Lecture Notes

Basics of Sensor Fusion

Roland Hostettler

Department of Electrical Engineering and Automation
Aalto University, Finland

November 27, 2018

Contents

Preface

These lecture notes are aimed for an entry-level M.Sc. course in sensor fusion. The goal is to provide a gentle and pragmatic introduction to the topic without the need for buying an expensive textbook. The reader should be familiar with:

1. linear algebra and calculus,
2. ordinary differential equations, and
3. basic statistics (manipulating expected values and understanding probability density functions).

The notes are based on the much more comprehensive and rigorous works on estimation theory and sensor fusion by Kay (1993); Gustafsson (2012), and Särkkä (2013) and the interested student is highly recommended to have a read in these books.

The topics covered in these notes range from simple linear observation models and least squares estimation, to modeling of dynamic systems, and inference in dynamic systems using bootstrap particle filtering. To lower the threshold to the topic, we sometimes trade mathematical rigor for easier understanding. This is in particular true for the probabilistic interpretations, especially in the beginning of the manuscript.

Chapter 1

Introduction

1.1 Background and Overview

Sensor fusion has applications in many different areas of daily life and plays an important role in modern society. For example, it is used to interpret traffic scenes in autonomous cars, for navigation and localization in robotics, or for the analysis of biosignals in biomedical engineering.

The common denominator and main objective of sensor fusion systems are that they take measurements from different sensors and *estimate* or *infer* one or more quantities of interest (which may be a latent, i.e., an indirectly observed quantity). On a high level, this involves three main components (Figure 1.1):

- One or more *sensor(s)* that measure an observable quantity,
- *model(s)* that relate the observed quantity to the quantity of interest, and
- an *estimation algorithm* that estimates the quantity of interest by combining the model and the measured data.

The quantity of interest is either a set of static *parameters*, the dynamic (i.e., time-varying) *state* of a system, or both. In this course, we will first concern ourselves with the static case in Chapters 2–3 and the dynamic case will be discussed in Chapters 4–5. The third case, simultaneous estimation of both static parameters and time-varying states are out of the scope of this course.

For the remainder, we will denote a scalar static parameter as θ and a vector of K parameters $\theta_1, \theta_2, \dots, \theta_K$ as $\boldsymbol{\theta} = \begin{bmatrix}\theta_1 & \theta_2 & \dots & \theta_K\end{bmatrix}^\mathsf{T}$. The symbol x_n will be used to denote a scalar, time-varying state of a dynamic system at time t_n, and $\boldsymbol{x}_n$ will be used for the vector state at t_n, that is, for the time-varying state of a higher order dynamic system.

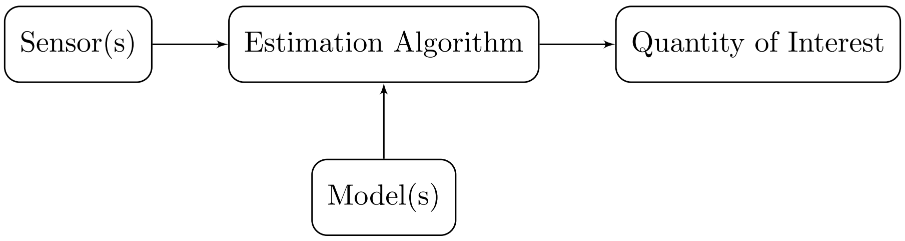


Figure 1.1: The basic components of a sensor fusion system.

Table 1.1: Examples of common sensors

Sensor	Measurement	Application Examples
Accelerometer	Gravity, acceleration	Inertial navigation, activity tracking, screen rotation
Gyroscope	Rotational velocity	Inertial navigation, activity tracking
Magnetometer	Magnetic field strength	Inertial navigation, digital compass, object tracking
Radar	Range, bearing, speed	Target tracking, autonomous vehicles
LIDAR	Range, bearing, speed	Target tracking, autonomous vehicles, robotics
Ultrasound	Range	Robotics
Camera	Visual scene	Security systems, autonomous vehicles, robotics
Strain gauge	Strain	Condition monitoring, scales

1.2 Sensors

A sensor is a device that provides a measurement related to the quantity of interest. The measurement may be direct or indirect: In direct measurements, the quantity observed by the sensor is the quantity of interest, for example the temperature in a thermometer. Sensors that measure indirectly provide a measurement of a quantity that is only related to the quantity of interest, for example an accelerometer which measures a body's acceleration when we might be interested in the position instead.

There are many different types of sensors for measuring a wide range of phenomena available today. These include electronic sensors, (micro-)mechanical sensors, or virtual sensors and a (non-exhaustive) overview of a few commonly used sensors, their measurements, and some of their application areas is shown in Table 1.1.

Sensors are characterized by many different properties that affect their performance. These include limitations such as measurement range, requirements and sensitivity for the environmental conditions (temperature, humidity, etc.), sampling frequency, as well as measurement noise, biases, and drifts. Some of these factors have to be taken into account when designing the hardware whereas others can be accurately dealt with when designing the estimation algorithm. In this course, we will focus on the latter aspects of the sensors, that is, we will introduce tools that are appropriate for taking into account the properties that can be handled by sensor fusion.

The measurement obtained by a sensor will be denoted using the symbol y_n or $\boldsymbol{y}_n$. The subscript n denotes the nth measurement, which may refer to an arbitrary measurement index, a time series, a sensor identification number in a sensor network, etc. The regular notation (y_n) is used to denote scalar measurements (e.g., from a temperature sensor) whereas the bold symbol ($\boldsymbol{y}_n$) denotes a vector observation (e.g., as measured by a tri-axial accelerometer).

1.3 Models

A model is a mathematical formulation that relates the quantity of interest to the measurements in a systematic way. Additionally, in the dynamic case, models are also used to describe how the quantity of interest evolves over time.

As mentioned earlier, sensors suffer from several drawbacks. Thus, an important aspect of every model is that it can take uncertainties such as sensor noise or *measurement noise* into account. Unfortunately, measurement noise is difficult to quantify analytically. Instead, a suitable approach is to use statistical modeling: Normally, the measurement noise exhibits statistical properties that can be quantified. Hence, while the actual value of the noise can not be observed or measured, its statistical properties can. A common assumption is that it is zero mean, that is, on average, it is zero (but each individual realization is not).

1.3.1 Basic Model

Next, we introduce a very simplistic model, which turns out to be quite generic, despite its simplicity. First, assume that a scalar measurement y_n of the parameters $\boldsymbol{\theta}$ is available. The objective is then to relate the measurement y_n to the parameters $\boldsymbol{\theta}$ such that the uncertainty is taken into account. We know that the measurement must be some function of the parameters plus measurement noise. Hence, we can write the model as

$$y_n = g_n(\boldsymbol{\theta}) + r_n \tag{1.1}$$

Equation (1.1) is called a *sensor model*, *measurement model*, or *observation model.* The generic anatomy of a measurement model is that

- the measured quantity y_n is found on the left hand side of the equation[1], and
- on the right hand side there are two terms, a function $g_n(\boldsymbol{\theta})$ that relates the parameters of interest $\boldsymbol{\theta}$ to the measurement y_n and a noise term r_n.

[1]There are more general formulations than (1.1) that do not share this trait, but we will not make use of these in this course.

As mentioned above, the measurement noise r_n is assumed to be a random variable. As such, r_n follows some kind of probability density function (pdf), that is, we have that

$$r_n \sim p(r_n), \tag{1.2}$$

which reads as "r_n is distributed according to $p(r_n)$" where $p(r_n)$ denotes the corresponding pdf. At this point, we will not concern ourselves with any particular form of $p(r_n)$ but only assume that the r_ns are zero-mean, independent random variables with variance $\sigma^2_{r,n}$. In other words, we have that

$$\mathrm{E}\{r_n\} = 0, \tag{1.3a}$$

$$\mathrm{var}\{r_n\} = \mathrm{E}\{r_n^2\} - (\mathrm{E}\{r_n\})^2 = \sigma^2_{r,n}, \tag{1.3b}$$

$$\mathrm{Cov}\{r_m, r_n\} = \mathrm{E}\{r_m r_n\} - \mathrm{E}\{r_m\}\,\mathrm{E}\{r_n\} = 0 \quad (m \neq n), \tag{1.3c}$$

where $\mathrm{E}\{r_n\}$ and $\mathrm{var}\{r_n\}$ denote the expected value and variance of r_n, respectively. Note that the measurement noise variance may vary for the different measurements, as indicated by the subscript n on $\sigma^2_{r,n}$ (e.g., if the different measurements are obtained by different sensors).

1.3.2 Vector Model

The basic model only considers scalar measurements. However, many sensors actually provide vector-valued measurements, for example accelerometers that provide full 3D-acceleration measurements at each sampling instant. The basic model (1.1) can easily be extended to account for such vector-valued measurements. In this case, the vector measurement model can be written as

$$\boldsymbol{y}_n = g_n(\boldsymbol{\theta}) + \boldsymbol{r}_n, \tag{1.4}$$

where $\boldsymbol{y}_n$ is an d_y-dimensional column vector. In this case, the measurement noise $\boldsymbol{r}_n$ becomes a multivariate random variable with pdf

$$\boldsymbol{r}_n \sim p(\boldsymbol{r}_n). \tag{1.5}$$

As for the scalar case, $p(\boldsymbol{r}_n)$ can have any arbitrary form. For simplicity, we again assume that the $\boldsymbol{r}_n$s are zero-mean, independent random variables with covariance matrix $\boldsymbol{R}_n$, that is,

$$\mathrm{E}\{\boldsymbol{r}_n\} = 0, \tag{1.6a}$$

$$\mathrm{Cov}\{\boldsymbol{r}_n\} = \mathrm{E}\{\boldsymbol{r}_n\boldsymbol{r}_n^\mathsf{T}\} - \mathrm{E}\{\boldsymbol{r}_n\}\,\mathrm{E}\{\boldsymbol{r}_n\}^\mathsf{T} = \boldsymbol{R}_n, \tag{1.6b}$$

$$\mathrm{Cov}\{\boldsymbol{r}_m, \boldsymbol{r}_n\} = \mathrm{E}\{\boldsymbol{r}_m\boldsymbol{r}_n^\mathsf{T}\} - \mathrm{E}\{\boldsymbol{r}_m\}\,\mathrm{E}\{\boldsymbol{r}_n\}^\mathsf{T} = 0 \quad (m \neq n). \tag{1.6c}$$

As it can be seen, the scalar model in (1.1) is just a special case of the vector model in (1.4), where the measurement and noise are scalars (i.e., $d_y = 1$). Hence, it is often more general to work with vector model, but sometimes more instructive to work with the scalar model.

1.3.3 Multiple Measurements and Measurement Stacking

In order to be able to *fuse* the measurements of one or more sensors, we need multiple measurements. These may either be from one sensor at different time instants (repeated measurements), from different sensors, or both. If we have obtained a total of N measurements $y_1, y_2, \dots, y_N$, we will refer to the set of all measurements using the notation $y_{1:N} = \{y_1, y_2, \dots, y_N\}$ for the scalar case and $\boldsymbol{y}_{1:N} = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \dots, \boldsymbol{y}_N\}$ for the vector case.

Sometimes, it is also helpful to write the complete set of measurements together with the measurement model more compactly. This can be achieved by stacking the measurements into a single *measurement vector*, which yields a much more compact model of the form

$$\boldsymbol{y} = g(\boldsymbol{\theta}) + \boldsymbol{r}. \tag{1.7}$$

For scalar measurements of the form (1.1), we have that

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}, \; g(\boldsymbol{\theta}) = \begin{bmatrix} g_1(\boldsymbol{\theta}) \\ g_2(\boldsymbol{\theta}) \\ \vdots \\ g_N(\boldsymbol{\theta}) \end{bmatrix}, \text{ and } \boldsymbol{r} = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_N \end{bmatrix}. \tag{1.8}$$

Furthermore, the overall noise covariance $\boldsymbol{R}$ for this model is a diagonal matrix of the form

$$\boldsymbol{R} = \operatorname{Cov}\{\boldsymbol{r}\} = \begin{bmatrix} \sigma_{r,1}^2 & 0 & \dots & 0 \\ 0 & \sigma_{r,2}^2 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \dots & 0 & \sigma_{r,N}^2 \end{bmatrix}. \tag{1.9}$$

For vector measurements, $\boldsymbol{y}$, $g(\boldsymbol{\theta})$, and $\boldsymbol{r}$ are constructed in the same way as for the scalar case in (1.8), that is,

$$\boldsymbol{y} = \begin{bmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \\ \vdots \\ \boldsymbol{y}_N \end{bmatrix}, \; g(\theta) = \begin{bmatrix} g_1(\boldsymbol{\theta}) \\ g_2(\boldsymbol{\theta}) \\ \vdots \\ g_N(\boldsymbol{\theta}) \end{bmatrix}, \text{ and } \boldsymbol{r} = \begin{bmatrix} \boldsymbol{r}_1 \\ \boldsymbol{r}_2 \\ \vdots \\ \boldsymbol{r}_N \end{bmatrix}. \tag{1.10}$$

The measurement noise covariance is a block-diagonal matrix with the individual covariance matrices $\boldsymbol{R}_n$ on the diagonal, that is,

$$\boldsymbol{R} = \operatorname{Cov}\{\boldsymbol{r}\} = \begin{bmatrix} \boldsymbol{R}_1 & 0 & \dots & 0 \\ 0 & \boldsymbol{R}_2 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \dots & 0 & \boldsymbol{R}_N \end{bmatrix}. \tag{1.11}$$

1.3.4 Gaussian Measurement Noise

A common assumption is that the measurement noise is zero-mean Gaussian noise. In this case, the pdf of the noise is the (multivariate) Gaussian distribution given by

$$p(\boldsymbol{r}) = \frac{1}{(2\pi)^{d_y N/2}|\boldsymbol{R}|^{1/2}} \exp\left(-\frac{1}{2}\boldsymbol{r}^{\mathsf{T}}\boldsymbol{R}^{-1}\boldsymbol{r}\right), \tag{1.12}$$

where $|\boldsymbol{R}|$ denotes the matrix determinant. Equation (1.12) may also be written more compactly as

$$p(\boldsymbol{r}) = \mathcal{N}(\boldsymbol{r}; 0, \boldsymbol{R}), \tag{1.13}$$

where $\mathcal{N}(\boldsymbol{z}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes the multivariate pdf of an M-dimensional Gaussian random variable $\boldsymbol{z}$ with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ defined as

$$\mathcal{N}(\boldsymbol{z}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{M/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{z}-\boldsymbol{\mu})^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{z}-\boldsymbol{\mu})\right). \tag{1.14}$$

Assuming the measurement noise to be Gaussian is indeed reasonable in many (but not all) applications. Furthermore, it also has some important implications on the algorithms that we will derive, in particular with respect to the estimators statistical properties. However, we will not focus on these aspects throughout most of this course.

1.4 Estimation Algorithm

The *estimation algorithm* or *inference algorithm* combines all the available measurements by using the measurement models to estimate the parameters (or states) in a way that is optimal with respect to some criterion. By combining the information from multiple measurements and sensors, we can exploit the diversity of the measurements to obtain a better parameter *estimate*. Hence, sensor fusion algorithms can essentially be seen as an application of estimation theory, which provides a statistically sound and flexible framework.

1.4.1 Cost Functions

In this course (with the exception of the particle filtering method introduced in Section 5.5), we sacrifice the generality and rigor of estimation theory for simplicity and only consider algorithms that minimize a *cost function* $J(\boldsymbol{\theta})$. These types of algorithms are a good starting point for many sensor fusion applications and have historically proved useful. They also have sound statistical interpretations. However, we will not discuss these explicitly, but for the interested student, there are a couple of exercises that delve into this topic.

Mathematically, minimizing a cost function $J(\boldsymbol{\theta})$ to find a parameter estimate $\hat{\boldsymbol{\theta}}$ (or state estimate $\hat{\boldsymbol{x}}_n$; the hat indicates that $\hat{\boldsymbol{\theta}}$ is an estimate of the parameters $\boldsymbol{\theta}$), can be written as the optimization problem

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, J(\boldsymbol{\theta}), \tag{1.15}$$

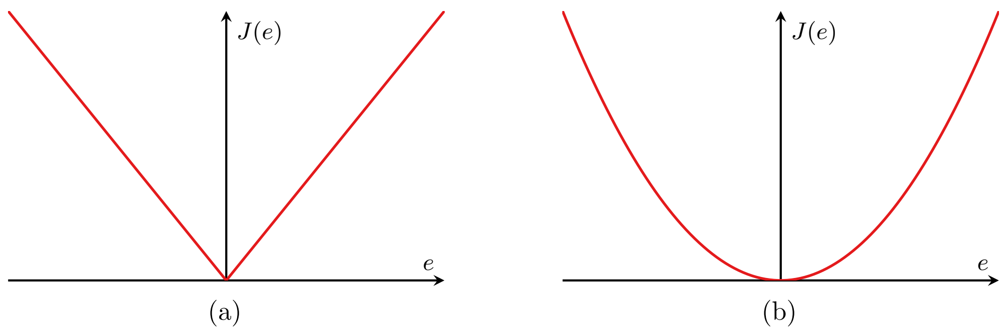


Figure 1.2: Examples of cost functions: (a) Absolute error, and (b) quadratic error.

which reads as "the estimate $\hat{\boldsymbol{\theta}}$ is the argument $\boldsymbol{\theta}$ that minimizes $J(\boldsymbol{y}, \boldsymbol{\theta})$". Typical cost functions minimize a function of the error

$$e_n = y_n - g_n(\boldsymbol{\theta}). \tag{1.16}$$

between the measurement and the output predicted by the estimated parameters. Two typical functions are the absolute error (Figure 1.2a)

$$|e_n| = |y_n - g_n(\boldsymbol{\theta})|, \tag{1.17}$$

which penalizes all errors equally, or the quadratic cost function (Figure 1.2b)

$$e_n^2 = (y_n - g_n(\boldsymbol{\theta}))^2. \tag{1.18}$$

1.4.2 Least Squares

In practice, the quadratic cost is much more common. It implicitly imposes a certain smoothness onto the problem by penalizing large errors more than small ones. Additionally, it is closely related to the case where the measurement noise is assumed to be Gaussian as discussed in Section 1.3. Minimizing the quadratic cost function is also referred to as the *least squares* method. As we will see, many of the most common estimation algorithms can be formulated based on this approach.

Naturally, the aim of the estimation algorithm must be to minimize the error over all measurements $y_{1:N}$ rather than only one measurement y_n. Hence, the resulting cost function is rather the sum over all the errors e_n. For the quadratic cost (1.18), the cost function is thus

$$\begin{aligned} J_{\text{LS}}(\boldsymbol{\theta}) &= \sum_{n=1}^{N} e_n^2 \\ &= \sum_{n=1}^{N} (y_n - g_n(\boldsymbol{\theta}))^2. \end{aligned} \tag{1.19}$$

Similarly, for vector measurements, the quadratic error can be written as

$$e_n^2 = (\boldsymbol{y}_n - g_n(\boldsymbol{\theta}))^\mathsf{T}(\boldsymbol{y}_n - g_n(\boldsymbol{\theta})), \tag{1.20}$$

and thus, the cost function becomes

$$J_{\text{LS}}(\boldsymbol{\theta}) = \sum_{n=1}^{M}(\boldsymbol{y}_n - g_n(\boldsymbol{\theta}))^\mathsf{T}(\boldsymbol{y}_n - g_n). \tag{1.21}$$

Finally, we can write (1.19) and (1.21) more compactly by using the vector notation introduced in Section 1.3. This yields the compact cost function

$$J_{\text{LS}}(\boldsymbol{\theta}) = (\boldsymbol{y} - g(\boldsymbol{\theta}))^\mathsf{T}(\boldsymbol{y} - g(\boldsymbol{\theta})). \tag{1.22}$$

The formulations (1.19) or (1.21) and (1.22) are equivalent. Hence, we will use these formulations interchangeably, depending on the context.

1.4.3 Weighted Least Squares

The least squares method assumes that all measurements are equally reliable. In other words, each measurement is weighed into the estimate equally. However, not every measurement might actually carry the same amount of information: Some measurements might be more important (in some sense) than others and thus, should contribute more to the solution than others.

This can be achieved by using the *weighted least squares* cost function instead. By introducing the positive weights w_n for each term in (1.19), the weighted least squares cost function for the scalar case becomes

$$J_{\text{WLS}}(\boldsymbol{\theta}) = \sum_{n=1}^{N} w_n(y_n - g_n(\boldsymbol{\theta}))^2. \tag{1.23}$$

Similarly, for the vector case (1.21), we can introduce a positive definite weighing matrix[2] $\boldsymbol{W}_n$ and weigh each term using this matrix. The resulting cost function becomes

$$J_{\text{WLS}}(\boldsymbol{\theta}) = \sum_{n=1}^{N}(\boldsymbol{y}_n - g_n(\boldsymbol{\theta}))^\mathsf{T}\boldsymbol{W}_n(\boldsymbol{y}_n - g_n(\boldsymbol{\theta})). \tag{1.24}$$

Again, (1.23)–(1.24) can be written using the compact notation as

$$J_{\text{WLS}}(\boldsymbol{\theta}) = (\boldsymbol{y} - g(\boldsymbol{\theta}))^\mathsf{T}\boldsymbol{W}(\boldsymbol{y} - g(\boldsymbol{\theta})), \tag{1.25}$$

where the overall weighing matrix is given by either

$$\boldsymbol{W} = \begin{bmatrix} w_1 & 0 & \dots & 0 \\ 0 & w_2 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \dots & 0 & w_N \end{bmatrix} \quad \text{and } \boldsymbol{W} = \begin{bmatrix} \boldsymbol{W}_1 & 0 & \dots & 0 \\ 0 & \boldsymbol{W}_2 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \dots & 0 & \boldsymbol{W}_N \end{bmatrix}$$

[2]That is, a matrix $\boldsymbol{M}$ for which it holds that $\boldsymbol{x}^\mathsf{T}\boldsymbol{M}\boldsymbol{x} > 0$ for any arbitrary non-zero vectors $\boldsymbol{x}$.

for the scalar and vector case, respectively. The forms (1.24)–(1.25) are indeed very general and they will be the basis for many of the estimation algorithms derived in this course.

An important question is the choice of the weights w_n (or $\boldsymbol{W}_n$). This may in principle be arbitrary as long as they are positive (or positive definite). In practice, however, a principled choice is to use

$$w_n = 1/\sigma_{r,n}^2 \text{ and } \boldsymbol{W}_n = \boldsymbol{R}_n^{-1},$$

that is, use the inverse (co)variance of the measurement noise as the weighting factor. The rationale behind this choice is as follows: The covariance is a measure for the amount of uncertainty in our measurement due to the measurement noise. A large covariance means that there is a large uncertainty and vice-versa. Hence, by weighing the measurements using the inverse of the covariance, measurements with high uncertainty are given lower weights and measurements with low uncertainty are given higher weights.

Chapter 2

Static Linear Models

In this chapter, we focus on static, linear models. These models are quite versatile already and have several important properties, one being that we can find a closed form estimation algorithm.

2.1 Linear Model

2.1.1 Scalar Model

The simplest measurement model arises when the measurement is a scaled and noisy version of a scalar parameter of interest. In this case, the scalar measurement y_n can be written as

$$y_n = c\theta + r_n, \tag{2.1}$$

where we assume that the scale factor c is known.

With (2.1) in mind, we can construct similar models where the scalar measurement linearly depends on a set of K unknown parameters $\theta_1, \dots, \theta_K$:

$$y_n = c_1\theta_1 + c_2\theta_2 + \cdots + c_K\theta_K + r_n, \tag{2.2}$$

with known scale factors $c_1, \dots, c_K$ and noise term r_n. We can rewrite (2.2) in matrix form as

$$y_n = \begin{bmatrix} c_1 & c_2 & \dots & c_K \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_K \end{bmatrix} + r_n = \boldsymbol{c}\boldsymbol{\theta} + r_n, \tag{2.3}$$

where $\boldsymbol{c} = \begin{bmatrix} c_1 & c_2 & \dots & c_K \end{bmatrix}$ is the vector of constants and $\boldsymbol{\theta} = \begin{bmatrix} \theta_1 & \theta_2 & \dots & \theta_K \end{bmatrix}^{\mathsf{T}}$ is the parameter vector.

For a total of N measurements $y_1, y_2, \dots, y_N$, we can use measurement stacking as discussed in Section 1.3 to obtain the compact model

$$\boldsymbol{y} = \boldsymbol{G}\boldsymbol{\theta} + \boldsymbol{r}, \tag{2.4}$$

where $\boldsymbol{y}$ and $\boldsymbol{r}$ are the stacked measurements and measurement noise, respectively. Moreover,

$$\boldsymbol{G} = \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{c} \\ \vdots \\ \boldsymbol{c} \end{bmatrix} \tag{2.5}$$

is a matrix that contains the known factors $c_1, \dots, c_K$.

2.1.2 Vector Model

The scalar model can also be extended to the case where each measurement itself is a vector. In this case, a single vector-valued measurement $\boldsymbol{y}_n$ can be written as

$$\begin{aligned} \boldsymbol{y}_n &= \begin{bmatrix} c_{11} & c_{12} & \dots & c_{1K} \\ c_{21} & c_{22} & \dots & c_{1K} \\ \vdots & \vdots & \ddots & \vdots \\ c_{d_y1} & c_{d_y2} & \dots & c_{d_yK} \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_K \end{bmatrix} + \boldsymbol{r}_n \\ &= \boldsymbol{C}\boldsymbol{\theta} + \boldsymbol{r}_n, \end{aligned} \tag{2.6}$$

where $\boldsymbol{C}$ is a known matrix.

Again, by stacking the measurements $\boldsymbol{y}_1, \boldsymbol{y}_2, \dots, \boldsymbol{y}_N$ into a single vector $\boldsymbol{y}$, we obtain the compact model

$$\boldsymbol{y} = \boldsymbol{G}\boldsymbol{\theta} + \boldsymbol{r}, \tag{2.7}$$

where the matrix $\boldsymbol{G}$ is given by

$$\boldsymbol{G} = \begin{bmatrix} \boldsymbol{C} \\ \boldsymbol{C} \\ \vdots \\ \boldsymbol{C} \end{bmatrix}. \tag{2.8}$$

Comparing (2.4) and (2.7), it is obvious that both the scalar and the vector model can be written in the same form, the only difference being the matrix $\boldsymbol{G}$. Furthermore, that common model is completely linear in all the parametes $\boldsymbol{\theta}$ and thus, this model is called the *linear model*.

2.2 Linear Least Squares

The linear model introduced in the previous section in (2.7) can be seen to be a linear equation system. Unfortunately, in addition to the K unknown parameters of interest in θ, there are also a Nd_y unknown noise values, which yields a total of $K + Nd_y$ unknowns. Since we only have Nd_y measurements ($\approx$ equations) the equation system is under-determined and can not be solved. Even adding more measurements does not solve this problem since every new measurement adds a new unknown (the noise values).

Fortunately, we have already introduced an alternative approach: Minimizing the *error cost* as discussed in Section 1.4. In this section, we will pursue the least squares approach for the linear model introduced in the previous section.

2.2.1 Scalar Model

We start our discussion based on the scalar model (2.1). For this model, the error for each measurement is given by

$$e_n = y_n - c\theta. \tag{2.9}$$

Hence, the least squares cost function becomes

$$J_{\mathrm{LS}}(\theta) = \sum_{n=1}^{N} (y_n - c\theta)^2. \tag{2.10}$$

The least squares estimate $\hat{\theta}_{\mathrm{LS}}$ is the value of θ that minimizes the cost function (2.10). The minima of the cost function with respect to θ are found by setting the derivative of $J_{\mathrm{LS}}(\theta)$ (with respect to θ) to zero and solving it for θ.

The derivative is given by

$$\begin{aligned}
\frac{\partial J_{\mathrm{LS}}(\theta)}{\partial \theta} &= \frac{\partial}{\partial \theta} \sum_{n=1}^{N} (y_n - c\theta)^2 \\
&= \sum_{n=1} -2c(y_n - c\theta) \\
&= -\sum_{n=1} 2cy_n + \sum_{n=1}^{N} 2c^2\theta \\
&= -2c \sum_{n=1} y_n + 2Nc^2\theta.
\end{aligned} \tag{2.11}$$

Setting (2.11) to zero yields

$$0 = -2c \sum_{n=1} y_n + 2Nc^2\theta,$$

and solving for θ finally yields the estimator

$$\hat{\theta}_{\mathrm{LS}} = \frac{1}{Nc} \sum_{n=1}^{N} y_n. \tag{2.12}$$

Two important aspects of an estimator are its mean and (co)variance. The mean indicates whether an estimator (on average) converges to the true parameter value θ and the covariance indicates how confident we are about the estimated value. For the scalar least squares estimator in (2.12), the mean is given by

$$\begin{aligned}
\mathrm{E}\{\hat{\theta}\} &= \mathrm{E}\left\{\frac{1}{Nc}\sum_{n=1}^{N} y_n\right\} \\
&= \frac{1}{Nc}\sum_{n=1}^{N} \mathrm{E}\{c\theta + r_n\} \\
&= \frac{1}{Nc}\sum_{n=1}^{N} c\theta + \mathrm{E}\{r_n\} \\
&= \frac{1}{Nc} Nc\theta + \sum_{n=1}^{N} \mathrm{E}\{r_n\} \\
&= \theta + \sum_{n=1}^{N} \mathrm{E}\{r_n\}.
\end{aligned}$$

Hence, if and only if the measurement noise is zero-mean, that is, $\mathrm{E}\{r_n\} = 0$, the mean reduces to

$$\mathrm{E}\{\hat{\theta}\} = \theta. \tag{2.13}$$

In this case, $\hat{\theta}$ converges to the true parameter value and the estimator is said to be an *unbiased estimator*. Furthermore, for the zero-mean noise case, the variance is given by

$$\begin{aligned}
\mathrm{var}\{\hat{\theta}\} &= \mathrm{E}\{(\hat{\theta} - \mathrm{E}\{\hat{\theta}\})^2\} \\
&= \mathrm{E}\left\{\left(\frac{1}{Nc}\sum_{n=1}^{N} y_n - \theta\right)^2\right\} \\
&= \mathrm{E}\left\{\left(\frac{1}{Nc}\sum_{n=1}^{N} (c\theta + r_n) - \theta\right)^2\right\} \\
&= \mathrm{E}\left\{\left(\frac{1}{Nc}\left(Nc\theta + \sum_{n=1}^{N} r_n\right) - \theta\right)^2\right\} \\
&= \frac{1}{N^2c^2}\mathrm{E}\left\{\left(\sum_{n=1}^{N} r_n\right)^2\right\},
\end{aligned}$$

and finally

$$\mathrm{var}\{\hat{\theta}\} = \frac{1}{N^2c^2}\sum_{n=1}^{N} \sigma_{r,n}^2, \tag{2.14}$$

where we have made use of the fact that the individual r_ns are independent (i.e., $\mathrm{Cov}\{r_i, r_j\} = 0$).

2.2.2 General Linear Model

We can now generalize the result (2.12) for general linear models of the form (2.7). Using the vector form of the least squares criterion in (1.22) together with the model (2.7) yields the cost function

$$J_{\mathrm{LS}}(\boldsymbol{\theta}) = (\boldsymbol{y} - \boldsymbol{G}\boldsymbol{\theta})^{\mathsf{T}}(\boldsymbol{y} - \boldsymbol{G}\boldsymbol{\theta}). \tag{2.15}$$

To minimize this cost function, we can follow the same steps as in the scalar case. First, we calculate the gradient of (2.15) with respect to the parameters $\boldsymbol{\theta}$. Since $\boldsymbol{\theta}$ is a vector, this can be done by calculating the partial derivative with respect to each individual entry θ_k in $\boldsymbol{\theta}$ individually. A more compact alternative is to use vector calculus. The latter is more compact and yields:

$$\begin{aligned}
\frac{\partial J_{\mathrm{LS}}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} &= \frac{\partial}{\partial \boldsymbol{\theta}}(\boldsymbol{y} - \boldsymbol{G}\boldsymbol{\theta})^{\mathsf{T}}(\boldsymbol{y} - \boldsymbol{G}\boldsymbol{\theta}) \\
&= \frac{\partial}{\partial \boldsymbol{\theta}}(\boldsymbol{y}^{\mathsf{T}}\boldsymbol{y} - \boldsymbol{y}^{\mathsf{T}}\boldsymbol{G}\boldsymbol{\theta} - \boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{G}^{\mathsf{T}}\boldsymbol{y} + \boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{G}^{\mathsf{T}}\boldsymbol{G}\boldsymbol{\theta} \\
&= -2\boldsymbol{G}^{\mathsf{T}}\boldsymbol{y} + 2\boldsymbol{G}^{\mathsf{T}}\boldsymbol{G}\boldsymbol{\theta}
\end{aligned}$$

Then, setting the gradient to zero and solving for $\boldsymbol{\theta}$ yields

$$\hat{\boldsymbol{\theta}}_{\mathrm{LS}} = (\boldsymbol{G}^{\mathsf{T}}\boldsymbol{G})^{-1}\boldsymbol{G}^{\mathsf{T}}\boldsymbol{y} \tag{2.16}$$

for the least squares estimator. Note that (2.16) is valid for both scalar measurements of the form (2.2) as well as (2.6), with the appropriate choice of the matrix $\boldsymbol{G}$.

Similar to the scalar case, we can calculate the statistical properties of the estimator (2.16). The mean is found from

$$\begin{aligned}
\mathrm{E}\{\hat{\boldsymbol{\theta}}_{\mathrm{LS}}\} &= \mathrm{E}\{(\boldsymbol{G}^{\mathsf{T}}\boldsymbol{G})^{-1}\boldsymbol{G}^{\mathsf{T}}\boldsymbol{y}\} \\
&= \mathrm{E}\{(\boldsymbol{G}^{\mathsf{T}}\boldsymbol{G})^{-1}\boldsymbol{G}^{\mathsf{T}}(\boldsymbol{G}\boldsymbol{\theta} + \boldsymbol{r})\} \\
&= \mathrm{E}\{(\boldsymbol{G}^{\mathsf{T}}\boldsymbol{G})^{-1}\boldsymbol{G}^{\mathsf{T}}\boldsymbol{G}\boldsymbol{\theta} + (\boldsymbol{G}^{\mathsf{T}}\boldsymbol{G})^{-1}\boldsymbol{G}^{\mathsf{T}}\boldsymbol{r}\} \\
&= \boldsymbol{\theta} + (\boldsymbol{G}^{\mathsf{T}}\boldsymbol{G})^{-1}\boldsymbol{G}^{\mathsf{T}}\,\mathrm{E}\{\boldsymbol{r}\},
\end{aligned}$$

and we can see that if (and only if) $\mathrm{E}\{\boldsymbol{r}\} = 0$, we have that

$$\mathrm{E}\{\hat{\boldsymbol{\theta}}_{\mathrm{LS}}\} = \boldsymbol{\theta}, \tag{2.17}$$

and the estimator is unbiased. Furthermore, for the zero-mean noise case, the variance can be derived from

$$\begin{aligned}
\mathrm{Cov}\{\hat{\boldsymbol{\theta}}_{\mathrm{LS}}\} &= \mathrm{E}\left\{\left(\hat{\boldsymbol{\theta}}_{\mathrm{LS}} - \mathrm{E}\{\hat{\boldsymbol{\theta}}_{\mathrm{LS}}\}\right)\left(\hat{\boldsymbol{\theta}}_{\mathrm{LS}} - \mathrm{E}\{\hat{\boldsymbol{\theta}}_{\mathrm{LS}}\}\right)^{\mathsf{T}}\right\} \\
&= \mathrm{E}\left\{\left((\boldsymbol{G}^{\mathsf{T}}\boldsymbol{G})^{-1}\boldsymbol{G}^{\mathsf{T}}(\boldsymbol{G}\boldsymbol{\theta} + \boldsymbol{r}) - \boldsymbol{\theta}\right)\left((\boldsymbol{G}^{\mathsf{T}}\boldsymbol{G})^{-1}\boldsymbol{G}^{\mathsf{T}}(\boldsymbol{G}\boldsymbol{\theta} + \boldsymbol{r}) - \boldsymbol{\theta}\right)^{\mathsf{T}}\right\} \\
&= \mathrm{E}\left\{\left((\boldsymbol{G}^{\mathsf{T}}\boldsymbol{G})^{-1}\boldsymbol{G}^{\mathsf{T}}\boldsymbol{r}\right)\left((\boldsymbol{G}^{\mathsf{T}}\boldsymbol{G})^{-1}\boldsymbol{G}^{\mathsf{T}}\boldsymbol{r}\right)^{\mathsf{T}}\right\} \\
&= (\boldsymbol{G}^{\mathsf{T}}\boldsymbol{G})^{-1}\boldsymbol{G}^{\mathsf{T}}\,\mathrm{E}\{\boldsymbol{r}\boldsymbol{r}^{\mathsf{T}}\}\boldsymbol{G}((\boldsymbol{G}^{\mathsf{T}}\boldsymbol{G})^{-1})^{\mathsf{T}},
\end{aligned}$$

and is given by

$$\operatorname{Cov}\{\hat{\boldsymbol{\theta}}_{\text{LS}}\} = (\boldsymbol{G}^\mathsf{T}\boldsymbol{G})^{-1}\boldsymbol{G}^\mathsf{T}\boldsymbol{R}\boldsymbol{G}((\boldsymbol{G}^\mathsf{T}\boldsymbol{G})^{-1})^\mathsf{T}. \tag{2.18}$$

2.2.3 Weighted Linear Least Squares

As mentioned in Section 1.4, it is sometimes desirable to use a weighted least squares criterion rather than the least squares criterion to take different levels of certainty into account. In this case, the cost function (1.25) with $\boldsymbol{W} = \boldsymbol{R}^{-1}$ is used together with the general linear model (2.7), which yields

$$J_{\text{WLS}}(\boldsymbol{\theta}) = (\boldsymbol{y} - \boldsymbol{G}\boldsymbol{\theta})^\mathsf{T}\boldsymbol{R}^{-1}(\boldsymbol{y} - \boldsymbol{G}\boldsymbol{\theta}). \tag{2.19}$$

Following the same steps for the derivation as for the least squares case, we obtain the weighted least squares estimator given by

$$\hat{\boldsymbol{\theta}}_{\text{WLS}} = (\boldsymbol{G}^\mathsf{T}\boldsymbol{R}^{-1}\boldsymbol{G})^{-1}\boldsymbol{G}^\mathsf{T}\boldsymbol{R}^{-1}\boldsymbol{y}. \tag{2.20}$$

Comparing the linear least squares estimator in (2.16) and the weighted version in (2.20), we see that the former is a special case of the latter if and only if the covariance matrix is proportional to the identity matrix, that is, $\boldsymbol{R} = \sigma_r^2\boldsymbol{I}$.

Similarly, the mean and covariance of the weighted least squares estimator can be shown to be

$$\begin{aligned}\operatorname{E}\{\hat{\boldsymbol{\theta}}_{\text{WLS}}\} &= \boldsymbol{\theta} + (\boldsymbol{G}^\mathsf{T}\boldsymbol{R}^{-1}\boldsymbol{G})^{-1}\boldsymbol{G}^\mathsf{T}\boldsymbol{R}^{-1}\operatorname{E}\{\boldsymbol{r}\} \\ &= \boldsymbol{\theta}\end{aligned}$$

and

$$\begin{aligned}\operatorname{Cov}\{\hat{\boldsymbol{\theta}}_{\text{WLS}}\} &= (\boldsymbol{G}^\mathsf{T}\boldsymbol{R}^{-1}\boldsymbol{G})^{-1}\boldsymbol{G}^\mathsf{T}\boldsymbol{R}^{-1}\boldsymbol{R}\boldsymbol{R}^{-1}\boldsymbol{G}(\boldsymbol{G}^\mathsf{T}\boldsymbol{R}^{-1}\boldsymbol{G})^{-1} \\ &= (\boldsymbol{G}^\mathsf{T}\boldsymbol{R}^{-1}\boldsymbol{G})^{-1}.\end{aligned} \tag{2.21}$$

respectively.

As discussed in Section 1.4, it is possible to make other choices for the weighing matrix $\boldsymbol{W}$. However, it can be shown that choosing the inverse noise covariance matrix $\boldsymbol{R}^{-1}$ is the optimal choice for $\boldsymbol{W}$ in the sense that it minimizes the variance of the estimator $\operatorname{Cov}\{\hat{\boldsymbol{\theta}}\}$.

2.3 Sequential Linear Least Squares

In many cases, the sensor data arrives sequentially at the estimator. Assume that we have calculated the least squares estimate $\hat{\boldsymbol{\theta}}_{n-1}$ according to (2.20) with covariance $\boldsymbol{P}_{n-1} = \operatorname{Cov}\{\hat{\boldsymbol{\theta}}_{n-1}\}$ as in (2.21) for the dataset $\boldsymbol{y}_{1:n-1} = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \dots, \boldsymbol{y}_{n-1}\}$. When a new measurement $\boldsymbol{y}_n$ arrives, the weighted least squares cost function can be written as

$$J_{\text{SLS}}(\boldsymbol{\theta}) = (\boldsymbol{y} - \boldsymbol{G}\boldsymbol{\theta})^\mathsf{T}\boldsymbol{R}^{-1}(\boldsymbol{y} - \boldsymbol{G}\boldsymbol{\theta}) + (\boldsymbol{y}_n - \boldsymbol{C}_n\boldsymbol{\theta})^\mathsf{T}\boldsymbol{R}_n^{-1}(\boldsymbol{y}_n - \boldsymbol{C}_n\boldsymbol{\theta}), \tag{2.22}$$

where the first part is the cost function that was minimized when $n-1$ measurements were available, and the second part is the additional cost for the nth sample.

Minimizing (2.22) is achieved in the same way as for the general linear model in the previous section, except for that there is an additional term. The gradient is given by

$$\frac{\partial J_{\text{SLS}}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = -2\boldsymbol{G}^\mathsf{T}\boldsymbol{R}^{-1}\boldsymbol{y} + 2\boldsymbol{G}^\mathsf{T}\boldsymbol{R}^{-1}\boldsymbol{G}\boldsymbol{\theta} - 2\boldsymbol{C}_n^\mathsf{T}\boldsymbol{R}_n^{-1}\boldsymbol{y}_n + 2\boldsymbol{C}_n^\mathsf{T}\boldsymbol{R}_n^{-1}\boldsymbol{C}_n\boldsymbol{\theta}.$$

Setting the gradient to zero and solving for $\boldsymbol{\theta}$ then yields the updated estimate

$$\begin{aligned}\hat{\boldsymbol{\theta}}_n &= (\boldsymbol{G}^\mathsf{T}\boldsymbol{R}^{-1}\boldsymbol{G} + \boldsymbol{C}_n^\mathsf{T}\boldsymbol{R}_n^{-1}\boldsymbol{C}_n)^{-1}(\boldsymbol{G}^\mathsf{T}\boldsymbol{R}^{-1}\boldsymbol{y} + \boldsymbol{C}_n^\mathsf{T}\boldsymbol{R}_n^{-1}\boldsymbol{y}_n) \\ &= (\boldsymbol{P}_{n-1}^{-1} + \boldsymbol{C}_n^\mathsf{T}\boldsymbol{R}_n^{-1}\boldsymbol{C}_n)^{-1}(\boldsymbol{G}^\mathsf{T}\boldsymbol{R}^{-1}\boldsymbol{y} + \boldsymbol{C}_n^\mathsf{T}\boldsymbol{R}_n^{-1}\boldsymbol{y}_n).\end{aligned}$$

Using the matrix inversion lemma and some further simplifications, we can rewrite this as

$$\hat{\boldsymbol{\theta}}_n = \hat{\boldsymbol{\theta}}_{n-1} + \boldsymbol{L}_n(\boldsymbol{y}_n - \boldsymbol{C}_n\hat{\boldsymbol{\theta}}_{n-1}) \tag{2.23}$$

with the gain

$$\boldsymbol{L}_n = \boldsymbol{P}_{n-1}\boldsymbol{C}_n^\mathsf{T}(\boldsymbol{C}_n\boldsymbol{P}_{n-1}\boldsymbol{C}_n^\mathsf{T} + \boldsymbol{R}_n)^{-1}. \tag{2.24}$$

Knowing that $\mathrm{E}\{\hat{\boldsymbol{\theta}}_{n-1}\} = \theta$ (which follows from Section 2.2), the mean of the updated estimate is given by

$$\begin{aligned}\mathrm{E}\{\hat{\boldsymbol{\theta}}_n\} &= \mathrm{E}\{\hat{\boldsymbol{\theta}}_{n-1}\} + \boldsymbol{L}_n(\mathrm{E}\{\boldsymbol{y}_n\} - \boldsymbol{C}_n\,\mathrm{E}\{\hat{\boldsymbol{\theta}}_{n-1}\}) \\ &= \boldsymbol{\theta},\end{aligned} \tag{2.25}$$

that is, even the updated estimate is unbiased. Furthermore, the covariance of the updated estimate is

$$\begin{aligned}\mathrm{Cov}\{\hat{\boldsymbol{\theta}}_n\} &= \mathrm{E}\{(\hat{\boldsymbol{\theta}}_{n-1} + \boldsymbol{L}_n(\boldsymbol{y}_n - \boldsymbol{C}_n\hat{\boldsymbol{\theta}}_{n-1}) - \boldsymbol{\theta})(\hat{\boldsymbol{\theta}}_{n-1} + \boldsymbol{L}_n(\boldsymbol{y}_n - \boldsymbol{C}_n\hat{\boldsymbol{\theta}}_{n-1}) - \boldsymbol{\theta})^\mathsf{T}\} \\ &= \mathrm{E}\{(\hat{\boldsymbol{\theta}}_{n-1} - \boldsymbol{\theta})(\hat{\boldsymbol{\theta}}_{n-1} - \boldsymbol{\theta})^\mathsf{T}\} + \mathrm{E}\{(\hat{\boldsymbol{\theta}}_{n-1} - \boldsymbol{\theta})(\boldsymbol{y}_n - \boldsymbol{C}_n\hat{\boldsymbol{\theta}}_{n-1})^\mathsf{T}\boldsymbol{L}_n^\mathsf{T}\} \\ &\quad + \mathrm{E}\{\boldsymbol{L}_n(\boldsymbol{y}_n - \boldsymbol{C}_n\hat{\boldsymbol{\theta}}_{n-1})(\hat{\boldsymbol{\theta}}_{n-1} - \boldsymbol{\theta})^\mathsf{T}\} \\ &\quad + \mathrm{E}\{\boldsymbol{L}_n(\boldsymbol{y}_n - \boldsymbol{C}_n\hat{\boldsymbol{\theta}}_{n-1})(\boldsymbol{y}_n - \boldsymbol{C}_n\hat{\boldsymbol{\theta}}_{n-1})^\mathsf{T}\boldsymbol{L}_n^\mathsf{T}\} \\ &= \boldsymbol{P}_{n-1} - \boldsymbol{L}_n(\boldsymbol{C}_n\boldsymbol{P}_{n-1}\boldsymbol{C}^\mathsf{T} + \boldsymbol{R}_n)\boldsymbol{L}_n^\mathsf{T}.\end{aligned} \tag{2.26}$$

Note how the covariance update in (2.26) illustrates how adding new measurements reduces the uncertainty of the estimate.

2.4 Regularized Linear Least Squares

So far, we have assumed that the parameters $\boldsymbol{\theta}$ are completely arbitrary and deterministic. However, in many cases, we have some prior information about the parameters, for example we know beforehand that they might be distributed around some mean $\mathrm{E}\{\boldsymbol{\theta}\} = \boldsymbol{m}$ with covariance $\mathrm{Cov}\{\boldsymbol{\theta}\} = \boldsymbol{P}$. This prior information can be incorporated into the estimator by adding a *regularization term* to the cost function, which leads to the regularized linear least squares estimator.

The cost function then becomes

$$J_{\text{ReLS}}(\boldsymbol{\theta}) = (\boldsymbol{y} - \boldsymbol{G}\boldsymbol{\theta})^{\mathsf{T}} \boldsymbol{R}^{-1} (\boldsymbol{y} - \boldsymbol{G}\boldsymbol{\theta}) + (\boldsymbol{\theta} - \boldsymbol{m})^{\mathsf{T}} \boldsymbol{P}^{-1} (\boldsymbol{\theta} - \boldsymbol{m}), \tag{2.27}$$

where the second term $(\boldsymbol{\theta} - \boldsymbol{m})^{\mathsf{T}} \boldsymbol{P}^{-1} (\boldsymbol{\theta} - \boldsymbol{m})$ is a regularization term that encodes the prior information.

As it can be seen, the cost function (2.27) has the same structure as the sequential least squares cost function in (2.22). Hence, finding the estimator for this case closely follows the derivation for the sequential case. The gradient is given by

$$\frac{\partial J_{\text{ReLS}}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = -2\boldsymbol{G}^{\mathsf{T}} \boldsymbol{R}^{-1} \boldsymbol{y} + 2\boldsymbol{G}^{\mathsf{T}} \boldsymbol{R}^{-1} \boldsymbol{G}\boldsymbol{\theta} - 2\boldsymbol{P}^{-1}\boldsymbol{m} + 2\boldsymbol{P}^{-1}\boldsymbol{\theta}.$$

Setting it to zero and solving for $\boldsymbol{\theta}$ then yields the regularized linear least squares estimator

$$\hat{\boldsymbol{\theta}}_{\text{ReLS}} = (\boldsymbol{G}^{\mathsf{T}} \boldsymbol{R}^{-1} \boldsymbol{G} + \boldsymbol{P}^{-1})^{-1} (\boldsymbol{G}^{\mathsf{T}} \boldsymbol{R}^{-1} \boldsymbol{y} + \boldsymbol{P}^{-1} \boldsymbol{m}). \tag{2.28}$$

The estimator in (2.28) shows that when using the regularization approach, the resulting estimate is a weighted average of the data, which enters through the term $\boldsymbol{G}^{\mathsf{T}} \boldsymbol{R}^{-1} \boldsymbol{y}$ and the prior information through $\boldsymbol{P}^{-1}\boldsymbol{m}$. The contributions of each of the terms are determined by two factors. First, there are the weighing matrices $\boldsymbol{R}^{-1}$ and $\boldsymbol{P}^{-1}$, which directly weigh the contributions. Second, the number of data points in $\boldsymbol{y}$ also affect how much emphasis is put on the data, and how much on the prior information.

Similar as in for the sequential least squares case, the estimator (2.28) can be reformulated using the matrix inversion lemma to find a second form. This form is given by

$$\hat{\boldsymbol{\theta}}_{\text{ReLS}} = \boldsymbol{m} + \boldsymbol{K}(\boldsymbol{y} - \boldsymbol{G}\boldsymbol{m}), \tag{2.29}$$

where $\boldsymbol{K}$ is a gain given by

$$\boldsymbol{K} = \boldsymbol{P}\boldsymbol{G}^{\mathsf{T}} (\boldsymbol{G}\boldsymbol{P}\boldsymbol{G}^{\mathsf{T}} + \boldsymbol{R})^{-1}. \tag{2.30}$$

In this form, a second interpretation of the regularized linear least squares estimator is possible. Recall that $\boldsymbol{m}$ was the prior mean. Then, the term $\boldsymbol{y} - \boldsymbol{G}\boldsymbol{m}$ is the error between the output and the output predicted by the prior mean. The final estimate is then the prior mean that is corrected by the error term through the weight $\boldsymbol{K}$.

The mean of the estimator follows to be

$$\begin{aligned} \mathrm{E}\{\hat{\boldsymbol{\theta}}_{\text{ReLS}}\} &= \mathrm{E}\{\boldsymbol{m} + \boldsymbol{K}(\boldsymbol{y} - \boldsymbol{G}\boldsymbol{m})\} \\ &= \boldsymbol{m} + \boldsymbol{K}(\boldsymbol{G}\,\mathrm{E}\{\boldsymbol{\theta}\} - \boldsymbol{G}\boldsymbol{m}). \end{aligned} \tag{2.31}$$

From (2.31) it can be seen that if $\boldsymbol{\theta}$ is indeed a random variable with mean $\boldsymbol{m}$, then the estimator will converge to that mean on average. However, if $\boldsymbol{\theta}$ does not follow the a prior assumed statistics, then a bias is introduced. The effect of that bias depends on several factors such as the number of data points, the prior covariance $\boldsymbol{P}$, or the measurement noise covariance $\boldsymbol{R}$ (the latter two enter through the gain $\boldsymbol{K}$).

Finally, we can also show that the covariance of the regularized linear least squares estimator is similar to (2.26) given by

$$\operatorname{Cov}\{\hat{\boldsymbol{\theta}}_{\text{ReLS}}\} = \boldsymbol{P} - \boldsymbol{K}(\boldsymbol{G}\boldsymbol{P}\boldsymbol{G}^{\mathsf{T}} + \boldsymbol{R})\boldsymbol{K}^{\mathsf{T}}. \tag{2.32}$$

Chapter 3

Static Nonlinear Models

In Chapter 2, we discussed problems where the sensor model is linear in the parameters of interest. For these model types, closed-form estimators exist and furthermore, we can analytically determine the estimators' properties such as biasedness or covariance. However, despite these desirable characteristics, linear models can be limiting and a nonlinear model may be more accurate in describing the relationship between the sensors' measurements and the parameters of interest. Hence, in this section, we develop estimation algorithms for general models of the form

$$\boldsymbol{y} = g(\boldsymbol{\theta}) + \boldsymbol{r}, \tag{3.1}$$

where $g(\boldsymbol{\theta})$ may be an arbitrary nonlinear (vector-valued) function. As for the linear models, they aim will be to minimize the weighted least squares cost function

$$J_{\text{WLS}}(\boldsymbol{\theta}) = (\boldsymbol{y} - g(\boldsymbol{\theta}))^{\mathsf{T}} \boldsymbol{R}^{-1} (\boldsymbol{y} - g(\boldsymbol{\theta})). \tag{3.2}$$

In some special cases, it is still possible to find analytical expressions at least for the (W)LS estimator and possibly even for its properties. Hence, it is always worth trying to derive it. However, for the vast majority of models, this is not possible and we have to resort to iterative, numerical optimization methods to minimize the cost function (3.2). For this purpose, two main approaches can be distinguished:

1. Methods that minimize the cost function directly, and
2. methods that find the zeros of the gradient.

Here, we will introduce three methods, all of which from the first class:

- Gradient descent,
- the Gauss–Newton algorithm, and
- the Levenberg-Marquardt algorithm.

Note that all of these algorithms typically only are able to find local minima, unless there is only a global minimum. Hence, it is important to have good initial guesses of the parameters.

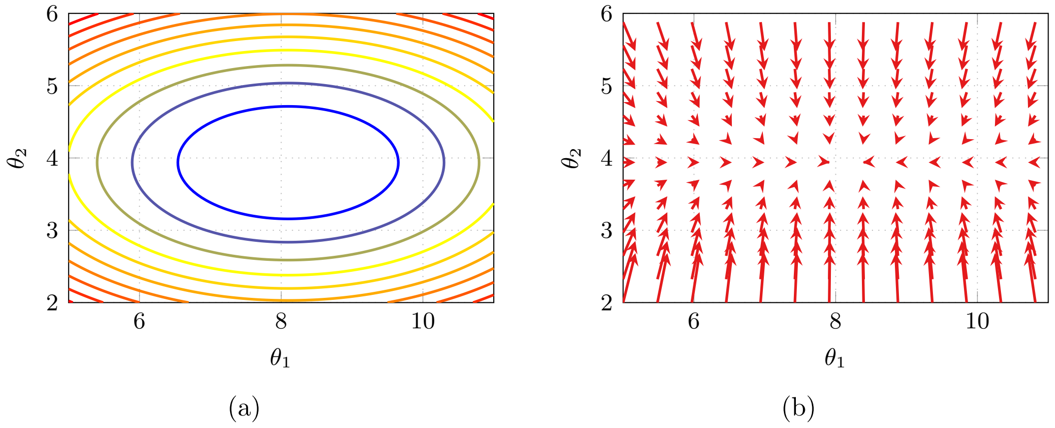


Figure 3.1: Illustration of the cost function for a nonlinear problem with parameters $\boldsymbol{\theta} = \begin{bmatrix}\theta_1 & \theta_2\end{bmatrix}^\mathsf{T}$. (a) Contours of the cost function, and (b) vector field defined by the negative gradient.

3.1 Gradient Descent

The first approach, *gradient descent* is a method based on the following strategy. Assume that we are given the current parameter estimate $\hat{\boldsymbol{\theta}}^{(i)}$ at the algorithm's ith iteration. Then, the gradient of the objective function at $\hat{\boldsymbol{\theta}}^{(i)}$ indicates the direction of the function input $\boldsymbol{\theta}$ in which the function value increases. Hence, taking steps from $\hat{\boldsymbol{\theta}}^{(i)}$ in the direction proportional to the negative gradient, the function value should decrease (see Figure 3.1).

This leads to the update rule given by

$$\hat{\boldsymbol{\theta}}^{(i+1)} = \hat{\boldsymbol{\theta}}^{(i)} - \gamma \nabla_{\boldsymbol{\theta}} J_{\mathrm{WLS}}(\boldsymbol{\theta})\big|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}^{(i)}}, \tag{3.3}$$

where $\nabla_{\boldsymbol{\theta}} J_{\mathrm{WLS}}(\boldsymbol{\theta})$ denotes the gradient of $J_{\mathrm{WLS}}(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$ and $\gamma > 0$ is a positive constant defining the step length. For simplicity, we start our derivation assuming scalar measurements y_n together with the (non-weighted) least squares cost function and later generalize it to the general model (3.1)–(3.2). In this case, we have that

$$J_{\mathrm{LS}}(\boldsymbol{\theta}) = \sum_{n=1}^{N} (y_n - g_n(\boldsymbol{\theta}))^2$$

and thus

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} J_{\mathrm{LS}}(\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}} \sum_{n=1}^{N} (y_n - g_n(\boldsymbol{\theta}))^2 \\ &= \sum_{n=1}^{N} -2(y_n - g_n(\boldsymbol{\theta}))\nabla_{\boldsymbol{\theta}} g_n(\boldsymbol{\theta}).\end{aligned}$$

The sum can be rewritten in vector form as

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} J_{\mathrm{LS}}(\boldsymbol{\theta}) &= -2\left[\nabla g_1(\boldsymbol{\theta}) \quad \nabla g_2(\boldsymbol{\theta}) \quad \dots \quad \nabla g_N(\boldsymbol{\theta})\right]\left(\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} - \begin{bmatrix} g_1(\boldsymbol{\theta}) \\ g_2(\boldsymbol{\theta}) \\ \vdots \\ g_N(\boldsymbol{\theta}) \end{bmatrix}\right) \\
&= -2 \begin{bmatrix} \frac{\partial g_1(\boldsymbol{\theta})}{\partial \theta_1} & \frac{\partial g_2(\boldsymbol{\theta})}{\partial \theta_1} & \dots & \frac{\partial g_N(\boldsymbol{\theta})}{\partial \theta_1} \\ \frac{\partial g_1(\boldsymbol{\theta})}{\partial \theta_2} & \frac{\partial g_2(\boldsymbol{\theta})}{\partial \theta_2} & & \vdots \\ \vdots & & \ddots & \frac{\partial g_N(\boldsymbol{\theta})}{\partial \theta_{K-1}} \\ \frac{\partial g_1(\boldsymbol{\theta})}{\partial \theta_K} & \dots & \frac{\partial g_{N-1}(\boldsymbol{\theta})}{\partial \theta_K} & \frac{\partial g_N(\boldsymbol{\theta})}{\partial \theta_K} \end{bmatrix} (\boldsymbol{y} - g(\boldsymbol{\theta})),
\end{aligned}$$

and the matrix of derivatives can be identified as the transpose of the Jacobian matrix given by

$$\boldsymbol{G}_\theta = \begin{bmatrix} \frac{\partial g_1(\boldsymbol{\theta})}{\partial \theta_1} & \frac{\partial g_1(\boldsymbol{\theta})}{\partial \theta_2} & \dots & \frac{\partial g_1(\boldsymbol{\theta})}{\partial \theta_K} \\ \frac{\partial g_2(\boldsymbol{\theta})}{\partial \theta_1} & \frac{\partial g_2(\boldsymbol{\theta})}{\partial \theta_2} & & \vdots \\ \vdots & & \ddots & \frac{\partial g_{N-1}(\boldsymbol{\theta})}{\partial \theta_K} \\ \frac{\partial g_N(\boldsymbol{\theta})}{\partial \theta_1} & \dots & \frac{\partial g_N(\boldsymbol{\theta})}{\partial \theta_K-1} & \frac{\partial g_N(\boldsymbol{\theta})}{\partial \theta_K} \end{bmatrix}, \tag{3.4}$$

which is indeed the gradient with respect to the vector $\boldsymbol{\theta}$ for vector valued $g(\boldsymbol{\theta})$ (Petersen and Pedersen, 2012). Hence, we have that

$$\nabla_{\boldsymbol{\theta}} J_{\mathrm{LS}}(\boldsymbol{\theta}) = -2\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}}(\boldsymbol{y} - g(\boldsymbol{\theta})). \tag{3.5}$$

We can now directly generalize (3.5) to the general cost (3.2), which yields

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} J_{\mathrm{WLS}}(\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}}(\boldsymbol{y} - g(\boldsymbol{\theta}))^{\mathsf{T}}\boldsymbol{R}^{-1}(\boldsymbol{y} - g(\boldsymbol{\theta})) \\
&= \nabla_{\boldsymbol{\theta}}\left[\boldsymbol{y}^{\mathsf{T}}\boldsymbol{R}^{-1}\boldsymbol{y} - \boldsymbol{y}^{\mathsf{T}}\boldsymbol{R}^{-1}g(\boldsymbol{\theta}) - g(\boldsymbol{\theta})^{\mathsf{T}}\boldsymbol{R}^{-1}\boldsymbol{y} + g(\boldsymbol{\theta})\boldsymbol{R}^{-1}g(\boldsymbol{\theta})\right] \\
&= -2\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}}\boldsymbol{R}^{-1}(\boldsymbol{y} - g(\boldsymbol{\theta})).
\end{aligned}$$

Note that the direction of the parameter update is given by $-\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}}\boldsymbol{R}^{-1}(\boldsymbol{y} - g(\boldsymbol{\theta}))$ and hence, the factor two can be dropped. This yields the gradient descent update given by

$$\hat{\boldsymbol{\theta}}^{(i+1)} = \hat{\boldsymbol{\theta}}^{(i)} + \gamma\Delta\boldsymbol{\theta}^{(i+1)}, \tag{3.6a}$$

$$\Delta\boldsymbol{\theta}^{(i+1)} = \boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}}\boldsymbol{R}^{-1}(\boldsymbol{y} - g(\hat{\boldsymbol{\theta}}^{(i)})), \tag{3.6b}$$

where the Jacobian matrix $\boldsymbol{G}_{\boldsymbol{\theta}}$ is evaluated at $\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}^{(i)}$.

It remains to determine how long the step length γ should be chosen. Choosing γ too large may cause the cost function to increase, whereas too small steps might cause unnecessarily slow convergence. Hence, the value for γ should be determined dynamically, at each iteration. This can be achieved using a *line search*, where γ is determined such

Algorithm 3.1 Gradient Descent with Line Search

```
Input: Initial parameter guess $\hat{\boldsymbol{\theta}}^{(0)}$, data $\boldsymbol{y}$, function $g(\boldsymbol{\theta})$, Jacobian $\boldsymbol{G}_{\boldsymbol{\theta}}$
Output: Parameter estimate $\hat{\boldsymbol{\theta}}_{\text{WLS}}$
 1: Set $i \leftarrow 0$
 2: repeat
 3:     Calculate the update direction
            $\Delta\boldsymbol{\theta}^{(i+1)} = \boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}} \boldsymbol{R}^{-1}(\boldsymbol{y} - g(\hat{\boldsymbol{\theta}}^{(i)}))$
 4:     Set $\gamma^{(i+1)} \leftarrow 1$
 5:     repeat                                              ▷ Line Search
 6:         Calculate
                $\hat{\boldsymbol{\theta}}^{(i+1)} = \hat{\boldsymbol{\theta}}^{(i)} + \gamma^{(i+1)}\Delta\boldsymbol{\theta}^{(i+1)}$
 7:         Set $\gamma^{(i+1)} \leftarrow \gamma^{(i+1)}/2$
 8:     until $J_{\text{WLS}}(\hat{\boldsymbol{\theta}}^{(i+1)}) < J_{\text{WLS}}(\hat{\boldsymbol{\theta}}^{(i)})$
 9:     Set $i \leftarrow i + 1$
10: until Converged
```

that it minimizes (or at least reduces) the cost function along the direction specified by $\Delta\boldsymbol{\theta}^{(i+1)}$. A simple strategy is to start with an initial step length (e.g., $\gamma = 1$), calculate the updated parameters, and check whether the cost has decreased or not. In the latter case, the step length is decreased (e.g., to $\gamma = 1/2$) and the cost is evaluated again. This procedure is repeated until the cost is decreased (Gustafsson, 2012). Note that this strategy only ensures that the cost is reduced, but not that it is minimized at each iteration. Naturally, there are other line search strategies, some of which are discussed in, for example, Boyd and Vandenberghe (2004). Finally, this leads to the gradient descent algorithm shown in Algorithm 3.1.

While intuitive, the gradient descent approach suffers from an important problem. In areas where the cost function is flat, the gradient is very small. Hence, the algorithm can only take small steps, which leads to the problem that a large number of iterations are needed to cross such areas.

3.2 Gauss–Newton Algorithm

The second method, the *Gauss–Newton* algorithm, is based on a different approach. Given the estimate $\hat{\boldsymbol{\theta}}^{(i)}$ of $\boldsymbol{\theta}$ at the ith iteration, the nonlinear function $g(\boldsymbol{\theta})$ can locally be approximated by using a first order Taylor series expansion. This approximation is given by

$$\begin{aligned} g(\boldsymbol{\theta}) &\approx g(\hat{\boldsymbol{\theta}}^{(i)}) + \nabla_{\boldsymbol{\theta}} g(\boldsymbol{\theta})(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)}) \\ &= g(\hat{\boldsymbol{\theta}}^{(i)}) + \boldsymbol{G}_{\boldsymbol{\theta}}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)}) \end{aligned} \tag{3.7}$$

where $\boldsymbol{G}_{\boldsymbol{\theta}}$ is the Jacobian matrix of $g(\boldsymbol{\theta})$ given by (3.4). Using this local linear approximation of the nonlinear function, the weighted least squares cost can be approximated by

$$\begin{aligned} J_{\text{WLS}}(\boldsymbol{\theta}) &\approx \left(\boldsymbol{y} - g(\hat{\boldsymbol{\theta}}^{(i)}) - \boldsymbol{G}_{\boldsymbol{\theta}}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)})\right)^{\mathsf{T}} \boldsymbol{R}^{-1} \left(\boldsymbol{y} - g(\hat{\boldsymbol{\theta}}^{(i)}) - \boldsymbol{G}_{\boldsymbol{\theta}}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)})\right) \\ &= \left(\boldsymbol{e}^{(i)} - \boldsymbol{G}_{\boldsymbol{\theta}}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)})\right)^{\mathsf{T}} \boldsymbol{R}^{-1} \left(\boldsymbol{e}^{(i)} - \boldsymbol{G}_{\boldsymbol{\theta}}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)})\right) \end{aligned} \tag{3.8}$$

were we have made use of $\boldsymbol{e}^{(i)} = \boldsymbol{y} - g(\hat{\boldsymbol{\theta}}^{(i)})$ (which is the error of the ith iteration). The approximated cost function (3.8) is linear in the parameters $\boldsymbol{\theta}$ (remember that $\hat{\boldsymbol{\theta}}^{(i)}$ is constant and given from the last iteration) and hence, we can derive a closed form solution for the parameter update. This is achieved by setting the approximative cost function's gradient to zero and solving for $\boldsymbol{\theta}$ as

$$\begin{aligned} \frac{\partial J_{\text{WLS}}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} &\approx \frac{\partial}{\partial \boldsymbol{\theta}} \Big((\boldsymbol{e}^{(i)})^{\mathsf{T}} \boldsymbol{R}^{-1} \boldsymbol{e}^{(i)} - (\boldsymbol{e}^{(i)})^{\mathsf{T}} \boldsymbol{R}^{-1} \boldsymbol{G}_{\boldsymbol{\theta}} (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)}) \\ &\qquad - (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)})^{\mathsf{T}} \boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}} \boldsymbol{R}^{-1} \boldsymbol{e}^{(i)} + (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)})^{\mathsf{T}} \boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}} \boldsymbol{R}^{-1} \boldsymbol{G}_{\boldsymbol{\theta}} (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)}) \Big) \\ &= -2\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}} \boldsymbol{R}^{-1} \boldsymbol{e}^{(i)} + 2\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}} \boldsymbol{R}^{-1} \boldsymbol{G}_{\boldsymbol{\theta}} (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)}) \end{aligned}$$

and thus

$$\begin{aligned} \hat{\boldsymbol{\theta}}^{(i+1)} &= \hat{\boldsymbol{\theta}}^{(i)} + (\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}} \boldsymbol{R}^{-1} \boldsymbol{G}_{\boldsymbol{\theta}})^{-1} \boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}} \boldsymbol{R}^{-1} \boldsymbol{e}^{(i)} \\ &= \hat{\boldsymbol{\theta}}^{(i)} + (\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}} \boldsymbol{R}^{-1} \boldsymbol{G}_{\boldsymbol{\theta}})^{-1} \boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}} \boldsymbol{R}^{-1} (\boldsymbol{y} - g(\hat{\boldsymbol{\theta}}^{(i)})). \end{aligned} \tag{3.9}$$

In practice, taking a full step according to (3.9) might be too large with respect to the neighborhood for which the Taylor series approximation (3.7) is valid. To avoid this, a scaled Gauss–Newton step can be done instead, proportional to the direction given by the local approximation. This yields

$$\hat{\boldsymbol{\theta}}^{(i+1)} = \hat{\boldsymbol{\theta}}^{(i)} + \gamma \Delta \hat{\boldsymbol{\theta}}^{(i+1)}, \tag{3.10a}$$

$$\Delta \hat{\boldsymbol{\theta}}^{(i+1)} = (\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}} \boldsymbol{R}^{-1} \boldsymbol{G}_{\boldsymbol{\theta}})^{-1} \boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}} \boldsymbol{R}^{-1} (\boldsymbol{y} - g(\hat{\boldsymbol{\theta}}^{(i)})), \tag{3.10b}$$

where γ is the scaling factor. Similar to the gradient descent method, the step length can be chosen such that it ensures reducing the cost. Hence, the Gauss–Newton algorithm using the same line search strategy as in Section 3.1 is given in Algorithm 3.2. As it can be seen, the only difference between the gradient descent and the Gauss–Newton algorithm is the direction of the parameter update.

A major disadvantage of the Gauss–Newton approach is that the linearization of the measurement model is local. In highly nonlinear problems, this means that it is dangerous to extrapolate too much and only small steps can be taken at a time to avoid missing local minima.

Algorithm 3.2 Gauss–Newton Algorithm with Line Search

```
Input: Initial parameter guess $\hat{\boldsymbol{\theta}}^{(0)}$, data $\boldsymbol{y}$, function $g(\boldsymbol{\theta})$, Jacobian $\boldsymbol{G}_{\boldsymbol{\theta}}$
Output: Parameter estimate $\hat{\boldsymbol{\theta}}_{\mathrm{WLS}}$
 1: Set $i \leftarrow 0$
 2: repeat
 3:     Calculate the update direction
            $$\Delta\boldsymbol{\theta}^{(i+1)} = (\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}} \boldsymbol{R}^{-1} \boldsymbol{G}_{\boldsymbol{\theta}})^{-1} \boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}} \boldsymbol{R}^{-1} (\boldsymbol{y} - g(\hat{\boldsymbol{\theta}}^{(i)}))$$
 4:     Set $\gamma^{(i+1)} \leftarrow 1$
 5:     repeat                                                  ▷ Line Search
 6:         Calculate
                $$\hat{\boldsymbol{\theta}}^{(i+1)} = \hat{\boldsymbol{\theta}}^{(i)} + \gamma^{(i+1)} \Delta\boldsymbol{\theta}^{(i+1)}$$
 7:         Set $\gamma^{(i+1)} \leftarrow \gamma^{(i+1)}/2$
 8:     until $J_{\mathrm{WLS}}(\hat{\boldsymbol{\theta}}^{(i+1)}) < J_{\mathrm{WLS}}(\hat{\boldsymbol{\theta}}^{(i)})$
 9:     Set $i \leftarrow i + 1$
10: until Converged
```

3.3 Levenberg–Marquardt Algorithm

As discussed, the gradient descent and Gauss–Newton algorithms suffer from problems that may make it difficult to use them in practice. The Levenberg–Marquardt algorithm tries to address these problems by combining the good properties of both algorithms (Levenberg, 1944; Marquardt, 1963).

One way of deriving the Levenberg–Marquardt algorithm is to see it as a regularized Gauss–Newton algorithm. The Taylor series approximation of the nonlinear function (3.7) is only valid in the local neighborhood of the current parameter estimate $\hat{\boldsymbol{\theta}}^{(i)}$. Hence, we can use this prior information to regularize the approximated weighted least squares cost function using a regularization term as introduced in Section 2.4. This yields the cost function approximation

$$\begin{aligned}
J_{\mathrm{ReLS}}(\boldsymbol{\theta}) &\approx \left(\boldsymbol{y} - g(\hat{\boldsymbol{\theta}}^{(i)}) - \boldsymbol{G}_{\boldsymbol{\theta}}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)})\right)^{\mathsf{T}} \boldsymbol{R}^{-1} \left(\boldsymbol{y} - g(\hat{\boldsymbol{\theta}}^{(i)}) - \boldsymbol{G}_{\boldsymbol{\theta}}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)})\right) \\
&\qquad + \lambda(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)})^{\mathsf{T}}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)}) \\
&= \left(\boldsymbol{e}^{(i)} - \boldsymbol{G}_{\boldsymbol{\theta}}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)})\right)^{\mathsf{T}} \boldsymbol{R}^{-1} \left(\boldsymbol{e}^{(i)} - \boldsymbol{G}_{\boldsymbol{\theta}}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)})\right) + \lambda(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)})^{\mathsf{T}}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)})
\end{aligned}$$

where the second term is the regularization around the last iteration's estimate with weight or *damping factor* λ. The gradient of the cost function approximation now becomes

$$\begin{aligned}
\frac{\partial J_{\mathrm{ReLS}}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} &\approx -2\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}} \boldsymbol{R}^{-1} \boldsymbol{e}^{(i)} + 2\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}} \boldsymbol{R}^{-1} \boldsymbol{G}_{\boldsymbol{\theta}}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)}) + 2\lambda(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)}) \\
&= -2\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}} \boldsymbol{R}^{-1} \boldsymbol{e}^{(i)} + 2(\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}} \boldsymbol{R}^{-1} \boldsymbol{G}_{\boldsymbol{\theta}} + \lambda \boldsymbol{I})(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)})
\end{aligned} \tag{3.11}$$

Setting (3.11) to zero and rearranging yields

$$(\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}}\boldsymbol{R}^{-1}\boldsymbol{G}_{\boldsymbol{\theta}} + \lambda\boldsymbol{I})(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)}) = \boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}}\boldsymbol{R}^{-1}\boldsymbol{e}^{(i)}$$

and thus,

$$\hat{\boldsymbol{\theta}}^{(i+1)} = \hat{\boldsymbol{\theta}}^{(i)} + \Delta\hat{\boldsymbol{\theta}}^{(i+1)}, \tag{3.12a}$$

$$\Delta\hat{\boldsymbol{\theta}}^{(i+1)} = (\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}}\boldsymbol{R}^{-1}\boldsymbol{G}_{\boldsymbol{\theta}} + \lambda\boldsymbol{I})^{-1}\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}}\boldsymbol{R}^{-1}(\boldsymbol{y} - g(\hat{\boldsymbol{\theta}}^{(i)})). \tag{3.12b}$$

Equation (3.12b) shows that when λ approaches zero, the algorithm converges to the Gauss–Newton algorithm. On the other hand, for large values of λ, the term $\lambda\boldsymbol{I}$ will dominate. In this case, we have that

$$\Delta\hat{\boldsymbol{\theta}}^{(i+1)} \approx \frac{1}{\lambda}\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}}\boldsymbol{R}^{-1}(\boldsymbol{y} - g(\hat{\boldsymbol{\theta}}^{(i)})),$$

which is a scaled gradient descent step. Thus, an important question is how to choose λ. Ideally, in flat regions of the cost function where the linear approximation is good, λ should be chosen small, whereas in steep regions, it should be chosen large. Unfortunately, there is no definite method on how to chose and adapt λ and several different approaches have been proposed, but the following adaption scheme has shown to work well (Nielsen, 1999).

First, note that the reduction in cost for the linearized model (3.7) is given by

$$\begin{aligned}
\Delta J_{\text{WLS}}(\boldsymbol{\theta}) &= J_{\text{WLS}}(\hat{\boldsymbol{\theta}}^{(i)}) - J_{\text{WLS}}(\boldsymbol{\theta}) \\
&\approx (\boldsymbol{y} - g(\hat{\boldsymbol{\theta}}^{(i)}))^{\mathsf{T}}\boldsymbol{R}^{-1}(\boldsymbol{y} - g(\hat{\boldsymbol{\theta}}^{(i)})) \\
&\quad - \left(\boldsymbol{y} - g(\hat{\boldsymbol{\theta}}^{(i)}) - \boldsymbol{G}_{\boldsymbol{\theta}}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)})\right)^{\mathsf{T}}\boldsymbol{R}^{-1}\left(\boldsymbol{y} - g(\hat{\boldsymbol{\theta}}^{(i)}) - \boldsymbol{G}_{\boldsymbol{\theta}}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)})\right) \\
&= 2(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)})^{\mathsf{T}}\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}}\boldsymbol{R}^{-1}(\boldsymbol{y} - g(\hat{\boldsymbol{\theta}}^{(i)})) - (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)})^{\mathsf{T}}\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}}\boldsymbol{R}^{-1}\boldsymbol{G}_{\boldsymbol{\theta}}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)}).
\end{aligned}$$

Next, we can introduce the ratio of the reduction in cost between the actual cost function and its linear approximation, that is,

$$\rho(\boldsymbol{\theta}) = \frac{J_{\text{WLS}}(\hat{\boldsymbol{\theta}}^{(i)}) - J_{\text{WLS}}(\boldsymbol{\theta})}{2(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)})^{\mathsf{T}}\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}}\boldsymbol{R}^{-1}(\boldsymbol{y} - g(\hat{\boldsymbol{\theta}}^{(i)})) - (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)})^{\mathsf{T}}\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}}\boldsymbol{R}^{-1}\boldsymbol{G}_{\boldsymbol{\theta}}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)})}. \tag{3.13}$$

Then, we can use the following adaption rule for the damping parameter λ where ν is a positive constant (Nielsen, 1999):

- If $\rho > 0$ (i.e., the cost is reduced), accept $\Delta\hat{\boldsymbol{\theta}}^{(i)}$ and decrease the damping by setting

$$\begin{aligned}
&\lambda \leftarrow \lambda \max(1/3, 1 - (2\rho - 1)^3), \\
&\nu = 2,
\end{aligned}$$

Algorithm 3.3 Levenberg–Marquardt Algorithm

```
Input: Initial parameter guess $\hat{\boldsymbol{\theta}}^{(0)}$, data $\boldsymbol{y}$, function $g(\boldsymbol{\theta})$, Jacobian $\boldsymbol{G}_{\boldsymbol{\theta}}$
Output: Parameter estimate $\hat{\boldsymbol{\theta}}_{\mathrm{WLS}}$
 1: Set $i \leftarrow 0$, $\lambda = \max(\mathrm{diag}(\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}}\boldsymbol{R}^{-1}\boldsymbol{G}_{\boldsymbol{\theta}}))$, and $\nu \leftarrow 2$
 2: repeat
 3:     repeat
 4:         Calculate
                $\Delta\hat{\boldsymbol{\theta}}^{(i+1)} = (\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}}\boldsymbol{R}^{-1}\boldsymbol{G}_{\boldsymbol{\theta}} + \lambda\boldsymbol{I})^{-1}\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}}\boldsymbol{R}^{-1}(\boldsymbol{y} - g(\hat{\boldsymbol{\theta}}^{(i)}))$
                $\rho = \frac{J_{\mathrm{WLS}}(\hat{\boldsymbol{\theta}}^{(i)}) - J_{\mathrm{WLS}}(\hat{\boldsymbol{\theta}}^{(i)} + \Delta\hat{\theta}^{(i+1)})}{2(\Delta\hat{\theta}^{(i+1)})^{\mathsf{T}}\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}}\boldsymbol{R}^{-1}(\boldsymbol{y} - g(\hat{\boldsymbol{\theta}}^{(i)})) - (\Delta\hat{\theta}^{(i+1)})^{\mathsf{T}}\boldsymbol{G}_{\boldsymbol{\theta}}^{\mathsf{T}}\boldsymbol{R}^{-1}\boldsymbol{G}_{\boldsymbol{\theta}}\Delta\hat{\theta}^{(i+1)}}$
 5:         if $\rho > 0$ then
 6:             Set
                    $\hat{\boldsymbol{\theta}}^{(i+1)} = \hat{\boldsymbol{\theta}}^{(i+1)} + \Delta\hat{\boldsymbol{\theta}}^{(i+1)}$
                    $\lambda \leftarrow \lambda \max(1/3, 1 - (2\rho - 1)^3)$
                    $\nu = 2$
 7:         else
 8:             Set
                    $\lambda \leftarrow \nu\lambda$
                    $\nu \leftarrow 2\nu$
 9:         end if
10:     until $\Delta\boldsymbol{\theta}^{(i+1)}$ is accepted
11:     Set $i \leftarrow i + 1$
12: until Converged
```

- otherwise, increase the damping by setting

$$\lambda \leftarrow \nu\lambda,$$
$$\nu \leftarrow 2\nu.$$

The final Levenberg–Marquardt algorithm with this adaption scheme for the damping parameter λ is shown in Algorithm 3.3.

3.4 Convergence Criteria

The algorithms introduced in the previous sections are all based on an iterative scheme, where the parameter estimates are improved sequentially, based on the last estimate. An important question therefore is, when to terminate the search. Three simple and easy to check criteria are:

- The absolute or relative change in the cost falls below a certain threshold,
- the absolute or relative change in the parameter estimate falls below a threshold,
- a maximum number of iterations is reached.

It is common practice to monitor all or at least a subset of these criteria (plus possibly others, too) and terminate the search when at least one of the criteria is fulfilled. In order to avoid infinite loops, the last criterion (a maximum number of iterations) should always be checked for, possibly issuing a warning when this is the reason for termination of the algorithm.

3.5 Separable Models

Sometimes, the model is only nonlinear in a subset of parameters, while it is linear in others. In this case, the model can be written in the form

$$\boldsymbol{y} = \boldsymbol{G}(\boldsymbol{\theta}_1)\boldsymbol{\theta}_2 + \boldsymbol{r}, \tag{3.14}$$

where $\boldsymbol{\theta}_1$ are the nonlinear parameters and $\boldsymbol{\theta}_2$ the linear ones.

In this case, we know that given the nonlinear parameters $\boldsymbol{\theta}_1$, the solution for $\boldsymbol{\theta}_2$ is (see Section 2.2)

$$\hat{\boldsymbol{\theta}}_2 = (\boldsymbol{G}(\boldsymbol{\theta}_1)^\mathsf{T}\boldsymbol{R}^{-1}\boldsymbol{G}(\boldsymbol{\theta}_1))^{-1}\boldsymbol{G}(\boldsymbol{\theta}_1)\boldsymbol{R}^{-1}\boldsymbol{y}.$$

Substituting this in the weighted least squares cost function (3.2) yields (after some simplifications)

$$\begin{aligned} J_{WLS}(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2) &= \boldsymbol{y}^\mathsf{T}\boldsymbol{y} - \boldsymbol{y}^\mathsf{T}\boldsymbol{R}^{-1}\boldsymbol{G}(\boldsymbol{\theta}_1)(\boldsymbol{G}(\boldsymbol{\theta}_1)^\mathsf{T}\boldsymbol{R}^{-1}\boldsymbol{G}(\boldsymbol{\theta}_1))^{-1}\boldsymbol{G}(\boldsymbol{\theta}_1)^\mathsf{T}\boldsymbol{R}^{-1}\boldsymbol{y} \\ &\quad - ((\boldsymbol{G}(\boldsymbol{\theta}_1)^\mathsf{T}\boldsymbol{R}^{-1}\boldsymbol{G}(\boldsymbol{\theta}_1))^{-1}\boldsymbol{G}(\boldsymbol{\theta}_1)\boldsymbol{R}^{-1}\boldsymbol{y})^\mathsf{T}\boldsymbol{G}(\boldsymbol{\theta}_1)^\mathsf{T}\boldsymbol{R}^{-1}\boldsymbol{y} \\ &\quad + ((\boldsymbol{G}(\boldsymbol{\theta}_1)^\mathsf{T}\boldsymbol{R}^{-1}\boldsymbol{G}(\boldsymbol{\theta}_1))^{-1}\boldsymbol{G}(\boldsymbol{\theta}_1)\boldsymbol{R}^{-1}\boldsymbol{y})^\mathsf{T} \\ &\quad \times \boldsymbol{G}(\boldsymbol{\theta}_1)^\mathsf{T}\boldsymbol{R}^{-1}\boldsymbol{G}(\boldsymbol{\theta}_1)(\boldsymbol{G}(\boldsymbol{\theta}_1)^\mathsf{T}\boldsymbol{R}^{-1}\boldsymbol{G}(\boldsymbol{\theta}_1))^{-1}\boldsymbol{G}(\boldsymbol{\theta}_1)^\mathsf{T}\boldsymbol{R}^{-1}\boldsymbol{y} \\ &= \boldsymbol{y}^\mathsf{T}\boldsymbol{y} - \boldsymbol{y}^\mathsf{T}\boldsymbol{R}^{-1}\boldsymbol{G}(\boldsymbol{\theta}_1)(\boldsymbol{G}(\boldsymbol{\theta}_1)^\mathsf{T}\boldsymbol{R}^{-1}\boldsymbol{G}(\boldsymbol{\theta}_1))^{-1}\boldsymbol{G}(\boldsymbol{\theta}_1)^\mathsf{T}\boldsymbol{R}^{-1}\boldsymbol{y}. \end{aligned} \tag{3.15}$$

Noting that only the second term in (3.15) depends on the nonlinear parameters $\boldsymbol{\theta}_1$, we can conclude that these can be found by minimizing this term, that is, by solving the optimization problem

$$\hat{\boldsymbol{\theta}}_1 = \operatorname*{argmin}_{\boldsymbol{\theta}_1} -\boldsymbol{y}^\mathsf{T}\boldsymbol{R}^{-1}\boldsymbol{G}(\boldsymbol{\theta}_1)(\boldsymbol{G}(\boldsymbol{\theta}_1)^\mathsf{T}\boldsymbol{R}^{-1}\boldsymbol{G}(\boldsymbol{\theta}_1))^{-1}\boldsymbol{G}(\boldsymbol{\theta}_1)^\mathsf{T}\boldsymbol{R}^{-1}\boldsymbol{y}. \tag{3.16}$$

The advantage of this approach is that the numerical optimization method only has to be employed for the lower-dimensional space of $\boldsymbol{\theta}_1$. This reduces the computational complexity and may make the estimation procedure more robust. Once the optimization problem (3.16) is solved, the remaining parameters $\boldsymbol{\theta}_2$ can be calculated in closed form.

Chapter 4

State-Space Models

Most sensor fusion problems involve dynamically changing variables. For example, in autonomous driving, other road users continuously appear, disappear, and move in the traffic scene. In this case, we are interested in estimating dynamically varying parameters. We refer to these parameters as *states* or *the state*, because they describe the state a system is in (e.g., the location and velocity of a car, or the position, attitude, and velocity of a unmanned aerial vehicle, etc.).

In such dynamic scenarios, we have to employ sensors that provide repeated measurements in time, that is, they sample periodically. In between those samples, the state of the system evolves according to some process and thus, the states at different times are different from each other. However, since the evolution of the state follows some dynamic process, they are related and thus, we can relate the states at two times to each other. To do that, we need a principled way of describing how the state evolves between samples. We find a suitable approach in differential equations (for continuous-time processes) and difference equations (for discrete-time processes), which can be used to describe dynamic processes. Indeed, we will make use of differential equations to derive *state-space models* that turn an Lth order differential (or difference) equation into a first order vector-valued differential (or difference) equation.

4.1 Continuous-Time State-Space Models

4.1.1 Deterministic Linear State-Space Models

We start the derivation of the state-space formulation based on an example. Figure 4.1 shows a spring-damper system, a typical mechanical system. This system is governed by Newton's second law of motion, from which we can find the following inhomogeneous ordinary differential equation (ODE):

$$ma(t) = -kp(t) - \eta v(t) + u(t), \tag{4.1}$$

where $p(t)$, $v(t)$, and $a(t)$ denote the position, velocity, and acceleration of the mass m, respectively, k is the spring constant, and η is the damping constant. Furthermore, the forcing function $u(t)$ is a deterministic input to the system.

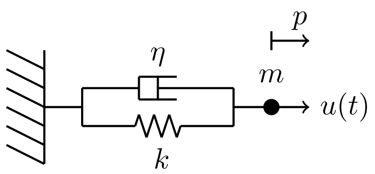


Figure 4.1: Example of a mechanical dynamic system: A spring-damper system with forcing function $u(t)$. The positive direction of the motion p is defined to the right.

By introducing a second equation, $v(t) = v(t)$, which always holds, and dividing (4.1) by m, we obtain the equation system

$$v(t) = v(t), \tag{4.2a}$$

$$a(t) = -\frac{k}{m}p(t) - \frac{\eta}{m}v(t) + \frac{1}{m}u(t). \tag{4.2b}$$

This can be rewritten in matrix form, such that we obtain

$$\begin{bmatrix} v(t) \\ a(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{\eta}{m} & -\frac{k}{m} \end{bmatrix} \begin{bmatrix} p(t) \\ v(t) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u(t). \tag{4.3}$$

Observe that in (4.3), the vector on the left hand side of the equation is the derivative of the vector on the right hand side. Hence, we can define the vector

$$\boldsymbol{x}(t) = \begin{bmatrix} p(t) \\ v(t) \end{bmatrix},$$

and rewrite (4.3) as

$$\dot{\boldsymbol{x}}(t) = \begin{bmatrix} 0 & 1 \\ -\frac{\eta}{m} & -\frac{k}{m} \end{bmatrix} \boldsymbol{x}(t) + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u(t), \tag{4.4}$$

where

$$\dot{\boldsymbol{x}}(t) = \frac{\mathrm{d}\boldsymbol{x}(t)}{\mathrm{d}t}$$

denotes the time-derivative.

The form (4.4) is the state-space representation of the dynamic system in Figure 4.1 and the vector $\boldsymbol{x}(t)$ is called the state vector (or simply state). The state vector now encodes all the information about the state the system is in (position, speed, etc.). Therefore, solving this first-order vector valued ODE representation rather than the original differential equation in 4.1 provides richer information about the system.

We can now generalize this approach. Consider the Lth order ODE of the form

$$\frac{\mathrm{d}^L x(t)}{\mathrm{d}t^L} = a_0 x(t) + a_1 \frac{\mathrm{d}x(t)}{\mathrm{d}t} + \cdots + a_{L-1} \frac{\mathrm{d}^{L-1} x(t)}{\mathrm{d}t^{L-1}} + b_1 u(t). \tag{4.5}$$

Then we can introduce $L-1$ equations of the form $\mathrm{d}^l x(t)/\mathrm{d}t^l = \mathrm{d}^l x(t)/\mathrm{d}t^l$ to obtain the equation system

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = \frac{\mathrm{d}x(t)}{\mathrm{d}t}, \tag{4.6a}$$

$$\frac{\mathrm{d}^2x(t)}{\mathrm{d}t^2} = \frac{\mathrm{d}^2x(t)}{\mathrm{d}t^2}, \tag{4.6b}$$

$$\vdots \tag{4.6c}$$

$$\frac{\mathrm{d}^Lx(t)}{\mathrm{d}t^L} = a_0x(t) + a_1\frac{\mathrm{d}x(t)}{\mathrm{d}t} + \cdots + a_{L-1}\frac{\mathrm{d}^{L-1}x(t)}{\mathrm{d}t^{L-1}} + b_1u(t). \tag{4.6d}$$

Rewriting (4.6) in vector form yields

$$\begin{bmatrix} \frac{\mathrm{d}x(t)}{\mathrm{d}t} \\ \frac{\mathrm{d}^2x(t)}{\mathrm{d}t^2} \\ \vdots \\ \frac{\mathrm{d}^Lx(t)}{\mathrm{d}t^L} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & & \vdots \\ \vdots & & & \ddots & 0 \\ a_0 & a_1 & & \dots & a_{L-1} \end{bmatrix} \begin{bmatrix} x(t) \\ \frac{\mathrm{d}x(t)}{\mathrm{d}t} \\ \vdots \\ \frac{\mathrm{d}^{L-1}x(t)}{\mathrm{d}t^{L-1}} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ b_1 \end{bmatrix} u(t).$$

This in turn, can be written compactly as the dynamic model

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}_u\boldsymbol{u}(t), \tag{4.7}$$

where

$$\boldsymbol{x}(t) = \begin{bmatrix} x(t) \\ \frac{\mathrm{d}x(t)}{\mathrm{d}t} \\ \vdots \\ \frac{\mathrm{d}^{L-1}x(t)}{\mathrm{d}t^{L-1}} \end{bmatrix}, \ \boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & & \vdots \\ \vdots & & & \ddots & 0 \\ a_0 & a_1 & & \dots & a_{L-1} \end{bmatrix}, \ \boldsymbol{B}_u = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ b_1 \end{bmatrix}, \ \boldsymbol{u}(t) = u(t) \tag{4.8}$$

While quite general, eq. (4.7) is a state-space representation for models of the type in (4.5). However, we can also derive the same type of representation for other dynamic models. For example, consider a freshly brewed hot cup of coffee. Newton's law of cooling then states that the change in temperature of the coffee is proportional to the temperature difference between the coffee and its surrounding (assuming that the surroundings are much larger than the coffee cup). This can be formulated as a differential equation of the form

$$\frac{\mathrm{d}T_\mathrm{c}(t)}{\mathrm{d}t} = -k_1(T_\mathrm{c}(t) - T_\mathrm{r}(t)), \tag{4.9}$$

where $T_\mathrm{c}(t)$ denotes the coffee's temperature, $T_\mathrm{r}(t)$ the room temperature, and k_1 is a constant. The change in room temperature on the other hand depends on the heat that is produced inside the room (e.g., due to heating) as well as the losses to the outside. Again assuming that Newton's law of cooling applies, we can write the differential equation as

$$\frac{\mathrm{d}T_\mathrm{r}(t)}{\mathrm{d}t} = -k_2(T_\mathrm{r}(t) - T_\mathrm{a}(t)) + h(t), \tag{4.10}$$

where k_2 is a constant, $T_\mathrm{a}(t)$ denotes the outside temperature and $h(t)$ is the heating input. Observe that (4.9)–(4.10) is a coupled system that can be written as the equation system

$$\begin{aligned}\frac{\mathrm{d}T_\mathrm{r}(t)}{\mathrm{d}t} &= -k_2(T_\mathrm{r}(t) - T_\mathrm{a}(t)) + h(t) &\quad (4.11\mathrm{a})\\ \frac{\mathrm{d}T_\mathrm{c}(t)}{\mathrm{d}t} &= -k_1(T_\mathrm{c}(t) - T_\mathrm{r}(t)), &\quad (4.11\mathrm{b})\end{aligned}$$

or equivalently on matrix form as

$$\begin{bmatrix}\frac{\mathrm{d}T_\mathrm{r}(t)}{\mathrm{d}t}\\ \frac{\mathrm{d}T_\mathrm{c}(t)}{\mathrm{d}t}\end{bmatrix} = \begin{bmatrix}-k_2 & 0\\ k_1 & -k_1\end{bmatrix}\begin{bmatrix}T_\mathrm{r}(t)\\ T_\mathrm{c}(t)\end{bmatrix} + \begin{bmatrix}k_2 & 1\\ 0 & 0\end{bmatrix}\begin{bmatrix}T_\mathrm{a}(t)\\ h(t)\end{bmatrix}. \tag{4.12}$$

This has again the form of (4.7), where

$$\boldsymbol{x}(t) = \begin{bmatrix}T_\mathrm{r}(t)\\ T_\mathrm{c}(t)\end{bmatrix},\ \boldsymbol{A} = \begin{bmatrix}-k_2 & 0\\ k_1 & -k_1\end{bmatrix},\ \boldsymbol{B}_u = \begin{bmatrix}k_2 & 1\\ 0 & 0\end{bmatrix},\ \boldsymbol{u}(t) = \begin{bmatrix}T_\mathrm{a}(t)\\ h(t)\end{bmatrix}. \tag{4.13}$$

Hence, the model (4.7) is a quite general *dynamic model* of the time-varying behavior for many different processes. Recall that our original aim was to estimate the dynamically changing state $\boldsymbol{x}(t)$. This requires that we measure $\boldsymbol{x}(t)$ in some way, that is, we need to combine the dynamic model with a *measurement model.* The simplest case is that of a linear measurement model of the form (2.7) where we replace the static parameters $\boldsymbol{\theta}$ with the state $\boldsymbol{x}(t)$. Since the measurements are obtained at discrete time instances t_n (i.e., at $t_1, t_2, \dots$), the measurement model becomes

$$\boldsymbol{y}_n = \boldsymbol{G}\boldsymbol{x}(t_n) + \boldsymbol{r}_n, \tag{4.14}$$

where $\boldsymbol{r}_n$ denotes the measurement noise with covariance matrix $\mathrm{Cov}\{\boldsymbol{r}_n\} = \boldsymbol{R}_n$.

Defining $\boldsymbol{x}_n \triangleq \boldsymbol{x}(t_n)$ and combining the dynamic model (4.7) and the measurement model (4.14) then yields the deterministic *linear state space model*

$$\begin{aligned}\dot{\boldsymbol{x}}(t) &= \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}_u\boldsymbol{u}(t), &\quad (4.15\mathrm{a})\\ \boldsymbol{y}_n &= \boldsymbol{G}\boldsymbol{x}_n + \boldsymbol{r}_n. &\quad (4.15\mathrm{b})\end{aligned}$$

4.1.2 Stochastic Linear State-Space Models

Often, the behavior of dynamic systems is not completely deterministic, or the input $u(t)$ is not known. For example, when tracking an airplane using radar, we do not know the inputs made by the pilots. Additionally, variables such as wind or pressure fields are not known either and it is difficult to model them accurately. Instead, such random effects can be modeled by including a stochastic process as an input to the differential equation (and hence to its state-space representation), which turns an ODE into a *stochastic differential equation* (SDE; Øksendal (2010); Särkkä and Solin (2018)).

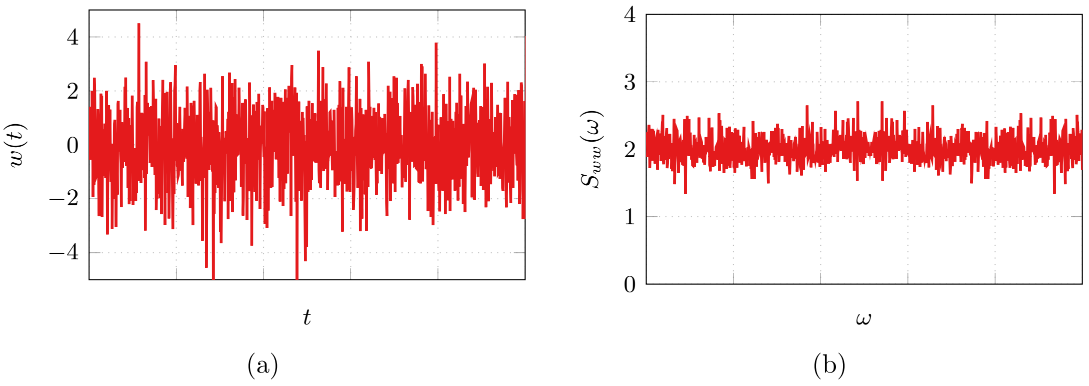


Figure 4.2: Example of a white noise process $w(t)$ with $\sigma_w^2 = 2$. (a) One realization of the process, and (b) its power spectral density $S_w(\omega)$ averaged over 100 realizations.

A stationary stochastic process $w(t)$ can be characterized by its autocorrelation function

$$R_{ww}(\tau) = \mathrm{E}\{w(t+\tau)w(t)\}$$

or, equivalently, the power spectral density (which is the Fourier transform of the autocorrelation function; Papoulis (1984)). A suitable assumption is that the stochastic process is a white noise process. In this case, the autocorrelation function is

$$R_{ww}(\tau) = \sigma_w^2 \delta(\tau), \tag{4.16}$$

where $\delta(\tau)$ denotes the Dirac delta function. Hence, the power spectral density is a constant, that is,

$$S_{ww} = \sigma_w^2. \tag{4.17}$$

In other words, the white noise process contains equal contributions from each frequency. One realization of such a process, together with its power spectral density are shown in Figure 4.2.

The derivation of the state-space representation of the resulting SDE follows the same steps as for the deterministic case, where the random process takes the role of the input. Consider the Lth order SDE given by

$$\frac{\mathrm{d}^L x(t)}{\mathrm{d}t^L} = a_0 x(t) + a_1 \frac{\mathrm{d}x(t)}{\mathrm{d}t} + \cdots + a_{L-1} \frac{\mathrm{d}^{L-1} x(t)}{\mathrm{d}t^{L-1}} + b_1 w(t), \tag{4.18}$$

which is of the same form as (4.5) but with the stochastic process $w(t)$ as the input. Rewriting (4.18) into matrix form yields

$$\begin{bmatrix} \frac{\mathrm{d}x(t)}{\mathrm{d}t} \\ \frac{\mathrm{d}^2 x(t)}{\mathrm{d}t^2} \\ \vdots \\ \frac{\mathrm{d}^L x(t)}{\mathrm{d}t^L} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & & \vdots \\ \vdots & & & \ddots & 0 \\ a_0 & a_1 & & \dots & a_{L-1} \end{bmatrix} \begin{bmatrix} x(t) \\ \frac{\mathrm{d}x(t)}{\mathrm{d}t} \\ \vdots \\ \frac{\mathrm{d}^{L-1} x(t)}{\mathrm{d}t^{L-1}} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ b_1 \end{bmatrix} w(t).$$

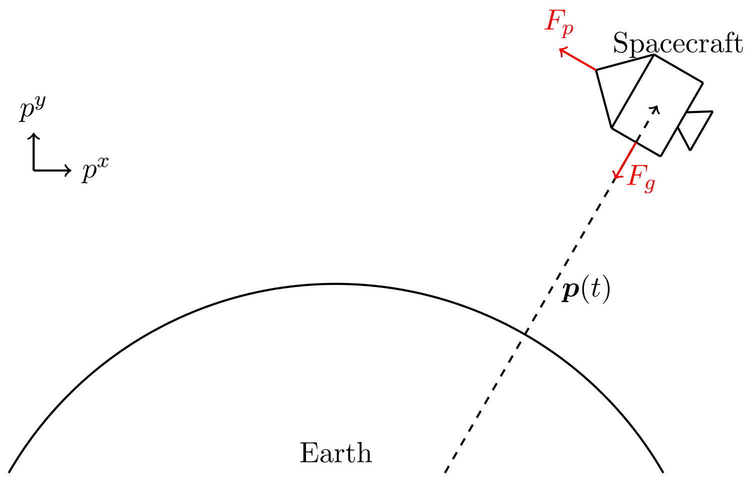


Figure 4.3: Illustration of a spacecraft orbiting the earth.

Hence, this can also be written compactly in the form of a first order vector valued SDE system

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}_w \boldsymbol{w}(t).$$

Naturally, also coupled models (like the coffee-cup example) can be written in this form. Together with a linear measurement equation, the stochastic linear state-space model becomes

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}_w \boldsymbol{w}(t), \tag{4.19a}$$

$$\boldsymbol{y}_n = \boldsymbol{G}\boldsymbol{x}_n + \boldsymbol{r}_n. \tag{4.19b}$$

Note that some dynamic models may contain both deterministic inputs $\boldsymbol{u}(t)$ and stochastic inputs $\boldsymbol{w}(t)$. Naturally, both of these can be incorporated in the model as well.

4.1.3 Nonlinear State-Space Models

So far, we have only discussed the state-space form of linear ODEs and SDEs. However, many dynamic systems actually behave nonlinearly and fortunately, the approach can be extended to nonlinear systems as well.

We start our discussion again based on an example. Figure 4.3 shows an illustration of a spacecraft orbiting the earth. The forces acting upon the craft are the gravitational pull of the earth F_g and the propulsion force F_p. The position $\boldsymbol{p}(t)$ of the craft is defined according to the coordinate system shown in Figure 4.3, with the origin at the center of the earth. The gravitational acceleration of an object at a distance $|\boldsymbol{p}(t)|$ from the earth's center is approximately

$$g \approx g_0 \left(\frac{r_e}{|\boldsymbol{p}(t)|}\right)^2,$$

where $g_0 = 9.81\,\mathrm{m/s^2}$ is the gravitational acceleration on the earth's surface and $r_e = 6371\,\mathrm{km}$ is the mean earth radius. The gravitational pull is in the opposite direction of $\boldsymbol{p}(t)$ and hence, we can write it on vector form as

$$\begin{aligned}\boldsymbol{F}_g &= -mg_0\left(\frac{r_e}{|\boldsymbol{p}(t)|}\right)^2\frac{\boldsymbol{p}(t)}{|\boldsymbol{p}(t)|}\\ &= -mg_0r_e^2\frac{\boldsymbol{p}(t)}{|\boldsymbol{p}(t)|^3}.\end{aligned}$$

The propulsion force is perpendicular to the direction of the position vector $\boldsymbol{p}(t)$. Hence, we can write

$$\boldsymbol{F}_p = F_p\frac{1}{|\boldsymbol{p}(t)|}\begin{bmatrix}-p^y(t)\\ p^x(t)\end{bmatrix}$$

When tracking the spacecraft from the ground, for example using radar, the magnitude of the propulsion force F_p is unknown. But since we know that the engines are only used to make small flight path corrections to conserve fuel, we can model this as a stochastic process, that is, we can assume that $F_p = w(t)$ with some spectral density σ_w^2.

Using Newton's second law it follows that the motion of the spacecraft is governed by the vector-valued differential equation

$$m\boldsymbol{a}(t) = -mg_0r_e^2\frac{\boldsymbol{p}(t)}{|\boldsymbol{p}(t)|^3} + \frac{1}{|\boldsymbol{p}(t)|}\begin{bmatrix}-p^y(t)\\ p^x(t)\end{bmatrix}w(t). \tag{4.20}$$

The highest order of the derivative here is the acceleration on the left hand side. Hence, a suitable state vector includes the position (zeroth derivative) and the speed (first derivative), that is,

$$\boldsymbol{x}(t) = \begin{bmatrix}p^x(t) & p^y(t) & v^x(t) & v^y(t)\end{bmatrix}^{\mathsf{T}}.$$

However, when trying to rewrite (4.20) into the form of a state-space representation (4.19), we notice that this is not possible since the right hand side of (4.20) is not linear in $\boldsymbol{p}(t)$. Nevertheless, we can still write it as a nonlinear equation system in terms of $\boldsymbol{x}(t)$ as

$$\begin{aligned}\begin{bmatrix}v^x(t)\\ v^y(t)\\ a^x(t)\\ a^y(t)\end{bmatrix} &= \begin{bmatrix}v^x(t)\\ v^y(t)\\ -g_0r_e^2\frac{p^x(t)}{|\boldsymbol{p}(t)|^3}\\ -g_0r_e^2\frac{p^y(t)}{|\boldsymbol{p}(t)|^3}\end{bmatrix} + \begin{bmatrix}0\\ 0\\ -\frac{p^y(t)}{m|\boldsymbol{p}(t)|}\\ \frac{p^x}{m|\boldsymbol{p}(t)|}\end{bmatrix}w(t)\\ &= \begin{bmatrix}f_1(\boldsymbol{x}(t))\\ f_2(\boldsymbol{x}(t))\\ f_3(\boldsymbol{x}(t))\\ f_4(\boldsymbol{x}(t))\end{bmatrix} + \begin{bmatrix}0\\ 0\\ -\frac{p^y(t)}{m|\boldsymbol{p}(t)|}\\ \frac{p^x}{m|\boldsymbol{p}(t)|}\end{bmatrix}w(t),\end{aligned} \tag{4.21}$$

where the $f_i(\boldsymbol{x}(t))$ are nonlinear functions of $\boldsymbol{x}(t)$.

This idea can be generalized to any arbitrary nonlinear dynamic system that is described by a nonlinear ODE or SDE, including coupled systems. Given the state vector $\boldsymbol{x}(t) = \begin{bmatrix} x_1(t) & x_2(t) & \dots & x_{d_x}(t) \end{bmatrix}^\mathsf{T}$ of dimension d_x, we can write the ODE/SDE as a vector-valued, nonlinear equation system of the form

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \vdots \\ \dot{x}_{d_x}(t) \end{bmatrix} = \begin{bmatrix} f_1(\boldsymbol{x}(t)) \\ f_2(\boldsymbol{x}(t)) \\ \vdots \\ f_{d_x}(\boldsymbol{x}(t)) \end{bmatrix} + \begin{bmatrix} b_{11}(\boldsymbol{x}(t)) & \dots & b_{1d_w}(\boldsymbol{x}(t)) \\ b_{21}(\boldsymbol{x}(t)) & & \vdots \\ \vdots & \ddots & \\ b_{d_x1}(\boldsymbol{x}(t)) & \dots & b_{d_xd_w}(\boldsymbol{x}(t)) \end{bmatrix} \boldsymbol{w}(t). \tag{4.22}$$

More compactly, this can be written as

$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t)) + \boldsymbol{B}_w(\boldsymbol{x}(t))\boldsymbol{w}(t), \tag{4.23}$$

where $f(\boldsymbol{x}(t))$ is a vector-valued nonlinear function of the state, and $\boldsymbol{B}_w(\boldsymbol{x}(t))$ is a matrix that depends on the state $\boldsymbol{x}(t)$.

In order to estimated the state $\boldsymbol{x}(t)$ based on observations $\boldsymbol{y}_n$, we again need to relate the measurements to the state. This is achieved by combining the dynamic model (4.23) with a measurement model as introduced in Chapters 2–3. The most general model arises if we chose a nonlinear observation model of the form (3.1) where the state $\boldsymbol{x}_n \triangleq \boldsymbol{x}(t_n)$ at time t_n takes the place of the parameters $\boldsymbol{\theta}$. This yields the general *nonlinear state-space model*

$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t)) + \boldsymbol{B}_w(\boldsymbol{x}(t))\boldsymbol{w}(t), \tag{4.24a}$$
$$\boldsymbol{y}_n = g(\boldsymbol{x}_n) + \boldsymbol{r}_n, \tag{4.24b}$$

where $f(\boldsymbol{x}(t))$ is the nonlinear *state transition function*, $\boldsymbol{w}(t)$ is the driving stochastic process, $g(\boldsymbol{x}_n)$ is the measurement model, and $\boldsymbol{r}_n$ is the measurement noise.

4.2 Discrete-Time State-Space Models

4.2.1 Deterministic Linear State-Space Models

As discussed in the previous section, ODEs and SDEs are a natural way for modeling the behavior of dynamic systems. Transforming them to state-space form furthermore provides a way of describing how the state of a system evolves over time.

However, not all dynamic systems can be modeled by using differential equations. For some systems, the state is only defined at some discrete points in time $t_1, t_2, \dots$, that is, only in *discrete-time*. In this case, another approach than differential equations is needed to model the dynamic behavior. The discrete-time equivalent to differential equations are *difference equations* that describe the relationship between the value of a variable at time t_n to its previous values at times $t_{n-1}, t_{n-2}, \dots$.

The process of obtaining a state-space representation from differential equations is closely related to the approach used for differential equations. Consider a d_xth order equation system with discrete-time, deterministic inputs $u_{1,n} \triangleq u_1(t_n), u_{2,n} \triangleq u_2(t_n), \ldots, u_{d_u,n} \triangleq u_{d_u}(t_n)$ of the form

$$\begin{aligned}
x_{1,n} &= a_{11}x_{1,n-1} + \cdots + a_{1d_x}x_{d_x,n-1} + b_{11}u_{1,n} + \cdots + b_{1d_u}u_{d_u,n} \\
x_{2,n} &= a_{21}x_{1,n-1} + \cdots + a_{2d_x}x_{d_x,n-1} + b_{21}u_{1,n} + \cdots + b_{2d_u}u_{d_u,n} \\
&\vdots \\
x_{d_x,n} &= a_{d_x1}x_{1,n-1} + \cdots + a_{d_xd_x}x_{d_x,n-1} + b_{d_x1}u_{1,n} + \cdots + b_{d_xd_u}u_{d_u,n}
\end{aligned}$$

where $x_{i,n} \triangleq x_i(t_n)$ is the ith variable at time t_n. This can be rewritten in matrix form according to

$$\begin{bmatrix} x_{1,n} \\ \vdots \\ x_{d_x,n} \end{bmatrix} = \begin{bmatrix} a_{11} & \ldots & a_{1d_x} \\ \vdots & \ddots & \vdots \\ a_{d_x1} & \ldots & a_{d_xd_x} \end{bmatrix} \begin{bmatrix} x_{1,n-1} \\ \vdots \\ x_{d_x,n-1} \end{bmatrix} + \begin{bmatrix} b_{11} & \ldots & b_{1d_u} \\ \vdots & \ddots & \vdots \\ b_{d_x1} & \ldots & b_{d_xd_u} \end{bmatrix} \begin{bmatrix} u_{1,n} \\ \vdots \\ u_{d_x,n} \end{bmatrix}.$$

More compactly, this can also be written as

$$\boldsymbol{x}_n = \boldsymbol{F}\boldsymbol{x}_{n-1} + \boldsymbol{B}_u\boldsymbol{u}_n, \tag{4.25}$$

with

$$\boldsymbol{x}_n = \begin{bmatrix} x_{1,n} \\ \vdots \\ x_{d_x,n} \end{bmatrix}, \; \boldsymbol{F} = \begin{bmatrix} a_{11} & \ldots & a_{1d_x} \\ \vdots & \ddots & \vdots \\ a_{d_x1} & \ldots & a_{d_xd_x} \end{bmatrix}, \; \boldsymbol{B}_u = \begin{bmatrix} b_{11} & \ldots & b_{1d_u} \\ \vdots & \ddots & \vdots \\ b_{d_x1} & \ldots & b_{d_xd_u} \end{bmatrix}, \; \boldsymbol{u}_n = \begin{bmatrix} u_{1,n} \\ \vdots \\ u_{d_u,n} \end{bmatrix}.$$

Equation (4.25) is the general dynamic model for linear discrete-time systems with deterministic inputs $\boldsymbol{u}_n$.

To convert the Lth order difference equation (with single input u_n)

$$z_n = c_1z_{n-1} + c_2z_{n-2} + \cdots + c_Lz_{n-L} + d_1u_n \tag{4.26}$$

to the form (4.25), we proceed as follows. First, we have to choose the state variables $\boldsymbol{x}_n$. This is easiest done at time $n-1$ and for the model (4.26), we can choose

$$x_{1,n-1} = z_{n-1}, \; x_{2,n-1} = z_{n-2}, \; \ldots, \; x_{d_x,n-1} = z_{n-L}.$$

Then we obtain that

$$\begin{aligned}
x_{1,n} &= z_n = c_1z_{n-1} + c_2z_{n-2} + \cdots + c_Lz_{n-L} + d_1u_n \\
&= c_1x_{1,n-1} + c_2x_{2,n-1} + \cdots + c_Lx_{d_x,n-1} + d_1u_n, \\
x_{2,n} &= z_{n-1} \\
&= x_{1,n-1}, \\
&\vdots \\
x_{d_x,n} &= z_{n-L+1} \\
&= x_{d_x+1,n-1},
\end{aligned}$$

where we have expressed the state at time n solely in terms of the state at previous times as well as the input u_n. Rewriting this on matrix form finally yields

$$\begin{bmatrix} x_{1,n} \\ x_{2,n} \\ \vdots \\ x_{d_x,n} \end{bmatrix} = \begin{bmatrix} c_1 & c_2 & \dots & c_L & \\ 1 & 0 & & \vdots & \\ \vdots & \ddots & & & \\ 0 & \dots & 1 & 0 & \end{bmatrix} \begin{bmatrix} x_{1,n-1} \\ x_{2,n-1} \\ \vdots \\ x_{d_x,n-1} \end{bmatrix} + \begin{bmatrix} d_1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} u_n,$$

which is of the form (4.25).

Equation (4.25) only models the dynamic behavior of the state. As for the continuous case, only noisy measurements of the dynamic process are available for estimation. Combining the dynamic model with a measurement model thus again yields the complete discrete-time state-space model. Assuming a linear measurement model of the form (2.7) then yields the linear discrete-time model

$$\boldsymbol{x}_n = \boldsymbol{F}\boldsymbol{x}_{n-1} + \boldsymbol{B}_u\boldsymbol{u}_n, \tag{4.27a}$$

$$\boldsymbol{y}_n = \boldsymbol{G}\boldsymbol{x}_n + \boldsymbol{r}_n. \tag{4.27b}$$

4.2.2 Stochastic Linear Dynamic Models

As for the continuous case, the deterministic discrete-time dynamic model can not take into account random effects, uncertainty, and unknown inputs. To account for such effects, we need an approach that is similar to the random process input in the continuous case. For the discrete-time case, we can model this using a *random variable* as the input to the dynamic model (compared to a stochastic process for the continuous case). We denote this random input, commonly called the *process noise*, as $\boldsymbol{q}_n$. Then, replacing the deterministic input $\boldsymbol{u}_n$ with $\boldsymbol{q}_n$ yields the stochastic discrete-time dynamic model

$$\boldsymbol{x}_n = \boldsymbol{F}\boldsymbol{x}_{n-1} + \boldsymbol{B}_q\boldsymbol{q}_n. \tag{4.28}$$

The random variable $\boldsymbol{q}_n$ follows some probability density function such that

$$\boldsymbol{q}_n \sim p(\boldsymbol{q}_n).$$

Here, we assume that $\boldsymbol{q}_n$ is zero-mean, that is $\mathrm{E}\{\boldsymbol{q}_n\} = 0$, and that it has covariance $\mathrm{Cov}\{\boldsymbol{q}_n\} = \boldsymbol{Q}_n$. Furthermore, we also assume that the cross-covariance between two random inputs $\boldsymbol{q}_m$ and $\boldsymbol{q}_n$ is $\mathrm{Cov}\{\boldsymbol{q}_m, \boldsymbol{q}_n\} = 0$ for $m \neq n$.

Together with a linear sensor model, we then obtain the stochastic linear discrete-time state-space model given by

$$\boldsymbol{x}_n = \boldsymbol{F}\boldsymbol{x}_{n-1} + \boldsymbol{B}_q\boldsymbol{q}_n, \tag{4.29a}$$

$$\boldsymbol{y}_n = \boldsymbol{G}\boldsymbol{x}_n + \boldsymbol{r}_n. \tag{4.29b}$$

4.2.3 Nonlinear Dynamic Model

Similar to differential equations, difference equations may be nonlinear. In this case, we start from the nonlinear equation system with the random variables $q_{1,n}, q_{2,n}, \dots, q_{d_q,n}$ as the inputs given by

$$\begin{aligned}
x_{1,n} &= f_1(x_{1,n-1}, x_{2,n-1}, \dots, x_{d_x,n-1}) + b_{11}q_{1,n} + \dots + b_{1d_q}q_{d_q,n}, \\
x_{2,n} &= f_2(x_{1,n-1}, x_{2,n-1}, \dots, x_{d_x,n-1}) + b_{21}q_{1,n} + \dots + b_{2d_q}q_{d_q,n}, \\
&\vdots \\
x_{d_x,n} &= f_{d_x}(x_{1,n-1}, x_{2,n-1}, \dots, x_{d_x,n-1}) + b_{d_x1}q_{1,n} + \dots + b_{d_xd_q}q_{d_q,n}.
\end{aligned}$$

This can directly be rewritten into vector form, which yields

$$\begin{bmatrix} x_{1,n} \\ \vdots \\ x_{d_x,n} \end{bmatrix} = \begin{bmatrix} f_1(x_{1,n-1}, x_{2,n-1}, \dots, x_{d_x,n-1}) \\ \vdots \\ f_{d_x}(x_{1,n-1}, x_{2,n-1}, \dots, x_{d_x,n-1}) \end{bmatrix} + \begin{bmatrix} b_{11} & \dots & b_{1d_u} \\ \vdots & \ddots & \vdots \\ b_{d_x1} & \dots & b_{d_xd_u} \end{bmatrix} \begin{bmatrix} q_{1,n} \\ \vdots \\ q_{d_u,n} \end{bmatrix},$$

or more compactly

$$\boldsymbol{x}_n = f(\boldsymbol{x}_{n-1}) + \boldsymbol{B}_q\boldsymbol{q}_n. \tag{4.30}$$

The nonlinear dynamic model (4.30) together with the general nonlinear measurement model (3.1) yields the nonlinear discrete-time state-space model

$$\boldsymbol{x}_n = f(\boldsymbol{x}_{n-1}) + \boldsymbol{B}_q\boldsymbol{q}_n, \tag{4.31a}$$
$$\boldsymbol{y}_n = g(\boldsymbol{x}_n) + \boldsymbol{r}_n, \tag{4.31b}$$

where $\boldsymbol{q}_n \sim p(\boldsymbol{q}_n)$, $\mathrm{E}\{\boldsymbol{q}_n\} = 0$, $\mathrm{Cov}\{\boldsymbol{q}_n\} = \boldsymbol{Q}_n$, and $\boldsymbol{r}_n \sim p(\boldsymbol{r}_n)$, $\mathrm{E}\{\boldsymbol{r}_n\} = 0$, $\mathrm{Cov}\{\boldsymbol{r}_n\} = \boldsymbol{R}_n$.

4.3 Discretization of Linear Dynamic Models

In practice, modern sensor fusion systems are implemented in a digital computer. Hence, dealing with continuous dynamic models, that are defined on continuous t is not possible. Instead, we have to discretize the continuous-time model to obtain an (approximately) equivalent discrete-time dynamic model. In other words, we have to solve the differential equation on the interval between times t_{n-1} and t_n, that is, we have to integrate the differential equation. For linear dynamic models, this can be achieved exactly and we develop the necessary tools in this section. For most nonlinear models, exact discretization is not possible and we have to resort to approximate methods. This is the subject of Section 4.4.

4.3.1 Deterministic Linear Dynamic Models

Scalar Model. To derive the necessary expressions for discretizing the linear dynamic model, we first review how to solve a first order scalar case. Consider the non-homogeneous first order ODE

$$\dot{x}(t) = ax(t) + bu(t), \tag{4.32}$$

which we want to solve on the interval between the two sampling points t_n and t_{n-1}. To do this, we introduce the *integrating factor* e^{-at} and note that it holds that

$$\frac{\mathrm{d}}{\mathrm{d}t}e^{-at}x(t) = e^{-at}\dot{x}(t) - e^{-at}ax(t). \tag{4.33}$$

Rearranging (4.32) by moving the term $ax(t)$ to the left hand side and multiplying by the integrating factor yields

$$e^{-at}\dot{x}(t) - e^{-at}ax(t) = e^{-at}bu(t)$$

and thus, using (4.33), we have that

$$\frac{\mathrm{d}}{\mathrm{d}t}e^{-at}x(t) = e^{-at}bu(t). \tag{4.34}$$

Equation (4.34) is separable in t. Hence, the ODE can directly be integrated from t_{n-1} to t_n according to

$$\int_{t_{n-1}}^{t_n} \mathrm{d}\left[e^{-at}x(t)\right] = \int_{t_{n-1}}^{t_n} e^{-at}bu(t)\mathrm{d}t,$$

which yields

$$\begin{aligned}\left[e^{-at}x(t)\right]_{t=t_{n-1}}^{t_n} &= \int_{t_{n-1}}^{t_n} e^{-at}bu(t)\mathrm{d}t,\\ e^{-at_n}x(t_n) - e^{-at_{n-1}}x(t_{n-1}) &= \int_{t_{n-1}}^{t_n} e^{-at}bu(t)\mathrm{d}t.\end{aligned}$$

Now we can solve for $x(t_n)$ by rearranging and multiplying by the factor e^{at_n}, which yields

$$\begin{aligned}x(t_n) &= e^{at_n - at_{n-1}}x(t_{n-1}) + e^{at_n}\int_{t_{n-1}}^{t_n} e^{-at}bu(t)\mathrm{d}t,\\ &= e^{a(t_n - t_{n-1})}x(t_{n-1}) + \int_{t_{n-1}}^{t_n} e^{a(t_n - t)}bu(t)\mathrm{d}t.\end{aligned}$$

Finally, letting $\Delta t \triangleq t_n - t_{n-1}$ and using the notation $x_n \triangleq x(t_n)$, we obtain

$$x_n = e^{a\Delta t}x_{n-1} + \int_{t_{n-1}}^{t_n} e^{a(t_n - t)}bu(t)\mathrm{d}t. \tag{4.35}$$

Note that the second term on the right hand side of (4.35) is the convolution between the factor $e^{a(t_n - t)}$ and the input $u(t)$ on the interval between t_{n-1} and t_n.

General Case. Based on the steps used for solving the scalar ODE (4.32), the general case can now be solved. First, recall that the general dynamic model on state-space form was given by the vector valued first order ODE system (4.15), that is,

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}_u\boldsymbol{u}(t). \tag{4.36}$$

The key here is to note that (4.36) is a first order vector ODE just like (4.32). Hence, we should be able to solve it using an integrating factor similar to the one for the scalar case. Indeed, such an integrating factor can be found through the *matrix exponential.* The matrix exponential for a square matrix $\boldsymbol{A}$ can be defined in the same way as the regular exponential, that is, as an infinite sum of the form

$$e^{\boldsymbol{A}} = \sum_{k=0}^{\infty} \frac{1}{k!}\boldsymbol{A}^k.$$

Similar to the (scalar) exponential, it holds that the derivative with respect to a scalar x of $e^{\boldsymbol{A}x}$ is

$$\frac{\mathrm{d}}{\mathrm{d}x}e^{\boldsymbol{A}x} = e^{\boldsymbol{A}x}\boldsymbol{A}. \tag{4.37}$$

Another useful property of the matrix exponential is that of its transpose, which is given by

$$(e^{\boldsymbol{A}})^{\mathsf{T}} = e^{\boldsymbol{A}^{\mathsf{T}}}. \tag{4.38}$$

With this in mind, we can solve the general case following the same steps as for the scalar case. First, we multiply (4.36) by the integrating factor $e^{-\boldsymbol{A}t}$ and rearrange it to obtain

$$e^{-\boldsymbol{A}t}\dot{\boldsymbol{x}}(t) - e^{-\boldsymbol{A}t}\boldsymbol{A}\boldsymbol{x}(t) = e^{-\boldsymbol{A}t}\boldsymbol{B}_u\boldsymbol{u}(t)$$

Next, similar to (4.33) (product rule), the left hand side is

$$\frac{\mathrm{d}}{\mathrm{d}t}e^{-\boldsymbol{A}t}\boldsymbol{x}(t) = e^{-\boldsymbol{A}t}\dot{\boldsymbol{x}}(t) - e^{-\boldsymbol{A}}\boldsymbol{A}\boldsymbol{x}(t),$$

and thus, we have that

$$\frac{\mathrm{d}}{\mathrm{d}t}e^{-\boldsymbol{A}t}\boldsymbol{x}(t) = e^{-\boldsymbol{A}t}\boldsymbol{B}_u\boldsymbol{u}(t).$$

This is again separable in t. Direct integration from t_{n-1} to t_n then leads to

$$\begin{aligned}
\int_{t_{n-1}}^{t_n} \mathrm{d}\left[e^{-\boldsymbol{A}t}\boldsymbol{x}(t)\right] &= \int_{t_{n-1}}^{t_n} e^{-\boldsymbol{A}t}\boldsymbol{B}_u\boldsymbol{u}(t)\mathrm{d}t. \\
\left[e^{-\boldsymbol{A}t}\boldsymbol{x}(t)\right]_{t=t_{n-1}}^{t_n} &= \int_{t_{n-1}}^{t_n} e^{-\boldsymbol{A}t}\boldsymbol{B}_u\boldsymbol{u}(t)\mathrm{d}t. \\
e^{-\boldsymbol{A}t_n}\boldsymbol{x}(t_n) - e^{-\boldsymbol{A}t_{n-1}}\boldsymbol{x}(t_{n-1}) &= \int_{t_{n-1}}^{t_n} e^{-\boldsymbol{A}t}\boldsymbol{B}_u\boldsymbol{u}(t)\mathrm{d}t.
\end{aligned}$$

Solving for $\boldsymbol{x}(t_n)$ by rearranging and multiplying both sides by $e^{\boldsymbol{A}t_n}$ yields

$$\boldsymbol{x}(t_n) = e^{\boldsymbol{A}(t_n - t_{n-1})}\boldsymbol{x}(t_{n-1}) + \int_{t_{n-1}}^{t_n} e^{\boldsymbol{A}(t_n - t)}\boldsymbol{B}_u\boldsymbol{u}(t)\mathrm{d}t,$$

or

$$\boldsymbol{x}_n = e^{\boldsymbol{A}\Delta t}\boldsymbol{x}_{n-1} + \int_{t_{n-1}}^{t_n} e^{\boldsymbol{A}(t_n - t)}\boldsymbol{B}_u\boldsymbol{u}(t)\mathrm{d}t. \tag{4.39}$$

In this case, the factor in front of x_{n-1} (i.e., the state at time t_{n-1}) is a matrix exponential that again depends on the *time difference* Δt between the two discrete times t_{n-1} and t_n. Note that Δt can be arbitrary, meaning that there is no need for uniform sampling between points t_{n-2}, t_{n-1}, t_n, and so forth. Furthermore, the second term in (4.39) is again the convolution between the deterministic input $u(t)$ and the matrix exponential on the same interval.

If the input $\boldsymbol{u}(t)$ is a zeroth-order-hold (ZOH) signal, which is common in control or robot navigation, the second term can further be simplified. In that case, $\boldsymbol{u}(t)$ is constant on the interval between t_{n-1} and t_n and we can denote it as $\boldsymbol{u}_{n-1}$. Thus, we then have that

$$\begin{aligned}\int_{t_{n-1}}^{t_n} e^{\boldsymbol{A}(t_n - t)}\boldsymbol{B}_u\boldsymbol{u}(t)\mathrm{d}t, &= \int_{t_{n-1}}^{t_n} e^{\boldsymbol{A}(t_n - t)}\boldsymbol{B}_u\boldsymbol{u}_{n-1}\mathrm{d}t, \\ &= \int_{t_{n-1}}^{t_n} e^{\boldsymbol{A}(t_n - t)}\boldsymbol{B}_u\mathrm{d}t\boldsymbol{u}_{n-1},\end{aligned}$$

where $\boldsymbol{B}_u$ may or may not depend on t.

Defining

$$\boldsymbol{F}_n \triangleq e^{\boldsymbol{A}(t_n - t_{n-1})}, \tag{4.40a}$$

$$\boldsymbol{L}_n \triangleq \int_{t_{n-1}}^{t_n} e^{\boldsymbol{A}(t_n - t)}\boldsymbol{B}_u\mathrm{d}t, \tag{4.40b}$$

we can rewrite (4.39) as

$$\boldsymbol{x}_n = \boldsymbol{F}_n\boldsymbol{x}_{n-1} + \boldsymbol{L}_n\boldsymbol{u}_{n-1}. \tag{4.41}$$

Equation (4.41) is an exact solution to (4.36) and thus it is an exact discretization and equivalent representation of the continuous-time model. Furthermore, it also has the same form as the discrete-time state-space model (4.27).

4.3.2 Stochastic Linear Dynamic Models

Discretization of the stochastic linear dynamic model in (4.19) and given by

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}_w\boldsymbol{w}(t), \tag{4.42}$$

with $R_{ww}(\tau) = \mathrm{E}\{\boldsymbol{w}(t+\tau)\boldsymbol{w}(t)\} = \boldsymbol{\Sigma}_w\delta(\tau)$, follows the same steps as the discretization of the deterministic model. This yields

$$\boldsymbol{x}(t_n) = e^{\boldsymbol{A}(t_n - t_{n-1})}\boldsymbol{x}(t_{n-1}) + \int_{t_{n-1}}^{t_n} e^{\boldsymbol{A}(t_n - t)}\boldsymbol{B}_w\boldsymbol{w}(t)\mathrm{d}t.$$

Now note that special attention has to be paid to the second term on the right hand side. The noise process $\boldsymbol{w}(t)$ does not have the properties required for regular integration rules to apply (Øksendal, 2010; Särkkä and Solin, 2018).

However, we can define a random variable $\boldsymbol{q}_n$ as

$$\boldsymbol{q}_n \triangleq \int_{t_{n-1}}^{t_n} e^{\boldsymbol{A}(t_n - t)}\boldsymbol{B}_w\boldsymbol{w}(t)\mathrm{d}t.$$

Next, we can calculate the properties of $\boldsymbol{q}_n$. The mean is given by

$$\begin{aligned}
\mathrm{E}\{\boldsymbol{q}_n\} &= \mathrm{E}\left\{\int_{t_{n-1}}^{t_n} e^{\boldsymbol{A}(t_n - t)}\boldsymbol{B}_w\boldsymbol{w}(t)\mathrm{d}t\right\} \\
&= \int_{t_{n-1}}^{t_n} \mathrm{E}\left\{e^{\boldsymbol{A}(t_n - t)}\boldsymbol{B}_w\boldsymbol{w}(t)\right\}\mathrm{d}t \\
&= \int_{t_{n-1}}^{t_n} e^{\boldsymbol{A}(t_n - t)}\boldsymbol{B}_w\,\mathrm{E}\left\{\boldsymbol{w}(t)\right\}\mathrm{d}t \\
&= 0.
\end{aligned}$$

Similarly, we can calculate the covariance matrix

$$\begin{aligned}
\mathrm{Cov}\{\boldsymbol{q}_n\} &= \mathrm{E}\{(\boldsymbol{q}_n - \mathrm{E}\{\boldsymbol{q}_n\})(\boldsymbol{q}_n - \mathrm{E}\{\boldsymbol{q}_n\})^\mathsf{T}\} \\
&= \mathrm{E}\{\boldsymbol{q}_n\boldsymbol{q}_n^\mathsf{T}\} \\
&= \mathrm{E}\left\{\left(\int_{t_{n-1}}^{t_n} e^{\boldsymbol{A}(t_n - t)}\boldsymbol{B}_w\boldsymbol{w}(t)\mathrm{d}t\right)\left(\int_{t_{n-1}}^{t_n} e^{\boldsymbol{A}(t_n - \tau)}\boldsymbol{B}_w\boldsymbol{w}(\tau)\mathrm{d}\tau\right)^\mathsf{T}\right\} \\
&= \int_{t_{n-1}}^{t_n}\int_{t_{n-1}}^{t_n} e^{\boldsymbol{A}(t_n - t)}\boldsymbol{B}_w\,\mathrm{E}\left\{\boldsymbol{w}(t)\boldsymbol{w}(\tau)^\mathsf{T}\right\}\boldsymbol{B}_w^\mathsf{T}(e^{\boldsymbol{A}(t_n - \tau)})^\mathsf{T}\mathrm{d}\tau\mathrm{d}t \\
&= \int_{t_{n-1}}^{t_n}\int_{t_{n-1}}^{t_n} e^{\boldsymbol{A}(t_n - t)}\boldsymbol{B}_w R_{ww}(t-\tau)\boldsymbol{B}_w^\mathsf{T}(e^{\boldsymbol{A}(t_n - \tau)})^\mathsf{T}\mathrm{d}\tau\mathrm{d}t \\
&= \int_{t_{n-1}}^{t_n}\int_{t_{n-1}}^{t_n} e^{\boldsymbol{A}(t_n - t)}\boldsymbol{B}_w\boldsymbol{\Sigma}_w\delta(t-\tau)\boldsymbol{B}_w^\mathsf{T}(e^{\boldsymbol{A}(t_n - \tau)})^\mathsf{T}\mathrm{d}\tau\mathrm{d}t \\
&= \int_{t_{n-1}}^{t_n} e^{\boldsymbol{A}(t_n - \tau)}\boldsymbol{B}_w\boldsymbol{\Sigma}_w\boldsymbol{B}_w^\mathsf{T}e^{\boldsymbol{A}^\mathsf{T}(t_n - \tau)}\mathrm{d}\tau \\
&\triangleq \boldsymbol{Q}_n.
\end{aligned}$$

Hence, the mean and covariance of the random variable $\boldsymbol{q}_n$ given that $\boldsymbol{w}(t)$ is a zero-mean white noise process with autocorrelation function $R_{ww}(\tau) = \boldsymbol{\Sigma}_w \delta(\tau)$ are given by

$$\mathrm{E}\{\boldsymbol{q}_n\} = 0, \tag{4.43a}$$

$$\mathrm{Cov}\{\boldsymbol{q}_n\} = \int_{t_{n-1}}^{t_n} e^{\boldsymbol{A}(t_n-\tau)} \boldsymbol{B}_w \boldsymbol{\Sigma}_w \boldsymbol{B}_w^\mathsf{T} e^{\boldsymbol{A}^\mathsf{T}(t_n-\tau)} \mathrm{d}\tau \triangleq \boldsymbol{Q}_n. \tag{4.43b}$$

Furthermore, it turns out that in this case, the process noise $\boldsymbol{q}_n$ is actually a Gaussian random variable, that is, it holds that

$$\boldsymbol{q}_n \sim \mathcal{N}(0, \boldsymbol{Q}_n). \tag{4.44}$$

In summary, the discretized stochastic dynamic model (4.42) is

$$\boldsymbol{x}_n = \boldsymbol{F}_n \boldsymbol{x}_{n-1} + \boldsymbol{q}_n \tag{4.45}$$

where

$$\boldsymbol{F}_n \triangleq e^{\boldsymbol{A}(t_n - t_{n-1})}, \tag{4.46a}$$

$$\boldsymbol{q}_n \sim \mathcal{N}(0, \boldsymbol{Q}_n), \tag{4.46b}$$

with $\boldsymbol{Q}_n$ as in (4.43). Again, the discretized model (4.45)–(4.46) is an exact discretization of the continuous-time model (4.42) and thus completely equivalent.

4.4 Discretization of Nonlinear Dynamic Models

Unlike the linear models discussed in Section 4.3, discretization of nonlinear dynamic models is generally harder. In particular, closed-form solutions to the discretization problem can only be found in a few special cases. In most other cases, approximations have to be used. In this section three approaches for approximate discretization are discussed.

4.4.1 Discretization of Linearized Nonlinear Models

The first approach is based on a Taylor series expansion of the nonlinear function. Truncating the Taylor series expansion of the nonlinear function $f(\boldsymbol{x}(t))$ around the state $\boldsymbol{x}_{n-1}$ at the linear term yields the following approximation of the (deterministic) nonlinear dynamic model:

$$\dot{\boldsymbol{x}}(t) \approx f(\boldsymbol{x}_{n-1}) + \boldsymbol{A}_x(\boldsymbol{x}(t) - \boldsymbol{x}_{n-1}) + \boldsymbol{B}_u \boldsymbol{u}(t),$$

where $\boldsymbol{A}_x$ is the Jacobian of $f(\boldsymbol{x}(t))$ with respect to $\boldsymbol{x}(t)$, evaluated at $\boldsymbol{x}(t) = \boldsymbol{x}_{n-1}$. Rearranging the terms on the right hand side yields

$$\dot{\boldsymbol{x}}(t) \approx \boldsymbol{A}_x \boldsymbol{x}(t) + f(\boldsymbol{x}_{n-1}) - \boldsymbol{A}_x \boldsymbol{x}_{n-1} + \boldsymbol{B}_u \boldsymbol{u}(t),$$

which is linear in $\boldsymbol{x}(t)$, and $f(\boldsymbol{x}_{n-1})$ and $\boldsymbol{A}_x x_{n-1}$ are constants. Thus, we can use the same approach as for linear models in Section 4.3, that is, we can use the integrating factor $e^{-\boldsymbol{A}_x t}$. This yields

$$\begin{aligned}\boldsymbol{x}_n \approx e^{\boldsymbol{A}_x \Delta t}\boldsymbol{x}_{n-1} + \int_{t_{n-1}}^{t_n} e^{\boldsymbol{A}_x(t_n - t)}\mathrm{d}t f(\boldsymbol{x}_{n-1}) - \int_{t_{n-1}}^{t_n} e^{\boldsymbol{A}_x(t_n - t)}\mathrm{d}t \boldsymbol{A}_x \boldsymbol{x}_{n-1} \\ + \int_{t_{n-1}}^{t_n} e^{\boldsymbol{A}_x(t_n - t)}\boldsymbol{B}_u \boldsymbol{u}(t)\mathrm{d}t.\end{aligned}$$

Noting that the third term is the integral of the derivative of the matrix exponential with respect to t (see (4.37)), we have that

$$\begin{aligned}\boldsymbol{x}_n &\approx e^{\boldsymbol{A}_x \Delta t}\boldsymbol{x}_{n-1} + \int_{t_{n-1}}^{t_n} e^{\boldsymbol{A}_x(t_n - t)}\mathrm{d}t f(\boldsymbol{x}_{n-1}) + \left[e^{\boldsymbol{A}_x(t_n - t)}\right]_{t=t_{n-1}}^{t_n} \boldsymbol{x}_{n-1} \\ &\quad + \int_{t_{n-1}}^{t_n} e^{\boldsymbol{A}_x(t_n - t)}\boldsymbol{B}_u \boldsymbol{u}(t)\mathrm{d}t \\ &= e^{\boldsymbol{A}_x \Delta t}\boldsymbol{x}_{n-1} + \int_{t_{n-1}}^{t_n} e^{\boldsymbol{A}_x(t_n - t)}\mathrm{d}t f(\boldsymbol{x}_{n-1}) + (\boldsymbol{I} - e^{\boldsymbol{A}_x(t_n - t_{n-1})})\boldsymbol{x}_{n-1} \\ &\quad + \int_{t_{n-1}}^{t_n} e^{\boldsymbol{A}_x(t_n - t)}\boldsymbol{B}_u \boldsymbol{u}(t)\mathrm{d}t,\end{aligned}$$

and finally,

$$\boldsymbol{x}_n \approx \boldsymbol{x}_{n-1} + \int_{t_{n-1}}^{t_n} e^{\boldsymbol{A}_x(t_n - t)}\mathrm{d}t f(\boldsymbol{x}_{n-1}) + \int_{t_{n-1}}^{t_n} e^{\boldsymbol{A}_x(t_n - t)}\boldsymbol{B}_u \boldsymbol{u}(t)\mathrm{d}t. \tag{4.47}$$

As usual, the drawback of this Taylor-series-based solution is that the linearization is only local and large Δts as well as highly nonlinear functions may be problematic. On the other hand, as it can be seen from (4.47), the approximation only affects the deterministic term of the model, whereas the input is calculated in the same way as for the linear model (but with the Jacobian $\boldsymbol{A}_x$). Hence, for the stochastic dynamic model, the result immediately follows from (4.47) to be

$$\boldsymbol{x}_n \approx \boldsymbol{x}_{n-1} + \int_{t_{n-1}}^{t_n} e^{\boldsymbol{A}_x(t_n - t)}\mathrm{d}t f(\boldsymbol{x}_{n-1}) + \boldsymbol{q}_n, \tag{4.48}$$

with

$$\begin{aligned}\boldsymbol{q}_n &\sim \mathcal{N}(0, \boldsymbol{Q}_n), \\ \boldsymbol{Q}_n &\approx \int_{t_{n-1}}^{t_n} e^{\boldsymbol{A}_x(t_n - \tau)}\boldsymbol{B}_w \boldsymbol{\Sigma}_w \boldsymbol{B}_w^{\mathsf{T}} e^{\boldsymbol{A}_x^{\mathsf{T}}(t_n - \tau)}\mathrm{d}\tau.\end{aligned}$$

4.4.2 Euler Discretization of Deterministic Nonlinear Models

The second approach is based on the so-called Euler approximation. We know that the solution for $\boldsymbol{x}_n$ can be written as

$$\boldsymbol{x}_n = \boldsymbol{x}_{n-1} + \int_{t_{n-1}}^{t_n} f(\boldsymbol{x}(t))\mathrm{d}t + \int_{t_{n-1}}^{t_n} \boldsymbol{B}_u \boldsymbol{u}(t)\mathrm{d}t,$$

and the problem is to calculate the integrals in the second and third terms on the right hand side. However, for sufficiently small $\Delta t = t_n - t_{n-1}$, we can approximate the integral by using the rectangle approximation. This means that we can approximate the integral as

$$\begin{aligned}\int_{t_{n-1}}^{t_n} f(\boldsymbol{x}(t))\mathrm{d}t &\approx f(\boldsymbol{x}_{n-1})(t_n - t_{n-1}) \\ &= \Delta t f(\boldsymbol{x}_{n-1}),\end{aligned}$$

and similar for the second integral with the input $\boldsymbol{u}(t)$.

This then yields the Euler discretization of the deterministic, nonlinear dynamic model given by

$$\boldsymbol{x}_n \approx \boldsymbol{x}_{n-1} + \Delta t f(\boldsymbol{x}_{n-1}) + \Delta t \boldsymbol{B}_u \boldsymbol{u}_{n-1}. \tag{4.49}$$

4.4.3 Euler–Maruyama Discretization of Stochastic Nonlinear Models

The Euler discretization is quite simple to implement. However, it does not readily work for stochastic dynamic models. In this case, we are interested in solving the integral equation

$$\boldsymbol{x}_n = \boldsymbol{x}_{n-1} + \int_{t_{n-1}}^{t_n} f(\boldsymbol{x}(t))\mathrm{d}t + \int_{t_{n-1}}^{t_n} \boldsymbol{B}_w \boldsymbol{w}(t)\mathrm{d}t.$$

We can still use the rectangle approximation for the integral involving the deterministic part (second term on the right hand side), but for the stochastic part, we are again lacking the integration rules. Thus, we can not use the rectangle approximation in this case. However, we can again define the resulting random variable as

$$\boldsymbol{q}_n \triangleq \int_{t_{n-1}}^{t_n} \boldsymbol{B}_w \boldsymbol{w}(t)\mathrm{d}t$$

and investigate its properties instead.

First, the mean is given by

$$\begin{aligned}\mathrm{E}\{\boldsymbol{q}_n\} &= \mathrm{E}\left\{\int_{t_{n-1}}^{t_n} \boldsymbol{B}_w \boldsymbol{w}(t)\mathrm{d}t\right\} \\ &= \int_{t_{n-1}}^{t_n} \boldsymbol{B}_w \,\mathrm{E}\left\{\boldsymbol{w}(t)\right\}\mathrm{d}t \\ &= 0,\end{aligned}$$

where we have made use of the assumption that $\boldsymbol{w}(t)$ is a zero-mean process. Second, the covariance can be found similar to the linear case as follows.

$$\begin{aligned}
\operatorname{Cov}\{\boldsymbol{q}_n\} &= \mathrm{E}\left\{\left(\int_{t_{n-1}}^{t_n} \boldsymbol{B}_w \boldsymbol{w}(t)\mathrm{d}t\right)\left(\int_{t_{n-1}}^{t_n} \boldsymbol{B}_w \boldsymbol{w}(\tau)\mathrm{d}\tau\right)^{\mathsf{T}}\right\} \\
&= \int_{t_{n-1}}^{t_n}\int_{t_{n-1}}^{t_n} \boldsymbol{B}_w \,\mathrm{E}\{\boldsymbol{w}(t)\boldsymbol{w}(\tau)^{\mathsf{T}}\}\boldsymbol{B}_w^{\mathsf{T}}\mathrm{d}\tau\mathrm{d}t \\
&= \int_{t_{n-1}}^{t_n}\int_{t_{n-1}}^{t_n} \boldsymbol{B}_w \boldsymbol{\Sigma}_w \delta(t-\tau)\boldsymbol{B}_w^{\mathsf{T}}\mathrm{d}\tau\mathrm{d}t \\
&= \int_{t_{n-1}}^{t_n} \boldsymbol{B}_w \boldsymbol{\Sigma}_w \boldsymbol{B}_w^{\mathsf{T}}\mathrm{d}\tau.
\end{aligned}$$

At this point, note that the remaining integral does not involve any random process anymore. Hence, we can again use the rectangle approximation of the integral, which yields

$$\begin{aligned}
\operatorname{Cov}\{\boldsymbol{q}_n\} &\approx (\boldsymbol{B}_w \boldsymbol{\Sigma}_w \boldsymbol{B}_w^{\mathsf{T}})(t_n - t_{n-1}) \\
&= \Delta t \boldsymbol{B}_w \boldsymbol{\Sigma}_w \boldsymbol{B}_w^{\mathsf{T}} \\
&\triangleq \boldsymbol{Q}_n.
\end{aligned}$$

This yields the *Euler–Maruyama* discretization of the stochastic nonlinear dynamic model given by

$$\boldsymbol{x}_n = \boldsymbol{x}_{n-1} + \Delta t f(\boldsymbol{x}_{n-1}) + \boldsymbol{q}_n, \tag{4.50}$$

with

$$\begin{aligned}
\boldsymbol{q}_n &\sim \mathcal{N}(0, \boldsymbol{Q}_n), \\
\boldsymbol{Q}_n &= \Delta t \boldsymbol{B}_w \boldsymbol{\Sigma}_w \boldsymbol{B}_w^{\mathsf{T}}.
\end{aligned}$$

Chapter 5

Filtering

In Chapter 4, we developed state-space models that describe the dynamic behavior of time-varying processes and how the sensor measurements relate to the state. Unfortunately, the estimation methods developed earlier do not work for such dynamic problems: These solve the estimation problem in the static case, but not when the parameters (i.e., the states) vary between samples. Instead, new methods are required which are able to combine the prior information from the dynamic model and the measurements at different points in time. This methodology is called filtering and the general approach is discussed in Section 5.1, which is followed by the exact solution of the filtering problem for linear state-space models in Section 5.2. Approximate filtering methods for nonlinear systems are discussed in Section 5.3–5.5.

5.1 The Filtering Approach

The basic discussion of the filtering approach is based on the general discrete-time state-space model of the form

$$\boldsymbol{x}_n = f(\boldsymbol{x}_{n-1}) + \boldsymbol{q}_n, \tag{5.1a}$$
$$\boldsymbol{y}_n = g(\boldsymbol{x}_n) + \boldsymbol{r}_n, \tag{5.1b}$$

where the process and measurement noises are zero-mean ($\mathrm{E}\{\boldsymbol{q}_n\} = \mathrm{E}\{\boldsymbol{r}_n\} = 0$) with covariances $\mathrm{Cov}\{\boldsymbol{q}_n\} = \boldsymbol{Q}_n$ and $\mathrm{Cov}\{\boldsymbol{r}_n\} = \boldsymbol{R}_n$, respectively. Note that the dynamic model (5.1a) may be an inherently discrete-time model (Section 4.2) or the result of discretizing a continuous-time dynamic model.

An aspect of the state-space model that has been neglected so far are the initial conditions. Since the dynamic model (5.1a) is based on a differential (difference) equation, it also requires knowledge of the initial conditions of the system. Naturally, in the sensor fusion and estimation context, the initial conditions are unknown (e.g., it is not known where exactly a target is located in the beginning). Instead, it is suitable to specify the initial conditions in a probabilistic way. In other words, we assume that the state at t_0, denoted as $\boldsymbol{x}_0 \triangleq \boldsymbol{x}(t_0)$, is a random variable which follows a probability density function

$p(\boldsymbol{x}_0)$, that is,

$$\boldsymbol{x}_0 \sim p(\boldsymbol{x}_0).$$

This allows us to specify the prior knowledge about where the initial state may be but also take into account that this is an uncertain guess. In practice, the distribution of the initial state is often assumed to follow a Gaussian distribution, but this does not need to be the case. Here, we only make the assumptions that the mean and covariance of the initial state are given by

$$\mathrm{E}\{\boldsymbol{x}_0\} = \boldsymbol{m}_0, \tag{5.2a}$$

$$\mathrm{Cov}\{\boldsymbol{x}_0\} = \boldsymbol{P}_0. \tag{5.2b}$$

Given the state-space model defined by (5.1) and (5.2), the filtering approach can now be formulated. In particular, filtering iterates between 1) a *prediction* step, which predicts the current state using the dynamic model and the previous estimate of the state (also called *time update*), and 2) a *measurement update* step that estimates the current state using the prediction and the new measurement. This prediction–update strategy is actually very general, and as it will be shown, all the algorithms discussed in this chapter make use of these two steps.

Prediction. The aim of the prediction step is to estimate the current state $\boldsymbol{x}_n$ at t_n given the set of all the previous measurement data $\boldsymbol{y}_{1:n-1} = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \dots, \boldsymbol{y}_{n-1}\}$, and hence, given the estimate of the state $\boldsymbol{x}_{n-1}$ at t_{n-1}. Here, the mean of the prediction is denoted as $\hat{\boldsymbol{x}}_{n|n-1}$, which is read as "the estimate of the state at t_n given the data up to t_{n-1}". The hat indicates that it is an estimation, whereas the subscript is closely related to the notation for conditional expectations and densities, that is, it is read as "n given $n-1$". Similarly, the covariance of the predicted $\boldsymbol{x}_n$ is denoted as $\boldsymbol{P}_{n|n-1}$.

Measurement Update. In the measurement update, the newly obtained measurement $\boldsymbol{y}_n$ is used together with the prediction (and its uncertainty) to estimate the current value of the state $\boldsymbol{x}_n$. Similar to the prediction, the mean of the updated state estimate at time t_n is denoted as $\hat{\boldsymbol{x}}_{n|n}$, which is read as "the estimate of the state at t_n given the data up to t_n". In other words, this estimate is now based on the set of all measurements including the latest measurement at t_n given by $\boldsymbol{y}_{1:n} = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \dots, \boldsymbol{y}_n\}$. Again, the covariance of the updated estimate is denoted as $\boldsymbol{P}_{n|n}$.

5.2 Kalman Filter

In this section, filtering for linear state-space models is considered. Recall that the linear state-space model with continuous-time dynamic model is (see Section 4.1)

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{w}(t), \tag{5.3a}$$

$$\boldsymbol{y}_n = \boldsymbol{G}_n\boldsymbol{x}(t_n) + \boldsymbol{r}_n. \tag{5.3b}$$

Together with the initial distribution (5.2), the model (5.3) has the discrete-time equivalent

$$\boldsymbol{x}_n = \boldsymbol{F}_n \boldsymbol{x}_{n-1} + \boldsymbol{q}_n \tag{5.4a}$$
$$\boldsymbol{y}_n = \boldsymbol{G}_n \boldsymbol{x}_n + \boldsymbol{r}_n \tag{5.4b}$$

with

$$\mathrm{E}\{\boldsymbol{x}_0\} = \boldsymbol{m}_0,\ \mathrm{Cov}\{\boldsymbol{x}_0\} = \boldsymbol{P}_0, \tag{5.5a}$$
$$\mathrm{E}\{\boldsymbol{q}_n\} = 0,\ \mathrm{Cov}\{\boldsymbol{q}_n\} = \boldsymbol{Q}_n, \tag{5.5b}$$
$$\mathrm{E}\{\boldsymbol{r}_n\} = 0,\ \mathrm{Cov}\{\boldsymbol{r}_n\} = \boldsymbol{R}_n. \tag{5.5c}$$

Based on (5.4)–(5.5), the prediction and update steps can now be derived. For ease of presentation, the measurement update step is presented first, followed by the prediction.

Measurement Update. For the measurement update at time t_n, assume that there exists a prediction of the mean of the state $\hat{\boldsymbol{x}}_{n|n-1}$ and its covariance $\boldsymbol{P}_{n|n-1}$. This can be seen as the *prior knowledge* of the state at t_n. Then, the new measurement $\boldsymbol{y}_n$ provides the actual (noisy) information about the true state. Hence, the objective is to minimize the error with respect to the measurement, taking the prior information (prediction) into account.

Recall that we have based our estimators mainly on cost functions in general and quadratic cost functions (least squares) in particular. A cost function that is particularly well suited for this kind of problem is the *regularized least squares* cost function. Hence, for the filtering problem with the linear measurement model, the cost function becomes

$$\begin{aligned} J_{\mathrm{ReLS}}(\boldsymbol{x}_n) = (&\boldsymbol{y}_n - \boldsymbol{G}_n \boldsymbol{x}_n)^\mathsf{T} \boldsymbol{R}_n^{-1} (\boldsymbol{y}_n - \boldsymbol{G}_n \boldsymbol{x}_n) \\ &+ (\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n|n-1})^\mathsf{T} \boldsymbol{P}_{n|n-1}^{-1} (\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n|n-1}), \end{aligned} \tag{5.6}$$

where the first term accounts for the measurement $\boldsymbol{y}_n$ and the second term, the regularization term, accounts for the prior information from the prediction.

To estimate the state, we can now solve the regularized least squares problem

$$\hat{\boldsymbol{x}}_{n|n} = \underset{\boldsymbol{x}_n}{\operatorname{argmin}}\, J_{\mathrm{ReLS}}(\boldsymbol{x}_n). \tag{5.7}$$

Since the cost function is completely linear in the unknown $\boldsymbol{x}_n$, solving (5.7) follows the same steps as solving the linear least squares problems in Chapter 2 and we can find a closed-form solution.

Fortunately, we have already solved this problem for the static case. The solution is given by (see Section 2.4)

$$\hat{\boldsymbol{x}}_{n|n} = \hat{\boldsymbol{x}}_{n|n-1} + \boldsymbol{K}_n (\boldsymbol{y}_n - \boldsymbol{G}_n \hat{\boldsymbol{x}}_{n|n-1}) \tag{5.8}$$

where $\boldsymbol{K}_n$ is called the *Kalman gain* that is given by

$$\boldsymbol{K}_n = \boldsymbol{P}_{n|n-1} \boldsymbol{G}_n^\mathsf{T} (\boldsymbol{G}_n \boldsymbol{P}_{n|n-1} \boldsymbol{G}_n^\mathsf{T} + \boldsymbol{R}_n)^{-1}. \tag{5.9}$$

Furthermore, we have also shown that the covariance of $\hat{\boldsymbol{x}}_{n|n}$ is

$$\boldsymbol{P}_{n|n} = \boldsymbol{P}_{n|n-1} - \boldsymbol{K}_n(\boldsymbol{G}_n\boldsymbol{P}_{n|n-1}\boldsymbol{G}_n^\mathsf{T} + \boldsymbol{R}_n)\boldsymbol{K}_n^\mathsf{T}. \tag{5.10}$$

An important property of the measurement update step is that the updated estimate and its covariance actually are the mean and covariance of the state given the set of all measurements $\boldsymbol{y}_{1:n} = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \dots, \boldsymbol{y}_n\}$ (Särkkä, 2013). In other words, it holds that

$$\mathrm{E}\{\boldsymbol{x}_n \mid \boldsymbol{y}_{1:n}\} = \hat{\boldsymbol{x}}_{n|n}, \tag{5.11a}$$

$$\mathrm{Cov}\{\boldsymbol{x}_n \mid \boldsymbol{y}_{1:n}\} = \boldsymbol{P}_{n|n}, \tag{5.11b}$$

where $\mathrm{E}\{\boldsymbol{x}_n \mid \boldsymbol{y}_{1:n}\}$ denotes the conditional expectation of $\boldsymbol{y}_n$ given the data $\boldsymbol{y}_{1:n}$ (and similar for the covariance).

Prediction. Given the estimated mean $\mathrm{E}\{\boldsymbol{x}_{n-1} \mid \boldsymbol{y}_{1:n-1}\} = \hat{\boldsymbol{x}}_{n-1|n-1}$ and its covariance $\mathrm{Cov}\{\boldsymbol{x}_{n-1} \mid \boldsymbol{y}_{1:n-1}\} = \boldsymbol{P}_{n-1|n-1}$ from the previous time step t_{n-1}, the predicted mean and covariance can now be calculated. The mean is given by

$$\hat{\boldsymbol{x}}_{n|n-1} = \mathrm{E}\{\boldsymbol{x}_n \mid \boldsymbol{y}_{1:n-1}\}.$$

Substituting $\boldsymbol{x}_n$ in the expectation using the dynamic model given in (5.4), the mean can be rewritten as

$$\mathrm{E}\{\boldsymbol{x}_n \mid \boldsymbol{y}_{1:n-1}\} = \mathrm{E}\{\boldsymbol{F}_n\boldsymbol{x}_{n-1} + \boldsymbol{q}_n \mid \boldsymbol{y}_{1:n-1}\}.$$

Distributing the expectation over the sum and noting that $\mathrm{E}\{\boldsymbol{q}_n \mid \boldsymbol{y}_{1:n-1}\} = 0$ yields

$$\begin{aligned}\mathrm{E}\{\boldsymbol{x}_n \mid \boldsymbol{y}_{1:n-1}\} &= \boldsymbol{F}_n \,\mathrm{E}\{\boldsymbol{x}_{n-1} \mid \boldsymbol{y}_{1:n-1}\} \\ &= \boldsymbol{F}_n\hat{\boldsymbol{x}}_{n-1|n-1},\end{aligned} \tag{5.12}$$

where the last equality makes use of (5.11).

Similarly, the covariance is found from

$$\begin{aligned}\boldsymbol{P}_{n|n-1} &= \mathrm{Cov}\{\boldsymbol{x}_n \mid \boldsymbol{y}_{1:n-1}\} \\ &= \mathrm{E}\{(\boldsymbol{x}_n - \mathrm{E}\{\boldsymbol{x}_n \mid \boldsymbol{y}_{1:n-1}\})(\boldsymbol{x}_n - \mathrm{E}\{\boldsymbol{x}_n \mid \boldsymbol{y}_{1:n-1}\})^\mathsf{T} \mid \boldsymbol{y}_{1:n-1}\}.\end{aligned}$$

Using the expression of the dynamic model (5.4) for $\boldsymbol{x}_n$ and (5.12) for $\mathrm{E}\{\boldsymbol{x}_n \mid \boldsymbol{y}_{1:n-1}\}$ yields

$$\begin{aligned}&\mathrm{Cov}\{\boldsymbol{x}_n \mid \boldsymbol{y}_{1:n-1}\} \\ &\quad= \mathrm{E}\{(\boldsymbol{F}_n\boldsymbol{x}_{n-1} + \boldsymbol{q}_n - \boldsymbol{F}_n\hat{\boldsymbol{x}}_{n-1|n-1})(\boldsymbol{F}_n\boldsymbol{x}_{n-1} + \boldsymbol{q}_n - \boldsymbol{F}_n\hat{\boldsymbol{x}}_{n-1|n-1})^\mathsf{T} \mid \boldsymbol{y}_{1:n-1}\}.\end{aligned}$$

By rewriting and expanding the quadratic term and distributing the expectation we obtain

$$\begin{aligned}\mathrm{Cov}\{\boldsymbol{x}_n \mid \boldsymbol{y}_{1:n-1}\} &= \mathrm{E}\{(\boldsymbol{F}_n\boldsymbol{x}_{n-1} - \boldsymbol{F}_n\hat{\boldsymbol{x}}_{n-1|n-1})(\boldsymbol{F}_n\boldsymbol{x}_{n-1} - \boldsymbol{F}_n\hat{\boldsymbol{x}}_{n|n-1})^\mathsf{T}\} \\ &\quad+ \mathrm{E}\{\boldsymbol{q}_n(\boldsymbol{F}_n\boldsymbol{x}_{n-1} - \boldsymbol{F}_n\hat{\boldsymbol{x}}_{n-1|n-1})^\mathsf{T} \mid \boldsymbol{y}_{1:n-1}\} \\ &\quad+ \mathrm{E}\{(\boldsymbol{F}_n\boldsymbol{x}_{n-1} - \boldsymbol{F}_n\hat{\boldsymbol{x}}_{n-1|n-1})\boldsymbol{q}_n^\mathsf{T} \mid \boldsymbol{y}_{1:n-1}\} + \mathrm{E}\{\boldsymbol{q}_n\boldsymbol{q}_n^\mathsf{T} \mid \boldsymbol{y}_{1:n-1}\}.\end{aligned}$$

Algorithm 5.1 Kalman Filter

```
1: Initialize $\hat{\boldsymbol{x}}_{0|0} = \boldsymbol{m}_0$, $\boldsymbol{P}_{0|0} = \boldsymbol{P}_0$
2: for $n = 1, 2, \dots$ do
3:     Prediction (time update):
           $\hat{\boldsymbol{x}}_{n|n-1} = \boldsymbol{F}_n \hat{\boldsymbol{x}}_{n-1|n-1}$
           $\boldsymbol{P}_{n|n-1} = \boldsymbol{F}_n \boldsymbol{P}_{n-1|n-1} \boldsymbol{F}_n^\mathsf{T} + \boldsymbol{Q}_n$
4:     Measurement update:
           $\boldsymbol{K}_n = \boldsymbol{P}_{n|n-1} \boldsymbol{G}_n^\mathsf{T} (\boldsymbol{G}_n \boldsymbol{P}_{n|n-1} \boldsymbol{G}_n + \boldsymbol{R}_n)^{-1}$
           $\hat{\boldsymbol{x}}_{n|n} = \hat{\boldsymbol{x}}_{n|n-1} + \boldsymbol{K}_n (\boldsymbol{y}_n - \boldsymbol{G}_n \hat{\boldsymbol{x}}_{n|n-1})$
           $\boldsymbol{P}_{n|n} = \boldsymbol{P}_{n|n-1} - \boldsymbol{K}_n (\boldsymbol{G}_n \boldsymbol{P}_{n|n-1} \boldsymbol{G}_n + \boldsymbol{R}_n) \boldsymbol{K}_n^\mathsf{T}$
5: end for
```

Noting that the middle terms are zero due to the factors being independent and $\boldsymbol{q}_n$ being zero-mean, what remains is

$$\begin{aligned}\operatorname{Cov}\{\boldsymbol{x}_n \mid \boldsymbol{y}_{1:n-1}\} = \mathrm{E}\{&(\boldsymbol{F}_n \boldsymbol{x}_{n-1} - \boldsymbol{F}_n \hat{\boldsymbol{x}}_{n-1|n-1})(\boldsymbol{F}_n \boldsymbol{x}_{n-1} - \boldsymbol{F}_n \hat{\boldsymbol{x}}_{n-1|n-1})^\mathsf{T} \mid \boldsymbol{y}_{1:n-1}\} \\ &+ \mathrm{E}\{\boldsymbol{q}_n \boldsymbol{q}_n^\mathsf{T} \mid \boldsymbol{y}_{1:n-1}\},\end{aligned}$$

and finally,

$$\operatorname{Cov}\{\boldsymbol{x}_n \mid \boldsymbol{y}_{1:n-1}\} = \boldsymbol{F}_n \boldsymbol{P}_{n-1|n-1} \boldsymbol{F}_n^\mathsf{T} + \boldsymbol{Q}_n. \tag{5.13}$$

Kalman Filter. Gathering the results (5.8)–(5.10) and (5.12)–(5.13), the filtering algorithm for linear state-space models can now be formulated. First, during the prediction (time update) step the predicted mean and covariance are calculated according to

$$\hat{\boldsymbol{x}}_{n|n-1} = \boldsymbol{F}_n \hat{\boldsymbol{x}}_{n-1|n-1}, \tag{5.14a}$$

$$\boldsymbol{P}_{n|n-1} = \boldsymbol{F}_n \boldsymbol{P}_{n-1|n-1} \boldsymbol{F}_n^\mathsf{T} + \boldsymbol{Q}_n. \tag{5.14b}$$

Second, in the measurement update step, the new measurement is incorporated and the new state is estimated using

$$\boldsymbol{K}_n = \boldsymbol{P}_{n|n-1} \boldsymbol{G}_n^\mathsf{T} (\boldsymbol{G}_n \boldsymbol{P}_{n|n-1} \boldsymbol{G}_n^\mathsf{T} + \boldsymbol{R}_n)^{-1}, \tag{5.15a}$$

$$\hat{\boldsymbol{x}}_{n|n} = \hat{\boldsymbol{x}}_{n|n-1} + \boldsymbol{K}_n (\boldsymbol{y}_n - \boldsymbol{G}_n \hat{\boldsymbol{x}}_{n|n-1}), \tag{5.15b}$$

$$\boldsymbol{P}_{n|n} = \boldsymbol{P}_{n|n-1} - \boldsymbol{K}_n (\boldsymbol{G}_n \boldsymbol{P}_{n|n-1} \boldsymbol{G}_n^\mathsf{T} + \boldsymbol{R}_n) \boldsymbol{K}_n^\mathsf{T}. \tag{5.15c}$$

Furthermore, the recursion is initialized by letting $\hat{\boldsymbol{x}}_{0|0} = \boldsymbol{m}_0$ and $\boldsymbol{P}_{0|0} = \boldsymbol{P}_0$. This leads to the algorithm shown in Algorithm 5.1, which is called the *Kalman filter* (KF) (Kalman, 1960).

It is worth pointing out several aspects of the prediction and measurement update (5.14)–(5.15). First, note that in the prediction, the covariance increases due to the scaling factor $\boldsymbol{F}_n$ and the added uncertainty by the process noise. On the other hand, during the measurement update, the covariance is decreased by a factor that scales quadratically with the gain $\boldsymbol{K}_n$. This follows the intuition that a prediction should increase the uncertainty, while incorporating new information should decrease it.

Second, the measurement update is a sum of the prediction $\hat{\boldsymbol{x}}_{n|n-1}$ and the term $\boldsymbol{y}_n - \boldsymbol{G}_n\hat{\boldsymbol{x}}_{n|n-1}$ scaled by the gain $\boldsymbol{K}_n$. Noting that $\boldsymbol{G}_n$ actually is the matrix relating the state $\boldsymbol{x}_n$ to the measurement $\boldsymbol{y}_n$, the term $\boldsymbol{G}_n\hat{\boldsymbol{x}}_{n|n-1}$ can be interpreted as a prediction of the output, which it in fact is. To show this, consider the expected value of the output $\boldsymbol{y}_n$ given all the data up to t_{n-1}, that is,

$$\begin{aligned}
\mathrm{E}\{\boldsymbol{y}_n \mid \boldsymbol{y}_{1:n-1}\} &= \mathrm{E}\{\boldsymbol{G}_n\boldsymbol{x}_n + \boldsymbol{r}_n \mid \boldsymbol{y}_{1:n-1}\} \\
&= \boldsymbol{G}_n \,\mathrm{E}\{\boldsymbol{x}_n \mid \boldsymbol{y}_{1:n-1}\} + \mathrm{E}\{\boldsymbol{r}_n \mid \boldsymbol{y}_{1:n-1}\}
\end{aligned}$$

and thus

$$\mathrm{E}\{\boldsymbol{y}_n \mid \boldsymbol{y}_{1:n-1}\} = \boldsymbol{G}_n\hat{\boldsymbol{x}}_{n|n-1}. \tag{5.16}$$

Hence, the difference between the measurement and its prediction gives an indication about how far the predicted state is from the true state: If the difference is large, the predicted state is far and should be corrected a lot, if it is small, it is close and does not need to be corrected much. Hence, this difference is also called the *innovation*.

Third, the covariance of the predicted output $\boldsymbol{y}_n$ is given by

$$\begin{aligned}
\mathrm{Cov}\{\boldsymbol{y}_n \mid \boldsymbol{y}_{1:n-1}\} &= \mathrm{E}\{(\boldsymbol{y}_n - \mathrm{E}\{\boldsymbol{y}_n \mid \boldsymbol{y}_{1:n-1}\})(\boldsymbol{y}_n - \mathrm{E}\{\boldsymbol{y}_n \mid \boldsymbol{y}_{1:n-1}\})^\mathsf{T} \mid \boldsymbol{y}_{1:n-1}\} \\
&= \mathrm{E}\{(\boldsymbol{G}_n\boldsymbol{x}_n + \boldsymbol{r}_n - \boldsymbol{G}_n\hat{\boldsymbol{x}}_{n|n-1})(\boldsymbol{G}_n\boldsymbol{x}_n + \boldsymbol{r}_n - \boldsymbol{G}_n\hat{\boldsymbol{x}}_{n|n-1})^\mathsf{T} \mid \boldsymbol{y}_{1:n-1}\} \\
&= \mathrm{E}\{(\boldsymbol{G}_n(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n|n-1})(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n|n-1})^\mathsf{T}\boldsymbol{G}_n^\mathsf{T} + \boldsymbol{r}_n\boldsymbol{r}_n^\mathsf{T} \mid \boldsymbol{y}_{1:n-1}\}
\end{aligned}$$

and finally,

$$\mathrm{Cov}\{\boldsymbol{y}_n \mid \boldsymbol{y}_{1:n-1}\} = \boldsymbol{G}_n\boldsymbol{P}_{n|n-1}\boldsymbol{G}_n^\mathsf{T} + \boldsymbol{R}_n, \tag{5.17}$$

which is the denominator of the Kalman gain $\boldsymbol{K}_n$. Similarly, the cross-covariance between the predicted state $\boldsymbol{x}_n$ and predicted output $\boldsymbol{y}_n$ is

$$\begin{aligned}
\mathrm{Cov}\{\boldsymbol{x}_n, \boldsymbol{y}_n \mid \boldsymbol{y}_{1:n-1}\} &= \mathrm{E}\{(\boldsymbol{x}_n - \mathrm{E}\{\boldsymbol{x}_n \mid \boldsymbol{y}_{1:n-1}\})(\boldsymbol{y}_n - \mathrm{E}\{\boldsymbol{y}_n \mid \boldsymbol{y}_{1:n-1}\})^\mathsf{T} \mid \boldsymbol{y}_{1:n-1}\} \\
&= \mathrm{E}\{(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n|n-1})(\boldsymbol{G}_n\boldsymbol{x}_n + \boldsymbol{r}_n - \boldsymbol{G}_n\hat{\boldsymbol{x}}_{n|n-1})^\mathsf{T} \mid \boldsymbol{y}_{1:n-1}\} \\
&= \mathrm{E}\{(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n|n-1})(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n|n-1})^\mathsf{T}\boldsymbol{G}_n^\mathsf{T} \mid \boldsymbol{y}_{1:n-1}\},
\end{aligned}$$

which yields

$$\mathrm{Cov}\{\boldsymbol{x}_n, \boldsymbol{y}_n \mid \boldsymbol{y}_{1:n-1}\} = \boldsymbol{P}_{n|n-1}\boldsymbol{G}_n^\mathsf{T}. \tag{5.18}$$

Thus, using (5.16)–(5.18), the measurement update can be written in the alternative form as

$$\boldsymbol{K}_n = \mathrm{Cov}\{\boldsymbol{x}_n, \boldsymbol{y}_n \mid \boldsymbol{y}_{1:n-1}\}\,\mathrm{Cov}\{\boldsymbol{y}_n \mid \boldsymbol{y}_{1:n-1}\}^{-1}, \tag{5.19a}$$

$$\hat{\boldsymbol{x}}_{n|n} = \hat{\boldsymbol{x}}_{n|n-1} + \boldsymbol{K}_n(\boldsymbol{y}_n - \mathrm{E}\{\boldsymbol{y}_n \mid \boldsymbol{y}_{1:n-1}\}), \tag{5.19b}$$

$$\boldsymbol{P}_{n|n} = \boldsymbol{P}_{n|n-1} - \boldsymbol{K}_n\,\mathrm{Cov}\{\boldsymbol{y}_n \mid \boldsymbol{y}_{1:n-1}\}\boldsymbol{K}_n^\mathsf{T}. \tag{5.19c}$$

From (5.19), we can see that the denominator of the Kalman gain $\boldsymbol{K}_n$ is the covariance of the predicted measurement. Hence, if the uncertainty of the prediction is large (either due to a large uncertainty in the predicted state or due to large measurement noise covariance $\boldsymbol{R}_n$), the gain becomes small. This in turn means that the innovation only contributes little information to the updated state. Conversely, if the uncertainty of the prediction is small, the gain becomes large and the innovation contributes a lot.

5.3 Extended Kalman Filter

Similar to the static, linear case discussed in Chapter 2, the Kalman filter is the closed-form solution for the state estimation problem in linear state-space models and thus an exact solution. However, if we consider general nonlinear state-space models of the form (5.1)–(5.2), closed-form solutions are normally not available. Similar to the nonlinear static case, we need approximative solutions instead. A first approximation is found in the *extended Kalman filter* (EKF), which is based on a linearization using a Taylor series approximation of the nonlinear dynamic and measurement models. Using this linearization, prediction and measurement update steps similar to the Kalman filter are found.

Prediction. For the prediction, assume that the state estimate and its covariance for t_{n-1} are given by $\hat{\boldsymbol{x}}_{n-1|n-1}$ and $\boldsymbol{P}_{n-1|n-1}$, respectively. Then, the linear (first order) Taylor series approximation of the dynamic model around the linearization point $\boldsymbol{x}_{n-1} = \hat{\boldsymbol{x}}_{n-1|n-1}$ is given by

$$\begin{aligned} \boldsymbol{x}_n &= f(\boldsymbol{x}_{n-1}) + \boldsymbol{q}_n \\ &\approx f(\hat{\boldsymbol{x}}_{n-1|n-1}) + \boldsymbol{F}_x(\boldsymbol{x}_{n-1} - \hat{\boldsymbol{x}}_{n-1|n-1}) + \boldsymbol{q}_n, \end{aligned} \tag{5.20}$$

where $\boldsymbol{F}_x$ is the Jacobian matrix of the vector-value function $f(\boldsymbol{x}_{n-1})$. Then, based on the linearized dynamic model (5.20), we can predict the mean and covariance of the state $\boldsymbol{x}_n$ at t_n given the set of all the measurements $\boldsymbol{y}_{1:n-1} = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \dots, \boldsymbol{y}_{n-1}\}$.

The mean is given by

$$\begin{aligned} \hat{\boldsymbol{x}}_{n|n-1} &= \mathrm{E}\{\boldsymbol{x}_n \mid \boldsymbol{y}_{1:n-1}\} \\ &\approx \mathrm{E}\{f(\hat{\boldsymbol{x}}_{n-1|n-1}) + \boldsymbol{F}_x(\boldsymbol{x}_{n-1} - \hat{\boldsymbol{x}}_{n-1|n-1}) + \boldsymbol{q}_n \mid \boldsymbol{y}_{1:n-1}\} \\ &= f(\hat{\boldsymbol{x}}_{n-1|n-1}) + \boldsymbol{F}_x \, \mathrm{E}\{\boldsymbol{x}_{n-1} \mid \boldsymbol{y}_{1:n-1}\} - \boldsymbol{F}_x\hat{\boldsymbol{x}}_{n-1|n-1} \\ &= f(\hat{\boldsymbol{x}}_{n-1|n-1}) + \boldsymbol{F}_x\hat{\boldsymbol{x}}_{n-1|n-1} - \boldsymbol{F}_x\hat{\boldsymbol{x}}_{n-1|n-1}, \end{aligned}$$

and thus

$$\hat{\boldsymbol{x}}_{n|n-1} = f(\hat{\boldsymbol{x}}_{n-1|n-1}). \tag{5.21}$$

Similarly, the covariance is

$$\begin{aligned}
\boldsymbol{P}_{n|n-1} &= \mathrm{E}\{(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n|n-1})(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n|n-1})^{\mathsf{T}} \mid \boldsymbol{y}_{1:n-1}\} \\
&\approx \mathrm{E}\{(f(\hat{\boldsymbol{x}}_{n-1|n-1}) + \boldsymbol{F}_x(\boldsymbol{x}_{n-1} - \hat{\boldsymbol{x}}_{n-1|n-1}) + \boldsymbol{q}_n - f(\hat{\boldsymbol{x}}_{n-1|n-1})) \\
&\qquad \times (f(\hat{\boldsymbol{x}}_{n-1|n-1}) + \boldsymbol{F}_x(\boldsymbol{x}_{n-1} - \hat{\boldsymbol{x}}_{n-1|n-1}) + \boldsymbol{q}_n - f(\hat{\boldsymbol{x}}_{n-1|n-1}))^{\mathsf{T}} \mid \boldsymbol{y}_{1:n-1}\} \\
&= \mathrm{E}\{(\boldsymbol{F}_x(\boldsymbol{x}_{n-1} - \hat{\boldsymbol{x}}_{n-1|n-1}) + \boldsymbol{q}_n)(\boldsymbol{F}_x(\boldsymbol{x}_{n-1} - \hat{\boldsymbol{x}}_{n-1|n-1}) + \boldsymbol{q}_n)^{\mathsf{T}} \mid \boldsymbol{y}_{1:n-1}\} \\
&= \boldsymbol{F}_x \, \mathrm{E}\{(\boldsymbol{x}_{n-1} - \hat{\boldsymbol{x}}_{n-1|n-1})(\boldsymbol{x}_{n-1} - \hat{\boldsymbol{x}}_{n-1|n-1})^{\mathsf{T}} \mid \boldsymbol{y}_{1:n-1}\}\boldsymbol{F}_x^{\mathsf{T}} \\
&\qquad + \mathrm{E}\{\boldsymbol{q}_n\boldsymbol{q}_n^{\mathsf{T}} \mid \boldsymbol{y}_{1:n-1}\}
\end{aligned}$$

and finally,

$$\boldsymbol{P}_{n|n-1} = \boldsymbol{F}_x\boldsymbol{P}_{n-1|n-1}\boldsymbol{F}_x^{\mathsf{T}} + \boldsymbol{Q}_n. \tag{5.22}$$

Measurement Update. The derivation of the measurement update follows very similar steps as the prediction in the EKF and the measurement update in the KF. Given the predicted mean $\boldsymbol{x}_{n|n-1}$ and covariance $\boldsymbol{P}_{n|n-1}$, the nonlinear measurement model is linearized using a first order Taylor series expansion around the prediction $\hat{\boldsymbol{x}}_{n|n-1}$. This yields

$$\begin{aligned}
\boldsymbol{y}_n &= g(\boldsymbol{x}_n) + \boldsymbol{r}_n \\
&= g(\hat{\boldsymbol{x}}_{n|n-1}) + \boldsymbol{G}_x(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n|n-1}) + \boldsymbol{r}_n,
\end{aligned} \tag{5.23}$$

where $\boldsymbol{G}_x$ denotes the Jacobian matrix of $g(\boldsymbol{x}_n)$ evaluated at $\boldsymbol{x}_n = \hat{\boldsymbol{x}}_{n|n-1}$.

Next, introducing the regularized least squares criterion for the linearized model (5.23) yields

$$\begin{aligned}
J_{\mathrm{ReLS}}(\boldsymbol{x}_n) &= (\boldsymbol{y}_n - g(\hat{\boldsymbol{x}}_{n|n-1}) - \boldsymbol{G}_x(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n|n-1}))^{\mathsf{T}}\boldsymbol{R}_n^{-1} \\
&\qquad \times (\boldsymbol{y}_n - g(\hat{\boldsymbol{x}}_{n|n-1}) - \boldsymbol{G}_x(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n|n-1})) \\
&\qquad + (\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n|n-1})^{\mathsf{T}}\boldsymbol{P}_{n|n-1}^{-1}(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n|n-1}).
\end{aligned} \tag{5.24}$$

By introducing a change of variables with $\boldsymbol{z}_n = \boldsymbol{y}_n - g(\hat{\boldsymbol{x}}_{n|n-1}) + \boldsymbol{G}_x\hat{\boldsymbol{x}}_{n|n-1}$, (5.24) can be rewritten as

$$\begin{aligned}
J_{\mathrm{ReLS}}(\boldsymbol{x}_n) &= (\boldsymbol{z}_n - \boldsymbol{G}_x\boldsymbol{x}_n)^{\mathsf{T}}\boldsymbol{R}_n^{-1}(\boldsymbol{z}_n - \boldsymbol{G}_x\boldsymbol{x}_n) \\
&\qquad + (\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n|n-1})^{\mathsf{T}}\boldsymbol{P}_{n|n-1}^{-1}(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n|n-1}),
\end{aligned} \tag{5.25}$$

which is linear in $\boldsymbol{x}_n$ and has the same form as (5.6) but with the Jacobian matrix $\boldsymbol{G}_x$ rather than the measurement matrix $\boldsymbol{G}_n$. Hence, solving

$$\hat{\boldsymbol{x}}_{n|n} = \operatorname*{argmin}_{\boldsymbol{x}_n} J_{\mathrm{ReLS}}(\boldsymbol{x}_n) \tag{5.26}$$

yields

$$\begin{aligned}
\hat{\boldsymbol{x}}_{n|n} &= \hat{\boldsymbol{x}}_{n|n-1} + \boldsymbol{K}_n(\boldsymbol{z}_n - \boldsymbol{G}_x\hat{\boldsymbol{x}}_{n|n-1}), \\
\boldsymbol{K}_n &= \boldsymbol{P}_{n|n-1}\boldsymbol{G}_x^{\mathsf{T}}(\boldsymbol{G}_x\boldsymbol{P}_{n|n-1}\boldsymbol{G}_x^{\mathsf{T}} + \boldsymbol{R}_n)^{-1},
\end{aligned}$$

Algorithm 5.2 Extended Kalman Filter

```
1: Initialize $\hat{\boldsymbol{x}}_{0|0} = \boldsymbol{m}_0$, $\boldsymbol{P}_{0|0} = \boldsymbol{P}_0$
2: for $n = 1, 2, \ldots$ do
3:     Prediction (time update):
           $\hat{\boldsymbol{x}}_{n|n-1} = f(\hat{\boldsymbol{x}}_{n-1|n-1})$
           $\boldsymbol{P}_{n|n-1} = \boldsymbol{F}_x \boldsymbol{P}_{n-1|n-1} \boldsymbol{F}_x^\top + \boldsymbol{Q}_n$
4:     Measurement update:
           $\boldsymbol{K}_n = \boldsymbol{P}_{n|n-1} \boldsymbol{G}_x^\top (\boldsymbol{G}_x \boldsymbol{P}_{n|n-1} \boldsymbol{G}_x^\top + \boldsymbol{R}_n)^{-1}$
           $\hat{\boldsymbol{x}}_{n|n} = \hat{\boldsymbol{x}}_{n|n-1} + \boldsymbol{K}_n (\boldsymbol{y}_n - g(\hat{\boldsymbol{x}}_{n|n-1}))$
           $\boldsymbol{P}_{n|n} = \boldsymbol{P}_{n|n-1} - \boldsymbol{K}_n (\boldsymbol{G}_x \boldsymbol{P}_{n|n-1} \boldsymbol{G}_x^\top + \boldsymbol{R}_n) \boldsymbol{K}_n^\top$
5: end for
```

with covariance

$$\boldsymbol{P}_{n|n} \approx \boldsymbol{P}_{n|n-1} - \boldsymbol{K}_n (\boldsymbol{G}_x \boldsymbol{P}_{n|n-1} \boldsymbol{G}_x^\top + \boldsymbol{R}_n) \boldsymbol{K}_n^\top.$$

Finally, changing $\boldsymbol{z}_n$ back to its definition yields

$$\begin{aligned}\hat{\boldsymbol{x}}_{n|n} &= \hat{\boldsymbol{x}}_{n|n-1} + \boldsymbol{K}_n (\boldsymbol{y}_n - g(\hat{\boldsymbol{x}}_{n|n-1}) + \boldsymbol{G}_x \hat{\boldsymbol{x}}_{n|n-1} - \boldsymbol{G}_x \hat{\boldsymbol{x}}_{n|n-1}) \\ &= \hat{\boldsymbol{x}}_{n|n-1} + \boldsymbol{K}_n (\boldsymbol{y}_n - g(\hat{\boldsymbol{x}}_{n|n-1}))\end{aligned}$$

for the measurement update.

Extended Kalman Filter. We can now summarize the prediction and measurement update steps for the EKF. First, during the predictions step, the predicted mean and covariance are calculated according to

$$\hat{\boldsymbol{x}}_{n|n-1} = f(\hat{\boldsymbol{x}}_{n-1|n-1}), \tag{5.27a}$$

$$\boldsymbol{P}_{n|n-1} = \boldsymbol{F}_x \boldsymbol{P}_{n-1|n-1} \boldsymbol{F}_x^\top + \boldsymbol{Q}_n, \tag{5.27b}$$

where $\boldsymbol{F}_x$ is the Jacobian matrix of the dynamic model. Second, the measurement update is

$$\boldsymbol{K}_n = \boldsymbol{P}_{n|n-1} \boldsymbol{G}_x^\top (\boldsymbol{G}_x \boldsymbol{P}_{n|n-1} \boldsymbol{G}_x^\top + \boldsymbol{R}_n)^{-1}, \tag{5.28a}$$

$$\hat{\boldsymbol{x}}_{n|n} = \hat{\boldsymbol{x}}_{n|n-1} + \boldsymbol{K}_n (\boldsymbol{y}_n - g(\hat{\boldsymbol{x}}_{n|n-1})), \tag{5.28b}$$

$$\boldsymbol{P}_{n|n} = \boldsymbol{P}_{n|n-1} - \boldsymbol{K}_n (\boldsymbol{G}_x \boldsymbol{P}_{n|n-1} \boldsymbol{G}_x^\top + \boldsymbol{R}_n) \boldsymbol{K}_n^\top. \tag{5.28c}$$

These two steps are then performed iteratively, and the algorithm is initialized with the initial conditions. This yields the complete EKF algorithm shown in Algorithm 5.2.

Note that this is essentially the same algorithm as the original KF where the nonlinear function takes the place of the prediction (both in the prediction step and the prediction of the output) and in the covariance updates, the Jacobian matrices take the place of the matrices in the dynamic model and measurement model. Furthermore, if either of the models is linear, the corresponding step (prediction or measurement update) may be replaced by an exact update step from the Kalman filter in Section 5.2. Finally, note that Algorithm 5.2 is an approximation of the filtering problem since the nonlinear function is approximated around the filtered and predicted state, meaning that both the state estimate and its covariance may or may not converge to the true state (similar as for static nonlinear problems).

5.4 Unscented Kalman Filtering

One of the major problems of the EKF is that the linearization is local, which often leads to problems such as the covariance of the state being underestimated (which in turn affects the Kalman gain in the next iteration). Another approach to solve the filtering problem within the Kalman filtering framework is the *unscented Kalman filter* (UKF) (Wan and Van Der Merwe, 2000; Julier and Uhlmann, 2004), which makes use of a so-called *unscented transform.* Hence, this transform will be discussed first, and then the prediction and measurement update steps for the UKF are derived.

Unscented Transform. The unscented transform is based on calculating the moments of the nonlinear transformation of a finite set of deterministic sampling points as follows. Given a random variable $\boldsymbol{x}$ with mean $\boldsymbol{m}$ and covariance $\boldsymbol{P}$, we can find a set of J points $\{\boldsymbol{x}^0, \boldsymbol{x}^2, \dots, \boldsymbol{x}^{J-1}\}$, so-called *sigma-points*, such that their weighted sum is equal to the mean and covariance of $\boldsymbol{x}$, that is, such that

$$\boldsymbol{m} = \sum_{j=0}^{J-1} w_m^j \boldsymbol{x}^j, \tag{5.29a}$$

$$\boldsymbol{P} = \sum_{j=0}^{J-1} w_P^j (\boldsymbol{x}^j - \boldsymbol{m})(\boldsymbol{x}^j - \boldsymbol{m})^\mathsf{T}, \tag{5.29b}$$

where it must hold that $\sum_{j=0}^{J-1} w_m^j = 1$ since the expected value of the sum should be unbiased.

Then, given the nonlinear function $\boldsymbol{z} = h(\boldsymbol{x})$, we can calculate the transformed sigma-points according to

$$\boldsymbol{z}^j = h(\boldsymbol{x}^j).$$

Based on the transformed sigma-points, the mean and covariance of the transformed variable $\boldsymbol{z}$ as well as the cross-covariance between $\boldsymbol{x}$ and $\boldsymbol{z}$ can then be calculated according to

$$\mathrm{E}\{\boldsymbol{z}\} \approx \sum_{j=1}^{J} w_m^j \boldsymbol{z}^j, \tag{5.30a}$$

$$\mathrm{Cov}\{\boldsymbol{z}\} \approx \sum_{j=1}^{J} w_P^j (\boldsymbol{z}^j - \mathrm{E}\{\boldsymbol{z}\})(\boldsymbol{z}^j - \mathrm{E}\{\boldsymbol{z}\})^\mathsf{T}, \tag{5.30b}$$

$$\mathrm{Cov}\{\boldsymbol{x}, \boldsymbol{z}\} \approx \sum_{j=1}^{J} w_P^j (\boldsymbol{x}^j - \boldsymbol{m})(\boldsymbol{z}^j - \mathrm{E}\{\boldsymbol{z}\})^\mathsf{T}. \tag{5.30c}$$

The question then is how to chose the sigma-points $\boldsymbol{x}^j$ and their weights for the mean w_m^j and covariance w_P^j. The unscented transform is one such approach for choosing sigma-points that fulfill the conditions (5.29). Here, $J = 2L + 1$ (with L being the dimension of the vector $\boldsymbol{x}$) sigma-points are chosen according to

$$\boldsymbol{x}^0 = \boldsymbol{m}, \tag{5.31a}$$

$$\boldsymbol{x}^j = \boldsymbol{m} + \sqrt{L+\lambda}[\sqrt{\boldsymbol{P}}]_j, \qquad j = 1, \dots, L, \tag{5.31b}$$

$$\boldsymbol{x}^j = \boldsymbol{m} - \sqrt{L+\lambda}[\sqrt{\boldsymbol{P}}]_{(j-L)}, \qquad j = L+1, \dots, 2L. \tag{5.31c}$$

In (5.31), $\sqrt{\boldsymbol{P}}$ denotes a matrix square root such that $\boldsymbol{P} = \sqrt{\boldsymbol{P}}\sqrt{\boldsymbol{P}}^\mathsf{T}$ (in practice, this can be implemented using the Cholesky factorization) and $[\sqrt{\boldsymbol{P}}]_j$ denotes the jth column of the matrix $\sqrt{\boldsymbol{P}}$. The weights corresponding to the above sigma-points are

$$w_m^0 = \frac{\lambda}{L+\lambda}, \tag{5.32a}$$

$$w_P^0 = \frac{\lambda}{L+\lambda} + (1 - \alpha^2 + \beta), \tag{5.32b}$$

$$w_m^j = w_P^j = \frac{1}{2(L+\lambda)}, \qquad j = 1, \dots, 2L. \tag{5.32c}$$

In both (5.31) and (5.32), λ is

$$\lambda = \alpha^2(L + \kappa) - L, \tag{5.33}$$

and α, β, and κ are tuning parameters. The parameters α and κ determine the scaling of the spread of the sigma-points in the direction of the covariance $\boldsymbol{P}$ whereas β is related the higher order moments of $\boldsymbol{x}$ and only affects the weight for the central sigma-point in the covariance calculations.

The choice of the tuning parameters may greatly affect the performance of the filter and several default choices have been suggested. The value for κ is most often chosen to be zero (i.e., $\kappa = 0$), which leaves the choices for α and β. First, note that with $\kappa = 0$, the weight of the central sigma-point becomes

$$w_m^0 = \frac{\lambda}{L+\lambda} = \frac{L\alpha^2 - L}{L + L\alpha^2 - L} = \frac{\alpha^2 - 1}{\alpha^2}, \tag{5.34}$$

and the scaling factor of the root of the covariance matrix becomes

$$\sqrt{L+\lambda} = \sqrt{L+\alpha^2 L - L} = \alpha\sqrt{L}. \tag{5.35}$$

One suggestion is to choose $\alpha = 1 \times 10^{-3}$ (Wan and Van Der Merwe, 2000). This gives a quite large and *negative* weight w_m^0 [see (5.34)] and a small scaling factor [see (5.35)], which makes the sigma-points lie close to the mean. To avoid this, it is sometimes more intuitive to instead start from the weight of the central point w_m^0 and then determine α based on reformulating (5.34), which yields

$$\alpha = \sqrt{\frac{1}{1-w_m^0}}. \tag{5.36}$$

Furthermore, since all the weights for $j > 0$ are the same [see (5.32)], it must also hold that

$$w_m^j = \frac{1-w_m^0}{2L}.$$

Finally, β only affects the central weight of the covariance and a good starting point is normally to chose $\beta = 2$ (see, for example, Wan and Van Der Merwe (2000)).

Prediction. The unscented transform as introduced above can directly be used in the prediction step to calculate the predicted mean and the covariance. In this case, the nonlinear transformation is the function of the dynamic model, that is, $f(\boldsymbol{x}_{n-1})$. The sigma-points are calculated using the estimated mean and covariance at t_n, that is,

$$\boldsymbol{x}_{n-1}^0 = \hat{\boldsymbol{x}}_{n-1|n-1} \tag{5.37a}$$

$$\boldsymbol{x}_{n-1}^j = \hat{\boldsymbol{x}}_{n-1|n-1} + \sqrt{L+\lambda}\left[\sqrt{\boldsymbol{P}_{n-1|n-1}}\right]_j, \qquad j = 1,\dots,L, \tag{5.37b}$$

$$\boldsymbol{x}_{n-1}^j = \hat{\boldsymbol{x}}_{n-1|n-1} - \sqrt{L+\lambda}\left[\sqrt{\boldsymbol{P}_{n-1|n-1}}\right]_{(j-L)}, \qquad j = L+1,\dots,2L. \tag{5.37c}$$

The weights of the sigma-points are given by (5.32) and λ is as in (5.33).

Then, the moments of the transformed variable, that is, the moments of the prediction become

$$\boldsymbol{x}_n^j = f(\boldsymbol{x}_{n-1}^j), \qquad j = 0,\dots,2L, \tag{5.38a}$$

$$\hat{\boldsymbol{x}}_{n|n-1} = \sum_{j=0}^{2L} w_m^j \boldsymbol{x}_n^j, \tag{5.38b}$$

$$\boldsymbol{P}_{n|n-1} = \sum_{j=0}^{2L} w_c^j (\boldsymbol{x}_n^j - \hat{\boldsymbol{x}}_{n|n-1})(\boldsymbol{x}_n^j - \hat{\boldsymbol{x}}_{n|n-1})^\mathsf{T} + \boldsymbol{Q}_n. \tag{5.38c}$$

Note that the additional term $\boldsymbol{Q}_n$ in the covariance is due to the fact that the dynamic model also includes the process noise term $\boldsymbol{q}_n$, which increases the uncertainty. In other words, the unscented transform only calculates the covariance of the $\boldsymbol{x}_{n-1}$ transformed by $f(\boldsymbol{x}_{n-1})$ and does not take the effect of the noise into account in this form.

Measurement Update. For the measurement update, first recall that it can be written in terms of the predicted output, its covariance, and the covariance between the predicted output and the state, see (5.19). Hence, we can also use the unscented transform to calculate these means and covariances.

In this case, the sigma points are calculated from the predicted mean $\hat{\boldsymbol{x}}_{n|n-1}$ and the covariance $\boldsymbol{P}_{n|n-1}$ according to

$$\boldsymbol{x}_n^0 = \hat{\boldsymbol{x}}_{n|n-1} \tag{5.39a}$$

$$\boldsymbol{x}_n^j = \hat{\boldsymbol{x}}_{n|n-1} + \sqrt{L+\lambda}\left[\sqrt{\boldsymbol{P}_{n|n-1}}\right]_j, \qquad j = 1, \dots, L, \tag{5.39b}$$

$$\boldsymbol{x}_n^j = \hat{\boldsymbol{x}}_{n|n-1} - \sqrt{L+\lambda}\left[\sqrt{\boldsymbol{P}_{n|n-1}}\right]_{(j-L)}, \qquad j = L+1, \dots, 2L, \tag{5.39c}$$

with the weights as in (5.32). Then, the necessary moments become

$$\boldsymbol{y}_n^j = g(\boldsymbol{x}_n^j), \qquad j = 0, \dots, 2L, \tag{5.40a}$$

$$\mathrm{E}\{\boldsymbol{y}_n \mid \boldsymbol{y}_{1:n-1}\} = \sum_{j=0}^{2L} w_m^j \boldsymbol{y}_n^j, \tag{5.40b}$$

$$\begin{aligned}\mathrm{Cov}\{\boldsymbol{y} \mid \boldsymbol{y}_{1:n-1}\} = \sum_{j=0}^{2L} w_P^j(\boldsymbol{y}_n^j - \mathrm{E}\{\boldsymbol{y}_n \mid \boldsymbol{y}_{1:n-1}\})(\boldsymbol{y}_n^j - \mathrm{E}\{\boldsymbol{y}_n \mid \boldsymbol{y}_{1:n-1}\})^\mathsf{T} \\ + \boldsymbol{R}_n,\end{aligned} \tag{5.40c}$$

$$\mathrm{Cov}\{\boldsymbol{x}_n, \boldsymbol{y}_n \mid \boldsymbol{y}_{1:n-1}\} = \sum_{j=0}^{2L} w_P^j(\boldsymbol{x}_n^j - \hat{\boldsymbol{x}}_{n|n-1})(\boldsymbol{y}_n^j - \mathrm{E}\{\boldsymbol{y}_n \mid \boldsymbol{y}_{1:n-1}\})^\mathsf{T}, \tag{5.40d}$$

and the measurement update given in (5.19) can be performed.

Unscented Kalman Filter. Based on the prediction and measurement update discussed above, the unscented Kalman filter then becomes as shown in Algorithm 5.3. Note that the UKF still performs the two steps very similar to the original Kalman filter for linear systems, that is, it basically estimates the mean and covariance of the state at each time step and propagates this information between time steps, but without analytical linearization. The latter also implies that the UKF does not require any Jacobians to be calculated and is thus somewhat easier to implement.

5.5 Bootstrap Particle Filter

Another approach of solving the estimation problem in dynamic systems is particle filtering, which differs quite a lot from the Kalman filtering approaches discussed in the previous sections. Rather than being based on predicting and updating the mean and covariance from time step to time step, particle filtering propagates a set of weighted random samples (called *particles*) and hence there is some resemblance to the unscented

Algorithm 5.3 Unscented Kalman Filter

```
1: Initialize $\hat{\boldsymbol{x}}_{0|0} = \boldsymbol{m}_0$, $\boldsymbol{P}_{0|0} = \boldsymbol{P}_0$
2: for $n = 1, 2, \ldots$ do
3:     Calculate $\boldsymbol{x}_{n-1}^j$, $w_m^j$, and $w_P^j$ using (5.37) and (5.32)
4:     Calculate $\hat{\boldsymbol{x}}_{n|n-1}$ and $\boldsymbol{P}_{n|n-1}$ using (5.38)
5:     Calculate $\boldsymbol{x}_n^j$, $w_m^j$, and $w_P^j$ using (5.39) and (5.32)
6:     Calculate $\mathrm{E}\{\boldsymbol{y}_n \mid \boldsymbol{y}_{1:n-1}\}$, $\mathrm{Cov}\{\boldsymbol{y}_n \mid \boldsymbol{y}_{1:n-1}\}$, and $\mathrm{Cov}\{\boldsymbol{x}_n, \boldsymbol{y}_n \mid \boldsymbol{y}_{1:n-1}\}$ using (5.40)
7:     Measurement update:
           $$\begin{aligned} \boldsymbol{K}_n &= \mathrm{Cov}\{\boldsymbol{x}_n, \boldsymbol{y}_n \mid \boldsymbol{y}_{1:n-1}\}\,\mathrm{Cov}\{\boldsymbol{y}_n \mid \boldsymbol{y}_{1:n-1}\}^{-1} \\ \hat{\boldsymbol{x}}_{n|n} &= \hat{\boldsymbol{x}}_{n|n-1} + \boldsymbol{K}_n(\boldsymbol{y}_n - \mathrm{E}\{\boldsymbol{y}_n \mid \boldsymbol{y}_{1:n-1}\}) \\ \boldsymbol{P}_{n|n} &= \boldsymbol{P}_{n|n-1} - \boldsymbol{K}_n\,\mathrm{Cov}\{\boldsymbol{y}_n \mid \boldsymbol{y}_{1:n-1}\}\boldsymbol{K}_n^\mathsf{T} \end{aligned}$$
8: end for
```

Kalman filter. The particles can be seen as qualified but random guesses of the state whereas the weights provide an indication of how good the guess is. Particle filtering is a very general methodology and comprises a large class of filtering algorithms. Here, we focus on the *bootstrap particle filter*, which has an intuitive explanation but also a rigorous mathematical background (Gordon et al., 1993; Doucet and Johansen, 2011).

To derive the particle filter, we first take another look at the general discrete-time state-space model given by

$$\begin{aligned} \boldsymbol{x}_n &= f(\boldsymbol{x}_{n-1}) + \boldsymbol{q}_n, \\ \boldsymbol{y}_n &= g(\boldsymbol{x}_{n-1}) + \boldsymbol{r}_n, \end{aligned}$$

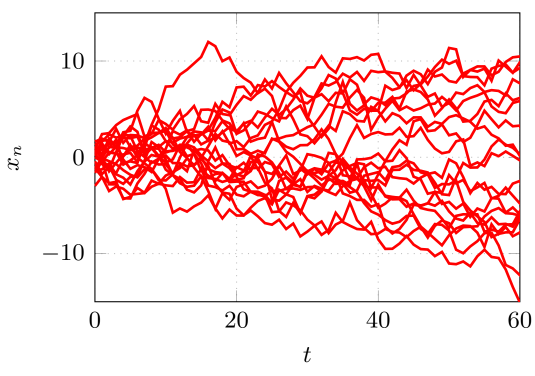
with $\boldsymbol{q}_n \sim p(\boldsymbol{q}_n)$ and $\boldsymbol{r}_n \sim p(\boldsymbol{r}_n)$. Recall that the dynamic model is a stochastic process, meaning that every time a system is observed, a new realization of that stochastic process is observed. For example, consider the scalar (one-dimensional) random walk dynamic model

$$x_n = x_{n-1} + q_n$$

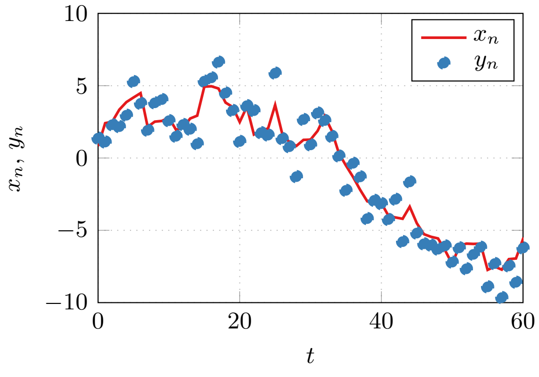
with

$$\begin{aligned} x_0 &\sim \mathcal{N}(0, 1), \\ q_n &\sim \mathcal{N}(0, 1). \end{aligned}$$

This model essentially states that the initial state is a random variable distributed around $x_0 = 0$ with variance 1 and as time elapses, random steps from that initial point are taken. Then, at any time t_n, there are infinitely many random steps that can be taken through q_n (i.e., q_n can take on any value, but with some values more likely than others



(a)



(b)

Figure 5.1: Random walk process example: (a) 20 realizations of the random walk process, and (b) one realization of the process together with its measurements.

as specified by the distribution of the random variable). This implies in turn that no two realizations of any random process are the same. As an example, Figure 5.1a illustrates 20 realization of the random walk process. As it can be seen, each realization takes a completely different path, and the realizations diverge as time elapses.

In practice, when estimating a system's state, all the observations (measurements) come from one single realization out of all the infinitely possible realizations of the process. The measurements then give us the information about the particular realization that is observed and the realization and the measurements are linked through the measurement model. For example, Figure 5.1b illustrates one particular realization of the random walk process together with the measurements from the linear measurement model

$$y_n = x_n + r_n$$

with $r_n \sim \mathcal{N}(0, 1)$.

This leads to a new strategy for filtering where we first simulate a set of trajectories and then evaluate how well each of these trajectories explain the measurements that we are observing. More formally, in the framework of the prediction and measurement update steps introduced in Section 5.1, this approach alternates between:

1. Prediction: Given a set of simulated states $\boldsymbol{x}_{n-1}^j$ (for $j = 1, \dots, J$), simulate from t_{n-1} to t_n to obtain a set of simulated states $\boldsymbol{x}_n^j$ $(j = 1, \dots, J)$;

2. Measurement update: Evaluate how well the simulated states $\boldsymbol{x}_n^j$ explain the observed measurement $\boldsymbol{y}_n$.

For example, Figure 5.2 combines the different realizations of the state trajectories in Figure 5.1a (gray) with the measurements (blue dots) in Figure 5.1b (together with the true trajectory that generated the measurements in red). As it can be seen from

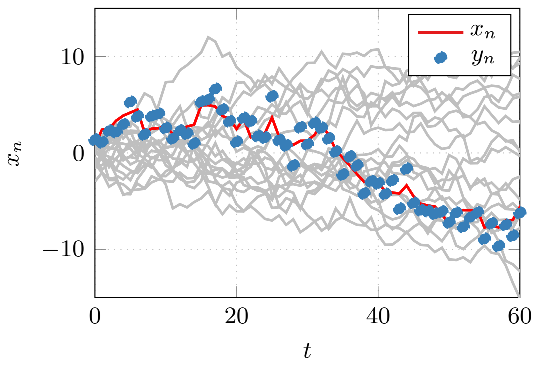


Figure 5.2: Random walk example with measurements y_n (blue) from one particular realization of the random walk process (red) and 20 other realizations.

the figure, several of the gray realizations could be the trajectories that generated the measurements in blue, at least at some points in time, whereas other realizations are very unlikely to be the trajectories that generated the measurements.

To develop a filtering algorithm based on this reasoning, we need to find a strategy to a) simulate samples $\boldsymbol{x}_n^j$ given the samples $\boldsymbol{x}_{n-1}^j$, and b) evaluate how well a particular sample $\boldsymbol{x}_n^j$ of the state explains the current measurement $\boldsymbol{y}_n$.

An intuitive way to generate new samples is to use the dynamic model to simulate one time step. In other words, we can generate a realization of the random variable $\boldsymbol{q}_n$ and use the nonlinear function $f(\boldsymbol{x}_{n-1})$ to pass the sample from t_{n-1} to t_n. This means performing the following two steps for each sample $j = 1, \dots, J$:

1. Sample $\boldsymbol{q}_n^j \sim p(\boldsymbol{q}_n)$,
2. Calculate $\boldsymbol{x}_n^j = f(\boldsymbol{x}_{n-1}^j) + \boldsymbol{q}_n^j$.

In practice, the process noise $\boldsymbol{q}_n$ is often Gaussian, that is, $\boldsymbol{q}_n$ is a Gaussian random variable (e.g., when the discrete-time model comes from the discretization of a continuous-time model with a white noise process as the input). In that case, sampling $\boldsymbol{q}_n$ amounts to sampling from the normal distribution $\mathcal{N}(0, \boldsymbol{Q}_n)$ where $\boldsymbol{Q}_n$ is the covariance matrix of $\boldsymbol{q}_n$.

To evaluate the importance of each sample with respect to the current measurement $\boldsymbol{y}_n$, we assign a weight w_n^j (called the *importance weight*) to each sample $\boldsymbol{x}_n^j$. Intuitively, the weight should represent how well each sample explains the measurement, and thus, the closer the sample is to the true state, the higher the weight should be. If we ensure that the weights sum to one, that is, if

$$\sum_{j=1}^{J} w_n^j = 1,$$

we can also loosely interpret the weight w_n^j as the probability of the jth sample representing the correct state. The question then is how the weights should be calculated, given the sample $\boldsymbol{x}_n^j$ and the measurement $\boldsymbol{y}_n$. So far, cost functions have been used to evaluate whether a state explains the measurements well. However, in this approach the cost should be low for a state that explains the measurements well, whereas the importance weight should be high and hence, we can not use cost functions[1].

Instead, we have another look at the measurement model. Recall that we have assumed that the measurement model is of the form

$$\boldsymbol{y}_n = g(\boldsymbol{x}_n) + \boldsymbol{r}_n,$$

where $\boldsymbol{r}_n \sim p(\boldsymbol{r}_n)$ is a random variable with probability density function $p(\boldsymbol{r}_n)$. Hence, $\boldsymbol{y}_n$ is a random variable too. Assuming that $\boldsymbol{x}_n$ is given, $\boldsymbol{y}_n$ follows the same distribution as $\boldsymbol{r}_n$ but offset by the constant term $g(\boldsymbol{x}_n)$. We can express this probability density function for $\boldsymbol{y}_n$ as $p(\boldsymbol{y}_n \mid \boldsymbol{x}_n)$ which is read as "the probability density function of $\boldsymbol{y}_n$ given that $\boldsymbol{x}_n$ is known". The probability density function $p(\boldsymbol{y}_n \mid \boldsymbol{x}_n)$ is commonly known as the *likelihood* because it indicates how likely a certain state $\boldsymbol{x}_n$ is when observing the measurement $\boldsymbol{y}_n$. Hence, the likelihood is a suitable measure for how well any given sample $\boldsymbol{x}_n^j$ explains the measurement $\boldsymbol{y}_n$, and we can define the non-normalized weight $\tilde{w}_n^j$ as

$$\tilde{w}_n^j = p(\boldsymbol{y}_n \mid \boldsymbol{x}_n^j).$$

$\tilde{w}_n^j$ is non-normalized because the sum over all weights does generally not fulfill the requirement that the weights should sum to one. This can be ensured by simply dividing each non-normalized weight by the sum of all the non-normalized weights, that is,

$$w_n^j = \frac{\tilde{w}_n^j}{\sum_{i=1}^{J} \tilde{w}_n^i}.$$

In practice, the measurement noise is often assumed to be a zero-mean Gaussian random variable with covariance matrix $\boldsymbol{R}_n$, that is, $p(\boldsymbol{r}_n) = \mathcal{N}(\boldsymbol{r}_n; 0, \boldsymbol{R}_n)$. In this case, the likelihood is also a Gaussian random variable with mean $g(\boldsymbol{x}_n)$ (due to the offset) and covariance $\boldsymbol{R}_n$ (due to the uncertainty introduced by $\boldsymbol{r}_n$). Hence, the likelihood becomes

$$p(\boldsymbol{y}_n \mid \boldsymbol{x}_n) = \mathcal{N}(\boldsymbol{y}_n; g(\boldsymbol{x}_n), \boldsymbol{R}_n).$$

Once the importance weights have been calculated, a point estimate of the current state and its covariance can be obtained. Similar to the unscented Kalman filter, these are given by the weighted sum of the individual samples $\boldsymbol{x}_n^j$, weighed by the importance weights w_n^j, that is,

$$\hat{\boldsymbol{x}}_{n|n} = \sum_{j=1}^{J} w_n^j \boldsymbol{x}_n^j, \tag{5.41a}$$

$$\boldsymbol{P}_{n|n} = \sum_{j=1}^{J} w_n^j (\boldsymbol{x}_n^j - \hat{\boldsymbol{x}}_{n|n})(\boldsymbol{x}_n^j - \hat{\boldsymbol{x}}_{n|n})^{\mathsf{T}}. \tag{5.41b}$$

[1] In practice, cost functions are closely related to the way the importance weights are calculated, but this is beyond the scope of this course.

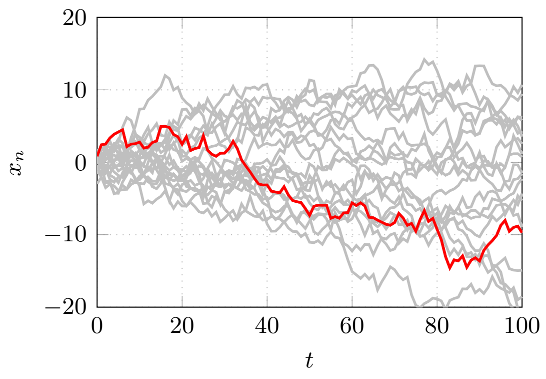


Figure 5.3: Illustration of the divergence problem when simulating trajectories.

Observe that the covariances for the process noise ($\boldsymbol{Q}_n$) and measurement noise ($\boldsymbol{R}_n$) do not enter these equations (contrary to the Kalman-filter-type algorithms). This is due to the fact that this uncertainty is accounted for when sampling and calculating the weights.

It would appear that this now yields a working algorithm. However, there is one important problem, illustrated for the random walk model in Figure 5.3: Assuming that the red trajectory represents the true realization of our state trajectory, the other simulated trajectories start to diverge from that trajectory as time elapses. Consequently, toward the end of the time scale, there are only very few trajectories in the neighborhood of the actual trajectory. In practice, this means that the weights for almost all but a few trajectories would become zero and eventually, after some more time has elapsed, there would be no trajectory close to the true realization anymore and the filter would break down completely.

To address this problem, one additional step called *resampling* has to be introduced. The basic idea of resampling is to make sure that samples with low weights, that is, samples that are unlikely to be close to the true state, are discarded and replaced with copies of the samples with high weight. Hence, resampling essentially regenerates the sample $\boldsymbol{x}_n^j$ such that there are a total of approximately $\lfloor w_n^j J \rceil$ copies of $\boldsymbol{x}_n^j$ in the resampled set of particles. Thus, if $\tilde{\boldsymbol{x}}_n^i$ (for $i = 1, \dots, J$) denotes the resampled particle, that particle is equal to particle $\boldsymbol{x}_n^j$ with probability w_n^j, that is, we have that

$$\Pr\{\tilde{\boldsymbol{x}}_n^i = \boldsymbol{x}_n^j\} = w_n^j,$$

where $\Pr\{\cdot\}$ denotes the probability. This ensures that trajectories with low weight are discarded and that the samples remain close to the true state.

The final bootstrap particle filtering algorithm then consists of the following three steps:

1. Propagate the particles using the dynamic model,

2. calculate the importance weights using the measurement model, and

3. resample the particles according to their weights.

These three steps are then repeated for the complete dataset and the recursion is initialized by sampling from the initial distribution of the state $p(\boldsymbol{x}_0)$. Assuming Gaussian distributions for the initial state, process noise, and measurement noise, that is,

$$\begin{aligned}
\boldsymbol{x}_0 &\sim \mathcal{N}(\boldsymbol{m}_0, \boldsymbol{P}_0), \\
\boldsymbol{q}_n &\sim \mathcal{N}(0, \boldsymbol{Q}_n), \\
\boldsymbol{r}_n &\sim \mathcal{N}(0, \boldsymbol{R}_n),
\end{aligned}$$

the resulting algorithm is as summarized in Algorithm 5.4.

Algorithm 5.4 Bootstrap Particle Filter (Gaussian Process and Measurement Noises)

```
1: Initialization: Sample ($j = 1, \dots, J$)
       $\boldsymbol{x}_0^j \sim \mathcal{N}(\boldsymbol{m}_0, \boldsymbol{P}_0)$
2: for $n = 1, 2, \dots$ do
3:     for $j = 1, 2, \dots, J$ do
4:         Sample
               $\boldsymbol{q}_n^j \sim \mathcal{N}(0, \boldsymbol{Q})$
5:         Propagate the state
               $\boldsymbol{x}_n^j = f(\boldsymbol{x}_{n-1}^j) + \boldsymbol{q}_n^j$
6:         Calculate the importance weights
               $\tilde{w}_n^j = \mathcal{N}(\boldsymbol{y}_n; g(\boldsymbol{x}_n^j), \boldsymbol{R}_n)$
7:     end for
8:     Normalize the importance weights ($j = 1, \dots, J$)
           $w_n^j = \frac{\tilde{w}_n^j}{\sum_{i=1}^J \tilde{w}_n^i}$
9:     Calculate the mean and covariance
           $\hat{\boldsymbol{x}}_{n|n} = \sum_{j=1}^J w_n^j \boldsymbol{x}_n^j$
           $\boldsymbol{P}_{n|n} = \sum_{j=1}^J w_n^j (\boldsymbol{x}_n^j - \hat{\boldsymbol{x}}_{n|n})(\boldsymbol{x}_n^j - \hat{\boldsymbol{x}}_{n|n})^\mathsf{T}$
10:    Resample such that
           $\Pr\{\tilde{\boldsymbol{x}}_n^i = \boldsymbol{x}_n^j\} = w_n^j$
11: end for
```

Bibliography

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization.* Cambridge University Press.

Doucet, A. and Johansen, A. M. (2011). A tutorial on particle filtering and smoothing: Fifteen years later. In Crisan, D. and Rozovskii, B., editors, *Handbook of Nonlinear Filtering*, volume 12 of *Oxford Handbooks*, pages 656–704. Oxford University Press, Oxford, UK.

Gordon, N. J., Salmond, D. J., and Smith, A. F. M. (1993). Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *Radar and Signal Processing, IEE Proceedings F*, 140(2):107–113.

Gustafsson, F. (2012). *Statistical Sensor Fusion.* Studentlitteratur, 2 edition.

Julier, S. J. and Uhlmann, J. K. (2004). Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME, Journal of Basic Engineering*, 82(1):35–45.

Kay, S. M. (1993). *Fundamentals of Statistical Signal Processing: Estimation Theory.* Prentice Hall, Upper Saddle River, NJ, USA.

Levenberg, K. (1944). A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics*, 2(2):164–168.

Marquardt, D. (1963). An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441.

Nielsen, H. B. (1999). Damping parameter in Marquardt's method. Technical report, Technical University of Denmark.

Papoulis, A. (1984). *Probability, Random Variables, and Stochastic Processes.* McGraw-Hill.

Petersen, K. B. and Pedersen, M. S. (2012). The matrix cookbook. Technical report, Technical University of Denmark.

Särkkä, S. (2013). *Bayesian Filtering and Smoothing.* Cambridge University Press, Cambridge, UK.

Särkkä, S. and Solin, A. (2018). *Applied Stochastic Differential Equations.* Cambridge University Press, Cambridge, UK.

Wan, E. A. and Van Der Merwe, R. (2000). The unscented Kalman filter for nonlinear estimation. In *Adaptive Systems for Signal Processing, Communications, and Control Symposium (AS-SPCC)*, pages 153–158.

Øksendal, B. (2010). *Stochastic Differential Equaitons: An Introduction with Applications.* Springer, 6 edition.