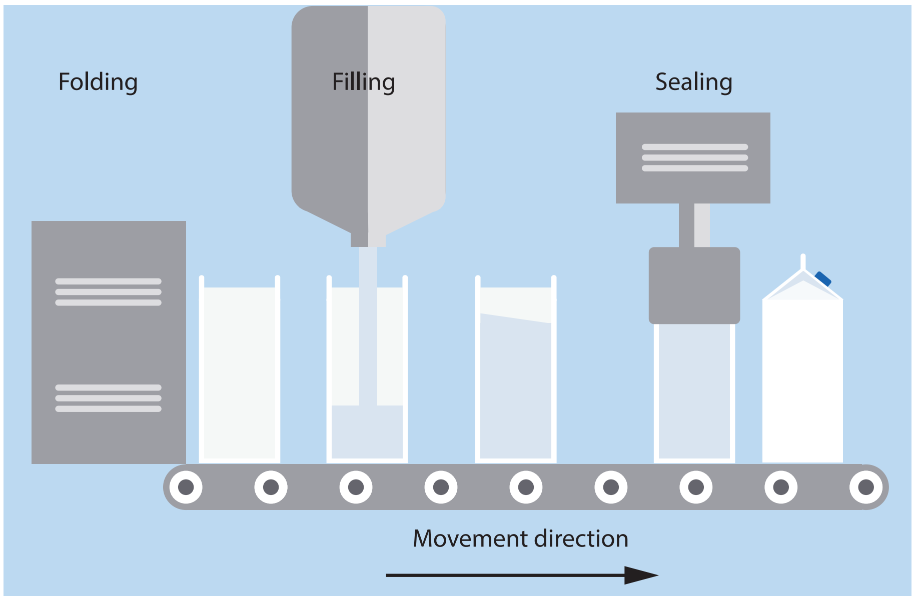


Figure 3.3: Schematic illustration of packaging machine. Packages are folded, filled with liquid and transported to a sealing machine. Our focus in this example is on the movement of the open filled container from the filling station to the sealing station.

We will use the following simple linear ODE model that describes the liquid surface evaluation at the rear container wall, and the horizontal position of the cart.

$$\dot{x}(t) = \begin{pmatrix} -2\zeta\omega_0 & -\omega_0 & 0 & 0 \\ \omega_0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} x(t) + \begin{pmatrix} \frac{a\omega_0}{2g} \\ 0 \\ 1 \\ 0 \end{pmatrix} u(t)$$

$$\begin{pmatrix} s(t) \\ p(t) \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} x(t) := \begin{pmatrix} C_s \\ C_p \end{pmatrix} x(t)$$

The two outputs $s(t)$ and $p(t)$ represent the surface evaluation at the rear of the package, and the container position on the conveyor belt, respectively.

We will consider a liquid container of height 0.2m, transported a distance of $L = 0.2$m. The liquid and container parameters are set to $\zeta = 0$, $\omega_0 = 21$ and $a = 0.07$; the parameter $g = 9.81$ is the standard constant of gravity. We are interested in moving the liquid container quickly from rest at $p = 0$ to rest at the target position $p = L$ without causing any slosh. Specifically, we require that

$$|s(t)| \leq s_{\max} \qquad \forall t$$

where $s_{\max} = 3.5$cm. In addition, the control input is limited

$$|u(t)| \leq u_{\max} \qquad \forall t.$$

The initial machine configuration has $u_{\max} = g$. We will use a sampling time of 5ms and zero-order-hold control inputs. This means that we can construct a corresponding discrete-time model using the formulas derived in Chapter 1. Although we are interested in minimizing the transfer time, we will start with the simpler problem with a fixed horizon T and try to minimize the total control energy. After zero-order hold sampling of the continuous-time dynamics, we can pose the corresponding finite-time optimal control problem

$$\begin{array}{ll} \text{minimize} & \sum_{t=0}^{T-1} u_t^2 \\ \text{subject to} & x_{t+1} = Ax_t + Bu_t, \quad t = 0, \dots, T-1 \\ & |C_s x_t| \leq s_{\max}, \quad t = 0, \dots, T-1 \\ & |u_t| \leq u_{\max}, \quad t = 0, \dots, T-1 \\ & x_T = x_{\text{tgt}} \end{array}$$

Thus, it fits our standard form with $Q = 0$, $R = I$ and $Q_T = 0$, while

$$M = \begin{pmatrix} C_s \\ -C_s \\ 0 \\ 0 \end{pmatrix}, \quad N = \begin{pmatrix} 0 \\ 0 \\ 1 \\ -1 \end{pmatrix}, \quad m = \begin{pmatrix} s_{\max} \\ -s_{\max} \\ u_{\max} \\ -u_{\max} \end{pmatrix}, \quad M_T = \begin{pmatrix} I \\ -I \end{pmatrix}, \quad m_T = \begin{pmatrix} x_{\text{tgt}} \\ -x_{\text{tgt}} \end{pmatrix}$$

The problem is readily transformed into a QP on standard form. Alternatively, we can use a modeling language such as cvx [19], that automatically converts the problem into a standard form for QPs, solves the problem, and returns a solution; see the listing in Figure 3.2.

The nominal transfer time for the package from its initial position to its target is $T = 0.5$ seconds, which leads to a planning problem over 100 sampling instances. The resulting quadratic problem is feasible and the optimal solution is shown in Figure 3.5. To speed up the movement, we proceed to find the smallest horizon T for which we can solve the state transfer problem. We do so by a simple bisection search. starting from $T_{\min} = 0$ and $T_{\max} = 0.5/h$. In each iteration, we try to solve the finite-time optimal control problem for $T_{\text{mid}} = \lfloor (T_{\min} + T_{\max})/2 \rfloor$. If the associated optimization problem is feasible, we let $T_{\max} = T_{\text{mid}}$, otherwise we set $T_{\min} = T_{\text{mid}}$. We then update T_{mid} and repeat, until we notice that $T_{\text{mid}} = T_{\min}$. The bisection search reveals that the minimum transfer time is $T = 0.39$s, a reduction of over 20% from the nominal solution. The cart position, liquid elevation, and the associated control input are shown in Figure 3.6.

In the time optimal state transfer, both the control and the slosh constraints are active during large periods of time. In fact, the optimal control has a intuitive interpretation: one first accelerates the cart until the positive slosh constraint becomes active and then adjust the input to keep the slosh at that level. This gives the maximum admissible acceleration of the cart. After an appropriate time, one then prepares the stopping by driving the system to the state where the negative slosh constraint is active and keeping it there (which results in the maximum admissible deceleration). Finally, all states are driven to rest at target position; see Figure 3.8. In fact, this is exactly how time-optimal control solution for the continuous-time system works. A formal proof of this claim requires continuous-time optimal control theory, which is beyond the scope of this course. However, we can recover the time-optimal control solution by using a smaller sampling interval, see Figure 3.7.

It is natural to ask if we could perform the transfer faster with a stronger motor (larger $u_{\max}$), or if we could use a weaker (and perhaps cheaper) motor without significant performance degradation. To this end, we solve the minimum-time optimal control problem for a range of values on $u_{\max}$. These experiments show that no dramatic performance improvements are obtained for $u_{\max}$ above g, see Figure 3.9. The reason for this is that the slosh constraint limits the transfer time. The control signal that is required to keep $s(t) = s_{\max}$ and $\dot{s}(t) = 0$ is exactly $u(t) = g$.

```
import numpy as np; import matplotlib.pyplot as plt
import control as ctrl; import cvxpy as cp

# Define continuous-time dynamics
zeta=0; omega0=21; a=0.07; g=9.81; L=0.2;
Ac=np.array([[-2*zeta*omega0, -omega0, 0, 0],
             [omega0, 0, 0, 0], [0, 0, 0, 0],[0, 0, 1, 0]])
Bc=np.array([[a*omega0/(2*g),0,1,0]]).T;
Cc=np.array([[0,1,0,0],[0,0,0,1]]); Dc=np.array([[0,0]]).T
sysc=ctrl.ss(Ac,Bc,Cc,Dc)

# Sample system to get discrete-time dynamics
h=5e-3
sysd=ctrl.c2d(sysc,h);
(A,B,C,D)=ctrl.ssdata(sysd)

# Define problem parameters and set-up optimization problem
Tf=0.5; T = int(Tf/h); time = np.linspace(0,T)
smax=0.035; umax=g
x0=np.array([[0,0,0,0]]).T; xT=np.array([[0,0,0,L]]).T
x=cp.Variable( (4, T+1) ); u=cp.Variable( (1, T) )
cost = cp.sum_squares(u);
constr = [ x[:,[0]]== x0 ]
for t in range(T):
        constr += [ x[:, t+1]==A@x[:,t] + B@u[:, t] ]
        constr += [-umax <= u[:,t], u[:,t] <= umax]
        constr += [-smax <= C[0,:]@x[:,t], C[0,:]@x[:,t] <= smax]
constr += [ x[:,[T]]==xT]
problem=cp.Problem(cp.Minimize(cost), constr)

# Solve problem and plot results
problem.solve()
fig, axs = plt.subplots(2)
axs[0].plot(C[1,:]@x.value)
axs[1].plot(u.value[0,:])
```

Figure 3.4: cvxpy code for finite-time energy optimal state transfer with magnitude constraints the slosh state and the control.

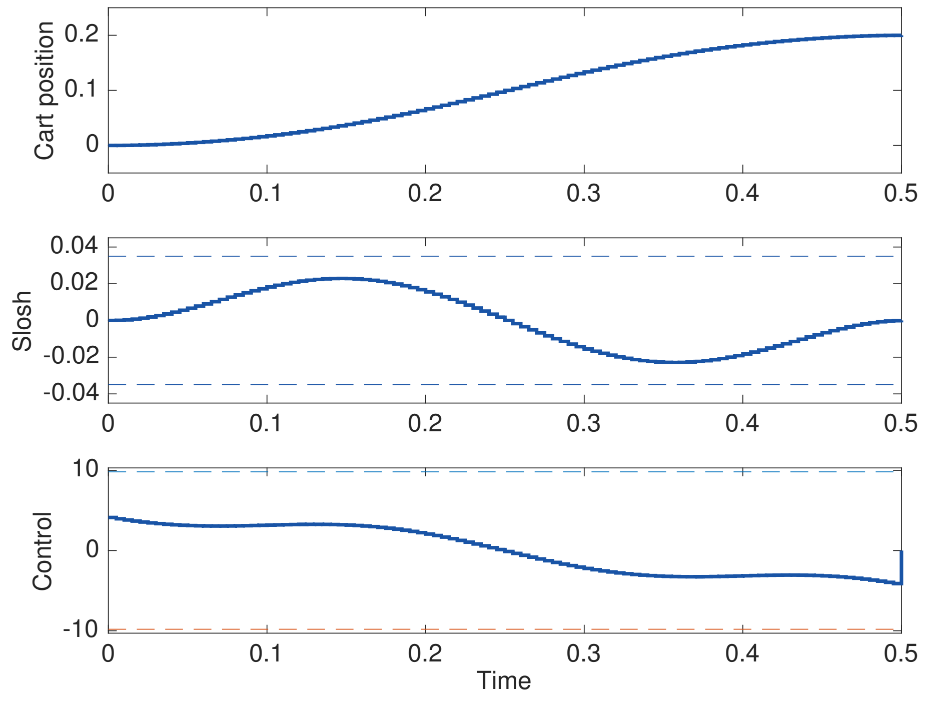


Figure 3.5: Minimum energy control of the cart with a nominal transfer time of 0.5 seconds. Neither control nor slosh constraints are active, indicating a potential for a more aggressive control.

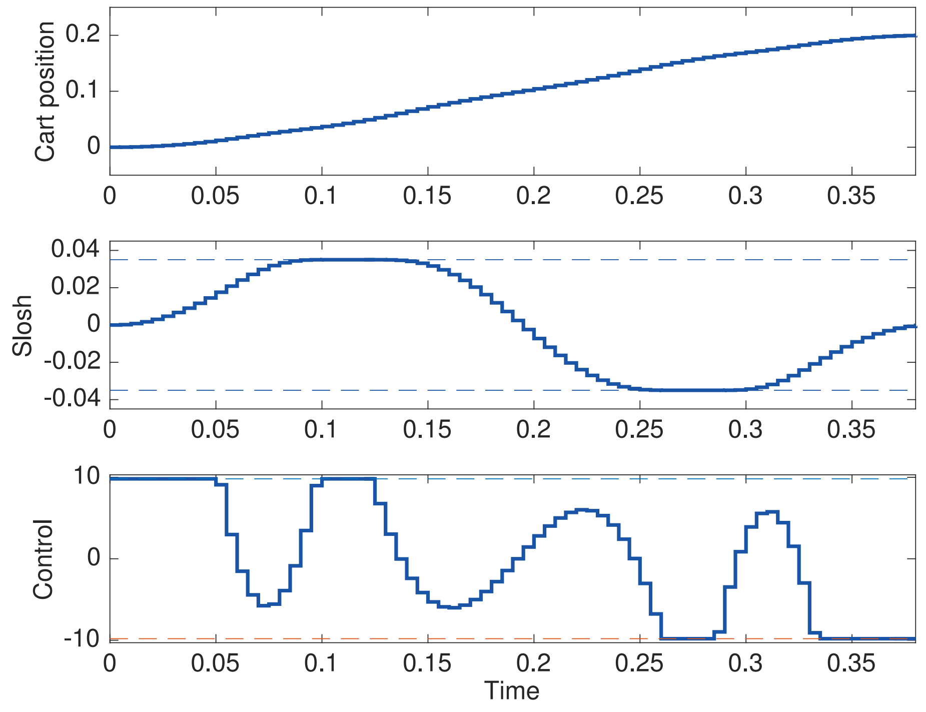


Figure 3.6: In the minimum time solution, both control and slosh constraints are active during large periods of time.

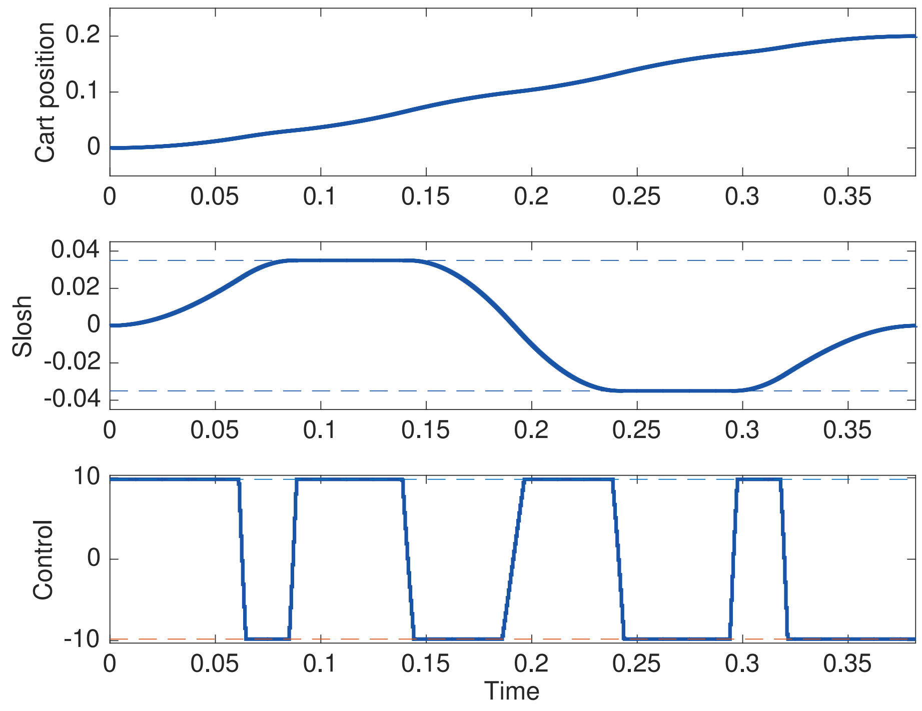


Figure 3.7: With a shorter sampling interval, the optimal input approaches the “bang-bang” nature of time-optimal control of continuous-time systems.

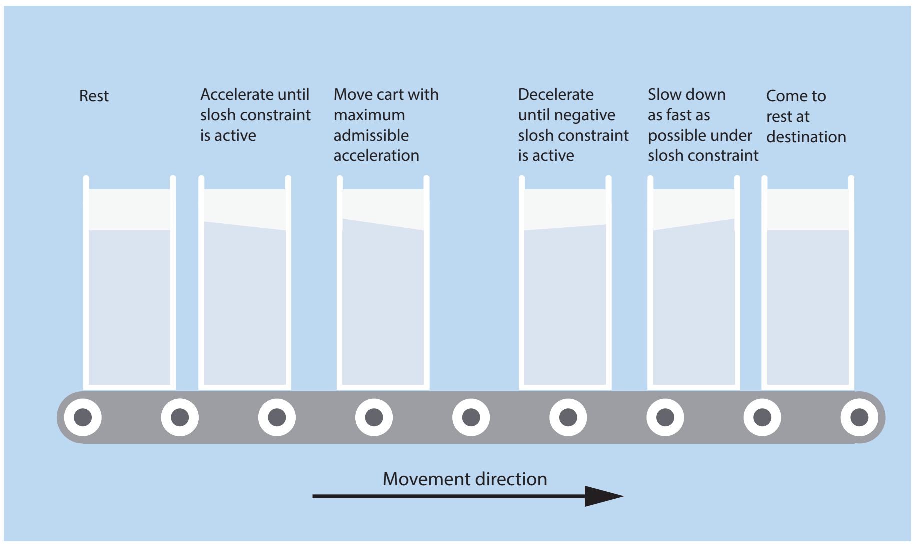

Rest
Accelerate until slosh constraint is active
Move cart with maximum admissible acceleration
Decelerate until negative slosh constraint is active
Slow down as fast as possible under slosh constraint
Come to rest at destination
Movement direction

Figure 3.8: Time-optimal movement strategy in pictures.

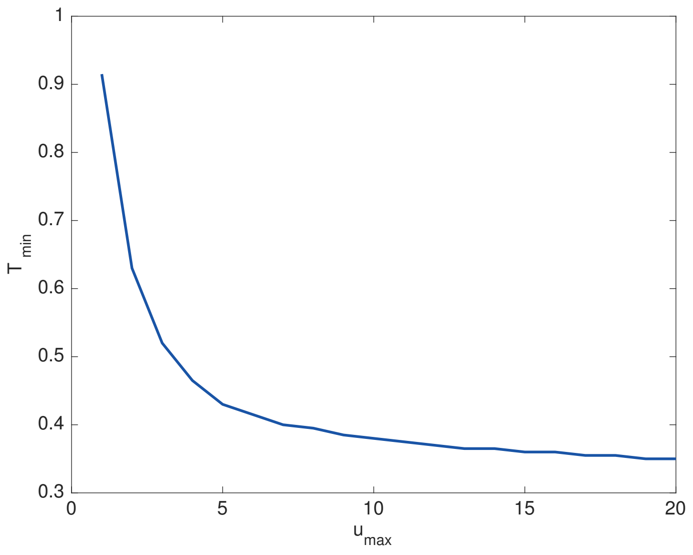


Figure 3.9: Minimum transfer time as function of the optimal control input $u_{\max}$. No significant reductions in the transfer time are possible when $u_{\max}$ exceeds g.

3.3 Optimal feedback policies via dynamic programming

Dynamic programming (DP) is a powerful approach for solving complex optimization problems by breaking them down into a series of smaller, more manageable subproblems. The idea is to solve each subproblem just once, store the solutions, and use them to build up solutions to bigger problems. In dynamical systems, DP is often used to transform an optimal control problem over a finite horizon into a sequence of subproblems over shorter horizons.

To understand dynamic programming, imagine that you are trying to find the shortest path from point A to point B. If you already know the shortest path and take a step along it to a new point, let's call it C, the path you intended to take from C to B must also be the shortest possible. Why? Because if there was a shorter path from C to B, you could have used that instead, and your original path wouldn't have been the shortest. This leads us to a key concept in dynamic programming called the *principle of optimality:*

> "Any optimal policy has the property that, whatever the current state and decision, the remaining decisions must constitute an optimal policy with regard to the state resulting from the current decision."

Dynamic programming uses this principle to solve problems by breaking them down into smaller subproblems. For example, in a shortest path problem, you can start from point B (the destination) and consider all the places that connect directly to B. For each of these points, you determine the shortest path to B and record the corresponding distance. When you expand the search to paths that are two steps away, any optimal path to B must pass through a directly connected point and then follow an optimal path from there on. To find the best route, you now combine the cost of reaching an intermediate point with the shortest cost-to-go from there to B (computed in the previous step). In this way, each intermediate cost-to-go calculation builds a roadmap that guides you toward the destination through the most efficient path. By repeatedly expanding the search to paths that are three steps long, four steps long (etc.), you layer solutions to these smaller subproblems to construct an optimal solution for the entire problem. Before we formalize the dynamic programming approach, we'll develop intuition through a few examples.

Example: computing the fastest route between two cities.

Figure 3.10 shows the travel times by car for popular routes between some of southern Sweden's largest cities. The nodes in the graph represent cities while arcs correspond to route choices. The numbers on each arc specify the estimated travel time in minutes.

We are interested in finding the fastest route that takes us from M (Malmö) to S (Stockholm). To this end, we will use our earlier observation. In each city, we have a choice where to move next, and will then follow a fastest path from that city to S. The total travel time from the city is the time to the next city plus the shortest travel time from there to S. To make an optimal route choice, we must know the shortest possible travel time (the "cost to go") from each nearby city to S.

To determine the cost-to-go, we proceed iteratively and compute a sequence of vectors $\{v_n\}$, whose entries represent the shortest travel time from each city to S that traverses at most n arcs. We use $+\infty$ to mark that S is not reachable in n or less arc traversals. Clearly, the only finite entry in v_0 (no arc traversals) is the one corresponding to S itself. Moving on to v_1, we see that we can reach S from 3 cities (V, Ö and N) by traversing a single arc. We thus record the corresponding direct distances in v_1; see Table 3.1. Things become a little more interesting when we allow for up to two arc traversals. For example, it is now possible to reach S from Ö in three different ways: the direct route, the route via V, or the one via N. Letting $\tau[A,B]$ denote the travel time on the arc connecting cities A and B, we can now compare these three options and use the best:

$$v_2[\ddot{O}] = \min\left\{v_1[\ddot{O}], \tau[\ddot{O},V] + v_1[V], \tau[\ddot{O},N] + v_1[N]\right\}.$$

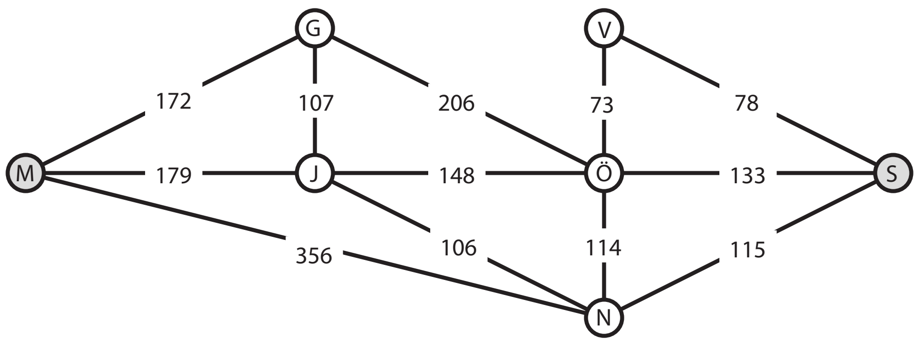


Figure 3.10: Travel times (minutes) by car between some of Sweden's largest cities.

We note that the direct route is the fastest, hence $v_2[\ddot{O}] = v_1[\ddot{O}]$. With two arc traversals, it is also possible to find a route from M to S via N. The corresponding shortest distance is

$$v_2[M] = \tau[M,N] + v_1[N] = 471$$

Moving on to v_3, we have more options for reaching S from M, since all of M's neighbors can reach S in two arc traversals or less. Specifically,

$$\begin{aligned} v_3[M] &= \min\{v_2[M], \tau[M,G] + v_2[G], \tau[M,J] + v_2[J], \tau[M,N] + v_2[N]\} = \\ &= \tau[M,J] + v_2[J] = 400. \end{aligned}$$

We have thus found a new shorter route from M to S. After completing all the entries of v_3, we move on to v_4 only to notice that it is identical to v_3. There is no need to try to repeat the process and compute v_5, since this would yield the same result. Thus, we are done and have found the shortest travel times from all cities to S.

Table 3.1: The vectors v_n contain the shortest travel times from all considered cities to S on routes that traverse at most n arcs.

city	v_0	v_1	v_2	v_3
S	0	0	0	0
V	$+\infty$	78	78	78
Ö	$+\infty$	133	133	133
N	$+\infty$	115	115	115
G	$+\infty$	$+\infty$	339	328
J	$+\infty$	$+\infty$	221	221
M	$+\infty$	$+\infty$	471	400

Although v_3 only contains the shortest distances, this is all we need to know to compute the optimal action. In each stage, the optimal policy is to use the one that minimizes the cost to the next city, plus the cost-to-go from that city in the next stage. For example, if we start in M, we note that

$$v_3[M] = v_2[J] + \tau[M,J],$$

Hence, the route from M to J will be on an optimal path. Then, since

$$v_2[J] = v_1[N] + \tau[J,N]$$

it will be optimal to continue from J to N. Finally, as

$$v_1[N] = v_0[S] + \tau[N,S]$$

the direct route from N to S terminates an optimal path. This optimal path is $M \to J \to N \to S$.

Let us continue by solving a problem of slightly different flavor.

Example: using the minimum number of coins to settle a debt.
Given coins valued at 1, 2 and 5 SEK, how can we determine the minimal number of coins necessary to pay a given amount? We can notice that the principle of optimality is at play: if you have found the coins necessary to settle your debt in an optimal way and pay one of these coins, the remaining coins must be an optimal way to settle the rest of what you owe. It therefore makes sense to form vectors v_n whose entries $v_n[i]$ are the minimal number of coins necessary to pay i SEK using at most n coins, see Table 3.2. As before, we use $+\infty$ to indicate that the corresponding amount cannot be paid using the allowed number of coins.

Table 3.2: The cost-to-go vectors v_n represent the minimal number of coins needed to settle each debt amount using at most n coins.

amount	v_0	v_1	v_2	v_3
0	0	0	0	0
1	$+\infty$	1	1	1
2	$+\infty$	1	1	1
3	$+\infty$	$+\infty$	2	2
4	$+\infty$	$+\infty$	2	2
5	$+\infty$	1	1	1
6	$+\infty$	$+\infty$	2	2
7	$+\infty$	$+\infty$	2	2
8	$+\infty$	$+\infty$	$+\infty$	3
9	$+\infty$	$+\infty$	$+\infty$	3

For notational simplicity, we begin with v_0 (the amounts that we can settle without any coins). Next, v_1 has entries $+1$ at positions corresponding to the denomination of the coins. For v_2, we note

$$\begin{aligned} v_2[i] &= \min\{v_1[i], v_1[i-1]+1, v_1[i-2]+1, v_1[i-5]+1\} = \\ &= \min\{v_1[i], 1+\min\{v_1[i-1], v_1[i-2], v_1[i-5]\}\} \end{aligned}$$

In words, this means that to settle i SEK with at most 2 coins, we should settle it with 1 coin (if possible), or use the best of the three possibilities that the different coin denominations offer. The computation of v_n for $n \geq 3$ is analogous.

The dynamic programming algorithm
Although the dynamic programming algorithm applies to finite-horizon problems on our standard form (3.1), it is convenient to rephrase the constraints as $u_t \in \mathcal{U}(x_t)$ and consider

$$\begin{array}{lll} \text{minimize} & \sum_{t=0}^{T-1} g_t(x_t,u_t) + g_T(x_T) & \\ \text{subject to} & x_{t+1} = f_t(x_t,u_t) & t = 0,\dots,T-1 \\ & u_t \in \mathcal{U}_t(x_t) & t = 0,\dots,T-1 \end{array} \tag{3.6}$$

To get a correspondence with (3.1), we let $\mathcal{U}_t(x_t) = \{u \mid (x_t,u) \in \mathcal{C}_t\}$ and $g_T(x) = +\infty$ if $x \notin \mathcal{X}_T$.

We are interested in finding an optimal policy $\mu^\star = \{\mu_0^\star, \mu_1^\star, \dots, \mu_{T-1}^\star\}$ such that $u_t = \mu_t^\star(x_t)$ defines the optimal action from state x_t in stage t. Associated to the optimal policy is its cost

$$v_0(x_0) = \sum_{t=0}^{T-1} g_t(x_t, \mu_t^\star(x_t)) + g_T(x_T)$$

where $x_{t+1} = f_t(x_t, \mu_t^\star(x_t))$. Note that v_0 is a function, mapping each initial state x_0 into the cost accumulated along the optimal trajectory that starts at x_0.

At an arbitrary stage t, the cost-to-go until the end of the horizon is

$$v_t(x_t) = \min_{\mu_t,\dots,\mu_{T-1}} \sum_{k=t}^{T-1} g_k(x_k, \mu_k(x_k)) + g_T(x_T)$$

where $x_{t+1} = f_t(x_t, \mu_t(u_t))$. By the principle of optimality, the truncated policy $\{\mu_{t+1}^\star, \dots, \mu_{T-1}^\star\}$ is optimal from stage $t+1$ and on. We can therefore express the cost-to-go as

$$\begin{aligned} v_t(x_t) &= \min_{\mu_t} \quad g_t(x_t, \mu_t(x_t)) + \sum_{k=t+1}^{T-1} g_k(x_k, \mu_k^\star(x_k)) + g_T(x_T) = \\ &= \min_{\mu_t} \quad g_t(x_t, \mu_t(x_t)) + v_{t+1}(x_{t+1}) = \\ &= \min_{\mu_t} \quad g_t(x_t, \mu_t(x_t)) + v_{t+1}(f_t(x_t, \mu_t(x_t))) \end{aligned}$$

Hence, given the cost-to-go function v_{t+1} at stage $t+1$, we can compute the cost-to-go function at stage t by optimizing the sum of the cost of stage t and the cost-to-go from the resulting state. Noticing that $v_T(x_T) = g_T(x_T)$ leads to the following result.

Theorem 3.3.1 — The DP algorithm. For every initial state x_0, the optimal cost of the basic problem is equal to $v_0(x_0)$, given by the last step of the following algorithm, which proceeds backwards from stage $T-1$ to stage 0:

$$v_T(x) = g_T(x) \tag{3.7}$$

$$v_t(x) = \min_{u \in \mathcal{U}_t(x_t)} \{g_t(x,u) + v_{t+1}(f_t(x,u))\} \qquad t = T-1, T-2, \dots, 0 \tag{3.8}$$

If $u_t^\star = \mu_t^\star(x)$ minimizes the right-hand-side of (3.8) for each x and each t, then the policy $\mu^\star = \{\mu_0^\star, \dots, \mu_{T-1}^\star\}$ is optimal.

As the next two examples show, both the fastest route and the debt settlement problem can be posed as finite-horizon optimal control problem on our standard form and hence solved using the dynamic programming algorithm detailed above.

■ **Example 3.6 — fastest route problem.** Here, the horizon T is the maximum number of arc traversals. The stage index t corresponds to the number of arc traversals left in our budget, *i.e.* in stage t we are allowed to traverse at most $T-t$ additional arcs. The state x_t is the current location (city) in the graph and the control is the decision which arc (road) to traverse next. The set $\mathcal{U}_t(x_t)$ contains the arcs connected to node x_t, and $f_t(x_t, u_t)$ maps the current node to the node at the end of the selected arc. Since we need to be in S after T arc traversals, we have $\mathcal{X}_T = \{S\}$. There is no terminal cost, $g_T \equiv 0$, but the stage cost $g_t(x_t, u_t)$ is the travel time along the arc selected by u_t. ■

■ **Example 3.7 — minimum coin settlement problem.** In this problem, the stage index corresponds to the maximal number of coins that we permit ourselves to use, and the stage index corresponds to how many coins that we (at most) have handed over to the creditor. The state x_t is the debt which remains to be settled, the control chooses what coin denomination to use, and the transition function adjusts the settled amount accordingly. Since the debt has to be cleared at stage T, we let $\mathcal{X}_T = \{0\}$ and set $g_T \equiv 0$. The stage cost is 1 if we use a coin, and zero otherwise. ■

Some remarks are in order. First, as we have already mentioned, $v_t(\cdot)$ is a function, mapping every possible state x_t in stage t into the cost of the associated optimal trajectory from stage t until stage T. If the state vector can only take a finite set of values $\{x^{(1)}, \dots, x^{(I)}\}$, as was the case in the fastest route and coin settlement problems, then we can represent v_t as a list of tuples $(x_t^{(i)}, v_t^{(i)})$ for $i = 1, \dots, I$. However, when the state vector is continuous, it can be challenging to find a closed-form representation of the cost-to-go functions. Second, it is not always easy to carry out the minimization step required to propagate the cost-to-go functions in (3.8). If the control action is discrete, then we can simply evaluate the cost of all possible control actions and take the one that yields the smallest cost-to-go. This was effectively what we did in the fastest route and coin settlement problems. But when u is continuous, this step can be challenging, both analytically (in the rare cases that an analytical solution exists) and numerically. The next example demonstrates a problem that can be solved analytically, despite that both the state and controls are continuous.

■ **Example 3.8** Consider the following sequential decision-making problem

$$\begin{array}{ll} \underset{u_0,\dots,u_{T-1}}{\text{minimize}} & \sum_{t=0}^{T-1} g_t(x_t, u_t) \\ \text{subject to} & x_{t+1} = f(x_t, u_t) \end{array} \tag{3.9}$$

where $g_t = d^t(u_t^2/x_t - pu_t)$ and $f(x_t, u_t) = x_t - u_t$ The problem is motivated by fair pricing of a T-year lease of an underground mine: the state x_t is the amount of ore left in the mine at the beginning of year t, and u_t the amount of ore extracted during that year. Selling the ore gives a profit of px_t but the extraction cost is u_t^2/x_t to model that extraction cost increases as the mine is depleted. The discount factor is $d = 1/(1+r)$ where r is the current interest rate. In this way, we maximize $-\sum_t g_t(x_t, u_t)$ which represents the net present value of the lease, *i.e.* the amount of money you would need to put in the bank at the beginning of the lease period to be able to replicate the income stream that can be generated from the lease.

Since we do not have a terminal cost or constraints, $v_T(x) \equiv 0$. To proceed, we consider year $t = T-1$, in which the optimal action minimizes $g_{T-1}(x, u)$. The first-order optimality condition

$$\frac{\partial g_{T-1}(x,u)}{\partial u} = d^{T-1}(2u/x - p) = 0$$

yields

$$u^\star = \mu_{T-1}^\star(x) = \frac{p}{2}x := L_{T-1}x$$

with the associated cost-to-go

$$v_{T-1}(x) = g_{T-1}(x, \mu_{T-1}^\star(x)) = -d^{T-1}L_{T-1}^2 x := d^{T-1}\alpha_{T-1}x$$

Note that v_{T-1} is a linear function of x. Next, we consider year $T-2$, when we need to minimize

$$\begin{aligned} & g_{T-2}(x,u) + v_{T-1}(f(x,u)) = g_{T-2}(x,u) + v_{T-1}(x-u) = \\ & = d^{T-2}\left(\frac{u^2}{x} - pu\right) + d^{T-1}\alpha_{T-1}(x-u) = \\ & = d^{T-2}\left(\frac{u^2}{x} - (d\alpha_{T-1} + p)u + d\alpha_{T-1}x\right) \end{aligned}$$

with respect to u. The first-order optimality conditions yield

$$u_{T-2}^\star(x) = \frac{1}{2}(p + d\alpha_{T-1})x := L_{T-2}x$$

Inserting this control in the expression for the cost-to-go gives

$$v_{T-2}(x) = d^{T-2}\left(-L_{T-2}^2 + d\alpha_{T-1}\right)x := d^{T-2}\alpha_{T-2}x$$

The first few steps of the DP algorithm suggest that $v_t(x) = d^t \alpha_t x$. We can prove this hypothesis formally using induction. The hypothesis holds true for $t = T$ with $\alpha_T = 0$. Given that the hypothesis holds for stage $t+1$, the cost-to-go at stage t is

$$v_t(x) = \min_u g_t(x,u) + v_{t+1}(x-u) = \min_u d^t \left(\frac{u^2}{x} - (d\alpha_{t+1} + p)u + d\alpha_{t+1}x \right)$$

Using the same calculations as in stage $T-2$, we find that $u_t^\star = L_t x$ and

$$v_t(x) = d^t \alpha_t x$$

where $\alpha_t = (-L_t^2 + d\alpha_{t+1})$ and $L_t = (p + d\alpha_{t+1})/2$. Hence, the hypothesis holds true. The fair price of the lease (the optimal value of (3.9)) is therefore $v_0(x_0) = \alpha_0 x_0$ where α_k can be computed recursively by initializing $\alpha_T = 0$ and then evaluating

$$L_k = \frac{1}{2}(p + d\alpha_{k+1})$$
$$\alpha_k = \left(-L_k^2 + d\alpha_{k+1}\right)$$

for $k = T-1, T-2, \ldots, 0$. ■

The example demonstrates the standard approach to solve dynamic programming problems with continuous states and controls. We first perform a few steps of the DP algorithm in order to form a hypothesis for how the cost-to-go function can be parameterized. In the above example, the cost-to-go was linear in the state. We then use induction to prove that the hypothesis holds for an arbitrary stage. Typically, the induction step also results in explicit update formulas for how the parameters of v_t (α_t in the example) can be computed from the ones of v_{t+1} and other problem data.

Solving dynamic programming problems numerically

A drawback with the dynamic programming approach described so far is that we need to find a representation of the value function that can be propagated from one stage to the next. When states and controls are continuous, this is sometimes difficult, and it may be more convenient to resort to numerical computations. A simple numerical approach to solving dynamic programming problems over a finite horizon is to define a grid on the state space and tabulate the value function at these grid points. A conceptual algorithm is shown as Algorithm 2 below.

Algorithm 2 Numerical solution to finite-time DP

```
1: Define grid $\mathcal{X}_\mathcal{G} = \{x^{(1)}, x^{(2)}, \ldots, x^{(\mathcal{G})}\}$ of test point to cover the state space
2: Initialize $v_T[i] = g_T(x^{(i)})$ for all $x^{(i)} \in \mathcal{X}_\mathcal{G}$.
3: for $k = T-1, T-2, \ldots, 1$ do
4:     for $i = 1, 2, \ldots, \mathcal{G}$ do
5:         $v_k[i] = \min_{u \in \mathcal{U}_k(x^{(i)})} \left\{ g_k(x^{(i)}, u) + \hat{v}_{k+1}(f_k(x^{(i)}, u)) \right\}$
```

Since the next state $x_{k+1} = f_k(x_k, u_k)$ is not guaranteed to coincide with any of the grid points, we introduce $\hat{v}_{k+1}(x)$, the approximation of v_{k+1} given its tabulated values $v_{k+1}[i]$. The simplest approximation is to use linear or multi-linear interpolation, but many options exist. One also needs to pay attention to edge effects that occur when there is no feasible control that can keep the next state inside the gridded part of the state space. We refer to [14] for details.

■ **Example 3.9** Consider the mechanical system from Example 3.3 and assume that we want to minimize a quadratic cost over a time horizon of $T = 10$, defined by the stage costs

$$g_t(x,u) = x^\top x + 10u^2 \quad \text{for } t = 0, 1, \ldots, 9, \quad g_{10}(x_t) = x^\top x.$$

We solve the dynamic programming problem numerically on a uniform grid of $\|x\|_\infty \leq 5$ with 51 gridpoints in each direction. We also add the constraint that the state must remain on the grid.

The optimal feedback and cost-to-go functions are shown in Figure 3.11 (top). The optimal control appears to be linear and the cost-to-go looks quadratic, apart from an edge effect that appears since we require that the state remains on $\|x\|_\infty \leq 5$. In Chapter 4, we will *prove* that in the absence of constraints, the optimal control policy for a linear system with quadratic cost is indeed linear, and that its cost-to-go function is quadratic. This solution can be computed very efficiently without the need to grid the state-space.

If we add the constraints that $|u_t| \leq 1$, we find the optimal control in Figure 3.11 (bottom left). Clearly, the optimal policy is no longer linear. In addition, there is a significant subset of states from which no admissible control can keep the future states within the grid. Although harder to verify visually, the cost-to-go function is no longer quadratic (same figure, bottom right). ■

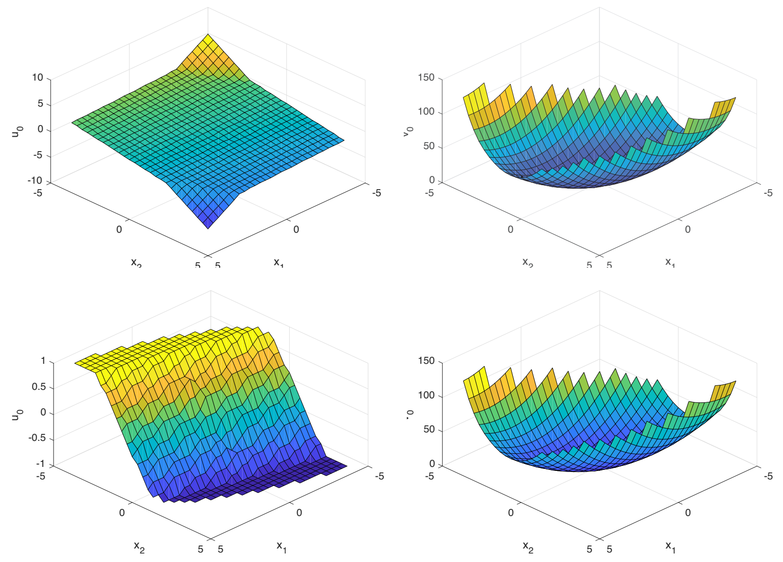


Figure 3.11: Optimal feedback policies (left column) and associated cost-to-go functions (right column). The upper row is for a linear system with quadratic cost: the optimal feedback is linear and the cost-to-go is quadratic. The lower row is for a linear system with state and control constraints: the optimal policy is now non-linear and the cost-to-go no longer quadratic.

Dynamic programming as optimization of multi-stage functions

While the dynamic programming formalism developed so far is elegant, it is not uncommon to find problems that are inconvenient to transform into the standard form. It may then be useful to have an alternative understanding of dynamic programming as a means for minimizing multistage problems

$$G_T(x_1,\dots,x_T;x_0) = g_0(x_0,x_1) + g_1(x_1,x_2) + \cdots + g_{T-1}(x_{T-1},x_T) + g_T(x_T).$$

As before, the state x_t couples the stage costs, since it appears in both g_{t-1} and g_t. To develop some intuition for how to find the optimal sequence of states, let us consider the cost

$$G_2(x_1,x_2;x_0) = g_0(x_0,x_1) + g_1(x_1,x_2) + g_2(x_2)$$

Due to the structure of the problem, we can re-write the optimal value

$$G_2^\star(x_0) = \min_{x_1,x_2} G_2(x_1,x_2;x_0) =$$
$$= \min_{x_1}\left\{ g_0(x_0,x_1) + \min_{x_2}\{g_1(x_1,x_2) + g_2(x_2)\}\right\}$$

In the inner minimization problem,

$$\underset{x_2}{\text{minimize}} \quad g_1(x_1,x_2)+g_2(x_2)$$

we can view x_1 as a fixed parameter and derive an optimizer $x_2^\star(x_1)$ which depends on x_1. If we assume that this optimizer is unique for each x_1, the optimal value of this problem

$$v_1(x_1)=g_1(x_1,x_2^\star(x_1))+g_2(x_2^\star(x_1))$$

is a function of x_1 and the outer minimization problem takes the form

$$\underset{x_1}{\text{minimize}} \quad g_0(x_0,x_1)+v_1(x_1).$$

By solving the outer problem in a similar manner, we find $G_2^\star(x_0)=v_0(x_0)$.

Using an analogous argument, the optimal value of the T-stage problem $G_T^\star(x_0)$ equals $v_0(x_0)$ where the value functions $v_n(x)$ are computed recursively:

$$\begin{aligned} v_T(x) &= g_T(x) \\ v_t(x) &= \min_z \{g_t(x,z)+v_{t+1}(z)\} \qquad\qquad t=T-1,T-2,\dots,0 \end{aligned}$$

If we instead are given a multistage problem on the form

$$G_T(x_0,\dots,x_{T-1};x_T)=g_0(x_0)+g_1(x_0,x_1)+\cdots+g_T(x_{T-1},x_T)$$

it is more natural to proceed using a forward induction

$$\begin{aligned} w_0(x) &= g_0(x) \\ w_{t+1}(x) &= \min_z \{w_t(z)+g_{t+1}(z,x))\} \qquad\qquad t=0,1,\dots,T-1 \end{aligned}$$

The function $w_k(x)$ has the interpretation of the smallest possible accumulated cost over the t first stages and is referred to as the *arrival cost* of stage t; furthermore, $G_T^\star(x_T)=w_T(x_T)$ quantifies the smallest total cost for which the decision of stage T can be made equal to x_T.

A few words about the infinite-horizon case

In many control problems, there is no natural fixed terminal time but we are rather interested in optimizing the performance over an infinite time-horizon. We will not cover such problems in detail in this course, as they are more technical to solve and would deviate our focus. Nevertheless, we will try to give high-level overview of some of the central ideas based on [7].

To this end, consider the infinite-horizon optimal control problem

$$\begin{array}{ll} \text{minimize} & \lim_{T\to\infty} \sum_{t=0}^T g(x_t,u_t) \\ \text{subject to} & x_{t+1}=f(x_t,u_t) \\ & u_t\in\mathcal{U}(x_t), \qquad x_t\in\mathcal{X} \end{array}$$

Note that both the stage costs and the dynamics are time-invariant (neither g nor f depend on t). We are interested in finding feedback policies on the form $\pi=\{\mu_0,\mu_1,\dots\}$, where each $\mu_t(x)\in\mathcal{U}(x)$ for all $x\in\mathcal{X}$. We let Π denote the set of all such policies, and call π stationary if it has the form $\{\mu,\mu,\dots\}$. By applying a policy $\pi\in\Pi$ to the system, we generate state and control pairs $(x_t,\mu_t(x_t))$ for $t=0,1,\dots$ whose cost

$$v_\pi(x_0)=\lim_{T\to\infty}\sum_{t=0}^T g(x_t,\mu_t(x_t)).$$

is a function of the initial state x_0. We require that the state costs are non-negative, *i.e.*, that

$$g(x,u) \geq 0 \tag{3.10}$$

for all $x \in \mathcal{X}$ and $u \in \mathcal{U}(x)$. To account for the possibility that there could be states for which the policy π is unable to obtain a finite cost, we allow v_π to be extended real-valued, *i.e.* take on values in $[0,\infty]$, The value function

$$v^\star(x) = \inf_{\pi \in \Pi} v_\pi(x)$$

describes the minimal infinite-horizon cost, and a policy $\pi^\star \in \Pi$ is optimal if it attains the smallest value of $v_\pi(x)$ for all x, *i.e.*

$$v_{\pi^\star}(x) = \inf_{\pi \in \Pi} v_\pi(x) = v^\star(x) \qquad \forall x.$$

Given our discussion for the finite-horizon case, one can expect that the value function $v^\star(x)$ under stationary policies satisfies the *Bellman equation*

$$v^\star(x) = \inf_{u \in \mathcal{U}(x)} \{g(x,u) + v^\star(f(x,u))\} \qquad \forall x \tag{3.11}$$

and that the stationary optimal policy can be found by minimizing the right-hand side of this equation. Note that the Bellman equation is a *functional equation*, i.e. the unknown quantity $v^\star(x)$ is a function and the equality should hold for all x. There is also a subtlety in the sense that if $\tilde{v}^\star$ satisfies the Bellman equation, then so does $\tilde{v}^\star + c$ for any positive constant c. It turns out that for our problem class, the optimal cost function is the smallest one which satisfies (3.11):

Theorem 3.3.2 Assume that the stage costs $g(x,u)$ satisfy (3.10). Then

(a) if $v^\star$ satisfies the Bellman equation (3.8) and $\tilde{v} : \mathcal{X} \mapsto [0,\infty]$ also satisfies (3.8), then $v^\star \leq \tilde{v}$.

(b) For all stationary policies π, we have

$$v_\pi(x) = g(x,\mu(x)) + v_\pi(f(x,\mu(x))) \qquad \forall x \in \mathcal{X}$$

(c) A stationary policy $\mu^\star$ is optimal if and only if

$$\mu^\star(x) \in \arg\min_{u \in \mathcal{U}(x)} \{g(x,u) + v^\star(f(x,u))\} \qquad \forall x \in \mathcal{X}$$

There are two principally different ways of finding the optimal stationary policy. The *value iteration* starts with some non-negative function $v_0 : X \mapsto [0,\infty]$ and generates a sequence $\{v_k\}$ via

$$v_{k+1}(x) = \inf_{u \in \mathcal{U}(x)} \{g(x,u) + v_k(f(x,u))\}.$$

Policy iteration, on the other hand, starts with an admissible μ_0 and interleaves policy evaluations

$$v_{\mu_k}(x) = g(x,\mu_k(x)) + v_{\mu_k}(f(x,\mu_k(x)))$$

with policy improvements

$$\mu_{k+1}(x) \in \arg\min_{u \in \mathcal{U}(x)} \{g(x,u) + v_{\mu_k}(f(x,u))\}$$

Both the policy and the value iterations are well-behaved and converge under some additional (and for many real-world systems mild) conditions; we refer to [7] and the exercises for details.

3.4 Exercises

Problem 3.1 The double integrator

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

models the movement of a friction-less unit mass subject to an external force $u(t)$. Under zero-order hold sampling with a sampling time of 1 second, the corresponding discrete-time model is

$$x_{t+1}=\begin{pmatrix}1&1\\0&1\end{pmatrix}x_t+\begin{pmatrix}0.5\\1\end{pmatrix}u_t$$

(a) Find a control sequence $\{u_0,u_1,\dots,u_9\}$ that drives the system from rest at the origin to rest at $x=5$ while minimizing the total energy $\sum_{t=0}^{9}u_t^2$. Plot the state trajectory and the control input. Explain briefly what you observe.

(b) Add the constraint $x_5=0$ (*i.e.* that the mass should be at the origin at time $t=5$). Before you compute the optimal control, try to figure out what you think it should be. Execute the code and plot the state trajectory and the input. Describe your observations.

Problem 3.2 Consider the triple integrator under zero-order hold sampling with $h=1$

$$x_{t+1}=\begin{pmatrix}1&1&0.5\\0&1&1\\0&0&1\end{pmatrix}x_t+\begin{pmatrix}0.167\\0.5\\1\end{pmatrix}u_t,\qquad y_t=\begin{pmatrix}1&0&0\end{pmatrix}x_t$$

It can be useful to interpret the states as the acceleration, velocity, and position of a unit mass.

(a) For what planning horizons T will you be able to find a control sequence $\{u_t\}$ that drives the system from $x_0=0$ to $x_T=\begin{pmatrix}5&0&0\end{pmatrix}^T$?

(b) Add the input constraint $|u_t|\le 1$. What is now the shortest time T for which you can find a solution? Explain what you observe! Try to solve the problem without any numerical computations, then validate your answer using cvxpy.

Hint. If you start from $x(0)=\dot{x}(0)=\ddot{x}(0)=0$, what constant control $u(t)=u$ should you apply to reach $x(T)=5$ as quickly as possible? Try to solve $x^{(3)}(t)=u$.

(c) Add the constraint that the velocity (the second state) can not exceed $1/2$. What is now the smallest planning horizon that you can have? Plot the state trajectories and the control input and comment on what you observe.

(d) Remove the state constraint from (c) and add a rate constraint on the input: $|u_{t+1}-u_t|\le 1/2$. What is the smallest value of T for which you can solve the planning problem? Plot the state trajectories and input sequence. Explain what you observe.

Problem 3.3 Modern space missions rely on rockets to be able to land safely. The following differential equations describe the linearized dynamics of a 60 meter rocket with separate horizontal and vertical thursters.

$$\begin{aligned}\ddot{x}(t)&=u_1(t)-u_2^0\theta(t)\\ \ddot{y}(t)&=u_2(t)-g\\ \ddot{\theta}(t)&=\alpha u_1(t)\end{aligned}$$

where $\alpha=0.1$, $g=9.81$ and u_2^0 is the average vertical control u_2 during the descent (the linearization is made around a trajectory for which $(u_1(t),u_2(t),\theta(t))\approx(0,u_2^0,0)$). The horizontal thrusters are subject to magnitude limitations $|u_1(t)|\le 5$.

We want to plan the descent from an initial position $x(0) = -20$, $\dot{x}(0) = 0$, $y(0) = 200$, $\dot{y}(0) = -40$, $\theta(0) = 0$ and $\dot{\theta} = 0$. Use sampling time $h = 0.1$ seconds and plan over a horizon of 10 seconds. One can show that the average vertical thrust for this maneuver is $u_2^0 = 14.41$. Find the optimal control sequence that brings the rocket to rest at the origin while minimizing $\sum_t \|u_t\|_1$.

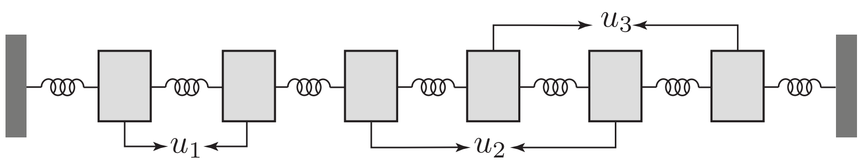


Figure 3.12: Moving masses in Problem 3.4

Problem 3.4 Consider the system shown in Figure 3.4. The system consists of six unit masses connected by springs to each other, and to walls on either side. There are three actuators, which exert tensions between certain pairs of masses. Its dynamics can be described by

$$\dot{x}(t) = \begin{pmatrix} 0 & I \\ A_{21} & 0 \end{pmatrix} x(t) + \begin{pmatrix} 0 \\ B_2 \end{pmatrix} u(t), \qquad y(t) = \begin{pmatrix} I & 0 \end{pmatrix} x(t)$$

where

$$A_{21} = \begin{pmatrix} -2 & 1 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 1 & -2 \end{pmatrix}, \text{ and } B_2 = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix}$$

The first six states are the velocities of the masses, and the last states are their displacements from their equilibrium positions. The actuators can exert a maximum force of ± 0.5, and the displacements of the masses cannot exceed ± 4.

(a) Pose the problem of minimizing the energy required to transfer the masses from rest at $x_0 = 0$ to rest at $x_T = 2$ as an optimal control problem on the standard form.

(b) Use a sampling time of $h = 0.5$ seconds, and a horizon of $T = 100$ samples. What is the minimum energy required for the state transfer?

(c) Plot how the total energy depends on the horizon length. What is the smallest time horizon in which you can manage to transfer the masses?

Problem 3.5 In this example, we will consider dynamic programming for the system

$$x_{t+1} = x_t + u_t$$

where the state is discrete and constrained, $x_t \in \{-2, -1, 0, 1, 2\}$, while $u_t \in \{-1, 0, 1\}$. We are interested in finding the control which minimizes the cost

$$x_3^2 + \sum_{t=0}^{2} x_t^2 + u_t^2.$$

(a) Use dynamic programming to compute the cost-to-go at time zero and derive the optimal control sequence $\{u_0, u_1, u_2\}$ for $x_0 = 2$.

(b) Let's now fix the final state to $x_3 = 1$. Compute the cost-to-go at time zero and determine the optimal control sequence for $x_0 = 2$.

(c) Write a short Python script that performs the dynamic programming recursion for (a) and (b).

Problem 3.6 Write a Python function for discrete dynamic programming with the signature

```
def discrete_dp(F, G, GT, T):
```

Here, F is a 3-dimensional array where $F[x,a,t]$ contains the next state, given that the system is at state x at time t and the control action is a. Similarly, G is a 3-dimensional array where $G[x,a,t]$ is the cost of being in state x and applying control action a at stage t, and GT is a vector of terminal costs for each state and T is the horizon length. Use your code to solve the fastest route and coin settlement problems in Section 3.3.

Problem 3.7 Consider the same system as in Problem 3.5, i.e.

$$x_{t+1} = x_t + u_t$$

and the cost

$$x_3^2 + \sum_{t=0}^{2} x_t^2 + u_t^2$$

but let the state and control variables be continuous and unconstrained. Compute the cost-to-go function $v_0(x)$, and compare it with your result for the discrete and constrained problem.

Problem 3.8 Solve the following linear-quadratic problems using dynamic programming. Specify the optimal strategy $\hat{u}_0, \dots, \hat{u}_3$, and the cost-to-go function at $t = 0$.

(a)

$$\begin{aligned} \underset{u_0,\dots,u_3}{\text{minimize}} \quad & x_3^2 + \sum_{k=0}^{2} u_k^2 \\ \text{subject to} \quad & x_{k+1} = x_k + u_k \end{aligned}$$

(b)

$$\begin{aligned} \underset{u_0,\dots,u_2}{\text{minimize}} \quad & \sum_{k=0}^{2} u_k^2 \\ \text{subject to} \quad & x_{k+1} = 2x_k - u_k \\ & x_3 = 0 \end{aligned}$$

(c)

$$\begin{aligned} \underset{u_0,\dots,u_2}{\text{minimize}} \quad & x_3^2 + \sum_{k=0}^{2} x_k^2 + u_k^2 \\ \text{subject to} \quad & x_{k+1} = 2x_k + u_k \end{aligned}$$

Problem 3.9 Solve the following nonlinear problems using dynamic programming. Specify the optimal strategy $\hat{u}_0, \dots, \hat{u}_3$, and the resulting objective.

(a)

$$\begin{aligned} \underset{u_0,\dots,u_2}{\text{maximize}} \quad & \sum_{k=0}^{2} x_k - x_3 \\ \text{subject to} \quad & x_{k+1} = x_k u_k \\ & 0 \le u_k \le 1, \quad k = 0, \dots, 2 \\ & x_0 = 1 \end{aligned}$$

(b)

$$\begin{aligned}
\underset{u_0,\dots,u_2}{\text{maximize}} \quad & \sum_{k=0}^{2} \log u_k x_k + \log x_3 \\
\text{subjec to} \quad & x_{k+1} = x_k(1-u_k) \\
& 0 < u_k < 1, \quad k = 0,\dots,2 \\
& x_0 = 4 \\
& (x_k > 0, \quad k = 0,\dots,3)
\end{aligned}$$

(c)

$$\begin{aligned}
\underset{u_0,\dots,u_2}{\text{maximize}} \quad & \sum_{k=0}^{2} u_k^4 + x_3^4 \\
\text{subject to} \quad & x_{k+1} = x_k + u_k \\
& x_0 = 4
\end{aligned}$$

Problem 3.10 A certain material is passed through two ovens in sequence. We are interested in how to optimally heat the material while it passes through the ovens. Let x_0 be the initial temperature of the material, x_k for $k = 1,2$ be the temperature when the material exits oven k, and u_{k-1} for $k = 1,2$ be the temperature in oven k. The temperature dynamics is modeled as

$$x_{k+1} = (1-a)x_k + au_k, \qquad k = 1,2,$$

where a is a known scalar in the interval $(0,1)$. The objective is to get the final temperature x_2 close to a given target T_{ref} while expending little energy. This objective is expressed by the cost

$$J(x_2,u_0,u_1) = r(x_2 - T_{\text{ref}})^2 + u_0^2 + u_1^2$$

where r is a given scalar. For simplicity, we assume that u is unconstrained.

(a) Formulate the problem as a finite-time optimal control problem on standard form.
(b) Solve the problem using dynamic programming for $a = 1/2$, $T_{\text{ref}} = 0$ and $r = 1$.
(c) Solve the problem for arbitrary feasible values of a, T_{ref} and r.

Problem 3.11 You currently own $x_c > T$ units of company shares, of some company C. You have reason to believe that company C will go bankrupt in N years, at which your shares would be worth nothing. However, up to that time, you receive dividends equal to $\theta\times$ your current share holdings, where $0 < \theta < 1$. In addition, every year you are allowed to buy or sell at most *one* unit of shares without any extra cost. Your intuition tells you that it should be possible to make a profit if you act in a clever way during these T years (assuming your bankruptcy assumption is true). Therefore, you formulate the following discrete optimal control problem:

$$\begin{aligned}
\underset{u_k}{\text{maximize}} \quad & \sum_{k=0}^{T-1} (\theta x_k) - 2x_T \\
\text{subject to} \quad & x_{k+1} = x_k + u_k \\
& |u_k| \leq 1, \quad k = 0,\dots,T-1 \\
& x_0 = x_c
\end{aligned}$$

The intuition of the objective is that you each year receive dividends θx_k that you do not reinvest. At the end of the period, you lose whatever amount x_T you have left as well as gain/lose $x_c - x_T$ depending on how many shares you sold/bought. Note, that x_c is constant and can be removed from the maximization. Furthermore, $x_c > T$ so it is not possible to sell off all shares and you can assume $x_k > 0$ throughout the period. (It is assumed that shares have unit costs, and that you have infinite capital.)

(a) For the specific case $x_c = 3$, $N = 2$, and $\theta = 0.4$, solve the above problem using dynamic programming. Specify the optimal strategy $\hat{u}_0, \hat{u}_1$, and the resulting profit.

(b) Formulate a value function $V_{k+1}(x_{k+1})$ and show by induction that it is indeed a valid value function for the dynamic programming recursion:

$$V_k(x_k) = \max_{|u_k|\le 1} \left(\theta x_k + V_{k+1}(x_{k+1})\right)$$

Hint: Think carefully about what structure the value function had in (a).

(c) Use $V_{N-1}(x_{N-1})$ as a starting point of the recursion obtained in (b), and determine an optimal control law $\hat{u}_k$ as a function of θ and N.

(d) Explain qualitatively how the optimal strategy depends on θ and N.

Problem 3.12

Consider a consumer who lives over T periods and must decide how much of a resource she will consume or save in each period. Let c_t be the consumption in period t and $\ln(c_t)$ be the associated utility for the consumer. Let x_t denote the resource level in period t and x_0 the initial resource level. The resource levels evolve according to $x_{t+1} = x_t - c_t$ and are constrained to be non-negative. The optimal consumption solves the following problem

$$\begin{array}{lll} \text{maximize} & \sum_{t=0}^{T-1} \ln(c_t) & \\ \text{subject to} & x_{t+1} = x_t - c_t & t = 0,1,\dots,T-1 \\ & x_t \ge 0 & t = 0,1,\dots,T \end{array}$$

We are interested in using dynamic programming to find the optimal consumption sequence $\{c_t\}$. Proceed in the following steps.

(a) Put the optimal consumption problem on our standard form for DPs

$$\begin{array}{lll} \text{minimize} & \sum_{t=0}^{T-1} g_t(x_t,u_t) + g_T(x_T) & \\ \text{subject to} & x_{t+1} = f_t(x_t,u_t) & t = 0,1,\dots,T-1 \\ & x_t \in X_t & t = 0,1,\dots,T \\ & u_t \in U_t & t = 0,1,\dots,T-1 \end{array}$$

Hint. $\{c_t\}$ maximizes $\sum_t \ln(c_t)$ if and only if it minimizes $\sum_t -\ln(c_t)$.

(b) Let $T = 1$, and show that it is optimal to consume all remaining resources, *i.e.* $c^\star_{T-1} = x_{T-1}$.

(c) Use dynamic programming to determine the optimal policy for the full T-step horizon.

Hint. Solve the problem for $T = 1, 2$ and 3, to see if you detect a pattern that you can use in a formal induction proof.

Problem 3.13 A public company has profit x_k at year k. This profit is distributed partly to the shareholders as dividends and partly as reinvestment in the company itself. Reinvesting increases the company profit by $\theta\times$ the invested capital. To boost its reputation, the company decides to maximize the amount distributed to the shareholders over an N year period. Therefore, the following discrete optimal control problem is formulated:

$$\begin{array}{ll} \underset{u_k}{\text{maximize}} & \sum_{k=0}^{N-1} (1-u_k)x_k \\ \text{subject to} & x_{k+1} = x_k + \theta u_k x_k \\ & 0 \le u_k \le 1, \quad k = 0,\dots,N-1 \\ & x_0 = x_c \end{array}$$

The intuition of the objective is that each year the profit x_k is divided into $u_k x_k$ which is reinvested and $(1-u_k)x_k$ which is given to the shareholders. $x_c > 0$ is the current profit that can be distributed during the first year.

(a) For the specific case $x_c = 3$, $N = 2$, and $\theta = 1.5$, solve the above problem using dynamic programming. Specify the optimal strategy $\hat{u}_0, \hat{u}_1$, and the total distributed profit.

(b) Formulate a value function $V_{k+1}(x_{k+1})$ and show by induction that it is indeed a valid value function for the dynamic programming recursion:

$$V_k(x_k) = \max_{0 \le u_k \le 1} \left((1-u_k)x_k + V_{k+1}(x_{k+1})\right)$$

Next, determine an optimal control law $\hat{u}_k$ as a function of θ and N.
Hint: Think carefully about what structure the value function had in (a).

(c) Qualitatively describe the optimal strategy.

Problem 3.14 A company has N types of goods that are to be exported through cargo freight. A single unit of the ith good weighs w_i tonnes, and each cargo has a capacity of W tonnes. The value of exporting a single unit of the ith good is given by v_i. The aim is to load the vessels with the distribution of goods that gives the most value possible. Hence, if a_i is the number of units of the ith good, the goal is captured by the following optimization problem:

$$\begin{aligned} \underset{a_1,\dots,a_N}{\text{maximize}} \quad & \sum_{i=1}^{N} v_i a_i \\ \text{subject to} \quad & \sum_{i=1}^{N} w_i a_i \le W \\ & a_i \in \mathbb{Z}_+, \quad i = 1,\dots,N \end{aligned}$$

Implement a MATLAB function `cargo_load(v,w,W)` that solves the cargo loading problem given values, weights and cargo capacity. Use the function to solve the following cargo loading problems.
Hint: Consider the following recursion:

$$V_i(x_i) = \max_{a_j = 0,1,\dots,\lfloor \frac{W}{a_i} \rfloor} \left(v_i a_j + V_{i+1}(x_i - w_i a_j)\right)$$

(a)

$$\begin{aligned} \underset{a_1,\dots,a_3}{\text{maximize}} \quad & 31a_1 + 47a_2 + 14a_3 \\ \text{subject to} \quad & 2a_1 + 3a_2 + a_3 \le 4 \\ & a_1,\dots,a_3 \in \mathbb{Z}_+ \end{aligned}$$

(b)

$$\begin{aligned} \underset{a_1,\dots,a_4}{\text{maximize}} \quad & 5a_1 + 2a_2 + 7a_3 + a_4 \\ \text{subject to} \quad & 4a_1 + 6a_2 + 2a_3 + a_4 \le 10 \\ & a_1,\dots,a_4 \in \mathbb{Z}_+ \end{aligned}$$

(c)

$$\begin{aligned} \underset{a_1,\dots,a_5}{\text{maximize}} \quad & 2a_1 + 4a_2 + 6a_3 + 3a_4 + 20a_5 \\ \text{subject to} \quad & 4a_1 + 2a_2 + 3a_3 + 2a_4 + 8a_5 \le 15 \\ & a_1,\dots,a_5 \in \mathbb{Z}_+ \end{aligned}$$

Problem 3.15 Consider the problem of fitting a function $\hat{f}$ to a set of N observations (x_i, y_i).

(a) Use the least-squares lemma to find the parameters (a,b) of the affine model

$$\hat{f}(x) = ax + b = \begin{pmatrix} a & b \end{pmatrix} \begin{pmatrix} x \\ 1 \end{pmatrix}$$

that minimize the least-squares cost

$$e = \sum_{i=1}^{N} \left(y_i - \hat{f}(x_i)\right)^2 \tag{3.12}$$

Write a program that takes (x_i, y_i) as input and returns the minimal value of the least-squares cost (3.12). Evaluate your code on $\{x_i\} = \{-10, -0, \dots, 9, 10\}$ and $y_i = \arctan(x_i)$.

(b) A better fit can be obtained by dividing the range of x_i into S segments and constructing individual affine estimates $\hat{f}_s$ in each segment s. The resulting piecewise affine (but not necessarily continuous) function will often allow a much more exact fit to the data. Assume that the data is ordered so that $x_1 \le x_2 \le \cdots \le x_N$. Use dynamic programming to construct an algorithm for finding the segmentation that minimizes the cost

$$\sum_{s=1}^{S} e_s + \lambda S$$

where e_s is the optimal least-squares cost for the data points belonging to segment s and $\lambda > 0$ is a parameter that allows you to shift the focus of the optimal solution between optimizing the total least-squares cost and finding a model with few segments. Evaluate your code for different values of λ on the same data as in (a).

Hint. If you cannot argue for the optimal solution direct using the dynamic programming principle, you can attempt to put the problem on our standard form. It is useful to identify stages with with segments, the state x_t as the number of data points covered by the t first segments, and u_t the number of points to include in the next segment.

Problem 3.16 In this problem, we will consider an idealized model of an inverted pendulum

$$\ddot{\theta}(t) = \sin(\theta(t)) - u(t)\cos(\theta(t)), \qquad |u(t)| \le u_{\max}$$

We are interested in finding the minimum energy control that takes the pendulum from rest at $\theta = \pi$ to rest at the unstable equilibrium point $\theta = 0$, and to explore how the optimal control depends on the maximal control magnitude $u_{\max}$.

(a) Argue from first principles about what you can expect the optimal control to be. Note that the total energy of the system is

$$E = \frac{1}{2}\left(\dot{\theta}(t)\right)^2 + \cos\theta(t) - 1$$

(b) Put the continuous-dynamics on state-space form with the state vector $x(t) = (\theta(t),\ \dot{\theta}(t))$, and use the Euler-forward discretization

$$\frac{dx(t)}{dt} \approx \frac{x(t+h) - x(t)}{h}$$

to find a discrete-time approximation $x_{t+1} = f(x_t, u_t)$ of the continuous-time dynamics.

(c) Use numerical dynamic programming to solve the optimal control problem

$$\begin{array}{ll} \text{minimize} & \sum_{t=0}^{T-1} u_t^2 + K x_T^\top x_T \\ \text{subject to} & x_{t+1} = f(x_t, u_t) \\ & |u_t| \le u_{\max} \end{array}$$

from $x_0 = (\pi,\ 0)$. Here the terminal cost models our desire that $x_T = \mathbf{0}$, and is more robust to numerical errors than using a hard terminal constraint. You can use the parameters $h = 0.1$, $K = 100$, $u_{\max} = 1$ and $T = 25$.

(d) Explore how the optimal solution computed in (c) depends on the discretization interval, the penalty parameter K and the horizon T. Repeat your experiments with $u_{\max} = 2$.

4. Linear-quadratic control

In this chapter, we will derive optimal control laws for linear systems with quadratic cost functions. A remarkable property of this linear-quadratic control problem is that the optimal controller is a linear state feedback. For a given state-space model of the system dynamics, the state feedback gains are completely determined by the cost function, whose parameters now become the "tuning knobs" in the controller design. In contrast to simpler techniques, such as eigenvalue assignment, the approach works equally well for scalar systems as for systems with multiple inputs and outputs.

The linear-quadratic regulator is an important control design methodology for linear systems, but it is also central to the model-predictive control strategies that we will develop later. In fact, when those strategies operate near the equilibrium, they will act as linear-quadratic regulators. It is therefore essential to understand how to tune a linear-quadratic regulator to obtain a desired system response. We provide a relatively complete treatment of important engineering aspects for linear-quadratic control, such as how to incorporate reference tracking, disturbance compensation, and integral action; how to address the dual problem of optimal state estimation; and how to combine the estimator with feedback from estimated states into an output feedback control strategy.

Finally, it is important to pay attention to the theory of linear-quadratic control. Not only is it an elegant and insightful theory, but our treatment of model-predictive control will be based on concepts that are more easily understood in the unconstrained case.

4.1 Finite-horizon linear-quadratic optimal control

The *finite-horizon linear quadratic control problem* considers a linear system

$$x_{t+1} = Ax_t + Bu_t \tag{4.1}$$

and aims at finding the control sequence $\{u_0, u_1, \dots, u_{T-1}\}$ that minimizes the quadratic cost

$$\sum_{t=0}^{T-1} (x_t^\top Q x_t + u_t^\top R u_t) + x_T^\top Q_T x_T$$

for given weight matrices $Q \succeq 0$, $R \succ 0$ and $Q_T \succeq 0$. The stage cost captures the trade-off between making the state vector converge quickly to zero and using control inputs with small energy. In particular, choosing a Q that is large relative to R makes state deviations more costly, and leads to an optimal controller that steers the states quickly to zero. Conversely, increasing the size of R shifts the focus of the stage cost to the control signal, leading to a lower energy input and state trajectories that tend to be closer to the open loop system's natural response. The terminal cost matrix Q_T plays a more subtle role, but larger values of Q_T will reinforce a desire to drive the terminal state to rest at the origin. We will discuss the controller tuning in more detail once we have derived the optimal controller for both finite and infinite-horizon problems.

For a fixed and finite value of T, the optimal control law can be computed using the dynamic programming approach described in Chapter 3, leading to the following result.

Theorem 4.1.1 The finite-horizon linear-quadratic optimal control problem

$$\begin{array}{ll} \text{minimize} & \sum_{t=0}^{T-1} x_t^\top Q x_t + u_t^\top R u_t + x_T^\top Q_T x_T \\ \text{subject to} & x_{t+1} = Ax_t + Bu_t, \qquad t = 0, 1, \ldots, T-1 \end{array}$$

with $Q \succeq 0$, $R \succ 0$ and $Q_T \succeq 0$ has the optimal solution $u_t = -L_t x_t$ where

$$L_t = (B^\top P_{t+1} B + R)^{-1} B^\top P_{t+1} A \tag{4.2}$$

and P_t satisfies the Riccati recursion

$$P_t = Q + A^\top P_{t+1} A - A^\top P_{t+1} B (B^\top P_{t+1} B + R)^{-1} B^\top P_{t+1} A \tag{4.3}$$

with boundary condition $P_T = Q_T$. The minimal value of the objective function is $x_0^\top P_0 x_0$.

Proof. We will derive the optimal control law using dynamic programming. To this end, we use induction to show that the cost-to-go function is quadratic

$$v_t(x_t) = x_t^\top P_t x_t.$$

First, we note that the induction hypothesis is satisfied for $t = T$ with $P_T = Q_T$. Next, assuming that the induction hypothesis holds for stage $t+1$, the dynamic programming recursion (3.8) gives

$$\begin{aligned} v_t(x_t) &= \min_{u_t} \left\{ x_t^\top Q x_t + u_t^\top R u_t + v_{t+1}(Ax_t + Bu_t) \right\} = \\ &= \min_{u_t} \left\{ x_t^\top Q x_t + u_t^\top R u_t + (Ax_t + Bu_t)^\top P_{t+1} (Ax_t + Bu_t) \right\} = \\ &= \min_{u_t} \left\{ u_t^\top (B^\top P_{t+1} B + R) u_t + 2u_t^\top B^\top P_{t+1} A x_t + x_t^\top (A^\top P_{t+1} A + Q) x_t \right\} \end{aligned}$$

Since $B^\top P_{t+1} B + R$ is positive definite, v_t is a convex quadratic function in u_t and Lemma 3.1 yields

$$u_t^\star(x_t) = -(B^\top P_{t+1} B + R)^{-1} B^\top P_{t+1} A x_t := -L_t x_t.$$

The associated cost-to-go is

$$v_t(x_t) = x_t^\top \left(Q + A^\top P_{t+1} A - A^\top P_{t+1} B (B^\top P_{t+1} B + R)^{-1} B^\top P_{t+1} A \right) x_t := x_t^\top P_t x_t$$

Since both stage costs and terminal costs are non-negative, the cost-to-go must also be non-negative and P_t must be a positive semidefinite matrix. Hence, by induction, the cost-to-go remains quadratic and positive semidefinite for $t = T, T-1, \ldots, 1, 0$. Since the expressions for L_t and P_t derived above agree with (4.2) and (4.3) for all t, the proof is complete. ■

The next example demonstrates the use of Theorem 4.1.1.

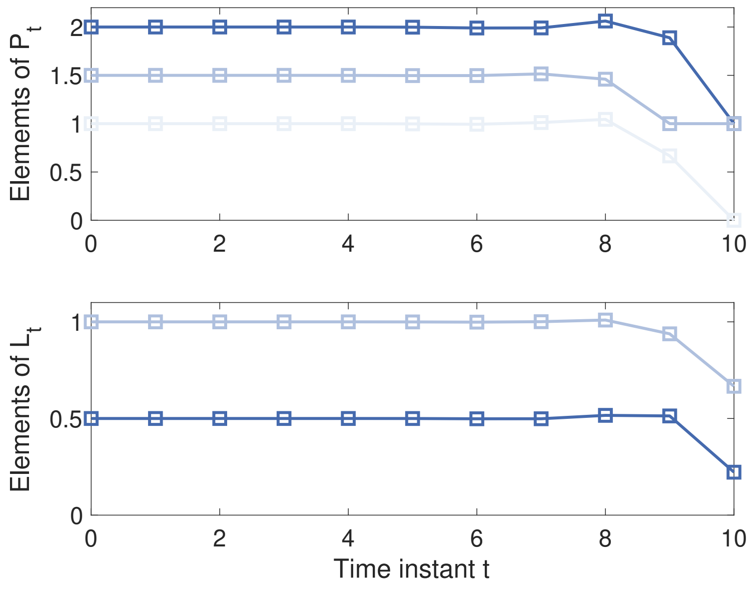


Figure 4.1: Elements of the iterates of the Riccati recursion (top) and the associated optimal feedback gains (bottom).

■ **Example 4.1** Consider linear-quadratic optimal control of the vertical quadrotor dynamics with a horizon of $T = 10$ samples and cost

$$x_T^\top x_T + \sum_{t=0}^{T-1} x_t^\top x_t + u_t^\top u_t$$

The problem is on our standard form with $Q = I$, $R = 1$ and $Q_T = I$. By Theorem 4.1.1, the optimal control is a time-varying state feedback, whose gains can be computed from the iterates P_t in the Riccati recursion (4.3) with boundary value $P_T = Q_T$. Figure 4.1 (top) shows the elements of the matrices P_t. These matrices are symmetric, so there are only three curves (two for the diagonal elements and one for the two identical off-diagonal elements). Recall that the Riccati recursion proceeds backward from $P_T = Q_T = I$ at stage T. The elements of P_t change significantly during the iterations $T, T-1, \ldots$ close to the end of the horizon, but converge quickly towards stationary values. The bottom plot visualizes how the associated optimal feedback gains stay constant until close to the end of the horizon, when the terminal cost becomes increasingly important. ■

Figure 4.1 indicates that the Riccati recursion (4.3) has two distinct regimes. When $t \ll T$, the matrices $\{P_t\}$ are almost constant. In this regime, there is enough time for the long-term optimal control to drive x_T close to zero and the impact of the terminal cost is negligible. When t is close to T, on the other hand, the matrices $\{P_t\}$ change significantly. Here, there is no longer enough time for the long-term optimal control to drive the terminal state close to zero, and the control actions have to be adapted to steer x_T to a state where the terminal cost is small.

Loss functions with cross-terms between control and state penalties*

It is sometimes useful to allow for cross-terms between the control and the state, *i.e.* to consider linear-quadratic control problems with stage costs on the form

$$g_t(x,u) = \begin{pmatrix} x \\ u \end{pmatrix}^\top \begin{pmatrix} Q & N \\ N^\top & R \end{pmatrix} \begin{pmatrix} x \\ u \end{pmatrix} = x^\top Q x + 2x^\top N u + u^\top R u$$

If $R \succ 0$, the completion-of-squares lemma guarantees that the stage cost is positive if

$$\min_u g_t(x,u) = x^\top Qx - x^\top NR^{-1}N^\top x = x^\top (Q - NR^{-1}N^\top)x \geq 0 \quad \text{for all } x$$

i.e., if $Q - NR^{-1}N^\top \succeq 0$. Moreover, we can then re-write the stage costs as

$$g_t(x,u) = (u-u^\star)^\top R(u-u^\star) + x^\top Qx - (u^\star)^\top Ru^\star := \tilde{u}^\top R\tilde{u} + x^\top \tilde{Q}x$$

where $\tilde{u} = u - u^\star = u + R^{-1}N^\top x$ and $\tilde{Q} = Q - NR^{-1}N^\top$.

These manipulations show that if $R \succ 0$ and $Q - NR^{-1}N^\top \succeq 0$, then we can solve the LQR problem with cross-terms for our original system by solving a standard LQR problem with cost

$$\tilde{J} = \sum_{t=0}^{T-1} x_t^\top \tilde{Q}x_t + \tilde{u}_t^\top R\tilde{u}_t + x_T^\top Q_T x_T \tag{4.4}$$

for a related system with state vector x_t and input $\tilde{u}_t$. The state-space equations for this system are

$$x_{t+1} = Ax_t + Bu_t = Ax_t + B(u_t - \tilde{u}_t) + B\tilde{u}_t = (A - R^{-1}N^\top)x_t + B\tilde{u}_t := \tilde{A}x_t + B\tilde{u}_t \tag{4.5}$$

The optimal control under modified dynamics (4.5) and modified cost (4.4) is on the form

$$\tilde{u}_t^\star = -\tilde{L}_t x_t.$$

By the relationship between $\tilde{u}_t$ and u_t, the optimal control for the original system is therefore

$$u_t^\star = -(\tilde{L}_t + R^{-1}N^\top)x_t.$$

4.2 Infinite-horizon linear-quadratic control: optimality and stability

It is natural to ask if the solution presented in Theorem 4.1.1 remains valid when we are interested in the behavior of the closed-loop system over an infinite horizon. To this end, we consider the *infinite-horizon linear-quadratic regulator (LQR)* problem

$$\begin{array}{ll} \text{minimize} & \sum_{t=0}^{\infty} x_t^\top Qx_t + u_t^\top Ru_t \\ \text{subject to} & x_{t+1} = Ax_t + Bu_t \end{array} \tag{4.6}$$

with $Q \succeq 0$ and $R \succ 0$. Since this problem has an infinite number of stages, there is no terminal time, and therefore no terminal cost. The word regulator is used to emphasize that the problem focuses on driving the system to rest at the origin. Since the stage costs are positive, we can use the infinite-horizon dynamic programming results in Theorem 3.3.2 to derive the optimal solution. The proofs are simplified using the following result, proven in Appendix D.3.

Lemma 4.1 The value function

$$v(x) = \min_{u_0,u_1,\dots} \left\{ \sum_{t=0}^{\infty} x_t^\top Qx_t + u_t^\top Ru_t \mid x_{t+1} = Ax_t + Bu_t,\ x_0 = x \right\}$$

of the infinite-horizon linear-quadratic regulation problem (4.6) is quadratic, *i.e.* $v(x) = x^\top Px$ for some positive semidefinite matrix P.

Combining this lemma with Theorem 3.3.2, we conclude that the value function must satisfy the following Bellman equation

$$\begin{aligned} x^\top Px &= \min_u \left\{ x^\top Qx + u^\top Ru + (Ax+Bu)^\top P(Ax+Bu) \right\} = \\ &= \min_u \left\{ x^\top (Q + A^\top PA)x + 2x^\top A^\top PBu + u^\top (R+B^\top PB)u \right\} = \\ &= x^\top \left(Q + A^\top PA - A^\top PB(R+B^\top PB)^{-1}B^\top PA \right) x \end{aligned} \tag{4.7}$$

where the last step follows from the completion-of-squares lemma (Lemma 3.1). Since this relationship holds for all x, P must satisfy the *discrete-time algebraic Riccati equation (DARE)*

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

We also note that the minimizing u in the Bellman equation is $u = -Lx$ where

$$L = (R+B^\top PB)^{-1}B^\top PA.$$

This controller is known as the *linear-quadratic regulator*. We have summarize the developments in the following proposition.

Proposition 4.2.1 Consider the infinite-horizon linear quadratic regulator problem

$$\begin{array}{ll} \text{minimize} & \sum_{t=0}^{\infty} x_t^\top Qx_t + u_t^\top Ru_t \\ \text{subject to} & x_{t+1} = Ax_t + Bu_t \end{array} \tag{4.8}$$

with $Q \succeq 0$ and $R \succ 0$. If the discrete-time algebraic Riccati equation (DARE)

$$P = Q + A^\top PA - A^\top PB(R+B^\top PB)^{-1}B^\top PA. \tag{4.9}$$

admits a positive semi-definite solution P, then the optimal control is $u_t = -Lx_t$ where

$$L = (R+B^\top PB)^{-1}B^\top PA \tag{4.10}$$

The minimal infinite-horizon cost from the initial state x_0 is $x_0^\top Px_0$.

Note that the discrete-time algebraic Riccati equation (4.9) characterizes the fixed-points $P_{t+1} = P_t = P$ of the Riccati recursion (4.3). The DARE is a *quadratic* matrix equation in P, typically solved numerically using specialized solvers. We illustrate these points with two examples.

■ **Example 4.2** Let us compute the linear-quadratic regulator for the quadcopter under the cost

$$\sum_{t=0}^{\infty} x_t^\top x_t + u_t^\top u_t.$$

To do so, we need to find a positive semi-definite solution to the DARE defined by the matrices

$$A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \qquad B = \begin{pmatrix} 0.5 \\ 1 \end{pmatrix}, \qquad Q = I, \qquad R = 1$$

It is easy to verify numerically that

$$P = \begin{pmatrix} 2 & 1 \\ 1 & 1.5 \end{pmatrix}$$

is such a solution. By comparing the elements of P with the elements of P_t for $t = 0$ in Figure 4.1, we note that this is indeed the stationary solution to the Riccati recursion in Example 4.1. ■

Like the Lyapunov equation in Chapter 2, it is sometimes possible to solve the algebraic Riccati equation analytically. However, since the equation is quadratic in P, it is much more difficult and rarely results in simple expressions. The next example illustrates the process on a scalar problem.

■ **Example 4.3** Let us compute the linear-quadratic regulator for the scalar system

$$x_{t+1} = \sqrt{\frac{3}{2}} x_t + u_t$$

under the cost

$$\sum_{t=0}^{\infty} x_t^2 + u_t^2.$$

We then need to solve the DARE defined by $A = \sqrt{3/2}$, $B = 1$, $Q = 1$ and $R = 1$, *i.e*

$$P = 1 + A^2 P - A^2 \frac{P^2}{1+P}.$$

It is convenient to re-write this equation as

$$P^2 - A^2 P - 1 = 0$$

Since P must be positive, the desired Riccati solution is

$$P = \frac{1}{2}A^2 + \frac{1}{2}\sqrt{A^4+4} = \frac{3}{4} + \frac{5}{4} = 2$$

and the optimal feedback gain is

$$L = a \cdot \frac{P}{1+P} = \sqrt{\frac{3}{2}} \cdot \frac{2}{3} = \sqrt{\frac{2}{3}}$$

Finally, we can verify that the closed-loop system

$$x_{t+1} = Ax_t + u_t = (A-L)x_t = \frac{\sqrt{6}}{6} x_t$$

is asymptotically stable. ■

Even if the optimal controllers computed in the examples above result in asymptotically stable closed-loop systems, this is not guaranteed by Proposition 4.2.3. Such a guarantee requires a few additional assumptions. To develop an intuition for the subtleties at play, let us first consider

$$x_{t+1} = 2x_t + 0 \cdot u_t.$$

Clearly, the control is unable to affect the state and stop it from growing exponentially. Hence, with a stage cost on the form $g(x,u) = Qx^2 + Ru^2$, the infinite-horizon LQ cost will be unbounded, no matter how we choose the weight $Q > 0$. The corresponding DARE becomes $P = Q + 4P$, which does not admit any positive solution. A similar phenomenon appears in the unreachable system

$$x_{t+1} = \begin{pmatrix} 1/2 & 1 \\ 0 & 2 \end{pmatrix} x_t + \begin{pmatrix} 1 \\ 0 \end{pmatrix} u_t \tag{4.11}$$

The second state is unstable and cannot be affected by the control. Hence, the infinite-horizon LQ-cost will be unbounded if this state appears in the stage cost.

If (A,B) is a reachable pair, on the other hand, then we know that there exists a control sequence $\{u_0,u_1,\dots,u_{n-1}\}$ that drives any initial state to the origin in n steps. Once the state has reached the origin, we can apply $u_t \equiv 0$ for $t \geq n$ to ensure that the state vector remains at the origin. This control sequence, which was selected without considering any optimality criterion, has a finite LQ-cost. Therefore, the *optimal* cost must also be finite. A similar argument holds for stabilizable systems: we can drive the reachable states to zero in finite time, and then let the unreachable states (which are then asymptotically stable) converge at their natural pace. Also this cost will be finite.

A bounded infinite-horizon cost does not by itself guarantee asymptotic stability of the closed loop. For example, the system (4.11) under the control policy $u_t \equiv 0$ has cost

$$\sum_{t=0}^{\infty} x_t^\top \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} x_t + u_t^\top R u_t = x_0^\top \begin{pmatrix} 4/3 & 0 \\ 0 & 0 \end{pmatrix} x_0,$$

which is finite for every finite x_0. We know that the system is unstable, since the second system state grows exponentially, but the stage cost does not account for this state. To make sure that a bounded cost also implies asymptotic stability of the closed loop, unstable states must either appear directly in the cost, or be observable in the cost through their influence on other states.

To perform a formal stability analysis of the closed-loop system under LQ-optimal control, we insert the optimal $u = -Lx$ in the Bellman equation (4.7). Since this u is optimal, it holds that

$$x^\top P x = x^\top \left(Q + L^\top R L + (A-BL)^\top P (A-BL) \right) x \qquad \forall x \in \mathbb{R}^n.$$

This, in turn, implies the following matrix identity must be satisfied

$$P = Q + L^\top R L + (A-BL)^\top P(A-BL)$$

We can re-write this equation as a Lyapunov equation in the closed-loop system matrix $A-BL$:

$$(A-BL)^\top P(A-BL) - P + Q + L^\top R L = 0. \tag{4.12}$$

Since the quantity $Q+L^\top RL$ can only be guaranteed to be positive semi-definite, we need to use the less restrictive Theorem 2.2.6 to assess closed-loop stability. To guarantee asymptotic stability, the theorem requires that $(A-BL,(Q+L^\top RL)^{1/2})$ is detectable. The next result shows that this detectability condition always holds when R is positive definite and $(A,Q^{1/2})$ detectable.

Proposition 4.2.2 Let $R \in \mathbb{R}^{m\times m}$ be positive definite and $(A,Q^{1/2})$ detectable. Then $(A-BL,(Q+L^\top RL)^{1/2})$ is also detectable for any $L \in \mathbb{R}^{m\times n}$.

Proof. Assume that $(A-BL,(Q+L^\top RL)^{1/2})$ is not detectable. Then, by Theorem 1.3.4, there exist $v \neq \mathbf{0}$ and λ with $|\lambda| > 1$ such that

$$(A-BL)v = \lambda v \qquad\qquad (Q+L^\top RL)^{1/2} v = 0$$

The second equality implies that

$$v^*(Q+L^\top RL)v = \|Q^{1/2}v\|_2^2 + \|R^{1/2}Lv\|_2^2 = 0,$$

i.e. that $Q^{1/2}v = 0$ and $R^{1/2}Lv = 0$. Since R is positive definite, this also means that $Lv = 0$. Thus,

$$(A-BL)v = Av = \lambda v, \qquad Q^{1/2}v = 0.$$

Since $(A,Q^{1/2})$ is detectable, any solution to these equations must have $|\lambda| < 1$, which is a contradiction. Hence, $(A-BL,(Q+L^\top RL)^{1/2}$ must be detectable. □

We can now summarize our results to a more complete theorem for infinite-horizon LQR.

Theorem 4.2.3 Consider the infinite-horizon linear-quadratic regulator problem

$$\begin{array}{ll} \text{minimize} & \sum_{t=0}^{\infty} x_t^\top Q x_t + u_t^\top R u_t \\ \text{subject to} & x_{t+1} = Ax_t + Bu_t \end{array}$$

with $Q \succeq 0$ and $R \succ 0$. If (A,B) is stabilizable, then the optimal cost is bounded. If, in addition, $(A, Q^{1/2})$ is detectable, then the DARE

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

admits a unique positive semidefinite solution, and the optimal control policy $u_t = -Lx_t$ with

$$L = (R + B^\top PB)^{-1}B^\top PA$$

renders the closed-loop system asymptotically stable.

■ **Example 4.4** Let us return to the vertical drone dynamics studied in earlier examples. We have already established that this system is controllable (and thus stabilizable). However, Theorem 4.2.3 also requires that $(A, Q^{1/2})$ should be detectable. Let us first only penalize the control action and the position of the drone, i.e., use

$$Q = \begin{pmatrix} 1 & 0 \end{pmatrix}^\top \begin{pmatrix} 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \quad \text{and} \quad R = 1$$

It is easy to verify that $Q^{1/2} = Q$ and that $(A, Q^{1/2})$ is observable. Solving the algebraic Riccati equation yields the optimal state feedback gains $L = \begin{pmatrix} 1/2 & 1 \end{pmatrix}$. The closed-loop system matrix has eigenvalues with magnitude $1/2$, and is therefore Schur stable.

If we instead of the position penalize the velocity, i.e. let

$$Q = \begin{pmatrix} 0 & 1 \end{pmatrix}^\top \begin{pmatrix} 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \quad \text{and} \quad R = 1$$

we also have $Q^{1/2} = Q$, but $(A, Q^{1/2})$ is not detectable (cf. Example 1.5). Standard DARE solvers are typically unable to solve the corresponding Riccati equation and return an error message. Nevertheless, it is easy to verify that

$$P = \begin{pmatrix} 0 & 0 \\ 0 & (1+\sqrt{5})/2 \end{pmatrix}$$

is a solution to the algebraic Riccati equation. The corresponding optimal control law, however, only drives the velocity (the second state) to zero, so the closed loop is not asymptotically stable. ■

A few words about the connection between Lyapunov and Riccati equations

Our analysis of the linear-quadratic regulator involves both Lyapunov and Riccati equations. Since these are new to many readers, it is useful to reflect on what they represent and how they were used.

The Lyapunov equation characterizes the set of admissible quadratic Lyapunov functions for a given linear system, while the Riccati equation determines the value function for a specific infinite-horizon linear-quadratic control problem. Like the Lyapunov functions derived from the Lyapunov equation, the LQR value function is also a quadratic function of the system state.

Theorem 4.2.3 relies on the observation that the value function for the infinite-horizon LQR problem also works as a Lyapunov function for the corresponding closed-loop system. This insight emerges by re-writing the Riccati equation as a Lyapunov equation for the closed-loop system under the LQ-optimal control law. Since the Riccati solution is positive semi-definite and $(A, Q^{1/2})$ is detectable, we can use Theorem 2.2.6 to conclude asymptotic stability.

Conversely, if we pick another feedback gain, say L', with $A - BL'$ Schur stable, Theorem 2.2.6 guarantees that the Lyapunov equation (4.12) admits a positive semidefinite solution P'. Moreover,

$$\sum_{t=0}^{\infty} x_t^\top Q x_t + u_t^\top R u_t = \sum_{t=0}^{\infty} x_t^\top \left(Q + (L')^\top R L' \right) x_t \leq x_0^\top P' x_0.$$

But since L' is not optimal, $x_0^\top P' x_0 \geq x_0^\top P x_0$ for all x_0. In other words, while the Lyapunov equation can be used to bound the quadratic cost of a given state feedback, the solution to the Riccati equation defines the value function – the minimum infinite-horizon cost achievable by any controller.

4.3 Tuning of the LQ-optimal control law

The LQ cost function attempts to strike a balance between the transient response and control effort. This trade-off is most obvious when we our system has a scalar control signal and a scalar output $y_t = Cx_t$, and we consider the cost defined by $Q = C^\top C$ and $R = \rho I$, i.e.

$$\sum_{t=0}^{\infty} x_t^\top Q x_t + u^\top R u_t = \sum_{t=0}^{\infty} x_t^\top C^\top C x_t + \rho u_t^\top u_t = \sum_{t=0}^{\infty} y_t^2 + \rho u_t^2 \tag{4.13}$$

A large value of ρ implies that it is costly to use the control signal, and we could expect the optimal controller to be gentle and the closed-loop dynamics to be similar to that of the uncontrolled plant. If we let ρ tend to zero, on the other hand, the cost function will only focus on deviations in the output and we could expect the optimal controller to be more aggressive and drive y_t to zero very quickly. The next example demonstrates this trade-off.

■ **Example 4.5** Let us study the vertical quadrotor dynamics from earlier examples and consider its position (the first state) as output signal. Figure 4.2 shows the optimal output and control signals from the same initial value but for three different values of ρ. As expected, a small value of ρ gives a fast response but large control actions, while a larger value of ρ results in restrictive use of input energy at the expense of a slower output response. ■

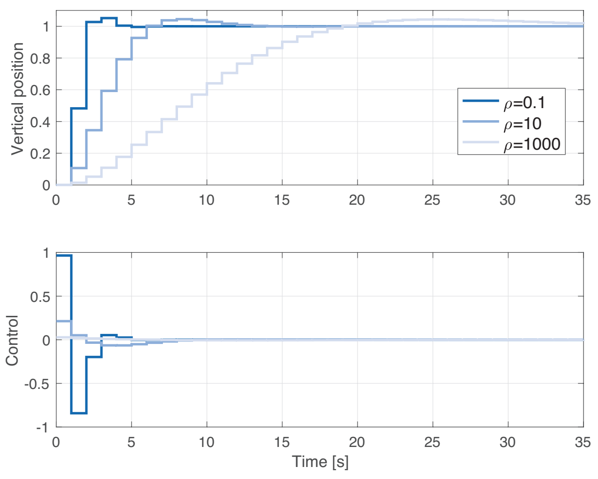


Figure 4.2: Initial value responses of LQR-controller for three different values of the ρ parameter.

Of course, the linear quadratic framework is much more expressive than this, especially for systems with many inputs and many outputs. However, the large number of tuning parameters can be challenging to manage when tuning the controller. Since Q and R are symmetric $n \times n$ and $m \times m$ matrices, the cost function itself has $n(n+1)/2 + m(m+1)/2$ free parameters. It can be difficult to obtain good results without a structured approach to select all these free parameters. Below, we outline a tuning procedure where the degrees of freedom are kept low and all tuning parameters play a clear role in shaping the controller response.

1. Define performance outputs to limit the degrees of freedom

A first step we will define performance outputs $z_t = Mx_t \in \mathbb{R}^{n_z}$ and consider the cost

$$\sum_{t=0}^{\infty} z_t^\top \bar{Q} z_t + u_t \bar{R} u_t = \sum_{t=0}^{\infty} x_t^\top M^\top \bar{Q} M x_t + u_t^\top \bar{R} u_t$$

where $\bar{Q}$ and $\bar{R}$ are diagonal matrices. This is then a standard LQ cost defined by $Q = M^\top \bar{Q} M$ and $R = \bar{R}$, with a single weight associated with each performance output and with each control signal. This parameterization is particularly useful when the number of performance outputs is much smaller than the system state dimension ($n_z \ll n$). If (A, C) is detectable, then it is common to begin with $M = C$, to only penalize the true system outputs, and to let $\bar{Q} = I$ and $\bar{R} = I$.

2. Normalize weights using Bryson's rule

It is often good idea to normalize the variables in any system model to have the same ranges. Even so, we may have very different acceptance levels for deviations in the different signals. *Bryson's rule* suggests the following initial weight tuning to give comparable deviations in different signals a comparable impact on the total linear-quadratic cost:

$$[\bar{Q}]_{ii} = \frac{1}{(z_{\max}^{(i)})^2}, \qquad [\bar{R}]_{ii} = \frac{1}{(u_{\max}^{(i)})^2}.$$

Here, $z_{\max}^{(i)}$ is the maximal transient error that we can accept in performance output $z^{(i)}$, and $u_{\max}^{(i)}$ is the magnitude of control signal i which we consider to be of equal harm as a transient error of $z_{\max}^{(i)}$.

We can already in this step start to evaluate the optimal controller in simulations. It is not uncommon that Bryson's rule gives a decent design without further tuning.

3. Tune the closed-loop bandwidth of the LQR controller

The closed-loop bandwidth is a key parameter in many control systems. We have already seen that a single parameter for adjusting the relative weighting between control cost and state errors is a good way to affect the closed-loop bandwidth. We thus propose to set

$$R = \rho \bar{R}$$

and adjust ρ to get the desired closed-loop bandwidth.

4. Fine-tune performance weights

If we still do not get the right balance between different performance outputs and between individual control signals, then we have to departure from Bryson's rule and add further degrees of freedom. We propose to simply use

$$\bar{Q} = \begin{pmatrix} \left(\frac{q_1}{z_{\max}^{(1)}}\right)^2 & 0 & \dots & 0 \\ 0 & \left(\frac{q_2}{z_{\max}^{(2)}}\right)^2 & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \dots & 0 & \left(\frac{q_{n_z}}{z_{\max}^{(n_z)}}\right)^2 \end{pmatrix}, \quad \bar{R} = \begin{pmatrix} \left(\frac{r_1}{u_{\max}^{(1)}}\right)^2 & 0 & \dots & 0 \\ 0 & \left(\frac{r_2}{u_{\max}^{(2)}}\right)^2 & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \dots & 0 & \left(\frac{r_m}{u_{\max}^{(m)}}\right)^2 \end{pmatrix}.$$

Initialized from $q_1 = q_2 = \cdots = q_{n_z} = 1$ and $r_1 = \cdots = r_m = \rho$. Avoid to adjust many parameters at the same time, and remember that it is the relative weighting between the terms that matters.

5. In necessary, introduce additional performance signals

If simulations reveal that certain internal states (or combinations of internal states) make large deviations and exceed desired operating ranges, one needs to introduce additional an performance outputs $z^{(i)}$ that capture these signals, and include them in the cost function. One then augments the M matrix with new rows and return to the previous steps to tune the new entry of the $\bar{Q}$ matrix.

The new performance outputs are often just states that were neglected before, but one can also attempt more advanced tricks. For example, if we would like the i^{th} system state to evolve as

$$[x_{t+1}]_i = a_{\text{des}}^\top x_t$$

we can add the performance output

$$z_t = [x_{t+1}]_i - a_{\text{des}}^\top x_t$$

However, since $[x_{t+1}]_i$ in general depends on both x_t and u_t, cross-terms between x_t and u_t may appear in the cost function when we form the square of the performance output.

If no new performance outputs are needed, one proceeds to the final step of the tuning procedure.

6. Evaluate, analyze and iterate.

While the linear-quadratic regulator is optimal, it is optimal in a specific sense: it optimizes a quadratic cost function (4.6) based on given weight matrices Q and R. If this cost captures all the closed-loop properties that we are interested in, then this is the best that we can do. But in many cases, we have a diverse set of requirements on our controller, and not all of them can be expressed as a quadratic cost on states and controls.

At this stage, it is therefore essential to examine the transfer functions from all inputs and disturbances to the control signal and system outputs. It is not uncommon to realize that previous design choices must be revised, and it may take several iterations to find the right tuning parameters.

The next example illustrates the proposed LQR tuning procedure on a simple inverted pendulum.

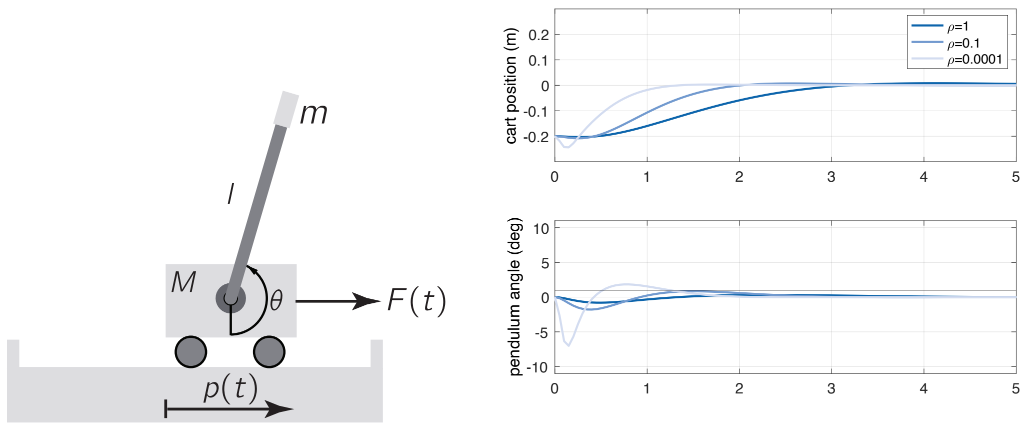


Figure 4.3: Inverted pendulum (left), and time responses for different weight ρ.

■ **Example 4.6** Consider the inverted pendulum system shown in Figure 4.3 (left). A pendulum is attached to a cart which we can accelerate by manipulating the external force F. A linearized model of the system dynamics around the upright equilibrium of the pendulum, with a sampling time of 0.05 seconds, is given by the following state-space model

$$x_{t+1} = \begin{pmatrix} 1 & 0.0498 & 0.0034 & 0.0001 \\ 0 & 0.9909 & 0.1348 & 0.0034 \\ 0 & -0.0006 & 1.0392 & 0.0507 \\ 0 & -0.0229 & 1.5779 & 1.0392 \end{pmatrix} x_t + \begin{pmatrix} 0.0023 \\ 0.0908 \\ 0.0057 \\ 0.2292 \end{pmatrix} u_t$$

The states represent the position and velocity of the cart, followed by the angle and angular velocity of the pendulum. Our aim is to design a controller that can move the cart 0.2 meters with a settling time of 1 second, without the pendulum angle ever exceeding 10 degrees ($= \pi/18$ radians) from its upright position. The force is limited to ± 10 N.

Following the proposed tuning procedure, we introduce the performance output $z_t = Mx_t$ with

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

to account for the cart position and the pendulum angle. Since the maximal values of the cart position and pendulum angle are comparable in magnitude (0.2 meters and $\pi/18 \approx 0,1745$ radians), we skip the normalization step and simulate the closed loop using $Q = M^T M$ and $R = \rho$ for different values of ρ. As we can see, we have to decrease ρ to 0.0001 to obtain the desired settling time. The maximum control magnitude during the cart movement is around 8 N, so there is still some room for speeding up the step response even further. We do so by fine-tuning the weight on the cart position from the default value of 1 to 1.6. Thus, our final design is given by the weight matrices

$$Q = M^\top \begin{pmatrix} 1.6 & 0 \\ 0 & 1 \end{pmatrix} M, \qquad R = 0.0001I$$

■

Next, we will develop some additional insight into properties of the linear-quadratic optimal controllers that can be useful for tuning.

The linear-quadratic regulator at the extremes: cheap and expensive control

Closed-loop systems under linear-quadratic optimal control have a very specific behavior when the control signal is either very cheap or very expensive. To develop some intuition for these properties, we return to the scalar system that we studied earlier.

■ **Example 4.7** Consider the LQ problem for $x_{t+1} = ax_t + u_t$ with a cost defined by $Q = 1$ and $R = \rho$. Following the same steps as in Example 4.3, we find that the corresponding DARE can be re-written as

$$P^2 - \phi(\rho)^2 P - 1 = 0 \quad \text{where} \quad \phi(\rho) = (1 + \rho(a^2 - 1))$$

The positive solution to this equation is

$$P = \frac{1}{2}\phi(\rho) + \frac{1}{2}\sqrt{\phi^2(\rho) + 4\rho}$$

and that the optimal feedback gain is

$$L = \frac{aP}{\rho + P}.$$

Note that these expressions agree with those in Example 4.3 when $\rho = 1$. Although it is difficult to get simple expressions for P and L for particular values of ρ, it is possible to study these expressions when the control effort becomes either very cheap or very expensive.

When $\rho \to 0$ ("control is cheap"), then $\phi(\rho) \to 1$, $P \to 1$, $L \to a$ and $x_{t+1} = (a-L)x_t \to 0 \cdot x_t$. Hence, the closed-loop pole tends to the origin and the state converges to zero in a single time step.

As $\rho \to \infty$ ("control is expensive"), on the other hand,

$$\frac{P}{\rho} \to \frac{a^2-1}{2} + \left|\frac{a^2-1}{2}\right|$$

and the closed-loop dynamics tends to

$$x_{t+1} = \begin{cases} ax_t & \text{if } |a| \leq 1 \\ a^{-1}x_t & \text{otherwise .} \end{cases}$$

Thus, if the open-loop is stable, it is optimal to not apply any control. If the system is unstable, on the other hand, we need to stabilize it and it is optimal to place the closed-loop pole in $1/a$. ■

It turns out that the properties for the cheap and expensive control scenarios that we have discovered in the scalar case also hold for linear systems with arbitrary state, input and output dimension. The result for multiple-input multiple-output systems require some concepts that are outside the scope of these notes, but we can state the results for single-input single-output systems.

Theorem 4.3.1 Consider the linear system

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t \\ z_t &= Mx_t \end{aligned}$$

with $x_t \in \mathbb{R}^n$, $u_t \in \mathbb{R}$ and $z_t \in \mathbb{R}$. Let (A,B) be reachable, (A,M) be observable and consider the LQ cost with $Q = M^\top M$ and $R = \rho I$, *i.e.*

$$J = \sum_{t=0}^{\infty} z_t^2 + \rho u_t^2$$

Assume that the open loop system from u_t to y_t has q zeros located at $z_1, \dots, z_q$, p poles at the origin and $n-p$ poles located at p_i. Then,

(a) as $\rho \to \infty$, p closed-loop poles remain at zero and the others tend to

$$\pi = \begin{cases} p_i & \text{if } |p_i| \leq 1 \\ p_i^{-1} & \text{if } |p_1| \geq 1 \end{cases}$$

(b) as $\rho \to 0$, $n-q$ closed-loop poles tend to zero, and the remaining ones to

$$\pi = \begin{cases} z_i & \text{if } |z_i| \leq 1 \\ z_i^{-1} & \text{if } |z_i| \geq 1 \end{cases}$$

The proof of this theorem, along with generalizations to the multiple-input multiple-output case can be found in [23]. The next few examples illustrate the use of the theorem.

■ **Example 4.8** Let us first study the quadrotor dynamics. As shown in Example 1.10, this system has a double pole at $z = 1$, and a (sampling) zero at $z = -1$. Hence, when the control is very expensive, minimal control will be applied to just ensure that the two poles are strictly inside the unit circle. As control becomes increasingly cheap, the two poles move towards the origin and the open-loop zero; see Figure 4.4. ■

The previous example highlights the dangers with pushing a single criterion to the extreme. As control cost ρ tends to zero, the optimal control policy tends to $u_t = -\begin{pmatrix} 2 & 2 \end{pmatrix} x_t$. This controller

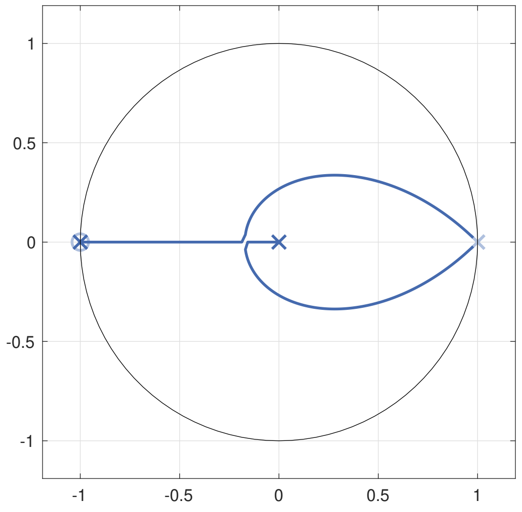


Figure 4.4: Open-loop poles (light crosses) and zeros (light rings). Dark crosses represent closed-loop poles under cheap control, and the dark line shows how the closed-loop poles move when control goes from expensive to cheap.

is able to bring the position to zero in a single time step, and then keep it there. However, the corresponding velocity (the second state) oscillates between $\pm v_0$. Hence, although the controller attains the minimal LQ cost, it is not a very attractive controller in other respects.

■ **Example 4.9** As another example, consider the inverted pendulum dynamics

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

This system has continuous-time poles at $s = \pm 1$. After zero-order-hold sampling with period $h = 0.1$, the corresponding discrete-time system has poles in $e^{\pm h}$ and a zero at -1. Since $e^h > 1$, the system is unstable, and we know that the expensive control solution is to move this pole to $1/e^h = e^{-h}$ and leave the stable pole unaltered. Hence, expensive LQR control will yield a double pole at $z = e^{-h}$. As the control action becomes increasingly cheap, the closed-loop poles move towards the origin and the system zero, creating an increasingly fast system, see Figure 4.5. ■

An optimal trade-off*

The trade-off that we have just explored is, in a certain sense, optimal. Specifally, consider the following energy-constrained optimal control problem

$$\begin{array}{ll} \text{minimize} & \sum_t z_t^2 \\ \text{subject to} & \sum_t u_t^2 \leq e_{\max} \\ & x_{t+1} = Ax_t + Bu_t, \quad z_t = Mx_t \end{array}$$

By introducing a Lagrange multiplier λ for the energy constraint (cf. Appendix C), the associated dual problem is to maximize the dual function

$$g(\lambda) = \inf_{\{u_t\}} \left\{ \sum_t x_t^\top M^\top M x_t + \lambda \sum_t u_t^2 \mid x_{t+1} = Ax_t + Bu_t \right\} - \lambda e_{\max}$$

We recognize the first term as the optimal LQ cost for $Q = M^\top M$ and $R = \lambda I$. If strong duality holds (which it does if the initial value is such that it is possible to drive the system to rest using less input energy than $e_{\max}$) then there is a $\lambda^\star$ such that the optimal solution to the energy-constrained problem is given by the LQ-optimal controller for the criterion (4.6) with $\rho = \lambda^\star$. Thus, by varying ρ in the LQ criterion, we are able to trace the optimal trade-off surface between $\sum_t z_t^2$ and $\sum_t u_t^2$.

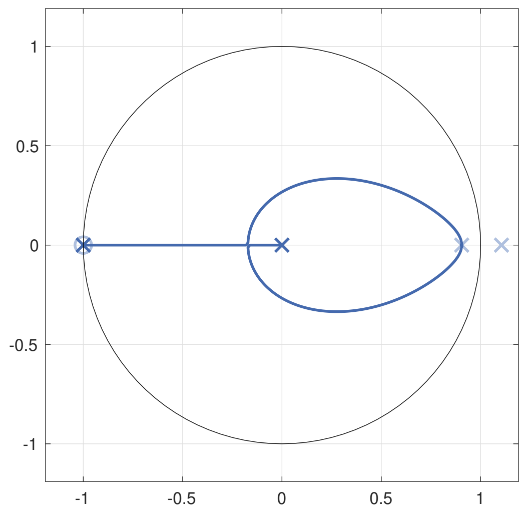


Figure 4.5: Open-loop poles (light crosses) and zeros (light rings). Dark crosses represent closed-loop poles under cheap control, and the dark line shows how the closed-loop poles move when control goes from expensive to cheap. Note how the unstable open-loop pole is mirrored in the stability boundary to ensure stability at a low control cost.

■ **Example 4.10** The boundary between the blue and white regions in Figure 4.6 defines the trade-off surface between control energy and output energy for the vertical dynamics of the quadcopter that we have studied in many of the examples. There is no controller that can do better in both of these criteria as long as it has access to the same information as the LQ-optimal controller. Therefore, the performance of every other controller will lie in the shaded area. To make this point, we evaluate the energy and control costs for the pole placement design from Example 1.6; see the white cross. It is slightly inside the trade-off curve since we can both attain a lower control cost for the same state cost, and a lower state cost for the same control cost. ■

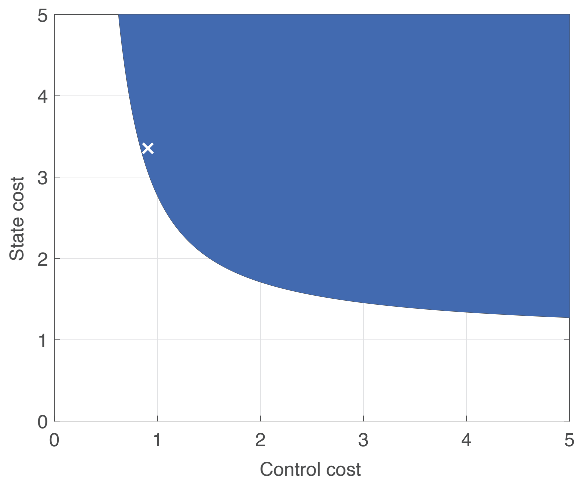


Figure 4.6: Trade-off between input energy (control cost) and output energy (state cost) for vertical quadcopter dynamics. The white cross indicates the two costs for a suboptimal design that places both closed-loop eigenvalues at $z = 0.5$.

4.4 Reference-tracking in the linear-quadratic framework

In many cases, the aim of the controller is not to drive the system state to zero, but rather to make the output $y_t = Cx_t$ follow a given reference sequence $\{r_t\}$. It turns out that such servo problems can also be dealt with using linear-quadratic regulator theory.

Tracking constant references: a feed-forward approach

When the reference is constant, *i.e.* when $r_t = r$ for all t, then error-free tracking is possible if there is an equilibrium state x^{ref} and a corresponding constant input u^{ref} such that

$$\begin{cases} x^{\text{ref}} &= Ax^{\text{ref}} + Bu^{\text{ref}} \\ r &= Cx^{\text{ref}} \end{cases} \tag{4.14}$$

These are $n+p$ equations in $n+m$ unknowns, so we will in general need that $m \geq p$, *i.e.* that we have to have at least as many control signals as the number of outputs that we want to track.

The standard LQ-control problem penalizes the (possibly weighted) norms of x_t and u_t, and thereby their distances from the origin. By a similar token, the tracking problem should penalize the differences between the true states and controls and their reference values:

$$\begin{array}{ll} \text{minimize} & \sum_{t=0}^{\infty} (x_t - x^{\text{ref}})^\top Q (x_t - x^{\text{ref}}) + (u_t - u^{\text{ref}})^\top R (u_t - u^{\text{ref}}) \\ \text{subject to} & x_{t+1} = Ax_t + Bu_t, \qquad x^{\text{ref}} = Ax^{\text{ref}} + Bu^{\text{ref}} \end{array}$$

Note that the constraints imply that

$$x_{t+1} - x^{\text{ref}} = (Ax_t + Bu_t) - (Ax^{\text{ref}} + Bu^{\text{ref}}) = A(x_t - x^{\text{ref}}) + B(u_t - u^{\text{ref}})$$

so this is just a linear-quadratic control problem in the coordinates $\Delta x_t = x_t - x^{\text{ref}}$ and $\Delta u_t = u_t - u^{\text{ref}}$:

$$\begin{array}{ll} \text{minimize} & \sum_{t=0}^{\infty} (\Delta x_t)^\top Q (\Delta x_t) + (\Delta u_t)^\top R (\Delta u_t) \\ \text{subject to} & \Delta x_{t+1} = A\Delta x_t + B\Delta u_t \end{array}$$

This means that the optimal control is

$$\Delta u_t^\star = -L\Delta x_t \Leftrightarrow u_t^\star = -L(x_t - x^{\text{ref}}) + u^{\text{ref}}$$

where L is the linear-quadratic optimal controller for a system defined by matrices A and B, and a cost defined by matrices Q and R. Under the conditions of Theorem 4.2.3, $A - BL$ will be Schur stable, and this controller will drive $x_t - x^{\text{ref}}$ and $u_t - u_t^{\text{ref}}$ to zero asymptotically. If x^{ref} and u^{ref} are chosen to satisfy (4.14), then this means that y_t will tend to r asymptotically.

In settings with multiple inputs and multiple outputs, there may be a scope to optimize the reference states and controls. Such an optimization could either determine the most efficient equilibrium point (if there are many), or find an equilibrium with small stationary tracking error if no equilibrium admits perfect tracking. When $m = p$, however, our options are limited and we can often to avoid to do an on-line optimization of $(x^{\text{ref}}, u^{\text{ref}})$ when r changes. To this end, note that the reference states and controls are not actually needed to compute the control input. Since

$$u_t = -L(x_t - x^{\text{ref}}) + u^{\text{ref}} = -Lx_t + (u^{\text{ref}} + Lx^{\text{ref}}) := -Lx_t + \bar{u}^{\text{ref}}$$

we only need to determine $\bar{u}^{\text{ref}}$. By replacing u^{ref} by $\bar{u}^{\text{ref}} - Lx^{\text{ref}}$ in (4.14), we see that we get error-free tracking in stationarity if

$$r = C(I - (A - BL))^{-1} B\bar{u}^{\text{ref}}$$

Since $A - BL$ is Schur stable, the inverse exists. If, in addition,

$$L^{\text{ref}} = \left(C(I - (A - BL))^{-1} B\right)^{-1}$$

exists then $\bar{u}^{\text{ref}} = L^{\text{ref}} r$ and the optimal tracking control is

$$u_t^\star = -Lx_t + L^{\text{ref}} r.$$

Note that this is the same feed-forward solution that we derived in Chapter 1. With the extension to reference tracking in place, we can delve deeper in control tuning and demonstrate the power of the LQ-framework on a multi-variable process.

■ **Example 4.11** The quadruple-tank apparatus shown in Figure 4.7 is a common laboratory process that is used to illustrate various aspects of control of systems with many inputs and outputs. It consists of two double-tank systems, where a fraction of the inflow generated by the pump to each upper tank is fed into the lower tank of the other system. This cross-coupling of the inflows allows

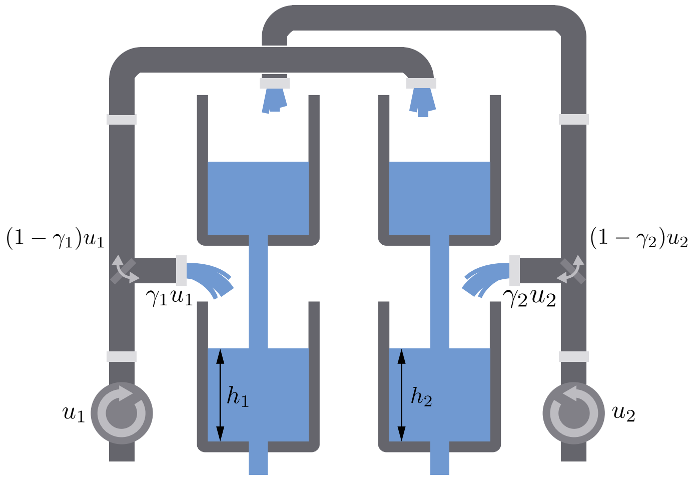


Figure 4.7: The quadruple tank apparatus.

to create a wide range of challenging dynamics. We will consider a relatively benign configuration described by the discrete-time linear system

$$x_{t+1} = \begin{pmatrix} 0.9921 & 0 & 0.0206 & 0 \\ 0 & 0.9945 & 0 & 0.0165 \\ 0 & 0 & 0.9793 & 0 \\ 0 & 0 & 0 & 0.9835 \end{pmatrix} x_t + \begin{pmatrix} 0.0415 & 0.0002 \\ 0.0001 & 0.0313 \\ 0 & 0.0237 \\ 0.0155 & 0 \end{pmatrix} u_t$$

$$y_t = \begin{pmatrix} 0.5 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 \end{pmatrix} x_t$$

Starting with $Q = I$ and $R = I$, we obtain the closed-loop step responses shown in light blue in Figure 4.8 (left). Letting $R = \rho I$ and decreasing ρ from 1 to 0.1 and 0.01 leads to the increasingly fast responses shown in darker blue in the same figure. Since the system has multiple inputs and outputs, it is also possible to shape the two outputs to behave differently. Figure 4.8 (right) shows the step responses of the LQ optimal controller for $Q = I$ and $R = I$ (light blue) along with a design that uses $R = I$ and $Q = M^\top \bar{Q} M$ with $M = C$ and $\bar{Q} = \text{diag}(10, 0)$. This design puts a larger cost on tracking errors in the first output than in the second, which results in a closed-loop with a distinctively faster response in the first output signal.

■

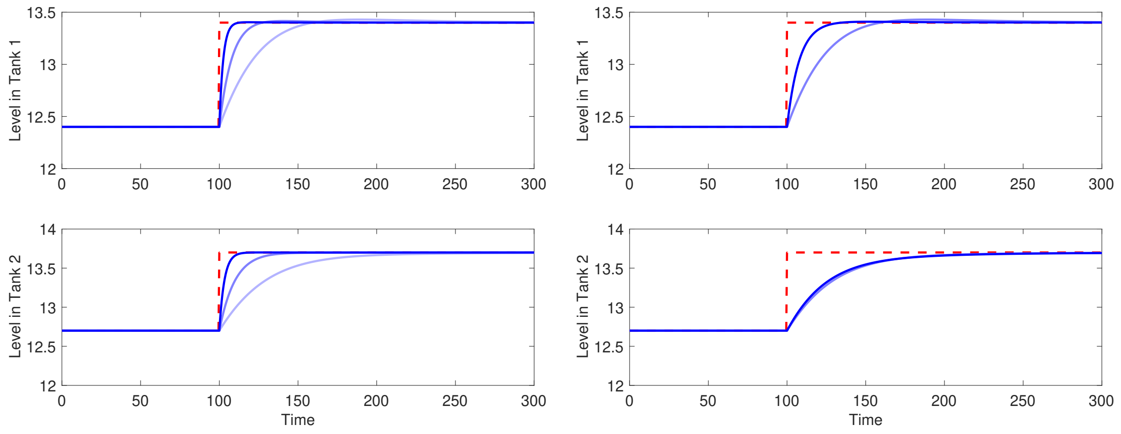


Figure 4.8: The simulations to the left show how a smaller penalty on the control signals results in faster step-responses. The right figure shows simulations of a controller where tracking errors in the first output are penalized much more than errors in the second output.

A feedback solution: integral action

A drawback with feed-forward compensation is that it tends to be sensitive to modeling errors. In the solution above, the reference states are computed under the assumption that we know A, B, and C perfectly. If the model differs from the true system, then it could very well be that the computed feed-forward control does not result in offset-free control. In practice, it is often more reliable to use feedback, and in particular integral action, to ensure error-free tracking of constant references.

The simplest way to include integral action is to introduce controller states that accumulate the errors between the reference signals and the performance outputs over time:

$$i_{t+1} = i_t + (r_t - y_t) = i_t + r_t - Cx_t$$

The closed-loop system will then have states of the plant (x_t) and the controller (i_t) that evolve as

$$\begin{pmatrix} x_{t+1} \\ i_{t+1} \end{pmatrix} = \begin{pmatrix} A & 0 \\ -C & I \end{pmatrix} \begin{pmatrix} x_t \\ i_t \end{pmatrix} + \begin{pmatrix} B \\ 0 \end{pmatrix} u_t + \begin{pmatrix} 0 \\ I \end{pmatrix} r_t \tag{4.15}$$

Note that any equilibrium point $(x^{\text{eq}}, i^{\text{eq}}, u^{\text{eq}})$ of the augmented system must also satisfy $Cx^{\text{eq}} = r$. Hence, any controller that makes the states of the extended system converge to an equilibrium point will guarantee offset-free tracking in stationarity.

We will consider optimal control problems with cost functions on the form

$$\sum_t \begin{pmatrix} x_t \\ i_t \end{pmatrix}^\top \begin{pmatrix} Q_x & 0 \\ 0 & Q_i \end{pmatrix} \begin{pmatrix} x_t \\ i_t \end{pmatrix} + u_t^\top R u_t. \tag{4.16}$$

The optimal controller for system (4.15) with cost function (4.16) is

$$u_t = \begin{pmatrix} L & L^i \end{pmatrix} \begin{pmatrix} x_t \\ i_t \end{pmatrix} = -Lx_t - L^i i_t$$

and hence combines state feedback with integral action.

Under the conditions of Theorem 4.2.3, this controller will guarantee that the augmented system is asymptotically stable in closed loop. This means that the augmented system reaches stationarity, and hence achieves offset-free tracking. The first condition of Theorem 4.2.3 demands that the augmented system is stabilizable. However, since the dynamics of the integral states are not asymptotically stable, it actually needs to be reachable. The following result is then useful.

Proposition 4.4.1 Assume that the system (A, B) is reachable. Then the augmented system is reachable if and only if the matrix

$$\begin{pmatrix} A - I & B \\ -C & 0 \end{pmatrix} \tag{4.17}$$

has row full rank.

Proof. By the PBH test, the augmented system is reachable if and only if there is no λ and no $w = (w_1, w_2) \neq 0$ such that

$$\begin{pmatrix} w_1 \\ w_2 \end{pmatrix}^\top \begin{pmatrix} A & 0 & B \\ -C & I & 0 \end{pmatrix} = \begin{pmatrix} \lambda w_1^\top & \lambda w_2^\top & 0 \end{pmatrix}$$

First note that if $w_2 = 0$, then w_1 must also be a zero vector, due to the assumption that the nominal system is reachable. On the other hand, if $w_2 \neq 0$, then any solution must have $\lambda = 1$ and satisfy

$$\begin{pmatrix} w_1 \\ w_2 \end{pmatrix}^\top \begin{pmatrix} A - I & B \\ -C & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 \end{pmatrix}$$

Due to the rank assumption, the only solution to this system of equations is $w = 0$. Hence, by the PBH test, the augmented system is reachable. ■

The rank condition requires that $m \geq p$ and that both B and C have rank of at least p. Thus, for error-free tracking, one need at least as many inputs as the number of outputs one wants to track.

The second condition of Theorem 4.2.3 requires that the augmented system state is observable through the state cost. We state the following result, whose proof is left as an exercise.

Proposition 4.4.2 Assume that (A,C) is detectable. Then

$$\left(\begin{pmatrix} A & 0 \\ -C & I \end{pmatrix}, \begin{pmatrix} Q_x^{1/2} & 0 \\ 0 & Q_i^{1/2} \end{pmatrix} \right)$$

is detectable if and only if $Q_i \succ 0$.

When tuning the LQ controller with integral action, it is convenient to view the integral state as the primary performance output. Since it is driven by the actual system output, it will converge slower, and be the limiting factor of the closed-loop bandwidth. It is therefore useful to begin with $Q_x = 0$ (assuming that (A,C) is detectable), $Q_i = I$ and $R = \rho I$ and adjust ρ to get the desired closed-loop bandwidth. The closed-loop system will typically have an overshoot in its step-response. To reduce this effect, it often works to penalize the system output (which is the rate-of-change of the integral state) by letting $Q_x = qC^\top C$ and increase q until we get the desired result.

The next example illustrates the use of integral action in linear-quadratic regulators.

■ **Example 4.12** Let us return to the quadruple tank from the previous example and add disturbances to the system. Figure 4.9 shows the previously designed LQ controller when the system is subject to disturbance inflows to the two lower tanks. We can see that the controller is unable to eliminate the effect of the disturbances.

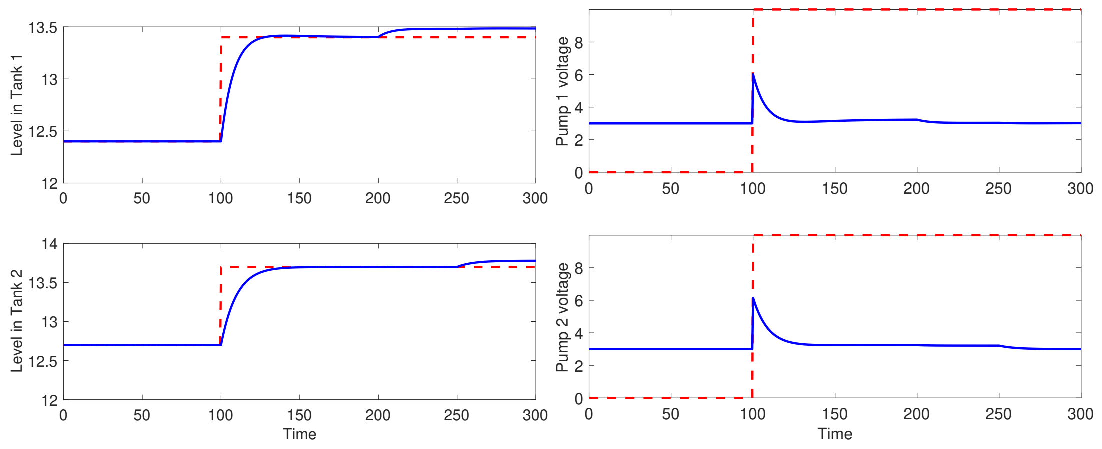


Figure 4.9: The tracking controller is able to follow the reference without error but is unable to fully eliminate the effect of constant disturbance inflows in the lower tanks.

To incorporate integral action in the controller, we note that the system has an equal number of inputs and outputs and that the rank condition on (4.17) is satisfied. We perform a LQR design with

$$Q_x = 100 \cdot C^\top C, \qquad Q_i = I, \qquad R = 0.01 \cdot I.$$

Figure 4.10 shows a simulation of reference changes in the two lower tank levels, followed by added disturbances (constant additional inflows) in the lower tanks. Notice how integral action allows the controller to perform error-free tracking and suppress the constant disturbances in stationarity. ■

Reference models and the servo problem

Although a simple feed-forward from the reference signal can achieve error-free tracking in stationarity, it can cause aggressive changes in the control signal when if the reference signal

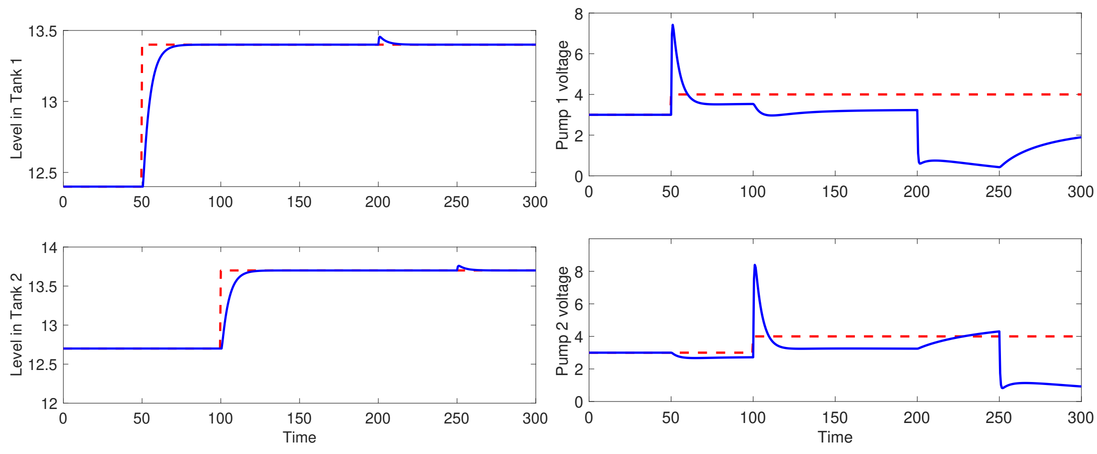


Figure 4.10: By incorporating integral action, the LQR controller allows to perform error free tracking of both outputs (left). In addition, constant disturbances are eliminated in stationarity. The associated control signals in the right figure demonstrate the multi-variable nature of the controller. For example, the reference change in the level of the first tank is dealt with by simultaneously increasing the voltage to the first pump and decreasing the voltage to the second.

changes abruptly. The standard LQ controller with integral action, on the other hand, is slower to react and can be more difficult to tune if we aim for a very specific closed-loop response. These limitations can sometimes be overcome by introducing a reference model

$$x_{t+1}^{\text{ref}} = \tilde{A}x_t^{\text{ref}} + \tilde{B}r_t, \qquad y_t^{\text{ref}} = \tilde{C}x_t^{\text{ref}}$$

that filters the reference signals and generates reference trajectories $\{x_t^{\text{ref}}\}$ for the system states. It is natural to define the reference model to have unit gain, so that y_t^{ref} tends to r_t if the reference is kept constant. To track the reference trajectory we consider the optimal control problem

$$\begin{array}{ll} \text{minimize} & \sum_{t=0}^{\infty}(x_t - x_t^{\text{ref}})^\top Q_x(x_t - x_t^{\text{ref}}) + i_t^\top Q_i i_t + u_t^\top R u_t \\ \text{subject to} & x_{t+1} = Ax_t + Bu_t \\ & x_{t+1}^{\text{ref}} = \tilde{A}x_t^{\text{ref}} + \tilde{B}r_t \\ & i_{t+1} = i_t + \tilde{C}x_t^{\text{ref}} - Cx_t \end{array}$$

Due to the presence of integrator states, y_t will be equal to y_t^{ref} in stationarity. The optimal controller can be found by solving a standard LQ problem for the augmented system

$$\bar{x}_{t+1} = \begin{pmatrix} x_{t+1} \\ x_{t+1}^{\text{ref}} \\ i_{t+1} \end{pmatrix} = \begin{pmatrix} A & 0 & 0 \\ 0 & \tilde{A} & 0 \\ -C & \tilde{C} & I \end{pmatrix} \bar{x}_t + \begin{pmatrix} B \\ 0 \\ 0 \end{pmatrix} u_t + \begin{pmatrix} 0 \\ \tilde{B} \\ 0 \end{pmatrix} r_t \tag{4.18}$$

with cost function defined by

$$\bar{Q} = \begin{pmatrix} Q_x & -Q_x & 0 \\ -Q_x & Q_x & 0 \\ 0 & 0 & Q_i \end{pmatrix}, \quad \bar{R} = R \tag{4.19}$$

The optimal controller will have the form

$$u_t = -Lx_t - L^{\text{ref}}x_t^{\text{ref}} - L^i i_t = -L(x_t - x_t^{\text{ref}}) - L^i i_t + (L - L^{\text{ref}})x_t^{\text{ref}}$$

so we can view it as a combination of a state feedback from the tracking errors, an integral term, and a feed-forward from the reference model states.

By a similar argument as for the servo problem without integral action, the augmented system (4.18) will be stabilizable if $\tilde{A}$ is Schur stable, (A,B) is stabilizable and the matrix (4.17) has full row rank. Similarly, if $Q_i \succ 0$, and both (A,C) and $(\tilde{A},\tilde{C})$ are detectable, then the detectability conditions that ensure solvability of the Riccati equation will be satisfied.

■ **Example 4.13** To demonstrate how the reference models can be used to reduce the transients during abrupt reference changes, we return to the quadruple tank system. As a reference model, we use a decoupled tank system with uniform time constants of $T = 3$ seconds, since this matches the tracking performance of our earlier controller. We use the same cost matrices as before and simulate the same sequence of reference changes and disturbances. As shown in Figure 4.11, the use of a reference model results in significantly reduced transients in the control signals compared to the ones in Figure 4.10, while the tracking performance remains virtually the same. ■

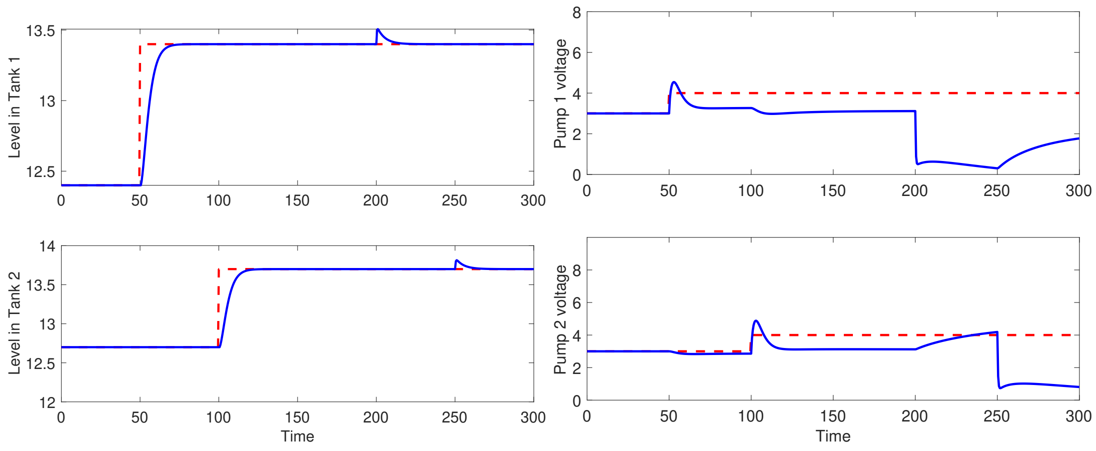


Figure 4.11: The use of a reference model simplifies the tuning of the reference tracking and allows us to reduce the transients in the control signals compared to ones in Figure 4.10.

4.5 Least-squares optimal state estimation

The linear-quadratic regulator has been derived under the assumption of perfect knowledge of the complete state vector. In practice, it is often costly or even impossible to measure all system states. Instead, one attempts to estimate the state vector from the available measurements. In Chapter 1, we discussed how observers and filters could be designed based on pole placement. However, the resulting estimator gains have no optimality guarantees and the design technique only applies to systems with a single sensor (since we would otherwise have more elements in the estimator gain matrix K than eigenvalues of the estimator error dynamics). In the next few pages, we will develop a least-squares approach to recursive state estimation. The estimate is optimal in a precise sense, and the approach applies to systems with an arbitrary number of sensors.

Recursive least-squares estimation

Consider the discrete-time linear system

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t + w_t \\ y_t &= Cx_t + v_t \end{aligned}$$

Here, the disturbance sequence $\{w_t\}$ models imperfections in our model of the state evolution, and the disturbance sequence $\{v_t\}$ models measurement noise and errors in the output equation. We will look for the disturbance sequences of smallest energy that allow us to match the predicted output with the sensor measurements (cf. the discussion in Example 3.2). Specifically, we will define the optimal state estimate via the solution to the following quadratic program

$$\begin{array}{ll} \underset{x_0,\{w_t\},\{v_t\}}{\text{minimize}} & (x_0-\bar{x}_0)^\top R_0(x_0-\bar{x}_0)+\sum_{t=0}^{T} w_t^\top R_w w_t + v_t^\top R_v v_t \\ \text{subject to} & \begin{array}{rcl} x_{t+1} & = & Ax_t + w_t \\ y_t & = & Cx_t + v_t \end{array} \end{array} \tag{4.20}$$

Here $\bar{x}_0$ is a prior guess of the initial state, while R_0, R_w, and R_v are positive semi-definite matrices. These matrices are tuning parameters for our estimator and play a similar role as the weight matrices in the linear-quadratic regulator problem. If we have high faith in the state update $x_{t+1} = Ax_t + Bu_t$, then we should choose a large R_w to prioritize making $\{w_t\}$ small; and if we believe that the output measurements are error-free, we should choose a large R_v.

The least-squares estimation problem (4.20) can be solved using dynamic programming, resulting in a the following recursive estimator of the state sequence $\{\hat{x}_{t|t}\}$

Theorem 4.5.1 The least-squares state estimation problem (4.20) can be solved recursively by repeated application of the following updates

Measurement update:

$$\begin{aligned} \bar{K}_t &= (S_t + C^\top R_v C)^{-1} C^\top R_v \\ \hat{x}_{t|t} &= \hat{x}_{t|t-1} + \bar{K}_t (y_t - C\hat{x}_{t|t-1}) \\ \bar{S}_t &= S_t + C^\top R_v C \end{aligned}$$

Prediction step:

$$\begin{aligned} \hat{x}_{t+1|t} &= A\hat{x}_{t|t} + Bu_t \\ S_{t+1} &= R_w - R_w A(\bar{S}_t + A^\top R_w A)^{-1} A^\top R_w \end{aligned}$$

from initial values $\hat{x}_{0|-1} = \bar{x}_0$ and $S_0 = R_0$.

Proof. The proof is given in Appendix D.3. ■

The least-squares estimator is intimately related to the better-known Kalman filter. Although the Kalman filter is derived from a stochastic perspective, it can be seen as a least-squares filter that maintains $P_t = S_t^{-1}$ and parameterizes the criterion in terms of $\Sigma_0 = R_0^{-1}$, $\Sigma_w = R_w^{-1}$ and $\Sigma_v = R_v^{-1}$. Specifically, it solves the quadratic program

$$\begin{array}{ll} \underset{x_0,\{w_t\},\{v_t\}}{\text{minimize}} & (x_0-\bar{x}_0)^\top \Sigma_0^{-1}(x_0-\bar{x}_0)+\sum_{t=0}^{T} w_t^\top \Sigma_w^{-1} w_t + v_t^\top \Sigma_v^{-1} v_t \\ \text{subject to} & \begin{array}{rcl} x_{t+1} & = & Ax_t + w_t \\ y_t & = & Cx_t + v_t \end{array} \end{array} \tag{4.21}$$

The corresponding filter updates can be derived by applying standard matrix inversion identities to the updates derived in Theorem 4.5.1, which leads to the following result.

Theorem 4.5.2 The estimation problem (4.20) with $R_0 = \Sigma_0^{-1}$, $R_w = \Sigma_w^{-1}$ and $R_v = \Sigma_v^{-1}$ can be solved recursively by repeated application of the updates

Measurement update:

$$\begin{aligned}\bar{K}_t &= P_t C^\top(\Sigma_v + CP_tC^\top)^{-1}\\ \hat{x}_{t|t} &= \hat{x}_{t|t-1} + \bar{K}_t(y_t - C\hat{x}_{t|t-1})\\ \bar{P}_t &= P_t - P_tC^\top(\Sigma_v + CP_tC^\top)^{-1}CP_t\end{aligned}$$

Prediction step:

$$\begin{aligned}\hat{x}_{t+1|t} &= A\hat{x}_{t|t} + Bu_t\\ P_{t+1} &= \Sigma_w + A\bar{P}_tA^\top\end{aligned}$$

from initial values $\hat{x}_{0|-1} = \bar{x}_0$ and $P_0 = \Sigma_0$.

Proof. See Appendix D.3. ■

By similar reasoning as for the least-squares filter, we should use a small value of Σ_w if we have high confidence in the state update equations, while high accuracy measurements should be reflected by selecting a small value of Σ_v.

Note that the state estimators have the same structure as the ones considered in Chapter 1, while the filtering gain is in general time-varying. More specifically, the one-step-ahead prediction is

$$\hat{x}_{t+1|t} = A\hat{x}_{t|t-1} + Bu_t + K_t(y_t - \hat{y}_{t|t-1}), \qquad \hat{y}_{t|t-1} = C\hat{x}_{t|t-1}$$

where $K_t = A\bar{K}_t$, while the optimal estimate of the current state is given by

$$\begin{aligned}\hat{x}_{t|t} &= \hat{x}_{t|t-1} + \bar{K}_t(y_t - C\hat{x}_{t|t-1})\\ \hat{x}_{t+1|t} &= A\hat{x}_{t|t-1} + Bu_t,\end{aligned}$$

In contrast to the state estimators that we have considered earlier, the filtering gain $\bar{K}_t$ has to be computed on-line, based on the matrix P_t (or S_t) that is also updated in every time step.

The stationary Kalman filter

Just like for the linear-quadratic regulator, it is easier to analyze and implement the stationary version of the Kalman filter, where the estimator gains are constant. The stationary solution can be found by first noticing that the filter gain $\bar{K}_t$ only depends on P_t. By eliminating the intermediate variable $\bar{P}_t$, the updates for P_t can be written as

$$P_{t+1} = \Sigma_w + AP_tA^\top - AP_tC^\top(\Sigma_v + CP_tC^\top)^{-1}CP_tA^\top. \tag{4.22}$$

This is a Riccati recursion for $\{P_t\}$ with strong connections to the recursion that is used to define the linear-quadratic regulator. Any stationary solution to (4.22) satisfies the algebraic Riccati equation

$$\begin{aligned}P &= \Sigma_w + APA^\top - APC^\top(\Sigma_v + CPC^\top)^{-1}CPA^\top = &(4.23)\\ &= (A-KC)P(A-KC)^\top + \Sigma_w + K\Sigma_vK^\top &(4.24)\end{aligned}$$

where $K = A\bar{K}$ and $\bar{K} = PC^\top(\Sigma_v + CPC^\top)^{-1}$. With these constant gain matrices, the optimal one-step-ahead predictor becomes

$$\hat{x}_{t+1|t} = (A-KC)\hat{x}_{t|t-1} + Bu_t + Ky_t$$

while the stationary Kalman filter simplifies to

$$\begin{aligned}\hat{x}_{t+1|t} &= (A-KC)\hat{x}_{t|t-1} + Bu_t + Ky_t \\ \hat{x}_{t|t} &= (I-\bar{K}C)\hat{x}_{t|t-1} + \bar{K}y_t\end{aligned}$$

Both observers are given by simple linear time-invariant systems. A short calculation verifies that the dynamics of the estimation errors $e_{t+1|t} = \hat{x}_{t+1|t} - x_t$ and $e_{t|t} = \hat{x}_{t|t} - x_t$ are asymptotically stable if $A-KC$ is Schur; see Exercise 4.12. The next result, which is an analog of Theorem 4.2.3, demonstrates that this is indeed the case under mild conditions.

Theorem 4.5.3 Consider the linear system $x_{t+1} = Ax_t + Bu_t$, $y_t = Cx_t$ and the cost defined by (4.21) with $\Sigma_w \succeq 0$, $\Sigma_v \succ 0$, and $T \to \infty$. If (A,C) is detectable, then the cost is bounded. If, in addition, $(A, \Sigma_w^{1/2})$ is stabilizable, then the algebraic Riccati equation (4.23) admits a unique positive semidefinite solution and $A-KC$ is Schur stable.

The next example develops an intuition for how the tuning parameters Σ_w and Σ_v affect the behavior of the associated stationary Kalman predictor.

■ **Example 4.14** Consider the scalar system

$$\begin{aligned}x_{t+1} &= ax_t + w_t \\ y_t &= x_t + v_t\end{aligned}$$

with $\Sigma_w = 1$ and $\Sigma_v = r$. The stationary Kalman filter gain is

$$K = \frac{aP}{r+P}$$

where P satisfies the Riccati equation

$$P = 1 + a^2P - \frac{a^2P^2}{r+P}$$

To gain insight into the optimal estimator, we consider the dynamics of the estimation error

$$e_{t+1} = (a-K)e_t$$

When $\Sigma_v = r \to 0$ (which corresponds to error-free measurements), $K \to a$ and

$$\hat{x}_{t+1} = ay_t = ax_t$$

Thus, the observer disregards any previous information and constructs the state estimate using the last measurement only. When $r \to \infty$ (very corrupted measurements), on the other hand, we have

$$a - K \to \begin{cases} a & \text{if } |a| \le 1 \\ a^{-1} & \text{otherwise} \end{cases}$$

Thus, if the process is open-loop stable, the estimator disregards the measurements and uses its model to predict the state estimate. If the system is unstable, the optimal gain is such that the error dynamics are stable and its pole is the inverse of the open-loop system pole. ■

A note on the duality between estimation and control*

As noted above, the infinite-horizon control and estimation problems bear striking resemblances: the optimal control is a linear state feedback $u_t = -Lx_t$ that drives the state (error) dynamics $x_{t+1} = (A - BL)x_t$ to zero, while the optimal estimator adjusts the state estimates by a factor $K_t(y_t - \hat{y}_t)$ such that the estimation error dynamics $e_{t+1} = (A - KC)e_t$ tends to zero asymptotically. The optimal gains are computed from the solutions to similar AREs: (4.9) and (4.23).

It turns out that this is no coincidence: estimation and control are dual in a precise mathematical sense. To the discrete-time linear system

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t \\ y_t &= Cx_t \end{aligned}$$

we can associate a dual system

$$\begin{aligned} \tilde{x}_{t+1} &= A^\top \tilde{x}_t + C^\top \tilde{u}_t \\ \tilde{y}_t &= B^\top \tilde{x}_t \end{aligned}$$

By applying the reachability and observability tests in Chapter 1, we notice that the original system is reachable if and only if its dual is observable (and vice versa). We can also verify that the optimal estimator for the primal system with LQR cost parameterized by Σ_w and Σ_v coincides with the optimal LQR controller for the dual system with $Q = \Sigma_w$ and $R = \Sigma_v$ (and vice versa).

One consequence of this duality is that it is enough that numerical control design packages provide support for solving either the LQR or the Kalman filter ARE. The other one can be solved by transforming the input data as revealed by the duality argument.

We will explore another insight of the duality argument, namely how to select weight matrices in the Kalman filter to ensure fast error dynamics. Note that if we apply $\tilde{u}_t = -K^\top \tilde{x}_t$ in the dual system, we get the closed-loop dynamics $\tilde{x}_{t+1} = (A^\top - C^\top K^\top)\tilde{x}_t = (A - KC)^\top \tilde{x}_t$. By the cheap control argument in Section 4.3, we know that this closed-loop dynamics will become increasingly fast if we use a criterion that puts an increasingly large penalty on $\tilde{y}_t^\top \tilde{y}_t = \tilde{x}_t^\top BB^\top \tilde{x}_t$; for example, by letting $Q = \sigma BB^\top$ and $R = I$ and increasing σ. By the duality argument, we should be able to compute this K by solving the estimation problem for the original system with $\Sigma_w = \sigma BB^\top$ and $\Sigma_v = I$. The next example illustrates that this intuition is indeed correct.

■ **Example 4.15** Consider the mechanical system from Example 3.3. The dual system has a two complex conjugate poles near the unit circle, and one zero at -1. Computing the optimal Kalman filter with $\Sigma_w = \sigma BB^\top$ and $\Sigma_v = I$, and letting σ vary from 1 to 10000 yields the eigenvalues of $A - KC$ shown in Figure 4.12. As can be expected from the cheap control analogy, one pole tends to the origin, while the other tends to the system zero. ■

4.6 Output feedback control

The LQ-optimal controller can be combined with a Kalman filter to form an output feedback controller. This controller measures the system output, estimates the state vector using the filter, and computes the control action as a linear feedback from the estimated states; see Figure 4.13.

When we use the infinite-horizon optimal controller and stationary estimator gains in the one-step ahead Kalman predictor, the controller is described by the equations

$$\begin{aligned} \hat{x}_{t+1} &= A\hat{x}_t + Bu_t + K(y_t - \hat{y}_t) = (A - BL - KC)\hat{x}_t + Ky_t \\ u_t &= -L\hat{x}_t \end{aligned}$$

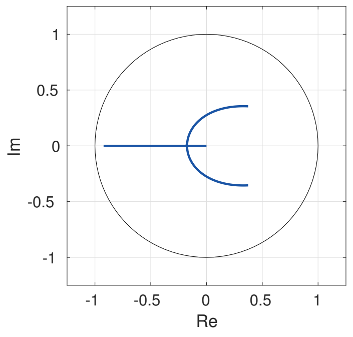


Figure 4.12: The poles of the estimation error dynamics in dark blue: one pole tends to the origin while the other one tends to the system zero.

If we use the Kalman filter, on the other hand, the controller dynamics is given by

$$\begin{aligned}\hat{x}_{t+1|t} &= (A - KC - BL(I - \bar{K}C))\hat{x}_{t|t-1} + (K - BL\bar{K})y_t \\ u_t &= -L(I - \bar{K}C)\hat{x}_{t|t-1} - L\bar{K}y_t\end{aligned}$$

When the conditions of Theorem 4.2.3 and Theorem 4.5.3 are met, then $A - BL$ and $A - KC$ are both Schur stable matrices and the closed-loop system is guaranteed to be asymptotically stable. However, the controller itself is not necessarily asymptotically stable, i.e. $A - BL - KC$ and $A - BL - KC + BL\bar{K}C$ are not necessarily Schur stable matrices. Although this may not always be a problem (in fact, there are systems that can only be stabilized using unstable controllers), it is judicious to always verify the internal stability of the computed control law.

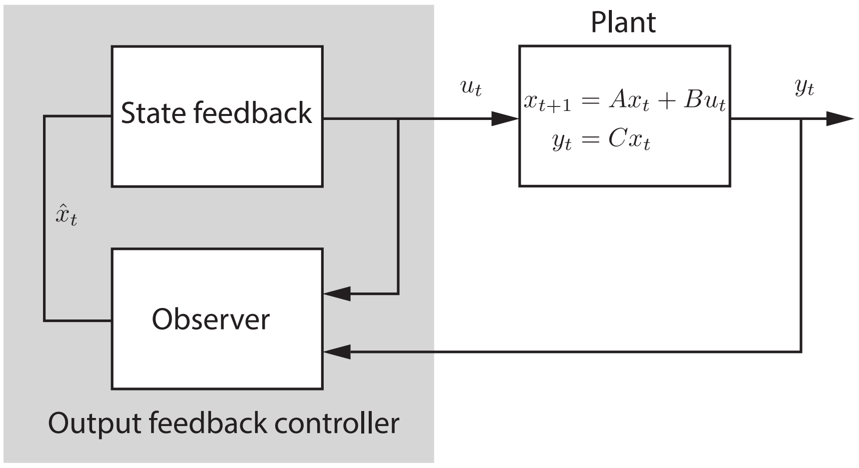


Figure 4.13: Output feedback control: combining a state estimator and linear feedback from the estimated states.

It is, in general, difficult to establish optimality properties of control laws where the estimator and feedback gains have been designed in separation. A notable exception is the stochastic setting of *linear-quadratic Gaussian (LQG)* control. If the state and output disturbance seqeunces $\{w_t\}$ and $\{v_t\}$ are independent white noise sequences with known covariance matrices Σ_w and Σ_v, respectively, LQR-optimal feedback from states estimated by the Kalman filter is indeed the ouput feedback policy that minimizes the expected value of the quadratic cost (4.6).

Even though the LQG controller is optimal in a precise way, it can be very sensitive to modeling errors, cf. Exercise 4.13. It is thus essential to evaluate the robustness properties of the controller prior to deployment. In practice, the combination of estimator and feedback from estimated states can often be made to work well, provided that the observer dynamics is significantly faster than the closed-loop dynamics. For the Kalman filter, we have discussed above how the bandwidth can be increased by adding artificial noise to the state equations, *e.g.* by letting

$$\Sigma_w = \Sigma_w^{\text{nom}} + \sigma_w BB^\top$$

and increasing the parameter σ_w. This procedure is known as *loop transfer recovery*. However, the procedure is not guaranteed to result a in robust controller, and the closed-loop system should always be analyzed for robustness and the influence of unmodeled disturbances.

4.7 Disturbance modeling and compensation

The linear-quadratic controller is designed from a regulation perspective, in the sense that it aims to drive the system states to the origin. We have also shown how it can also be used to design controllers that track a given reference signal. In this section, we will describe how persistent exogeneous disturbances can be compensated for in the linear-quadratic framework. The approach is based on modelling the exogenous signals as outputs of autonomous linear systems, and including these systems in the model used for linear-quadratic controller design.

To describe the approach, consider the system

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t + B_d d_t \\ y_t &= Cx_t + C_d d_t \end{aligned}$$

where d_t is a vector of disturbance signals which we do not measure. Rather than considering the disturbances as arbitrary, we assume that we can model them as the output of a linear system

$$\begin{aligned} z_{t+1} &= A_z z_t \\ d_t &= C_z z_t \end{aligned} \tag{4.25}$$

In this way, the dynamics of the system and the disturbances that act on it can be described by

$$\begin{aligned} \begin{pmatrix} x_{t+1} \\ z_{t+1} \end{pmatrix} &= \begin{pmatrix} A & B_d C_z \\ 0 & A_z \end{pmatrix} \begin{pmatrix} x_t \\ z_t \end{pmatrix} + \begin{pmatrix} B \\ 0 \end{pmatrix} u_t \\ y_t &= \begin{pmatrix} C & C_d C_z \end{pmatrix} \begin{pmatrix} x_t \\ z_t \end{pmatrix} \end{aligned} \tag{4.26}$$

By designing a stabilizing controller for this extended system model, we can ensure that $y_t \to 0$, despite the presence of disturbances. In particular, we will design an output feedback controller for the extended system based on a Kalman filter and linear feedback from the estimated extended state vector. The resulting control strategy, illustrated in Figure 4.14, combines feedback from the estimated process states and compensation for the estimated disturbances.

Many common disturbances in control systems can be modeled as outputs of autonomous linear systems. For example, a constant disturbance of unknown amplitude can be modeled as

$$z_{t+1} = z_t, \qquad d_t = z_t$$

The unknown initial value z_0 represents the constant level of the disturbance. A sinusoidal disturbance with natural frequency ω_0 rad/sec can be modelled as the output of a harmonic oscillator

$$z_{t+1} = \begin{pmatrix} \cos(\omega_0 h) & \sin(\omega_0 h) \\ -\sin(\omega_0 h) & \cos(\omega_0 h) \end{pmatrix} z_t, \qquad d_t = \begin{pmatrix} 1 & 0 \end{pmatrix} z_t$$

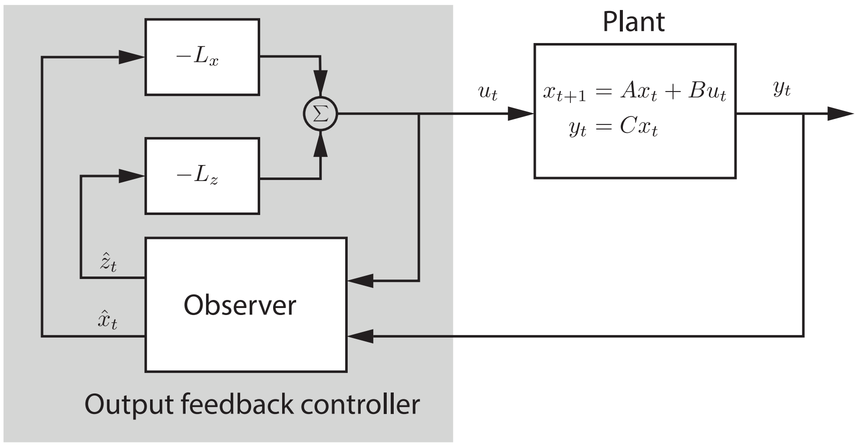


Figure 4.14: Output feedback controller with combined state and disturbance estimator.

where h is the sampling time of the discrete system (in seconds). The first state of this model is the amplitude of the disturbance, while the second one describes the rate-of-change. A general periodic disturbance of unknown shape but with known period of n_d samples can be described by

$$z_{t+1} = \begin{pmatrix} 0_{(n_d-1)\times 1} & I_{(n_d-1)\times(n_d-1)} \\ 1 & 0_{1\times(n_d-1)} \end{pmatrix} z_t, \qquad d_t = \begin{pmatrix} 1 & 0_{(1\times(n_d-1))} \end{pmatrix} z_t$$

A problem with the disturbance models described above is that their dynamics is only stable, and not asymptotically stable. Since the control signal cannot affect the disturbance state vector in (4.26), the extended system is not stabilizable. A practical engineering solution to this problem is to perturb the nominal disturbance dynamics slightly, *e.g.* replacing A_z by $A_z - \varepsilon I$ for some small value of ε that makes $A_z - \varepsilon I$ Schur stable.

The framework can also be extended to deal with reference tracking problems. To this end, note that when the reference r_t is constant, the desired output signal generated by the model

$$x^{\text{ref}}_{t+1} = A^{\text{ref}} x^{\text{ref}}_t + B^{\text{ref}} r_t, \qquad y^{\text{ref}}_t = C^{\text{ref}} x^{\text{ref}}_t$$

can be represented as the output of the following autonomous linear system

$$\xi_{t+1} = \begin{pmatrix} A^{\text{ref}} & B^{\text{ref}} \\ 0 & 1 \end{pmatrix} z_t := A_r \xi_t, \qquad y^{\text{ref}}_t = \begin{pmatrix} C^{\text{ref}} & 0 \end{pmatrix} \xi_t := C_r \xi_t.$$

The constant value of r_t is now reflected by the final component of z_t, while the first components of z_t contain x^{ref}_t. This technique can then be combined with the approach to disturbance compensation that we have described above. For example, we can consider the reference tracking problem

$$\begin{array}{ll} \text{minimize} & \sum_t (y_t - y^{\text{ref}}_t)^\top (y_t - y^{\text{ref}}_t) + u_t^\top R u_t \\ \text{subject to} & x_{t+1} = Ax_t + Bu_t + B_d d_t, \quad y_t = Cx_t + C_d d_t \\ & z_{t+1} = A_z z_t, \quad d_t = C_z z_t \\ & \xi_{t+1} = A_r \xi_t \quad y^{\text{ref}}_t = C_r \xi_t \end{array}$$

We then first perturb A_z and A_r so that they are Schur stable. Then, we form the extended system

$$\begin{pmatrix} x_{t+1} \\ z_{t+1} \\ \xi_{t+1} \end{pmatrix} = \begin{pmatrix} A & B_d C_z & 0 \\ 0 & A_z & 0 \\ 0 & 0 & A_r \end{pmatrix} \begin{pmatrix} x_t \\ z_t \\ \xi_t \end{pmatrix} + \begin{pmatrix} B \\ 0 \\ 0 \end{pmatrix} u_t$$

and design an LQR controller using the weights $Q = C_e^\top Q_e C_e$, $R = \rho_e I$ where

$$C_e = \begin{pmatrix} C & C_d C_z & -C_r \end{pmatrix}$$

The resulting control law computes u_t as a linear combination of x_t, z_t and ξ_t, which are typically not all available for direct measurements. Thus, as a final step, we design a Kalman filter for the extended system. Note that the reference model is something that we implement. We can therefore design the Kalman filter with

$$y_t = \begin{pmatrix} C & C_d C_z & 0 \\ 0 & 0 & I \end{pmatrix} \begin{pmatrix} x_t \\ z_t \\ \xi_t \end{pmatrix} + v_t$$

We demonstrate the disturbance modeling and compensation idea for the design of a high-performance servo control for an optical data storage device.

■ **Example 4.16** DVD is an optical storage format where digital data is read from (and written to) compact discs using laser. The discs are organized in tracks, 0.076μm apart, which the laser head needs to find and follow. Since all discs are slightly asymmetric, the tracks oscillate relative to a fixed laser position when the disc spins. The magnitude of these oscillations is dramatic, about 5000 times larger than the acceptable tracking error. To realize reliable high-throughput DVD drives, it is therefore absolutely essential to have a high-performance control system for positioning the laser head above the tracks; see Figure 4.15. In fact, the control system is so important that its performance requirements is part of the DVD standard.

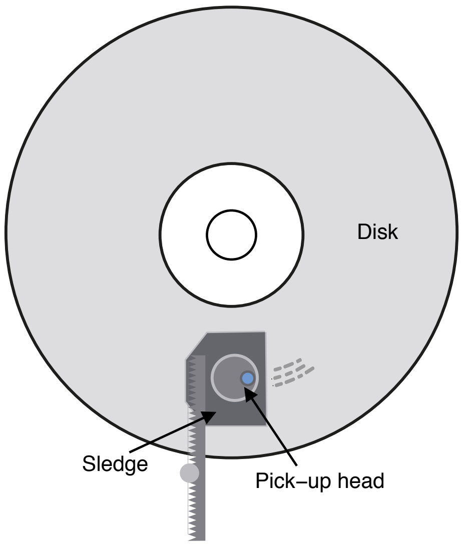


Figure 4.15: A typical DVD design: a pickup head with laser, lens, and the light detector is mounted on a sledge that can move radially across the disc. To achieve the desired position accuracy, the laser is mounted onto the sledge using springs and can be moved relative to the sledge by electromagnets. In this way, it can also be moved small amounts quickly and with high accuracy.

The following linear system describes the identified radial dynamics of a DVD servo

$$\begin{aligned} x_{t+1} &= \begin{pmatrix} 0.9982 & 0.0073 \\ -0.0073 & 0.9965 \end{pmatrix} x_t + \begin{pmatrix} -9.0497 \\ -9.0647 \end{pmatrix} u_t \\ y_t &= \begin{pmatrix} -9.0497 & 9.0647 \end{pmatrix} x_t + d_t \end{aligned}$$

with a sampling frequency of 40 kHz. The system has a high static gain, but also a non-minimum phase zero that limits the achievable bandwidth of the system. The disturbance d_t is sinusoidal with a frequency of roughly 20 Hz. It can therefore be modelled as a harmonic oscillator as described above. After a slight perturbation to ensure that the disturbance dynamics is asymptotically stable, we find the disturbance model

$$z_{t+1} = \begin{pmatrix} 0.9990 & 0.0031 \\ -0.0031 & 0.9990 \end{pmatrix}, \qquad d_t = \begin{pmatrix} 1 & 0 \end{pmatrix}$$

We then form the corresponding extended system (A_e, B_e, C_e) described in (4.26) and design an output feedback controller for it using Kalman filtering and LQ-optimal control.

The performance specifications for the DVD drive are given in terms of its frequency response (or, rather, its loop gain). Nevertheless, we can use the linear-quadratic methodology and tune the weight matrices to get the desired response. We start out with $Q = C_e^\top C_e$, $R_e = 1$, $\Sigma_w = I$ and $\Sigma_v = 1$. With $R_e = 0.1$, we get the right bandwidth of the system, but the low-frequency gain of the loop gain is too low. By adjusting Σ_w to $2I + 1.5BB^\top$, we get a reasonable loop gain, and it is difficult to increase the low-frequency gain further without losing internal stability of the controller.

The light blue lines in Figure 4.16 show a simulation of a nominal controller, designed without accounting for the output disturbance. We apply a unit reference change at $t = 0.01$ seconds. At time $t = 0.02$ seconds, we add the output disturbance $d(t) = 100\sin(40\pi(t - 0.02))$. Although the controller is able to reduce the effect of the disturbance, there is still a substantial tracking error.

To reduce the tracking error, we add a disturbance model to our system. It turns out that this controller is sensitive to abrupt reference changes, so we also add a reference model to our design. We chose a second-order reference model $(A^{\text{ref}}, B^{\text{ref}}, C^{\text{ref}}, D^{\text{ref}})$ with unit gain, no zeros, and two real poles placed at $z = 0.1$ and $z = 0.11$. For the controller, we use $Q_e = 1$ and $\rho_e = 0.1$, while the Kalman filter is designed using $\Sigma_v = 1$ and

$$\Sigma_w = \begin{pmatrix} 0.1BB^T & 0 & 0 \\ 0 & 10^{12} B_d B_d^T & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

The large differences in scaling of the blocks is mainly due to the different magnitudes of B and B_d. The resulting response, shown as dark lines in Figure 4.16 illustrate a significantly improved disturbance suppression (the peak error is reduced by a factor of roughly 50). ■

Disturbance observers and integral action

Although we have not proven this formally in these notes, it is well-known that integral action in a controller allows to compensate for constant disturbances. It is thus natural to ask if the combination of a disturbance observer and feedback from estimated states gives a controller with integral action when the disturbance is assumed to be constant. It turns out that this is indeed true. We will demonstrate it on a system with a scalar input disturbance, which can be described by the extended system (4.26) with matrices

$$B_d = B, \qquad C_d = 0, \qquad A_z = 1, \qquad C_z = 1$$

The combination of Kalman filter and state feedback from the estimated states of the extended system gives an output feedback controller whose dynamics is described by

$$\begin{aligned} \hat{x}_{t+1} &= A\hat{x}_t + B\hat{z}_t + Bu_t + K(y_t - \hat{y}_t) \\ \hat{z}_{t+1} &= \hat{z}_t + K_d(y_t - \hat{y}_t) \\ u_t &= -L\hat{x}_t - L_z\hat{z}_t \end{aligned}$$

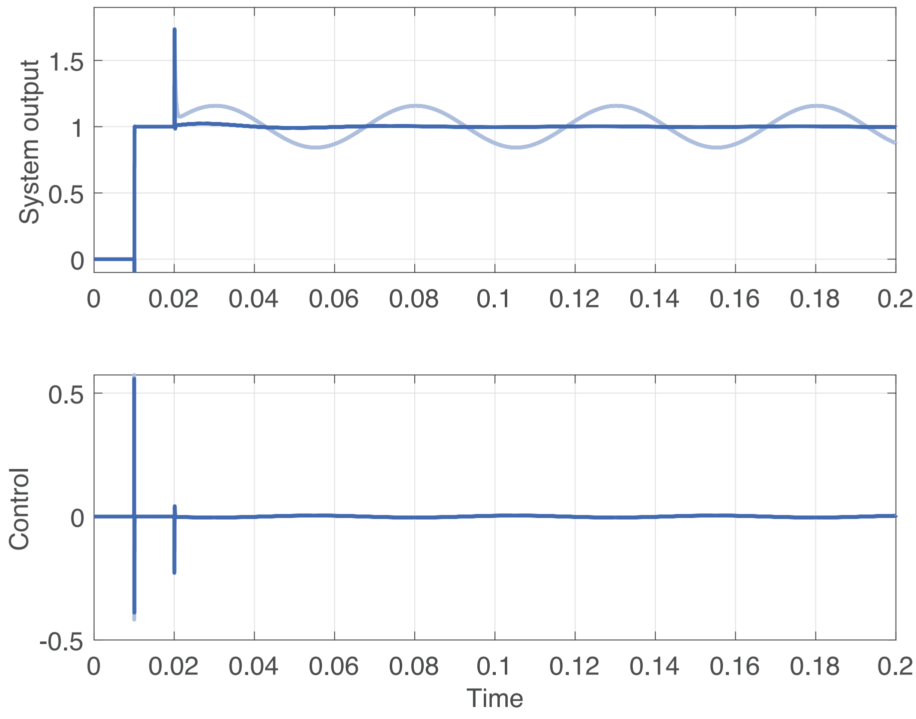


Figure 4.16: Effect of 20Hz output disturbance on nominal closed-loop system (light lines) and system under the output feedback which uses an internal disturbance model (dark line). Note that the disturbance is only applied from $t = 0.02$, which creates a transient while the estimation errors in the Kalman filter converge.

which we can write as

$$\begin{pmatrix} \hat{x}_{t+1} \\ \hat{z}_{t+1} \end{pmatrix} = \begin{pmatrix} A - BL & B(1 - L_z) \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \hat{x}_t \\ \hat{z}_t \end{pmatrix} + \begin{pmatrix} K \\ K_d \end{pmatrix} (y_t - \hat{y}_t)$$

$$u_t = -\begin{pmatrix} L & L_z \end{pmatrix} \begin{pmatrix} \hat{x}_t \\ \hat{z}_t \end{pmatrix}$$

Due to the block diagonal structure of the system matrix, the controller will have n poles at the eigenvalues of $A - BL$ and one pole at $+1$. Thus, the controller will have an integral state, and the control law can be seen as a feedback $L\hat{x}_t$ from the estimated plant state plus an integral term $L_z\hat{z}_t$.

4.8 Exercises

Problem 4.1 Consider the scalar system

$$x_{t+1} = x_t + u_t$$

(a) Compute the sequence of positive scalars $\{P_t\}$ that characterize the cost-to-go function $v_t(x_t) = P_t x_t^2$ for the quadratic cost

$$\sum_{t=0}^{T-1} x_t^2 + u_t^2 + 10x_T^2$$

Verify numerically that the sequence converges and the value that P_0 converges to as $T \to \infty$. What is the corresponding optimal control $u_0(x_0)$?

(b) Repeat the same investigation for the double integrator

$$x_{t+1} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} x_t + \begin{pmatrix} 1/2 \\ 1 \end{pmatrix} u_t, \qquad y_t = \begin{pmatrix} 1 & 0 \end{pmatrix} x_t \tag{4.27}$$

with the cost

$$\sum_{t=0}^{T-1} y_t^\top y_t + u_t^2 + 10x_T^\top x_T.$$

Plot the entries of the matrices $\{P_t\}$ of the Riccati recursion. Verify that as $T \to \infty$, P_0 converges to the solution of the Riccati equation

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

Problem 4.2 Let us now consider the double integrator under the infinite-horizon cost

$$\sum_{t=0}^{\infty} x_t^\top \begin{pmatrix} q_x & 0 \\ 0 & q_v \end{pmatrix} x_t + u_t^2$$

(a) Is it possible to find a solution to the DARE that ensures an asymptotically stable closed loop when $q_x = 0$ or $q_y = 0$? Justify your answer.

(b) How do you expect the closed-loop to behave under the optimal control law defined by $q_x = 0.1$ and $q_v = 100$? Verify your intuition by computing the optimal control law and simulating an initial response from $x_0 = \mathbf{1}$. What happens when you increase or decrease the value of q_v?

(c) How do you expect the closed-loop to behave when $q_x = 100$ and $q_v = 0.1$? Verify your intuition by computing the optimal control law and simulating an initial response from $x_0 = \mathbf{1}$. What happens when you increase or decrease the value of q_x?

(d) Choose weights so that the initial response from $x_0 = \mathbf{1}$ settles in 10 seconds without any significant overshoot. Begin by letting $q_v = 0.001$ and tune q_x to get roughly the right bandwidth. Then adjust q_v to decrease the overshoot until you get the desired response. What weights do you find?

Problem 4.3 Consider the discrete-time linear system

$$x_{t+1} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} x_t + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} u_t$$

Compute the LQ-optimal feedback gains and the associated stationary Riccati solution for the cost defined by the state weights $Q = I$ and the three control weights

$$R = \begin{pmatrix} 1000 & 0 \\ 0 & 1 \end{pmatrix}, \qquad R = \begin{pmatrix} 1 & 0 \\ 0 & 1000 \end{pmatrix}, \text{ and } \quad R = \begin{pmatrix} 1000 & 0 \\ 0 & 1000 \end{pmatrix}$$

Discuss qualitatively what you observe.

Problem 4.4 Consider the discrete-time linear system

$$x_{t+1} = \begin{pmatrix} 0.8 & 0 \\ 1 & 1.2 \end{pmatrix} x_t + \begin{pmatrix} 1 \\ 0 \end{pmatrix} u_t$$

with cost

$$J = \sum_{t=0}^{\infty} x_t^\top x_t + \rho u_t^2$$

Find the optimal feedback gain numerically for different values of $\rho = 10^k$ and $k = -3, -2, \dots, 2, 3$. Plot how the eigenvalues of the resulting closed-loop system matrix $A - BL$ vary with ρ. Explain qualitatively what you observe.

Problem 4.5 Figure 4.17 shows pole-zero maps and closed-loop step-responses for three LQR designs for the system

$$x_{t+1} = \begin{pmatrix} 0.9904 & 0.1939 \\ -0.0950 & 0.9361 \end{pmatrix} x_t + \begin{pmatrix} 0.0196 \\ 0.1939 \end{pmatrix} u_t, \qquad y_t = \begin{pmatrix} 1 & 0 \end{pmatrix} x_t$$

All three designs use $Q = I$ but differ in their selection of $\rho \in \{0.001, 0.1, 10\}$. Can you deduce which row in Figure 4.17 corresponds to what value of ρ? Justify your answer.

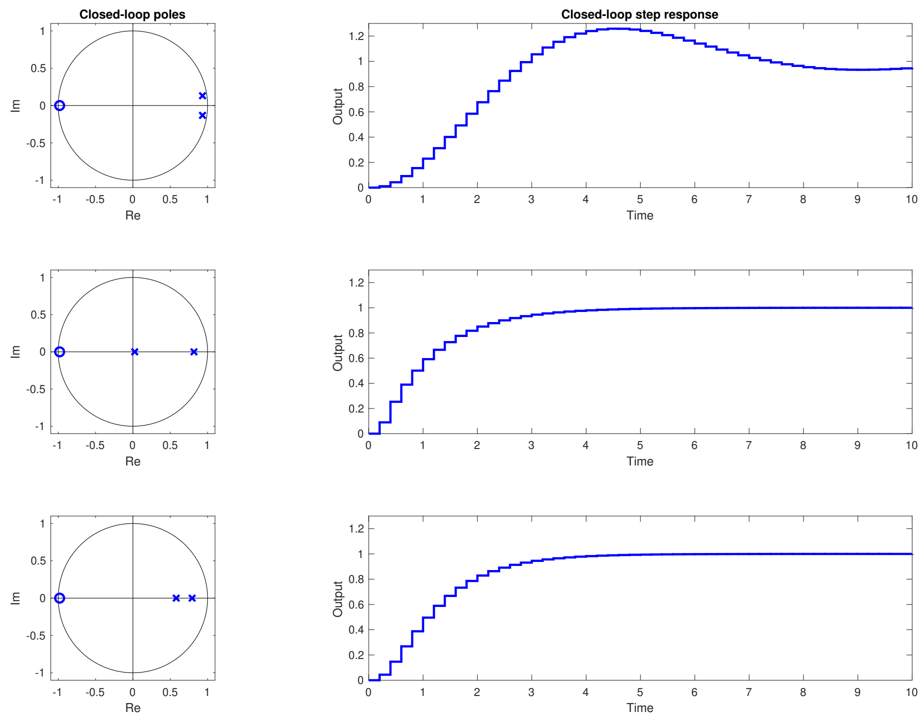


Figure 4.17: Pole-zero maps and step-responses of closed-loop systems in Problem 4.8.

Problem 4.6 Consider the discrete-time linear system

$$x_{t+1} = \begin{pmatrix} 1.2 & 0 \\ 1 & 0.8 \end{pmatrix} x_t + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u_t$$

with cost

$$J = \sum_{t=0}^{\infty} x_t^\top x_t + u_t^2$$

and $x_0 = \mathbf{1}$. What is the smallest value of J that is possible to attain? Verify your answer by computing the optimal state-feedback, simulating the closed-loop system from $x_0 = \mathbf{1}$, and recording the value of the LQ-cost along the closed-loop trajectory.

Problem 4.7 We consider the system

$$\begin{aligned} x_{t+1} &= \begin{pmatrix} 1 & 0 \\ 2 & 3 \end{pmatrix} x_t + \begin{pmatrix} 1 \\ 0 \end{pmatrix} u_t \\ y_t &= \begin{pmatrix} 0 & 1 \end{pmatrix} x_t \end{aligned}$$

(a) Design an LQR controller of the form $u_t = -Lx_t - l_r r_t$ using performance weights

$$Q = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad R = 1$$

and the reference feedforward is designed so that the static gain from r to y is equal to one.

(b) Assume that there is a constant disturbance acting on the input, so $u_t' = u_t + d$. Assume $r_t = 1$ and $d_t = 1$. Do you expect the static error $\lim_{t\to\infty} e_t = \lim_{t\to\infty} y_t - r_t$ to be present? Compute the stationary output and compare it with your intuition. What do you observe?

(c) Introduce an additional integral state i_t in the system and design LQR controller $u_t = -Lx_t - l_r r_t - l_i i_t$ for the performance criterion

$$J = \sum_{t=0}^{\infty} e_t^\top e_t + i_t^2 + u_t^2$$

where $e = x - x_{ref}$ denotes the state error.

(d) Assume again that $d_t = 1$ and $r_t = 1$. Do you expect the static error to be present now? Compute the stationary output and compare it with your observation. Comment your observations!

Problem 4.8 A common variation on the LQR problem includes explicit time-varying weighting factors on the state and input costs,

$$J = \sum_{\tau=0}^{N-1} \gamma^\tau (x_\tau^\top Q x_\tau + u_\tau^\top R u_\tau) + \gamma^N x_N^\top Q_N x_N,$$

where $x_{t+1} = Ax_t + Bu_t$, $y_t = Cx_t$, x_0 is given, and, as usual, we assume $Q = Q^\top \geq 0$, $Q_N = Q_N^\top \geq 0$, and $R = R^\top > 0$ are constant. The parameter γ, called the exponential weighting factor, is positive. For $\gamma = 1$, this reduces to the standard LQR cost function. For $\gamma < 1$, the penalty for future state and input deviations is smaller than in the present; in this case we call γ the *discount factor* or *forgetting factor*. When $\gamma > 1$, future costs are accentuated compared to present costs. This gives added incentive for the input to steer the state towards zero quickly.

(a) Note that we can find the input sequence $u_0^\star, \dots, u_{N-1}^\star$ that minimizes J using standard LQR methods, by considering the state and input costs as time-varying, with $Q_t = \gamma^t Q$, $R_t = \gamma^t R$, and final cost given by $\gamma^N Q_N$. Thus, we know at least one way to solve the exponentially weighted LQR problem. Use this method to find the recursive equations that give $u^\star$.

(b) Exponential weights can also be incorporated directly into a dynamic programming formulation. We define

$$W_t(z) = \min \sum_{\tau=t}^{N-1} \gamma^{\tau-t}(x_\tau^\top Q x_\tau + u_\tau^\top R u_\tau) + \gamma^{N-t} x_N^\top Q_N x_N,$$

where $x_t = z$, $x_{\tau+1} = Ax_\tau + Bu_\tau$, and the minimum is over $u_t, \dots, u_{N-1}$. This is the minimum cost-to-go, if we started in state z at time t, with time weighting also starting at t. Argue that

$$W_N(z) = x_N^\top Q_N x_N$$

$$W_t(z) = \min_w \left(z^\top Q z + w^\top R w + \gamma W_{t+1}(Az + Bw)\right),$$

and that the minimizing w is in fact $u_t^\star$. In other words, work out a backward recursion for W_t, and give an expression for $u_t^\star$ in terms of W_t. Show that this method yields the same $u^\star$ as the first method.

(c) Yet another method can be used to find $u^\star$. Define a new system as

$$y_{t+1} = \gamma^{1/2} A y_t + \gamma^{1/2} B z_t$$

Argue that we have $y_t = \gamma^{t/2} x_t$, provided $z_t = \gamma^{t/2} u_t$, for $t = 0, \dots, N-1$. With this choice of z, the exponentially weighted LQR cost J for the original system is given by

$$J = \sum_{\tau=0}^{N-1} (y_\tau^\top Q y_\tau + z_\tau^\top R z_\tau) + y_N^\top Q_N y_N,$$

i.e., the unweighted LQR cost for the modified system. We can use the standard formulas to obtain the optimal input for the modified system $z^\star$, and from this, we can get $u^\star$. Do this, and verify that once again, you get the same $u^\star$.

Problem 4.9 This problem considers an extension of linear-quadratic regulation to tracking of a reference trajectory that accounts for the full control cost of doing so (instead of only penalizing the deviations from a reference control). To that end, consider a discrete-time linear system

$$x_{t+1} = Ax_t + Bu_t$$

with the cost

$$J = \sum_{k=0}^{T} (x_k - x_k^{\mathrm{ref}})^\top Q (x_k - x_k^{\mathrm{ref}}) + \sum_{k=0}^{T-1} u_k^\top R u_k$$

where $Q \succ 0$ and $R \succ 0$. The reference trajectory $\{x_k^{\mathrm{ref}}\}$ is assumed to be known. We want to use dynamic programming to derive the optimal control law and show that it consists of a feedback term from the system state x_k and a feed-forward term from the reference state x_k^{ref}.

(a) Show that the cost-to-go function at time T, $V_T(x)$ has the form

$$V_T(x) = x^\top P_T x + 2q_T^\top x + r_T$$

for some $P_T \succ 0$, q_T and r_T.

(b) Assume that

$$V_{t+1}(x) = x^\top P_{t+1} x + 2q_{t+1}^\top x + r_{t+1}$$

with $P_{t+1} \succ 0$. Show that the optimal control at time t is on the form

$$u_t^\star = -L_t x_t + m_t$$

where

$$L_t = (R + B^\top P_{t+1} B)^{-1} B^\top P_{t+1} A, \qquad m_t = -(R + B^\top P_{t+1} B)^{-1} B^\top q_{t+1}$$

(c) Use dynamic programming (backward induction) to show that $V_T, V_{T-1}, \dots V_0$ all have the given form. Verify that

$$\begin{aligned}
P_t &= Q + A^\top P_{t+1} A - A^\top P_{t+1} B (R + B^\top P_{t+1} B)^{-1} B^\top P_{t+1} A,\\
q_t &= (A - BL_t)^\top q_{t+1} - Q x_t^{\text{ref}}\\
r_t &= r_{t+1} + (x_t^{\text{ref}})^\top Q x_t^{\text{ref}} + q_{t+1}^\top B m_t
\end{aligned}$$

Problem 4.10 We have proven that the LQR control law guarantees an asymptotically stable closed-loop. In other words, the optimal solution $u_t = -Lx_t$ to

$$\begin{array}{ll}
\text{minimize} & \sum_{t=0}^{\infty} x_t^\top Q x_t + u_t^\top R u_t\\
\text{subject to} & x_{t+1} = Ax_t + Bu_t
\end{array}$$

guarantees that $(A - BL)$ has all eigenvalue inside the unit disc.

We will now extend these results and explore how linear-quadratic control theory can be used to design a feedback control law that places the poles in a disc with a given radius α and center $(\beta, 0)$ in the complex plane.

(a) Let us begin by the case $\beta = 0$, *i.e.* the desired pole locations are a disc centered at the origin with radius α. Show that the optimal solution to

$$\begin{array}{ll}
\text{minimize} & \sum_{t=0}^{\infty} \left(\frac{1}{\alpha}\right)^{2t} [x_t^\top Q x_t + u_t^\top R u_t]\\
\text{subject to} & x_{t+1} = Ax_t + Bu_t
\end{array} \tag{4.28}$$

results in closed-loop poles in the desired region.

Hint Show that the variable transformation $\bar{x}_t = x_t/\alpha^t$, $\bar{u}_t = u_t/\alpha^t$ allows to transform (4.28) into a LQR problem on standard form

$$\begin{array}{ll}
\text{minimize} & \sum_{t=0}^{\infty} \bar{x}_t^\top Q \bar{x}_t + \bar{u}_t^\top R \bar{u}_t\\
\text{subject to} & \bar{x}_{t+1} = \bar{A}\bar{x}_t + \bar{B}\bar{u}_t.
\end{array} \tag{4.29}$$

and argue about what the known properties of $\bar{x}_t$ in closed loop implies for the corresponding x_t under the optimal control u_t.

(b) Show that modifying $\bar{A}$ to $\bar{A} - (\beta/\alpha)I$ in (4.29) yields a feedback law $u_t = -Lx_t$ which places all eigenvalues of $A - BL$ in a disc with radius α and center at $(\beta, 0)$.

(c) Use the approach derived above to compute a feedback that places the closed-loop poles of

$$x_{t+1} = \begin{pmatrix} 1 & 0.1 \\ 0 & 1 \end{pmatrix} x_t + \begin{pmatrix} 0.005 \\ 0.1 \end{pmatrix} u_t$$

in a disc with radius 0.25, centered at $(0.5, 0)$. Please use the cost matrices $Q = I$ and $R = 1$ when you perform your calculations.

Problem 4.11 One disadvantage of more advanced control strategies is it can take a significant time to compute the control signal. This computational delay can have a detrimental effect on the stability of the closed loop. In this problem, we will explore the effects of computational delays and how to design optimal controllers that compensate for them.

(a) To gain some intuition, let us first consider a scalar system

$$x_{t+1} = 2x_t + u_t$$

and compute the control law $u_t = -Lx_t$ that minimizes the cost

$$\sum_{t=0}^{\infty} x_t^2 + u_t^2$$

Is the closed-loop system asymptotically stable?

(b) Assume now that the system is subject to a computational delay of one sample. This means that the system evolves according to

$$x_{t+1} = 2x_t + u_{t-1}.$$

Write the closed-loop dynamics on state-space form with state vector (x_t, u_{t-1}) and input u_t. Is the system with $u_t = -Lx_t$ computed in (a) asymptotically stable?

(c) One way to compensate for the delay is to replace the current state by a prediction of its future value. Let the nominal system be given by the state-space model

$$x_{t+1} = Ax_t + Bu_t$$

and assume that the state feedback $u_t = -Lx_t$ renders $(A - BL)$ is Schur stable.
Let us now assume that the system is subject to a computational delay of one sample. In other words, the state vector evolves as

$$x_{t+1} = Ax_t + Bu_{t-1}$$

To compensate for this delay, we will modify the nominal state feedback to use a prediction of the future state

$$u_t = -L\hat{x}_{t+1|t} = -L(Ax_t + Bu_{t-1}).$$

Show that this strategy ensures the stability of the associated closed-loop

$$\begin{pmatrix} x_{t+1} \\ u_t \end{pmatrix} = \begin{pmatrix} A & B \\ -LA & -LB \end{pmatrix} \begin{pmatrix} x_t \\ u_{t-1} \end{pmatrix}$$

Hint. Recall that A and TAT^{-1} have the same eigenvalues and that the eigenvalues of

$$\left(\begin{array}{c|c} A_{11} & A_{12} \\ \hline 0 & A_{22} \end{array}\right)$$

is the eigenvalues of the matrices A_{11} and A_{22}.

(d) We can take the result in (c) even further. Prove the following statement:

Let $u_t = -Lx_t$ be the optimal control law for the reachable linear system $x_{t+1} = Ax_t + Bu_t$ and the infinite-horizon cost

$$\sum_{t=0}^{\infty} x_t^\top Q x_t + u_t^\top R u_t$$

Then, the control law that minimizes the same cost for the input delayed system

$$x_{t+1} = Ax_t + Bu_{t-1}$$

is the feedback law

$$u_t = -L(Ax_t + Bu_{t-1}).$$

Problem 4.12 In this exercise, we will analyze the error dynamics of the stationary Kalman one-step-ahead predictor and the Kalman filter for the linear system

$$x_{t+1} = Ax_t + Bu_t, \qquad y_t = Cx_t$$

(a) Consider the one-step-ahead predictor

$$\hat{x}_{t+1|t} = A\hat{x}_{t|t-1} + Bu_t + K(y_t - C\hat{x}_{t|t-1})$$

and the estimation error $e_{t|t-1} = \hat{x}_{t|t-1} - x_t$. Show that the estimation error evolves as

$$e_{t+1|t} = (A - KC)e_{t|t-1}$$

This shows that the estimation errors are asymptotically stable if $A - KC$ is Schur stable.

(b) For the stationary Kalman filter

$$\begin{aligned}\hat{x}_{t+1|t} &= (A - KC)\hat{x}_{t|t-1} + Bu_t + Ky_t\\ \hat{x}_{t|t} &= (I - \bar{K}C)\hat{x}_{t|t-1} + \bar{K}y_t\end{aligned}$$

show that the dynamics of $e_{t|t} = \hat{x}_{t|t} - x_t$ evolves as

$$e_{t+1|t+1} = (A - \bar{K}CA)e_{t|t}$$

Show that $A - \bar{K}CA$ has the same eigenvalues as $A - KC$, and that also this error dynamics is asymptotically stable if $A - KC$ is Schur stable.

Hint. Recall that $K = A\bar{K}$ and that for any real and square matrices X and Y of the same dimensions, the matrix products XY and YX have the same eigenvalues.

Problem 4.13 In this problem, we will explore the robustness properties (or, rather, the lack or robustness properties) of the linear-quadratic output feedback controller.

(a) Consider the discrete-time linear system $x_{t+1} = Ax_t + Bu_t$, $y_t = Cx_t$ with

$$A = \begin{pmatrix} -0.6 & 0.7 \\ 1.0 & -0.7 \end{pmatrix}, \qquad B = \begin{pmatrix} 0.8 \\ 0.9 \end{pmatrix}, \qquad C = \begin{pmatrix} 1 & 0 \end{pmatrix}$$

Design an LQ controller with weights

$$Q = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}, \qquad R = 1$$

and a stationary one-step ahead Kalman predictor with $\Sigma_w = Q$ and $\Sigma_v = R$. Verify that the eigenvalues of $A - BL$ and $A - KC$ are both stable. Is the corresponding output feedback controller also stable?

(b) Assume that you use the model $x_{t+1} = Ax_t + Bu_t$, $y_t = Cx_t$ for designing your controller, but the actual system dynamics is given by $x_{t+1} = A_a x_t + B_a u_t$, $y_t = C_a x_t$. Show that the closed-loop error dynamics are given by the system

$$\begin{pmatrix} x_{t+1} \\ e_{t+1} \end{pmatrix} = \begin{pmatrix} A_a - B_a L & B_a L \\ (A_a - A) - (B_a - B)L - K(C_a - C) & Aa + (B_a - B)L - KC \end{pmatrix} \begin{pmatrix} x_t \\ e_t \end{pmatrix}$$

where $e_t = x_t - \hat{x}_t$.

(c) Assume that the controller designed in (b) used the correct values of A and C, while $B_a = B + \begin{pmatrix} 0 & 10^{-4} \end{pmatrix}^\top$. Are the error dynamics still asymptotically stable?

Problem 4.14 Two pendulums with lengths l_1 and l_2 and with masses m_1 and m_2 are attached to a moving cart with mass M. The system can be affected by an external force F, which is under our control; see Figure 4.18. Assuming that the masses of the two pendulums are much smaller than

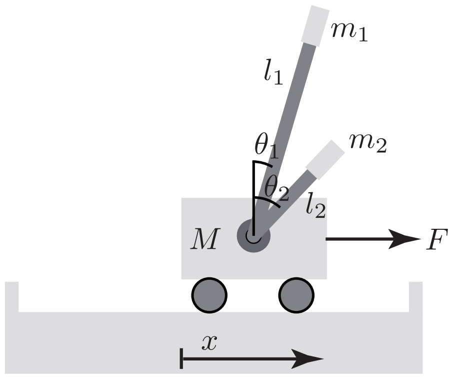


Figure 4.18: Schematic illustration of the double-inverted pendulum system.

the cart mass, the dynamics of the system is given by

$$\begin{aligned} l_1\ddot{\theta}_1 &= g\sin(\theta_1) + \cos(\theta_1)u \\ l_2\ddot{\theta}_2 &= g\sin(\theta_2) + \cos(\theta_2)u \\ \ddot{x} &= u \end{aligned}$$

where $u = F/M$. Linearization around $\theta_1 \approx \theta_2 \approx 0$ gives the linear model

$$\frac{d}{dt}\begin{pmatrix} \theta_1(t) \\ \dot{\theta}_1(t) \\ \theta_2(t) \\ \dot{\theta}_2(t) \\ x(t) \\ \dot{x}(t) \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ g/l_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & g/l_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \theta_1(t) \\ \dot{\theta}_1(t) \\ \theta_2(t) \\ \dot{\theta}_2(t) \\ x(t) \\ \dot{x}(t) \end{pmatrix} + \begin{pmatrix} 0 \\ 1/l_1 \\ 0 \\ 1/l_2 \\ 0 \\ 1 \end{pmatrix} u(t)$$

We will initially consider a system with $l_1 = 0.5$ and $l_2 = 0.1$. You can use the value $g = 9.81$ for the standard acceleration due to gravity.

(a) Sample the system with $h = 0.02$ seconds and determine the associated discrete-time system description.

$$x_{t+1} = Ax_t + Bu_t, \qquad y_t = Cx_t + Du_t$$

(b) Plot the eigenvalues of A and identify the eigenvalues (open-loop poles) which correspond to the dynamics of the cart, to the dynamics of the first pendulum, and to the dynamics of the second pendulum.

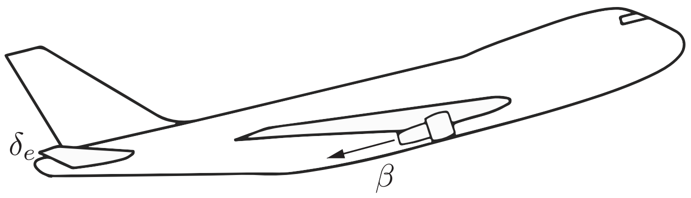
(c) Consider an initial state where the cart is in rest, and the pendulum angles are $+5^\circ$ and -5°, respectively. In other words,

$$x_0 = \begin{pmatrix} \pi/36 & 0 & -\pi/36 & 0 & 0 & 0 \end{pmatrix} \tag{4.30}$$

Perform an LQR design which is able to drives the pendulum angles to within $\pm 1^\circ$ in less than 2.5 seconds, while keeping all angles within $\pm 25^\circ$ during the transient.

(d) Prove that the system is not reachable if the two pendulums have equal lengths, $l_1 = l_2 = 0.5$.

(e) Intuitively, it should be more difficult to stabilize the system the more similar the two pendulums are in length. Quantify this difficulty by solving an "expensive control" problem defined by $Q = 10^{-6}I$ and $R = 1$ for different values of l_2. Let l_2 vary from 1.01 to 0.11 meter in steps of 0.05 meter, and calculate the infinite-horizon cost-to-go from x_0 defined in (4.30). Plot the cost-to-go against l_2. What can you observe?



Problem 4.15 The longitudinal dynamics of an aircraft in a steady climb of 15° with a constant speed of 200 km/h is given by

$$\dot{x}(t) = \begin{pmatrix} -0.045 & 1.96 & -9.48 & 0 \\ -0.0065 & -1.96 & -0.046 & 1 \\ 0 & 0 & 0 & 1 \\ 0.0059 & -6.83 & 0.042 & -2.9 \end{pmatrix} x(t) + \begin{pmatrix} 0 & 0.25 \\ 0.30 & 0 \\ 0 & 0 \\ 0.12 & 0 \end{pmatrix} u(t).$$

Here, the state vector $x = \begin{pmatrix} v & \alpha & \theta & q \end{pmatrix}$ represents deviations in airspeed [m/s], angle of attack [rad], pitch angle [rad] and pitch rate [rad/s]from their respective steady states; the control vector $u = \begin{pmatrix} \delta_e & \beta \end{pmatrix}$ comprises deviations in the elevator deflection [rad] and the throttle angle [rad]. We assume that the plane can measure the air speed v and the pitch rate q.

We would like to design an airspeed control system that responds to a step-change in reference speed of 1m/s with a maximal settling time of 15s without steady-state error, while the maximum elevator deflection should be limited to $\pm 10^\circ$ and the maximum throttle deflection should not exceed $\pm 0.5^\circ$.

(a) Sample the system with $h = 0.1$ seconds and compute the feed-forward gain L^{ref} in the control law

$$u = L^{\text{ref}} x^{\text{ref}} - L(x_t - x^{\text{ref}}) \tag{4.31}$$

to track a given reference state x^{ref}.

(b) Verify that the following two vectors are valid reference states:

$$\bar{x}^{(1)} = \begin{pmatrix} 1 \\ 0.0014 \\ -0.0045 \\ 0 \end{pmatrix}, \qquad \bar{x}^{(2)} = \begin{pmatrix} 1 \\ 0.0014 \\ 0 \\ 0 \end{pmatrix}$$

Is there any reason to prefer one over the other?

(b) Consider the control strategy (4.31) and the second suggestion for x^{ref} above. Let $Q = I$ and determine an appropriate control penalty matrix R, given the desired maximal values of the control signals.

5. Model predictive control

In this chapter, we will shift focus to control of constrained systems. The presence of constraints makes the control design problem much more challenging, even if both the system dynamics and the constraints are linear. For example, we have already seen in Chapter 3 that the cost-to-go functions for such problems are non-quadratic and therefore non-trivial to compute and represent.

To circumvent this difficulty, we will rely on receding horizon control. At each sampling instant, these strategies measure the current process state and compute an open-loop optimal control sequence over a fixed horizon. However, rather than executing the entire sequence, they implement only the first control action. At the next sampling instant, they repeat the process—measuring the current state, computing a new optimal control sequence, and executing the first action in that sequence. By continuously updating the control sequence based on the most recent state information, the controller adapts to the true system dynamics. Since this approach uses a model of the process to predict the consequences of the control actions, it is known as *model predictive control* (MPC). In contrast to open-loop strategies, where the complete control sequence is predetermined based on the initial state, MPC incorporates feedback by updating the control at each sampling instant.

In this chapter, we focus on linear systems with linear constraints and (convex) quadratic costs. From Chapter 3, we know that open-loop optimal control sequences for such problems can be computed efficiently using quadratic programming. However, using open-loop optimal sequences in a receding-horizon fashion results in a control policy that seems very different from traditional control laws, since the control signal is specified indirectly as the outcome of an optimization problem. Nevertheless, we will show that in the absence of constraints, the model-predictive control strategy results in a linear state feedback law that can be tuned to be identical to the linear-quadratic controller that we have studied in the previous chapter. In the presence of constraints, on the other hand, the model-predictive control strategy results in a nonlinear state feedback law that is typically only defined for a subset of the process states. We devote a significant effort to developing an understanding of how the different parameters of a model predictive controller affect the stability and performance of the closed-loop system. Special attention is given to techniques that reduce the computational cost and enlarge the operating range of the MPC strategy. Finally, we discuss important engineering aspects such as reference tracking and disturbance suppression.

5.1 Model-predictive control: two basic ideas

Model predictive control is based on two key ideas. The first idea is to determine the control input by solving an (open-loop) optimal control problem. This step is enabled by developing a finite-dimensional approximation of the underlying infinite-horizon optimal control problem that can be solved quickly and reliably online. The second idea is to introduce feedback using the receding-horizon principle. At every sample, we measure the system state, plan an optimal input trajectory over a finite horizon, but only implement the first action in the planned sequence. The combination of these ideas leads to a basic MPC policy that we will eventually be able to analyze, tune, and extend in many interesting directions.

A finite-horizon approximation of infinite-horizon optimal control

Consider the following infinite-horizon linear quadratic control problem under constraints

$$\begin{array}{lll} \text{minimize} & \sum_{t=0}^{\infty} x_t^\top Q x_t + u_t^\top R u_t & \\ \text{subject to} & x_{t+1} = Ax_t + Bu_t & t = 0,1,\dots \\ & x_t \in X, \quad u_t \in U & t = 0,1,\dots \end{array} \tag{5.1}$$

We can view (5.1) as an optimization problem with an infinite number of variables $\{x_t, u_t\}_{t=0}^{\infty}$. Intuitively, as long as there exists a control sequence that drives the state to zero, we should be able to approximate this problem by one that has a finite (but possibly long) horizon. To this end, we split the objective function of (5.1) into a *finite-horizon cost* for the first T time steps, followed by an infinite-horizon *tail cost* that captures the behaviour from sample T and on:

$$\begin{array}{lll} \text{minimize} & \sum_{t=0}^{T-1} x_t^\top Q x_t + u_t^\top R u_t + \sum_{t=T}^{\infty} x_t^\top Q x_t + u_t^\top R u_t & \\ \text{subject to} & x_{t+1} = Ax_t + Bu_t & t = 0,1,\dots \\ & x_t \in X, \quad u_t \in U & t = 0,1,\dots \end{array}$$

From the previous chapters, we know that the minimal value of the cost over the tail is given by the cost-to-go function from state x_T at time T. Hence, we can re-write our infinite-horizon optimal control problem as an equivalent optimization problem over a finite horizon T:

$$\begin{array}{lll} \text{minimize} & \sum_{t=0}^{T-1} x_t^\top Q x_t + u_t^\top R u_t + v(x_T) & \\ \text{subject to} & x_{t+1} = Ax_t + Bu_t & t = 0,1,\dots,T-1 \\ & x_t \in X, \quad u_t \in U & t = 0,1,\dots,T-1 \end{array} \tag{5.2}$$

So far, this reformulation is only formal, since we do not have an efficient way for computing and representing $v(\cdot)$ when x and u are constrained. However, we know that v is quadratic in the absence of constraints. Hence, if we can drive x_T to a region X_T in which the system is guaranteed to operate without ever violating the constraints, then we can replace $v(x)$ by a quadratic approximation $\hat{v}(x) = x^\top Q_T x$; ; see Figure 5.1. This leads to the following finite-horizon approximation of (5.1):

$$\begin{array}{lll} \text{minimize} & \sum_{t=0}^{T-1} x_t^\top Q x_t + u_t^\top R u_t + x_T^\top Q_T x_T & \\ \text{subject to} & x_{t+1} = Ax_t + Bu_t & t = 0,1,\dots,T-1 \\ & x_t \in X, \quad u_t \in U & t = 0,1,\dots,T-1 \\ & x_T \in X_T & \end{array} \tag{5.3}$$

Note that this problem includes both a terminal cost $x_T^\top Q_T x_T$ and a terminal constraint, $x_T \in X_T$.

When Q, R and Q_T are positive semidefinite and the sets X, U and X_T are polyhedra, the optimization problem (5.3) is a convex quadratic program. We can therefore hope to solve it quickly and reliably, even for relatively large state dimensions and long planning horizons. The quality of the approximation (5.3) depends on the planning horizon T. If T is long enough so that the optimal solution to the infinite-horizon problem (5.1) naturally drives the system state to X_T at time T, then there is no loss. But if the planning horizon is short, then we may need to deviate from the infinite-horizon optimal control to guarantee that the state reaches X_T in just T samples. This can lead to a suboptimal performance, and it can also restrict the set of initial states for which we can solve the planning problem (5.3). We will discuss these issues in much more detail later.

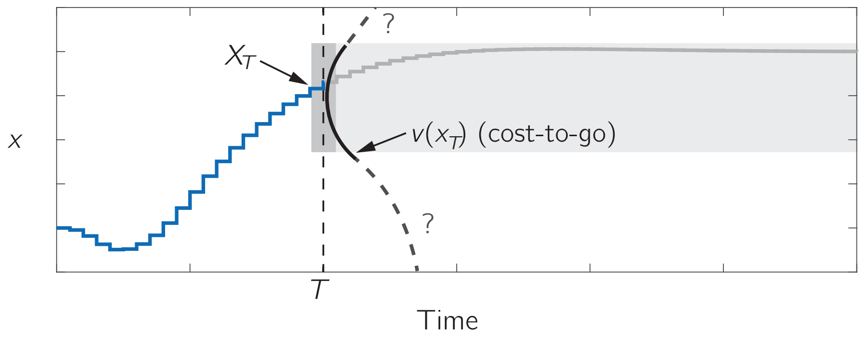


Figure 5.1: The terminal state ensures that the state at the end of the prediction horizon is in a region of the state space where we have a good (typically quadratic) estimate of the remaining cost-to-go for the associated infinite-horizon problem.

The receding-horizon control principle

Although the reformulation (5.3) is interesting from a computational perspective, it defines an open-loop control sequence. As we discussed in Chapter 3, open-loop controls can be sensitive to errors in the model that is used to predict the future states. To introduce feedback, we will use a *receding-horizon control approach.* At each sampling time, we measure the system state x_t, solve a finite-horizon planning problem to predict the optimal controls (and states) for the next T samples, and apply the first control action in the computed sequence. We then wait until the next sampling time and repeat this procedure. Specifically, at every time t, we solve the planning problem

$$\begin{array}{lll} \text{minimize} & \sum_{k=0}^{T-1} \hat{x}_k^\top Q \hat{x}_k + \hat{u}_k^\top R \hat{u}_k + \hat{x}_T^\top Q_T \hat{x}_T & \\ \text{subject to} & \hat{x}_{k+1} = A\hat{x}_k + B\hat{u}_k & k = 0, \dots, T-1 \\ & \hat{x}_k \in X, \quad \hat{u}_k \in U & k = 0, \dots, T-1 \\ & \hat{x}_T \in X_T & \\ & \hat{x}_0 = x_t & \end{array} \tag{5.4}$$

to find its optimal solution $\{\hat{x}_0^\star, \dots, \hat{x}_T^\star\}$ and $\{\hat{u}_0^\star, \dots, \hat{u}_{T-1}^\star\}$, and apply the control

$$u_t = \hat{u}_0^\star \tag{5.5}$$

(that is, the first action control in the planned sequence). In the finite-horizon planning problem, we have used the notation $\hat{u}_k$ and $\hat{x}_k$ for the predicted optimal constraints k steps into the future. The notation is chosen to emphasize that the solution to the planning problem *predicts* future optimal states and controls based on knowledge of the current state and under the assumption that our model agrees with the true system dynamics[1]; see Figure 5.2 for a pictorial illustration.

[1]This notation is not standard: some literature use $u_{t+k|t}$ and $x_{t+k|t}$ instead of $\hat{x}_k$ and $\hat{u}_k$. But in this chapter x_t and u_t (without hats) refer to the true system state and the control action applied to the system, respectively.

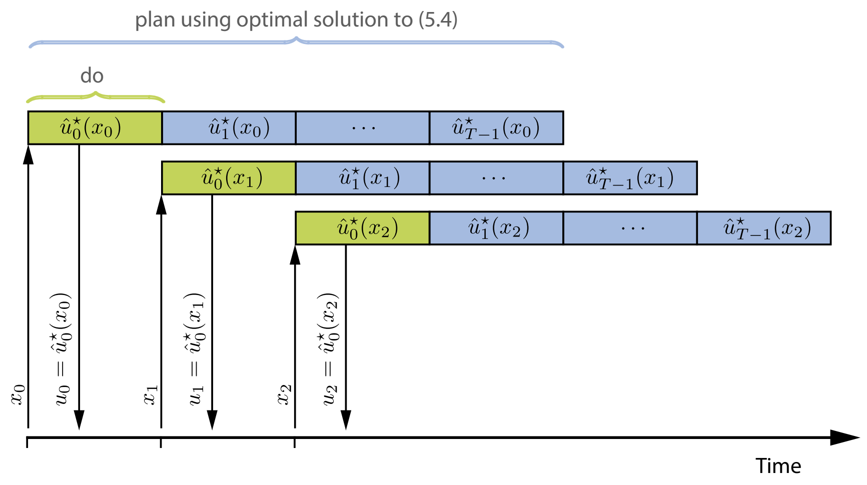

plan using optimal solution to (5.4)
do
$\hat{u}_0^\star(x_0)$
$\hat{u}_1^\star(x_0)$
$\cdots$
$\hat{u}_{T-1}^\star(x_0)$
$\hat{u}_0^\star(x_1)$
$\hat{u}_1^\star(x_1)$
$\cdots$
$\hat{u}_{T-1}^\star(x_1)$
$\hat{u}_0^\star(x_2)$
$\hat{u}_1^\star(x_2)$
$\cdots$
$\hat{u}_{T-1}^\star(x_2)$
x_0
$u_0 = \hat{u}_0^\star(x_0)$
x_1
$u_1 = \hat{u}_0^\star(x_1)$
x_2
$u_2 = \hat{u}_0^\star(x_2)$
Time

Figure 5.2: The receding-horizon principle. At time $t = 0$, we measure the true state x_0 and plan an optimal control $\{\hat{u}_0, \ldots, \hat{u}_{T-1}\}$ with respect to predicted states $\{\hat{x}_1, \ldots, \hat{x}_T\}$ over a horizon T. We implement only the first control, *i.e.* let $u_0 = \hat{u}_0^\star$. At time $t = 1$, we measure x_1 (which may differ from what we predicted it to be at $t = 0$ due to model inaccuracies) and re-plan over the prediction horizon. We implement the first move, wait until the next sampling instant and repeat.

■ **Example 5.1** To demonstrate the feedback aspect of receding-horizon control, we return to the quadcopter and consider a landing maneuver from a height of 40 m. We plan over a horizon of $T = 10$ steps, use a quadratic cost defined by $Q = Q_T = I$ and $R = 1$, and require that $|u_t| \leq g$. We do not impose any state constraints and do not add any terminal constraint since the drone reaches rest at zero well before the end of the horizon; see the full blue lines in Figure 5.3.

Assume that the true system is subject to an input disturbance so that its dynamics is given by $x_{t+1} = Ax_t + B(u_t + 1)$. If we still use the open-loop optimal control policy computed for the nominal dynamics, the landing fails miserably (dashed blue line). Since the open-loop policy is zero from time $t = 5$ and onward, there is nothing that compensates for the input disturbance. However, if we use a receding-horizon approach and re-plan the control action in each sampling instant, the quadcopter remains well-behaved and reaches a position close to zero, even though it plans using the nominal model. Although our receding-horizon controller can be improved in many ways, the example already demonstrates that it is much more robust than an open-loop policy. ■

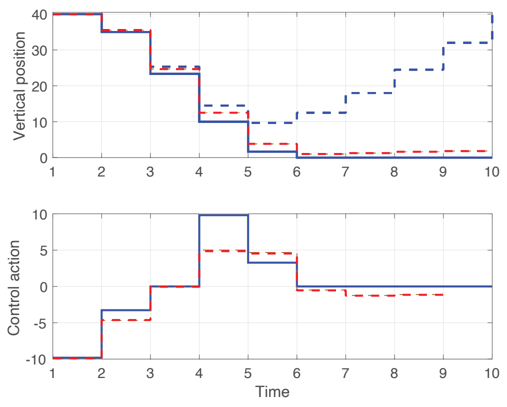


Figure 5.3: Open-loop optimal control on nominal model (full blue line) and actual system (dashed line). The receding-horizon policy (dashed red line) is much more robust on the actual system.

Model-predictive control: a first attempt

To get some experience of MPC on a more complex problem, we consider an aircraft control problem from [26]. As we will see, the MPC framework makes it easy to include multiple constraints on states and controls, and the controller can generate control inputs that respect the stated constraints without compromising the transient response. But we will also see that a poorly tuned MPC controller can go unstable, and may even fail to generate an input signal at all.

A simple Python implementation of the basic MPC control strategy for linear systems with simple bounds on states and controls is listed in Figure 5.4. Given the current state x, `mpcController` solves the planning horizon over T samples and returns the first control in the optimal input sequence. To solve the planning problem, it calls `computePredictedOptimalControls`, which formulates and solves (5.4) using the algebraic modeling language `cvxpy`. We have used a modeling language for clarity. In an actual implementation, it is typically more efficient to perform the transformation to a QP on standard form using the techniques described in Chapter 3, and only revise the matrices that depend on x_0 in each sampling instant.

```python
import numpy as np; import cvxpy as cp; import control as ctrl

def computePredictedOptimalControls(mpcProblemData, x0):
    # Extract system matrices
    A=mpcProblemData['A']; B=mpcProblemData['B']; C=mpcProblemData['C'];
    (n,m)=B.shape
    # Planning horizon
    T=mpcProblemData['T']
    # Cost function definition
    Q=mpcProblemData['Q']; R=mpcProblemData['R']; QT=mpcProblemData['QT']
    # Constraints
    ulim=mpcProblemData['ulim']; ylim=mpcProblemData['ylim']
    # Terminal constraint: MT*xT <= mT
    MT=mpcProblemData['MT']; mT=mpcProblemData['mT']

    # Set-up and solve planning problem
    x=cp.Variable((n,T+1)); u=cp.Variable((m,T))
    cost = 0; constr = [ x[:,[0]]== x0 ]
    for t in range(T):
        cost += cp.quad_form(x[:,t],Q)+ cp.quad_form(u[:,t],R)
        constr += [ x[:, t+1]==A@x[:,t] + B@ u[:, t] ]
        constr += [-ulim <= u[:,[t]], u[:,[t]] <= ulim]
        constr += [-ylim <= C@x[:,[t]], C@x[:,[t]] <= ylim]
    constr += [ MT@x[:,[T]] <= mT ]
    cost += cp.quad_form(x[:,T], QT)
    problem=cp.Problem(cp.Minimize(cost), constr)

    problem.solve()
    return (x.value, u.value)

def mpcController(mpcProblemData, x):
    # Plan optimal controls and states over the next T samples
    (xPred, uPred)=computePredictedOptimalControls(mpcProblemData, x)

    # Apply the first control action in the predicted optimal sequence
    return uPred[:,[0]]
```

Figure 5.4: Basic Python code for MPC controller: compute predicted optimal controls over the next T steps, but apply only the first move. The finite-horizon planning problem is readily formulated and solved using an algebraic modeling language such as `cvxpy`.

■ **Example 5.2 — Altitude control of an aircraft.** The following model describes the linearized dynamics of a Cessna citation aircraft flying with a speed of 128.2 m/s at an altitude of 5000 m.

$$\dot{x}(t) = \begin{pmatrix} -1.2822 & 0 & 0.98 & 0 \\ 0 & 0 & 1 & 0 \\ -5.4293 & 0 & -1.8366 & 0 \\ -128.2 & 128.2 & 0 & 0 \end{pmatrix} x(t) + \begin{pmatrix} -0.3 \\ 0 \\ -17 \\ 0 \end{pmatrix} u(t)$$

$$y(t) = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

The control input is the elevator angle, and the states represent the angle of attack, the pitch angle, the pitch rate, and the altitude, respectively; see Figure 5.5. The control input is limited to $\pm 15^\circ$ and has a slew-rate (rate-of-change) limit of $\pm 30^\circ$/s. For passenger comfort, the pitch angle (the first component of $y(t)$) is limited to $\pm 20^\circ$. Our aim is to design a controller which can perform swift changes in altitude while maintaining passenger comfort.

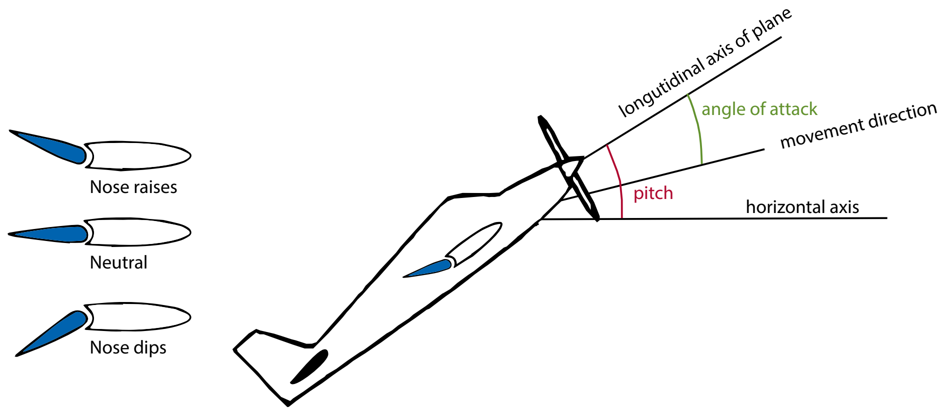


Figure 5.5: The altitude is the vertical distance between the center of mass of the plane and the ground; the angle between the horizontal axis and the longitudinal axis of the plane is called the pitch, while the angle between the moment direction and the longitudinal axis is called the angle of attack. The control problem is to manipulate the elevator surfaces on the wings in order to control the altitude, pitch rate, and angle of attack (positive angles are downward, leading the nose to dip).

We base our design on a discrete-time model sampled with $h = 0.25$s. To begin with, we disregard the slew-rate constraint on the actuator and consider a small descent of 10 m by letting $x_0 = (0, 0, 0, 10)$. A linear controller designed with the LQR methodology for penalty matrices $Q = I$ and $R = 10$ behaves well in simulations with the linear model, but oscillates widely when the actuator magnitude constraints are included in the simulations, see Figure 5.6 (left). In addition, the comfort constraint on the pitch rate is violated. To account for these constraints, we turn to an MPC controller that uses the same penalty matrices, a prediction horizon of $T = 10$, and incorporates the magnitude constraints on the control action and on the pitch angle. For simplicity, we set $Q_T = 0$. In contrast to the linear design, the MPC controller produces a well-behaved response that satisfies both control and state constraints; see Figure 5.6 (right). When we include the slew-rate constraints in the simulations, the (linear) LQR controller results in an unstable closed loop. The MPC design, on the other hand, deals well with the actuator limitations, see Figure 5.7 (left).

The MPC controller that we have simulated does not use a terminal penalty (i.e. $Q_T = 0$) or terminal constraint. Such a design needs a relatively long prediction horizon to have near-optimal performance. It should therefore come as no surprise that when we reduce the prediction horizon to $T = 4$, the closed-loop is no longer stable, but the aircraft goes into an undamped oscillation;

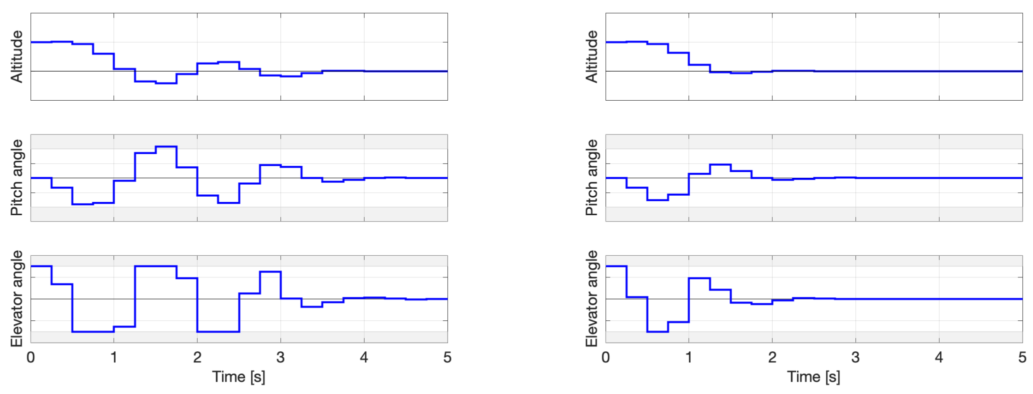


Figure 5.6: The optimal LQ controller, which disregards actuator and state constraints, behaves well in linear operation, but is unable to handle a descent of 10 meters in altitude without causing large oscillations (left). The MPC controller, on the other hand, acounts for the constraints in the design and is able to execute a well-damped descent while respecting control and state limits.

see Figure 5.7 (right). Even worse, if we request a altitude change of 30 meters, the planning problem becomes infeasible and the receding-horizon policy is unable to suggest a control action. ■

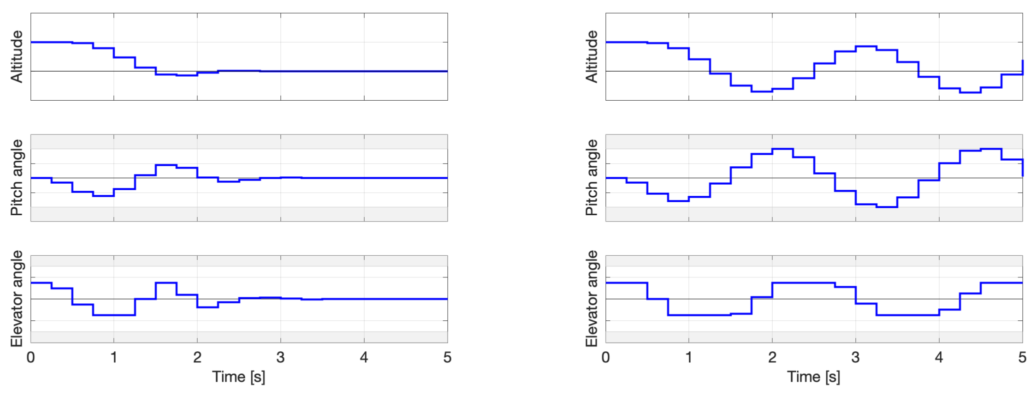


Figure 5.7: The MPC controller also handles slew-rate constraints on the elevators (left). However, when we reduce the prediction horizon to $T = 4$, the controller is no longer able to stabilize the altitude (right). In this example, this problem can be remedied by adding the appropriate terminal constraint and terminal penalty (not shown).

The example indicates that MPC has a great potential to deal with constrained systems, as long as the controller is properly tuned. But there are also several things that can go wrong: the planning problem can become infeasible, and the closed-loop system may become unstable even when it plans using an exact model of the true system dynamics. To understand how we can analyze the potential instability problems in model predictive control, we will try to gain insight by temporarily removing the constraints and study the simpler linear-quadratic receding horizon control problem.

5.2 Linear-quadratic receding-horizon control

It is easy to get blinded by how the MPC controller works. The fact that it solves an optimization problem to plan the control actions in every sampling interval appears, at first, to put it beyond the reach of standard analysis. But the essential property is not *how* the controller works, but rather *what* it does. The MPC controller controls the system to attain a minimal value of a cost function. We will show that if we construct the cost function appropriately, then driving the state to a point where the cost is minimized also guarantees asymptotic stability of the closed loop.

To make this point, we will briefly study the properties of the linear-quadratic receding-horizon controller. Just like the MPC controller, this controller solves a finite-horizon planning problem at every sampling instant. But unlike more complete MPC, it does not have any state or control constraints. Instead, it simply attempts to solve the infinite-horizon LQR problem

$$\begin{array}{ll} \text{minimize} & \sum_{t=0}^{\infty} x_t^\top Q x_t + u_t^\top R u_t \\ \text{subject to} & x_{t+1} = Ax_t + Bu_t \end{array}$$

using a receding-horizon policy. Specifically, the controller relies on the planning problem

$$\begin{array}{lll} \text{minimize} & \sum_{k=0}^{T-1} \hat{x}_k^\top Q \hat{x}_k + \hat{u}_k^\top R \hat{u}_k + \hat{x}_T^\top Q_T \hat{x}_T & \\ \text{subject to} & \hat{x}_{k+1} = A\hat{x}_k + B\hat{u}_k & k = 0, \dots, T-1 \\ & \hat{x}_0 = x_t & \end{array} \tag{5.6}$$

Just like before, the decision variables $\{\hat{x}_k\}$ represent the predicted optimal state trajectory from $\hat{x}_0 = x_t$ at time t and T steps into the future, while $\{\hat{u}_k\}$ represents the associated optimal control sequence. Once the planning problem (5.6) is solved, the controller applies

$$u_t = u_0^\star \tag{5.7}$$

The control signal is held constant until the next sampling instant when the procedure is repeated.

In Chapter 4, we have shown how the finite-horizon LQR problem (5.6) can be solved using dynamic programming. By Theorem 4.1.1, the optimal control sequence $\{\hat{u}_k^\star\}$ is on the form

$$\hat{u}_k^\star = -L_k \hat{x}_k^\star \qquad k = 0, \dots, T-1$$

where

$$L_k = (R + B^\top P_{k+1} B)^{-1} B^\top P_{k+1} A$$

and the matrix sequence $\{P_k\}$ are computed via the Riccati recursion

$$P_{k-1} = Q + A^\top P_k A - A^\top P_k B (R + B^\top P_k B)^{-1} B^\top P_k A \tag{5.8}$$

initiated with $P_T = Q_T$. The receding-horizon control (5.7) can therefore be expressed as

$$u_t = \hat{u}_0^\star = -L_0 \hat{x}_0^\star = -L_0 x_t.$$

Thus, although we formally re-plan the control actions over a finite horizon in every sample, we always apply the same linear feedback $u_t = -L_0 x_t$ based on the current state. Unfortunately, as the next example shows, this policy does not necessarily yield a stable closed loop.

■ **Example 5.3 — Instability of linear-quadratic RHC.** Consider the discrete-time linear system

$$A = \begin{pmatrix} 5/4 & -1/4 \\ 1/4 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} 1/4 \\ 1/4 \end{pmatrix} \tag{5.9}$$

and the receding-horizon problem given by $Q = Q_T = I$, $R = 1$ and $T = 1$. The one-step optimal receding-horizon control is

$$u_t = -L_0 x_t = -\begin{pmatrix} 1/3 & 1/6 \end{pmatrix} x_t$$

which results in the closed-loop system

$$x_{t+1} = (A - BL_0)x_t = \begin{pmatrix} 7/6 & -7/24 \\ 1/6 & 23/24 \end{pmatrix} x_t$$

The closed-loop system matrix has two complex-conjugate eigenvalues with magnitude $\sqrt{7/6} > 1$, and is therefore not Schur stable. Admittedly, a one-step prediction horizon is very short and it is natural to ask if a larger horizon would yield a stable closed-loop. For this system, receding-horizon control will yield a stable closed loop for $T = 3$, result in closed-loop instability for T in the range $4 - 7$, and then ensure stability for larger horizons. ■

The example indicates that it can be difficult to characterize the relationship between prediction horizon and closed-loop stability. Although we can expect to recover asymptotic stability for sufficiently long prediction horizons, this is not a very attractive possibility. Increasing the horizon length results in a larger planning problem that takes a longer time to solve. However, we should also be able to guarantee closed-loop stability using the proper choice of terminal cost. Recall that the value function of the infinite-horizon linear-quadratic control problem is $v(x) = x^\top Px$ where

$$P = Q + A^\top PA - A^\top PB(R + B^\top PB)^{-1}B^\top PA. \tag{5.10}$$

If we use this value function as terminal cost in (5.6), *i.e.*, let $Q_T = P$ where P satisfies (5.10), then the receding-horizon planning problem is an *exact* reformulation of the infinite-horizon linear-quadratic control problem. The computed control signal (5.7) will therefore be identical to the the infinite-horizon LQR policy. Since that policy results in an asymptotically stable closed loop, so does the receding-horizon control. The next result formalizes our intuitive argument.

Proposition 5.2.1 Consider the linear system $x_{t+1} = Ax_t + Bu_t$ with (A, B) stabilizable. Let $Q \succeq 0, R \succ 0$ and $(A, Q^{1/2})$ be detectable. Then, if $Q_T = P$ where

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

the receding-horizon control (5.6), (5.7) results in an asymptotically stable closed-loop system. Moreover, the control is equivalent to a linear state feedback $u_t = -Lx_t$ where L satisfies

$$L = (R + B^\top PB)^{-1}B^\top PA$$

Proof. Letting $Q_T = P$ in (5.8) yields $P_t = P$ for all times. Hence, the optimal control equals the infinite-horizon optimal LQ solution, which, by Theorem 4.2.3, is guaranteed to yield an asymptotically closed-loop under the given conditions. ■

■ **Example 5.4 — Terminal penalty and stability of linear-quadratic RHC.** We can now return to the set-up in Example 5.3, and set Q_T equal to the stationary Riccati solution for the linear system (5.9) and the cost matrices $Q = I$ and $R = 1$. We then find

$$Q_T = P = \begin{pmatrix} 22.4351 & -26.6541 \\ -26.6541 & 48.6485 \end{pmatrix}, \qquad L_0 = \begin{pmatrix} 0.0266 & 2.7297 \end{pmatrix}$$

One can verify that $A - BL_0$ is Schur stable, so the closed-loop system is asymptotically stable. ■

Proposition 5.2.1 is useful since it defines an easily computable terminal cost that guarantees closed-loop stability of the receding-horizon control law. But it is also restrictive since it only supports a single option for the terminal cost and relies on reducing the receding-horizon policy to a linear-quadratic regulator. As we will demonstrate next, it is possible to perform a direct analysis of the closed-loop system using Lyapunov theory. This approach will reveal a full family of terminal costs that can be used to ensure closed-loop stability.

Recall that the stability proof for LQR uses the value function $v(x) = x^\top Px$ as Lyapunov function. When we use $v(x)$ as terminal cost in (5.6), the linear-quadratic receding-horizon policy (5.6), (5.7) generates the same control actions as LQR, so we should be able to use the same Lyapunov function to analyze its properties. Moreover, with this terminal cost, the infinite-horizon cost-to-go is identical to the optimal value of the planning problem

$$J^\star(x) = \begin{array}{lll} \underset{\substack{\hat{u}_0,\dots,\hat{u}_{T-1}\\ \hat{x}_0,\dots,\hat{x}_T}}{\min} & \sum_{k=0}^{T-1} \hat{x}_k^\top Q\hat{x}_k + \hat{u}_k^\top R\hat{u}_k + \hat{x}_T^\top Q_T\hat{x}_T & \\ \text{subject to} & \hat{x}_{k+1} = A\hat{x}_k + B\hat{u}_k & k = 0,1,\dots,T-1 \\ & \hat{x}_0 = x & \end{array} \tag{5.11}$$

with $Q_T = P$. We refer to $J^\star(x)$ as the *predicted cost* of the receding-horizon policy (5.6), (5.7). The next theorem is the result of a stability analysis that uses $J^\star(x)$ as Lyapunov function.

Theorem 5.2.2 Consider the system $x_{t+1} = Ax_t + Bu_t$ with (A,B) stabilizable. If $Q \succeq 0$ with $(Q^{1/2}, A)$ detectable, $R \succ 0$ and $Q_T = P$ where P satisfies the Lyapunov equation

$$(A - B\tilde{L})^\top P(A - B\tilde{L}) - P + (Q + \tilde{L}^\top R\tilde{L}) = 0$$

for some $\tilde{L}$ such that $A - B\tilde{L}$ is Schur stable, then the receding horizon control (5.6), (5.7) results in an asymptotically stable closed loop for all $T \geq 1$.

Proof. To ease notation, we define $q(x,u) = x^\top Qx + u^\top Ru$ and $q_T(x) = x^\top Q_T x$, and note that Theorem 2.2.6 implies that the suggested terminal cost matrix Q_T is positive semi-definite. Let $\{\hat{x}_k^\star\}$ and $\{\hat{u}_k^\star\}$ be the optimal solution to (5.11) with $x = x_t$ at time t. Then

$$J^\star(x_t) = \sum_{k=0}^{T-1} q(\hat{x}_k^\star, \hat{u}_k^\star) + q_T(\hat{x}_T^\star).$$

Since $J^\star(x_t) \geq 0$ for all x_t, and $J^\star(0) = 0$, the predicted cost is a positive semi-definite function. We have shown in Section 3.2 that $J^\star$ is a quadratic function of x_t. We will now show that it can be used as a Lyapunov function for the receding-horizon control law.

Assume that we apply the first element of the optimal predicted control sequence, i.e. let $u_t = \hat{u}_0^\star$. If our model agrees with the actual system, this leads us to $x_{t+1} = Ax_t + B\hat{u}_0^\star = \hat{x}_1^\star$. We would now like to show that it is possible to find a solution to (5.11) from $x = x_{t+1}$ so that $J^\star(x_{t+1}) \leq J^\star(x_t)$. To that end, consider the control sequence $\{\hat{u}_1^\star, \dots, \hat{u}_{T-1}^\star, \tilde{u}_T\}$. This control yields the predicted state trajectory $\{\hat{x}_1^\star, \hat{x}_2^\star, \dots, \hat{x}_T^\star, \tilde{x}_{T+1}\}$ where $\tilde{x}_{T+1} = A\hat{x}_T^\star + B\tilde{u}_T$. The associated cost is

$$\begin{aligned} J(x_{t+1}) &= \sum_{k=1}^{T-1} q(\hat{x}_k^\star, \hat{u}_k^\star) + q(\hat{x}_T^\star, \tilde{u}_T) + q_T(\tilde{x}_{T+1}) = \\ &= -q(\hat{x}_0^\star, \hat{u}_0^\star) + \underbrace{q(\hat{x}_0^\star, \hat{u}_0^\star) + \sum_{k=1}^{T-1} q(\hat{x}_k^\star, \hat{u}_k^\star) + q_T(\hat{x}_T^\star)}_{J^\star(x_t)} - q_T(\hat{x}_T^\star) + q(\hat{x}_T^\star, \tilde{u}_T) + q_T(\tilde{x}_{T+1}) \\ &= -q(\hat{x}_0^\star, \hat{u}_0^\star) + J^\star(x_t) - q_T(\hat{x}_T^\star) + q(\hat{x}_T^\star, \tilde{u}_T) + q_T(\tilde{x}_{T+1}). \end{aligned}$$

We now let $\tilde{u}_T = -\tilde{L}\hat{x}_T^\star$, so that $\tilde{x}_{T+1} = (A - B\tilde{L})\hat{x}_T^\star$ and observe that the last three terms satisfy

$$-q_T(\hat{x}_T^\star) + q(\hat{x}_T^\star, \tilde{u}_T) + q_T(\tilde{x}_{T+1}) = (\hat{x}_T^\star)^\top(-Q_T + Q + \tilde{L}^\top R\tilde{L} + (A - B\tilde{L})^\top Q_T(A - B\tilde{L}))\hat{x}_T^\star$$

Therefore, if we pick Q_T and $\tilde{L}$ as suggested in the theorem, these terms vanish, and it holds that

$$J(x_{t+1}) = J^\star(x_t) - q(x_0^\star, u_0^\star).$$

Since $\tilde{u}_T$ was chosen without minimizing $J(x_{t+1})$, it holds that $J^\star(x_{t+1}) \leq J(x_{t+1})$, and thus that

$$J^\star(x_{t+1}) \leq J^\star(x_t) - q(\hat{x}_0^\star, \hat{u}_0^\star) = J^\star(x_t) - \left(x_t^\top Q x_t + u_t^\top R u_t\right)$$

where we have used that $\hat{x}_0^\star = x_t$ and $\hat{u}_0^\star = u_t$. By summing both sides of the inequality

$$\lim_{k\to\infty} \sum_{t=0}^{k} \left(x_t^\top Q x_t + u_t^\top R u_t\right) = J^\star(x_0) - \lim_{k\to\infty} J^\star(x_k) \leq J^\star(x_0)$$

where the inequality follows since $J^\star$ is non-negative for all arguments. Since (A,B) is stabilizable, $J^\star$ is bounded, and Cauchy's convergence criterion implies that

$$\lim_{t\to\infty} \left(x_t^\top Q x_t + u_t^\top R u_t\right) = 0.$$

As both terms inside the parentheses are non-negative, they must both tend to zero. Since $R \succ 0$, $\lim_{t\to\infty} u^\top R u = 0$ implies that $\lim_{t\to\infty} u_t = 0$. Similarly, $x_t^\top Q x_t \to 0$ implies that $Q^{1/2} x_t \to 0$ and, by the detectability assumption on $(A, Q^{1/2})$, that $\lim_{t\to\infty} x_t = 0$. ■

Receding-horizon linear quadratic control is somewhat artificial, since the optimal infinite-horizon feedback policy is already known and can be computed off-line. However, this will no longer be the case when we introduce state and control constraints. Yet, as we will show shortly, the line of argument used in the proof of Theorem 5.2.2 can be extended to the constrained case.

5.3 Stability and recursive feasibility of MPC

The approach that we have used in Theorem 5.2.2 can be extended to prove stability of model predictive control of systems constrained systems. However, to do so, we need to address two new challenges that appear due to the presence of constraints. First, we must ensure that the planning problem remains feasible, *i.e.* that it will always admit at least one solution. Second, since the true value function is difficult to compute, we need to find an alternative terminal cost that ensures closed-loop stability without significantly increasing the complexity of the planning problem. Let us address these problems one at a time.

Initially feasible and recursively feasible states

A model-predictive controller can only operate in states for which the planning problem (5.4) has a feasible solution. These are the states from which it is possible to reach X_T in T steps while the predicted controls and states satisfy their respective constraints. We call such states *initially feasible*. However, we have seen in Example 5.2 that even if the initial state is feasible and we are able to solve the planning problem for some time, we may still end up in a state for which the planning problem has no solution. To avoid such a situation, we need to ensure that the planning problem remains feasible for all future states that will be encountered under the model-predictive control policy. The next example helps to build additional intuition.

■ **Example 5.5 — Initial feasibility does not imply recursive feasibility.** Consider the vertical quadcopter dynamics studied in earlier examples, and assume that the system is subject to constraints $|u_t| \leq 1$ and $\|x_t\|_\infty \leq 10$ for all t. The system is controlled using an MPC controller with

$$Q = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad R = 1, \quad T = 3$$

but no terminal constraint or terminal cost (*i.e.* we let $Q_T = 0$ and $X_T = \mathbb{R}^2$). Clearly, we must require that $\|x_0\|_\infty \leq 10$, *i.e.*, that the initial state satisfies the constraints in (5.4). However, as shown in Figure 5.8 (left), the set of initially feasible states is even smaller. Intuitively, if the system is close to the constraint boundary and has a high outward velocity, then will will not have sufficient control authority to reverse the motion. By the same intuition, it is clear that even if the initial state is feasible, this does not necessarily guarantee that all future states will be feasible. In this example, the initial value $x_0 = \begin{pmatrix} -9 & 7 \end{pmatrix}$ leads to a feasible planning problem with $u_0 = -1$. Applying this control yields $x_1 = \begin{pmatrix} -2.5 & 6 \end{pmatrix}$, which does not belong to the set of initially feasible states. Hence, we cannot solve the MPC planning problem from this state.

In Chapter 2, we have discussed maximal control invariant sets. These are the set of states for which there exists an admissible control so that the constraints are satisfied for all future times. As shown in Figure 5.8 (right), x_0 does not belong to the maximal control invariant set of this system, so there is in fact no controller (and, in particular, no MPC controller) that is able to satisfy the constraints for all future times. We should, therefore, not be surprised that we lose feasibility. ■

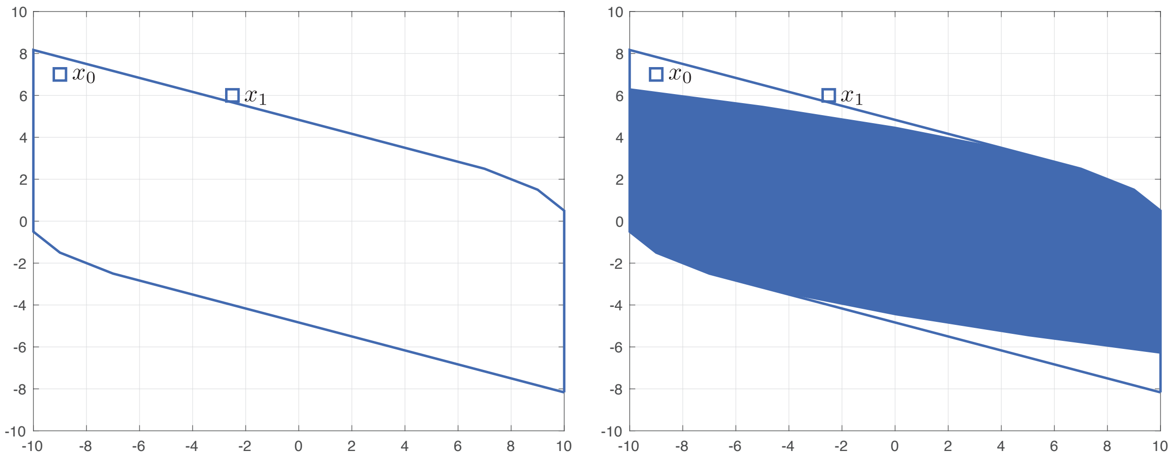


Figure 5.8: Left figure shows the set of initial values for which the MPC planning problem in Example 5.5 admits a feasible solution. Even for initial states in this set, the MPC planning problem can become infeasible in the future. As the right figure shows, x_0 lies outside the maximal control invariant set for the system (filled blue), so no admissible controller can handle this initial value.

The property that the MPC planning problem remains feasible for all future times is referred to as *recursive feasibility*, defined next.

Definition 5.3.1 — Recursive feasibility. The MPC problem (5.4) is *recursively feasible* if the existence of a feasible solution with initial state x_t implies that the problem (5.4) is feasible also when initialized with the next state x_{t+1} under the control law (5.5).

The next result shows that a control invariant terminal set guarantees recursive feasibility.

Proposition 5.3.1 If the terminal set X_T is control invariant, then the MPC problem (5.4) is recursively feasible.

Proof. Let us consider an initial state x_0 from which the MPC planning problem is feasible. Thus, x_0 allows us to solve (5.4) and find optimal controls $\{\hat{u}_0^\star, \dots, \hat{u}_{T-1}^\star\}$, with an associated predicted state evolution $\{\hat{x}_0^\star, \dots, \hat{x}_T^\star\}$. Since $x_1 = Ax_0 + Bu_0^\star = x_1^\star$, we can apply the control sequence $\{\hat{u}_1^\star, \dots, \hat{u}_{T-1}^\star, \tilde{u}_T\}$ from x_1 resulting in the predicted state trajectory $\{\hat{x}_1^\star, \dots, \hat{x}_T^\star, \tilde{x}_{T+1}\}$ where $\tilde{x}_{T+1} = A\hat{x}_T^\star + B\tilde{u}_T$. Since $\hat{x}_T^\star \in X_T$ and X_T is control invariant, there exists an admissible $\tilde{u}_T$ which brings $\tilde{x}_{T+1} = A\hat{x}_T^\star + B\tilde{u}_T$ into X_T. Using such a $\tilde{u}_T$ in the proposed control sequence (and the corresponding $\tilde{x}_{T+1}$ as the final element of the predicted state sequence) yields a feasible solution to (5.4). Recursive feasibility is established.

As we have discussed in Chapter 2, many control invariant sets can be represented as linear inequalities and are therefore easy to incorporate in the MPC planning problem. This includes the maximal control invariant set itself, the maximal invariant set for a given state feedback law, and the origin. Since the MPC controller has to drive the state into the terminal state at the end of the horizon, a smaller terminal set results in a smaller set of recursively feasible states. We will discuss this in more detail once that we have derived conditions for asymptotic stability of the closed loop.

Asymptotic stability of the model predictive controller in closed-loop

Now that we know how to ensure recursive feasibility, we can guarantee closed-loop stability using essentially the argument that we used for linear-quadratic receding-horizon control.

Theorem 5.3.2 — MPC stability. Consider the linear system $x_{t+1} = Ax_t + Bu_t$ subject to constraints $x_t \in X$ and $u_t \in U$ for all $t \geq 0$ under the model predictive control (5.4),(5.5). If the following conditions hold:

(a) $Q \succeq 0$ with $(Q^{1/2}, A)$ detectable, $R \succ 0$ and $Q_T \succ 0$;
(b) X, U and X_T are closed and contain 0 in their interior;
(c) X_T is control invariant under the given dynamics and constraints; and
(d) For every $x \in X_T$, there exists $\tilde{u} \in U$ such that $Ax + B\tilde{u} \in X_T$ and

$$(Ax + B\tilde{u})^\top Q_T (Ax + B\tilde{u}) - x^\top Q_T x + x^\top Q x + \tilde{u}^\top R \tilde{u} \leq 0$$

then $\lim_{t\to\infty} x_t = 0$ from all initial values x_0 for which (5.4) admits a feasible solution.

Proof. By Proposition 5.3.1, Condition (c) implies recursive feasibility. To prove stability, we will use a Lyapunov function argument based on the predicted cost

$$J^\star(x) = \begin{array}{lll} \min\limits_{\substack{\hat{u}_0,\dots,\hat{u}_{T-1} \\ \hat{x}_0,\hat{x}_1,\dots,\hat{x}_T}} & \sum\limits_{k=0}^{T-1} \hat{x}_k^\top Q \hat{x}_k + \hat{u}_k^\top R \hat{u}_k + \hat{x}_T Q_T \hat{x}_T & \\ \text{subject to} & \hat{x}_{k+1} = A\hat{x}_k + B\hat{u}_k & k = 0,\dots,T-1 \\ & \hat{x}_k \in X,\ \hat{u}_k \in U & k = 0,\dots,T-1 \\ & \hat{x}_T \in X_T & \\ & \hat{x}_0 = x & \end{array} \tag{5.12}$$

By the same arguments as in the proof of Theorem 5.2.2, $J^\star(x)$ is a positive semi-definite function. Moreover, by Theorem 5.4.1, $J^\star(x)$ is continuous, and therefore a valid Lyapunov function candidate. Let x_t admit a feasible solution to (5.4) at time t and let $\{\hat{u}_0^\star, \dots, \hat{u}_{T-1}^\star\}$ and $\{\hat{x}_1^\star, \dots, \hat{x}_T^\star\}$ be the corresponding optimal predicted control sequence and state trajectory, respectively. By the same arguments used in Proposition 5.3.1, there is a $\tilde{u}_T$ such that $\{\hat{u}_1^\star, \hat{u}_2^\star, \dots, \hat{u}_{T-1}^\star, \tilde{u}_T\}$ is feasible when evaluating the predicted cost for the system state $x_{t+1} = Ax_t + Bu_0^\star$ at time $t+1$. Letting $q(x,u) = x^\top Q x + u^\top R u$ and $q_T(x) = x^\top Q_T x$, the associated cost can be written as

$$\begin{aligned} J(x_{t+1}) &= \sum_{k=1}^{T-1} q(\hat{x}_k^\star, \hat{u}_k^\star) + q(\hat{x}_T^\star, \tilde{u}_T) + q_T(A\hat{x}_T^\star + B\tilde{u}_T) = \\ &= J^\star(x_t) - q(\hat{x}_0^\star, \hat{u}_0^\star) - q_T(\hat{x}_T^\star) + q(\hat{x}_T^\star, \tilde{u}_T) + q_T(A\hat{x}_T^\star + B\tilde{u}_T) \end{aligned}$$

Since $\hat{x}_T^\star \in X_T$, (c) and (d) imply that we can find $\tilde{u}_T$ such that the last terms are negative, *i.e.*

$$J(x_{t+1}) \leq J^\star(x_t) - q(\hat{x}_0^\star, \hat{u}_0^\star) = J^\star(x_t) - q(x_t, u_t)$$

Since $\tilde{u}_T$ was chosen without minimizing the predicted cost, $J^\star(x_{t+1}) \leq J(x_{t+1})$ and therefore

$$J^\star(x_{t+1}) \leq J^\star(x_t) - q(x_t, u_t).$$

Asymptotic stability now follows by summing both sides of the inequality over time and proceeding just as in the proof of Theorem 5.2.2. ■

Condition (c) demands that the terminal set is control invariant, while (d) requires that $x^\top Q_T x$ is an upper bound on the infinite horizon LQR cost for all initial states in the terminal set. Together, the two conditions are quite complex to deal with in their full generality. The simplest solution is to let $X_T = \{0\}$, in which case $\tilde{u} = 0$ satisfies condition (d) for all Q_T. But this terminal set is very small and requires large prediction horizons to avoid a small operating region of the MPC controller. The largest terminal set that we can use is the maximal control invariant set itself. With this terminal set, condition (d) is difficult to guarantee, since it requires a quadratic upper bound on the cost-to-go for a control policy that makes the state stay in X_T. We have neither given an explicit construction of such a policy nor discussed tools that can estimate its infinite-horizon cost. A good compromise is to let $\tilde{u}_T = -\tilde{L}\tilde{x}_T$ for some state feedback gain $\tilde{L}$. One can then let X_T be the maximal positive invariant set of $x_{t+1} = (A - B\tilde{L})x_t$ under the given constraints, and find the terminal weight Q_T by solving a Lyapunov equation as in Theorem 5.2.2.

Proposition 5.3.3 Consider the discrete-time linear system $x_{t+1} = Ax_t + Bu_t$ under the constraints $x_t \in X$ and $u_t \in U$ for all $t \geq 0$. Let u_t be defined by the model predictive control law (5.4), (5.5). If conditions (a) and (b) of Theorem 5.3.2 hold and

(c') $X_T \subseteq X \cap \{x \mid -\tilde{L}x \in U\}$ is positively invariant under $x_{t+1} = (A - B\tilde{L})x_t$

(d') $Q_T = P$ where P satsifies the Lyapunov equation

$$(A - B\tilde{L})^\top P(A - B\tilde{L}) - P = -(Q + \tilde{L}^\top R\tilde{L})$$

then $\lim_{t\to\infty} x_t = 0$ from all initial values x_0 for which (5.4) admits a feasible solution.

A natural choice for $\tilde{L}$ is to use the infinite-horizon linear-quadratic controller for the weight matrices Q and R. As shown in Equation (4.12), the Q_T proposed in (d') is then equivalent to the solution to the algebraic Riccati equation (4.9) for the given system and cost matrices.

How terminal constraints and horizon length impact recursive feasibility and optimality

The set of recursively feasible states on an MPC controller with a control invariant terminal set X_T and planning horizon T is simply the set of initial states from which we are able to reach X_T in T steps while respecting the constraints in x and u. In other words, it is the T-step controllable set of X_T. This set becomes larger if we choose a larger terminal set or a longer planning horizon.

The smallest valid choice for a terminal set is the origin, $X_T = \{0\}$, and the largest choice is the maximal control invariant set for the given dynamics and constraints. The positively invariant set of the infinite horizon LQR controller is often able to strike a nice balance between the two, and has some interesting properties that we will explore in Proposition 5.3.4 below. But let us first illumuniate the discussion with an example.

■ **Example 5.6 — The terminal set impacts the set of recursive feasible states.** Let us return to the constrained quadcopter dynamics studied in Example 5.5 and add a terminal constraint to the planning problem. Figure 5.9 (left) shows the maximal control invariant set along with the set of recursive feasible states for $X_T = \{0\}$ (in blue), and for X_T being the invariant set of the LQ-optimal state feedback for $Q = I$ and $R = 1$ (grey). Clearly, the larger terminal set gives a larger set of initially feasible states. Similarly, Figure 5.9 (right) shows that a larger horizon gives a larger set of initially feasible states; in fact, for large values of T, we recover the maximal control invariant set. ■

When we increase the control horizon, we expect that the set set of recursively feasible states becomes larger and that the MPC control approaches the infinite-horizon optimal (constrained) control. Recall that we introduced MPC by dividing an infinite-horizon optimal control problem into a T-step constrained LQR problem and an infinite-horizon tail problem which we only can

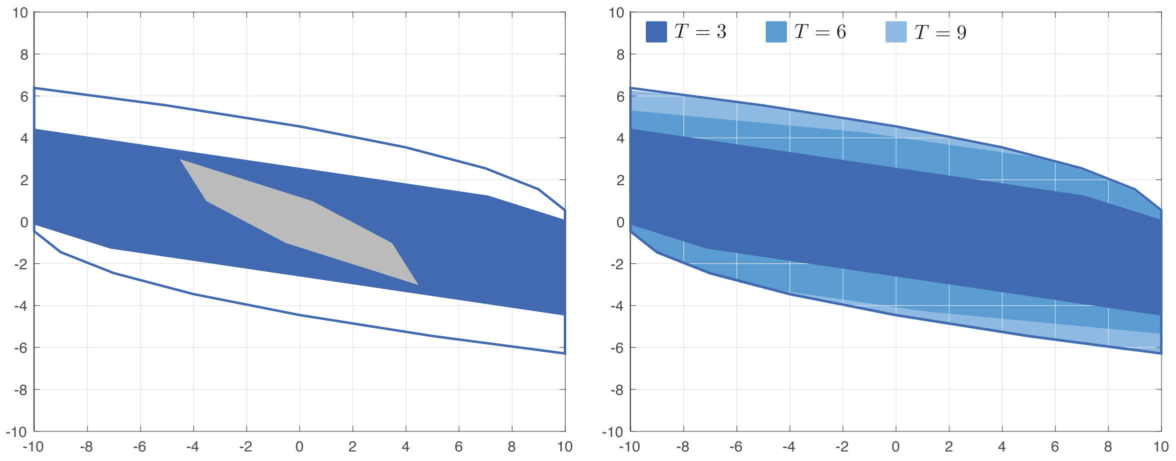


Figure 5.9: The left figure shows the set of recursively feasible states when we use the origin as terminal set (in gray), and when we use the maximal invariant set of the LQ-optimal state feedback as terminal set (in blue). The smaller terminal set gives a smaller set of recursively feasible states. The right figure shows how a longer prediction horizon T allows us to enlarge the set of recursively feasible states, approaching the maximal control invariant set of the system.

solve in the absence of constraints. For short planning horizons, we should expect some degree of suboptimality, since we "glue" the two solutions together by forcing x_T into X_T. At the same time, if the constraint sets X, X_T and U contain origin in their interior, and if the stage cost vanishes only for $(x,u) = (0,0)$, then the finite-horizon optimal control will drive $x_T \to 0$ as $T \to \infty$. Hence, if the horizon is long enough, we should expect that $x_T \in X_T$ even if this is not explicitly enforced, which means that no optimality is lost. One can prove that this intuition is indeed true.

Proposition 5.3.4 Consider the same set-up as in Proposition 5.3.3 and let (a), (b), (c') and (d') hold with $\tilde{L}$ chosen as infinite-horizon LQ-optimal state feedback gains defined by matrices Q and R. Then there exists a finite horizon T_∞, dependent on x_0, with the property that whenever $T \geq T_\infty$, the sequence $\{\hat{u}_0^\star, \dots, \hat{u}_{T-1}^\star\}$ that attains the minimum cost in (5.4) is equal to the first T elements of the infinite sequence $\{u_0^\star, u_1^\star, \dots\}$ that minimizes the infinite-horizon constrained LQR cost (5.1).

Proof. First note that in the absence of constraints, the optimal solution to (5.1) is the infinite-horizon LQ-optimal control $u_t = -Lx_t$ where L is given by (4.10). Its infinite-horizon cost-to-go is equal to $V(x) = x^\top Px$, where P is the solution to the associated Riccati equation (4.9); V is also a Lyapunov function for the closed-loop system under the LQ-optimal control, so every level set of V is invariant under $x_{t+1} = (A - BL)x_t$. Since the origin lies in the interior of both X and U, there exists a level set $\mathcal{L}_V(\alpha)$ of V fully contained in X in which the LQ-optimal control is admissible.

Now, since the system is reachable, the infinite-horizon optimal controller (5.1) will drive the state to the origin. This means that there exists some T_∞ so that the optimal control will drive the state to $\mathcal{L}_V(\alpha)$. Its cost-to-go at this point will be equal to $V(x)$, since the optimal control will be the linear LQ-optimal controller for all future times. Hence, with $T \geq T_\infty$, the proposed model-predictive controller solves the same problem as the infinite-horizon optimal controller. Equivalence of the two control sequences follows since they are both unique as the objective of the two control problems are strongly convex in u. ■

The negative aspect of using a long horizon is the computational cost of solving a large optimization problem at every sample. In practice, we may therefore need to use T smaller than T_∞. As the next example shows, this does not necessarily mean a large loss in performance.

■ **Example 5.7** Consider the system

$$x_{t+1} = \begin{pmatrix} 0.5 & -1.25 \\ 1.0 & 0.0 \end{pmatrix} x_t + \begin{pmatrix} 1 \\ 0 \end{pmatrix} u_t$$

under constraints $\|x_t\|_\infty \leq 5$, and $|u_t| \leq 1$. We let the stage-cost be defined by $Q = I$ and $R = \rho I$. We apply Proposition 5.3.3 and use the infinite-horizon LQR cost-to-go function as terminal penalty, and the maximal invariant set for the closed-loop dynamics of the corresponding LQR-optimal controller as terminal set. Figure 5.10 demonstrates how the predicted trajectories for different horizon lengths differ. When the horizon is short, one has to accept to deviate from the infinite-optimal control to drive the system into the terminal state at the end of the prediction horizon. As shown in the same figure (right), the two control laws nevertheless generate the same closed-loop trajectories. The reason is that the initial parts of the predicted optimal trajectories agree, and since only the first predicted action is used in each sample, the closed-loop trajectories will also agree. ■

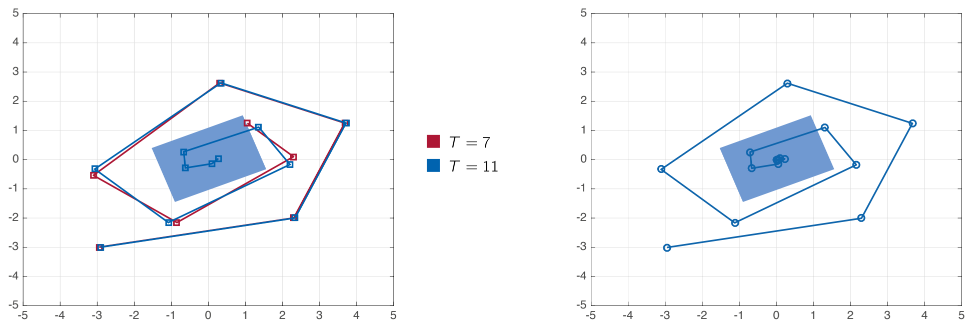


Figure 5.10: Predicted trajectories for two different horizon lengths (left), and corresponding closed-loop trajectories (right). The need to drive the terminal state into the terminal state (light blue polyhedron) requires the control with short horizon to deviate from the infinite-horizon optimal controller. Still, the initial control signals are the same, and the closed loop trajectories coincide.

5.4 The MPC policy is a static nonlinear state feedback law

The model-predictive control policy is defined through the solution of an optimization problem that depends on the current process state. This makes it different from traditional approaches that define the control action as an explicit function of the state. Nevertheless, we have shown that the linear-quadratic receding-horizon controller, which also solves an optimization problem in every sample, is actually a linear state feedback. In this section, we will show that the model predictive control law for linear systems defines a static nonlinear mapping from the process state to the control action. More specifically, the control law is piecewise linear. This fact can be proven using properties of multiparametric quadratic programs reviewed in Appendix C.

Theorem 5.4.1 The MPC control law defined by (5.4), (5.5) is a continuous function of x_t. It is piecewise affine, in the sense that there exists a partition of the feasible set of (5.4) into a finite number of polyhedra $\{R_1, \ldots, R_K\}$ such that u_t is an affine function of x_t in each R_k. Moreover, the predicted cost (5.12) is a continuous and piecewise quadratic function of the initial state, defined on the same polyhedral partition as the control law.

Proof. Recall from Chapter 3 that the planning problem (5.4) can be formulated in condensed form and described by a quadratic program

$$\begin{array}{ll} \text{mininimze} & z^\top Pz + 2q^\top z + r \\ \text{subject to} & Az \leq b \end{array}$$

where q, r, and b depend on the initial value response (and therefore on x_t). To simplify the dependency on x_t, we perform a variable-transformation $x = z - P^{-1}q$, which results in the problem

$$\begin{array}{ll} \text{minimize} & x^\top Px + r - q^\top P^{-1}q \\ \text{subject to} & Ax \leq b - AP^{-1}q \end{array}$$

Constant terms in the objective do not influence the optimal solution (only the optimal value of the problem). We can therefore disregard $r - q^\top P^{-1}q$ when we find its optimal solution. Moreover, since b is affine in x_t and q is linear in x_t, the inequality constraint can be written as $Ax \leq w + Sx_t$. Hence, for a given x_t, we can find the MPC control action (5.5) by first solving

$$\begin{array}{ll} \text{minimize} & x^\top Px \\ \text{subject to} & Ax \leq w + Sx_t \end{array} \tag{5.13}$$

for $x^\star$, then forming $z^\star = x^\star + P^{-1}q$, and extracting the first component(s) of $z^\star$ to obtain $u_t = \hat{u}_0^\star$

One can view the components of x_t as parameters of the planning problem. With this perspective, (5.13) is a multiparametric quadratic program reviewed in Appendix C. For such problems, one can characterize precisely how the optimizer and the optimal value depend on the parameters. By Proposition C.6.1, the optimal solution to (5.13) $x^\star(x_t)$ is continuous and piecewise affine in x_t. Since $P^{-1}q$ is linear in x_t, it follows that $z^\star(x_t)$ is also piecewise affine and continuous in x_t. Hence, u_t, which is defined as the m first components of $z^\star(x_t)$ is continuous and piecewise affine.

As $z^\star(x_t)$ is continuous and piecewise affine, it is clear that $z^\star(x_t)^\top Pz^\star(x_t)$ is continuous and piecewise quadratic. Similarly, since q depends linearly on x_t, $2q^\top z^\star(x_t)$ is also continuous and piecewise quadratic. Finally, as r is a quadratic function of x_t, we conclude that the optimal value of (5.13) is a continuous and piecewise quadratic function of x_t. □

There are a number of algorithms for multiparametric quadratic programming that construct a polyhedral partition of the feasible set of (5.13) and determine the affine expressions that describe how the control signal depends on the state in each such region. This information is typically computed off-line and stored in memory. During run-time, finding an optimal solution to (5.13) then simply amounts to finding which polyhedron in the partition the current state belongs to, and then evaluating the corresponding affine expression for how u_t depends on x_t. In the context of model predictive control, this approach is known as *explicit MPC*. The advantage of explicit MPC is that it requires a very limited online effort, and that it is easy to get an estimate of the worst-case execution time of the MPC controller. The disadvantage of explicit MPC is that mpQP solvers sometimes generate partitions with many polyhedra, which means that the corresponding explicit control law requires significant memory to store. This is especially true for systems with high state dimensions and planning problems with long horizons and many constraints.

■ **Example 5.8** Let us return to the quadcopter dynamics $|u_t| \leq 1$ and states constrained to $\|x_t\|_\infty \leq 10$. For simplicity, we use $T = 3$, $Q = I$, $R = 1$, $Q_T = 0$ and the terminal set defined by the maximal invariant set of the associated linear-quadratic controller. Figure 5.11 shows the partition of the state space into convex 21 polyhedra (left), and the resulting continuous piecewise affine feedback law (right). Note that the feasible set of the controller agrees with the one in Example 5.6. With horizons $T = 6$ and 9, the explicit controller has 55 and 85 critical regions, respectively. ■

We have given a brief exposition of the nonlinear nature of the basic MPC controller based on results for a specific class of multiparametric quadratic programs. A few additional details of how

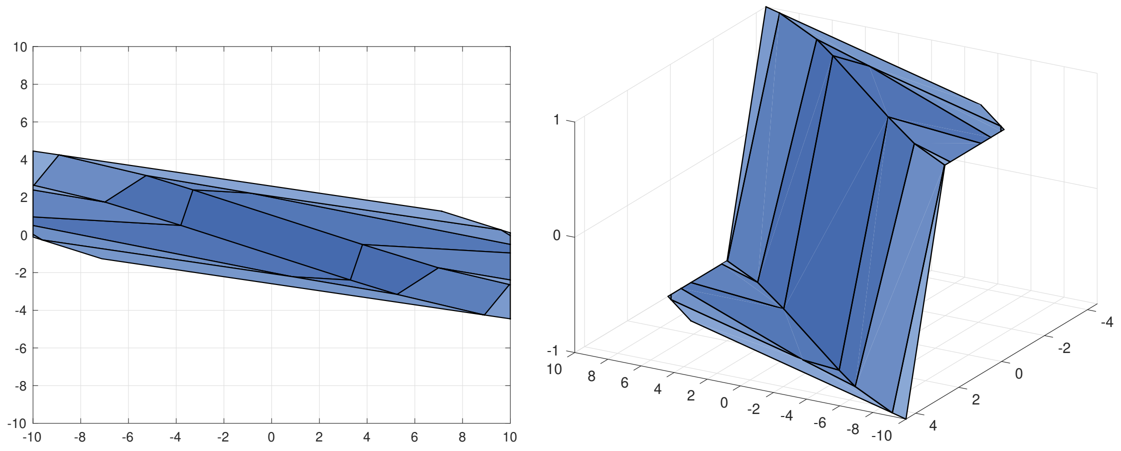


Figure 5.11: Partition for explicit MPC controller for the quadrotor (left) and the corresponding control law (right).

such a solver works is given in Appendix C. With a small additional effort, the approach can be extended to derive explicit expressions for many of the more advanced MPC strategies that we will encounter in the remainder of this chapter; see [5] for details.

5.5 A few practical enhancements to the basic MPC policy

There are several variations of the basic MPC formulation that are very useful in practice, but result in control policies that are somewhat more difficult to analyze theoretically. We will explore a few of them here. In particular, we will elaborate on how we can guarantee that the planning problem always remains feasible by softening its constraints, discuss how the planning problem can be expressed in terms of separate control and prediction horizons, and illustrate how the fixed terminal set can be replaced with explicit constraint checking over an extended horizon.

Ensuring a feasible planning problem by softening constraints

Although we have proven that a careful combination of terminal set and terminal cost guarantees both recursive feasibility and closed-loop stability, there are reasons to be critical to our analysis. In particular, we have assumed that the model used for predictions in the controller is identical to the actual process dynamics and that no unknown disturbances are acting on the system. So even if our idealized analysis tells us that we should have recursive feasibility, the optimization problem (5.4) may still become infeasible when the controller runs against the real process.

To increase the likelihood that the planning problem remains feasible, we can allow the optimizer to violate certain constraints if this becomes necessary. To this end, we distinguish between *hard constraints* that must be respected at all times, and *soft constraints* that can be breached without severe consequences. Examples of hard constraints include limits on the control that the actuator can deliver and safety-critical operating limits on process states. Soft constraints, on the other hand, are typically introduced from a performance perspective and can be violated without significant consequences. To signal to the optimizer that a constraint of the form

$$a^\top z \leq b$$

can be violated, we introduce a non-negative *slack variable* $s \geq 0$ and modify the constraint to

$$a^\top z \leq b + s.$$

This inequality can always be satisfied with $s = \max\{0, a^\top z - b\}$. Constraint violations are discouraged by adding a positive slack cost $\sigma(s)$ to the objective of the planning problem. Common costs include the linear $\sigma(s) = \kappa s$ and quadratic $\sigma(s) = \kappa s^2$, each with its distinct characteristics. The quadratic penalty starts with a zero slope at the origin but increases rapidly with s, making it lenient for minor violations while penalizing larger ones heavily. Conversely, the linear penalty begins with a non-zero slope and increases gradually, discouraging even small violations but being more forgiving for large constraint violations. The next example attempts to quantify this intuition.

■ **Example 5.9** Let us consider the quadratic function $f(x) = (x-1)^2$ under the constraint $x \leq 0$. Clearly the optimal solution is $x^\star = 0$, for which $f^\star = f(x^\star) = 1$. Now consider the softened version

$$\begin{array}{ll} \text{minimize} & f(x) + \sigma(s) \\ \text{subject to} & x \leq s, \qquad s \geq 0 \end{array}$$

with $\sigma(s) = \kappa s^2$. Then, compared to $x^\star = s = 0$ letting $x = x^\star + s$ with $s \geq 0$ changes the cost by

$$f(x) + \kappa\sigma(s) - f(x^\star) = (x^\star + s - 1)^2 + \kappa s^2 - (x^\star - 1)^2 = s^2 - s + \kappa s^2 = (\kappa+1)s^2 - s$$

which is negative for $s \in (0, 1/(\kappa+1))$. Hence, no matter how large we choose κ, it will always be optimal to pick $s^\star = 1/2(\kappa+1) > 0$ and to violate the constraint (if even by a little).

For the linear penalty $\sigma(s) = \kappa s$, a similar small variation in x and s changes the cost by

$$(x^\star + s - 1)^2 + \kappa s - (x^\star - 1)^2 = s^2 + (\kappa - 2)s$$

which is positive for all s if $\kappa \geq 2$. Therefore, there will be no constraint violation if $\kappa \geq 2$. ■

One can demonstrate that the observations from our simple example hold true for more general quadratic programs. A small relaxation of the constraints will lead to a decrease in the optimal value of the QP that depends linearly on the perturbation. Thus, if $\sigma(s)$ is quadratic, it will always be better to violate the constraint slightly (decreasing the cost proportionally to the violation, and accepting a slack penalty cost with the slower quadratic growth for small violations).

In the context of model-predictive control, we often enforce multiple constraints at multiple (possibly all) times in the planning horizon. When softening these constraints, we can either use a single slack variable for all constraints or assign individual slack variables to each constraint at every sample. Opting for a single slack variable focuses on minimizing the most significant violation, but allows all soft constraints to be violated to the same level. This may lead the optimizer to breach constraints that actually could have been met. On the other hand, employing individual slack variables for each constraint increases the total number of variables but enables us to discourage violations of every single constraint. In the latter case, original constraints on the form

$$M_x \hat{x}_k \leq m_x$$

are softened using different slack vectors s_k for each time in the horizon, i.e.

$$M_x \hat{x}_k \leq m_x + s_k$$

and the objective of the planning problem becomes

$$\sum_{k=0}^{T-1} \hat{x}_k^\top Q \hat{x}_k + \hat{u}_k^\top R \hat{u}_k + \sigma(s_k) + \hat{x}_T^\top Q_t \hat{x}_T$$

where $\sigma(s) = \kappa\|s\|_1$ or $\sigma(s) = \kappa\|s\|_2^2$. Although certainly possible, it is rather uncommon to use different weights κ for different points in time (or different weights for different components of s_k).

The next example demonstrates the effect of constraint softening in an MPC problem.

■ **Example 5.10 — Softening constraints with quadratic and linear penalties.** Consider

$$x_{t+1} = \begin{pmatrix} 0.8 & -0.6 & 0.2 \\ 1 & 0 & 0 \\ 0 & 0.8 & 0.6 \end{pmatrix} x_t + \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} u_t$$
$$y_t = \begin{pmatrix} -1 & 1 & 1 \end{pmatrix} x_t$$

subject to the constraints $|u_t| \leq 1$ and $|y_t| \leq 1$. For simplicity, we use an MPC controller without terminal state and terminal cost, and let $Q = C^\top C$, $R = 1$ and $T = 20$.

From the initial value $x_0 = (0.8, 0.8, 0.8)$, it is impossible to respect the upper limit on the output, and without softened constraints the MPC problem would have been infeasible. Figure 5.12 demonstrates the effect of constraint softening with linear and quadratic penalties, respectively. The simulations show how larger penalties on the slacks give better constraint satisfaction. We also see that the quadratic penalty is better at avoiding large deviations, while the linear penalty is better at ensuring that the constraints are violated for a short period of time, when possible. ■

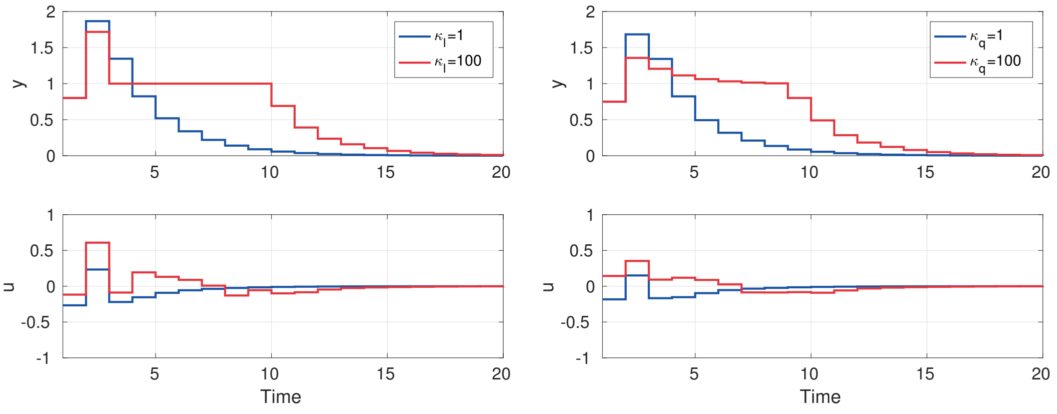


Figure 5.12: Larger penalties on slacks reduce constraint violations. Quadratic penalties are better at reducing the amplitude of violations, while linear penalties are better at limiting the duration of constraint violations.

Although there are theoretical guidelines for adjusting the slack penalty weights, many practical applications use a quadratic slack penalty with its weight tuned to be large enough in simulations. However, it's important to be cautious, as setting the weight too high can cause numerical issues, making the planning problem difficult to solve due to ill-conditioning.

The dual mode concept and separate control and prediction horizons

We motivated the MPC policy from the division of an infinite-horizon optimal control problem into two distinct modes: the first one steers the system through a constrained transient, and the second one applies an infinite-horizon optimal LQR controller to drive the state to rest; see Figure 5.13. In Proposition 5.3.1 we demonstrated that a corresponding MPC policy will be stable and recursively feasible if we choose the terminal set and terminal cost appropriately, and in Proposition 5.3.4 we demonstrated that the policy will also be optimal if the planning horizon is long enough.

The *dual mode* strategy of planning the optimal control inputs $\{\hat{u}_0, \hat{u}_1, \dots, \hat{u}_{T-1}\}$ over a finite horizon T and then switching to a fixed control policy suggests several interesting modifications of the planning problem (5.4). One such change is to replace the terminal set by *explicit constraint checking*. We then reformulate the terminal constraint $\hat{x}_T \in X_T$ as a series of conditions that ensure that the second mode policy remains admissible and the predicted states satisfy their constraints.

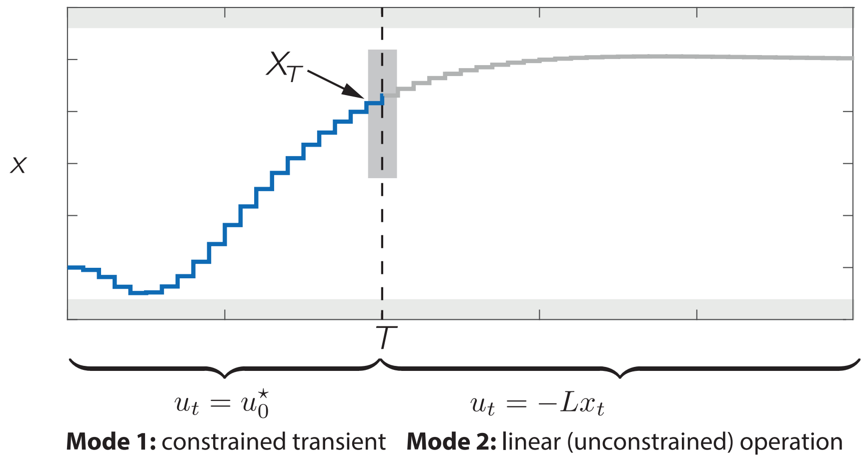


Figure 5.13: The dual mode perspective: the MPC controller plans for a first mode optimizing a constrained transient, and a second mode that uses a fixed linear state feedback to regulate the state around its target and away from constraints.

When the controller in the second mode is linear, $\hat{u}_k = -L\hat{x}_k$, it holds that $\hat{x}_{T+1} = A\hat{x}_T + B\hat{u}_T = (A-BL)\hat{x}_T$, and more generally that

$$\hat{x}_{T+k} = (A-BL)^k\hat{x}_T \qquad \text{for } k \geq 0$$

Assume that the origin lies in the interior of X and U so that we can express the state constraints as

$$\hat{x}_k \in X = \{x \mid M_x x \leq \mathbf{1}\}$$

and the control constraints as

$$\hat{u}_k \in U = \{u \mid M_u u \leq \mathbf{1}\}.$$

Then, $\hat{x}_{T+k}$ satisfies its constraints and results in an admissible second mode control if

$$\begin{pmatrix} M_x\hat{x}_{T+k} \\ M_u\hat{u}_{T+k} \end{pmatrix} \leq \mathbf{1} \Leftrightarrow \begin{pmatrix} M_x \\ -M_uL \end{pmatrix} \hat{x}_{T+k} \leq \mathbf{1} \Leftrightarrow \bar{M}\hat{x}_{T+k} \leq \mathbf{1}$$

with the obvious definition of $\bar{M}$. Since $\hat{x}_{T+k} = (A-BL)^k\hat{x}_T$, the requirement that $\hat{x}_{T+k}$ and $\hat{u}_{T+k}$ should satisfy their constraints for all future times can therefore be expressed as

$$\bar{M}(A-BL)^k\hat{x}_T \leq \mathbf{1} \qquad \text{for } k = 0, 1, \ldots \tag{5.14}$$

If we compare these conditions with the ones in Theorem 2.3.1, we see that they define the maximal invariant set for $\hat{x}_{k+1} = (A-BL)\hat{x}_k$ contained in $\{x \mid \bar{M}x \leq \mathbf{1}\}$. Hence, if $(A-BL)$ is Schur and $(A-BL, \bar{M})$ is observable, the determinedness index ν of the invariant set is finite, and it is enough to impose (5.14) over a finite horizon $T_c \geq \nu$. The corresponding MPC planning problem is

$$\begin{array}{lll} \text{minimize} & \sum_{k=0}^{T-1} \hat{x}_k^\top Q\hat{x}_k + \hat{u}_k^\top R\hat{u}_k + \hat{x}_T^\top Q_T\hat{x}_T & \\ \text{subject to} & \hat{x}_{k+1} = A\hat{x}_k + B\hat{u}_k & k = 0, \ldots, T-1 \\ & M\hat{x}_k + N\hat{u}_k \leq m & k = 0, \ldots, T-1 \\ & \bar{M}(A-BL)^{k-T}\hat{x}_T \leq \mathbf{1} & k = T, \ldots, T+T_c \\ & \hat{x}_0 = x_t & \end{array} \tag{5.15}$$

Note that the terminal constraint is replaced by constraint checking over $k = T, T+1, \dots, T+T_c$.

In the literature, T is known as the *control horizon* since we optimize the control action over the T first samples, while $T+T_c$ is known as the *prediction horizon* since we predict the effect of our control actions over all these samples. We will mainly use dual mode formulation (5.15) when it is difficult to compute an invariant-set off-line. However, the original reason for using a separate control horizon and prediction horizon was to decrease the computational effort required to solve the associated quadratic program. A small control horizon gives an optimization problem with fewer decision variables, but also a more shortsighted controller. As the analogy with the terminal set shows, stability can still be guaranteed if the prediction horizon is sufficiently long and if Q_T is chosen to reflect the infinite-horizon cost of the dual mode controller. Of course, a short control horizon T leads to a small set of recursively feasible states also in this formulation.

The use of pre-stabilized predictions to improve numerical conditioning

Another modification that comes naturally in the dual-mode framework is to re-parameterize the complete planning problem in terms of offsets v_k from a fixed control law. Specifically, if we let

$$\hat{u}_k = -L\hat{x}_k + \hat{v}_k$$

with $\hat{v}_k = 0$ for $k \geq T$, then we can express the state predictions as

$$\hat{x}_{k+1} = A\hat{x}_k + B\hat{u}_k = (A - BL)\hat{x}_k + B\hat{v}_k.$$

These *pre-stabilized* predictions may lead to a better numerical conditioning of the resulting quadratic program (and hence faster and more reliable solutions) than the original formulation.

An inconvenience of this approach is that it introduces a coupling between $\hat{x}_k$ and $\hat{v}_k$. For example, simple bound constraints on $\hat{u}_k$ are replaced by joint constraints on $\hat{x}_k$ and $\hat{v}_k$, and the objective function will have terms that depend on both $\hat{x}_l$ and $\hat{v}_k$ since

$$\hat{u}_k^\top R\hat{u}_k = (\hat{v}_k - L\hat{x}_k)^\top R(\hat{v}_k - L\hat{x}_k) = \hat{v}_k^\top R\hat{v}_k + \hat{x}_k^\top L^\top L\hat{x}_k - 2\hat{v}_k^T RL\hat{x}_k$$

This coupling can increase the solution times for some solvers, counteracting some of the gains that come from better numerical conditioning of the prediction equations.

Pre-stabilized predictions can also be used without the dual mode (i.e. in our original formulation of the planning problem with a terminal set constraint).

Move-blocking to reduce the number of decision variables

The number of free control variables in the planning problem is proportional to the horizon T. One way to limit the number of decision variables without reducing the planning horizon length is to use *move blocking*, *i.e.* to constrain the control input to remain constant over predefined intervals of the prediction horizon. For example, we could only allow the predicted optimal controls to change every second sample. This would require that $\hat{u}_1 = \hat{u}_0$, $\hat{u}_3 = \hat{u}_2$, etc and lead to an intermediate approach between standard MPC and on that uses a double sampling interval. The planning problem adjusts the control input every second sample, but it accounts for the system behavior and enforces state constraints in every sample. In addition, since the planning problem is re-solved in every sampling interval, the actual implemented control input typically changes in every sample.

Move-blocking schemes can be challenging to analyze theoretically. The main reason for this is that the shifted input trajectory from one sampling instance often violates the move-blocking constraints in the next. Hence, there is no guarantee that a feasible control sequence at the current time step will lead to a state where a feasible solution exists at the next time step. The situation is slightly better for "offset-blocking" where one uses pre-stabilized predictions and limits the offsets v_k to be zero at certain time steps. In this case, for states within the maximal invariant set of the feedback controller, setting all offsets to zero yields an admissible solution. This improves the likelihood of maintaining feasibility, but does not guarantee recursive feasibility for all initial states within the feasible region. To rigorously prove stability and recursive feasibility for move-blocking schemes, more structured approaches are needed, see [11].

5.6 Model predictive control for reference tracking

Let us begin by considering the problem of making the output $y_t = Cx_t$ converge to a stationary reference value r. This problem can be addressed using the same ideas that we used for the linear-quadratic regulator, *i.e.* to determine x^{ref} and u^{ref} such that

$$\begin{cases} x^{\text{ref}} & = & Ax^{\text{ref}} + Bu^{\text{ref}} \\ r & = & Cx^{\text{ref}} \end{cases} \tag{5.16}$$

and then penalize the deviations $\Delta x_t = x_t - x^{\text{ref}}$ and $\Delta u_t = u_t - u^{\text{ref}}$ from these reference values. In the absence of constraints, we have shown that the optimal tracking problem can be posed as a standard LQR problem in the deviations

$$\begin{array}{ll} \text{minimize} & \sum_{t=0}^{\infty} \Delta x_t^{\top} Q \Delta x_t + \Delta u_t^{\top} R \Delta u_t \\ \text{subject to} & \Delta x_{t+1} = A\Delta x_t + B\Delta u_t \end{array}$$

This allowed us to argue that the optimal control is on the form $\Delta u_t = -L\Delta x_t$ where L is obtained by solving a standard LQR problem, and that the cost-to-go function for this problem is $v(x_t) = \Delta x_t^{\top} P \Delta x_t$ where P satisfies the corresponding discrete-time algebraic Riccati equation.

In light of these observations, it is natural to formulate the MPC planning problem in the coordinates $(\Delta\hat{x}_t, \Delta\hat{u}_t)$ using a terminal cost on the form $\Delta\hat{x}_T^{\top} Q_T \Delta\hat{x}_T$. Since the constraints are typically stated in terms of the actual states, *e.g.* $M_x x_t \leq m_x$ and $M_u u_t \leq m_u$, we use the relationships $x_t = \Delta x_t + x^{\text{ref}}$ and $u_t = \Delta u_t + u^{\text{ref}}$ and pose the planning problem as

$$\begin{array}{lll} \text{minimize} & \sum_{k=0}^{T} \Delta\hat{x}_k^{\top} Q \Delta\hat{x}_k + \Delta\hat{u}_k^{\top} R \Delta\hat{u} + \Delta\hat{x}_T^{\top} Q_T \Delta\hat{x}_T & \\ \text{subject to} & \Delta\hat{x}_{k+1} = A\Delta\hat{x}_k + B\Delta\hat{u}_k & t = 0,1,\ldots,T-1 \\ & M_x(\Delta\hat{x}_k + x^{\text{ref}}) \leq m_x & t = 0,1,\ldots,T-1 \\ & M_u(\Delta\hat{u}_k + u^{\text{ref}}) \leq m_u & t = 0,1,\ldots,T-1 \\ & \Delta\hat{x}_T \in \bar{X}_T & \\ & \Delta\hat{x}_0 + x^{\text{ref}} = x_t & \end{array}.$$

The applied control is simply $\hat{u}_0^{\star} = \Delta\hat{u}_0^{\star} + u^{\text{ref}}$. To guarantee recursive feasibility, the terminal set $\bar{X}_T$ should be control invariant for the dynamics of $\Delta\hat{x}_k$ under the control constraint

$$\Delta\hat{u}_k \in \bar{U} = \left\{\Delta u \mid M_u \Delta u \leq m_u - M_u u^{\text{ref}}\right\}$$

and $\bar{X}_T$ should be contained in the set

$$\bar{X} = \left\{\Delta x \mid M_x \Delta x \leq m_x - M_x x^{\text{ref}}\right\}.$$

Since these sets depend on the reference (through x^{ref} and u^{ref}), the terminal set will typically need to be re-calculated whenever the reference changes. The terminal set $\bar{X}_T = \{0\}$ is a valid choice if x^{ref} is feasible, but it severely limits the set of recursively feasible states. A better solution is to replace the control invariant set with explicit constraint checking for a fixed control law $\Delta\hat{u}_k = -L\Delta\hat{x}_k$ for $k = T, \ldots, T+T_c$. An inconvenience is that the determinedness index of the associated invariant set changes as the reference changes, so one may need to accept a heuristic choice of T_c.

It is possible to integrate the reference calculation in the planning problem. Doing so has the advantage that we can deal with infeasible reference requests by softening the requirement on perfect tracking. We then replace $r = Cx^{\text{ref}}$ by $r + s = Cx^{\text{ref}}$ and add a slack penalty $\sigma(s)$ to the objective function of the planning problem This leads to the following MPC formulation

$$\begin{array}{lll}
\text{minimize} & \sum_{k=0}^{T-1} \Delta\hat{x}_k^\top Q \Delta\hat{x}_k + \Delta\hat{u}_k^\top R \Delta\hat{u}_k + \Delta\hat{x}_T^\top Q_T \Delta\hat{x}_T + \sigma(s) & \\
\text{subject to} & \Delta\hat{x}_{k+1} = A\Delta\hat{x}_k + B\Delta\hat{u}_k & k = 0,\dots,T-1 \\
& M_x(\Delta\hat{x}_k + x^{\text{ref}}) \le m_x & k = 0,\dots,T-1 \\
& M_u(\Delta\hat{u}_k + u^{\text{ref}}) \le m_u & k = 0,\dots,T-1 \\
& \Delta\hat{x}_T \in \bar{X}_T & \\
& \begin{pmatrix} A-I & B \\ C & 0 \end{pmatrix} \begin{pmatrix} x^{\text{ref}} \\ u^{\text{ref}} \end{pmatrix} = \begin{pmatrix} 0 \\ r+s \end{pmatrix} & \\
& \Delta\hat{x}_0 + x^{\text{ref}} = x_t &
\end{array} \tag{5.17}$$

in variables $\{\Delta\hat{x}_k, \Delta\hat{u}_k\}, x^{\text{ref}}, u^{\text{ref}}$ and s. The applied control is

$$u_t = \Delta\hat{u}_0^\star + u^{\text{ref}}. \tag{5.18}$$

The next example demonstrates this approach on a simple tracking problem.

■ **Example 5.11** The following model represents the influence of the elevator surface deflection δ on the pitch rate q of an aircraft, see Figure 5.14.

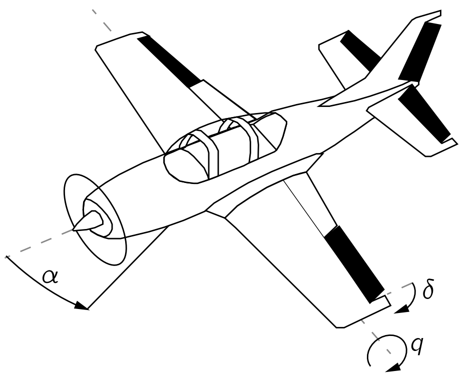


Figure 5.14: The elevator surface deflection δ affects the angle of attack α and the pitch rate q.

$$\begin{pmatrix} \alpha_{k+1} \\ q_{k+1} \end{pmatrix} = \begin{pmatrix} 0.9719 & 0.0155 \\ 0.2097 & 0.9705 \end{pmatrix} \begin{pmatrix} \alpha_k \\ q_k \end{pmatrix} + \begin{pmatrix} 0.0071 \\ 0.3263 \end{pmatrix} \delta_k$$

The control problem is to determine the appropriate elevator surface deflection δ_k for tracking a given reference angle α_r. Both states and the control are subject to upper and lower bounds

$$X = \{(\alpha, q) : -15 \le \alpha \le 30 \text{ and } -100 \le q \le 100\} \qquad U = \{\delta \ : -25 \le \delta \le 25\}$$

We consider the cost defined by

$$Q = \begin{pmatrix} 10 & 0 \\ 0 & 1 \end{pmatrix}. \quad R = 1$$

whose stationary optimal control in linear operation is

$$u_t = -L\begin{pmatrix}\alpha_k \\ q_k\end{pmatrix} + l_r\alpha_r = -\begin{pmatrix}1.9603 & 0.8385\end{pmatrix}\begin{pmatrix}\alpha_k \\ q_k\end{pmatrix} + 3.1973\alpha_r$$

The planning problem uses a horizon of $T = 40$ samples.

As shown in Figure 5.15, constraint softening allows the MPC controller to act on both feasible and infeasible reference values (*e.g.*, $\alpha_r < \alpha_{\min} = -15$). The phase plane plot shown in Figure 5.16 (left) gives an alternative perspective of how the MPC controller steers the system state trajectory (blue crosses) to track the given reference (green circles).

Figure 5.16 (right) shows the terminal sets defined by the maximal invariant sets of the closed-loop under the LQ-optimal feedback for the different reference requests. In this example, the reference states in the interior of the constraint set (shown in grey) have a determinedness index of 2 and would thus only require $T_c = 2$. However, the equilibrium corresponding to $\alpha = -15$ lies on the boundary and is (much) more complex to represent with a determinedness index of 38. ■

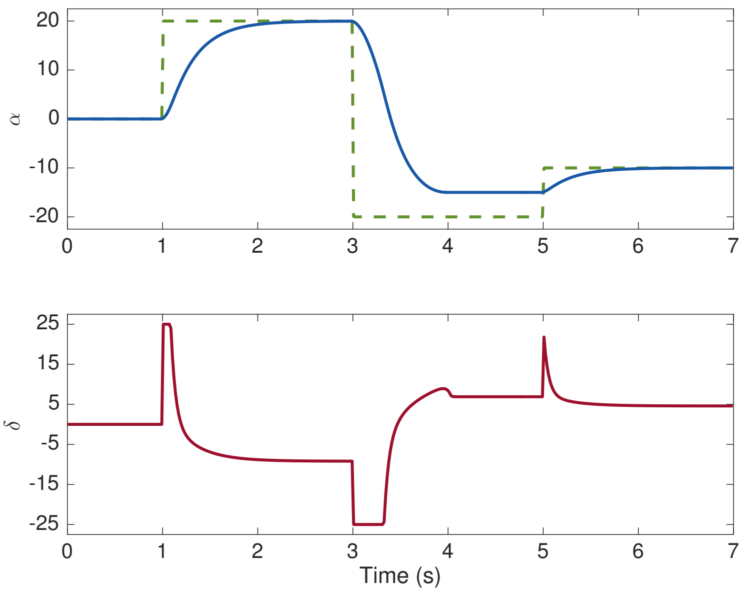


Figure 5.15: Upper plot shows how the controller is able to track feasible references without error and at the same time deal with unfeasible reference requests.

Preview of known future signals

If the reference changes are known in advance, then we can include this information in the MPC planning problem to optimize the state transition even further. Thus, rather than using a constant reference r in (5.17), we supply the known reference trajectory $\{r_k\}$ over the prediction horizon. In the MPC planning problem, we replace the constant target state, controls and slack vectors $(x^{\text{ref}}, u^{\text{ref}}, s)$ by sequences $(\{x_k^{\text{ref}}\}, \{u_k^{\text{ref}}\}, \{s_k\})$ which are allowed to vary over the prediction horizon as long as they satisfy (5.16). The next example illustrates the potential benefits of preview.

■ **Example 5.12** We return to the aircraft problem studied in Example 5.11 and assume that the future reference is known over the prediction horizon. As shown in Figure 5.17, the preview allows the controller to prepare the transition earlier and ensure a faster convergence to the new set-point. ■

Preview is not limited to only reference signals; it can also be used to improve disturbance suppression if we can obtain a forecast of the future disturbances over the planning horizon. When the disturbances can be modelled as outputs of linear systems, we can can include these in the

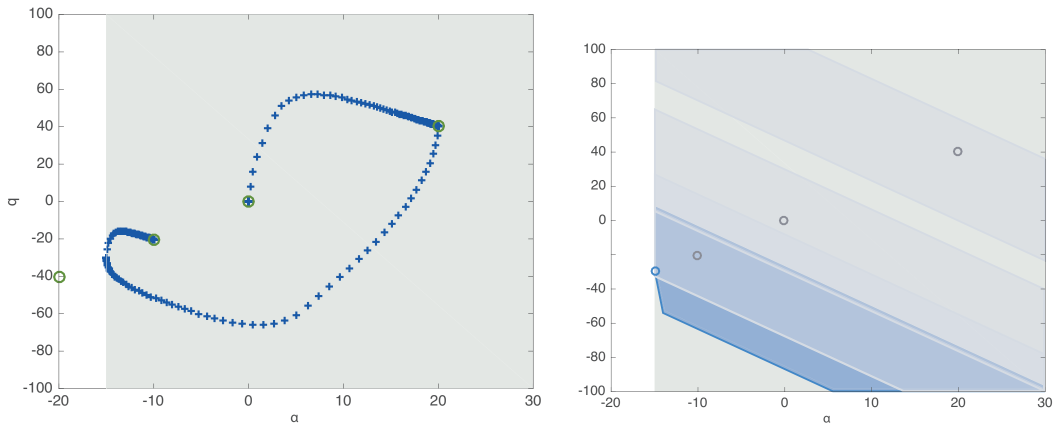


Figure 5.16: The closed-loop trajectories in the phase-plane shows how the system deals with the feasible and infeasible reference requests (left). The right figure shows the invariant sets for the LQ-optimal controller under the given constraints and for the different set-points. The three set-points inside the constraint sets result in invariant sets with simple representations (grey) while the reference on the constraint boundary gives a more complex invariant set (blue).

prediction equations following a method similar to what we described in the previous chapter. Alternatively, the controller might receive forecasts of future disturbances from another system component, as illustrated by the reference example. In such instances, we simply incorporate the time-varying disturbance sequence in the prediction equations.

A brief remark on integrated planning and model-predictive control

A natural next step after mastering reference tracking is to include the optimization of the references themselves. MPC offers a compelling framework to integrate planning and control into a single optimization problem. By jointly optimizing over future reference trajectories and control actions, one can, in principle, achieve truly optimal solutions that respect both long-term objectives and dynamic constraints. However, this integration comes at a significant computational cost, and solving high-dimensional planning problems over long time horizons is challenging to do in real-time, especially for systems with fast dynamics. In practice, many applications – such as autonomous navigation or industrial processes – exhibit a natural time-scale separation where planning operates at a slower, strategic level, while feedback control requires rapid, reactive adjustments. A decoupled strategy – conducting planning over longer intervals with a faster feedback control loop – often leads to a better balance between optimality and computational effort.

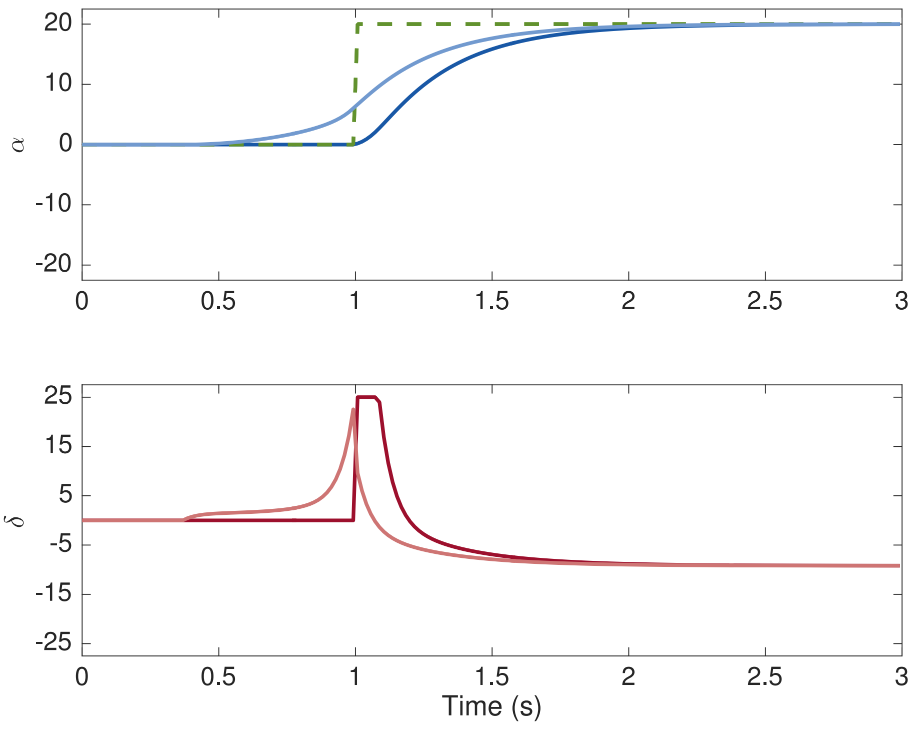


Figure 5.17: By including preview (lighter colors), the controller acts on the reference change ahead of time, ensuring a faster and smoother transition.

5.7 Disturbance compensation and offset-free MPC

We will now consider the slightly more complex problem of reference tracking in the presence of disturbances. As discussed above, if we have a good model of the disturbances, then we can simply include this model in the prediction equations, estimate the disturbances using an observer, and compensate for them in the MPC planning problem. This is also the essence of the approach to offset-free MPC which we will describe here. When the disturbances are constant and we are only concerned about the disturbance compensation in stationarity, it is possible to give strong guarantees. Under rather mild conditions, any unconstrained equilibrium of the closed-loop system then attains offset-free tracking. Much like statements about integral action in linear systems, this argument does not rely on a perfect match between the system dynamics and the model, neither when it comes to the system matrices nor the disturbances. Instead, the argument relies on the assumption that the MPC controller will be able to remain feasible and attain an equilibrium.

To account for the effect of disturbances, we consider the following system model

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t + B_d d_t \\ y_t &= Cx_t + C_d d_t \end{aligned} \tag{5.19}$$

where $d_t \in \mathbb{R}^{n_d}$ is the disturbance vector, while $B_d \in \mathbb{R}^{n\times n_d}$ and $C_d \in \mathbb{R}^{p\times n_d}$ are matrices that describe how the disturbance vector acts on the state evolution and on the measured output. We would like to ensure that $y_t \to r$ as $t \to \infty$ for any feasible reference vector $r \in \mathbb{R}^p$.

Assume that the disturbance is constant, *i.e.* $d_{t+1} = d_t$ for all t. We can then re-write (5.19) as

$$\begin{aligned} \begin{pmatrix} x_{t+1} \\ d_{t+1} \end{pmatrix} &= \begin{pmatrix} A & B_d \\ 0 & I \end{pmatrix} \begin{pmatrix} x_t \\ d_t \end{pmatrix} + \begin{pmatrix} B \\ 0 \end{pmatrix} u_t &:= A_e \begin{pmatrix} x_t \\ d_t \end{pmatrix} + B_e u_t \\ y_t &= \begin{pmatrix} C & C_d \end{pmatrix} \begin{pmatrix} x_t \\ d_t \end{pmatrix} &:= C_e \begin{pmatrix} x_t \\ d_t \end{pmatrix} \end{aligned} \tag{5.20}$$

The state vector of this extended system includes both the system state x_t and the disturbance vector d_t. If the full state vector cannot be measured, it is natural to ask under what conditions it can be estimated from output and input sequences $\{y_t\}$ and $\{u_t\}$. The next result answers this question.

Proposition 5.7.1 Assume that the nominal system (A,C) is observable. Then (A_e,C_e) is observable if and only if the matrix

$$\begin{pmatrix} A-I & B_d \\ C & C_d \end{pmatrix}$$

has rank $n+n_d$.

Proof. By the PBH test, the extended system is observable if and only if there is no λ and no $(v_1,v_2) \neq 0$ such that

$$\begin{pmatrix} A & B_d \\ 0 & I \\ C & C_d \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \lambda \begin{pmatrix} v_1 \\ v_2 \\ 0 \end{pmatrix} \tag{5.21}$$

Note that if we let $v_2 = 0$, these equations reduce to the conditions in the PBH test for observability of the nominal system which, by assumption, only admits the solution $v_1 = 0$. If $v_2 \neq 0$, on the other hand, we must have $\lambda = 1$ and (5.21) simplifies to

$$\begin{pmatrix} A-I & B_d \\ C & C_d \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = 0$$

Under the rank condition posed in the theorem, the only solution to these equations is $(v_1,v_2) = 0$. Hence, the extended system is observable and the proof is complete. ■

The rank condition requires that $p \geq n_d$, so we must measure at least as many signals as the number of disturbances in our model. If the extended system is observable, then we can attempt to estimate the extended state vector using an observer on the form

$$\begin{aligned} \begin{pmatrix} \hat{x}_{t+1} \\ \hat{d}_{t+1} \end{pmatrix} &= \begin{pmatrix} A & B_d \\ 0 & I \end{pmatrix} \begin{pmatrix} \hat{x}_t \\ \hat{d}_t \end{pmatrix} + \begin{pmatrix} B \\ 0 \end{pmatrix} u_t + \begin{pmatrix} K_x \\ K_d \end{pmatrix} (y_t - \hat{y}_t) \\ \hat{y}_t &= \begin{pmatrix} C & C_d \end{pmatrix} \begin{pmatrix} \hat{x}_t \\ \hat{d}_t \end{pmatrix} \end{aligned} \tag{5.22}$$

$A_e - KC_e$ is Schur stable.

Given that we have estimated the disturbance correctly, we can determine reference states x^{ref} and controls u^{ref} that make the output equal to the desired reference, *i.e.*

$$\begin{aligned} x^{\text{ref}} &= Ax^{\text{ref}} + Bu^{\text{ref}} + B_d d \\ r &= Cx^{\text{ref}} + C_d d \end{aligned}$$

We re-write these conditions as

$$\begin{pmatrix} A-I & B \\ C & 0 \end{pmatrix} \begin{pmatrix} x^{\text{ref}} \\ u^{\text{ref}} \end{pmatrix} = \begin{pmatrix} -B_d d \\ r - C_d d \end{pmatrix} \tag{5.23}$$

and note that we can find $(x^{\text{ref}}, u^{\text{ref}})$ for every right-hand side if the matrix

$$\begin{pmatrix} A-I & B \\ C & 0 \end{pmatrix}$$

has rank $n+p$. This requires that $m \geq p$, so to ensure offset-free tracking we must, in general, have at least as many control signals as we have outputs that we want to track.

We can now use an MPC controller whose objective function penalizes deviations from x^{eq} and u^{eq}. The values of the target state and control are computed from the estimated disturbance under the assumption that the observer has converged, $\hat{d}_t = d$:

$$\begin{array}{lll} \text{minimize} & \displaystyle\sum_{k=0}^{T-1} \Delta\hat{x}_k^\top Q \Delta\hat{x}_k + \Delta\hat{u}_k^\top R \Delta\hat{u}_k + \Delta\hat{x}_T^\top Q_T \Delta\hat{x}_T & \\ \text{subject to} & \Delta\hat{x}_{k+1} = A\Delta\hat{x}_k + B\Delta\hat{u}_k & k = 0,\dots,T-1 \\ & M_x(\Delta\hat{x}_k + x^{\text{ref}}) \leq m_x & k = 0,\dots,T-1 \\ & M_u(\Delta\hat{u}_k + u^{\text{ref}}) \leq m_u & k = 0,\dots,T-1 \\ & \Delta\hat{x}_T \in \bar{X}_T & \\ & \begin{pmatrix} A-I & B \\ C & 0 \end{pmatrix} \begin{pmatrix} x^{\text{ref}} \\ u^{\text{ref}} \end{pmatrix} = \begin{pmatrix} -B_d d \\ r - C_d d \end{pmatrix} & \\ & d = \hat{d}_t & \\ & \Delta\hat{x}_0 + x^{\text{ref}} = \hat{x}_t & \end{array} \tag{5.24}$$

in variables $\{\Delta\hat{x}_k, \Delta\hat{u}_k\}, x^{\text{ref}}$ and u^{ref}. The applied control is

$$u_t = \Delta\hat{u}_0^\star + u^{\text{ref}} \tag{5.25}$$

The following theorem shows that offset-free tracking can be ensured under mild conditions.

Theorem 5.7.2 Consider the discrete-time linear system (5.19) with $n_d = p$,

$$\text{rank} \begin{pmatrix} A-I & B_d \\ C & C_d \end{pmatrix} = n + n_d \tag{5.26}$$

and

$$\operatorname{rank}\begin{pmatrix} A-I & B \\ C & 0 \end{pmatrix} = n+p \tag{5.27}$$

Let $(\hat{x}_t, \hat{d}_t)$ be estimated by an observer on the form (5.22), designed to have asymptotically stable estimation error dynamics, and let u_t be given by the model predictive control law (5.24), tuned to ensure a asymptotically stable closed loop when r_t and d_t are known and fixed.

Assume that the MPC problem is feasible for all times and unconstrained for all $t \geq T_{\text{lin}}$ for some fixed time T_{lin}. If the closed-loop system reaches a steady-state, then $\lim_{t\to\infty} y_t = r$.

Proof. If the closed-loop system reaches a steady state $(\bar{x}, \bar{d}, \bar{u})$, then so does the observer, since its error dynamics are asymptotically stable. Let $(\hat{x}^{\text{eq}}, \hat{d}^{\text{eq}})$ be the stationary estimates of the observer, and note that stationarity of the observer equations necessitates that

$$K_d(\bar{y} - \hat{y}^{\text{eq}}) = 0 \tag{5.28}$$

where $\bar{y} = C\bar{x}$ and $\hat{y}^{\text{eq}} = C\hat{x}^{\text{eq}} + C_d\hat{d}^{\text{eq}}$. Moreover, by the asymptotic stability of the observer

$$\begin{aligned}\det\left(I - \left(A_e - \begin{pmatrix} K_x \\ K_d \end{pmatrix} C_e\right)\right) &= \det\begin{pmatrix} I - A + K_xC & -B_d + K_xC_d \\ K_dC & K_dC_d \end{pmatrix} = \\ &= \det\begin{pmatrix} I & 0 \\ 0 & K_d \end{pmatrix} \det\begin{pmatrix} I - A + K_xC & -B_d + K_xC_d \\ C & C_d \end{pmatrix} \neq 0.\end{aligned}$$

This condition requires that both the determinants are non-zero, which implies that $K_d \in \mathbb{R}^{p\times p}$ must have full rank. Hence, (5.28) can only hold if $\bar{y} = \hat{y}^{\text{eq}}$. Next, note that when the MPC controller uses the stationary observer estimates, the target state x^{ref} satisfies $Cx^{\text{ref}} = r - C_d\hat{d}^{\text{eq}}$, so

$$\bar{y} = \hat{y}^{\text{eq}} = C\hat{x}^{\text{eq}} + C_d\hat{d}^{\text{eq}} = C\hat{x}^{\text{eq}} + r - Cx^{\text{ref}} = r + C(\hat{x}^{\text{eq}} - x^{\text{ref}})$$

To ensure that $\bar{y} = r$, it remains to show that $\hat{x}^{\text{eq}} - x^{\text{ref}} = 0$. To this end, we combine the steady-state conditions of the estimator with the target calculation in the MPC controller

$$\begin{aligned}\hat{x}^{\text{eq}} - x^{\text{ref}} &= A\hat{x}^{\text{eq}} + B\bar{u} + B_d\hat{d}^{\text{eq}} - (Ax^{\text{ref}} + Bu^{\text{ref}} + B_d\hat{d}^{\text{eq}}) = \\ &= A(\hat{x}^{\text{eq}} - x^{\text{ref}}) + B(\bar{u} - u^{\text{ref}})\end{aligned}$$

Since we have assumed that the MPC controller operates without violating any constraints when $t \geq T_{\text{lin}}$, the MPC control will be linear and on the form $u_t - u^{\text{ref}} = -L(\hat{x}_t - x^{\text{ref}})$. In particular, for $u_t = \bar{u}$ and $\hat{x}_t = \hat{x}^{\text{eq}}$ we have $\bar{u} - u^{\text{ref}} = -L(\hat{x}^{\text{eq}} - x^{\text{ref}})$. It must therefore hold that

$$\hat{x}^{\text{eq}} - x^{\text{ref}} = (A - BL)(\hat{x}^{\text{eq}} - x^{\text{ref}}) \Leftrightarrow (I - A + BL)(\hat{x}^{\text{eq}} - x^{\text{ref}}) = 0$$

Since we have assumed that the MPC controller is asymptotically stable, the matrix $I - A + BL$ is invertible and the only solution is $\hat{x}^{\text{eq}} = x^{\text{ref}}$. Hence, $\bar{y} = r$ and we have offset-free tracking. $\square$

Several remarks are in order. First, note that the theorem does not assume that the equilibrium states and control $(\bar{x}, \bar{u})$ are equal to the reference states and controls $(x^{\text{ref}}, u^{\text{ref}})$ computed in the planning problem. Neither does it assume that the model used in the controller and the actual system dynamics match. It does, however, assume that the estimator and the planning problem are based on the same model, and that the closed-loop attains an unconstrained steady-state. Hence, when we design the model to be used in the disturbance observer and the MPC predictions, the disturbances do not need to correspond to physical signals acting on the system. Rather, they can be seen as a means for introducing integral action and ensuring offset-free tracking. If there are only a few real disturbance signals, one may need to introduce artificial disturbances to meet the requirement that $n_d = p$. These artificial disturbances should then be introduced so that the corresponding matrices B_d and C_d make the matrix (5.26) have full rank.

Second, the proof can be adapted to the use observers on the form

$$\begin{aligned}\begin{pmatrix}\hat{x}_{t|t-1}\\ \hat{d}_{t|t-1}\end{pmatrix} &= \begin{pmatrix}A & B_d\\ 0 & I\end{pmatrix}\begin{pmatrix}\hat{x}_{t-1|t-1}\\ \hat{d}_{t-1|t-1}\end{pmatrix} + \begin{pmatrix}B\\ 0\end{pmatrix}u_{t-1}\\ \begin{pmatrix}\hat{x}_{t|t}\\ \hat{d}_{t|t}\end{pmatrix} &= \begin{pmatrix}\hat{x}_{t|t-1}\\ \hat{d}_{t|t-1}\end{pmatrix} + \begin{pmatrix}K_x\\ K_d\end{pmatrix}\left(y_t - C\hat{x}_{t|t-1} - C_d\hat{d}_{t|t-1}\right).\end{aligned}$$

Recall that asymptotic stability of the error dynamics for this filter implies that

$$\det\left(I - \left(A_e - \begin{pmatrix}K_x\\ K_d\end{pmatrix}C_eA_e\right)\right) \neq 0$$

However, we can use the same factorization trick as before to conclude that K_d must have full rank. All other parts of the proof remain the same.

Finally, a limitation of the theorem is that it assumes that the planning problem is feasible at all times. This is not trivial to guarantee when the MPC controller is driven by estimated states. During transients, these state estimates will typically exhibit errors which may render the planning problem infeasible. Careful tuning of the observer helps, but it is often a good practice to soften constraints, especially the constraint on perfect reference tracking.

■ **Example 5.13** Let us return to the quadruple tank system used in Example 4.11, but now reconfigured to have a more challenging dynamics. Specifically, we use $\gamma_1 = 0.25$ and $\gamma_2 = 0.35$, which means that a majority of the inflow will enter the opposing tank system through its upper tank, creating a lingering effect on its lower tank level. The dynamics of this configuration becomes non-minimum phase and much slower than that in Example 4.11. The control objective is to track reference levels in the two lower tanks, using measurements of only these tanks. We use a sampling time of $h = 3$ seconds and design a one-step ahead predictor for the system states using the Kalman filter approach with $\Sigma_w = 100 \cdot I$ and $\Sigma_v = I$. The tank levels are limited to the interval $[0, 19.8]$ cm (the natural constraint of 20 cm has been tightened a little to get a safety margin to avoid overflow), and the pump voltages are restricted to the interval $[0, 10]$ V.

Figure 5.18 shows a simulation of two different controllers in two different scenarios. The dashed line is a reference tracking controller, designed without integral action, simulated for a scenario where no disturbances act on the system. The design uses $Q = I$, $R = 0.01 \cdot I$ and $T = 40$. The lower tank level has a distinct non-minimum phase behaviour, where the response first moves in the wrong direction, but the controller attains perfect tracking in steady-state. The MPC controller handles the constraints on both the states and the control signals and avoids overflow. We then simulate a scenario where a unit disturbance is acting on the second inflow from $t = 500$. Since the disturbance is not modeled, both the estimator and the MPC controller struggle and the overall controller is unable to track the references. Finally, we design an offset-free MPC with $B_d = B$ (to model input disturbances) and $C_d = 0$. The simulation, in full lines, shows how the controller is able to compensate for the disturbance and attain perfect tracking in stationarity. ■

Alternative approaches for offset-free MPC

The literature contains a few seemingly different techniques for achieving offset-free MPC. However, it turns out that many of them can either be seen as special cases of the approach described above or analyzed using the same techniques [29].

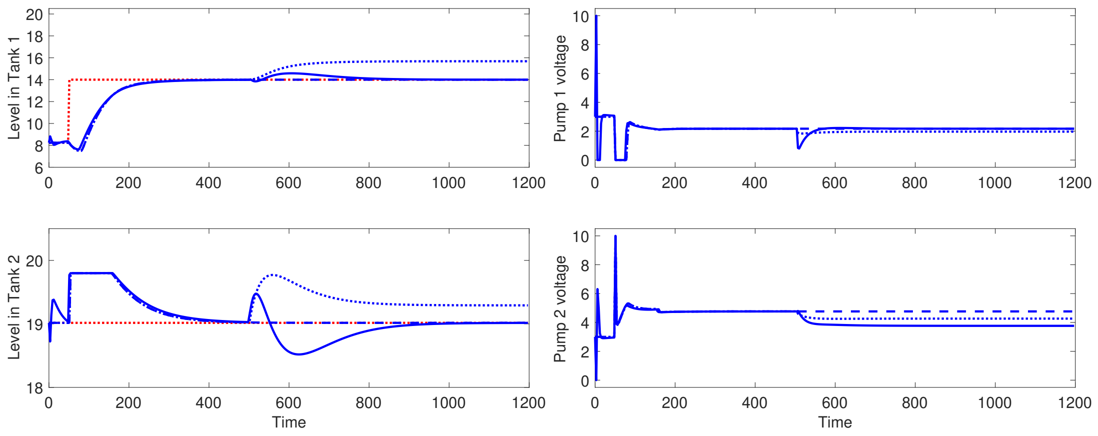


Figure 5.18: While the standard reference tracking solution works well (dashed blue) works well in the absence of disturbances, both the observer and the controller struggle in the presence of disturbances (dotted blue). In contrast, the off-set free MPC controller (full blue) attains perfect tracking in stationarity. Note how the MPC controller handles both control and state constraints.

One such technique is the state disturbance observer approach. It attempts to explain all discrepancies in predicted states and output measurements by the presence of constant disturbances:

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t + d_x \\ y_t &= Cx_t + d_y \end{aligned}$$

Compared to our previous model (5.19), we have replaced $B_d d$ by a disturbance $d_x \in \mathbb{R}^n$ and $C_d d$ by another disturbance $d_y \in \mathbb{R}^p$. Hence, if we knew x_t and x_{t+1}, we could immediately compute

$$\begin{aligned} d_x &= x_{t+1} - (Ax_t + Bu_t) \\ d_y &= Cx_t - y_t \end{aligned}$$

and use these as predictions of future disturbances. If we instead estimate x_{t+1} by an observer

$$\hat{x}_{t+1} = A\hat{x}_t + Bu_t + K(y_t - C\hat{x}_t), \qquad \hat{y}_t = C\hat{x}_t$$

the corresponding estimates of the disturbances become

$$\begin{aligned} \hat{d}_x &= \hat{x}_{t+1} - (A\hat{x}_t + Bu_t) = K(y_t - \hat{y}_t) \\ \hat{d}_y &= C\hat{x}_t - y_t \end{aligned}$$

We then use $\hat{d}_x$ and $\hat{d}_y$ instead of $B_d\hat{d}_t$ and $C_d\hat{d}_t$ in the MPC planning problem.

From an implementation perspective, this controller has a slightly reduced complexity since we do not need to estimate the disturbances, but instead compute them from the estimates produced by an observer of reduced order. However, as the next result states, this approach can be seen as a special case of the disturbance observer approach.

Proposition 5.7.3 The state disturbance observer is a particular case of the disturbance observer approach where $B_d = K$, $C_d = I$, while $K_x = K$ and $K_d = I$. If $(A - KC)$ has all eigenvalues inside the unit disc, then so does

$$\begin{pmatrix} A & K \\ 0 & I \end{pmatrix} - \begin{pmatrix} K \\ I \end{pmatrix} \begin{pmatrix} C & I \end{pmatrix}$$

and, in addition,

$$\operatorname{rank}\begin{pmatrix} A-I & K \\ C & I \end{pmatrix} = n+p$$

Proof. With the particular choices of disturbance models and estimator gains, the observer equations for the extended system read

$$\begin{aligned} \hat{x}_{t+1} &= A\hat{x}_t + Bu_t + K\hat{d}_t + K(y_t - C\hat{x}_t - \hat{d}_t) = A\hat{x}_t + Bu_t + K(y_t - C\hat{x}_t) \\ \hat{d}_{t+1} &= \hat{d}_t + (y_t - C\hat{x}_t - \hat{d}_t) = y_t - C\hat{x}_t \end{aligned}$$

For asymptotic stability of the error dynamics of the extended system, note that

$$\begin{pmatrix} A & K \\ 0 & I \end{pmatrix} - \begin{pmatrix} K \\ I \end{pmatrix}\begin{pmatrix} C & I \end{pmatrix} = \begin{pmatrix} A-KC & 0 \\ 0 & 0 \end{pmatrix}$$

Since the matrix is block-diagonal, its eigenvalues are the eigenvalues of $(A-KC)$ and 0. So the extended systems has n eigenvalues at the locations of the eigenvalues of $A-KC$ and p eigenvalues at the origin. Finally, to show that the rank condition holds, we show that any solution

$$\begin{pmatrix} A-I & K \\ C & I \end{pmatrix}\begin{pmatrix} x \\ d \end{pmatrix} = 0$$

must be identical to zero. Specifically, any solution must satisfy $d = -Cx$ and thereby $(A-I-KC)x = -(I-(A-KC))x = 0$. But since $(A-KC)$ is Schur, $I-(A-KC)$ has full rank, the only solution is $x=0$ and thereby $d=0$. The claim is proven. ■

Another common technique for offset-free MPC considers the control increments $\delta u_t = u_t - u_{t-1}$ as inputs and implements a receding-horizon control strategy with the system

$$\begin{pmatrix} x_{t+1} \\ u_t \end{pmatrix} = \begin{pmatrix} A & B \\ 0 & I \end{pmatrix}\begin{pmatrix} x_t \\ u_{t-1} \end{pmatrix} + \begin{pmatrix} B \\ I \end{pmatrix}\delta u_t, \qquad \hat{y}_t = \begin{pmatrix} C & 0 \end{pmatrix}\begin{pmatrix} x_t \\ u_{t-1} \end{pmatrix} \tag{5.29}$$

and cost

$$\sum_{k=0}^{T-1} (y_k - r)^\top Q (y_k - r) + \delta u_k^\top R \delta u_k.$$

for some positive definite matrices Q and R. These implementations typically do not use a terminal cost, which means that they are not covered by our stability guarantees and may need long prediction horizons in practice. However, the absence of terminal cost and constraint simplifies the implementation, since the reference state and reference control are not needed, and the planning problem can be phrased directly in terms of the predicted control increments and extended states. Specifically, when x_t and u_{t-1} are estimated by a one-step ahead predictor

$$\begin{pmatrix} \hat{x}_{t+1} \\ \hat{u}_t \end{pmatrix} = \begin{pmatrix} A & B \\ 0 & I \end{pmatrix}\begin{pmatrix} \hat{x}_t \\ \hat{u}_{t_1} \end{pmatrix} + \begin{pmatrix} B \\ I \end{pmatrix}\delta u_t + \begin{pmatrix} K_x \\ K_u \end{pmatrix}(y_t - C\hat{x}_t) \tag{5.30}$$

the MPC controller solves the planning problem

$$\begin{array}{lll} \text{minimize} & \displaystyle\sum_{k=0}^{T-1} (C\hat{x}_k - r)^\top Q (C\hat{x}_k - r) + \delta\hat{u}_k^\top R \delta\hat{u}_k & \\ \text{subject to} & \hat{x}_{k+1} = A\hat{x}_k + B\hat{u}_{k-1} + B\delta\hat{u}_k & k = 0,\dots,T-1 \\ & \hat{u}_k = \hat{u}_{k-1} + \delta\hat{u}_k & k = 0,\dots,T-1 \\ & M_x\hat{x}_k \le m_x & k = 0,\dots,T-1 \\ & M_u(\hat{u}_{k-1} + \delta\hat{u}_k) \le m_u & k = 0,\dots,T-1 \\ & \hat{u}_{-1} = \hat{u}_t & \\ & \hat{x}_0 = \hat{x}_t & \end{array} \tag{5.31}$$

and applies the control

$$u_t = u_{t-1} + \delta\hat{u}_t^\star \tag{5.32}$$

Although this formulation is not formally a special case of the disturbance compensation approach, one can use the same argument as in Theorem 5.7.2 to establish offset-free tracking.

Proposition 5.7.4 Consider the discrete-time linear system (5.29) with $p = m$, (A, C) detectable and (A, B) reachable, and assume that

$$\operatorname{rank}\begin{pmatrix} A - I & B \\ C & 0 \end{pmatrix} = n + m$$

Let $(\hat{x}_t, \hat{u}_{t-1})$ be estimated by a one-step ahead predictor (5.30) designed to have asymptotically stable estimation error dynamics, and let u_t be given by the model predictive control law (5.31), (5.32), tuned to ensure an asymptotically stable closed loop. Assume that the MPC problem is feasible for all times and unconstrained for all $t \geq T_{\text{lin}}$ for some fixed time T_{lin}. If the closed-loop reaches a steady-state, then $\lim_{t\to\infty} y_t = r$.

Proof. For notational convenience, we introduce

$$A_e = \begin{pmatrix} A & B \\ 0 & I \end{pmatrix}, \qquad B_e = \begin{pmatrix} B \\ I \end{pmatrix}, \qquad C_e = \begin{pmatrix} C & 0 \end{pmatrix}$$

By the PBH tests, reachability of (A, B) implies reachability of (A_e, B_e), and the rank condition and the assumption that (A, C) is detectable implies that (A_e, C_e) is also detectable. Although there is no explicit reference state, we can re-write the problem in terms of $z_e^{\text{ref}} = (x^{\text{ref}}, u^{\text{ref}})$ that satisfies

$$\begin{aligned} C_e z_e^{\text{ref}} &= r \\ z_e^{\text{ref}} &= A_e z_e^{\text{ref}} \end{aligned}$$

By the rank condition, z_e^{ref} exist and is unique for all values of r. Now, by re-phrasing the optimal control problem in linear operation in terms of $z_t = \begin{pmatrix} x_t & u_{t-1} \end{pmatrix}$ and the reference state,

$$\begin{array}{ll} \text{minimize} & \sum_{t=0}^{T} (z_t - z_e^{\text{ref}})^\top C_e^\top Q C_e (z_t - z_e^{\text{ref}}) + \delta u_t^T R \delta u_t \\ \text{subject to} & z_{t+1} - z_e^{\text{ref}} = A_e(z_t - z_e^{\text{ref}}) + B_e \delta u_t \end{array}$$

we see that the optimal control has the form

$$\delta u_t = -L(z_t - z_e^{\text{ref}}) = -L\begin{pmatrix} x_t \\ u_{t-1} \end{pmatrix} + L z_e^{\text{ref}}$$

so that the closed-loop dynamics satisfy $z_{t+1} - z_e^{\text{ref}} = (A_e - B_e L)(z_t - z_e^{\text{ref}})$. By assumption, the MPC controller is tuned so that $A_e - B_e L$ is Schur stable (under the reachability and detectability conditions, this is guaranteed to happen as $T \to \infty$). We are now ready to proof our claim. If the closed-loop system reachabes a steady-state $(\bar{x}, \bar{u})$, the so does the observer, since its error dynamics is asymptotically stable. Let $\hat{z}_e^{\text{eq}}$ be the stationary estimate of the observer and note that K_u has full rank, so stationarity of the observer necessitates that $\bar{y} = \hat{y}^{\text{eq}}$. When the MPC controller uses the stationary estimate, it produces a control input

$$\delta\bar{u} = -L(\hat{z}_e^{\text{eq}} - z_e^{\text{ref}})$$

In addition,

$$\hat{z}_e^{\text{eq}} - z^{\text{ref}} = A_e \hat{z}^{\text{eq}} + B_e \delta\bar{u} - A_e z_e^{\text{ref}} = (A_e - B_e L)(\hat{z}_e^{\text{eq}} - z_e^{\text{ref}})$$

which implies that $\hat{z}_e^{\text{eq}} = z_e^{\text{ref}}$, and thereby that $\bar{y} = \hat{y}^{\text{eq}} = C_e \hat{z}_e^{\text{eq}} = C_e \hat{z}_e^{\text{ref}} = r$. ∎

Note, that although the MPC controller could maintain $u_t = \sum_{k=0}^{t} \delta u_k^\star$, we still need to estimate the full extended state vector and use the estimated values of both x_t and u_{t-1} in the planning problem to ensure offset-free control in the presence of disturbances and model mismatch.

5.8 Tuning rules

A model-predictive controller has a wealth of parameters to tune, ranging from sampling times and horizon lengths to system model, cost criterion, and constraints. Some of these parameters are selected based on the same principles that you are familiar with from other design techniques, while others are new and unique to model-predictive control. In this subsection, we briefly discuss the various parameters and suggest rules-of-thumb of design principles tailored for MPC.

Sample time selection

As discussed in Chapter 1, the sampling time is fundamentally limited by the Nyquist sampling theorem. Once we sample faster than the Nyquist frequency, the sampling time selection involves a compromise between the computational load on the controller (since we have to compute a new control signal in every sampling interval), and the effectiveness in tracking references and rejecting disturbances (since the controller cannot react until the next sampling instance after the disturbance hits the system). In addition, the delay incurred by slow sampling can reduce stability margins of the controller, resulting in increased sensitivity to model uncertainties.

We propose to use least $4-10$ samples in the rise-time that we target for the closed-loop system. Once you have determined enough parameters to be able to design and simulate a simple controller, it is a good idea to add an unmeasured disturbance to the simulation and see if you obtain significant performance improvements when you sample faster. If you do, you should consider decreasing the sampling interval.

System model

Like many advanced control methodologies, LQR and MPC require a model of the system to be controlled. In addition, we may need a system model to evaluate our controller in simulations. The simulation model is typically not the same as the one that we use for design. For example, we may choose to design the controller based on a linear model, but simulate its response on a more detailed nonlinear system model. Another common situation is that we use a simpler model of lower order for control design in order to reduce the complexity of the resulting controller. Finally, we may purposefully perturb the simulation model (either by changing the system matrices, or by adding additional disturbances) to evaluate the robustness of our control design.

Models of dynamical systems are typically derived from physical knowledge or identified from experimental data. Physics-based models are most often developed in continuous-time and therefore need to be linearized and sampled as discussed in Chapter 1. Experimental models, on the other hand, are typically obtained in discrete-time, since data collection is naturally performed by sampling of analog sensor signals. The identification of linear system models from data is well understood in theory and practice, see, e.g. [24] for a thorough treatment.

For model-predictive control, it is particularly attractive to use a subspace identification method since they estimate a state-space model that attempts to match the dynamics in the collected input-output signals. However, subspace methods only return one state-space realization (out of many possible ones), and there is no guarantee that the state vector in the proposed realization corresponds to physical states. It is therefore a good idea to set-up the system identification experiments to measure (a) the actual outputs to be used by the output feedback MPC, and (b) the process states which we would like to constrain. It may also be good for system insight to transform the identified system model to a form where each output in the identification experiment maps to a single state.

Finally, it is good practice to scale the model so that inputs and outputs have unit ranges, i.e.

$$\|u_t\|_\infty \leq 1, \qquad \|y_t\|_\infty \leq 1.$$

Weight matrices

It is convenient to tune the MPC controller for linear operation first. After all, most controllers have set-points that stay constant for extended periods of time, and they spend most of their operational time regulating the system outputs around these set-points.

For the weight matrices, we simply propose to follow the LQR controller tuning procedure described in Chapter 4. At this point, it is useful to evaluate frequency domain properties of the closed-loop system. Similarly, it is advisable to simulate the effect of typical references, disturbances, and measurement noise sequences on the the control signal and the system output.

Horizon lengths

The horizon length T is determined by the slowest process dynamics, since we need to reach the terminal set in T steps. If we have selected the sampling time to have $8-10$ samples in a closed-loop rise time, then we suggest to use a horizon of at least $4-5$ samples. We may need to extend this value in order to handle larger reference changes. However, if there is As the horizon lengths increase, matrices that define the planning problem grow in size and require more memory for storage. In addition, the solution times for the planning problem typically increase, and so does the delay between the time that the sensor signals were read and the new control input can be applied.

In theory, when we increase the prediction horizon, the performance gets closer to that of the optimal infinite-horizon controller. However, this property only holds if the model that we use in our design is an accurate description of the true system dynamics. If the prediction model is inaccurate, extending the horizon length beyond a certain value may actually lead to worse performance. By a similar token, if the system is unstable, then the prediction equations may become very poorly conditioned over long horizons, and we may need to limit our selection.

Terminal set and terminal penalty

The terminal penalty and the terminal set are essential for our theoretical convergence proof. At the same time, we have shown that these features are also, effectively, imposed by having a long prediction horizon. Hence, the terminal penalty and terminal state are most important when we want to keep the control horizon T small.

In order to limit the number of tuning parameters, it is convenient to use the solution to the infinite-horizon Riccati equation (4.9) to define the terminal penalty, and the maximal invariant set of the associated state-feedback law as terminal set. The invariant set is readily calculated off-line, as described in Chapter 2. We can spend a significant amount of computations to find, and possibly also simplify, this invariant set before the MPC controller is configured and deployed.

However, if the MPC controller is intended to track a time-varying reference, then it can be more convenient to rely on explicit constraint checking. For tuning of the constraint checking horizon, we propose to compute the invariant sets for the closed-loop system under the infinite-horizon LQR controller and a set of different reference values. The determinedness indexes of these sets give a reasonable range for the constraint-checking horizon. A more heuristic choice is to let $T+T_c$ be equal to the number of samples in the target closed-loop settling time.

Constraint softening

We recommend to soften all constraints that we can accept to violate, in order to ensure feasibility of the QP also in the presence of unforeseen disturbances. In many cases, only the control magnitude constraints and some safety-critical outputs cannot be softened.

The slack-penalty weights should be large enough to avoid constraint violation whenever this is possible. However, too large slack weights may lead to poor conditioning of the underlying QP (whose other variables, after the scaling proposed above, typically have unit ranges).

Disturbance modeling and integral action

Although there are many ways of incorporating integral action into an MPC controller, we advise to stick with the uniform approach to disturbance compensation that we have used throughout these notes. In this approach, we model the disturbances as outputs of linear systems, estimate the system and disturbance model states using an observer, and compensate for the estimated disturbances in the MPC design. In this framework, constant disturbance models yields integral action.

To avoid performance degradation due to the observer dynamics, we recommend to use a Kalman filter that estimates the states using observations up until the current sampling instant. Note that modeling errors may lead to biased state estimates (irrespective of the estimator that we use), so we may need to physically measure all states that are subject to hard constraints.

Observer gain selection

Even if the observer or filter gains can be selected using a variety of methods, we suggest to use the Kalman filter approach, since it naturally extends to systems with many inputs and outputs. The observer bandwidth should be (significantly) faster than the closed-loop dynamics, while avoiding unnecessary overshoots and oscillations in the observer dynamics.

For the output feedback MPC, it is absolutely essential to simulate the effect of disturbances, measurement noises, and process variations, since it may be much less robust than a controller which has access to the full state vector. It is also advisable to analyze the frequency domain properties of the controller in linear operation (i.e., of the output feedback LQ controller corresponding to the proposed weight matrices and noise covariances).

Choosing the right quadratic programming solver

For problems with relatively long sampling times (say in the order of seconds) and applications where we can afford to perform the computations on a standard computer, this is hardly an issue. For such scenarios, quadratic programming is a mature technology with a wealth of reliable commercial and free solvers. However, for systems that require fast sampling, or applications with significant constraints on the implementation platform, one has to be more careful. If we have a lot of memory, it can be convenient to rely on explicit MPC, since the execution times (and therefore also the computational delay from sensor measurements to actuator commands) is small and easy to bound. If we cannot afford to store the explicit MPC controller, then we have to use on-line optimization. Fast (dual) gradient methods are the simplest to implement and cheapest to execute, but they may generate (slightly) suboptimal or (slightly) infeasible solutions that the controller then needs to cope with. Interior-point methods are reliable and generate accurate solutions, but require rather advanced numerical computations on-line, while active set methods often strike a good balance between numerical accuracy, memory and code footprint, and execution time.

5.9 Exercises

Problem 5.1 Consider an MPC problem:

$$\begin{array}{lll} \text{minimize} & \sum_{k=0}^{T-1} \hat{x}_k^T Q \hat{x}_k + \hat{u}_k^T R \hat{u}_k + \hat{x}_T^T Q_T \hat{x}_T & \\ \text{subject to} & \hat{x}_{k+1} = A\hat{x}_k + B\hat{u}_k & k = 0, 1, \dots, T-1 \\ & -y_{\text{lim}} \le C\hat{x}_k \le y_{\text{lim}} & k = 0, 1, \dots, T \\ & -u_{\text{lim}} \le \hat{u}_k \le u_{\text{lim}} & k = 0, \dots, T-1 \end{array}$$

Assume that you have a system with $m = 1$ inputs, $p = 1$ outputs and a state dimension of $n = 2$.

(a) Formulate the planning problem for $T = 2$ as a quadratic program

$$\begin{array}{ll} \text{minimize} & z^T H z \\ \text{subject to} & Pz \le h \\ & Cz = b. \end{array}$$

where

$$z = \begin{pmatrix} \hat{u}_0 & \hat{x}_1 & \dots & \hat{u}_{T-1} & \hat{x}_T \end{pmatrix}^T.$$

(b) How do the dimensions of the matrices depend on T? What are their dimesions for $T = 5$, $T = 10$, and $T = 50$?

(c) You would also like to impose a rate limitation on the input. Add rows in the P and h matrices so that the input rate is limited to $\pm r_{\text{lim}}$

Problem 5.2 In this exercise, we will explore stability properties of the receding-horizon LQ control law. In each sample, this control policy solves a planning problem on the form

$$\begin{array}{ll} \text{minimize} & \sum_{k=0}^{T-1} \hat{x}_k^\top Q \hat{x}_k + \hat{u}_k^\top R \hat{u}_k + \hat{x}_T^\top Q_T \hat{x}_T \\ \text{subject to} & \hat{x}_{k+1} = A\hat{x}_k + B\hat{u}_k \\ & \hat{x}_0 = x_t \end{array}$$

and implements the first control move, $u_t = u_0^\star$. At the next sampling instant, the controller measures the full state vector again and repeats the procedure.

Consider the specific system given by

$$A = \begin{pmatrix} 4/3 & -2/3 \\ 1 & 0 \end{pmatrix}, \qquad B = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

Note that the open-loop system is stable but oscillatory (in fact, it is also non-minimum phase and generally hard to control). Let the cost be defined by

$$Q = \begin{pmatrix} -2/3 & 1 \end{pmatrix}^\top \begin{pmatrix} -2/3 & 1 \end{pmatrix} + 0.01I, \qquad R = 0.0001, \qquad Q_T = Q.$$

(a) Let $T = 1$. Use the Riccati recursion to compute L such that

$$u_0^\star = -Lx_t$$

Does the receding-horizon control result in an asymptotically stable closed-loop?

(b) What is the smallest value of T for which the receding-horizon policy results in an asymptotically stable closed-loop?

(c) Compute the infinite-horizon control law. Does it yield an asymptotically stable closed-loop?

(d Per definition, the receding-horizon control law should never be able to attain a lower infinite-horizon cost than the optimal controller. Quantify the difference between the two costs when $T = 5$ and $Q_T = Q$. For what initial states do you get the largest difference in cost?

Problem 5.3 Consider the double integrator under zero-order hold sampling

$$
\begin{aligned}
x_{t+1} &= \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} x_t + \begin{pmatrix} 0.5 \\ 1 \end{pmatrix} u_t \\
y_t &= \begin{pmatrix} 1 & 0 \end{pmatrix} x_t
\end{aligned}
$$

(a) Use an algebraic modeling language to implement an MPC controller based on the following planning problem

$$
\begin{array}{lll}
\text{minimize} & \sum_{k=0}^{T-1} \hat{x}_k^\top Q \hat{x}_k + \hat{u}_k^\top R \hat{u}_k + \hat{x}_T^\top Q_T \hat{x}_T & \\
\text{subject to} & \hat{x}_{k+1} = A\hat{x}_k + B\hat{u}_t & k = 0, 1, \dots, T-1 \\
 & |\hat{u}_k| \le u_{\text{lim}} & k = 0, 1, \dots, T-1 \\
 & |C\hat{x}_k| \le y_{\text{lim}} & k = 0, 1 \dots, T
\end{array}
$$

(b) Use $Q = I$, $R = 10I$, $Q_T = Q$ and $T = 10$. Set $u_{\text{lim}} = 1$ and $y_{\text{lim}} = 20$. Simulate the system and plot the control input along with the state trajectory.

(c) You are not happy with the overshoot in the position (the first state). Try to reduce the overshoot by decreasing the value of y_{lim}. Will this work? How small can you make y_{lim}? What stops you from making it even smaller?

(d) Set $y_{\text{lim}} = 12.5$ and reduce the planning horizon to $T = 2$. Explain what you observe!

Problem 5.4 The discrete-time dynamics of an inverted pendulum can be described by the system

$$
\begin{aligned}
x_{t+1} &= \begin{pmatrix} 1.16 & 0.1053 \\ 3.283 & 1.16 \end{pmatrix} x_t + \begin{pmatrix} 0.02332 \\ 0.4785 \end{pmatrix} u_t \\
y_t &= \begin{pmatrix} 1 & 0 \end{pmatrix} x_t
\end{aligned}
$$

The control signal is amplitude limited, $|u_t| \le 10$ and you would like to keep $|y_t| \le \pi/6$.

You are interested in designing a model predictive controller that (approximately) minimizes the infinite-horizon criterion

$$
\sum_{k=0}^{\infty} x_t^\top x_t + u_t^\top u_t
$$

(a) Determine the optimal LQR state-feedback $u_t = -Lx_t$ and draw the intersection of the region where this controller does not saturate, *i.e.*

$$
\{x \mid -Lx \le 10\}
$$

and the set of states that obey the output constraint,

$$
\{x \mid Cx \le \pi/6\}
$$

(b) Compute the maximal invariant set of the closed-loop dynamics $x_{t+1} = (A - BL)x_t$ contained in the region that you constructed in (a). Does $x_0 = \begin{pmatrix} 0 & 5 \end{pmatrix}$ belong to the computed set? What does this imply?

(c) Construct the 10-step controllable set for the control invariant set that you constructed in (b). Does $x_0 = \begin{pmatrix} 0 & 5 \end{pmatrix}$ belong to the computed set? What about $x_0 = \begin{pmatrix} 0 & 7.5 \end{pmatrix}$? What is the implication of this?

(d) Modify your MPC code from Exercise 5.3 to accommodate for a terminal set. Use $T = 10$, $Q_T = P$ where P solves the Riccati equation for the infinite-horizon LQR problem, and use the terminal set computed in (b). Simulate the closed-loop system from initial values $x_0 = \begin{pmatrix} 0 & 5 \end{pmatrix}$ and $x_0 = \begin{pmatrix} 0 & 7.5 \end{pmatrix}$. Comment on what you observe!

(e) In some cases, it is the limited control magnitude that hinders us from reaching the terminal set during the planning horizon, and in other cases, it is the output constraint. Modify your code to soften the output constraints. Use a quadratic slack penalty on the form $\sigma(s) = 100s^2$. Simulate the closed-loop system from the two initial states in (d). Explain what you observe!

Problem 5.5 Consider the discrete-time system

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t, \\ y_t &= Cx_t \end{aligned}$$

with

$$A = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \qquad B = \frac{1}{2}\begin{pmatrix} -1 \\ 1 \end{pmatrix} \qquad C = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \end{pmatrix}.$$

We want to design a model predictive controller that minimizes the cost function

$$J = \sum_{t=0}^{\infty} \frac{1}{2}(y_t^2 + u_t^2).$$

(a) Use the results from Chapter 2 to show that if $u_t = \frac{1}{\sqrt{2}} y_t$, then

$$\sum_{t=0}^{\infty} \frac{1}{2}(y_t^2 + u_t^2) = \|x_0\|^2.$$

(b) A receding-horizon LQ controller is defined at each time step t by

$$u_t = \hat{u}_0^\star$$

where $\{\hat{u}_0^\star, \hat{u}_1^\star, \ldots, \hat{u}_T^\star\}$ is the minimizing argument of

$$\begin{array}{ll} \text{minimize} & \sum_{k=0}^{T-1} \frac{1}{2}(\hat{y}_k^2 + \hat{u}_k^2) + \|\hat{x}_T\|^2 \\ \text{subject to} & \hat{x}_{k+1} = A\hat{x}_k + B\hat{u}_k, \quad \hat{y}_k = C\hat{x}_k \end{array}$$

Show that the control law yields a stable closed-loop system.

(c) The system is now subject to the constraint $-1 \le y_t \le 1$, for all t. Show that if $u_t = \frac{1}{\sqrt{2}} y_t$, then $|y_t| \le 1$ and $|y_{t+1}| \le 1$ imply that

$$-1 \le y_{t+k} \le 1 \qquad\qquad \forall k \ge 0$$

In other words, the set $\mathcal{I} = \left\{ x \mid |Cx| \le 1 \wedge |C(A + BC/\sqrt{2})x| \le 1 \right\}$ is positively invariant.

(d) Now, consider the model predictive control law $u_t = \hat{u}_0^\star$ where $\{\hat{u}_0^\star, \ldots, \hat{u}_T^\star\}$ is the minimizing argument of

$$\begin{array}{lll} \text{minimize} & \sum_{k=0}^{T-1} \frac{1}{2}(\hat{y}_k^2 + \hat{u}_k^2) + \|\hat{x}_T\|^2 & \\ \text{subject to} & \hat{x}_{k+1} = A\hat{x}_k + B\hat{u}_k & k = 0, 1, \ldots, T-1 \\ & \hat{y}_k = C\hat{x}_k & k = 0, 1, \ldots, T \\ & -1 \le \hat{y}_k \le 1, & k = 0, 1, \ldots, T \end{array}$$

Will this MPC controller guarantee a stable closed-loop system? Justify your answer!

Problem 5.6 Consider the linear system

$$x_{t+1} = \begin{pmatrix} 0.8 & -0.1 \\ 1 & 1.6 \end{pmatrix} x_t + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u_t := Ax_t + Bu_t \tag{5.33}$$

subject to the input and state constraints

$$x_t \in X = \left\{ x \in \mathbb{R}^2 \text{ such that } \begin{pmatrix} -5 \\ -5 \end{pmatrix} \leq x \leq \begin{pmatrix} 5 \\ 5 \end{pmatrix} \right\}, \quad \forall t \geq 0$$
$$u_t \in U = \{ u \in \mathbb{R} \text{ such that } -3 \leq u \leq 3 \}, \quad \forall t \geq 0$$

We are interested in solving the following infinite-horizon optimal control problem

$$\begin{array}{ll} \underset{\{u_t\}_{t=0}^{\infty}}{\text{minimize}} & \sum_{t=0}^{\infty} x_t^T Q x_t + u_t^T R u_t \\ \text{such that} & x_{t+1} = \begin{pmatrix} 0.8 & -0.1 \\ 1 & 1.6 \end{pmatrix} x_t + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u_t, \quad t = 0, \dots, \infty \\ & x_t \in X, \quad t = 0, \dots, \infty \\ & u_t \in U, \quad t = 0, \dots, \infty \\ & x_0 = x \end{array} \tag{5.34}$$

where the penalty matrices are given by

$$Q = \begin{pmatrix} 0.8 & 0.18 \\ 0.18 & 1.05 \end{pmatrix}, \quad R = 1.$$

(a) Consider the stabilizing control law

$$u_t = -Lx_t = -\begin{pmatrix} 1 & 1 \end{pmatrix} x_t \tag{5.35}$$

Estimate the infinite-horizon cost of this control law as a function of x. Verify that the proposed feedback law is indeed stabilizing.

(b) Show that the set

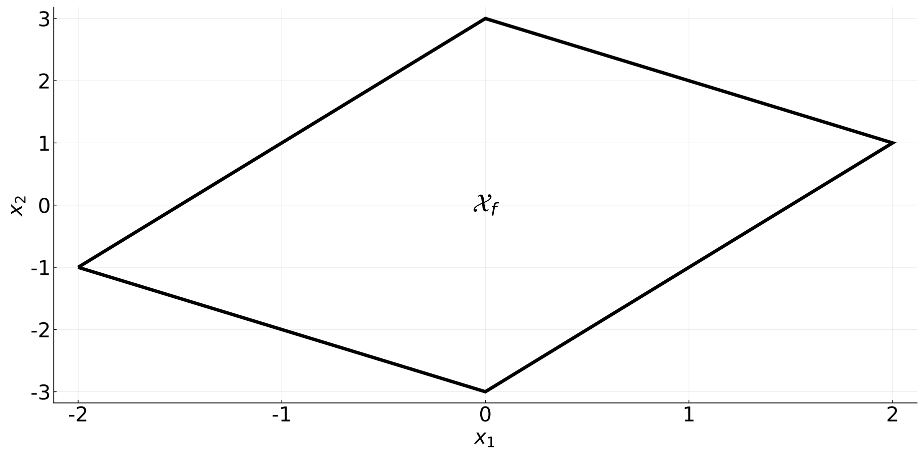
$$X_T = \left\{ x \in \mathbb{R}^2 \text{ such that } \begin{array}{r} x_1 + x_2 \leq 3 \\ x_1 + x_2 \geq -3 \\ 2x_2 - x_1 \leq 3 \\ 2x_2 - x_1 \geq -3 \end{array} \right\}$$

is control invariant under $x_{t+1} = Ax_t + Bu_t$ by verifying that it is positive invariant under $x_{t+1} = (A - BL)x_t$ for L as defined in (5.35). Moreover, show that the feedback law (5.35) is admissible in X_T.

(c) Consider the following finite-dimensional optimization problem

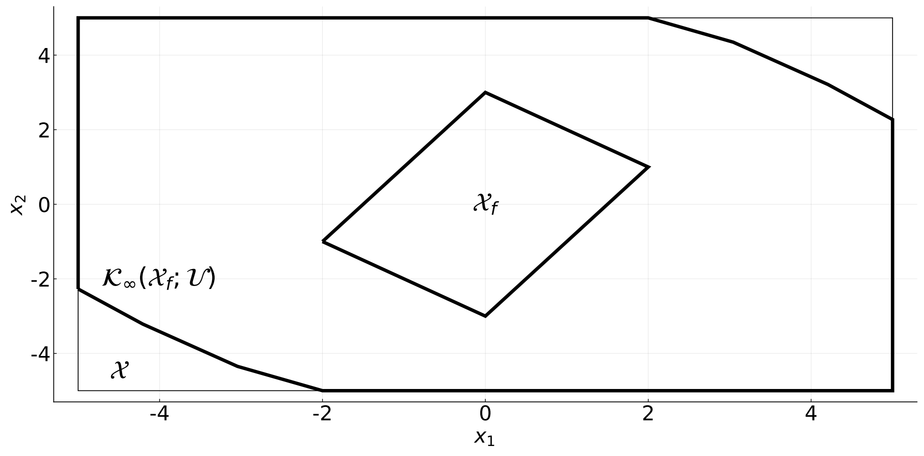
$$\begin{array}{ll} \underset{\hat{u}_0, \dots, \hat{u}_{T-1}}{\text{minimize}} & \sum_{k=0}^{T-1} \hat{x}_t^T Q \hat{x}_t + \hat{u}_t^T R \hat{u}_t + \hat{x}_T^T Q_T \hat{x}_T \\ \text{such that} & \hat{x}_{t+1} = \begin{pmatrix} 0.8 & -0.1 \\ 1 & 1.6 \end{pmatrix} \hat{x}_t + \begin{pmatrix} 0 \\ 1 \end{pmatrix} \hat{u}_t, \quad t = 0, \dots, T-1 \\ & \hat{x}_t \in X, \quad t = 0, \dots, T \\ & \hat{u}_t \in U, \quad t = 0, \dots, T-1 \\ & \hat{x}_T \in X_T \\ & \hat{x}_0 = x \end{array} \tag{5.36}$$

where $Q_T = P$ and P is the solution to the Riccati equation of the infinite-horizon LQR problem in (a) and X_T is the set defined in (b). Explain, with reference to the theory presented



in the course, why a feasible solution to (5.36) defines an input-admissible control sequence $\{\hat{u}_0, \ldots, \hat{u}_{T-1}\}$ that drives the system to a state from which the feedback law (5.39) can stabilize the system while satisfying the state constraints. Comment on the optimality of solutions to (5.36) in relation to (5.34).

(d) The largest control invariant set $\mathcal{C}^{\infty}_{X \times U} \subseteq X$ under $x_{t+1} = Ax_t + Bu_t$ is shown below



Comment on the feasibility of (5.36) for

- $x_0 = \begin{pmatrix} -4 & -4 \end{pmatrix}^T$
- $x_0 = \begin{pmatrix} -4 & 2 \end{pmatrix}^T$

for any given horizon length $T > 0$.

Problem 5.7 Consider the linear system

$$x_{t+1} = \begin{pmatrix} 0.5 & 0 \\ -0.5 & 1.5 \end{pmatrix} x_t + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u_t := Ax_t + Bu_t \tag{5.37}$$

subject to the input and state constraints

$$\begin{aligned} & x_t \in X = \left\{ x \in \mathbb{R}^2 \mid \begin{pmatrix} -5 \\ -5 \end{pmatrix} \leq x \leq \begin{pmatrix} 5 \\ 5 \end{pmatrix} \right\}, \quad \forall t \geq 0 \\ & u_t \in U = \{ u \in \mathbb{R} \mid -2 \leq u \leq 2 \}, \quad \forall t \geq 0 \end{aligned}$$

We are interested in solving the following infinite-horizon optimal control problem

$$\begin{array}{ll} \underset{\{u_k\}_{k=0}^{\infty}}{\text{minimize}} & \sum_{k=0}^{\infty} x_k^T Q x_k + u_k^T R u_k \\ \text{such that} & x_{k+1} = \begin{pmatrix} 0.5 & 0 \\ -0.5 & 1.5 \end{pmatrix} x_k + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u_k, \quad k = 0, \dots, \infty \\ & x_k \in X, \quad k = 0, \dots, \infty \\ & u_k \in U, \quad k = 0, \dots, \infty \\ & x_0 = x \end{array} \tag{5.38}$$

where the penalty matrices are given by

$$Q = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}, \quad R = 1.$$

(a) Consider the stabilizing control law

$$u_t = -Lx_t = -\begin{pmatrix} -1 & 1 \end{pmatrix} x \tag{5.39}$$

Estimate the infinite-horizon cost of applying this control law to (5.37). Verify that the feedback law is indeed stabilizing.

(b) Show that the set

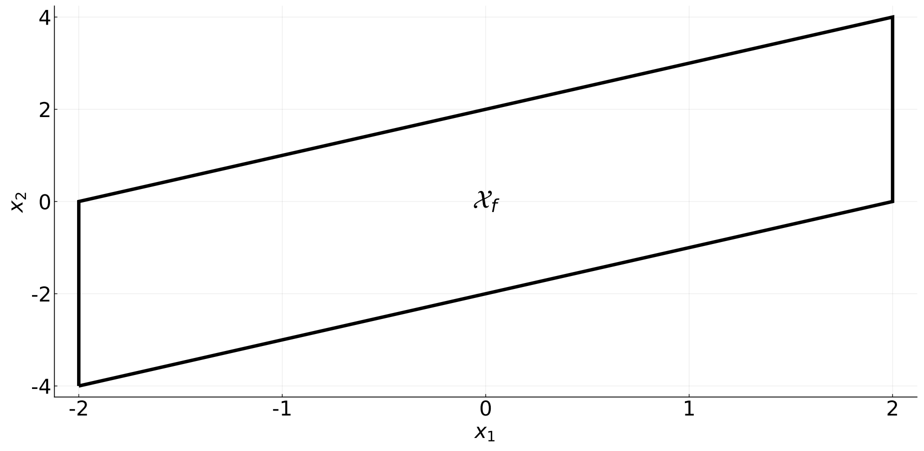
$$X_T = \left\{ x \in \mathbb{R}^2 \;\middle|\; \begin{array}{r} -2 \leq x_1 \leq 2 \\ -2 \leq x_1 - x_2 \leq 2 \end{array} \right\}$$

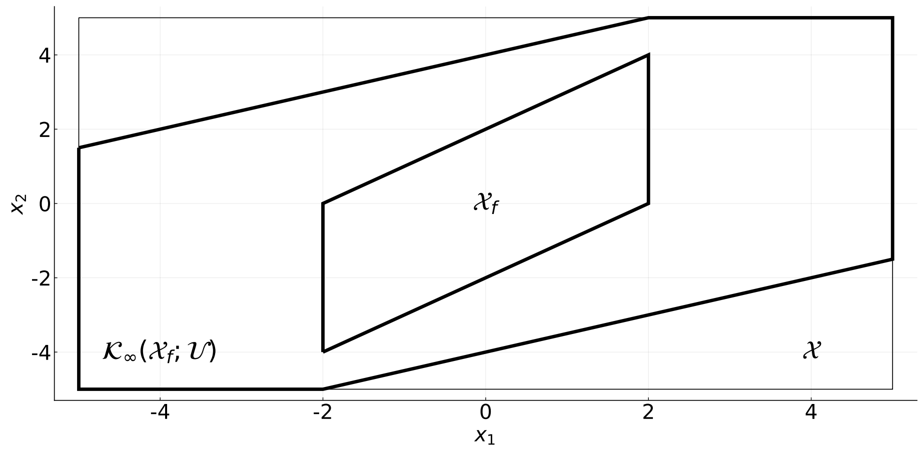
is control invariant under $x_{t+1} = Ax_t + Bu_t$ by verifying that it is positive invariant under $x_{t+1} = (A - BL)x_t$ for L as defined in (5.39). Moreover, show that the feedback law (5.39) is admissible in X_T.

(c) Consider the following finite-dimensional optimization problem

$$\begin{array}{lll} \underset{\hat{u}_0, \dots, \hat{u}_{T-1}}{\text{minimize}} & \sum_{k=0}^{T-1} \hat{x}_k^T Q \hat{x}_k + \hat{u}_k^T R \hat{u}_k + \hat{x}_T^T Q_T \hat{x}_T & \\ \text{such that} & \hat{x}_{t+1} = \begin{pmatrix} 0.5 & 0 \\ -0.5 & 1.5 \end{pmatrix} \hat{x}_t + \begin{pmatrix} 0 \\ 1 \end{pmatrix} \hat{u}_t, & t = 0, \dots, N-1 \\ & \hat{x}_k \in X & k = 0, \dots, T \\ & \hat{u}_t \in U, & k = 0, \dots, T-1 \\ & \hat{x}_T \in X_T & \\ & \hat{x}_0 = x & \end{array} \tag{5.40}$$

where $Q_T = P$ and P is the solution in (a) and X_T is the set defined in (b). Explain, with reference to theory presented in the course, why a feasible solution to (5.40) defines an input-admissible control sequence $\{\hat{u}_0, \dots, \hat{u}_{T-1}\}$ that drives the system to a state from which the feedback law (5.39) can stabilize the system. Comment on the optimality of solutions to (5.40) in relation to (5.38).





(d) The largest control invariant set $\mathcal{C}^\infty_{X\times U} \subseteq X$ under $x_{t+1} = Ax_t + Bu_t$ is shown below. Comment on the feasibility of (5.40) for

- $x_0 = \begin{pmatrix} 4 & -4 \end{pmatrix}^T$
- $x_0 = \begin{pmatrix} 4 & 4 \end{pmatrix}^T$

for any given horizon length $T > 0$.

Problem 5.8 For certain classes of systems, the general stability conditions for the closed-loop system under model-predictive control can be simplified significantly. In this problem, we will study one such class of systems, namely asymptotically stable linear systems with input constraints:

$$x_{t+1} = Ax_t + Bu_t, \qquad u_t \in U = \{u \mid |u| \leq 1\}$$

(a) Let P satisfy the Lyapunov equation

$$A^T PA - P + Q = 0$$

Justify why the MPC controller based on the planning problem

$$\begin{array}{lll} \text{minimize} & \sum_{k=0}^{T-1} \hat{x}_k^\top Q\hat{x}_k + \hat{u}_k^\top R\hat{u}_k + \hat{x}_T^\top P\hat{x}_T & \\ \text{subject to} & \hat{x}_{k+1} = A\hat{x}_k + B\hat{u}_k & k = 0,1,\ldots,T-1 \\ & |\hat{u}_k| \leq 1 & k = 0,1,\ldots,T-1 \\ & \hat{x}_0 = x_t & \end{array}$$

with $Q \succ 0$ and $R \succ 0$ results in an asymptotically stable closed-loop for all horizon lengths.

Note. it is the same matrix Q in the stage cost and the Lyapunov equation.

(b) Let $T = 1$ and assume that $u_t \in \mathbb{R}$ (i.e. that there is a single control input). Determine an explicit expression for how the MPC control policy resulting from the planning problem in (a) depends on the state x_t.

(c) Given the explicit solution in (b), it is attractive to use $T = 1$. Is there any reason to use $T > 1$, even if this means that you will need to solve the planning problem numerically in every sample?
Hint. How does the MPC controller in (a) behave in linear operation (near $x = 0$) as T increases?

Problem 5.9 In the stability proof for linear MPC, we have assumed that the system is reachable. We will now explore some potential challenges that can appear in the absence of reachability.

(a) Use the PBH test to show that the system

$$x_{t+1} = \begin{pmatrix} 1 & 0 \\ 0 & 1/2 \end{pmatrix} x_t + \begin{pmatrix} 1 \\ 0 \end{pmatrix} u_t$$

is not reachable. Is it stabilizable?

(b) Now consider the infinite-horizon LQR problem defined by the cost

$$\sum_{t=0}^{\infty} x_t^T x_t + u_t^T u_t$$

Show that the algebraic Riccati equation admits a diagonal solution, despite your findings in (a). Compute the optimal control law and verify that the closed-loop system is asymptotically stable.

(c) You feel ready to include the control signal constraint

$$|u_t| \leq 1$$

and to attempt to control the system using a MPC control law. For simplicity, you use the terminal set $X_T = \{0\}$. However, you quickly realize that this does not always work. Explain why!

(d) You decide to change the terminal set to

$$X_T = \{x \mid |x_1| \leq 1 \text{ and } |x_2| \leq 1\}$$

Will this work? What is the minimal horizon you need to use to ensure that the initial state

$$x_0 = \begin{pmatrix} 100 \\ 100 \end{pmatrix}$$

is feasible? (you can disregard any other constraints)

Problem 5.10 A model aircraft has the following discrete dynamics in the vertical direction

$$\begin{pmatrix} h_{t+1} \\ \gamma_{t+1} \end{pmatrix} = \begin{pmatrix} 1 & 2 \\ 0 & 0.9 \end{pmatrix} \begin{pmatrix} h_t \\ \gamma_t \end{pmatrix} + \begin{pmatrix} 0.1 \\ 0.1 \end{pmatrix} u_t$$

where h_t is the altitude above ground and γ_t is the flight path angle, see Figure 5.19. Note that a negative angle means that the aircraft is decreasing its altitude. We have the following two constraints on the system

$$\begin{aligned} -0.25 &\leq \gamma_t \leq 0.25 \quad [\text{rad}] \\ 0 &\leq h_t \leq 100 \quad [\text{m}]. \end{aligned} \tag{5.41}$$

We will design a controller to bring the aircraft to land, that is, drive the altitude to $h = 0$.

(a) We first try to solve this problem using LQR, where we minimize

$$J = \sum_{t=0}^{\infty} (x_t^T Q x_t + u_t^T R u_t), \qquad Q = \begin{pmatrix} 1 & 0 \\ 0 & 35 \end{pmatrix}, \ R = \begin{pmatrix} 633 \end{pmatrix}$$

The solution to the discrete-time algebraic Riccati equation is given by

$$P = Q + A^T PA - A^T PB(R + B^T PB)^{-1} B^T PA \implies P \approx \begin{pmatrix} 20 & 240 \\ 240 & 3805 \end{pmatrix}$$

- Given this P, what is the optimal LQR feedback solution $u_t = -Lx_t$?
- In practice, will this controller be good with respect to (5.41)?

(b) Next, we formulate a finite-horizon predictive control law. At each step $u_t = \hat{u}_0^\star$, where $\{\hat{u}_0^\star, \hat{u}_1^\star, \ldots, \hat{u}_{T-1}^\star\}$ is the minimizing argument of

$$\begin{array}{lll} \text{minimize} & \sum_{k=0}^{T-1} (\hat{x}_k^T Q \hat{x}_k + \hat{u}_k^T R \hat{u}_k) + \hat{x}_T^T Q_T \hat{x}_T & \\ \text{subject to} & \hat{x}_{k+1} = A\hat{x}_k + B\hat{u}_k & k = 0, 1, \ldots, T-1 \end{array}$$

where $Q_T = P$ from the previous task. Show that if the system is not subject to any state constraints, then this control law yields a stable closed-loop system.

(c) In addition to the constraints (5.41), we should also limit the vertical velocity at touchdown. The vertical velocity is approximated as

$$\dot{h} = 20 \sin \gamma,$$

where we have assumed that the plane is traveling with a constant velocity $v_0 = 20$ m/s. If we want the vertical velocity at touchdown to be limited to

$$\dot{h} \geq -1 \text{ [m/s]}$$

what is the approximately equivalent constraint

$$\gamma \geq \gamma^{\text{td}} \text{ [rad]}$$

which must be enforced at touchdown, assuming that γ is small?

(d) Next, you will add a constraint on the flight path angle such that there is a continuous transition between the constraint $\gamma \geq \gamma^{\min} = -0.25$ at $h = 10$ m and $\gamma \geq \gamma^{\text{td}}$ at $h = 0$ m. Start by formulating the constraint on the form

$$\gamma + k_1 \cdot h \leq k_2.$$

After this, find the matrix H and vector h such that you can write all the state constraints at time t, including (5.41), on the form

$$Hx_t \leq h \tag{5.42}$$

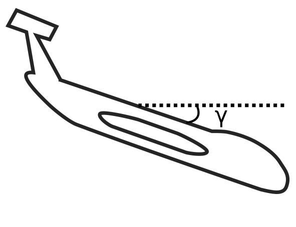


Figure 5.19: The aircraft vertical state is here described with two variables – altitude [m] and flight path angle [rad].

(e) Assume that the mode two control is given by

$$u_t = \begin{pmatrix} -0.04 & 0.6 \end{pmatrix} x_t.$$

Is the state constraint set (5.42) which you found in (d), a suitable terminal set with regards to invariance?

Problem 5.11 In this task, we will use a longitudinal truck model, illustrated in Figure 5.20, where the truck displacement $s(t)$ is given by

$$\dot{s}(t) = v(t). \tag{5.43}$$

The truck velocity is modeled by

$$\dot{v}(t) = -\frac{c_{\text{air}}}{m} v^2(t) + \frac{c_{\text{trac}}}{m} u(t), \tag{5.44}$$

where $v(t)$ is the velocity of the truck, $u(t)$ is the input torque and $m = 40000$ kg is the mass of the vehicle. The velocity and input torque should stay within the limits

$$\begin{aligned} v_{\min} &\le v(t) \le v_{\max}, & v_{\min} &= 55 \text{ km/h}, & v_{\max} &= 85 \text{ km/h} \\ u_{\min} &\le u(t) \le u_{\max}, & u_{\min} &= 0 \text{ Nm}, & u_{\max} &= 1850 \text{ Nm}. \end{aligned}$$

Further, the reference speed is

$$v_{\text{ref}} = 70 \text{ km/h}.$$

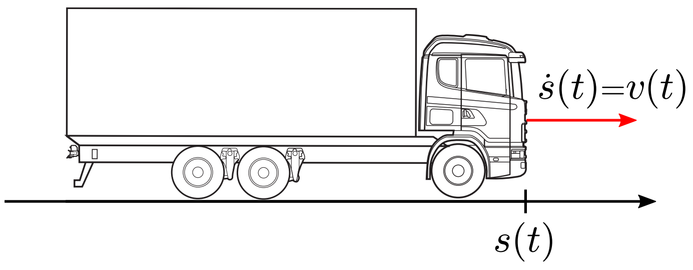


Figure 5.20: The forces acting on a truck moving on a road with slope α.

The standard way to predict the state evolution in MPC is based on differential equations with respect to time

$$\frac{dx}{dt} = f_t(x).$$

Now, we shall instead use equations with respect to the displacement of the truck

$$\frac{dx}{ds} = f_d(x).$$

To do this, we will perform a variable change from velocity $v(t)$ to kinetic energy $E(t)$ as the state variable. Remember that kinetic energy is given by $E = \frac{mv^2}{2}$.

(a) Perform the change of variable described above, starting with the dynamics of equation (5.44). You should get a linear equation of the form

$$\frac{dE}{ds} = A_s E + B_s u.$$

Hint: Derive the relation between the derivatives using

$$\frac{dE}{ds} = \frac{dE}{dt}\frac{dt}{ds} = \frac{1}{v}\frac{dE}{dt}$$

(b) What is the corresponding value of the minimum, maximum, and reference energy, i.e. E_{min}, E_{max} and E_{ref}?

(c) Above, we approximate the maximum input constraint u_{max} as a constant value. However, as illustrated in Figure 5.21, the maximum torque is dependent on the engine speed and thus it is also dependent on the velocity of the truck.
The relationship between engine speed $\omega(t)$ in RPM and velocity (m/s) is given by

$$v(t) = \frac{2\pi \cdot 0.5}{2.45 \cdot 60}\omega(t).$$

Derive a piecewise linear constraint for the maximum input, by calculating constants $m_1, m_2, m_3, k_1, k_2, k_3$ such that

$$\begin{pmatrix} 1 & -k_1 \\ 1 & -k_2 \\ 1 & -k_3 \end{pmatrix} \begin{pmatrix} u(t) \\ E(t) \end{pmatrix} \leq \begin{pmatrix} m_1 \\ m_2 \\ m_3 \end{pmatrix},$$

is the approximation of the upper bound, as illustrated in Figure 5.22.

(d) Write down the optimization problem to be solved in the spatial MPC truck problem, where a tradeoff between input and velocity deviation should be penalized. Assume that the discretized dynamics is given by

$$E_{k+1} = AE_k + Bu_k + B_w w_k$$

where A, B and B_w are the discretized matrices of the spatial dynamics.

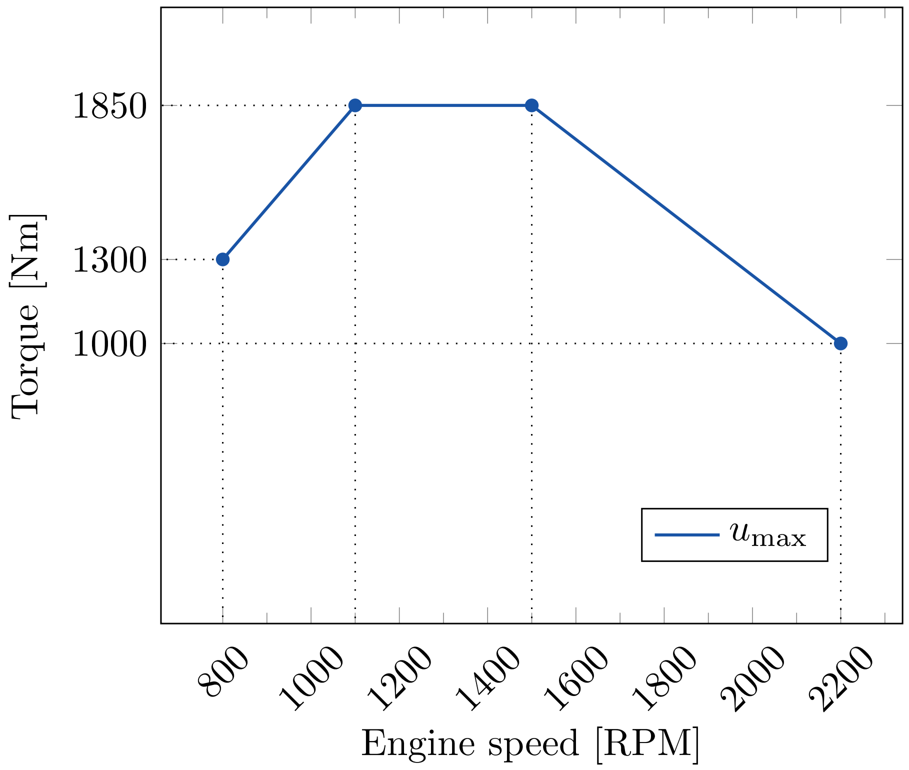


Figure 5.21: The maximum input torque as a function of engine speed.

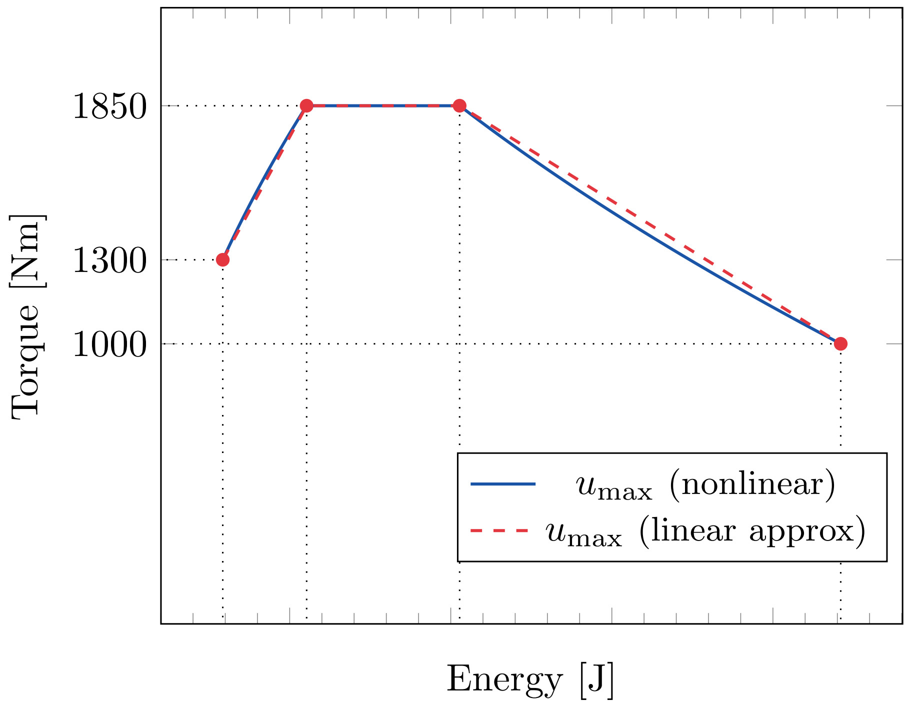


Figure 5.22: The maximum input torque as a function of kinetic energy, and a piecewise linear approximation (red dashed line).

Problem 5.12 This problem considers model-predictive control for ship heading; see Figure 5.23. Specifically, the following linear system

$$x_{t+1} = \begin{pmatrix} 0.9480 & 0 \\ 0.4869 & 1 \end{pmatrix} x_t + \begin{pmatrix} 0.0014 \\ 0.0003 \end{pmatrix} u_t$$

describes the linearized dynamics of a container ship with sampling time 0.5 seconds. The second state is the heading angle (in radians), and the first state describes its rate of change (also referred to as yaw rate). The system input u_t is the rudder angle.

The system is subject to two constraints: the input is limited to ± 35 degrees (that is, $35\pi/180 \approx 0.6109$ radians) and the yaw rate is limited to ± 0.006 radians/second.

(a) Figure 5.24 shows three designs for the weights $R = 10^{-3}$ and $Q = \text{diag}(1, q_{22})$ where $q_{22} \in \{0.1, 1, 10\}$. Which plot (full line, dashed line, dotted line) corresponds to what value of q_{22}? Justify your answer.

(b) Moving on, we focus on the weights $R = 10^{-3}$ and $Q = I$. Verify that

$$P = \begin{pmatrix} 84.8235 & 22.3528 \\ 22.3528 & 10.4803 \end{pmatrix}$$

solves the algebraic Riccati equation for the corresponding LQ-optimal design. Determine the corresponding optimal state feedback gains L.

(c) In the state-space, draw the set of states for which the controller from (b) does not saturate, and for which the yaw rate constraint is also satisfied.

(d) In general, the maximal invariant set for the closed-loop dynamics under LQR control for given constraints is just a subset of the set that you have constructed above. What about this specific case? Is the set that you have constructed in (c) invariant under the LQ-optimal state feedback computed in (b)?

Hint. What is the next state at the vertices of the polyhedron in (c)?

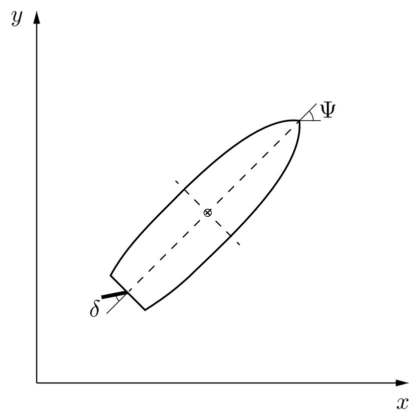


Figure 5.23: Ship heading control.

(e) You consider an MPC design defined by the planning problem

$$\begin{array}{ll} \text{minimize} & \sum_{t=0}^{T-1} x_t^T Q x_t + u_t^T R u_t + x_T^T Q_T x_T \\ \text{subject to} & x_{t+1} = A x_t + B u_t, \quad x_t \in X, \quad u_t \in U \qquad t = 0, \dots, T-1 \\ & x_T \in X_T \end{array}$$

where X_T is the maximal invariant set for the LQR optimal controller, and X and U define the state and control constraints described above.

You select $T = 10$ but find that the set of initially feasible states is too small. Of course, you could increase T in hope of increasing the size of the set, but are there any other changes to your design that could also increase the set of initially feasible states? Please motivate your answer!

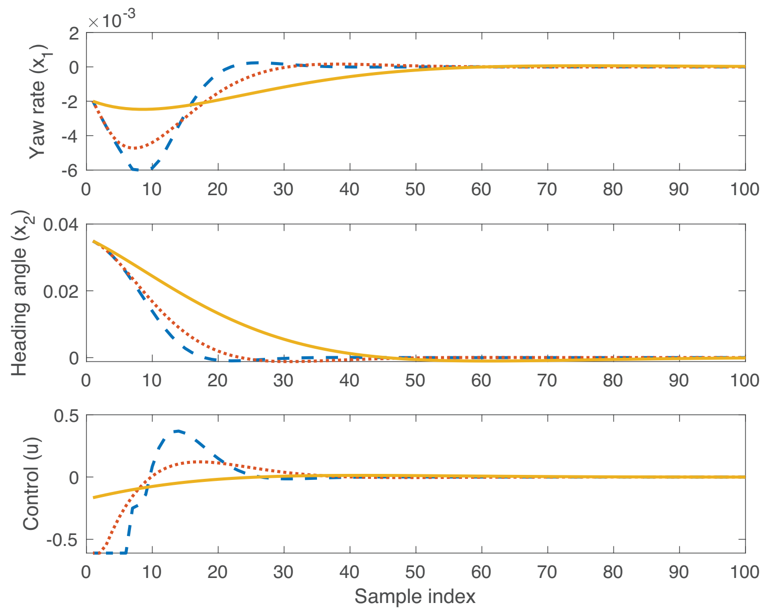


Figure 5.24: MPC designs for three different weight choices in subproblem (a).

Problem 5.13 A reactor is used to decompose a chemical A into a product B. The reaction is exothermic and produces heat, which means that the reactor needs to be cooled during operations.

Our aim is to design a control system that adjusts the flow rate of the product A and of the coolant to follow desired set-points in the production rate of B and in the reactor temperature.

The system dynamics can be written as

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t + w_t \\ y_t &= Cx_t + v_t \end{aligned}$$

where w and v are disturbances acting on the system and

$$A = \begin{pmatrix} 0.9580 & 0 & 0 & 0 \\ 0 & 0.9418 & 0 & 0 \\ 0 & 0 & 0.9048 & 0 \\ 0 & 0 & 0 & 0.9277 \end{pmatrix}, \quad B = \begin{pmatrix} 0.25 & 0 \\ 0.25 & 0 \\ 0 & 0.50 \\ 0 & 0.50 \end{pmatrix},$$

$$C = \begin{pmatrix} 0.1678 & 0.0 & 0.9516 & 0.0 \\ 0.0 & 0.2329 & 0.0 & 0.2890 \end{pmatrix}$$

(a) Implement a reference-tracking MPC controller. Evaluate the design given by $Q = 10I$, $R = I$, $Q_T = Q$ and $T = 10$. You can use a quadratic slack penalty for the reference with weight $\kappa = 100$. The control signals are magnitude limited, $|[u_t]_1| \leq 2$ and $|[u_t]_2| \leq 0.6$.
Simulate the closed-loop system with a reference $\begin{pmatrix} 0.45 & -0.45 \end{pmatrix}$ for the first 25 time steps, and $\begin{pmatrix} 0 & 0 \end{pmatrix}$ for the last 25 time steps. Do you achieve offset-free tracking?

(b) Simulate the effect of an input disturbance on the feed flow (first input). Use the reference $\begin{pmatrix} 0.45 & -0.45 \end{pmatrix}$ for the full simulation, and add the input disturbance of $d = -0.5$ after 25 simulation steps. Do you still achieve offset-free tracking?

(c) Since the disturbance acts on the first input, it is natural to use

$$B_d = \begin{pmatrix} 0.25 & 0 \\ 0.25 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}.$$

Our theoretical result for offset-free tracking requires that the number of disturbances is equal to the number of outputs. Hence, we need to add an additional (artificial) disturbance to our model. Does it matter what output you add the disturbance to?

(d) Augment the system model with the two constant disturbances and design an observer for the augmented system so that the error dynamics has poles in $\{0.50, 0.51, 0.60, 0.61, 0.62, 0.63\}$. Implement an offset-free MPC controller and re-do the simulations in (b). Do you achieve offset-free tracking despite the input disturbance?

Problem 5.14 The stability analysis for MPC presented in this chapter assumes that the planning problem can be solved to optimality and uses the predicted cost $J^\star(x)$ as Lyapunov function. If the optimizer can only be guaranteed to produce a feasible solution with a predicted cost $J_\delta(x)$ where $J_\delta(x) \leq J^\star(x) + \delta$ for some $\delta > 0$, can we still guarantee that the closed-loop system is stable?

To answer this question, assume that the linear system

$$x_{t+1} = Ax_t + Bu_t$$

is controlled using an MPC controller based on the following planning problem

$$\begin{array}{ll} \text{minimize} & \sum_{k=0}^{T-1} \hat{x}_k^\top Q \hat{x}_k + \hat{u}_k^\top R \hat{u}_k + \hat{x}_T^\top Q_T \hat{x}_T \\ \text{subject to} & \hat{x}_{k+1} = A\hat{x}_k + B\hat{u}_k \\ & \hat{x}_k \in X,\ \hat{u}_k \in U,\ \hat{x}_T \in X_T \\ & \hat{x}_0 = x_t \end{array} \tag{5.45}$$

In each sampling interval, we compute a feasible solution to the planning problem that attains an objective value of $J_\delta(x_t) \le J^\star(x_t) + \delta$. The first move in the computed (suboptimal) control sequence is applied, before the procedure is repeated in the next sampling instance. For simplicity, we assume that (A,B) is controllable, that Q and R are positive definite, that $Q_T = P$ where P solve the algebraic Riccati equation for the associated infinite-horizon LQR problem, and that the LQR controller $u_t = -Lx_t$ is admissible in X_T and that X_T is invariant for $x_{t+1} = (A-BL)x_t$.

(a) Demonstrate that the closed-loop under the inexact MPC control satisfies

$$J_\delta(x_{t+1}) - J_\delta(x_t) \le -q(x_t,u_t) + \delta$$

where $q(x,u) = x^\top Qx + u^\top Ru$.

(b) Use the following Lemma (proven in the Exercises of Chapter 2) to prove convergence of the suboptimal MPC controller to a ball around the origin.

Lemma. Let $x_{t+1} = f(x_t)$. Assume that there exists a continuous function V that satisfies

$$\begin{aligned} &\alpha_1\|x\|_2^2 \le V(x) \le \alpha_2\|x\|_2^2 \\ &V(f(x)) - V(x) \le -\beta\|x\|_2^2 + m \end{aligned}$$

for some positive scalars α_1, α_2, m, and $\beta \in (0,\alpha_2)$. Then,

$$\lim_{t\to\infty} \|x_t\|^2 \le \frac{\alpha_2}{\alpha_1\beta} m$$

To be able to apply the lemma, we need to bound J_δ and q. One can show, although the proof is somewhat technical, that

$$J_\delta(x_t) \le c\lambda_{\max}(P)\|x_t\|_2^2$$

for some constant c. Justify why

$$\begin{aligned} J_\delta(x_t) &\ge \lambda_{\min}(Q)\|x_t\|_2^2 \\ -q(x_t,u_t) &\le -\lambda_{\min}(Q)\|x_t\|_2^2 \end{aligned}$$

and determine the solution accuracy δ needed to guarantee that the state converges to the ball

$$\mathcal{B}_\varepsilon = \{x \mid \|x\|_2^2 \le \varepsilon\}$$

Problem 5.15 It is easy to design an open-loop optimal control policy that reaches a desired target in finite time. Specifically, consider the discrete-time linear system

$$x_{t+1} = Ax_t + Bu_t$$

and let $\{\hat{x}_0^\star, \dots, \hat{x}_T^\star, \hat{u}_0^\star, \dots, \hat{u}_{T-1}^\star\}$ be the optimal solution to

$$\begin{array}{lll} \text{minimize} & \sum_{t=0}^{T-1} \hat{x}_t^\top Q\hat{x}_t + \hat{u}_t^\top R\hat{u}_t & \\ \text{subject to} & \hat{x}_{t+1} = A\hat{x}_t + B\hat{u}_t & t = 0,1,\dots,T-1 \\ & \hat{x}_t \in X, \quad \hat{u}_t \in U & t = 0,1,\dots,T-1 \\ & \hat{x}_T = x_{\text{tgt}} & \\ & \hat{x}_0 = x_t & \end{array} \tag{5.46}$$

at $t = 0$. Then, letting $u_t = \hat{u}_t^\star$ for $t = 0,1,\dots,T-1$ ensures that $x_T = x_{\text{tgt}}$. However, we know that open-loop policies are sensitive to model uncertainties, so it is interesting to see if we can get a similar closed-loop behavior using MPC.

(a) Let us first prove that the terminal constraint itself is not enough to ensure finite-time convergence when (5.46) is used as a planning problem for MPC.
To simplify the calculations, we consider the scalar integrator

$$x_{t+1} = x_t + u_t$$

and a receding-horizon control law based on solving the planning problem

$$\begin{array}{ll} \text{minimize} & \sum_{t=0}^{2} \hat{x}_t^2 + \hat{u}_t^2 \\ \text{subject to} & \hat{x}_{t+1} = \hat{x}_t + \hat{u}_t \quad t = 0,1 \\ & \hat{x}_2 = 0 \\ & \hat{x}_0 = x_t \end{array}$$

and then applying $u_t = \hat{u}_0^\star$ (note that this is a special case of (5.46) with $A = B = Q = R = 1$, $x_{\text{tgt}} = 0$ and $T = 2$.). Derive an explicit expression for how the receding-horizon control u_t depends on x_t. Justify why the state under this control law will not reach origin in finite time.

(b) It is, of course, the receding-horizon implementation that hinders you from reaching the target state in finite time: at $t = 0$, you plan to reach the target at $t = T$, at $t = 1$ you plan to reach at $T+1$, etc. Let us instead consider an MPC controller that decreases T in each step. In other words, at $t = 0$ it solves the planning problem (5.46) with $T = T_0$ and applies $u_0 = \hat{u}_0^\star$, at $t = 1$ plans using (5.46) with $T = T_0 - 1$ and applies $u_1 = \hat{u}_0^\star$, etc.
Prove that this policy is recursively feasible, *i.e.*, if you can solve the planning problem from x_0 with horizon T_0, then you can solve it from $x_1 = Ax_0 + Bu_0 = Ax_0 + B\hat{u}_0^\star$ with horizon $T_0 - 1$. Will the state vector reach the target in finite time under this control policy?

(c) A limitation with the policy proposed in (b) is that the tuning of the Q and R matrices does not affect the behaviour of the controller when the controller regulates the state close to the target. Can you explain why? (it may help to revisit your solution to (a)).

(d) To avoid the issue discussed in (c), one may put a lower limit $T_{\min}$ on T. In other words, starting from some T_0, the horizon is initially decreased at every sampling time, until $T = T_{\min}$ (which happens at $t = T_0 - T_{\min}$), after which the horizon is kept at $T_{\min}$ for all future times. Will this policy be able to drive the system state to the target in finite time? Will it behave as the infinite-horizon LQR controller near the target? Justify your answers!

In practice, existing proposals for (near) time-optimal MPC adapt $T_{\min}$ and T in every sample.

III

Appendix

A. Mathematical preliminaries

This appendix summarizes a few key results from basic calculus and linear algebra that we make use of in the proofs. A more complete treatment of these topics can be found in a variety of textbooks.

A.1 Sequences and series

Recall that a sequence of real numbers $\{a_k\}_{k=0}^{\infty}$ is said to converge to a limit L if, for any $\varepsilon > 0$, there is an integer N such that if $n \geq N$, then $|a_n - L| \leq \varepsilon$. To each sequence $\{a_k\}_{k=0}^{\infty}$ we can associate a series $\sum_{k=0}^{\infty} a_k$. We say that the series converges if the sequence of partial sums $s_n = \sum_{k=0}^{n} a_k$ converges. We will make use of the following fact.

Theorem A.1.1 — Cauchy's convergence criterion. If the series $\sum_{k=0}^{\infty} a_k$ of real numbers converges, then $\lim_{k\to\infty} a_k = 0$

There are many ways to test if a series convergence. One such test is by comparison.

Theorem A.1.2 If the series of non-negative terms $\sum_{k=0}^{\infty} b_k$ converges and $|a_k| \leq b_k$ for all k, then the series $\sum_{k=0}^{\infty} a_k$ also converges.

If we use $b_k = |a_k|$ in the comparison theorem, we conclude that if the series is absolutely convergent, i.e. if $\sum_{k=0}^{\infty} |a_k|$ converges, then the series itself $\sum_{k=0}^{\infty} a_k$ is also convergent.

A.2 Linear systems of equations

A linear system of equations consists of a set of m equations where each equation represents a linear relationship among the same set of n variables:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1, \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2, \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= b_m, \end{aligned}$$

Here, $x_1, \dots, x_n$ are the variables, and the coefficients a_{ij} along with the constants b_i are real or complex numbers. The solution of such a system is a set of values for $x_1, \dots, x_n$ that simultaneously satisfies all m equations. These equations are conveniently represented in matrix form

$$Ax = b$$

where

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}, \quad x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}.$$

The next few results are central to characterizing the solution set to systems of linear equations.

Definition A.2.1 — Range of a matrix. The *range* of the matrix $A \in \mathbb{R}^{m\times n}$ is defined as

$$\mathcal{R}(A) = \{Ax \mid x \in \mathbb{R}^n\} \subseteq \mathbb{R}^m$$

The range of A defines the vectors b for which the linear system $Ax = b$ has a solution.

Definition A.2.2 — Matrix rank. For $A \in \mathbb{R}^{m\times n}$, the dimension of $\mathcal{R}(A)$ is denoted $\operatorname{rank}(A)$. The rank of A is equal to the number of linearly independent columns in A.

The number of linearly independent columns in a matrix is sometimes referred to as the *column rank*. In a similar manner, one can define the *row rank* as the number of linearly independent rows in the matrix. It turns out, however, that the column rank is always equal to the row rank, so we simply refer to it as the rank of the matrix. Since row and column ranks are equal, we must have $\operatorname{rank}(A) < \min(m,n)$.

Theorem A.2.1 — Rank and Solutions of Linear Systems. A linear system $Ax = b$ has at least one solution if and only if $\operatorname{rank}(A) = \operatorname{rank}([A|b])$, where $[A|b]$ is the augmented matrix of A and b. If $\operatorname{rank}(A) = n$, the system has a unique solution. If $\operatorname{rank}(A) < n$, the system may have infinitely many solutions.

Definition A.2.3 — Nullspace of a matrix. The *nullspace* of a matrix $A \in \mathbb{R}^{m\times n}$ is defined as

$$\mathcal{N}(A) = \{x \mid Ax = 0\}$$

The nullspace of A is the set of vectors x that are mapped to zero by the linear mapping $x \mapsto Ax$. If a matrix has a non-trivial null-space (a nullspace with dimension one or more), then there will be many vectors x that result in the same $y = Ax$. In particular, if $y = Ax$ and $\tilde{x} \in \mathcal{N}(A)$, then it also holds that $A(x + \tilde{x}) = y$.

Theorem A.2.2 — Dimension Theorem. For any $A \in \mathbb{R}^{m\times n}$, it holds that

$$\dim(\mathcal{R}(A)) + \dim(\mathcal{N}(A)) = n.$$

In words, the dimension of $\mathcal{R}(A)$ plus the dimension of $\mathcal{N}(A)$ equals the number of columns of A.

For square matrices, *i.e.* matrices with the same number of columns and rows, the rank is intimately connected to the matrix determinant.

Definition A.2.4 — Determinant. Let $A = [a_{ij}] \in \mathbb{R}^n$ and $A_{ij} \in \mathbb{R}^{(n-1)\times(n-1)}$ be the matrix obtained by deleting the ith row and jth column from A. If $n = 1$, the determinant of A, denoted $\det(A)$ is the scalar value A itself. If $n > 1$, the determinant can be computed recursively via

$$\det(A) = \sum_{j=1}^{n} (-1)^{\bar{i}+j} a_{\bar{i}j} \det(A_{\bar{i}j})$$

where $\bar{i} \in \{1,2,\dots,n\}$ is an arbitrary row.

The quantity $c_{ij} = (-1)^{i+j}\det(A_{ij})$ is known as the *cofactor* of the element a_{ij}. The definition can be readily used to derive an explicit expression for the determinant of 2×2 matrices

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \Rightarrow \det(A) = a_{11}a_{22} - a_{12}a_{21}$$

by expanding along the first row ($\bar{i} = 1$). Similarly, for a lower-triangular matrix

$$A = \begin{pmatrix} a_{11} & 0 & \dots & 0 \\ a_{12} & a_{22} & \ddots & \vdots \\ \vdots & & \ddots & 0 \\ a_{1n} & a_{2n} & \dots & a_{nn} \end{pmatrix}$$

an expansion along the first row reveals that $\det(A) = a_{11}a_{22}\dots a_{nn}$. It is also possible (albeit with more work) that the determinant satisfies the following properties.

Proposition A.2.3 Let $A, B \in \mathbb{R}^{n\times n}$. Then

(a) $\det(A^\top) = \det(A)$
(b) $\det(AB) = \det(A)\det(B)$

The next results provides a link between determinants and full matrix rank.

Proposition A.2.4 $A \in \mathbb{R}^{n\times n}$ has full rank if and only if $\det(A) \neq 0$.

Definition A.2.5 — Invertible Matrix. A matrix $A \in \mathbb{R}^{n\times n}$ is said to be *invertible* (or non-singular) if there exists another matrix $B \in \mathbb{R}^{n\times n}$ such that $AB = BA = I$, where I is the identity matrix of dimension n. The matrix B is called the inverse of A and is denoted by A^{-1}.

The next results collects some of the many equivalent conditions for a matrix to be invertible.

Theorem A.2.5 — Matrix Invertibility. Let $A \in \mathbb{R}^{n\times n}$. The following statements are equivalent

(a) A is invertible.
(b) $\det(A) \neq 0$.

(c) The columns of A are linearly independent.
(c) $Ax = b$ has a unique solution for every $b \in \mathbb{R}^n$.

When A is invertible, it is given by

$$A^{-1} = \frac{\text{adj}(A)}{\det(A)}$$

where $\text{adj}(A) = [c_{ij}]^\top$ is the adjoint matrix of A, *i.e.* the cofactor matrix of A transposed. Hence,

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \Rightarrow A^{-1} = \frac{1}{\det(A)} \begin{pmatrix} a_{22} & -a_{21} \\ -a_{12} & a_{11} \end{pmatrix}^\top = \frac{1}{a_{11}a_{22} - a_{12}a_{21}} \begin{pmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{pmatrix}$$

Determining the inverses of large matrices by hand can be tedious, unless they have a special structure. However, it is easy to verify that a given matrix expression B is the inverse of A: we simply need to multiply the two matrices and verify that $AB = BA = I$.

Proposition A.2.6 Let $X \in \mathbb{R}^{n \times n}$, $Y \in \mathbb{R}^{k \times k}$ and $Z \in \mathbb{R}^{n \times k}$ be real matrices of appropriate dimensions with X and Y invertible. Then

$$(X + ZYZ^\top)^{-1} = X^{-1} - X^{-1}Z(Y^{-1} + Z^\top X^{-1} Z)^{-1} Z^\top X^{-1} \tag{A.1}$$

and

$$Y^{-1}Z^\top (X^{-1} + ZY^{-1}Z^\top)^{-1} = (Y + Z^\top XZ)^{-1} Z^\top X. \tag{A.2}$$

In many cases, matrices have a structure that makes it natural to partition them into blocks along the rows or columns of the original matrix. For example

$$A = \left(\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right), \qquad B = \left(\begin{array}{c|c} B_{11} & B_{12} \\ \hline B_{21} & B_{22} \end{array}\right)$$

As long as the matrix blocks have compatible dimensions, we can add, subtract and multiply block matrices just as if the blocks would have been elements. This, for the block matrices just defined

$$A + B = \left(\begin{array}{c|c} A_{11} + B_{11} & A_{12} + B_{12} \\ \hline A_{21} + B_{21} & A_{22} + B_{22} \end{array}\right), \qquad AB = \left(\begin{array}{c|c} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ \hline A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{array}\right)$$

The next results summarize important properties of the determinant and the inverse of specific block matrices.

Proposition A.2.7 The determinant of a block triangular matrix is the product of the determinants of its diagonal blocks,

$$\det \left(\begin{array}{c|c} A_{11} & 0 \\ \hline A_{21} & A_{22} \end{array}\right) = \det(A_{11})\det(A_{22}) = \det \left(\begin{array}{c|c} A_{11} & A_{12} \\ \hline 0 & A_{22} \end{array}\right)$$

Proposition A.2.8 Consider the matrix $A \in \mathbb{R}^{n \times n}$ partitioned as

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$$

where $A_{11} \in \mathbb{R}^{k \times k}$ and $A_{22} \in \mathbb{R}^{(n-k)\times(n-k)}$ are both invertible. Then

$$A^{-1} = \begin{pmatrix} (A_{11} - A_{12}A_{22}^{-1}A_{21})^{-1} & -A_{11}^{-1}A_{12}(A_{22} - A_{21}A_{11}^{-1}A_{12})^{-1} \\ -A_{22}^{-1}A_{21}(A_{11} - A_{12}A_{22}^{-1}A_{21})^{-1} & (A_{22} - A_{21}A_{11}^{-1}A_{12})^{-1} \end{pmatrix}$$

A.3 Eigenvalues and eigenvectors

Eigenvalues and eigenvectors are fundamental concepts in linear algebra, particularly in the study of linear transformations.

Definition A.3.1 — Eigenvalue and Eigenvector. Given a square matrix A, a scalar λ is called an *eigenvalue* of A if there exists a non-zero vector v such that

$$Av = \lambda v.$$

The vector v is called an *eigenvector* of A corresponding to the eigenvalue λ.

In some cases, we do not need to know all the eigenvalues, but only bound their magnitude. It can then be convenient to define the spectral radius.

Definition A.3.2 — Spectral Radius. The spectral radius of a matrix A, denoted as $\rho(A)$, is the largest absolute value of its eigenvalues, mathematically given by $\rho(A) = \max\{|\lambda| : \lambda \text{ is an eigenvalue of } A\}$.

Eigenvalues are often computed numerically using specialized routines. The easiest way to compute the eigenvalues of a given matrix by hand is often through its characteristic polynomial.

Definition A.3.3 — Characteristic Polynomial. The *characteristic polynomial* of a square matrix A is

$$p(\lambda) = \det(\lambda I - A),$$

where I is the identity matrix of the same dimension as A, and λ is a scalar.

The roots of the characteristic polynomial are the eigenvalues of the matrix A. Another useful property of the characteristic polynomial is given by the Cayley-Hamilton theorem.

Theorem A.3.1 — Cayley-Hamilton Theorem. Every square matrix A satisfies its own characteristic polynomial. If $p(\lambda)$ is the characteristic polynomial of A, then $p(A) = 0$.

In these notes, we use the Cayley-Hamilton theorem in our discussion about reachability and observability, but we can also use it together with Proposition A.2.3 to prove the following result.

Proposition A.3.2 The matrix A and its transpose $A^\top$ have the same eigenvalues.

A.4 Vector and matrix norms

Vector norms are used to measure the length of vectors.

Definition A.4.1 — Vector norm. A norm on $\mathbb{C}^n$ is a function $\|\cdot\| : \mathbb{C}^n \mapsto \mathbb{R}_+$ that assigns a nonnegative real number to each vector $x \in \mathbb{C}^n$ and satisfies the following conditions:

(a) $\|x\| \geq 0$ and $\|x\| = 0$ if and only if $x = 0$
(b) $\|\lambda x\| = |\lambda|\|x\|$ for every $\lambda \in \mathbb{R}$
(c) $\|x + y\| \leq \|x\| + \|y\|$ for every $y \in \mathbb{C}^n$

Common vector norms are the Euclidean norm, the one norm and the infinity norm, defined as

$$\|x\|_2 = \sqrt{\sum_i x_i^2}, \qquad \|x\|_1 = \sum_i |x_i|, \qquad \|x\|_\infty = \max_i |x_i|$$

respectively. In a similar way, matrix norms are used to measure the "size" of matrices:

Definition A.4.2 — Matrix Norm. A matrix norm on a set of matrices $\mathcal{M}$ is a function $||\cdot||$: $\mathcal{M} \to \mathbb{R}$ that satisfies the following properties for all matrices A, B in $\mathcal{M}$ and scalar c:

- Non-negativity: $||A|| \geq 0$ and $||A|| = 0$ if and only if $A = 0$.
- Scalar multiplication: $||cA|| = |c|||A||$.
- Triangle inequality: $||A+B|| \leq ||A|| + ||B||$.
- Sub-multiplicativity: $||AB|| \leq ||A||||B||$.

A particularly useful class of matrix norms are those that are induced by a given vector norm. These norms measure the maximum amount by which the matrix A can stretch any vector x

Definition A.4.3 — Induced Matrix Norm. For a given vector norm $||\cdot||$, the induced matrix norm $||\cdot||$ is defined for any matrix A as:

$$||A|| = \max_{x \neq 0} \frac{||Ax||}{||x||}$$

where $||Ax||$ and $||x||$ are computed using the same vector norm.

The Euclidean vector norm induces the *spectral norm*

$$||A||_2 = \sqrt{\rho(A^*A)},$$

where A^* is the conjugate transpose of A. From the definition of the induced norm, it follows that

$$||Ax|| \leq ||A||||x||$$

for all x. Matrix norms do not necessarily need to be induced by an underlying vector norm. An important norm that is not induced is the *Frobenius norm*

$$||A||_F = \sqrt{\sum_{i,j} |a_{ij}|^2}$$

where a_{ij} are the entries of A. The following useful result states that if $\rho(A) < 1$, then we can always find a norm such that $||A|| < 1$.

Theorem A.4.1 — Matrix norms and the spectral radius. For every $A \in \mathbb{R}^{n\times n}$ and any $\varepsilon > 0$, there is an induced matrix norm $||\cdot||$ such that

$$||A|| \leq \rho(A) + \varepsilon$$

Moreover, in this norm, it also holds that $||A^\top|| \leq \rho(A) + \varepsilon$.

The introduction of matrix norms allows us to define convergence and matrix sequences and series. We say that a matrix sequence $\{A_k\}_{k=0}^{\infty}$ converges to a matrix L if, for any $\varepsilon > 0$, there is an integer such that if $n \geq N$, then $||A_n - L|| \leq \varepsilon$. The following result generalizes the absolute convergence test to matrix series.

Theorem A.4.2 Consider the sequence $\{A_k\}_{k=0}^{\infty}$ with $A_k \in \mathbb{R}^{n\times n}$ and let $||\cdot||$ be an induced matrix norm on $\mathbb{R}^{n\times n}$. If $\sum_{k=0}^{\infty} ||A_k||$ converges, then so does $\sum_{k=0}^{\infty} A_k$.

A.5 The matrix exponential

The matrix exponential is a matrix-valued function used for solving systems of differential equations. Analogous to how the solution to a scalar differential equation $\dot{x}(t) = ax(t)$ is given in terms of the exponential function $x(t) = e^{at}x(0)$, the solution to a system of linear ordinary differential equations

$$\dot{x}(t) = Ax(t) \tag{A.3}$$

with $x(t) \in \mathbb{R}^n$ is given by

$$x(t) = e^{At}x(0) \tag{A.4}$$

where e^{At} is the matrix exponential (of the matrix At).

Definition A.5.1 The matrix exponential of $M \in \mathbb{R}^{n\times n}$ is defined by the power series

$$e^M = I + M + \frac{1}{2!}M^2 + \frac{1}{3!}M^3 + \cdots$$

which converges for all $M \in \mathbb{R}^{n\times n}$.

We can directly verify that the solution (A.4) satisfies (A.3):

$$\begin{aligned}\dot{x}(t) &= \frac{d}{dt}\left(I + At + \frac{1}{2!}A^2t^2 + \cdots\right)x(0) = \\ &= \left(A + A^2t + \frac{1}{2!}A^3t^2 + \cdots\right)x(0) = \\ &= A(I + At + \frac{1}{2!}A^2t^2 + \cdots)x(0) = Ae^{At}x(0) = Ax(t)\end{aligned}$$

For nilpotent matrices, the definition is practical for evaluating the matrix exponential (since the series converges after a finite number of terms). In general, however, it is often more convenient to use the Laplace transform. To understand how this works, recall that

$$(I - M)^{-1} = I + M + M^2 + M^3 + \cdots$$

provided that the series converges (you can verify the identity by multiplying both sides of the equation with $(I - M)$). Thus,

$$(sI - A)^{-1} = \frac{1}{s}(I - \frac{A}{s})^{-1} = \frac{1}{s}I + \frac{1}{s^2}A + \frac{1}{s^3}A^2 + \cdots$$

which converges for large enough $|s|$. By the inverse Laplace transform, we find

$$\mathcal{L}^{-1}\left((sI - A)^{-1}\right) = I + tA + \frac{t^2}{2!}A^2 + \cdots = e^{At}.$$

The next example demonstrates the two techniques for computing the matrix exponential.

■ **Example A.1** Consider

$$A = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$$

then since $A^k = 0$ for $k \geq 2$ (A is nilpotent) the power series allows us to conclude that

$$e^{At} = I + At = \begin{pmatrix} 1 & t \\ 0 & 1 \end{pmatrix}.$$

Using the inverse Laplace transform approach, we would first evaluate

$$(sI - A)^{-1} = \begin{pmatrix} s & -1 \\ 0 & s \end{pmatrix}^{-1} = \frac{1}{s^2}\begin{pmatrix} s & 1 \\ 0 & s \end{pmatrix} = \begin{pmatrix} 1/s & 1/s^2 \\ 0 & 1/s \end{pmatrix}.$$

Then we determine the matrix exponential by taking the inverse Laplace transform to find

$$e^{At} = \mathcal{L}^{-1}\left((sI - A)^{-1}\right) = \begin{pmatrix} \mathcal{L}^{-1}(1/s) & \mathcal{L}^{-1}(1/s^2) \\ 0 & \mathcal{L}^{-1}(1/s) \end{pmatrix} = \begin{pmatrix} 1 & t \\ 0 & 1 \end{pmatrix}$$

■

A.6 Quadratic forms and functions

A quadratic form $f : \mathbb{R}^n \mapsto \mathbb{R}$ is a polynomial where each term is of order two:

$$f(x) = \sum_{i=1}^{n} \sum_{j\geq i}^{n} m_{ij} x_i x_j = \sum_{i=1}^{n} m_{ii} x_i^2 + \sum_{i=1}^{n} \sum_{j>n}^{n} m_{ij} x_i x_j.$$

It is convenient to represent such functions in matrix form

$$f(x) = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}^\top \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{12} & p_{22} & & p_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{1n} & p_{2n} & \cdots & p_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = x^\top P x$$

Note that

$$x^\top P x = \sum_{i=1}^{n} \sum_{j=1}^{n} p_{ij} x_i x_j = \sum_{i} p_{ii} x_i^2 + \sum_{i=1}^{n} \sum_{j>i}^{n} (p_{ij} + p_{ji}) x_i x_j$$

Hence, the two parameterizations are equal if we let $p_{ii} = m_{ii}$ and $p_{ij} = p_{ji} = m_{ij}/2$. This means that we can always assume that P is a symmetric matrix, i.e. that $P = P^\top$.

When performing optimization of functions of several variables, we repeatedly need to compute gradients and Hessians. Let $f : \mathbb{R}^n \mapsto \mathbb{R}$. Then its gradient at x, $\nabla f(x) : \mathbb{R}^n \mapsto \mathbb{R}^n$, is defined by

$$\nabla f(x) = \begin{pmatrix} \frac{\partial f(x)}{\partial x_1} \\ \frac{\partial f(x)}{\partial x_2} \\ \vdots \\ \frac{\partial f(x)}{\partial x_n} \end{pmatrix}$$

The quadratic form $f(x) = x^\top P x$ has gradient

$$\nabla f(x) = \begin{pmatrix} \frac{\partial f(x)}{\partial x_1} \\ \frac{\partial f(x)}{\partial x_2} \\ \vdots \\ \frac{\partial f(x)}{\partial x_n} \end{pmatrix} = \begin{pmatrix} 2p_{11}x_1 + (p_{12}+p_{21})x_2 + (p_{13}+p_{31})x_3 + \cdots + (p_{1n}+p_{n1})x_n \\ (p_{12}+p_{21})x_1 + 2p_{22}x_2 + (p_{23}+p_{32})x_3 + \cdots + (p_{2n}+p_{n2})x_n \\ \vdots \\ (p_{1n}+p_{n1})x_1 + (p_{2n}+p_{n2})x_2 + (p_{3n}+p_{n3})x_3 + \cdots + 2p_{nn}x_n \end{pmatrix}$$

So if P is symmetric, *i.e.* $p_{ij} = p_{ji}$ then we find that

$$\nabla f(x) = 2Px$$

A quadratic function is a polynomial where each term is at most two. We can write it as a sum of a quadratic form, a linear form, and a constant:

$$f(x) = x^\top P x + 2q^\top x + r \tag{A.5}$$

A simple calculation reveals that the linear form $q^\top x$ has gradient q, so the (A.5) has gradient

$$\nabla f(x) = 2Px + 2q$$

A.7 Positive definite matrices and functions

Definition A.7.1 — Positive definite functions. We say that the function $f : \mathbb{R}^n \mapsto \mathbb{R}$ is *positive semidefinite* if

$$f(x) \geq 0 \qquad \forall x \in \mathbb{R}^n$$

and that f is *positive definite* if

$$f(x) > 0 \qquad \forall x \in \mathbb{R}^n \text{ with } x \neq 0$$

Since a quadratic form $f(x) = x^\top Px$ is characterized by the matrix P, it is natural to link positive definiteness of the quadratic form to properties of P. This leads to the following definition.

Definition A.7.2 A square symmetric matrix $P = P^\top \in \mathbb{R}^{n\times n}$ is called *positive semidefinite* if $f(x) = x^\top Px$ is a positive semidefinite function; it is called *positive definite* if $f(x) = x^\top Px$ defines a positive definite function.

We use the notation $P \succeq 0$ to denote that P is positive semidefinite, and $P \succ 0$ to denote that it is positive definite. The following result will be useful.

Theorem A.7.1 The square symmetric matrix $P = P^\top \in \mathbb{R}^{n\times n}$ is positive semidefinite if and only if all its eigenvalues are non-negative and real. A positive semidefinite matrix admits a representation $P = R^\top R$ where $R \in \mathbb{R}^{n\times n}$.

The square symmetric $P = P^\top \in \mathbb{R}^{n\times n}$ is positive definite if and only if all its eigenvalues are positive and real. A positive definite matrix admits a representation $P = R^\top R$ where $R \in \mathbb{R}^{n\times n}$ is of full rank. A positive definite matrix is invertible (and its inverse is itself positive definite).

This theorem tells us that a positive definite quadratic form $x^\top Px$ can be written as $\sum_{i=1}^n z_i^2 = \sum_{i=1}^n (r_i^\top x)^2$ for some vectors $r_i \in \mathbb{R}^n$. The quadratic form is zero for x which satisfy $r_i^\top x = 0$ for all i; in a positive definite quadratic form, the matrix R spans $\mathbb{R}^n$ and $Rx = 0$ only for $x = 0$.

The definitions of positive definite matrices and functions extend to complex matrices and vectors. In particular, we say that a complex-valued matrix $M \in \mathbb{C}^{n\times n}$ is positive definite if z^*Mz is real and positive for all non-zero complex column vectors $z \in \mathbb{C}^n$, where z^* is the complex conjugate transpose of z. We will only use the fact that if $P \in \mathbb{R}^{n\times n}$ is positive definite, then

$$z^*Pz > 0$$

for all $z \in \mathbb{C}^n$ with $z \neq 0$. This follows by writing $z = x + iy$ with $x, y \in \mathbb{R}^n$ and evaluating

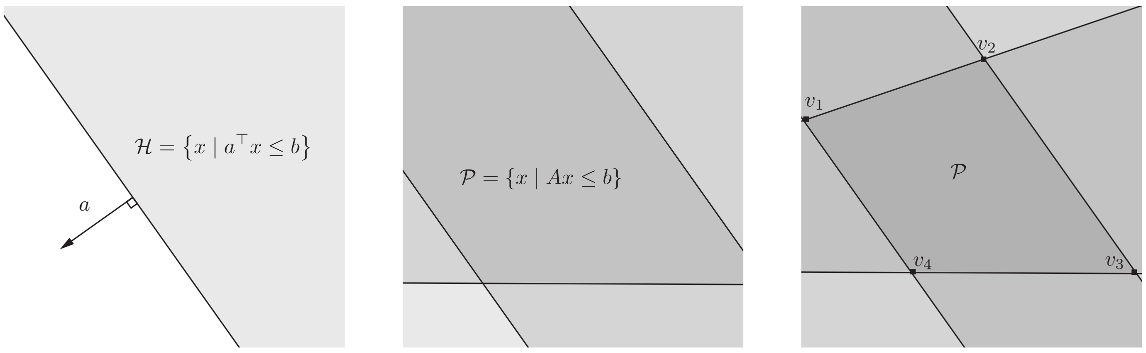
$$z^*Pz = (x^\top - iy^\top)P(x+iy) = x^\top Px + y^\top Py.$$

The right-hand side is positive unless $x = y = 0$, which proves our claim.

B. Polyhedra and ellipsoids

This appendix reviews some basic properties of polytopes, polyhedra and ellipsoids.

B.1 Polyhedra and polytopes



$\mathcal{H} = \{x \mid a^\top x \leq b\}$
a
$\mathcal{P} = \{x \mid Ax \leq b\}$
v_1
v_2
$\mathcal{P}$
v_4
v_3

Figure B.1: Halfspace (left), polyhedron (middle), and polytope (right).

Halfspaces, polytopes and polyhedra, shown in Figure B.1 and defined below, give a geometrical meaning to the solution set to one or more linear inequalities.

Definition B.1.1 A (closed) *half-space* is the solution set to a linear inequality

$$\mathscr{H}(a,b) = \left\{x \in \mathbb{R}^n \mid a^\top x \leq b\right\}$$

The vector $a \in \mathbb{R}^n$ is the *normal* vector of the half-space.

Definition B.1.2 A *polyhedron* is the solution set to a finite number of linear inequalities

$$\mathscr{P} = \{x \in \mathbb{R}^n \mid Ax \leq b\} = \left\{x \in \mathbb{R}^n \mid a_i^\top x \leq b_i, \quad i = 1, \ldots, m\right\}$$

where $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$. Equivalently, it is the intersection of a finite number of halfspaces

$$\mathcal{P} = \mathcal{H}(a_1,b_1) \cap \mathcal{H}(a_2,b_2) \cap \cdots \cap \mathcal{H}(a_m,b_m)$$

Note that any convex set (e.g. the disk) is the intersection of all half-spaces that include it. It is only when the set is finitely generated that we call it a polyhedron.

Definition B.1.3 A *polytope* $\mathcal{P}(A,b)$ is a bounded polyhedron, *i.e.* one that does not contain any ray $\{x+t(y-x) \mid x,y \in \mathcal{P}, \quad t \geq 0\}$.

The representation (A,b) of a polyhedron $\mathcal{P}$ is not unique. For example, A and b can be multiplied by the same positive diagonal matrix without changing the underlying set. Hence, if $0 \in \mathcal{P}$ then $b \geq 0$ and we can always normalize the description to have $b = 1$. More interestingly, a representation can contain *redundant constraints*, *i.e.* constraints that do not alter the underlying polyhedron. Examples of redundant constraints are repeated intersections with the same halfspace, or intersections with a halfspace $\mathcal{H}(\tilde{a},\tilde{b})$ that contains $\mathcal{P}$ in its strict interior. For example,

$$\mathcal{P} = \{x \mid x \leq 0\} = \{x \mid x \leq 0\} \cap \{x \mid x \leq 0\} = \{x \mid x \leq 0\} \cap \{x \mid x \leq 1\}$$

are three representations of the same set (the set of non-positive reals).

To determine if $\mathcal{H}(\tilde{a},\tilde{b})$ is redundant for $\mathcal{P}$, we can simply solve a linear program

$$\begin{array}{ll} \text{maximize} & \tilde{a}^\top x \\ \text{subject to} & Ax \leq b \end{array}$$

to determine the maximal value of $\tilde{a}^\top x$ for $x \in \mathcal{P}$. If this value is strictly smaller than $\tilde{b}$, then $\mathcal{P} \subset \mathcal{H}(\tilde{a},\tilde{b})$, so the constraint is redundant. To remove redundant constraints from a given representation (A,b) of $\mathcal{P}$ we can check, for each of the m constraints defined by (A,b), if it is redundant relative to the polyhedron $\mathcal{P}'$ obtained by removing the corresponding row from (A,b).

The projection of a polyhedron

$$\mathcal{P} = \{(x,u) \mid Ax + Du \leq b\}$$

onto x is defined as

$$\mathsf{proj}_x(\mathcal{P}) = \{x \mid (x,u) \in \mathcal{P} \text{ for some } u\}.$$

This projection is computed by eliminating u from the inequalities. To understand how this is done, consider the case where u is scalar and each element d_i of D is non-zero. Then, each inequality $a_i^\top x + d_i u \leq b_i$ implies a constraint on u. Specifically, if $d_i > 0$, then

$$u \leq \frac{b_i - a_i^\top x}{d_i} := U_i(x)$$

while if $d_i < 0$

$$u \geq \frac{b_i - a_i^\top x}{d_i} := L_i(x).$$

The projection is the set of x for which u can satisfy these bounds,

$$\mathsf{proj}_x(\mathcal{P}) = \{x \mid \max_{i\in I_l} L_i(x) \leq \min_{i\in I_u} U_i(x)\}$$

This set is a polyhedron described by the $|I_l| \times |I_u|$ inequalities

$$L_i(x) - U_j(x) \leq 0 \qquad i \in I_l,\ j \in I_u$$

In practice, many of these inequalities are redundant and can be removed by the procedure discussed earlier. In the case that some $d_i = 0$, these inequalities are already independent of u and should be included in the definition of $\mathrm{proj}_x(\mathcal{P})$.The procedure that we have just described is known as Fourier-Motzkin elimination and is treated in more detail in, *e.g.*, [36].

Fourier-Motzkin elimination is a useful procedure when we do not have too many constraints or eliminate too many variables. In the worst-case, the number of inequalities go from m to $m^2/4$ in a single round of elimination; k rounds of this procedure (eliminating k variables sequentially without removing redundant constraints) would result in $m^{2^k}/(2^{2^{k+1}-2})$ inequalities (a doubly exponential growth in k). Although the practical performance of Fourier-Motzkin tends to be better than this worst-case, it remains computationally challenging to project many variables in polyhedra defined by many inequalities (for example, in high dimensions).

As an alternative, one can also represent a polytope as the convex hull of a finite number of extreme points $v_1, \dots, v_K$, *i.e.*

$$\mathcal{P} = \{x = \sum_{i=1}^{K} \alpha_i v_i \text{ for some } \alpha_i \in [0,1] \text{ with } \sum_{i=1}^{K} \alpha_i = 1\}$$

The extreme points v_i are called *vertices* and the associated representation a V-representation (in contrast to the H-representation that we have explored above). The V-representations of polyhedra (which may be unbounded) are slightly more complex and also involve extremal rays; see [**Zie;95**] for details. We will not explore the V-representation in detail here, but take the opportunity to highlight a few important facts. First, there are polytopes with small H-representations and complex V-representations. For example, the infinity-norm ball $\|x\|_\infty \leq 1$ can be described by the intersection of $2n$ hyperplanes $\mathcal{H}_i^{\pm} = \{x \mid \pm[x]_i \leq 1\}$, but it has 2^n extreme points v with $[v]_i = \pm 1$. There are also polyhedra with compact V-representations but large H-representations. Second, some operations that are hard to perform on the H-representation may be easy to perform using the V-representation (and vice versa). Since the transformation between H and V representations may be computationally expensive, it is important to use the representation that is most efficient (with respect to memory and computations) for each particular problem.

B.2 Ellipsoids

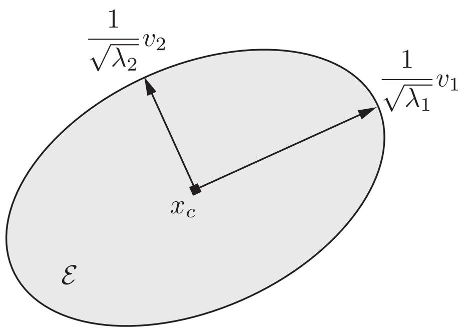

$\frac{1}{\sqrt{\lambda_2}}v_2$
$\frac{1}{\sqrt{\lambda_1}}v_1$
x_c
$\mathcal{E}$

Figure B.2: Ellipsoid characterized by its center, its semi-axes and their lengths

In a similar way as the solution set to a linear inequality defines a halfspace, the solution set to a (convex) quadratic inequality defines an ellipsoid:

Definition B.2.1 An ellipsoid is the solution set to a convex quadratic inequality

$$\mathcal{E} = \left\{ x \mid x^T P x + 2q^\top x + r \leq 0 \right\} \tag{B.1}$$

for some positive definite matrix P.

Let us first consider the case $q = 0$ and $r = -1$, *i.e.* ellipsoids centered around the origin:

$$\mathcal{E} = \left\{ x \mid x^\top P x \leq 1 \right\}$$

Since P is positive definite, it is real and symmetric and admits and eigendecomposition

$$P = V \Lambda V^\top$$

where V is an orthogonal matrix whose columns are the eigenvectors of P and Λ is a diagonal matrix whose entries are the eigenvalues of P. Thus, in a coordinate system described by $z = V^\top x$,

$$\mathcal{E} = \{ z \mid \lambda_1 z_1^2 + \cdots + \lambda_n z_n^2 \leq 1 \}$$

defining an ellipsoid with axes aligned to the coordinate system and lengths proportional to $1/\sqrt{\lambda_i}$. In the original coordinates $x = Vz$, this corresponds to an ellipsoid whose axes are aligned with the eigenvectors v_i of P and stretched a factor $1/\sqrt{\lambda_i}$; see Figure B.2.

Introducing $R = \Lambda^{1/2} V^\top$, we can re-write the ellipsoid on the form

$$\mathcal{E} = \left\{ x \mid \|Rx\|_2^2 \leq 1 \right\} = \left\{ x = R^{-1} w \mid \|w\|_2^2 \leq 1 \right\}$$

The final representation is convenient for plotting the ellipsoid, since it represents $\mathcal{E}$ as a linear transformation of the unit disc.

Let us now go back to the original description (B.1) and assume that $q \neq 0$. Then, by Lemma 3.1, we can write

$$\mathcal{E} = \left\{ x \mid (x - x_c)^\top P (x - x_c) + r - x_c^\top P x_c \leq 0 \right\} \tag{B.2}$$

where $x_c = P^{-1} q$. Hence, this ellipsoid is centered around x_c and has semi-axes aligned with the eigenvectors of P. To get the right semi-axes lengths, we must divide both sides of the inequality in (B.2) by $x_c^\top P x_c - r$, i.e. consider the eigenvalues of $P/(q^\top P^{-1} q - r)$.

C. Mathematical programming

This section contains a tutorial introduction to selected topics in mathematical programming and convex optimization. They are written to support the developments in these notes, but are by no means complete. We refer to the literature, such as [6, 10], for details.

C.1 Optimality conditions for unconstrained optimization

Consider the minimization of a multivariable function $f(x)$. We write this problem as

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \quad f(x) \tag{C.1}$$

and refer to $f : \mathbb{R}^n \mapsto \mathbb{R}$ as the *objective function* and $x \in \mathbb{R}^n$ as the *decision vector*. We are thus interested in finding the vector x which attains the smallest possible value of f. If f is continuously differentiable, then any minimizer $x^\star$ must satisfy the *first-order optimality condition*

$$\nabla f(x^\star) = 0. \tag{C.2}$$

(otherwise, we could improve the objective by adjusting $x^\star$ in the direction of the negative gradient). However, this condition is not sufficient for guaranteeing optimality. A vector x for which $\nabla f(x) = 0$ could be a minimum, a maximum or a saddle point; see Figure C.1. In order to say more, we must put additional restrictions on the objective function, or further conditions on $x^\star$. In these notes we will therefore limit the objective functions and constraints to be convex, as defined next.

C.2 Convexity and optimization

We begin by the definition of convex sets.

Definition C.2.1 The set $X \subseteq \mathbb{R}^m$ is *convex* if for any $x_1, x_2 \in X$ and every $\theta \in [0,1]$ we have $\theta x_1 + (1-\theta)x_2 \in X$.

Thus, a set X is convex if the line segment between any two points in X also lies in X; see Figure C.2.

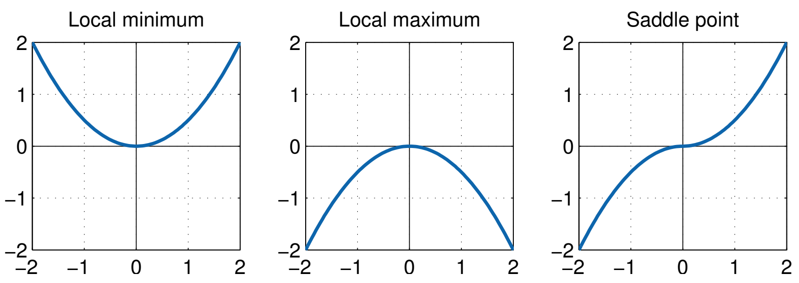


Figure C.1: Three continuously differentiable functions, whose gradients vanish at the origin. In the different functions, the origin is a local minima, local maxima and a saddle points, respectively.

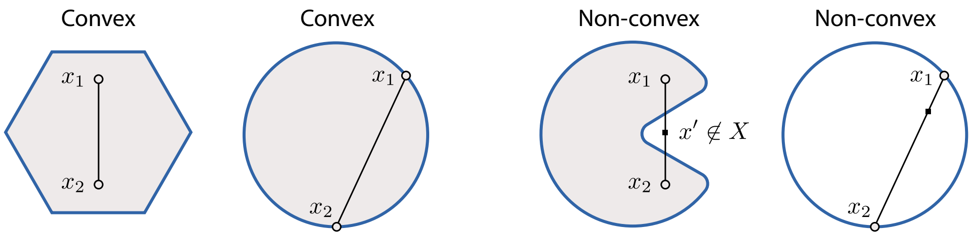


Figure C.2: A set is convex if every line segment between points in the set also lies in the set. The unit disc (the second set from the left) is convex, while the unit cirle (the rightmost set) is not.

■ **Example C.1** One of the most basic convex sets is the half-space, defined by the solution set of a linear inequality

$$X = \left\{ x \mid a^\top x \le b \right\}.$$

To prove that halfspaces define convex sets, we can use the definition and consider $y = \theta x_1 + (1 - \theta)x_2$ with $x_1, x_2 \in X$. Then, for $\theta \in [0, 1]$ it holds that

$$a^\top y = a^\top(\theta x_1 + (1-\theta)x_2) = \theta a^\top x_1 + (1-\theta)a^\top x_2 \le \theta b + (1-\theta)b = b$$

where the inequality follows since x_1 and x_2 both lie in X (and therefore satisfy $a^\top x_1 \le b$ and $a^\top x_2 \le b$, respectively). Hence, $y \in X$ for every value of $\theta \in [0, 1]$, so X is a convex set. ■

Definition C.2.2 The function $f : \mathbb{R}^n \mapsto \mathbb{R}$ is *convex* if its domain is a convex set and it holds that for all $x_1, x_2 \in \mathrm{dom} f$ and for every $\theta \in [0, 1]$ we have

$$f(\theta x_1 + (1-\theta)x_2) \le \theta f(x_1) + (1-\theta) f(x_2) \tag{C.3}$$

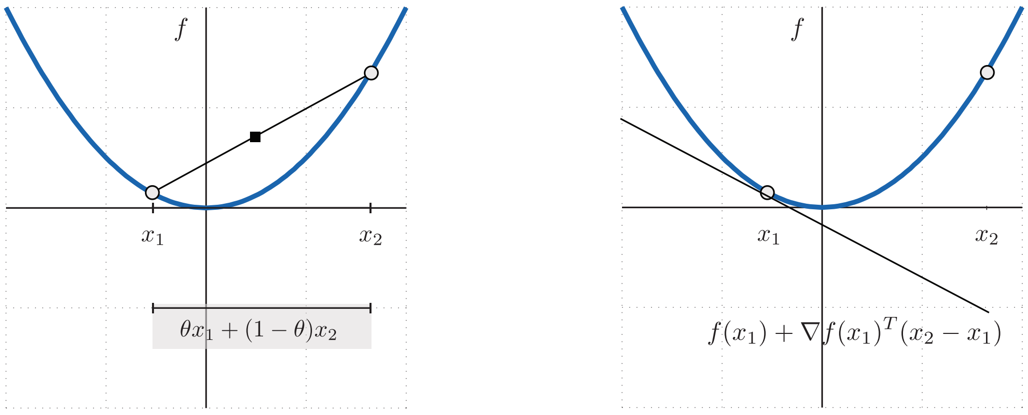
The geometric interpretation of (C.3) is that the line segment between $(x_1, f(x_1))$ and $(x_2, f(x_2))$ always lies above the graph of f; see Figure C.3 (right).

■ **Example C.2** By a similar calculation as in Example C.1, we can easily show that affine functions

$$f(x) = a^\top x + b$$

are convex. We will now show that quadratic functions

$$f(x) = x^\top Px$$



f
x_1
x_2
$\theta x_1 + (1-\theta)x_2$
f
x_1
x_2
$f(x_1) + \nabla f(x_1)^T(x_2 - x_1)$

Figure C.3: A function f is convex if the line segment between $(x_1, f(x_1))$ and $(x_2, f(x_2))$ lies above the graph of f for every x_1 and x_2 in its domain (left). A continuously differentiable function f is convex if its linearization in any point x_1 is a global lower bound of the function (right).

are convex when P is a semidefinite matrix. We then have that

$$
\begin{aligned}
&f(\theta x_1 + (1-\theta)x_2) - \theta f(x_1) - (1-\theta)f(x_2) = \\
&= \theta^2 x_1^\top P x_1 + (1-\theta)^2 x_2^\top P x_2 - 2\theta(1-\theta)x_1^\top P x_2 - \theta x_1^\top P x_1 - (1-\theta)x_2^\top P x_2 = \\
&= \theta(\theta-1)\left[(x_1-x_2)^\top P(x_1-x_2)\right] \le 0,
\end{aligned}
$$

where the last inequality follows since $\theta(\theta-1) < 0$ for $\theta \in [0,1]$ and since we have assumed P to be positive semidefinite.

It follows from the definition that the sum of two convex functions is also convex. Hence, the more general quadratic form

$$f(x) = \frac{1}{2}x^\top P x + q^\top x + r$$

is convex whenever P is positive semidefinite. ■

If f is continuously differentiable, then we can give an alternative characterization of convex functions which is often easier to work with.

Proposition C.2.1 A continuously differentiable function $f : \mathbb{R}^n \mapsto \mathbb{R}$ is convex if and only if

$$f(x_2) \ge f(x_1) + \nabla f(x_1)^\top (x_2 - x_1) \qquad \text{for all } x_1, x_2 \in \text{dom } f \tag{C.4}$$

Condition (C.4) implies that for differentiable convex functions, the linearization at any point is a global lower bound on the function; see Figure C.3 (right). This also implies that stationary points $\tilde{x}$ where $\nabla f(\tilde{x}) = 0$ define global minima of f, since $f(x_2) \ge f(\tilde{x})$ for all $x_2 \in$ dom f. Thus, the first-order optimality condition (C.2) is both necessary and sufficient when f is convex.

Finally, we note in passing that the set

$$X = \{x \mid f(x) \le 0\}$$

is convex when f is a convex function. The proof of this claim is analogous to Example C.1 .

C.3 Constrained convex optimization problems

In practice, the decision vector often has to satisfy additional constraints. We will thererfore study constrained optimization problems on the form

$$\begin{array}{lll} \text{minimize} & f_0(x) & \\ \text{subject to} & f_i(x) \leq 0 & i=1,\dots,m \\ & g_i^\top x = h_i & i=1,\dots,p \end{array} \tag{C.5}$$

In this formulation, $f_0(x)$ is the *objective function* representing the operating cost of the system while $x \in \mathbb{R}^n$ is the *decision vector* containing the free variables. In addition $f_i(x) \leq 0$ describe *inequality constraints* and $g_i^\top x = h_i$ describe *linear equality constraints*. We say that x is *feasible* if it satisfies all constraints. The optimization problem (C.5) is *feasible* if it admits at least one feasible x. A vector $x^\star$ is said to be *optimal* if it attains the smallest value of f_0 among all feasible x. We let $p^\star = f_0(x^\star)$ denote the *optimal value* of (C.5).

It is sometimes convenient to write the optimization problem (C.5) on the compact form

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & x \in X \end{array}$$

where we have introduced the *feasible set*

$$X = \left\{ x \mid f_i(x) \leq 0,\ i=1,\dots,m \ \text{ and } \ g_i^\top x - h_i = 0,\ i=1,\dots,p \right\}. \tag{C.6}$$

as the set of decision vectors that satisfy all the constraints.

We will focus on *convex optimization problems* in which f_0 is a convex function and X is a convex set. As mentioned in the previous section, X defined in (C.6) is a convex set when all inequality constraint functions f_i are convex.

Convex optimization problems admit a powerful and elegant theory, and can be solved to global optimality using a range of numerical solvers; see, *e.g.*, [10]. The two most well-known classes of convex programming problems are linear and quadratic programs. In a *linear programming (LP) problem* one minimizes a linear function subject to linear inequality and equality constraints:

$$\begin{array}{lll} \text{minimize} & c^\top x & \\ \text{subject to} & a_i^\top x \leq b_i & i=1,\dots,m \\ & g_i^\top x = h_i & i=1,\dots,p \end{array} \tag{C.7}$$

A (convex) *quadratic programming (QP) problem*, on the other hand, considers the minimization of a convex quadratic function subject to linear constraints:

$$\begin{array}{lll} \text{minimize} & x^\top Px + 2q^\top x + r & \\ \text{subject to} & a_i^\top x \leq b_i & i=1,\dots,m \\ & g_i^\top x = h_i & i=1,\dots,p \end{array} \tag{C.8}$$

Both linear and quadratic programming problems have been studied extensively and admit a rich and useful theory. In addition, the numerical solvers for these problems have reached a high level of maturity and can routinely solve problems with millions of variables and constraints.

C.4 Modeling decision problems as quadratic programs

To make practical use of mathematical programming, it is essential to know how to translate a given decision problem into the optimization formalism. Although this “optimiztion modeling” is conceptually simple, different classes of optimization problems have slightly different properties, and differ in the type of objectives and constraints that they are able to describe. In this section, we will give a brief introduction to modeling for quadratic programming.

In an optimization model, the decision vector x contains the free system parameters that we can choose or influence; the objective function describes the cost of operating the system under the decisions in x; and the constraints describe the restrictions that x has to obey. For example, in an investment application, the decision vector may represent the amount invested in each asset, the objective may be to maximize the expected return, and the constraints may limit the total investment budget and the risk exposure. In a control application, the decision vector can be the sequence of controls that we apply, the objective can be to minimize the total energy consumed, and the constraints may encode the system dynamics and limitations on states and controls; see the exercises for more examples. The quadratic programming formalism restricts the objective function to be linear or quadratic, and limits the constraints to be expressed as linear (inequality and equality) constraints. However, as we will see, several important nonlinear functions and constraints can also be expressed in the quadratic programming formalism if we introduce additional decision variables.

Objective function modeling

If there are no constraints on x, a linear objective function $f_0(x) = c^\top x$ will result in an unbounded optimal value. The optimal decision will then be to let the elements of x have the opposite sign of those of c, and to let their magnitudes grow (arbitrarily) large. Hence, with a linear objective function, the constraints are essential in defining meaningful solutions.

As an example of this, consider the following problem

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{n} c_i x_i \\ \text{subject to} & \sum_{i=1}^{n} x_i = 1 \\ & x_i \geq 0 \qquad i = 1,\dots,n \end{array}$$

You can think of x_i as the volume of resource i, and $c_i \geq 0$ as its unit cost. The problem looks for the cheapest set of resources with a total volume of one. Of course, the optimal solution is to find the resource j with smallest unit cost (smallest value c_j), set the corresponding x_j to one, and all other decision variables to zero.

Even if this problem is trivial, it exposes some general features of linear programs. In particular, the optimal solution is always attained at the boundary of the constraint set. It is useful to think about the weights c_i as priorities: the larger the value of c_i, the more important it is to make the corresponding x_i small. In the simple resource allocation problem, we could set all variables but one to zero, but this is of course not necessarily true with other constraints.

If the objective function is quadratic $f_0(x) = x^\top P x + q^\top x + r$, and P is positive definite, then minimum of f_0 is attained at $x^\star = -P^{-1}q$, which is bounded if P and q are. Hence, the optimal solution does not necessarily lie on the boundary of the constraints. To make this more clear, let us consider a simple resource allocation problem with two resources and quadratic costs:

$$\begin{array}{ll} \text{minimize} & c_1 x_1^2 + c_2 x_2^2 \\ \text{subject to} & x_1 + x_2 = 1 \\ & x_1, x_2 \geq 0 \end{array}$$

After substituting $x_2 = 1 - x_1$ into the objective, a simple calculation yields that

$$x_1^\star = \frac{1}{1 + c_1/c_2}, \qquad x_2^\star = 1 - x_1^\star$$

We can still interpret the c_i's as (relative) priorities. The larger the ratio c_1/c_2, the more important it is to make x_1 small, and the smaller the allocation $x_1^\star$ in the optimal solution. Conversely, a small ratio c_1/c_2 gives an optimal allocation of x_1 close to one. Note that the effect of changing the priorities is much less pronounced than it was for a linear objective function.

Nonlinear functions in the QP formalism

Although linear and quadratic programs only allow for linear constraints, it is also straightforward to include convex piecewise linear constraints. For example, the constraint

$$|x| \leq 1$$

with $x \in \mathbb{R}$ can be written as the two linear constraints $x \leq 1$ and $x \geq -1$, *i.e.*

$$\begin{pmatrix} x \\ -x \end{pmatrix} \leq \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

Similarly, the constraint

$$\max_{i=1,\dots,m} a_i^\top x + b_i \leq 1$$

with $x \in \mathbb{R}^n$ can be written as

$$\begin{pmatrix} a_1^\top \\ \vdots \\ a_m^\top \end{pmatrix} x \leq \begin{pmatrix} 1 - b_1 \\ \vdots \\ 1 - b_m \end{pmatrix}$$

If we want to minimize $|x|$, we can minimize an auxiliary variable $t \geq |x|$, *i.e.*

$$\begin{array}{ll} \text{minimize} & t \\ \text{subject to} & \begin{pmatrix} 1 & -1 \\ -1 & -1 \end{pmatrix} \begin{pmatrix} x \\ t \end{pmatrix} \leq \mathbf{0} \end{array}$$

In a similar way, for $x \in \mathbb{R}^n$, we can minimize $\|x\|_\infty = \max_i |x_i|$, by minimizing $t \geq |x_i|$, *i.e.*

$$\begin{array}{ll} \text{minimize} & t \\ \text{subject to} & \begin{pmatrix} I & -\mathbf{1} \\ -I & \mathbf{1} \end{pmatrix} \begin{pmatrix} x \\ t \end{pmatrix} \leq \mathbf{0} \end{array}$$

Finally, we can minimize $\|x\|_1 = \sum_{i=1}^n |x_i|$ by

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^n t_i \\ \text{subject to} & |x_i| \leq t_i \end{array}$$

We leave it as an exercise to put this optimization problem on the standard form for linear or quadratic programs. Sums of the functions described above can be modeled in a similar manner and can be expressed as linear or quadratic programs. Many algebraic modeling languages for optimization, such as YALMIP and cvx, implement a rich catalog of functions that can be represented by linear inequalities and perform the reformulations to standard form automatically.

Hard and soft constraints

A mathematical programming problem only optimizes over the decision vectors that satisfy the stated constraints. The constraints are therefore hard, in the sense that they have to be satisfied. However, we can sometimes accept to violate some of the constraints, if this is necessary to find a solution. Such soft constraints are typically dealt with using slack variables. To make the decision more precise, consider the following optimization problem

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & A_\text{h} x \leq b_\text{h} \\ & A_\text{s} x \leq b_\text{s} \end{array}$$

where the second set of constraints are soft. To allow for constraint violations, we consider

$$\begin{array}{ll} \text{minimize} & f_0(x) + \kappa s^\top s \\ \text{subject to} & A_\text{h}x \leq b_\text{h} \\ & A_\text{s}x \leq b_\text{s} + s \\ & s \geq 0 \end{array}$$

In this formulation, the soft constraints can always be satisfied, possibly by a large violation s. To discourage violations, we add a slack penalty $\kappa s^\top s$ to the objective function. A large value of the parameter κ puts a high priority on keeping s small. Softening of constraints is essential to MPC.

C.5 Optimality conditions for constrained convex optimization

For unconstrained convex optimization problems, the first-order optimality condition allows for a simple analytical characterization of the optimizers. The corresponding condition for constrained convex optimization problems reads that $x^\star$ must satisfy

$$\nabla f(x^\star)^\top (x - x^\star) \geq 0 \qquad \forall x \in X.$$

This condition says that $x^\star$ is optimal if every perturbation of $x^\star$ in a feasible direction leads to an increase in the function value. Unfortunately, this condition is much more difficult to check than the unconstrained counterpart, and it has therefore limited practical use.

A more convenient technique for studying constrained convex optimization is the *method of Lagrange multipliers*. Given a problem on the form (C.5), one constructs an associated *Lagrangian function* $L : \mathbb{R}^n \times \mathbb{R}^m_+ \times \mathbb{R}^p \mapsto \mathbb{R}$

$$L(x, \lambda, \mu) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \mu_i (g_i^\top x - h_i).$$

Here $\lambda_i \geq 0$ is a *Lagrange multiplier* for the inequality constraint $f_i(x) \leq 0$, while μ_i is a Lagrange multiplier for the equality constraint $g_i^\top x - h_i = 0$. Note that if x is feasible (*i.e.*, satisfies the constraints) and $\lambda_i \geq 0$ for $i = 1, \dots, m$, then

$$L(x, \lambda, \mu) \leq f_0(x)$$

and that equality holds if $\lambda_i f_i(x) = 0$ for all i. To explore this lower bound further, one introduces the *dual function* $g : \mathbb{R}^m_+ \times \mathbb{R}^p \mapsto \mathbb{R}$ as

$$g(\lambda, \mu) = \inf_{x \in \mathcal{D}} L(x, \lambda, \mu) = \inf_{x \in \mathcal{D}} \left\{ f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \mu_i (g_i^\top x - h_i) \right\}$$

Here, $\mathcal{D}$ is the intersection of the domain of f_0 and the feasible set X. We assume that $\mathcal{D}$ is non-empty. A key property of the dual function is that it lower bounds the optimal value of (C.5). Specifically, if $\lambda_i \geq 0$ for $i = 1, \dots, m$, then

$$g(\lambda, \mu) \leq p^\star$$

Since g is a lower bound to $p^\star$, it is natural to try to make this bound as tight as possible, *i.e.* to

$$\begin{array}{ll} \text{maximize} & g(\lambda, \mu) \\ \text{subject to} & \lambda \succeq 0 \end{array}$$

This optimization problem is called the *dual problem* to (C.5), and we will denote its optimal value by $d^\star$. The next result follows immediately from our discussion.

Proposition C.5.1 Weak duality, *i.e.* $d^\star \leq p^\star$ always holds.

Under various conditions, convex optimization problems allow us to guarantee *strong duality*, *i.e.* that $d^\star = p^\star$. The next result is known as the *Slater conditions* for strong duality:

Theorem C.5.2 If $f_i(x)$, $i = 0, \dots, m$ are convex and there exists an $x \in \operatorname{int} \operatorname{dom} f_0$ such that

$$f_i(x) < 0, \quad i = 1, \dots, m \text{ and } g_i^\top x - h_i = 0, \quad i = 1, \dots, p$$

then $d^\star = p^\star$.

Under strong duality, we can derive several additional useful results. To this end, let $x^\star$ be primal optimal and $(\lambda^\star, \mu^\star)$ be a dual optimal point. Then, if strong duality holds, we have

$$\begin{aligned} f_0(x^\star) = g(\lambda^\star, \mu^\star) = \inf_{x \in \mathcal{D}} \left\{ f_0(x) + \sum_{i=1}^m \lambda_i^\star f_i(x) + \sum_{i=1}^p \mu_i^\star (g_i^\top x - h_i) \right\} \leq \\ \leq f_0(x^\star) + \sum_{i=1}^m \lambda_i^\star f_i(x^\star) + \sum_{i=1}^p \mu_i^\star (g_i^\top x^\star - h_i) \leq f_0(x^\star). \end{aligned}$$

Since all inequalities must hold with equality, we conclude that $x^\star$ minimizes $L(x, \lambda^\star, \mu^\star)$ and that the so-called *complementary slackness* $\lambda_i^\star f_i(x^\star) = 0$ must hold for all $i = 1, \dots, m$. This leads to the *Karush-Kuhn-Tucker (KKT) conditions* for optimality:

$$\begin{aligned} \nabla f_0(x^\star) + \sum_{i=1}^m \lambda_i^\star \nabla f_i(x^\star) + \sum_{i=1}^p \mu_i^\star g_i &= 0 \\ \lambda_i^\star f_i(x^\star) &= 0 \qquad i = 1, \dots, m \\ f_i(x^\star) &\leq 0 \qquad i = 1, \dots, m \\ g_i^T x^\star - h_i &= 0 \qquad i = 1, \dots, p \\ \lambda_i^\star &\geq 0 \qquad i = 1, \dots, m. \end{aligned}$$

Here, the first equality is the first-order optimality condition for $x^\star$ minimizing $L(x, \lambda^\star, \mu^\star)$, the second set of equations describe complementary slackness, the third and fourth set of equations are primal feasibility and the final set of conditions describe dual feasibility.

Theorem C.5.3 Consider the constrained optimization problem (C.5) under the assumption that $f_0, \dots, f_m$ are continuously differentiable and convex. Then, if Slater's condition holds, the KKT conditions are necessary and sufficient for optimality.

The next example illustrates a typical application of the method of Lagrange multipliers.

■ **Example C.3** To solve the constrained optimization problem

$$\begin{aligned} &\text{minimize} && x^\top x \\ &\text{subject to} && Cx = d \end{aligned}$$

using the method of Lagrange multipliers, we first form the Lagrangian function

$$L(x, \mu) = x^\top x + \mu^\top (Cx - d)$$

The associated dual function is

$$g(\mu) = \inf_x L(x, \mu).$$

By the first-order optimality conditions, the minimizing x is $x = -C^\top \mu/2$ so

$$g(\mu) = L(-\frac{1}{2}C^\top \mu, \mu) = -\frac{1}{4}\mu^\top CC^\top \mu - \mu^\top d.$$

The optimal dual value

$$d^\star = \sup_\mu g(\mu)$$

can also be computed by application of the first-order optimality conditions. Doing so, we find that the optimal Lagrange multiplier must satisfy

$$-\frac{1}{2}CC^\top \mu^\star - d = 0$$

Under the assumption that $CC^\top$ is invertible, $\mu^\star = -2(CC^\top)^{-1}d$ and

$$d^\star = d^\top (CC^\top)^{-1} d.$$

Since $x^\top x$ is convex, if there is an x such that $Cx = d$, then by Theorem C.5.2

$$p^\star = d^\star = d^\top (CC^\top)^{-1} d$$

and an optimal solution is given by the minimizer of $L(x, \mu^\star)$, *i.e.*

$$x^\star = C^\top (CC^\top)^{-1} d.$$

■

C.6 A brief overview of quadratic programming solvers

For many applications, quadratic programming is a mature technology. One can trust that standard numerical routines are quick to provide an accurate solution or detect that no solution exists. However, when limits are stretched, e.g. in terms of problem size, acceptable execution times, or constraints on the underlying execution platform, this may no longer be true. We may then be restricted in what type of quadratic programming solver that we can use, and be more exposed to the limitations of different solvers. This is currently the case for many MPC applications.

This section gives a brief overview of the QP solver technologies that are currently used in model predictive control. It describes the basic underlying ideas and tries to convey intuition that is useful for selecting the solver for a particular application. For a deeper theoretical treatment of the details required to implement your own QP solver, we refer to the literature.

The gradient descent method

Let us first begin by studying an unconstrained quadratic program

$$\underset{z\in\mathbb{R}^n}{\text{minimize}} \quad f(z) = \tfrac{1}{2}z^\top Qz + p^\top z \tag{C.9}$$

where Q is a positive definite matrix with $\lambda_{\min}(Q) = \mu$ and $\lambda_{\max}(Q) = L$. Of course, we know from the first-order optimality conditions that the optimizer $z^\star$ of (C.9) satisfies

$$Qz^\star + p = 0$$

Hence, $z^\star$ can (and should) be found by solving a system of linear equations. Nevertheless, we can also find the optimizer by minimizing $f(x)$ numerically.

One of the simplest methods for minimizing a convex function is the *gradient descent* algorithm. Given $z_0 \in \mathbb{R}^n$, it generates a sequence of iterates $\{z_k\}$ via

$$z_{k+1} = z_k - \gamma \nabla f(z_k).$$

where γ is a step-size parameter. Clearly, any fixed-point z of these iterations satisfies $\nabla f(z) = 0$. As we will see, the algorithm converges to such a fixed-point if the step-size parameter is set appropriately. Specifically, for our problem (C.9), $\nabla f(z) = Qz + p = Q(z - z^\star)$ and

$$\begin{aligned} \|z_{k+1} - z^\star\|_2 &= \|z_k - \gamma \nabla f(z_k) - z^\star\|_2 = \\ &= \|z_k - z^\star - \gamma Q(z_k - z^\star)\|_2 = \\ &= \|(I - \gamma Q)(z_k - z^\star)\|_2 = \\ &\leq \|(I - \gamma Q)\|_2 \|(z_k - z^\star)\|_2 = \\ &= \left(\max_i |1 - \gamma \lambda_i(Q)|\right) \|(z_k - z^\star)\|_2 \end{aligned}$$

Hence, with the step-size $\gamma = 1/L$, we have

$$\|z_{k+1} - z^\star\|_2 \leq \left(1 - \frac{1}{\kappa}\right) \|z_k - z^\star\|_2$$

where we have introduced the condition number $\kappa = L/\mu \geq 1$. The analysis shows that the iterates move closer to the optimizer in every iteration and that the convergence is slower when the condition number κ of Q is large. With some additional effort, it is possible to do a more careful analysis and demonstrate an even faster convergence. Such a refined analysis also shows that the gradient descent iterations converge slower when the condition number is large.

Gradient descent can also be adapted to QP problems with simple bounds,

$$\begin{array}{ll} \text{minimize} & \frac{1}{2} z^\top Q z + p^\top z \\ \text{subject to} & \|z\|_\infty \leq z_{\max} \end{array}$$

by simply projecting the iterates onto the feasible set

$$z_{k+1} = \text{Proj}_Z\left(z_k - \gamma \nabla f(z_k)\right)$$

Here, $Z = \{z \mid \|z\|_\infty \leq z_{\max}\}$ and $\text{Proj}_Z(x) = \arg\min_{z \in Z} \|z - x\|_2^2$ is the Euclidean projection of x onto Z. One can show that this projection simply amounts to projecting each element of the iterate vector to the interval $[-z_{\max}, z_{\max}]$. Since $z^\star \in Z$ and the projection operator is non-expansive (see Problem C.15), it holds that

$$\|z_{k+1} - z^\star\|_2 = \|\text{Proj}(z_k - \gamma \nabla f(x_k)) - \text{Proj}(z^\star)\|_2 \leq \|z_k - \gamma \nabla f(z_k) - z^\star\|_2,$$

so the same convergence proof (and guarantee) holds for the projected GD.

The projected gradient method can be used with more general constraints, as long as the projection operator can be evaluated efficiently. However, projecting onto a general polyhedron $\{z \mid Az \leq b\}$, as we are interested in doing when solving MPC problems, requires solving a quadratic program. Hence, evaluating the projection operator is as hard as solving the original problem. However, the corresponding dual optimization problem has simple constraints and can therefore be solved using the projected gradient descent method; see Problem C.16.

The (projected) gradient method is surprisingly simple to implement, but it has a few disadvantages. First, it is relatively sensitive to the conditioning of the problem (the condition number κ) and the convergence may be slow if we need high accuracy solutions. There are a number of ways to improve the practical convergence speed of the gradient descent iterations, and there are several related algorithms that can accelerate the convergence in both theory and practice, see [18]. The second problem is that convergence of these methods is only asymptotic, so in the context of MPC, one may have to accept to operate using (slightly) suboptimal solutions; cf. Exercise 5.14.

Interior-point methods

Interior-point methods are state-of-the-art for solving medium-sized QPs on standard computers. They can deal with general quadratic programs and benefit for treating equality and inequality constraints differently. However, for simplicity, we only explain them for a QP on the form

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}z^\top Pz + q^\top z \\ \text{subject to} & Az \leq b \end{array}$$

Recall that the optimal solution to such a problem satisfies the KKT conditions

$$\begin{aligned} Pz + q + A^\top \lambda &= 0 \\ Az - b &\leq 0 \\ \lambda \odot (Az - b) &= 0 \\ \lambda &\geq 0 \end{aligned}$$

where $\odot$ in the complementarity condition denotes elementwise multiplication of the two vectors. One popular class of interior point methods approach the solution to these conditions along a *central path.* To define the central path, we re-write the optimality conditions using a slack-variable s and relax the complementarity condition with a parameter μ

$$\begin{aligned} Pz + q + A^\top \lambda &= 0 \\ Az - b + s &= 0 \\ \lambda \odot s &= \mu \mathbf{1} \\ \lambda, s &\geq 0 \end{aligned}$$

The central path is the unique point $(z(\mu), s(\mu), \lambda(\mu))$ that simultaneously satisfies these relaxed optimality conditions. The conditions are solved for a sequence of parameters μ that converges to zero. Non-negativity of λ and s is accounted for separately, and the remaining conditions define a nonlinear system of $n+2m$ equations in $n+2m$ variables. When the QP is convex, one can show that the equations have a unique solution in the strict interior of the orthant $\lambda, s \geq 0$.

As μ tends to zero, the central path converges to an optimal solution to both the primal and dual problems. A primal-dual path-following algorithm is defined as any iterative process that starts from a point in the strict interior of (2.5) and at each iteration estimates a value of μ representing a point on the central path that is in some sense closer to the optimal solution than the current point and then attempts to step toward this central-path point making sure that the new point remains in the strict interior of the appropriate orthant.

The interior point methods have excellent theoretical convergence guarantees and tend to find high-accuracy solutions in a relatively modest (typically 10-20) number of iterations. However, they require more memory than the simple gradient iterations and rely on advanced numerical linear algebra in each iteration to solve the relaxed KKT conditions. Representative interior-point methods that target embedded MPC controllers incude cvxgen [27] and HPMPC [17].

Active set methods

Active set methods attempt to solve quadratic programs on the form

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}z^\top Pz + q^\top z \\ \text{subject to} & Az \leq b \end{array} \tag{C.10}$$

by determining what constraints hold with equality at optimal solution. To describe these methods, we introduce the following terminology. We say that an inequality constraint $a_i^\top z \leq b_i$ is *active* at $\bar{z}$ if $a_i^\top \bar{z} = b_i$. For a system of q inequality constraints, $Az \leq b$, we define the *active set* at $\bar{z}$, $\mathcal{A}(\bar{z}) \subseteq \{1,\dots,q\}$, as the set of indices of the constraints that are active at $\bar{z}$. For ease of notation, we let $\mathcal{A}^\star = \mathcal{A}(z^\star)$. If $M \in \mathbb{R}^{q\times r}$ and $\mathcal{A} \subseteq \{1,\dots,q\}$, we let $M_{\mathcal{A}} \in \mathbb{R}^{|\mathcal{A}|\times r}$ represent the matrix formed by the rows of M whose indicies belong to $\mathcal{A}$.

Now, if we would know the optimal optimal active set $\mathcal{A}^\star$ of (C.10), then the optimal solution to (C.10) would also be a solution to the equality-constrained QP

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}z^\top Pz + q^\top z \\ \text{subject to} & A_{\mathcal{A}^\star}z = b_{\mathcal{A}^\star} \end{array}$$

The optimality conditions for this problem are the KKT conditions

$$\begin{pmatrix} P & A_{\mathcal{A}^\star}^\top \\ A_{\mathcal{A}^\star} & 0 \end{pmatrix} \begin{pmatrix} z \\ \lambda \end{pmatrix} = \begin{pmatrix} -q \\ b_{\mathcal{A}^\star} \end{pmatrix} \tag{C.11}$$

so its solution can be found by solving a simple system of linear equations. Active set methods leverage on this observation and find the optimal solution by maintaining a working set $\mathcal{W}$ of constraints, and updating $\mathcal{W}$ so that it eventually converges to $\mathcal{A}^\star$. More specifically, it solves the KKT system (C.11) defined by $\mathcal{A}^\star = \mathcal{W}$ and checks if the corresponding solution satisfies $\lambda \geq 0$ and $Az \leq b$. If this is the case, an optimal solution has been found. Otherwise, the working set has to be updated and a new KKT system has to be solved.

Most active set methods replace one constraint in the working set in every iteration. There are several approaches for selecting what constraint to add and remove, leading to primal, dual, and primal-dual active set methods, respectively. Primal methods ensure that the primal iterate z is always feasible, dual methods guarantee that the dual iterate λ is always non-negative, while primal-dual methods do not enforce feasibility of iterates until convergence.

Active set methods are attractive for MPC applications, since the effort of solving the KKT system is limited, and the optimal active set can often be found in a few iterations. This is especially true if the MPC solver is *warm-started*, *i.e.* initialized from a good starting guess based on the solution to the problem solved in the previous sampling time. Once the optimal set is found, a high-accuracy solution can be computed also with limited precision arithmetics.

A drawback of active set methods is that one has to be careful to avoid cycling, *i.e.* that the new working set is equal to some previous working set. In addition, their theoretical worst-case complexity is poor, since one can construct contrived problems that require an exponential number of updates of the working set to find the optimal solution. Nevertheless, for the class of problems that is encountered in MPC applications, it is possible to perform an exact complexity analysis offline to determine the maximum number of iterations that can possibly be required. Representative active set solvers for embedded systems include qpOASES [15] and DAQP [3].

Explicit solutions to quadratic programs

For the family of quadratic programs that appear in model predictive control applications, it is also possible to avoid on-line optimization altogether, and compute an explicit expression of the optimal solution as function of the initial state. This approach is known as *explicit MPC* and is based on the concept of multi-parametric quadratic programming.

Recall that we in Chapter 3 demonstrated how MPC problems on standard form could be written in a condensed form with only inequality constraints

$$\begin{array}{ll} \text{minimize} & z^\top Pz + 2q^\top z + r \\ \text{subject to} & Az \leq b \end{array}$$

Here, z contains the predicted optimal control sequence $\{\hat{u}_0, \hat{u}_1, \dots, \hat{u}_{T-1}\}$, and q, r and b depend on the free response (and therefore on the initial state). To simplify the exposition, we perform a variable transformation $x = z + P^{-1}q$, which results in the problem

$$\begin{array}{ll} \text{minimize} & x^\top P x + r - q^\top P^{-1} q \\ \text{subject to} & Ax \leq b - P^{-1} q \end{array}$$

Since the constant term in the objective does not influence the optimal solution (only the optimal value of the optimization problem), we will initially disregard it. Moreover, since b is affine in x_0 and q is linear in x_0, we can re-write the constraint as $Ax \leq w + S\theta$, where we have replaced x_0 by θ to signify that it is a parameter in the optimization problem. This leads to the following *multiparametric Quadratic Programming (mpQP)* problem

$$\begin{array}{ll} \text{minimize} & x^\top P x \\ \text{subject to} & Ax \leq w + S\theta \end{array} \tag{C.12}$$

where $x \in \mathbb{R}^d$, $w \in \mathbb{R}^q$, $\theta \in \Theta \subseteq \mathbb{R}^n$, and the remaining matrices are of compatible dimensions. We are interested in understanding how its optimal solution $x^\star(\theta)$ depends on the parmeter θ. To this end, we note that the optimal solution to (C.12) is characterized by the KKT conditions

$$\begin{aligned} 2Px + A^\top \lambda &= 0 \\ Ax &\leq w + S\theta \\ \lambda &\geq 0 \\ \lambda \odot (Ax - w - S\theta) &= 0 \end{aligned}$$

As we will see below, given an optimal active set, we can use the KKT conditions to express the optimal solution of (C.12) as an explicit function of θ. This expression will be valid for all θ that result in the same optimal active set. We call such a region in the parameter space a *critical region*, defined next.

Definition C.6.1 Given an index set $\mathcal{A} \subseteq \{1, 2, \dots, q\}$, the critical region $\mathcal{R}_\mathcal{A}$ associated with $\mathcal{A}$ is the set of parameters for which the optimal active set is equal to $\mathcal{A}$, *i.e.* $\mathcal{R}_\mathcal{A} = \{\theta \in \mathbb{R}^n \mid \mathcal{A}^\star(\theta) = \mathcal{A}\}$.

As discussed in the section on active set QP solvers, all Lagrange multipliers that are not in the active set will be zero and can be eliminated from the KKT conditions. Using the same notation as in the previous section, we can therefore write the KKT conditions for a given active set $\mathcal{A}$ as

$$\begin{aligned} 2Px + A_\mathcal{A}^\top \lambda_\mathcal{A} &= 0 \\ Ax &\leq w + S\theta \\ \lambda_\mathcal{A} &\geq 0 \\ A_\mathcal{A} x - w_\mathcal{A} - S_\mathcal{A}\theta &= 0 \end{aligned}$$

For simplicity of exposition, let us assume that $P \succ 0$ and that the rows of $A_\mathcal{A}$ are linearly independent. Then, we can combine the first and fourth KKT conditions to find explicit expressions for the optimal dual and primal variables:

$$\lambda_\mathcal{A}(\theta) = -2(A_\mathcal{A} P^{-1} A_\mathcal{A}^\top)^{-1}(w_\mathcal{A} + S_\mathcal{A}\theta) \tag{C.13}$$

$$x(\theta) = P^{-1} A_\mathcal{A}^\top (A_\mathcal{A} P^{-1} A_\mathcal{A}^\top)^{-1}(w_\mathcal{A} + S_\mathcal{A}\theta) \tag{C.14}$$

Note that the optimal solutions are affine in θ, and that the inverses exist since, by assumption, $P \succ 0$ and the rows of $A_\mathcal{A}$ are linearly independent. The expressions are valid for all θ that do not violate primal and dual feasibility, *i.e.*, for which

$$\left\{ \begin{array}{rcl} AP^{-1}A_\mathcal{A}^\top (A_\mathcal{A} P^{-1} A_\mathcal{A}^\top)^{-1}(w_\mathcal{A} + S_\mathcal{A}\theta) & \leq & w + S\theta \\ 2(A_\mathcal{A} P^{-1} A_\mathcal{A}^\top)^{-1}(w_\mathcal{A} + S_\mathcal{A}\theta) & \leq & 0 \end{array} \right. \tag{C.15}$$

Since these inequalities are affine in θ, the critical region $\mathcal{R}_{\mathcal{A}}$ is a polyhedron in $\mathbb{R}^n$. For all θ that lie in the critical region defined by (C.15), the optimal primal solution $x(\theta)$ is an affine function of θ, whose explicit expression is given in (C.14).

To define the explicit solution for *all* feasible θ, one needs to explore all critical regions. Although there are many possible ways to do this, we will only explain a simple but computationally demanding procedure. It begins with a convex and compact subset $\Theta \subset \mathbb{R}^n$ of the parameter space. In every step of the algorithm, it picks a $\theta \in \Theta$ and solves (C.12) using a standard QP solver to find the optimal active set $\mathcal{A}^\star(\theta)$ and the corresponding critical region $\mathcal{R}_{\mathcal{A}^\star(\theta)}$ using (C.15). It then partitions $\Theta \backslash \mathcal{R}_{\mathcal{A}^\star(\theta)}$ into convex polyhedra and repeats the procedure inside each such polyhedon; see Figure C.4. More specifically, if $\mathcal{R}_{\mathcal{A}^\star(\theta)}$ is defined by m inequalities, it partitions the complement of this set into m polyhedra, where the i^{th} polyhedron is defined from the inequalities of the critical region by dropping the first $i-1$ constraints, reversing the direction of the i^{th} inequality, and keeping the remaining constraints as is. The development of a numerically reliable algorithm that can also deal with the case that the rows of $A_{\mathcal{A}^\star(\theta)}$ become linearly dependent requires some additional effort. We will not provide the details here, but refer to the literature and describe the approach on a simple integrator example.

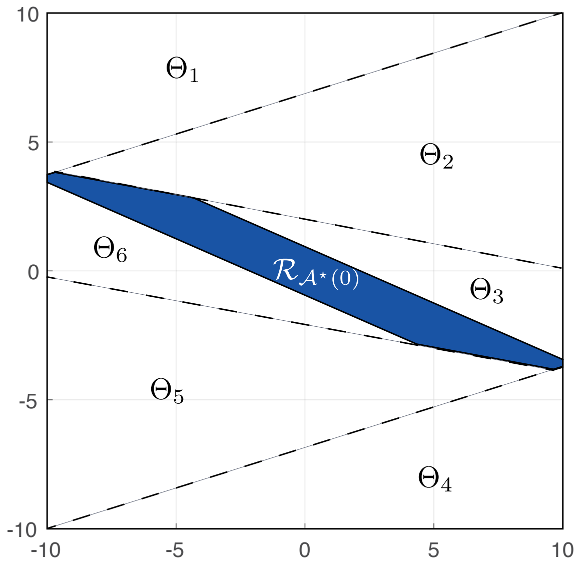


Figure C.4: A simple approach to subdivide a compact region Θ of the parameter space into critical regions. From an interior point of Θ, the associated QP is solved and an active set is identified. The complement of the corresponding critical region is subdivided into convex polyhedra (here, $\Theta_1, \ldots, \Theta_6$, and the procedure is repeated.

■ **Example C.4** Let us consider the planning problem for an integrator process with magnitude limits on the control

$$\begin{array}{ll} \text{minimize} & \sum_{t=0}^{T-1} \frac{1}{2}\left(\hat{x}_t^2 + \hat{u}_t^2\right) + \frac{1}{2}\hat{x}_T^2 \\ \text{subject to} & \hat{x}_{t+1} = \hat{x}_t + \hat{u}_t \\ & \hat{u}_t \in [-1,1] \end{array} \tag{C.16}$$

For simplicity, we consider $T = 1$ and use the dynamics to eliminate the terminal state, leading to the planning problem

$$\begin{array}{ll} \text{minimize} & \hat{x}_0^2 + \hat{x}_0\hat{u}_0 + \hat{u}_0^2 \\ \text{subject to} & \hat{u}_0 \in [-1,1] \end{array}$$

To find the explicit solution, we apply the variable transformation $\hat{u} = \hat{u}_0 + \hat{x}_0/2$ and drop terms in the objective that do not depend on $\hat{u}$, leading to

$$\begin{array}{ll} \text{minimize} & \hat{u}^2 \\ \text{subject to} & \hat{u} \le 1 + \hat{x}_0/2 \\ & -\hat{u} \le 1 - \hat{x}_0/2 \end{array} \tag{C.17}$$

Beginning with $\hat{x}_0 = 0$, the optimal solution to the QP is $\hat{u} = 0$ and $\mathcal{A}(0) = \emptyset$. The primal feasibility conditions reveal that this solution is valid for $\hat{x}_0 \in \mathcal{R}_{\mathcal{A}^\star(0)} = \{x_0 \mid x_0 \le 2 \wedge -x_0 \le 2\}$. To find the complete solution, we subdivide the remaining parameter space into the two polyhedra $\Theta_1 = \{x_0 \mid x_0 \ge 2 \wedge -x_0 \le 2\}$ and $\Theta_2 = \{x_0 \mid -x_0 \ge 2\}$. For $\hat{x}_0 \in \Theta_1$, we pick $\hat{x}_0 = 6$ and solve the resulting QP to find $\hat{u}^\star = 2$ and $\mathcal{A}^\star(6) = \{2\}$. The KKT conditions reveal that $\hat{u}^\star = \hat{x}_0/2 - 1$, and that this expression is valid for all $\hat{x}_0 \ge 2$. For $\hat{x}_0 \in \Theta_2$, we choose $\hat{x}_0 = -6$ and solve the resulting QP to find $\mathcal{A}^\star(-6) = \{1\}$. The KKT conditions imply that $\hat{u}^\star = 1 + \hat{x}_0/2$, valid for $\hat{x}_0 \le 2$. Hence, we have found the solution to (C.17):

$$\hat{u}^\star = \begin{cases} \hat{x}_0/2 + 1 & \text{if } \hat{x}_0 \le -2 \\ 0 & \text{if } -2 \le \hat{x}_0 \le 2 \\ \hat{x}_0/2 - 1 & \text{if } \hat{x}_0 \ge 2 \end{cases}$$

Finally, we transform the solution back to the original coordinates to find

$$\hat{u}_0^\star = \begin{cases} 1 & \text{if } \hat{x}_0 \le -2 \\ -\hat{x}_0/2 & \text{if } -2 \le \hat{x}_0 \le 2 \\ -1 & \text{if } \hat{x}_0 \ge 2 \end{cases}$$

Note that the minimizer is a continuous and piecewise affine function of the parameter $\hat{x}_0$. ■

Multi-parametric quadratic programming algorithms, such as [5, 35] are run off-line, and partition feasible parameter set into non-overlapping polyhedra and compute the associated affine expressions that determine the optimizer as a function of the parameters in each such polyhedron. The algorithm itself can take significant time to run, and the partition can sometimes have a (very) large number of polyhedra, especially when the optimization problem has a large decision vector and many constraints. However, the online computations are reduced to finding the polyhedron that contains the current parameters and then evaluating the affine expression valid in that region.

Another advantage of the multi-parametric framework is that it allows us to understand how the optimal solution and the optimal value depend on the parameters. The next proposition, which is Theorem 4 in [5] adapted to our notation, summarizes some properties of multi-parametric quadratic programs that are useful in the analysis of model predictive controllers.

Proposition C.6.1 Consider the mpQP (C.12) with $P \succ 0$ and Θ convex. Then the set of feasible parameters $\Theta_f \subseteq \Theta$ is convex. The minimizer function $x^\star(\theta) : \Theta_f \mapsto \mathbb{R}^d$ is continuous and piecewise affine in the sense that there exists a partitioning of Θ_f into a finite set of full-dimensional polyhedra $\mathcal{R} := \{R_1, \dots, R_K\}$ with $\cup_{k=1}^K R_k = \Theta_f$ and $\text{int}(R_i) \cap \text{int}(R_j) = \emptyset$ for all $i \neq j$, such that $x^\star(\theta)$ is an affine function of θ in each R_k. The value function

$$V(\theta) = \min_x \left\{ x^\top P x \mid Ax \le w + S\theta \right\}$$

is convex, continuous, and piecewise quadratic.

C.7 Exercises

Problem C.1 Show that the probability simplex

$$X = \left\{ x \in \mathbb{R}^n \mid x_i \geq 0, \sum_i x_i = 1 \right\}$$

is a convex set.

Problem C.2 Recall that a norm $\|\cdot\|$ satisfies the following three properties

1. $\|v\| \geq 0$ for all v and $\|v\| = 0$ if and only if $v = 0$.
2. $\|\lambda v\| = |\lambda| \|v\|$ for all λ, v.
3. $\|v + w\| \leq \|v\| + \|w\|$ for all v, w.

Show that the norm is a convex function. Use this result to show that the unit norm ball

$$\mathbb{B} = \{x \in \mathbb{R}^n \mid \|x\| \leq 1\}$$

and the second-order cone

$$\mathbb{K} = \left\{(x,t) \in \mathbb{R}^{n+1} \mid \|x\|_2 \leq t\right\}$$

are convex sets.

Problem C.3 Show that the pointwise maximum of two convex functions is convex. In other words, if $f_1(x)$ and $f_2(x)$ are both convex functions, then so is $f(x) = \max\{f_1(x), f_2(x)\}$.

Problem C.4 Let $f : \mathbb{R} \mapsto \mathbb{R}$ be a sclar and twice continuously differentiable scalar function. Prove that f is convex if and only if

$$\nabla^2 f(x) \geq 0 \quad \forall x$$

(In $\mathbb{R}^n$, the corresponding result states that f is convex if and only if its Hessian is positive semidefinite for all x; you can also try to prove this more general result.)

Problem C.5 Draw the feasible set for the following LP. Indicate lines along which the objective function is constant, and identify the optimal solution.

$$\begin{array}{ll} \text{maximize} & x_1 + x_2 \\ \text{subject to} & x_1 + 2x_2 \leq 10 \\ & 2x_1 + x_2 \leq 16 \\ & -x_1 + x_2 \leq 3 \\ & x_1 \geq 0,\ x_2 \geq 0 \end{array}$$

Validate your results by solving the LP numerically.

Problem C.6 A city has the following minimum requirement for the number of police on duty

00.00 – 04.00	04.00 – 08 – 00	08.00 – 12.00	12/000 – 16.00	16.00 – 20.00	20.00 – 24.00
15	35	65	80	40	25

Each police comes on duty at 0.00, 4.00, 8.00, 12.00, 16.00 or 20.00 and works for eight consecutive hours. Assume that the same schedule is repeated day after day. Formulate the problem of finding the duty schedule that minimizes the total number of police as a linear program. How many policemen are needed?

Hint. Disregard the fact that the number of police in each shift should be an integer number.

Problem C.7 A health food shop packages three types of snack foods: chewy, crunchy, and nutty. These are made by mixing sunflower seeds, raisins, and peanuts. The specifications for each food are given in the following table:

Mixture	Sunflower seeds	Raisins	Peantus	Retail price/kg
Chewy	–	at least 60%	at most 25%	20 SEK
Crunchy	at least 60%	–	at most 25%	16 SEK
Nutty	at most 20%	–	at least 60%	12 SEK

The supplier of ingredients can deliver at most 100kg of sunflower seeds at 10 SEK/kg, 80kg or raisins at 15 SEK/kg and 60kg of peanuts at 8 SEK/kg. Determine the mixing scheme that maximizes profit under the assumption that all produced goods will be sold.

Hint. It can be convenient to introduce decision variables x_{ij} that symbolize the amount (in kilos) of ingredient j used to make snack i.

Problem C.8 An airport is accepting N arriving aircraft in a fixed order $1, 2, \dots, N$. Each aircraft i is given a time-interval $[l_i, u_i]$ when it is allowed to land. The airport wants to assign arrival times t_i that maximize the smallest inter-arrival times, *i.e.* find t_i in order to

$$\text{maximize} \quad \min_{i=1,\dots,N-1} t_{i+1} - t_i$$

while maintaining the ordering of the aircraft and respecting the given time windows. Formulate a linear program that computes the optimal arrival times t_i.

Problem C.9 There are many ways to define the center of a polyhedron

$$\mathcal{P} = \left\{ x \in \mathbb{R}^n \mid a_i^T x \leq b_i,\ i = 1, \dots, m \right\}$$

The Chebyshev center of $\mathcal{P}$ is defined as the center x_c of the largest Euclidean ball

$$\mathcal{B}(x_c, r) = \{ x \in \mathbb{R}^n \mid \|x - x_c\|_2 \leq r \}$$

that fits inside the polyhedron. Derive a linear program for computing x_c.

Hint. Note that the Euclidean ball can also be represented as

$$\mathcal{B} = \{ x = x_c + u \mid \|u\|_2 \leq r \}$$

Use this representation to derive a condition for $\mathcal{B}$ to be fully contained in a single halfspace $\mathcal{H}_u = \left\{ x \in \mathbb{R}^n \mid a_i^T x \leq b_i \right\}$ and apply this condition to all hyperplanes that define the polyhedron.

Problem C.10 Consider the following quadratic program

$$\begin{array}{ll} \text{minimize} & q(x_1, x_2) \\ \text{subject to} & x_1 + x_2 \leq 1 \\ & x_1 \geq 0,\ x_2 \geq 0 \end{array}$$

Draw the feasible set, and the level curves where the values of objective function

$$q(x) = (x_1 - 1)^2 + (x_2 - 1)^2$$

take on the values $0, 4$ and 9. Identify the optimal solution. Repeat the exercise for the objective

$$q(x) = x_1^2 + x_2^2.$$

Problem C.11 You are given the following data

$$x = \begin{pmatrix} 0.53 & 1.77 & 1.95 & 2.15 & 2.86 & 3.50 & 3.71 & 3.79 & 4.11 & 4.21 \end{pmatrix}$$
$$y = \begin{pmatrix} -0.67 & 3.57 & 4.59 & 7.04 & 16.75 & 33.49 & 40.24 & 43.54 & 54.85 & 59.90 \end{pmatrix}$$

and would like to fit a third-degree polynomial $\hat{y} = \hat{f}(x) = ax^3 + bx^2 + cx + d$ to minimize the loss

$$\sum_{i=1}^{10} (\hat{y}_i - y_i)^2$$

Formulate the problem as a least-squares problem and compute the optimal parameters $\hat{f}(x)$.

Problem C.12 You are given the following data

$$\begin{aligned} x = \big(\; & 0.38 \quad 0.61 \quad 1.09 \quad 2.35 \quad 2.37 \quad 2.72 \quad 4.18 \quad 4.28 \quad 4.37 \quad 4.50 \quad \ldots \\ & 5.38 \quad 5.61 \quad 6.09 \quad 7.35 \quad 7.37 \quad 7.72 \quad 9.18 \quad 9.28 \quad 9.37 \quad 9.50 \quad \big) \\ y = \big(\; & -0.79 \quad -0.54 \quad -0.34 \quad 7.94 \quad 9.87 \quad 14.63 \quad 57.93 \quad 63.99 \quad 67.31 \quad 74.26 \quad \ldots \\ & 130.0 \quad 143.3 \quad 171.0 \quad 234.2 \quad 235.3 \quad 250.6 \quad 303.5 \quad 306.7 \quad 309.4 \quad 313.0 \quad \big) \end{aligned}$$

Visual inspection of the data reveals that there is an apparent break-point around $x = 5$. Use quadratic programming to estimate the parameters of a piecewise polynomial

$$f = \begin{cases} ax^3 + bx^2 + cx + d & \text{if } x \leq 5 \\ a'x^3 + b'x^2 + c'x + d' & \text{if } x \geq 5 \end{cases}$$

under the constraint that f should be continuous and continuously differentiable.

Problem C.13 We are given two set of points $\{x_1, \ldots, x_N\}$ and $\{y_1, \ldots, y_M\}$ and would like to find a linear classifier, *i.e.*, an affine function $f(x) = a^\top x + b$ such that

$$a^T x_i + b < 0, \quad i = 1, \ldots, N, \qquad a^T y_i + b > 0, \quad i = 1, \ldots, M.$$

Since the inequalities are homogeneous in (a, b) we normalize the inequalities and require that

$$a^T x_i + b \leq -1, \quad i = 1, \ldots, N, \qquad a^T y_i + b \geq 1, \quad i = 1, \ldots, M.$$

instead. The set

$$\mathcal{S} = \{ z \in \mathbb{R}^n \mid -1 \leq a^T z + b \leq 1 \}$$

defines a "slab" that separates the two point clouds $\{x_i\}$ and $\{y_i\}$. Verify that the width (also known as the margin) of the slab is given by $2/\|a\|_2$ and formulate the problem of finding the linear classifier with maximum margin as a quadratic programming problem.

Problem C.14 We have considered linear programming problems on the form

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax \leq b \end{array} \tag{C.18}$$

Show that the associated dual problem is itself a linear program on the form

$$\begin{array}{ll} \text{maximize} & d^T z \\ \text{subject to} & Ez = f, \quad z \geq 0. \end{array} \tag{C.19}$$

Given an optimal solution $z^\star$ to (C.19), what is the corresponding optimal solution $x^\star$ to (C.18)?

Problem C.15 The projection of a vector u onto a convex set X, $\mathrm{Proj}_X(u)$, is the vector $x^\star \in X$ that is closest to u, i.e. the solution to the optimization problem

$$\begin{array}{ll} \underset{x}{\text{minimize}} & \frac{1}{2}\|x-u\|_2^2 \\ \text{subject to} & x \in X \end{array}$$

(a) Use the definition of the projection operator to show that the projection onto the non-negative orthant, $X = \{x \mid x \geq 0\}$ is simply $\max(0, x_i)$ for each component x_i of x.
(b) Show that the projection is non-expansive, i.e. that it satisfies

$$\|\mathrm{Proj}_X(u) - \mathrm{Proj}_X(v)\|_2 \leq \|u-v\|_2 \qquad \text{for all } u, v \in \mathbb{R}^n.$$

Hint. Note that the optimality condition for constrained optimization implies that $(\mathrm{Proj}_X(u)-u)^\top(x-\mathrm{Proj}_X(u)) \geq 0$ for all $x \in X$, and that $\mathrm{Proj}_X(v) \in X$.

Problem C.16 Consider the quadratic program

$$\begin{array}{ll} \text{minimize} & z^\top P z + 2q^\top z + r \\ \text{subject to} & Az \leq b \end{array}$$

(a) Form the associated Lagrangian $L(z, \lambda)$ and use the completion-of-squares lemma to show that the corresponding dual problem is a QP on the form

$$\begin{array}{ll} \text{maximize} & \lambda^\top P_d \lambda + 2q_d^\top \lambda + r_d \\ \text{subject to} & \lambda \geq 0 \end{array}$$

where $P_d = -AP^{-1}A^\top/4$, $q_d = -(AP^{-1}q+b)/2$ and $r_d = r - q^\top P^{-1} q$. Determine an expression for how the optimal primal solution $z^\star$ depends on the optimal dual solution $\lambda^\star$.
(b) Verify your expressions by numerically solving the QP defined by $P = I$, $q = -2\mathbf{1}$, $r = 0$, $A = \begin{pmatrix} I & -I \end{pmatrix}^\top$ and $b = \mathbf{1}$. This QP attempts to minimize the Euclidean distance between $(2,2)$ and the feasible set $\|z\|_\infty \leq 1$, so $z^\star = (1,1)$. But what is the optimal solution to the dual problem, and can you recover the primal optimizer from the dual?
(c) Implement the projected gradient descent method and solve the dual QP. How many iterations do you need to get a solution of good accuracy?

Hint. Recall that the x that maximizes $f(x)$ minimizes $-f(x)$. You can use this fact in (b) and (c).

Problem C.17 A set of N players in a market share a common resource. The utility of player i being assigned x_i amount of resource is characterized by a concave utility function $u_i(x_i)$. We are interested in finding the allocation that maximizes the "social welfare" under the resource constraint:

$$\begin{array}{ll} \text{maximize} & \sum_{i=1}^N u_i(x_i) \\ \text{subject to} & \sum_i x_i = x_{\text{tot}}. \end{array}$$

Use duality theory to show that the players can be aligned to the social optimum by introducing a price p on the resource under the assumption that they maximize their utility minus resource cost

$$\text{maximize} \quad u_i(x_i) - px_i$$

Hint. A function u is concave if and only if $-u$ is convex. You can therefore consider the convex minimization problem

$$\begin{array}{ll} \text{minimize} & \sum_i -u_i(x_i) \\ \text{subject to} & \sum_i x_i = x_{\text{tot}}. \end{array}$$

D. Additional proofs

D.1 Proofs from Chapter 1

Proof of reachable subspace decomposition

The proof relies on the following lemma.

Lemma D.1 The set of reachable states is A-invariant, *i.e.* if x_{tgt} is reachable, then so is Ax_{tgt}.

Proof. We begin by establishing that $\text{range}(A\mathcal{C}_n) \subseteq \text{range}(\mathcal{C}_n)$. To this end, note that

$$A\mathcal{C}_n = \begin{pmatrix} A^nB & A^{n-1}B & \dots & AB \end{pmatrix}$$

The last $n-1$ blocks are present in the controllability matrix, and by the Cayley-Hamilton theorem, we can write A^nB as a linear combination of the columns of the controllability matrix. Hence, $\text{range}(A\mathcal{C}_n) \subseteq \text{range}(\mathcal{C}_n)$. Now, if x_{tgt} is reachable, then there exists $\mathcal{U}_n$ such that $x_{\text{tgt}} = \mathcal{C}_n\mathcal{U}_n$ and $Ax_{\text{tgt}} = A\mathcal{C}_n\mathcal{U}_n$. As the range of $A\mathcal{C}_n$ is contained in the range of $\mathcal{C}_n$, there exists $\mathcal{V}_n$ such that $Ax_{\text{tgt}} = \mathcal{C}_n\mathcal{V}_n$. Hence Ax_{tgt} is also reachable. ■

We are now ready to proceed and prove Theorem 1.2.2. Let $V \in \mathbb{R}^{n\times n_r}$ be a matrix whose columns span the range of $\mathcal{C}_n$, and let $W \in \mathbb{R}^{n\times n-n_r}$ be a matrix whose columns are independent of each other and those of V. Consider the state transformation $z = T^{-1}x$ with $T = \begin{pmatrix} V & W \end{pmatrix}$, i.e.

$$x = \sum_{i=1}^{n_r} z_i v_i + \sum_{j=n_r+1}^{n} z_j w_{j-n_r}.$$

Note that $z_j = 0$ for all $j = n_r+1,\dots,n$ if x lies in the range of $\mathcal{C}_n$. In other words, the last $n-n_r$ components of $T^{-1}x$ are zero if $x \in \text{range}(\mathcal{C}_n)$.

Since the range of $\mathcal{C}_n$ is A-invariant, Av_i lies in the range of $\mathcal{C}_n$ and the last $n-n_r$ components of $T^{-1}Av_i$ must equal zero. Hence, the $n-n_r$ last rows of $T^{-1}AV$ must be zero, and $T^{-1}AT = T^{-1}A\begin{pmatrix} V & U \end{pmatrix}$ has the desired block structure. Similarly, B lies in the range of $\mathcal{C}_n$ so $T^{-1}B$ must also have the desired structure.

D.2 Proofs from Chapter 2

Proof of existence of matrix series

Lemma D.2 Let $Q \in \mathbb{R}^{n\times n}$ be positive semidefinite and $A \in \mathbb{R}^{n\times n}$ have spectral radius strictly less than one. Then, the series

$$P = \sum_{k=0}^{\infty} (A^\top)^k Q A^k$$

converges.

Proof. We will show that there is an induced matrix norm, in which the matrix series is absolutely convergent. Since A has spectral radius strictly less than one, Theorem A.4.1 guarantees that there is a matrix norm $\|\cdot\|$ such that $\|A\| \le r < 1$. In this norm, we also have $\|A^\top\| \le r$ and thus

$$\sum_{k=0}^{\infty} \|(A^\top)^k Q A^k\| \le \sum_{k=0}^{\infty} \|A^\top\|^k \|Q\| \|A\|^k = \|Q\| \sum_{k=0}^{\infty} \left(r^2\right)^k = \|Q\| \frac{1}{1-r^2}.$$

Now, by Theorem A.4.2 this implies that the matrix series itself converges.

D.3 Proofs from Chapter 4

Proof that the infinite-horizon LQ cost-to-go function is quadratic

We begin by establishing that $v(\lambda x_0) = \lambda^2 x_0$. Recall that

$$x_t = A^t x_0 + \sum_{k=0}^{t-1} A^k B u_{t-1-k}.$$

so multiplying both x_0 and the input sequence by the same scalar λ leads to

$$\lambda x_t = A^t \lambda x_0 + \sum_{k=0}^{t-1} A^k B \lambda u_{t-1-k}.$$

Introducing

$$J(x_0, u) = \sum_{t=0}^{\infty} x_t^\top Q x_t + u_t^\top R u_t$$

we note that $J(\lambda x_0, \lambda u) = \lambda^2 J(x_0, u)$, and therefore

$$v(\lambda x_0) = \lambda^2 v(x_0). \tag{D.1}$$

Next, consider two initial states x_0 and $\tilde{x}_0$. We would like to show that

$$V(x_0 + \tilde{x}_0) + V(x_0 - \tilde{x}_0) = 2V(x_0) + 2V(\tilde{x}_0).$$

To this end, we let u and $\tilde{u}$ be two input sequences and consider

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t \\ \tilde{x}_{t+1} &= A\tilde{x}_t + B\tilde{u}_t \end{aligned}$$

Adding or subtracting the above equations yields

$$x_{t+1} + \tilde{x}_{t+1} = A(x_t + \tilde{x}_t) + B(u_t + \tilde{u}_t), \qquad x_{t+1} - \tilde{x}_{t+1} = A(x_t - \tilde{x}_t) + B(u_t - \tilde{u}_t).$$

Therefore,

$$\begin{aligned} J(u+\tilde{u},x_0+\tilde{x}_0)+J(u-\tilde{u},x_0-\tilde{x}_0) &= \sum_{t=0}^{\infty}((x_t+\tilde{x}_t)^T Q(x_t+\tilde{x}_t)+(x_t-\tilde{x}_t)^T Q(x_t-\tilde{x}_t) \\ &\quad+\sum_{t=0}^{\infty}(u_t+\tilde{u}_t)^T R(u_t+\tilde{u}_t)+(u_t-\tilde{u}_t)^T R(u_t-\tilde{u}_t)) \\ &= \sum_{t=0}^{\infty} 2x_t^T Q x_t + 2\tilde{x}_t^T Q\tilde{x}_t + 2u_t^T R u_t + 2\tilde{u}_t^T R\tilde{u}_t \\ &= 2J(u,x_0)+2J(\tilde{u},\tilde{x}_0) \end{aligned}$$

By minimizing both sides with respect to u and $\tilde{u}$ we obtain

$$\min_{u,\tilde{u}}\{J(u+\tilde{u},x_0+\tilde{x}_0)+J(u-\tilde{u},x_0-\tilde{x}_0)\} = \min_{u} 2J(u,x_0)+\min_{\tilde{u}} 2J(\tilde{u},\tilde{x}_0)$$

The right-hand side of this equation is simply $2v(x_0)+2v(\tilde{x}_0)$. When it comes to the left-hand side,

$$\min_{u,\tilde{u}}\{J(u+\tilde{u},x_0+\tilde{x}_0)+J(u-\tilde{u},x_0-\tilde{x}_0)\} \geq \min_{u,\tilde{u}} J(u+\tilde{u},x_0+\tilde{x}_0)+\min_{u,\tilde{u}} J(u-\tilde{u},x_0-\tilde{x}_0)$$

so we have established the inequality

$$v(x_0+\tilde{x}_0)+v(x_0-\tilde{x}_0) \leq 2v(x_0)+2v(\tilde{x}_0). \tag{D.2}$$

Applying this inequality to $x_0=(x_0'+\tilde{x}_0')/2$ and $\tilde{x}_0=(x_0'-\tilde{x}_0')/2$, as $v(\lambda x_0)=\lambda^2 v(x_0)$, we find

$$v(x_0')+v(\tilde{x}_0') \leq \frac{1}{2}v(x_0'+\tilde{x}_0')+\frac{1}{2}v(x_0'-\tilde{x}_0')$$

i.e.

$$v(x_0'+\tilde{x}_0')+v(x_0'-\tilde{x}_0') \geq 2v(x_0')+2v(\tilde{x}_0') \tag{D.3}$$

Since (D.2) and (D.3) hold for all arguments, it must be that

$$v(x_0+\tilde{x}_0)+v(x_0-\tilde{x}_0) = 2v(x_0)+2v(\tilde{x}_0). \tag{D.4}$$

We are now ready to show that $v'(x_0)=\nabla v(x_0)$ is linear in x_0. To this end, we take the derivative of both sides of (D.4) with respect to x_0 and $\tilde{x}_0$, respectively

$$\begin{aligned} v'(x_0+\tilde{x}_0)+v'(x_0-\tilde{x}_0) &= 2v'(x_0) \\ v'(x_0+\tilde{x}_0)-v'(x_0-\tilde{x}_0) &= 2v'(\tilde{x}_0) \end{aligned}$$

and add these equations to find

$$v'(x_0+\tilde{x}_0) = v'(x_0)+v'(\tilde{x}_0)$$

In addition, differentiating both sides of (D.1) with respect to x_0 yields $\lambda v'(\lambda x_0)=\lambda^2 v'(x_0)$, i.e.

$$v'(\lambda x_0) = \lambda v'(x_0)$$

We can therefore conclude that v' is linear in x_0, *i.e.* on the form

$$v'(x_0) = Mx_0$$

for some $M \in \mathbb{R}^{n\times n}$.

Next, (D.1) implies that $v(0) = 0$, and therefore

$$v(x_0) = \int_{s=0}^{1} v'(sx_0)^T x_0 \, ds = \int_{s=0}^{1} s \cdot x_0^T M^T x_0 \, ds = \frac{1}{2} x_0^T M^T x_0$$

From the appendix about quadratic forms, we know that we can let M be symmetric without loss of generality, i.e. that we can write

$$v(x_0) = x_0^T P x_0$$

with $P = (M/2 + M^T/2)/2$. Since the cost is non-negative, P must be positive semi-definite. This concludes our proof.

Proof of the least-squares filter equations

We begin by eliminating the disturbances w_t and v_t from (4.20). Since

$$\begin{aligned} w_t &= x_{t+1} - (Ax_t + Bu_t) \\ v_t &= y_t - Cx_t \end{aligned}$$

the optimal estimate time t is obtained by minimizing

$$J_t(x_0,\ldots,x_t) = r_0(x_0) + \sum_{k=0}^{t-1} r_v(x_k) + r_w(x_k, x_{k+1}) \tag{D.5}$$

where

$$\begin{aligned} r_0(x_0) &= (x_0 - \bar{x}_0)^\top R_0 (x_0 - \bar{x}_0) \\ r_v(x_t) &= (y_t - Cx_t)^\top R_v (y_t - Cx_t) \\ r_w(x_t, x_{t+1}) &= (x_{t+1} - (Ax_t + Bu_t))^\top R_w (x_{t+1} - (Ax_t + Bu_t)). \end{aligned}$$

We note that $J_t(x_0,\ldots,x_t)$ has the multi-stage structure

$$J(x_0,\ldots,x_t) = g_0(x_0) + \sum_{k=1}^{t} g_k(x_{k-1}, x_k)$$

discussed in Section 3.3 with $g_0(x_0) = r_0(x_0)$ and $g_k(x_{k-1}, x_k) = r_v(x_{k-1}) + r_w(x_{k-1}, x_k)$. The optimal solution can therefore be computed using forward induction. It will be convenient to structure the solution in terms of $\hat{x}_{t|s}$, *i.e.* the best estimate of x_t given information up until time s. We let $\hat{x}_{0|-1}$ denote the best guess of x_0 before any measurements are made. Hence, $\hat{x}_{0|-1} = \bar{x}_0$ and the initial arrival cost has the form

$$w_0(x_0) = (x_0 - \hat{x}_{0|-1})^T R_0 (x_0 - \hat{x}_{0|-1})$$

To be able to apply dynamic programming in an efficient way, it would be convenient if the arrival cost remains quadratic. It turns out that the arrival costs will admit the parameterization

$$w_k(x_k) = (x_k - \hat{x}_{k|k-1})^\top S_k (x_k - \hat{x}_{k|k-1}) + s_k.$$

for some positive semidefinite matrix S_k and non-negative constant s_k. This claim is clearly true for $k = 0$. We can now proceed by induction from an arbitrary stage k. The forward induction step is

$$w_{k+1}(x_{k+1}) = \min_{x_k} \{ w_k(x_k) + g_{k+1}(x_k, x_{k+1}) \} = \min_{x_t} \{ w_k(x_k) + r_v(x_k) + r_w(x_k, x_{k+1}) \}$$

To structure the optimal solution into a measurement update and a prediction step, we will derive the optimal solution by applying the completion-of-squares lemma twice. First, we combine the two first terms into a single quadratic

$$\begin{aligned}\bar{w}_k(x_k) &= w_k(x_k) + r_v(x_k) = \\ &= (x_k - \hat{x}_{k|k-1})^\top S_k (x_k - \hat{x}_{k|k-1}) + s_k + (y_k - Cx_k)^\top R_v (y_k - Cx_k) = \\ &= x_k^\top (S_k + C^\top R_v C) x_k - 2(S_k \hat{x}_{k|k-1} + C^\top R_v y_k)^\top x_k + s_k + \hat{x}_{k|k-1}^\top S_k \hat{x}_{k|k-1} + y_k^\top R_v y_k\end{aligned}$$

This function represents the accumulated cost up until measurement k, and depends on $y_0, \dots, y_k$. We will refer to the minimizer of this expression as $\hat{x}_{k|k}$. By the completion-of-squares lemma,

$$\hat{x}_{k|k} = (S_k + C^\top R_v C)^{-1}(S_k \hat{x}_{k|k-1} + C^\top R_v y_k) = \hat{x}_{k|k-1} + K_k(y_k - C\hat{x}_{k|k-1})$$

where we have introduced $K_k = (S_k + C^\top R_v C)^{-1} C^\top R_v$. Moreover, $\bar{w}_k(x_k)$ can be expressed as

$$\bar{w}_k(x_k) = (x_k - \hat{x}_{k|k})^\top \bar{S}_k (x_k - \hat{x}_{k|k}) + \bar{s}_k$$

where

$$\begin{aligned}\bar{S}_k &= S_k + C^\top R_v C \\ \bar{s}_k &= s_k + \hat{x}_{k|k-1}^\top S_k \hat{x}_{k|k-1} + y_k^\top R_v y_k - \hat{x}_{k|k}^\top (S_k + C^\top R_v C)\hat{x}_{k|k}\end{aligned}$$

While this is an important intermediate result, we need to complete the induction to find how S_k and s_k evolve. To this end, we re-write the induction step in terms of $\bar{w}_k$ as

$$\begin{aligned}w_{k+1}(x_{k+1}) &= \min_{x_k} \{\bar{w}_k(x_k) + r_w(x_k, x_{k+1})\} = \\ &= \min_{x_k} \Big\{ x_k^\top (\bar{S}_k + A^\top R_w A) x_k - 2(\bar{S}_k \hat{x}_{k|k} + A^\top R_w (x_{k+1} - Bu_k))^\top x_k + \\ &\qquad + \hat{x}_{k|k}^\top \bar{S}_k \hat{x}_{k|k} + (x_{k+1} - Bu_k)^\top R_w (x_{k+1} - Bu_k) + \bar{s}_k \Big\}\end{aligned} \tag{D.6}$$

By the least-squares lemma, the optimizer is

$$x_k^\star = (\bar{S}_k + A^\top R_w A)^{-1}(\bar{S}_k \hat{x}_{k|k} + A^\top R_w (x_{k+1} - Bu_k))$$

and the arrival cost can be expressed as

$$w_{k+1}(x_{k+1}) = \hat{x}_{k|k}^\top \bar{S}_k \hat{x}_{k|k} + (x_{k+1} - Bu_k)^\top R_w (x_{k+1} - Bu_k) + \bar{s}_k - (x_k^\star)^\top (\bar{S}_k + A^\top R_w A) x_k^\star$$

It now remains to massage this expression to show that it can be written on the form

$$w_{k+1}(x_{k+1}) = (x_{k+1} - \hat{x}_{k+1|k})^\top S_{k+1} (x_{k+1} - \hat{x}_{k+1|k}) + s_{k+1}$$

To this end, let $\hat{x}_{k+1|k} = A\hat{x}_{k|k} + Bu_k$ and $e_{k+1} = x_{k+1} - \hat{x}_{k+1|k}$. Then $x_{k+1} - Bu_k = e_k + A\hat{x}_{k|k}$ and

$$\begin{aligned}w_{k+1}(x_{k+1}) &= \hat{x}_{k|k}^\top \bar{S}_k \hat{x}_{k|k} + (e_{k+1} + A\hat{x}_{k|k})^\top R_w (e_{k+1} + A\hat{x}_{k|k}) + \bar{s}_k \\ &- \quad (\bar{S}_k \hat{x}_{k|k} + A^\top R_w (e_{k+1} + A\hat{x}_{k|k}))^\top (\bar{S}_k + A^\top R_w A)^{-1} (\bar{S}_k \hat{x}_{k|k} + A^\top R_w (e_{k+1} + A\hat{x}_{k|k}) = \\ &= e_{k+1}^\top (R_w - R_w A (\bar{S}_k + A^\top R_w A)^{-1} A^\top R_w) e_{k+1} + \bar{s}_k\end{aligned}$$

which is on the desired form. Hence, $S_{k+1} = R_w - R_w A(\bar{S}_k + A^\top R_w A)^{-1} A^\top R_w$ and $s_{k+1} = \bar{s}_k$ satisfy the recursion and we are done.

Proof of the Kalman filter equations

Recall that the Kalman filter cost is parameterized as

$$J_N = (x_{0|-1}^T - \mu_0)^T \Sigma_0^{-1} (x_{0|-1} - \mu_0) + \sum_{k=0}^{N-1} w_k^T \Sigma_1^{-1} w_k + v_k^T \Sigma_2^{-1} v_k$$

i.e. the same as the least-squares filter with $\Sigma_0 = R_0^{-1}$, $\Sigma_1 = R_1^{-1}$ and $\Sigma_2 = R_2^{-1}$. It maintains an arrival cost parameterized in terms of $P_t = S_t^{-1}$. Hence, we do not need to re-derive the update equations, but rather manipulate the update equations so that they express P_{t+1} in terms of P_t, Σ_0, Σ_1, Σ_2 and the system matrices. To this end, we use the matrix identities in Proposition A.2.6.

We can now transform the update equations of the least-squares filter one by one. For $\bar{K}_t$, we let $Y = S_t$, $Z = C$, $X = R_2$ and apply (A.2):

$$\begin{aligned} \bar{K}_t &= (S_t + C^T R_2 C)^{-1} C^T R_2 = \\ &= S_t^{-1} C^T (R_2^{-1} + C S_t^{-1} C)^{-1} = P_t C^T (\Sigma_2 + C P_t C^T)^{-1} \end{aligned}$$

No modification is needed for $\hat{x}_{t|t-1}$, while the update for $\bar{P}_t = \bar{S}_t^{-1}$ can be derived by direct application of (A.1) with $X = S_t$, $Z = C$ and $Y = R_2$:

$$\begin{aligned} \bar{P}_t &= (S_t + C^T R_2 C)^{-1} = S_t^{-1} - S_t^{-1} C^T (R_2^{-1} + C S_t^{-1} C^T)^{-1} C S_t^{-1} = \\ &= P_t - P_t C^T (\Sigma_2 + C P_t C^T)^{-1} C P_t. \end{aligned}$$

Again, no change is needed for $\hat{x}_{t|t}$, while $P_{t+1} = S_{t+1}^{-1}$ can be found by the reverse application of (A.1) with $X = R_1^{-1}, Y = \bar{P}_t$ and $Z = A$:

$$P_{t+1} = \Sigma_1 + A \bar{P}_t A^T.$$

The proof is complete.

Bibliography

[1] Abdul Afram and Farrokh Janabi-Sharifi. *Model Predictive Control for Building Energy Systems*. Technical report. Pacific Northwest National Laboratory, 2019 (cited on page 12).

[2] ANDRITZ Automation. *BrainWave Model Predictive Control for the Process Industries*. Technical report. ANDRITZ, 2019. URL: `https://www.andritz.com/resource/blob/15102/cb1881abbfb17573ecb6f60c28e04b9a/aa-intech-brainwave-model-predictive-control-for-the-process-industries-data.pdf` (cited on page 12).

[3] Daniel Arnström, Alberto Bemporad, and Daniel Axehill. "A Dual Active-Set Solver for Embedded Quadratic Programming Using Recursive LDL^T Updates". In: *IEEE Transactions on Automatic Control* 67.8 (2022), pages 4362–4369. DOI: `10.1109/TAC.2022.3176430` (cited on page 224).

[4] K. J. Åström and B. Wittenmark. *Computer controlled systems: theory and design*. 2nd. Prentice Hall, 1990 (cited on page 32).

[5] Alberto Bemporad et al. "The explicit linear quadratic regulator for constrained systems". In: *Automatica* 38.1 (2002), pages 3–20. ISSN: 0005-1098. DOI: `https://doi.org/10.1016/S0005-1098(01)00174-1`. URL: `https://www.sciencedirect.com/science/article/pii/S0005109801001741` (cited on pages 160, 227).

[6] D. Bertsekas. *Nonlinear programming*. 2nd. Athena Scientific, 1999 (cited on page 213).

[7] D. P. Bertsekas. *Dynamic Programming and Optimal Control, Vol. II: Approximate Dynamic Programming*. Athena Scientific, 2012 (cited on pages 91, 92).

[8] Lars Blackmore. "Autonomous precision landing of space rockets". In: *The Bridge* 40.4 (2010), pages 15–20 (cited on page 12).

[9] Francesco Borrelli, Alberto Bemporad, and Manfred Morari. *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press, 2017. DOI: `10.1017/9781139061759` (cited on page 3).

[10] S. P. Boyd and L. Vandenberghe. *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2004. ISBN: 0521833787 (cited on pages 213, 216).

[11] R. Cagienard et al. "Move blocking strategies in receding horizon control". In: *Journal of Process Control* 17.6 (2007), pages 563–570. ISSN: 0959-1524. DOI: https://doi.org/10.1016/j.jprocont.2007.01.001. URL: https://www.sciencedirect.com/science/article/pii/S0959152407000030 (cited on page 165).

[12] Control Global. *2022 Control Process Automation Hall of Fame: Brian Ramaker*. 2022. URL: https://www.controlglobal.com/home/article/11287750/2022-control-process-automation-hall-of-fame-brian-ramaker (cited on page 11).

[13] Charles R Cutler and Brian L Ramaker. "Dynamic matrix control–a computer control algorithm". In: *Proceedings of the Joint Automatic Control Conference* 17 (1980), page 72 (cited on page 11).

[14] Philipp Elbert, Soren Ebbesen, and Lino Guzzella. "Implementation of Dynamic Programming for n-Dimensional Optimal Control Problems With Final State Constraints". In: *IEEE Transactions on Control Systems Technology* 21.3 (2013), pages 924–931 (cited on page 89).

[15] Hans Joachim Ferreau et al. "qpOASES: a parametric active-set algorithm for quadratic programming". In: *Mathematical Programming Computation* 6.4 (2014), pages 327–363. DOI: 10.1007/s12532-014-0071-1. URL: https://doi.org/10.1007/s12532-014-0071-1 (cited on page 224).

[16] Gene Franklin, J. D. Powell, and Abbas Emami-Naeini. *Feedback Control of Dynamic Systems (5th Edition)*. 5th edition. Prentice Hall, 2005 (cited on page 32).

[17] Gianluca Frison et al. "High-performance small-scale solvers for linear Model Predictive Control". In: *2014 European Control Conference (ECC)*. 2014, pages 128–133. DOI: 10.1109/ECC.2014.6862490 (cited on page 223).

[18] Pontus Giselsson. "Improved Fast Dual Gradient Methods for Embedded Model Predictive Control". In: *IFAC Proceedings Volumes* 47.3 (2014). 19th IFAC World Congress, pages 2303–2309. ISSN: 1474-6670. DOI: https://doi.org/10.3182/20140824-6-ZA-1003.00295. URL: https://www.sciencedirect.com/science/article/pii/S1474667016419555 (cited on page 222).

[19] Michael Grant and Stephen Boyd. *CVX: Matlab Software for Disciplined Convex Programming, version 2.1*. http://cvxr.com/cvx. Mar. 2014 (cited on pages 77, 79).

[20] M. Grundelius. "Methods for Control of Liquid Slosh". eng. Lund University, 2001, page 167 (cited on page 78).

[21] Jiefeng Hu, Jianguo Zhu, and David G Dorrell. "Model predictive control for power grid applications: A systematic review". In: *IEEE Transactions on Power Systems* 36.5 (2021), pages 4327–4335 (cited on page 12).

[22] Kalypso. *Advanced Process Control*. 2023. URL: https://kalypso.com/services/data-science/advanced-process-control (cited on page 12).

[23] H. Kwakernaak and R. Sivan. *Linear Optimal Control Systems*. Chichester: Wiley-Interscience, 1972 (cited on page 113).

[24] Lennart Ljung. *System Identification: Theory for the User*. USA: Prentice-Hall, Inc., 1986. ISBN: 0138816409 (cited on page 177).

[25] J. Lofberg. "YALMIP : a toolbox for modeling and optimization in MATLAB". In: *2004 IEEE International Conference on Robotics and Automation (IEEE Cat. No.04CH37508)*. Sept. 2004, pages 284–289. DOI: 10.1109/CACSD.2004.1393890 (cited on page 77).

[26] J.M. Maciejowski. *Predictive Control with Constraints*. England.: Prentice Hall, 2002 (cited on pages 3, 147).

[27] J. Mattingley and S. Boyd. “CVXGEN: A Code Generator for Embedded Convex Optimization”. In: *Optimization and Engineering* 12.1 (2012), pages 1–27 (cited on page 223).

[28] Mordor Intelligence. *Advanced Process Control Market - Growth, Trends, and Forecasts (2025-2030).* Technical report. Mordor Intelligence, 2024. URL: `https://www.mordorintelligence.com/industry-reports/global-advanced-process-control-market-industry-analysis-gro` (cited on page 12).

[29] Gabriele Pannocchia. “Offset-free tracking MPC: A tutorial review and comparison of different formulations”. In: *2015 European Control Conference (ECC)*. 2015, pages 527–532. DOI: `10.1109/ECC.2015.7330597` (cited on page 173).

[30] Samuel Privara et al. “Experimental evaluation of model predictive control for thermal comfort and energy efficiency in buildings”. In: *Energy and Buildings* 153 (2017), pages 150–157 (cited on page 12).

[31] S Joe Qin and Thomas A Badgwell. “A survey of industrial model predictive control technology”. In: *Control Engineering Practice* 11.7 (2003), pages 733–764 (cited on page 11).

[32] James Rawlings. *Model predictive control : theory and design*. Madison, Wis: Nob Hill Pub, 2009. ISBN: 0975937707 (cited on page 3).

[33] Jacques Richalet et al. “Model predictive heuristic control: Applications to industrial processes”. In: *Automatica* 14.5 (1978), pages 413–428 (cited on page 11).

[34] Rockwell Automation. *Advanced Process Control Solutions*. Technical report. Rockwell Automation, 2019. URL: `https://literature.rockwellautomation.com/idc/groups/literature/documents/pp/rsppfb-pp001_-en-p.pdf` (cited on page 12).

[35] Petter Tøndel, Tor Arne Johansen, and Alberto Bemporad. “An algorithm for multi-parametric quadratic programming and explicit MPC solutions”. In: *Automatica* 39.3 (2003), pages 489–497. ISSN: 0005-1098. DOI: `https://doi.org/10.1016/S0005-1098(02)00250-9`. URL: `https://www.sciencedirect.com/science/article/pii/S0005109802002509` (cited on page 227).

[36] Günter M. Ziegler. *Lectures on polytopes*. New York: Springer-Verlag, 1995, pages –. URL: `http://www.worldcat.org/search?qt=worldcat_org_all&q=9780387943657` (cited on page 211).