Lecture Notes on Optimal Estimation and Control

Moritz Diehl

April 28, 2014

Contents

Chapter 1

Introduction

Optimal control regards the *optimization* of *dynamic systems*. In this lecture we identify dynamic systems with processes that are evolving with time and that can be characterized by *states* x that allow us to predict the future behavior of the system. If the state is not known, we first need to *estimate* it based on the available measurement information. The estimation process is very often optimization-based and uses the same optimization methods that are used for optimal control. This is the reason why this lecture bundles both optimal control and estimation in one single course. Often, a dynamic system can be controlled by a suitable choice of inputs that we denote as *controls* u in this script. In optimal control, these controls shall be chosen optimally in order to optimize some *objective function* and respect some *constraints*.

For optimal control, we might think of an electric train where the state x consists of the current position and velocity, and where the control u is the engine power that the train driver can choose at each moment. We might regard the motion of the train on a time interval $[t_{\mathrm{init}}, t_{\mathrm{fin}}]$, and the objective could be to minimize the consumed energy to drive from Station A to Station B, and one of the constraints would be that the train should arrive in Station B at the fixed final time, t_{fin}.

On the other hand, in optimization-based estimation, we treat unknown disturbances as inputs, and the objective function is the misfit between the actual measurements and their model predictions. The resulting optimization problems are mathematically of the same form as the problems of optimal control, with the disturbances as controls. For this reason, we focus the larger part of the course on the topic "optimal control". At several occasions we specialize to estimation problems as a special case.

A typical property of a dynamic system is that knowledge of an *initial state* x_{init} and a *control input trajectory* $u(t)$ for all $t \in [t_{\mathrm{init}}, t_{\mathrm{fin}}]$ allows one to determine the whole *state trajectory* $x(t)$ for $t \in [t_{\mathrm{init}}, t_{\mathrm{fin}}]$. As the motion of a train can very well be modelled by Newton's laws of motion, the usual description of this dynamic system is deterministic and in continuous time and with continuous states.

But dynamic systems and their mathematical models can come in many variants, and it is useful to properly define the names given commonly to different dynamic system classes, which we do in the next section. Afterwards, we will discuss two important classes, continuous time and discrete time systems, in more mathematical detail, before we give an overview of optimization problem classes and finally outline the contents of the lecture chapter by chapter.

1.1 Dynamic System Classes

In this section, let us go, one by one, through the many dividing lines in the field of dynamic systems.

Continuous vs Discrete Time Systems

Any dynamic system evolves over time, but time can come in two variants: while the physical time is continuous and forms the natural setting for most technical and biological systems, other dynamic systems can best be modelled in discrete time, such as digitally controlled sampled-data systems, or games.

We call a system a *discrete time system* whenever the time in which the system evolves only takes values on a predefined time grid, usually assumed to be integers. If we have an interval of real numbers, like for the physical time, we call it a *continuous time system*. In this lecture, we usually denote the continuous time by the variable $t \in \mathbb{R}$ and write for example $x(t)$. In case of discrete time systems, we use an index, usually $k \in \mathbb{N}$, and write x_k for the state at time point k.

Continuous vs Discrete State Spaces

Another crucial element of a dynamic system is its state x, which often lives in a continuous state space, like the position of the train, but can also be discrete, like the position of the figures on a chess game. We define the *state space* $\mathbb{X}$ to be the set of all values that the state vector x may take. If $\mathbb{X}$ is a subset of a real vector space such as $\mathbb{R}^{n_x}$ or another differentiable manifold, we speak of a *continuous state space*. If $\mathbb{X}$ is a finite or a countable set, we speak of a *discrete state space*. If the state of a system is described by a combination of discrete and continuous variables we speak of a *hybrid state space*.

A *multi-stage system* is the special case of a system with hybrid state space that develops through a sequence of stages and where the state space on each stage is continuous. An example for a multi-stage system is walking, where consecutive stages are characterized by the number of feet that are on the ground at a given moment. For multi-stage systems, the time instant when one stage ends and the next one starts can often be described by a *switching function*. This function is positive on one and negative on the other stage, and assumes the value zero at the time instant that separates the stages.

Another special case are systems that develop in a continuous state space and in continuous time, but are sometimes subject to discontinuous jumps, such as bouncing billiard balls. These can often be modelled as multi-stage systems with switching functions, plus so called *jump conditions* that describe the discontinuous state evolution at the time instant between the stages.

Finite vs Infinite Dimensional Continuous State Spaces

The class of continuous state spaces can be further subdivided into the finite dimensional ones, whose state can be characterized by a finite set of real numbers, and the infinite dimensional ones, which have a state that lives in function spaces. The evolution of finite dimensional systems in continuous time is usually described by *ordinary differential equations (ODE)* or their generalizations, such as *differential algebraic equations (DAE)*.

Infinite dimensional systems are sometimes also called *distributed parameter systems*, and in the continuous time case, their behaviour is typically described by *partial differential equations (PDE)*. An example for a controlled infinite dimensional system is the evolution of the airflow and temperature distribution in a building that is controlled by an air-conditioning system.

Continuous vs Discrete Control Sets

We denote by $\mathbb{U}$ the set in which the controls u live, and exactly as for the states, we can divide the possible control sets into *continuous control sets* and *discrete control sets*. A mixture of both is a *hybrid control set*. An example for a discrete control set is the set of gear choices for a car, or any switch that we can can choose to be either on or off, but nothing in between.

In the systems and control community, the term *hybrid system* denotes a dynamic system which has either a hybrid state or hybrid control space, or both. Generally speaking, hybrid systems are more difficult to optimize than systems with continuous control and state spaces.

However, an interesting and relevant class are hybrid systems that have continuous time and continuous states, but discrete controls. They might be called hybrid systems with *external switches* or *integer controls* and turn out to be tremendously easier to optimize than other forms of hybrid systems, if treated with the right numerical methods.

Time-Variant vs Time-Invariant Systems

A system whose dynamics depend on time is called a *time-variant system*, while a dynamic system is called *time-invariant* if its evolution does not depend on the time and date when it is happening. As the laws of physics are time-invariant, most technical systems belong to the latter class, but for example the temperature evolution of a house with hot days and cold nights might best be described by a time-variant system model. While the class of time-variant systems trivially comprises all time-invariant systems, it is an important observation that also the other direction holds: each time-variant system can be modelled by a nonlinear time-invariant system if the state space is augmented by an extra state that takes account of the advancement of time, and which we might call the “clock state”.

Linear vs Nonlinear Systems

If the state trajectory of a system depends linearly on the initial value and the control inputs, it is called a *linear system*. If the dependence is affine, one should ideally speak of an *affine system*, but often the term linear is used here as well. In all other cases, we speak of a *nonlinear system*.

A particularly important class of linear systems are *linear time invariant (LTI)* systems. An LTI system can be completely characterized in at least three equivalent ways: first, by two matrices that are typically called A and B; second, by its *step response function*; and third, by its *frequency response function*. A large part of the research in the control community is devoted to the study of LTI systems.

Controlled vs Uncontrolled Dynamic Systems

While we are in this lecture mostly interested in *controlled dynamic systems*, i.e. systems that have a control input that we can choose, it is good to remember that there exist many systems that cannot be influenced at all, but that only evolve according to their intrinsic laws of motion. These *uncontrolled systems* have an empty control set, $\mathbb{U} = \emptyset$. If a dynamic system is both uncontrolled and time-invariant it is also called an *autonomous system*.

Note that an autonomous system with discrete state space that also lives in discrete time is often called an *automaton*.

Within the class of controlled dynamic systems, of special interest are the so called *controllable systems*, which have the desirable property that their state vector x can be steered from any initial state x_{init} to any final state x_{fin} in a finite time with suitably chosen control input trajectories. Many controlled systems of interest are not completely controllable because some parts of their state space cannot be influenced by the control inputs. If these parts are stable, the system is called *stabilizable*.

Stable vs Unstable Dynamic Systems

A dynamic system whose state trajectory remains bounded for bounded initial values and controls is called a *stable system*, and an *unstable system* otherwise. For autonomous systems, *stability* of the system around a fixed point can be defined rigorously: for any arbitrarily small neighborhood $\mathcal{N}$ around the fixed point there exists a region so that all trajectories that start in this region remain in $\mathcal{N}$. *Asymptotic stability* is stronger and additionally requires that all considered trajectories eventually converge to the fixed point. For autonomous LTI systems, stability can be computationally characterized by the eigenvalues of the system matrix.

Deterministic vs Stochastic Systems

If the evolution of a system can be predicted when its initial state and the control inputs are known, it is called a *deterministic system*. When its evolution involves some random behaviour, we call it a *stochastic system*.

The movements of assets on the stockmarket are an example for a stochastic system, whereas the motion of planets in the solar system can usually be assumed to be deterministic. An interesting special case of deterministic systems with continuous state space are *chaotic systems*. These systems are so sensitive to their initial values that even knowing these to arbitrarily high, but finite, precisions does not allow one to predict the complete future of the system: only the near future can be predicted. The partial differential equations used in weather forecast models have this property, and one well-known chaotic system of ODE, the *Lorenz attractor*, was inspired by these.

Note that also games like chess can be interpreted as dynamic systems. Here the evolution is neither deterministic nor stochastic, but determined by the actions of an adverse player. If we assume that the adversary always chooses the worst possible control action against us, we enter the field of *game theory*, which in continuous state spaces and engineering applications is often denoted by *robust optimal control*.

Open-Loop vs Closed-Loop Controlled Systems

When choosing the inputs of a controlled dynamic system, one first way is decide in advance, before the process starts, which control action we want to apply at which time instant. This is called *open-loop control* in the systems and control community, and has the important property that the control u is a function of time only and does not depend on the current system state.

A second way to choose the controls incorporates our most recent knowledge about the system state which we might observe with the help of measurements. This knowledge allows us to apply feedback to the system by adapting the control action according to the measurements. In the systems and control community, this is called *closed-loop control*, but also the more intuitive term *feedback control* is used. It has the important property that the control action does depend on the current state. The map from the state to the control action is called a *feedback control policy*. In case this policy optimizes our optimization objective, it is called the *optimal feedback control policy*.

Open-loop control can be compared to a cooking instruction that says: cook the potatos for 25 minutes in boiling water. A closed-loop, or feedback control of the same process would for example say: cook the potatos in boiling water until they are so soft that they do not attach anymore to a fork that you push into them. The feedback control approach promises the better result, but requires more work as we have to take the measurements.

This lecture is mainly concerned with numerical methods of how to compute optimal open-loop controls for given objective and constraints. But the last part of the lecture is concerned with a powerful method to approximate the optimal feedback control policy: *model predictive control*, a feedback control technique that is based on the repeated solution of open-loop optimal control problems.

Focus of This Script

In this script we have a strong focus on deterministic systems with continuous state and control spaces. Mostly, we consider discrete time systems, while in a follow up lecture on numerical optimal control we discuss the treatment of continuous time systems in much more detail.

The main reason for our focus on continuous state and control spaces is that the resulting optimal control problems can efficiently be treated by derivative-based optimization methods. They are thus tremendously easier to solve than most other classes, both in terms of the solvable system sizes and of computational speed. Also, these continuous optimal control problems comprise the important class of convex optimal control problems, which allow us to find a global solution reliably and fast. Convex optimal control problems are important in their own right, but also serve as an approximation of nonconvex optimal control problems within Newton-type optimization methods.

1.2 Continuous Time Systems

Most systems of interest in science and engineering are described in form of differential equations which live in continuous time. On the other hand, all numerical simulation methods have to discretize the time interval of interest in some form or the other and thus effectively generate discrete time systems. We will thus only briefly sketch some relevant properties of continuous time systems in this section, and sketch how they can be transformed into discrete time systems. Throughout the lecture, we will mainly be concerned with discrete time systems, while we occasionally come back to the continuous time case.

Ordinary Differential Equations

A controlled dynamic system in continuous time can in the simplest case be described by an ordinary differential equation (ODE) on a time interval $[t_{\text{init}}, t_{\text{fin}}]$ by

$$\dot{x}(t) = f(x(t), u(t), t), \quad t \in [t_{\text{init}}, t_{\text{fin}}] \tag{1.1}$$

where $t \in \mathbb{R}$ is the time, $u(t) \in \mathbb{R}^{n_u}$ are the controls, and $x(t) \in \mathbb{R}^{n_x}$ is the state. The function f is a map from states, controls, and time to the rate of change of the state, i.e. $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times [t_{\text{init}}, t_{\text{fin}}] \to \mathbb{R}^{n_x}$. Due to the explicit time dependence of the function f, this is a time-variant system.

We are first interested in the question if this differential equation has a solution if the initial value $x(t_{\text{init}})$ is fixed and also the controls $u(t)$ are fixed for all $t \in [t_{\text{init}}, t_{\text{fin}}]$. In this context, the dependence of f on the fixed controls $u(t)$ is equivalent to a a further time-dependence of f, and we can redefine the ODE as $\dot{x} = \tilde{f}(x, t)$ with $\tilde{f}(x, t) := f(x, u(t), t)$. Thus, let us first leave away the dependence of f on the controls, and just regard the time-dependent uncontrolled ODE:

$$\dot{x}(t) = f(x(t), t), \quad t \in [t_{\text{init}}, t_{\text{fin}}]. \tag{1.2}$$

Initial Value Problems

An initial value problem (IVP) is given by (1.2) and the initial value constraint $x(t_{\text{init}}) = x_{\text{init}}$ with some fixed parameter x_{init}. Existence of a solution to an IVP is guaranteed under continuity of f with respect to to x and t according to a theorem from 1886 that is due to Giuseppe Peano. But existence alone is of limited interest as the solutions might be non-unique.

Example 1 (Non-Unique ODE Solution) *The scalar ODE with $f(x) = \sqrt{|x(t)|}$ can stay for an undetermined duration in the point $x = 0$ before leaving it at an arbitrary* *time t_0. It then follows a trajectory $x(t) = (t - t_0)^2/4$ that can be easily shown to satisfy the ODE* (1.2)*. We note that the ODE function f is continuous, and thus existence of the solution is guaranteed mathematically. However, at the origin, the derivative of f approaches infinity. It turns out that this is the reason which causes the non-uniqueness of the solution.*

As we are only interested in systems with well-defined and deterministic solutions, we would like to formulate only ODE with unique solutions. Here helps the following theorem by Charles Émile Picard (1890) and Ernst Leonard Lindelöf (1894).

Theorem 1 (Existence and Uniqueness of IVP) *Regard the initial value problem* (1.2) *with $x(t_{\text{init}}) = x_{\text{init}}$, and assume that $f : \mathbb{R}^{n_x} \times [t_{\text{init}}, t_{\text{fin}}] \to \mathbb{R}^{n_x}$ is continuous with respect to x and t. Furthermore, assume that f is Lipschitz continuous with respect to x, i.e., that there exists a constant L such that for all $x, y \in \mathbb{R}^{n_x}$ and all $t \in [t_{\text{init}}, t_{\text{fin}}]$*

$$\|f(x,t) - f(y,t)\| \leq L\|x - y\|. \tag{1.3}$$

Then there exists a unique solution $x : [t_{\text{init}}, t_{\text{fin}}] \to \mathbb{R}^{n_x}$ of the IVP.

Lipschitz continuity of f with respect to x is not easy to check. It is much easier to verify if a function is differentiable. It is therefore a helpful fact that every function f that is differentiable with respect to x is also locally Lipschitz continuous, and one can prove the following corollary to the Theorem of Picard-Lindelöf.

Corollary 1 (Local Existence and Uniqueness) *Regard the same initial value problem as in Theorem 1, but instead of global Lipschitz continuity, assume that f is continuously differentiable with respect to x for all $t \in [t_{\text{init}}, t_{\text{fin}}]$. Then there exists a possibly shortened, but non-empty interval $[t_{\text{init}}, t'_{\text{fin}}]$ with $t'_{\text{fin}} \in (t_{\text{init}}, t_{\text{fin}}]$ on which the IVP has a unique solution.*

Note that for nonlinear continuous time systems – in contrast to discrete time systems – it is very easily possibly even with innocently looking and smooth functions f to obtain an "explosion", i.e., a solution that tends to infinity for finite times.

Example 2 (Explosion of an ODE) *Regard the scalar example $f(x) = x^2$ with $t_{\text{init}} = 0$ and $x_{\text{init}} = 1$, and let us regard the interval $[t_{\text{init}}, t_{\text{fin}}]$ with $t_{\text{fin}} = 10$. The IVP has the explicit solution $x(t) = 1/(1-t)$, which is only defined on the half open interval $[0, 1)$, because it tends to infinity for $t \to 1$. Thus, we need to choose some $t'_{\text{fin}} < 1$ in order to have a unique and finite solution to the IVP on the shortened interval $[t_{\text{init}}, t'_{\text{fin}}]$. The existence of this local solution is guaranteed by the above corollary. Note that the explosion in finite time is due to the fact that the function f is not globally Lipschitz continuous, so Theorem 1 is not applicable.*

Discontinuities with Respect to Time

It is important to note that the above theorem and corollary can be extended to the case that there are finitely many discontinuities of f with respect to t. In this case the ODE solution can only be defined on each of the continuous time intervals separately, while the derivative of x is not defined at the time points at which the discontinuities of f

occur, at least not in the strong sense. But the transition from one interval to the next can be determined by continuity of the state trajectory, i.e. we require that the end state of one continuous initial value problem is the starting value of the next one.

The fact that unique solutions still exist in the case of discontinuities is important because, first, many optimal control problems have discontinuous control trajectories $u(t)$ in their solution, and, second, many algorithms discretize the controls as piecewise constant functions which have jumps at the interval boundaries. Fortunately, this does not cause difficulties for existence and uniqueness of the IVPs.

Linear Time Invariant (LTI) Systems

A special class of tremendous importance are the linear time invariant (LTI) systems. These are described by an ODE of the form

$$\dot{x} \quad = \quad Ax + Bu \tag{1.4}$$

with fixed matrices $A \in \mathbb{R}^{n_x \times n_x}$ and $B \in \mathbb{R}^{n_x \times n_u}$. LTI systems are one of the principal interests in the field of automatic control and a vast literature exists on LTI systems. Note that the function $f(x, u) = Ax + Bu$ is Lipschitz continuous with respect to x with Lipschitz constant $L = \|A\|$, so that the global solution to any initial value problem with a piecewise continuous control input can be guaranteed.

Many important notions such as *controllability* or *stabilizability*, and computational results such as the *step response* or *frequency response function* can be defined in terms of the matrices A and B alone. Note that in the field of linear system analysis and control, usually also output equations $y = Cx$ are present, where the outputs y may be the only physically relevant quantities. Only the linear operator from u to y - the input-output-behaviour - is of interest, while the state x is just an intermediate quantity. In that context, the states are not even unique, because different state space realizations of the same input-output behavior exist. In this lecture, however, we are not interested in input-outputs-behaviours, but assume that the state is the principal quantity of interest. Output equations are not part of the models in this lecture. If one wants to make the connection to the LTI literature, one might set $C = \mathbb{I}$.

Zero Order Hold and Solution Map

In the age of digital control, the inputs u are often generated by a computer and implemented at the physical system as piecewise constant between two sampling instants. This is called *zero order hold*. The grid size is typically constant, say of fixed length $\Delta t > 0$, so that the sampling instants are given by $t_k = k \cdot \Delta t$. If our original model is a differentiable ODE model, but we have piecewise constant control inputs with fixed values $u(t) = u_k$ wtih $u_k \in \mathbb{R}^{n_u}$ on each interval $t \in [t_k, t_{k+1}]$, we might want to regard the transition from the state $x(t_k)$ to the state $x(t_{k+1})$ as a discrete time system. This is indeed possible, as the ODE solution exists and is unique on the interval $[t_k, t_{k+1}]$ for each initial value $x(t_k) = x_{\text{init}}$.

If the original ODE system is time-invariant, it is enough to regard one initial value problem with constant control $u(t) = u_{\text{const}}$

$$\dot{x}(t) = f(x(t), u_{\text{const}}), \quad t \in [0, \Delta t], \quad \text{with} \quad x(0) = x_{\text{init}}. \tag{1.5}$$

The unique solution $x : [0, \Delta t] \to \mathbb{R}^{n_x}$ to this problem is a function of both, the initial value x_{init} and the control u_{const}, so we might denote the solution by

$$x(t; x_{\text{init}}, u_{\text{const}}), \quad \text{for} \quad t \in [0, \Delta t]. \tag{1.6}$$

This map from $(x_{\text{init}}, u_{\text{const}})$ to the state trajectory is called the *solution map*. The final value of this short trajectory piece, $x(\Delta t; x_{\text{init}}, u_{\text{const}})$, is of major interest, as it is the point where the next sampling interval starts. We might define the transition function $f_{\text{dis}} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ by $f_{\text{dis}}(x_{\text{init}}, u_{\text{const}}) = x(\Delta t; x_{\text{init}}, u_{\text{const}})$. This function allows us to define a discrete time system that uniquely describes the evolution of the system state at the sampling instants t_k:

$$x(t_{k+1}) = f_{\text{dis}}(x(t_k), u_k). \tag{1.7}$$

Solution Map of Linear Time Invariant Systems

Let us regard a simple and important example: for linear continuous time systems

$$\dot{x} = Ax + Bu$$

with initial value x_{init} at $t_{\text{init}} = 0$, and constant control input u_{const}, the solution map $x(t; x_{\text{init}}, u_{\text{const}})$ is explicitly given as

$$x(t; x_{\text{init}}, u_{\text{const}}) = \exp(At)x_{\text{init}} + \int_0^t \exp(A(t-\tau))Bu_{\text{const}}\mathrm{d}\tau,$$

where $\exp(A)$ is the matrix exponential. It is interesting to note that this map is well defined for all times $t \in \mathbb{R}$, as linear systems cannot explode. The corresponding discrete time system with sampling time Δt is again a linear time invariant system, and is given by

$$f_{\text{dis}}(x_k, u_k) = A_{\text{dis}}x_k + B_{\text{dis}}u_k \tag{1.8}$$

with

$$A_{\text{dis}} = \exp(A\Delta t) \quad \text{and} \quad B_{\text{dis}} = \int_0^{\Delta t} \exp(A(\Delta t - \tau))B\mathrm{d}\tau. \tag{1.9}$$

Sensitivities

In the context of optimal control, derivatives of the dynamic system simulation are needed for nearly all numerical algorithms. Following Theorem 1 and Corollary 1 we know that the solution map to the IVP (1.5) exists on an interval $[0, \Delta t]$ and is unique under mild conditions even for general nonlinear systems. But is it also differentiable with respect to the initial value and control input?

In order to discuss the issue of derivatives, which in the dynamic system context are often called *sensitivities*, let us first ask what happens if we call the solution map with different inputs. For small perturbations of the values $(x_{\text{init}}, u_{\text{const}})$, we still have a unique solution $x(t; x_{\text{init}}, u_{\text{const}})$ on the whole interval $t \in [0, \Delta t]$. Let us restrict ourselves to a neighborhood $\mathcal{N}$ of fixed values $(x_{\text{init}}, u_{\text{const}})$. For each fixed $t \in [0, \Delta t]$, we can now regard the well defined and unique solution map $x(t; \cdot) : \mathcal{N} \to \mathbb{R}^{n_x},\ (x_{\text{init}}, u_{\text{const}}) \mapsto x(t; x_{\text{init}}, u_{\text{const}})$. A natural question to ask is if this map is differentiable. Fortunately, it is possible to show that if f is m-times continuously differentiable with respect to both x and u, then the solution map $x(t; \cdot)$, for each $t \in [0, \Delta t]$, is also m-times continuously differentiable with respect to $(x_{\text{init}}, u_{\text{const}})$.

In the general nonlinear case, the solution map $x(t; x_{\text{init}}, u_{\text{const}})$ can only be generated by a numerical simulation routine. The computation of derivatives of this numerically generated map is a delicate issue that we discuss in detail in a follow up course on numerical optimal control. To mention already the main difficulty, note that most numerical integration routines are adaptive, i.e., might choose to do different numbers of integration steps for different IVPs. This renders the numerical approximation of the map $x(t; x_{\text{init}}, u_{\text{const}})$ typically non-differentiable in the inputs $x_{\text{init}}, u_{\text{const}}$. Thus, multiple calls of a black-box integrator and application of finite differences might result in very wrong derivative approximations.

Numerical Integration Methods

A numerical simulation routine that approximates the solution map is often called an *integrator*. A simple but very crude way to generate an approximation for $x(t; x_{\text{init}}, u_{\text{const}})$ for $t \in [0, \Delta t]$ is to perform a linear extrapolation based on the time derivative $\dot{x} = f(x, u)$ at the initial time point:

$$\tilde{x}(t; x_{\text{init}}, u_{\text{const}}) = x_{\text{init}} + t f(x_{\text{init}}, u_{\text{const}}), \quad t \in [0, \Delta t]. \tag{1.10}$$

This is called one *Euler integration step*. For very small Δt, this approximation becomes very good. In fact, the error $\tilde{x}(\Delta t; x_{\text{init}}, u_{\text{const}}) - x(\Delta t; x_{\text{init}}, u_{\text{const}})$ is of second order in Δt. This motivated Leonhard Euler to perform several steps of smaller size, and propose what is now called the *Euler integration method*. We subdivide the interval $[0, \Delta t]$ into M subintervals each of length $h = \Delta t / M$, and perform M such linear extrapolation steps consecutively, starting at $\tilde{x}_0 = x_{\text{init}}$:

$$\tilde{x}_{j+1} = \tilde{x}_j + h f(\tilde{x}_j, u_{\text{const}}), \quad j = 0, \ldots, M - 1. \tag{1.11}$$

It can be proven that the Euler integration method is *stable*, i.e. that the propagation of local errors is bounded with a constant that is independent of the step size h. Therefore, the approximation becomes better and better when we decrease the step size h: since the *consistency* error in each step is of order h^2, and the total number of steps is of order $\Delta t / h$, the accumulated error in the final step is of order $h \Delta t$. As this is linear in the step size h, we say that the Euler method has the *order one*. Taking more steps is more accurate, but also needs more computation time. One measure for the computational effort of an integration method is the number of evaluations of f, which for the Euler method grows linearly with the desired accuracy.

In practice, the Euler integrator is rarely competitive, because other methods exist that deliver the desired accuracy levels at much lower computational cost. We discuss several numerical simulation methods later, but present here already one of the most widespread integrators, the ***Runge-Kutta Method of Order Four***, which we will often abbreviate as ***RK4***. One step of the RK4 method needs four evaluations of f and stores the results in four intermediate quantities $k_i \in \mathbb{R}^{n_x}$, $i = 1, \ldots, 4$. Like the Euler integration method, the RK4 also generates a sequence of values $\tilde{x}_j$, $j = 0, \ldots, M$, with $\tilde{x}_0 = x_{\text{init}}$. At $\tilde{x}_j$, and using the constant control input u_{const}, one step of the RK4 method proceeds as follows:

$$k_1 = f(\tilde{x}_j, u_{\text{const}}) \tag{1.12a}$$

$$k_2 = f(\tilde{x}_j + \frac{h}{2} k_1, u_{\text{const}}) \tag{1.12b}$$

$$k_3 = f(\tilde{x}_j + \frac{h}{2} k_2, u_{\text{const}}) \tag{1.12c}$$

$$k_4 = f(\tilde{x}_j + h\, k_3, u_{\text{const}}) \tag{1.12d}$$

$$\tilde{x}_{j+1} = \tilde{x}_j + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{1.12e}$$

One step of RK4 is thus as expensive as four steps of the Euler method. But it can be shown that the accuracy of the final approximation $\tilde{x}_M$ is of order $h^4 \Delta t$. In practice, this means that the RK4 method usually needs tremendously fewer function evaluations than the Euler method to obtain the same accuracy level.

From here on, and throughout the major part of the lecture, we will leave the field of continuous time systems, and directly assume that we control a discrete time system $x_{k+1} = f_{\text{dis}}(x_k, u_k)$. Let us keep in mind, however, that the transition map $f_{\text{dis}}(x_k, u_k)$ is usually not given as an explicit expression but can instead be a relatively involved computer code with several intermediate quantities. In the exercises of this lecture, we will usually discretize the occuring ODE systems by using only one Euler or RK4 step per control interval, i.e. use $M = 1$ and $h = \Delta t$. The RK4 step often gives already a sufficient approximation at relatively low cost.

1.3 Discrete Time Systems

Let us now discuss in more detail the discrete time systems that are at the basis of the control problems in the first part of this lecture. In the general time-variant case, these systems are characterized by the dynamics

$$x_{k+1} = f_k(x_k, u_k), \quad k = 0, 1, \ldots, N-1 \tag{1.13}$$

on a time horizon of length N, with N control input vectors $u_0, \ldots, u_{N-1} \in \mathbb{R}^{n_u}$ and $(N+1)$ state vectors $x_0, \ldots, x_N \in \mathbb{R}^{n_x}$.

If we know the initial state x_0 and the controls $u_0, \ldots, u_{N-1}$ we could recursively call the functions f_k in order to obtain all other states, $x_1, \ldots, x_N$. We call this a *forward simulation* of the system dynamics.

Definition 1 (Forward simulation) *The* forward simulation *is the map*

$$\begin{array}{rrcl} f_{\text{sim}}: & \mathbb{R}^{n_x+Nn_u} & \to & \mathbb{R}^{(N+1)n_x} \\ & (x_0; u_0, u_1, \dots, u_{N-1}) & \mapsto & (x_0, x_1, x_2, \dots, x_N) \end{array} \tag{1.14}$$

that is defined by solving Equation (1.13) *recursively for all* $k = 0, 1, \dots, N-1$.

The inputs of the forward simulation routine are the initial value x_0 and the controls u_k for $k = 0, \dots, N-1$. In many practical problems we can only choose the controls while the initial value is fixed. Though this is a very natural assumption, it is not the only possible one. In optimization, we might have very different requirements: We might, for example, have a free initial value that we want to choose in an optimal way. Or we might have both a fixed initial state and a fixed terminal state that we want to reach. We might also look for periodic sequences with $x_0 = x_N$, but do not know x_0 beforehand. All these desires on the initial and the terminal state can be expressed by suitable constraints. For the purpose of this manuscript it is important to note that the fundamental equation that is characterizing a dynamic optimization problem are the system dynamics stated in Equation (1.13), but no initial value constraint, which is optional.

Linear Time Invariant (LTI) Systems

As discussed already for the continuous time case, linear time invariant (LTI) systems are not only one of the simplest possible dynamic system classes, but also have a rich and beautiful history. In the discrete time case, they are determined by the system equation

$$x_{k+1} = Ax_k + Bu_k, \quad k = 0, 1, \dots, N-1. \tag{1.15}$$

with fixed matrices $A \in \mathbb{R}^{n_x \times n_x}$ and $B \in \mathbb{R}^{n_x \times n_u}$. An LTI system is stable if all eigenvalues of the matrix A are in the unit disc of the complex plane, i.e. have a modulus smaller or equal to one, and *asymptotically stable* if all moduli are strictly smaller than one. It is easy to show that the forward simulation map for an LTI system on a horizon with length N is given by

$$f_{\text{sim}}(x_0; u_0, \dots, u_{N-1}) = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} x_0 \\ Ax_0 + Bu_0 \\ A^2x_0 + ABu_0 + Bu_1 \\ \vdots \\ A^N x_0 + \sum_{k=0}^{N-1} A^{N-1-k} Bu_k \end{bmatrix}$$

In order to check controllability, due to linearity, one might ask the question if after N steps any terminal state x_N can be reached from $x_0 = 0$ by a suitable choice of control inputs. Because of

$$x_N = \underbrace{\begin{bmatrix} A^{N-1}B & A^{N-2}B & \cdots & B \end{bmatrix}}_{=\mathcal{C}_N} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}$$

this is possible if and only if the matrix $\mathcal{C}_N \in \mathbb{R}^{n_x \times N n_u}$ has the rank n_x. Increasing N can only increase the rank, but one can show that the maximum possible rank is already reached for $N = n_x$, so it is enough to check if the so called *controllability matrix* $\mathcal{C}_{n_x}$ has the rank n_x.

Affine Systems and Linearizations along Trajectories

An important generalization of linear systems are affine time-varying systems of the form

$$x_{k+1} \quad = \quad A_k x_k + B_k u_k + c_k, \quad k = 0, 1, \ldots, N-1. \tag{1.16}$$

These often appear as linearizations of nonlinear dynamic systems along a given reference trajectory. To see this, let us regard a nonlinear dynamic system and some given reference trajectory values $\bar{x}_0, \ldots, \bar{x}_{N-1}$ as well as $\bar{u}_0, \ldots, \bar{u}_{N-1}$. Then the Taylor expansion of each function f_k at the reference value $(\bar{x}_k, \bar{u}_k)$ is given by

$$(x_{k+1} - \bar{x}_{k+1}) \approx \frac{\partial f_k}{\partial x}(\bar{x}_k, \bar{u}_k)(x_k - \bar{x}_k) + \frac{\partial f_k}{\partial u}(\bar{x}_k, \bar{u}_k)(u_k - \bar{u}_k) + (f_k(\bar{x}_k, \bar{u}_k) - \bar{x}_{k+1})$$

thus resulting in affine time-varying dynamics of the form (1.16). Note that even for a time-invariant nonlinear system the linearized dynamics becomes time-variant due to the different linearization points on the reference trajectory.

It is an important fact that the forward simulation map of an affine system (1.16) is again an affine function of the initial value and the controls. More specifically, this affine map is for any $N \in \mathbb{N}$ given by:

$$x_N = (A_{N-1} \cdots A_0)\, x_0 + \sum_{k=0}^{N-1} \left(\Pi_{j=k+1}^{N-1} A_j \right) (B_k u_k + c_k)\,.$$

1.4 Optimization Problem Classes

Mathematical optimization refers to finding the best, or *optimal* solution among a set of possible decisions, where optimality is defined with the help of an *objective function*. Some solution candidates are *feasible*, others not, and it is assumed that *feasibility* of a solution candidate can be checked by evaluation of some *constraint functions* that need for example be equal to zero. Like the field of dynamic systems, the field of mathematical optimization comprises many different problem classes, which we will briefly try to classify in this section.

Historically, optimization has been identified with programming, where a program was understood as a deterministic plan, e.g., in logistics. For this reason, many of the optimization problem classes have been given names that contain the words *program* or *programming*. In this script we will often use these names and their abbreviations, because they are still widely used. Thus, we use e.g. the term *linear program (LP)* as a synonym for a *linear optimization problem*. It is interesting to note that the major society for mathematical optimization, which had for decades the name *Mathematical*

Programming Society (MPS), changed its name in 2011 to *Mathematical Optimization Society (MOS)*, while it decided not to change the name of its major journal, that still is called *Mathematical Programming*. In this script we chose a similarly pragmatic approach to the naming conventions.

Finite vs Infinite Dimensional Optimization

An important divididing line in the field of optimization regards the dimension of the space in which the decision variable, say x, is chosen. If x can be represented by finitely many numbers, e.g. $x \in \mathbb{R}^n$ with some $n \in \mathbb{N}$, we speak of a *finite dimensional optimization problem*, otherwise, of an *infinite dimensional optimization problem*. The second might also be referred to as *optimization in function spaces*. Discrete time optimal control problems fall into the first, continuous time optimal control problems into the second class.

Besides the dimension of the decision variable, also the dimension of the constraint functions can be finite or infinite. If an infinite number of inequality constraints is present while the decision variable is finite dimensional, one speaks of a *semi-infinite optimization problem*. This class naturally arises in the context of *robust optimization*, where one wants to find the best choice of the decision variable that satisfies the constraints for all possible values of an unknown but bounded disturbance.

Continuous vs Integer Optimization

A second dividing line concerns the type of decision variables. These can be either *continuous*, like for example real valued vectors $x \in \mathbb{R}^n$, or any other elements of a smooth manifold. On the other hand, the decision variable can be *discrete*, or *integer valued*, i.e. we have $z \in \mathbb{Z}^n$, or, when a set of binary choices has to be made, $z \in \{0,1\}^n$. In this case one often also speaks of *combinatorial optimization*. If an optimization problem has both, continuous and integer variables, it is called a *mixed-integer optimization problem*.

An important class of continuous optimization problems are the so called *nonlinear programs (NLP)*. They can be stated in the form

$$
\begin{aligned}
\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & f(x) && \text{(1.17a)}\\
\text{subject to} \quad & g(x) = 0, && \text{(1.17b)}\\
& h(x) \leq 0, && \text{(1.17c)}
\end{aligned}
$$

where $f : \mathbb{R}^n \rightarrow \mathbb{R}$, $g : \mathbb{R}^n \rightarrow \mathbb{R}^{n_g}$, and $h : \mathbb{R}^n \rightarrow \mathbb{R}^{n_h}$ are assumed to be at least once continuously differentiable. Note that we use function and variable names such as f and x with a very different meaning than before in the context of dynamic systems. In the first part of the lecture we discuss algorithms to solve this kind of optimization problems, and the discrete time optimal control problems treated in this lecture can also be regarded as a specially structured form of NLPs. Two important subclasses of NLPs are the *linear programs (LP)*, which have affine problem functions f, g, h, and the *quadratic programs (QP)*, which have affine constraint functions g, h and a more general linear quadratic objective $f(x) = c^T x + \frac{1}{2} x^T H x$ with a symmetric matrix $H \in \mathbb{R}^{n \times n}$.

A large class of mixed-integer optimization problems are the so called *mixed integer nonlinear programs (MINLP)*, which can be stated as

$$\begin{aligned}
\underset{\substack{x \in \mathbb{R}^n \\ z \in \mathbb{Z}^m}}{\text{minimize}} \quad & f(x, z) && \text{(1.18a)} \\
\text{subject to} \quad & g(x, z) = 0, && \text{(1.18b)} \\
& h(x, z) \leq 0. && \text{(1.18c)}
\end{aligned}$$

Among the MINLPs, an important special case arises if the problem functions f, g, h are affine in both variables, x and z, which is called a *mixed integer linear program (MILP)*. If the objective is allowed to be linear quadratic, one speaks of a *mixed integer quadratic program (MIQP)*. If in an MILP only integer variables are present, one usually just calls it an *integer program (IP)*. The field of (linear) integer programming is huge and has powerful algorithms available. Most problems in logistics fall into this class, a famous example being the *travelling salesman problem*, which concerns the shortest closed path that one can travel through a given number of towns, visiting each town exactly once.

An interesting class of mixed-integer optimization problems arises in the context of optimal control of hybrid dynamic systems, which in the discrete time case can be regarded a special case of MINLP. In continuous time, we enter the field of infinite dimensional mixed-integer optimization, often also called *Mixed-integer optimal control problems (MIOCP)*.

Convex vs Nonconvex Optimization

Arguably the most important dividing line in the world of optimization is between convex and nonconvex optimization problems. Convex optimization problems are a subclass of the continuous optimization problems and arise if the objective function is a convex function and the set of feasible points a convex set. In this case one can show that any *local solution*, i.e. values for the decision variables that lead to the best possible objective value in a neighborhood, is also a *global solution*, i.e. has the best possible objective value among all feasible points. Practically very important is the fact that convexity of a function or a set can be checked just by checking convexity of its building blocks and if they are constructed in a way that preserves convexity.

Several important subclasses of NLPs are convex, such as LPs. Also QPs are convex if they have a convex objective f. Another example are *Quadratically Constrained Quadratic Programs (QCQP)* which have quadratic inequalities and whose feasible set is the intersection of ellipsoids. Some other optimization problems are convex but do not form part of the NLP family. Two widely used classes are *second-order cone programs (SOCP)* and *semi-definite programs (SDP)* which have linear objective functions but more involved convex feasible sets: for SOCP, it is the set of vectors which have one component that is larger than the Euclidean norm of all the other components and which it is called the *second order cone*, and for SDP it is the set of symmetric matrices that are positive semi-definite, i.e. have all eigenvalues larger than zero. SDPs are often used when designing linear feedback control laws. Also infinite dimensional optimization problems such as optimal control problems in continuous time can be convex under fortunate circumstances.

In this context, it is interesting to note that a sufficient condition for convexity of an optimal control problem is that the underlying dynamic system is linear and that the objective and constraints are convex in controls and states. On the other hand, optimal control problems with underlying nonlinear dynamic systems, which are the focus of this lecture, are usually nonconvex.

Optimization problems with integer variables can never be convex due to the non-convexity of the set of integers. However, it is of great algorithmic advantage if mixed-integer problems have a convex substructure in the sense that convex problems arise when the integer variables are allowed to also take real values. These so called *convex relaxations* are at the basis of nearly all competitive algorithms for mixed-integer optimization. For example, linear integer programs can be solved very efficiently because their convex relaxations are just linear programs, which are convex and can be solved very efficiently.

1.5 Overview and Notation

The chapters of these lecture notes can roughly be divided into six major areas.

- Introduction
- Optimization Background
- Discrete Time Optimal Control
- Continuous Time Optimal Control
- Model Predictive Control and Moving Horizon Estimation

Notation

Within this lecture we use $\mathbb{R}$ for the set of real numbers, $\mathbb{R}_+$ for the non-negative ones and $\mathbb{R}_{++}$ for the positive ones, $\mathbb{Z}$ for the set of integers, and $\mathbb{N}$ for the set of natural numbers including zero, i.e. we identify $\mathbb{N} = \mathbb{Z}_+$. The set of real-valued vectors of dimension n is denoted by $\mathbb{R}^n$, and $\mathbb{R}^{n\times m}$ denotes the set of matrices with n rows and m columns. By default, all vectors are assumed to be column vectors, i.e. we identify $\mathbb{R}^n = \mathbb{R}^{n\times 1}$. We usually use square brackets when presenting vectors and matrices elementwise. Because will often deal with concatenations of several vectors, say $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$, yielding a vector in $\mathbb{R}^{n+m}$, we abbreviate this concatenation sometimes as (x, y) in the text, instead of the correct but more clumsy equivalent notations $[x^\top, y^\top]^\top$ or

$$\begin{bmatrix} x \\ y \end{bmatrix}.$$

Square and round brackets are also used in a very different context, namely for intervals in $\mathbb{R}$, where for two real numbers $a < b$ the expression $[a, b] \subset \mathbb{R}$ denotes the closed interval containing both boundaries a and b, while an open boundary is denoted by a round bracket, e.g. (a, b) denotes the open interval and $[a, b)$ the half open interval containing a but not b.

When dealing with norms of vectors $x \in \mathbb{R}^n$, we denote by $\|x\|$ an arbitrary norm, and by $\|x\|_2$ the Euclidean norm, i.e. we have $\|x\|_2^2 = x^\top x$. We denote a weighted Euclidean norm with a positive definite weighting matrix $Q \in \mathbb{R}^{n\times n}$ by $\|x\|_Q$, i.e. we have $\|x\|_Q^2 = x^\top Qx$. The L_1 and L_∞ norms are defined by $\|x\|_1 = \sum_{i=1}^n |x_i|$ and $\|x\|_\infty = \max\{|x_1|, \dots, |x_n|\}$. Matrix norms are the induced operator norms, if not stated otherwise, and the Frobenius norm $\|A\|_\mathrm{F}$ of a matrix $A \in \mathbb{R}^{n\times m}$ is defined by $\|A\|_\mathrm{F}^2 = \mathrm{trace}(AA^\top) = \sum_{i=1}^n \sum_{j=1}^m A_{ij}A_{ij}$.

When we deal with derivatives of functions f with several real inputs and several real outputs, i.e. functions $f : \mathbb{R}^n \to \mathbb{R}^m, x \mapsto f(x)$, we define the Jacobian matrix $\frac{\partial f}{\partial x}(x)$ as a matrix in $\mathbb{R}^{m\times n}$, following standard conventions. For scalar functions with $m = 1$, we denote the gradient vector as $\nabla f(x) \in \mathbb{R}^n$, a column vector, also following standard conventions. Slightly less standard, we generalize the gradient symbol to all functions $f : \mathbb{R}^n \to \mathbb{R}^m$ even with $m > 1$, i.e. we generally define in this lecture

$$\nabla f(x) = \frac{\partial f}{\partial x}(x)^\top \in \mathbb{R}^{n\times m}.$$

Using this notation, the first order Taylor series is e.g. written as

$$f(x) = f(\bar{x}) + \nabla f(\bar{x})^\top (x - \bar{x})) + o(\|x - \bar{x}\|)$$

The second derivative, or Hessian matrix will only be defined for scalar functions $f : \mathbb{R}^n \to \mathbb{R}$ and be denoted by $\nabla^2 f(x)$.

For square symmetric matrices of dimension n we sometimes use the symbol $\mathbb{S}_n$, i.e. $\mathbb{S}_n = \{A \in \mathbb{R}^{n\times n} | A = A^\top\}$. For any symmetric matrix $A \in \mathbb{S}_n$ we write $A \succcurlyeq 0$ if it is a positive semi-definite matrix, i.e. all its eigenvalues are larger or equal to zero, and $A \succ 0$ if it is positive definite, i.e. all its eigenvalues are positive. This notation is also used for *matrix inequalities* that allow us to compare two symmetric matrices $A, B \in \mathbb{S}_n$, where we define for example $A \succcurlyeq B$ by $A - B \succcurlyeq 0$.

When using logical symbols, $A \Rightarrow B$ is used when a proposition A implies a proposition B. In words the same is expressed by "If A then B". We write $A \Leftrightarrow B$ for "A if and only if B", and we sometimes shorten this to "A iff B", with a double "f", following standard practice.