Lecture Notes on Modelling and System Identification

Moritz Diehl

February 18, 2019

Contents

[1] not covered in the lecture
[2] not covered in the lecture

Preface to the Manuscript

This lecture manuscript is written to accompany a lecture course on "Modelling and System Identification" given since winter 2013 at the University of Freiburg. This script was written by the author with tremendous help by Jesus Lago Garcia during a student job contract, who wrote some sections based on his notes taken during the lecture in 2014 and some based on the lecture slides. The script also contains some input by Honghu Xue and from Robin Verscheuren and Jonas Koenemann, who helped to proofread the text. Some parts and figures are based on a previous draft manuscript from the same course a year earlier by Benjamin Völker, and on the lecture notes from a course on numerical optimization the author has previously taught at the University of Leuven. The script is far from perfect and surely contains some errors, which can be reported to `moritz.diehl@imtek.uni-freiburg.de`.

Aim of the manuscript is that it shall serve to the students of the course as a reference for self study and exam preparation, together with the video lectures and the slides. In addition, we strongly recommend to consult the book by Lennart Ljung [Lju99] as well as the script by Johan Schoukens [Sch13].

Freiburg, November 2014 -October 2015 *Moritz Diehl*

Part I

Introduction and Foundations

Chapter 1

Introduction

The lecture course on Modelling and System Identification (MSI) has as its aim to enable the students to create models that help to predict the behaviour of systems. Here, we are particularly interested in dynamic system models, i.e. systems evolving in time. With good system models, one can not only predict the future (like in weather forecasting), but also control or optimize the behaviour of a technical system, via feedback control or smart input design. Having a good model gives us access to powerful engineering tools. This course focuses on the process to obtain such models. It builds on knowledge from three fields: Systems Theory, Statistics, and Optimization. We will recall necessary concepts from these three fields on demand during the course. For a motivation for the importance of statistics in system identification, we first look at a very simple example taken from the excellent lecture notes of J. Schoukens [Sch13]. The course will then first focus on identification methods for static models and their statistical properties, and review the necessary concepts from statistics and optimization where needed. Later, we will look at different ways to model dynamic systems and how to identify them. For a much more detailed and complete treatment of system identification, we refer to the textbook by L. Ljung [Lju99].

1.1 Mathematical 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 $\|x\|_\infty = \max\{|x_1|, \ldots, |x_n|\}$. 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.

A remark is in order about how we refer to the components of a vector, say $x \in \mathbb{R}^n$. Here, we sometimes use the standard convention, i.e. refer to the components by $x_1, x_2, \ldots, x_n$, i.e. have $x = [x_1, x_2, \ldots, x_n]^\top$. But sometimes, in particular if the components of the vector represent a series of measurements or a sequence of other quantities, we use round brackets to refer to its components (as in the script [Sch13] and the book [Lju99]), i.e. we write $x(1), x(2), \ldots, x(n)$. Sometimes we want to express that the length of the whole vector increases with time, and we give an index to the whole vector, i.e. we write $x_n \in \mathbb{R}^n$, which conflicts with the notation for a single component. In order to avoid confusion with the single components, we usually indicate the dimension of the respective vectors. In particular for the vector that contains all measurements taken until time point N, we use the notation $y(1), \ldots, y(N)$ for the components, and $y_N = [y(1), \ldots, y(N)]^\top$ for the whole sequence, to be consistent with the notation in most of the system identification literature.

We usually use the "hat" above some estimated quantities, e.g. $\hat{R}$ in order to denote that they are estimates of R rather than the true value, which we sometimes denote by the same variable name but an index zero, i.e. R_0 is the true value of R. If the estimate depends on a series of samples, we often give it also an index. e.g. $\hat{R}(N)$ is the estimate of R given the first N measurements.

1.2 A Simple Example: Resistance Estimation

To motivate the need to think about statistical properties, we start with a simple example taken from [Sch13]. Aim is to estimate the resistance of a resistor based on measurements of current and voltage, see Fig. 1.1. In order to increase the accuracy of our estimate, we repeat the same experiment many times, where we measure each time current $i(k)$ and voltage $u(k)$. Due to measurement noise, all these values will be different, see Fig. 1.2.

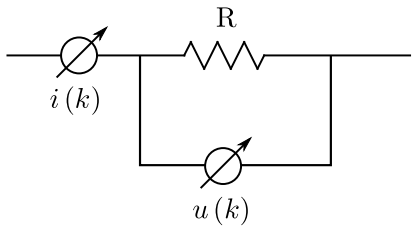


Figure 1.1: Resistance estimation example with resistor R , current measurements $i(k)$, and voltage measurements $u(k)$ for $k = 1, 2, ..., N$. [Sch13]

Another way to look at the same series of measurements would be the voltage-current diagram in Fig. 1.3. In a perfect world without noise, we would expect all measurements to be at the same point, but in reality they are spread out.

We know by Ohm's law that $u = Ri$. For a single measurement, $u(1)$ and $i(1)$, we would clearly have $\hat{R}(1) = \frac{u(1)}{i(1)}$. We would expect that taking many measurements should help us to reduce the effect of noise and improve our estimates. But given a series of measurements $u(k)$ and $i(k)$, how should we compute an estimate $\hat{R}(N)$ for the unknown resistance? Let us look at three different approaches, which we will denote by names that are partly ad-hoc and partly motivated by later considerations.

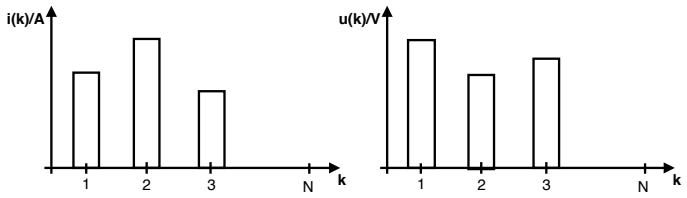


Figure 1.2: The measurements are a time series of discrete noisy values.

Figure 1.3: The same measurements in a voltage-current diagram.

1.2.1 Simple Approach

A first simple approach is to compute the average of the resistance estimates one would obtain for each experiment.

$$\hat{R}_{\mathrm{SA}}(N) = \frac{1}{N} \cdot \sum_{k=1}^{N} \frac{u(k)}{i(k)} \tag{1.1}$$

We can take a long sequence of measurements, and apply this formula each time. The resulting sequence of numbers $\hat{R}_{\mathrm{SA}}(N)$ is shown as a function of N in the curve in Fig. 1.4 at the top. It seems to converge for large N. But does it converge to the right value?

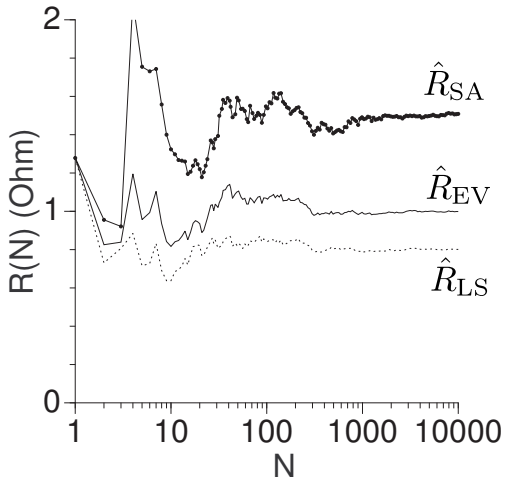


Figure 1.4: Different estimations of R for increasing sample size N. At least two of the estimators are wrong. But which estimator gives us the true value of R when N goes to infinity ?

1.2.2 Error-in-Variables

A different approach to reduce the noise would be to first take the average of all voltage measurements and of all current measurements, and then divide the two averages. This gives the following formula that we call "error-in-variables" because it will correspond to an estimator with this name that we investigate later.

$$\hat{R}_{\mathrm{EV}}(N) = \frac{\frac{1}{N}\sum_{k=1}^{N} u(k)}{\frac{1}{N}\sum_{k=1}^{N} i(k)} \tag{1.2}$$

The corresponding sequence of numbers $\hat{R}_{\mathrm{EV}}(N)$ is shown in Fig. 1.4 in the middle for the same measurements as before. Unfortunately, the two approaches give different results, and the curve does not converge to the same value as the previous approach.

1.2.3 Least Squares

A third approach uses the method of least squares. Idea is to find R by minimizing the sum of the squared differences between the *model predictions* for the voltage based on the measurement $i(k)$, i.e. $Ri(k)$, and the measurement of the voltage, $u(k)$, see Fig. 1.5. The defining formula is thus given by the following expression.

$$\hat{R}_{\mathrm{LS}}(N) = \arg\min_{R\in\mathbb{R}} \sum_{k=1}^{N} (R\cdot i(k) - u(k))^2 \tag{1.3}$$

$$= \frac{\frac{1}{N}\sum_{k=1}^{N} u(k)\cdot i(k)}{\frac{1}{N}\sum_{k=1}^{N} i(k)^2} \tag{1.4}$$

Here, the second line is obtained by differentiating the function to be optimized, i.e. $f(R) = \sum_{k=1}^{N}(R\cdot i(k) - u(k))^2$, and setting the derivative to zero, $\frac{\partial f}{\partial R}(R^*) = 0$, and resolving this equation to get the minimizer R^* (deriving the formula is left as an easy exercise to the reader). The least squares approach is very powerful, in particular when we deal with more than one unknown parameter, and is very often used in this lecture.

The sequence of numbers $\hat{R}_{\mathrm{LS}}(N)$ resulting from the method of least squares is shown in Fig. 1.4 in the lowest graph. The values seem to converge to a fixed value, but they converge to a different value than the two previous approaches. We still do not know which of the three might be true value of R, unfortunately.

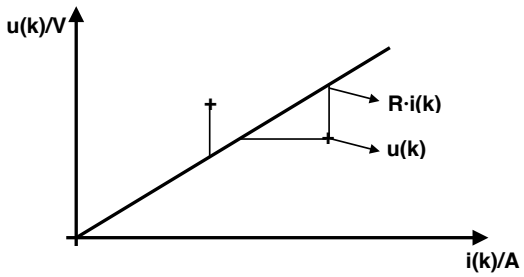


Figure 1.5: Principle behind the Least Squares approach: minimize the squared differences between $u(k)$ and $Ri(k)$.

In order to resolve the problem to decide which of the three estimators gives the correct value (or if they are all three wrong) we need to look into the statistical properties of estimators. These are based on probability theory, which we treat in the next chapter.

Chapter 2

Probability and Statistics in a Nutshell

In this chapter we review concepts from the fields of mathematical statistics and probability that we will need often in this course.

2.1 Random Variables and Probabilities

Random variables are used to describe the possible outcomes of experiments. Random variables are slightly different than the usual mathematical variables, because a random variable does not yet have a value. A random variable X can take values from a given set, typically the real numbers. A specific value $x \in \mathbb{R}$ of the random variable X will typically be denoted by a lower case letter, while the random variable itself will usually be denoted by an upper case letter; we will mostly, but not always stick to this convention. Note that the random variable X itself is not a real number, but just takes values in $\mathbb{R}$. Nevertheless, we sometimes write sloppily $X \in \mathbb{R}$, or $Y \in \mathbb{R}^n$ to quickly indicate that a random variable takes scalars or vectors as values.

The probability that a certain event A occurs is denoted by $P(A)$, and $P(A)$ is a real number in the interval $[0, 1]$. The event A is typically defined by a condition that a random variable can satisfy or not. For example, the probability that the value of a random variable X is larger than a fixed number a is denoted by $P(X > a)$. If the event contains all possible outcomes of the underlying random variable X, its probability is one. If two events A and B are *mutually exclusive*, i.e. are never true at the same time, the probability that one or the other occurs is given by the sum of the two probabilities: $P(A \vee B) = P(A) + P(B)$. Two events can also be *independent* from each other. In that case, the probability that they both occur is given by the product: $P(A \wedge B) = P(A)P(B)$. If this is not the case, the two events are called *dependent*. We will often also write $P(A, B)$ for the joint probabily $P(A \wedge B)$. One can define the *conditional probability* $P(A|B)$ that an event A occurs given that event B has already occured. It is easy to verify the identity

$$P(A|B) = \frac{P(A, B)}{P(B)}.$$

An immediate consequence of this identity is

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}.$$

which is known as *Bayes Theorem* after Thomas Bayes (1701-1761), who investigated how new evidence (event B has occured) can update prior beliefs - the *a-priori* probability $P(A)$ - in order to obtain the *a-posteriori* conditional probability of A given B.

2.2 Scalar Random Variables and Probability Density Functions

For a real valued random variable X, one can define the *Probability Density Function (PDF)* $p_X(x)$ which describes the behaviour of the random variable, and which is a function from the real numbers to the real numbers, i.e. $p_X : \mathbb{R} \to \mathbb{R}$, $x \mapsto p_X(x)$. Note that we use the name of the underlying random variable (here X) as index,

when needed, and that the input argument x of the PDF is just a real number. We will sometimes drop the index when the underlying random variable is clear from the context.

The PDF $p_X(x)$ is related to the probability that X takes values in any interval $[a, b]$ in the following way:

$$P(X \in [a, b]) = \int_a^b p_X(x)\, \mathrm{d}x$$

Conversely, one can define the PDF as

$$p_X(x) = \lim_{\Delta x \to 0} \frac{P(X \in [x, x + \Delta x])}{\Delta x}$$

Two random variables X, Y are independent if the joint PDF $p_{X,Y}(x, y)$ is the product of the individual PDFs, i.e. $p_{X,Y}(x, y) = p_X(x)p_Y(y)$, otherwise they are dependent. The conditional PDF $p_{X|Y}$ of X for given Y is defined by

$$p_{X|Y}(x|y) = \frac{p_{X,Y}(x, y)}{p_Y(y)}.$$

As the above notation can become very cumbersome, we will occasionally also omit the index of the PDF and for example just express the above identity as

$$p(x|y) = \frac{p(x, y)}{p(y)}.$$

2.2.1 Mean and Variance

The *expectation value* or *mean* of a random variable is often denoted by μ_X and computed as $\int_{-\infty}^{\infty} x\ p_X(x)\mathrm{d}x$. More generally, one can compute the expectation of any function $f(X)$ of a random variable, which is by itself a random variable. It is convenient to introduce the expectation operator $\mathbb{E}\{\cdot\}$, which is defined by

$$\mathbb{E}\{f(X)\} := \int_{-\infty}^{\infty} f(x)\ p_X(x)\mathrm{d}x.$$

Due to the linearity of the integral, the expectation operator is also linear. This means that for any affine function $f(X) = a + bX$ with fixed numbers $a, b \in \mathbb{R}$, we have that

$$\mathbb{E}\{a + bX\} = a + b\,\mathbb{E}\{X\}.$$

Note that this is not possible for nonlinear functions $f(X)$, i.e. in general $\mathbb{E}\{f(X)\} \neq f(\mathbb{E}\{X\})$.

The *variance* of a random variable X is a measure of how much the variable varies around the mean and is denoted by σ_X^2. It is defined as

$$\sigma_X^2 := \mathbb{E}\{(X - \mu_X)^2\}.$$

The square root of the variance, $\sigma_X = \sqrt{\sigma_X^2}$, is called the *standard deviation*.

2.2.2 Simple Example: Uniform Distribution

One simple example for a PDF is the *uniform* distribution of on an interval $[a, b] \subset \mathbb{R}$ with $a < b$. Here, the PDF of the random variable X is defined as

$$p_X(x) = \begin{cases} \frac{1}{b-a} & \text{if } x \in [a, b] \\ 0 & \text{else} \end{cases}$$

It is easy to check that $\int_{-\infty}^{\infty} p_X(x)\mathrm{d}x = 1$. The mean is given by

$$\int_{-\infty}^{\infty} p_X(x)\, x\, \mathrm{d}x = \int_a^b \frac{1}{b-a}\, x\, \mathrm{d}x = \frac{b^2 - a^2}{2(b-a)} = \frac{a+b}{2} =: \mu_X.$$

The variance σ_X^2 is given by

$$\int_{-\infty}^{\infty} p_X(x)\, (x - \mu_X)^2\, \mathrm{d}x = \int_a^b \frac{1}{b-a}\, (x - \mu_X)^2\, \mathrm{d}x.$$

By a change of variables, $y = x - \mu_X$ and using the shorthand $c = b - a$ one obtains

$$\sigma_X^2 = \frac{1}{c} \int_{-c/2}^{c/2} y^2\, \mathrm{d}y = \frac{1}{c} \left[\frac{1}{3} \left(\frac{c}{2}\right)^3 - \frac{1}{3} \left(-\frac{c}{2}\right)^3 \right] = \frac{c^2}{12} = \frac{(b-a)^2}{12}$$

2.2.3 The Normal Distribution

An important PDF in many applications is given by the normal distribution. A random variable X is called Gaussian or normally distributed, if its PDF is given by

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right).$$

Here, the two numbers $\mu \in \mathbb{R}$ and $\sigma > 0$ are numbers that characterize the distribution. One can again show by integration that this PDF is normed, i.e. $\int_{-\infty}^{\infty} p_X(x)\mathrm{d}x = 1$. Also, one can compute its mean μ_X and variance σ_X^2, which turn out to be given exactly by the parameters μ and σ^2. For this reason, one denotes the above PDF as a normal distribution with mean μ and variance σ^2. To express this fact in a compact way, one also writes $X \sim \mathcal{N}(\mu, \sigma^2)$, where the calligraphic $\mathcal{N}$ stands for "normal". The normal distribution is also called "Gaussian distribution" due to the fact that it was discovered by Carl Friedrich Gauss (1777 - 1855), a German mathematician who contributed to a variety of fields, including statistics.

2.2.4 Covariance and Correlation

Let us now regard two scalar random variables Y and Z with joint PDF $p_{Y,Z}(y, z)$. The PDF of Y is then given by the integral over z, i.e. $p_Y(y) = \int_{-\infty}^{\infty} p_{Y,Z}(y, z)\mathrm{d}z$, and likewise for Z. These are called the marginal distributions, and with their help one can compute the individual means μ_Y, μ_Z and variances σ_Y^2, σ_Z^2. In addition to the variances, one is also interested in the so-called *covariance* $\sigma(Y, Z)$ between Y and Z that is defined as the expectation of $(Y - \mu_Y)(Z - \mu_Z)$, i.e.

$$\sigma(Y, Z) := \mathbb{E}\{(Y - \mu_Y)(Z - \mu_Z)\} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} (y - \mu_Y)(z - \mu_Z)\, p_{Y,Z}(y, z)\, \mathrm{d}y\, \mathrm{d}z.$$

The physical unit of the covariance is the product of the units of the two random variables. To obtain a unitless quantity, one often divides the covariance by the two standard deviations, which yields the *correlation* $\rho(Y, Z)$ between Y and Z:

$$\rho(Y, Z) := \frac{\sigma(Y, Z)}{\sigma_Y\, \sigma_Z}.$$

One can show that the correlation is always in the interval $[-1, 1]$. If the correlation is zero, one says that the two variables are *uncorrelated*. If two variables are statistically independent, then they are always uncorrelated, but the converse is not true in general.

2.3 Multidimensional Random Variables

When we regard two random variables $Y \in \mathbb{R}$ and $Z \in \mathbb{R}$, one could also form the vector $X = [Y, Z]^\top \in \mathbb{R}^2$. The random variable is now characterized by the joint PDF $p_{Y,Z}(y, z) = p_X(x)$. More generally, we can regard a random vector $X \in \mathbb{R}^n$. The expectation of any function $f : \mathbb{R}^n \to \mathbb{R}^m$ of X can then be computed by

$$\mathbb{E}\{f(X)\} = \int_{\mathbb{R}^n} f(x)\, p_X(x)\, \mathrm{d}^n x.$$

2.3.1 Mean and Covariance Matrix

The expectation operator can also be applied to vector valued random variables, where the expectation is just computed for each component separately. We denote the mean of a random vector X by $\mu_X = \mathbb{E}\{X\}$. Note that μ_X is a vector of the same dimension as X. We generalize the variance to the so-called *covariance matrix* $\Sigma_X \in \mathbb{R}^{n\times n}$, which contains all variances and covariances in a single matrix. It is given by $\Sigma_X = \mathrm{cov}(X)$ where the covariance operator is defined by

$$\mathrm{cov}(X) = \mathbb{E}\{(X - \mu_X)(X - \mu_X)^\top\}.$$

It is easy to verify the identity $\mathrm{cov}(X) = \mathbb{E}\{XX^\top\} - \mu_X\mu_X^\top$.

2.3.2 Multidimensional Normal Distribution

We say that a vector valued random variable X is normally distributed with mean μ and covariance Σ if its PDF $p(x)$ is given by a multidimensional Gaussian as follows

$$p(x) = \frac{1}{\sqrt{\det(2\pi\Sigma)}} \exp\left(-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)\right)$$

As a shorthand, one also writes $X \sim \mathcal{N}(\mu, \Sigma)$ to express that X follows a normal distribution with mean μ and covariance Σ. One can verify by integral computations that indeed $\mathbb{E}\{X\} = \mu$ and $\mathrm{cov}(X) = \Sigma$.

2.4 Statistical Estimators

An *estimator* uses possibly many measurements in order to estimate the value of some parameter vector that we typically denote by θ in this script. The parameter is not random, but its true value, θ_0, is not known to the estimator. If we group all the measurements in a vector valued random variable $Y_N \in \mathbb{R}^N$, the estimator is a function of Y_N. We can denote this function by $\hat{\theta}_N(Y_N)$. Due to its dependence on Y_N, the estimate $\hat{\theta}_N(Y_N)$ is itself a random variable, for which we can define mean and covariance. Ideally, the expectation value of the estimator is equal to the true parameter value θ_0. We then say that the estimator is unbiased.

Definition 1 (Biased- and Unbiasedness) *An estimator $\hat{\theta}_N$ is called* unbiased *iff $\mathbb{E}\{\hat{\theta}_N(Y_N)\} = \theta_0$, where θ_0 is the true value of a parameter. Otherwise, it is called* biased*.*

Example for unbiasedness: estimating the mean by an average One of the simplest estimators tries to estimate the mean $\theta \equiv \mu_Y$ of a scalar random variable Y by averaging N measurements of Y. Each of these measurements $Y(k)$ is random, and overall the random vector Y_N is given by $Y_N = [Y(1), \ldots, Y(N)]^\top$. The estimator $\hat{\theta}_N(Y_N)$ is given by

$$\hat{\theta}_N(Y_N) = \frac{1}{N}\Sigma_{k=1}^N Y(k).$$

It is easy to verify that this estimator is unbiased, because

$$\mathbb{E}\{\hat{\theta}_N(Y_N)\} = \frac{1}{N}\Sigma_{k=1}^N \mathbb{E}\{Y(k)\} = \frac{1}{N}\Sigma_{k=1}^N \mu_Y = \mu_Y$$

Because this estimator is often used it has a special name. It is called the *sample mean*.

In order to assess the performance of an unbiased estimator, one can regard the covariance matrix of the estimates, i.e.

$$\mathrm{cov}(\hat{\theta}_N(Y_N))$$

The smaller this symmetric positive semi-definite matrix, the better the estimator. If two estimators $\hat{\theta}^A$ and $\hat{\theta}^B$ are both unbiased, and if the matrix inequality $\mathrm{cov}(\hat{\theta}^A) \succcurlyeq \mathrm{cov}(\hat{\theta}^B)$ holds, we can conclude that the estimator $\hat{\theta}^B$ has a better performance than estimator $\hat{\theta}^A$. Typically, the covariance of an estimator becomes smaller when an increasing number N of measurements is used. Often the covariance even tends to zero as $N \to \infty$.

Some estimators are not unbiased, but if N tends to infinity, their bias – i.e. the difference between true value and the mean of the estimate – tends to zero.

Definition 2 (Asymptotic Unbiasedness) *An estimator $\hat{\theta}_N$ is called* asymptotically unbiased *iff*

$$\lim_{N\to\infty} \mathbb{E}\{\hat{\theta}_N(Y_N)\} = \theta_0.$$

Example for asymptotically unbiasedness: estimating the variance by the mean squared deviations One of the simplest biased, but asymptotically unbiased estimators is tries to estimate the variance $\theta \equiv \sigma_Y^2$ of a scalar random variable Y by taking N measurements of Y, computing the experimental mean $M(Y_N) = \frac{1}{N}\Sigma_{k=1}^N Y(k)$, and then averaging the squared deviations from the mean

$$\hat{\theta}_N(Y_N) = \frac{1}{N}\Sigma_{k=1}^N (Y(k) - M(Y_N))^2$$

To show that it is biased, one has to consider that the sample mean $M(Y_N)$ is a random variable that is not independent from Y_N. One can compute its expectation value, which after some algebra is evaluated to be

$$\mathbb{E}\{\hat{\theta}_N\} = \frac{N-1}{N}\sigma_Y^2.$$

Only for $N \to \infty$, this estimator tends to the true value, so it is indeed *asymptotically unbiased*.

Because the bias is very easy to correct, in practice one rarely uses the above formula. Instead, to estimate the variance of a random variable Y, one uses the so called *sample variance* S^2 that is defined by

$$S^2 = \frac{1}{N-1}\Sigma_{n=1}^{N}(Y(n) - M(Y_N))^2.$$

Note the division by $N-1$ instead of N.

A stronger and even more desirable property than asymptotic unbiasedness is called *consistency*.

Definition 3 (Consistency) *An estimator $\hat{\theta}_N(Y_N)$ is called* consistent *if, for any $\epsilon > 0$, the probability $P(\hat{\theta}_N(Y_N) \in [\theta_0 - \epsilon, \theta_0 + \epsilon])$ tends to one as $N \to \infty$.*

It can be shown that an estimator is consistent if (a) it is asymptotically unbiased and (b) its covariance tends to zero as $N \to \infty$.

2.5 Analysis of the Resistance Estimation Example

In the previous chapter, we introduced the Resistance Estimation Example where we used three different ways to generate an estimator: the so-called "Simple approach" represented by Equation (2.1), the "Error in Variables" estimator represented by Equation (2.2) and finally the "Least Squares" estimator represented by Equation (2.3).

$$\hat{R}_{\text{SA}}(N) = \frac{1}{N} \cdot \sum_{k=1}^{N} \frac{u(k)}{i(k)} \tag{2.1}$$

$$\hat{R}_{\text{EV}}(N) = \frac{\frac{1}{N}\sum_{k=1}^{N} u(k)}{\frac{1}{N}\sum_{k=1}^{N} i(k)} \tag{2.2}$$

$$\hat{R}_{\text{LS}}(N) = \arg\min_{R\in\mathbb{R}} \sum_{k=1}^{N} (R \cdot i(k) - u(k))^2 = \frac{\frac{1}{N}\sum_{k=1}^{N} u(k) \cdot i(k)}{\frac{1}{N}\sum_{k=1}^{N} i(k)^2} \tag{2.3}$$

The aim of this section is to analyse each of the estimators and in particular, because this is easiest to analyze, to find out if they are asymptotically unbiased or not. In order to perform this analysis, we need to make some assumptions on the origin of the measurement errors, i.e. we analyse a model situation where one assumes to know the true value and the noise properties. We model the voltage $u(k)$ and the current $i(k)$ at time k as the true value of the voltage u_0 and current i_0, plus a zero mean i.i.d. noise $n_u(k)$ and $n_i(k)$. This means that we write

$$\begin{aligned} u(k) &= u_0 + n_u(k) \\ i(k) &= i_0 + n_i(k). \end{aligned} \tag{2.4}$$

Of course, the quotient of the true values u_0 and i_0 is the true value of the resistance, i.e. $R_0 = \frac{u_0}{i_0}$.

We also, make some assumptions on the disturbing noise, namely that $n_i(k)$ and $n_u(k)$ are independent from each other and are each independent identically distributed (i.i.d.), have symmetric distributions with zero mean, and have the (finite) variances σ_i^2 and σ_u^2.

In order to analyze the estimators, we need to make use of a result that is intuitively easy to accept, while a rigorous treatment would be involved. The result regards an infinite sequence of measurements $y(k)$ of a random variable Y and states that

$$\lim_{N\to\infty} \frac{1}{N} \sum_{k=1}^{N} y(k) = \mathbb{E}\{Y\}.$$

This implies for example that

$$\lim_{N\to\infty} \frac{1}{N} \sum_{k=1}^{N} u(k) = u_0.$$

Because the noises are zero mean, symmetrically distributed and independent, also their product has zero expectation, i.e.

$$\lim_{N\to\infty} \frac{1}{N} \sum_{k=1}^{N} n_i(k) n_u(k) = 0.$$

However, the limit of their squares is non-zero, because the mean of their squares is given by the variance. For example,

$$\lim_{N\to\infty} \frac{1}{N} \sum_{k=1}^{N} n_i(k) n_i(k) = \sigma_i^2.$$

In the following, we regard the three estimators in a different order than before, because the analysis of the simple approach is more involved than the one of the other two, and is therefore postponed to the end of the section.

2.5.1 Error in Variables estimator

Let us first look at the "error in variables" estimator:

$$\hat{R}_{\text{EV}}(N) = \frac{\frac{1}{N}\sum_{k=1}^{N} u(k)}{\frac{1}{N}\sum_{k=1}^{N} i(k)} \tag{2.5}$$

Let us now calculate the limit of $\hat{R}_{\text{EV}}(N)$ for $N \to \infty$, as follows.

$$\begin{aligned} \lim_{N\to\infty} \hat{R}_{\text{EV}}(N) &= \lim_{N\to\infty} \left\{ \frac{\frac{1}{N}\sum_{k=1}^{N} u(k)}{\frac{1}{N}\sum_{k=1}^{N} i(k)} \right\} = \frac{\lim_{N\to\infty}\{\frac{1}{N}\sum_{k=1}^{N}(u_0 + n_u(k))\}}{\lim_{N\to\infty}\{\frac{1}{N}\sum_{k=1}^{N}(i_0 + n_i(k))\}} \\ &= \frac{u_0 + \lim_{N\to\infty}\{\frac{1}{N}\sum_{k=1}^{N} n_u(k)\}}{i_0 + \lim_{N\to\infty}\{\frac{1}{N}\sum_{k=1}^{N} + n_i(k)\}} = \frac{u_0}{i_0} = R_0 \end{aligned} \tag{2.6}$$

We can conclude from this result that the Error in Variables estimator is asymptotically unbiased.

2.5.2 Least Squares estimator

Assuming the same properties for the measurements $u(k)$ and $i(k)$ as before, the Least Squares estimator can be analysed as follows. We start by separating true values and noise in the formula.

$$\hat{R}_{\text{LS}}(N) = \frac{\frac{1}{N}\sum_{k=1}^{N} u(k) \cdot i(k)}{\frac{1}{N}\sum_{k=1}^{N} i(k)^2} = \frac{\frac{1}{N}\sum_{k=1}^{N} (u_0 + n_u(k)) \cdot (i_0 + n_i(k))}{\frac{1}{N}\sum_{k=1}^{N} (i_0 + n_i(k))^2}. \tag{2.7}$$

Now, following the same procedure as before, we can calculate the limit when $N \to \infty$ for the numerator and denominator separately.

Numerator:

$$\begin{aligned} &\lim_{N\to\infty} \frac{1}{N} \sum_{k=1}^{N} (u_0 + n_u(k)) \cdot (i_0 + n_i(k)) \\ &= \lim_{N\to\infty} \frac{1}{N} \sum_{k=1}^{N} u_0 \cdot i_0 + \lim_{N\to\infty} \frac{1}{N} \sum_{k=1}^{N} i_0 \cdot n_u(k) + \\ &\quad + \lim_{N\to\infty} \frac{1}{N} \sum_{k=1}^{N} u_0 \cdot n_i(k) + \lim_{N\to\infty} \frac{1}{N} \sum_{k=1}^{N} n_i(k) \cdot n_u(k)) \\ &= u_0 \cdot i_0 \end{aligned} \tag{2.8}$$

The last line follows since $n_u(k), n_i(k)$ are zero mean, and $\lim_{N\to\infty} \frac{1}{N} \sum_{k=1}^{N} n_u(k), n_i(k) = 0$.

Denominator:

$$\begin{aligned}
&\lim_{N\to\infty} \frac{1}{N} \sum_{k=1}^{N} (i_0 + n_i(k))^2 \\
&= \lim_{N\to\infty} \frac{1}{N} \sum_{k=1}^{N} {i_0}^2 + \lim_{N\to\infty} \frac{1}{N} \sum_{k=1}^{N} 2 \cdot i_0 \cdot n_i(k) + \lim_{N\to\infty} \frac{1}{N} \sum_{k=1}^{N} n_i(k)^2 \\
&= {i_0}^2 + \sigma_i^2
\end{aligned} \tag{2.9}$$

The last line follows since $\lim_{N\to\infty} \frac{1}{N} \sum_{k=1}^{N} (n_i(k))^2 = \sigma_i^2$. As a result, the limit of the expected value of $\hat{R}_{\text{LS}}(N)$ is not the real value R_0, but instead:

$$\lim_{N\to\infty} \mathbb{E}\{\hat{R}_{\text{LS}}(N)\} = \frac{u_0 i_0}{i_0^2 + \sigma_i^2} = \frac{R_0}{1 + \frac{\sigma_i^2}{i_0^2}} \tag{2.10}$$

This means that the estimated value is always lower than the real value, making this estimator a biased estimator. It is interesting to note that the size of the bias depends only on the size of the noise on the current measurements relative to the true current, but not on the size of the noise of the voltage measurements. This is a general property of least squares estimators where the noise on the regression variable (here: the current) has an influence on the bias.

2.5.3 Simple Approach

Finally we take a look to the Simple Approach estimator defined by:

$$\hat{R}_{\text{SA}}(N) = \frac{1}{N} \cdot \sum_{k=1}^{N} \frac{u(k)}{i(k)} \tag{2.11}$$

Using the same assumptions as before we could already conclude that $\lim_{N\to} \infty \hat{R}_{\text{SA}}(N)$ might not even converge, since $i(k)$ can take zero values, and $\hat{R}_{\text{SA}}(N)$ would be an average of numbers, some of which take the value of infinity.

In spite of this fact, we can still try to find an approximate value for the expected value, assuming that no $i(k)$ would take the value of zero. For this aim, let us first expand the denominator using a Taylor series:

$$\frac{1}{i(k)} = \frac{1}{i_0 \cdot \left(1 + \frac{n_i(k)}{i_0}\right)} \approx \frac{1}{i_0} \cdot \left(1 - \frac{n_i(k)}{i_0} + \left(\frac{n_i(k)}{i_0}\right)^2 - \left(\frac{n_i(k)}{i_0}\right)^3 + \dots\right) \tag{2.12}$$

Though this series might not converge for noise values that are too large, we can use it to analyse the estimator under optimistic assumptions, namely under the assumption that all noise terms are small enough that the series converges. If we find a bias even under these optimistic assumptions, we can conclude that the overall estimator is biased as well. Let us continue as follows:

$$\begin{aligned}
&\lim_{N\to\infty} \frac{1}{N} \sum_{k=1}^{N} \hat{R}_{\text{SA}}(N) \\
&= \lim_{N\to\infty} \frac{1}{N} \sum_{k=1}^{N} \left\{ \frac{u_0 + n_u(k)}{i_0} \cdot \left(1 - \frac{n_i(k)}{i_0} + \left(\frac{n_i(k)}{i_0}\right)^2 - \left(\frac{n_i(k)}{i_0}\right)^3 + \dots\right) \right\} \\
&= \left\{ \frac{u_0}{i_0} \lim_{N\to\infty} \frac{1}{N} \sum_{k=1}^{N} \left(1 + \frac{n_u(k)}{u_0} - \frac{n_i(k)}{i_0} - \frac{n_i(k) \cdot n_u(k)}{i_0 \cdot u_0} + \frac{n_i(k)^2}{i_0^2} + \dots\right) \right\} \\
&\approx \frac{u_0}{i_0} \cdot \left(1 + \frac{\sigma_i^2}{i_0^2}\right)
\end{aligned} \tag{2.13}$$

Therefore, even when every $i(k) \neq 0$, the estimator would be biased. In this case, in contrast to the least squares estimator, the biased value is higher than the true value.

2.5.4 Discussing the results

When we first looked at the graph in Fig. 1.4, it was not possible to state which was the real value, and which of the estimators were biased. Now, with the analysis done in the previous section, it is pretty obvious that only the error in variables estimator $\hat{R}_{\mathrm{EV}}(N)$ converges to the true value of the resistance for high values of N, due to the fact that it is the only asymptotically unbiased estimator. In the case of the least squares estimator we can check that the theoretical analysis matches the graph: the least squares estimator produces a value smaller than the true one. Finally, in the case of the simple approach, we observed that it might not even converge. But if it does, the value that it estimates is higher than the real one. All this can be seen in the graphs in Fig. 1.4.

Chapter 3

Unconstrained Optimization in a Nutshell

Very often estimators for unknown parameters are obtained by minimizing some form of "misfit" between measurement data and model predictions. One simple example for this is the least-squares estimator that we encountered in the introduction. Most often, the optimization is with respect to the unknown parameters as decision variables, that are usually called θ in this script, and the objective function of the optimization problem depends on the measurement data, that we often call y. For least squares, we would for example minimize the misfit between the vector $y \in \mathbb{R}^N$ and the model predictions $M(\theta)$, i.e. the function to be minimized would be $f(\theta) := \|y - M(\theta)\|_2^2$. But let us abstract from the estimation context in this chapter, and simply use, as customary in the field of optimization, the variable $x \in \mathbb{R}^n$ as the unknown decision variable in the optimization problem. Thus, let us in this chapter regard unconstrained optimization problems of the form

$$\min_{x \in D} f(x), \tag{3.1}$$

where we regard objective functions $f : D \to \mathbb{R}$ that are defined on some open domain $D \subset \mathbb{R}^n$. We are only interested in minimizers that lie inside of D. We might have $D = \mathbb{R}^n$, but often this is not the case, e.g. as in the following example where the domain are only the positive numbers:

$$\min_{x \in (0,\infty)} \frac{1}{x} + x. \tag{3.2}$$

3.1 Optimality Conditions

Let us state a few simple and well-known results from unconstrained optimization that are often used in this course.

Theorem 1 (First Order Necessary Conditions) *If $x^* \in D$ is local minimizer of $f : D \to \mathbb{R}$ and $f \in C^1$ then*

$$\nabla f(x^*) = 0. \tag{3.3}$$

Definition 4 (Stationary Point) *A point $\bar{x}$ with $\nabla f(\bar{x}) = 0$ is called a stationary point of f.*

Given the above theorem, stationarity is a necessary, but of course not a sufficient condition for optimality. There will be one surprisingly large class of functions $f(x)$ for which stationarity is also sufficent for global optimality, namely the class of convex functions which we treat in the next section. For general nonlinear cost functions $f(x)$, however, we need to look at second order derivatives in order to decide if a stationary point is a minimizer or not. There exist necessary and sufficient conditions that are straightforward generalizations of well-known results from one dimensional analysis to $\mathbb{R}^n$. First, we recall the definition of positive (semi) definiteness for matrices.

Definition 5 (Generalized Inequality for Symmetric Matrices) *We write for a symmetric matrix $B = B^\top$, $B \in \mathbb{R}^{n \times n}$ that "$B \succcurlyeq 0$" if and only if B is positive semi-definite i.e., if $\forall z \in \mathbb{R}^n : \; z^\top B z \geq 0$, or, equivalently, if all (real) eigenvalues of the symmetric matrix B are non-negative:*

$$B \succcurlyeq 0 \iff \min \operatorname{eig}(B) \geq 0.$$

We write for two such symmetric matrices that "$A \succcurlyeq B$" iff $A - B \succcurlyeq 0$, and "$A \preccurlyeq B$" iff $B \succcurlyeq A$. We say $B \succ 0$ iff B is *positive definite*, i.e., if $\forall z \in \mathbb{R}^n \setminus \{0\} : z^\top B z > 0$, or, equivalently, if all eigenvalues of B are positive

$$B \succ 0 \iff \min \operatorname{eig}(B) > 0.$$

Theorem 2 (Second Order Necessary Conditions) *If $x^* \in D$ is local minimizer of $f : D \to \mathbb{R}$ and $f \in C^2$ then*

$$\nabla^2 f(x^*) \succcurlyeq 0. \tag{3.4}$$

Note that the matrix inequality is identical with the statement that all eigenvalues of the Hessian $\nabla^2 f(x^*)$ must be non-negative. It is possible that the Hessian has one or more zero eigenvalues – whose eigenvectors corresponds to directions of zero-curvature in the cost function. Due to this fact, the second order necessary condition (3.4) is not sufficient for a stationary point x^* to be a minimizer. This is illustrated by the simple one-dimensional functions $f(x) = x^3$ or $f(x) = -x^4$ for which $x^* = 0$ is a saddle point and a maximizer, respectively, though for both the necessary conditions $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*) \succcurlyeq 0$ are satisfied. How can we obtain a sufficient optimality condition for general nonlinear, but smooth functions f?

Theorem 3 (Second Order Sufficient Conditions and Stability under Perturbations) *Assume that $f : D \to \mathbb{R}$ is C^2. If $x^* \in D$ is a stationary point and*

$$\nabla^2 f(x^*) \succ 0. \tag{3.5}$$

then x^* is a strict local minimizer of f. In addition, this minimizer is locally unique and is stable against small perturbations of f, i.e.there exists a constant C such that for sufficiently small $p \in \mathbb{R}^n$ holds

$$\|x^* - \arg\min_x (f(x) + p^\top x)\| \leq C\|p\|.$$

This last theorem, which ensures stability under perturbations for optimal points that satisfy the second order sufficient conditions, is particularly important in the context of parameter estimation. Note that it requires that the Hessian matrix of the objective function is positive definite. If this condition is not satisfied, and we will see that it is not satisfied for so called "ill-posed" estimation problems, the optimization result, i.e. the estimate, will not be unique or vary strongly when the data vary.

3.2 Convex Optimization

As said above, convex optimization problems are particularly easy to solve to global optimality. In this section we first define convexity of sets and functions, state the main properties of convex optimization problems, and give some conditions to check if a function is convex.

First, a set is convex if all connecting lines between two points are fully contained in the set as well. Mathematically, this can be expressed in the following definition.

Definition 6 (Convex Set) *A set $\Omega \subset \mathbb{R}^n$ is convex if*

$$\forall x, y \in \Omega, t \in [0, 1] : \; x + t(y - x) \in \Omega. \tag{3.6}$$

Second, a function is convex, if all secants are above the graph:

Definition 7 (Convex Function) *A function $f : \Omega \to \mathbb{R}$ is convex, if its domain Ω is a convex set and if*

$$\forall x, y \in \Omega, t \in [0, 1] : \; f(x + t(y - x)) \leq f(x) + t(f(y) - f(x)). \tag{3.7}$$

Note that this definition is equivalent to saying that the Epigraph of f, i.e., the set $\{(x, s) \in \mathbb{R}^n \times \mathbb{R} | x \in \Omega, \; s \geq f(x)\}$, is a convex set. Finally, a *convex optimization problem* is an optimization problem where we want to minimize a convex function over a convex set. For unconstrained convex optimization problems, we assume that the minimizer lies in the interior of the convex set. The reason that statisticians and other scientists like convex optimization problems much more than the non-convex ones are the following two theorems.

Theorem 4 (Local Implies Global Optimality for Convex Problems) *For a convex optimization problem, every local minimum is also a global one.*

Theorem 5 (Convex First Order Sufficient Conditions) *Assume that $f : D \to \mathbb{R}$ is C^1 and convex. If $x^* \in D$ is a stationary point of f, then x^* is a global minimizer of f.*

We will extensively make use of this second theorem, because many of the optimization problems formulated in system identification are convex and we can then easily compute the minimizer by setting the gradient $\nabla f(x)$ to zero. An important convex objective function is the least squares cost function $f(x) = \|y - \Phi x\|_2^2$ with $\Phi \in \mathbb{R}^{N \times n}$, that is the subject of the next chapter.

There exists a whole algebra of operations that preserve convexity of functions and sets, which is excellently explained in good text books on convex optimization such as [BV04]. Here we only mention one important fact that is related to the positive curvature of a function.

Theorem 6 (Convexity for C^2 Functions) *Assume that $f : \Omega \to \mathbb{R}$ is twice continuously differentiable and Ω convex and open. Then f is convex if and only if for all $x \in \Omega$ the Hessian is positive semi-definite, i.e.,*

$$\forall x \in \Omega : \quad \nabla^2 f(x) \succcurlyeq 0. \tag{3.8}$$

Example: Let us regard the simple scalar optimization problem from the beginning, where we want to minimize $f(x) = \frac{1}{x} + x$ for all positive x. We compute the gradient $\nabla f(x) = -\frac{1}{x^2} + 1$ and the Hessian $\nabla^2 f(x) = \frac{2}{x^3}$. For positive numbers, the Hessian is always positive, i.e. $\nabla^2 f(x) \succcurlyeq 0$, and therefore, the function is convex. Thus, we can find its global minimum x^* by setting the gradient to zero:

$$\nabla f(x) = -\frac{1}{x^2} + 1 = 0.$$

This equation admits only one positive solution, $x^* = 1$. This point must be the global minimizer.

It is interesting to also investigate the stability of this minimizer under perturbations, which is guaranteed by the fact that it also satisfies the second order sufficient conditions for local optimality (because the Hessian is strictly positive). Thus, let us compute the minimizer of $f(x) + px$ for arbitrary $p \in \mathbb{R}$. By the same reasoning as above, we obtain

$$-\frac{1}{x^2} + 1 + p = 0,$$

and for small p the only positive solution is given by

$$x^*(p) = \frac{1}{\sqrt{1+p}},$$

and indeed, the difference is bounded by a linear term

$$\|x^*(0) - x^*(p)\| = \frac{|\sqrt{1+p} - 1|}{\sqrt{1+p}} \leq C|p|$$

e.g. with $C = 1$ for sufficiently small p.

Part II

General Parameter Estimation Methods

Chapter 4

Linear Least Squares Estimation

Linear least squares (LLS or just LS) is a technique that helps us to find a model that is linear in some unknown parameters $\theta \in \mathbb{R}^d$. For this aim, we regard a sequence of measurements $y(1), \ldots, y(N) \in \mathbb{R}$ that shall be explained – they are also called the *dependent variables* – and another sequence of *regression vectors* $\phi(1), \ldots, \phi(N) \in \mathbb{R}^d$, which are regarded as the inputs of the model and are also called the *independent or explanatory variables*. Prediction errors are modelled by additive measurement noise $\epsilon(1), \ldots, \epsilon(N)$ with zero mean such that the overall model is given by

$$y(k) = \phi(k)^\top \theta + \epsilon(k), \quad \text{for} \quad k = 1, \ldots, N.$$

Let us in this section regard only scalar measurements $y(k)$, though LLS can be generalized easily to the case of several dependent variables. The task of LLS is to find an estimate $\hat{\theta}_{\mathrm{LS}}$ for the true but unknown parameter vector θ_0. Often the ultimate aim is to be able to predict a y for any given new values of the regression vector ϕ by the model $y = \phi^\top \hat{\theta}_{\mathrm{LS}}$.

4.1 Least Squares Problem Formulation

Idea of linear least squares is to find the θ that minimizes the sum of the squares of the prediction errors $y(k) - \phi(k)^\top \theta$, i.e. the *least squares cost function*

$$\sum_{k=1}^{N} \left(y(k) - \phi(k)^\top \theta\right)^2.$$

Stacking all values $y(k)$ into one long vector $y_N \in \mathbb{R}^N$ and all regression vectors as rows into one matrix $\Phi_N \in \mathbb{R}^{N \times d}$, i.e.,

$$y_N = \begin{bmatrix} y(1) \\ \vdots \\ y(N) \end{bmatrix} \quad \text{and} \quad \Phi_N = \begin{bmatrix} \phi(1)^\top \\ \vdots \\ \phi(N)^\top \end{bmatrix}$$

we can write the least squares cost function[1] as

$$f(\theta) = \|y_N - \Phi_N \theta\|_2^2.$$

The least squares estimate $\hat{\theta}_{\mathrm{LS}}$ is the value of θ that minimizes this function. Thus, we are faced with an *unconstrained optimization problem* that can be written as

$$\min_{\theta \in \mathbb{R}^d} f(\theta).$$

In estimation, we are mainly interested in the input arguments of f that achieve the minimal value, which we call the minimizers. The set of minimizers S^* is denoted by

$$S^* = \arg\min_{\theta \in \mathbb{R}^d} f(\theta).$$

[1] We recall that for any vector $x \in \mathbb{R}^n$, we define the Euclidean norm as $\|x\|_2 = \left(\sum_{i=1}^n x_i^2\right)^{1/2} = (x^\top x)^{1/2}$.

Note that there can be several minimizers. If the minimizer is unique, we have only one value in the set, that we denote θ^*, and we can slightly sloppily identify θ^* with $\{\theta^*\}$. The least squares estimator $\hat{\theta}_{\mathrm{LS}}$ is given by this unique minimizer, such that we will often write

$$\hat{\theta}_{\mathrm{LS}} = \arg\min_{\theta \in \mathbb{R}^d} f(\theta).$$

But in order to compute the minimizer (or the set of minimizers), we need to solve an optimization problem. Let us therefore recall a few concepts from optimization, and then give an explicit solution formula for $\hat{\theta}_{\mathrm{LS}}$.

4.2 Solution of the Linear Least Squares Problem

The function $f(\theta) = \frac{1}{2} \parallel y_N - \Phi_N \theta \parallel_2^2$ is convex. Therefore local minimizers are found by just setting the gradient to zero. For notational convenience, we will in this section omit the subindex N and write $f(\theta) = \frac{1}{2} \parallel y - \Phi\theta \parallel_2^2$, and we will refer to the components of y with a simple subindex, i.e. write y_k instead of $y(k)$. Also, we have introduced a factor $\frac{1}{2}$ in the objective, which does not change the minimizer. We introduced it because it will cancel a factor two that would otherwise be present in the first and second derivatives of f. To find the minimizer, let us compute the gradient of f.

$$\begin{aligned} \nabla f(\theta^*) = 0 \quad &\Leftrightarrow \quad \Phi^\top \Phi \theta^* - \Phi^\top y = 0 \\ &\Leftrightarrow \quad \theta^* = \underbrace{(\Phi^\top \Phi)^{-1} \Phi^\top}_{=\Phi^+} y \end{aligned} \tag{4.1}$$

Definition 8 (Pseudo inverse) *Φ^+ is called the pseudo inverse of the matrix Φ and is a generalization of the inverse matrix. If $\Phi^\top \Phi \succ 0$, the pseudo inverse Φ^+ is given by*

$$\Phi^+ = (\Phi^\top \Phi)^{-1} \Phi^\top \tag{4.2}$$

So far, $(\Phi^\top \Phi)^{-1}$ is only defined when $\Phi^\top \Phi \succ 0$. This holds if and only if $\mathrm{rank}(\Phi) = d$, i.e., if the columns of Φ are linearly independent. In this context, it is interesting to note that $\nabla^2 f(\theta) = \Phi^\top \Phi$, i.e. the pseudo inverse is well-defined if and only if the second order sufficient conditions for optimality are satisfied.

Later, we will generalize the pseudo inverse to the case that Φ has linearly dependent column vectors, i.e. that the matrix $\Phi^\top \Phi$ has one or more zero eigenvalues. Due to convexity of f, points with $\nabla f(\theta) = 0$ will still be minimizers in that case, but they will not be unique anymore. But let us first illustrate the regular case with $\Phi^\top \Phi \succ 0$ in two examples.

Example 1 (Fitting a constant equals taking the average) *Let us regard the simple optimization problem:*

$$\min_{\theta \in \mathbb{R}} \frac{1}{2} \sum_{i=1}^{N} (y_i - \theta)^2.$$

This is a linear least squares problem, where the vector y and the matrix $\Phi \in \mathbb{R}^{N \times 1}$ are given by

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}, \; \Phi = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}. \tag{4.3}$$

Because $\Phi^\top \Phi = N$, it can be easily seen that

$$\Phi^+ = (\Phi^\top \Phi)^{-1} \Phi^\top = \frac{1}{N} \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix} \tag{4.4}$$

so we conclude that the local minimizer equals the average of the given points y_i:

$$\theta^* = \Phi^+ y = \frac{1}{N} \sum_{i=1}^{N} y_i. \tag{4.5}$$

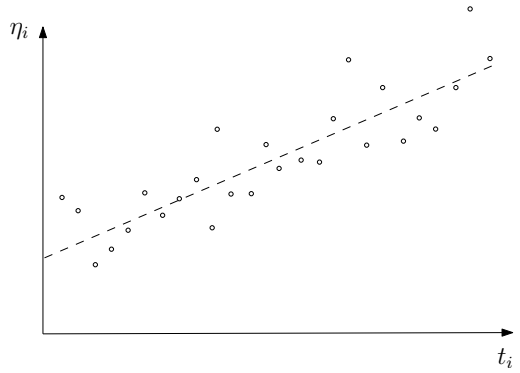


Figure 4.1: Linear regression for a set of data points (t_i, y_i)

Example 2 (Fitting a line) *Given data points $\{t_i\}_{i=1}^N$ with corresponding values $\{y_i\}_{i=1}^N$, find the 2-dimensional parameter vector $\theta = (\theta_1, \theta_2)$, so that the polynomial of degree one $p(t;\theta) = \theta_1 + \theta_2 t$ provides a prediction of y at time t. The corresponding optimization problem looks like:*

$$\min_{\theta\in\mathbb{R}^2} \frac{1}{2}\sum_{i=1}^{N}(y_i - p(t_i;\theta))^2 = \min_{\theta\in\mathbb{R}^2} \frac{1}{2}\left\| y - \Phi \begin{bmatrix}\theta_1\\ \theta_2\end{bmatrix}\right\|_2^2 \tag{4.6}$$

where y is the same vector as in (4.3) *and Φ is given by*

$$\Phi = \begin{bmatrix} 1 & t_1 \\ 1 & t_2 \\ \vdots & \vdots \\ 1 & t_N \end{bmatrix}. \tag{4.7}$$

The local minimizer is found by equation (4.1)*, where the calculation of $(\Phi^\top\Phi)$ is straightforward:*

$$\Phi^\top\Phi = \left[\begin{array}{c|c} N & \sum t_i \\ \hline \sum t_i & \sum t_i^2 \end{array}\right] \tag{4.8}$$

4.3 Weighted Least Squares

One might want to give different weights to different residuals in the sum of the linear least squares cost function. This is important if the measurement errors $\epsilon(k)$ have zero mean and are independent, but are not identically distributed, such that they have different variances $\sigma_\epsilon^2(k)$. We would intuitively like to give less weight to those measurements which are corrupted by stronger noise. Weighting is mandatory if different measurements represent different physical units, if we want to avoid that we add squared apples to squared pears. Fortunately, the variance of each measurement has the same unit as the measurement squared, such that a division of each residual by the variance would make all terms free of units. For this reason one nearly always uses the following weighted least squares cost function:

$$f_{\mathrm{WLS}}(\theta) = \sum_{k=1}^{N} \frac{(y(k) - \phi^\top\theta)^2}{\sigma_\epsilon^2(k)},$$

and we will see that this cost function ensures the best possible performance of the least squares estimator. To bring this into a more compact notation, we can introduce a diagonal weighting matrix

$$W = \begin{bmatrix} \sigma_\epsilon^{-2}(1) & & \\ & \ddots & \\ & & \sigma_\epsilon^{-2}(N) \end{bmatrix}$$

and then write[2]

$$f_{\mathrm{WLS}}(\theta) = \|y - \Phi\theta\|_W^2.$$

Even more general, one might use any symmetric positive definite matrix $W \in \mathbb{R}^{N\times N}$ as weighting matrix. The optimal solution is given by

$$\hat{\theta}_{\mathrm{WLS}} = \arg\min f_{\mathrm{WLS}}(\theta) = (\Phi^\top W\Phi)^{-1}\Phi^\top W y.$$

There is an alternative way to represent the solution, using the matrix $\tilde{\Phi} = W^{\frac{1}{2}}\Phi$ and its pseudo inverse. To derive this alternative way, let us first state the fact that there exists a unique symmetric square root $W^{\frac{1}{2}}$ for any symmetric positive definite matrix W. For example, for a diagonal weighting matrix as above, the square root is given by

$$W^{\frac{1}{2}} = \begin{bmatrix} \sigma_\epsilon^{-1}(1) & & \\ & \ddots & \\ & & \sigma_\epsilon^{-1}(N) \end{bmatrix}.$$

With this square root matrix, we have the trivial identity $\|x\|_W^2 = \|W^{\frac{1}{2}}x\|_2^2$ for any vector $x \in \mathbb{R}^N$ and can therefore write

$$f_{\mathrm{WLS}}(\theta) = \|\underbrace{W^{\frac{1}{2}}y}_{=:\tilde{y}} - \underbrace{W^{\frac{1}{2}}\Phi}_{=:\tilde{\Phi}}\theta\|_2^2.$$

Thus, the weighted least squares problem is nothing else than an unweighted least squares problem with rescaled measurements $\tilde{y} = W^{\frac{1}{2}}y$ and rescaled regressor matrix $\tilde{\Phi} = W^{\frac{1}{2}}\Phi$, and the solution can be computed using the pseudo inverse of $\tilde{\Phi}$ and is simply given by

$$\hat{\theta}_{\mathrm{WLS}} = \tilde{\Phi}^+\tilde{y}.$$

This way of computing the estimate is numerically more stable so it is in general preferable. An important observation is that the resulting solution vector $\hat{\theta}_{\mathrm{WLS}}$ does not depend on the total scaling of the entries of the weighting matrix, i.e. for any positive real number α, the weighting matrices W and αW deliver identical results $\hat{\theta}_{\mathrm{WLS}}$. Only for this reason it is meaningful to use unweighted least squares – they deliver the optimal result in the case that the measurement errors are assumed to be independent and identically distributed. But generally speaking, all least squares problems are in fact weighted least squares problems, because one always has to make a choice of how to scale the measurement errors. If one uses unweighted least squares, one implicitly chooses the unit matrix as weighting matrix, which makes sense for i.i.d. measurement errors, but otherwise not. For ease of notation, we will in the following nevertheless continue discussing the unweighted LS formulation, keeping in mind that any weighted least squares problem can be brought into this form by the above rescaling procedure.

4.4 Ill-Posed Least Squares and the Moore Penrose Pseudo Inverse

In some cases, the matrix $\Phi^\top\Phi$ is not invertible, i.e. it contains at least one zero eigenvalue. In this case the estimation problem is called ill-posed, because the solution is not unique. But there is still the possibility to obtain a solution of the least squares problem that might give a reasonable result. For this we have to use a special type of pseudo inverse. Let us recall that definition (4.2) of the pseudo inverse does only hold if $\Phi^\top\Phi$ is invertible. This implies that the set of optimal solutions S^* has only one optimal point θ^*, given by $S^* = \{\theta^*\} = (\Phi^\top\Phi)^{-1}\Phi y$. If $\Phi^\top\Phi$ is not invertible, the set of solutions S^* is given by

$$S^* = \{\theta \mid \nabla f(\theta) = 0\} = \{\theta|\Phi^\top\Phi\theta - \Phi^\top y = 0\} \tag{4.9}$$

In order to pick a unique point out of this set, we might choose to search for the "minimum norm solution", i.e. the vector θ^* with minimum norm satisfying $\theta^* \in S^*$.

$$\min_{\theta\in\mathbb{R}^n} \quad \frac{1}{2}\|\theta\|_2^2 \quad \text{subject to} \quad \theta \in S^* \tag{4.10}$$

We will show below that this minimal norm solution is given by the so called "Moore Penrose Pseudo Inverse".

[2]Recall that for any positive definite matrix W the weighted Euclidean norm $\|x\|_W$ is defined as $\|x\|_W = \sqrt{x^\top Wx}$.

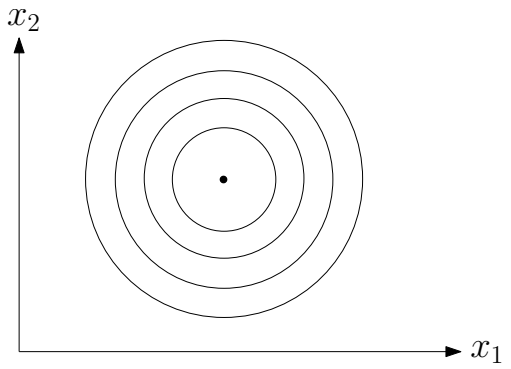


Figure 4.2: $\Phi^\top \Phi$ is invertible, resulting in a unique minimum.

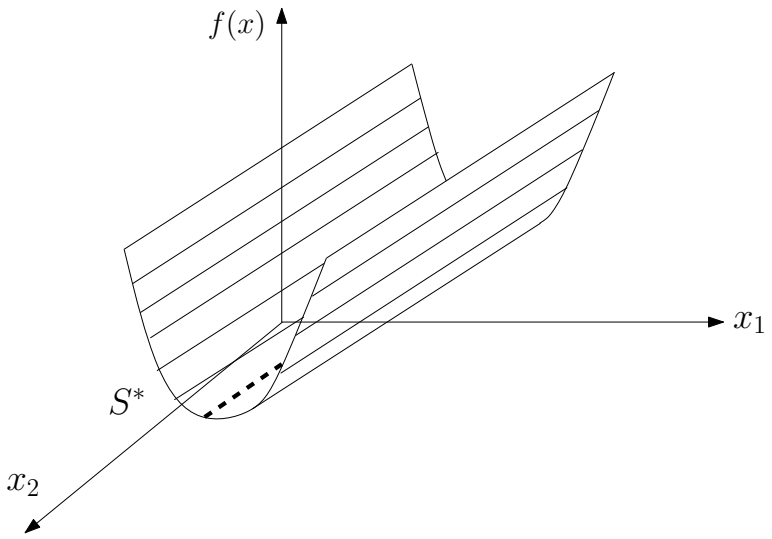


Figure 4.3: An example of an ill-posed problem, $\Phi^\top \Phi$ is not invertible

Definition 9 (Singular value decomposition (SVD)) *Assume $A \in \mathbb{R}^{mxn}$ with $\operatorname{rank}(A) = r \leq \min(m, n)$. In that case, A can be factored as:*

$$A = USV^\top, \tag{4.11}$$

where $U \in \mathbb{R}^{mxm}$, $V \in \mathbb{R}^{nxn}$, and both of them are unitary matrices, i.e, both satisfy $U^\top U = I$ and $V^\top V = I$, where I is the identity matrix. Furthermore, $S \in \mathbb{R}^{mxn}$ is a matrix with non-negative elements $(\sigma_1, \ldots, \sigma_r, 0, \ldots, 0)$ on the diagonal and zeros everywhere else, where $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r \geq 0$. This is known as the singular value decomposition (SVD) of A.

Definition 10 (Moore Penrose Pseudo Inverse) *Assume $\Phi \in \mathbb{R}^{m \times n}$ and that the singular value decomposition (SVD) of Φ is given by $\Phi = USV^\top$. Then, the Moore Penrose Pseudo Inverse Φ^+ is given by:*

$$\Phi^+ = VS^+U^\top, \tag{4.12}$$

where for

$$S = \left[\begin{array}{cccccc} \sigma_1 & & & & & \\ & \sigma_2 & & & & \\ & & \ddots & & & \\ & & & \sigma_r & & \\ & & & & \ddots & \\ & & & & & 0 \\ \hline 0 & \ldots & 0 & \ldots & & 0 \end{array}\right] \quad \textit{holds} \quad S^+ = \left[\begin{array}{cccccc|c} \sigma_1^{-1} & & & & & & 0 \\ & \sigma_2^{-1} & & & & & \vdots \\ & & \ddots & & & & 0 \\ & & & \sigma_r^{-1} & & & \vdots \\ & & & & 0 & & \\ & & & & & \ddots & \\ & & & & & 0 & 0 \end{array}\right] \tag{4.13}$$

If $\Phi^\top\Phi$ is invertible, then $\Phi^+ = (\Phi^\top\Phi)^{-1}\Phi^\top$ what easily can be shown:

$$\begin{aligned} (\Phi^\top\Phi)^{-1}\Phi^\top &= (VS^\top U^\top USV^\top)^{-1}VS^\top U^\top \\ &= V(S^\top S)^{-1}V^\top VS^\top U^\top \\ &= V(S^\top S)^{-1}S^\top U^\top \\ &= V \begin{bmatrix} \sigma_1^2 & & & \\ & \sigma_2^2 & & \\ & & \ddots & \\ & & & \sigma_r^2 \end{bmatrix}^{-1} \left[\begin{array}{ccc|c} \sigma_1 & & & 0 \\ & \sigma_2 & & \\ & & \ddots & \\ & & \sigma_r & \end{array}\right] U^\top \\ &= VS^+U^\top \end{aligned}$$

4.4.1 Regularization for Least Squares

The minimum norm solution can be approximated by a "regularized problem"

$$\min_\theta \ \frac{1}{2}\|y - \Phi\theta\|_2^2 + \frac{\alpha}{2}\|\theta\|_2^2, \tag{4.14}$$

with small $\alpha > 0$, to get a unique solution

$$\begin{aligned} \nabla f(\theta) &= \Phi^\top\Phi\theta - \Phi^\top y + \alpha\theta && (4.15)\\ &= (\Phi^\top\Phi + \alpha\mathbb{I})\theta - \Phi^\top y, && (4.16)\\ \text{thus} \quad \theta^* &= (\Phi^\top\Phi + \alpha\mathbb{I})^{-1}\Phi^\top y && (4.17)\\ & && (4.18) \end{aligned}$$

Lemma 1 $\lim_{\alpha\to 0}(\Phi^\top\Phi + \alpha\mathbb{I})^{-1}\Phi^\top = \Phi^+$, *with Φ^+ the Moore Penrose Pseudo Inverse.*

Proof: Taking the SVD of $\Phi = USV^\top$, $(\Phi^\top\Phi + \alpha\mathbb{I})^{-1}\Phi^\top$ can be written in the form:

$$\begin{aligned}(\Phi^\top\Phi + \alpha\mathbb{I})^{-1}\Phi^\top &= (VS^\top U^\top USV^\top + \alpha\underbrace{\mathbb{I}}_{VV^\top})^{-1}\underbrace{\Phi^\top}_{US^\top V^\top} \\ &= V(S^\top S + \alpha\mathbb{I})^{-1}V^\top VS^\top U^\top \\ &= V(S^\top S + \alpha\mathbb{I})^{-1}S^\top U^\top\end{aligned}$$

Rewriting the right hand side of the equation explicitly:

$$= V\begin{bmatrix}\sigma_1^2+\alpha & & & & & \\ & \ddots & & & & \\ & & \sigma_r^2+\alpha & & & \\ & & & \alpha & & \\ & & & & \ddots & \\ & & & & & \alpha\end{bmatrix}^{-1}\left[\begin{array}{cccc|c}\sigma_1 & & & & 0 \\ & \ddots & & & \vdots \\ & & \sigma_r & & \\ & & & 0 & \\ & & & & \ddots \\ & & & & 0 \;\; 0\end{array}\right]U^\top$$

Calculating the matrix product simplifies the equation:

$$= V\left[\begin{array}{cccccc|c}\frac{\sigma_1}{\sigma_1^2+\alpha} & & & & & & 0 \\ & \ddots & & & & & \\ & & \frac{\sigma_r}{\sigma_r^2+\alpha} & & & & \vdots \\ & & & \frac{0}{\alpha} & & & \\ & & & & \ddots & & \\ & & & & & \frac{0}{\alpha} & 0\end{array}\right]U^T$$

It can be easily seen that for $\alpha \to 0$ each diagonal element has the solution:

$$\lim_{\alpha\to 0}\frac{\sigma_i}{\sigma_i^2+\alpha} = \begin{cases}\frac{1}{\sigma_i} & \text{if} \quad \sigma_i \neq 0 \\ 0 & \text{if} \quad \sigma_i = 0\end{cases} \tag{4.19}$$

With the above lemma, we have shown that the Moore Penrose Pseudo Inverse Φ^+ solves the problem (4.14) for infinitely small $\alpha > 0$. Thus it selects $\theta^* \in S^*$ with minimal norm.

4.5 Statistical Analysis of the Weighted Least Squares Estimator

In the following, we analyse the weighted least squares estimator, and we make the following assumptions:

- the N measurements $y(k)$ are generated by a model $y(k) = \phi(k)^\top\theta_0 + \epsilon(k)$ with θ_0 the true but unknown parameter value
- the N regression vectors $\phi(k)$ are deterministic and not corrupted by noise (attention: this assumption is often violated in practice)
- the N noise terms $\epsilon(k)$ have zero mean. Often, we can also assume that they are independent from each other, or even that they are i.i.d., and this will have consequences on the optimal choice of weighting matrix, as we will see. We abbreviate $\epsilon_N = [\epsilon(1), \ldots, \epsilon(N)]^\top$.
- the weighted least squares estimator is computed as $\hat{\theta}_{\text{WLS}} = (\Phi_N^\top W\Phi_N)^{-1}\Phi_N^\top W y_N$.

We are most interested in the following questions: is the estimator biased or unbiased? What is its performance, i.e. what covariance matrix does the estimate have?

4.5.1 The expectation of the least squares estimator

Let us compute the expectation of $\hat{\theta}_{\mathrm{WLS}}$.

$$\begin{aligned}
\mathbb{E}\{\hat{\theta}_{\mathrm{WLS}}\} &= \mathbb{E}\left\{(\Phi_N^\top W \Phi_N)^{-1}\Phi_N^\top W y_N\right\} && (4.20)\\
&= (\Phi_N^\top W \Phi_N)^{-1}\Phi_N^\top W \;\; \mathbb{E}\{y_N\} && (4.21)\\
&= (\Phi_N^\top W \Phi_N)^{-1}\Phi_N^\top W \quad \mathbb{E}\{\Phi_N\theta_0 \qquad + \epsilon_N\} && (4.22)\\
&= (\Phi_N^\top W \Phi_N)^{-1}(\Phi_N^\top W \Phi_N) \quad \theta_0 \qquad + \; (\Phi_N^\top W \Phi_N)^{-1}\Phi_N^\top W \,\mathbb{E}\{\epsilon_N\} && (4.23)\\
&= \qquad \theta_0 \qquad + \; 0 && (4.24)
\end{aligned}$$

Thus, the weighted least squares estimator has the true parameter value θ_0 as expectation value, i.e. it is an unbiased estimator. This fact is true independent of the choice of weighting matrix W.

4.5.2 The covariance of the least squares estimator

In order to assess the performance of an unbiased estimator, one can look at the covariance matrix of $\hat{\theta}_{\mathrm{WLS}}$. The smaller this covariance matrix in the matrix sense, the better is the estimator. Let us therefore compute the covariance matrix of $\hat{\theta}_{\mathrm{WLS}}$. Using the identity $\hat{\theta}_{\mathrm{WLS}} - \theta_0 = (\Phi_N^\top W \Phi_N)^{-1}\Phi_N^\top W \epsilon_N$, it is given by

$$\begin{aligned}
\mathrm{cov}(\hat{\theta}_{\mathrm{WLS}}) &= \mathbb{E}\{(\hat{\theta}_{\mathrm{WLS}} - \theta_0)(\hat{\theta}_{\mathrm{WLS}} - \theta_0)^\top\} && (4.25)\\
&= (\Phi_N^\top W \Phi_N)^{-1}\Phi_N^\top W \,\mathbb{E}\left\{\epsilon_N \epsilon_N^\top\right\} W\Phi_N(\Phi_N^\top W \Phi_N)^{-1} && (4.26)\\
&= (\Phi_N^\top W \Phi_N)^{-1}\Phi_N^\top W \quad \Sigma_{\epsilon_N} \quad W\Phi_N(\Phi_N^\top W \Phi_N)^{-1}. && (4.27)
\end{aligned}$$

Here, we have used the shorthand $\Sigma_{\epsilon_N} = \mathrm{cov}(\epsilon_N)$. For different choices of W, the covariance $\mathrm{cov}(\hat{\theta}_{\mathrm{WLS}})$ will be different. However, there is one specific choice that makes the above formula very easy: if we happen to know Σ_{ϵ_N} and would choose $W := \Sigma_{\epsilon_N}^{-1}$, we would obtain

$$\begin{aligned}
\mathrm{cov}(\hat{\theta}_{\mathrm{WLS}}) &= (\Phi_N^\top W \Phi_N)^{-1}\Phi_N^\top W \quad W^{-1} \quad W\Phi_N(\Phi_N^\top W \Phi_N)^{-1} && (4.28)\\
&= (\Phi_N^\top W \Phi_N)^{-1}(\Phi_N^\top W \Phi_N)(\Phi_N^\top W \Phi_N)^{-1} && (4.29)\\
&= (\Phi_N^\top W \Phi_N)^{-1} && (4.30)\\
&= (\Phi_N^\top \Sigma_{\epsilon_N}^{-1} \Phi_N)^{-1}. && (4.31)
\end{aligned}$$

Interestingly, it turns out that this choice of weighting matrix is the optimal choice, i.e. for all other weighting matrices W one has

$$\mathrm{cov}(\hat{\theta}_{\mathrm{WLS}}) \succcurlyeq (\Phi_N^\top \Sigma_{\epsilon_N}^{-1} \Phi_N)^{-1}.$$

Even more, one can show that in case of Gaussian noise with zero mean and covariance Σ_{ϵ_N}, the weighted linear least squares estimator with optimal weights $W = \Sigma_{\epsilon_N}^{-1}$ achieves the lower bound on the covariance matrix that any unbiased estimator can achieve (the so called Cramer-Rao lower bound).

4.6 Measuring the Goodness of Fit using R-Squared

In the previous sections we have provided an indicator of how good an unbiased estimator can be: the covariance matrix. Nevertheless, the covariance is a matrix which provides information relative to the problem that is solved, so a simple change of units of the parameters would change the values of the covariance. Also, it does not really say if we have a good fit or not. Furthermore, because of its matrix structure, it is a rather difficult indicator to read. As a consequence, one would like to use another indicator hat is just assesssing the "goodness of fit" and that is easier to interpret. Ideally, we would like to have one scalar number, which can take values from a fixed range of values that is independent of the problem data size. Such an indicator it is given by the so called *coefficient of determination* or *R-squared* (R^2) value which is represented by the following expression:

$$R^2 = 1 - \frac{\|y_N - \Phi_N \cdot \hat{\theta}\|_2^2}{\|y_N\|_2^2} \tag{4.32}$$

Here, one usually first subtracts the mean from the measurement data before computing the denominator $\|y_N\|_2^2$, i.e. one ensures that $\sum_{k=1}^N y(k) = 0$.

The R^2 value is always between zero and one. A value of one means that the fit is perfect, i.e. the estimated values of the linear fit, $\hat{y}_N := \Phi_N \cdot \hat{\theta}$, coincide with the measurements, y_N. A value of zero means that the linear model is not able to explain any of the data. To get a more intuitive interpretation of intermediate R^2 values, it is interesting to regard the residuals $\epsilon_N = y_N - \hat{y}_N$ and exploit the fact that $\hat{y}_N$ and ϵ_N are orthogonal vectors. This implies that

$$R^2 = 1 - \frac{\|\epsilon_N\|_2^2}{\|y_N\|_2^2} = \frac{\|y_N\|_2^2 - \|\epsilon_N\|_2^2}{\|y_N\|_2^2} = \frac{\|\hat{y}_N\|_2^2}{\|y_N\|_2^2}.$$

This means that the R^2 value can be regarded as a measure of how much of the variation in the data can be explained by the linear model. In the derivation above, the relation $\|\hat{y}_N\|_2^2 = \|y_N\|_2^2 - \|\epsilon_N\|_2^2$ can be proved taking into account the orthogonality condition of ϵ_N and $\hat{y}_N$ that is due to optimality of $\hat{\theta}$. The optimality condition of $\hat{\theta}$ in the least squares problem problem is $\Phi_N^\top \cdot (\Phi_N \cdot \hat{\theta} - y_N) = 0$ i.e. $\Phi_N^T \cdot \epsilon_N = 0$. This implies that $\hat{\theta}^\top \cdot \Phi_N^\top \cdot \epsilon_N = \hat{y}_N^\top \cdot \epsilon_N = 0$ i.e. that $\hat{y}_N$ and ϵ_N are orthogonal. Orthogonality and the fact that $y_N = \hat{y}_N + \epsilon_N$ implies that $\|y_N\|_2^2 = \|\hat{y}_N\|_2^2 + \|\hat{\epsilon}_N\|_2^2$.

4.7 Estimating the Covariance with a Single Experiment

So far, we have analysed the theoretical properties of the LS estimator, and we know that for independent identically distributed measurement errors, the unweighted least squares estimater gives us the optimal estimator. If the variance of the noise is σ_ϵ^2, the least squares estimator $\hat{\theta}_{\mathrm{LS}} = \Phi_N^+ y_N$ is a random variable with the true parameter value θ_0 as mean and the following covariance matrix:

$$\Sigma_{\hat{\theta}} := \mathrm{cov}(\hat{\theta}_{\mathrm{LS}}) = \sigma_\epsilon^2 (\Phi_N^\top \Phi_N)^{-1}.$$

In addition, if the number of measurements N in one experiment is large, by a law of large numbers, the distribution of $\hat{\theta}_{\mathrm{LS}}$ follows approximately a normal distribution, even if the measurement errors were not normally distributed. Thus, if one repeats the same experiment with the same N regression vectors many times, the estimates $\hat{\theta}_{\mathrm{LS}}$ would follow a normal distribution characterized by these two parameters, i.e. $\hat{\theta}_{\mathrm{LS}} \sim \mathcal{N}(\theta_0, \Sigma_{\hat{\theta}})$. In a realistic application, however, the situation is quite different than in this analysis:

- First, we do of course *not* know the true value θ_0
- Second, we do not repeat our experiment many times, but just do *one single experiment*.
- Third, we typically do *not* know the variance of the measurement noise σ_ϵ^2.

Nevertheless, and surprisingly, if one makes the assumption that the noise is independent identically distributed, one is able to make a very good guess of the covariance matrix of the estimator, which we will describe here. The main reason is that we know the deterministic matrix Φ_N exactly. The covariance is basically given by the matrix $(\Phi_N^\top \Phi)^{-1}$, which only needs to be scaled by a factor, the unknown σ_ϵ^2. Thus, we only need to find an estimate for the noise variance. Fortunately, we have N measurements $y(k)$ as well as the corresponding model predictions $\phi(k)^\top \hat{\theta}_{\mathrm{LS}}$ for $k = 1, \ldots, N$, so their average difference can be used to estimate the measurement noise. Because the predictions are based on fitting the d-dimensional vector $\hat{\theta}_{\mathrm{LS}}$ to the same measurements $y(k)$ that we want to use to estimate the measurement errors, we should not just take the average of the squared deviations $(y(k) - \phi(k)^\top \hat{\theta})^2$ – this would be a biased (though asymptotically unbiased) estimator. It can be shown that an unbiased estimate for σ_ϵ^2 is obtained by

$$\hat{\sigma}_\epsilon^2 := \frac{1}{(N-d)} \sum_{k=1}^N (y(k) - \phi(k)^\top \hat{\theta}_{\mathrm{LS}})^2 = \frac{\|y_N - \Phi_N \hat{\theta}_{\mathrm{LS}}\|_2^2}{(N-d)}$$

Thus, our final formula for a good estimate $\hat{\Sigma}_{\hat{\theta}}$ of the true but unknown covariance $\mathrm{cov}(\theta_{\mathrm{LS}})$ is

$$\hat{\Sigma}_{\hat{\theta}} := \hat{\sigma}_\epsilon^2 (\Phi_N^\top \Phi_N)^{-1} = \frac{\|y_N - \Phi_N \hat{\theta}_{\mathrm{LS}}\|_2^2}{(N-d)} (\Phi_N^\top \Phi_N)^{-1}.$$

What we have now are two quantities, an estimate $\hat{\theta}_{\mathrm{LS}}$ of the true parameter value θ_0, as well as an estimate $\hat{\Sigma}_{\hat{\theta}}$ for the covariance matrix of this estimate. This knowledge helps us to make a strong statement about how probable it is that our our estimate is close to the true parameter value. Under the assumption that our linear model structure is correct and thus our estimator is unbiased, and the (slightly optimistic) assumption that our covariance estimate $\hat{\Sigma}_{\hat{\theta}}$ is equal to the true covariance $\Sigma_{\hat{\theta}}$ of the estimator $\hat{\theta}_{\mathrm{LS}}$, we can compute the probability that an uncertainty ellipsoid around the estimated $\hat{\theta}_{\mathrm{LS}}$ contains the (unknown) true parameter value θ_0.

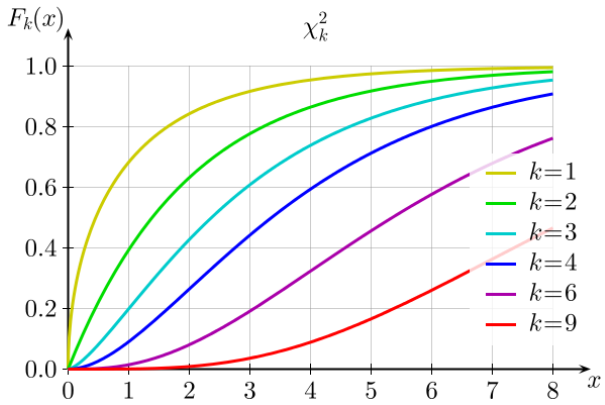


Figure 4.4: Cumulative density function $F(x, k)$ for the χ^2-distribution with k degrees of freedom (image: wikipedia).

In order to compute this probability, we need to use the cumulative density function (CDF) – i.e. the integral of the PDF – of the so called χ^2-distribution (Chi-squared) with $k := d$ degrees of freedom. We denote this CDF, which is illustrated for up to $k = 9$ degrees of freedom in Figure 4.4, by $F(x, k)$. This function tells us how probable it is that the square of a k-dimensional, normally distributed variable $X \sim \mathcal{N}(0, \mathbb{I})$ with zero mean and unit covariance has a value smaller than x, i.e.

$$P(\|X\|_2^2 \leq x) = F(x, k).$$

Using the fact that under the above assumptions, the random variable $X := \Sigma_{\hat{\theta}}^{-\frac{1}{2}}(\hat{\theta}_{\mathrm{LS}} - \theta_0)$ is normally distributed with zero mean and unit covariance, we thus know that for any positive x we have

$$P(\|\theta_0 - \hat{\theta}_{\mathrm{LS}}\|^2_{\Sigma^{-1}_{\hat{\theta}}} \leq x) = F(x, d).$$

We can give another interpretation to the same fact: the probability that the true value θ_0 is contained in the *confidence ellipsoid* $\mathcal{E}_x(\hat{\theta}_{\mathrm{LS}})$ defined by

$$\mathcal{E}_x(\hat{\theta}_{\mathrm{LS}}) := \{\theta \in \mathbb{R}^d \mid \|\theta - \hat{\theta}_{\mathrm{LS}}\|^2_{\Sigma^{-1}_{\hat{\theta}}} \leq x\}$$

is given by

$$P\left(\ \theta_0 \in \mathcal{E}_x(\hat{\theta}_{\mathrm{LS}})\ \right) = F(x, d).$$

Note that in this expression, it is the ellipsoid which is random, not the true, but unknown, value θ_0. We call the confidence ellipsoid for $x = 1$, i.e. the set

$$\mathcal{E}_1(\hat{\theta}_{\mathrm{LS}}) := \{\theta \in \mathbb{R}^d \mid \|\theta - \hat{\theta}_{\mathrm{LS}}\|^2_{\Sigma^{-1}_{\hat{\theta}}} \leq 1\}$$

the *one-sigma confidence ellipsoid*. The probability that the true value is contained in it decreases with increasing dimension $d = k$ of the parameter space and can be found in Figure 4.4 at $x = 1$.

Note that the variance for one single component of the parameter vector can be found as a diagonal entry in the covariance matrix, and that the probability that the true value of this single component is inside the one sigma interval around the estimated value is always 68.3%, independent of the parameter dimension d. This is due to the fact that each single component of $\hat{\theta}$ follows a one dimensional normal distribution.

For mathematical correctness, we have to note that we had to assume that the covariance matrix $\Sigma_{\hat{\theta}}$ is exactly known in order to make use of the χ^2-distribution. On the other hand, in practice, we can only use its estimate $\hat{\Sigma}_{\hat{\theta}}$ in the definition of the confidence ellipsoid. A refined analysis, which is beyond our ambitions, would need to take into account that also $\hat{\Sigma}_{\hat{\theta}}$ is a random variable, which implies that $X := \hat{\Sigma}_{\hat{\theta}}^{-\frac{1}{2}}(\hat{\theta}_{\mathrm{LS}} - \theta_0)$ follows a distribution which is similar to, but not equal to a standard normal distribution. For the practice of least squares estimation, however, the above characterization of confidence ellipsoids with the χ^2-distribution is accurate enough and can help us to assess the quality of an estimation result after a single experiment.

Chapter 5

Maximum Likelihood and Bayesian Estimation

In this chapter we are mostly concerned with models described by a possibly nonlinear function $M(\theta)$ that maps from $\theta \in \mathbb{R}^d$ into the space $\mathbb{R}^N$ of measurements, y_N. We sometimes drop the index N, i.e. write just $y \in \mathbb{R}^N$, and we sometimes denote the components of y by $y_1, \ldots, y_N$ instead of $y(1), \ldots, y(N)$. Also, the measurements are disrupted by noise, which we call $\epsilon \in \mathbb{R}^N$. Altogether, the prediction model for obtaining the measurements is

$$y = M(\theta) + \epsilon.$$

We will treat two related types of estimators, the maximum likelihood estimator and the Bayesian estimator, which both need nonlinear optimization solvers. We conclude the chapter with a fundamental inequality that every unbiased estimator needs to satisfy, the famous Cramer-Rao inequality which gives a lower bound on the covariance matrix.

5.1 Maximum Likelihood Estimation

Definition 11 (Likelihood) *The likelihood function $L(\theta)$ is a function of θ for given measurements y that describes how likely the measurements would have been if the parameter would have the value θ. It is defined as $L(\theta) = p(y|\theta)$, using the PDF of y for given θ.*

Definition 12 (Maximum-Likelihood Estimate) *The maximum-likelihood estimate of the unknown parameter θ is the parameter value that maximizes the likelihood function $L(\theta) = p(y|\theta)$.*

Assume $y_i = M_i(\overline{\theta}) + \epsilon_i$ with $\overline{\theta}$ the "true" parameter, and ϵ_i *Gaussian noise* with expectation value $\mathbb{E}(\epsilon_i) = 0$, $\mathbb{E}(\epsilon_i\, \epsilon_i) = \sigma_i^2$ and ϵ_i, ϵ_j independent for $i \neq j$. Then holds

$$p(y|\theta) = \prod_{i=1}^{N} p(y_i \mid \theta) \tag{5.1}$$

$$= C \prod_{i=1}^{N} \exp\left(\frac{-(y_i - M_i(\theta))^2}{2\sigma_i^2}\right) \tag{5.2}$$

with $C = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma_i^2}}$. Taking the logarithm of both sides gives

$$\log p(y|\theta) = \log(C) + \sum_{i=1}^{N} -\frac{(y_i - M_i(\theta))^2}{2\sigma_i^2} \tag{5.3}$$

with a constant C. Due to monotonicity of the logarithm holds that the argument maximizing $p(y|\theta)$ is given by

$$\begin{aligned} \arg\max_{\theta\in\mathbb{R}^d} p(y|\theta) &= \arg\min_{\theta\in\mathbb{R}^d} \; -\log(p(y|\theta)) && (5.4)\\ &= \arg\min_{\theta\in\mathbb{R}^d} \; \sum_{i=1}^{m} \frac{(y_i - M_i(\theta))^2}{2\sigma_i^2} && (5.5)\\ &= \arg\min_{\theta\in\mathbb{R}^d} \; \frac{1}{2}\,\|S^{-1}(y - M(\theta))\|_2^2 && (5.6) \end{aligned}$$

Thus, the least squares problem has a statistical interpretation. Note that due to the fact that we might have different standard deviations σ_i for different measurements y_i we need to scale both measurements and model functions in order to obtain an objective in the usual least squares form $\|\hat{y} - \hat{M}(\theta)\|_2^2$, as

$$\begin{aligned} \min_\theta \; \frac{1}{2}\sum_{i=1}^{N}\left(\frac{y_i - M_i(\theta)}{\sigma_i}\right)^2 &= \min_\theta \; \frac{1}{2}\,\|S^{-1}(y - M(\theta))\|_2^2 && (5.7)\\ &= \min_\theta \; \frac{1}{2}\,\|S^{-1}y - S^{-1}M(\theta)\|_2^2 && (5.8) \end{aligned}$$

$$\text{with } S = \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_N \end{bmatrix}.$$

Statistical Interpretation of Regularization terms: Note that a regularization term like $\alpha\|\theta - \bar{\theta}\|_2^2$ that is added to the objective can be interpreted as a "pseudo measurement" $\bar{\theta}$ of the parameter value θ, which includes a statistical assumption: the smaller α, the larger we implicitly assume the standard deviation of this pseudo-measurement. As the data of a regularization term are usually given before the actual measurements, regularization is also often interpreted as "a priori knowledge". Note that not only the Euclidean norm with one scalar weighting α can be chosen, but many other forms of regularization are possible, e.g. terms of the form $\|A(\theta - \bar{\theta})\|_2^2$ with some matrix A.

5.1.1 L_1-Estimation

Instead of using $\|.\|_2^2$, i.e. the L_2-norm in the fitting problem, we might alternatively use $\|.\|_1$, i.e., the L_1-norm. This gives rise to the so called L_1-estimation problem:

$$\min_\theta \|y - M(\theta)\|_1 = \min_\theta \; \sum_{i=1}^{N} |y_i - M_i(\theta)| \tag{5.9}$$

Like the L_2-estimation problem, also the L_1-estimation problem can be interpreted statistically as a maximum-likelihood estimate. However, in the L_1-case, the measurement errors are assumed to follow a Laplace distribution instead of a Gaussian.

An interesting observation is that the optimal L_1-fit of a constant θ to a sample of different scalar values $y_1, \ldots, y_N$ just gives the median of this sample, i.e.

$$\arg\min_{\theta\in\mathbb{R}} \; \sum_{i=1}^{N} |y_i - \theta| \quad = \quad \text{median of } \{y_1, \ldots, y_N\}. \tag{5.10}$$

Remember that the same problem with the L_2-norm gave the average of y_i. Generally speaking, the median is less sensitive to outliers than the average, and a detailed analysis shows that the solution to general L_1-estimation problems is also less sensitive to a few outliers. Therefore, L_1-estimation is sometimes also called "robust" parameter estimation.

5.2 Bayesian Estimation and the Maximum A Posteriori Estimate

The Maximum Likelihood estimation solves the problem $\arg\max_{\theta\in\mathbb{R}^d} p(y_N|\theta)$. However, one might instead want to maximise the probability of θ given the measurements y_N, i.e. $p(\theta|y_N)$. This last probability it is not known beforehand, but Bayes' rule can be applied to obtain a computable expression:

$$p(\theta|y_N) = \frac{p(y_N,\theta)}{p(y_N)} = \frac{p(y_N|\theta)\cdot p(\theta)}{p(y_N)} \tag{5.11}$$

From Equation (5.11) it can be seen that the probability to maximise $p(\theta|y_N)$ depends on 3 factors:

1. A constant term, $p(y_N)$, that need not be regarded when optimizing over θ.
2. The same term which was maximised for Maximum Likelihood, $p(y_N|\theta)$.
3. An extra term $p(\theta)$ that introduces the need for a-priory knowledge, or an assumption, on the value of θ.

The systematic approach to incorporate the prior knowledge is always the same: to incorporate an extra minimisation term representing the negative log-probability of this prior knowledge (as it was done with the ML term). This resulting estimate is known as the Maximum a Posteriori Estimate (MAP) and represented by:

$$\hat{\theta}_{\text{MAP}} = \arg\min_{\theta\in\mathbb{R}}\{-\log(p(y_N|\theta)) - \log(p(\theta))\} \tag{5.12}$$

The problem with this method is that it leads to bias when the prior knowledge on θ is not accurate (which it usually is not, otherwise we would not need to estimate θ).

5.2.1 MAP example: Regularised Least Squares

Assumptions:

1. The measurements $y_N \in \mathbb{R}^N$ have i.i.d. noise, the model is linear $M(\theta) = \Phi_N \cdot \theta$, and $\theta \in \mathbb{R}$.
2. Prior knowledge on the parameter θ is given: $\theta = \bar{\theta} \pm \sigma_\theta$, where $\bar{\theta}$ is the a-priori most probable value σ_θ its standard deviation.

With the first assumption, it is obvious that the minimisation problem will have a term that will minimize a standard least squares problem, and with the second assumption a second minimization term proportional to the difference $\theta - \bar{\theta}$ and representing the probability of the noise σ_θ must be included. The final expression is:

$$\hat{\theta}_{\text{MAP}} = \arg\min_{\theta\in\mathbb{R}} \frac{1}{2}\cdot\frac{1}{\sigma_\epsilon^2}\cdot\|y_N - \Phi_N\cdot\theta\|_2^2 + \frac{1}{2}\cdot\frac{1}{\sigma_\theta^2}\cdot(\theta-\bar{\theta})^2 \tag{5.13}$$

5.2.2 Differences between ML and MAP estimation

MAP and ML are very related. On the one hand, MAP could be considered as a generalisation of ML, where in the case of ML the weight in the prior-knowledge is zero. On the other hand, ML can be regarded as a special case of MAP with "pseudo-measurements" $\bar{\theta}$ of θ.

5.3 Recursive Linear Least Squares

In all previous sections, we have introduced several optimisation methods which provide a useful tool for parameter estimation. Nevertheless, they have a main disadvantage: they can only be used offline, i.e. they optimise and fit a finite number N of measurements to a given model.

In real-life applications, problems where the flow of data is continuous are not unusual. Here, online parameter estimators must be used, i.e. the number of measurements is infinite $[y(1), y(2), \dots, y(N), y(N+1), \dots]$.

The main aim of this section is to provide a tool that can minimise the problem:

$$\hat{\theta}_{\text{ML}}(N) = \arg\min_{\theta\in\mathbb{R}} \frac{1}{2}\|y_N - \Phi_N\cdot\theta\|_2^2 \tag{5.14}$$

for increasing N but without increasing the computational time. This task is fairly complex since $\Phi_N \in \mathbb{R}^{N\times d}$ becomes infinitely long when $N \to \infty$. In order to achieve the goal stated above, the main idea is to find a recursive equation which allows the calculation of $\hat{\theta}_{\mathrm{ML}}(N+1)$ using the previous optimal estimator $\hat{\theta}_{\mathrm{ML}}(N)$, the new measurement $y(N+1)$ and the new regressor $\varphi(N+1)$. We want to do so without the need of storing and/or using in the calculation the previous measurement or regressor values. We make the following assumptions:

1. The sequence of measurements $[y(1), y(2), \ldots, y(N), y(N+1), \ldots]$ is infinite and updated with a new value at each time step.

2. The sequence of regressors $[\varphi(1), \varphi(2), \ldots, \varphi(N), \varphi(N+1), \ldots]$ is infinite and updated with a new value at each time step

3. There is only measurement noise and it is i.i.d. and Gaussian.

Lemma 2 *Let $y_N \in \mathbb{R}^N$ be a set of N measurements used on a Least Squares problem, $\Phi_N \in \mathbb{R}^{N\times d}$ the regressor of the problem, and $\theta \in \mathbb{R}^d$ the estimator, then:*

$$\frac{1}{2}\cdot\|y_N - \Phi_N\cdot\theta\|_2^2 = \frac{1}{2}\cdot\|\theta - \hat{\theta}_{\mathrm{ML}}(N)\|^2_{\Phi_N^\top\cdot\Phi_N} + \mathrm{const} \tag{5.15}$$

Proof: The linear least squares equation we are solving is $\Phi^\top\cdot\Phi\cdot\hat{\theta} = \Phi^\top\cdot y$. Remember that this equation only has a unique solution if the inverse of $\Phi^\top\cdot\Phi$ exists, which we assume here. In the following, we will use the fact that $\Phi_N^\top\cdot\Phi_N\cdot\hat{\theta}_{\mathrm{ML}}(N) = \Phi_N^\top\cdot y_N$ holds, after N measurements have been made.

We can expand the minimisation term of the LS problem as follows:

$$\begin{aligned}
&\frac{1}{2}\cdot\|y_N - \Phi_N\cdot\theta\|_2^2 = \\
&= \frac{1}{2}\cdot\|y_N\|_2^2 + \frac{1}{2}\cdot\theta^\top\cdot\Phi_N^\top\cdot\Phi_N\cdot\theta - \theta^\top\cdot\Phi_N^\top\cdot y_N = \\
&= \frac{1}{2}\cdot\|y_N\|_2^2 + \frac{1}{2}\cdot\theta^\top\cdot\Phi_N^\top\cdot\Phi_N\cdot\theta - \theta^\top\cdot\Phi_N^\top\cdot\Phi_N\cdot\hat{\theta}_{\mathrm{ML}} = \\
&= \frac{1}{2}\cdot\|y_N\|_2^2 - \frac{1}{2}\cdot\hat{\theta}_{\mathrm{ML}}^\top\cdot\Phi_N^\top\cdot\Phi_N\cdot\hat{\theta}_{\mathrm{ML}} + \frac{1}{2}\cdot(\theta-\hat{\theta}_{\mathrm{ML}})^\top\cdot\Phi_N^\top\cdot\Phi_N\cdot(\theta-\hat{\theta}_{\mathrm{ML}}) = \\
&= \mathrm{const} + \frac{1}{2}\cdot\|\theta - \hat{\theta}_{\mathrm{ML}}(N)\|^2_{\Phi_N^\top\cdot\Phi_N}
\end{aligned} \tag{5.16}$$

Theorem 7 (Recursive Least Squares) *Let $\hat{\theta}_{\mathrm{ML}}(N+1)$ be the optimal estimator to the problem* (5.14)*, then $\hat{\theta}_{\mathrm{ML}}(N+1)$ can be calculated by a recursive algorithm consisting of two steps that are computed with every new measurement N and which are defined by the equations below:*

$$Q_{N+1} = Q_N + \varphi(N+1)\cdot\varphi(N+1)^\top \tag{5.17}$$

$$\hat{\theta}_{\mathrm{ML}}(N+1) = \hat{\theta}_{\mathrm{ML}}(N) + Q_{N+1}^{-1}\cdot\varphi(N+1)\cdot[y(N+1) - \varphi(N+1)^\top\cdot\hat{\theta}_{\mathrm{ML}}(N)] \tag{5.18}$$

Remark: The first term of Equation (5.18) represents the best prior guess, and the second term the so called "innovation".

Proof: Let Q_N be $\Phi_N^\top\Phi_N \in \mathbb{R}^{d\times d}$, then Q_N follows a recursion:

$$\begin{aligned}
Q_{N+1} = \Phi_{N+1}^\top\cdot\Phi_{N+1} &= \begin{bmatrix}\Phi_N\\ \varphi(N+1)^\top\end{bmatrix}^\top\cdot\begin{bmatrix}\Phi_N\\ \varphi(N+1)^\top\end{bmatrix} = \\
&= Q_N + \varphi(N+1)\cdot\varphi(N+1)^\top
\end{aligned} \tag{5.19}$$

Let $\hat{\theta}_{\mathrm{ML}}(N+1)$ be the optimal estimator given $N+1$ measurements, then $\hat{\theta}_{\mathrm{ML}}(N+1)$ can be expressed in a recursive manner:

$$\begin{aligned}
\hat{\theta}_{\mathrm{ML}}(N+1) &= (\Phi_{N+1}^{\top} \cdot \Phi_{N+1})^{-1} \cdot \Phi_{N+1}^{\top} \cdot y_{N+1} \\
&= Q_{N+1}^{-1} \cdot (\Phi_{N+1}^{\top} \cdot y_{N+1}) = Q_{N+1}^{-1} \cdot \begin{bmatrix} \Phi_N \\ \varphi(N+1)^{\top} \end{bmatrix}^{\top} \cdot \begin{bmatrix} y_N \\ y(N+1) \end{bmatrix} \\
&= Q_{N+1}^{-1} \cdot (\Phi_N^{\top} \cdot y_N + \varphi(N+1) \cdot y(N+1)) \\
&= Q_{N+1}^{-1} \cdot (\Phi_N^{\top} \cdot \Phi_N \cdot \hat{\theta}_{\mathrm{ML}}(N) + \varphi(N+1) \cdot y(N+1)) \\
&= Q_{N+1}^{-1} \cdot ([\Phi_N^{\top} \cdot \Phi_N + \varphi(N+1) \cdot \varphi(N+1)^{\top}] \cdot \hat{\theta}_{\mathrm{ML}}(N) \\
&\quad - \varphi(N+1) \cdot \varphi(N+1)^{\top} \cdot \hat{\theta}_{\mathrm{ML}}(N) + \varphi(N+1) \cdot y(N+1)) \\
&= Q_{N+1}^{-1} \cdot Q_{N+1} \cdot \hat{\theta}_{\mathrm{ML}}(N) + Q_{N+1}^{-1} \cdot (\varphi(N+1) \cdot [y(N+1) - \varphi(N+1)^{\top} \cdot \hat{\theta}_{\mathrm{ML}}(N)]) \\
&= \hat{\theta}_{\mathrm{ML}}(N) + Q_{N+1}^{-1} \cdot \varphi(N+1) \cdot [y(N+1) - \varphi(N+1)^{\top} \cdot \hat{\theta}_{\mathrm{ML}}(N)]
\end{aligned} \tag{5.20}$$

It is an interesting observation that the RLS algorithm solves in each step the following minimisation problem:

$$\begin{aligned}
\hat{\theta}_{\mathrm{ML}}(N+1) &= \arg\min_{\theta \in \mathbb{R}} \frac{1}{2} \|y_{N+1} - \Phi_{N+1} \cdot \theta\|_2^2 = \\
&= \arg\min_{\theta \in \mathbb{R}^d} \left(\frac{1}{2} \cdot \|\theta - \hat{\theta}_{\mathrm{ML}}(N)\|_{Q_N}^2 + \frac{1}{2} \cdot \|y(N+1) - \varphi(N+1)^{\top} \cdot \theta\|_2^2 \right)
\end{aligned} \tag{5.21}$$

5.3.1 Initialisation and implementation considerations of RLS

In practise, two fundamental questions arises when implementing a RLS algorithm:

1. How should Q_0 be initialised? It is important to avoid singularity.
   Answer: considering the way that Q_N enters Equation (5.21), it is clear that Q_N is in a way the inverse of the covariance of the θ parameter, $Q_N \approx \Sigma_{\hat{\theta}_{\mathrm{ML}}(N)}^{-1}$.

   Moreover, if some prior knowledge on θ can be assumed, for example, $\theta \sim \mathbb{N}(\theta_0, Q_0^{-1})$, this knowledge can be included as a regularisation term of the form $\|\theta\|_{Q_0}^2$, where Q_0 is small and positive definite. This regularisation with a positive definite matrix Q_0 leads to consistently non-singular Q_N, because adding successive terms would only increase the eigenvalues of the matrix, keeping it positive definite and thus non-singular.

2. How to avoid that $Q_N \to \infty$ when $N \to \infty$? The higher the Q_N the less influence will have the new measurements.

   The solution of this problem is easier than the previous one. The only thing that is necessary is to down-weight past information using a "forgetting factor" α. The main idea is that α multiplies Q_N in the equation of Q_{N+1}, so that past measurements have less weight than the most recent ones, and on top of that, Q_N does not increase to infinity (if the new contributions $\phi(N+1)\phi(N+1)^{\top}$ are bounded).

These two implementations can be seen on the set of equations below. Thes equations re-write Equation (5.21) as:

$$\hat{\theta}_{\mathrm{ML}}(N+1) = \arg\min_{\theta \in \mathbb{R}^d} \left(\alpha \cdot \frac{1}{2} \cdot \|\theta - \hat{\theta}_{\mathrm{ML}}(N)\|_{Q_N}^2 + \frac{1}{2} \cdot \|y(N+1) - \varphi(N+1)^{\top} \cdot \theta\|_2^2 \right) \tag{5.22}$$

In the recursive update this translates to:

$$\begin{aligned}
Q_0 \quad \text{given,} \quad \text{and} \quad \hat{\theta}_{\mathrm{ML}}(0) \quad \text{given,} \\
Q_{N+1} &= \alpha \cdot Q_N + \varphi(N+1) \cdot \varphi(N+1)^{\top}, \\
\hat{\theta}_{\mathrm{ML}}(N+1) &= \hat{\theta}_{\mathrm{ML}}(N) + Q_{N+1}^{-1} \cdot \varphi(N+1) \cdot [y(N+1) - \varphi(N+1)^{\top} \cdot \hat{\theta}_{\mathrm{ML}}(N)]
\end{aligned} \tag{5.23}$$

5.3.2 Example of RLS: estimate position of a robot in a plane

Imagine that we have a moving robot in a 2D plane whose position is given as the cartesian coordinates $Z = [x, y]$. Given a continuous flow of position measurements $[(x_1, y_1), \ldots, (x_N, y_N), (x_{N+1}, y_{N+1}), \ldots]$, the goal is to estimate the position of the robot at every time point t_{N+1} with every new set of noisy measurements (x_{N+1}, y_{N+1}), but considering all the past measurements $[(x_1, y_1), \ldots, (x_N, y_N)]$ and without increasing the computational time with increasing number of measurements. In figure 5.1 we show a set of measured data for an example system with sampling time $\nabla T = 0.1282$:

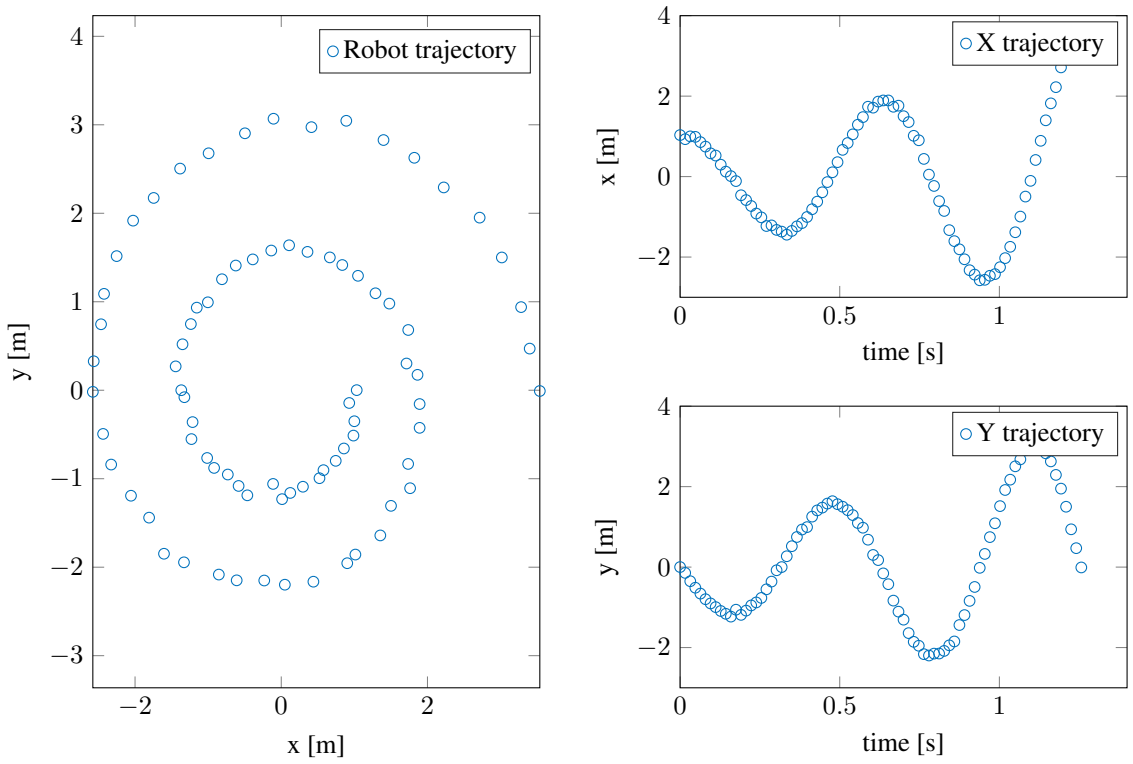


Figure 5.1: Robot movement in a 2D plane in space and time

Due to a lack of information on the dynamics of the robot as well as on the controls, we propose a fourth order polynomial as the local model for the position in the x-axis $x(t_k) = a_0 + a_1 \cdot t_k + a_2 \cdot t_k^2 + a_3 \cdot t_k^3 + a_4 \cdot t_k^4$, and another polynomial for the position in the y-axis $y(t_k) = b_0 + b_1 \cdot t_k + b_2 \cdot t_k^2 + b_3 \cdot t_k^3 + b_4 \cdot t_k^4$. The estimation problem is linear in the parameters $\theta = (a_0, a_1, a_2, a_3, a_4, b_0, b_1, b_2, b_3, b_4)$, thus we could in principle use both Linear Least Squares or Recursive Least Square for estimation purposes.

If we use LLS, not only the computational time would increase with every new measurement, but since the proposed models $x(t_k)$ and $y(t_k)$ are expected to be valid only locally, increasing the number of measurements and giving the same weight to every measurement leads to a very poor estimation. Figure 5.2 shows the LLS estimation using the total set of measurements.

On the other hand, in RLS we can include a forgetting factor α to indicate that the polynomial models are only local approximations of the movement, and therefore, that the last measurement values are much more important. Figure 5.3 represents this estimation using RLS, it can be seen how the estimation now is quite similar to the original measured data. Furthermore, we can see how the estimated local polynomial at each time interval approximates the trajectory quite well in a neighbourhood of the last measurement point, and how they deviate for farther points.

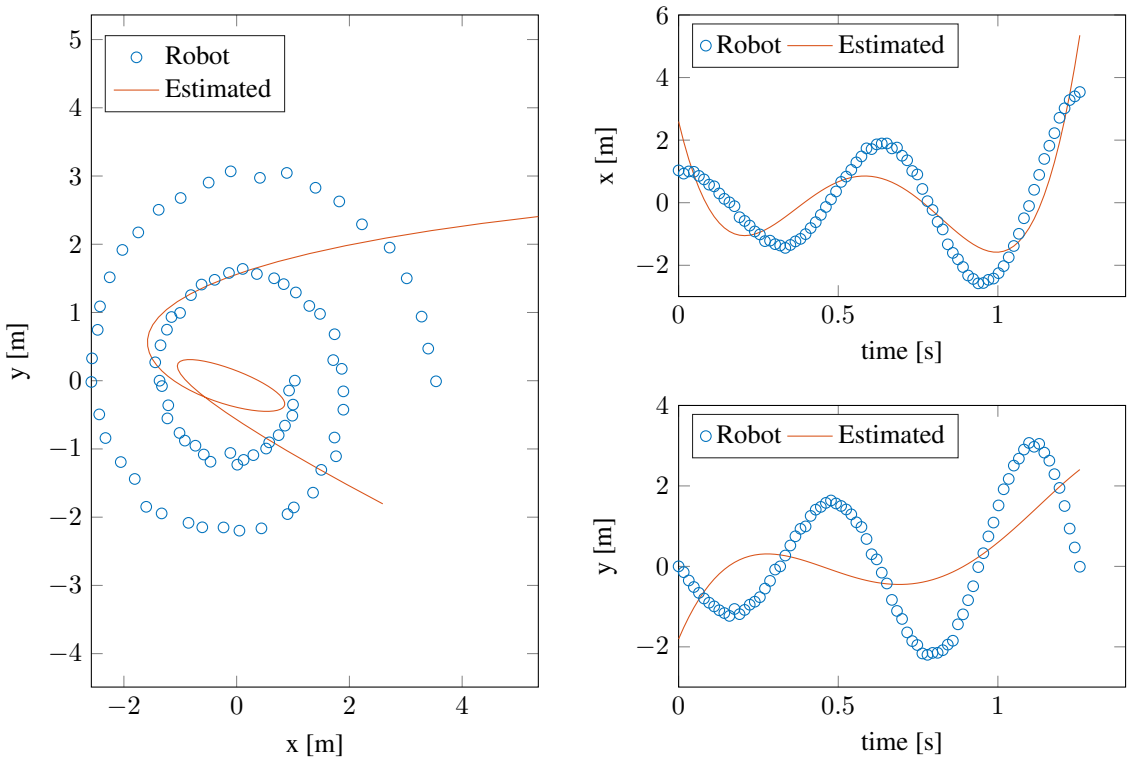


Figure 5.2: Robot position estimation of the robot using LLS (without downweighting)

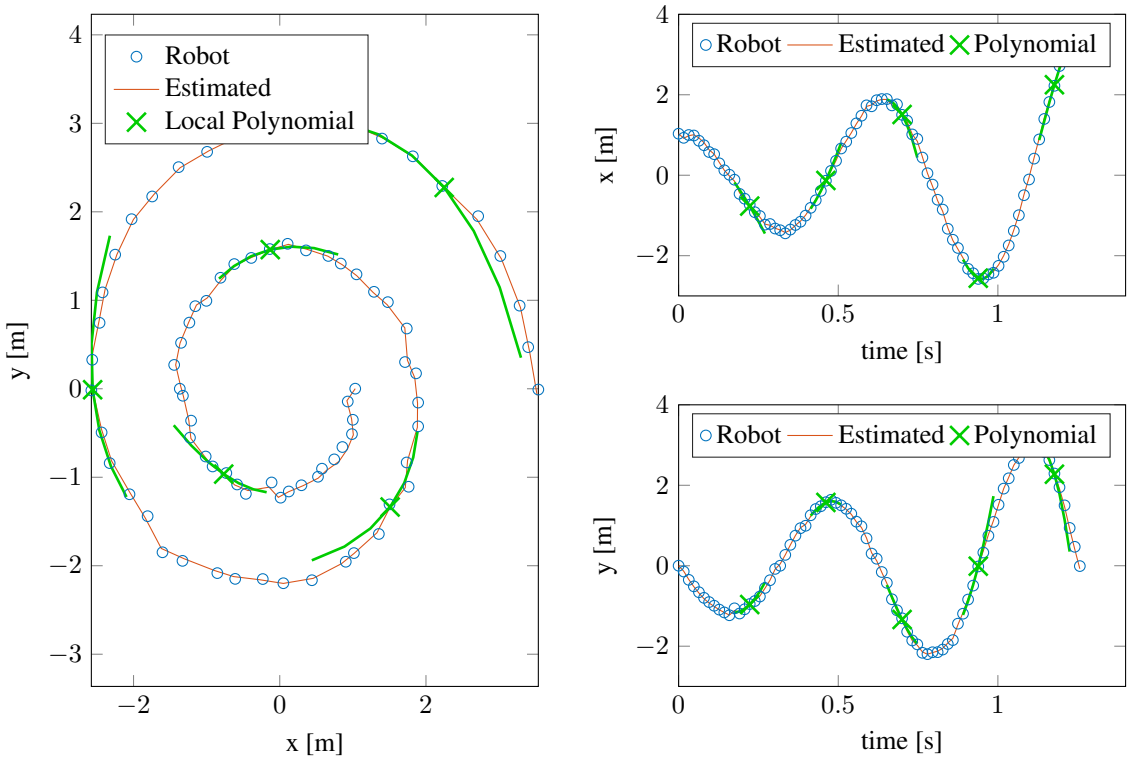


Figure 5.3: Robot position estimation of the robot using RLS (with downweighting)

5.4 Cramer-Rao-Inequality

Theorem 8 (Cramer-Rao-Inequality: lower bound on the covariance of an unbiased estimator) *Let $p(y_N|\theta)$ be the probability density function of obtaining the measurements y_N given the parameters θ (the one to be maximized in a Maximum Likelihood problem). Furthermore, assume that the probability density function $p(y_N|\theta_0)$ for the true θ_0 is known. Then, any unbiased estimator $\hat{\theta}(y_N)$ is a random variable due to its dependence on the random variable y_N, and it has a covariance matrix $\Sigma_{\hat{\theta}} := \mathrm{cov}(\hat{\theta}(y_N))$ that is bigger than the inverse of the so-called **Fisher information matrix** M:*

$$\Sigma_{\hat{\theta}} \succeq M^{-1} \tag{5.24}$$

Definition 13 (Fisher information matrix M) *Let $L(\theta, y_N) := -\log p(y_N|\theta)$ be the negative log likelihood function (which is minimised in a general Maximum Likelihood problem). Then, the Fisher information matrix associated with such a problem is defined as:*

$$M = \mathbb{E}\{\nabla_\theta^2 L(\theta_0, y_N)\} = \int_{y_N} \nabla_\theta^2 L(\theta_0, y_N) \cdot p(y_N|\theta_0) \cdot \mathrm{d}y_N \tag{5.25}$$

We will not prove the Cramer-Rao theorem here but refer e.g. to the book of Ljung [Lju99] for the proof. We can however apply the theorem to the case that we are minimising a linear model with gaussian noise $\mathcal{N}(0, \Sigma)$. Then the Cramer-Rao-Inequality states that

$$\Sigma_{\hat{\theta}} \succeq M^{-1} = (\Phi_N^\top \cdot \Sigma^{-1} \cdot \Phi)^{-1} \tag{5.26}$$

To see this, we use the fact that in the case of a linear model with Gaussian noise, $L(\theta, y_N)$ becomes:

$$L(\theta, y_N) = \frac{1}{2} \cdot (\Phi_N \cdot \theta - y_N)^\top \cdot \Sigma^{-1} \cdot (\Phi_N \cdot \theta - y_N) \tag{5.27}$$

Its Hessian can be easily calculated as:

$$\nabla_\theta^2 L(\theta, y_N) = \Phi_N^\top \cdot \Sigma^{-1} \cdot \Phi_N \tag{5.28}$$

where it is easily seen that the Hessian is constant and independent from θ and y_N, thus its expectation, which equals the Fisher information matrix, is identical to this constant value:

$$M = \mathbb{E}\{\nabla_\theta^2 L(\theta, y_N)\} = \nabla_\theta^2 L(\theta, y_N) = \Phi_N^\top \cdot \Sigma^{-1} \cdot \Phi_N \tag{5.29}$$

This application of the Cramer-Rao inequality confirms the result mentioned previously on Section 4.5.2, where the smallest covariance for a weighted least squares estimator was obtained when choosing the correct weighting matrix $W = \Sigma^{-1}$.

An interesting result which we will not formalise and prove here is that the covariance matrix of the ML-estimator for i.i.d. measurements $y(1), \ldots, y(N)$ approaches, for $N \to \infty$, the Cramer-Rao lower bound, i.e. the inverse of the corresponding Fisher information matrix $M_{\mathrm{ML}}(N)$ and the distribution of $\hat{\theta}_{\mathrm{ML}}(N)$ becomes asymptotically normal, i.e. $\hat{\theta}_{\mathrm{ML}}(N) \sim \mathcal{N}(\theta_0, M_{\mathrm{ML}}(N)^{-1})$.

For models which are more general than a linear model with Gaussian noise, the Fisher information matrix M can not be easily computed. The reason for that is that in general the true value of the parameter θ_0 is unknown, and therefore, the Hessian matrix $\nabla_\theta^2 L(\theta_0, y_N)$ and the probability $p(y_N|\theta_0)$ have to be approximated using the estimated parameter $\hat{\theta}$. Furthermore, the integral in (5.25) is difficult to compute in practice.

An heuristic approximation that can be used to compute an estimate of the covariance of a ML-estimate is to first assume that $\Sigma_{\bar{\theta}} \approx M^{-1}$, i.e. the Cramer Rao limit is reached. Second, one could approximate the Fisher information matrix by $M \approx \nabla_\theta^2 L(\hat{\theta}, y_N)$ using the actual measurements y_N

5.5 Practical solution of the Nonlinear Least Squares Problem

When we formulate and solve a nonlinear least squares problem, we need to use a numerical optimization routine to find the maximum likelihood estimate. In order to do this, we first scale the vector of model-measurement-mismatch residuals $y - M(\theta)$ by using a – usually diagonal – guess Σ_ϵ of the covariance matrix of the noise, in

order to obtain the scaled residual vector $R(\theta) := \Sigma_\epsilon^{-\frac{1}{2}} (M(\theta) - y)$ such that the maximum likelihood estimate $\hat{\theta} = \theta^*$ is obtained by the solution of the optimization problem

$$\theta^* = \arg\min_\theta \underbrace{\frac{1}{2}\|R(\theta)\|_2^2}_{=:f(\theta)} \tag{5.30}$$

Note that the *residual function* R maps from $\mathbb{R}^d$ to $\mathbb{R}^N$, and that most solution algorithms require the user to pass this function $R(\theta)$ – and not the objective function $f(\theta) = \frac{1}{2}\|R(\theta)\|_2^2$ – to the solver. We want to answer three questions in this section:

- How do we solve the nonlinear least squares optimization problem (5.30) in practice?
- How can we obtain an estimate of the covariance matrix of the parameter estimate?
- How can we assess if the modelling assumptions, in particular on the noise, were correct?

We will answer the three questions in the following three subsections.

5.5.1 The Gauss-Newton Algorithm

In practice, nonlinear least squares problems are solved with specialized nonlinear optimization routines like MATLAB's `lsqnonlin`, which expect the user to provide an initial guess for the parameter θ – which we call $\theta_{[0]}$ in this section – and a pointer to the function $R : \mathbb{R}^d \to \mathbb{R}^N$. Most available algorithms can only guarantee to find local minima. Starting at the *initial guess* $\theta_{[0]}$, they generate a sequence of iterates that we call $\theta_{[0]}, \theta_{[1]}, \theta_{[2]}, \ldots$ Note that each $\theta_{[k]}$ is a vector in the space $\mathbb{R}^d$ and that we use rectangular parentheses $\cdot_{[k]}$ in the index in order to distinguish it from the component index. A basic requirement of all algorithms is that they should converge to a point θ^* that satisfies at least the first order necessary optimality condition, i.e., to a point that satisfies $\nabla f(\theta^*) = 0$. Most algorithms are variants of the so-called Gauss-Newton method described next, though often these variants come under very different names such as "Levenberg-Marquardt Algorithm" or "Trust-Region-Reflective Method".

Idea: Because we know very well how to solve linear least squares problems and because all nonlinear functions can locally be approximated by their first order Taylor series, a straightforward idea would be to solve a linear least squares problem based on the linearization at a solution guess $\theta_{[k]}$ in order to obtain a better solution guess $\theta_{[k+1]}$. More concretely, for any solution guess $\theta_{[k]}$ we have that $R(\theta) = R(\theta_{[k]}) + \frac{\partial R}{\partial \theta}(\theta_{[k]})(\theta - \theta_{[k]}) + O(\|\theta - \theta_{[k]}\|_2^2)$, and if we use the first order Taylor series to formulate a linear least squares problem in order to find $\theta_{[k+1]}$, we obtain the expression

$$\begin{aligned}
\theta_{[k+1]} &= \arg\min_\theta \frac{1}{2}\Big\| R(\theta_{[k]}) + \underbrace{\frac{\partial R}{\partial \theta}(\theta_{[k]})}_{=:J(\theta_{[k]})}(\theta - \theta_{[k]})\Big\|_2^2 && (5.31)\\
&= \arg\min_\theta \frac{1}{2}\Big\| -J(\theta_{[k]})\,\theta_{[k]} + R(\theta_{[k]}) \;+\; J(\theta_{[k]})\,\theta\,\Big\|_2^2 && (5.32)\\
&= (J(\theta_{[k]})^\top J(\theta_{[k]}))^{-1} J(\theta_{[k]})^\top\,(J(\theta_{[k]})\,\theta_{[k]} - R(\theta_{[k]})) && (5.33)\\
&= \theta_{[k]} - (J(\theta_{[k]})^\top J(\theta_{[k]}))^{-1} J(\theta_{[k]})^\top R(\theta_{[k]}) && (5.34)\\
&= \theta_{[k]} - J(\theta_{[k]})^+ R(\theta_{[k]}) && (5.35)
\end{aligned}$$

Note that the iteration above is only well defined if the Jacobian matrix $J(\theta_{[k]})$ is of full rank, but that in practical implementations, small algorithm modifications ensure that each iteration is well defined. With the above expression, we have already defined the basic Gauss-Newton algorithm. One can show that – if it converges – the Gauss-Newton algorithm converges linearly to a stationary point θ^* with $\nabla f(\theta^*) = 0$, but a proof of this result is beyond our interest here.

However, in order to understand the algorithm a bit better and to see at least why the algorithm does not move away from a stationary point, it is useful to look at explicit expressions for the derivatives of the objective function

f, which are given by

$$f(\theta) = \frac{1}{2}\|R(\theta)\|_2^2 = \frac{1}{2}\sum_{i=1}^{N} R_i(\theta)^2 \tag{5.36}$$

$$\nabla f(\theta) = \sum_{i=1}^{N} \nabla R_i(\theta) R_i(\theta) = J(\theta)^\top R(\theta) \tag{5.37}$$

$$\nabla^2 f(\theta) = \underbrace{J(\theta)^\top J(\theta)}_{=:B_{\mathrm{GN}}(\theta)} + \sum_{i=1}^{N} \nabla^2 R_i(\theta) R_i(\theta) \tag{5.38}$$

Using some of the above expressions, the iterations of the Gauss-Newton algorithm can be written as

$$\theta_{[k+1]} = \theta_{[k]} - B_{\mathrm{GN}}(\theta_{[k]})^{-1} \nabla f(\theta_{[k]})$$

It can be seen, as expected from an optimization algorithm, that the algorithm would not move away from a stationary point with $\nabla f(\theta_{[k]}) = 0$. But the inverted matrix $B_{\mathrm{GN}}(\theta_{[k]})^{-1}$ in front of the gradient could also be chosen differently. If one would choose the inverse of the exact Hessian matrix, $\nabla^2 f(\theta_{[k]})^{-1}$, one would obtain the so-called *Newton method*; different choices of Hessian approximation give rise to different members in the class of so-called *Newton-type optimization* methods, which comprises the family of Gauss-Newton methods. The matrix $B_{\mathrm{GN}}(\theta)$ is called the *Gauss-Newton Hessian approximation*. Note that it is a positive semidefinite matrix, but not necessarily positive definite. In variants of the Gauss-Newton method, for example in the Levenberg-Marquardt algorithm, the Gauss-Newton Hessian approximation is first computed but then modified in the actual step computation, for example to ensure that the Hessian approximation becomes positive definite or that the steps remain small enough for the first order Taylor series to remain a good approximation of the actual function.

Independent of which algorithm is used, at the end of the call of the optimization solver, the solver will return a value θ^* that is an approximate local minimizer of $f(\theta)$. We will use it as the maximum-likelihood estimate, i.e., set $\hat{\theta} := \theta^*$. Interestingly, it is useful to also obtain from the algorithm – or to recompute afterwards – the inverse Gauss-Newton Hessian $B_{\mathrm{GN}}(\theta^*)^{-1}$, because it can serve as approximation of the covariance matrix of this estimate.

5.5.2 Estimating the Covariance Matrix and Extracting its Relevant Entries

The easiest way to obtain a rough estimate of the covariance matrix $\Sigma_{\hat{\theta}}$ of the parameter estimate θ^* would be to assume that the linearization of R at the solution is the correct model, and that all the statistical assumptions we made in the formulation of the function R were correct, i.e., that we indeed had Gaussian noise with covariance matrix Σ_ϵ). Following the linear least squares analysis, we could then directly use $B_{\mathrm{GN}}(\theta^*)^{-1}$ as parameter covariance matrix. Note that, due to the scaling in the expression $R(\theta) = \Sigma_\epsilon^{-\frac{1}{2}}(M(\theta) - y)$, our assumption would be that the scale of the residual vector components is not only unitless, but also in the order of one. For this reason, the optimal squared residual value is expected to be in the order of N. More precisely, due the fact that we have a d-dimensional vector fitting the data and minimizing the residual, we expect $\|R(\theta^*)\|_2^2 \approx N - d$, as already discussed in Section 4.7. In practice, however, we might have made an error in estimating the absolute size of the noise covariance Σ_ϵ, such that the size of the squared residual $\|R(\theta^*)\|_2^2$ can be different from $N - d$. Because this is easy to correct, we follow the reasoning of Section 4.7, and in practice use the parameter covariance estimate

$$\Sigma_{\hat{\theta}} := \frac{\|R(\theta^*)\|_2^2}{N - d} \left(J(\theta^*)^\top J(\theta^*)\right)^{-1}$$

If one wants to express the result of the parameter estimation, one often only uses the diagonal entries from this matrix, which contain, for $i = 1, \ldots, d$, the variances σ_i^2 of the respective parameter components $\hat{\theta}_i$, as seen in the following detailed matrix expression:

$$\Sigma_{\hat{\theta}} = \begin{bmatrix} \sigma_1^2 & * & * & * \\ * & \sigma_2^2 & * & * \\ * & * & \ddots & * \\ * & * & * & \sigma_d^2 \end{bmatrix}$$

Taking the square root of the variances yields the standard deviations, such that the final result of the whole parameter estimation procedure could be presented by only $2d$ numbers in the form

$$\theta_i = \theta_i^* \pm \sqrt{\sigma_i^2}, \quad \text{for} \quad i = 1, \ldots, d$$

5.5.3 Checking the Optimal Residual Vector

Because all analysis in this section is based on the measurement data that we use for the estimation, we will not be able to completely answer the question of model validation, namely if our model is able to make valid predictions for new situations. For model validation, we would need another set of measurement data that were not involved in the estimation procedure, for example a new experiment that is performed after the estimation procedure is finished, or a previously conducted experiment that was kept secret during the parameter estimation procedure and was just reserved for model validation.

However, what we can do with the existing data of the single experiment that we use for parameter estimation, is to look at the residual values $R_i(\theta^*)$ for $i = 1, \ldots, N$. If we plot them as a function of i, they should look like a sequence of random numbers. In order to check this in more details, one typically creates and plots a histogram of the residual values $R_i(\theta^*)$. If the histogram looks like the histogram of a zero mean Gaussian with unit variance, the model assumptions on the system and noise are likely to be correct. If not, some part of the modelling assumptions was probably wrong. One should then think hard and change the system or noise model and restart the parameter estimation procedure, based on the same data, but on a different model.

Part III

Dynamic System Modelling and Identification

Chapter 6

Dynamic Systems in a Nutshell

In this lecture, our major aim is to model and identify *dynamic systems*, i.e. processes that are evolving with time. These systems can be characterized by *states* x and parameters p that allow us to predict the future behavior of the system. If the state and the parameters are not known, we first need to *estimate* them based on the available measurement information. Often, a dynamic system can be controlled by a suitable choice of inputs that we denote as *controls* u in this script, and the ultimate purpose of modelling and system identification is to be able to design and test control strategies.

As an example of a dynamic system, 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 ultimate aim of controller design could be to minimize the consumption of electrical energy while arriving in time. Before we can decide on the control strategy, we need to know the current state of the train. Even more important, we should know important model parameters such as the mass of the train or how the motor efficiency changes with speed.

To determine the unknown system parameters, we typically perform experiments and record measurement data. In optimization-based state and parameter estimation, the objective function is typically the misfit between the actual measurements and the model predictions.

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.

6.1 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 typically use the index variable $k \in \mathbb{N}$, and write x_k or $x(k)$ 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*.

Finite vs Infinite Dimensional 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. Systems with delay are another class of systems with infinite dimensional state space.

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.

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*.

Stable vs Unstable Dynamic Systems

A dynamic system whose state trajectory remains bounded in an arbitrarily small neighbourhood of an equilibrium point for suitably 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.

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 or the latest measurements.

6.2 Continuous Time Systems

Most systems of interest in science and engineering are described in form of deterministic 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 briefly sketch some relevant properties of continuous time systems in this section, and show how they can be transformed into discrete time systems. Later, we will mainly be concerned with discrete time systems, while we occasionally come back to the continuous time case.

Continuous time systems can often be described by ordinary differential equations (ODE), but sometimes they are described by other forms of differential equations such as differential-algebraic equations (DAE), partial differential equations (PDE) or delay differential equations (DDE). As a part of the main content of the script, only ODE and DAE are explained in more detail here, but the reader is referred to Appendix B to see how PDE and DDE could be addressed with similar estimation methods after some discretization.

6.2.1 Ordinary Differential Equations (ODE)

A controlled dynamic system in continuous time can be described by a general ODE which is represented by:

$$\dot{x}(t) = f(x(t), u(t), \epsilon(t), p). \tag{6.1}$$

Here, the different elements of the ODE are vector valued, and can be defined as:

- States $x(t)$, which describe the internal behaviour of the system. Together they form the state vector, which describes the memory of the system.

- Control inputs $u(t)$, which modify the behaviour of the system.
- Disturbances $\epsilon(t)$, representing the noise that any model/system has.
- Unknown parameters p (constant in time).

In the case of an ODE, we can state that $\dot{x} = \frac{\mathrm{d}x}{\mathrm{d}t} = \frac{\partial x}{\partial t}$, i.e. the total and partial derivative coincide as the state x only depends on t. For notational simplicity, we usually omit time dependence and write $\dot{x} = f(x, u, \epsilon, p)$. For even simpler notation, we sometimes even omit $\epsilon(t)$ and p, getting the following ODE on a time interval $[t_{\text{init}}, t_{\text{fin}}]$ by

$$\dot{x}(t) = f(x(t), u(t), t), \quad t \in [t_{\text{init}}, t_{\text{fin}}] \tag{6.2}$$

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. But we should keep in mind that in practise they need to be included again when necessary. After these simplifications, the standard form of ODE can be defined as:

$$\dot{x} = f(x) \tag{6.3}$$

Let us give some examples of systems that can be described by an ODE and state for each some of the typically needed states.

- Pendulum: rotational angle and angular speed.
- Hot plate with pot: temperature of the pot and the plate.
- Continuously Stirred Tank Reactors (CSTR): concentrations of reactants, temperatures.
- Robot arms: angles and angular speeds of the joints.
- Moving robots: position (x, y) and orientation.
- Race cars: position (x, y), orientation, time derivatives of these three.
- Airplanes in free flight: 3 location parameters and 3 orientation parameters plus the derivatives (speeds) of all these 6.

6.2.2 Differential Algebraic Equations (DAE)

As discussed above, continuous time systems can often be described by ODE, but sometimes they are described by other forms of differential equations such as differential-algebraic equations (DAE). They are very similar to ODE, the only difference is that besides the *differential states* $x \in \mathbb{R}^{n_x}$ there are also the so called *algebraic states* $z \in \mathbb{R}^{n_z}$. The main difference between them is that the derivative of z does not appear in the model equations, and instead, z is implicitly determined by an algebraic equation. Thus, we can write the standard form of DAE (a so called semi-explicit DAE, because the time derivative appears explicitly):

$$\begin{aligned} \dot{x} &= f(x, z) \\ 0 &= g(x, z). \end{aligned}$$

Here, the algebraic equations $g(x, z)$ implicitly determine z. Obviously, z and g must have the same dimension, i.e. $g(x, z) \in \mathbb{R}^{n_z}$. The difference with $\dot{x} = f(x, z)$ is that whereas for each differential state there is one equation determining its time derivative, $g(x, z)$ determines z implicitly, i.e. all the equations have to be considered as a set, and usually, the individual algebraic equations cannot be solved independently. In order to ensure uniqueness and numerical solvability, the Jacobian $\frac{\partial g}{\partial z} \in \mathbb{R}^{n_z \times n_z}$ must be invertible (this is called "index one"). There exist also DAE with higher index, but they are beyond the topics of this course.

Equivalence of DAE with ODE

Any index-one DAE (invertible Jacobian) can in theory be differentiated to obtain a standard ODE, and that is why index-one DAE can be also solved with an ODE solver. The procedure can be defined as follows: first take the total time derivative of the algebraic equation w.r.t. time t, where we know that g is always equal to zero:

$$g(x,z)=0 \quad \Rightarrow \quad \frac{\mathrm{d}g}{\mathrm{d}t}(x,z)=0. \tag{6.4}$$

Here the right equation is equivalent to:

$$\frac{\partial g}{\partial z}\dot{z}+\frac{\partial g}{\partial x}\dot{x}=0 \tag{6.5}$$

Using the property of index one, namely invertible Jacobian, $\frac{\partial g}{\partial z}$ is invertible, and we can write $\dot{z}$ explicitly as:

$$\dot{z}=-\left(\frac{\partial g}{\partial z}\right)^{-1}\frac{\partial g}{\partial x}f(x,z) \tag{6.6}$$

This whole procedure is called index reduction. After this index reduction, we obtain the following ODE (where ODE could be interpreted as a DAE of index zero):

$$\begin{aligned}\dot{x}&=f(x,z)\\ \dot{z}&=-\left(\frac{\partial g}{\partial z}\right)^{-1}\frac{\partial g}{\partial x}f(x,z)\end{aligned}$$

In the way that this ODE was derived, the only thing that can be ensured is that $\frac{\mathrm{d}g}{\mathrm{d}t}=0$, i.e. the value of $g(x(t),z(t))$ remains constant along trajectories: g is an invariant. But, in order to have a complete valid model, we have to ensure also the algebraic equation $g(x,z)=0$ for all the trajectories. However, this algebraic equation is satisfied for all t if it holds for the initial value, i.e. $g(x(0),z(0))=0$ has to be satisfied.

For even simpler cases, an expression of the form $z=h(x)$ could be obtained directly by computer algebra tools.

More General DAE Formulations

In practice there are two more generalisations or extensions of the DAE formulation, namely fully implicit DAE and high index DAE, therefore it is important to take a look at these concepts.

Definition 14 *A fully-implicit DAE is a DAE that is described by one large "implicit" nonlinear equation system:*

$$f(\dot{x},x,z)=0 \tag{6.7}$$

with $f(\dot{x},x,z)\in\mathbb{R}^{(n_x+n_z)}$ and assuming that $\frac{\partial f}{\partial(\dot{x},z)}$ is invertible (index one).

A very special case of this are implicit ODE $f(\dot{x},x)=0$.
Fully implicit DAE often appear in mechanical or chemical applications. Good examples are conservation equations, like the thermal energy in a basin of water: defining the water heat capacity k and the state $x=[m,T]$, with water mass $m(t)$ and temperature $T(t)$, the fully implicit DAE can be derived from energy conservation $\dot{E}=0$ as:

$$0=\dot{E}=k\dot{m}T+km\dot{T} \tag{6.8}$$

Fully-implicit equations can be easily solved in MATLAB using the ode15i solver. It takes as all states in one vector $y=(x,z)^\top$. Grammar: $f(t,y,\dot{y})=0$. So let's look at the example of:

$$f(t,y,\dot{y})=\begin{bmatrix}\dot{y_1}+y_1+y_2=0\\ y_2-\sin(t)=0\end{bmatrix} \tag{6.9}$$

where y_2 is the algebraic state. We can simulate and solve this system in Matlab as follows:

1. Define implicit DAE:

```
function [ resid ] = mydae( t, y, ydot )
resid=zeros(2,1);
resid(1)=ydot(1)+y(1)+y(2);
resid(2)=y(2)-sin(t);
end
```

2. Create consistent initial values:

```
y0=[10;0];
ydot0=[-10;1];
```

3. Call solver (on time interval $[0, 10]$):

```
[tout, yout]=ode15i(@mydae,[0, 10], y0,ydot0)
plot(tout,yout)
```

The output of such a script can be seen on Figure 6.1

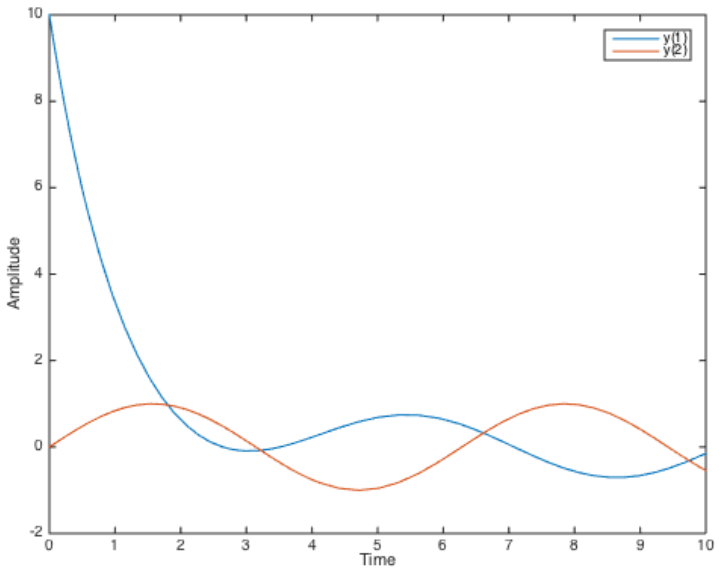


Figure 6.1: Numerical solution for Matlab defined DAE

Definition 15 (High Index DAE) *We can define a high index DAE as a DAE of index n, where $n \geq 2$, where index refers to the number of total time derivatives that it takes to reduce the DAE to an index zero DAE, i.e. reduce to an ODE.*

In practice nobody reduces further than index one, since good DAE solvers exist already for index one e.g. MATLAB ode15i.

6.2.3 Properties of Continuous Time Systems

Let us regard the simplified ODE (6.2) to sketch some relevant properties of continuous time systems. 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:

$$\dot{x}(t) = f(x(t), t), \quad t \in [t_{\text{init}}, t_{\text{fin}}]. \tag{6.10}$$

Initial Value Problems

An initial value problem (IVP) is given by (6.10) 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 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 3 (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* (6.10)*. 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 9 (Existence and Uniqueness of IVP) *Regard the initial value problem* (6.10) *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 $t \in [t_{\text{init}}, t_{\text{fin}}]$*

$$\|f(x,t) - f(y,t)\| \leq L\|x - y\|. \tag{6.11}$$

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 9, 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 possible to obtain an "explosion", i.e., a solution that tends to infinity for finite times, even with innocently looking and smooth functions f.

Example 4 (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 9 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 many state and parameter estimation problems are based on discontinuous control trajectories $u(t)$. 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

$$\dot{x} \quad = \quad Ax + Bu \tag{6.12}$$

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.

For system identification, we usually need to add output equations $y = Cx + Du$ to our model, where the outputs y may be the only physically measurable quantities. In that context, it is important to remark that the states are not even unique, because different state space realizations of the same input-output behavior exist.

Many important notions such as *controllability* or *stabilizability*, and *observability* or *detectability*, and concepts such as the *impulse response* or *frequency response function* can be defined in terms of the matrices A, B, C and D alone. In particular, the *transfer function* $G(s)$ of an LTI system is the Laplace transform of the impulse response and can be shown to be given by

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

The frequency response is given by the transfer function evaluated at values $s = j\omega$ where j is the imaginary unit.

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 $u(t) = u_k$ with $u_k \in \mathbb{R}^{n_u}$ 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 $u(t) = u_{\text{const}}$

$$\dot{x}(t) = f(x(t), u_{\text{const}}), \quad t \in [0, \Delta t], \quad \text{with} \quad x(0) = x_{\text{init}}. \tag{6.13}$$

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

$$x(t; x_{\text{init}}, u_{\text{const}}), \quad \text{for} \quad t \in [0, \Delta t]. \tag{6.14}$$

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:

$$x(t_{k+1}) = f_{\text{dis}}(x(t_k), u_k). \tag{6.15}$$

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

$$f_{\text{dis}}(x_k, u_k) = A_{\text{dis}} x_k + B_{\text{dis}} u_k \tag{6.16}$$

with

$$A_{\text{dis}} = \exp(A\Delta t) \quad \text{and} \quad B_{\text{dis}} = \int_0^{\Delta t} \exp(A(\Delta t - \tau)) B \mathrm{d}\tau. \tag{6.17}$$

One interesting observation is that the discrete time system matrix A_{dis} resulting from the solution of an LTI system in continuous time is by construction an invertible matrix, with inverse $A_{\text{dis}}^{-1} = \exp(-A\Delta t)$. For systems with strongly decaying dynamics, however, the matrix A_{dis} might have some very small eigenvalues and will thus be nearly singular.

Sensitivities

In the context of estimation, derivatives of the dynamic system simulation are often needed. Following Theorem 9 and Corollary 1 we know that the solution map to the IVP (6.13) 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. The reason is 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:

$$\tilde{x}(t; x_{\text{init}}, u_{\text{const}}) = x_{\text{init}} + t f(x_{\text{init}}, u_{\text{const}}), \quad t \in [0, \Delta t]. \tag{6.18}$$

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 $\tilde{x}_0 = x_{\text{init}}$:

$$\tilde{x}_{j+1} = \tilde{x}_j + h f(\tilde{x}_j, u_{\text{const}}), \quad j = 0, \ldots, M-1. \tag{6.19}$$

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:

$$
\begin{aligned}
k_1 &= f(\tilde{x}_j, u_{\text{const}}) && \text{(6.20a)}\\
k_2 &= f(\tilde{x}_j + \frac{h}{2} k_1, u_{\text{const}}) && \text{(6.20b)}\\
k_3 &= f(\tilde{x}_j + \frac{h}{2} k_2, u_{\text{const}}) && \text{(6.20c)}\\
k_4 &= f(\tilde{x}_j + h\, k_3, u_{\text{const}}) && \text{(6.20d)}\\
\tilde{x}_{j+1} &= \tilde{x}_j + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) && \text{(6.20e)}
\end{aligned}
$$

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.

6.3 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. Discrete time system models can be divided in four main categories: 1) Deterministic models in state space model form, 2) deterministic models in input-output model form, 3) stochastic models in state space model form and 4) stochastic models in input-output model form. An input-output model can be easily transformed to an equivalent state space model. However, the transformation from a state space model to an input-output model is not always possible. Defining a system to be stochastic or deterministic represents whether random perturbations of the real system are considered in the model or not.

Deterministic Model as State Space Model	Stochastic Model as State Space Model
Deterministic Model as Input-Output Model	Stochastic Model as Input-Output Model

6.3.1 State Space Model

In the general time-variant case, these systems are characterized by the dynamics

$$
x_{k+1} = f_k(x_k, u_k), \quad k = 0, 1, \ldots, N-1 \qquad \text{(6.21)}
$$

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 16 (Forward simulation) *The* forward simulation *is the map*

$$\begin{array}{rccc} f_{\text{sim}}: & \mathbb{R}^{n_x+Nn_u} & \to & \mathbb{R}^{(N+1)n_x} \\ & (x_0; u_0, u_1, \ldots, u_{N-1}) & \mapsto & (x_0, x_1, x_2, \ldots, x_N) \end{array} \tag{6.22}$$

that is defined by solving Equation (6.21) *recursively for all* $k = 0, 1, \ldots, N-1$.

The inputs of the forward simulation routine are the initial value x_0 and the controls u_k for $k = 0, \ldots, N-1$. 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 (6.21), 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

$$x_{k+1} \quad = \quad Ax_k + Bu_k, \quad k = 0, 1, \ldots, N-1. \tag{6.23}$$

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 asymptotically stable if all eigenvalues of the matrix A are strictly inside the unit disc of the complex plane, i.e. have a modulus 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

$$f_{\text{sim}}(x_0; u_0, \ldots, u_{N-1}) = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} x_0 \\ Ax_0 + Bu_0 \\ A^2x_0 + ABu_0 + Bu_1 \\ \vdots \\ A^Nx_0 + \sum_{k=0}^{N-1} A^{N-1-k}Bu_k \end{bmatrix}$$

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 $\mathcal{C}_N \in \mathbb{R}^{n_x \times Nn_u}$ 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.

Eigenvalues and Eigenvectors of LTI Systems

Every square matrix $A \in \mathbb{R}^{n_x \times n_x}$ can be brought into the Jordan canonical form $A = QJQ^{-1}$ with non-singular $Q \in \mathbb{C}^{n_x \times n_x}$ and J block diagonal, consisting of m-Jordan blocks J_i. Thus, it holds that

$$J = \begin{bmatrix} J_1 & & & \\ & J_2 & & \\ & & \ddots & \\ & & & J_m \end{bmatrix} \quad \text{with} \quad J_i = \begin{bmatrix} \lambda_i & 1 & & \\ & \lambda_i & 1 & \\ & & \ddots & \ddots \\ & & & \lambda_i \end{bmatrix}.$$

Many of the Jordan blocks might just have size one, i.e. $J_i = [\lambda_i]$. To better understand the uncontrolled system evolution with dynamics $x_{k+1} = Ax_k$ and initial condition $x_0 = x_{\text{init}}$, one can regard the solution map $x_N = A^N x_0$ in the eigenbasis, which yields the expression

$$x_N = Q\ J^N\ (Q^{-1}x_0)$$

First, it is seen that all Jordan blocks evolve independently, after the initial condition is represented in the eigenbasis. Second, a simple Jordan block J_i will just result in the corresponding component being multiplied by a factor λ_i^N. Third, for nontrivial Jordan blocks, one obtains more complex expressions with N upper diagonals of the form

$$J_i^N = \begin{bmatrix} \lambda_i^N & N\lambda_i^{N-1} & \cdots & 1 & \\ & \lambda_i^N & N\lambda_i^{N-1} & & \ddots \\ & & & \ddots & \\ & & & & \lambda_i^N \end{bmatrix}.$$

If one eigenvalue has a larger modulus $|\lambda_i|$ than all others, the Jordan block J_i^N will grow faster (or shrink slower) than the others for increasing N. The result is that the corresponding eigenvector(s) will dominate the final state x_N for large N, while all others "die out". Here, the second largest eigenvalues will result in the most slowly decaying components, and their corresponding eigenvectors will keep a visible contribution in x_N the longest.

Interestingly, complex eigenvalues as well as eigenvectors appear in complex conjugate pairs. If an eigenvalue λ_i is complex, the (real part of) the corresponding eigenvector will perform oscillatory motion. To understand the behaviour of complex eigenvectors, let us regard a complex conjugate pair of simple eigenvalues λ_i and $\lambda_j = \bar{\lambda}_i$, and their eigenvectors $v_i, v_j \in \mathbb{C}^{n_x}$, i.e. $Av_i = \lambda_i v_i$ and $Av_j = \bar{\lambda}_i v_j$. It is easy to see that, because A is real, $v_j = \bar{v}_i$ is a possible choice for the eigenvector corresponding to $\bar{\lambda}_i$. Then holds that $\text{Re}\{v_i\} = \frac{1}{2}(v_i + v_j)$. Therefore,

$$A^N\ \text{Re}\{v_i\} = \frac{1}{2}(\lambda_i^N v_i + \lambda_j^N v_j) = \frac{1}{2}(\lambda_i^N v_i + \bar{\lambda}_i^N \bar{v}_i) = \text{Re}\{\lambda_i^N v_i\}.$$

If we represent λ_i as $\lambda_i = re^{\phi i}$ (where the i in the exponent is the imaginary unit while the other i remains just an integer index), then $\lambda_i^N = r^N e^{N\phi i}$. If ϕ is a fraction of 2π, there is an N such that $N\phi = 2\pi$, and after N iterations we will obtain the same real part as in the original eigenvector, but multiplied with r^N. We can conclude that the real part of the eigenvector to a complex eigenvalue $re^{\phi i}$ performs a form of damped or growing oscillatory motion with period duration $N = 2\pi/\phi$ and growth constant r.

Affine Systems and Linearizations along Trajectories

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

$$x_{k+1} \quad = \quad A_k x_k + B_k u_k + c_k, \quad k = 0, 1, \ldots, N-1. \tag{6.24}$$

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 (6.24). 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 (6.24) 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:

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

6.3.2 Input Output Models

So far, all system models we have treated were so called "state space models" given by Equation (6.21) in the case of discrete time systems, and by Equation (6.2) in the continuous time case. This representation is particularly interesting when the state of the system represents meaningful information like the chemical composition in a chemical process, or the velocity of a car, etc.

However, there are many systems where the only information required is just the output of the system. The representation of the output without considering any internal states is the main idea of the input-output representation. This model is shown by Equation (6.25) in the case of a continuous time system and Equation (6.26) in the discrete time case, where y represents the output variable of the system, and as it can be seen it depends only on the output values and control values in the past, but not on an internal state.

$$\frac{\partial^n y}{\partial t^n} = g(y, \frac{\partial y}{\partial t}, \frac{\partial^2 y}{\partial t^2}, \ldots, \frac{\partial^{n-1} y}{\partial t^{n-1}}, u, \frac{\partial u}{\partial t}, \frac{\partial^2 u}{\partial t^2}), \ldots, \frac{\partial^n u}{\partial t^n}) \tag{6.25}$$

$$y(k) = h(u(k), \ldots, u(k-n), y(k-1), \ldots, y(k-n)) \tag{6.26}$$

Transformation from Input-Output to State Space model

Input-output models can be easily transformed to an equivalent state space model. For instance, the discrete time equation of an input-output model given by (6.26) can easily be transformed to an equivalent state space model as follows: We first define the state $x(t)$ to be the concatenation of all relevant past inputs and outputs, $x(t) = [y(t-1)^\top, u(t-1)^\top, \ldots, y(t-n)^\top, u(t-n)^\top]^\top$ and then

$$x(t+1) = f(x(t), u(t)) = \begin{bmatrix} h(u(t), \ldots, y(t-1)\ldots) \\ u(t) \\ y(t-1) \\ u(t-1) \\ \vdots \\ y(t-n+1) \\ u(t-n+1) \end{bmatrix} \tag{6.27}$$

$$y(t) = g(x(t), u(t)) = h(u(t), \ldots, y(t-1)\ldots)$$

The transformation from a state space to an input-output model is not always possible, but in many cases it is, e.g. LTI systems which are observable and controllable.

LTI systems as Input-Output Models

One of the characteristics of an input-output model is that any LTI system can be always represented as an input-output model, and this model can be brought in turn to the unique representation of the system as a transfer function $G(s)$. This can be seen in Equation (6.28) (representing the LTI input output model) and Equation (6.29) (representing the transfer function in continuous time). Note that the change of sign in $a_0, a_1, \ldots$ is done by convention.

$$\frac{\partial^n y}{\partial t^n} = -a_0 y - a_1 \frac{\partial y}{\partial t} - a_2 \frac{\partial^2 y}{\partial t^2} - \ldots - a_{n-1} \frac{\partial^{n-1} y}{\partial t^{n-1}} + b_o u + b_1 \frac{\partial u}{\partial t} + b_2 \frac{\partial^2 u}{\partial t^2} + \ldots + \frac{\partial^n u}{\partial t^n}) \tag{6.28}$$

$$G(s) = \frac{b_0 + b_1 s + \ldots + b_n s^n}{a_0 + a_1 s + \ldots + a_{n-1} s^{n-1} + s^n} \tag{6.29}$$

Furthermore, considering the most general expression for a discrete time LTI system, the transfer function using the z-transform can be derived. These two concepts can be observed in the following two equations:

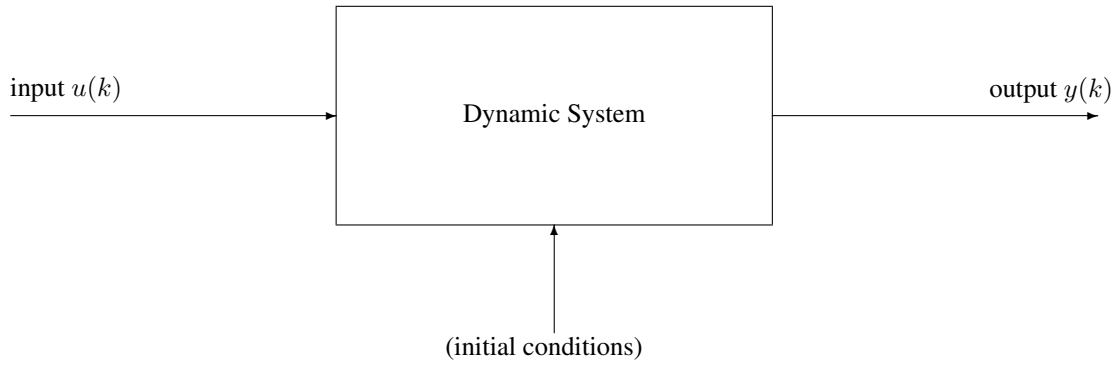


Figure 6.2: Simple scheme of a discrete dynamic system

$$\begin{aligned} &a_0 y(k) + \ldots + a_{n_a} y(k-n_a) = b_0 u(k) + \ldots + b_{n_b} u(k-n_b) \\ &n_a, n_b < 0, n = \max(n_a, n_b), \quad t \in [n+1, n+2, \ldots] \\ &\text{Initial conditions:} \quad y(1) = y_1, \ldots, y(n) = y_n \\ &\qquad\qquad\qquad\quad u(1) = u_1, \ldots, u(n) = u_n. \end{aligned} \tag{6.30}$$

$$G(z) = \frac{b_0 + b_1 z^{-1} + \ldots + b_{n_b} z^{-n_b}}{a_0 + a_1 z^{-1} + \ldots + a_{n_a} z^{-n_a}} \tag{6.31}$$

The expression given by (6.31) is also known as a polynomial model, and we will discuss them later in section 6.4. Furthermore, using the stated definition of $n = \max(n_a, n_b)$, (6.31) can be transformed in the more general transfer function expression (just adding the necessary zero coefficients) given by the next equation:

$$G(z) = \frac{b_0 + b_1 z^{-1} + \ldots + b_n z^{-n}}{a_0 + a_1 z^{-1} + \ldots + a_n z^{-n}} = \frac{b_0 z^n + b_1 z^{n-1} + \ldots + b_n}{a_0 z^n + a_1 z^{n-1} + \ldots + a_n} \tag{6.32}$$

6.4 Deterministic Models

A dynamic system model can be defined as the set of governing equations that describe the behaviour of the system, and which allows the computation of a sequence of system outputs $y(1), \ldots, y(N)$ for any horizon length N, provided that the sequence of inputs $u(1), \ldots, u(N)$ as well as the initial conditions of the system are known.

Generally, when speaking of dynamic system models for discrete time, there has been theory developed to model systems in many different ways. However, for the scope of this course the distinction between different models has been narrowed down to just three: Deterministic Models, Models with Measurement Noise (a.k.a. Output Error Model) and Models with Stochastic Disturbances (a.k.a. Equation Error Models.)

As the name already states, a deterministic model is a model which establishes a set of non-stochastic governing equations, meaning that the output of the system y or the state x can be obtained with absolute certainty, assuming that the inputs of the system $u(1), \ldots, u(N)$, the previous outputs $y(1), \ldots, y(N)$ or state $x(n-1)$ and the initial conditions are known. Examples of such a models are the already presented State Space Model:

$$\begin{aligned} &x(t+1) = f(x(k), u(k)) \\ &y(k) = g(x(k), u(k)), \quad t \in [1, 2, \ldots], \\ &\text{Initial conditions:} \quad x(1) = x_{\text{init}}. \end{aligned} \tag{6.33}$$

or the previously introduced input-output model:

$$\begin{aligned} &y(k) = h(u(k), \ldots, u(k-n), y(k-1), \ldots, y(k-n)), \quad k \in [n+1, n+2, \ldots] \\ &\text{Initial conditions:} \quad y(1) = y_1, \ldots, y(n) = y_n \\ &\qquad\qquad\qquad\quad u(1) = u_1, \ldots, u(n) = u_n. \end{aligned} \tag{6.34}$$

Finite Impulse Response (FIR) Models

In Section 6.3.2, the polynomial model has been introduced as a representation of LTI systems in an input-output model which could be then transformed to a Transfer Function. One special case of this kind of model, which also is an example of a deterministic model, is the finite impulse response (FIR) model.

In this case, and as the name states, the response of the system is finite, meaning that the output of the system at any moment k is a weighted sum of the n_b previous inputs $[u(k-n_\text{b}), \ldots, u(k)]$, therefore, if the input is set to 0, after n_b time intervals the output of the system will also be zero, making the response of the system finite. The general equation of a FIR system can be seen in Equation (6.35), and its transfer function in Equation (6.36). Note that using the same nomenclature as in (6.3.2), a FIR system can also be defined as a general LTI system where $n_a = 0$ (n_a was the number of past outputs entering the system as inputs).

$$y(k) = b_0 u(k) + \ldots + b_{n_b} u(k - n_b) \tag{6.35}$$

$$G(z) = b_0 + b_1 z^{-1} + \ldots + b_{n_b} z^{-n_b} = \frac{b_0 z^{n_b} + b_1 z^{n_b - 1} + \ldots + b_{n_b}}{z^{n_b} + 0 + \ldots + 0} \tag{6.36}$$

Auto Regressive Models with Exogenous Inputs (ARX) Models

This kind of model is a generalisation of the FIR, representing a system where the output is a weighted sum of the past inputs $[u(1), \ldots, u(N)]$ and past outputs $[y(1), \ldots, y(N)]$, and it is defined by:

$$\begin{aligned} &a_0 y(k) + \ldots + a_{n_a} y(k - n_a) = b_0 u(k) + \ldots + b_{n_b} u(k - n_b) \\ &n_a, n_b \geq 0, n = \max(n_a, n_b), \quad t \in [n+1, n+2, \ldots] \\ &\text{Initial conditions:} \quad y(1), \ldots, y(n) \\ &\qquad\qquad\qquad\quad u(1), \ldots, u(n). \end{aligned} \tag{6.37}$$

$$G(z) = \frac{b_0 z^n + b_1 z^{n-1} + \ldots + b_n}{a_0 z^n + a_1 z^{n-1} + \ldots + a_n} \tag{6.38}$$

These models are also known as infinite impulse response models (IIR) due to the fact that the dependence on the previous inputs results in a non-zero value for the output (with the exception of the trivial case where the output is 0 at the beginning and no further input is applied).

Another kind of model is the autoregressive (AR) model, which is completely autonomous, meaning that the response does not depend on the input. Such systems have no transfer function since the numerator of a classic transfer function using a z-transform would be always 0. An example of a system that could be represented by an AR model would be the Fibonacci numbers: $[1, 1, 2, 3, 5, 8, 13, 21, \ldots]$. The equation of the system is represented by:

$$y(k) = -a_1 y(k-1) - \ldots - a_{n_a} y(k - n_a) \tag{6.39}$$

Simulation of Deterministic Models

One of the best features of a deterministic model is that the simulation of the system is very easy to obtain given the initial conditions x_init, the control trajectory of the system $U = [u(1), u(2), \ldots, u(N)]$, and the system parameters p. In particular, using any of the forward simulation methods presented at the beginning of Section (6.3) (Euler or RK4), the computation of the outputs $[y(1), \ldots, y(N)]$ can be easily obtained. This mapping from U, p and x_init to a general output $y(k)$ is represented by the function M in the following equation:

$$y(k) = M(k; U, x_\text{init}, p) \tag{6.40}$$

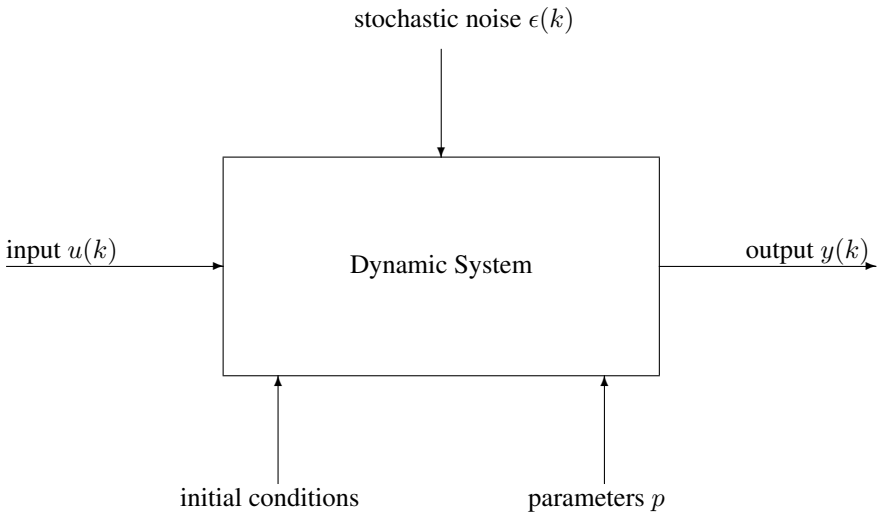


Figure 6.3: Model with stochastic disturbances (equation errors)

6.5 Stochastic Models

In reality, dynamic systems are far from being deterministic. Instead we have so-called stochastic models, and the three main differences with respect to a deterministic model are:

- In reality, we always have stochastic noise $\epsilon(k)$, e.g. external disturbances or measurement errors.
- also, we have unknown, but constant system parameters p.
- measured outputs $y(k)$ depend on both, $\epsilon(k)$ and p:

It can be assumed that the noise is i.i.d., and that these noise terms $\epsilon(k)$ enter the model like a normal input, but as a random value. In this way, we can define the different stochastic models. First, we introduce the equivalent model to the state space model, the so-called stochastic state space model:

$$\begin{aligned} x(t+1) &= f(x(k), u(k), \epsilon(k)) \\ y(k) &= g(x(k), u(k), \epsilon(k)) \\ &\text{for} \quad t = 1, 2, \dots \end{aligned}$$

and secondly the equivalent stochastic input output model:

$$\begin{aligned} y(k) = h(&u(k), \dots, u(k-n), y(k-1), \dots, y(k-n), \epsilon(k), \dots, \epsilon(k-n)) \\ &\text{for} \quad k = n+1, n+2, \dots \end{aligned}$$

6.5.1 Model with Measurement Noise (Output Error Model)

One special case of stochastic models are models with the error only modelled in the output, and where the error $\epsilon(k)$ can be assumed as additive measurement noise. In this case, the equation of the output can be defined as:

$$y(k) = M(k; U, x_{\text{init}}, p) + \epsilon(k) \tag{6.41}$$

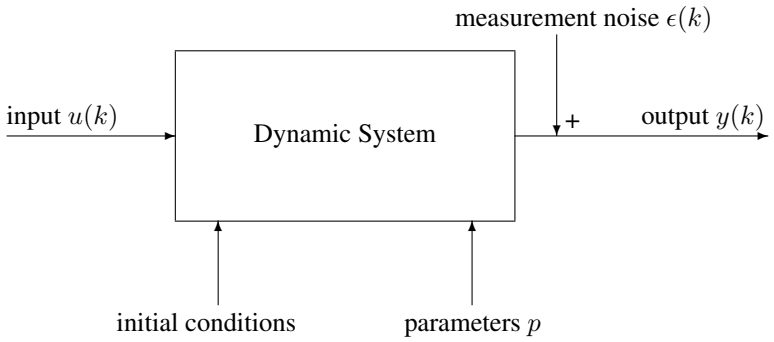


Figure 6.4: Model with measurement noise (output errors)

where $M(k; U, x_{\text{init}}, p)$ can be the deterministic model used in the previous section. The picture below illustrates this concept in a graphical manner:

Its main advantage is that assuming that there is only noise on the output is often a realistic noise assumption. Of course, there are many systems where this assumption is not valid, and where noise should also be modeled on e.g. the input of the system.

The main disadvantage of this model is that M is typically nonlinear, thus non-convex, and finding a global minimum is often an impossible task.

6.5.2 Model with Stochastic Disturbances (Equation Errors)

As was said previously, the assumption of having only noise at the output is not always correct. Sometimes it is more correct to assume that stochastic noise $\epsilon(k)$ can enter also the model internally, and not only at the output equation. Furthermore, the measured outputs $y(k)$ depend on both $\epsilon(k)$ and p. A schematic representation is given in Figure (6.3).

One special case arises when the i.i.d. noise $\epsilon(k)$ enters the input-output equation as additive disturbance:

$$\begin{aligned} y(k) &= h(p, u(k), \ldots, u(k-n), y(k-1), \ldots, y(k-n)) + \epsilon(k) \\ &\text{for} \quad k = n+1, n+2, \ldots \end{aligned}$$

Linear In the Parameter models (LIP)

A general form of LIP model with equation error noise is given below:

$$y(k) = \sum_{i=1}^{d} \theta_i \, \phi_i(u(k). \ldots, y(k-1), \ldots) + \epsilon(k) \tag{6.42}$$

where $\phi_1, \ldots, \phi_d$ are called the basis functions of the LIP model. The unknown coefficients enter the above equation linearly. In the case of a general input-output model, the basis functions depend on the past inputs u and the outputs y.

Another way to express the model is given by:

$$y(k) = \varphi(k)^\top \theta + \epsilon(k) \tag{6.43}$$

where $\varphi(k) = (\phi_1(\cdot), \ldots, \phi_d(\cdot))^\top$ is the regression vector and it is formed by the basis functions of the LIP model.

Special Case: LIP-LTI Models with Equation Errors (ARX)

A general ARX model with equation errors can be stated by Equation (6.44). This model is both LTI and LIP, thus combining the best of two worlds:

$$a_0 y(k) + \ldots + a_{n_a} y(k-n_a) = b_0 u(k) + \ldots + b_{n_b} u(k-n_b) + \epsilon(k). \tag{6.44}$$

Other Stochastic System Classes

Besides the two explained models, there are plenty of different stochastic models out there. Some of the most well-known are:

- Auto-regressive moving average with eXogeneous input (ARMAX), where the noise term $\epsilon(k)$ goes through a FIR filter before becoming part of the equation error:

$$\begin{aligned} & a_0 y(k) + \ldots + a_{n_a} y(k-n_a) \\ =& b_0 u(k) + \ldots + b_{n_b} u(k-n_b) + \epsilon(k) + c_1 \epsilon(k-1) + \ldots c_{n_c} \epsilon(k-n_c) \end{aligned}$$

- Auto-regressive moving average models without inputs (ARMA):

$$a_0 y(k) + \ldots + a_{n_a} y(k-n_a) = \epsilon(k) + c_1 \epsilon(k-1) + \ldots c_{n_c} \epsilon(k-n_c)$$

Where the c_i represent the noise coefficients, and when estimating them, we will have to use nonlinear least squares, because c_i are multiplied with the unknown noise terms $\epsilon(k-i)$.

Chapter 7

Parameter Estimation with Dynamic System Models

Let's do a short summary of the previous chapter, which will help us to better understand this chapter: General Stochastic Models are a type of model that included a variable noise term $\epsilon(k)$. Such stochastic noise $\epsilon(k)$ enters the model either internally or in the output equation. The model has the output $y(k)$, which depends on the initial values: $u(k)$, p, and $\epsilon(k)$. A diagram of the model that includes the error internally is depicted below:

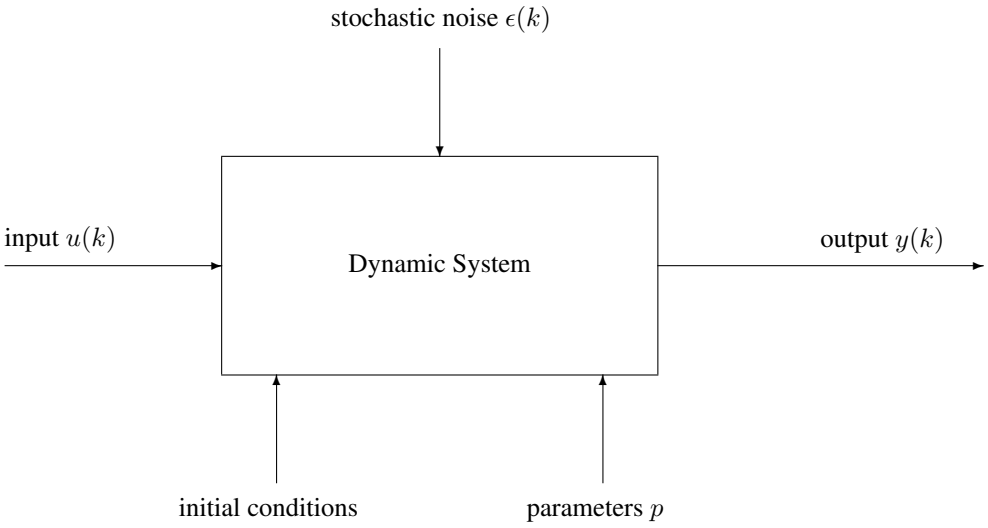


Figure 7.1: Model with stochastic disturbances

The noise terms $\epsilon(k)$ are i.i.d. noise, and they enter the model as stochastic input. In the case of Stochastic state-space models, the model is:

$$\begin{aligned} x(k+1) &= f(x(k), u(k), \epsilon(k)) \\ y(k) &= g(x(k), u(k), \epsilon(k)) \quad \text{for} \quad k = 1, 2, \ldots \end{aligned}$$

whereas for input-output models, it is:

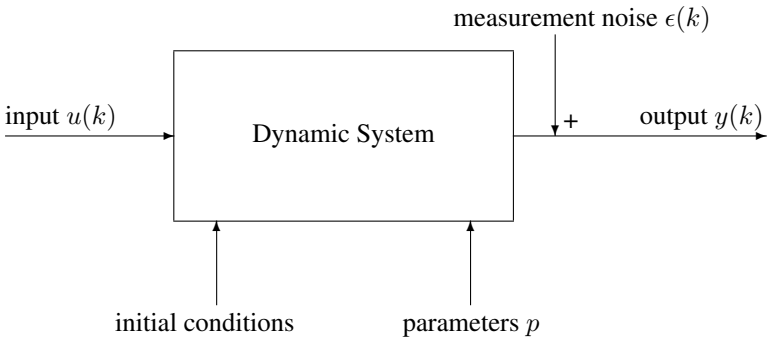


Figure 7.2: Model with measurement noise (output errors)

$$y(k) = h(u(k), \ldots, u(k-n), y(k-1), \ldots, y(k-n), \epsilon(k), \ldots, \epsilon(k-n))$$
$$\text{for} \quad k = n+1, n+2, \ldots$$

In the case of additive measurement noise $\epsilon(t)$, a deterministic model can be used:

$$y(k) = M(k; U, x_{\text{init}}, p) + \epsilon(k)$$

A schematic representation of this model is given in Figure (7.2).

7.1 Pure Output Error (OE) Minimization

When we can assume i.i.d. Gaussian noise that is only affecting the output, a maximum likelihood estimate for $\theta = (x_{\text{init}}^\top, p^\top)^\top$ can be obtained by non-linear least-squares using the following algorithm:

$$\theta_{\text{ML}} = \arg\min_\theta \sum_{k=1}^{N} \quad (\, y(k) - M(k; U, x_{\text{init}}, p) \,)^2 \tag{7.1}$$

Here $U = (u(1), \ldots, u(N))^\top$ and x_{init} represents the initial conditions. This algorithm has several disadvantages:

1. Usual models are non linear and non-convex,therefore a global minimum cannot be guaranteed.
2. Unstable systems are very difficult to identify using this kind of model.
3. In reality, also the inputs of the systems have noise, which the model does not account for.

7.1.1 Output Error Minimization for FIR models: Convex minimization

A nice property of FIR models is that when using the Output Error representation, they lead to convex problems, which implies that a global minimum can be found.

Let $\theta = p = (b_0, \ldots, b_{n_b})^\top$ be the unknown parameter that needs to be estimated. A FIR model with output errors predicts measurements which are modelled by the following equation:

$$y(k) = (u(k), u(k-1), \ldots, u(k-n_{n_b})) \cdot \theta + \epsilon(k) \tag{7.2}$$

Therefore, for $k \geq n_b + 1$, the deterministic part of this model can be expressed as $M(k; U, x_{\text{init}}, p) = (u(k), u(k-1), \ldots, u(k-n_{n_b})) \cdot \theta$, and the OE minimization problem for a generic FIR model is the linear least squares problem stated below:

$$\min_{\theta} \sum_{k=n_b+1}^{N} \; (\, y(k) - (u(k), u(k-1), \ldots, u(k-n_{n_b}))\theta \,)^2 \tag{7.3}$$

The main disadvantage of this problem is that even though we could represent our system by a FIR model, they often need a very high dimension n_b to obtain a reasonable fit. As a consequence ARX models are usually used instead. Furthermore, with FIR models no physical interpretation possible.

7.2 Equation Error Minimization

One special case of equation error models arises when the i.i.d. noise $\epsilon(k)$ enters the input-output equation as additive disturbance:

$$\begin{aligned} y(k) &= h(p, u(k), \ldots, u(k-n), y(k-1), \ldots, y(k-n)) + \epsilon(k) \\ &\text{for} \quad k = n+1, n+2, \ldots \end{aligned}$$

In this case, if the noise is i.i.d. Gaussian, a maximum likelihood formulation to estimate the unknown parameter vector $\theta = p$ is given by:

$$\theta_{\text{ML}} = \arg\min_{\theta} \sum_{k=n+1}^{N} \; (\, y(k) - h(p, u(k), \ldots, y(k-1), \ldots)) \,)^2 \tag{7.4}$$

where $u(k)$ and $y(k)$ are the known input and output measurements, and where the algorithm minimises the sum of the so called **equation errors** or **prediction errors**, which are represented by the differences of $y(k) - h(p, u(k), \ldots, y(t-1), \ldots))$.

The problem (7.4) is also known as Prediction error minimisation (PEM).

Such a problem is convex if p enters linearly in f, i.e. if the model is **linear-in-the-parameters (LIP)**. In this case, because of the convexity of the problem, the prediction error minimization is globally solvable.

PEM of LIP Models

A way to express a LIP model is given by:

$$y(k) = \varphi(k)^\top \theta + \epsilon(k) \tag{7.5}$$

where $\varphi(k) = (\phi_1(\cdot), \ldots, \phi_d(\cdot))^\top$ is the regression vector and it is formed by the basis functions of the LIP model. Considering this last expression, the prediction error minimisation (PEM) problem then can be formulated as:

$$\min_{\theta} \underbrace{\sum_{k=n+1}^{N} \; (\, y(k) - \varphi(k)^\top \theta) \,)^2}_{= \| y_N - \Phi_N \theta \|_2^2} \tag{7.6}$$

which can be solved using the well-known analytical solution for Linear Least Square: $\theta^* = \Phi_N^+ y_N$

Special Case: PEM of LIP-LTI Models with Equation Errors (ARX)

A general ARX model with equation errors can be stated by Equation (6.44):

In order to have a determined estimation problem, a_0 has to be fixed, otherwise the number of optimal solutions would be infinite. Therefore, we usually fix $a_0 = 1$, and use $\theta = (a_1, \ldots, a_{n_a}, b_0, \ldots, b_{n_b})^\top$ as the parameter estimator vector. Then the regression vector is given by:

$$\varphi(k) = (-y(k-1), \ldots, -y(k-n_a), u(k), \ldots, u(k-n_b))^\top \tag{7.7}$$

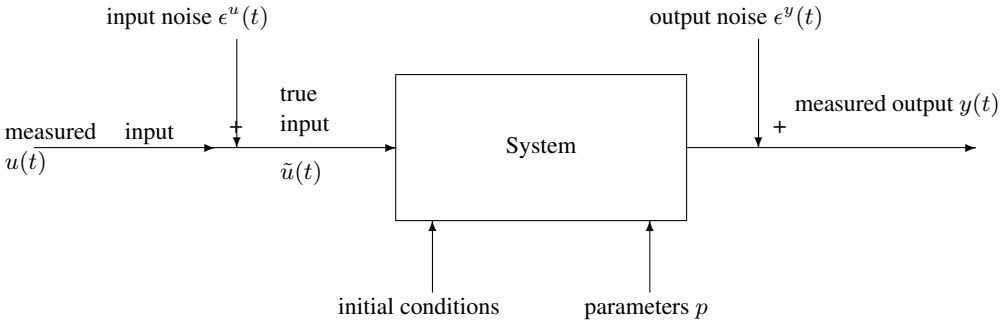
leading to the unique optimal solution provided by the LLS algorithm:

$$y(k) = \varphi(k)^\top \theta + \epsilon(k) \tag{7.8}$$

7.3 Models with Input and Output Errors

To model input and output errors, we make the simple assumption that we have two noise terms $\epsilon^u(t)$ and $\epsilon^y(t)$ affecting the input and the output of our deterministic model:

$$y(k) = M(k; U + \epsilon_N^u, x_{\text{init}}, p) + \epsilon^y(k) \tag{7.9}$$



For this particular problem, assuming once again i.i.d. Gaussian noise on both, inputs and outputs, with variances σ_u^2 for the inputs and σ_y^2 for the outputs, it is straightforward to establish the ML estimation problem for the unknown parameter vector $\theta = (x_{\text{init}}^\top, p^\top, {\epsilon_N^u}^\top)^\top$. The noise vector is defined by $\epsilon_N^u = (\epsilon^u(1), \ldots, \epsilon^u(N))^\top$. This is a non-linear least squares problem, given by:

$$\min_\theta \sum_{k=1}^{N} \frac{1}{\sigma_y^2}(y(k) - M(k; U + \epsilon_N^u, x_{\text{init}}, p))^2 + \frac{1}{\sigma_u^2}(\epsilon^u(k))^2 \tag{7.10}$$

Making the transformation of $\theta \to \hat{\theta}$, where the $\tilde{\theta} = (x_{\text{init}}^\top, p^\top, \tilde{U}^\top)^\top$ with $\tilde{U} = U + \epsilon_N^u$, the equation stated before can be written as:

$$\min_{\tilde{\theta}} \sum_{k=1}^{N} \frac{1}{\sigma_y^2}(y(k) - M(k; \tilde{U}, x_{\text{init}}, p))^2 + \frac{1}{\sigma_u^2}(u(k) - \tilde{u}(k))^2 \tag{7.11}$$

In contrast to Output-Error Models, Input-Output-Error Models are more complete and accurate because they take into account the input noise. They can also better deal with unstable systems.

7.4 State Space Models with Process and Measurement Noise

We consider discrete time state space models of the following form

$$x_{k+1} = f(x_k, u_k, p) + w_k \tag{7.12}$$

$$y_k = g(x_k, u_k, p) + v_k \tag{7.13}$$

with states $x_k \in \mathbb{R}^{n_x}$, outputs $y_k \in \mathbb{R}^{n_y}$, control inputs $u_k \in \mathbb{R}^{n_u}$ that we assume to be perfectly known and (constant) parameters $p \in \mathbb{R}^{n_p}$.

In the following, we assume that both process and measurement noise, $w_k \in \mathbb{R}^{n_x}$ and $v_k \in \mathbb{R}^{n_y}$ respectively, are i.i.d. and follow a zero-mean Gaussian distribution, i.e.

$$w_k \sim \mathcal{N}(0, \Sigma_w),\ v_k \sim \mathcal{N}(0, \Sigma_v).$$

Given a control trajectory $(u_0, \ldots, u_N)$ and measurements $(y_0, \ldots, y_N)$, we would like to compute the most likely values of the parameters p as well as the most likely state trajectory $(x_0, \ldots, x_N)$. The problem of estimating both p and $(x_0, \ldots, x_N)$ is referred to as *parameter and state trajectory estimation*.

Let us summarize the unknown variables by the vector $\theta = (p, x_0, \ldots, x_N)$. We define the unweighted residual function $\tilde{R}(\theta)$ as

$$\tilde{R}(\theta) = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_{N-1} \\ v_0 \\ v_1 \\ \vdots \\ v_N \end{bmatrix} = \begin{bmatrix} x_1 - f(x_0, u_0, p) \\ x_2 - f(x_1, u_1, p) \\ \vdots \\ x_N - f(x_{N-1}, u_{N-1}, p) \\ y_0 - g(x_0, u_0, p) \\ y_1 - g(x_1, u_1, p) \\ \vdots \\ y_N - g(x_N, u_N, p) \end{bmatrix}$$

The weighted function $R(\theta)$ is obtained from $\tilde{R}(\theta)$ by multiplying it with the inverse square root of the covariance matrix $\Sigma = \operatorname{cov}((w_0, \ldots, w_{N-1}, v_0, \ldots, v_N))$ (cf. Section 4.3 on weighted least squares):

$$R(\theta) = \Sigma^{-\frac{1}{2}} \tilde{R}(\theta)$$

where Σ is given as

$$\Sigma = \begin{bmatrix} \Sigma_w & & & & & & \\ & \ddots & & & & & \\ & & \Sigma_w & & & & \\ & & & \Sigma_v & & & \\ & & & & \ddots & & \\ & & & & & \Sigma_v \end{bmatrix}.$$

We can obtain parameter and state trajectory estimates by solving the following nonlinear least squares problem

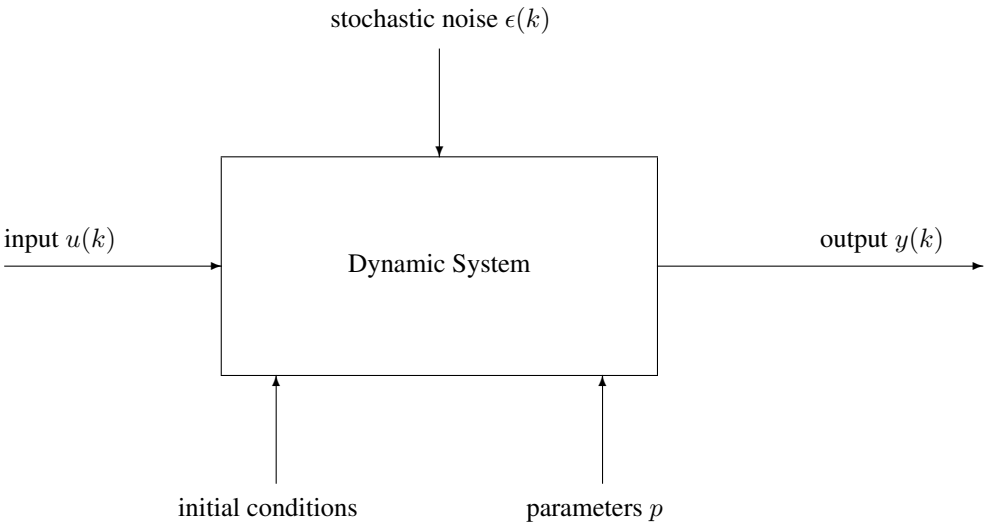
$$\theta^* = \arg\min_{\theta} \frac{1}{2} \|R(\theta)\|_2^2$$

which can be solved using the Gauss-Newton algorithm (cf. Section 5.5.1). In Matlab, this is implemented by the function `lsqnonlin`.

Chapter 8

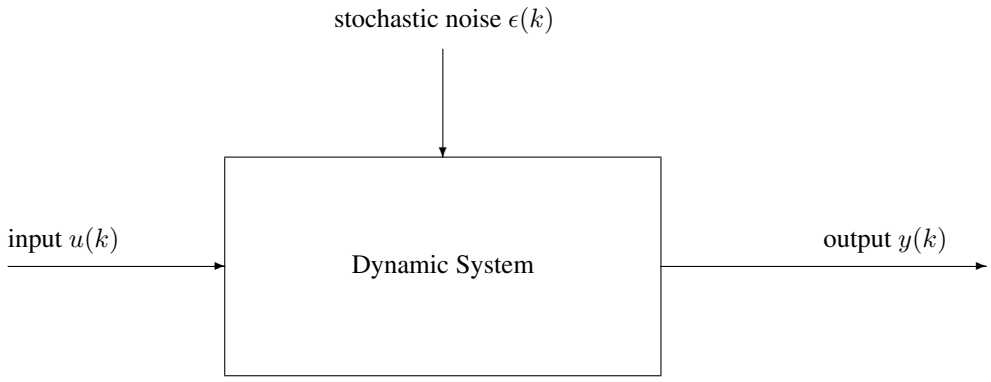
Nonparametric and Frequency Domain Identification Methods[1]

As seen in the previous Chapter, dynamics system models have inputs $u(t)$ and outputs $y(t)$ that can be measured. We assume i.i.d. noise $\epsilon(t)$ that disturbs our experiments and an unknown system model illustrated below:



In nonparametric identification we do not model a physical representation of the system but we consider the model as a "black box". We rather directly want to obtain the transfer function of the system. Thus, in this approach neither the parameters nor initial conditions matter. Such a model is depicted below:

[1] not covered in the lecture



The main features of a nonparametric identification problem are the following:

- Its aim is to make a prediction of the system without doing any real modelling work, related to the physical laws involved in the system's behaviour.
- The approach that follows is to choose a generic model class and then identify the "black-box" model
- For Linear time invariant (LTI) systems we get the nonparametric model by identifying the impulse response function (see next section).

8.1 Nonparametric identification of LTI systems

A continuous time LTI system is a system that allows us to compute, for any horizon $[0, T]$ and control trajectory $u(t)$ for $t \in [0, T]$, the output trajectory $y(t)$ for $t \in [0, T]$. Typically, we assume the initial conditions to be zero. Moreover, it is very easy to transform a continuous LTI system to a discrete one, and MATLAB provides an easy interface to do so.

As already mentioned before, for LTI systems it is sufficient to identify the impulse response function to get the transfer function of the system. Another tool to analyse the behaviour of an LTI system is the Bode diagram.

8.1.1 The impulse response and transfer function

If the impulse response function $g(t)$ is known, the output for any input signal $u(t)$ can be computed by a convolution operation. Let $y(t)$ be the output for a given input $u(t)$, and $g(t)$ be the impulse response, then the output can be calculated as follows:

$$y(t) = \int_0^\infty g(\tau)u(t-\tau)\mathrm{d}\tau \tag{8.1}$$

By transforming the input and impulse response functions into the Laplace domain, we get the output in the Laplace domain by a simple multiplication:

$$Y(s) = G(s)U(s) \tag{8.2}$$

where $G(s)$ is the transfer function that characterises the system completely. It can be obtained directly with the Laplace transform of the impulse response:

$$G(s) = \int_0^\infty e^{-st}g(t)\mathrm{d}t \tag{8.3}$$

where $s \in \mathbb{C}$, and $G : \mathbb{C} \to \mathbb{C}$. As an example, we can take a look at the impulse response and the step response of the transfer function $G(s) = \frac{1}{a+s}$, which are illustrated in Figure 8.1 and 8.2.

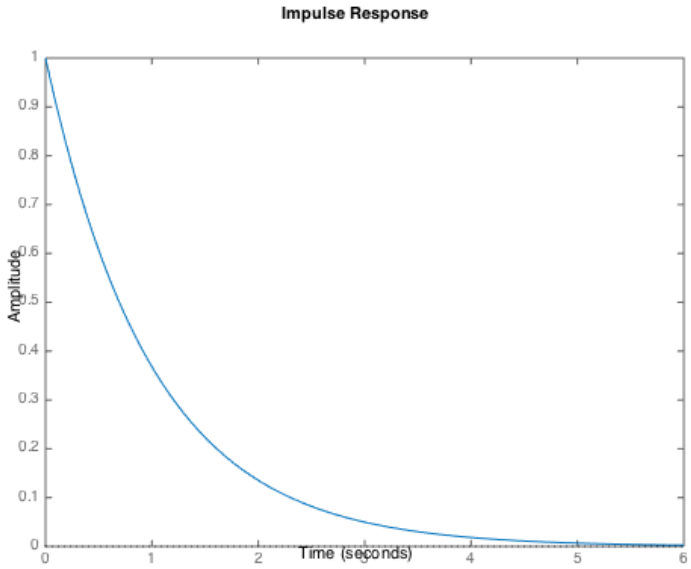


Figure 8.1: Impulse response for $G(s) = \frac{1}{a+s}$

8.1.2 Bode diagram

Bode diagrams are a way to visualize the transfer function $G(s)$. This kind of diagram shows the values of $G(j\omega)$ for all positive values of ω (where j is the imaginary unit, and ω is measured in rad/s and represented the frequency of the input $u(t)$.

The diagram consists of two parts, a magnitude and a phase plot, both with logarithmically spaced frequencies ω on the x-axis. The magnitude plot shows the magnitudes $|G(j\omega)|$ also in a logarithmic scale (log-log plot). However, the phase plot shows the argument $\arg G(j\omega)$ of the complex number $G(j\omega)$, i.e. its angle in the complex plane, in a normal scale.

A Bode diagram can be easily obtained in MATLAB using the command `bode`, which can generate the Bode diagram of a known system. As an example, we can take a look on the Bode Plots of the transfer function $G(s) = \frac{1}{a+s}$, which is shown in Figure 8.3

This plot is a different way to look at the system, but it is also very representative. It represent the behaviour of the system for different frequencies. In the example plot, we see how the output is attenuated and the phase shifted at high frequencies.

Bode diagram from Frequency Sweeps

One of the exciting results of LTI system theory is that whenever the system is excited with a sinusoidal input, the output will be always sinusoidal. However, the sine at the output is shifted in phase and its amplitude is modified.

$$u(t) = A \cdot \sin(\omega \cdot t), \quad y(t) = \|G(j \cdot \omega)\| A \cdot \sin(\omega \cdot t + \alpha) \tag{8.4}$$

The Bode diagram shows for any frequency ω the phase shift α and amplitude modification. A frequency sweep goes trough all frequencies ω, waits till transients die out, and records phase and magnitude for all frequencies. This method is not efficient, since it consumes a lot of time waiting in each iteration for the transients to die out.

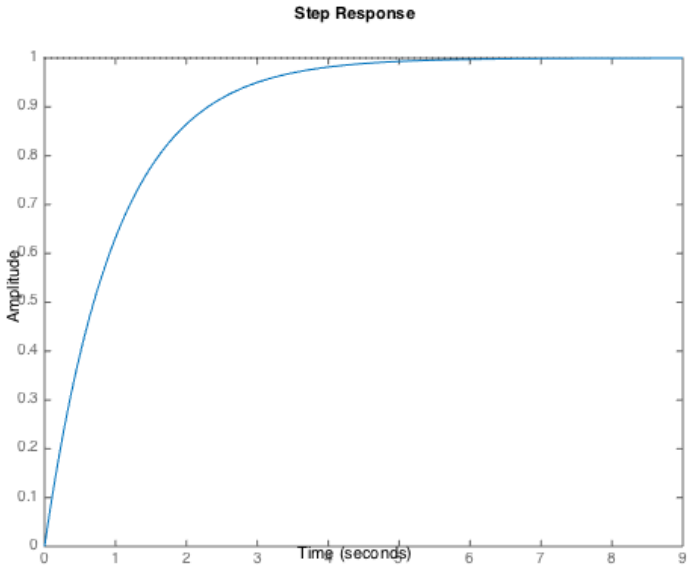


Figure 8.2: Step response for $G(s) = \frac{1}{a+s}$

8.1.3 Discrete time LTI systems

Recall from Section 6.3.2 the definition of a discrete LTI system. It is a system with discrete input $u(k)$ and discrete output $y(k)$, with a time difference between the samples equal to the sampling time ΔT. Its most general expression is defined by:

$$\begin{aligned} &a_0 y(k) + \ldots + a_{n_a} y(k-n_a) = b_0 u(k) + \ldots + b_{n_b} u(k-n_b) \\ &n_a, n_b < 0, n = \max(n_a, n_b), \quad t \in [n+1, n+2, \ldots] \\ &\text{Initial conditions:} \quad y(1) = y_1, \ldots, y(n) = y_n \\ &\qquad\qquad\qquad\quad u(1) = u_1, \ldots, u(n) = u_n. \end{aligned} \tag{8.5}$$

In Section 6.3.2 we also saw the structure of its transfer function, which was defined by:

$$G(z) = \frac{b_0 + b_1 z^{-1} + \ldots + b_{n_b} z^{-n_b}}{a_0 + a_1 z^{-1} + \ldots + a_{n_a} z^{-n_a}} \tag{8.6}$$

however back then, we did not show how to compute such a function, and that is something that we look in detail in this section.

The main idea is that if the so-called discrete time impulse response values $g(0), g(1), \ldots$ are known, the general output can be computed by a linear combination of past inputs using the convolution equation presented before, but adapted to discrete time ($\int \to \sum$):

$$y(t) = \sum_{k=0}^{\infty} g(k) u(t-k) \mathrm{d}\tau \tag{8.7}$$

In the discrete time domain, the z-transform replaces the Laplace-transform, and in the so called z-domain, a convolution translates to a multiplication of the z-transforms:

$$Y(z) = G(z)U(z) \tag{8.8}$$

Here, the z-transform of any signal, like g, u, y can be defined as:

$$G(z) := \sum_{t=0}^{\infty} z^{-t} g(t) \tag{8.9}$$

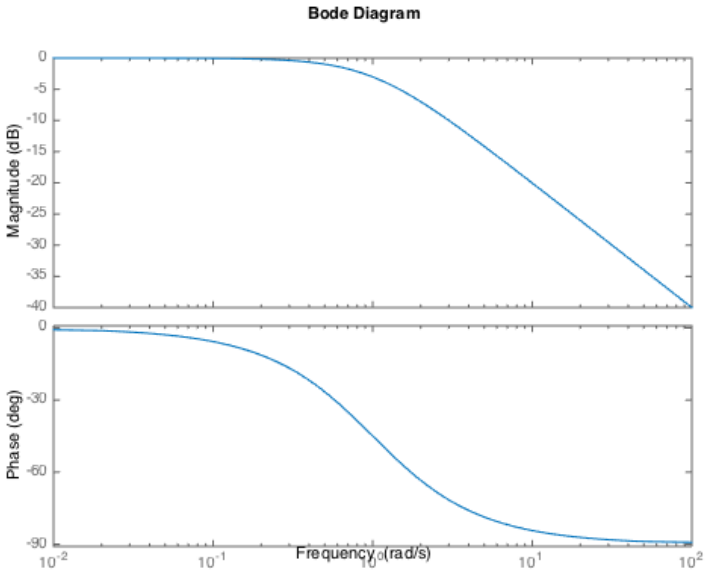


Figure 8.3: Bode plot for $G(s) = \frac{1}{a+s}$

Remember the difference between FIR and IRR system: we have a finite impulse response (FIR) models if and only if $g(k)$ has finitely many non-zero values (the impulse response is finite), otherwise it means that the model is an infinite impulse response (IIR).

Experimental method to obtain impulse response in discrete time

From a mathematical point of view, the impulse response of a system is nothing else but the derivative of the step response. Therefore, one way to compute the impulse response of the system is to measure the step response (which is typically easy):

$$h(t) = \sum_{k=0}^{t} g(k) \cdot u(t-k) \cdot \Delta T = \sum_{k=0}^{t} g(k) \cdot \Delta T \quad \rightarrow \quad g(t) = \frac{h(t) - h(t-1)}{\Delta T} \tag{8.10}$$

However this approach is very sensitive to noise.

Discrete time Bode diagrams

A discrete time Bode diagram shows the values of the complex function $G(z)$ on the unit circle (instead of the imaginary axis as in the Laplace-domain), i.e. $z = e^{j\omega T}$ where T is the sampling time.

For values $\omega > \frac{2\pi}{T}$, the values of z are repeated. In fact, only the values on the upper semi-circle are plotted, up to the so-called Nyquist frequency $\omega_{\max} = \frac{\pi}{T}$, so that the Bode diagram has a limited range of ω (in contrast to continuous time). The Nyquist frequency is known from the Nyquist Theorem, which establishes that it is the maximum frequency that can be resolved by sampling time T.

Finally, it is important to note that $G(e^{j\omega T})$ is given by:

$$G(e^{j\omega T}) := \sum_{k=0}^{\infty} e^{-jk\omega T} g(k) \tag{8.11}$$

This expression looks similar to the definition of the discrete Fourier Transform (DFT or FFT), that is introduced in the following chapter.

8.2 The Discrete Fourier Transform

Everything we have defined in the previous section leads to the so-called transfer function, which when represented on the imaginary axis or unit circle, leads to the so call Frequency Response Function (FRF).

The aim of nonparametric identification of LTI systems is to get the transfer function $G(s)$, where the magnitudes and phases of $G(j\omega)$ for different positive frequencies ω create the Bode Diagram. We also saw a fundamental fact of LTI systems: sinusoidal inputs $u(t) = \mathrm{Re}\{U \cdot e^{j\omega t}\}$ lead to sinusoidal outputs $y(t)$ with a phase shift and a new magnitude described by $G(j\omega)$:

$$y(t) = \mathrm{Re}\{G(j\omega) \cdot U \cdot e^{j\omega t}\} = |G(j\omega)| \cdot U \cdot \mathrm{Re}\{e^{j[\omega t + \arg G(j\omega)]}\} \tag{8.12}$$

Precisely for this reason, $G(j\omega)$ is called the Frequency Response Function (FRF). It is important to note that all these concepts are only applicable to LTI systems. For non-linear or time dependent systems, the frequency behaviour is more difficult to describe.

We can also recall from the last section the definition of a Sine Wave Testing (a.k.a. Frequency Sweep). Remember that one way to obtain $G(j\omega)$ for a specific frequency ω was to use a sine wave $u(t) = U_0 \sin(\omega t)$ as input and record the magnitude Y_0 and the phase shift ϕ of $y(t) = Y_0 \sin(\omega t + \phi)$ to form the following expression:

$$G(j\omega) = \frac{Y_0}{U_0} e^{j\phi}$$

In the case of a frequency sweep, the previous procedure would be repeated through all frequencies ω, waiting until transients had died out, and recording the magnitude and phase of the output for each frequency. The main disadvantage of this procedure is that for each new frequency, we have to wait until transients died out, so that the total measurement time is very long. The scope of this section is to introduce the Fourier Transform to find a more efficient way to estimate the FRF.

Different names for the FRF can be found in the literature, among them empirical transfer function estimate (ETFE) [Lju99].

8.2.1 Laplace vs Fourier Transform

Before entering the definition of the Fourier transform it is interesting to see the differences between the Laplace and the Fourier transform. In different fields they are used in place of each other because they represent the same operation, however, there are subtle differences that we would like to point out.

We can recall from the previous section that $G(s) = \frac{Y(s)}{U(s)}$, where the Laplace transform $G(s)$ was defined for any $g(t)$, $g(t)$ that was defined as 0 for $t < 0$, as:

$$G(s) := \int_0^\infty g(t) e^{-st} \mathrm{d}t = \int_{-\infty}^\infty g(t) e^{-st} \mathrm{d}t \tag{8.13}$$

where in the case of using the Laplace transform for the Frequency Response Function $G(j\omega)$, we only need pure imaginary values $s = j\omega$, for frequency ω, and therefore the final expression is defined by:

$$G(j\omega) = \int_{-\infty}^\infty g(t) e^{-j\omega t} \mathrm{d}t \tag{8.14}$$

This last expression is identical to the Fourier Transform (FT), defined for any function $f : \mathbb{R} \to \mathbb{R}$ by

$$\mathcal{F}\{f\}(\omega) := \int_{-\infty}^\infty f(t) e^{-j\omega t} \mathrm{d}t \tag{8.15}$$

With that in mind we can go a bit deeper on the similarity and the subtle difference between them:

1. Both transformations basically contain the same information.
2. Both transform a time signal $f(t)$ from the time domain into frequency domain.
3. Both transformations have inverse transformations that give the original time signal back.
4. Both transformations generate complex valued functions.

5. Laplace transform has complex a input argument $s \in \mathbb{C}$, while Fourier transform has a real $\omega \in \mathbb{R}$ as input.

6. For the Laplace transform, all input signals are by definition zero for $t < 0$, while the Fourier transform deals with functions defined for any $t \in \mathbb{R}$ (i.e. functions with infinite support)

7. The Laplace transform is often used by engineers, the Fourier transform more often used by mathematicians and physicists

8.2.2 Inverse Fourier Transform

If we define the Fourier transform of the function $f(t)$ as $F(\omega) = \mathcal{F}\{f\}(\omega)$, then $f(t)$ can be recoverd by inverse Fourier transformation $\mathcal{F}^{-1}$ given by:

$$f(t) = \mathcal{F}^{-1}\{F\}(t) := \frac{1}{2\pi}\int_{-\infty}^{\infty} F(\omega)e^{j\omega t}\mathrm{d}\omega \tag{8.16}$$

It is important to remark the similarity of normal and inverse FT: just the sign in the exponent and the factor are different (some definitions even use twice the same factor, $\frac{1}{\sqrt{2\pi}}$, to make both expressions completely symmetric). Furthermore it is clear that the inverse FT can be used to construct the inverse Laplace transform, since they basically are the same.

One interesting related fact to the Fourier transform is that the Dirac-delta function is the superposition of all frequencies with equal weight:

$$\delta(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{j\omega t}\mathrm{d}\omega \tag{8.17}$$

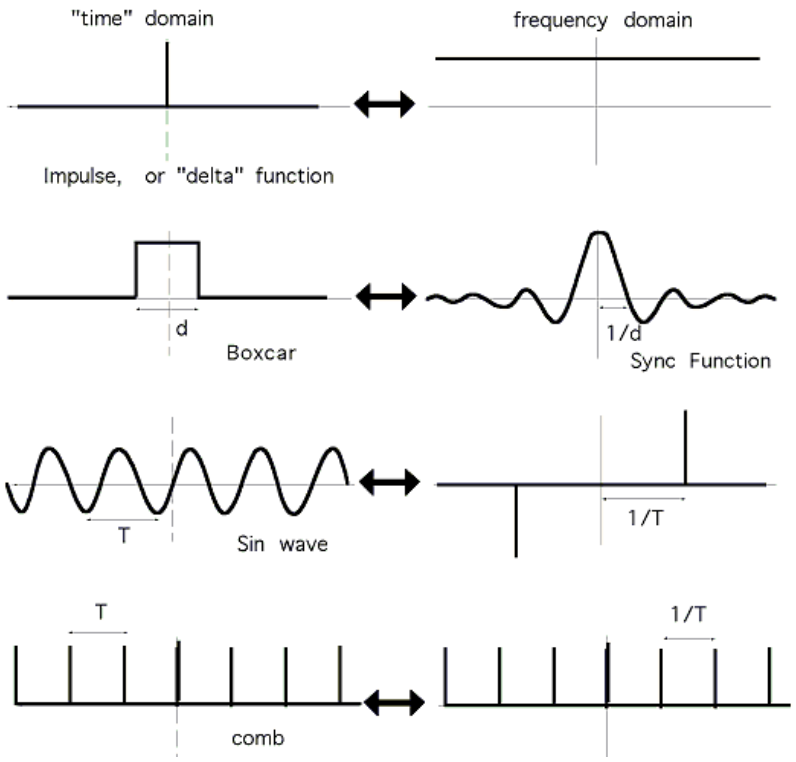
expression that also represent the inverse Fourier transform of a function in a frequency domain representing the whole spectre with the same magnitude for every frequency.

8.2.3 Fourier Transform examples

In the Figure 8.2.3 we can see some of the most well-known functions on the time and frequency domain. Note that the first example is exactly the Dirac-delta case explained before.

The second one is the impulse function that leads to the sync function on the frequency domain. Notice that the window width on the time domain is inversely proportionally to the window width on the frequency domain, and this fact is very exciting and important because it states that in any analysis, the more precise you need have the function in time domain, then the less precise will be the function on the frequency domain.

The third example is the standard sinusoidal function in time domain, that lead to a single positive frequency. Finally the last example is a set of impulse function.



8.2.4 Estimating the FRF with the Fourier Transform

We can now use the previous explained concept as a practical tool to calculate the FRF. The idea behind this method is that if we have recorded two arbitrary time signals, $u(t)$ as the input and $y(t)$ as the output signal of a system, we can the use their Fourier transforms to estimate the frequency response function (FRF) by the following equation:

$$G(j\omega) = \frac{\mathcal{F}\{y\}(\omega)}{\mathcal{F}\{u\}(\omega)} \tag{8.18}$$

where this fact is implicitly used in sine wave testing with frequency ω_0 to determine the frequency of the sine wave. To show that let's compute first the Fourier Transform of $f_1(t) = \frac{e^{j\omega_0 t}}{2\pi}$:

$$\mathcal{F}\{f_1\}(\omega) = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{j(\omega_0-\omega)t}\mathrm{d}t = \delta(\omega-\omega_0) \tag{8.19}$$

then, considering that a real sine is described by $f_2(t) = \frac{e^{j\omega_0 t} - e^{-j\omega_0 t}}{2\pi}$, its Fourier transform can be easily calculated (by means of the distributive property of the Fourier Transform) as:

$$\mathcal{F}\{f_2\}(\omega) = \delta(\omega - \omega_0) - \delta(\omega + \omega_0) \tag{8.20}$$

Thus, whenever we wanna obtain the frequency of any sine wave function, its Fourier transform can be applied and since the 2 Dirac impulses of $\mathcal{F}$ will be located at $\pm\omega_0$, ω_0 can be easily obtained.

Nevertheless, in practice, this procedure has a huge main disadvantage: in reality, even for sine waves of frequency ω_0 , the signals $u(t)$ and $y(t)$ will have finite duration, and thus the FT when applied to this finite signals will also be finite. In order to avoid undefined expressions (such as division by 0), the computation of the FT should be done only taking into account the finite values of $\mathcal{F}\{y\}(\omega_0)$ and $\mathcal{F}\{y\}(\omega_0)$:

$$G(j\omega_0) = \frac{\mathcal{F}\{y\}(\omega_0)}{\mathcal{F}\{y\}(\omega_0)} \tag{8.21}$$

Nevertheless this procedure of having to manually discard values does not seems good, since it is a waste of computational time. Another drawback is the condition of continuous time signals, since most of the signal nowadays are digital and not analog, thus their values are discretised. We have seen that Fourier Transform works with continuous time signals on infinite horizons, therefore two questions naturally (with their two answers) arises:

1. How to compute FT in practice? **Answer**: by the Discrete Fourier Transform, which solves the problem of finite time and discrete values.

2. Can we use an input with many frequencies to get many FRF values in a single experiment? Remember that so far the only procedure introduce was the frequency sweeping, which lead to high computational times due to repetition of the process for each frequency. **Answer**: yes, we should then use multisines.

8.2.5 Discrete Fourier Transform

As we have mentioned, FT works with continuous time signals on infinite horizons, and this condition is most of the time not satisfied: practical signals are very often discrete and finite. The Discrete Fourier Transform (DFT) works with discrete signals on finite horizons, solving therefore the problem.

Basically, the main idea of the DFT is that it takes any vector of N numbers $u(0), u(1), \ldots, u(N-1)$ and generates a new vector $U(0), \ldots, U(N-1))$ which also has N components (here we start with index zero for convenience). Moreover, th DFT also has an inverse transformation that recovers the original vector

Fast Fourier Transform (FFT)

One efficient algorithm to compute the DFT is called Fast Fourier Transform (FFT), and in practise the DFT is nearly always computed by the FFT algorithm, therefore many people (and MATLAB) use the word FFT synonymously with DFT. In MATLAB for instance, the commands `fft` and `ifft` provide an easy interface to compute the FFT and its inverse.

DFT definition

Given a vector of discrete values $u(0), \ldots, u(N-1)$, the discrete fourier transform of them are represented by $U(0), \ldots, U(N-1)$, which are computed from $u(0), \ldots, u(N-1)$ using the following equation:

$$U(m) := \sum_{k=0}^{N-1} u(t)\alpha_N^{-mk}, \quad \text{with} \quad \alpha_N := e^{j\frac{2\pi}{N}} \tag{8.22}$$

It is important to note that α_N is an N-th complex root of 1, i.e. $\alpha_N^N = 1$ It is also important to note that $\alpha_N^{-mk} = e^{-j\frac{2\pi}{N}mk}$ and $\overline{\alpha_N^{-mk}} = \alpha_N^{mk}$.

From the very basic definition we can try to grasp what the DFT does. Since α_N is a point on the unit circle of the complex plane with phase $\frac{2\pi}{N}$, and since the k multipliying α_N means that the new point α_N^k point is shifted $\frac{2\pi}{N}$ from the originally α_N. With that it mind, we can understand the DFT as a weighted sum of the vector u to compute each of the U elements, where the weights are the N point on the unit circle equally spaced by $\frac{2\pi}{N}$. Note that the factor m represents the fact that when computing the DFT, each U has a different weight sequence, otherwise the DFT for any of the $U(0), \ldots, U(N-1)$ points would be exactly the same.

DFT properties

In the scope of this lecture we will only study the most important and useful properties of the DFT. The properties says that the DFT of a real valued signal consists of N complex numbers, but only the first half contain useful information, since second half of vector are complex conjugates of first half, and thus redundant information:

$$U(N-m) = \overline{U(m)} \tag{8.23}$$

Proof 1

$$U(N-m) = \sum_{k=0}^{N-1} u(t)\alpha_N^{-(N-m)k} = \sum_{k=0}^{N-1} u(t)\alpha_N^{mk} = \sum_{k=0}^{N-1} u(t)\overline{\alpha_N^{-mk}} \tag{8.24}$$

This property is completely related to the Nyqvist theorem, which stablishes the maximum frequency that can be analysed giving a certain sampling frequency That limitation is due to something called aliasing, and we will treat it in the following sections.

As an example of this important property, Figure 8.4 represent a DFT of some signal. On it we can see how the second half of the values are a conjugate copy of the first half:

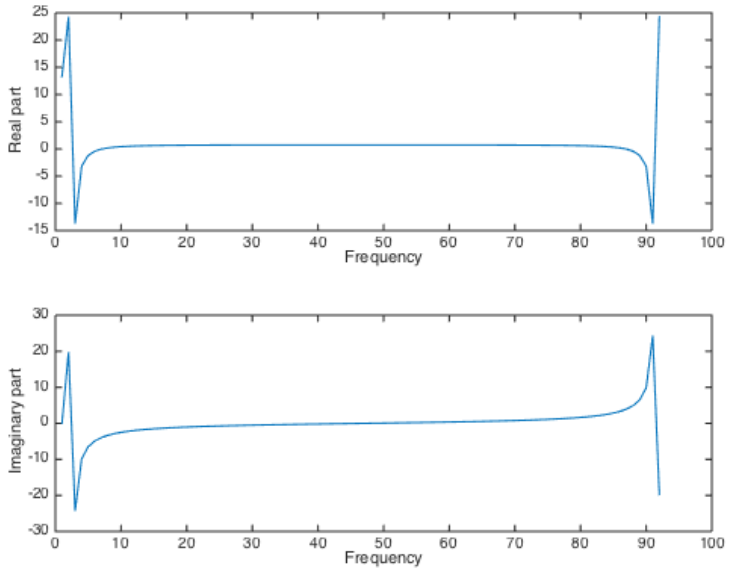


Figure 8.4: DFT example to visualise DFT property

8.2.6 Aliasing and Leakage Errors

In order to understand why the DFT is not that perfect, it is important to remember the differences between the FT and the DFT, and what does it implies in practice.

Basically, FT works on continuous time signals $u_c(t)$ with infinite support, whereas the DFT is forced to do make two approximations:

1. **Sampling:** the DFT has to work on sampled (discrete time) signals as:

$$u_d(k) := u_c(k \cdot \Delta t) \tag{8.25}$$

   where Δt is the sampling time.

2. **Windowing:** DFT only uses only N samples, i.e. limits the signal to a finite window of horizon length $T = N\Delta t$

Unfortunately these two approximations lead to characteristic errors, errors that we will define and explain in the following sections.

Aliasing Errors

Sampling can introduce so called aliasing errors when the continuous time signal contained too high frequencies. Basically if we introduce a sampling rate $f_s = \frac{1}{\Delta t}$, then any signal with frequencies higher than half the sampling rate will suffer from aliasing. This limit on the maximum frequency that can be sampled is called the Nyquist frequency, and represented by:

$$f_{\text{Nyquist}} = \frac{1}{2\Delta t}[Hz] \quad \text{or} \quad \omega_{\text{Nyquist}} = \frac{2\pi}{2\Delta t}[rad/s] \tag{8.26}$$

In Figure 8.5 we can see this effect. There we can see three different sine waves, the first one with $\omega_1 = 6\frac{rad}{s}$, a second one with $\omega_2 = 20\frac{rad}{s}$, and a third one with $\omega_3 = 60\frac{rad}{s}$. Since the sampling frequency is $\omega_s = 62\frac{rad}{s} < 2 \cdot \omega_3$, the third sine wave is not sampled correct and aliasing occurs.

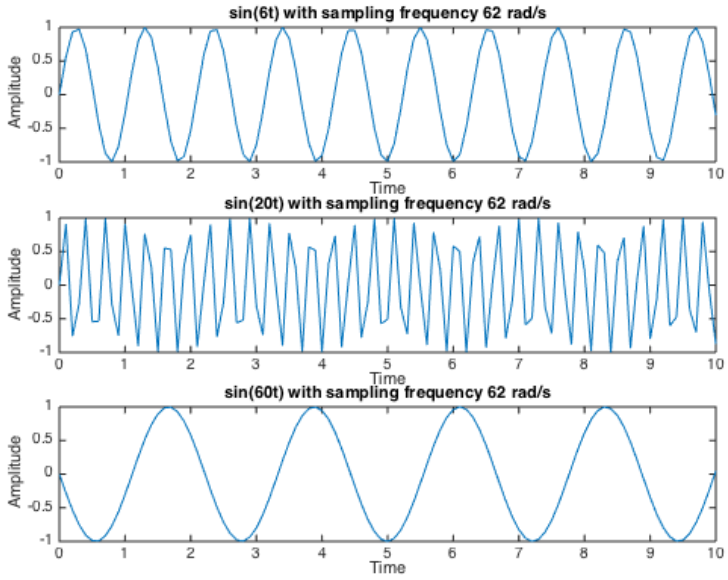


Figure 8.5: Aliasing example

Leakage Errors

Leakage is defined as the effect that occurs when the DFT spectrum shows frequencies that were not present in original signal, but are close to the true frequencies. To understand the reason for that it is important to look at again at the difference between the FT and the DFT:

$$\int_{-\infty}^{\infty} u_c(t) \cdot e^{-j\omega t}\, \mathrm{d}t \approx \sum_{k=0}^{N-1} u_d(k) \cdot \underbrace{e^{-j\omega(k\cdot\Delta t)}}_{=e^{-j\frac{2\pi}{N}km}} \cdot \Delta t \tag{8.27}$$

where the integral represents the definition of the FT, and the sum the ideal translation to a discrete version. From that it is clear that the FT and DFT expressions are only similar when:

$$-j\omega(k\cdot\Delta t) = -j\frac{2\pi}{N}km \quad \text{i.e.} \quad \omega = m\frac{2\pi}{\Delta t \cdot N} \tag{8.28}$$

So from this last expression we get that the number of samples N, the sampling frequency $\omega_s = \frac{2\pi}{\Delta T}$ and the signal frequency ω are interrelated, and unless their values are properly chosen to satisfy (8.28), then leakage occurs and the DFT spectrum is not correct. This interaction says that the signal frequency must be a multiple of a base frequency defined by $\frac{2\pi}{\Delta t \cdot N}$. The Figure 8.6 illustrates this problem.

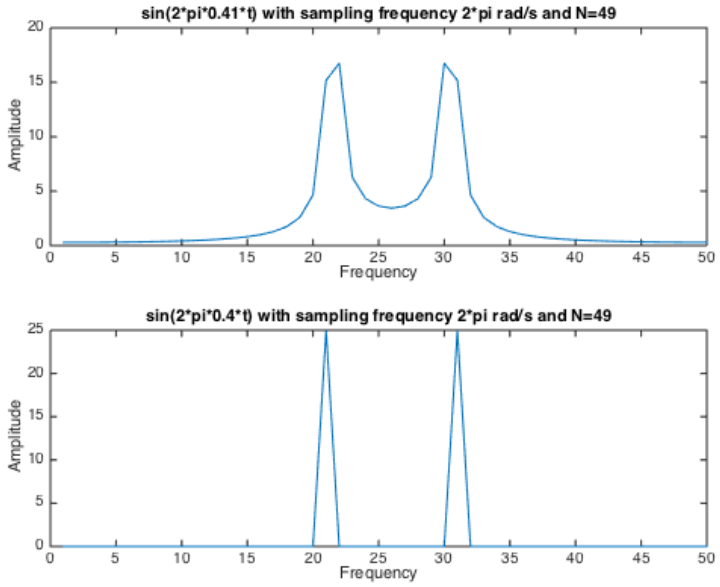


Figure 8.6: The first graph shows leakage, and the second ones does not

The Base Frequency and its Harmonics

One of the concepts strongly related with the leakage effect is the Base Frequency and the Harmonics. We can define the base frequency as:

$$\omega_{\text{base}} := \frac{2\pi}{N \cdot \Delta t} = \frac{2\pi}{T} \tag{8.29}$$

frequency that represents the slowest sine that fits exactly into the sampling window $N \cdot \Delta t$. Considering this, we can define and harmonic as follows: a sine signal $\sin(\omega t)$ is called the m-th harmonic if $\omega = m \cdot \omega_{\text{base}}$.

Leakage errors are produced the DFT contains only the first $N/2$ harmonics of the base signal, thus the frequency resolution (difference of two frequencies that are distinguished by the DFT) is equal to the base frequency, and whenever the signal frequency is not a multiple of the base frequency leakage will occur. On top of that it is easy to identify that the highest harmonic that the DFT can resolve correspond with the Nyqvist effect.

In summary we can say that the finite length of the window limits the frequency resolution, because the longer the time window, the smaller the base frequency, and thus the finer the frequencies that can be resolved in the signal.

As an illustration, Figure 8.7 represent sine waves the base frequency with the first 4 harmonics in time domain, and Figure 8.8 does the same for frequency domain.

8.3 Multisine Excitation Signals

If $u(t)$ is a superposition of a set of specially chosen sine waves, the $u(t)$ is known as a multisine, and we can use this input $u(t)$ to excite and identify in a nonparametric way the LTI system under study.

The multisine is know as the most perfect excitation signal for nonparametric identification, since using this signal as an excitation signal, and the using the DFT to the input and output signal, we can identify the system in a more shorter way than using a frequency sweep. Nevertheless, in order to avoid both aliasing and leakage, the following three conditions have to be met:

1. The DFT window length T has to be an integer multiple of the sampling time Δt, i.e. $T = N \cdot \Delta t$.

2. The multisine can contain only the harmonics of the base frequency $\omega_{\text{base}} = \frac{2\pi}{T}$, i.e. it is periodic with period T (or an integer fraction of T).

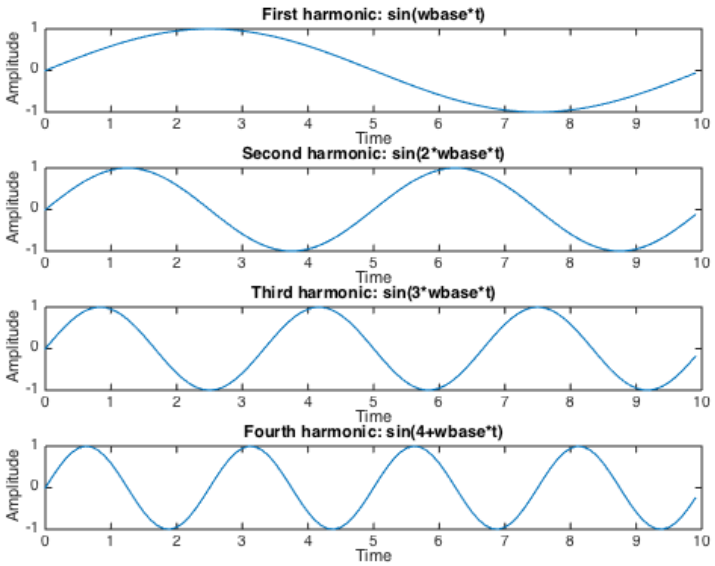


Figure 8.7: Base frequency, second, third and fourth harmonic in time domain

3. The multisine does not contain any frequency higher than the Nyquist frequency $\omega_{\text{Nyquist}} = \frac{\pi}{\Delta t}$. In reality the maximum frequency is chosen to be as one half of the Nyquist frequency.

On top of that, in order to achieve optimal excitation without too large input amplitudes, one chooses the phases of the multisine carefully to avoid positive interference. The practical purpose of that is that systems have a limitation on the maximum input amplitude, if there is positive interference on the multisine, the amplitude of the input might be bigger than the maximum, and the signal is clipped, obtaining a different input than the desired.

One way to create a multisine in a programming language as Matlab, is to define the discrete vector containing the spectrum (which is very easy to define) of the multisine, defining each phase and amplitude, then doing the inverse fourier transform of such a vector, the multisine is easy obtained.

As an illustration, Figure 8.9 represents a multisine where positive interference occurs due to the fact that the phases were initialised with the same value, therefore there are high peaks. In Figure 8.10 we can see how this signal when entering a system with a maximum input amplitude of 0.2 clips the multisine. Finally in Figure 8.11 we can see a multiline which phases were randomly chosen and thus no positive interference occurs.

8.3.1 The Crest Factor

One way to measure how god a multiline is, in the sense of how much positive interference it has, it is to use the so-called crest factor.

The crest factor can be defined as the ratio between the highest peak $u_{\max}$ and the root mean square u_{rms} of the input signal, and thus a measure of how nice and smooth a multisine is.

$$C_{\text{CR}} := \frac{u_{\max}}{u_{\text{rms}}} \tag{8.30}$$

where

$$u_{\max} := \max_{t \in [0,T]} |u(t)| \tag{8.31}$$

and

$$u_{\text{rms}} := \sqrt{\frac{1}{T}\int_0^T u(t)^2 \mathrm{d}t} \tag{8.32}$$

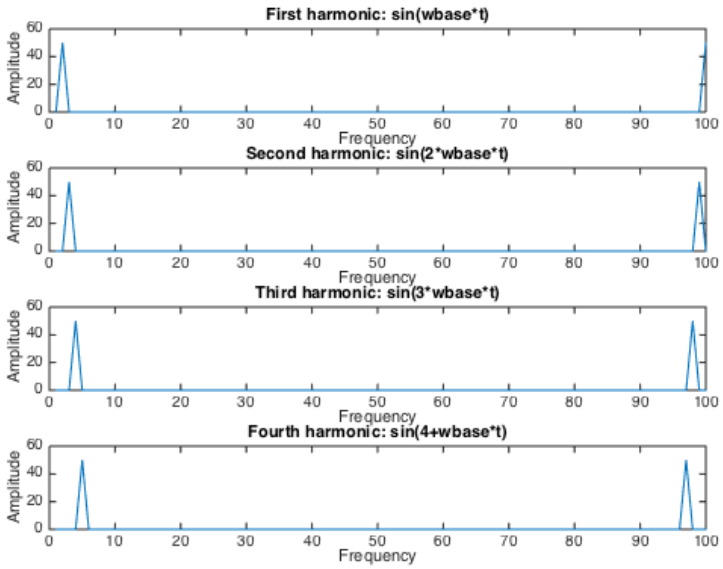


Figure 8.8: Base frequency, second, third and fourth harmonic in frequency domain

We can look at the previous section to obtain a rough estimator of the crest factor. It is obvious that the multiline depicted by Figure 8.9 will have a very bad crest factor, whereas that Figure 8.11 will have a much better crest factor.

8.3.2 Optimising Multisine for optimal crest factor

There are two main actions that can be taken in order to have e good multisine in terms of crest factor:

1. Because most of the representation on nonparametric identification (like Bode diagrams) are done in logarithmic spaces, then it is not a good idea to choose the frequencies in a linear fashion. Instead, we can choose approximately logarithmically spaced frequencies $\omega_{k+1}/\omega_k \approx 1.05$ (many high frequencies are left empty). In this case 1.05 is just an example, $1.1, 1.03, \ldots$ can be also used instead. Keep also in mind that the relation $\omega_{k+1}/\omega_k \approx 1.05$ would produce very likely a frequency ω_{k+1} which is not a multiple of the base frequency, therefore rounding must executed so that all the chosen frequencies are multiples of the base frequency.

2. Optimise the phases to minimize the crest factor. The only problem with this approach is that it is nonconvex problem, therefore it requires the use one of many heuristics. Because of that reason, in practice many times a random algorithm to choose the phases is used instead, it is suboptimal, but still produces a much better result than having the signal in synchronisation.

As an illustration, Figure 8.12 (this figure was obtained from [RP12]) shows the example of a good designed multisine. We see how the frequencies are after 10 $[Hz]$ are equally spaced in a logarithmic scale. The fact that at the beginning is not the case it is because of the rounding which makes use of all the harmonics. Furthermore, it can be seen that the multsine is quite smooth because of the phase optimisation, leading to an optimal crest factor.

8.3.3 Multsine Identification Implementation Procedure

As a final recall we can summarise the procedure to design a multisine, and use it for nonparametric identification. For that we can list the main 2 steps.

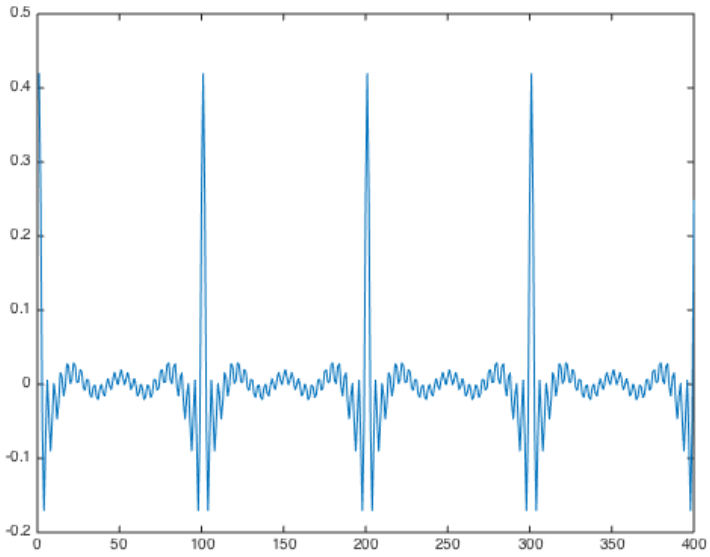


Figure 8.9: Multsine with positive interference

Multisine Procedure Step 1: Input Design

The first step is to choose and design the input $u(t)$ as a periodic multisine, keeping in mind the 3 guidelines pointed before to avoid aliasing and leakage errors. We can summarise this three guidelines as:

1. Choosing as window length T an integer multiple of the sampling time Δt, i.e. $T = N \cdot \Delta t$, e.g. $N = 4096$.
2. Adding to the multisine only harmonics of the base frequency $\omega_{\text{base}} = \frac{2\pi}{T}$.
3. Not adding to the multisine any frequency higher than about half of the Nyquist frequency $\omega_{\text{Nyquist}} = \frac{2\pi}{2\Delta t}$

Furthermore, it is important to choose nonzero amplitudes only for those frequencies of interest. The most clear example was to use P logarithmically spaced frequencies $\omega_{k(p)} = \omega_{\text{base}} \cdot k(p)$, with $k(p)$ being integers between 1 and $N/4$, and $p = 1, \ldots, P$, so that it matches the log-plots of Bode diagrams.

The final consideration when designing the multisine is to choose phases randomly (or smarter) to have a small crest factor.

Multisine Procedure - Step 2: Experiment and Analysis

Once the multisine is designed, it is injected into the system in a periodic manner, i.e. the same multisine is injected for many periods: $u(1), \ldots, u(m)$.

After that, it is necessary to wait until the transients died out, discard these first periods, and then record the input/output data over the last M periods (e.g. $M = 100$) of duration T (i.e. in total NM time samples).

Once the data is recorded, the last M periods have to be averaged in time domain, so that we the averages $\hat{u}(k)$, $\hat{y}(k)$, for $k = 0, \ldots, N-1$ can be obtained. After that, the DFT of $\hat{u}(k)$ and $\hat{y}(k)$ must be taken to get $\hat{U}(k)$ and $\hat{Y}(k)$ for $k = 0, \ldots, N-1$

Finally, with $\hat{U}(k)$ and $\hat{Y}(k)$ the transfer function can be estimated at the excited frequencies ω_p by:

$$\hat{G}(j\omega_{k(p)}) = \frac{\hat{Y}(k(p))}{\hat{U}(k(p))}, \quad p \in [1, \ldots, P] \tag{8.33}$$

Finally the Bode diagram can be plotted considering that the non-excited frequencies might contain some noise.

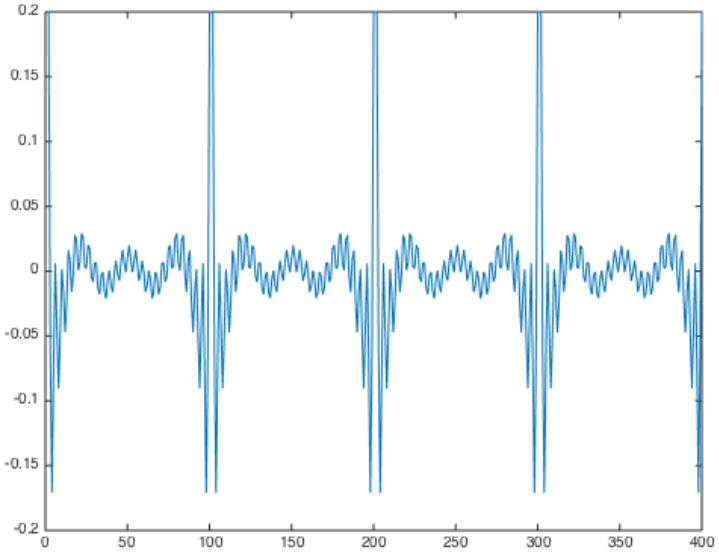


Figure 8.10: Multsine with positive interference clipped at the entrance

8.3.4 Multsine Error Analysis

In general the multisine procedure is good, it leads to no leakage or aliasing errors, no transient effects (if we waited long enough), however it still has to deal with noisy measurements of u and y, and thus, noisy averages of $\hat{u}(k)$ and $\hat{y}(k)$, so that the DFT of them is also noisy.

The main goal of this section is to analysis the estimator $\hat{G}(j\omega_k)$, where regarding one of the excited frequencies with index k, $\omega_k = \omega_{\text{base}} \cdot k$, the $\hat{G}(j\omega_k)$ is defined as:

$$\hat{G}(j\omega_k) = \frac{\hat{Y}(k)}{\hat{U}(k)} \tag{8.34}$$

And with that goal, two main question regarding the estimator arise:

1. Is the estimate unbiased ?
2. How to estimate the variance of $\hat{G}(j\omega_k)$?

Before continuing the analysis it is important to do one remark regarding the average step: due to linearity of DFT, the average can be done right before or after computing the DFT, but never after the quotient, since division is not a linear operation (remember resistance operation). Thus, two different flow execution can be done:

- Method 1:
  1. Average windows of u, y in time domain
  2. Take DFT of average to get $\hat{Y}(k)$ and $\hat{U}(k)$
  3. Build quotient of DFTs, i.e. compute
  $$\hat{G}(j\omega_k) = \frac{\hat{Y}(k)}{\hat{U}(k)} \tag{8.35}$$
- Method 2:
  1. Take DFT of each window
  2. Average DFTs in frequency domain to get $\hat{Y}(k)$ and $\hat{U}(k)$
  3. Build quotient of average DFTs

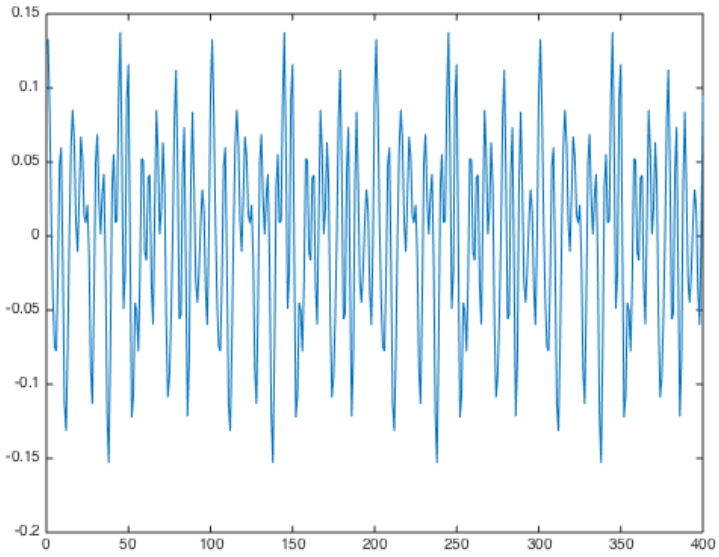


Figure 8.11: Multsine without positive interference, specially designed

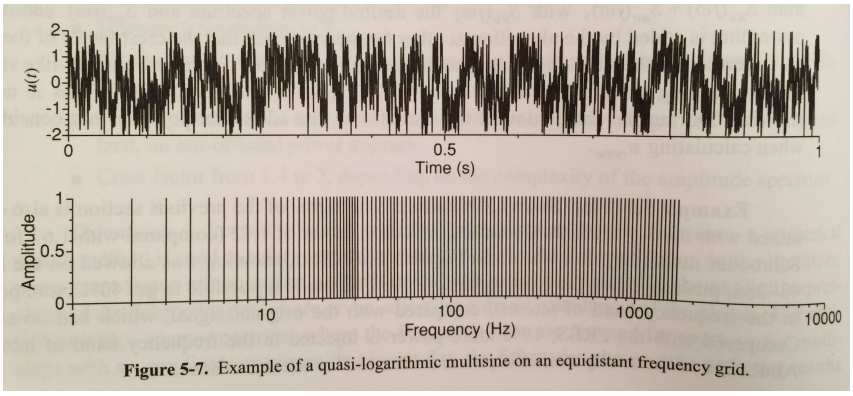


Figure 8.12: Multsine without positive interference

Assumptions for Simplified Analysis

Before starting the analysis, some assumptions has to be made so that the analysis becomes simpler. The main assumptions that are done can be summarised as:

1. The noise in all M data windows is uncorrelated.
2. The noise on $u(k)$ and $y(k)$ is also uncorrelated.
3. After the DFT computation(linear transformation with complex numbers), the noise on each $U(k)$ and $Y(k)$ is *circular complex normally distributed* (i.e. both real and imaginary parts are Gaussian with the same variance and independent) with variances $\sigma^2_U(k)$ and $\sigma^2_Y(k)$
4. Because of averaging over M independent samples, the noise on the averages, $N_{\hat{U}}(k)$ and $N_{\hat{Y}}(k)$, is of variance $\sigma^2_{\hat{U}}(k) = \frac{\sigma^2_U(k)}{M}$ and $\sigma^2_{\hat{Y}}(k) = \frac{\sigma^2_Y(k)}{M}$ (and assumed to have zero mean). Furthermore they enter the model for analysis as:

$$\hat{U}(k) = U_0(k) + N_{\hat{U}}(k) \quad \text{and} \quad \hat{Y}(k) = Y_0(k) + N_{\hat{Y}}(k) \tag{8.36}$$

with U_0 and Y_0 the true periodic values

Simplified Analysis

Once the assumptions were made, we can proceed with the analysis. The transfer function considering the error terms is:

$$\hat{G}(j\omega_k) = \frac{Y_0(k) + N_{\hat{Y}}(k)}{U_0(k) + N_{\hat{U}}(k)} = \frac{Y_0(k)}{U_0(k)} \frac{\Big(1 + \frac{N_{\hat{Y}}(k)}{Y_0(k)}\Big)}{\Big(1 + \frac{N_{\hat{U}}(k)}{U_0(k)}\Big)} \tag{8.37}$$

Applying a Taylor expansion and considering only the first order terms, the final expression that we obtain is:

$$\hat{G}(j\omega_k) \approx G_0(j\omega_k)\Big(1 + \frac{N_{\hat{Y}}(k)}{Y_0(k)} - \frac{N_{\hat{U}}(k)}{U_0(k)}\Big) \tag{8.38}$$

being

$$G_0(j\omega_k) = \frac{Y_0(k)}{U_0(k)} \tag{8.39}$$

the true transfer function.

Because the noises are zero mean, it is clear that no bias exist (up to first order Taylor expansion). Considering the first order Taylor expansion:

$$\hat{G}(j\omega_k) = G_0(j\omega_k)\Big(1 + \frac{N_{\hat{Y}}(k)}{Y_0(k)} - \frac{N_{\hat{U}}(k)}{U_0(k)}\Big) \tag{8.40}$$

The expected value of the transfer function is the true value of the transfer function, thus, we have an unbiased estimator,i.e.:

$$\mathbb{E}\{\hat{G}(j\omega_k)\} = G_0(j\omega_k) \tag{8.41}$$

Regarding the variance of the transfer function, we have due to the independence of the 2 noise terms that:

$$\sigma^2_{\hat{G}}(k) = |G_0(j\omega_k)|^2\Big(\frac{\sigma^2_{\hat{Y}}(k)}{|Y_0(k)|^2} + \frac{\sigma^2_{\hat{U}}(k)}{|U_0(k)|^2}\Big) \tag{8.42}$$

To this variance we still have to take the averaging into account, which gives a final variance for the transfer function:

$$\sigma^2_{\hat{G}}(k) = \frac{|G_0(j\omega_k)|^2}{M}\Big(\frac{\sigma^2_{Y}(k)}{|G_0(k)U_0(k)|^2} + \frac{\sigma^2_{U}(k)}{|U_0(k)|^2}\Big) \tag{8.43}$$

Thus, we can conclude that the quality of the estimation is better for higher signal to noise ratio (SNR) $\frac{|U_0(k)|}{\sigma_U(k)}$. This obviously justifies our desire to choose high amplitudes for the frequencies of interest.

8.3.5 Practical guidelines

In reality, when implementing a excitation signal to identify the system, there are three different alternatives for three different situations.

1. In theory, multisines are the most efficient way to get accuracy in limited time.
2. However, if the system is very fast compared to the human scale (almost nfinite testing time is possible), then frequency sweep might be the easiest way to identifying. That is way in high frequency electronic applications, frequency sweeps are used instead of multilines.
3. On the other hand, if the system is very slow compared to the human time scale, then a step response might be more wise to identify the system, because for the multsines we have to wait until the transients die out, and in very slow systems that might take many days. A step response won't give an accurate estimate of the system, but would give an estimator of the timing of the system, delays...
4. For those systems in the middle (Hz to kHz), multisines are definitely the most efficient implementation.

Chapter 9

Online estimation for dynamic systems

9.1 Recursive Least Squares

Before explaining the Kalman filter from the perspective of system identification, it is important to revise the algorithm called Recursive Least Squares, which was presented in Section 5.3, and it is convenient to remember.

This algorithm was basically used for online parameter/state estimation, i.e. to estimate the optimal parameters when the flow of incoming information is not finite but always updated, and the need for an algorithm that computes the update optimised state from the previous optimised state (and without increasing the computational load) is necessary.

Basically we saw that for linear models $y_k = \phi_k^\top \theta + \epsilon_k$ and i.i.d. Gaussian noise, we could set the following recursive algorithm to solve the problem stated above:

1. First of all, we had to start with some a-priori knowledge on θ in form of a mean $\hat{\theta}_0$ and inverse covariance Q_0, so that the first loop at $k = 1$ could be computed.

2. Then, at each time k, when a new measurement y_k arrives, the first thing to do is to compute the new inverse covariance Q_k as:
$$Q_k = Q_{k-1} + \phi_k \phi_k^\top \tag{9.1}$$
What that equation represents is the fact that adding measurements increases the "information", where the amount of information is measured by the inverse of the covariance matrix:

3. After the computation of the covariance, we can compute the new estimate $\hat{\theta}_k$ ("innovation update")
$$\hat{\theta}_k = \hat{\theta}_{k-1} + \underbrace{Q_k^{-1}\phi_k(y_k - \phi_k^\top \hat{\theta}_{k-1})}_{\text{"innovation"}} \tag{9.2}$$

In that algorithm Step 1 is only executed once, at the beginning, whereas the Steps 2 and 3 are done recursively for each new measurement. This problem can also be expressed in the form of a general optimisation problem as follows:

$$\hat{\theta}_k = \arg\min_\theta \quad (\theta - \hat{\theta}_0)^\top Q_0 \, (\theta - \hat{\theta}_0) \; + \; \sum_{i=1}^{k} (y_i - \phi_k^\top \theta)^2 \tag{9.3}$$

9.2 Recursive Least Squares for State Estimation: an approach to Kalman Filter derivation

Let's assume now that we have a known deterministic linear system defined by

$$x_{i+1} = A_i x_i \tag{9.4}$$

with also a linear measurement equation defined by:

$$y_i = C_i x_i + v_i \tag{9.5}$$

where v_i is a i.i.d. zero mean noise, and initial knowledge on the initial state is known and has the form of a mean $\hat{x}_0$ and inverse covariance Q_0.

Clearly, due to linearity, the state at any point x_k can be expressed as $x_k = A_{k-1} \cdots A_0 x_0$ and in turn the measurement y_k at any given point k becomes:

$$y_k = C_k A_{k-1} \cdots A_0 x_0 + v_k \tag{9.6}$$

Taking a look at the structure of this algorithm and comparing it with the RLS algorithm presented before, it is obvious that using $\theta \equiv x_0$, $\epsilon_k \equiv v_k$ and $\phi_k^\top = C_k A_{k-1} \cdots A_0$, this problem can be casted into the standard RLS framework defined before. Particularly, our recursion becomes:

$$\begin{aligned} Q_k &= Q_{k-1} + (C_k A_{k-1} \cdots A_0)^\top C_k A_{k-1} \cdots A_0 \\ \hat{\theta}_k &= \hat{\theta}_{k-1} + Q_k^{-1} (C_k A_{k-1} \cdots A_0)^\top (y_k - C_k A_{k-1} \cdots A_0 \hat{\theta}_{k-1}) \end{aligned} \tag{9.7}$$

However, this last expression is not really that helpful, since very often, what we are most interested in is the current state x_k rather than x_0, thus let's try to derive a recursive expression for x_k. Let us denote the optimal estimator of x_k given the data $(y_1, \ldots, y_m)$ as $\hat{x}_{[k|m]}$. Clearly,

$$\hat{x}_{[k|m]} = A_{k-1} \cdots A_0 \hat{\theta}_m \tag{9.8}$$

and

$$\underbrace{\operatorname{cov}\{x_{[k|m]}\}}_{=:P_{[k|m]}} = A_{k-1} \cdots A_0 \cdot Q_m^{-1} \cdot A_0^\top \cdots A_{k-1}^\top \tag{9.9}$$

Where $\hat{x}_{[k|m]}$ represents the best estimator of $\hat{x}_k$ given m measurements. Ideally we would like to obtain $\hat{x}_{[k|k]}$, to have always the state update with the most recent measurement data. If know we multiply every term of the first equation of (9.7) by $A_{k-1} \cdots A_0$ we obtain:

$$\begin{aligned} A_{k-1} \cdots A_0 \cdot \hat{\theta}_k &= A_{k-1} \cdots A_0 \cdot \hat{\theta}_{k-1} + A_{k-1} \cdots A_0 \cdot Q_k^{-1} (C_k A_{k-1} \cdots A_0)^\top (y_k - C_k A_{k-1} \cdots A_0 \hat{\theta}_{k-1}) \\ \hat{x}_{[k|k]} &= \hat{x}_{[k|k-1]} + P_{[k|k]} \cdot C_k^\top (y_k - C_k \hat{x}_{[k|k-1]}) \end{aligned} \tag{9.10}$$

where this expression is usually known as the update of the mean. Basically what it does is, it checks how good the prediction $\hat{x}_{[k|k-1]}$ is, and if it is not perfect it corrects it by using the covariance.

Furthermore, assuming for simplicity invertibility of $A_{k-1} \cdots A_0$, and multiplying every term of the first equation of (9.7) by $(A_{k-1} \cdots A_0)^{-1}$ on the right and by $(A_0^\top \cdots A_{k-1}^\top)^{-1}$ on the left, we obtain the covariance update step:

$$P_{[k|k]}^{-1} = P_{[k|k-1]}^{-1} + C_k^\top C_k \tag{9.11}$$

That equation basically means that the previous $P_{[k|k-1]}^{-1}$ predicted inverse covariance, is still modified by the new information coming in. Moreover, to obtain the $\hat{x}_{[k|k-1]}$ and $P_{[k|k-1]}^{-1}$, the so-called time propagation (or prediction) step, we can use directly:

$$\hat{x}_{[k|k-1]} = A_{k-1} \cdot \hat{x}_{[k-1|k-1]} \tag{9.12}$$

and

$$P_{[k|k-1]} = A_{k-1} \cdot P_{[k-1|k-1]} \cdot A_{k-1}^\top \tag{9.13}$$

Where this propagation step means to predict the future value of the variables, without new information coming in, prediction values which are checked and updated in the previously defined update step.

Thus, we can summary the RLS for State Estimation as the computation of two steps in order to compute the estimates $x_{[k|k]}$ and covariances $P_{[k|k]}$:

1. Prediction Step (before measurement):

$$\hat{x}_{[k+1|k]} = A_k \cdot \hat{x}_{[k|k]} \tag{9.14}$$
$$P_{[k+1|k]} = A_k \cdot P_{[k|k]} \cdot A_k^\top \tag{9.15}$$

2. Innovation Update Step (after measurement):

$$P_{[k|k]} = \left(P_{[k|k-1]}^{-1} + C_k^\top C_k\right)^{-1} \tag{9.16}$$
$$\hat{x}_{[k|k]} = \hat{x}_{[k|k-1]} + P_{[k|k]} \cdot C_k^\top (y_k - C_k \hat{x}_{[k|k-1]}) \tag{9.17}$$

Another thing to keep in mind is the meaning of P in this equations. If P represent the covariance of the estimator $\hat{x}$, then it is clear that the bigger the covariance the less we should trust our model. If we look at P in the innovation step, we see that the bigger P is, the more we change the updated state according to the new mesasurement. However, if P is small, it means that the covariance is small, therefore we should trust the model more than the measurements, and that is why in this case the expression $(y_k - C_k \hat{x}_{[k|k-1]})$ gets less weight than the previously predicted state $x_{[k|k-1]}$.

So far every equation derived is completely deterministic, we have just solved the problem of RLS. However, with all the previous equations in mind, a natural question should arise: can we interpret the $\hat{x}_{[k|k-1]}$ and $P_{[k|k-1]}$ as a-priori information on x_k based on a prediction model? If so, how can we model the uncertainty for that prediction model?

The answer to these two questions is that we can model the priori-information on x_k by $x_k = A_{k-1}x_{k-1} + w_{k-1}$, where w_{k-1} is a i.i.d. state noise on the state model with zero mean and covariance W_{k-1}. Does this make sense? In practice it does, from an intuitive point of view we can say that since no model is perfect, if we do not know anything about the model, a good way to approximate the uncertainty of a model is adding i.i.d. noise, so that we take into account that the model predictions are not 100% accurate.

Because of these perturbations on the state model, an extra term must be added to the equation of the prediction step where the covariance of the estimator is predicted. A good way to point this uncertainty is by not only predicting the new covariance by a linear mapping, but also add at each iteration the covariance of the noise, W_{k-1} so that the uncertainty on the model at each prediction step is accounted for. The predicted covariance is now defined by:

$$P_{[k+1|k]} = A_k \cdot P_{[k|k]} \cdot A_k^\top + W_k \tag{9.18}$$

This set of equations that we have derived are basically known as Kalman Filter.

9.3 Kalman Filter

In the previous section we have basically obtained the Kalman Filter, only one remark more is necessary. We can consider the more generally model, where we assume affine state and measurement models that are both perturbed by additive zero-mean Gaussian noise, i.e. we consider state space models of the following form:

$$x_{k+1} = A_k x_k + b_k + w_k \tag{9.19}$$
$$y_k = C_k x_k + d_k + v_k \tag{9.20}$$

where $w_k \sim \mathcal{N}(0, W_k)$ and $v_k \sim \mathcal{N}(0, V_k)$ are independent. In this case, the Kalman Filter prediction and innovation update steps are given by:

1. Prediction Step:

$$\hat{x}_{[k+1|k]} = A_k \hat{x}_{[k|k]} + b_k \tag{9.21}$$
$$P_{[k+1|k]} = A_k P_{[k|k]} A_k^\top + W_k \tag{9.22}$$

2. Innovation Update Step

$$P_{[k|k]} = \left(P_{[k|k-1]}^{-1} + C_k^\top V_k^{-1} C_k\right)^{-1} \tag{9.23}$$
$$\hat{x}_{[k|k]} = \hat{x}_{[k|k-1]} + P_{[k|k]} C_k^\top V_k^{-1} (y_k - C_k \hat{x}_{[k|k-1]} - d_k) \tag{9.24}$$

Note that from a computational point of view, it is beneficial to rewrite the innovation update step for the covariance estimate as follows:

$$\begin{aligned} P_{[k|k]} &= \left(P_{[k|k-1]}^{-1} + C_k^\top V_k^{-1} C_k\right)^{-1} \\ &= \left(P_{[k|k-1]}^{-1}\left(\mathbb{I} + P_{[k|k-1]} C_k^\top V_k^{-1} C_k\right)\right)^{-1} \\ &= \left(\mathbb{I} + P_{[k|k-1]} C_k^\top V_k^{-1} C_k\right)^{-1} P_{[k|k-1]} \end{aligned}$$

With this formulation, we do not need to invert the matrix $P_{[k|k-1]}$ which can become close to singular when the uncertainty of our estimate gets close to zero (in some directions). This is only one way of reformulating the innovation update step and there are indeed several other formulas.
Similarly to RLS, we can state the optimization problem that the Kalman Filter recursively solves as:

$$(x_0^*, \dots, x_N^*) = \arg\min_{x_0,\dots,x_N} \|x_0 - \hat{x}_0\|^2_{P_0^{-1}} + \sum_{i=1}^{N} \|y_i - C_i x_i - d_i\|^2_{V_i^{-1}} + \sum_{i=1}^{N-1} \|x_{i+1} - A_i x_i - b_i\|^2_{W_i^{-1}} \quad (9.25)$$

Despite the minimisation over $(x_0, \dots, x_N)$, the only thing that is of interest is the last value x_N which corresponds to our current state estimate $\hat{x}_{[k|k]}$. Note that this optimization problem that we can recursively solve using the Kalman Filter is the same problem we introduced in Section 7.4 as *trajectory estimation problem*.

9.4 Kalman Filter in continuous time[1]

So far the equations that we have derived corresponded to the discrete version of the Kalman Filter. However, we can easily transfer this knowledge to the continuous Kalman Filter considering first of all a continuous time model with noises w^c, v^c:

$$\begin{aligned} \dot{x}(t) &= A^c(t)x(t) + w^c(t) \\ y(t) &= C^c(t)x(t) + v^c(t) \end{aligned}$$

Noting that $w^c(t)$ has unit $[x]/[t]$ and $v^c(t)$ has unit $[y]$. Furthermore it is reasonable assumption to consider that we have white noises noises in both, state model and measurement, and with that we can define the covariance as diagonal matrix of the form:

- $\operatorname{cov}(w^c(t_1), w^c(t_2)) = \delta(t_1 - t_2) \cdot W^c$ where $[W^c] = [x]^2/[t]$
- $\operatorname{cov}(v^c(t_1), v^c(t_2)) \ = \delta(t_1 - t_2) \cdot V^c$ where $[V^c] = [y]^2 \cdot [t]$

We can then try to transfer this continuous model to discrete time, so that we can compare with the previously derived Kalman Filter. For that we can use small time step Δt and time points $t_k = k \cdot \Delta t$. Then, we can identifying the discrete variables as follow:

- $x_k = x(t_k)$.
- $w_k = \int_{t_k}^{t_{k+1}} w^c(t)\,\mathrm{d}t$. This makes sense, since the random walk theory says: $W = \Delta t \cdot W^c$ (the longer we wait, the more uncertain we become).
- $C_k = C(t_k)$.
- $y_k = \frac{1}{\Delta t}\int_{t_k}^{t_{k+1}} y(t)\,\mathrm{d}t$, and thus $v_k = \frac{1}{\Delta t}\int_{t_k}^{t_{k+1}} v^c(t)\,\mathrm{d}t$. That equation represents the fact that the measurement errors have a completely opposite behaviour than state noise, since because of the averaging, the covariance matrix shrinks with longer time intervals, i.e. $V = \frac{V_c}{\Delta t}$.

[1] not covered in the lecture

With that set, we can directly write the equivalent discrete time model as follows.

$$x_{k+1} = \underbrace{[I + \Delta t \cdot A^c(t_k)]}_{=:A_k} x_k + w_k$$
$$y_k = \underbrace{C^c(t_k)}_{=C_k} x_k + v_k$$

Where the covariances can are also described as: $\mathrm{cov}(w_k) = W = \Delta t \cdot W^c$ and $\mathrm{cov}(v_k) = V = \Delta t^{-1} \cdot V^c$

And with that we can set directly the discretised Kalman Filter in the case that we discretise the continuous time model:

- Prediction step (up to first order):

$$\hat{x}_{[k+1|k]} = \hat{x}_{[k|k]} + \Delta t \cdot A^c(t_k)\hat{x}_{[k|k]}$$
$$P_{[k+1|k]} = P_{[k|k]} + \Delta t \left[A^c(t_k)P_{[k|k]} + P_{[k|k]}A^c(t_k)^\top + W^c\right]$$

- Innovation Update Step:

$$P_{[k|k]} = \left(P^{-1}_{[k|k-1]} + \Delta t \cdot C_k^\top (V^c)^{-1} C_k\right)^{-1} \tag{9.26}$$
$$\hat{x}_{[k|k]} = \hat{x}_{[k|k-1]} + \Delta t \cdot P_{[k|k]} \cdot C_k^\top (V^c)^{-1}(y_k - C_k\hat{x}_{[k|k-1]})$$

By Taylor expansion, Eq. (9.26) becomes

$$P_{[k|k]} = P_{[k|k-1]} - \Delta t \cdot P_{[k|k-1]} C_k^\top (V^c)^{-1} C_k P_{[k|k-1]} \tag{9.27}$$

Finally adding both steps together, identifying $\hat{x}(t_k) \equiv \hat{x}_{[k+1|k]}$ and $P(t_k) \equiv P_{[k+1|k]}$, and taking the limit $\Delta t \to 0$, the previously discretised Kalman Filter yields the the two differential equations:

$$\dot{\hat{x}}(t) = A^c(t_k)\hat{x}(t) + P(t)C^c(t)^\top (V^c)^{-1}\left(y(t) - C^c(t)\hat{x}(t)\right)$$
$$\dot{P}(t) = A^c(t)P(t) + P(t)A^c(t)^\top + W^c - P(t)C^c(t)^\top (V^c)^{-1} C^c(t)P(t)$$

These continuous time equations are called the "Kalman-Bucy-Filter" and represent the Kalman Filter in continuous time.

9.5 Extended Kalman Filter

The Extended Kalman Filter (EKF) is the nonlinear version of the Kalman filter, i.e. it can be applied to any dynamic system of the form

$$x_{k+1} = f_k(x_k) + w_k \tag{9.28a}$$
$$y_k = g_k(x_k) + v_k \tag{9.28b}$$

where $f_k : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$ and $g_k : \mathbb{R}^{n_x} \to \mathbb{R}^{n_y}$ are nonlinear differentiable functions. Furthermore, we assume that w_k, v_k are independent and follow Gaussian distributions with zero mean and covariance matrices W_k and V_k respectively. Note that by augmenting the state to also include constant parameters and using the equivalence

$$f_k(x_k) \;\leftrightarrow\; f(x_k, u_k),$$
$$g_k(x_k) \;\leftrightarrow\; g(x_k, u_k),$$

any model of the form introduced in Section 7.4 can be rewritten in the form of 9.28.

The main idea of the EKF is to first linearize f_k and g_k at the current state estimate and then apply the same update and prediction formulas we derived in Section 9.3 for the case of linear systems. Let $\bar{x}_k$ denote the linearization point (which we will specify later on). The first order Taylor expansion of f_k at $\bar{x}_k$ is then given by

$$f_k(x_k) = \underbrace{f_k(\bar{x}_k) + A_k(x_k - \bar{x}_k)}_{f_k^{\text{LIN}}(x_k;\bar{x}_k)} + O\left(\|x_k - \bar{x}_k\|^2\right)$$

where $A_k = \frac{\partial}{\partial x_k} f_k(\bar{x}_k)$. The function $f_k^{\text{LIN}}(x_k;\bar{x}_k)$ denotes the linearization of f_k at $\bar{x}_k$. Similarly, linearizing the function g at the linearization point $\bar{\bar{x}}_k$ yields

$$g_k(x_k) = \underbrace{g_k(\bar{\bar{x}}_k) + C_k(x_k - \bar{\bar{x}}_k)}_{g_k^{\text{LIN}}(x_k;\bar{\bar{x}}_k)} + O\left(\|x_k - \bar{\bar{x}}_k\|^2\right)$$

where $C_k = \frac{\partial}{\partial x_k} g_k(\bar{\bar{x}}_k)$. Thus, the linearized system is given by:

$$\begin{aligned} x_{k+1} &= f_k(\bar{x}_k) + A_k(x_k - \bar{x}_k) + w_k \\ &= A_k x_k + \underbrace{f_k(\bar{x}_k) - A_k\bar{x}_k}_{b_k} \\ y_k &= g_k(\bar{\bar{x}}_k) + C_k(x_k - \bar{\bar{x}}_k) \\ &= C_k x_k + \underbrace{g_k(\bar{\bar{x}}_k) - C_k\bar{\bar{x}}_k}_{d_k} \end{aligned}$$

where $A_k = \frac{\partial}{\partial x_k} f_k(\bar{x}_k)$ and $C_k = \frac{\partial}{\partial x_k} g_k(\bar{\bar{x}}_k)$. Choosing the current estimate $\hat{x}_{[k|k]}$ as linearization point $\bar{x}_k$ and applying the update step we derived in Section 9.3, we obtain

$$\begin{aligned} \hat{x}_{[k|k+1]} &= A_k\hat{x}_{[k|k]} + f_k(\bar{x}_k) - A_k\bar{x}_k = f_k\left(\hat{x}_{[k|k]}\right) \\ P_{[k|k+1]} &= A_k P_{[k|k]} A_k^\top + W_k \end{aligned}$$

where $A_k = \frac{\partial}{\partial x_k} f_k(\hat{x}_{[k|k]})$. For the innovation update step, we choose our current prediction $\hat{x}_{[k+1|k]}$ as linearization point $\bar{\bar{x}}_{k+1}$ and obtain:

$$\begin{aligned} P_{[k+1|k+1]} &= \left(P_{[k+1|k]}^{-1} + C_{k+1}^\top V_{k+1}^{-1} C_{k+1}\right)^{-1} \\ \hat{x}_{[k+1|k+1]} &= \hat{x}_{[k+1|k]} + P_{[k+1|k+1]} C_{k+1}^\top V_{k+1}^{-1}\left(y_{k+1} - C_{k+1}\hat{x}_{[k+1|k]} - g_{k+1}(\bar{\bar{x}}_{k+1}) + C_{k+1}\bar{\bar{x}}_{k+1}\right) \\ &= \hat{x}_{[k+1|k]} + P_{[k+1|k+1]} C_{k+1}^\top V_{k+1}^{-1}\left(y_{k+1} - g_{k+1}(\hat{x}_{[k+1|k]})\right) \end{aligned}$$

where $C_k = \frac{\partial}{\partial x_{k+1}} g_{k+1}(\hat{x}_{[k+1|k]})$. Analogously to the linear Kalman Filter, the term $y_{k+1} - g(x_{[k+1|k]})$ corresponds to the prediction error, i.e. the deviation of the predicted output $g_{k+1}(\hat{x}_{[k+1|k]})$ and the measured output y_{k+1}. If our prediction coincides with the actual measurement, this term is zero and we do not update the current state estimate.
In summary, the prediction and innovation update step are given by

1. Prediction Step:

$$\hat{x}_{[k+1|k]} = f_k\left(\hat{x}_{[k|k]}\right) \tag{9.29}$$
$$P_{[k+1|k]} = A_k P_{[k|k]} A_k^\top + W_k \tag{9.30}$$

   where $A_k = \frac{\partial}{\partial x_k} f_k(\hat{x}_{[k|k]})$.

2. Innovation Update Step

$$P_{[k+1|k+1]} = \left(P_{[k+1|k]}^{-1} + C_{k+1}^\top V_{k+1}^{-1} C_{k+1}\right)^{-1} \tag{9.31}$$
$$\hat{x}_{[k+1|k+1]} = \hat{x}_{[k+1|k]} + P_{[k+1|k+1]} C_{k+1}^\top V_{k+1}^{-1}\left(y_{k+1} - g_{k+1}\left(\hat{x}_{[k+1|k]}\right)\right) \tag{9.32}$$

   where $C_{k+1} = \frac{\partial}{\partial x_{k+1}} g_{k+1}(\hat{x}_{[k+1|k]})$.

As for the linear Kalman Filter, the Extended Kalman Filter estimates the current state of the system in a recursive manner. The computations that have to be performed in each recursive step are rather cheap and in particular constant for each timestep. In contrast to the linear Kalman Filter, the EKF is no longer an optimal estimator due to the linearization of the dynamics. Thus, we have to carefully choose the initial guess of the state of the system, as the EKF might not recover from a large initial estimation error and diverge from the actual system state.

An alternative state estimation method which is closely related to EKF is the so-called *Unscented Kalman Filter* (UKF) [JU04]. While the EKF propagates the covariance of the estimate through the linearized model, the UKF tries to capture the nonlinearity of the system by (deterministically) sampling the nonlinear dynamics at several so-called *sigma points*. These sigma points are propagated through the nonlinear function f and then combined to compute the predicted state and covariance.

Yet another approach to state estimation are *Particle Filters*. Here, the main idea is to approximate the conditional probability distribution of the current state of the system given the available observations by *particles* which are random samples from this distribution. As the number of particles that are required to effectively represent the conditional distribution increases exponentially with the dimension of the state space, these methods are usually only applicable for low dimensional systems.

Another powerful alternative to EKF is presented in the next section.

9.6 Moving Horizon Estimation for State Estimation

In this section, we introduce *Moving Horizon Estimation* (MHE) as an optimization-based approach to (nonlinear) state estimation. As for the EKF, we consider a nonlinear dynamic system of the following form:

$$\begin{aligned} x_{k+1} &= f_k(x_k) + w_k \\ y_k &= g_k(x_k) + v_k \end{aligned}$$

with random state and measurement noise $w_k \in \mathbb{R}^{n_x}$ and $v_k \in \mathbb{R}^{n_y}$ with known probability distributions that are described by the following PDFs:

$$p_{w_k}(w_k) = \text{const} \cdot \exp(-\phi_k(w_k)) \tag{9.33}$$

$$p_{v_k}(v_k) = \text{const} \cdot \exp(-\psi_k(v_k)) \tag{9.34}$$

Additionally, we assume that we have some prior knowledge of the initial state of the system which is given in terms of a prior probability distribution of x_0 with PDF:

$$p_0(x_0) = \text{const} \cdot \exp(-\Pi_0(x_0)).$$

Note that for a zero-mean Gaussian prior as well as zero-mean Gaussian state and measurement noise with covariance matrix Σ, we would have $\Pi_0(\cdot) = \phi_k(\cdot) = \psi_k(\cdot) = \frac{1}{2}\| \cdot \|^2_{\Sigma^{-1}}$. With the more general formulation, it is, however, possible to assume non-Gaussian noise distributions, e.g. Laplacian noise.

Let $\theta = (x_0, \dots, x_N)$. The *Full Information Estimation* (FIE) [RM09] problem can then be formulated as:

$$\hat{\theta}^{\text{FIE}} = \theta^* = \arg\min_{\theta} \; \Pi_0(x_0) + \sum_{k=0}^{N} \psi_k(y_k - g_k(x_k)) + \sum_{k=0}^{N-1} \phi_k(x_{k+1} - f_k(x_k)) \tag{9.35}$$

If $\theta^* = (x_0^*, \dots, x_N^*)$ is the solution to the above optimization problem, then our current state estimate $\hat{x}^{\text{FIE}}_{[N|N]}$ would be given by x_N^*. In this formulation, we estimate the complete state trajectory $(x_0, \dots, x_N)$ given all the available data $(y_0, \dots, y_N)$. Thus, the size of the FIE problem increases with every iteration. Eventually, solving the above optimization problem will become computationally intractable. Especially for online state estimation, where we have typically only very limited computation time available, we would require an estimation algorithm whose computational cost is constant for every iteration.

To meet this requirement, we introduce *Moving Horizon Estimation* (MHE) which is an approximation of the FIE problem. With MHE, we consider only a fixed number of measurements and summarize the contributions of all previous measurements in an additional term which we call *arrival cost*.

Let $\theta = (x_{N-M}, \dots, x_N)$ where M denotes the horizon length. The MHE problem is then defined as

$$\hat{\theta}^{\text{MHE}} = \arg\min_{\theta} \Pi_{N-M}(x_{N-M}) + \sum_{k=N-M}^{N} \psi_k(y_k - g_k(x_k)) + \sum_{k=N-M}^{N-1} \phi_k(x_{k+1} - f_k(x_k)) \tag{9.36}$$

The term $\Pi_{N-M}(x_{N-M})$ is called *arrival cost* and tries to summarize the contributions of the previous measurements $(y_0, \dots, y_{N-M})$ to the cost function.
In the following, we discuss three different variants of computing the arrival cost.

9.6.1 Exact Arrival Cost

In theory, the exact arrival cost is given by

$$\Pi_{N-M}(x_{N-M}) = \min_{x_0,\dots,x_{N-M-1}} \Pi_0(x_0) + \sum_{k=0}^{N-M-1} \psi_k(y_k - g_k(x_k)) + \phi_k(x_{k+1} - f_k(x_k)) \tag{9.37}$$

where x_{N-M} enters the optimization problem as a parameter. Note that the MHE problem (9.36) with true arrival cost (9.37) is equivalent to the FIE problem (9.35). Obviously, computing the true arrival cost and then solving the MHE problem is as expensive as solving the FIE problem.

There is, however, a special case in which we can compute the arrival cost analytically and that is the case of linear systems with Gaussian state and measurement noise as well as a Gaussian prior. Under these assumptions, the arrival cost and the MHE problem (and thus also the FIE problem) can be solved analytically yielding the linear Kalman Filter.
For the general nonliner case, where computing the exact arrival cost is computationally intractable, we have to use an approximation in order to obtain a state estimator whose computational cost is fixed for every iteration.

9.6.2 Zero Arrival Cost

The simplest approximation of the arrival cost is to completely ignore all measurements that are outside of the horizon by setting

$$\Pi_{N-M}(x_{N-M}) = 0.$$

9.6.3 Quadratic Arrival Cost

Another way of approximating the arrival cost is to use a quadratic cost term, i.e.

$$\Pi_{N-M}(x_{N-M}) = \frac{1}{2} \left\| x_{N-M} - \bar{x}_{N-M} \right\|^2_{P^{-1}_{N-M}} .$$

which implicitly means that we assume a Gaussian distribution for x_{N-M},

$$x_{N-M} \sim \mathcal{N}\left(\bar{x}_{N-M}, P_{N-M}\right).$$

A quadratic approximation of the arrival cost has the advantage that the resulting MHE problem is a convex optimization problem assuming that ϕ_k and ψ_k are convex functions as well.
There are several approaches for computing the values of $\bar{x}_{N-M}$ and P_{N-M}. We will briefly discuss two of them.

1. A simple way of computing $\bar{x}_{N-M}$ and P_{N-M} is to run an additional extended Kalman Filter that lags behind M timesteps. We then use the prediction $\hat{x}_{[N-M|N-M-1]}$ with covariance $P_{[N-M|N-M-1]}$ from the EKF to define the arrival cost.

2. Another possible approach is to use an EKF with a more up-to-date linearization point, i.e. we use our current best estimate that we obtained using all the available data. Assume that we just solved the MHE problem on the horizon $(N-M)$ to N with arrival cost defined by $\bar{x}_{N-M}$ and P_{N-M}. To update the arrival cost, we use our current best estimate $\hat{x}^{\text{MHE}}_{[N-M|N]}$ as linearization point for both update and prediction step of the EKF. This yields the following formulas:

(a) Innovation Update Step for the Arrival Cost Update:

$$P_{N-M}^{+} = \left(P_{N-M}^{-1} + \tilde{C}_{N-M}^{\top} V_{N-M}^{-1} \tilde{C}_{N-M}\right)^{-1} \tag{9.38}$$

$$\bar{x}_{N-M}^{+} = \bar{x}_{N-M} + P_{N-M}^{+} \tilde{C}_{N-M}^{\top} V_{N-M}^{-1} \left(y_{N-M} - g_{N-M}\left(\bar{x}_{N-M}\right)\right) \tag{9.39}$$

where $\tilde{C}_{N-M} = \frac{\partial}{\partial x} g_{N-M}\left(\hat{x}_{[N-M|N]}^{\text{MHE}}\right)$.

(b) Prediction Step for the Arrival Cost Update:

$$\bar{x}_{N-M+1} = f_{N-M}\left(\bar{x}_{N-M}^{+}\right) \tag{9.40}$$

$$P_{N-M+1} = \tilde{A}_{N-M} P_{N-M}^{+} \tilde{A}_{N-M}^{\top} + \tilde{W}_{N-M} \tag{9.41}$$

where $\tilde{A}_{N-M} = \frac{\partial}{\partial x} f_{N-M}\left(\hat{x}_{[N-M|N]}^{\text{MHE}}\right)$.

In practice, we usually choose $\tilde{W}_{N-M} = W_{N-M} + Q_{\text{s}}$ with $Q_{\text{s}} \succ 0$. Note that $P_{N-M+1} \succ \tilde{W}_{N-M} \succ Q_{\text{s}}$, i.e. by adding Q_{s} we can ensure that P_{N-M+1} is always well-conditioned.
Additionally, by introducing this larger covariance $\tilde{W}_{N-M}$, we decrease the influence of measurements outside of the horizon and account for linearization errors. Note that if we would choose $\tilde{W}_{N-M}$ *infinitely large*, we obtain the zero arrival cost approximation that we introduced in the previous subsection.

9.7 Moving Horizon Estimation for State and Parameter Estimation

In this section, we extend the MHE formulation that we introduced in the previous section to also account for unknown parameters p included in our model formulation. Thus, we consider nonlinear dynamic systems of the following form:

$$\begin{aligned} x_{k+1} &= f_k(x_k, p) + w_k \\ y_k &= g_k(x_k, p) + v_k \end{aligned}$$

with constant parameters $p \in \mathbb{R}^{n_p}$ and random state and measurement noise $w_k \in \mathbb{R}^{n_x}$, $v_k \in \mathbb{R}^{n_y}$ with PDFs as given in (9.33) and (9.34). Additionally, we assume a prior probability distribution of x_0 given by:

$$p_0(x_0, p) = \text{const} \cdot \exp(-\Pi_0(x_0, p)).$$

As introduced in the previous section, the MHE problem for horizon length M and $\theta = (p, x_{N-M}, \ldots, x_N)$ is given by

$$\hat{\theta}^{\text{MHE}} = \arg\min_{\theta} \; \Pi_{N-M}(x_{N-M}, p) + \sum_{k=N-M}^{N} \psi_k(y_k - g_k(x_k, p)) + \sum_{k=N-M}^{N-1} \phi_k(x_{k+1} - f_k(x_k, p)). \tag{9.42}$$

As stated in the previous section, there are several ways to compute the arrival cost. In the following, we discuss only a quadratic approximation of the arrival cost. To this end, we first augment the state x to also include the parameter p, i.e. we define $z_k = (x_k, p_k)$. In addition, we allow the parameter to (slightly) vary with respect to the previous MHE problem, i.e. for updating the arrival cost we assume the following dynamics:

$$z_{k+1} = \tilde{f}_k(z_k) + \tilde{w}_k = \begin{bmatrix} f_k(x_k, p_k) \\ p_k \end{bmatrix} + \begin{bmatrix} w_k + q_{\text{s}} \\ q_{\text{p}} \end{bmatrix} \tag{9.43}$$

where $\tilde{w}_k \sim \mathcal{N}(0, \tilde{W}_k)$ and the covariance matrix $\tilde{W}_k$ is given by

$$\tilde{W}_k = \begin{bmatrix} W_k + Q_{\text{s}} & 0 \\ 0 & Q_{\text{p}} \end{bmatrix}$$

with $Q_{\text{s}} \succ 0$, $Q_{\text{p}} \succ 0$.

With the augmented state, a quadratic approximation of the arrival cost is given by

$$\Pi_{N-M}(x_{N-M}, p_{N-M}) = \tilde{\Pi}_{N-M}(z_{N-M}) = \frac{1}{2} \left\| z_{N-M} - \bar{z}_{N-M} \right\|^2_{P^{-1}_{N-M}} .$$

The arrival cost for the next MHE problem, i.e. $\bar{z}_{N-M+1}$ and P_{N-M+1}, can then be computed from equations (9.38) to (9.41) using the linearization of the augmented dynamics (9.43) at the current best estimate $\hat{z}^{\mathrm{MHE}}_{[N-M|N]}$.

As in the previous section, we choose a larger covariance matrix $\tilde{W}_k$ to ensure invertibility of P_{N-M+1} and to diminish the influence of measurements outside of the horizon.
In this formulation, the parameter p is constant only on the horizon, i.e. within a single MHE problem, and it might vary with respect to the previous MHE problem. We can control how much the value of p changes by choosing the covariance matrix Q_{p} accordingly. Note that with $Q_{\mathrm{p}} = 0$, which corresponds to a constant value of p, the matrix P_{N-M+1} would not be invertible.

Appendices

Appendix A

Optimization

A.1 Newton-type optimization methods

Presented optimisation problems have as a common feature linearity on the estimator, i.e. the model $M(x)$ describes the measurements y_N as a linear function of x: $M(x) = \Phi \cdot x$.

Furthermore, it has always been assumed that there is Gaussian noise on the measurements which can be modeled with a covariance matrix Σ_x. When both facts are combined, the classical Weighted Linear Least Squares Problem appears, and its analytical solution $\hat{x}_{\mathrm{WLS}} = (\Phi^\top \cdot W \cdot \Phi)^{-1} \cdot \Phi^\top \cdot W \cdot y$ can be easily extracted.

In general, $M(x)$ is not linear, and the measurement noise might not be Gaussian, which leads to a general problem where a non-analytical solution needs to be found. In such cases, it is important to have a method which can find a local or even global minimum of the problem. In order to present a general optimisation method, let us first recall the general optimality conditions.

A.1.1 Optimality conditions

$$\arg\min_{x \in \mathbb{R}^d} f(x), \quad f \in \mathbb{R} \tag{A.1}$$

Let (A.1) be a general optimisation problem, then the following conditions can be established:

- **First order necessary condition (FONC):**
  If $x^\star$ minimizes (A.1) then $\nabla f(x^\star) = 0$
- **Second order necessaary condition (SONC):**
  If $x^\star$ minimizes (A.1) then $\nabla^2 f(x^\star) \succeq 0$
- **Second order sufficient condition (SOSC):**
  If $\nabla^2 f(x^\star) \succ 0$ and $\nabla f(x^\star) = 0$ then $x^\star$ minimizes (A.1).
- **FONC for convex functions:**
  $x^\star$ is a global minimizer of (A.1) $\iff$ $\nabla f(x^\star) = 0$ (if $f(x)$ is convex).

A.1.2 Descent direction methods

Let again consider the general optimisation problem:

$$\arg\min_{x \in \mathbb{R}^d} f(x), \quad f \in \mathbb{R} \tag{A.2}$$

By the optimisation conditions, we could calculate all the stationary points ($\nabla f(x) = 0$) and then evaluate their Hessian to see whether it is positive definite. However, solving the set of equations $\nabla f(x) = 0$ is usually a very difficult task.

Because of that, we will introduce an iterative algorithm for finding the stationary points instead of trying to find the analytical solution. The iterative algorithms that we will present have the form of:

$$x_{k+1} = x_k + t_k \cdot d_k, \quad k = 0, 1, 2, \ldots \tag{A.3}$$

where $d_k \in \mathbb{R}^d$ it is called a descent direction and $t_k \in \mathbb{R}$ the stepsize. In order to choose the step size there are many different algorithms, three popular of which are: constant step size, exact line search and backtracking. However, since the scope of this lecture is not an optimisation class, we encourage those readers who are more curious to take a look at one of the excellent textbooks [NW06], [BV04] or [Bec14]. In the case of a descent direction, it is important however to look at its definition and one of its property, so any descent direction method can be understood.

Definition 17 (Descent direction) *A vector $d_k \in \mathbb{R}^d$ is called a descent direction of f at x if the directional derivative is negative:*

$$f'(x; d) = \nabla f(x)^\top \cdot d < 0 \tag{A.4}$$

One of the most important properties of descent directions is that steps small enough along this directions lead to a decrease in the objective function:

Lemma 3 (descent property of descent direction) *Let f be a continuous differentiable function over $\mathbb{R}^d$, and let $x \in \mathbb{R}^d$. Let a vector $d_k \neq 0 \in \mathbb{R}^d$ be a descent direction of f at x. Then there exists and $\epsilon > 0$ such that:*

$$f(x + t \cdot d_k) < f(x) \tag{A.5}$$

for any $t \in (0, \epsilon]$.

With the last lemma in mind, it can be understood that the idea of algorithm (A.3) is to iterate over the objective function, reducing its value at each iteration by choosing a descent direction and suitably short stepsize.

Gradient or Steepest Descent Method

It can be proved that the direction represented by the negative gradient $d_k = -\nabla f(x)$ is a descent direction. Thus, one of the well-known implementation of algorithm (A.3) takes the form of:

$$x_{k+1} = x_k - t_k \cdot \nabla f(x_k), \quad k = 0, 1, 2, \ldots \tag{A.6}$$

Which is iterated until $\|\nabla f(x_{k+1})\| < \epsilon$. One can show that the negative gradient points into the direction of steepest descent in Euclidean space. Though this sounds like a good property, the gradient method suffers from very slow convergence in practice.

A.1.3 Newton's method

The steepest descent method only uses first order information. Let us now assume that f is twice continuously differentiable, and let us introduce a method that uses second order information. The main idea is that this new method tries to locally minimise the quadratic Taylor approximation of $f(x)$ at some point x_k. Considering first the quadratic approximation of $f(x)$ around the vicinity of x_k:

$$f(x) = f(x_k) + \nabla f(x_k)^\top \cdot (x - x_k) + \frac{1}{2} \cdot (x - x_k)^\top \nabla^2 f(x_k) \cdot (x - x_k) \tag{A.7}$$

It can be easily calculated the point that minimises such a expression as:

$$x_{k+1} = \arg\min_{x \in \mathbb{R}^d} f(x_k) + \nabla f(x_k)^\top \cdot (x - x_k) + \frac{1}{2} \cdot (x - x_k)^\top \nabla^2 f(x_k) \cdot (x - x_k) \tag{A.8}$$

which is only defined if $\nabla^2 f(x_k)$ is positive definite. The unique minimiser of (A.8) is given by

$$\nabla f(x_k) + \nabla^2 f(x_k) \cdot (x_{k+1} - x_k) \tag{A.9}$$

which leads to the exact Newthon method defined by:

$$x_{k+1} = x_k - (\nabla^2 f(x_k))^{-1} \cdot \nabla f(x_k) \tag{A.10}$$

It can be seen that Newton's method is just a special case of a scaled gradient method where the step size t is substituted by the inverse of the Hessian. Computing the Hessian can be costly but in practice the Newton Method performs better than the gradient method.

Gauss-Newton method

If we have a general Non-linear Least Squares problem of the form:

$$\arg\min_{x\in\mathbb{R}^d} f(x) = \arg\min_{x\in\mathbb{R}^d} \frac{1}{2}\cdot\|y_N - M(x)\|_2^2 \tag{A.11}$$

a good and faster approximation of Newton's iterative method is easy to prove and can be introduced.

Definition 18 (Gauss-Newton's iterative method for finding stationary points) *Let $\hat{x}$ be the optimal solution of the problem* (A.11)*, then $\hat{x}$ is also a stationary point of $f(x)$, and can be found with the iterative method described below:*

$$\begin{aligned} x_{k+1} &= x_k - \left(\frac{\partial M}{\partial x}(x_k)^\top \cdot \frac{\partial M}{\partial x}(x_k)\right)^{-1} \cdot \frac{\partial M}{\partial x}^\top (x_k)\cdot(M(x_k)-y) = \\ &= \arg\min_{x\in\mathbb{R}^d} \frac{1}{2}\cdot\|(M(x_k)-y) + \frac{\partial M}{\partial x}(x_k)\cdot(x-x_k)\| \end{aligned} \tag{A.12}$$

Proof: In order to use Newton's method the Gradient $\nabla f(x)$ and the Hessian $\nabla^2 f(x)$ must be calculated, so let's derive an expression for them in the case of problem (A.11):

$$\begin{aligned} \nabla f(x) = & \\ &= \frac{\partial}{\partial x}[\frac{1}{2}(M(x)-y_N)^\top\cdot(M(x)-y_N)]^\top = \\ &= \frac{\partial M}{\partial x}(x)^\top\cdot(M(x)-y_N) = \sum_{k=1}^{N}\nabla M_k(x)\cdot(M_k(x)-y(k)) \end{aligned} \tag{A.13}$$

$$\begin{aligned} \nabla^2 f(x) = & \\ &= \frac{\partial}{\partial x}(\frac{\partial}{\partial x}[(M(x)-y_N)^\top\cdot(M(x)-y_N)]^\top) = \\ &= \frac{\partial}{\partial x}\left[\sum_{k=1}^{N}\nabla M_k(x)\cdot(M_k(x)-y(k))\right] = \\ &= \sum_{k=1}^{N}\nabla^2 M_k(x)\cdot(M_k(x)-y(k)) + \sum_{k=1}^{N}\nabla M_k(x)\cdot\nabla M_k(x)^\top \approx \\ &\approx \frac{\partial M}{\partial x}(x)^\top\cdot\frac{\partial M}{\partial x}(x) \end{aligned} \tag{A.14}$$

Where the approximation $\sum_{k=1}^{N}\nabla^2 M_k(x)\cdot(M_k(x)-y(k)) \ll \sum_{k=1}^{N}\nabla M_k(x)\cdot(\nabla M_k(x))^\top$ is made, which is usually correct since normally either $\nabla^2 M_k(x)$ or $(M_k(x)-y(k))$ are small.

Taking into account these two results, Equation (A.12) is directly obtained substituting the values of $\nabla f(x)$ and $\nabla^2 f(x)$ into the Newthon's method algorithm defined by Equation (A.10).

Appendix B

Differential Equations

On chapter 6.2.1 ODE and DAE are explained. This appendix intends to expand this theory by defining PDEs and DDEs.

B.1 Partial Differential Equations (PDE)

Definition 19 (Partial Differential Equations) *A Partial Differential Equations (PDE) is a differential equation which has partial derivatives of several variables not only with respect of time t, but also with respect of some spatial coordinates x. We can name the solution as $u(t, x)$, which represents the vector of variables. Then, the general expression of a PDE can be written as:*

$$f(u, \frac{\partial u}{\partial t}, \ldots, \frac{\partial^n u}{\partial t^n}, x, \frac{\partial u}{\partial x}, \ldots, \frac{\partial^m u}{\partial x^m}) = 0 \tag{B.1}$$

where $n, m \in \mathbb{Z}^+$. Please note that here u and x have totally different meaning than in the rest of the text, u is the vector of variables, and x the set of spatial coordinates.

PDEs typically arise from spatially distributed parameters, and because of that they are also called "distributed parameter systems". The easiest example we can illustrate is the heat (diffusion) equation in one dimension, also known as Fick's Second Law of Diffusion.

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2} \tag{B.2}$$

with D diffusion constant

In order to solve any PDE, it is necessary to specify boundary conditions in space (what is happening at the boundaries of the system) and initial conditions at time zero, i.e. the value for $u(x, 0)$ must be known for any x. Since in theory x is continuous, the set of initial conditions must be given by a profile in space, i.e. the initial conditions form a set of infinite states.

Because solving the set of equations with infinitely initial conditions can be extremely hard, normally the spatial derivatives are discretised keeping the time derivatives as continuous, so that we generate a set of ODEs, where each ODE represents the numerical solution of the time evolution of one of these discretised $u(k)$, and then the ODE solver can be used. The method of discretising all variables but one to solve a PDE is called "method of lines".

In order to discretise the spatial derivative many different algorithms exists, among the most famous ones, one could name the Finite Element Method (FEM), the Finite Volume Method or the Finite Difference Method.

B.1.1 Examples of systems described by a PDE

- Any temperature profile since it is described always by the Fick's Second Law of Diffusion. The temperature profile can be in a microchip, a water tank, a wall or even the inner part of the earth.
- Any fluid flow, since it is based on the Navier-Stokes PDE. For instance any airflow in a computer, or around an airplane, in a building or in the atmosphere. The algorithm/field that takes care of such PDEs is know as computational fluid dynamics (CFD).

- The growth of bacteria in a rotten vegetable.
- Chemical concentrations in a tubular reactor

Let us derive the PDE in the case of the Heat equation defined by Fick's Second Law of Diffusion. We can describe the heat equation as:

$$\frac{\partial u(x,t)}{\partial t} = D\frac{\partial^2 u(x,t)}{\partial x^2} \tag{B.3}$$

with $x \in [0,1]$, and with boundary conditions: $u(0,t) = \sin(t), u(1,t) = 0$. We can imagine this situation as having any sort of 1D heat transmission, where in one side the temperature remains constant and equal to 0, and in the other side the temperature is a sine wave due to some outside temperature variations, e.g. due to the day and night differences.

We can then apply the method of lines, i.e. keep the time derivatives while applying finite differences to the spatial derivatives. If then we use a grid of size $\Delta x = 1/N$, we can make a spatial discretisation of u as $u_k \approx u(k \cdot \Delta x, t)$ with $k \in [1, N-1]$. Then, applying the approximation of a derivative for discrete variables $\frac{\Delta u}{\Delta x}$, the final discretised ODE is obtained:

$$\dot{u}_k = D\frac{(u_{k+1} - 2u_k + u_{k-1})}{(\Delta x)^2} \tag{B.4}$$

where since $k \in [1, N-1]$, the values of u_0 and u_N are undefined but needed. However, they are given when we incorporate the boundary conditions to the problem:

$$u_0 = \sin(t) \quad \text{and} \quad u_N = 0 \tag{B.5}$$

Finally we should set the initial condition, e.g. $u(x,0) = 0$, i.e. $u_0(0) = u_1(0) = \ldots = u_N(0) = 0$. With that, we obtain a ODE where each variable of the ODE represents one of the discretised spatial states of the problem, and with such an ODE we can use for instance the solver `ode15s` from Matlab in order to simulate the system

B.1.2 MATLAB example

In order to see the whole procedure, it is good to solve the previous example using Matlab.

1. The first thing to do is to set the discretised ODE, which given a certain time and profile u calculates the derivative of each of the discretised spatial states:

```
function [ udot] = mypde(t,u )
N=20; D=0.1; udot=zeros(N,1);
u0=sin(t);
udot(1)=N*N*D*(u0-2*u(1)+u(2));
for k=2:N-1
udot(k)=N*N*D*(u(k-1)-2*u(k)+u(k+1));
end
uN=0;
udot(N)=N*N*D*(uN-2*u(N)+uN);
```

2. Then, the initial conditions must be set for each of the states u: `u0=zeros(20,0);`

3. Finally we can simulate each one of the spatial discretised states on a time interval calling the ODE solver:

```
[tout,uout]=ode15s(@mypde, [0 10], u0)
figure(1); plot(tout,yout);
figure(2); surf(tout,linspace(0,1,20),uout')
```

On Figure B.1 we can observe the evolution of some of the states through time, only some of them are represented for the sake of visualisation. We can see how the evolution through time is to follow the sine wave that enters the system, but with a attenuation in each sequentially state. That is the expected behaviour in any diffusion process, where the places with lower concentration (in this case temperature) get their values increased from the places where the concentration is bigger, but unless the process last forever, the concentrations never get to be equal.

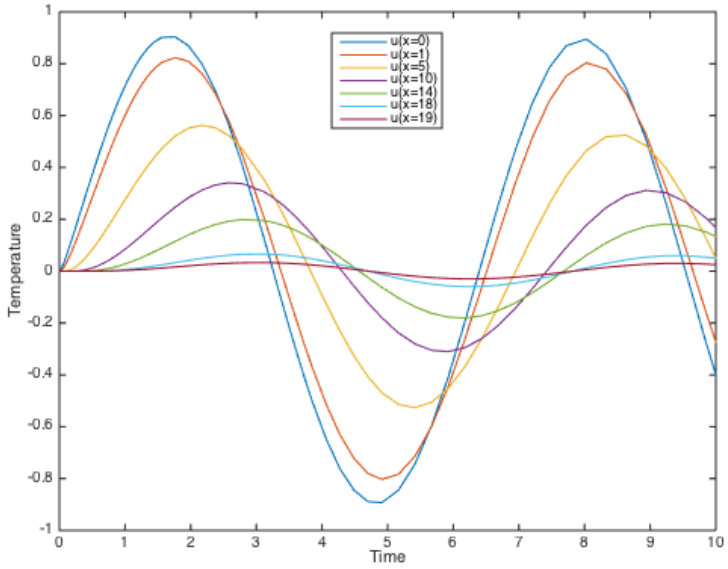


Figure B.1: Numerical solution for some of the spatial state $u(k)$ in the defined Heat Equation PDE

On Figure B.2 we can see an evolution on time and space in a 3D plot of the temperature. From it, we can appreciate the boundary conditions at $u(t, x = 0) = \sin(t), u(t, x = 20) = 0$ and the initial condition of $u(t = 0, x) = 0$.

B.2 Delay Differential Equations (DDE)

A Delay Differential Equation (DDE) occurs if the rate of change of the current state depends on a state in the past. One of the simplest form of DDE is the following, with one single delay $d > 0$:

$$\dot{x}(t) = f(x(t - d)) \tag{B.6}$$

where $x \in \mathbb{R}^n$ represents the state vector of the system.

The main problem with DDE is the simulation, because in order to simulate the system, we need to know the state $x(t)$ on a complete continuous interval $t \in [0, d]$. Thus, as we had for PDE, now we have again infinitely many initial conditions, since at the very beginning it is necessary to know a continuous path of the state x, from $x(0)$ until $x(d)$.

In order to solve this problem we have to do the same as we did for PDE, namely discretize the problem. In particular, we can model the time delay as a "pipe flow", where the pipe flow represents a flow of the state in time. To do that, we can add an extra variable $y \in [0, 1]$ representing the position inside the "pipe", and another variable u representing the flow in the pipe: we can think of u as the continuous past memory of x flowing backwards through the "time pipe", so that at the entrance of the pipe we will have the state x at the current time t, and in the output of the pipe the state x will be state at some point in past $(t - d)$ where d is the duration of travel through the pipe. Considering that, we can set the PDE of this pipe flow representing the continuous memory as:

$$\frac{\partial u}{\partial t} = -\frac{1}{d}\frac{\partial u}{\partial y}$$

with $u(0, t) = x(t)$ representing the input into pipe and $x(t - d) = u(1, t)$ representing the output of the pipe. Then, this PDE can be solved as we did in the previous section: we can spatially discretise the pipe. With such a discretisation, when a special scheme called "upwind-discretization" is usde, the delay ends up being modelled by a sequence of first oder delays ("PT1"), and we can refer to each one of these discretised states u_k representing the past memory as "helper states".

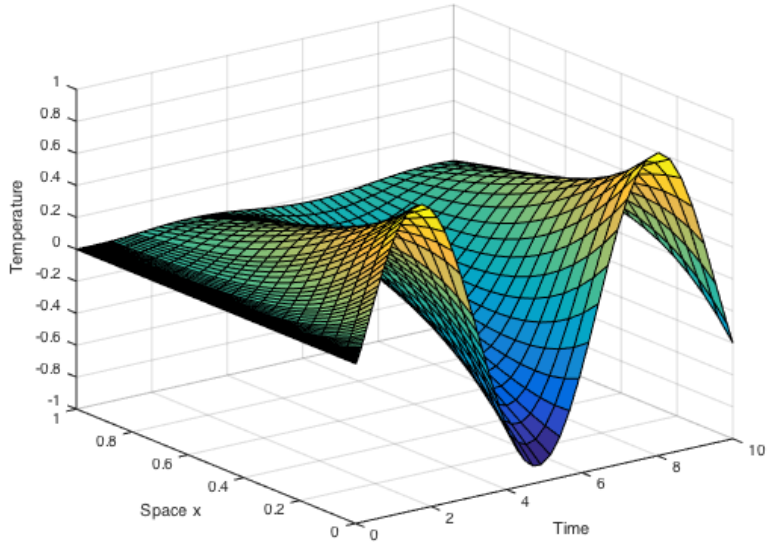


Figure B.2: Numerical solution in 3D for the defined Heat Equation PDE

B.2.1 Example of DDE

We show a simple example with $x \in \mathbb{R}$, where the scalar DDE is:

$$\dot{x}(t) = -x(t-d) \tag{B.7}$$

To solve the problem we would have to introduce N "helper states" $[u_1, \ldots, u_N]$, which once we spatially discretise the PDE modelling the pipe flow, we would end up with:

$$\dot{u}_k = -\frac{N}{d}(u_k - u_{k-1}), \quad \forall k \in [1, \ldots, N]: \tag{B.8}$$

with the boundary conditions on the PDE $u_0(t) = x(t)$, and where $\frac{d}{N}$ represents Δu. As we have explained already before, the last helper state would approximate the desired delayed value, i.e.:

$$x(t-d) \approx u_N \tag{B.9}$$

What it is important to keep in mind is that in practice, often $N = 2$ to 5 approximate a real delay with sufficient accuracy.

We can set as we did before the example in Matlab, and the represent the results in a figure.

1. The first thing to do is to set the discretised ODE. To understand the algorithm keep in mind that $udot$ represent the discretise partial derivates of the helper states, and the idea is since for the DDE properties $x(t) = u_0(t)$ and $x(t-d) \approx u_N(t)$, and since the DDE is $\dot{x}(t) = -x(t-d)$, it is clear that $\dot{x}(t) = \dot{u_0}(t) \approx u_N(t)$, and that is represented by the last line $udot(1) = -u(N)$.

```
function [ udot] = mydde(t, u)
d=1; N=20; udot=zeros(N,1);
for k=2:N
udot(k)=-N/d*(u(k)-u(k-1));
end
udot(1)= - u(N);
end
```

2. Then, we specify the initial conditions: `u0=zeros(20,0); u0(1)=1`

   where we basically have that $x(t=0) = u_0(t=0) = 1$, and then we see the evolution of the helper state $u_0(t) = x(t)$.

3. Finally we can simulate each one of the spacial discretised states on a time interval calling the ODE solver:

```
[tout,uout]=ode15s(@mypde, [0 10], u0)
figure(1); plot(tout,yout);
figure(2); surf(tout,linspace(0,1,20),uout')
```

As before we can check the simulation results. On Figure B.3 we can observe the evolution of the first helper state $u_0(t)$, which represents the main state $x(t)$, when the pipe is modelled as a discretisation of 3 helper states (3 PT1). We would expect a delay at $t = 1$ since d was also set as 1. However, we can see that the delay is not really perfect, nevertheless it delays the state almost up to $t = 1$. We can compare this result with the same value for 10 helper states on Figure B.4, where we can see how adding more helper states, i.e. increasing the number of discretised small delays, improves the approximation of the delay function.

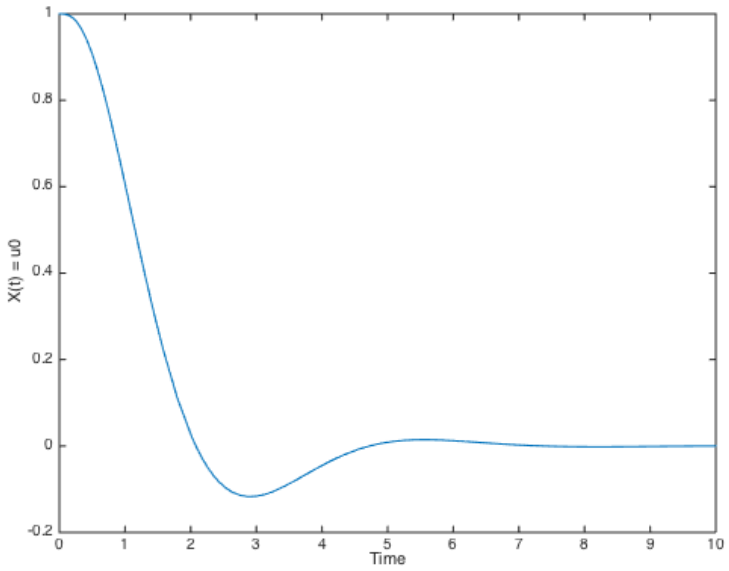


Figure B.3: Numerical solution of $x(t) = u_0(t)$ DDE using 3 helper states

On Figure B.5 we can see the evolution on time of the three helper states, $x(t) = u_0(t)$, $x(t - d/2) = u_1(t)$ and $x(t - d) = u_2(t)$. We can compare again this result with the approximation for 10 helper states on Figure B.6, where once again we can see how the transition between the states is smoother (smaller first order delays, but more amount of them leads to smother transitions between states, and thus to a better delay approximation).

Finally, on Figure B.7 we can see an evolution on time and space in a 3D plot of the 3 helper states. From it, we can appreciate the boundary conditions $u_0(t = 0) = x(t = 0) = 1$ and $u_1(t = 0) = u_2(t = 0) = 0$.

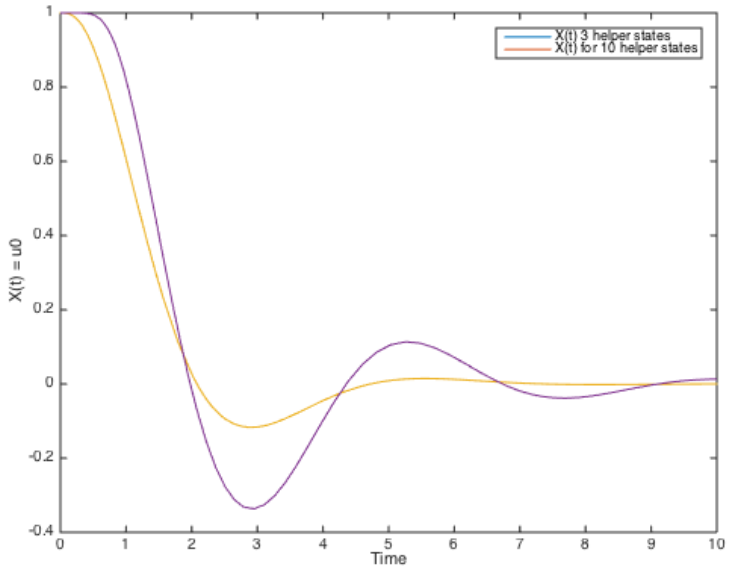


Figure B.4: Numerical solution of $x(t) = u_0(t)$ DDE using 10 helper states

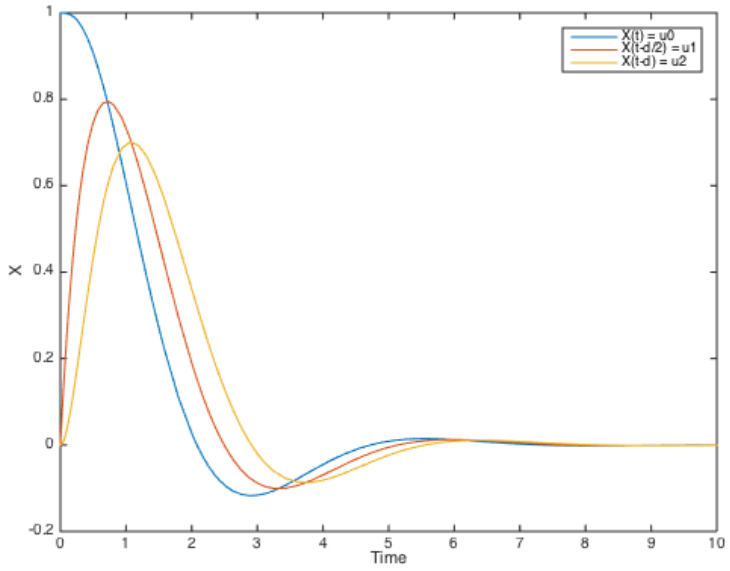


Figure B.5: Numerical solution of $x(t) = u_0(t)$, $x(t - d/2) = u_1(t)$ and $x(t - d) = u_2(t)$

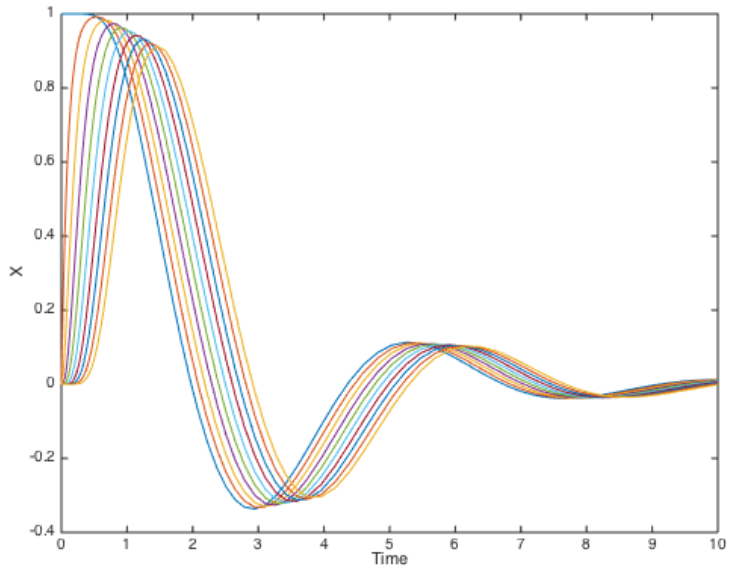


Figure B.6: Numerical solution of of all the helper states using 10 helper states

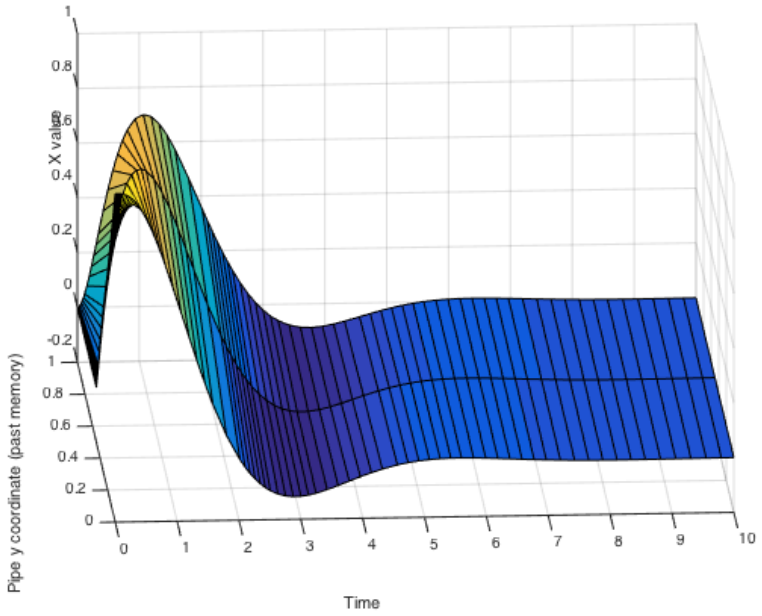


Figure B.7: Numerical solution of the delay equation with 3 helper states in 3D

Appendix C

Dynamic Systems

Model class	**IIR**	**FIR**	**AR**	**ARMA**	**ARMAX**	**NARMA[1]**	**NARMAX[2]**	**OE**	**Box-Jenkins**
linear	X	X	X	X	X			X	X
non-linear						X	X		
time-invariant	X[3]	X[3]	X[3]						
time-variant	X	X	X	X	X	X	X	X	X
deterministic	X	X	X						
stochastic	X	X	X	X	X	X	X	X	X
dependent on outputs	X		X	X	X	X	X	X	
dependent on inputs	X	X			X		X	X	X
linear-in-the-parameters (LIP)	X	X	X	X	X			(X)	X

Table C.1: Model Classes and their Characteristics

[1] Non-linear ARMA
[2] Non-linear ARMAX
[3] Model class is time-invariant if the model is deterministic.

Bibliography

[Bec14] Amir Beck. *Introduction to Nonlinear Optimization: Theory, Algorithms and Applications with MATLAB*. MOS-SIAM, 2014.

[BV04] S. Boyd and L. Vandenberghe. *Convex Optimization*. University Press, Cambridge, 2004.

[JU04] S.J. Julier and J.K. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401– 422, 2004.

[Lju99] L. Ljung. *System identification: Theory for the User*. Prentice Hall, Upper Saddle River, N.J., 1999.

[NW06] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, 2 edition, 2006.

[RM09] J. B. Rawlings and D. Q. Mayne. *Model Predictive Control: Theory and Design*. Nob Hill, 2009.

[RP12] J. Schoukens R. Pintelon. *System Identification - A Frequency Domain Approach*. Wiley, first edition, 2012.

[Sch13] J. Schoukens. System identification, April 2013. Lecture Manuscript.