Fundamentals of Sensor Fusion

Target tracking, navigation and SLAM

Edmund Brekke

HTTPS://WWW.NTNU.EDU/EMPLOYEES/EDMUND.BREKKE

First edition, October 13, 2019

Contents

1. Overview of tracking, navigation and SLAM

Wikipedia defines sensor fusion as *combining of sensory data or data derived from disparate sources such that the resulting information has less uncertainty than would be possible when these sources were used individually*. Depending on the application, this entails different levels of interpretation of the data. Sensor fusion is closely related to *state estimation*, as the goal is typically to estimate a state vector for an underlying dynamical system based on noisy measurements.

In the context of state estimation, there are two fundamental approaches to sensor fusion. On the one hand, it is possible to process measurements from the different sensors as they arrive, updating the state estimate every time this happens. This is known as *measurement level fusion*. On the other hand, it is possible to maintain different estimates for the different sensors, and then fuse them into a combined estimate. This latter approach is in the context of target tracking known as *track-to-track fusion*. In this book, the main focus will be on the former approach. Therefore, this is not really a book about sensor fusion, but rather about state estimation. However, the estimation problems that dominate in typical sensor fusion applications have many things in common which distinguish them from other estimation problems, and and a solid understanding of applied state estimation is essential as a fundament for more advanced work in sensor fusion.

1.1 Target tracking, navigation and SLAM

In this book the focus will be on three sensor fusion applications: Target tracking, inertial navigation and simultaneous localization and mapping (SLAM). In target tracking the goal is to estimate the motion of one or several moving objects by means of sensors that observe these objects. Such sensors can include cameras, radar, laser scanners and sonar. In inertial navigation the goal is to estimate the motion of a moving object by means of sensors attached to that object. This includes inertial sensors such as an accelerometer and a gyroscope, and sensors such as GNSS and compass which measure location and orientation relative to an external coordinate system. In SLAM the goal is again to estimate own motion as in inertial navigation, but it is also a goal to build a map of the environment. For this reason, SLAM must also use imaging sensors such as those used in target tracking.

We shall refer to the sensors used in target tracking as *exteroceptive*, while we shall refer

to the sensors used in inertial navigation as *interoceptive*. We may also refer to exteroceptive sensors as imaging sensors. While it perhaps is an abuse of terminology to describe GPS as interoceptive, this terminology makes it easy to distinguish the two categories of sensors. The processing of interoceptive and exteroceptive data is radically different. Interoceptive data (e.g., a GPS measurement) can be fed directly to a Kalman filter, which then will estimate position and other kinematic quantities[1]. For exteroceptive data, which come as images or scans, we must first extract relevant features by means of detection and segmentation procedures. This will typically give a list of blobs, which can be reduced to point features by means of clustering procedures. To be able to estimate anything from these features, it is necessary to establish correspondences between features in consecutive scans. This is known as *data association*.

Consequently, the processing pipeline in target tracking and SLAM is considerably more complex than in inertial navigation. but it is also important to understand that solutions to the different applications are dictated by rather different requirements. Inertial navigation may seem like the simplest problem since it does not involve data association, but the requirements of accuracy and robustness are generally very strict. Therefore inertial navigation uses more sophisticated models, and clever choices of estimator design are needed. On the other hand, target tracking tends to use extremely simple motion models, since it would be futile and even dangerous to attempt to estimate more complex models. While a typical navigation system uses 16 or more dimensions in the state vector, most tracking methods use only 4 or 5 dimensions in the state vector.

In inertial navigation it is often desired to make solutions with global stability properties. Such properties provide a guarantee that the filter will not diverge. For target tracking, where all reasonable motion models are marginally stable, it is impossible to guarantee stability. Indeed, if the system is allowed to run for long enough time, divergence is bound to happen. The best safeguard against this is ensure that all plausible hypotheses for data association are considered. For this reason, the computational complexity of target tracking will typically be substantially higher than for inertial navigation.

For SLAM, data association is also an issue that must be taken seriously, but it is less critical than for target tracking. The reason for this is that SLAM typically uses hundreds and thousands of landmarks in the environment. Even if the method fails to associate a large percentage of these, the remaining landmarks are still likely to carry enough information to support estimation of own motion. However, the large number of landmarks is also something that drives computational complexity up. A key research topic in SLAM has therefore been to exploit structure inherent in the SLAM problem to enable fast state estimation.

1.2 A brief history of sensor fusion

Research in target tracking has historically been driven by military applications. The radar was developed just before and during the early phases of World War 2. Most of the tracking algorithms currently in use were developed during the cold war. These were typically designed for guiding antiaircraft artillery or antisubmarine torpedoes. Variations of these methods have dissipated into virtually any defence application where the goal is to shoot someone or something.

Target tracking found its first civilian applications in air traffic control (ATC) and as a maritime navigation aid, where it is known as automatic radar plotting aid (ARPA). The main purpose of these systems is to enable human operators to prevent collisions, by keeping track of the whereabouts of other airplanes and ships, as well as unwelcome intruders such as drones. These systems have undergone progressive steps towards autonomy. Since the Überlingen aircraft collision in 2002 pilots have been instructed to trust their automatic collision avoidance system over human instructions.

[1] Wild-point filtering should of course be used to prevent obviously erroneous measurements to do any damage.

Target tracking also plays a role in all kind of surveillance applications. This can be surveillance of secured areas where one wants to know if unwelcome intruders are passing through or loitering. It can be biological or industrial applications where one wants to follow the motion of fish, bacteria, insects, etc. Tracking methods can also be used to keep track of space debris that poses a threat to satellites or astronauts.

In the same way as for target tracking, early applications of inertial navigation also emerged in World War 2 due to military needs. Long-range missiles such as the German V2 rockets needed a navigation system to hit their targets. The development of inertial navigation continued with the more advanced missiles developed during the cold war. At the same time, the space race between the United States and the Soviet Union also led to the usage of advanced navigation technology in spacecraft and satellites. The moonlanding is often credited as the first application of the Kalman filter. Since then, inertial navigation has become commonplace technology in all kinds of vehicles, and also electrical gadgets such as smartphones. The next stepping stone was the introduction of Global Navigation Satellite Systems (GNSS), including the Global Positioning System (GPS), which since 1995 has since then provided a global reference frame for navigation systems, so that location anywhere on the earth can be estimated with an accuracy of a handful of meters, or possibly centimeters if differential techniques are used.

However, access to satellites is limited in many situations, such as for underwater vehicles or indoor robotics. This limitation has since the late 90s been a major driving force for the development of SLAM. Using SLAM, a robot can estimate its own location relative to its local environments, such as the rooms in a building. SLAM also has roots in computer vision, virtual reality and 3-D reconstruction.

The primary motivation behind the course TTK4250 Sensor Fusion is current research on autonomous vehicles. Since the DARPA Grand Challenge was won by the driverless car Stanley in 2005 it has been clear that autonomous cars can successfully cope with fairly complex environments, and the automotive industry has since then been putting huge efforts into the development of autonomous cars. Both target tracking, inertial navigation and SLAM are essential components of the software for autonomous cars. Norway does not have a large automotive industry, but its maritime industry is world-leading. Increased levels of autonomy are expected to have a huge impact on this industry in the near future.

A secondary motivation behind TTK4250 Sensor Fusion is that estimation in general plays an increasing role both in cybernetics and related disciplines. A general trend is that as simpler problems are being solved, researchers and innovators move on to tackle more challenging problems. The simpler problems are in this context those that can be solved by feedback control alone, possibly including some filtering or observers to process sensor data. The more challenging problems are those that involve a more complex interpretation of the data, and where the control system consequently must perform some kind of reasoning in the presence of uncertainty. The fundamental tools such as Kalman filters, particle filters, Gaussian mixtures and graphical models appear again and again in such applications. The more general management of uncertainty is highly relevant in several other fields, including subsurface imaging for oil and gas exploration, medical imaging, meterology and climate science, system identification of physical and biological processes and financial markets, to mention a few areas.

1.3 Modeling - important or not so important?

Imagine that you hear the roar of an F16 somewhere to the west. Soon after you hear it right above you. Instinctively, you then try to spot it to your east, where you already can see it fly into the horizon. In this process, you utilize several models. First, you have a *plant model*, also known as process model or kinematic prior, which tells you that the plane probably continues more or less in the same direction as it moved earlier. Second, you have a *measurement model*, which in this

particular example would account for the fact that the airplane probably was in the direction from which the sound was emitted when the sound was emitted, and which also would account for the fact that the speed of sound is not that much higher than the speed of the airplane.

It is fairly evident that models have a central role to play in sensor fusion. For the choice of models, the Einstein maxim applies: "As simple as possible, but not simpler". Theoretically, we could make a model of the F16 plane which accounts for its motion in 6 degrees of freedom, including various dynamical forces. Unfortunately, there is no way that we can estimate all the state variables of such a model. Not only would it be meaningless to use such a model, but it would most certainly be quite disastrous, as the inevitable failures to estimate some of the states very soon will cause the track to diverge significantly from the truth. In contrast, most tracking methods use models assuming nearly constant velocity (the CV model) or nearly constant turn rate (the CT model).

The modeling task does not necessarily stop with such kinematic models, e.g., the kind of models that are used in a Kalman filter. If you see several airplanes instead of only one, then you are forced to evaluate which of these is that particular F16 plane. If you do not see the plane again, perhaps you will consider the possibility that the roar was from a meteorite fall from the sky instead. However, you are in possession of some vague prior knowledge about how frequently F16s and meteorites fly above your head, and this knowledge would probably make you extremely reluctant to accept this alternative explanation, and you would still search for the F16 in the sky. Especially in target tracking, this kind of knowledge is often encoded in models for target existence, false alarms and misdetections. Even if all these models on their own are very simple, the overall tracking problem, where all the models are combined, will quickly become rather complex.

It is important that models make sense and exhibit sufficient degree of realism, especially when expanding modeling from basic kinematic properties to more exotic properties. Sometimes one may discover that an estimation method that in theory should work extremely well, fails in any realistic scenario because its modeling assumptions are entirely unrealistic. Also, adapting a tested and tried method to a new problem may fail to be successful if that problem has characteristics which violate the assumptions of that method.

Machine learning methods are often viewed as an alternative to model-based methods. This is a valid point of view, although with two important caveats. It cannot be emphasized strongly enough that the term machine learning is a wide umbrella that covers several loosely related methods. As the first caveat, so-called probabilistic machine learning methods do involve explicit models, which are to be learned from the data. As the second caveat, artificial neural networks are fundamentally nonlinear regression engines which encode relationships between inputs and outputs in a black box fashion. In principle there are no limits to the relationships that such an engine can learn. It could for example learn the relationship between GPS and IMU measurements and the true state vector in a navigation system. But this relationship can be viewed as an implicit representation of the kinematic model. The question then is whether the learned model will give better predictive power than a model specified manually from prior knowledge. For complex problems such as image recognition this is probably the case, while it is much more questionable for, e.g., inertial navigation using GPS and IMU.

1.4 The deterministic and probabilistic paradigms

Uncertainty can be dealt with in many ways. First of all, we may ignore it alltogether. Many classical estimation techniques exist which do not actually utilize models of uncertainty. In standard unweighted least squares estimation one simply aims to find the underlying parameter that minimizes the average error that results between observations and predicted observations. As another example, to match two images one may extract some features in both images, and attempt to find a transformation that makes as many of the features as possible coincide.

State estimation is often referred to as filtering, and state estimators are often known as filters (e.g., the Kalman filter). Filtering is a fairly general concept. One can design filters which do not correspond to a meaningful estimators, and one can design filters that estimate a state without any explicit treatment of uncertainty. Instead, a filter is typically designed by specifying requirements on the spectral properties of the estimation error. For example, in dynamical positioning the goal is to estimate long term motion of a vessel, while short term disturbances due to sea waves should be ignored. This way of thinking leads naturally to the theory of linear and nonlinear observers, whose tuning parameters (gains) are tuned according to the desired response time of the observer. The designer of the observer will typically present a stability proof to convince that the observer can be relied upon.

The conventional approach in sensor fusion is, however, to quantify uncertainty by means of probabilities. This is quite natural, as probability has been the established language of uncertainty since the time of Laplace. There are several reasons why we would like to quantify uncertainty, as opposed to merely coping with uncertainty. First of all, established estimation algorithms such as the Kalman filter have been developed in a probabilistic context. Furthermore, quantifying the uncertainty is useful to help us with interpreting the results. Returning to the F16 example, having an uncertainty estimate makes it possible for you to decide how much you should move your gaze around the spot where you expect to see it. As a third reason, quantified uncertainty enables us to make precise rules for how to weigh different pieces of evidence against each other in a (multi-) sensor fusion system.

Such rules may be perceived as a straightjacket or as the beams of a house construction. This book will attempt to convince you towards the latter perception. It is extremely important to have sensible structures to rely upon when designing complex sensor fusion systems. At the core of these structures we often find Bayes' rule, which is a very powerful tool, because it governs how information changes in the presence of new evidence. Proving that a sensor fusion method obeys Bayes' rule is in some sense the equivalent of stability proofs in control theory. The virtue of a stability proof is not necessarily that the system will converge to zero error. If noise continuously enter the system, that will not happen. The stability proof may, however, ensure that the system will not do anything crazy. Similarly, if a sensor fusion method is faithful to Bayes, we can trust it to behave in a rational manner.

1.5 References and rationale

Several excellent books on estimation and sensor fusion exist. Why was it then necessary to write a textbook from scratch for the course TTK4250? The answer has to do with the limitations of how much knowledge a student can be expected to digest during a 7.5 study points course. In TTK4250 the goal is to reach sufficient comprehension of state-of-the-art methods for tracking, navigation and SLAM to be able to use these from day one in research problems in the 5th year specialization projects and MSc theses. Furthermore, this goal is to be achieved from a foundation consisting of little more than a basic probability course and a basic Kalman filtering and estimation course (i.e., linear systems theory). There exists no textbook today that covers this entire span. Nevertheless, I am convinced that it is possible to cover this in a one-semester course, if there is a strong focus on the core methods and their theoretical foundations, and less focus on general theory or the many possible variations of the methods.

If this course was to use an existing textbook, the most natural candidates would have been the 2001 book by Bar-Shalom, Li & Kirubarajan [4] or the 2012 book by Gustafsson [48]. Both cover the fundamentals of probability and estimation extremely well, and include authoritative in-depth treatments of Kalman filtering, several nonlinear methods (such as EKF) and multiple model estimation. Any student who wants to do serious work in sensor fusion should own a copy of at least one of these. There are, however, two reasons why I have chosen not to use any of these as

curriculum for TTK4250. Both books are very rich on details, and I am concerned that the amount of details will simply be too much for many students to digest. Furthermore, neither book covers data association, which in my opinion is one of the most important topics in sensor fusion.

Fundamental probability and estimation theory is covered in Chapters 2 and 3, but only to a limited extent. For more details the reader is referred to Kay's book [59] from 1993 and to Papoulis & Pillai's textbook on probability and stochastic processes from 2002 [85]. Again, any serious sensor fusion student should be familiar with these books. While [59] is very readable and takes a discrete-time approach, [85] is the definitive reference on the theory of continuous-time stochastic processes.

For target tracking, I am first and foremost inspired by Bar-Shalom and Li, represented by the book [3], which was written in 1995. A more expanded version of this book came in 2011 with the title title "Tracking and Data Fusion: A Handbook of Algorithms" [6]. The most comprehensive textbook on target tracking that exists is probably [11]. It is indispensable for diving into the building blocks of MHT implementations, but cannot be used as an authoritative source on the state-of-the-art with its 20 years of age. To become familiar with the modern paradigm in target tracking the reader would have to supplement the mentioned books with Challa [21] and Mahler [71] (or the research articles that these books build upon). In [21] methods that use the concept of existence probability are described. The book is to an even larger extent than [6] written as a handbook of algorithms. In [71], the focus is on Mahler's theory of random finite sets. While Mahler has made significant efforts to present this material in a pedagogical manner, the material itself is far too demanding to be used in a basic course on sensor fusion.

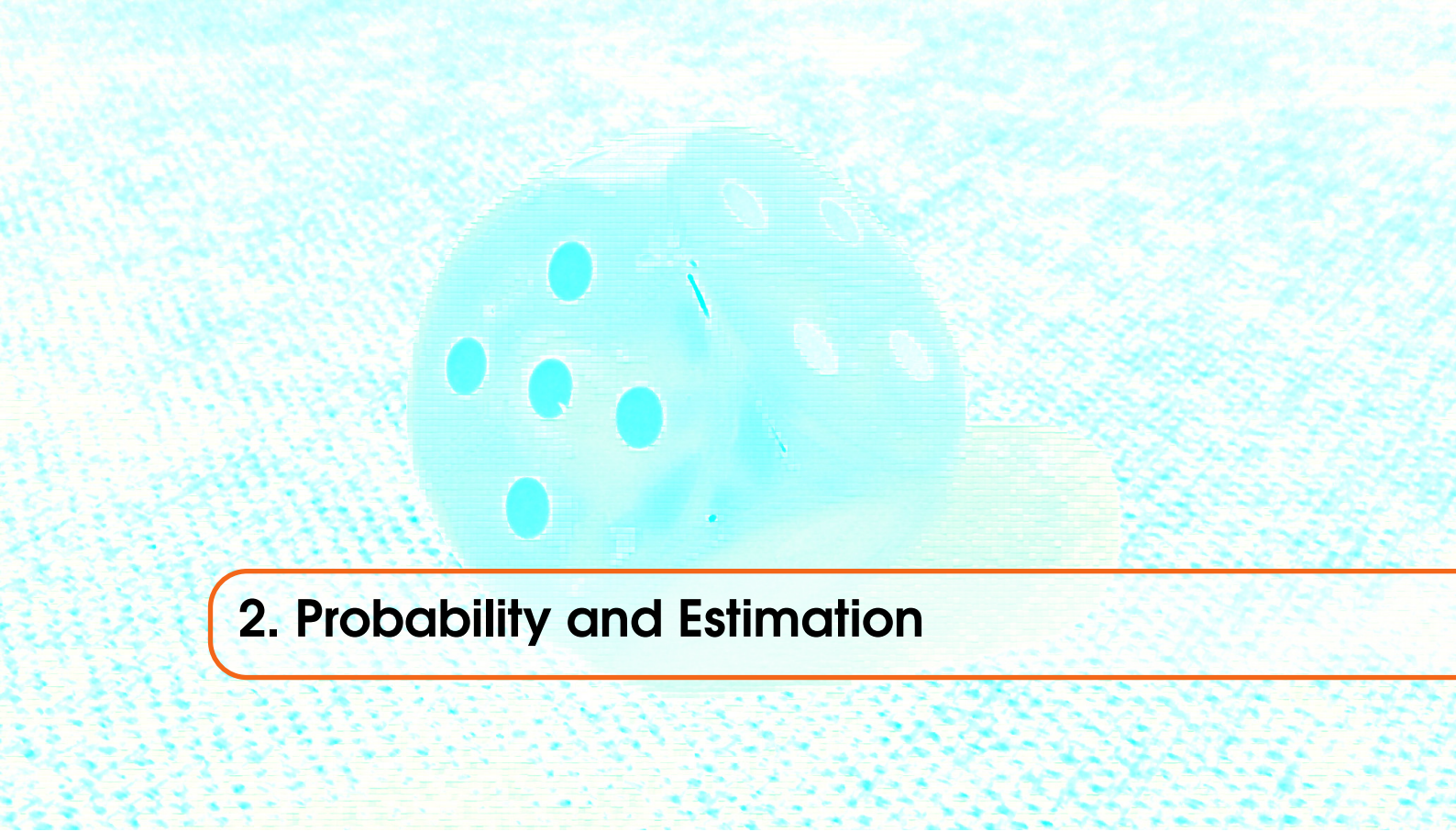
As for navigation, established textbooks include the books by Groves [46] and Titterton & Weston [105]. These delve much deeper into the fundamentals of GNSS technology etc. than we would like to do in this course, while their exposition of the error state formulation of the Kalman filter is much less systematic. Inertial navigation has previously been taught in the course TTK5 using Vik's book [109]. Again, the main shortcoming is no systematic exposition of the error state formulation. This is, however, available in Sola's text [100], which forms the basis for our treatment of inertial navigation.

There exists no textbook on SLAM that is up to date with the significant developments that have found place during the last 10 years. The closest one gets to a canonical textbook on SLAM remains Thrun's book [104] which is from 2006. It gives a very detailed treatment of EKF-SLAM and particle filtering SLAM. Recursive SLAM methods, together with the related topic of localization, are also covered in [48]. While not primarily about SLAM, Barfoot's book [7] may be very relevant for anyone working on SLAM and navigation.

1.6 Acknowledgement

This book is largely inspired by my experience in supervising MSc and PhD students working on autonomous ship projects during the last 5 years. All of these students can be found on the website `folk.ntnu.no/edmundfo/autoseastudents/autoseastudents.html`. In particular, I would like to express special appreciation to Lars-Christian Tokle, who has contributed ideas, proofreading and some of the central proofs, and Michael Ernesto Lopez who has made several of the TiKZ-coded figures. The book has been written using Texmaker and the Legrand Orange Book template (`www.latextemplates.com/template/the-legrand-orange-book`). During the writing I upgraded my macbook to a 2018 model with butterfly keys. I blame all typos on the butterfly keys.

Finally the reader should be aware that this is still work in progress. As of 14th of August, only the first three chapters are completed and will be released on Blackboard. Additional chapters will be released as soon as they are completed.



2. Probability and Estimation

2.1 Foundations of probability theory

Probability theory is about assigning numbers to *events*. These numbers, known as probabilities, tell us something about how likely these events are to happen. This intuitive notion of probability can be expressed more precisely in terms of three axioms. Let the letter E refer to any event, and let Ω be the outcome space, that is, the collection of all possible events. We use the notation $\Pr\{\cdot\}$ to signify probability of the events. The axioms of probability can then be formulated as follows

1. $\Pr\{E\} \in \mathbb{R}$, $\Pr\{E\} \geq 0$.
2. $\Pr\{\Omega\} = 1$.
3. For any sequence of disjoint events $E_1, \ldots, E_n$ we have $\Pr\{\bigcup_{i=1}^{n} E_i\} = \sum_{i=1}^{n} \Pr\{E_i\}$.

The first axiom then tells us that the probability of any event is a non-negative real number. The second axiom tells us that the probability of the entire outcome space is one. The third axiom tells us that the probabilities of disjoint events can be added in order to find the probability that any of the events considered should happen.

In addition to the axioms, probability theory also relies on some fundamental definitions, which provide a language for how different events can be related to each other.

Definition 2.1.1 — Conditional probability. Given two events A and B, the conditional probability of A given B is

$$\Pr\{A|B\} = \frac{\Pr\{A \cap B\}}{\Pr\{B\}} \tag{2.1}$$

Definition 2.1.2 — Independence. We say that two events A and B are independent if

$$\Pr\{A \cap B\} = \Pr\{A\}\Pr\{B\} \tag{2.2}$$

Equipped with these definitions, we can establish some basic results in probability theory. The following two results will play a key role in very many of the derivations in the subsequent chapters.

Theorem 2.1.1 — The total probability theorem. Let $\{B_n\} = \{B_1, B_2, B_3, \ldots\}$ be a countable partition of the outcome space, and let A be some event. Then the following is true:

$$\Pr\{A\} = \sum_n \Pr\{A|B_n\}\Pr\{B_n\}.$$

Proof. The probability of A is the same as the probability of the union $\bigcup_n (A \cap B_n)$ insofar as $\{B_n\}$ is a partition of the outcome space. Thus

$$\Pr\{A\} = \sum_n \Pr\{A \cap B_n\} = \sum_n \Pr\{A|B_n\}\Pr\{B_n\}$$

where the first equality follows from axiom number 3 and the last equality follows from the definition of conditional probability. ■

Theorem 2.1.2 — Bayes' rule. For any two events A and B, the following is true:

$$\Pr\{A|B\} = \frac{\Pr\{B|A\}\Pr\{A\}}{\Pr\{B\}}.$$

Proof. According to the definition of conditional probability the joint probability of A and B is $\Pr\{A \cap B\} = \Pr\{A|B\}\Pr\{B\} = \Pr\{B|A\}\Pr\{A\}$. Bayes' rule follows from dividing by $\Pr\{B\}$ on both sides. ■

2.2 Random variables and probability distributions

It would be very cumbersome if we always were to work explicitly with events. In probability theory and its applications we are typically interested in the behavior of quantities that are of a random nature. By the phrase "of a random nature" we mean that the quantity is uncertain, and that our knowledge about it is modeled by means of probability theory. Such a quantity could for example be the speed of an airplane. In a probabilistic model we would have some probability that it is smaller than $100\,\mathrm{m/s}$, while the probability that it is larger than or equal to $100\,\mathrm{m/s}$ would be one minus that probability. Statements such as "the velocity is less than $100\,\mathrm{m/s}$" correspond to the events in Section 2.1.

2.2.1 Random variables and probability measures

Random variables are in the mathematical approach to probability theory defined as functions from an abstract outcome space to a more concrete outcome space such as $\mathbb{R}^n$. For any element in the abstract space the random variable attains a certain element in the concrete space, known as the *realization* of the random variable. The formal definition of a random variable is based on measure theory, which is one of the core foundations of mathematical analysis (other foundations being algebra and topology). The conceptual framework revolves around the concept of a *probability space*, which again builds on the concepts of σ*-algebras* and *measures*.

Definition 2.2.1 — σ-algebra. A σ-algebra $\mathscr{F}$ on a set Ω is a collection of subsets of Ω such that

- Ω itself is a member of $\mathscr{F}$.
- If the subset $A \subseteq \Omega$ is a member of $\mathscr{F}$, then its complement $\Omega \backslash A$ is also a member of $\mathscr{F}$.
- If A_1, A_2, A_3, etc are members of $\mathscr{F}$, then the union $\bigcup_n A_n$ is also a member of $\mathscr{F}$.

In probability theory, the elements of the σ-algebra plays the role of meaningful events. We can illustrate this with a simple example.

■ **Example 2.1 — Throw of a dice.** The outcome space when we throw a dice is $\Omega = \{1,2,3,4,5,6\}$. All the possible subsets of these 6 elements constitute a corresponding σ-algebra:

$$\mathscr{F} = \{\{1\},\{2\},\ldots,\{1,2\},\ldots,\{1,2,3,4,5,6\}\}.$$

We see that for any outcome at least one, and possibly several members of the σ-algebra will be active. ■

Measures are functions from σ-algebras to the real numbers. In other words, a measure is a systematic way of assigning numbers to subsets. In probability theory we are only concerned with a special class of measures known as probability measures.

Definition 2.2.2 — Probability measure. A probability measure P on the σ-algebra $\mathscr{F}$ is a function from $\mathscr{F}$ to the unit interval $[0,1]$ that obeys the three axioms of probability.

We see that the probability measure by definition obeys the fundamental axioms of probability that we introduced in the beginning of Section 2.1. We refer to the combination of the outcome space, the σ-algebra, and the probability measure as a probability space.

Definition 2.2.3 — Probability space. A probability space is a triplet $(\Omega,\mathscr{F},P)$ where

- Ω is a space of possible events.
- $\mathscr{F}$ is a σ-algebra on Ω.
- $P : \mathscr{F} \to [0,1]$ is a probability measure.

Definition 2.2.4 — Random variable. A random variable X is a function from Ω into another space $\mathbb{O}$, which we henceforth shall know as the outcome space.

The outcome space $\mathbb{O}$ can be many different things. In game of dice it will consist of the possible faces of a dice, which can be represented as $\{1,2,\ldots,6\}$. Most typically, $\mathbb{O}$ will be the space of real numbers $\mathbb{R}$, or it may the n-dimensional space of real-valued vectors $\mathbb{R}^n$. In the latter case, we often talk about random vectors instead of random variables. The random variable X connects the abstract space with the outcome space $\mathbb{O}$, so that probability becomes distributed on $\mathbb{O}$. The probability that the value of X is a member of B, where $B \subseteq \mathbb{O}$, is given by $P(X^{-1}(B))$, i.e., by summing the probabilities of all elements $\omega \in \Omega$ for which $X(\omega)$ ends up in B.

Why is it useful to introduce the abstract space Ω in addition to the more tangible outcome space $\mathbb{O}$? At least two answers can be provided. First, the definition of random variables as functions provides a clear mechanism for distinguishing random variables from their realizations. While the difference is intuitively easy to grasp, it is also nice to have a precise mathematical distinction. Second, the tangible outcome space $\mathbb{O}$ is not necessarily as neat and tidy as $\mathbb{R}^n$. For example, we shall in Chapter 13 let $\mathbb{O}$ consist of all finite subsets of $\mathbb{R}^n$. As another example, $\mathbb{O}$ could be a closed manifold such as $SO(3)$ or the surface of the earth. In such cases it may not be obvious how probability mass should be distributed in $\mathbb{O}$. By defining the random variables as functions we get a mechanism for mapping the probability mass from a space where probabilities by definition make sense to this more exotic space.

Definition 2.2.5 — Realization. The output of the random variable X for a particular $\omega \in \Omega$ is called a realization of X. We can write this as $x = X(\omega)$.

The concepts of a random variable and its realization are often confused, accidentally or deliberately. Often the same notation is used for both. This is often the only sensible thing to do, as notation would quickly become very complicated if we were to distinguish random variables and their realizations all the time. Nevertheless, the distinction should be remembered, as it sometimes can be quite important.

The abstract measure $P(\cdot)$ induces a probability measure on the tangible outcome space $\mathbb{O}$. We can relate this measure to probabilities as follows:

Definition 2.2.6 — Probability measure of a random variable. The probability measure of X is a function $\beta_X(\cdot)$ from subsets of $\mathbb{O}$ to $[0,1]$ so that

$$\beta_X(S) = \Pr\{X \in S\} = P(X^{-1}(S)). \tag{2.3}$$

2.2.2 Probability distributions

The main issue with probability measures, and the equivalent formulation in terms of primitive events in Section 2.1, is that these representations are highly redundant. To see this, and make the conceptual leap over to more useful representations, let us look at scalar continuous-valued random variables, i.e., let $\mathbb{O} = \mathbb{R}$. We do not need to assign a probability value to all possible subsets of $\mathbb{R}$ to specify the random properties of the random variable X. It would suffice to assign a probability value to all subsets of the form $(-\infty, x)$ where x is a real number, i.e., a possible realization of X. Or it would suffice to define a function whose integrals from $-\infty$ to x yields these values. These two viewpoints lead to the concept of cumulative distribution functions and probability density functions.

Definition 2.2.7 — Cumulative distribution function (cdf). The cdf, denoted $P(x)$, of $X \in \mathbb{R}$ is the probability $\Pr\{X < x\}$.

Definition 2.2.8 — Probability density function (pdf). The pdf of the scalar random variable X is the derivative

$$p(x) = \frac{\partial P(x)}{\partial x}.$$

Generally, the term probability distribution is used to mean the same as a pdf. It must be emphasized that *pdf's and probabilities are two different things.* We get probabilities when we integrate a pdf over a subset of the outcome space.

For discrete-valued random variables (e.g., throwing a dice) we do not normally talk about the pdf, but rather about the point mass function (pmf), which is given by $p(x) = \Pr\{X = x\}$. We can obtain the cdf in the same manner as for continuous-valued random variables by replacing integration with summation. Notice that such expressions such as $\Pr\{X = x\}$ in general are meaningless for continuous-valued random variables: The probability of X attaining a particular value will be zero if X is distributed over a continuous space.

The extension of cdfs and pdfs to vector-valued random variables is fairly straightforward. For example, if $Z = [X,Y]^\mathsf{T}$ for scalar X and Y, then the cdf is defined as

$$P_Z(z) = P_{X,Y}(x,y) = \Pr\{X \leq x, Y \leq y\}$$

and the pdf is found as

$$p_Z(z) = p_{X,Y}(x,y) = \frac{\partial^2}{\partial x \partial y} P_{X,Y}(x,y) = \frac{\partial^2}{\partial y \partial x} P_{X,Y}(x,y).$$

It must again be emphasized that the abstract machinery of Section 2.2.1 is much more general than random variables in $\mathbb{R}^n$. More care is needed if one aims to extend the constructions of cdfs and pdfs to exotic state spaces such as closed manifolds.

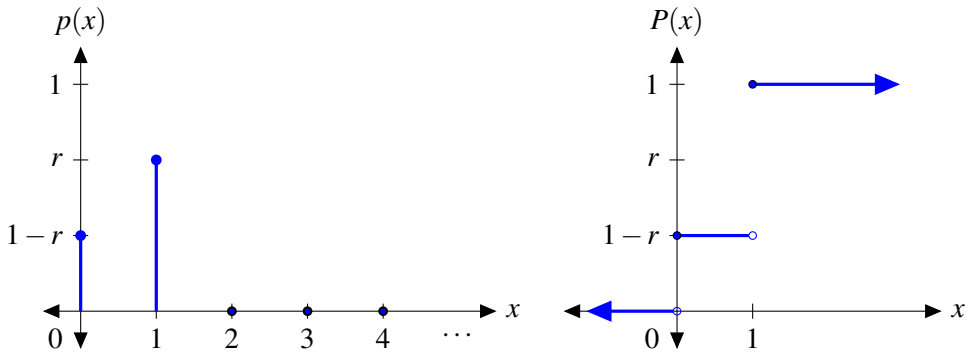
2.2.3 Examples of probability distributions

It is time to look at some examples of random variables that play important roles in sensor fusion. We shall first look at a handful of discrete examples.

■ **Example 2.2 — Bernoulli random variable.** A Bernoulli random variable with parameter r has a binary outcome space: $E = \{0, 1\}$, and its probability distribution is given by

$$p(x) = \begin{cases} 1-r & \text{if } x = 0 \\ r & \text{if } x = 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.4}$$

We write $p(x) = \text{Bernoulli}(x; r)$ to signify this distribution. ■



(a) PMF. (b) CDF.

Figure 2.1: Example of Bernoulli distribution.

■ **Example 2.3 — Binomial random variable.** A binomial random variable with parameters $r \in [0, 1]$ and $n \in \mathbb{N}$ has outcome space $\{0, \ldots, n\}$ and its probability distribution is given by

$$p(x) = \binom{n}{x} r^x (1-r)^{n-x}. \tag{2.5}$$

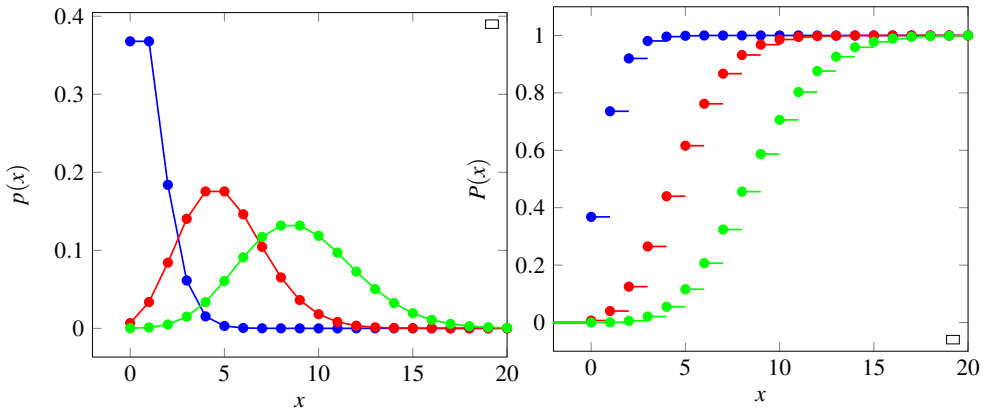
■

■ **Example 2.4 — Poisson random variable.** A Poisson random variable with parameter λ has the countable (that means infinite, but discrete) outcome space $\{0, 1, 2, 3, \ldots\}$ and its probability distribution is given by

$$p(x) = \frac{\lambda^x e^{-\lambda}}{x!} \tag{2.6}$$

We write $p(x) = \text{Poisson}(x; \lambda)$ to signify this distribution. ■

Bernoulli random variables are often used to model whether a target exists or does not exist, or whether a target is detected or not detected. Poisson random variables are often used to model how many false alarms there are in a radar scan. The binomial random variable can be viewed as a generalization of the Bernoulli random variable when there is more than one yes-no question, e.g., two potential targets, or two sensor cells which may or may not contain an observation. According to the Poisson limit theorem, also known as the law of rare events, the binomial distribution can be approximated by the Poisson distribution if n tends towards infinity and r tends towards zero in such a manner that the product nr tends towards λ.



(a) PMF (b) CDF

Figure 2.2: Poisson distribution.

As for continuous random variables, the simplest possible example is the uniform distribution, which for instance often is used in target tracking to model the spatial distributions of false alarms or newborn targets.

■ **Example 2.5 — Uniform random variable.** A uniformly distributed random variable X on the interval $[a,b]$ has the pdf

$$p(x) = \text{Uniform}(x;[a,b]) = \frac{1}{b-a}\chi_{[a,b]}(x) = \begin{cases} \frac{1}{b-a} & \text{if } a \leq x \leq b \\ 0 & \text{otherwise.} \end{cases} \tag{2.7}$$

Its cdf can be written as

$$p(x) = \begin{cases} 0 & \text{if if } x \leq a \\ \frac{x-a}{b-a} & \text{if } a \leq x \leq b \\ 1 & \text{if } x \geq a. \end{cases} \tag{2.8}$$

■

By far the most important class of continuous random variable are Gaussian random variables. In a similar manner as for uniform random variables, the pdf of a Gaussian random variable is given by two parameters. It is possible to parameterize the pdf in different ways.

■ **Example 2.6 — Gaussian random variable.** A Gaussian random variable with expectation μ and variance σ^2 has the pdf

$$p(x) = \mathcal{N}(x;\mu,\sigma^2) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{2.9}$$

This is known as the moment-based parameterization. Another parameterization is the canonical parameterization. If we define $\eta = \mu/\sigma^2$ and $\lambda = 1/\sigma^2$, then the same pdf can also be expressed as

$$p(x) = \exp\left(\alpha + \eta x - \frac{1}{2}\lambda x^2\right) \quad \text{where} \quad \alpha = -\frac{1}{2}\left(\ln(2\pi) - \ln(\lambda) + \eta^2/\lambda\right). \tag{2.10}$$

The cdf of the Gaussian does not exist in closed form. It is typically expressed in terms of the so-called error function according to

$$P(x) = \frac{1}{2}\left[1 + \text{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right] \quad \text{where} \quad \text{erf}(x) = \frac{1}{\sqrt{\pi}}\int_0^x \exp(-t^2)\text{d}t. \tag{2.11}$$

■

The Gaussian distribution is the most well-known example of a distribution from the so-called exponential family. Another important member of this family is the Gamma distribution.

■ **Example 2.7 — Gamma and χ^2 random variables.** A Gamma random variable with shape parameter k and scale parameter θ has the pdf

$$p(x) = \text{Gamma}(x; k, \theta) = \frac{x^{k-1}\exp(-x/\theta)}{\theta^k \Gamma(k)} \tag{2.12}$$

Again different parameterizations are possible, but the parameterization in terms of shape and scale is sufficient for us. Several other exponential-family distributions are intimately linked to the Gamma distribution. Both the Rayleigh distribution, the exponential distribution and the χ^2-distribution are special cases of the Gamma distribution. The exponential distribution $p(x) = \lambda e^{\lambda t}$ arises as a special case of (2.12) when $k = 1$ and $\theta = 1/\lambda$. The Rayleigh distribution $p(x) = \frac{x}{\sigma^2} e^{-x^2/(2\sigma^2)}$ arises as a special case when $k = 2$ and $\theta = 2\sigma^2$. As for the χ^2 distribution, we say that X is χ^2 distributed with n degrees of freedom if

$$p(x) = \frac{1}{2^{n/2}\Gamma(n/2)} x^{n/2-1} \exp\left(-\frac{x}{2}\right). \tag{2.13}$$

We can see that (2.13) is a special case of (2.12) when the scale parameter is equal to 2 and the shape parameter is equal to $\frac{n}{2}$. For general Gamma and χ^2 random variables the cdf is again non-analytic, while closed-form expressions are easily found for Rayleigh and exponential RVs. Nevertheless, it is often important to find such cdf values, and to solve for x when a cdf value is given, and this can easily be done with Matlab functions such as `chi2cdf` and `chi2inv`, respectively. ■

2.2.4 Sampling from probability distributions

To play with random variables, it is very useful to be able to simulate them. Programming languages such as Matlab comes with support or add-on packages which allow the user to draw samples from well-known probability distributions without putting much thought into how the simulation is done. For examples, to draw a 4×1 vector of independent samples from $\mathcal{N}(x; 0, 1)$ we can use the Matlab command

```
randn(4,1).
```

More complicated random variables can often be simulated by exploiting their relationships to other random variables. For example, we shall later see (Example 2.11 on page 27) that the absolute value of such a Gaussian random variable is a χ^2 distributed random variable with one degree of freedom. Thus, we have a means of simulating such χ^2 distributed random variables.

If we are not so lucky to have such a relationship to exploit, we can still use *cdf inversion*, also known as inverse transform sampling or the Smirnov transform. Consider the problem of generating N independent samples of a random variable X with pdf $p_X(x)$. In this approach, we draw a random N numbers u^i from the uniform distribution $\text{Uniform}(u; [0, 1])$. The we look at the cdf $P_X(x)$. Every sample x^i is then found as the value of x^i such that

$$P_X(x^i) = u^i. \tag{2.14}$$

In other words, we find the samples as $x^i = P_X^{-1}(u^i)$. We see that this approach relies fundamentally on our ability to invert the cdf. This may or may not be possible to do in closed form. However, even if no closed-form solution exist, approximations may be good enough. For example, we can approximate the cdf by "segmented" functions, in the shape of a staircase or a C^0 collection of straight lines.

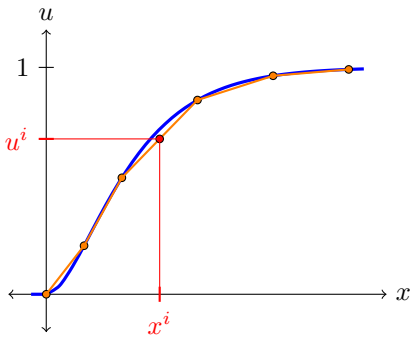


Figure 2.3: Illustration of how the Smirnov transform can be used to sample a random variable even if its inverse cdf has no analytical expression.

2.3 Moments

In many cases, we are not primarily interested in the full pdf of a random variable, but only in certain descriptive measures which give us more tangible information about how the random variable behaves. Among such measures are moments, which are expectations of power of the random variable. The two most well known moments are expectation and variance.

The expectation and variance of a scalar-valued random variable X with realization x are given by the integrals

$$E[X] = \int xp(x)\mathrm{d}x \tag{2.15}$$

$$\mathrm{Var}[X] = E[(X - E[X])^2] = \int (x - E[X])^2 p(x)\mathrm{d}x. \tag{2.16}$$

The notations $\mu = E[X]$ and $\sigma = \sqrt{\mathrm{Var}[X]}$ are commonly used. The generalizations for a vector-valued random variable X with realization $\mathbf{x}$ are given by

$$E[X] = \int \mathbf{x}p(\mathbf{x})\mathrm{d}\mathbf{x} \tag{2.17}$$

$$\mathrm{Var}[X] = E[(X - E[X])(X - E[X])^\mathsf{T}] = \int (\mathbf{x} - E[X])(\mathbf{x} - E[X])^\mathsf{T} p(\mathbf{x})\mathrm{d}\mathbf{x}. \tag{2.18}$$

In the vector case $\mathrm{Var}[X]$ becomes a matrix, and not merely a number. It is known as the covariance matrix and consists of elements of the form

$$\mathrm{Cov}[X_1, X_2] = E[(X_1 - E[X_1])(X_2 - E[X_2])^\mathsf{T}] = \int (x_1 - E[X_1])(x_2 - E[X_2])p(x_1, x_2)\mathrm{d}x_1\mathrm{d}x_2. \tag{2.19}$$

For discrete random variables the integrals are replaced by sums. The expectation and the variance are closely related to the sample mean and the sample variance, which can be calculated for a collection of N samples according to

$$\bar{\mathbf{x}} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_i \tag{2.20}$$

$$\mathbf{P} = \frac{1}{N-1}\sum_{i=1}^{N} (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^\mathsf{T} \tag{2.21}$$

If the samples are drawn from the distribution $p(x)$, then the mean will converge towards its expectation as the number of samples goes towards infinity. Notice that these sample-dependent quantities in contrast to the expectation and variance are random variables. The distribution of the sample mean can in some cases be calculated analytically, while in other cases it may be intractable. However, as the number of samples goes towards infinity we are guaranteed some regularity thanks to the Central Limit Theorem (CLT).

Theorem 2.3.1 — Central Limit Theorem, Lindeberg-Lévy version. Let $X_1, X_2, \ldots X_n$ be a sequence of i.i.d. random variables with expectation $E[X_i] = \mu$ and $\text{Var}[X_i] = \sigma^2 < \infty$. Let

$$S_n = \frac{1}{n}\sum_{i=1}^{n} X_i.$$

Then, as n approaches infinity, the distribution of the random variable $\sqrt{n}(S_n - \mu)$ converges to $\mathcal{N}(0, \sigma^2)$.

Proof. This is left as an exercise to the reader (Exercise 2.4) since it depends on generating functions, which will be introduced in Section 2.4. ■

The CLT is often used as a justification for assuming that something is Gaussian distributed. In order to assess such justifications it is important to be aware of its requirements. The random variables must be i.i.d. If one wants to analyze the sum of random variables with slightly different distributions, the CLT may break down. Also notice that $\text{Var}[X_i]$ must exist. There exist distributions for which the variance is infinity, and for such distributions the CLT cannot be used.

Since expectation is given by an integral it inherits the linearity property of the integral. Therefore, we can in general write

$$E[\mathbf{A}X + \mathbf{B}Y] = \mathbf{A}E[X] + \mathbf{B}E[Y] \tag{2.22}$$

where X and Y are random vectors and $\mathbf{A}$ and $\mathbf{B}$ are matrices. Variance involves squaring and is therefore not a linear operation. Nevertheless, the squaring operation is so benign that a similar result also can be stated for variance:

$$\text{Var}[\mathbf{A}X + \mathbf{B}Y] = \mathbf{A}\text{Var}(X)\mathbf{A}^\mathsf{T} + \mathbf{B}\text{Var}(Y)\mathbf{B}^\mathsf{T} + \mathbf{A}\text{Cov}(X,Y)\mathbf{B}^\mathsf{T} + \mathbf{B}\text{Cov}(Y,X)\mathbf{A}^\mathsf{T}. \tag{2.23}$$

We notice that variance also is a kind of expectation, which involves taking the expectation of an expectation. It can also be useful to calculate the expectation of an expectation in other contexts. The law of total expectation states that

$$E_X[X] = E_Y[E_X[X|Y]]. \tag{2.24}$$

Here we have introduced subscripts on the expectation operation to specify which random variable it applies to. If the random variables X and Y are independent, then $E_X[X|Y]$ does not depend on Y, and it follows that

$$E_{X,Y}[XY] = E_X[E_Y[XY]] = E_Y[E_X[X|Y]Y] = E_X[X]E_Y[Y]. \tag{2.25}$$

Expectation tends to have a smoothing behavior. For this reason there exist several inequalities that expectations must obey. The most famous such inequality is Jensen's inequality, which states that for an convex function $f(\cdot)$ the following must hold:

$$f(E[X]) \leq E[f(X)] \tag{2.26}$$

Such a convex function could for example be $f(x) = x^2$. Jensen's inequality can be proved by noting that the convexity of $f(\cdot)$ implies that $f(X) \geq f(E[X]) + f'(E[X])(X - E[X])$, which again implies that $f(X) \geq f(E[X])$ because $E[X - E[X])] = 0$.

We can also study moments of higher order than expectation (first-order) and variance (second-order). In particular, the fourth order moment is often used to measure how heavytailed a distribution is. By normalizing this according to the variance squared, we get the kurtosis

$$\text{Kurt}[X] = \frac{E[(X-\mu)^4]}{\sigma^4} \tag{2.27}$$

which is the standard measure for heavytailedness. For the univariate Gaussian distribution it can be shown that the kurtosis is 3. Distributions with higher kurtosis are known as leptokurtic or heavytailed, while distributions with lower kurtosis are known as platykurtic. We can also analyse how much a distribution differs from the shape of the Gaussian by evaluating its skewness, which is defined as

$$\gamma = \frac{E[(X-\mu)^3]}{\sigma^3}. \tag{2.28}$$

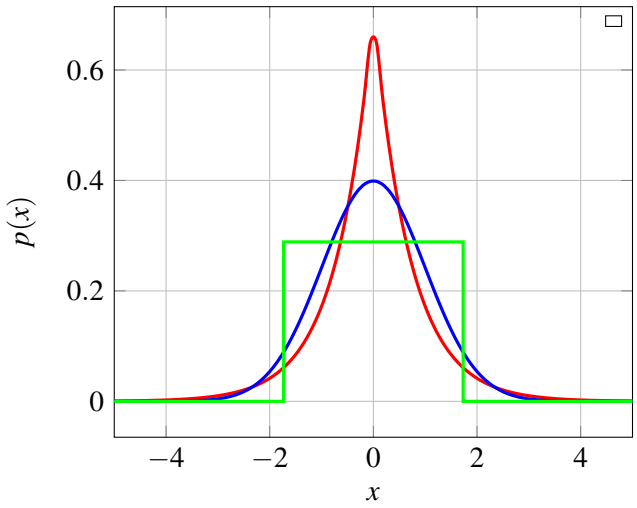


Figure 2.4: Kurtosis values for three symmetric distributions that have mean of zero and variance of one: Laplace distribution (red), normal distribution (blue) and uniform distribution (green).

A variety of information measures which describe random variables can be expressed and understood as expectations. These include the entropy

$$H[X] = E[-\ln(p(X))] \tag{2.29}$$

and the Kullback-Leibler divergence

$$D_{\text{KL}}(p,q) = E\left[-\ln\left(\frac{q(X)}{p(X)}\right)\right] \tag{2.30}$$

which provides a measure for how similar an approximating distribution $q(X)$ is to the true distribution $p(X)$. Such measures play a central role in many modern estimation methods.

2.4 Generating functions and transformations of RVs

We recall from basic control theory that sometimes it is easier to manipulate functions through their Laplace and Fourier transforms, than to work with the functions themselves. This is for example often the case when solving differential equations. In the same way, several transform-domain representations of probability distributions exist, and do often provide convenient representations. These are known as generating functions.

A generating function of a random variable is the expectation of a certain transformation of that random variable. The three most commonly encountered generating functions are known as the characteristic function, the moment-generating function and the probability-generating function. The first two are used for continuous random variables, and are in the scalar case given by

$$\Phi_X(\omega) = E_X[e^{\mathrm{i}\omega x}] = \int_{-\infty}^{\infty} p(x)e^{\mathrm{i}\omega x}\mathrm{d}x \tag{2.31}$$

$$M_X(s) = E_X[e^{sx}] = \int_{-\infty}^{\infty} p(x)e^{sx}\mathrm{d}x. \tag{2.32}$$

The latter function is used for discrete random variables, and is (again in the scalar case) given by

$$G(t) = E_X[t^X] = \sum_{n=-\infty}^{\infty} p(x_n)t^{x_n}. \tag{2.33}$$

All of these representations share three important properties:

1. The generating function determines the distribution and vice versa.
2. The generating function of a sum of independent random variables is the product of the generating functions.
3. The moments can be found by differentiating the generating function.

Property 1 should not come as a surprise. We see that $\Phi_X(\omega)$ and $M_X(s)$ work in a manner very similar to Fourier and Laplace transforms, respectively, and we know that the same principle applies to Fourier and Laplace transforms. We recognize the probability generating function as the z-transform of the pmf.

Property 2 is the most important reason why one should be familiar with generating functions: Whenever a random variable is a sum of two or more other random variables with known distributions, generating functions is a convenient tool to find the distribution of the random variable in question. Property 3 tends to be the rationale under which generating functions are presented in basic statistics courses. It is also important: It can be much more convenient to find moments by derivation than through the alternative route of integration.

■ **Example 2.8 — Sum of Bernoulli and Poisson.** Let $X \sim \text{Bernoulli}(x;p)$ and let $Y \sim \text{Poisson}(y;\lambda)$, and let $S = X + Y$. What is the probability distribution of S?
Solution: The probability generating functions of X and Y can be found as

$$G_X(t) = 1 - r + rt \tag{2.34}$$

$$G_Y(t) = \exp(\lambda(t-1)). \tag{2.35}$$

The probability generating function of S is therefore

$$G_S(t) = (1-r+rt)\exp(\lambda(t-1)) = (1-r)\exp(\lambda(t-1)) + rt\exp(\lambda(t-1)) \tag{2.36}$$

We get the probabilities of S by differentiating the probability generating function. Let us first notice that

$$\frac{\mathrm{d}^k}{\mathrm{d}t^k}\left[te^{\lambda t}\right] = \frac{\mathrm{d}^{k-1}}{\mathrm{d}t^{k-1}}\left[e^{\lambda t} + \lambda te^{\lambda t}\right] = \ldots = k\lambda^{k-1}e^{\lambda t} + \lambda^k te^{\lambda t}. \tag{2.37}$$

Based on this we find the probability that $S = k$ according to

$$\begin{aligned} p(k) &= \frac{1}{k!}\frac{\mathrm{d}^k}{\mathrm{d}t^k}G_S(t)\Big|_{t=0} \\ &= \frac{1}{k!}\left((1-r)e^{-\lambda}\frac{\mathrm{d}^k}{\mathrm{d}t^k}\left[e^{\lambda t}\right] + re^{-\lambda}\frac{\mathrm{d}^k}{\mathrm{d}t^k}\left[te^{\lambda t}\right]\right)\Big|_{t=0} \\ &= \frac{1}{k!}\left(\lambda^k(1-r)e^{-\lambda} + rk\lambda^{k-1}e^{-\lambda}\right). \end{aligned} \tag{2.38}$$

■

■ **Example 2.9 — Sum of Exponentials.** Let X_i, $i = 1,\ldots,N$ be N i.i.d. exponential RVs with parameter λ, and define $Y = \sum_{i=1}^{N} X_i$. What is the probability distribution of Y?
Solution: The moment-generating function for each of the i.i.d. exponentials is

$$M_X(s) = \lambda\int_0^\infty e^{(s-\lambda)x}\mathrm{d}x = \frac{\lambda}{\lambda - s} \quad \text{if} \quad s < \lambda. \tag{2.39}$$

According to Property 2 of the generating function this implies that

$$M_Y(s) = \left(\frac{\lambda}{\lambda - s}\right)^N. \tag{2.40}$$

If we were to calculate the inverse Laplace transformation of such an expression, we would end up with terms looking something like $x^N\exp(-\lambda x)$. Since this is precisely what we have in the Gamma function, let us check its moment-generating function:

$$M_Y(s) = \frac{1}{\theta^k\Gamma(k)}\int_0^\infty x^{k-1}\exp\left(-x\left(\frac{1}{\theta} - s\right)\right)\mathrm{d}x = \frac{1}{\theta^k\Gamma(k)}\Gamma(k)\left(\frac{1}{\frac{1}{\theta} - s}\right)^k = \left(\frac{1}{1/-\theta s}\right)^k. \tag{2.41}$$

By comparing (2.40) with (2.41) we see that the sum of N exponentials becomes a Gamma distributed random variable with scale parameter $1/\lambda$ and shape parameter N. ■

While all the three generating functions are valid for a continuous random variable, only the probability-generating function is used for discrete RVs. Generalization to vector-valued random variables is straightforward. For a vector-valued random variable X the characteristic function is

$$\Phi_X(\boldsymbol{\omega}) = E_X[e^{\mathrm{i}\boldsymbol{\omega}^\mathsf{T}\mathbf{x}}] = \int_\infty p(\mathbf{x})e^{\mathrm{i}\boldsymbol{\omega}^\mathsf{T}\mathbf{x}}\mathrm{d}\mathbf{x}. \tag{2.42}$$

While we now have a tool for finding the distributions of sums of random variables, we lack a systematic methodology for discovering the distributions that result when a scalar or vector-valued random variable is subject to a non-linear transformation. While closed-form solutions in most such cases are unattainable, fundamental formulas nevertheless exist.

Theorem 2.4.1 — Nonlinear transformations of random variables. Suppose that $\mathbf{y} = \mathbf{f}(\mathbf{x})$ where $\mathbf{h} : \mathbb{R}^n \to \mathbb{R}^n$ and $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$. Denote the pdf's of $\mathbf{x}$ and $\mathbf{y}$ by $g(\mathbf{x})$ and $h(\mathbf{y})$, respectively. Then we have that

$$h(\mathbf{y}) = \sum_i g(\mathbf{f}_i^{-1}(\mathbf{y}))\,|\det(\mathbf{F}_i^{-1}(\mathbf{y}))|$$

where $\mathbf{f}_i^{-1}(\mathbf{y})$ range over all solutions of $\mathbf{y} = \mathbf{f}(\mathbf{x})$ with respect to $\mathbf{x}$, and $\mathbf{F}_i^{-1}(\mathbf{y})$ is the corresponding Jacobian matrix of the inverse mapping $\mathbf{f}_i^{-1}(\mathbf{y})$.

Proof. For a proof in the 2-dimensional case, see [85] pages 199-201. ■

The sum in Theorem 2.4.1 reduces to a single term whenever $\mathbf{f}$ is invertible.

■ **Example 2.10 — Amplitude and phase of circularly symmetric Gaussian.** Let $\mathbf{x} = [x_1, x_2]^\mathsf{T}$ be a 2-dimensional random variable given by

$$g(\mathbf{x}) = \mathcal{N}(\mathbf{x}_1\,; 0, \sigma^2)\mathcal{N}(\mathbf{x}_2\,; 0, \sigma^2) = \mathcal{N}(\mathbf{x}; \mathbf{0}, \sigma^2\mathbf{I}). \tag{2.43}$$

In the last expression we have introduced the notation of the multivariate Gaussian, which will be extensively studied in Chapter 3. We want to find the pdf of the 2-dimensional random variable $\mathbf{y}$ given by $\mathbf{y} = \mathbf{f}(\mathbf{x})$ where $\mathbf{f} : \mathbb{R}^2 \to [0,\infty) \times [0, 2\pi)$ is given by

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} \|\mathbf{x}\|_2 \\ \text{atan2}(x_2, x_1) \end{bmatrix}. \tag{2.44}$$

Solution: First we notice that $\mathbf{f}$ is invertible on the given domain, since it simply is a conversion from Cartesian to polar coordinates. We can therefore drop the subscript i. Let us denote the components of $\mathbf{y}$ by r and θ, so that $\mathbf{y} = [r, \theta]^\mathsf{T}$. The inverse mapping and its Jacobian are given by

$$\mathbf{f}^{-1}(\mathbf{y}) = \begin{bmatrix} r\cos\theta \\ r\sin\theta \end{bmatrix} \quad \text{and} \quad \mathbf{F}^{-1}(\mathbf{y}) = \begin{bmatrix} \cos\theta & -r\sin\theta \\ \sin\theta & r\cos\theta \end{bmatrix}. \tag{2.45}$$

It is easy to see that $|\mathbf{F}^{-1}(\mathbf{y})| = r$, and it follows that

$$h(\mathbf{y}) = \frac{r}{2\pi\sigma^2}\exp\left(-\frac{1}{2\sigma^2}\left(r^2\cos^2\theta + r^2\sin^2\theta\right)\right) = \frac{r}{2\pi\sigma^2}\exp\left(-\frac{r^2}{2\sigma^2}\right) \tag{2.46}$$

While the expression in (2.46) technically solves the given problem, we are not entirely satisfied before we have the marginal pdfs of r and θ. Since only r is present in (2.46), and since θ is defined on an interval of length 2π, we do not need to perform any aditional calculations to find these: We simply factorize the joint density as $h(\mathbf{y}) = p_r(r)p_\theta(\theta)$ where

$$p(r) = \frac{r}{\sigma^2}\exp\left(-\frac{r^2}{2\sigma^2}\right) = \text{Rayleigh}(r\,; \sigma^2) \tag{2.47}$$

$$p(\theta) = \frac{1}{2\pi}\chi_{[0,2\pi]}(\theta) = \text{Uniform}(\theta\,; [0, 2\pi]). \tag{2.48}$$

■

■ **Example 2.11 — Square of zero-mean univariate Gaussian.** If $X \sim \mathcal{N}(0,1)$, what is then the pdf of $Y = X^2$?

Solution: In this case, the mapping $\mathbf{f}$ in Theorem 2.4.1 is not invertible. For any realization of Y, we have two possible realizations of X:

$$\mathbf{f}_1^{-1} = \sqrt{y} \quad \text{and} \quad \mathbf{f}_2^{-1} = -\sqrt{y}. \tag{2.49}$$

The corresponding Jacobians are

$$\mathbf{F}_1^{-1}(y) = \frac{1}{2\sqrt{y}} \quad \text{and} \quad \mathbf{F}_2^{-1}(y) = \frac{-1}{2\sqrt{y}}. \tag{2.50}$$

By stitching this together, we arrive at

$$h(y) = \frac{1}{\sqrt{2\pi}}\left[\frac{1}{2\sqrt{y}}e^{-\frac{1}{2}(\sqrt{y})^2} + \frac{1}{2\sqrt{y}}e^{-\frac{1}{2}(-\sqrt{y})^2}\right] = \frac{1}{\sqrt{2\pi y}}e^{-y/2}. \tag{2.51}$$

We recognize this as a χ^2 distribution with 1 degree of freedom. ■

2.5 Frequentist and Bayesian approaches to probability

While everyone can agree on the fundamental framework outlined in the previous section, significant controversy exists when it comes to the more philosophical interpretation of what probability really is.

First, there is the question of whether randomness actually occur in nature, or whether randomness only serves as a model for stuff we do not have any better model for. In target tracking, the latter interpretation is much more relevant than the former, but this entails that we always must consider whether randomness actually provides a realistic model.

Second, there is a distinction between frequentist and Bayesian interpretation of probability. According to a die-hard frequentist, all probabilities should have an interpretation of a frequency of some sort. For example, if it rains in 250 out of 365 days in Bergen, then the probability of rain in Bergen is about 0.68. In contrast, most Bayesians are willing to consider statistical models that involve subjective assignments of probability. Such information is encoded in a prior distribution $p(X)$ which together with the likelihood $p(z|x)$ yields the posterior distribution

$$p(x|z) \propto p(z|x)p(x) \tag{2.52}$$

This allows the Bayesian to say that the unknown x has so and so probability for being this or that, or within a given interval. Frequentists will do no such thing. Instead a frequentist will analyze how plausible the data are given x. The philosophical difference between the two schools can be seen in the different interpretations of frequentist *confidence interval* and Bayesian *credible intervals*. A 95% confidence interval is constructed so that if the experiment is repeated several times, then it will cover the true and deterministic value of x 95% of the time. A 95% credible interval will contain the true and random value of x 95% of the time.

The capability of treating estimation in purely probabilistic terms is a huge advantage for the Bayesian approach. The Bayesian can always present the full posterior as a solution to his estimation problem, and ask the customer what information she would like to see extracted from it. The caveat is of course that the prior distribution, which necessarily has a subjective element, normally will play an important role in shaping the posterior. For some problems it may be appropriate to choose a so-called noninformative prior, which is designed to minimize the amount of information imposed by the prior. In other cases, physical models do indeed provide prior knowledge that it would be rather silly not to utilize. This is typically the case in sensor fusion.

2.5.1 Bayes and conditional probability for continuous RVs

While Bayes' rule itself is not a controversial result, it is nevertheless not obvious for continuous random variables, but something that must be proved.

Theorem 2.5.1 — Bayes' rule for pdfs. Let the continuous random variables X and Z have the joint distribution $p(x,z)$. Then

$$p(x|z) = \frac{p(z|x)p(x)}{p(z)} \tag{2.53}$$

Proof. For simplicity assume that both X and Z are scalar. Let us first define the events

$$\begin{aligned} A &= \{\, z \le Z \le z + \Delta z \,\}. \\ B &= \{\, x \le X \le x + \Delta x \,\}. \end{aligned} \tag{2.54}$$

Let us denote all the cdfs of x and z by $P(\cdot)$, with and without conditioning on A or B. Bayes' law

for probabilities yields

$$\frac{\Pr\{A\,|\,B\}}{\Pr\{A\}} = \frac{\Pr\{B\,|\,A\}}{\Pr\{B\}} \tag{2.55}$$

We can rewrite this expression in terms of cdfs as follows.

$$\frac{\dfrac{P(z+\Delta z\,|\,B)-P(z\,|\,B)}{\Delta z}}{\dfrac{P(z+\Delta z)-P(z)}{\Delta z}} = \frac{\dfrac{P(x+\Delta x\,|\,A)-P(x\,|\,A)}{\Delta x}}{\dfrac{P(x+\Delta x)-P(x)}{\Delta x}} \tag{2.56}$$

We have included division by Δz and Δx above and below the original fraction bar because we obviously intend to convert the cdfs to pdfs, which will involve differentiation. Let us then define the conditional pdfs according to

$$\begin{aligned} p(z\,|\,x) &= \lim_{\Delta x\to 0} p(z\,|\,B) = \lim_{\Delta z\to 0}\lim_{\Delta x\to 0} \frac{P(z+\Delta z\,|\,B)-P(z\,|\,B)}{\Delta z} \\ p(x\,|\,z) &= \lim_{\Delta z\to 0} p(x\,|\,A) = \lim_{\Delta x\to 0}\lim_{\Delta z\to 0} \frac{P(x+\Delta x\,|\,A)-P(x\,|\,A)}{\Delta x}. \end{aligned} \tag{2.57}$$

If we now take the double limit of (2.56) for $\Delta x \to 0$ and $\Delta z \to 0$ we can insert these pdfs in the numerators, while the denominators obviously turn into $p(z)$ and $p(x)$. Thus, (2.56) becomes

$$\frac{p(z\,|\,x)}{p(z)} = \frac{p(x\,|\,z)}{p(x)} \tag{2.58}$$

which is nothing more than a restatement of Bayes' rule for pdfs. ■

It is important to keep in mind that all results concerning pdf's of continuous random variables, including Bayes and conditional probability, are based on differentiation of proper probabilities. Differentiation is always relative to the metric properties, i.e., parametrization and units, of the underlying space, which may be $\mathbb{R}$, $\mathbb{R}^d$ or something more exotic, e.g., a sphere. The choice of coordinate system may not only affect the values of the pdf's, but also their overall shape.

2.6 Estimators

Estimation is the task of inferring knowledge about an unknown quantity x from data z which are related to x. In the probabilistic paradigm, the relationship between z and x is in the form of a probabilistic model $p(z\,|\,x)$. In the frequentist approach, this is all that is given. The Bayesian approach, prior knowledge about x is also given in the form of another probabilistic model $p(x)$, and one must then weigh the two models against each other to find an estimate of x.

For a given estimation problem, an *estimator* is a procedure that attempts to guess a concrete value for what x is based on the data. The output of the estimator is called an *estimate*.

Definition 2.6.1 — Estimator. Let $\mathscr{Z}$ and $\mathscr{X}$ be the spaces in which z and x belong, respectively. An estimator is a function $\theta : \mathscr{Z} \to \mathscr{X}$ so that $\theta(z)$ gives an estimate of x. We use a hat to denote the estimate, i.e., $\hat{x} = \theta(z)$.

Estimators can be categorized according to various categories. First, there is of course the divide between frequentist and Bayesian estimators. We may also distinguish between *maximizing* and *averaging* estimators. A maximizing estimator aims to find the one best value of x given z. An averaging estimator aims to find a value of x which is representative of our knowledge about x given z. While maximizing estimators can be defined in both frequentist and Bayesian estimation, the concept of an averaging estimator necessarily involves elements of Bayesian thinking. If one wants to have such tools available, it is hard remain a die-hard frequentist.[1]

[1] "Everyone is either a Bayesian or a closet Bayesian", Josef Knoll.

2.6.1 Maximizing estimators

Two well-known maximizing estimators are the maximum likelihood (ML) and maximum *a posteriori* (MAP) estimators. The ML estimator is given by

$$\hat{x} = \arg\max_x p_{Z|X}(z|x). \tag{2.59}$$

while the MAP estimator is given by

$$\hat{x} = \arg\max_x p_{X|Z}(x|z) = \arg\max_x p_{Z|X}(z|x)p_X(x). \tag{2.60}$$

In some special cases closed-form expressions for the ML or MAP estimators exist. It does probably not come as a big surprise that such cases include estimation of expectation and covariance for a Gaussian distribution. Another example is given in Example 2.12. In general, however, numerical search techniques are needed to implement these estimators. The menu that such techniques can be chosen from is fairly daunting, and a solid understanding of the objective functions in (2.59) or (2.60) is essential, both to succeed at all, and to achieve acceptable computational efficiency.

To which extent are the ML and MAP estimators reliable? The answer is to a larger extent affirmative for the ML estimator than for the MAP estimator. The ML estimator has some nice properties that statisticians refer to as *consistency* and *efficiency*. The former means that given enough data it will always recover the correct parameter. The latter means that it as it approaches that limit will attain a higher accuracy than any other kind of estimator that utilizes the same information.

For the MAP estimator we need to be more careful, since it also involves a prior distribution and Bayes' rule. Recalling that pdf's are only defined by differentiation relative to the coordinate system, it is clear that a change of coordinates can alter the location of the peak of a pdf. Since the MAP estimator maximizes the posterior pdf, it is therefore clear that the MAP estimator may not be invariant under coordinate transformations. This should not, however, be taken as a criticism of the Bayesian paradigm. A bona-fide Bayesian will never think of the MAP estimator as the solution to her estimation problem. In the Bayesian mindset, the entire posterior distribution is the solution, and more or less reliable information about this solution can be extracted by various estimators. For discrete estimation problems, also known as classification problems, the picture is somewhat different. Then there is a (hopefully nonzero) probability that the estimator hits the right value or class, and it can be shown that the MAP estimator has the highest success rate of all possible estimators, see Exercise 2.7.

2.6.2 Estimators as random variables

Since an estimator depends on the data, and the data both in Bayesian and frequentist schools are random variables, estimators are random variables as well. This means that a given estimator for a given model has a particular distribution. It is important to be aware of this for several reasons. Often, one wants to know the mean square error (MSE) of an estimator, and the MSE is given by this distribution. Furthermore, knowledge about an estimator's distribution can be important in subsequent processing of the estimate. For example, if two different estimates are to be combined in a sensor fusion system, then we would typically put most weight on the estimate with the lowest MSE.

■ **Example 2.12 — ML estimator of Rayleigh distribution.** Let $\mathbf{z} = [z_1, \ldots, z_M]^\mathsf{T}$ consist of i.i.d. samples from a Rayleigh distribution:

$$p(\mathbf{z}\,|\,\eta) = \prod_{i=1}^{M} \frac{z_i}{\eta} \exp\left(-\frac{z_i^2}{\eta^2}\right). \tag{2.61}$$

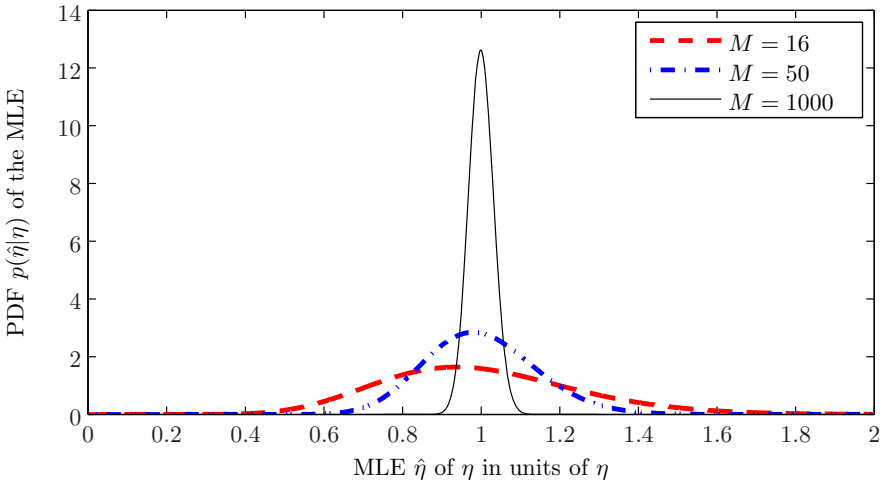


Figure 2.5: Distribution of the Rayleigh MLE for different sample sizes. Notice that a rather large number of samples is needed if η is to be estimated with accuracy of say 5%.

By differentiating the logarithm of $p(\mathbf{z}\,|\,\eta)$ and equating the derivative to zero, we get the ML estimator

$$\hat{\eta} = \frac{1}{2M}\sum_{i=1}^{M} z_i^2. \tag{2.62}$$

It can be shown that this quantity has a Gamma distribution. In Exercise 2.3 you are asked to show that each z_j^2 has an exponential distribution. Furthermore, we know from Example 2.9 that such a sum of exponential random variables is Gamma distributed. Through these steps we arrive at

$$p(\hat{\eta}\,|\,\eta) = \text{Gamma}\left(\hat{\eta}\,; M, \frac{2\eta}{M}\right) = \frac{M^M}{\Gamma(M)}\left(\frac{\hat{\eta}}{\eta}\right)^M \exp\left(-M\frac{\hat{\eta}}{\eta}\right). \tag{2.63}$$

This result is of central importance in detection theory, because we can use it to tune detection probabilities and false alarm rates of radar detectors. ∎

Related to the concept of estimators is the concept of *statistics*. A statistic is a single measure of some attribute of a sample. Estimators are statistics, but we can also construct statistics that not necessarily correspond to a meaningful estimator. As an example, the term $\sum_{i=1}^{M} z_i^2$ in Example 2.12 is a statistic. When we talk about statistics in this sense, we are primarily interested in *sufficient statistics*.

Definition 2.6.2 — Sufficient statistic. A sufficient statistic $g(\mathbf{z})$ for the parameter $\mathbf{x}$ summarizes the information about $\mathbf{x}$ contained in the data $\mathbf{z}$.

Returning to Example 2.12, we see that the term $\sum_{i=1}^{M} z_i^2$ is also a sufficient statistic for the parameter η. This is evident, because in (2.62) the individual data values z_j only occur in this expression. In other words, there is no dependence of the data which are not encapsulated by this term, and the term is then by definition a sufficient statistic. More generally, according to the Neyman-Fisher factorization theorem, it holds that $g(\mathbf{z})$ is a sufficient statistic if the likelihood can be factorized as $f(\mathbf{z}\,|\,\mathbf{x}) = f_1(g(\mathbf{z}),\mathbf{x})f_2(\mathbf{z})$. By a slight abuse of terminology, the parameters required to describe a given distribution, e.g., expectation and covariance of a Gaussian, are sometimes referred to as sufficient statistic. This is abusive because these are not functions of the data. On the other hand, the sample mean and the sample covariance *are* statistics.

2.6.3 LS and MMSE estimators

The least squares (LS) estimator and the minimum mean square error (MMSE) estimator both represent philosophies different from the the ML or MAP estimators. Consider an estimation problem of the form $\mathbf{z} = \mathbf{h}(\mathbf{x}) + \mathbf{w}$ where $\mathbf{w}$ is an unknown vector of measurement noise. The least squares estimator is

$$\hat{\mathbf{x}}_{\text{LS}} = \arg\min_{\mathbf{x}} \|\mathbf{z} - \mathbf{h}(\mathbf{x})\|_2^2. \tag{2.64}$$

It is evident from (2.64) that the LS estimator does not make any assumptions about the measurement "noise" $\mathbf{w}$. It is therefore of a non-probabilistic nature, and is a prime candidate to be considered in problems where a probabilistic model is not available. If $\mathbf{w}$ consists of i.i.d. Gaussian samples, then the LS estimator becomes identical to the MLE.

The idea of minimizing quadratic cost functions also plays a central role in the Bayesian paradigm. In accordance with the Bayesian philosophy of accounting for all uncertainty, the central concept in Bayesian decision theory is to make decisions that minimize given risk measures. An estimator that can be shown to minimize some risk function, is said to be Bayes-optimal with regard to that risk function. This is a very general framework. The decisions can in principle be any actions that are based on observing the data. The risk function can be any quantification of the consequences of the decision. Within the limited scope of estimation, however, the decisions will typically be nothing more than choosing a particular possibility of the estimatee as the estimate. In mathematical terms, if $l(\hat{\mathbf{x}}, \mathbf{x})$ is a loss function describing the deviation between the estimatee θ and our guess of its value $\hat{\theta}$, then the corresponding Bayes-optimal estimator is

$$\hat{\mathbf{x}}_l = \arg\min_{\hat{\mathbf{x}}} E_{\mathbf{x}}[l(\hat{\mathbf{x}}, \mathbf{x})]. \tag{2.65}$$

The most popular risk function is the expectation of the squared error, which in the vector case means the expectation of the function

$$l(\hat{\mathbf{x}}, \mathbf{x}) = (\hat{\mathbf{x}} - \mathbf{x})^{\mathsf{T}}(\hat{\mathbf{x}} - \mathbf{x}). \tag{2.66}$$

The estimator that results is known as the minimum mean square error (MMSE) estimator, which also is known as the expected *a posteriori* estimator, for reasons that will become obvious in the following theorem.

Theorem 2.6.1 — MMSE estimator. The MMSE estimator is given by

$$\hat{\mathbf{x}}_{\text{MMSE}} = \arg\min_{\hat{\mathbf{x}}} E\left[(\hat{\mathbf{x}} - \mathbf{x})^{\mathsf{T}}(\hat{\mathbf{x}} - \mathbf{x})\right] = E[\mathbf{x}|\mathbf{z}] = \int \mathbf{x} p(\mathbf{x}|\mathbf{z}) \mathrm{d}\mathbf{x}. \tag{2.67}$$

Proof. If $\hat{\mathbf{x}}_{\text{MMSE}}$ minimizes $E[l(\hat{\mathbf{x}}, \mathbf{x}) \,|\, \mathbf{z}]$ for all $\mathbf{z}$, then it will also minimize $E[l(\hat{\mathbf{x}}, \mathbf{x})]$. We proceed to show that it indeed does that by means of differentiation.

$$\frac{\partial E[(\hat{\mathbf{x}} - \mathbf{x})^{\mathsf{T}}(\hat{\mathbf{x}} - \mathbf{x}) \,|\, \mathbf{z}]}{\partial \hat{\mathbf{x}}} = E\left[\frac{\partial (\hat{\mathbf{x}} - \mathbf{x})^{\mathsf{T}}(\hat{\mathbf{x}} - \mathbf{x})}{\partial \hat{\mathbf{x}}} \,|\, \mathbf{z}\right] = E\left[-2(\hat{\mathbf{x}} - \mathbf{x})^{\mathsf{T}} \,|\, \mathbf{z}\right] \tag{2.68}$$

To identify the minimizer of the MSE we equate this with zero, which yields

$$\mathbf{0} = E\left[\hat{\mathbf{x}} - \mathbf{x} \,|\, \mathbf{z}\right] = E\left[\hat{\mathbf{x}} \,|\, \mathbf{z}\right] - E\left[\mathbf{x} \,|\, \mathbf{z}\right]. \tag{2.69}$$

Since $\hat{\mathbf{x}}$ is a function of the data we have that $\hat{\mathbf{x}} = E\left[\hat{\mathbf{x}} \,|\, \mathbf{z}\right]$ and it follows that $\hat{\mathbf{x}} = E\left[\mathbf{x} \,|\, \mathbf{z}\right]$. ■

2.6.4 Bias, MSE and variance of estimators

We generally want estimators that have low bias and MSE. Let $\tilde{\mathbf{x}} = \mathbf{x} - \hat{\mathbf{x}}$ be the estimation error. The bias of an estimator is then defined as the expectation $E[\tilde{\mathbf{x}}]$. We say that an estimator is *unbiased* if $E[\tilde{\mathbf{x}}] = 0$. The MMSE estimator is always unbiased. The ML and MAP estimators are, on the other hand, in general biased. Unbiasedness is obviously desirable in general. However, there may exist biased estimators with lower MSE than the best unbiased estimator, and there may exist estimation problems where the requirement of unbiasedness will lead to unacceptable degradation of the MSE.

In the scalar case, the variance and MSE of an estimator are given by

$$\mathrm{Var}(\hat{x}) = E[(\hat{x} - E[\hat{x}])^2] \ , \ \mathrm{MSE}(\hat{x}) = E[(\hat{x} - x)^2]. \tag{2.70}$$

From this it follows that the MSE can be decomposed into variance and bias as follows:

$$\mathrm{MSE}(\hat{x}) = \mathrm{Var}(\hat{x}) + \mathrm{Bias}(\hat{x}, x)^2. \tag{2.71}$$

In the vector case, the variance becomes a matrix:

$$\mathrm{Cov}(\hat{\mathbf{x}}) = E\left[(\hat{\mathbf{x}} - E[\hat{\mathbf{x}}])(\hat{\mathbf{x}} - E[\hat{\mathbf{x}}])^\mathsf{T}\right], \tag{2.72}$$

The MSE is on the other hand a scalar number also in the vector case, as already indicated by (2.67):

$$\mathrm{MSE}(\hat{\mathbf{x}}) = E\left[\|\mathbf{x} - \hat{\mathbf{x}}\|_2^2\right] = E\left[(\hat{\mathbf{x}} - \mathbf{x})^\mathsf{T}(\hat{\mathbf{x}} - \mathbf{x})\right] = \mathrm{tr}\left(E[(\hat{\mathbf{x}} - \mathbf{x})(\hat{\mathbf{x}} - \mathbf{x})^\mathsf{T}]\right). \tag{2.73}$$

2.6.5 LMMSE estimators

When the MMSE estimator is too complicated we may settle for the best linear estimator.

Theorem 2.6.2 — LMMSE estimator. The best estimator of the form $\hat{\mathbf{x}} = \mathbf{A}\mathbf{z} + \mathbf{b}$ that minimizes $\mathrm{MSE}(\hat{\mathbf{x}}) = E\left[\|\mathbf{x} - \hat{\mathbf{x}}\|_2^2\right]$ is given by

$$\hat{\mathbf{x}} = E[\mathbf{x}] + \mathrm{Cov}(\mathbf{x}, \mathbf{z})\mathrm{Cov}(\mathbf{z})^{-1}(\mathbf{z} - E[\mathbf{z}]). \tag{2.74}$$

Proof. We prove the case where the estimatee is a scalar x, i.e., when the estimator is of the form $\hat{x} = \mathbf{a}^\mathsf{T}\mathbf{z} + b$. First we find what b should be. The estimation error and its square can be written

$$\tilde{x} = x - \mathbf{a}^\mathsf{T}\mathbf{z} - b \tag{2.75}$$

$$\tilde{x}^2 = (x - \mathbf{a}^\mathsf{T}\mathbf{z})^2 - 2b(x - \mathbf{a}^\mathsf{T}\mathbf{z}) + b^2, \tag{2.76}$$

respectively. The MSE is

$$E[\tilde{x}^2] = E[(x - \mathbf{a}^\mathsf{T}\mathbf{z})^2] - 2b(E[x] - \mathbf{a}^\mathsf{T}E[\mathbf{z}]) + b^2. \tag{2.77}$$

To find the value of b that minimizes this we calculate the derivative with respect to b and equate it with zero. This results in

$$b = E[x] - \mathbf{a}^\mathsf{T}E[\mathbf{z}]. \tag{2.78}$$

We could also have arrived at this result by requiring that LMMSE estimator should be unbiased. It is easy to see that if we insert (2.78) into the expression for $\tilde{x}$ in (2.75) then the bias becomes zero.

We proceed to look for the optimal vector $\mathbf{a}$. By demanding the derivative of the MSE with respect to $\mathbf{a}$ to be zero, we obtain

$$\frac{\partial}{\partial \mathbf{a}} E[\tilde{x}^2] = \frac{\partial}{\partial \mathbf{a}} E\left[(x - E[x] - \mathbf{a}(\mathbf{z} - E[\mathbf{z}]))^2\right] = 2E[\tilde{x}(\mathbf{z} - E[\mathbf{z}])^\mathsf{T}] = 0. \tag{2.79}$$

The second equality follows from the chain rule. The result that $E[\tilde{x}(\mathbf{z} - E[\mathbf{z}])^\mathsf{T}] = 0$ is known as the orthogonality principle. By further analyzing what this entails we obtain

$$\begin{aligned} E[\tilde{x}\mathbf{z}^\mathsf{T}] &= E\left[\left(x - E[x] - \mathbf{a}^\mathsf{T}(\mathbf{z} - E[\mathbf{z}]\right)(\mathbf{z} - E[\mathbf{z}])^\mathsf{T}\right] \\ &= \mathrm{Cov}[x, \mathbf{z}] - \mathbf{a}^\mathsf{T}\mathrm{Cov}[\mathbf{z}] = 0. \end{aligned} \tag{2.80}$$

This leads to the optimal choice of

$$\mathbf{a} = \mathrm{Cov}[x, \mathbf{z}]\,\mathrm{Cov}[\mathbf{z}]^{-1}. \tag{2.81}$$

Combining this with the expression for b yields the formula in the theorem. ■

It can also be seen that the matrix MSE of the LMMSE estimator is given by

$$\begin{aligned} E[\tilde{\mathbf{x}}\tilde{\mathbf{x}}^\mathsf{T}] =& E\Big[\left(\mathbf{x} - E[\mathbf{x}] - \mathrm{Cov}[\mathbf{x}, \mathbf{z}]\mathrm{Cov}[\mathbf{z}]^{-1}(\mathbf{z} - E[\mathbf{z}])\right) \\ & \qquad \left(\mathbf{x} - E[\mathbf{x}] - \mathrm{Cov}[\mathbf{x}, \mathbf{z}]\mathrm{Cov}[\mathbf{z}]^{-1}(\mathbf{z} - E[\mathbf{z}])\right)^\mathsf{T}\Big] \\ =& \mathrm{Cov}[\mathbf{x}] - \mathrm{Cov}[\mathbf{x}, \mathbf{z}]\mathrm{Cov}[\mathbf{z}]^{-1}\mathrm{Cov}[\mathbf{x}, \mathbf{z}]^\mathsf{T}. \end{aligned} \tag{2.82}$$

The LMMSE equations (2.74) and (2.82) bears some resemblance to the Kalman filter formulas. In fact, we could have derived the Kalman filter as an example of LMMSE estimation by means of the orthogonality principle. We shall instead, however, study the Kalman filter in a more directly Bayesian setting, because this will make it easier to understand how the Kalman filter is used in sensor fusion applications such as target tracking and SLAM. For this purpose, the next chapter is entirely devoted to the multivariate Gaussian distribution.

2.7 References and chapter remarks

The aim of this chapter has largely been to provide a condensed summary of the first three chapters in [4]. Thus, if the reader is looking for further details, that would be a good place to start. For fundamentals of probability theory, the reader may consult a basic probability textbook such as [112], or go straight to the more comprehensive [85]. The proof for Bayes' rule for pdf's in Theorem 2.5.1 is based on a somewhat more cursorial proof found in [85]. To do anything with the measure theoretic approach to probability must students who follow TTK4250 would first need to become familiar with measure theory, which is rigorously explained in [34]. The best source on Bayesian estimation and decision theory known to the author is [32].

2.8 Exercises

Exercise 2.1 Let $Y = X_1 + \ldots + X_n$ be the sum of n independent random vectors. Show that the characteristic function of Y as given by (2.42) is equal to the product of the characteristic functions of $X_1, \ldots, X_n$. ■

Exercise 2.2 Let $X_1, \ldots, X_n$ be n i.i.d. random variables, each χ^2 distributed with k degrees of freedom. Show that $Y = X_1 + \ldots + X_n$ is χ^2 distributed with nk degrees of freedom. ■

Exercise 2.3 Let X be a Rayleigh distributed random variable with parameter σ^2 as defined on page 21. Show that X^2 is an exponential random variable with parameter $\lambda = 1/\sigma^2$. ■

Exercise 2.4 Prove the Lindeberg-Lévy version of the CLT. That is, prove Theorem 2.3.1. **Hint:** Use generating functions. ■

Exercise 2.5 Let $X_1, \ldots, X_n$ be n independent Bernoulli random variables with success probabilities $r_1, \ldots, r_n$, respectively.

a) What is distribution of $Y = \sum_{i=1}^{n} X_n$?
b) What is the expectation of Y?
c) What is the covariance of Y?

■

Exercise 2.6 Let X be a random variable with cumulative distribution function F. Show that the random variable $Y = F(X)$ is uniformly distributed over $[0,1]$. ■

Exercise 2.7 Show that the MAP classifier has the highest success rate of all possible classifiers for a Bayesian classification problem. ■

3. The multivariate Gaussian

The key construction that underlies the Kalman filter, and virtually all of sensor fusion, is the multivariate Gaussian, which generalizes the univariate Gaussian from Example (2.6). The distribution of a multivariate Gaussian RV is given by its expectation vector μ and symmetric positive definite covariance matrix $\mathbf{P}$, according to

$$\mathcal{N}(\mathbf{x};\mu,\mathbf{P}) = \frac{1}{(2\pi)^{\frac{n}{2}}|\mathbf{P}|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^{\mathsf{T}}\mathbf{P}^{-1}(\mathbf{x}-\mu)\right) \tag{3.1}$$

Recall that the univariate Gaussian pdf always looks like a bell curve whose peak location and spread are governed by its expectation and covariance, respectively. A two-dimensional Gaussian gives a similar bell surface, whose peak location is given by its expectation vector, and whose shape is given by its covariance matrix. This generalizes in the obvious manner to higher dimensions.

3.1 Quadratic forms and covariance ellipses

The exponent in (3.1) is a quadratic form in the variable $\mathbf{x}$. Let us more precisely define the quadratic form corresponding to (3.1) as

$$q(\mathbf{x}) = (\mathbf{x}-\mu)^{\mathsf{T}}\mathbf{P}^{-1}(\mathbf{x}-\mu). \tag{3.2}$$

The value of this function for a given $\mathbf{x}$ and μ is also known as the Mahalanobis distance between $\mathbf{x}$ and μ. In terms of this function we can write the logarithm of the Gaussian pdf as

$$\ln\mathcal{N}(\mathbf{x};\mu,\mathbf{P}) = c - \frac{1}{2}q(\mathbf{x}) \tag{3.3}$$

where c is a constant that is given by the normalization requirement, and therefore carries no information about $\mathbf{x}$. From this we make three very important observations.

First, this means that the curvature, or more precisely the Hessian, of $\ln\mathcal{N}(\mathbf{x};\mu,\mathbf{P})$ is constant. In fact, if $\mathcal{H}_{\mathbf{x}}$ denotes the Hessian, the following identity holds for any $\mathbf{x}$:

$$\mathcal{H}_{\mathbf{x}}\ln\mathcal{N}(\mathbf{x};\mu,\mathbf{P}) = -\mathbf{P}^{-1}. \tag{3.4}$$

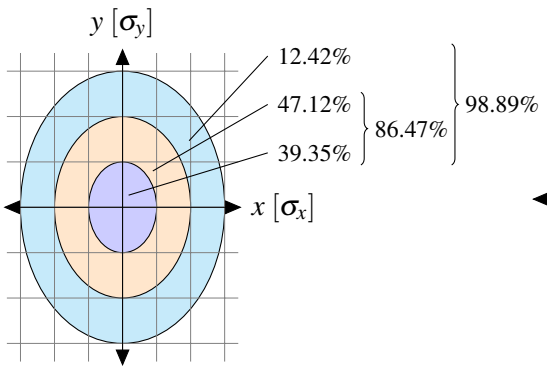
Second, the level curves for a 2-dimensional Gaussian are ellipses, since any expression of the form $Ax^2 + Bxy + Cy^2 = 1$ describes an ellipse in the $x-y$-plane. More generally, we get ellipsoids for a 3-dimensional Gaussian, and so on. For simplicity, we will just refer to these as covariance ellipses, irrespectively of the dimension. The axes of these ellipses can be found by means of eigenvalue decomposition of $\mathbf{P}$. If $(\lambda_i, \mathbf{e}_i)$ are the eigenvalues and eigenvectors of $\mathbf{P}$, the the 1σ ellipse of $\mathbf{P}$ has axes given by $\sqrt{\lambda_i}\mathbf{e}_i$, and the more general $g\sigma$-ellipse has axes given by $g\sqrt{\lambda_i}\mathbf{e}_i$. The area, or more generally hypervolume, within such an ellipse is given by

$$\frac{\pi^{n/2}}{\Gamma(n/2+1)} g^n \sqrt{|\mathbf{P}|}.$$

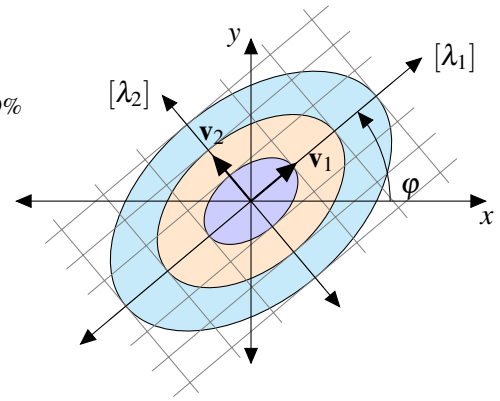
Third, it is evident that a Gaussian is entirely specified in terms of its quadratic form. Conversely, if we multiply a convex quadratic form in $\mathbf{x}$ by $\left(-\frac{1}{2}\right)$, and then exponentiate it, we get a function which is proportional to a particular Gaussian distribution in $\mathbf{x}$. The normalization requirement implies that this function cannot be proportional to any other pdf than this particular Gaussian. The consequence of this is that when we study how multivariate Gaussians behave under various operations (marginalization, conditioning, etc.) *we only need to study the quadratic forms*.

■ **Example 3.1 — Conditioning is entirely given by quadratic forms.** Let $p(x,y)$ be a bivariate Gaussian, and let $f(x,y)$ be its corresponding quadratic form. The conditional distribution of x given y is $p(x|y) = p(x,y)/p(y)$. We know that both $p(x|y)$, $p(x,y)$ and $p(y)$ are valid pdfs. We also know that $p(y)$ is a function of y alone. The dependency of $p(x|y)$ on x is therefore given by the quadratic form $f(x,y)$, this time treated as a function of x alone, with y fixed. Thus, we see that $p(x|y)$ also must be Gaussian, and that we can determine $p(x|y)$ from $p(x,y)$ just by studying quadratic forms. The exact details for how this is to be done are left for Theorem 3.2.3 in the next section. ■

The amount of probability within a $g\sigma$-ellipse decreases as the dimension n increases. This is because higher dimension provides more directions in which probability mass can be spread. to quantify this, the trick is to transform the Gaussian random vector into a scalar χ^2 random variable whose cdf can be written in terms of the error function or directly evaluated using, e.g., Matlab. More precisely, if $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{P})$ it can be shown that $(\mathbf{x}-\mu)^\mathsf{T}\mathbf{P}^{-1}(\mathbf{x}-\mu)$ has a χ^2 distribution with n degrees of freedom (Exercise 3.4). The probability mass within 1, 2 and 3 σ for a bivariate Gaussian is illustrated in Figure 3.1.



(a) $\Sigma = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}$.



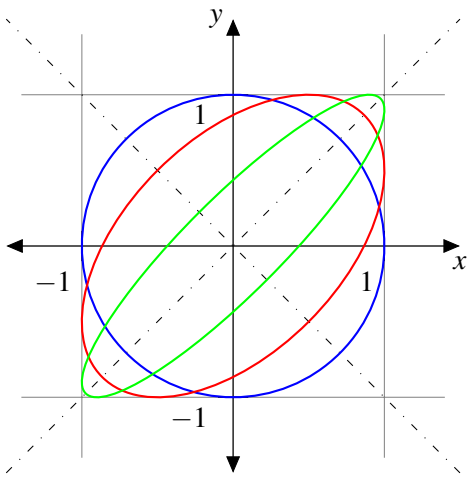
(b) $\Sigma = \mathbf{P}\Lambda\mathbf{P}^T$, $\mathbf{P} = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 \end{bmatrix}$.

Figure 3.1: Two-dimensional Gaussian distribution. Probability of ellipses.

In the right hand side of Figure 3.1 we see how the similarity transform (also known as the eigendecomposition or spectral transform) of the covariance matrix yields an alternative coordinate

system, spanned by the eigenvectors, where all cross-correlations vanish. This can be useful for a variety of computational purposes. It is also possible to transform a correlated Gaussian vector to a correlation-free Gaussian vector by means of other decompositions such as the Cholesky factorization (see Example 3.4). We see that correlations make the covariance ellipses tilted. In the right hand side of Figure 3.1 there is a positive correlation between x and y. The ellipses would have been tilted the other way if it was negative.

■ **Example 3.2 — Correlations make the covariance ellipses narrower.** In Figure 3.2 we see the 1σ covariance ellipses of three bivariate Gaussians with unity marginal variance in the x- and y-directions. The top and bottom of all the covariance ellipses touch the -1 and $+1$ lines, and the same happens horizontally. However, the Gaussians with the highest cross-correlation a, have the narrowest covariance ellipses. This shows that the presence of correlations actually decrease uncertainty. For this reason, estimation methods such as the Kalman filter exploit correlations.



$$\begin{bmatrix} x \\ y \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & a \\ a & 1 \end{bmatrix} \right)$$

$$\begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 1 & a \\ a & 1 \end{bmatrix}^{-1} \begin{bmatrix} x \\ y \end{bmatrix} = 1$$

$a = 0.0$
$a = 0.5$
$a = 0.9$

Figure 3.2: For $a \in (-1,1)$, the semiaxes are $\sqrt{1+a}$ and $\sqrt{1-a}$ long and lie on the $y = x$ and $y = -x$ lines, respectively.

■

3.2 Rules for working with Gaussians

We will now establish several important results that make it much easier to work with with multivariate Gaussians than with any other kind of multivariate RVs. Based on the argument elaborated in Example 3.1, we will prove all these results by simply studying how the quadratic form in the exponent of (3.1) behaves.

Theorem 3.2.1 — Independence. Two random vectors $\mathbf{x}$ and $\mathbf{y}$ with probability density functions $\mathcal{N}(\mathbf{x};\mathbf{a},\mathbf{A})$ and $\mathcal{N}(\mathbf{y};\mathbf{b},\mathbf{B})$ are independent if and only if

$$p(\mathbf{x},\mathbf{y}) = p\left(\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}; \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}, \begin{bmatrix} \mathbf{A} & 0 \\ 0 & \mathbf{B} \end{bmatrix}\right). \tag{3.5}$$

Proof. We need to prove that zero covariance implies that $p(\mathbf{x},\mathbf{y}) = p(\mathbf{x})p(\mathbf{y})$ and vice versa. According to the inversion rule of Appendix P.1 we know that the inverse of a block-diagonal matrix is found by simply inverting the blocks. Thus, the quadratic form in the exponent of the

joint Gaussian becomes

$$\begin{bmatrix} \mathbf{x}-\mathbf{a} \\ \mathbf{y}-\mathbf{b} \end{bmatrix}^\mathsf{T} \begin{bmatrix} \mathbf{A} & 0 \\ 0 & \mathbf{B} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{x}-\mathbf{a} \\ \mathbf{y}-\mathbf{b} \end{bmatrix} = \begin{bmatrix} (\mathbf{x}-\mathbf{a})^\mathsf{T} & (\mathbf{y}-\mathbf{b})^\mathsf{T} \end{bmatrix} \begin{bmatrix} \mathbf{A}^{-1} & 0 \\ 0 & \mathbf{B}^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{x}-\mathbf{a} \\ \mathbf{y}-\mathbf{b} \end{bmatrix}$$
$$= (\mathbf{x}-\mathbf{a})^\mathsf{T}\mathbf{A}^{-1}(\mathbf{x}-\mathbf{a}) + (\mathbf{y}-\mathbf{b})^\mathsf{T}\mathbf{B}^{-1}(\mathbf{y}-\mathbf{b})$$

which is a sum of the quadratic forms from the marginal distributions. Based on this, we see that zero covariance implies that $\ln p(\mathbf{x},\mathbf{y}) = \ln p(\mathbf{x}) + \ln p(\mathbf{y})$, which again implies that $p(\mathbf{x},\mathbf{y}) = p(\mathbf{x})p(\mathbf{y})$. The converse implications also follow. ■

This relationship between independence and zero covariance does not hold for arbitrary non-Gaussian random vectors.

There is a strong relationship between Gaussianity and linearity, to the extent that these concepts often are used interchangeably in the literature. If a Gaussian RV goes through a linear transform, the new RV is also Gaussian.

Theorem 3.2.2 — Linearity. If $p(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \mathbf{a}, \mathbf{A})$ and $\mathbf{y} = \mathbf{F}\mathbf{x}$, then $p(\mathbf{y}) = \mathcal{N}(\mathbf{y}; \mathbf{F}\mathbf{a}, \mathbf{F}\mathbf{A}\mathbf{F}^\mathsf{T})$.

Proof. A general proof of this property can be constructed by means of the moment-generating function. In the special case that $\mathbf{F}$ is square and invertible, a simpler proof can be constructed by means of the nonlinear transformation formula in Theorem 2.4.1. We only sketch this proof here. Denote the original pdf by $g(\mathbf{x})$ and the pdf of the new RV by $h(\mathbf{y})$. We then have that

$$\begin{aligned} g(\mathbf{x}) &\propto \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{a})^\mathsf{T}\mathbf{A}^{-1}(\mathbf{x}-\mathbf{a})\right) \\ h(\mathbf{y}) &\propto \exp\left(-\frac{1}{2}(\mathbf{F}^{-1}\mathbf{y}-\mathbf{a})^\mathsf{T}\mathbf{A}^{-1}(\mathbf{F}^{-1}\mathbf{y}-\mathbf{a})\right) \\ &= \exp\left(-\frac{1}{2}(\mathbf{F}^{-1}\mathbf{y}-\mathbf{a})^\mathsf{T}\mathbf{F}^\mathsf{T}(\mathbf{F}^\mathsf{T})^{-1}\mathbf{A}^{-1}\mathbf{F}^{-1}\mathbf{F}(\mathbf{F}^{-1}\mathbf{y}-\mathbf{a})\right) \\ &= \exp\left(-\frac{1}{2}(\mathbf{y}-\mathbf{F}\mathbf{a})^\mathsf{T}(\mathbf{F}^\mathsf{T})^{-1}\mathbf{A}^{-1}\mathbf{F}^{-1}(\mathbf{y}-\mathbf{F}\mathbf{a})\right). \end{aligned} \tag{3.6}$$

The theorem then follows from recognizing that $(\mathbf{F}^\mathsf{T})^{-1}\mathbf{A}^{-1}\mathbf{F}^{-1} = (\mathbf{F}\mathbf{A}\mathbf{F}^\mathsf{T})^{-1}$. ■

A related result is that if $\mathbf{x}$ has the distribution $\mathcal{N}(\mathbf{a},\mathbf{A})$, then the random vector $\mathbf{y} = \mathbf{x}+\mathbf{b}$ has the distribution $\mathcal{N}(\mathbf{a}+\mathbf{b},\mathbf{A})$. This follows almost trivially from the nonlinear transformation formula, because all we have to is shift $\mathbf{b}$ from the random vector slot to the expectation slot.

▪ **Example 3.3 — Sum of variances.** Let $x_1 \sim \mathcal{N}(0,\sigma^2)$ and $x_2 \sim \mathcal{N}(0,\sigma^2)$. Then it follows that $x = x_1 + x_2 \sim \mathcal{N}(0, 2\sigma^2)$. This follows from the theorem with $\mathbf{F} = [1,1]$, $\mathbf{x} = [x_1, x_2]^\mathsf{T}$ and $\mathbf{y} = x_3$. ▪

▪ **Example 3.4 — Cholesky decomposition of the covariance matrix.** Let $\mathbf{x} \sim \mathcal{N}(\mathbf{0},\mathbf{A})$. Since $\mathbf{A}$ is symmetric positive definite we know that it has a Cholesky factorization $\mathbf{L}$ so that $\mathbf{A} = \mathbf{L}\mathbf{L}^\mathsf{T}$. Let us then define the transformed RV $\mathbf{y} = \mathbf{L}^{-1}\mathbf{x}$. The expectation of $\mathbf{y}$ is then obviously zero as well. Its covariance becomes

$$\mathrm{Cov}[\mathbf{y}] = \mathbf{L}^{-1}\mathbf{A}(\mathbf{L}^{-1})^\mathsf{T} = \mathbf{L}^{-1}\mathbf{L}\mathbf{L}^\mathsf{T}(\mathbf{L}^{-1})^\mathsf{T} = \mathbf{I}. \tag{3.7}$$

The effect that the linear transformation $\mathbf{L}^{-1}$ has on $\mathbf{x}$ is often described as "whitening" or "prewhitening". This is a common and important operation, especially in signal processing, where $\mathbf{x}$ may be thought of as a long time sequence constituting a signal. By whitening a signal

we can perform operations on it (e.g, hypothesis tests) which require whiteness, i.e., independence between the samples. We can also use a reversal of the above argument to generate correlated Gaussian RVs. If we have a Gaussian RV $\mathbf{y} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, then the transformed RV $\mathbf{x} = \mathbf{L}\mathbf{y}$ will have the covariance

$$\mathrm{Cov}[\mathbf{x}] = \mathbf{L}\mathbf{L}^{\mathsf{T}} = \mathbf{A}. \tag{3.8}$$

In Matlab, we can for example draw `N` indepedendent random vectors from $\mathcal{N}(\mathtt{a}, \mathtt{A})$ using

$$\mathtt{x} = \mathtt{repmat}(\mathtt{a}, [1, \mathtt{N}]) + \mathtt{cholA'*randn}(\mathtt{n}, \mathtt{N}). \tag{3.9}$$

The transpose is required because Matlab's `chol` function calculates the upper Cholesky matrix by default, and not the lower Cholesky matrix. ∎

Marginalization and conditioning are frequent tasks that must be done with the multivariate Gaussian. In the moment-based representation of the Gaussian, the former is trivial, while the latter is somewhat more complicated.

Theorem 3.2.3 — Marginalization and conditioning. Let $\mathbf{x}$ and $\mathbf{y}$ have the joint distribution

$$p(\mathbf{x}, \mathbf{y}) = \mathcal{N}\left(\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}; \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}, \begin{bmatrix} \mathbf{P}_{xx} & \mathbf{P}_{xy} \\ \mathbf{P}_{xy}^{\mathsf{T}} & \mathbf{P}_{yy} \end{bmatrix}\right)$$

Then the marginal distribution of $\mathbf{y}$ is $p(\mathbf{y}) = \mathcal{N}(\mathbf{y}; \mathbf{b}, \mathbf{P}_{yy})$, and the conditional distribution pf $\mathbf{x}$ given $\mathbf{y}$ is $p(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_{x|y}, \mathbf{P}_{x|y})$ where $\boldsymbol{\mu}_{x|y} = \mathbf{a} + \mathbf{P}_{xy}\mathbf{P}_{yy}^{-1}(\mathbf{y} - \mathbf{b})$ and $\mathbf{P}_{x|y} = \mathbf{P}_{xx} - \mathbf{P}_{xy}\mathbf{P}_{yy}^{-1}\mathbf{P}_{xy}^{\mathsf{T}}$.

Proof. For simplicity, assume that $\mathbf{a}$ and $\mathbf{b}$ are zero. We use the matrix inversion rule (P.3) to decompose the inverse covariance matrix as follows:

$$\begin{bmatrix} \mathbf{P}_{xx} & \mathbf{P}_{xy} \\ \mathbf{P}_{xy}^{\mathsf{T}} & \mathbf{P}_{yy} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{P}_{yy}^{-1}\mathbf{P}_{xy}^{\mathsf{T}} & \mathbf{I} \end{bmatrix} \begin{bmatrix} (\mathbf{P}_{xx} - \mathbf{P}_{xy}\mathbf{P}_{yy}^{-1}\mathbf{P}_{xy}^{\mathsf{T}})^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{P}_{yy}^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{I} & -\mathbf{P}_{xy}\mathbf{P}_{yy}^{-1} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}.$$

We insert this into the joint Gaussian, and obtain

$$p(\mathbf{x}, \mathbf{y}) \propto \exp\left(-\frac{1}{2}\begin{bmatrix} \mathbf{x} - \mathbf{P}_{xy}\mathbf{P}_{yy}^{-1}\mathbf{y} \\ \mathbf{y} \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} (\mathbf{P}_{xx} - \mathbf{P}_{xy}\mathbf{P}_{yy}^{-1}\mathbf{P}_{xy}^{\mathsf{T}})^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{P}_{yy}^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{x} - \mathbf{P}_{xy}\mathbf{P}_{yy}^{-1}\mathbf{y} \\ \mathbf{y} \end{bmatrix}\right).$$

Since the center matrix is diagonal, this becomes

$$p(\mathbf{x}, \mathbf{y}) \propto \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{P}_{xy}\mathbf{P}_{yy}^{-1}\mathbf{y})^{\mathsf{T}}(\mathbf{P}_{xx} - \mathbf{P}_{xy}\mathbf{P}_{yy}^{-1}\mathbf{P}_{xy}^{\mathsf{T}})^{-1}(\mathbf{x} - \mathbf{P}_{xy}\mathbf{P}_{yy}^{-1}\mathbf{y})\right) \exp\left(-\frac{1}{2}\mathbf{y}^{\mathsf{T}}\mathbf{P}_{yy}^{-1}\mathbf{y}\right).$$

Both the desired results follow from inspection of this expression. We look at marginalization first. In this case, the goal is to find $p(\mathbf{y}) = \int p(\mathbf{x}, \mathbf{y})\mathrm{d}\mathbf{x}$. Irrespectively on the value of $\mathbf{y}$, the first exponential will describe a Gaussian in $\mathbf{x}$. For this reason, the integral over $\mathbf{x}$ eliminates this entire exponential, including its dependency on $\mathbf{y}$, and we are left with the second exponential. This proves the marginalization result. Then we look at conditioning. In this case, the goal is to find $p(\mathbf{x}|\mathbf{y}) = p(\mathbf{x}, \mathbf{y})/p(\mathbf{y})$. Carrying out this division amounts to removing the second exponential, so that we are left with only the first one. That is, the conditional density $p(\mathbf{x}|\mathbf{y})$ is a Gaussian with expectation $\mathbf{P}_{xy}\mathbf{P}_{yy}^{-1}\mathbf{y}$ and covariance $\mathbf{P}_{xx} - \mathbf{P}_{xy}\mathbf{P}_{yy}^{-1}\mathbf{P}_{xy}^{\mathsf{T}}$. Re-introducing the expectations $\mathbf{a}$ and $\mathbf{b}$ at their appropriate slots is straightforward, and concludes the proof. ∎

■ **Example 3.5 — Marginalization and conditioning.** When we condition on part of a Gaussian vector, we are essentially making a cut through the ellipse at the conditioning variables. Since the presence of correlations make the ellipse narrower, as seen in Example 3.2, we would expect the uncertainty that remains after conditioning to be smaller than the marginal uncertainty. This is of course also what the formulas in Theorem 3.2.3 tell us. Figure 3.3 illustrates this.

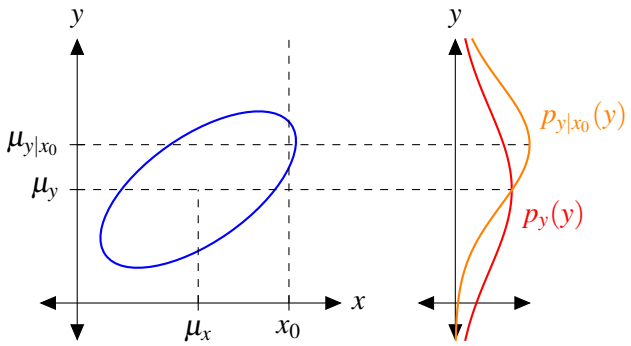


Figure 3.3: Two-dimensional Gaussian distribution. Marginal distribution (red) and conditional distribution (orange) for a value $x = x_0$.

■

3.3 The product identity

After having tasted some samples of the niceness of the multivariate Gaussian in Sections 3.2 - 3.4, it should not come as any surprise that the product of two Gaussians is another Gaussian. After all, a Gaussian is what we get when we exponentiate a negative definite quadratic form, and when we add two quadratic forms we get another quadratic form, whose exponential becomes a Gaussian after normalization. What is perhaps more surprising is that the normalization constant also becomes a Gaussian.

Recall that the expression for a Gaussian, $\mathcal{N}(\mathbf{x}; \mu, \mathbf{P})$, has two "slots" where vectors can be inserted, i.e., the argument slot and the expectation slot. What is particularly interesting is to see what happens when we have a product of two Gaussians, where $\mathbf{x}$ enters the argument slot in one of the Gaussians, and the expectation slot in the other Gaussian. This leads to the fundamental product identity.

Theorem 3.3.1 — The product identity. The following identity

$$\mathcal{N}(\mathbf{z}; \mathbf{H}\mathbf{x}, \mathbf{R})\mathcal{N}(\mathbf{x}; \bar{\mathbf{x}}, \bar{\mathbf{P}}) = \mathcal{N}(\mathbf{z}; \bar{\mathbf{z}}, \mathbf{S})\mathcal{N}(\mathbf{x}; \hat{\mathbf{x}}, \hat{\mathbf{P}}) \tag{3.10}$$

is true if the vectors and matrices involved are related according to

$$\begin{aligned}
\bar{\mathbf{z}} &= \mathbf{H}\bar{\mathbf{x}} \\
\hat{\mathbf{x}} &= \bar{\mathbf{x}} + \mathbf{W}(\mathbf{z} - \mathbf{H}\bar{\mathbf{x}}) \\
\mathbf{S} &= \mathbf{R} + \mathbf{H}\bar{\mathbf{P}}\mathbf{H}^\mathsf{T} \\
\hat{\mathbf{P}} &= (\mathbf{I} - \mathbf{W}\mathbf{H})\bar{\mathbf{P}} \\
\mathbf{W} &= \bar{\mathbf{P}}\mathbf{H}^\mathsf{T}\mathbf{S}^{-1}.
\end{aligned}$$

Proof. We can prove the product identity by showing that both the left-hand-side and the right-hand-side are two possible factorizations of a joint Gaussian $p(\mathbf{x}, \mathbf{z})$. In other words, we are going

to use the relationships

$$p(\mathbf{x},\mathbf{z}) = p(\mathbf{z}|\mathbf{x})p(\mathbf{x}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{z}). \tag{3.11}$$

Step 1: Construct the joint density. If we start by looking at the first factorization in (3.11), we may simply define $p(\mathbf{z}|\mathbf{x})$ and $p(\mathbf{x})$ as identical to the two Gaussians on the left hand side of (3.10). This also defines $p(\mathbf{x},\mathbf{z})$ as identical to their product, i.e.,

$$p(\mathbf{z},\mathbf{x}) = \mathcal{N}(\mathbf{z};\mathbf{H}\mathbf{x},\mathbf{R})\mathcal{N}(\mathbf{x};\bar{\mathbf{x}},\bar{\mathbf{P}}). \tag{3.12}$$

Step 2: Manipulate the quadratic form. The quadratic form in (3.12) can be written as follows:

$$\begin{aligned}
&(\mathbf{x}-\bar{\mathbf{x}})^{\mathsf{T}}\bar{\mathbf{P}}^{-1}(\mathbf{x}-\bar{\mathbf{x}}) + (\mathbf{z}-\mathbf{H}\mathbf{x})^{\mathsf{T}}\mathbf{R}^{-1}(\mathbf{z}-\mathbf{H}\mathbf{x}) \\
&= \begin{bmatrix}\mathbf{x}-\bar{\mathbf{x}}\\ \mathbf{z}-\mathbf{H}\mathbf{x}\end{bmatrix}^{\mathsf{T}} \begin{bmatrix}\bar{\mathbf{P}}^{-1} & \mathbf{0}\\ \mathbf{0} & \mathbf{R}^{-1}\end{bmatrix} \begin{bmatrix}\mathbf{x}-\bar{\mathbf{x}}\\ \mathbf{z}-\mathbf{H}\mathbf{x}\end{bmatrix} \\
&= \begin{bmatrix}\mathbf{x}-\bar{\mathbf{x}}\\ \mathbf{z}-\mathbf{H}\bar{\mathbf{x}}\end{bmatrix}^{\mathsf{T}} \begin{bmatrix}\mathbf{I} & -\mathbf{H}^{\mathsf{T}}\\ \mathbf{0} & \mathbf{I}\end{bmatrix} \begin{bmatrix}\bar{\mathbf{P}}^{-1} & \mathbf{0}\\ \mathbf{0} & \mathbf{R}^{-1}\end{bmatrix} \begin{bmatrix}\mathbf{I} & \mathbf{0}\\ -\mathbf{H} & \mathbf{I}\end{bmatrix} \begin{bmatrix}\mathbf{x}-\bar{\mathbf{x}}\\ \mathbf{z}-\mathbf{H}\bar{\mathbf{x}}\end{bmatrix} \\
&= \begin{bmatrix}\mathbf{x}-\bar{\mathbf{x}}\\ \mathbf{z}-\mathbf{H}\bar{\mathbf{x}}\end{bmatrix}^{\mathsf{T}} \left(\begin{bmatrix}\mathbf{I} & \mathbf{0}\\ \mathbf{H} & \mathbf{I}\end{bmatrix} \begin{bmatrix}\bar{\mathbf{P}} & \mathbf{0}\\ \mathbf{0} & \mathbf{R}\end{bmatrix} \begin{bmatrix}\mathbf{I} & \mathbf{H}^{\mathsf{T}}\\ \mathbf{0} & \mathbf{I}\end{bmatrix} \right)^{-1} \begin{bmatrix}\mathbf{x}-\bar{\mathbf{x}}\\ \mathbf{z}-\mathbf{H}\bar{\mathbf{x}}\end{bmatrix} \\
&= \begin{bmatrix}\mathbf{x}-\bar{\mathbf{x}}\\ \mathbf{z}-\mathbf{H}\bar{\mathbf{x}}\end{bmatrix}^{\mathsf{T}} \begin{bmatrix}\bar{\mathbf{P}} & \bar{\mathbf{P}}\mathbf{H}^{\mathsf{T}}\\ \mathbf{H}\bar{\mathbf{P}} & \mathbf{H}\bar{\mathbf{P}}\mathbf{H}^{\mathsf{T}}+\mathbf{R}\end{bmatrix}^{-1} \begin{bmatrix}\mathbf{x}-\bar{\mathbf{x}}\\ \mathbf{z}-\mathbf{H}\bar{\mathbf{x}}\end{bmatrix}
\end{aligned} \tag{3.13}$$

The second equality in this development is the only one that is not straightforward. The background for this equality is that we need to remove $\mathbf{x}$ from the outer vector if we are going to arrive at the product identity. This is done by subjecting the outer vector to an appropriate linear transform. More precisely, this is achieved by

$$\begin{bmatrix}\mathbf{I} & \mathbf{0}\\ -\mathbf{H} & \mathbf{I}\end{bmatrix} \begin{bmatrix}\mathbf{x}-\bar{\mathbf{x}}\\ \mathbf{z}-\mathbf{H}\bar{\mathbf{x}}\end{bmatrix} = \begin{bmatrix}\mathbf{x}-\bar{\mathbf{x}}\\ -\mathbf{H}\mathbf{x}+\mathbf{H}\bar{\mathbf{x}}+\mathbf{x}-\mathbf{H}\bar{\mathbf{x}}\end{bmatrix} = \begin{bmatrix}\mathbf{x}-\bar{\mathbf{x}}\\ \mathbf{z}-\mathbf{H}\mathbf{x}\end{bmatrix} \tag{3.14}$$

In other words, we can replace the right-hand-side of (3.14) with the left-hand-side of (3.14), which is done in the third line of (3.13).

Step 3: Factorize into marginal and conditional. The last expression in (3.13) is the quadratic form of a Gaussian in $\mathbf{x}$ and $\mathbf{z}$, and since it encapsulates all dependency on $\mathbf{x}$ and $\mathbf{z}$, we can rest assured that this Gaussian is the joint density $p(\mathbf{x},\mathbf{z})$. The covariance matrix in (3.13) is not block-diagonal, and therefore some further work is needed to split the quadratic form into a sum of two separate quadratic forms, as we need to arrive at the product identity. However, if we can construct the alternative factorization $p(\mathbf{x},\mathbf{z}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{z})$ we may perhaps achieve this. To find the quadratic forms corresponding to $p(\mathbf{z})$ and $p(\mathbf{x}|\mathbf{z})$ we use Theorem 3.2.3, which provided expressions for marginal and conditional Gaussians. If we compare with the entities involved in Theorem 3.2.3, we can make the following identifications

Role	In Theorem 3.2.3	In (3.13)
Conditioned RV	$\mathbf{x}$	$\mathbf{x}$
RV that we condition on	$\mathbf{y}$	$\mathbf{z}$
Expectation of conditioned RV	$\mathbf{a}$	$\bar{\mathbf{x}}$
Expectation of RV that we condition on	$\mathbf{b}$	$\mathbf{H}\bar{\mathbf{x}} = \bar{\mathbf{z}}$
Covariance of conditioned RV	$\mathbf{P}_{xx}$	$\bar{\mathbf{P}}$
Cross-covariance	$\mathbf{P}_{xy}$	$\bar{\mathbf{P}}\mathbf{H}^{\mathsf{T}}$
Covariance of RV that we condition on	$\mathbf{P}_{yy}$	$\mathbf{H}\bar{\mathbf{P}}\mathbf{H}^{\mathsf{T}}+\mathbf{R} = \mathbf{S}$

Based on this, we may furthermore identify $\boldsymbol{\mu}_{x|y}$ and $\mathbf{P}_{x|y}$ in Theorem 3.2.3 with the entities

$$\hat{\mathbf{x}} = \bar{\mathbf{x}} + \bar{\mathbf{P}}\mathbf{H}^\mathsf{T}(\mathbf{H}\bar{\mathbf{P}}\mathbf{H}^\mathsf{T} + \mathbf{R})^{-1}(\mathbf{z} - \mathbf{H}\bar{\mathbf{x}}) \tag{3.15}$$

$$\hat{\mathbf{P}} = \bar{\mathbf{P}} - \bar{\mathbf{P}}\mathbf{H}^\mathsf{T}(\mathbf{H}\bar{\mathbf{P}}\mathbf{H}^\mathsf{T} + \mathbf{R})^{-1}\mathbf{H}\bar{\mathbf{P}}. \tag{3.16}$$

We have thus specified all the entities involved in the quadratic forms corresponding to $p(\mathbf{z})$ and $p(\mathbf{x}|\mathbf{z})$, and it follows that

$$(\mathbf{x} - \bar{\mathbf{x}})^\mathsf{T}\bar{\mathbf{P}}^{-1}(\mathbf{x} - \bar{\mathbf{x}}) + (\mathbf{z} - \mathbf{H}\mathbf{x})^\mathsf{T}\mathbf{R}^{-1}(\mathbf{z} - \mathbf{H}\mathbf{x}) = (\mathbf{z} - \bar{\mathbf{z}})^\mathsf{T}\mathbf{S}^{-1}(\mathbf{z} - \bar{\mathbf{z}}) + (\mathbf{x} - \hat{\mathbf{x}})^\mathsf{T}\hat{\mathbf{P}}^{-1}(\mathbf{x} - \hat{\mathbf{x}})$$

In order to arrive at the final expressions in the theorem, we notice first that the term $\mathbf{W} = \bar{\mathbf{P}}\mathbf{H}^\mathsf{T}\mathbf{S}^{-1}$ is present in both (3.15) and (3.16). In the first of these, recognizing $\mathbf{W}$ leads to the expression $\hat{\mathbf{x}} = \bar{\mathbf{x}} + \mathbf{W}(\mathbf{z} - \mathbf{H}\bar{\mathbf{x}})$. In the second of these, recognizing $\mathbf{W}$ leads to $\hat{\mathbf{P}} = \bar{\mathbf{P}} - \mathbf{W}\mathbf{H}\bar{\mathbf{P}} = (\mathbf{I} - \mathbf{W}\mathbf{H})\bar{\mathbf{P}}$, and we have successfully separated (3.13) into the two quadratic forms on the right-hand-side of the product identity. ■

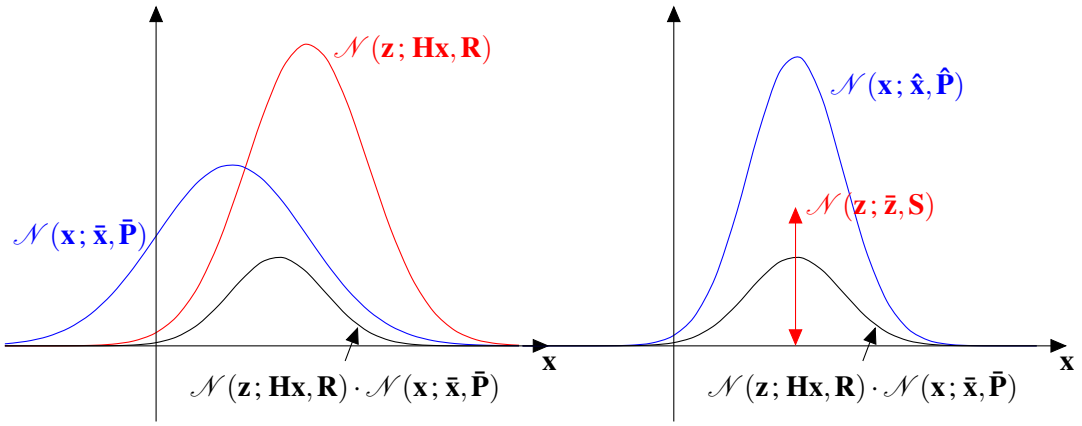


Figure 3.4: Illustration of the product identity.

3.4 The canonical form

As an alternative to the moment-based parameterization (3.1), the multivariate Gaussian can also parameterized in the canonical form, which in the multivariate case becomes

$$\mathcal{N}(\mathbf{x}; \mu, \mathbf{P}) = \exp\left(a + \boldsymbol{\eta}^\mathsf{T}\mathbf{x} - \frac{1}{2}\mathbf{x}^\mathsf{T}\boldsymbol{\Lambda}\mathbf{x}\right) \tag{3.17}$$

where

$$\boldsymbol{\Lambda} = \mathbf{P}^{-1} \tag{3.18}$$

$$\boldsymbol{\eta} = \boldsymbol{\Lambda}\mu \tag{3.19}$$

$$a = -(1/2)n\ln(2\pi) - \ln|\boldsymbol{\Lambda}| + \boldsymbol{\eta}^\mathsf{T}\boldsymbol{\Lambda}\boldsymbol{\eta}. \tag{3.20}$$

The entity $\boldsymbol{\eta}$ is sometimes referred to as an information state. The entity $\boldsymbol{\Lambda}$ is known as the information matrix or precision matrix. The term "information" refers to the fact that $\boldsymbol{\Lambda}$, being the inverse of $\mathbf{P}$, is a measure of the opposite of uncertainty.

We saw in the previous section that in the moment-based parametrization, marginalization was easy while conditioning was more complicated. In canonical form, it is opposite.

Theorem 3.4.1 — Marginalization and conditioning in canonical form. If $\mathbf{x}$ and $\mathbf{y}$ have the joint distribution

$$p(\mathbf{x},\mathbf{y}) \propto \exp\left(\begin{bmatrix} \boldsymbol{\eta}_a^\mathsf{T} & \boldsymbol{\eta}_b^\mathsf{T} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} - \frac{1}{2} \begin{bmatrix} \mathbf{x}^\mathsf{T} & \mathbf{y}^\mathsf{T} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Lambda}_{xx} & \boldsymbol{\Lambda}_{xy} \\ \boldsymbol{\Lambda}_{xy}^\mathsf{T} & \boldsymbol{\Lambda}_{yy} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \right) \tag{3.21}$$

then the marginal distribution of $\mathbf{y}$ is has potential vector $\boldsymbol{\eta}_* = \boldsymbol{\eta}_b - \boldsymbol{\Lambda}_{xy}^\mathsf{T}\boldsymbol{\Lambda}_{xx}^{-1}\boldsymbol{\eta}_a$ and information matrix $\boldsymbol{\Lambda}_* = \boldsymbol{\Lambda}_{yy} - \boldsymbol{\Lambda}_{xy}^\mathsf{T}\boldsymbol{\Lambda}_{xx}^{-1}\boldsymbol{\Lambda}_{xy}$, and the conditional distribution of $\mathbf{x}$ given $\mathbf{y}$ has potential vector $\boldsymbol{\eta}_{x|y} = \boldsymbol{\eta}_a - \boldsymbol{\Lambda}_{xy}\boldsymbol{\eta}_b$ and information matrix $\boldsymbol{\Lambda}_{x|y} = \boldsymbol{\Lambda}_{xx}$.

Proof. First we show the marginalization, and then we use this to show the conditioning. For the marginalized density, the information matrix must be the inverse of the corresponding marginalized covariance matrix. This means that

$$\begin{aligned}
\boldsymbol{\Lambda}_* =& \mathbf{P}_{yy}^{-1} \\
=& \left([\mathbf{0},\mathbf{I}] \begin{bmatrix} \mathbf{P}_{xx} & \mathbf{P}_{xy} \\ \mathbf{P}_{xy}^\mathsf{T} & \mathbf{P}_{yy} \end{bmatrix} \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix} \right)^{-1} = \left([\mathbf{0},\mathbf{I}] \begin{bmatrix} \boldsymbol{\Lambda}_{xx} & \boldsymbol{\Lambda}_{xy} \\ \boldsymbol{\Lambda}_{xy}^\mathsf{T} & \boldsymbol{\Lambda}_{yy} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix} \right)^{-1} \\
=& \left([\mathbf{0},\mathbf{I}] \begin{bmatrix} \cdots & \cdots \\ \cdots & (\boldsymbol{\Lambda}_{yy} - \boldsymbol{\Lambda}_{xy}^\mathsf{T}\boldsymbol{\Lambda}_{xx}^{-1}\boldsymbol{\Lambda}_{xy})^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix} \right)^{-1} \\
=& \boldsymbol{\Lambda}_{yy} - \boldsymbol{\Lambda}_{xy}^\mathsf{T}\boldsymbol{\Lambda}_{xx}^{-1}\boldsymbol{\Lambda}_{xy}.
\end{aligned} \tag{3.22}$$

We have inverted the $\boldsymbol{\Lambda}$ matrix by means of the inverse block matrix formula from Appendix P.1. Since the inverted matrix is to be pre- and post-multiplied by the matrix $[\mathbf{0},\mathbf{I}]$ we have simply written "…" instead of spelling out the irrelevant blocks. In order to express the marginalized potential, we first express the marginalized expectation in terms of the joint potential vector:

$$\mathbf{b} = [\mathbf{0},\mathbf{I}] \begin{bmatrix} \boldsymbol{\Lambda}_{xx} & \boldsymbol{\Lambda}_{xy} \\ \boldsymbol{\Lambda}_{xy}^\mathsf{T} & \boldsymbol{\Lambda}_{yy} \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{\eta}_a \\ \boldsymbol{\eta}_b \end{bmatrix} \tag{3.23}$$

The marginalized potential vector can then be found as

$$\boldsymbol{\eta}_* = \boldsymbol{\Lambda}_*\mathbf{b} = \boldsymbol{\Lambda}_*(\boldsymbol{\Lambda}_*^{-1}\boldsymbol{\Lambda}_{yy} - \boldsymbol{\Lambda}_*^{-1}\boldsymbol{\Lambda}_{xy}^\mathsf{T}\boldsymbol{\Lambda}_{xx}^{-1}\boldsymbol{\eta}_a) = \boldsymbol{\eta}_b - \boldsymbol{\Lambda}_{xy}^\mathsf{T}\boldsymbol{\Lambda}_{xx}^{-1}\boldsymbol{\eta}_a. \tag{3.24}$$

The conditional Gaussian $p(\mathbf{x}|\mathbf{y})$ can then be found according to

$$\begin{aligned}
p(\mathbf{x}|\mathbf{y}) =& \frac{p(\mathbf{x},\mathbf{y})}{p(\mathbf{y})} \\
\propto& \exp\left(\boldsymbol{\eta}_a^\mathsf{T}\mathbf{x} + \boldsymbol{\eta}_b^\mathsf{T}\mathbf{y} - \frac{1}{2}\left(\mathbf{x}^\mathsf{T}\boldsymbol{\Lambda}_{xx}\mathbf{x} + \mathbf{x}^\mathsf{T}\boldsymbol{\Lambda}_{xy}\mathbf{y} + \mathbf{y}^\mathsf{T}\boldsymbol{\Lambda}_{xy}^\mathsf{T}\mathbf{x} + \mathbf{y}^\mathsf{T}\boldsymbol{\Lambda}_{yy}\mathbf{y} \right) \right. \\
& \left. - \left(\boldsymbol{\eta}_b - \boldsymbol{\Lambda}_{xy}^\mathsf{T}\boldsymbol{\Lambda}_{xx}^{-1}\boldsymbol{\eta}_a \right)^\mathsf{T}\mathbf{y} - \frac{1}{2}\mathbf{y}^\mathsf{T}\left(\boldsymbol{\Lambda}_{yy} - \boldsymbol{\Lambda}_{xy}^\mathsf{T}\boldsymbol{\Lambda}_{xx}^{-1}\boldsymbol{\Lambda}_{xy} \right)\mathbf{y} \right) \\
\propto& \exp\left(\boldsymbol{\eta}_a^\mathsf{T}\mathbf{x} - \mathbf{y}^\mathsf{T}\boldsymbol{\Lambda}_{xy}\mathbf{x} - \frac{1}{2}\mathbf{x}^\mathsf{T}\boldsymbol{\Lambda}_{xx}\mathbf{x} \right)
\end{aligned} \tag{3.25}$$

The last proportionality is obtained by discarding all terms that do not contain $\mathbf{x}$, as they provide no information about the shape of the conditional distribution of $\mathbf{x}$. We see that $\boldsymbol{\eta}_a^\mathsf{T} - \mathbf{y}^\mathsf{T}\boldsymbol{\Lambda}_{xy}$ plays the role of the potential vector, while $\boldsymbol{\Lambda}_{xx}$ plays the role of the information matrix in the conditional distribution. ■

The canonical form is useful for several reasons. First, as shown in Theorem 3.4.1, conditioning is in general easier to do in canonical form than in the moment-based form. Since estimation often boils down to calcuating the marginal density of state, given the data, this is not unimportant. Furthermore, as already pointed out in (3.4), the information matrix equals the curvature of the logarithm of the Gaussian.

3.5 References and chapter remarks

The multivariate Gaussian is so important that extensive treatments can be found in many textbooks. From a statistical perspective, a standard reference is [56]. Readers more inclined towards machine learning, may prefer machine learning textbooks such as [76] or [10]. Sensor fusion textbooks such as [4] and [48] also contain all the standard stuff. The main purpose for the treatment in this book has been to present the most important results in manner as condensed and pedagogical as possible, while still provide rigorous proofs for all the results. It is therefore important to be aware that there is much more to read in the mentioned textbooks.

The strong focus on understanding the Gaussian as the exponential of a quadratic form is somewhat original to this book, although a similar perspective can also be found in [10]. The proof of Theorem 3.2.3 (marginalization and conditioning of Gaussians in moment-form) is due to Gary B. Huang [52], and goes along the lines of the proof in [76]. An alternative proof, formulated in terms of random variables, can be found in [56]. A third proof, which uses completion of the square, is found in [95]. The proof of the product identity follows the structure of the proof in [106]. Alternative proofs can be found in [92] and [71].

3.6 Exercises

Exercise 3.1 Let $\mathbf{A}$ and $\mathbf{B}$ be symmetric and invertible matrices of the same dimension. Show that

$$\mathbf{A} - \mathbf{A}(\mathbf{A}+\mathbf{B})^{-1}\mathbf{A} = (\mathbf{A}^{-1}+\mathbf{B}^{-1})^{-1}.$$

Hint: The equality holds if and only if the inverse of the right-hand-side times the left-hand-side equals the identity matrix. ∎

Exercise 3.2 Let $\mathbf{z} = \mathbf{H}\mathbf{x} + \mathbf{w}$ where $\mathbf{x}$ and $\mathbf{w}$ are independent random vectors distributed according to $\mathcal{N}(\mathbf{x}; |\bar{\mathbf{x}}, \mathbf{P})$ and $\mathcal{N}(\mathbf{w}; |\mathbf{0}, \mathbf{R})$. Show that the joint distribution of $\mathbf{x}$ and $\mathbf{z}$ is

$$\mathcal{N}\left(\begin{bmatrix}\mathbf{x}\\ \mathbf{z}\end{bmatrix}; \begin{bmatrix}\bar{\mathbf{x}}\\ \mathbf{0}\end{bmatrix}, \begin{bmatrix}\mathbf{P} & \mathbf{P}\mathbf{H}^{\mathsf{T}}\\ \mathbf{H}\mathbf{P} & \mathbf{H}\mathbf{P}\mathbf{H}^{\mathsf{T}}+\mathbf{R}\end{bmatrix}\right).$$

∎

Exercise 3.3 Derive the covariance formula $\hat{\mathbf{P}}^{-1} = \mathbf{H}^{\mathsf{T}}\mathbf{R}^{-1}\mathbf{H} + \bar{\mathbf{P}}^{-1}$. ∎

Exercise 3.4 Let $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{P})$ be an n-dimensional RV. Show that $(\mathbf{x}-\mu)^{\mathsf{T}}\mathbf{P}^{-1}(\mathbf{x}-\mu)$ has a χ^2 distribution with n degrees of freedom.
Hint: You can build on relevant results from Examples 2.9, 2.10 and 3.4 as well as Theorem 3.2.2 and Exercise 2.2. ∎

4. The Kalman Filter From a Bayesian Perspective

Typical applications of sensor fusion involve temporal processing of sensor data. We are dealing with a dynamical system whose state we want to estimate as measurements arrive. For example, in a dynamical positioning system, we may receive measurements of position and orientation, and want to estimate the full state vector including position, orientation, velocities and bias terms for unmodeled dynamics, every time a measurement is received. This is known as state estimation. It is also commonly referred to as filtering in the estimation literature. As mentioned in Chapter 1 we can solve address state estimation with techniques based on both deterministic and probabilistic theory. On the latter is a focus here.

Having established a probabilistic framework in Chapters 2 - 3, we formulate the probabilistic filtering problem in relatively general terms in Section 4.1. After this is done, we invoke the common assumptions of Gaussianity and linearity in Section 4.2. This leads directly to the Kalman filter. In the subsequent sections we discuss how we can tune the Kalman filter according to knowledge about the noise processes.

4.1 The Bayes filter

In the filtering problem, the state of a stochastic dynamic system is to be estimated from a series of noisy measurements. By definition of the concept of a state vector, all information that describes the system at a given time is contained in the state vector.

■ **Example 4.1** In the most common kinematic model for target tracking, the so-called constant velocity model (CV model), the state vector contains only position and velocity. For a two-dimensional scenario, the state vector then becomes

$$\mathbf{x} = \begin{bmatrix} \rho_x \\ \rho_y \\ \mathbf{v}_x \\ \mathbf{v}_y \end{bmatrix} \tag{4.1}$$

where ρ_x is the position of the target along the x-coordinate etc. This entails an assumption that any

knowledge about, e.g., accelerations, previously or at this time, would not provide any additional information about how the target is going to move in the future. ∎

The filtering problem is formulated in terms of two models. The first model is known as the Markov model, kinematic prior or plant model, and concerns how the state evolves in time. This model can be specified as $p(\mathbf{x}_k \,|\, \mathbf{x}_{k-1})$. The second model is known as the measurement model or likelihood, and concerns how the measurements that we actually observe are related to the state vector. This model can be specified as $p(\mathbf{z}_k \,|\, \mathbf{x}_k)$.

A couple of independence assumptions underlie this way of treating the filtering problem. For the given specification of the plant model to make sense, the Markov property must hold:

$$p(\mathbf{x}_k \,|\, \mathbf{x}_1, \ldots, \mathbf{x}_{k-2}, \mathbf{x}_{k-1}, \mathbf{z}_1, \ldots, \mathbf{z}_{k-2}, \mathbf{z}_{k-1}) = p(\mathbf{x}_k \,|\, \mathbf{x}_{k-1}). \tag{4.2}$$

In other words, $\mathbf{x}_k$ should be independent of $\mathbf{x}_0, \ldots, \mathbf{x}_{k-1}$ when $\mathbf{x}_k$ is given. A similar requirement is assumed to hold for the measurements:

$$p(\mathbf{z}_k \,|\, \mathbf{x}_1, \ldots, \mathbf{x}_{k-2}, \mathbf{x}_{k-1}, \mathbf{x}_k, \mathbf{z}_1, \ldots, \mathbf{z}_{k-2}, \mathbf{z}_{k-1}) = p(\mathbf{z}_k \,|\, \mathbf{x}_k). \tag{4.3}$$

We can visualize this structure using the graphical model in Figure 4.1. The independence assumptions manifest themselves as follows: The only node that affects $\mathbf{x}_{k+1}$ is $\mathbf{x}_k$, and the only node that affects $\mathbf{z}_k$ is also $\mathbf{x}_k$. Every horizontal arrow connects two state nodes at different times according to the process model. Every vertical arrow connects a measurement node with the corresponding state node according to the measurement model.

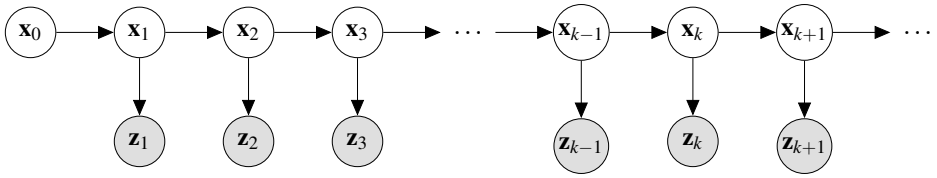


Figure 4.1: The probabilistic graphical model that underlies recursive Bayesian estimation.

Now that we have specified the filtering problem, it remains to solve it. In the Bayesian approach, we consider the posterior distribution $p(\mathbf{x}_k \,|\, \mathbf{z}_{1:k})$ as the solution to the filtering problem. The notation $\mathbf{z}_{1:k}$ signifies the sequence of all the measurements $\mathbf{z}_1, \ldots, \mathbf{z}_k$. The posterior distribution contains all the information that we have available about $\mathbf{x}_k$ conditional on these measurements. Once we have the posterior distribution specified, we can calculate various estimates of $\mathbf{x}_k$ based on MAP estimation, MMSE estimation etc.

The filtering is done in a cyclic manner, which consists of a prediction step and an update step. The prediction step, also known as the Chapman-Kolmogorov equation, follows from the total probability theorem and is given by

$$p(\mathbf{x}_k|\mathbf{z}_{1:k-1}) \;=\; \int p(\mathbf{x}_k, \mathbf{x}_{k-1} \,|\, \mathbf{z}_{1:k-1}) \mathrm{d}\mathbf{x}_{k-1} = \int p(\mathbf{x}_k|\mathbf{x}_{k-1}) p(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1}) \mathrm{d}\mathbf{x}_{k-1}. \tag{4.4}$$

The update step follows from Bayes' rule and is given by

$$p(\mathbf{x}_k|\mathbf{z}_{1:k}) \;=\; \frac{p(\mathbf{z}_k|\mathbf{x}_k) p(\mathbf{x}_k|\mathbf{z}_{1:k-1})}{p(\mathbf{z}_k \,|\, \mathbf{z}_{1:k-1})} \propto p(\mathbf{z}_k|\mathbf{x}_k) p(\mathbf{x}_k|\mathbf{z}_{1:k-1}). \tag{4.5}$$

Equations (4.4) and (4.5) are together known as the Bayes filter. These equations are the foundations for everything that follows in this book.

4.2 The Kalman filter

We cannot expect to find closed-form solutions to the Bayes filter in general. Most candidates for the pdfs $p(\mathbf{x}_k|\mathbf{x}_{k-1})$ and $p(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1})$ will not lead to any closed-form expression for the Chapman-Kolmogorov integral. Furthermore, when we multiply the pdfs $p(\mathbf{z}_k|\mathbf{x}_k)$ and $p(\mathbf{x}_k|\mathbf{z}_{1:k-1})$ in the Bayes formula, we have no guarantee that the normalized product will be recognizable as a known pdf. So, are there any circumstances under which we are guaranteed to get a closed-form solution?

The answer to this question is that a closed-form solution exist when both the Markov model and the likelihood are Gaussian and linear, and the initial density is Gaussian as well. That is, these models must be of the forms

$$\begin{aligned} p(\mathbf{x}_k|\mathbf{x}_{k-1}) &= \mathcal{N}(\mathbf{x}_k\,;\mathbf{F}\mathbf{x}_{k-1},\mathbf{Q}) \\ p(\mathbf{z}_k|\mathbf{x}_k) &= \mathcal{N}(\mathbf{z}_k\,;\mathbf{H}\mathbf{x}_k,\mathbf{R}) \\ p(\mathbf{x}_0) &= \mathcal{N}(\mathbf{x}_0\,;\hat{\mathbf{x}}_0,\mathbf{P}_0). \end{aligned} \tag{4.6}$$

This is often written in the equivalent form

$$\mathbf{x}_k = \mathbf{F}\mathbf{x}_{k-1} + \mathbf{v}_k, \qquad \mathbf{v}_k \sim \mathcal{N}(0,\mathbf{Q}) \tag{4.7}$$

$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{w}_k, \qquad \mathbf{w}_k \sim \mathcal{N}(0,\mathbf{R}) \tag{4.8}$$

$$\mathbf{x}_0 \sim \mathcal{N}(\hat{\mathbf{x}}_0,\mathbf{P}_0). \tag{4.9}$$

which is more similar to the conventional state space formulation of linear systems in deterministic systems theory. Some remarks on notation are in order. The matrix $\mathbf{F}$ is known as the transition matrix while the matrix $\mathbf{H}$ is known as the measurement matrix. The symmetric positive definite matrices $\mathbf{Q}$ and $\mathbf{R}$ govern the statistical properties of the plant noise $\mathbf{v}_k$ and measurement noise $\mathbf{w}_k$, respectively. All noise vectors are assumed mutually independent. This follows from the Markov assumptions (4.2) and (4.3). The fact that $\mathbf{v}_k$ is independent of $\mathbf{v}_l$ for all $l \neq k$ is referred to as whiteness. Notice that whiteness does not imply Gaussianity, nor does Gaussianity imply whiteness.

Under these assumptions, the Kalman filter is an optimal solution to the estimation problem. It consists of the expressions provided in Algorithm 1:

Algorithm 1 The Kalman filter

procedure KF($\hat{\mathbf{x}}_{k-1}$, $\mathbf{P}_{k-1}$, $\mathbf{z}_k$)
- $\hat{\mathbf{x}}_{k|k-1} \leftarrow \mathbf{F}\hat{\mathbf{x}}_{k-1}$ ▷ The predicted state estimate
- $\mathbf{P}_{k|k-1} \leftarrow \mathbf{F}\mathbf{P}_{k-1}\mathbf{F}^\mathsf{T} + \mathbf{Q}$ ▷ The predicted covariance
- $\hat{\mathbf{z}}_{k|k-1} \leftarrow \mathbf{H}\hat{\mathbf{x}}_{k|k-1}$ ▷ The predicted measurement
- $\boldsymbol{\nu}_k \leftarrow \mathbf{z}_k - \hat{\mathbf{z}}_{k|k-1}$ ▷ The innovation
- $\mathbf{S}_k \leftarrow \mathbf{H}\mathbf{P}_{k|k-1}\mathbf{H}^\mathsf{T} + \mathbf{R}$ ▷ The innovation covariance
- $\mathbf{W}_k \leftarrow \mathbf{P}_{k|k-1}\mathbf{H}^\mathsf{T}\mathbf{S}_k^{-1}$ ▷ The Kalman gain
- $\hat{\mathbf{x}}_k \leftarrow \hat{\mathbf{x}}_{k|k-1} + \mathbf{W}_k\boldsymbol{\nu}_k$ ▷ The posterior state estimate
- $\mathbf{P}_k \leftarrow (\mathbf{I} - \mathbf{W}_k\mathbf{H})\mathbf{P}_{k|k-1}$ ▷ The posterior covariance
- **return** $\hat{\mathbf{x}}_k$, $\mathbf{P}_k$, $\hat{\mathbf{x}}_{k|k-1}$, $\mathbf{P}_{k|k-1}$, $\mathbf{S}_k$

end procedure

It is important to be familiar with all the equations and terminology in Algorithm 1. Keep in mind that some of the entities have different equivalent expressions, of which only one is given in Algorithm 1. In particular, the posterior covariance $\mathbf{P}_k$ can be written in more lengthy forms with

better numerical properties, such as the Joseph form

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{W}_k\mathbf{H})\mathbf{P}_{k|k-1}(\mathbf{I} - \mathbf{W}_k\mathbf{H})^\mathsf{T} + \mathbf{W}\mathbf{R}\mathbf{W}^\mathsf{T}. \tag{4.10}$$

Recall that our aim is to obtain expressions for the predicted pdf $p(\mathbf{x}_k|\mathbf{z}_{1:k-1})$ and the posterior pdf $p(\mathbf{x}_k|\mathbf{z}_{1:k})$. The Kalman filter provides such expressions, and the relationship is explained in Theorems 4.2.1 and 4.2.1.

Theorem 4.2.1 — The Kalman filter prediction. If the posterior density at the previous time step is $p(\mathbf{x}_{k-1}\,|\,\mathbf{z}_{1:k-1}) = \mathcal{N}(\mathbf{x}_{k-1},;\hat{\mathbf{x}}_{k-1},\mathbf{P}_{k-1})$ and the Markov model is given as in (4.6), then the predicted density is $p(\mathbf{x}_k\,|\,\mathbf{z}_{1:k-1}) = \mathcal{N}(\mathbf{x}_k,;\hat{\mathbf{x}}_{k|k-1},\mathbf{P}_{k|k-1})$ where $\hat{\mathbf{x}}_{k|k-1}$ and $\mathbf{P}_{k|k-1}$ are as given in Algorithm 1.

Proof. We prove this by means of the fundamental product identity that was derived in Section 3.3. The predicted density is given by

$$\begin{aligned}
p(\mathbf{x}_k|\mathbf{z}_{1:k-1}) &= \int p(\mathbf{x}_k|\mathbf{x}_{k-1})p(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1})\mathrm{d}\mathbf{x}_{k-1} \\
&= \int \mathcal{N}(\mathbf{x}_k\,;\mathbf{F}\mathbf{x}_{k-1},Q)\mathcal{N}(\mathbf{x}_{k-1}\,;\hat{\mathbf{x}}_{k-1},\mathbf{P}_{k-1})\mathrm{d}\mathbf{x}_{k-1} \\
&= \mathcal{N}(\mathbf{x}_k\,;\mathbf{F}\hat{\mathbf{x}}_{k-1},\mathbf{F}\mathbf{P}_{k-1}\mathbf{F}^\mathsf{T}+Q) \\
&\quad \cdot \int \mathcal{N}(\mathbf{x}_{k-1}\,;\text{ some vector }\,,\text{ some covariance matrix }\,)\mathrm{d}\mathbf{x}_{k-1} \\
&= \mathcal{N}(\mathbf{x}_k\,;\hat{\mathbf{x}}_{k|k-1},\mathbf{P}_{k|k-1}).
\end{aligned} \tag{4.11}$$

where $\hat{\mathbf{x}}_{k|k-1}$ and $\mathbf{P}_{k|k-1}$ are as given in the theorem. ■

Theorem 4.2.2 — The Kalman filter update. If the predicted density at the current time step is $p(\mathbf{x}_k\,|\,\mathbf{z}_{1:k-1}) = \mathcal{N}(\mathbf{x}_{k-1},;\hat{\mathbf{x}}_{k|k-1},\mathbf{P}_{k|k-1})$ and the likelihood is given as in (4.6), then the posterior density is $p(\mathbf{x}_k\,|\,\mathbf{z}_{1:k}) = \mathcal{N}(\mathbf{x}_k,;\hat{\mathbf{x}}_k,\mathbf{P}_k)$ where $\hat{\mathbf{x}}_k$ and $\mathbf{P}_k$ are as given in Algorithm 1.

Proof. Again the proof is an application of the fundamental product identity. The posterior density is given by

$$\begin{aligned}
p(\mathbf{x}_k|\mathbf{z}_{1:k}) &\propto p(\mathbf{z}_k|\mathbf{x}_k)\,p(\mathbf{x}_k|\mathbf{z}_{1:k-1}) \\
&= \mathcal{N}(\mathbf{z}_k\,;\mathbf{H}\mathbf{x}_k,\mathbf{R})\,\mathcal{N}(\mathbf{x}_k\,;\hat{\mathbf{x}}_{k|k-1},\mathbf{P}_{k|k-1}) \\
&= \mathcal{N}(\mathbf{z}_k\,;\mathbf{H}\hat{\mathbf{x}}_{k|k-1},\mathbf{H}\mathbf{P}_{k|k-1}\mathbf{H}^\mathsf{T}+\mathbf{R})\mathcal{N}(\mathbf{x}_k\,;\hat{\mathbf{x}}_k,\mathbf{P}_k) \\
&\propto \mathcal{N}(\mathbf{x}_k\,;\hat{\mathbf{x}}_k,\mathbf{P}_k).
\end{aligned} \tag{4.12}$$

where $\hat{\mathbf{x}}_k$ and $\mathbf{P}_k$ are as given in the theorem. ■

This way of understanding the Kalman filter has both benefits and disadvantages compared to other approaches. As the reader hopefully can agree, the proof of the Kalman filter becomes quite trivial once the product identity is accepted as a premise. Proving the product identity was, however, non-trivial. When we later will extend the Kalman filter to tracking methods that operate in scenarios with multiple targets and clutter, the product identity will play an important role, so it is important to be familiar with it. Also, working directly in terms of Gaussian distributions, instead of only with expectation vectors and covariance matrices, highlights the relationship between the Kalman filter and the optimal Bayes filter. A disadvantage of this approach is that it fails to convey that the Kalman filter also has some interesting optimality properties when the noise is non-Gaussian, but zero-mean with a given variance, which Section 2.6.5 on LMMSE estimation

hinted at. Neither does it tell us anything about the stability of the Kalman filter from a control theoretic perspective.

We see that the design of a Kalman filter largely consists of choosing the matrices $\mathbf{F}$, $\mathbf{H}$, $\mathbf{Q}$ and $\mathbf{R}$. While $\mathbf{F}$ and $\mathbf{H}$ typically is given by the model, some degree of tuning is often required to set appropriate values in the covariance matrices $\mathbf{Q}$ and $\mathbf{R}$. In the remainder of this chapter we will discuss this in greater detail. Since the plant model in many tracking problems is most naturally specified in continuous time, the determination of $\mathbf{Q}$ is intimately linked with discretization of continuous time stochastic systems. For this reason, we need to take a closer look at the theory of stochastic processes.

4.3 Stochastic processes

We can view a stochastic process as an infinite-dimensional random variable. Recall that the possible outcomes of a scalar random variable are different numbers in $\mathbb{R}$. Similarly, the possible outcomes of a random vector (i.e., vector valued random variable) are different vectors in $\mathbb{R}^n$. For a (vector valued) stochastic process, the possible outcomes are different *functions* of the form $\mathbf{x}(t) : \mathbb{R} \to \mathbb{R}^n$. That is, a stochastic process describes a random function of time. The reader is strongly encouraged to spend some time digesting the following examples to get some tastes of the different structures that stochastic processes may have.

■ **Example 4.2 — The Wiener process.** This is the grandfather of most of the interesting stochastic processes. We first define a stochastic process $x(t)$ by

$$x(nT) = \sum_{i=1}^{n} x_i \tag{4.13}$$

where x_i $(i = 1, \ldots, n)$ are i.i.d. random variables with expectation 0 and variance T. The Wiener process $w(t)$ can then be defined as the limit

$$b(t) = \lim_{T \to 0} x(t). \tag{4.14}$$

Notice that we only need to specify the expectation and variance of the variables x_i, while their exact distribution can be arbitrary as long as they are i.i.d.. This is because of the central limit theorem. In the limit $T \to 0$ the number of i.i.d. random variables x_i that contribute to $b(t)$ will go towards infinity. Then the central limit theorem guarantees that the distribution of $b(t)$ is Gaussian. Furthermore, it is easy to determine the expectation and covariance of this Gaussian, for all values of t. Its expectation is zero, because the expectation of a sum of Gaussian random variables is the same as the sum of their individual expectations, which all are zero. A similar argument reveals that the variance is

$$E[b(t)^2] = \text{Var}\left[\sum_{i=1}^{n} x_i\right] = nT = \frac{t}{T}T = t. \tag{4.15}$$

The key step in (4.15) is recognizing that the number of steps n is equal to the time t divided by the time step T. Thus, we see that the variance incrases linearly as well move away from the $t = 0$. The Wiener process is often described as a random walk, and this behavior is illustrated in Figure 4.2.

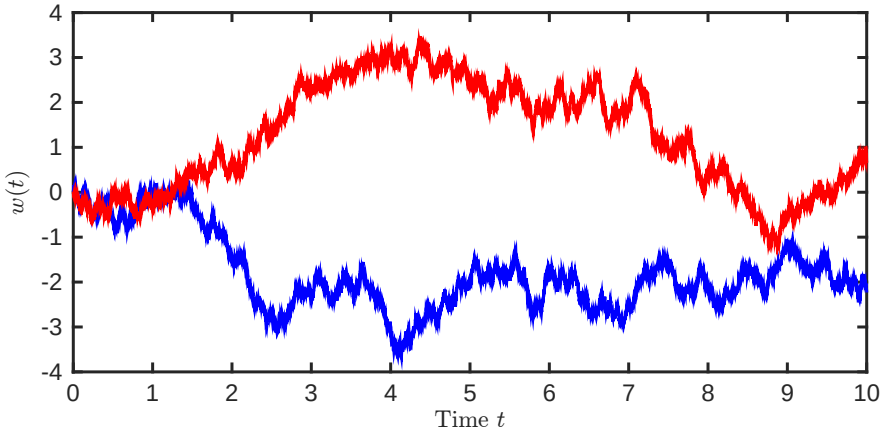


Figure 4.2: Two realizations of the Wiener process

■

■ **Example 4.3 — White Gaussian noise.** It is easy to define the concept of a *discrete-time* white Gaussian noise process, i.e., a white sequence of random variables: We just need to insist that the sequence consists of i.i.d. Gaussian random variables. For a *continuous-time process* $n(t)$ the concept of whiteness is more troublesome. First, we need a tangible definition. Here we define white noise as the derivative of the Wiener process:

$$n(t) = \lim_{\Delta \to 0} \frac{b(t+\Delta) - b(t)}{\Delta}. \tag{4.16}$$

We shall later show that this process indeed has the whiteness property when $\Delta \to 0$: In that limit, $n(t_1)$ is inependendent of $n(t_2)$ whenever $t_1 \neq t_2$. The reader may then ask: Why did we not simply define white Gaussian noise as the increments x_i that we used to construct the Wiener process? The answer has to do with the scaling of the variance. Let us take a look at the following two questions, which seem innocent enough:

a) What is $\mathrm{Var}\left[\int_0^s n(t)\mathrm{d}t\right]$?
b) What is $\mathrm{Var}\left[n(t)\right]$?

Question **a)** is important because we generally want to use white noise as a driving mechanism: In order to see how a linear system governed by a differential equation responds to white noise we must be able to say something meaningful about integrals such as the one in question **a)**. Clearly, in most applications, we would like this variance to be a finite number, neither zero nor infinity. Together with the requirement that $n(t_1)$ should be independent of $n(t_2)$ for all $t_1 \neq t_2$, this implies that the answer to question **b)** is infinity. To see this, let us first define the following approximation to the integral:

$$S = \sum_{k=0}^{t/\Delta} \Delta n(k\Delta) \approx \int_0^t n(\tau)\mathrm{d}\tau \tag{4.17}$$

The variance of S is then given by

$$\mathrm{Var}\left[S\right] = \mathrm{Var}\left[\sum_{k=0}^{t/\Delta} \Delta n(k\Delta)\right] = \Delta^2 \mathrm{Var}\left[\sum_{k=0}^{t/\Delta} n(k\Delta)\right] = \Delta^2 \left(\frac{t}{\Delta}\right) \mathrm{Var}\left[n(t)\right] \tag{4.18}$$

Solving for $\mathrm{Var}[n(t)]$ yields

$$\mathrm{Var}[n(t)] = \frac{1}{t\Delta}\mathrm{Var}[S] = \frac{1}{\Delta} \tag{4.19}$$

where the last step follows from the requirement that S should be identical with the Wiener process. For $n(t)$ to be truly white we need to take the limit $\Delta \to 0$, which leads to the blow-up of $\mathrm{Var}[n(t)]$. Therefore it would have been problematic to define white noise simply as the increments of the Wiener process. ∎

∎ **Example 4.4 — A random constant.** Let the function $f(t)$ be given by $f(t) = a$ where $a \sim \mathcal{N}(0,1)$. In contrast to the previous examples, this is a very simply stochastic process which does not require any sophisticated machinery. Again, it is interesting to study time integrals of the process. We find that

$$\mathrm{Var}\left[\int_0^\tau f(\tau)\mathrm{d}\tau\right] = \mathrm{Var}[at] = t^2. \tag{4.20}$$

By comparing this with (4.15) we see that the uncertainty caused by a integration of a fixed bias increases faster than the uncertainty caused by integration of white noise. This should not come as a surprise, since white noise has a self-cancelling property that the fixed bias does not have. ∎

4.3.1 Stationarity and the autocorrelation function

Let us then return to the general theory of stochastic properties. In principle, the pdf of an arbitrary stochastic process would need to involve an infinite number of entries. It would not only need to contain the full distributions of $\mathbf{x}(t)$ for all $t \in \mathbb{R}$, but it would also need to contain the joint distributions for all tuples $\mathbf{x}(t_1), \ldots, \mathbf{x}(t_k)$ for any number k. However, such a complicated specification is obviously not needed for the examples given above. We can restrict the general collection of arbitrary stochastic processes to classes of nice and tractable stochastic processes. One such class which obviously is nice to work with is the class of all Gaussian processes. Another, and unrelated, concept that tells us something about which processes are nicer than other processes, is stationarity. The literature makes a distinction between strict-sense-stationary and wide-sense-stationary. We will only study the latter kind of stationarity here, since it is most useful from a practical perspective.

Wide-sense stationary is described in terms of the autocorrelation function (ACF). For an arbitrary stochastic process it is defined as the expectation

$$R(t_1,t_2) = E[\mathbf{x}(t_1)\mathbf{x}(t_2)^\mathsf{T}]. \tag{4.21}$$

A stochastic process is said to be wide sense stationary if its mean is constant

$$E[\mathbf{x}(t)] = \eta \tag{4.22}$$

and the ACF can be written as a function of $\tau = t_2 - t_1$:

$$R(\tau) = E[\mathbf{x}(t)\mathbf{x}(t+\tau)^\mathsf{T}]. \tag{4.23}$$

In other words, for a wide sense stationary process, the correlations between $\mathbf{x}(t)$ at different times should only depend on the difference between these times. More explicitly, if we define $\mathbf{x}_1 = \mathbf{x}(t)$ and $\mathbf{x}_2 = \mathbf{x}(t+\tau)$, then we can also write the autocorrelation function as

$$R(\tau) = \int\int \mathbf{x}_1\mathbf{x}_2^\mathsf{T} p(\mathbf{x}_1,\mathbf{x}_2)\mathrm{d}\mathbf{x}_1\mathrm{d}\mathbf{x}_2. \tag{4.24}$$

Another nice property for a stochastic process to possess is Gaussianity. For a wide-sense stationary process this means that the joint density $p(\mathbf{x}_1, \mathbf{x}_2)$ always is a Gaussian. If the constant mean η is zero, then this distribution is of the form

$$p(\mathbf{x}_1, \mathbf{x}_2) = \mathcal{N}\left(\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix}; \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} R(0) & R(\tau) \\ R(\tau) & R(0) \end{bmatrix} \right). \tag{4.25}$$

Let us first look at the ACF of a non-stationary process.

Theorem 4.3.1 The Wiener process is non-stationary, and its ACF is given by $R(t_1, t_2) = \min(t_1, t_2)$.

Proof. Assume that $t_1 < t_2$. Since the Wiener process has independent increments we have that $E[b(t_1)(b(t_2) - b(t_1))] = 0$. On the other hand, we have that $E[b(t_1)(b(t_2) - b(t_1))] = E[b(t_1)b(t_2)] - E[b(t_1)^2] = E[b(t_1)b(t_2)] - t$. The last equality follows from (4.15). By comparing these two results it follows that $E[b(t_1)b(t_2)] = t$. The proof for the case when $t_1 > t_2$ is as good as identical. ■

Let us then show that white Gaussian noise is wide-sense stationary, and find its covariance.

Theorem 4.3.2 The ACF of the white noise process defined by (4.15) is given by $R(\tau) = \delta(\tau)$.

Proof. Let us first keep Δ fixed and define

$$\tilde{n}(t) = \frac{b(t+\Delta) - b(t)}{\Delta}. \tag{4.26}$$

The ACF of $\tilde{n}$ is then given by

$$\begin{aligned} E\left[\tilde{n}(t)\tilde{n}(t+\tau)\right] &= E\left[\frac{b(t+\Delta) - b(t)}{\Delta} \cdot \frac{b(t+\tau+\Delta) - b(t+\tau)}{\Delta} \right] \\ &= \frac{1}{\Delta^2} E\Big[b(t+\Delta)b(t+\tau+\Delta) - b(t)b(t+\tau+\Delta) \\ &\quad - b(t+\Delta)b(t+\tau) + b(t)b(t+\tau) \Big] \\ &= \frac{1}{\Delta^2} \Big(\min(t+\Delta, t+\Delta+\tau) - \min(t, t+\Delta+\tau) \\ &\quad - \min(t+\Delta, t+\tau) + \min(t, t+\tau) \Big) \end{aligned} \tag{4.27}$$

To find exact expressions for $E\left[\tilde{n}(t)\tilde{n}(t+d)\right]$ we have to distinguish between four cases:

1) $0 < \Delta < \tau$
2) $0 < \tau < \Delta$
3) $-\Delta < \tau < 0$
4) $\tau < -\Delta < 0$.

For the first case we find that

$$E\left[\tilde{n}(t)\tilde{n}(t+d)\right] = \frac{1}{\Delta}\Big(t + \Delta - t - (t+\Delta) + t \Big) = 0. \tag{4.28}$$

A similar argument yields 0 also for the fourth case. The middle cases require a bit more work. In the second case, the ACF is given by

$$E\left[\tilde{n}(t)\tilde{n}(t+\tau)\right] = \frac{1}{\Delta^2}\Big(t + \Delta - t - (t+\tau) + t \Big) = \frac{\Delta - \tau}{\Delta^2}. \tag{4.29}$$

In the third case, the ACF is given by

$$E\left[\tilde{n}(t)\tilde{n}(t+\tau)\right] = \frac{1}{\Delta^2}\left(t+\Delta+\tau-(t+\Delta+\tau)-(t+\tau)+t+\tau\right) = \frac{\Delta+\tau}{\Delta^2}. \tag{4.30}$$

We can sum this up as

$$E\left[\tilde{n}(t)\tilde{n}(t+\tau)\right] = \begin{cases} 0 & \text{if } 0<\Delta<\tau \\ \frac{1}{\Delta}\left(1-\frac{\tau}{\Delta}\right) & \text{if } 0<\tau<\Delta \\ \frac{1}{\Delta}\left(1+\frac{\tau}{\Delta}\right) & \text{if } -\Delta<\tau<0 \\ 0 & \text{if } \tau<-\Delta<0. \end{cases} \tag{4.31}$$

This is a "triangle pulse" which is entirely given by τ (and not by t). Thus, we have actually shown that white noise is stationary. To show that this triangle pulse indeed converges to the δ-function in the limit $\Delta \to 0$, we must verify that the integral of $E\left[\tilde{n}(t)\tilde{n}(t+\tau)\right]$ over τ is equal to one, no matter what Δ is. This is indeed the case:

$$\int_{-\Delta}^{0} \frac{1}{\Delta}\left(1+\frac{\tau}{\Delta}\right)\mathrm{d}\tau + \int_{0}^{\Delta} \frac{1}{\Delta}\left(1-\frac{\tau}{\Delta}\right)\mathrm{d}\tau = \frac{1}{2}+\frac{1}{2} = 1. \tag{4.32}$$

■

We can of course scale the strength of white noise: A white noise process with ACF $q\delta(\tau)$ is equal to a standard white noise process multiplied with $\sqrt{q}$. We refer to q as the variance of the white noise process, and $\sqrt{q}$ as the corresponding standard deviation.

4.3.2 Linear systems and the power spectral density

When a stochastic process is used as input to a dynamic system, the output will be another stochastic process. Recall from conventional (i.e., deterministic) state space theory that the response of a linear dynamical system can be modeled both in the time-domain and in the transform-domain. In the time domain the response to an arbitrary input signal is given by a convolution with the impulse response. In the transform domain the response is given by multiplication with the transfer function. The same rules applies when the input is a stochastic prosess. However, in this case we are mainly interested in how this affects the ACF or its Fourier transform, the power spectral density (PSD).

The convolution formulas for the ACF are given in the following theorem [85]. Notice that the theorem suggests evaluating the cross-correlation between the input and the output first, and then the autocorrelation of the output.

Theorem 4.3.3 Let the stochastic process $\mathbf{x}(t)$ with ACF $R_{xx}(t_1,t_2)$ be input to a linear system with impulse response $h(t)$ such that $\mathbf{y}(t) = \int_{-\infty}^{\infty} h(t-\tau)\mathbf{x}(\tau)\mathrm{d}\tau$. Assume that both $\mathbf{x}(t)$ and $h(t)$ are scalar and real-valued. The ACF $R_{yy}(t_1,t_2)$ of $\mathbf{y}(t)$ is then given by

$$R_{xy}(t_1,t_2) = \int_{-\infty}^{\infty} R_{xx}(t_1,t_2-\alpha)h(\alpha)\mathrm{d}\alpha \tag{4.33}$$

$$R_{yy}(t_1,t_2) = \int_{-\infty}^{\infty} R_{xy}(t_1-\alpha,t_2)h(\alpha)\mathrm{d}\alpha. \tag{4.34}$$

Proof. We only prove (4.33). From commutativity of the convolution operation and linearity of the integral it follows that

$$\mathbf{x}(t_1)\mathbf{y}(t_2) = \int \mathbf{x}(t_1)h(\alpha)\mathbf{x}(t_2-\alpha)\mathrm{d}\alpha.$$

Taking the expectation yields

$$E[\mathbf{x}(t_1)\mathbf{y}(t_2)] = \int E[\mathbf{x}(t_1)h(\alpha)\mathbf{x}(t_2-\alpha)]\mathrm{d}\alpha = \int h(\alpha)R_{xx}(t_1,t_2-\alpha)\mathrm{d}\alpha.$$

Proving (4.34) follows along very similar steps. ■

■ **Example 4.5 — The Gauss-Markov process.** As an example of this theorem we can look at the case when white noise $v(t)$ with ACF $R_{vv}(\tau) = q\delta(\tau)$ is used as input to a linear system with impulse response $h(t) = e^{-ct}u(t)$ where $u(t)$ is the Heaviside function. Since no real system has infinite memory, we assume that the white noise is switched on at $t = 0$. Therefore, we define $\mathbf{x}(t) = v(t)u(t)$ as the actual input to the system. The ACF of $\mathbf{x}(t)$ is given by

$$R_{xx}(t_1,t_2) = \begin{cases} 0 & \text{if } t_1 < 0 \text{ or } t_2 < 0 \\ R_{vv}(t_2-t_1) & \text{otherwise.} \end{cases} = R_{vv}(t_2-t_1)u(t_1)u(t_2) \tag{4.35}$$

Notice that $v(t)$ is a stationary stochastic process while $\mathbf{x}(t)$ is a non-stationary stochastic process. Defining $\mathbf{y}(t)$ as the convolution of $h(t)$ and $\mathbf{x}(t)$, our next step is to evaluate the ACF $R_{xy}(t_1,t_2)$. It is given by

$$\begin{aligned} R_{xy}(t_1,t_2) &= \int_{-\infty}^{\infty} R_{xx}(t_1,t_2-\alpha)h(\alpha)\mathrm{d}\alpha \\ &= \int_{-\infty}^{\infty} R_{vv}(t_2-\alpha-t_1)u(t_1)u(t_2-\alpha)h(\alpha)\mathrm{d}\alpha \\ &= \int_{-\infty}^{\infty} q\delta(t_2-t_1-\alpha)u(t_1)u(t_2-\alpha)e^{-c\alpha}u(\alpha)\mathrm{d}\alpha \\ &= qe^{-c(t_2-t_1)}u(t_1)u(t_2-t_1) \end{aligned} \tag{4.36}$$

The second equality in (4.36) follows from inserting (4.35) in the formula for R_{xy}, and accordingly replace t_2 with $t_2-\alpha$. In the third equality we have inserted concrete expressions for R_{vv} and h. Notice that a third Heaviside $u(\alpha)$ enters the picture because it is part of the impulse response. To arrive at the fourth equality we use the sifting property of the delta function and notice that the middle Heaviside then becomes redundant. Notice that $R_{xy}(t_1,t_2)$ according to (4.36) is zero whenever whenever $t_1 < 0$, and also whenever $t_2 < t_1$. This reflects the fact that $\mathbf{y}(t)$ only contains contributions from $\mathbf{x}(t)$ at previous time steps. In the remainder of the example, we therefore assume that $0 < t_1 < t_2$. We proceed to insert R_{xy} into the ACF for the filter output.

$$\begin{aligned} R_{yy}(t_1,t_2) &= \int_{-\infty}^{\infty} qe^{-c\alpha}e^{-c(t_2-t_1+\alpha)}u(\alpha)u(t_1-\alpha)u(t_2-t_1+\alpha)\mathrm{d}\alpha \\ &= qe^{-c(t_2-t_1)}\int_{-\infty}^{\infty} e^{-2c\alpha}u(\alpha)u(t_1-\alpha)u(t_2-t_1+\alpha)\mathrm{d}\alpha. \end{aligned} \tag{4.37}$$

The triple product of Heaviside functions need careful treatment. The first Heaviside yields a lower integration limit of zero. The second Heaviside can be rewritten as $u(t_1-\alpha) = 1-u(\alpha-t_1)$. Before dealing with the third Heaviside we recall the assumption that $t_1 < t_2$. This Heaviside can then be rewritten as $u(t_2-t_1+\alpha) = u(\alpha-(t_1-t_2))$ where $t_1-t_2 < 0$. Thus, this Heaviside becomes irrelevant since its changepoint is lower than for the first Heaviside. This leaves us with

$$R_{yy}(t_1,t_2) = qe^{-c(t_2-t_1)}\int_0^{t_1} e^{-2c\alpha}\mathrm{d}\alpha = \frac{q}{2c}(1-e^{-ct_1})e^{-c(t_2-t_1)} \tag{4.38}$$

which is our final expression for the ACF of the Gauss-Markov process. Recall that this expression is valid if $0 < t_1 < t_2$. Otherwise, the ACF is zero. It is interesting to notice that the term e^{-ct_1} goes towards zero as t_1 increases. In the limit as $t_1 \to \infty$ the Gauss-Markov process indistinguishable from a stationary process with ACF $\frac{q}{2c}e^{-c(t_2-t_1)}$. ■

Convolutions are in general cumbersome to evaluate. Furthermore, the expressions in Theorem 4.3.3 are only applicable to scalar systems. As in deterministic system theory, we can overcome these challenges by working in the transform domain. The main role is then played by the PSD, which is defined as the Fourier transform of the ACF:

$$S(\omega) = \int_{-\infty}^{\infty} R(\tau) e^{-\mathrm{i}\omega\tau} \mathrm{d}\tau \quad \Leftrightarrow \quad R(\tau) = \frac{1}{2\pi} \int_{-\infty}^{\infty} S(\omega) e^{\mathrm{i}\omega\tau} \mathrm{d}\omega. \tag{4.39}$$

The name power spectral density describes the fact that the PSD tells us how the energy of a signal is distributed among different frequencies (this is known as the Wiener-Khinchin theorem). Notice that $S(\omega)$ is only defined for wide-sense stationary processes, because the definition takes for granted that the ACF can be written as

$$R(t_1, t_2) = R(t_2 - t_1) = R(\tau) \text{ where } \tau = t_2 - t_1. \tag{4.40}$$

As in deterministic linear systems theory, it is also useful to define the transfer function, which is the Fourier transform of the impulse response:

$$H(\omega) = \int_{-\infty}^{\infty} h(\tau) e^{-\mathrm{i}\omega\tau} \mathrm{d}\tau. \tag{4.41}$$

To derive formulas similar to Theorem 4.3.3 in the frequency domain, let us first rewrite the expressions for $R_{xy}(t_1, t_2)$ and $R_{yy}(t_1, t_2)$ in the wide-sense stationary case:

$$R_{xy}(\tau) = \int R_{xx}(\tau - \alpha) h(\alpha) \mathrm{d}\alpha \tag{4.42}$$

$$R_{yy}(\tau) = \int R_{xy}(\tau - \alpha) h(-\alpha) \mathrm{d}\alpha. \tag{4.43}$$

The verify these expression is left as a small exercise to the reader.

Theorem 4.3.4 Let the wide-sense stationary stochastic process $\mathbf{x}(t)$ be input to a linear system with impulse response $h(t)$ such that $\mathbf{y}(t) = \int_{-\infty}^{\infty} h(t-\tau)\mathbf{x}(\tau)\mathrm{d}\tau$. Assume that both $\mathbf{x}(t)$ and $h(t)$ are scalar and real-valued. Let $S_{xx}(\omega)$ be the PSD of $\mathbf{x}(t)$, and let $H(\omega)$ be the Fourier transform of $h(t)$. Then $\mathbf{y}(t)$ is also wide-sense stationary and its PSD is given by

$$S_{xy}(\omega) = H(\omega) S_{xx}(\omega) \quad \text{and} \quad S_{yy}(\omega) = H^*(\omega) S_{xy}(\omega). \tag{4.44}$$

Proof. This follows directly from taking the Fourier transform of (4.42) and (4.43) by means of the convolution theorem, and utilizing the fact that if $H(\omega)$ is the Fourier transform of $h(t)$, then $H^*(\omega)$ is the Fourier transform of $h(-t)$. ■

In other words, we can find the PSD according to

$$S_{yy}(\omega) = H(\omega) S_{xx}(\omega) H^*(\omega) = |H(\omega)|^2 S_{xx}(\omega). \tag{4.45}$$

■ **Example 4.6 — PSD of white noise.** We recall that the ACF of white noise is a δ-function $q\delta(\tau)$. The PSD is the Fourier transform of this, which is a constant

$$S_{ww}(\omega) = q. \tag{4.46}$$

The physical interpretation of this is that white noise has energy equally distributed among all frequencies, from zero to infinity. Clearly this is not possible in the real world, and it demonstrates that white noise is a mathematical abstraction that must be treated with some care. ■

■ **Example 4.7 — PSD of Gauss-Markov process.** We recall that the Gauss-Markov process was generated by passing white noise with strength q through a linear filter with impulse response $h(t) = e^{ct}u(t)$. We know from linear systems theory that the corresponding transfer function is

$$H(\omega) = \frac{1}{c + \mathrm{i}\omega} \;\Rightarrow\; |H(\omega)]^2 = \frac{1}{c^2 + \omega^2}. \tag{4.47}$$

To find the ACF, we should multiply this with q and take the inverse transform. To do this, we may consult any table of Fourier transform pairs, which hopefully contains the pair

$$e^{-c|\tau|} \;\leftrightarrow\; \frac{2c}{c^2 + \omega^2}. \tag{4.48}$$

Notice that the function $e^{-c|t|}$ is different from $h(t)$ because it also is nonzero for negative τ. Thus we arrive at the ACF

$$R_{yy}(\tau) = \frac{q}{2c} e^{-c|\tau|}. \tag{4.49}$$

This is identical to the ACF that we obtained by means of convolution in Example 4.5 when we omit the middle factor $(1 - e^{-ct_1})$. The reason for the discrepancy is that we in Example 4.5 replaced the stationary pure white noise input with a non-stationary version defined by $\mathbf{x}(t) = v(t)u(t)$ in order to maintain causality. As $t_1 \to \infty$ we see that the middle term tends towards unity, and the two ACFs become indistinguishable. This means that in the long run we can treat the Gauss-Markov process as a stationary process. This would not be possible for, e.g., the Wiener process. ■

From comparing Examples 4.5 and 4.7 we see that obtaining the ACF of a process that comes out of a linear filter is much easier in the frequency domain than in the time domain. The price to pay is that we must confine ourselves to stationary processes.

There are at least two other reasons why the frequency domain representation of stochastic processes can be useful. First, we may want to estimate the correlation properties of a given time series of noise samples. By working in the frequency domain we get access to many tools of system identification. Particular frequencies (from, e.g., a pattern of ocean waves) that characterize the data are much easier to spot in the empirical PSD than in the empirical ACF. Second, we may want to simulate noise with given correlation properties. In general it takes less computational resources to do this in the frequency domain than in the time domain. We do not delve any deeper into these topics here, as this is material that more naturally would belong in a course in digital signal processing.

Related to simulation is also modeling. If we know that our process noise has a certain PSD, then we can find a transfer function $H(\omega)$ such that (4.45) holds, and we can then construct a differential equation that would have $H(\omega)$ or some approximation of it as its transfer function. For example, for the Gauss-Markov process the obvious differential equation would be

$$\dot{y} = -cy + w, \;\; w \sim \mathcal{N}(0, q\delta(t - \tau)) \tag{4.50}$$

where the notation $\delta(t - \tau)$ signifies that w is a white noise process.

4.4 Continuous-time models

In typical filtering problems the process model (4.7) is naturally specified in continuous time using the framework of stochastic processes from the previous section. A continuous-time linear process model is of the form

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} + \mathbf{G}\mathbf{n} \tag{4.51}$$

where $\mathbf{x}$ is the state vector, $\mathbf{u}$ is the control input and $\mathbf{n}$ is the process noise. To specify such a model we need to specify the elements in $\mathbf{x}$, the matrices $\mathbf{A}$, $\mathbf{B}$ (often zero) and $\mathbf{G}$, and the stochastic properties of $\mathbf{n}$. The process noise is typically assumed to be white, as is the case in Example 4.8. If the noise indeed is correlated, we extend the state vector with the correlated noise process, and include a filter that would produce noise with the desired correlation properties as part of the state space model, see Example 4.9.

■ **Example 4.8 — The CV model.** In the tracking literature, the most common kinematic model is the 2-dimensional constant velocity (CV) model. In continuous time it is given by $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{G}\mathbf{n}$ where $\mathbf{x}$ is the state vector and $\mathbf{n}$ is the process noise. The matrices are given by

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{and} \quad \mathbf{G} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{4.52}$$

while the process noise is assumed white with diagonal covariance, i.e.,

$$\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \mathbf{D}\delta(t-\tau)) \quad \text{where} \quad \mathbf{D} = \begin{bmatrix} \sigma_a^2 & 0 \\ 0 & \sigma_a^2 \end{bmatrix}. \tag{4.53}$$

The number σ_a is a measure of how much acceleration the target is expected to undergo. Thus, knowledge about the likely acceleration can be used as a crude way of determining σ_a. Notice that this model essentially integrates white noise. The last two states in $\mathbf{x}$ are therefore independent Wiener processes. ■

■ **Example 4.9 — Accelerometer with slowly varying bias.** In this example, we look at inertial navigation of a vehicle that moves along a straight road. Let the state vector consist of the following states:

$$\mathbf{x} = \begin{bmatrix} \text{Position of the vehicle} \\ \text{Velocity of the vehicle} \\ \text{Bias of the accelerometer} \end{bmatrix}$$

The accelerometer provides measurements of acceleration. Since these measurements are the derivatives of velocity, they cannot be modeled as a linear combination of elements in the state vector. Instead, the accelerometer measurements are modeled as a control input. The corresponding noise on the accelerometer measurements must then enter the system as process noise, and not as *bona-fide* measurement noise. Matters are complicated by the fact that accelerometers tend to suffer from slowly varying biases. An obvious candidate for modeling this is the Gauss-Markov process from example 4.5. All of this leads to a continuous-time model of the form $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} + \mathbf{G}\mathbf{n}$ where

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & -c \end{bmatrix}, \ \mathbf{B} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \ \mathbf{G} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{4.54}$$

and where

$$\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \mathbf{D}\delta(t-\tau)) \text{ where } \mathbf{D} = \begin{bmatrix} \sigma_a^2 & 0 \\ 0 & \sigma_b^2 \end{bmatrix}. \tag{4.55}$$

Here $\mathbf{u}$ are the accelerometer readings extrapolated as a continuous-time signal and $\mathbf{n}$ is the plant noise. The model is governed by the constants c, σ_a and σ_b. If we define $T = \frac{1}{c}$ as the time constant of the bias process, and we know that the mean square value of the bias is b, then we find that

$$\sigma_b^2 = 2bc = \frac{2b}{T}. \tag{4.56}$$

The standard deviation σ_a quantifies the uncertainty that remains in the accelerometer readings when the bias is accounted for. In the above model, this uncertainty is modeled as white Gaussian noise. ∎

4.5 Discretization

Continuous-time models such as those from the last two examples are useful from a conceptual perspective because they are invariant to the discretization time interval. Nevertheless, discretization is required for practical implementation and state estimation by means of the Kalman filter as formulated in Section 4.2. More precisely, discretization is the task of deriving the Markov model $p(\mathbf{x}_k \,|\, \mathbf{x}_{k-1})$ from the continuous time model (4.51), where $\mathbf{x}_k = \mathbf{x}(t_k)$ and $\mathbf{x}_{k-1} = \mathbf{x}(t_{k-1})$. For notational simplicity, less us also denote the discretization interval by

$$T = t_k - t_{k-1}. \tag{4.57}$$

From basic control theory, it is evident that the solution of (4.51) can be written as

$$\mathbf{x}_k = \mathbf{F}\mathbf{x}_{k-1} + \mathbf{u}_k + \mathbf{v}_k \tag{4.58}$$

where

$$\mathbf{F} = e^{\mathbf{A}(t_k - t_{k-1})}, \;\; \mathbf{u}_k = \int_{t_{k-1}}^{t_k} e^{\mathbf{A}(t_k-\tau)}\mathbf{B}\mathbf{u}(\tau)\,\mathrm{d}\tau \;\; \text{and} \;\; \mathbf{v}_k = \int_{t_{k-1}}^{t_k} e^{\mathbf{A}(t_k-\tau)}\mathbf{G}\mathbf{n}(\tau)\,\mathrm{d}\tau. \tag{4.59}$$

Evaluation of $\mathbf{F}$ and $\mathbf{u}_k$ is straightforward. All the randomness, conditional on $\mathbf{x}_{k-1}$, is encapsulated in $\mathbf{v}_k$. To complete the specification of $p(\mathbf{x}_k \,|\, \mathbf{x}_{k-1})$ we therefore have to quantify the statistical properties of $\mathbf{v}_k$. It is fairly obvious that the discrete-time plant noise inherits whiteness and Gaussianity from its continuous-time counterpart. All we need to know about $\mathbf{v}_k$ is then encapsulated in its instantaneous covariance matrix $\mathbf{Q}$.

Theorem 4.5.1 — Discretization of LTI stochastic systems. Let the continuous-time model be given by (4.51) where $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \mathbf{D}\delta(t-\tau))$. Also, assume that the discretization time is fixed at $T = t_k - t_{k-1}$. The covariance matrix of $\mathbf{v}_k$ in the discrete-time model (4.58) is then given by

$$\mathbf{Q} = E[\mathbf{v}_k \mathbf{v}_k^\mathsf{T}] = \int_0^T e^{(T-\tau)\mathbf{A}}\mathbf{G}\mathbf{D}\mathbf{G}^\mathsf{T} e^{(T-\tau)\mathbf{A}^\mathsf{T}}\,\mathrm{d}\tau \tag{4.60}$$

Proof. We prove the result in the slightly more general context of establishing the correlation between the discrete-time plant noise contributions at two different time steps. Following along the lines of page 189 in [4], we have that

$$\begin{aligned}
E[\mathbf{v}_k \mathbf{v}_l^\mathsf{T}] &= E\left[\left(\int_{t_{k-1}}^{t_k} e^{\mathbf{A}(t_k-\tau_1)}\mathbf{G}\mathbf{n}(\tau_1)\,\mathrm{d}\tau_1\right)\left(\int_{t_{l-1}}^{t_l} e^{\mathbf{A}(t_l-\tau_2)}\mathbf{G}\mathbf{n}(\tau_2)\,\mathrm{d}\tau_2\right)^\mathsf{T}\right] \\
&= E\left[\int_{t_{k-1}}^{t_k}\int_{t_{l-1}}^{t_l} e^{\mathbf{A}(t_k-\tau_1)}\mathbf{G}\mathbf{n}(\tau_1)\mathbf{n}(\tau_2)^\mathsf{T}\mathbf{G}^\mathsf{T} e^{\mathbf{A}^\mathsf{T}(t_l-\tau_2)}\mathrm{d}\tau_1\mathrm{d}\tau_2\right] \\
&= \int_{t_{k-1}}^{t_k}\int_{t_{l-1}}^{t_l} e^{\mathbf{A}(t_k-\tau_1)}\mathbf{G}E\left[\mathbf{n}(\tau_1)\mathbf{n}(\tau_2)^\mathsf{T}\right]\mathbf{G}^\mathsf{T} e^{\mathbf{A}^\mathsf{T}(t_l-\tau_2)}\mathrm{d}\tau_1\mathrm{d}\tau_2 \\
&= \int_{t_{k-1}}^{t_k}\int_{t_{l-1}}^{t_l} e^{\mathbf{A}(t_k-\tau_1)}\mathbf{G}\mathbf{D}\delta(\tau_1-\tau_2)\mathbf{G}^\mathsf{T} e^{\mathbf{A}^\mathsf{T}(t_l-\tau_2)}\mathrm{d}\tau_1\mathrm{d}\tau_2 \\
&= \int_{t_{k-1}}^{t_k} e^{(t_k-\tau)\mathbf{A}}\mathbf{G}\mathbf{D}\mathbf{G}^\mathsf{T} e^{(t_k-\tau)\mathbf{A}^\mathsf{T}}\mathrm{d}\tau\,\delta_{kl}.
\end{aligned} \tag{4.61}$$

Shifting the integration limits and recognizing the Kronecker delta leads to formula in the theorem. Notice that the third line of (4.61) is valid even if $\mathbf{n}$ is colored, while whiteness is used to make the transition to the fourth line. ■

It is not immediately obvious how one should evaluate the integral in Theorem 4.5.1. In a first-order approximation we could assume that $e^{(t_k-\tau)\mathbf{A}} \approx \mathbf{I}$, which would lead to

$$\mathbf{Q} \approx \mathbf{G}\mathbf{D}\mathbf{G}^{\mathsf{T}} T. \tag{4.62}$$

This is in general not recommended. The matrix product $\mathbf{G}\mathbf{G}^{\mathsf{T}}$ is not necessarily positive definite, even if the true $\mathbf{Q}$ in (4.61) is positive definite. However, a closed-form solution is also possible, and is generally to be preferred over approximations whenever one is in possession efficient techniques for evaluating the matrix exponential:

Theorem 4.5.2 — Van Loan's formula. Let us define the matrices $\mathbf{V}_1$ and $\mathbf{V}_2$ according to

$$\exp\left(\begin{bmatrix} -\mathbf{A} & \mathbf{G}\mathbf{D}\mathbf{G}^{\mathsf{T}} \\ \mathbf{0} & \mathbf{A}^{\mathsf{T}} \end{bmatrix} T\right) = \begin{bmatrix} \times & \mathbf{V}_2 \\ \mathbf{0} & \mathbf{V}_1 \end{bmatrix}. \tag{4.63}$$

Then we can find $\mathbf{Q}$ from Theorem 4.5.1 according to $\mathbf{Q} = \mathbf{V}_1^{\mathsf{T}}\mathbf{V}_2$.

Proof. A proof of the theorem can be found in Section II of Van Loan's article [108]. The proof is nontrivial, and does not provide much insights into genuine sensor fusion issues, and is therefore not repeated here. ■

The presence of a matrix exponential in this formula means that everything is not yet entirely resolved. Different programming languages have different levels of support for evaluating matrix exponentials. In some cases, the direct evaluation of a matrix exponential may be to slow to be of real-time utility. Sometimes it is possible to evaluate (4.63) exactly in terms of elementary functions. For example, for the discrete-time CV model all terms of order 4 and above in the series expansion s of the matrix exponential in (4.63) become zero, so that it can be truncated to order 3. For this model, the transition matrix and discrete-time plant noise covariance become

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{Q} = \begin{bmatrix} \frac{T^3}{3} & 0 & \frac{T^2}{2} & 0 \\ 0 & \frac{T^3}{3} & 0 & \frac{T^2}{2} \\ \frac{T^2}{2} & 0 & T & 0 \\ 0 & \frac{T^2}{2} & 0 & T \end{bmatrix} \sigma_a^2. \tag{4.64}$$

The derivation of these matrices is given to the reader as Exercise 4.2.

4.6 Tuning of the Q matrix

Having established the fundamental procedure for converting a continuous-time plant model into a discrete-time process model, we also have to provide sensible values for process noise parameters such as σ_a^2 in (4.64). This is made tricky by the fact that white noise ultimately is a dubious description of reality. For example, the acceleration of a moving object will be more of an on-off kind of thing than a pure white noise process. However, any such acceleration pattern is a possible realization from the ensemble of a white noise process. Let us once again keep in mind that white noise with variance q really is a limit of Wiener process increments as the time step goes towards zero. Thinking in terms of such increments, it is clear that a realization containing many increments larger than the standard deviation will have lower probability than a realization where

the increments are similar to or smaller than the standard deviation. From this mechanism we could expect that a large value of q would be more plausible to explain whatever increments there are.

On the other hand, assuming a smaller value of q, which is tantamount to assuming that the increments on average are smaller, would concentrate more probability mass at realizations that involve small increments. Thus, a balance results, between the plausibility of a large q to explain increments observed, and the plausibility of a small q in the absence of increments. A rigorous approach to determining this equilibrium can be found in maximum likelihood estimation, which we will study in Section 4.8.

In practical applications such as target tracking it is, however, of limited value to have a theoretically optimal estimate of the plant noise strength according to such considerations, because the white noise construction after all is so artificial. The actual process noise will typically vary a lot with time. While methods exist to deal with this variability, see Chapter 6, it is desirable at least from the perspective of simplicity to have a fixed plant noise covariance. Setting the plant noise covariance low will yield better accuracy when no significant maneuvers find place. Setting it high will yield inferior accuracy in this situation, but may be necessary to achieve acceptable performance during stronger maneuvers. A simple guideline that can be used is therefore to set the plant noise as high as required to account for the strongest maneuvers that are likely to happen.

■ **Example 4.10 — The maneuverability of a human diver.** The following example is taken from [17], where an active sonar was used to track a human diver with re-breather system. the diver is capable of going from resting to swimming in about one second, and is able to sustain a speed of approximately $0.5\,\mathrm{m/s}$. We see this happening in Figure 4.3. A CV model was used to track the diver.

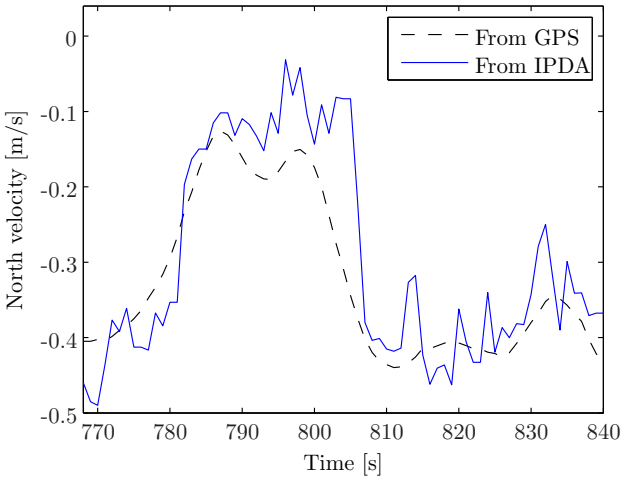


Figure 4.3: Target velocity for the diver scenario over a limited time interval.

Based on these considerations, $\sigma_a = 0.5\,\mathrm{m/s^2}$ appears to be reasonable value for the process noise, which was again used in the tracker that generated the results in Figure 4.3. The tracking method is described in greater detail in Section 7.6. ■

4.6.1 Filter consistency

While simple physical considerations as in Example 4.10 are always mandatory and often sufficient, a more systematic way of tuning the $\mathbf{Q}$ matrix is by means of *filter consistency*. We say that a filter is consistent if its errors on average are well described by the output of the filter. This leads to the following requirements:

1. The state errors should be acceptable as zero mean.

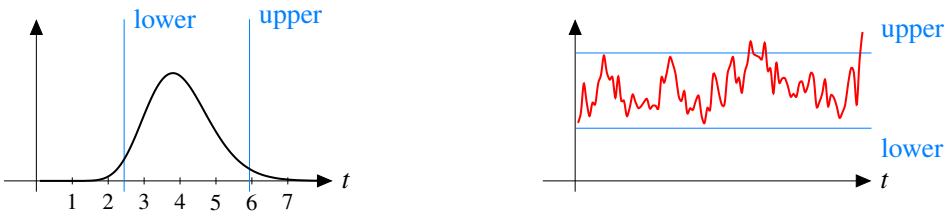


Figure 4.4: On the left: χ^2 distributions with confidence intervals. On the right: The average of ε_k over 10 Monte-Carlo simulations compared with the corresponding χ^2-boundaries.

2. The state errors should have magnitude commensurate with the state covariance yielded by the filter.
3. The innovations should be acceptable as zero mean.
4. The innovations should have magnitude commensurate with the innovation covariance yielded by the filter.
5. The innovations should be acceptable as white.

The typical test for consistency is to check whether criteria 2 or 4 holds. We refer the reader to [4] pp. 235-236 for bias and whiteness tests. The standard tools for testing criteria 2 and 4 are the normalized estimation error squared (NEES)

$$\varepsilon_k = (\hat{\mathbf{x}}_k - \mathbf{x}_k)^{\mathsf{T}} \mathbf{P}_k^{-1} (\hat{\mathbf{x}}_k - \mathbf{x}_k) \tag{4.65}$$

and the normalized innovations squared (NIS)

$$\varepsilon_k^{\nu} = \boldsymbol{\nu}_k^{\mathsf{T}} \mathbf{S}_k^{-1} \boldsymbol{\nu}_k. \tag{4.66}$$

In the remainder of this section, suppose that $\mathbf{x} \in \mathbb{R}^d$ and that $\mathbf{z} \in \mathbb{R}^s$. If the filter model indeed is correct, then ε_k and ε_k^{ν} will be χ^2 distributed random variables with d and s, respectively, as their degrees of freedoms (cf., Exercise 3.4). Furthermore, when we add together several ε_k or ε_k^{ν}, either from N different time steps or from N different realizations, then we get a new χ^2 distributed random variable with Nd or Ns degrees of freedom (cf., Exercise 2.2). The *average* NEES or NIS (known as ANEES and ANIS) should therefore obey a scaled χ^2 distribution, or more precisely a Gamma distribution with scale parameter $2/N$ and shape parameter $Nd/2$ or $Ns/2$. For any such distribution we can construct a confidence interval between the α quantile and the $1-\alpha$ quantile for a sensible value of α. Say we are looking for a 95% confidence interval for the ANEES. Then its lower and upper bound can be found according to

$$\texttt{lower} = \texttt{chi2inv}(0.025, Nd)/N \quad \text{and} \quad \texttt{upper} = \texttt{chi2inv}(0.975, Nd)/N \tag{4.67}$$

using the Matlab function for inverse χ^2 cdf. Assuming that everything else, including the measurement noise covariance, is correct, we can expect too large values in **Q** to push the ANEES below `lower`, while too small values in **Q** to push the ANEES above `upper`. Both situations are undesirable as they will lead to suboptimal estimation, but the latter is the most serious situation. If the ANEES tends to be too high it means that $\mathbf{P}_k$ tends to be too low, meaning that the Kalman filter is overconfident. Both in target tracking (with its measurement origin uncertainty) and in inertial navigation (with its nonlinearities) this is a common cause of divergence. In any case, a consistency analysis should always be carried out when designing a Kalman filter, as it also can be a useful debugging technique.

■ **Example 4.11 — Tuning of the CV model in the Autosea tracker.** In [102] this procedure was used to determine the plant noise strength for a CV model used in a maritime radar tracking system.

In this work, a Kalman filter based on CV model was used for measurements from the automatic identification system (AIS). The AIS measurements included both position and velocity, leading to the measurement matrix $\mathbf{H} = \mathbf{I}_4$. To compensate for quantization effects[1], the measurement noise covariance was set to

$$\mathbf{R} = \begin{bmatrix} 0.5^2 & 0 & 0 & 0 \\ 0 & 0.5^2 & 0 & 0 \\ 0 & 0 & 0.1^2 & 0 \\ 0 & 0 & 0 & 0.1^2 \end{bmatrix} + \frac{1}{12} \begin{bmatrix} v_x^2 & 0 & 0 & 0 \\ 0 & v_y^2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{4.68}$$

Kalman filters were then used to track the following ships with their AIS data as measurements:

Name	Type	Length × Breadth	Mean SOG (m/s)
GLUTRA	Passenger	94.8 m × 16.0 m	5.3
SULA	Cargo	87.9 m × 12.8 m	5.7
KORSFJORD	Passenger	122 m × 16.7 m	5.6
TRONDHEIMSFJORD II	High speed	24.5 m × 8 m	12.8
TELEMETRON	Pleasure craft	8 m × 3 m	13.1

Figure 4.5: Ships used to determine the plant noise in the Autosea tracker.

The Kalman filters were given both $\sigma_a = 0.05$ and $\sigma_a = 0.5$ as plant noise strength. Based on the tracking results, average NIS values and corresponding normalized χ^2 intervals (r_1, r_2) were calculated for the two values of σ_a:

Name	$\sigma_a = 0.05$		$\sigma_a = 0.5$		(r_1, r_2)	N
	NIS	AI	NIS	AI		
GLUTRA	**4.67**	−0.02	0.90	0.01	(3.47, 4.55)	109
SULA	**3.61**	−0.22	0.51	−0.10	(3.49, 4.56)	106
KORSFJORD	71.8	−1.33	**4.31**	−0.44	(3.52, 4.51)	127
TR.FJORD II	11.3	−0.62	**3.24**	−0.16	(3.76, 4.24)	533
TELEMETRON	371	−0.04	**4.45**	−0.01	(3.77, 4.23)	579

Figure 4.6: Consistency results

For the slower ships GLUTRA and SULA, the low value of $\sigma_a = 0.05$ appears most adequate. However, this value led to unacceptable NIS-values for the ships KORSFJORD, TRONDHEIMSFJORD II and TELEMETRON. When $\sigma_a = 0.5$ is used, the average NIS resides inside its 95% confidence interval for KORSFJORD, while it is a little bit too low for TRONDHEIMSFJORD II and a little bit too high for TELEMETRON. This value of σ_a was therefore chosen. The price to be paid for this is suboptimal tracking performance for slower ships such as GLUTRA and SULA: Their tracks will fluctuate more than they would have done if the lower value of σ_a had been chosen. ∎

A final remark is in order with regard to filter consistency. Any kind of model mismatches, not only a bad $\mathbf{Q}$ matrix, can lead to bad consistency properties. For nonlinear filters, the underlying approximation techniques will also have an impact on consistency. This concept is therefore very important in most of the subsequent chapters.

[1] AIS messages are given time stamps measured in an integer number of seconds, see [102] for details.

4.7 Tuning of the R matrix

The measurement noise is generally more tangible than the process noise. Any sensor has finite resolution, which yields a lower bound for the elements in the measurement noise matrix $\mathbf{R}$. The resolution is typically quantized (e.g., resolution cells for a radar), so that the measurements in principle will be discrete-valued random variables. To maintain the assumption of Gaussian measurement noise, this discrete measurement model must be approximated by a Gaussian. This is typically done by moment matching.

■ **Example 4.12** Consider a scenario with point targets and a 2-dimensional sensor that covers the surveillance region with square cells of fixed resolution Δx. Assume that the state vector $\mathbf{x}$ contains positions and velocities according to $\mathbf{x} = [x, y, v_x, v_y]^\mathsf{T}$ With the measurement matrix $\mathbf{H} = [\mathbf{I}_2, \mathbf{0}]$ we find that the distribution of $\mathbf{z} - \mathbf{H}\mathbf{x}$ conditional on $\mathbf{x}$ is uniform according to

$$p(\mathbf{z} - \mathbf{H}\mathbf{x}|\mathbf{x}) = \begin{cases} 1/\Delta x^2 & \text{if } \|\mathbf{z} - \mathbf{H}\mathbf{x}\|_\infty < \Delta x/2 \\ 0 & \text{otherwise.} \end{cases} \tag{4.69}$$

This distribution can be approximated by a Gaussian with the same covariance, with is

$$\mathbf{R} = \begin{bmatrix} \frac{\Delta x^2}{12} & 0 \\ 0 & \frac{\Delta x^2}{12} \end{bmatrix}. \tag{4.70}$$

With this approximation, (4.69) can be replaced by

$$p(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \mathbf{H}\mathbf{x}, \mathbf{R}) \tag{4.71}$$

where $\mathbf{z}$ is to be understood as the centroid of the corresponding resolution cell. ■

In reality, a sensor's resolution cells do not normally constitute squares or rectangles in Cartesian space. For example, the resolution cells of a 2-dimensional surveillance radar represent constant range and bearing. In other words, the resolution cells are rectangles in polar coordinates. This makes the filtering problem nonlinear. One possible solution is to use nonlinear filtering techniques such as the extended Kalman filter, which we will discuss in the next chapter. An alternative is to convert the measurements from polar coordinates to Cartesian coordinates, so that the standard Kalman filter still can be used. Several conversion methods have been proposed [3] [64] [70]. These methods perform both transformation of the measurement noise covariance, and also *bias compensation* due to the fact that an ellipse in polar coordinates becomes a banana in Cartesian coordinates. If the sensor resolution is sufficiently fine, there is no real need to conduct bias compensation, and we can transform the covariance matrix by pre- and post-multiplication by the Jacobian of polar-to-Cartesian mapping.

■ **Example 4.13** Assume that the measurement model is of the form $\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k) + \mathbf{w}_k$ where $\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$, and where

$$\mathbf{h}(\mathbf{x}_k) = \begin{bmatrix} \sqrt{x^2 + y^2} \\ \operatorname{atan2}(y, x) \end{bmatrix} \quad \text{and} \quad \mathbf{R} = \begin{bmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_\theta^2 \end{bmatrix}. \tag{4.72}$$

In other words, the measurements are given in polar coordinates, and we denote the elements in $\mathbf{z}_k$ correspondingly: $\mathbf{z}_k = [r, \theta]^\mathsf{T}$. Again, assume that the state vector is $\mathbf{x} = [x, y, v_x, v_y]^\mathsf{T}$. The converted measurement model, where the measurements are given in Cartesian coordinates without debiasing, is then of the form $\mathbf{y}_k = \mathbf{H}\mathbf{x}_k + \mathbf{w}_k$ where $\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_c)$, and where

$$\mathbf{y}_k = \begin{bmatrix} r\cos\theta \\ r\sin\theta \end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad \text{and} \quad \mathbf{R}_c = \mathbf{J}\mathbf{R}\mathbf{J}^\mathsf{T}, \quad \text{where} \quad \mathbf{J} = \frac{\partial}{\partial \mathbf{z}_k}\mathbf{h}^{-1}(\mathbf{z}_k). \tag{4.73}$$

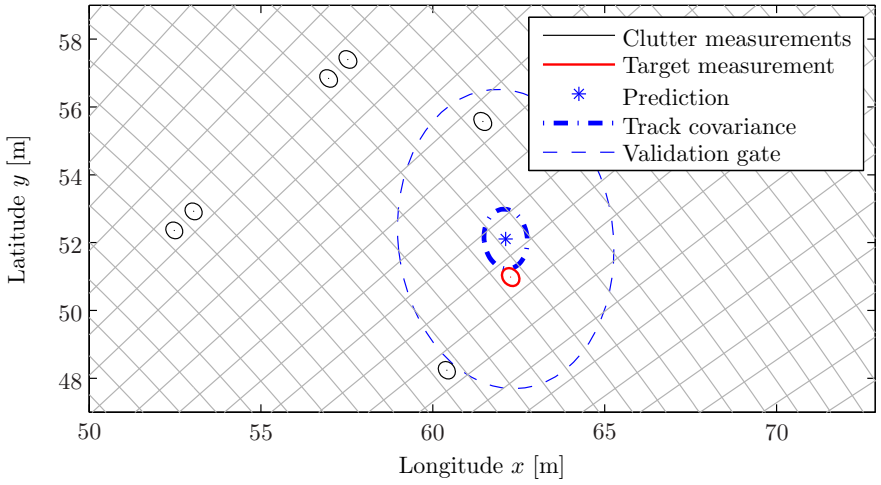


Figure 4.7: Illustration of Gaussian distributions matched to the discrete resolution cells in sonar tracking [15]. In this figure we also see a track prediction with corresponding uncertainty (blue stuff). We will study this in greater detail in Chapters 7 and 8.

By differentiating $\mathbf{h}^{-1}(\mathbf{z}_k)$, which simply is the mapping from $\mathbf{z}_k$ to $\mathbf{y}_k$, we find the the elements of $\mathbf{J}$ to be

$$\mathbf{J} = \begin{bmatrix} \cos\theta & -r\sin\theta \\ \sin\theta & r\cos\theta \end{bmatrix}. \tag{4.74}$$

A combined visualization of what is going on in Examples 4.12 and 4.13 is shown in Figure 4.7. ∎

To further complicate matters, sensor resolution may not be the only source of uncertainty that needs to be accounted for by $\mathbf{R}$. In target tracking and SLAM, a single target or landmark may cover several resolution cells. The standard approach then is to extract the centroids of local "blobs" as measurements by means of a clustering procedure. It is in general hard or impossible to provide analytical expressions for the resulting uncertainty, and one must resort to empirical investigations. This can be done means of real data similar to those the system is expected to handle, or simulations from a realistic sensor model that includes the entire measurement extraction machinery. For an example of the former, the reader is referred to [102]. For an example of the latter, the reader is referred to [17].

4.8 System identification and joint estimation of Q and R

Theorem 4.8.1 — Maximum likelihood estimation of system parameters. Consider the discrete-time state-space model (4.6), and assume that both $\mathbf{Q}$ and $\mathbf{R}$ depend on an unknown parameter vector $\boldsymbol{q}$. The maximum likelihood estimate of $\boldsymbol{q}$, if it exists, can then be found as

$$\boldsymbol{q}_{\mathrm{ML}} = \arg\max_{\boldsymbol{q}} \sum_k \log \mathcal{N}(\mathbf{z}_k; \mathbf{H}\hat{\mathbf{x}}_{k|k-1}, \mathbf{S}_k) \tag{4.75}$$

where $\hat{\mathbf{x}}_{k|k-1}$ and $\mathbf{S}_k$ are as given in Algorithm 1.

Proof. The likelihood of all the recieved measurements conditional on $\boldsymbol{q}$ is $p(\mathbf{z}_{1:k} \,|\, \boldsymbol{q})$. As a first step, we expand it as an integral over all the unknown states $\mathbf{x}_1, \ldots, \mathbf{x}_k$ in accordance with the total

probability theorem:

$$p(\mathbf{z}_{1:k} \,|\, \boldsymbol{q}) = \int p(\mathbf{z}_{1:k} \,|\, \mathbf{x}_{1:k}, \boldsymbol{q}) p(\mathbf{x}_{1:k}, \boldsymbol{q}) \,\mathrm{d}\mathbf{x}_{1:k} \tag{4.76}$$

By utilizing the Markov structure, we can write it in greater detail as

$$p(\mathbf{z}_{1:k} \,|\, \boldsymbol{q}) = \int \left(\prod_{l=1}^{k} p(\mathbf{z}_l \,|\, \mathbf{x}_l, \boldsymbol{q}) \right) \left(\prod_{l=2}^{k} p(\mathbf{x}_l \,|\, \mathbf{x}_{l-1}, \boldsymbol{q}) \right) p(\mathbf{x}_1) \,\mathrm{d}\mathbf{x}_{1:k}. \tag{4.77}$$

Furthermore, by invoking the Gaussian-linear assumptions of the model (4.6), it becomes

$$p(\mathbf{z}_{1:k} \,|\, \boldsymbol{q}) = \int \left(\prod_{l=1}^{k} \mathcal{N}(\mathbf{z}_l \,;\, \mathbf{H}\mathbf{x}_l, \mathbf{R}) \right) \left(\prod_{l=2}^{k} \mathcal{N}(\mathbf{x}_l \,;\, \mathbf{F}\mathbf{x}_{l-1}, \mathbf{Q}) \right) \mathcal{N}(\mathbf{x}_1 \,;\, \hat{\mathbf{x}}_{1|0}, \mathbf{P}_{1|0}) \,\mathrm{d}\mathbf{x}_{1:k}. \tag{4.78}$$

For simplicity, we have removed reference to the parameter vector $\boldsymbol{q}$. We can rewrite this expression as a sequence of nested integrals $I_1, I_2, \ldots$ so that

$$\begin{aligned}
I_1(\mathbf{x}_2) &= \int \mathcal{N}(\mathbf{x}_2 \,;\, \mathbf{F}\mathbf{x}_1, \mathbf{Q}) \mathcal{N}(\mathbf{z}_1 \,;\, \mathbf{H}\mathbf{x}_1, \mathbf{R}) \mathcal{N}(\mathbf{x}_1 \,;\, \hat{\mathbf{x}}_{1|0}, \mathbf{P}_{1|0}) \,\mathrm{d}\mathbf{x}_1, \\
I_2(\mathbf{x}_3) &= \int \mathcal{N}(\mathbf{x}_3 \,;\, \mathbf{F}\mathbf{x}_2, \mathbf{Q}) \mathcal{N}(\mathbf{z}_2 \,;\, \mathbf{H}\mathbf{x}_2, \mathbf{R}) I_1(\mathbf{x}_2) \,\mathrm{d}\mathbf{x}_2 \\
&\vdots \\
p(\mathbf{z}_{1:k} \,|\, \boldsymbol{q}) &= \int \mathcal{N}(\mathbf{z}_k \,;\, \mathbf{H}\mathbf{x}_k, \mathbf{R}) I_{k-1}(\mathbf{x}_k) \,\mathrm{d}\mathbf{x}_k.
\end{aligned} \tag{4.79}$$

By utilizing the product identity as used in Theorems 4.2.1 and 4.2.2 we can write the first integral as

$$I_1(\mathbf{x}_2) = \mathcal{N}(\mathbf{z}_1 \,;\, \mathbf{H}\hat{\mathbf{x}}_{1|0}, \mathbf{S}_1) \mathcal{N}(\mathbf{x}_2 \,;\, \mathbf{F}, \mathbf{F}\mathbf{P}_{1|0}\mathbf{F}^{\mathsf{T}} + \mathbf{Q}). \tag{4.80}$$

The second integral becomes

$$I_2(\mathbf{x}_3) = \mathcal{N}(\mathbf{z}_1 \,;\, \mathbf{H}\hat{\mathbf{x}}_{1|0}, \mathbf{S}_1) \mathcal{N}(\mathbf{z}_2 \,;\, \mathbf{H}\hat{\mathbf{x}}_{2|1}, \mathbf{S}_2) \mathcal{N}(\mathbf{x}_2 \,;\, \mathbf{F}, \mathbf{F}\mathbf{P}_{2|1}\mathbf{F}^{\mathsf{T}} + \mathbf{Q}). \tag{4.81}$$

and so on. We see that as we progress through the time indices we accumulate factors of the form $\mathcal{N}(\mathbf{z}_l \,;\, \mathbf{H}\hat{\mathbf{x}}_{l|l-1}, \mathbf{S}_l)$, and by taking the logarithm the product of these factors becomes the sum in the theorem. ■

▪ **Example 4.14** To study how maximum likelihood estimation of $\mathbf{Q}$ and $\mathbf{R}$ may work in practice, let us attempt to estimate the tuning parameters σ_z and σ_a in a CT model with straightforward Cartesian position measurements:

$$\dot{\mathbf{x}} = \begin{bmatrix} \mathbf{0}_{2\times 2} & \mathbf{I}_2 \\ \mathbf{0}_{2\times 2} & \mathbf{0}_{2\times 2} \end{bmatrix} \mathbf{x} + \begin{bmatrix} \mathbf{0}_{2\times 2} \\ \mathbf{I}_2 \end{bmatrix} \mathbf{n}, \;\; \mathbf{z}_k = \begin{bmatrix} \mathbf{I}_2 & \mathbf{0}_{2\times 2} \end{bmatrix} + \mathbf{w} \tag{4.82}$$

where

$$\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \sigma_a \mathbf{I}_2 \delta(t-\tau)) \;\; \text{and} \;\; \mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \sigma_z \mathbf{I}_2 \delta_{kl}). \tag{4.83}$$

We use the discretization interval $T = 1$ and 40 time steps, and $\sigma_z = 5$ and $\sigma_a = 0.2$ as ground truth parameters. We then plot the likelihood, i.e., the exponential of the sum in (4.75), as a function of possible values of σ_a and σ_z. For two different random seeds the results shown in Figure 4.8 were

obtained. We see that the estimates of σ_z are fairly sharp: In both cases it seems unlikely that σ_z should deviate more than, say, 20% from the maximum likelihood estimate. The estimates on σ_a are, on the other hand, much less reliable. In the simulation used in the left hand side of Figure 4.8, it seems clear that σ_a must be relatively close to its ground truth. But in the simulation used on the right hand side, much lower values of σ_a appear to be more plausible.

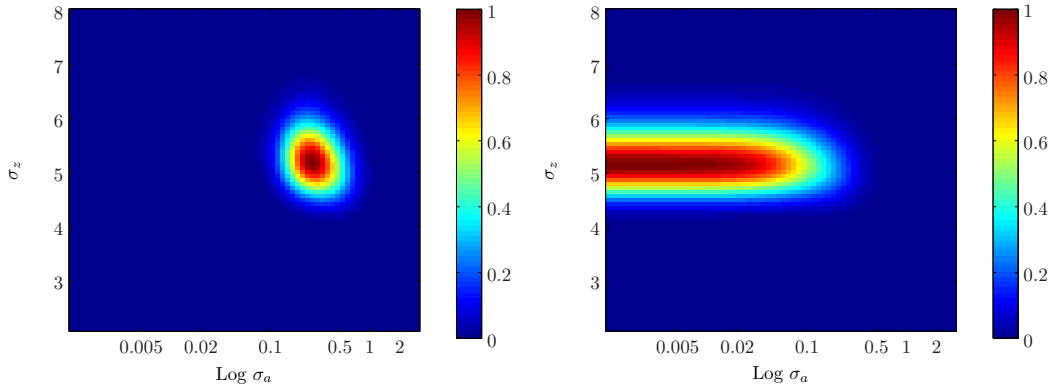


Figure 4.8: The likelihood function, normalized relative to its maximum, for two realizations of the CV model.

■

This example demonstrates that it may or may not be possible to obtain reliable estimates for the model parameters used in a Kalman filter by means of maximum likelihood. It is therefore generally preferable to use physical domain knowledge and common sense to set such values, as done in Sections 4.5 and 4.7. If there nevertheless is a need or desire to estimate these values from data, it may be a good idea to analyze whether the parameters actually can be estimated with a reasonable accuracy. A possible tool for this analysis is the Cramer Rao Lower Bound (CRLB) [88].

4.9 LTI and LTV systems

We have tacitly assumed that the matrices $\mathbf{A}$, $\mathbf{F}$, $\mathbf{Q}$, $\mathbf{R}$ and so on are constant in time. This means that we are looking at linear time-invariant (LTI) systems. Such systems are of course the easiest systems to analyze. In particular, most of the tools from frequency domain analysis require the systems to be of this form. However, the Kalman filter does not depend on this restriction. The Kalman filter remains an optimal estimator also for linear time-variant (LTV) systems where the matrices are allowed to change in time, insofar as they are known. In the discrete-time case we should then use a notation such as $\mathbf{A}_k$, $\mathbf{F}_k$, $\mathbf{Q}_k$, $\mathbf{R}_k$ and so on to emphasize the variability with time. Many sources (e.g., [93]) do indeed use such a notation when they introduce the Kalman filter in order to strive for utmost generality. In this book the philosophy is a bit different: We avoid "cluttered notation" as much as possible, and take for granted that the reader will be able to understand the most obvious generalizations.

In contrast to the LTI and the LTV cases, now consider the case where there is uncertainty in the system matrices. Also consider the case where the system matrices depend on the state vector. In these cases the system is neither LTI nor LTV, but nonlinear. Then the Kalman filter is no longer optimal, and possible not even usable at all, and we may want to consider nonlinear filtering techniques such as those described in the next chapter.

4.10 References and chapter remarks

The treatment of the main topic of this chapter, the Kalman filter, is largely based on Chapter 5 in [4]. The product identity is not used to derive the Kalman filter in [4]. Instead, it is derived as a case of MMSE estimation for Gaussian random variables. While the mathematical details are very similar to our derivation of the product identity, there is a huge conceptual difference: Our derivation of the Kalman filter is purely based on probability, and does not rely on any concepts related to estimators and estimation. Another perspective on the Kalman filter can be found in [43], where the Kalman filter is derived from the criterion that it should minimize the trace of the covariance of the state estimate. Finally, we could also have arrived at the Kalman filter from the orthogonality principle that we used to derive the LMMSE estimator.

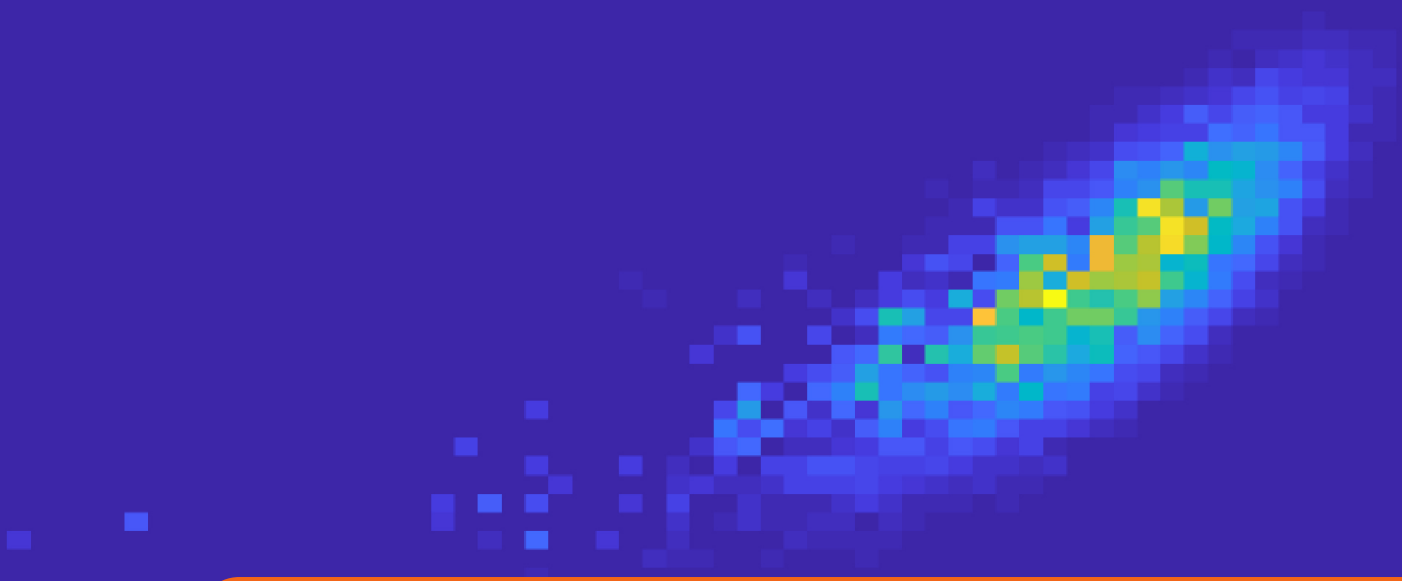
The treatment of stochastic processes is largely taken from [85]. For wide-sense stationary processes, one must decide on a sign convention for the cross-correlation $R_{xy}(\tau)$. While [85] uses the convention $\tau = t_1 - t_2$ we have used the convention $\tau = t_2 - t_1$, which also is used by Wikipedia.

The use of filter consistency as a tuning tool is recommended in [4]. If one instead wants to go for a maximum likelihood technique then [97] could be a reference worth checking out.

4.11 Exercises

Exercise 4.1 In inertial navigation, an accelerometer suffers from a slowly varying bias according to the Gauss-Markov model (4.54). What should the driving noise be to model a bias with average strength $1\,\mathrm{m/s^2}$, if the time constant is $T = 1\,\mathrm{s}$? ∎

Exercise 4.2 Derive the transition matrix $\mathbf{F}$ and discrete-time covariance $\mathbf{Q}$ for the CV model that are given in (4.64). ∎



5. Non-linear filtering

In the real world, estimation problems are seldom simple enough to fit the Gaussian-linear model studied in Chapter 4. Depending on how serious the deviations from Gaussianity and linearity are, a variety of filtering techniques are available. The purpose of this chapter is to review non-linear filtering techniques that are commonly used in target tracking.

First of all it should be mentioned that even if a problem technically is nonlinear, none of these techniques may be required. For example, if measurements are given in polar coordinates, a simple coordinate conversion such as the one that was suggested in Section 4.7 is most often preferable to nonlinear filtering techniques.

The main focus in this chapter is on two nonlinear filtering techniques: the Extended Kalman filter (EKF) and particle filters. The former is commonly used for problems whose nonlinearities are easy to linearize. The latter is a very general methodology that can be used for virtually any Bayesian filtering problem which involves a Markov model and a likelihood. Both have their own weaknesses, which may or may not lead to unacceptable performance degradation. Both of the two fundamental approaches, i.e., linearization and sampling, can be combined and expanded, leading to a wealth of different nonlinear estimation methods.

5.1 Linearization and the Extended Kalman Filter

It is often possible to attack nonlinearities by means of linearization, so that linear estimation techniques still can be used. Different options are possible for the choice of linearization technique. In some cases it can make sense to linearize around a fixed operating point. Such a fixed linearization point could be the initial estimate of a constant parameter, or the origin for a system that always remains so close to the origin that this linearization remains reasonably valid. In contrast, the operating point that the EKF linearizes around is the most recent estimate for the prediction step, and the most recent prediction for the update step.

Before we delve into the details of the EKF, let us also realize that more complex or sophisticated options for linearization exist as well. Having linearized once around the most recent prediction, and then performed an EKF update, we can once again linearize around the new estimate, and calculate the measurement update around this linearization point, and we can proceed in this manner

until we are sufficiently happy. Finally, we can also obtain a linear model by means of the LMMSE philosophy from Section 2.6.5. The we do not linearize around an operating point at all. Instead, we build the linear model by means of moment approximations, which typically are found from deterministic sample-based integration techniques.

The EKF is typically used for systems whose Markov model and measurement model can be written as follows:

$$\begin{aligned} \mathbf{x}_k &= \boldsymbol{f}(\mathbf{x}_{k-1}) + \mathbf{v}_k, & \mathbf{v}_k &\sim \mathcal{N}(0, \mathbf{Q}) \\ \mathbf{z}_k &= \mathbf{h}(\mathbf{x}_k) + \mathbf{w}_k, & \mathbf{w}_k &\sim \mathcal{N}(0, \mathbf{R}). \end{aligned} \tag{5.1}$$

Comparing with the model (4.9) from the previous chapter, we see that $\boldsymbol{f}$ and $\mathbf{h}$ are nonlinear functions, as opposed to the linear matrices $\mathbf{F}$ and $\mathbf{H}$ in (4.9). In order to construct linearized approximations of $\boldsymbol{f}$ and $\mathbf{h}$, we first introduce the error variables

$$\Delta\mathbf{x}_{k-1} = \mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1} \tag{5.2}$$

$$\Delta\mathbf{x}_k = \mathbf{x}_k - \hat{\mathbf{x}}_{k|k-1} \tag{5.3}$$

and then make Taylor expansions in terms of the error variables:

$$\boldsymbol{f}(\mathbf{x}_{k-1}) \approx \boldsymbol{f}(\hat{\mathbf{x}}_{k-1}) + \mathbf{F}(\hat{\mathbf{x}}_{k-1})\,\Delta\mathbf{x}_{k-1} \tag{5.4}$$

$$\mathbf{h}(\mathbf{x}_k) \approx \mathbf{h}(\hat{\mathbf{x}}_{k|k-1}) + \mathbf{H}(\hat{\mathbf{x}}_{k|k-1})\,\Delta\mathbf{x}_k. \tag{5.5}$$

The matrices $\mathbf{F}$ and $\mathbf{H}$ are now found as Jacobians of $\boldsymbol{f}$ and $\mathbf{h}$ with respect to $\mathbf{x}_{k-1}$ and $\mathbf{x}_k$:

$$\mathbf{F}(\hat{\mathbf{x}}_{k-1}) = \left.\frac{\partial}{\partial \mathbf{x}_{k-1}} \boldsymbol{f}(\mathbf{x}_{k-1})\right|_{\mathbf{x}_{k-1}=\hat{\mathbf{x}}_{k-1}} \tag{5.6}$$

$$\mathbf{H}(\hat{\mathbf{x}}_{k|k-1}) = \left.\frac{\partial}{\partial \mathbf{x}_k} \mathbf{h}(\mathbf{x}_k)\right|_{\mathbf{x}_k=\hat{\mathbf{x}}_{k|k-1}} \tag{5.7}$$

In the sequel we omit the dependencies of the Jacobians on $\hat{\mathbf{x}}_{k-1}$ and $\hat{\mathbf{x}}_{k|k-1}$ for notational simplicity. The prediced density can then be expressed as

$$\begin{aligned} p(\mathbf{x}_k|\mathbf{z}_{1:k-1}) &\approx \int \mathcal{N}(\mathbf{x}_k - \boldsymbol{f}(\hat{\mathbf{x}}_{k-1})\,;\, \mathbf{F}\Delta\mathbf{x}_{k-1}, \mathbf{Q})\mathcal{N}(\Delta\mathbf{x}_{k-1}\,;\, \mathbf{0}, \mathbf{P}_{k-1})\mathrm{d}\Delta\mathbf{x}_{k-1} \\ &= \mathcal{N}(\mathbf{x}_k\,;\, \boldsymbol{f}(\hat{\mathbf{x}}_{k-1}), \mathbf{F}\mathbf{Q}\mathbf{F}^\mathsf{T} + \mathbf{Q}) \int \mathcal{N}(\Delta\mathbf{x}_{k-1}\,;\, \ldots)\mathrm{d}\Delta\mathbf{x}_{k-1} \\ &= \mathcal{N}(\mathbf{x}_k\,;\, \boldsymbol{f}(\hat{\mathbf{x}}_{k-1}), \mathbf{F}\mathbf{Q}\mathbf{F}^\mathsf{T} + \mathbf{Q}) \end{aligned} \tag{5.8}$$

while the posterior density can be expressed as

$$\begin{aligned} p(\mathbf{x}_k|\mathbf{z}_{1:k}) &\propto \mathcal{N}(\mathbf{z}_k\,;\, \mathbf{h}(\mathbf{x}_k), \mathbf{R})\mathcal{N}(\mathbf{x}_k\,;\, \hat{\mathbf{x}}_{k|k-1}, \mathbf{P}_{k|k-1}) \\ &\approx \mathcal{N}(\mathbf{z}_k - \mathbf{h}(\hat{\mathbf{x}}_{k|k-1})\,;\, \mathbf{H}\Delta\mathbf{x}_k, \mathbf{R})\mathcal{N}(\Delta\mathbf{x}_k\,;\, \mathbf{0}, \mathbf{P}_{k|k-1}) \\ &= \mathcal{N}(\ldots)\mathcal{N}(\Delta\mathbf{x}_k\,;\, \mathbf{0} + \mathbf{W}(\mathbf{z}_k - \mathbf{h}(\hat{\mathbf{x}}_{k|k-1}) - \mathbf{H}\mathbf{0}), (\mathbf{I} - \mathbf{W}\mathbf{H}\mathbf{P}_{k|k-1}) \\ &\propto \mathcal{N}(\mathbf{x}_k\,;\, \hat{\mathbf{x}}_{k|k-1} + \mathbf{W}(\mathbf{z}_k - \mathbf{h}(\hat{\mathbf{x}}_{k|k-1})), (\mathbf{I} - \mathbf{W}\mathbf{H}\mathbf{P}_{k|k-1}) \end{aligned} \tag{5.9}$$

where the Kalman gain is given by the usual expression $\mathbf{W}_k = \mathbf{P}_{k|k-1}\mathbf{H}^\mathsf{T}\mathbf{S}_k^{-1}$. Again, the product identity has been used to obtain the desired results. In the same way as we did for the standard KF, we also express the EKF in algorithmic form below:

Algorithm 2 The extended Kalman filter

procedure EKF($\hat{\mathbf{x}}_{k-1}$, $\mathbf{P}_{k-1}$, $\mathbf{z}_k$)
  $\hat{\mathbf{x}}_{k|k-1} \leftarrow \boldsymbol{f}(\hat{\mathbf{x}}_{k-1})$ ▷ The predicted state estimate
  $\mathbf{F} \leftarrow \frac{\partial}{\partial \mathbf{x}_{k-1}} \boldsymbol{f}(\mathbf{x}_{k-1})$ ▷ The prediction Jacobian
  $\mathbf{P}_{k|k-1} \leftarrow \mathbf{F}\mathbf{P}_{k-1}\mathbf{F}^\mathsf{T} + \mathbf{Q}$ ▷ The predicted covariance
  $\hat{\mathbf{z}}_{k|k-1} \leftarrow \mathbf{h}(\hat{\mathbf{x}}_{k|k-1})$ ▷ The predicted measurement
  $\boldsymbol{\nu}_k \leftarrow \mathbf{z}_k - \hat{\mathbf{z}}_{k|k-1}$ ▷ The innovation
  $\mathbf{H} \leftarrow \frac{\partial}{\partial \mathbf{x}_k} \mathbf{h}(\mathbf{x}_k)$ ▷ The measurement Jacobian
  $\mathbf{S}_k \leftarrow \mathbf{H}\mathbf{P}_{k|k-1}\mathbf{H}^\mathsf{T} + \mathbf{R}$ ▷ The innovation covariance
  $\mathbf{W}_k \leftarrow \mathbf{P}_{k|k-1}\mathbf{H}^\mathsf{T}\mathbf{S}_k^{-1}$ ▷ The Kalman gain
  $\hat{\mathbf{x}}_k \leftarrow \hat{\mathbf{x}}_{k|k-1} + \mathbf{W}_k\boldsymbol{\nu}_k$ ▷ The posterior state estimate
  $\mathbf{P}_k \leftarrow (\mathbf{I} - \mathbf{W}_k\mathbf{H})\mathbf{P}_{k|k-1}$ ▷ The posterior covariance
  return $\hat{\mathbf{x}}_k$, $\mathbf{P}_k$, $\hat{\mathbf{x}}_{k|k-1}$, $\mathbf{P}_{k|k-1}$, $\mathbf{S}_k$
end procedure

■ **Example 5.1 — EKF for the CT model.** In this example we shall introduce the coordinated turn (CT) model, which in target tracking is the most important alternative to the CV model. The fundamental premise behind this model is that the target moves with a nearly constant turn-rate, which we denote ω. We use the right-hand convention that a positive turn rate results when the cross product of two consecutive velocity vectors has a positive z-component. The following relationships between velocities and accelerations can then be derived:

$$\ddot{x} = -\omega\dot{y} \quad \text{and} \quad \ddot{y} = \omega\dot{x}. \tag{5.10}$$

If we define a state vector as $\mathbf{y} = [x, y, \dot{x}, \dot{y}]^\mathsf{T}$ then (5.10) can be rewritten as a state space model of the form

$$\dot{\mathbf{y}} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -\omega \\ 0 & 0 & \omega & 0 \end{bmatrix} \mathbf{y}. \tag{5.11}$$

So far, this is a purely linear system. If we include ω in the state vector, then it becomes a nonlinear system. Also, (5.11) is a purely deterministic system. We make it into a a stochastic system by introducing plant noise processes for x- and y-accelerations with standard deviation σ_a, and also for the turn rate ω with standard deviation σ_ω. Furthermore, we assume that the turn rate evolves according to a Wiener process multiplied by σ_ω. With the state vector $\mathbf{x} = [x, y, \dot{x}, \dot{y}, \omega]^\mathsf{T}$, this leads to the nonlinear plant model

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -\omega & 0 \\ 0 & 0 & \omega & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{n} \tag{5.12}$$

where $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \mathbf{D})$ and $\mathbf{D} = \operatorname{diag}([\sigma_a^2, \sigma_a^2, \sigma_\omega^2])$. The model is nonlinear because the turn rate ω both appears in the state vector and in the system matrix. An exact discretization of (5.12) is therefore not possible. However, if we assume that ω is slowly varying, in other words that σ_ω is sufficiently small compared to the time step T, then it should be a fair approximation to decouple

the dynamics of x, y, $\dot{x}$ and $\dot{y}$ from the dynamics of ω, leading to two linear models. Stitching these together, and calculating matrix exponentials, we arrive at

$$\mathbf{x}_k = \begin{bmatrix} 1 & 0 & \frac{\sin T\omega_{k-1}}{\omega_{k-1}} & \frac{-1+\cos T\omega_{k-1}}{\omega_{k-1}} & 0 \\ 0 & 1 & \frac{1-\cos T\omega_{k-1}}{\omega_{k-1}} & \frac{\sin T\omega_{k-1}}{\omega_{k-1}} & 0 \\ 0 & 0 & \cos T\omega_{k-1} & -\sin T\omega_{k-1} & 0 \\ 0 & 0 & \sin T\omega_{k-1} & \cos T\omega_{k-1} & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{x}_{k-1} + \mathbf{v}_k \tag{5.13}$$

where the discrete-time plant noise $\mathbf{v}_k$ is zero-mean Gaussian white with covariance matrix

$$\mathbf{Q} = \begin{bmatrix} \frac{T^3}{3}\sigma_a^2 & 0 & \frac{T^2}{2}\sigma_a^2 & 0 & 0 \\ 0 & \frac{T^3}{3}\sigma_a^2 & 0 & \frac{T^2}{2}\sigma_a^2 & 0 \\ \frac{T^2}{2}\sigma_a^2 & 0 & T\sigma_a^2 & 0 & 0 \\ 0 & \frac{T^2}{2}\sigma_a^2 & 0 & T\sigma_a^2 & 0 \\ 0 & 0 & 0 & 0 & T\sigma_\omega^2 \end{bmatrix} \tag{5.14}$$

Again, (5.13) is a nonlinear model due to the presence of ω_{k-1} both in the transition matrix and in the previous state vector $\mathbf{x}_{k-1}$. An annoying artifact of (5.13) the fact that it is not defined for $\omega_{k-1} = 0$. Nevertheless, the limit value can be calculated using L'Hopital's rule, or we may simple take the matrix exponential of (5.12) with $\omega = 0$, and (5.13) then boils down to the CV model.

To implement an EKF for (5.13) we have to calculate the Jacobian for the mapping from $\mathbf{x}_{k-1}$ to $\mathbf{x}_k$. While somewhat tedious, the calculations are in principle straightforward[1], and lead to

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & \frac{1}{\omega}\sin T\omega & -\frac{1}{\omega}(1-\cos T\omega) & \mathbf{F}_{1,5} \\ 0 & 1 & \frac{1}{\omega}(1-\cos T\omega) & \frac{1}{\omega}\sin T\omega & \mathbf{F}_{2,5} \\ 0 & 0 & \cos T\omega & -\sin T\omega & \mathbf{F}_{3,5} \\ 0 & 0 & \sin T\omega & \cos T\omega & \mathbf{F}_{4,5} \\ 0 & 0 & 0 & 0 & \mathbf{F}_{5,5} \end{bmatrix} \tag{5.15}$$

where the last column in $\mathbf{F}$ is

$$\mathbf{F}_{\cdot,5} = \begin{bmatrix} \mathbf{F}_{1,5} \\ \mathbf{F}_{2,5} \\ \mathbf{F}_{3,5} \\ \mathbf{F}_{4,5} \\ \mathbf{F}_{5,5} \end{bmatrix} = \begin{bmatrix} \frac{1}{\omega^2}(\dot{y} - (T\omega\dot{y} + \dot{x})\sin T\omega + (T\omega\dot{x} - \dot{y})\cos T\omega) \\ \frac{1}{\omega^2}(-\dot{x} + (\omega T\dot{x} - \dot{y})\sin T\omega + (\omega T\dot{y} + \dot{x})\cos T\omega) \\ -T(\dot{x}\sin T\omega + \dot{y}\cos T\omega) \\ T(\dot{x}\cos T\omega - \dot{y}\sin T\omega) \\ 1 \end{bmatrix}. \tag{5.16}$$

For notational simplicity the time index $k-1$ has been dropped in (5.15) and (5.16). Again, special attention is required to deal with the limit $\omega \to 0$. After wrestling a bit with the maths, the following matrix results:

$$\lim_{\omega\to 0} \mathbf{F} = \begin{bmatrix} 1 & 0 & T & 0 & -\frac{T^2\dot{y}}{2} \\ 0 & 1 & 0 & T & \frac{T^2\dot{x}}{2} \\ 0 & 0 & 1 & 0 & -T\dot{y} \\ 0 & 0 & 0 & 1 & T\dot{x} \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.17}$$

[1] I recommend that you always verify Jacobian calculations both using a symbolic mathematics program such as Maple, and using numerical differentiation in, e.g., Matlab.

Table 5.1: Parameter values used in CT EKF example

Symbol	Description	Value
σ_a	Acceleration plant noise standard deviation	$0.02\ \mathrm{m/s^2}$
σ_ω	Turn rate plant noise standard deviation	$0.005\ /\ \mathrm{s}$
σ_z	Measurement noise standard deviation	5 m
T	Discretization time interval	0.5 s

Notice that the upper left 4×4 sub-matrix in (5.17) is the transition matrix from the CV model. We have now specified all the mathematics that is geniune to the CT model. To design an EKF for the CT model we must also specify a measurement model. We opt for standard position measurements, i.e.,

$$\mathbf{z}_k = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} \mathbf{x}_k + \mathbf{w}_k, \ \ \mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \sigma_z^2 \mathbf{I}_2). \tag{5.18}$$

We benchmark the CT model against the CV model using simulations. We investigate a simulated scenario with the model parameters given in able 5.1. The CV model uses the same values for σ_a, σ_z and T as the CT model. We draw the initial state vector according to $\mathcal{N}(\hat{\mathbf{x}}_1, \mathbf{P}_1)$ where $\hat{\mathbf{x}}_1 = [0, 0, 5, 0, 0.05]^\mathsf{T}$ and $\mathbf{P}_1 = \mathrm{diag}([25, 25, 0.25, 0.25, 0.0025])$. In Figure 5.1 we see the comparison of the CT model with the CV model in a single Monte-Carlo simulation. For fairness of comparison, the CT filter model is initialized with zero turn rate, i.e., with $\hat{\mathbf{x}}_1 = [0, 0, 5, 0, 0]^\mathsf{T}$. We see that the CT model nevertheless has a fairly good concept of the actual turn rate after only three measurements. The CV model is also able to follow the turn, but it lags with $5 - 10$ m, which may or may not be acceptable. Notice the different sizes of the covariance ellipses in Figure 5.1. The CV model gives a smaller covariance than the CT model despite being worse off. In other words, the CV model has poor consistency properties. Keep in mind that the CV model was implemented with the same acceleration process noise as the CT model. Increasing the *assumed* process noise for the CV model would improve its consistency.

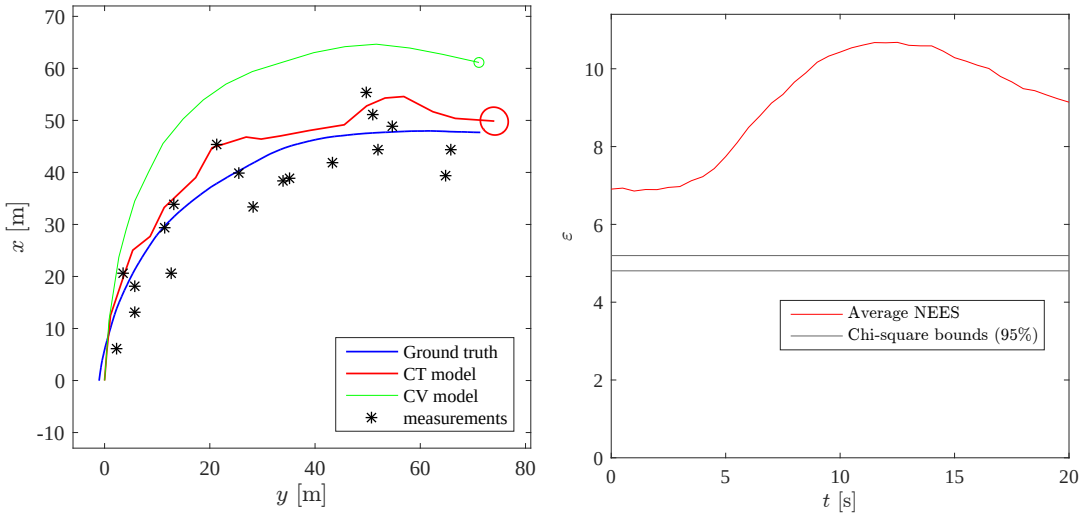


Figure 5.1: Filtering results for a single simulation of the CT scenario and consistency analysis from 1000 Monte-Carlo simulations.

Despite the observed improvements, caution is mandatory. By running 1000 Monte-Carlo

Table 5.2: Parameter values used in the Van der Pol example

Symbol	Description	Value
σ_a	Plant noise standard deviation	0.1
σ_z	Measurement noise standard deviation	0.1
T	Discretization time interval	0.01 s
T	Time interval between measurements	1 s

simulations, we can investigate whether the EKF actually attains ideal consistency properties. The answer, provided by the right-hand-side of Figure 5.1, is ultimately negative. The ANEES of the full 5-dimensional state vector of the CT-EKF is up to twice as large as it should be. While this is not very severe inconsistency, it could make a tracking system more susceptible to track loss. For other choices of the system parameters, the inconsistency problem could be better or worse. ∎

∎ **Example 5.2 — The Van der Pol oscillator.** The archtypical nonlinear system in deterministic system theory is the Van der Pol oscillator. It is similar to a standard linear mass-spring-damper system, that also has a negative damping that comes into play when the state vector approaches its equilibrium point at the origin. A stochastic version of this system, driven by white noise, can be written as $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{G}\mathbf{n}$ where

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} x_2 \\ \mu(1-x_1^2)x_2 - x_1 \end{bmatrix}, \ \mathbf{G} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \ \mathbf{n} \sim \mathcal{N}(0, \mathbf{D}\delta(t-\tau)) \tag{5.19}$$

where $\mathbf{D}$ is the (scalar) covariance of the continuous-time white noise process. Let us furthermore assume that only the second state is measured, so the measurement model becomes $\mathbf{z} = \mathbf{H}\mathbf{x} + \mathbf{w}$ where

$$\mathbf{H} = \begin{bmatrix} 0 & 1 \end{bmatrix}, \ \mathbf{w} \sim \mathcal{N}(0, R\delta_{kl}) \tag{5.20}$$

and where R is the (scalar) discrete-time measurement covariance. The first challenge that we encounter in this example is that we are given a *continuous-time* nonlinear plant model, while our formulation of the EKF in Algorithm 2 presumes a *discrete-time* nonlinear plant model. Unfortunately, discretization of arbitrary nonlinear systems do not lead to closed-form solutions. In the EKF we must then devise approximations both for prediction of the state estimate and its covariance. The former is done by any kind of numerical ODE solver, such as Euler's method or your favourite Runge-Kutta method. For the latter, one possibility is to proceed as closely as possible to how we would do this for a linear system, by evaluating the transition matrix for the linearized system and the propagating the covariance using the standard Kalman filter approach for this transition matrix. The system matrix and the plant noise matrix of the linearized model are

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -1-2\mu xy & \mu(1-x^2) \end{bmatrix}, \ \mathbf{G} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{5.21}$$

Over the time interval $T = t_k - t_{k-1}$ we find the corresponding transition matrix $\mathbf{F}$ and the discrete time plant noise covariance $\mathbf{Q}$ according to the standard discretization formulas in (4.59) and (4.61). Then the covariance prediction is given by the standard formula $\mathbf{P}_{k|k-1} = \mathbf{F}\mathbf{P}_{k-1}\mathbf{F}^\mathsf{T} + \mathbf{Q}$. This completes the specification of the EKF for the Van der Pol example. In addition some model parameters and simulation setup parameters must be decided to implement a simulation of this system. These are provided in Table 5.2. ∎

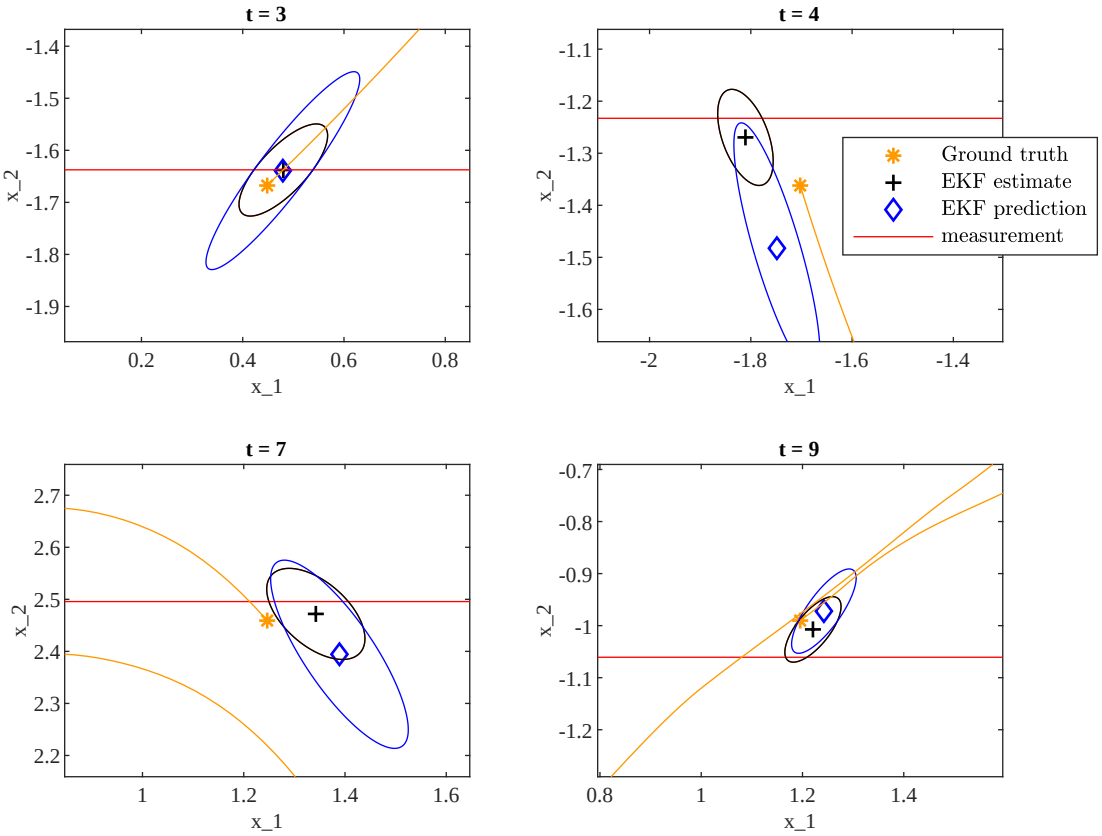


Figure 5.2: Snapshots of EKF outputs for the Van der Pol example.

The output from the EKF in this example is visualized in Figure 5.2 for 4 different time steps. We see that both the predicted and the updated covariance ellipse tend to be aligned with the trajectory of the state vector. This is related to the very strong predictive power inherent in this system: It is well known that the Van der Pol oscillator tends towards a limit cycle. This information is not used in the measurement update, but it does indirectly affect the development of the covariance during the prediction.

5.2 Particle Filters

A key challenge in nonlinear filtering is that, even if all the noise processes are Gaussian, the nonlinear transformations quickly turn the posterior density into something non-Gaussian. Exact solutions are then very unlikely to exist. If the posterior indeed does not resemble a Gaussian at all, then we have little reason to expect that an EKF will work well, and we may desire a more flexible technique that can represent arbitrary posteriors with finite computational resources.

Possible solutions to this problem of non-Gaussian filtering tend to belong to one of three categories. The first category consists of grid-based approaches. For problems in 1 or 2 dimensions it may be feasible to evaluate the posterior density in a brute force manner over a grid with a suitable resolution. For higher dimensions this quickly becomes computationally infeasible. The second category consist of approaches based on random sampling. Instead of placing the samples in a pre-defined grid, we may draw them from a probability distribution which tends to place them where the bulk of the posterior density is likely to be. The third category consists of methods that aim to represent the posterior as a sum of simpler functions, typically Gaussians. In this section

we will explore particle filters, which is the most popular approach from the second category of non-Gaussian filters.

The theory of particle filters is founded upon three main ideas. These can be summarized as *1)* Monte-Carlo integration and importance sampling, *2)* Sampling of trajectories and *3)* Resampling. Our development of the fundamental theory of particle filtering will be organized according to these ideas in the following three subsections, before we present the simplest possible particle filter in Section 5.2.4. More advanced particle filter methods follow in Sections 5.2.5 - 5.2.7.

5.2.1 Idea 1: Monte-Carlo integration and importance sampling

To begin, recall that interesting properties are often extracted from pdf's by means of integration. For example, the expectation of $\mathbf{x}$ under the pdf $\pi(\mathbf{x})$ is given by the integral $\int \mathbf{x}\pi(\mathbf{x})\,\mathrm{d}\mathbf{x}$. For an arbitrary pdf $\pi(\mathbf{x})$ we may easily encounter a situation where this integral has no closed form solution. Does that mean that we have no means of evaluating the expectation of $\mathbf{x}$? Not necessarily. If we are able to produce a large number of independent samples $\mathbf{x}^i$ from $\pi(\mathbf{x})$ then we can simply approximate the integral as the average of these samples.

This can be generalized to a situation where we want to evaluate an integral of the form

$$I = \int \mathbf{f}(\mathbf{x})\,\pi(\mathbf{x})\,\mathrm{d}\mathbf{x}. \tag{5.22}$$

Again, we draw samples $\mathbf{x}^i$ from $\pi(\mathbf{x})$, and approximate the integral as the average of $\mathbf{f}(\mathbf{x}^i)$. In more mathematical terms, we let $\{\mathbf{x}^i\}_{i=1}^N$ be N i.i.d. samples from $\pi(\cdot)$, and we define the corresponding sample mean of $\mathbf{f}(\mathbf{x})$ as

$$I_N = \frac{1}{N}\sum_{i=1}^{N}\mathbf{f}(\mathbf{x}^i). \tag{5.23}$$

Then we can rest assured that

$$I_N \longrightarrow I \quad \text{as} \quad N \longrightarrow \infty. \tag{5.24}$$

This technique for evaluating integrals is known as importance sampling, and $\pi(\mathbf{x})$ is called an importance density.

■ **Example 5.3 — Integral evalutated by importance sampling.** Consider the function displaye in Figure 5.3. This function is given by the expression

$$f(x) = 0.7\mathcal{N}(x;1,0.1) + 0.3\mathcal{N}(x;2.8,0.9). \tag{5.25}$$

Since it is a sum of two pdfs, weighted with numbers that sum to one, it is easy to verify that the integral $\int f(x)\mathrm{d}x$ indeed is one. But let us pretend that we are not in possession of this knowledge. We want to evaluate the integral by means of importance sampling. We consider four possible importance densities:

$$\begin{aligned}\pi_1(x) &= \mathcal{N}(x;1,0.1)\\ \pi_2(x) &= \mathcal{N}(x;2.8,0.9)\\ \pi_3(x) &= \mathcal{N}(x;1.54,0.34)\\ \pi_4(x) &= f(x).\end{aligned}$$

We recognize importance densities 1 and 2 as the Gaussians that make up the summands in (5.25). Neither of these are good representatives for the overall shape of $f(x)$. The third importance density is a Gaussian that is constructed in order to provide a better representation of the shape of $f(x)$ in

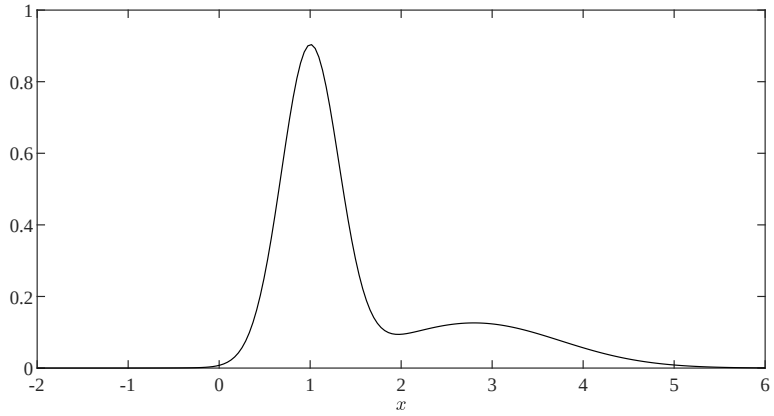


Figure 5.3: Function to be integrated by means of importance sampling.

Table 5.3: Results of importance sampling for the integral of (5.3)

Scheme	$N = 10$	$N = 100$	$N = 10^3$	$N = 10^4$	$N = 10^5$
π_1	0.7256	0.7432	0.7806	0.7710	0.8183
π_2	2.2569	0.9333	0.8900	1.0114	1.0091
π_3	1.0048	1.0164	0.9883	1.0100	0.9995
π_4	1.0000	1.0000	1.0000	1.0000	1.0000

the following sense: It has the same expectation and covariance as $f(x)$ would have if interpreted as a pdf. This is known as moment-matching. Finally, since $f(x)$ actually is a pdf we can also use $f(x)$ itself as an importance density. To use importance sampling, we calculate sample averages of the form

$$\frac{1}{N}\sum_{i=1}^{N}\frac{f(x^i)}{\pi_k(x^i)},\ k \in \{1,2,3,4\}$$

with samples from $\pi_1(\cdot),\ldots,\pi_4(\cdot)$, respectively. Let us then take a look at how the accuracy of the different integration schemes depend on the number N of samples. In Table 5.3 we see some results from single trials of the integration schemes for different values of of N. The first scheme does not perform very well. Even for $N = 10^5$ it is not capable of giving an accurate evaluation. Also notice that it *always* underestimates the integral. The reason for that is that $\pi_1(x)$ is narrower than $f(x)$. Consequently, it is difficult for it to place samples at the plateau between $x = 2$ and $x = 4$, and it fails to take this part of the integrand properly into account. The second scheme uses the widest of the two Gaussians involved in $f(x)$, and consequently it does manage to place samples everywhere where the integrand differs significantly from zero. We see a significant improvement in its performance as N increases. The third scheme appears to have very good performance already for low values of N. Because it is better tailored to the overal structure of $f(x)$ than the second scheme, it manages to place its samples more strategically. Finally, the fourth scheme is able to find the exact value of the integral no matter what N is, because once its proposal density is factored out of the integrand, all we are left with is the constant one.

There are two take-home lessons from this example. First, it is clearly desirable with a proposal density that is a good approximation of the integrand. Second, it is important that the proposal density places samples wherever the integrand is has significant mass, and therefore it is better with a proposal density that is a little bit too wide than a proposal density that is a little bit too narrow.

Keep in mind that these conclusions were made only based on a single Monte-Carlo simulation. To state anything with certainty a larger number (typically hundreds or thousands) of Monte-Carlo simulations may be needed. We save that for the next example. ∎

∎ **Example 5.4 — 2-dimensional integral evalutated by importance sampling.** In this example we want to evaluate the integral $\int_D \mathcal{N}(\mathbf{x}; \mathbf{a}, \mathbf{A})\mathrm{d}\mathbf{x}$ the region of integration is $D = (-\infty, \infty) \times [0, 1]$, and where

$$\mathbf{a} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \ \mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

This scenario is illustrated in Figure 5.4. It is motivated by the problem of calculating collision probabilities in a collision avoidance system [103]. Let us consider two possible importance densities:

$$\pi_1(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \mathbf{a}, \mathbf{A})$$
$$\pi_2(\mathbf{x}) = \mathcal{N}(x_1; 0, 1)\mathrm{Uniform}(x_2; [0, 1])$$

The accuracy of the two proposal densities is evaluated for different sample numbers over 1000

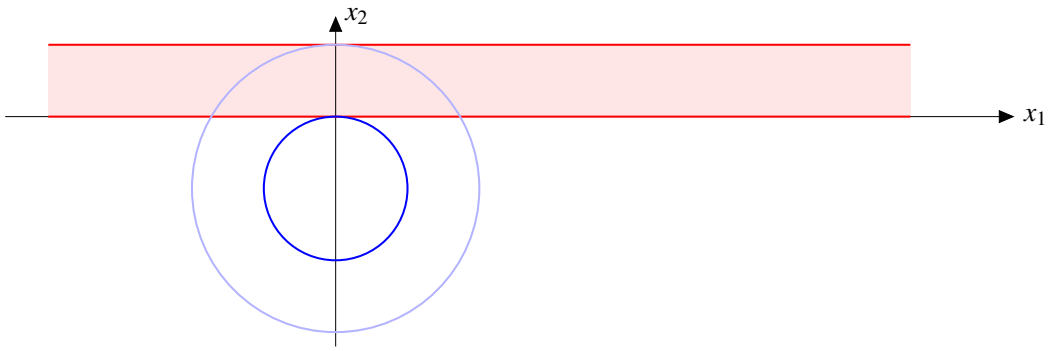


Figure 5.4: Illustration of the problem to be solved in Example 5.4. The integration domain is the red strip, while the integrand is the Gaussian whose first and second covariance ellipses are shown in blue.

independent Monte-Carlo simulations. In Figure 5.5 we see the relative RMSE of the two integration schemes, that is, RMSE divided by the exact value of the integral, which is 0.1359. Overall, the integration scheme using $\pi_1(\mathbf{x})$ has 10 times larger errors than the integration scheme using $\pi_2(\mathbf{x})$. To reach a relative accuracy of about 0.01 we need about 1000 samples with $\pi_2(\mathbf{x})$, while we need close to 10^5 samples with $\pi_1(\mathbf{x})$. The reason is that $\pi_1(\mathbf{x})$ is much wider than it should be in the x_2-direction in order to place its samples within the red strip. ∎

So far we have explored importance sampling for integrals where we have a closed-form integrand. The situation that we soon shall encounter in particle filtering is slightly more complex. In a particle filter we are again dealing with an integral of the form $I = \int \mathbf{f}(\mathbf{x})\, \pi(\mathbf{x})\, \mathrm{d}\mathbf{x}$, but in this case we are only able to express $\pi(\mathbf{x})$ indirectly. More to the point, we will not be able to sample from $\pi(\mathbf{x})$ itself. The solution is to introduce yet another proposal density, called $q(\mathbf{x})$, so that the integral also can be written as

$$I = \int \mathbf{f}(\mathbf{x}) \frac{\pi(\mathbf{x})}{q(\mathbf{x})} q(\mathbf{x})\, \mathrm{d}\mathbf{x}. \tag{5.26}$$

We introduce the so-called importance weights

$$\tilde{w}(\mathbf{x}^i) = \frac{\pi(\mathbf{x}^i)}{q(\mathbf{x}^i)} \tag{5.27}$$

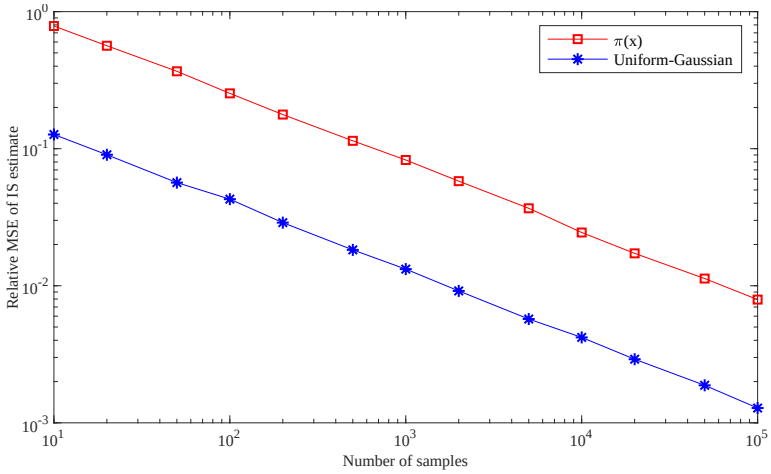


Figure 5.5: Accuracy of Monte-Carlo integration in Example 5.4.

and approximate the integral by the sample mean

$$I_N = \frac{1}{N}\sum_{i=1}^{N} \mathbf{f}(\mathbf{x}^i)\tilde{w}(\mathbf{x}^i). \tag{5.28}$$

Thanks to (5.26) we can again rest assured that $I_N \longrightarrow I$ as $N \longrightarrow \infty$. In particle filtering applications, the pdf $\pi(\mathbf{x})$ will typically only be known up to proportionality. We can then use the normalized importance weights and still get the correct result (see Exercise 5.2):

$$w(\mathbf{x}^i) = \frac{\tilde{w}(\mathbf{x}^i)}{\sum_{j=1}^{N} \tilde{w}(\mathbf{x}^j)}. \tag{5.29}$$

We saw in the examples that the choice of the importance density is critical to the efficiency of Monte-Carlo integration. It is desirable to choose a $q(\mathbf{x})$ that is a good approximation of $\pi(\mathbf{x})$. In general, our choice of $q(\mathbf{x})$ will typically be either narrower or wider than $\pi(\mathbf{x})$. The former is generally worse than the latter. A central concept here is *support*: The support of a function $f(\mathbf{x})$ is the set of all $\mathbf{x}$ such that $f(\mathbf{x}) > 0$. For importance sampling to work at all, the support of $q(\mathbf{x})$ must include the entire support of $\pi(\mathbf{x})$. This is equivalent to demanding that $\pi(\mathbf{x})/q(\mathbf{x})$ is upper bounded. Even if this is satisfied, importance sampling may still be horribly inefficient if there is a significant difference in where $\pi(\mathbf{x})$ has the bulk of its probability mass and where $q(\mathbf{x})$ has its probability mass.

5.2.2 Idea 2: Sampling of trajectories

To use importance sampling in recursive Bayesian estimation, we need mechanisms to represent the posterior $p(\mathbf{x}_k \,|\, \mathbf{z}_{1:k})$ in terms of samples from an importance density $q(\mathbf{x}_k \,|\, \mathbf{z}_{1:k})$. The challenge here is that we do not have any closed form expression for $p(\mathbf{x}_k \,|\, \mathbf{z}_{1:k})$ available to insert in the weight formula (5.27). We must therefore come up with a sampling scheme that allows us to only use known densities, i.e. $p(\mathbf{z}_k \,|\, \mathbf{x}_k)$ and $p(\mathbf{x}_k \,|\, \mathbf{x}_{k-1})$, in the weight evaluation.

The solution that a conventional particle filter uses to overcome this challenge is to sample not only the current state $\mathbf{x}_k$, but actually the entire trajectory $\mathbf{x}_{1:k}$. This works as follows. First N samples of the initial states $\mathbf{x}_1$ are drawn, and their weights are updated according to how well they explain the first measurement $\mathbf{z}_1$. Then one sample for each of these are drawn of the next state vector $\mathbf{x}_2$ and weights are calculated. As the particle filter iterates through the time steps, each sample thus becomes a possible trajectory of the system.

Delving into the mathematics, we notice that the full posterior of the trajectory $\mathbf{x}_{1:k}$ is given by

$$p(\mathbf{x}_{1:k}|\mathbf{z}_{1:k}) \propto p(\mathbf{z}_k|\mathbf{x}_k)\, p(\mathbf{x}_k|\mathbf{x}_{k-1})\, p(\mathbf{x}_{1:k-1}|\mathbf{z}_{1:k-1}). \tag{5.30}$$

We assume that the old posterior $p(\mathbf{x}_{1:k-1}\,|\,\mathbf{z}_{1:k-1})$ is given by a set of particles $\mathbf{x}^i_{1:k-1}$, which are drawn according to a proposal density $q(\mathbf{x}_{1:k-1}\,|\,\mathbf{z}_{1:k-1})$ and with weights

$$w^i_{k-1} \propto \frac{p(\mathbf{x}^i_{1:k-1}|\mathbf{z}_{1:k-1})}{q(\mathbf{x}^i_{1:k-1}|\mathbf{z}_{1:k-1})}. \tag{5.31}$$

The key step in the derivation is then to assume that the new proposal density $q(\mathbf{x}_{1:k}\,|\,\mathbf{z}_{1:k})$ is chosen in such a way that it factorizes as follows:

$$q(\mathbf{x}_{1:k}|\mathbf{z}_{1:k}) = q(\mathbf{x}_k|\mathbf{x}_{1:k-1},\mathbf{z}_{1:k-1})q(\mathbf{x}_{1:k-1}|\mathbf{z}_{1:k-1}) \tag{5.32}$$

If we combine the last three equations we arrive at the following expression for the current importance weights

$$\begin{aligned} w^i_k &\propto \frac{p(\mathbf{x}^i_{1:k}|\mathbf{z}_{1:k})}{q(\mathbf{x}^i_{1:k}|\mathbf{z}_{1:k})} \\ &\propto \frac{p(\mathbf{z}_k\,|\,\mathbf{x}^i_k)p(\mathbf{x}^i_k\,|\,\mathbf{x}^i_{k-1})p(\mathbf{x}^i_{1:k-1}\,|\,\mathbf{z}_{1:k-1})}{q(\mathbf{x}^i_k\,|\,\mathbf{x}_{1:k-1},\mathbf{z}_{1:k-1})q(\mathbf{x}^i_{1:k-1}|\mathbf{z}_{1:k-1})} \\ &\propto \frac{p(\mathbf{z}_k\,|\,\mathbf{x}^i_k)p(\mathbf{x}^i_k\,|\,\mathbf{x}^i_{k-1})}{q(\mathbf{x}^i_k\,|\,\mathbf{x}_{1:k-1},\mathbf{z}_{1:k-1})} w^i_{k-1} \end{aligned} \tag{5.33}$$

The main take-home message from (5.33) is that we have a *recursive* formula for the weight updates. Without that, we would have needed to evaluate the trajectory pdf's $p(\mathbf{x}_{1:k}|\mathbf{z}_{1:k})$ and $q(\mathbf{x}_{1:k}|\mathbf{z}_{1:k})$ every time a new measurement is being processed.

In the standard Bayesian filtering problem we rely on the Markov assumption $p(\mathbf{x}_k\,|\,\mathbf{x}_{1:k-1}) = p(\mathbf{x}_k\,|\,\mathbf{x}_{k-1})$. That is, we assume that earlier states $\mathbf{x}_1,\ldots,\mathbf{x}_{k-2}$ carry no information about $\mathbf{x}_k$ that is not carried by $\mathbf{x}_{k-1}$. Under this assumption there should not be much to gain from invoking earlier states or measurements in the proposal density, and it makes sense to restrict it to the form

$$q(\mathbf{x}_k|\mathbf{x}_{1:k-1},\mathbf{z}_{1:k}) = q(\mathbf{x}_k|\mathbf{x}_{k-1},\mathbf{z}_k). \tag{5.34}$$

With this proposal density, the weight update becomes

$$w^i_k \propto w^i_{k-1} \frac{p(\mathbf{z}_k|\mathbf{x}^i_k)\, p(\mathbf{x}^i_k|\mathbf{x}^i_{k-1})}{q(\mathbf{x}^i_k|\mathbf{x}^i_{k-1},\mathbf{z}_k)} \tag{5.35}$$

While further specification of the proposal density differs between the many variations of the particle filter, all particle filters known to the author is based on the general framework specified by (5.34) and (5.35).

While (5.34) and (5.35) really describe a methodology for importance sampling of trajectories, we are in filtering (as opposed to, e.g., smoothing) only interested in the marginal posterior distribution at the current time step. It is given by

$$p(\mathbf{x}_k\,|\,\mathbf{z}_{1:k}) = \int p(\mathbf{x}_{1:k}\,|\,\mathbf{z}_{1:k})\,\mathrm{d}\mathbf{x}_{1:k-1}. \tag{5.36}$$

If $p(\mathbf{x}_{1:k}\,|\,\mathbf{z}_{1:k})$ is represented by a collection of samples, then the marginalization boils down to simply forgetting the first $k-1$ state vectors in every sample. Since this is what we do in practice, it may on the surface seem like we only sample the most recent state vector in a particle filter. But the derivations of this section shows that this is an illusion.

5.2.3 Idea 3: Resampling

Since a particle filter really samples trajectories and not just state vectors, the dimension of its samples increases during every iteration. This is problematic, because as the sample space becomes larger and larger, it becomes more and more difficult for the proposal density to hit the right regions. A symptom of this problem is that the variance of the weights will increase until one particle has all the probability mass. A proof that this indeed will happen can be found in [61]. An intuitive understanding can be gained from recalling that the logarithm of any Gaussian is a quadratic form. Assume for this little thought exercise that the pdf that we want to sample from is approximately Gaussian. If the dimension of the state space increases, and it becomes more difficult to place samples near the peak of the Gaussian, then the samples will tend to end up further away where the quadratic form is steeper. It is illustrated in Figure 5.6 how the rightmost couple of samples x_3 and x_4 will get very different pdf-values, while the difference between the pdf-values for the samples near the peak (x_1 and x_2) is negligible.

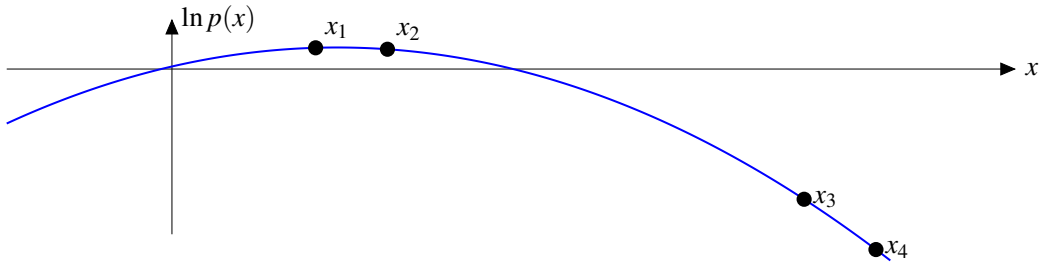


Figure 5.6: Illustration of how the ratio of pdf-values from a Gaussian increases for samples drawn further away from the expectation. A particle filter attempting to estimate this posterior will get more and more imbalanced weights the further away the particle cloud is from the maximum.

The problem that only one or a few of the particles take all the probability mass is known as *degeneracy*. It can be monitored by a quantity known as the effective sample size:

$$\hat{N}_{\text{eff}} = \frac{1}{\sum_{i=1}^{N} (w_k^i)^2}. \tag{5.37}$$

We are generally happy if $\hat{N}_{\text{eff}} \approx N$. On the other hand, if $\hat{N}_{\text{eff}} \to 1$ then we have degeneracy. It is therefore advisable to do something if $\hat{N}_{\text{eff}}$ goes below some pre-determined threshold.

The solution to this problem is to *resample* the particles. We generate a new particle cloud, where each new particle is drawn from the collection of old particles, with probabilities equal to the old particle weights. This way, we get several copies of the particles with high weights, while particles with lower weights get fewer or no copies. All the new particles get uniform weights, and the degeneracy problem is thus solved.

The most straightforward approach to resampling is to use *cdf inversion* as described in Section 2.2.4. How this works for a particle filter is illustrated in Figure 5.7. First, we construct the cdf of the old particle cloud. This will be a staircase function with jumps for every integer j between 1 and N. For each new particle (indexed by i) we draw a random number u^i uniformly distributed between 0 and 1. Then we determine which step covers u^i. The corresponding particle then becomes the new particle number i. After repeating this procedure N times, a new particle cloud has been generated.

A naive implementation of resampling by cdf inversion would work in $O(N^2)$ time. For each new particle, we find the corresponding old particle. It is possible to reduce the complexity to $O(N \log N)$ by sorting the random numbers u^i first. Further reduction to $O(N)$ complexity is possible by utilizing algorithms based on order-statistics.

A more convenient approach to resampling is *systematic resampling*. In this approach, only a single random number is drawn, and then new particle copies are generated by moving along

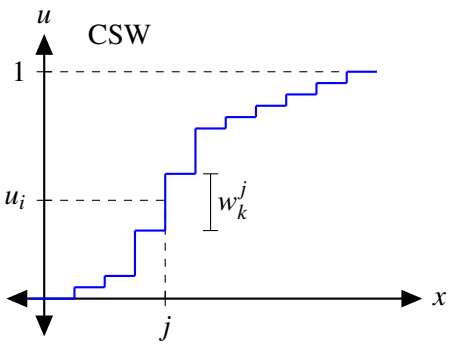


Figure 5.7: Resampling by CDF inversion.

the cumulative sum of weights, without any sorting needed. A pseudo-code description of the algorithm is given in Algorithm 3. The scrambling in Line 13 of the pseudocode is not strictly needed, but recommended to ensure that everything remains as random as possible.

Algorithm 3 Systematic resampling

```
1:  procedure ROULETTESYSTEMATIC(weights, N)
2:      cumweights ← cumsum(weights)
3:      indicesout ← zeros(N, 1)
4:      noise ← rand(1, 1)/N
5:      i ← 1
6:      for j ← 1 : N do
7:          u_j ← (j − 1)/N
8:          while u_j > cumweights[i] do
9:              i ← i + 1
10:         end while
11:         indicesout[j] ← i
12:     end for
13:     indicesout ← indicesout(randperm(size(indicesout, 1)))
14:     return indicesOut
15: end procedure
```

5.2.4 The SIR filter and practical implementation

We now have all the building blocks in place to design our first particle filter. The main question when designing a particle filter is always: What should our proposal density $q(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{z}_k)$ be? The simplest answer to that question is that it can be given by the process model:

$$q(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i, \mathbf{z}_k) = p(\mathbf{x}_k^i \,|\, \mathbf{x}_{k-1}^i). \tag{5.38}$$

For this proposal density, the weight update formula becomes

$$w_k^i =\propto w_{k-1}^i \frac{p(\mathbf{z}_k|\mathbf{x}_k^i)\, p(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i)}{p(\mathbf{x}_k \,|\, \mathbf{x}_{k-1})} = w_{k-1}^i p(\mathbf{z}_k|\mathbf{x}_k^i) \tag{5.39}$$

The particle filter that results is known as the Sequential Importance-sampling-Resampling filter, abbreviated as the SIR filter. Before we delve into our first example of a particle filter, an important remark is in order regarding how the weight update (5.39) is implemented in practice. The likelihood

$p(\mathbf{z}_k|\mathbf{x}_k^i)$ can vary tremendously in value, and is very likely to encounter situations where all the likelihood values become indistinguishable from zero due to the limited numerical precision of any computer. The solution to this problem is to work with the logarithms of $p(\mathbf{z}_k|\mathbf{x}_k^i)$ and w_k^i. The product in (5.39) then becomes a sum. This may, however, not be sufficient. To carry out the resampling step, we have to calculate the normalized weights, and before we can do this we have to exponentiate the logarithmic weights, which again carries a danger of numerical underflow. To ensure that at least the largest weights give meaningful numbers after this exponentiation, we can rescale the logarithmic weights before the exponentiation by adding a constant to all the logarithmic weights so that the largest logarithmic weight becomes zero. The overall workflow of the SIR filter is summarized by the block diagram in Figure 5.8.

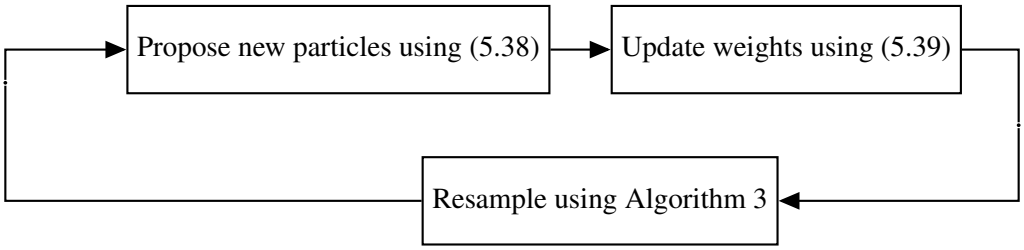


Figure 5.8: Workflow of the SIR filter. Each estimation cycle consists of three steps.

■ **Example 5.5 — SIR filter for the CV model.** Before we consider any nonlinear applications, it is a good idea to study how well a particle filter is able to deal with a straightforward linear scenario. Therefore we implement the SIR filter for the standard CV model. We use the same values for σ_z and T as were used for the CV model in Table 5.1. We investigate the two values for the process noise strength σ_a that were considered in Example 4.11, that is $\sigma_a = 0.5$ and $\sigma_a = 0.05$. We generate the simulated trajectories from initial state and covariance given by

$$\mathbf{x}_0 = \begin{bmatrix} 0 \\ 0 \\ 5 \\ 0 \end{bmatrix}, \quad \mathbf{P}_0 = \begin{bmatrix} 25 & 0 & 0 & 0 \\ 0 & 25 & 0 & 0 \\ 0 & 0 & 25 & 0 \\ 0 & 0 & 0 & 25 \end{bmatrix}.$$

The simulation models and the filter models are identical. The only remaining tuning constant is the number of particles, so let us study how this choice affects the performance of the SIR filter, in addition to the process noise covariance.

In Figure 5.9 we compare the performance of the SIR filter and the Kalman filter in the high noise scenario $\sigma_a = 0.5$ when 10, 100, 1000, 10000 and 100000 particles are used. We see that all the particle filters with more than 100 particles have performance on par with the Kalman filter. However, when only 10 particles are used, the performance is not acceptable.

One would expect that performance should increase when the process noise is lowered to 0.05. However, Figure 5.10 reveals a very different picture. In this case, we need more than 10000 particles to get performance comparable to the Kalman filter. With 1000 particles the RMSE is several times larger for the particle filter than for the Kalman filter. Further reductions in the value of σ_a only makes the performance even worse.

If we dive deeper into these bewildering results, it is revealed that the SIR filter in the low-noise scenario often diverges because none of its particles are good enough. Once divergence happens, it has no mechanism to restore its performance. In Figure 5.11 we see a case study of how this happens for a Monte-Carlo simulation with 1000 particles. After 10 seconds (20 time steps) the particle cloud has collapsed into a collection of clusters, of which none really manages to capture the true posterior given by the Kalman filter. After 20 seconds only one of these clusters remain, and it drifts slowly and indefinitely away from the true state for future time steps. ■

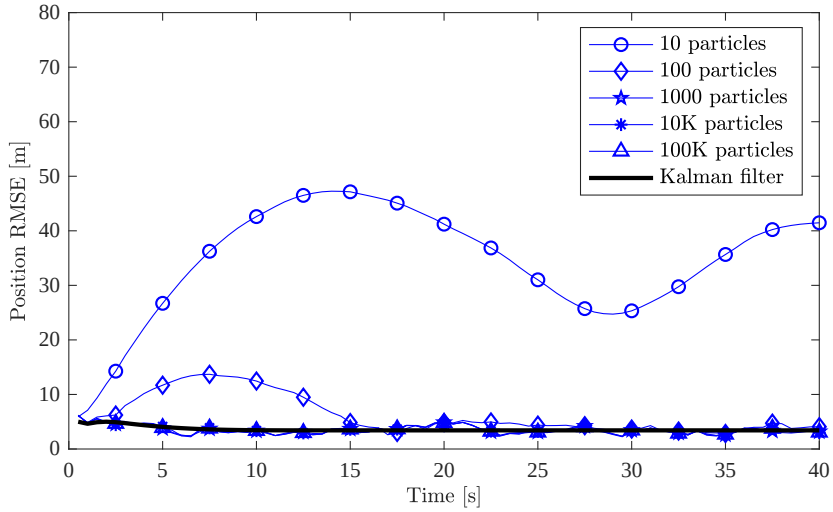


Figure 5.9: Comparison of SIR filter and Kalman filter for the CV model when $\sigma_a = 0.5$

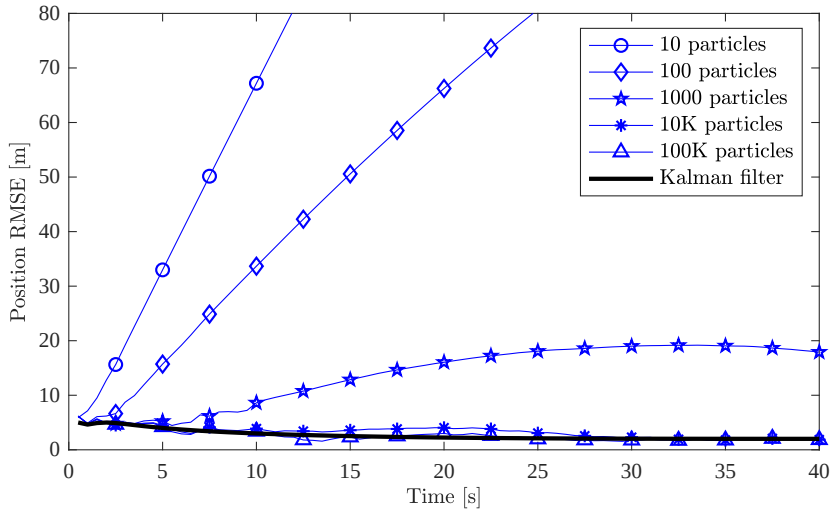


Figure 5.10: Comparison of SIR filter and Kalman filter for the CV model when $\sigma_a = 0.05$

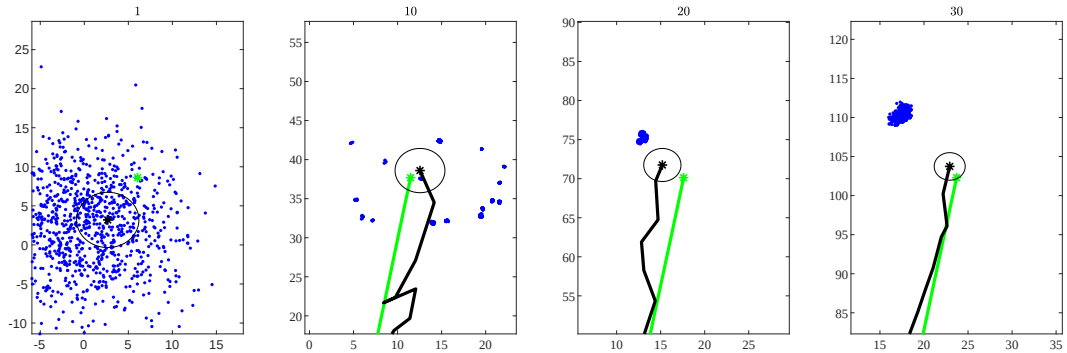


Figure 5.11: Development of particle depletion when $\sigma_a = 0.05$.

The effect seen in Figure 5.11 is known as *particle depletion*. It is clearly a reason to be cautious with particle filters. We can always circumvent the problem by increasing the process noise in the filter model. But this is cheating. If we are not able to present the particle filter with an honest model of the scenario, but must tell it that the scenario is noisier than it actually is, then any uncertainty statistics from the particle filter will be fundamentally unreliable.

The main take-home message from the example is that it is not trivial to design a particle filter that works well. In the remainder of this section on particle filters we will cover three standard approaches to improve particle filter performance: Data-dependent proposal densities, regularization and Rao-Blackwellization. These can very well be combined in the same filter. Among these techniques, only regularization does actually solve the collapse problem seen in Figure 5.11. This is kind of disappointing, as regularization is the least elegant and most heuristic of the three techniques.

5.2.5 Designing a good proposal density

In our derivation of the mathematics of the particle filter in Section 5.2.2 we arrived at a proposal density of the general form $q(\mathbf{x}_k|\mathbf{x}_{1:k-1},\mathbf{z}_{1:k}) = q(\mathbf{x}_k|\mathbf{x}_{k-1},\mathbf{z}_k)$. That is, given a previous particle $\mathbf{x}^i_{1:k-1}$, we draw its child particle $\mathbf{x}^i_{1:k}$ from a pdf that is given by $\mathbf{x}^i_{k-1}$ and $\mathbf{z}_k$. The SIR filter conforms to this scheme, but only $\mathbf{x}^i_{k-1}$ was used in its proposal density, while $\mathbf{z}_k$ was ignored. This is clearly suboptimal.

The *optimal proposal density*, given that $\mathbf{x}^i_{k-1}$ is the previous state, is given by

$$q(\mathbf{x}_k\,|\,\mathbf{x}^i_{k-1},\mathbf{z}_k)_{\text{opt}} = p(\mathbf{x}_k\,|\,\mathbf{x}^i_{k-1},\mathbf{z}_k) \propto p(\mathbf{z}_k\,|\,\mathbf{x}^i_{k-1},\mathbf{x}_k)\,p(\mathbf{x}_k\,|\,\mathbf{x}^i_{k-1}). \tag{5.40}$$

In other words, it is the posterior density of $\mathbf{x}_k$ given that $\mathbf{x}_{k-1}$ had the value $\mathbf{x}^i_{k-1}$. The weight update in (5.35) now becomes

$$w^i_k \propto w^i_{k-1} p(\mathbf{z}_k\,|\,\mathbf{x}^i_{k-1}) = w^i_{k-1} \int p(\mathbf{z}_k\,|\,\mathbf{x}_k) p(\mathbf{x}_k\,|\,\mathbf{x}^i_{k-1})\,\mathrm{d}\mathbf{x}_k. \tag{5.41}$$

With this scheme we are likely to encounter two difficulties. First, the integral in (5.41) will probably not have a closed-form solution if either $p(\mathbf{z}_k\,|\,\mathbf{x}_k)$ or $p(\mathbf{x}_k\,|\,\mathbf{x}^i_{k-1})$ is non-Gaussian or nonlinear[2], which presumably is the case. Second, it may be difficult to sample from $p(\mathbf{x}_k\,|\,\mathbf{x}^i_{k-1},\mathbf{z}_k)$.

Nevertheless, special circumstances exist where the optimal proposal scheme can be used. More precisely, it can be used if the measurement model is linear and both plant noise and measurement noise are Gaussian. In other words, a prerequisite for using the optimal proposal density without approximations is that nonlinearities only occur in the process model. Such a model is of the form

$$\begin{aligned} \mathbf{x}_k &= \boldsymbol{f}(\mathbf{x}_{k-1}) + \mathbf{v}_{k-1} \\ \mathbf{z}_k &= \mathbf{H}\mathbf{x}_k + \mathbf{w}_k \end{aligned} \tag{5.42}$$

where $\mathbf{v}_{k-1}$ and $\mathbf{w}_k$ are mutually independent zero-mean white Gaussian sequences with covariances $\mathbf{Q}$ and $\mathbf{R}$. For particle number i, let us define

$$\begin{aligned} \mathbf{a}^i_k &= \boldsymbol{f}(\mathbf{x}^i_{k-1}) + \boldsymbol{\Sigma}\mathbf{H}^\mathsf{T}\mathbf{R}^{-1}(\mathbf{z}_k - \mathbf{b}^i_k) \\ \boldsymbol{\Sigma} &= \mathbf{Q} - \mathbf{Q}\mathbf{H}^\mathsf{T}\mathbf{S}^{-1}\mathbf{H}\mathbf{Q} \\ \mathbf{S} &= \mathbf{H}\mathbf{Q}\mathbf{H}^\mathsf{T} + \mathbf{R} \\ \mathbf{b}^i_k &= \mathbf{H}\boldsymbol{f}(\mathbf{x}^i_{k-1}). \end{aligned} \tag{5.43}$$

[2] By stating that $p(\mathbf{z}_k\,|\,\mathbf{x}_k)$ is nonlinear it is mean that the relationship between $\mathbf{z}_k$ and $\mathbf{x}_k$ is nonlinear.

The optimal proposal density is then given by

$$q(\mathbf{x}_k \,|\, \mathbf{x}_{k-1}^i, \mathbf{z}_k)_{\text{opt}} = \mathcal{N}(\mathbf{x}_k \,;\, \mathbf{a}_k^i, \boldsymbol{\Sigma}) \tag{5.44}$$

while the corresponding weight update is given by $w_k^i \propto w_{k-1}^i p(\mathbf{z}_k \,|\, \mathbf{x}_k^i)$ where

$$p(\mathbf{z}_k \,|\, \mathbf{x}_{k-1}) = \mathcal{N}(\mathbf{z}_k \,;\, \mathbf{b}_k^i, \mathbf{S}). \tag{5.45}$$

The proof of (5.44) and (5.45) is left as an exercise to the reader.

In the more general case with nonlinear measurements or non-Gaussian noise we can still attempt to *approximate* the optimal proposal density using closed-form techniques, and this is in general a recommended design strategy. If the nonlinear mappings $\boldsymbol{f}(\mathbf{x}_{k-1})$ and $\mathbf{h}(\mathbf{x}_k)$ of the general model are sufficiently smooth, the most obvious strategy is to use an EKF to obtain this approximation. This is known as a local linearization particle filter. In this approach, the proposal density is of the form

$$q(\mathbf{x}_k^i \,|\, \mathbf{x}_{k-1}^i, \mathbf{z}_k) = \mathcal{N}(\mathbf{x}_k^i \,;\, \hat{\mathbf{x}}_k^i, \hat{\mathbf{P}}_k^i) \tag{5.46}$$

where the vector $\hat{\mathbf{x}}_k^i$ and the matrix $\hat{\mathbf{P}}_k^i$ are the output of an EKF for particle i. More precisely, this EKF takes the previous estimate-covariance pair and uses it to make a prediction at the current time step, after which the measurement $\mathbf{z}_k$ is used to obtain the current pair through a standard EKF update:

$$(\mathbf{x}_{k-1}^i, \hat{\mathbf{P}}_{k-1}^i) \longrightarrow (\bar{\mathbf{x}}_k^i, \bar{\mathbf{P}}_k^i) \longrightarrow (\hat{\mathbf{x}}_k^i, \hat{\mathbf{P}}_k^i).$$

With this scheme, the particle weights are updated according to

$$w_k^i \propto w_{k-1}^i \frac{p(\mathbf{z}_k \,|\, \mathbf{x}_k^i) p(\mathbf{x}_k^i \,|\, \mathbf{x}_{k-1}^i)}{\mathcal{N}(\mathbf{x}_k^i \,;\, \hat{\mathbf{x}}_k^i, \hat{\mathbf{P}}_k^i)}. \tag{5.47}$$

5.2.6 Regularized particle filter

Unfortunately, using an optimal or near-optimal proposal density is often not enough to ensure that a particle filter behaves well, even in low-dimensional spaces. This has to do with the particle depletion phenomenon that we encountered in Example 5.5. The culprit behind this is the resampling step: Resampling inevitably has a cluster-generating effect on the particle cloud. Each particle that survives resampling becomes a collection of new particles. Each particle in one such collection will subsequently evolve independently according to the proposal density. If the proposal density is narrow, these particles may take a long time to spread out and fill the gaps between the collections generated by the resampling, and a cluster pattern can be observed.

In contrast to the fragmented particle cloud seen in Figure 5.11, we know that the true posterior in Example 5.5 is a Gaussian. Based on this, it seems that one possible strategy to mitigate the particle depletion problem could be to break up the clusters by some kind of jittering. If we calculate the sample covariance of the particle cloud we get a measure of how much the particles should be jittered. The regularized particle filter uses this information to supplement the conventional resampling step with an additional jittering step.

A key concept in the jittering step is that of a *kernel density*. The kernel density is used to draw a new value of $\mathbf{x}_k^i$ for each resampled particle. The new value should not be too far from the old one, so that the overall particle cloud retains its shape. But it should still be sufficiently far away that the gaps between the depletion fragments are filled. This trade-off is solved by choosing an optimal bandwidth for the kernel. Different kernel shapes are possible. A so-called Epanechnikov

kernel, which basically is a pdf shaped as a concave parabola, is considered the optimal shape [88]. However, a Gaussian kernel is generally easier to implement, and can be written as

$$k(\mathbf{x}_k \,|\, \mathbf{x}_k^i) = \mathcal{N}(\mathbf{x}_k \,;\, \mathbf{x}_k^i, h^2 \hat{\mathbf{P}}) \tag{5.48}$$

where $\hat{\mathbf{P}}$ is the sample covariance of the particle cloud and h is the bandwidth. The sample covariance is given by

$$\hat{\mathbf{x}}_k = \sum_{i=1}^{N} w_k^i \mathbf{x}_k^i \tag{5.49}$$

$$\hat{\mathbf{P}} = \sum_{i=1}^{N} w_k^i (\mathbf{x}_k^i - \hat{\mathbf{x}}_k)(\mathbf{x}_k^i - \hat{\mathbf{x}}_k)^{\mathsf{T}}. \tag{5.50}$$

The optimal bandwidth for the Gaussian kernel is given by

$$h = A \cdot N^{\frac{1}{n+4}} \quad \text{with} \quad A \left(\frac{4}{n+2} \right)^{\frac{1}{n+4}} \tag{5.51}$$

where N is the number of particles and n is the dimension of the state space.

The reason why the regularized particle filter is described as inelegant above is that this step makes it impossible to guarantee that the particles actually do represent the true posterior. For a standard particle filter, one can rest assured that if one just have enough particles, then these particles will be distributed exactly according to the posterior (of course, "enough" may in reality mean billions upon billions). For a regularized particle filter no such convergence result can be established.

5.2.7 Rao-Blackwellization

To make particle filters work with sufficient efficiency to be of practical interest, it is important to utilize whatever knowledge we have about the *structure* of the problem. One such structure of importance is the placement of nonlinearities in the model. A nonlinear system will typically have some state variables that go through nonlinearities, and some state variables that only occur linearly in the process and measurement models. As an example, let us recall the CT model of Example 5.1, and more precisely its discrete-time process model (5.13). In this model, the current state is given by a matrix times the old state plus some process noise. This does indeed sound like a rather linear model. The only reason why it is nonlinear is that the turn-rate appears in that matrix, subject to different trigonometric, and thus nonlinear, functions. In other words, the turn-rate is the only one of the 5 state variables that actually is exposed to nonlinearities. The question therefore arises: Can we use a particle filter to estimate the nonlinear turn-rate state, and leave the remaining states to a Kalman filter? This is known as Rao-Blackwellization.

Since our aim is to understand how a Rao-Blackwellized particle filter (RBPF) works, rather than having an RBPF available for all eventualities, we shall develop the RBPF for a somewhat limited model, that will be sufficient for our purposes. Our model is

$$\mathbf{x}_k^n = \boldsymbol{f}_k(\mathbf{x}_{k-1}^n) + \mathbf{v}_k^n \tag{5.52}$$

$$\mathbf{x}_k^l = \mathbf{A}_k(\mathbf{x}_{k-1}^n)\mathbf{x}_{k-1}^l + \mathbf{v}_k^l \tag{5.53}$$

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k^n) + \mathbf{C}_k(\mathbf{x}_k^n)\mathbf{x}_k^l + \mathbf{w}_k \tag{5.54}$$

where the noises are zero-mean Gaussian with covariances $\mathbf{Q}^n$, $\mathbf{Q}^l$ and $\mathbf{R}$, respectively, and the usual independence assumptions apply. For this model to be feasible it must be possible to partition the state vector $\mathbf{x}_k$ into a nonlinear part $\mathbf{x}_k^n$ and a linear part $\mathbf{x}_k^l$. We see that $\mathbf{x}_k^l$ only occurs in linear

relationships both in the process model and in the measurement model. to separate the linear part from the nonlinear part, we make use of the following factorization of the posterior:

$$p(\mathbf{x}_k^l, \mathbf{x}_{1:k}^n \,|\, \mathbf{z}_{1:k}) = p(\mathbf{x}_k^l \,|\, \mathbf{x}_{1:k}^n, \mathbf{z}_{1:k}) p(\mathbf{x}_{1:k}^n \,|\, \mathbf{z}_{1:k}). \tag{5.55}$$

It is obvious that we can make such a factorization; this result hinges on nothing more than the definition of conditional probability. We have chosen to condition on the trajectory $\mathbf{x}_{1:k}^n$, and not merely on the latest state vector $\mathbf{x}_k^n$, for reasons that will become evident in connection with the following theorem. Let us furthermore assume that the linear part of (5.55) at the previous time step is given by

$$p(\mathbf{x}_{k-1}^l \,|\, \mathbf{x}_{1:k-1}^n, \mathbf{z}_{1:k-1}) = \mathcal{N}(\mathbf{x}_{k-1}^l \,;\, \hat{\mathbf{x}}_{k-1}^l(\mathbf{x}_{1:k-1}^n), \mathbf{P}_{k-1}^l(\mathbf{x}_{1:k-1}^n)). \tag{5.56}$$

In the sequel, we shall suppress the dependencies on $\mathbf{x}_k^n$ and $\mathbf{x}_{1:k-1}^n$, so that we simply can write $\boldsymbol{f}_k^n$, $\mathbf{h}_k^n$, $\mathbf{A}_k$, $\mathbf{C}_k$, $\hat{\mathbf{x}}_{k-1}^l$ and $\mathbf{P}_{k-1}^l$.

Theorem 5.2.1 — The Rao-Blackwellized particle filter. For the linear part of the state vector, the predicted and posterior densities are

$$p(\mathbf{x}_k^l \,|\, \mathbf{x}_{1:k}^n, \mathbf{z}_{1:k-1}) = \mathcal{N}(\mathbf{x}_k^l \,;\, \hat{\mathbf{x}}_{k|k-1}^l, \mathbf{P}_{k|k-1}^l) \tag{5.57}$$

$$p(\mathbf{x}_k^l \,|\, \mathbf{x}_{1:k}^n, \mathbf{z}_{1:k}) = \mathcal{N}(\mathbf{x}_k^l \,;\, \hat{\mathbf{x}}_k^l, \mathbf{P}_k^l) \tag{5.58}$$

where expectations and covariances are given by

$$[\hat{\mathbf{x}}_k^l, \mathbf{P}_k^l, \hat{\mathbf{x}}_{k|k-1}^l, \mathbf{P}_{k|k-1}^l, \mathbf{S}_k^l] = \mathrm{KF}(\hat{\mathbf{x}}_{k-1}^l, \mathbf{P}_{k-1}^l, \mathbf{z}_k - \mathbf{h}_k). \tag{5.59}$$

with $\mathbf{A}_k$, $\mathbf{C}_k$, $\mathbf{Q}^l$ and $\mathbf{R}^l$ playing the role as transition matrix, measurement matrix, process noise matrix and measurement noise matrix, respectively. For the nonlinear part of the state vector, the predicted and posterior densities are given according to

$$p(\mathbf{x}_k^n \,|\, \mathbf{x}_{1:k-1}^n, \mathbf{z}_{1:k-1}) = \mathcal{N}(\mathbf{x}_k^n \,;\, \boldsymbol{f}_k^n, \mathbf{Q}_k^n) \tag{5.60}$$

$$p(\mathbf{z}_k \,|\, \mathbf{x}_{1:k}^n, \mathbf{z}_{1:k-1}) = \mathcal{N}(\mathbf{z}_k \,;\, \mathbf{h}_k + \mathbf{C}_k \,;\, \hat{\mathbf{x}}_{k|k-1}^l, \mathbf{S}_k^l). \tag{5.61}$$

Proof. For the linear part, the key observation to make is that everything is done conditional on the nonlinear trajectory $\mathbf{x}_{1:k}^n$. Thus, both $\mathbf{x}_{1:k-1}^n$ and $\mathbf{x}_k^n$ are treated as if they were known, i.e., deterministic. Under this conditioning, both the process model and the measurement model of the linear part constitute an LTV system (see page 68). The prediction and the posterior are therefore given by a Kalman filter with the appropriate system matrices. Observe that we have to move $\mathbf{h}_k^n$ over to the left-hand side of (5.54) in order to ensure that it is of the same form as a standard linear measurement model.

For the nonlinear part, the prediction formula (5.60) is nothing more than a restatement of the nonlinear process model (5.53). As for the update formula, a quick glance at (5.61) should give a hint that the product identity again will be our friend. The trick is to re-introduce the linear state by means of the total probability theorem:

$$\begin{aligned} p(\mathbf{z}_k \,|\, \mathbf{x}_{1:k}, \mathbf{z}_{1:k-1}) &= \int p(\mathbf{z}_k \,|\, \mathbf{x}_k^l, \mathbf{x}_k^n) p(\mathbf{x}_k^l \,|\, \mathbf{x}_{1:k}^n, \mathbf{z}_{1:k-1}) \, \mathrm{d}\mathbf{x}_k^l \\ &= \int \mathcal{N}(\mathbf{z}_k - \mathbf{h}_k \,;\, \mathbf{C}_k \mathbf{x}_k^l, \mathbf{R}) \mathcal{N}(\mathbf{x}_k^l \,;\, \hat{\mathbf{x}}_{k|k-1}^l, \mathbf{P}_{k|k-1}^l) \mathrm{d}\mathbf{x}_k^l \\ &= \mathcal{N}(\mathbf{z}_k - \mathbf{h}_k \,;\, \mathbf{C}_k \hat{\mathbf{x}}_{k|k-1}^l, \mathbf{C}_k \mathbf{P}_{k|k-1}^l \mathbf{C}_k^\mathsf{T} + \mathbf{R}). \end{aligned} \tag{5.62}$$

The covariance matrix in this Gaussian is nothing else than the matrix $\mathbf{S}_k^l$ that comes out of the Kalman filtering cycle (5.59). To finalize the proof, we see that (5.61) follows by moving $\mathbf{h}_k$ from the random variable slot to the expectation slot. ■

The densities involved in Theorem 5.2.1 provide everything we need to know to implement an RBPF. Conditional on any particle, i.e., realization of the trajectory $\mathbf{x}_{1:k-1}^n$ we use a Kalman filter to estimate $\mathbf{x}_k^l$. Particles can be sampled using the same tools that were available for non-Rao-Blackwellized particle filters. To obtain the posterior particle weights we use the likelihood (5.61) in the same way as we would do for a non-Rao-Blackwellized particle filter.

5.3 References and chapter remarks

We have only scratched the surface of nonlinear filtering in this chapter. This topic will be covered in more depth in the PhD-level course TK8102 Nonlinear State Estimation. Concerning the EKF and linearization-based methods there are at least three possible directions that the reader should be aware of. First, there is the possibility of implementing iterated EKFs, or solving an EKF-style estimation problem through Gauss-Newton optimization. The relationship between these two viewpoints is a central topic in [48]. Second, there is the possibility of calculating more correct covariances in EKF-style problems. This can in some instances be achieved by deterministic sample-based techniques such as the Unscented Kalman filter [57] or by the so-called Laplace approximation [16]. Third, a general framework for linearization possibilities, that include both the EKF, the UKF and several other possibilities, has recently been published in [42].

Regarding sample-based filtering methods, the most important alternative to the particle filter is the point mass filter. This is just an evaluation of the pdfs involved in the Bayes recursion (4.4) and (4.5) on a discrete grid. While straightforward in one dimension, it becomes intractable in dimensions ≥ 4. For treatment of the point mass filter the reader is referred to [48], and also to PhD theses such as [9] and [1]. Other grid-based techniques include variations of dynamic programming [8] and formulation of state estimation as the solution of a stochastic partial differential equation [72] [26].

The standard textbook on nonlinear filtering by means of particle filters is [88]. Particle filtering has been thoroughly treated in a series of PhD theses at Linköping University. In particular, our treatment of the RBPF is based on Schön's thesis [94]. An alternative version of the RBPF has also been proposed in Hendeby's thesis [49]. This version treats the RBPF as a filter bank, consisting of several linear filters running in parallel. We shall not pursue this line of thought any further here. Instead, the next chapter will focus on the standard approach to filter banks used in target tracking.

There exist several other nonlinear filtering techniques that are reminiscent of particle filtering. In particular, we mention the Ensemble Kalman filter (ENKF) [58] which is used for very high-dimensional problems in e.g. meteorology, and the Daum-Huang filter [27] which attempts to implement Bayes' rule by *moving* the particles instead of adjusting their weigths as done in a conventional particle filter. During the presentation of that paper at the SPIE conference, which the author attended, Fred Daum pointed out that this kind of filter was specifically designed to solve "the problem of high SNR", i.e., low process noise.

The CT model is extensively studied in Chapter 4 and 11 of [4]. Our version of the CT model in (5.15) - (5.17) is slightly different than the "canonical" CT model presented in [4]. This is partly due to a different choice of the order of elements in $\mathbf{x}$. More importantly, we have here opted to develop the CT model from a continuous-time perspective, while a pure discrete-time perspective was used in [4], leading to a different $\mathbf{Q}$ matrix. A comparison of EKF and particle filters for the CT model was published in [33].

5.4 Exercises

Exercise 5.1 In Example 5.1, we notice that the CT model gets a larger positional covariance than the CV model even though it uses exactly the same values of σ_a and σ_z. Explain where the the additional uncertainty comes from, and how it is channeled into the position covariance. ■

Exercise 5.2 On page 81 it was claimed that importance sampling for the integral in (5.28) also can be used when the proportionality constant in the pdf $\pi(\mathbf{x})$ is unknown. Show that this works when the normalized importance weights in (5.29) are used. ■

Exercise 5.3 Derive the formulas for optimal proposal density with linear measurement model and Gaussian noise. ■

Exercise 5.4 It was claimed on page 88 that in the case of a linear measurement model the optimal proposal density can be implemented using the closed-form expressions in (5.44) and (5.45). Prove this result. **Hint:** Use the product identity. ■

6. Maneuvering targets and multiple models

In a tracking system, the standard CV model is clearly appropriate if targets move straight ahead in a predictable manner. This is probably the case if we for example want to track a jumbojet crossing the Atlantic. Various non-linear models can be considered if we have knowledge that such a model is more appropriate. This cold for instance be the case if we want to track a satellite in orbit around the Earth. In this case, we have precise nonlinear equations for how the gravity from the Earth, the Moon and the Sun will affect the orbit of the satellite. It could also be tempting to develop more refined nonlinear models for predicting and estimating the state of a target that *maneuvers*. One can, however, raise two objections against this way of dealing with target maneuvers.

First, more complicated model increases the chance that model mismatch can lead to poor performance. No model is perfect, and simplicity both makes tuning easier and prevents the need for introducing unnecessary and possibly harmful prior assumptions. For example, using a model that includes turn rate as a state variable may be problematic if the target actually moves straight ahead. As another example, more refined maneuver models may require us to include acceleration, and possibly even its derivative, also known as jerk, in the state vector. However, white noise models for jerk or its derivatives are not necessarily any more reasonable than white noise models for acceleration, and the consequence of model mismatch become more severe before the model mismatch now is propagated to three integration levels as opposed to two integration levels for the CV model.

Second, maneuvers tend to be something that begins at a definite time, and ends at another time. In other words, it is an on-off-phenomena. To model this as as a process driven by white noise is absurd.

The standard solution to these problems is to use a bank of filters whose motion models are attuned to different reasonable maneuvers. One can for example have one CV model and one CT model. Or one can have one model with low process noise and one model with high process noise. Running all the models in parallell allows the tracking filter to compare the performance of all the models, and then rely the most on the models that perform the best.

The main goal of this chapter is to introduce the Interacting Multiple Models (IMM) algorithm, which is a practical method for using multiple models in parallell. The chapter is organized as

follows. In Section 6.1 we will introduce the generic framework for modeling a given system with the use of multiple models. This is followed by Section 6.2 were we develop the brute-force optimal approach to state estimation for such a system. We will see that the optimal approach is not feasible in practice, and we therefore introduce mixture reduction in Section 6.3 as a tool to keep the computational complexity bounded. Mixture reduction will also play a very central role in Chapters 7 and 8. In Section 6.4 we introduce the IMM algorithm, which is a particularly elegant approach to mixture reduction for multiple model systems. Finally we will discuss some examples in Section **??**.

6.1 Modeling with Multiple Models

Instead of assuming a single model (i.e., combined plant and measurement model) of the form (4.6), we now have M models, also referred to as modes, indexed by $s = 1, \dots, M$, of the form

$$\mathbf{x}_k = \mathbf{f}^{(s)}(\mathbf{x}_{k-1}) + \mathbf{v}_k, \ \mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}^{(s)}) \tag{6.1}$$

$$\mathbf{z}_k = \mathbf{h}^{(s)}(\mathbf{x}_k) + \mathbf{w}_k, \ \mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}^{(s)}). \tag{6.2}$$

We write the models in this nonlinear form because it often is the case that one or more of the models are nonlinear. We will maintain this level of generality as much as possible in this chapter.

At any particular time, the system is in a particular mode. The system changes mode according to a Markov process with transition probabilities

$$\pi^{ij} = \Pr\{\text{The target goes from model } j \text{ at time } k-1 \text{ to model } i \text{ at time } k\} \tag{6.3}$$

$$= \Pr\{s_k = j \,|\, s_{k-1} = i\}. \tag{6.4}$$

We may simply write this as $\Pr\{s_k \,|\, s_{k-1}\}$ if we do not care about spelling out the realizations. It can be convenient to collect the transition probabilities in a matrix

$$\pi = \begin{bmatrix} \pi^{11} & \dots & \pi^{1M} \\ \vdots & & \vdots \\ \pi^{M1} & \dots & \pi^{MM} \end{bmatrix}$$

and also to denote the vector of mode probabilities at time k by $\mathbf{p}_k = [p_k^{(1)}, \dots, p_k^{(M)}]^\mathsf{T}$. Then the mode probabilities evolve according to

$$\mathbf{p}_k = \pi \mathbf{p}_{k-1}. \tag{6.5}$$

■ **Example 6.1** The classical example of where multiple models can be useful is in tracking of civilian airplanes in an air traffic control (ATC) system. Civilian airplanes do generally not pull a lot of crazy stunts. They fly straight ahead, or they perform neat and tidy turns. This is well modeled by the CV model and the CT model, respectively. Furthermore, their general process noise is generally low compared to the maneuver strength of the turns. Using a radar with a resolution of say 100 m will also yield sufficiently accurate measurements to detect that a turn is happening when the high velocities are taken into account. In Figure 6.1 an example simulation of an ATC scenario is displayed. In the bottom plot we see that the probability of the CT model increases soon after the onset of the two maneuvers, and then decrases somewhat more slowly. ■

6.2 Estimation with multiple models: Optimal approach

Before delving into the IMM algorithm, we should have some idea about what this algorithm tries to approximate. In other words, we shall look at the optimal Bayesian solution to multiple model

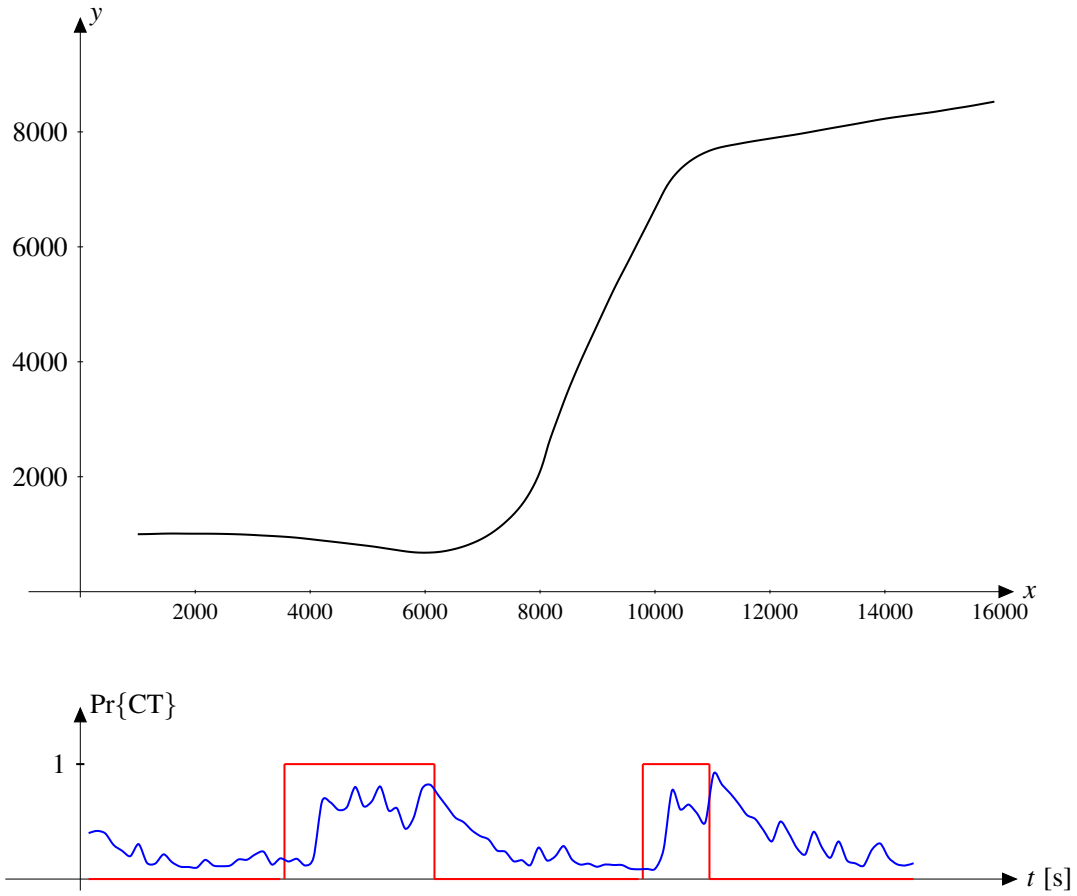


Figure 6.1: Above: Simulation of ATC scenario with two maneuvers. Below: Probabilities of the CT model. Red is ground truth while blue is estimated using the IMM method.

estimation. Every time we predict the system to a new time step, the system could have changed mode. This leads to an ever expanding collection of mode histories. At time $k-1$ we have M^{k-1} mode histories, and *each* of these give rise to M possible mode histories at time k. Let us denote the current mode by s_k, while the previous mode is denoted s_{k-1}, and $s_{1:k}$ denotes the history of modes up to and including time step k. As an inductive hypothesis, we shall assume that the posterior density of $\mathbf{x}_{k-1}$, i.e., the density of the state vector at the previous time step, is of the form

$$p(\mathbf{x}_{k-1} \,|\, \mathbf{z}_{1:k-1}) = \sum_{s_{1:k-1}} \Pr(s_{1:k-1} \,|\, \mathbf{z}_{1:k-1})\, p(\mathbf{x}_{k-1} \,|\, s_{1:k-1}, \mathbf{z}_{1:k-1}) \tag{6.6}$$

This construction is an example of what is known as a *mixture*. It consists of a collection of pdfs, the history-conditional densities $p(\mathbf{x}_{k-1} \,|\, s_{1:k-1}, \mathbf{z}_{1:k-1})$, which are weighted by the probabilities $\Pr(s_{1:k-1} \,|\, \mathbf{z}_{1:k-1})$. We shall then see that after prediction and measurement update, we get a mixture of the same form, but with more components.

Theorem 6.2.1 Let us write the kinematic models in (6.1) and (6.2) as $p(\mathbf{x}_k \,|\, \mathbf{x}_{k-1}, s_k)$ and $p(\mathbf{z}_k \,|\, \mathbf{x}_k, s_k)$. We assume that the model sequence obeys the Markov chain (6.4) and that the pdf of $\mathbf{x}_k$ can be written in the form (6.6). Let us introduce the densities

$$p(\mathbf{x}_k \,|\, s_{1:k}, \mathbf{z}_{1:k-1}) = \int p(\mathbf{x}_k \,|\, \mathbf{x}_{k-1}, s_k) p(\mathbf{x}_{k-1} \,|\, s_{1:k-1}, \mathbf{z}_{1:k-1})\, \mathrm{d}\mathbf{x}_{k-1} \tag{6.7}$$

$$p(\mathbf{x}_k \,|\, s_{1:k}, \mathbf{z}_{1:k}) = \frac{p(\mathbf{z}_k \,|\, \mathbf{x}_k, s_k) p(\mathbf{x}_k \,|\, s_{1:k}, \mathbf{z}_{1:k-1})}{\Lambda^{1:k}} \tag{6.8}$$

where the normalization constant in (6.8) is

$$\Lambda^{1:k} = p(\mathbf{z}_k | s_{1:k}, \mathbf{z}_{1:k-1}) = \int p(\mathbf{z}_k \,|\, \mathbf{x}_k, s_k) p(\mathbf{x}_k \,|\, s_{1:k}, \mathbf{z}_{1:k-1})\, \mathrm{d}\mathbf{x}_k. \tag{6.9}$$

Let us also introduce the posterior model history probability

$$\Pr(s_{1:k} \,|\, \mathbf{z}_{1:k}) \propto \Lambda^{1:k} \Pr(s_k \,|\, s_{k-1}) \Pr\{s_{1:k-1} \,|\, \mathbf{z}_{1:k-1}\}. \tag{6.10}$$

The posterior density of the state vector for multiple model estimation at time step k is then

$$p(\mathbf{x}_k \,|\, \mathbf{z}_{1:k}) = \sum_{s_{1:k}} \Pr\{s_{1:k} \,|\, \mathbf{z}_{1:k}\}\, p(\mathbf{x}_k \,|\, s_{1:k}, \mathbf{z}_{1:k}). \tag{6.11}$$

Before proving the theorem, let us briefly recapitulate what is going on here. The main result is (6.11), whose components are given by (6.10) and (6.8), which again are given by (6.9) and (6.7), respectively. The density $p(\mathbf{x}_k \,|\, s_{1:k}, \mathbf{z}_{1:k-1})$ in (6.7) is the predicted density of the state vector, based on the previous measurements and a particular model history $s_{1:k}$. The current mode s_k is also included, because it governs the process model (6.1). The density $p(\mathbf{x}_k \,|\, \mathbf{z}_{1:k}, s_{1:k})$ in (6.8) is the posterior density of the state vector for the same model history. Its normalization constant $\Lambda^{1:k}$ works as a likelihood for calculating the posterior probability $\Pr(s_{1:k} \,|\, \mathbf{z}_{1:k})$ of the history $s_{1:k}$.

Proof. Let us first take a look at the expressions for the predicted and posterior history-conditional densities in (6.7) and (6.8), respectively. The expression for the predicted density is as usual nothing more than an application of the total probability theorem, with the simplifications that can be made thanks to the Markov assumptions. For instance, if $\mathbf{x}_{k-1}$ and s_k are known, the neither previous states, modes nor measurements carry any additional information about $\mathbf{x}_k$. The expression for the posterior density follows from Bayes' rule in a similar straightforward manner.

What remains is to show that the expressions in (6.9), (6.10) and (6.11) play the prescribed roles in the total posterior $p(\mathbf{x}_k \,|\, \mathbf{z}_{1:k})$. We start by working on the general expression for the posterior of

the state vector. We expand it to range over all the model histories by means of the total probability theorem, we use Bayes' rule to bring forward the current measurement, and we factorize the predicted joint distribution of the state vector and the model history according to the definition of conditional probability:

$$\begin{aligned} p(\mathbf{x}_k \,|\, \mathbf{z}_{1:k}) &= \sum_{s_{1:k}} p(\mathbf{x}_k, s_{1:k} \,|\, \mathbf{z}_{1:k}) \\ &= \frac{1}{p(\mathbf{z}_k \,|\, \mathbf{z}_{1:k-1})} \sum_{s_{1:k}} p(\mathbf{z}_k \,|\, \mathbf{x}_k, s_{1:k}) p(\mathbf{x}_k, s_{1:k} \,|\, \mathbf{z}_{1:k-1}) \\ &= \frac{1}{p(\mathbf{z}_k \,|\, \mathbf{z}_{1:k-1})} \sum_{s_{1:k}} p(\mathbf{z}_k \,|\, \mathbf{x}_k, s_{1:k}) p(\mathbf{x}_k \,|\, s_{1:k}, \mathbf{z}_{1:k-1}) \Pr\{s_{1:k} \,|\, \mathbf{z}_{1:k-1}\}. \end{aligned} \tag{6.12}$$

Thanks to the Markov properties of the system we can simplify the first term in the sum as follows:

$$p(\mathbf{z}_k \,|\, \mathbf{x}_k, s_{1:k}) = p(\mathbf{z}_k \,|\, \mathbf{x}_k, s_k). \tag{6.13}$$

We can then recognize the product of the first two terms in the sum as proportional to the history-conditional posterior density in (6.8). That is,

$$p(\mathbf{z}_k \,|\, \mathbf{x}_k, s_{1:k}) p(\mathbf{x}_k \,|\, s_{1:k}, \mathbf{z}_{1:k-1}) = p(\mathbf{x}_k \,|\, s_{1:k}, \mathbf{z}_{1:k}) \Lambda^{1:k}. \tag{6.14}$$

If we insert this in (6.12) we obtain

$$p(\mathbf{x}_k \,|\, \mathbf{z}_{1:k}) = \frac{1}{p(\mathbf{z}_k \,|\, \mathbf{z}_{1:k-1})} \sum_{s_{1:k}} p(\mathbf{x}_k \,|\, s_{1:k}, \mathbf{z}_{1:k}) \Lambda^{1:k} \Pr\{s_k \,|\, s_{k-1}\} \Pr\{s_{1:k-1} \,|\, \mathbf{z}_{1:k-1}\}. \tag{6.15}$$

However, we could also, by re-arranging the first step in (6.12) according to the definition of conditional probability, have written the posterior as was done in (6.11). By comparing this with (6.15) it follows that $\Pr\{s_{1:k} \,|\, \mathbf{z}_{1:k}\}$ must be equal to the product of all the terms that do not contain $\mathbf{x}_k$ in (6.15). In other words,

$$\Pr(s_{1:k} \,|\, \mathbf{z}_{1:k}) \frac{1}{p(\mathbf{z}_k \,|\, \mathbf{z}_{1:k-1})} \Lambda^{1:k} \Pr(s_k \,|\, s_{k-1}) \Pr\{s_{1:k-1} \,|\, \mathbf{z}_{1:k-1}\}. \tag{6.16}$$

This concludes the proof. ■

The recursion in terms of histories $s_{1:k}$ quickly becomes infeasible due to the exponential number of possible histories. For every parent history $s_{1:k-1}$ there are M possible child histories, and again M possible child histories for each of these, and so on. Any implementation of this scheme must therefore rely on pruning techniques. A possibility is therefore to prune away histories with low probabilities. Another possibility is to employ N-scan pruning, in which only the best history at time step $k-N$ is kept, and all other histories are discarded. A more popular alternative to pruning is to merge old histories together. We must then replace several pdfs (typically Gaussians) by fewer pdfs (typically a single Gaussian) which constitute an approximation of the same information. This is known as *mixture reduction*. Both the IMM method, and the tracking methods to be presented in subsequent chapters, utilize moment-based reduction of Gaussian mixtures, and we shall therefore discuss this in detail in the next section.

6.3 Gaussian mixtures, their moments and mixture reduction

A Gaussian mixture is a function of the form

$$f(\mathbf{x}) = \sum_{i=1}^{M} w^i \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}^i, \mathbf{P}^i) \quad \text{where} \quad \sum_{i=1}^{M} w^i = 1 \quad \text{and} \quad w^i \geq 0 \quad \text{for all} \quad i. \tag{6.17}$$

The normalization requirement ensures that the mixture integrates to one, and thus is a valid pdf itself. We see that a Gaussian mixture with M components is given by the weights, expectations and covariances of the individual components. It is fairly evident that the expectation of $\mathbf{x}$ over the mixture $f(\mathbf{x})$ is given by

$$\bar{\boldsymbol{\mu}} = \sum_{i=1}^{M} w^i \boldsymbol{\mu}^i. \tag{6.18}$$

The expression for the covariance is more complicated: In addition to a sum of individual covariances as in (6.18), it also contains a term called the spread-of-the-innovations which quantifies how much the individual expectations differ.

Theorem 6.3.1 — Covariance of Gaussian mixture. The covariance of the Gaussian mixture $f(\mathbf{x})$ in (6.17) is

$$\bar{\mathbf{P}} = \sum_{i=1}^{M} w^i \mathbf{P}^i + \tilde{\mathbf{P}} \tag{6.19}$$

where the spread-of-the-innovovations term is given by

$$\tilde{\mathbf{P}} = \sum_{i=1}^{M} w^i (\boldsymbol{\mu}^i - \bar{\boldsymbol{\mu}})(\boldsymbol{\mu}^i - \bar{\boldsymbol{\mu}})^\mathsf{T} = \sum_{i=1}^{M} w^i \boldsymbol{\mu}^i (\boldsymbol{\mu}^i)^\mathsf{T} - \bar{\boldsymbol{\mu}}\bar{\boldsymbol{\mu}}^\mathsf{T}. \tag{6.20}$$

Proof. The following proof is adapted from [4] p. 56. We think of the mixture in the following way. Let A^i be one of M possible events that happens with probability w^i. If the event A^i happens, then $\mathbf{x}$ is distributed according to $\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}^i, \mathbf{P}^i)$. Based on this perspective, we can express the covariance of the mixture by means of conditional expectation:

$$\begin{aligned}
\bar{\mathbf{P}} &= \sum_{i=1}^{M} w^i E\left[(\mathbf{x} - \bar{\boldsymbol{\mu}})(\mathbf{x} - \bar{\boldsymbol{\mu}})^\mathsf{T} \,|\, A^i\right] \\
&= \sum_{i=1}^{M} w^i E\left[(\mathbf{x} - \boldsymbol{\mu}^i + \boldsymbol{\mu}^i - \bar{\boldsymbol{\mu}})(\mathbf{x} - \boldsymbol{\mu}^i + \boldsymbol{\mu}^i - \bar{\boldsymbol{\mu}})^\mathsf{T} \,|\, A^i\right].
\end{aligned} \tag{6.21}$$

The expectation in the second step can also be written as

$$\begin{aligned}
&E[(\mathbf{x} - \boldsymbol{\mu}^i)(\mathbf{x} - \boldsymbol{\mu}^i)^\mathsf{T} \,|\, A^i] + E[(\mathbf{x} - \boldsymbol{\mu}^i)(\boldsymbol{\mu}^i - \bar{\boldsymbol{\mu}})^\mathsf{T} \,|\, A^i] \\
&+ E[(\boldsymbol{\mu}^i - \bar{\boldsymbol{\mu}})(\mathbf{x} - \boldsymbol{\mu}^i)^\mathsf{T} \,|\, A^i] + E[(\boldsymbol{\mu}^i - \bar{\boldsymbol{\mu}})(\boldsymbol{\mu}^i - \bar{\boldsymbol{\mu}})^\mathsf{T} \,|\, A^i].
\end{aligned} \tag{6.22}$$

Here we notice that the two middle terms actually become zero, because $\mathbf{x}$ under the event A^i is distributed according to a Gaussian that is symmetric around $\boldsymbol{\mu}^i$. It follows that

$$\begin{aligned}
\bar{\mathbf{P}} &= \sum_{i=1}^{M} w^i E\left[(\mathbf{x} - \boldsymbol{\mu}^i)(\mathbf{x} - \boldsymbol{\mu}^i)^\mathsf{T} \,|\, A^i\right] + \sum_{i=1}^{M} w^i E\left[(\boldsymbol{\mu}^i - \bar{\boldsymbol{\mu}})(\boldsymbol{\mu}^i - \bar{\boldsymbol{\mu}})^\mathsf{T} \,|\, A^i\right] \\
&= \sum_{i=1}^{M} w^i \mathbf{P}^i + \sum_{i=1}^{M} w^i (\boldsymbol{\mu}^i - \bar{\boldsymbol{\mu}})(\boldsymbol{\mu}^i - \bar{\boldsymbol{\mu}})^\mathsf{T}
\end{aligned} \tag{6.23}$$

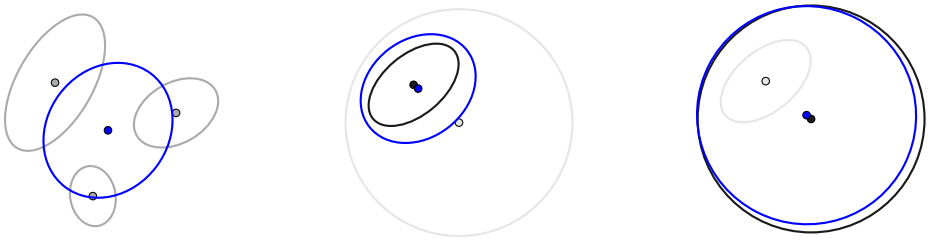
where the second sum was handled by noting that both $\boldsymbol{\mu}^i$ and $\bar{\boldsymbol{\mu}}$ are non-random.

The second expression for $\tilde{\mathbf{P}}$ can be derived as follows:

$$\begin{aligned}\tilde{\mathbf{P}} &= \sum_{i=1}^{M} w^i \left(\boldsymbol{\mu}^i (\boldsymbol{\mu}^i)^\mathsf{T} - \boldsymbol{\mu}^i \bar{\boldsymbol{\mu}}^\mathsf{T} - \bar{\boldsymbol{\mu}} (\boldsymbol{\mu}^i)^\mathsf{T} + \bar{\boldsymbol{\mu}} \bar{\boldsymbol{\mu}}^\mathsf{T} \right) \\ &= \sum_{i=1}^{M} w^i \boldsymbol{\mu}^i (\boldsymbol{\mu}^i)^\mathsf{T} - \left(\sum_{i=1}^{M} w^i \boldsymbol{\mu}^i \right) \bar{\boldsymbol{\mu}}^\mathsf{T} - \bar{\boldsymbol{\mu}} \left(\sum_{i=1}^{M} w^i \boldsymbol{\mu}^i \right)^\mathsf{T} + \bar{\boldsymbol{\mu}} \bar{\boldsymbol{\mu}}^\mathsf{T} \sum_{i=1}^{M} w^i. \end{aligned} \tag{6.24}$$

We see that each of the last three terms in (6.24) is equal to $\bar{\boldsymbol{\mu}} \bar{\boldsymbol{\mu}}^\mathsf{T}$. Since two are negative and one is positive, we are left with one negative after cancellation, and the last equality of (6.20) results. ■

■ **Example 6.2** In Figure 6.2 we see three examples of Gaussian mixtures. In the leftmost example the mixture consists of three well separated Gaussians with equal weights. The covariance of the mixture is shown by means of the blue ellipse, and its expectation is the blue dot. We see that approximating the mixture by a single Gaussian gives a reasonable uncertainty representation, but the expectation ends up in a region that is not representative of the mixture at all. The two illustrations to the right display scenarios that are more typical of what one often will encounter in a tracking system. In the middle illustration we see a mixture with one strong and certain component, and a weaker and more uncertain component. In this particular case, we see that the covariance ellipse of the single-Gaussian approximation is just large enough to almost include the expectation of the weaker component. In the rightmost illustration we have a situation where a weak and precise component possibly should be considered, but is dominated by a stronger component with a larger covariance. In this case the weak component only causes a marginal decrease in the uncertainty of the single-Gaussian approxmation.



(a) (b) (c)

Figure 6.2: Examples of Gaussian mixtures.

■

The moment-based approach described by (6.18) and Theorem 6.3.1 is the simplest approach to mixture reduction beyond pure pruning, but many other approaches exist as well. If one have M Gaussians, and only want to reduce it to $N \leq M$ Gaussians, then one can devise various heuristic schemes that merges two Gaussians if they are sufficiently similar, and otherwise leave them alone until the desired reduction has been achieved. A similar philosophy is used by the K-means clustering algorithm, which can be used to reduce a mixture to a given number of Gaussians.

Another approach to the mixture reduction problem is found in *variational inference*. The idea behind variational inference is to specify a cost function that describes how much an approximative pdf (e.g., a single Gaussian) differs from the full mixture. The parameters of the approximation are then adjusted through optimization in order to minimize the cost function. An example of such a cost function could be the Kullback-Leibler divergence that we briefly encountered on page 24. For more in-depth reading on mixture reduction the reader may consult references such as [92] and [116].

6.4 Interacting Multiple Models

Assume that the previous posterior $p(\mathbf{x}_{k-1}\,|\,\mathbf{z}_{1:k-1})$ is a mixture over M components, one for each mode, at time step $k-1$. Let us furthermore assume that that every mode-matched density in this mixture is a Gaussian. Then the previous posterior can be written as

$$p(\mathbf{x}_{k-1}\,|\,\mathbf{z}_{1:k-1}) = \sum_{s_{k-1}} \Pr(s_{k-1}\,|\,\mathbf{z}_{1:k-1})\,\mathcal{N}\left(\mathbf{x}_{k-1}\,;\,\hat{\mathbf{x}}_{k-1}^{(s_{k-1})}, \mathbf{P}_{k-1}^{(s_{k-1})}\right). \tag{6.25}$$

We use the paranthesis-in-the-superscript notation for the parameters of the Gaussians to emphasize that these superscripts have to do with modes. The expression (6.25) describes the situation at time k. The new model s_k will thereafter govern both the state transition and the measurement at time k. Without mixture reduction, we would then get M^2 components at time k, and this would continue to grow exponentially for subsequent time steps. A natural solution to the mixture reduction problem could be to reduce these M^2 components to M components *after* the prediction, one for each of the new models s_k. This is known as the Generalized Pseudo-Bayesian 2 (GBP2) approach. However, other mixture reduction strategies are also possible, and may be perform better or worse than the suggested strategy. In particular, the IMM method is based on a small twist of the suggested scheme: It reduces the M^2 components to M components, one for each value of s_k, *before* the prediction. In mathematical terms, the difference between GBP2 and IMM is whether we approximate $p(\mathbf{x}_k\,|\,\mathbf{z}_{1:k-1}, s_k)$ or $p(\mathbf{x}_{k-1}\,|\,\mathbf{z}_{1:k-1}, s_k)$ by a single Gaussian. There is an important computational benefit in making the latter choice: While GBP2 needs to maintain M^2 filters, IMM only needs to maintain M filters.

6.4.1 Workflow of the IMM method

Each estimation cycle of the IMM method consists of four steps:

1. Calculation of mixing probabilities.
2. Mixing.
3. Mode matched filtering.
4. Update of mode probabilities.

In the first two steps, the mixture-reduction approximation from Section (6.3) is invoked, while the current data set is processed in the last two steps.

Step 1: The mixing probabilities, denoted $\mu_{s_{k-1}|s_k}$, are the probabilities that model s_{k-1} was the true mode at time step $k-1$, given that model s_k is the true model at time step k. We find these probabilities as follows

$$\begin{aligned}
\mu_{s_{k-1}|s_k} &= \Pr\{s_{k-1}\,|\,s_k, \mathbf{z}_{1:k-1}\} \\
&= \frac{\Pr\{s_k\,|\,s_{k-1}, \mathbf{z}_{1:k-1}\}\Pr\{s_{k-1}\,|\,\mathbf{z}_{1:k-1}\}}{\Pr\{s_k\,|\,\mathbf{z}_{1:k-1}\}} \\
&\propto \Pr\{s_k\,|\,s_{k-1}, \mathbf{z}_{1:k-1}\}\Pr\{s_{k-1}\,|\,\mathbf{z}_{1:k-1}\} \\
&= \pi^{s_{k-1}s_k} p_{k-1}^{(s_{k-1})}.
\end{aligned} \tag{6.26}$$

Notice how Bayes' rule is used: We need an expression describing s_{k-1} conditional on s_k, but what we have is the reverse relationship. Since we only are interested in probabilities of s_{k-1}, we do not need to worry about the normalization constant $\Pr\{s_k\,|\,\mathbf{z}_{1:k-1}\}$, and we hide it inside the proportionality sign on the third line. On the fourth line we have simply revived the matrix-vector notation that we introduced in (6.5) as an alternative expression for the mixing probabilities, since this is how the mixing typically is implemented in a computer.

Step 2: For every model s_k at time k we know want to obtain a Gaussian $\mathcal{N}(\mathbf{x}_{k-1}\,;\,\hat{\mathbf{x}}_{k-1}^{0,(s_k)}, \mathbf{P}_{k-1}^{0,(s_k)})$ that is a best fit description of the pdf $p(\mathbf{x}_{k-1}\,|\,s_k, \mathbf{z}_{1:k-1})$. By invoking the mixing probabilities and

the assumption that we had M Gaussians at time $k-1$ we can write it as

$$p(\mathbf{x}_{k-1}\,|s_k,\mathbf{z}_{1:k-1}) = \sum_{s_{k-1}} \mu_{s_{k-1}|s_k}\,\mathcal{N}(\mathbf{x}_{k-1}\,;\hat{\mathbf{x}}_{k-1}^{(s_{k-1})},\mathbf{P}_{k-1}^{(s_{k-1})}). \tag{6.27}$$

This is a Gaussian mixture, and its moment-based approximation of the form $\mathcal{N}(\mathbf{x}_{k-1}\,;\hat{\mathbf{x}}_{k-1}^{0,(s_k)},\mathbf{P}_{k-1}^{0,(s_k)})$ is given by

$$\hat{\mathbf{x}}_{k-1}^{0,(s_k)} = \sum_{s_{k-1}} \mu_{s_{k-1}|s_k}\hat{\mathbf{x}}_{k-1}^{(s_{k-1})} \tag{6.28}$$

$$\mathbf{P}_{k-1}^{0,(s_k)} = \sum_{s_{k-1}} \mu_{s_{k-1}|s_k}\left\{\mathbf{P}_{k-1}^{(s_{k-1})} + (\hat{\mathbf{x}}_{k-1}^{0,(s_k)} - \hat{\mathbf{x}}_{k-1}^{(s_{k-1})})(\hat{\mathbf{x}}_{k-1}^{0,(s_k)} - \hat{\mathbf{x}}_{k-1}^{(s_{k-1})})^{\mathsf{T}}\right\}. \tag{6.29}$$

Step 3: Having approximated the posterior at time $k-1$ as a mixture over s_k, we are ready to conduct mode-matched filtering for reach realization of s_k. This involves an EKF cycle which we summarize by referring to Algorithm 2:

$$[\hat{\mathbf{x}}_k^{(s_k)}, \mathbf{P}_k^{(s_k)}, \hat{\mathbf{x}}_{k|k-1}^{(s_k)}, \mathbf{P}_{k|k-1}^{(s_k)}, \mathbf{S}_k^{(s_k)}] = \mathrm{EKF}(\hat{\mathbf{x}}_{k-1}^{0,(s_k)}, \mathbf{P}_{k-1}^{0,(s_k)}, \mathbf{z}_k). \tag{6.30}$$

We see that the moments $\hat{\mathbf{x}}_{k-1}^{0,(s_k)}$ and $\mathbf{P}_{k-1}^{0,(s_k)}$ from the mixing step are used as input to the prediction-estimation cycle. The appropriate model parameters (the functions $\boldsymbol{f}^{(s_k)}$ and $\mathbf{h}^{(s_k)}$, and the matrices $\mathbf{Q}^{(s_k)}$ and $\mathbf{R}^{(s_k)}$) of the given realization of s_k are used within the EKF-cycle. As a byproduct of this step, we also calculate the mode-conditional likelihood, which is given by

$$\begin{aligned}\Lambda_k^{(s_k)} &= p(\mathbf{z}_k\,|\,s_k,\mathbf{z}_{1:k-1})\\ &= \int p(\mathbf{z}_k\,|\,\mathbf{x}_k)p(\mathbf{x}_k\,|\,s_k,\mathbf{z}_{1:k-1})\,\mathrm{d}\mathbf{x}_k\\ &= \mathcal{N}\left(\mathbf{z}_k\,;\mathbf{h}^{(s_k)}\left(\hat{\mathbf{x}}_{k|k-1}^{(s_k)}\right),\mathbf{S}_k^{(s_k)}\right).\end{aligned} \tag{6.31}$$

The last equality in (6.31) is, as the reader perhaps can guess, due to the product identity.
Step 4: For the last step, the update of mode probabilities, we use the mode conditional likelihood and the predicted probability of the mode s_k, which we denote $p_{k|k-1}^{(s_k)}$, and which is given by (6.5). The posterior probability of the mode s_k is then, according to Bayes' rule, given by

$$p_k^{(s_k)} = \Pr\{s_k\,|\,\mathbf{z}_{1:k}\} = \frac{\Lambda_k^{(s_k)}p_{k|k-1}^{(s_k)}}{\sum_{s_k}\Lambda_k^{(s_k)}p_{k|k-1}^{(s_k)}}. \tag{6.32}$$

As for any other filtering technique, challenges not directly encapsulated by the description of the algorithm may arise during practical implementation. The normalization constant in (6.32) is likely to reach so small values that it becomes indistinguishable from zero. To prevent this from happening, the mode probabilities should not be allowed to sink below some predetermined limit. Another common challenge arises when different state spaces are used for the different modes.

Also, as any other filtering technique, the IMM method has its limitations. It can only decide which mode the system is in if the abilities of the modes to explain the data are sufficiently different. Said in different words, if all modes are equally probable, then the IMM method is of limited utility. We will investigate some quantification of this in the next section.

6.5 References and chapter remarks

The IMM method and its relatives GPB1 and GPB2 are extensively studied in Chapter 11 of [4]. Our treatment of multiple models and the IMM method is a very brief summary of the essential information from that chapter. Treatments can also be found in other textbooks on sensor fusion, for instance in Chapter 10 of [48]. Extensive treatments of mixture reduction and variational inference can be found in probabilistic machine learning textbooks such as [10] and [76].

7. Single-target tracking: The PDAF and relatives

In this chapter, we begin our study of target tracking, as opposed to pure filtering, which has been discussed in Chapters 4 - 6. The difference is that in target tracking, we generally receive several measurements. This raises the question about which measurement should be used in the filter update. A standard assumption is that the target of interest generates at most one measurement. If it does not generate a measurement, then we have a *misdetection*. Other measurements are generated by other targets and also by clutter. Clutter is also known as false alarms. In this first chapter on target tracking, we will assume that only one target is present, so that all non-target measurements necessarily are clutter.

Because of this measurement origin uncertainty, a straightforward Kalman filter is not sufficient to solve our problem any longer, even in the Gaussian-linear case. Instead, we need more complex methods which associate measurements to the target. This problem of data association can be solved in different manners. In very broad terms, we have two possibilities, which in some sense corresponds to MAP and MSEE estimation: We can choose the best measurement, and discard all other measurements, or we can attempt to calculate a weighed average over all the possible measurements, where the weights quantify how likely the measurement is to be the one true measurement.

This latter approach can be implemented by adapting the mixture reduction formulas from Section 6.3 to single-target tracking. This leads to a very efficient tracking method known as the probabilistic data association filter (PDAF). This tracking method, and a couple of close relatives, are the main topic of this chapter.

7.1 The prior and the validation gate

7.2 Clutter

The measurements that we feed to a tracking method are typically extracted from a sensor array, such as a radar image. For a given resolution cell we have a Boolean detector, which is supposed to report whenever the target resides in that resolution cell. Otherwise, it should not report anything. However, since no detector is perfect, there will be some probability that it also reports a false

alarm when no target is present. This probability is known as the false alarm rate P_{FA}. Typical values for the false alarm rate are in the range of 10^{-6} to 10^{-2}.

If we assume that the detection processes in different resolution cells are i.i.d., then the total number of false alarms, which we denote φ, is distributed according to a Binomial distribution

$$\mu(\phi) = \binom{N}{\varphi} P_{\text{FA}}^{\varphi}(1-P_{\text{FA}})^{N-\varphi} \tag{7.1}$$

where N is the total number of resolution cells. The binomial distribution is cumbersome to work with, but when N is large, say > 30, it can be approximated with either a Gaussian distribution or a Poisson distribution, depending on how the limit is taken. For modeling the cardinality of clutter, the Gaussian approximation is clearly not appropriate, since it will assign some probability to have a negative number of clutter points, and also since it is a continuous distribution. The Poisson approximation, on the other hand, does not assign any probability to negative numbers, and is a standard model for clutter in target tracking. If we define $\Lambda = NP_{\text{FA}}$, the Poisson clutter model becomes

$$\mu(\phi) = e^{-\Lambda}\frac{\Lambda^{\varphi}}{\varphi!} \tag{7.2}$$

The Poisson distribution (7.2) is related to a more complicated construction known as a Poisson Point Process (PPP). We postpone the discussion of general point processes to Chapter 13, but a very brief introduction is necessary at this stage. In a PPP, we first sample a random number of points according to (7.2), and then sample the locations of these points i.i.d. according to a uniform pdf in some region $\mathscr{S}$ with volume V. We refer to $\lambda = \Lambda/V$ as the intensity of the PPP. This collection of points will have two properties:

- The numbers of points in two disjoint regions will be independent.
- For any subregion $\mathscr{R} \subset \mathscr{S}$, the number of point will be Poisson distributed, and the rate of this Poisson distribution will be the integral of the intensity over $\mathscr{R}$.

Based on intuition, both these properties are reasonable for clutter. The Poisson model is, however, not without weaknesses. The main problem is that it is difficult to know the correct value of λ. Estimates of this quality will under most circumstances suffer from huge uncertainties, which may or may not have a significant impact on the tracking method.

An alternative clutter model is the diffuse cardinality model, in which all cardinalities have the same probability:

$$\mu(\phi = 0) = \mu(\phi = 1) = \mu(\phi = 2) = \ldots = \varepsilon$$

The spatial density of clutter measurements is again assumed uniform within the region of interest. It is sometimes claimed that the diffuse pmf is improper, in the sense that it does not sum up to one if $\varepsilon > 0$. This is, however, really not an issue to be concerned about, as one can always truncate it for a sufficiently large number (say ten billion) so that it remains a well-defined pmf.

A major caveat in this discussion is that blurring effects and target extent in reality will cause a target to be seen in several resolution cells, and clustering is therefore normally used as part of the detection procedure. This invalidates the idea that the Binomial pmf of the clutter cardinality can be found directly from the false alarm rate and the number of resolution cells. Furthermore, it can be very difficult to design a detector that actually manages to maintain the desired false alarm rate. It may then be unreasonable to assume a fixed clutter intensity, and instead we may treat it as a function $\lambda(\mathbf{z})$ in the measurement space. In this case we find the total clutter rate as the integral of the intensity

$$\Lambda = \int_{\mathscr{S}} \lambda(\mathbf{z})\mathrm{d}\mathbf{z}.$$

7.3 Misdetections

The second complication, which makes target tracking more complicated than pure filtering, is that the target may or may not be detected. If we assume that the detection event (i.e., whether or not the target is detected) at time k is independent of the detection events at all other time steps, then the detection model is extremely simple: It is given by a Bernoulli random variable:

$$P(\delta) = \begin{cases} P_{\mathrm{D}} & \text{if } \delta = 1 \\ 1 - P_{\mathrm{D}} & \text{if } \delta = 0. \end{cases}$$

Again, the main complication is the need to guess a suitable value for the underlying parameter, which for this model is the detection probability P_{D}. Typical values are in the range from 0.5 (e.g., sonar tracking applications) to 0.95 (e.g., radar tracking of airplanes).

It should be mentioned that the false alarm rate and the detection probability are not independent of each other. If we want a higher detection probability, we can achieve that by lowering the detection threshold, which also will yield a higher false alarm rate. Choosing a good trade-off (known as operating point) is largely a matter of engineering judgement. We will return to this topic on page 119 and in Chapter 14.

7.4 The PDAF: Moment-based mixture reduction

Having introduced the main conceptual phenomena which makes target tracking more complex than pure filtering, we see that in target tracking we receive a (possibly empty) set of measurements $Z_k = \{\mathbf{z}_k^1, \dots, \mathbf{z}_k^{m_k}\}$ at each time step instead of just a single measurement vector $\mathbf{z}_k$. We use the notation $Z_{1:k}$ to denote the collection of measurement sets received at time step k.

7.4.1 Single-target assumptions

The goal of a Bayesian single-target tracking method is to estimate the predicted pdf $p_{k|k-1}(\mathbf{x}_k) = p(\mathbf{x}_k|Z_{1:k-1})$ and the posterior pdf $p_k(\mathbf{x}_k) = p(\mathbf{x}_k|Z_{1:k})$. To do this we need to specify a model that is sufficiently rich to make these densities well-defined. We will first familiarize ourselves with a somewhat general version of this model, before we make it more concrete by invoking Gaussian-linearity and so on. The benefit of this approach is that it provides us with expressions of general pdfs which make the derivations of the PDAF and other tracking methods significantly less cluttered than it would be if we were going to write everything in terms of Gaussians. The general model that we start with consist of the following 7 assumptions:

- S1 At time step $k-1$ one and only one target exists in the surveillance region $\mathscr{S}$ with state vector $\mathbf{x}_{k-1}$.
- S2 The prior density of $\mathbf{x}_{k-1}$ is given as $p_{k-1}(\mathbf{x}_{k-1})$.
- S3 The state vector of the target evolves from time step $k-1$ to k according to a Markov model of the form $f_{\mathbf{x}}(\mathbf{x}_k|\mathbf{x}_{k-1})$.
- S4 A measurement of the target is detected with probability P_{D}.
- S5 If a target-originating measurement exists, then it is related to $\mathbf{x}_k$ according to a likelihood of the form $f_{\mathbf{z}}(\mathbf{z}_k|\mathbf{x}_k)$.
- S6 An unknown number of φ_k measurements originate from clutter, where the discrete-valued random variable φ_k is distributed according to $\mu(\varphi)$.
- S7 If $\mathbf{z}$ is a clutter measurement, then it is distributed according to a pdf $c(\mathbf{z})$, independently of all other measurements.

Assumptions S2, S3 and S5 are similar to the assumptions underlying conventional filtering. In the Gaussian-linear case, the pdfs described in these assumptions can be written as follows:

$$\begin{aligned} p_{k-1}(\mathbf{x}_{k-1}) &= \mathcal{N}(\mathbf{x}_k\,;\hat{\mathbf{x}}_{k|k-1},\mathbf{P}_{k|k-1}) \\ f_{\mathbf{x}}(\mathbf{x}_k|\mathbf{x}_{k-1}) &= \mathcal{N}(\mathbf{x}_k\,;\mathbf{F}\mathbf{x}_{k-1},\mathbf{Q}) \\ f_{\mathbf{z}}(\mathbf{z}_k|\mathbf{x}_k) &= \mathcal{N}(\mathbf{z}_k\,;\mathbf{H}\mathbf{x}_k,\mathbf{R}). \end{aligned}$$

Assumption S1 does not really provide anything new either, but it is included in order to distinguish the single-target assumptions from the multi-target assumptions that we will study in Chapter 8.

Assumptions S4, S6 and S7 are, however, not present in standard filtering problems. Notice in particular that we can have several (or zero) clutter measurements, but we can have only one or zero target measurements. From a physical point of view this may appear questionable. In a radar image of a large ship, several cells are likely to contain returns from the ship. There exist tracking methods, so-called extended object tracking, which attempt to process such a collection of target-originating measurements collectively. However, the standard solution is to cluster the detections together to a single blob, whose centroid is then given as a measurement to the tracking method. The assumption that at most one measurement comes from the target is very convenient from a computational perspective: Instead of having to consider which of several combinations of m_k measurements are from the target, the data association problem boils down to determining which of the m_k measurements, if any, comes from the target.

7.4.2 Thinking in terms of mixtures

Based on this discussion, we introduce an association variable a_k which accounts for which, if any, measurement comes from the target. We represent the case of misdetection by zero, and a_k can then attain integer values ranging from 0 to m_k. In accordance with the possible realizations of the association variable, the outcome space can be partitioned into the following m_k+1 mutually exclusive and exhaustive events at time step k:

$$\begin{aligned} &a_k = 0 && \text{No measurement originates from the target} \\ &a_k = 1 && \text{Measurement 1 originates from the target} \\ &\vdots \\ &a_k = m_k && \text{Measurement } m_k \text{ originates from the target.} \end{aligned} \tag{7.3}$$

We known from our assumptions that one and only one of these events must take place at time step k. The problem of data association can be understood as making inference about a_k. We may take both an MAP or an MMSE approach in this endeavour, and the two estimation philosophies lead to different tracking methods. The PDAF is based on the MMSE approach. In any case, to do probabilistic inference concerning a_k we have to calculate probabilities for the different values that a_k can attain.

According to the total probability theorem the posterior density of the target state can be written as

$$p_k(\mathbf{x}_k) = \sum_{i=0}^{m_k} p(\mathbf{x}_k\,|\,a_k = i, Z_{1:k}) \Pr\{a_k = i\,|\,Z_{1:k}\}. \tag{7.4}$$

For notational convenience, we will in the sequel confuse the random variable a_k with its realization, so that (7.4) is written

$$p_k(\mathbf{x}_k) = \sum_{a_k} p(\mathbf{x}_k\,|\,a_k, Z_{1:k}) \Pr\{a_k\,|\,Z_{1:k}\}. \tag{7.5}$$

We recognize this as a mixture of the same form as those we encountered in Section 6.3. The goal of the PDAF is to reduce this mixture to a single Gaussian that approximates $p_k(\mathbf{x}_k)$ by means of mixture reduction. To achieve this we must specify the event-conditional pdfs $p(\mathbf{x}_k \,|\, a_k, Z_{1:k})$ and the event probabilities $\Pr\{a_k \,|\, Z_{1:k}\}$.

7.4.3 The event-conditional posterior densities

The first of these tasks is fairly straightforward. The prediction from time step $k-1$ to time step k follows a standard Chapman-Kolmogorov equation

$$p_{k|k-1}(\mathbf{x}_k) = \int f_{\mathbf{x}}(\mathbf{x}_k|\mathbf{x}_{k-1}) p_k(\mathbf{x}_{k-1}) \mathrm{d}\mathbf{x}_{k-1}. \tag{7.6}$$

The exact details of (7.6) will of course depend on the prior $p_k(\mathbf{x}_{k-1})$ and the process model $f_{\mathbf{x}}(\mathbf{x}_k|\mathbf{x}_{k-1})$. Only when these are Gaussian can we expect a closed form solution. For now, we will just take for granted that $p_{k|k-1}(\mathbf{x}_k)$ is somehow provided. The event-conditional posteriors are given by

$$p(\mathbf{x}_k \,|\, a_k, Z_{1:k}) \propto \begin{cases} p_{k|k-1}(\mathbf{x}_k) & \text{if } \; a_k = 0 \\ f_{\mathbf{z}}(\mathbf{z}_k^{a_k} \,|\, \mathbf{x}_k) p_{k|k-1}(\mathbf{x}_k) & \text{if } \; a_k > 0. \end{cases} \tag{7.7}$$

The notation $\mathbf{z}_k^{a_k}$ signifies that measurement number a_k out of the m_k measurements is used.

These expressions are fairly straightforward to understand, but let us nevertheless spend some time reflecting over them. In the case when $a_k = 0$ we have no measurement. In this case, the measurement set Z_k provides no information about $\mathbf{x}_k$, and the posterior density conditional on this event is therefore identical to the predicted pdf.[1] On the other hand, if we assume that one measurement came from the target (i.e., that $a_k > 0$) and that we known which measurement it was, then the state estimation boils down to conventional usage of Bayes rule, which is what the second line in (7.7) expresses. Of course, under Gaussian-linear assumptions this boils down to a Kalman filter.

7.4.4 The event probabilities

The probabilities $\Pr\{a_k \,|\, Z_{1:k}\}$ require some more work. Let us first define the likelihood shortcut

$$l^{a_k} = \int f_{\mathbf{z}}(\mathbf{z}_k^{a_k} \,|\, \mathbf{x}_k) p_{k|k-1}(\mathbf{x}_k) \, \mathrm{d}\mathbf{x}_k. \tag{7.8}$$

We then use this quantity to express the posterior association probabilities as follows:

Theorem 7.4.1 — Association probabilities for single-target tracking. Under the assumptions S1-S7 the posterior association probabilities are given by

$$\Pr\{a_k \,|\, Z_{1:k}\} \propto \begin{cases} (1 - P_{\mathrm{D}}) m_k \dfrac{\mu(m_k)}{\mu(m_k - 1)} & \text{if } \; a_k = 0 \\ \dfrac{P_{\mathrm{D}}}{c(\mathbf{z}_k^{a_k})} l^{a_k} & \text{if } \; a_k > 0. \end{cases}$$

Proof. The main conceptual difficulties in target tracking arise from the fact that the measurement set Z_k has random cardinality. To deal with this, we treat its cardinality m_k as a separate piece of

[1] It can be argued that the absence of target-originating measurements actually does carry information about $\mathbf{x}_k$. The core of this argument is that one possible reason why a measurement was not found could be that the covariance was too small, so that the measurement ended up outside of the validation gate. See [66]. In this book we will, however, generally assume that the validation gates are so large that this is not an issue.

information. According to Bayes' rule we then have

$$\Pr\{a_k \,|\, Z_{1:k}\} = \Pr\{a_k \,|\, Z_k, m_k, Z_{1:k-1}\} \propto p(Z_k|m_k, a_k, Z_{1:k-1}) \Pr\{a_k \,|\, m_k\}. \tag{7.9}$$

Notice that the conditioning on $Z_{1:k-1}$ has been removed in the last term. This can be done because when the only available information about Z_k is m_k, we have no knowledge of how well any individual measurement in Z_k will fit together with the previously observed measurements. Nevertheless, the cardinality m_k does carry important information: When more measurements are present we will inevitably be less confident in our belief that any given measurement comes from the target. The remaining part of the proof is about obtaining expressions for the likelihiood $p(Z_k|m_k, a_k, Z_{1:k-1})$ and the prior probability $\Pr\{a_k \,|\, m_k\}$ under the two distinct cases of $a_k = 0$ and $a_k > 0$.

Likelihood for $a_k = 0$: In this case the pdfs of all measurements are given by the clutter model of assumption S7, and the likelihood becomes just a product of these densities:

$$\begin{aligned} p(Z_k \,|\, m_k, a_k = 0, Z_{1:k-1}) &= \int p(Z_k \,|\, m_k, a_k = 0, \mathbf{x}_k, Z_{1:k-1}) p(\mathbf{x}_k \,|\, Z_{1:k-1}) \mathrm{d}\mathbf{x}_k \\ &= \int \prod_{j=1}^{m_k} c(\mathbf{z}_k^j)\, p_{k|k-1}(\mathbf{x}_k)\, \mathrm{d}\mathbf{x}_k = \prod_{j=1}^{m_k} c(\mathbf{z}_k^j). \end{aligned} \tag{7.10}$$

Likelihood for $a_k > 0$: In this case, one of the measurements will be given by the measurement model of assumption S5, while the other ones are given by a product of clutter pdfs. Having established (7.10), we also write the likelihood in this case so that the same product is present:

$$\begin{aligned} p(Z_k \,|\, m_k, a_k, Z_{1:k-1}) &= \int p(Z_k \,|\, m_k, a_k, \mathbf{x}_k, Z_{1:k-1}) p(\mathbf{x}_k \,|\, Z_{1:k-1}) \mathrm{d}\mathbf{x}_k \\ &= \int \prod_{j \neq a_k} c(\mathbf{z}_k^j) f_{\mathbf{z}}(\mathbf{z}_k^{a_k} \,|\, \mathbf{x}_k)\, p_{k|k-1}(\mathbf{x}_k)\, \mathrm{d}\mathbf{x}_k = \frac{l^{a_k}}{c(\mathbf{z}^{a_k})} \prod_{j=1}^{m_k} c(\mathbf{z}_k^j). \end{aligned} \tag{7.11}$$

Prior probability for $a_k = 0$: In what follows we will make a clear distinction between the random variable m_k and its realization, which we denote $\underline{\mathrm{m}}$. Instead of simply writing $\Pr\{a_k \,|\, m_k\}$ we here write $\Pr\{a_k = 0 \,|\, m_k = \underline{\mathrm{m}}\}$, and we find this probability as follows

$$\begin{aligned} \Pr\{a_k = 0 \,|\, m_k = \underline{\mathrm{m}}\} &= \Pr\{\varphi_k = \underline{\mathrm{m}} \,|\, m_k = \underline{\mathrm{m}}\} \\ &= \frac{\Pr\{m_k = \underline{\mathrm{m}} | \varphi_k = \underline{\mathrm{m}}\} \Pr\{\varphi_k = \underline{\mathrm{m}}\}}{\Pr\{m_k = \underline{\mathrm{m}}\}} \\ &= \frac{(1 - P_{\mathrm{D}}) \mu(\underline{\mathrm{m}})}{\Pr\{m_k = \underline{\mathrm{m}}\}}. \end{aligned} \tag{7.12}$$

The first equality in (7.12) follows from the fact that if zero measurements comes from the target, then the number of clutter measurements must be equal to the total number of measurements. The second equality is just Bayes' rule. In the third equality we insert the clutter cardinality distribution, and notice that if all the measurements are clutter measurements, then this implies a misdetection, which has probability $1 - P_{\mathrm{D}}$.

Prior probability for $a_k > 0$: This case proceeds along very similar lines to the previous case:

$$\begin{aligned} \Pr\{a_k = i \,|\, m_k = \underline{\mathrm{m}}\} &= \Pr\{a_k = i \,|\, \varphi_k = \underline{\mathrm{m}} - 1, m_k = \underline{\mathrm{m}}\} \Pr\{\varphi_k = \underline{\mathrm{m}} - 1 \,|\, m_k = \underline{\mathrm{m}}\} \\ &= \frac{1}{\underline{\mathrm{m}}} \Pr\{\varphi_k = \underline{\mathrm{m}} - 1 \,|\, m_k = \underline{\mathrm{m}}\} \\ &= \frac{\Pr\{m_k = \underline{\mathrm{m}} \,|\, \varphi = \underline{\mathrm{m}} - 1\} \Pr\{\varphi = \underline{\mathrm{m}} - 1\}}{m \Pr\{m_k = \underline{\mathrm{m}}\}} \\ &= \frac{P_{\mathrm{D}} \mu(\underline{\mathrm{m}} - 1)}{\underline{\mathrm{m}} \Pr\{m_k = \underline{\mathrm{m}}\}}. \end{aligned} \tag{7.13}$$

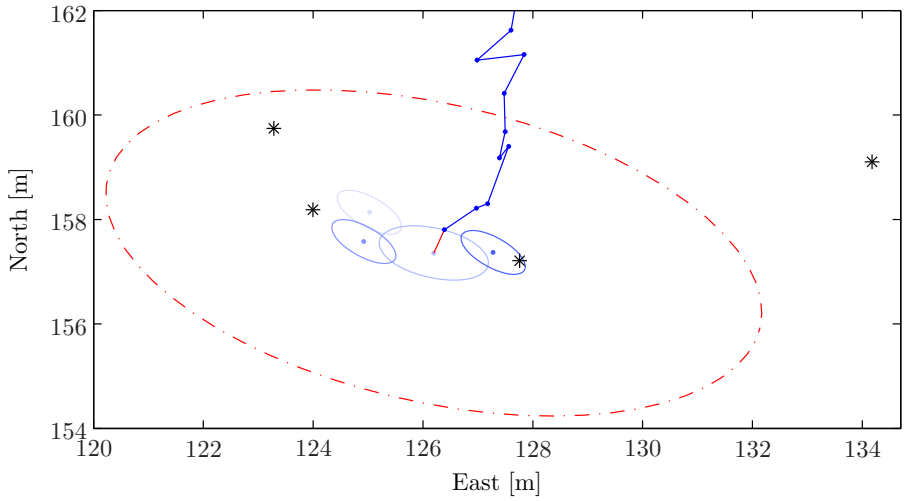


Figure 7.1: Validation gate for a PDAF tracker with three measurements inside of the gate and one measurement outside it. Hypothesis-conditional state estimates and their covariances are displayed with blue dots and ellipses, where the darkness of the color reflects the association probability.

We can summarize the results in the following two expressions, which together yield the posterior probabilities of the theorem:

$$p(Z_k\,|a_k,m_k,Z_{1:k-1}) \propto \begin{cases} 1 & \text{if } a_k = 0 \\ l^{a_k}/c(\mathbf{z}_k^{a_k}) & \text{if } a_k > 0 \end{cases} \quad \text{and} \quad \Pr\{a_k\,|\,m_k\} \propto \begin{cases} (1-P_{\mathrm{D}})\mu(m_k) & \text{if } a_k = 0 \\ \dfrac{P_{\mathrm{D}}\mu(m_k-1)}{m_k} & \text{if } a_k > 0. \end{cases}$$

■

After this proof of what may be considered the main theorem of this chapter, let us recapitulate what we have so far. Our goal was to find the constituents of the mixture (7.5). We found the hypothesis-conditional densities $p(\mathbf{x}_k\,|\,a_k,Z_{1:k})$ in (7.7), and we found the hypothesis probabilities $\Pr\{a_k\,|\,Z_{1:k}\}$ in Theorem 7.4.1. We also refer to these by the notation $\beta_k^{a_k} \triangleq \Pr\{a_k\,|\,Z_{1:k}\}$.

7.4.5 Gaussian-linear assumptions and the validation gate

To make this mixture into something more tangible we need to invoke additional assumptions. We need to specify forms for four continous pdfs, namely the prior distribution, the process model, the measurement model and the clutter pdf. Let these be given as follows:

$$\begin{aligned} p_{k-1}(\mathbf{x}_{k-1}) &= \mathcal{N}(\mathbf{x}_{k-1}\,;\hat{\mathbf{x}}_{k-1},\mathbf{P}_{k-1}) \\ f_{\mathbf{x}}(\mathbf{x}_k\,|\,\mathbf{x}_{k-1}) &= \mathcal{N}(\mathbf{x}_k\,;\mathbf{F}\mathbf{x}_{k-1},Q) \\ f_{\mathbf{z}}(\mathbf{z}_k\,;\mathbf{x}_k) &= \mathcal{N}(\mathbf{z}_k\,|\,\mathbf{H}\mathbf{x}_k,\mathbf{R}) \\ c(\mathbf{z}_k) &= \frac{1}{V_k}. \end{aligned} \tag{7.14}$$

According to the product identity, the first two expressions in (7.14) together imply that the predicted density becomes

$$p_{k|k-1}(\mathbf{x}_k) = \mathcal{N}(\mathbf{x}_k\,;\hat{\mathbf{x}}_{k|k-1},\mathbf{P}_{k|k-1}) \tag{7.15}$$

where, just as for a regular Kalman filter, we have the prediction and its covariance given by

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}\hat{\mathbf{x}}_{k-1} \quad \text{and } \mathbf{P}_{k|k-1} = \mathbf{F}\mathbf{P}_{k-1}\mathbf{F}^{\mathsf{T}} + \mathbf{Q}. \tag{7.16}$$

For convenience, let us also define the predicted measurement and its covariance given by

$$\hat{\mathbf{z}}_{k|k-1} = \mathbf{H}\hat{\mathbf{x}}_{k|k-1} \text{ and } \mathbf{S}_k = \mathbf{H}\mathbf{P}_{k|k-1}\mathbf{H}^\mathsf{T} + \mathbf{R}. \tag{7.17}$$

It remains to define the quantity V_k. This is the volume of the region of the measurement space in which measurements are considered. While this region could be comprised by the entire surveillance region, it is in practice always restricted to a *validation gate* in the vicinity of the predicted measurement $\hat{\mathbf{z}}_{k|k-1}$. The validation gate is typically constructed as an ellipse or ellipsoid, whose shape is given by the predicted covariance $\mathbf{S}_k$ and a scaling parameter g:

$$\mathscr{G} = \left\{ \mathbf{z} \text{ such that } (\mathbf{z} - \hat{\mathbf{z}}_{k|k-1})^\mathsf{T} \mathbf{S}_k^{-1} (\mathbf{z} - \hat{\mathbf{z}}_{k|k-1}) < g^2 \right\}.$$

We see that g is the number of standard deviations that we are willing to consider. The smaller the gate, the higher the risk that a true measurement may fall outside of the gate. If the dimension of $\mathbf{z}$ is ≤ 3, and $g \geq 3$, then this probability is negligible, assuming our models of the process and measurement noise are valid. Otherwise, we can introduce a gate probability P_G which quantifies the probability that a true measurement is inside the gate, given that it exists. We must then replace P_D with $P_\mathrm{D}P_\mathrm{G}$ in all our calculations, and we also divide the likelihood $f_\mathbf{z}(\cdot)$ by P_G in the weight calculation to compensate for the truncation. We will not pursue the issue of gating probabilities any further here, and the reader is referred to [3] pp. 95-96 and 134-135.

The volume of the gate may or may not be needed depending on our choice of clutter cardinality model (diffuse or Poisson). In any case, it is given by

$$V_k = c_{n_z} |g^2 \mathbf{S}_k|^{1/2}, \; c_1 = 2, \; c_2 = \pi, \; c_3 = 4\pi/3, \ldots$$

where c_{n_z} is the dimension of the measurement space (e.g., 2 for range-bearing measurements). Again, the reader is referred to [3] pp. 95-96 for further details.

Under the Gaussian-linear assumptions we can express the event-conditional posterior densities as

$$p_k(\mathbf{x}_k \,|\, a_k, Z_{1:k}) = \mathscr{N}(\mathbf{x}_k \,;\, \hat{\mathbf{x}}_k^{a_k}, \mathbf{P}_k^{a_k}) \tag{7.18}$$

where the expectations $\hat{\mathbf{x}}_k^{a_k}$ and covariances $\mathbf{P}_k^{a_k}$ are given by standard Kalman filter updates. For completeness we will go through the details. The Kalman gain is

$$\mathbf{W} = \mathbf{P}_{k|k-1}\mathbf{H}^\mathsf{T}(\mathbf{H}\mathbf{P}_{k|k-1}\mathbf{H}^\mathsf{T} + \mathbf{R})^{-1}. \tag{7.19}$$

Notice that the same Kalman gain is used for all the measurements. In practice, it may be desirable to let the measurement noise matrix $\mathbf{R}$ depend on the measurements, and in that case a different Kalman gain must be used for each measurement. Let us also introduce the event-conditional innovations

$$\boldsymbol{\nu}_k^{a_k} = \mathbf{z}_k^{a_k} - \mathbf{H}\hat{\mathbf{x}}_{k|k-1}.$$

Then the event-conditional state estimate $\hat{\mathbf{x}}_k^{a_k}$ is given by

$$\hat{\mathbf{x}}_k^{a_k} = \begin{cases} \hat{\mathbf{x}}_{k|k-1} & \text{if } a_k = 0 \\ \hat{\mathbf{x}}_{k|k-1} + \mathbf{W}_k \boldsymbol{\nu}_k^{a_k} & \text{if } a_k > 0 \end{cases} \tag{7.20}$$

while its covariance $\mathbf{P}_k^{a_k}$ is given by

$$\mathbf{P}_k^{a_k} = \begin{cases} \mathbf{P}_{k|k-1} & \text{if } a_k = 0 \\ (\mathbf{I} - \mathbf{W}_k\mathbf{H})\mathbf{P}_{k|k-1} & \text{if } a_k > 0. \end{cases} \tag{7.21}$$

The exact expressions for the association probabilities depend on the choice of clutter cardinality model. We present the results for the diffuse and the Poisson models in the following two corollarys to Theorem 7.4.1.

Corollary 7.4.2 — Association probabilities for diffuse clutter model. If $\mu(\varphi)$ is constant, then the posterior association probabilities with Gaussian-linear assumptions are given by

$$P\{a_k \,|\, Z_{1:k}\} \propto \begin{cases} \dfrac{m_k(1-P_{\mathrm{D}})}{V_k} & \text{if } \; a_k = 0 \\ P_{\mathrm{D}} \mathcal{N}(\mathbf{z}_k^{a_k}; \hat{\mathbf{z}}_{k|k-1}, \mathbf{S}_k) & \text{if } \; a_k > 0. \end{cases}$$

Corollary 7.4.3 — Association probabilities for Poisson clutter model. If the number of clutter measurements in the validation gate is distributed according to $\mu(\varphi_k) = \exp(-\lambda V_k)\frac{(\lambda V_k)^{\varphi_k}}{\varphi_k!}$, then the posterior association probabilities with Gaussian-linear assumptions are given by

$$P\{a_k \,|\, Z_{1:k}\} \propto \begin{cases} \lambda(1-P_{\mathrm{D}}) & \text{if } \; a_k = 0 \\ P_{\mathrm{D}} \mathcal{N}(\mathbf{z}_k^{a_k}; \hat{\mathbf{z}}_{k|k-1}, \mathbf{S}_k) & \text{if } \; a_k > 0. \end{cases}$$

The steps needed to prove Corollary 7.4.2 from Theorem 7.4.1 are straigthforward: This includes nothing more than the product identity for Gaussian distributions, some moving around of the terms, and cancellations of the clutter cardinality terms since they are the same in all cases. The proof of Corollary 7.4.3 is slightly more complicated, and we leave it as an exercise for the reader.

The expressions in Corollaries 7.4.2 and 7.4.3 are the key formulas in the PDAF. Together with the filtering formulas (7.20) and (7.21) they define the Gaussian mixture that results after observing a measurement set Z_k. Before we delve into the last step of the PDAF algorithm, the mixture reduction, we shall discuss some pros and cons with the two versions of the PDAF that the diffuse and the Poisson clutter models lead to.

We see from the Corollaries that, depending on the clutter model chosen, we must either specify V_k or λ. It can be difficult to obtain a reliable estimate of λ, and this is a strong argument in favor of the diffuse model. It can also be seen from the formulas in the Corollaries that in the diffuse PDAF the quantity m_k/V_k takes the role as an estimate of λ. Reliable estimates of λ can be obtained if the underlying detector actually achieves its nominal false alarm rate. Unfortunately, the actual false alarm rate may often deviate from the nominal false alarm rate by several decades. It is also possible to estimate λ by counting the number of measurements. Such estimators must rely on some continuity of λ either in time or space, which again may be difficult to ensure. Clutter estimation remains a field of active research. One reason why the Poisson version of the PDAF may be preferred is that when we later are going to generalize the PDAF to multi-target tracking, calculating the volume of the joint validation region for several targets becomes more tricky. The vast majority of research papers on multi-target tracking use the Poisson clutter model.

7.4.6 Mixture reduction: The last step of the PDAF cycle

Since we have a mixture after the measurement update, the Gaussian assumption on the prior is obviously violated at the beginning of the next estimation cycle. We would replace the single Gaussian with a mixture, which then would be extended to a larger mixture during the next measurement update, and so on. This is sometimes referred to as the track-split-filter or the optimal Bayesian approach (OBA). For such an approach to work, we would soon have to prune away mixture components with low probabilities in order to maintain computational feasibility. We will look more closely at this approach in Chapter 9, which covers multiple hypothesis tracking.

The PDAF is instead based on the concept of mixture reduction from Section 6.3, which gives much bigger computational savings. As the last step of the estimation cycle, it merges all the components of the mixture into a single Gaussian. As in Section 6.3 this is based on moment-matching: The new Gaussian should have the same expectation and covariance as the original

mixture. This is known as moment matching. The expectation of the mixture is

$$\hat{\mathbf{x}}_k = \beta_k^0 \hat{\mathbf{x}}_{k|k-1} + \sum_{a_k>0} \beta_k^{a_k} \hat{\mathbf{x}}_k^{a_k} \tag{7.22}$$

If we define $\boldsymbol{\nu}_k = \sum_{a_k>0} \beta_k^{a_k} \boldsymbol{\nu}_k^{a_k}$ then this can also be written as

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_{k|k-1} + \mathbf{W}_k \boldsymbol{\nu}_k. \tag{7.23}$$

Matching of the second-order moments, i.e. covariances, leads to two equivalent expressions for the covariance of the mixture

$$\mathbf{P}_k = \beta_k^0 \mathbf{P}_{k|k-1} + \sum_{a_k>0} \beta_k^{a_k} \left(\mathbf{P}_{k|k-1} - \mathbf{W}_k \mathbf{S}_k \mathbf{W}_k\right) + \tilde{\mathbf{P}}_k \tag{7.24}$$

$$= \mathbf{P}_{k|k-1} - (1-\beta_k^0)\mathbf{W}_k \mathbf{S}_k \mathbf{W}_k + \tilde{\mathbf{P}}_k \tag{7.25}$$

where

$$\tilde{\mathbf{P}}_k = \mathbf{W}_k \left[\sum_{a_k>0} \beta_k^{a_k} \boldsymbol{\nu}_k^{a_k} (\boldsymbol{\nu}_k^{a_k})^\mathsf{T} - \boldsymbol{\nu}_k \boldsymbol{\nu}_k^\mathsf{T} \right] \mathbf{W}_k^\mathsf{T} \tag{7.26}$$

is the "spread of the innovations". The derivations of (7.24) - (7.26) build directly on the mixture-reduction formulas from Theorem 6.3.1. See also [3] p. 155 for further details. To summarize, after moment-matching the PDAF approximates the posterior as

$$p(\mathbf{x}_k \,|\, \mathbf{Z}_{1:k}) \approx \mathcal{N}(\mathbf{x}_k; \hat{\mathbf{x}}_k, \mathbf{P}_k). \tag{7.27}$$

This can now be predicted to the next time step using standard KF techniques. This concludes our development of the PDAF.

7.5 Implementing the PDAF

Assumption S1 means that the PDAF is a single-target tracking method, as opposed to a multi-target tracking method. Based on this, one could conclude that the PDAF cannot be used in scenarios where multiple targets are present. But such a dismissal would be imprudent. It is perfectly possible to run several PDAFs in parallell on different targets. If their validation gates do not share measurements this is hardly any issue at all. If the validation gates intersect, using a PDAF instead of a bona-fide multi-target tracking method will lead to a performance loss, but it can still be done. Some additional bookeeping will be advisable to prevent or mitigate the consequences of phenomena such as track coalescence (see Section 8.4).

However, designing a complete tracking system is significantly more complex than the PDAF itself, even if it is designed purely in a single-target fashion. Most importantly, tracks must be initialized and terminated according to given criteria. It is in this context important to keep track of the tracks (pun intended). In the real world, the purely linear models that we have used in (7.14) will often have to replaced with nonlinear models, and this can lead to both moderate and more severe complications. Finally, to do all of this in a coherent and systematic way it is important to include tools for visualization.

7.5.1 Visualization

Target tracking is about making sense of what a sensor observes. In, e.g., conventional ARPA systems, the entire purpose of the tracking system is to visualize what is going on to the human

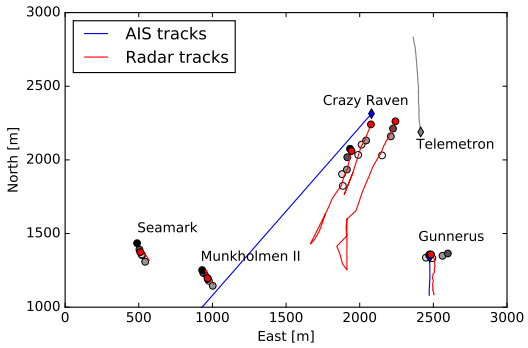
seafarer. But visualization is also extremely important in the process of designing and debugging a tracking system.

Tracks, state estimates and measurements are typical things that should be visualized. Furthermore, at least for a tracking system based on the PDAF, it is desirable to visualize covariance ellipses and validation gates. The latter is crucial in order to know which measurements that a given track is considering. In some cases it may also be desirable to visualize things like sensor resolution or covariance ellipses of the measurement noise. The sensor FOV and land contours may also be useful components in this world picture. If ground truth is available (either because it is a simulation or because the target has a GPS transmitter) then it would probably be useful to include this as well.

We see that quite an amount of information can easily become part of this visualization. Furthermore, as we will discuss in greater detail in Section 7.5.3, a tracking system will typically distinguish between preliminary tracks and confirmed tracks. These will typically be given different colors.

Since tracking is about estimating the state of a dynamical system, it is often desirable to present the visualization as a movie. However, for presentation in a paper or report this is not possible. One option can then be to present the measurements and state estimates for all time steps simultaneously, with a color scale that indicates time. If there is very much clutter present, such a picture may not be very informative, and it may be better to just present a handful of snapshots, e.g. to illustrate a scenario where track-loss happens so that it can be studied in further detail.

In order to avoid information overload, it is generally desirable to visualize the tracking results separate from the underlying sensor images. Nevertheless, comparison with the contents of the raw data can be very valuable if the raw data are available. This can be useful to identify weaknesses of the methods for detection and measurement extraction.



(a) Radar tracks from PDAF.

(b) The same scenario seen from drone video.

Figure 7.2: Example of display for a PDAF-based radar tracker used in the Autosea project on collision avoidance for autonomous ships. Measurements used in the tracking systems are given shades of gray based on their time stamps. The blue AIS tracks were not used in the tracking system. The radar was mounted onboard Telemetron, which also was equipped with a very precise navigation system.

7.5.2 Dealing with nonlinearities

Many nonlinearities can be dealt with by KF formulas such as those in (7.19), (7.20) and (7.21) with corresponding EKF or UKF formulas. In such cases the generalization of the PDAF to deal with nonlinear kinematics or measurement models is straightforward.

The most common nonlinearity in target tracking is the one caused by measurements received in polar coordinates. As discussed in Section 4.7, the standard way of addressing this is by converting

the measurements to Cartesian coordinates, and thus avoid the need for linearization. This raises another tricky issue, though. The converted measurement noise covariance matrix $\mathbf{R}$ will now depend on the measurement. In some cases this may be the way to go, but most often it will lead to undesirable complications, since the Kalman gain in (7.19) then will have to be calculated for every validated measurement. However, in most practical cases the variations of $\mathbf{R}$ within a the validation gate will be so small that a common matrix based on what a conversion of $\hat{\mathbf{z}}_{h|h-1}$ would yield can be used for all the measurements within the gate.

Regarding design of $\mathbf{R}$, it may also in some situations be tempting to let $\mathbf{R}$ depend on uncertainty due to target extent. A measurement that is generated by a bigger blob in the sensor image is likely to have a larger uncertainty than a measurement that is generated by a smaller blob. While it may be wise to include such uncertainties in the design of $\mathbf{R}$, the reader is strongly warned against allowing this design to depend on the measurement. The reason for this is that larger blobs in typical applications are more likely to come from the true target, while smaller blobs are more likely to be clutter, and such a relationship would therefore lead to an undesirable preference for clutter.

In contrast, it may be possible to improve the association accuracy of the PDAF by using the signal strength of the measurements to calculate the association probabilities β_k^i. The reader is referred to papers such as [63], [18] and [22] for further suggestions on this topic.

If more severe nonlinearities are present, particle filtering may be a promising alternative to KF-based techniques. Despite being intensely explored during the last couple of decades, the combination of particle filtering with single-target tracking remains an open research topic. Questions such as the following must be answered: Should one sample just the kinematic state or should one also sample the associations? The reader is referred to [33] for a possible approach.

7.5.3 Track management

The biggest hurdle in a tracking system is that tracks are generally not initialized *a priori*. The tracking system must itself make a decision whether a sequence of measurements are likely to originate from a new target, and if so establish a track on that target. The simplest recipe for this process is to count the number of times the validation gate contains a detection over some limited time interval. If we again and again receive a detection within the validation gate, this could be taken as evidence that a target is present, which repeated absense of detections is evidence that no target is present.

A track initialization method based on this reasoning is the 2/2&M/N logic [3]. First, a so-called tentative track is established whenever we receive two consecutive detections in close vicinity. The threshold for declaring a tentative track could for example be based on the maximal speed of the target combined with the measurement noise. After these two initial time steps, the tentative track is declared a preliminary track, which is propagated using the PDAF for a maximum of N time steps. After those N time steps a decision is made. If the track did successfully gate measurements in M of those time steps, it is declared a confirmed track. If it did not gate measurements in that many time steps, it is on the other hand terminated.

Confirmed tracks are maintained without being subject to the same test again. However, the target of a confirmed track may eventually disappear. This could be because the target moves outside of the surveillance region, because it dissolves, because it temporarily becomes invisible or simply because of track loss, i.e., the track fails to follow correct measurements and eventually has no idea where its target is. In such cases it is desirable to terminate the track, since invalid tracks in the tracking system can cause a lot of confusion. This is especially so because the validation gates of a lost track quickly will grow. This can turn a neat collection of single-target scenarios into a big and messy multi-target scenario. A tracking system should therefore also contain a track termination logic, for which the M/N logic again is the simplest candidate.

A vast variety of more advanced track management techniques exist. Broadly speaking, these techniques can be divided into methods based on detection theory and methods that utilize the concept of existence probability. In the former approach, the track initialization problem is viewed as a detection problem, with a corresponding track-level false alarm rate and detection probability. The goal is then to develop tests that achieve a higher detection probability for a given false alarm rate than he M/N logic, which also making the decision faster than the M/N logic. The most important such method is the sequential probability ratio test (SPRT) which makes its decision as soon as it is sufficiently confident. The SPRT is further discussed in Section 14.6.

7.6 Extension to target existence: IPDA

In this section we discuss the other approach to track management: The existence-based approach. In this approach the state vector of the PDAF is extended to also include a discrete valued existence state: Either the target exists or it does not exist. The goal is then not only to estimate the kinematic state of the target, but also to evaluate the posterior probability of target existence. This generalization of the PDAF is known as the integrated probabilistic data association (IPDA) and was originally proposed by Darko Musicki in 1994 [78].

The key difference between the detection theoretic approach and the existence-based approach is worth stressing: In the former approach a binary decision is sought. In the latter approach no decision is being made as part of the approach itself. We may be content with having the probability that a track corresponds to a real target quantified by the existence probability of the IPDA, or we may use this existence probability to make a binary decision. Since the 90's, the existence-based paradigm has to a large extent dominated research in target tracking. One of the central ideas, which largely has been developed by Ronald Mahler [71], is to formulate multi-target tracking with existence uncertainty in terms of random finite sets: The goal is to estimate how many elements there is in the set of targets, and also to estimate the pdfs of the individual elements. The IPDA is the first step towards this modern paradigm of multi-target tracking, and we will revisit this perspective in Chapter 13.

We recall that the PDAF was based on the assumptions S1-S7 on page 105. Among these, only the first, which says that one and only one target exists, needs to be fundamentally changed when we introduce existence uncertainty. For the IPDA, the corresponding assumption can be stated as follows

S1* At time step $k-1$ the set X_{k-1} of targets is equal to the singleton set $\{\mathbf{x}_{k-1}\}$ with probability r_k, and equal to empty set $\emptyset$ with probability $1-r_k$. If X_{k-1} is singleton, then X_k is singleton with probability P_S, and empty with probability $1-P_S$. If X_{k-1} is empty, then X_k is also empty.

In plain English, this means that the target may survive with probability P_S, and it may die with probability $1-P_S$. If it is already dead, then it cannot be reborn. However, even if its prior existence probability is rather low, support from good measurements may bring it back up.

The remaining assumptions S2-S7 are only affected in the sense that the statements of S2-S5 now are conditional on the hypothesis that a target exists. Notice that the amended assumption S1* also specifies a Markov model for target existence. In the standard formulation of the IPDA this Markov model allows a currently existing target to die, but it does not allow non-existence to suddenly turn into existence. Thus, the limit of the Markov chain will be 100% non-existence, unless sufficiently promising measurements counteract this development.

The workflow of the IPDA consists of 5 steps: *existence prediction, state prediction, association weights, state update and existence update*. Of these, the first and the last are not part of the conventional PDAF. Furthermore, the 3 middle steps are entirely identical to the corresponding steps in the PDAF. We therefore focus on the first and the last steps below.

7.6.1 The existence prediction

Both for the target existence and for the kinematic state, prediction is done by means of Chapman-Kolmogorov, a.k.a. the total probability theorem. If we denote the predicted existence probability by $r_{k|k-1}$, then we have

$$\begin{bmatrix} r_{k|k-1} \\ 1-r_{k|k-1} \end{bmatrix} = \begin{bmatrix} P_S & 0 \\ 1-P_S & 1 \end{bmatrix} \begin{bmatrix} r_{k-1} \\ 1-r_{k-1} \end{bmatrix} = \begin{bmatrix} P_S r_{k-1} \\ 1-P_S r_{k-1} \end{bmatrix} \tag{7.28}$$

where the 2×2 matrix in (7.28) is known as the Markov chain transition matrix.

7.6.2 The existence update

The main distinction of the IPDA is the formula for calculating the posterior existence probability r_k. We state the formula for updating r_k as a theorem below. The IPDA uses the same association events a_k as the PDAF. Again $a_k = 0$ means that no measurement is assigned to the target, while $a_k = i$ means that measurement number i is assigned to the target. However, the possibility of non-existence means that we have to be more careful concerning what we condition on when we define probabilities of these events. Let us define the following events

$$\begin{aligned} &E: \text{ The target exists.} \\ &D: \text{ The target is detected.} \end{aligned} \tag{7.29}$$

For the IPDA we must define the association probabilities as

$$\beta_k^{a_k} = P\{a_k \,|\, E, Z_{1:k}\} = \frac{P\{a_k, E \,|\, Z_{1:k}\}}{P\{E \,|\, Z_{1:k}\}} \tag{7.30}$$

as opposed to simply $P\{a_k \,|\, Z_{1:k}\}$ as we did for the PDAF. This is because the event $a_k = 0$, which is the same as $\neg D$, covers both misdetection of an existing target, and non-existence. This is illustrated in Figure 7.3. Nevertheless, as we also will see in the next theorem, the expressions for $\beta_k^{a_k}$ remain exactly the same as for the PDAF.

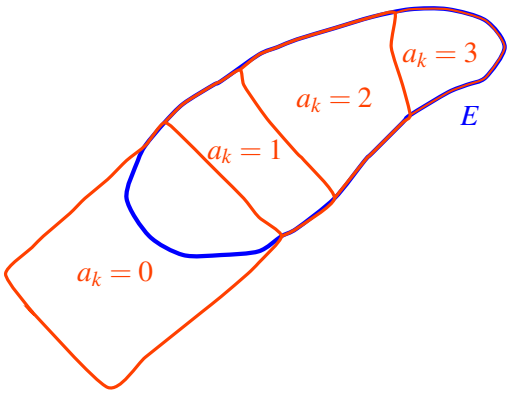


Figure 7.3: Illustration of events involved in the calculation of association probabilities for the IPDA for a case with 3 measurements within the validation gate.

The Poisson clutter model of Corollary 7.4.3 has always been used for the IPDA. Several suggestions for how to estimate the clutter rate λ have been proposed in the literature (See Section 7.8). In the seminal reference [78] the following estimator was suggested:

$$\hat{\lambda} = \begin{cases} 0 & \text{if } m_k = 0 \\ \frac{1}{V_k}\left(m_k - P_D r_{k|k-1}\right) & \text{if } m_k > 0. \end{cases} \tag{7.31}$$

This should be contrasted with the interpretation of m_k/V_k as a clutter estimate in the diffuse PDAF. The existence probability predicted by the IPDA allows a more refined clutter estimate. Having established the prediction of the existence probability, we also have to establish its measurement update.

Theorem 7.6.1 — Existence probability for the IPDA. Let the predicted existence probability be $r_{k|k-1}$, and otherwise we retain the same assumptions as in Corollary 7.4.3 (Poisson clutter, Gaussian-linear kinematics, etc.). The posterior existence probability is then

$$r_k = \frac{\mathscr{L}_k r_{k|k-1}}{1-(1-\mathscr{L}_k)r_{k|k-1}} r_{k|k-1} \tag{7.32}$$

where

$$\mathscr{L}_k = 1 - P_{\mathrm{D}} + \frac{P_{\mathrm{D}}}{\lambda}\sum_{a_k=1}^{m_k} l^{a_k} = 1 - P_{\mathrm{D}} + \frac{P_{\mathrm{D}}}{\lambda}\sum_{a_k=1}^{m_k} \mathcal{N}(\mathbf{z}_k^{a_k}; \hat{\mathbf{z}}_{k|k-1}, \mathbf{S}_k) \tag{7.33}$$

The association probabilities $\beta_k^{a_k}$, $a_k = 0, \dots, m_k$ are exactly the same as in Corollary 7.4.3.

Proof. Let us first calculate the probability of existence in the event that no measurement from the target is detected. It is given by

$$\begin{aligned}
\Pr\{E \,|\, a_k = 0\} =& \Pr\{E \,|\, \neg D, Z_{1:k}\} \\
=& \Pr\{E \,|\, \neg D, Z_{1:k-1}\} \\
=& \frac{\Pr\{\neg D \,|\, E\}\Pr\{E \,|\, Z_{1:k-1}\}}{\Pr\{\neg D \,|\, E\}\Pr\{E \,|\, Z_{1:k-1}\} + \Pr\{\neg D \,|\, \neg E\}\Pr\{\neg E \,|\, Z_{1:k-1}\}} \\
=& \frac{(1-P_{\mathrm{D}})r_{k|k-1}}{(1-P_{\mathrm{D}})r_{k|k-1} + 1\cdot(1-r_{k|k-1}} \\
=& \frac{r_{k|k-1}(1-P_{\mathrm{D}})}{1-r_{k|k-1}+r_{k|k-1}(1-P_{\mathrm{D}})} \\
=& \frac{r_{k|k-1}(1-P_{\mathrm{D}})}{1-P_{\mathrm{D}} r_{k|k-1}}
\end{aligned} \tag{7.34}$$

Let us then study the probabilities of the different realizations of the association variable a_k. We can find these by adapting the proof of Theorem 7.4.1 to the IPDA. The kinematic pdfs and the clutter description remain the same. The only thing that remains is then the occurences of the detection probability P_{D} in the prior hypothesis probabilities in (7.12) and (7.13). For the PDAF we had that $P\{D\} = P_{\mathrm{D}}$. For the IPDA, we get instead that

$$P\{D\} = P\{D,E\} = P\{D|E\}P\{E\} = P_{\mathrm{D}} r_{k|k-1}. \tag{7.35}$$

Thus, all occurences of P_{D} in Theorem 7.4.1 must be replaced by $P_{\mathrm{D}} r_{k|k-1}$. In the general framework of Theorem 7.4.1 we then have that

$$\Pr\{a_k \,|\, Z_{1:k}\} \propto \begin{cases} (1 - P_{\mathrm{D}} r_{k|k-1}) m_k \dfrac{\mu(m_k)}{\mu(m_k - 1)} & \text{if } a_k = 0 \\ P_{\mathrm{D}} r_{k|k-1} \dfrac{l^{a_k}}{c(\mathbf{z}_k^{a_k})} & \text{if } a_k > 0. \end{cases} \tag{7.36}$$

From (7.30) we have that the association probabilities $\beta_k^{a_k}$ are proportional to the joint probabilities $\Pr\{a_k, E \,|\, Z_{1:k}\}$ since the event E always must be conditioned upon. For the case that we have a

detection, i.e., $a_k > 0$, the probability of existence conditional on a_k is unity, yielding

$$\Pr\{a_k = i, E \,|\, Z_{1:k}\} = \Pr\{a_k = i \,|\, Z_{1:k}\}. \tag{7.37}$$

For the case that we $a_k = 0$, we must make some non-trivial manipulations to relate $\Pr\{a_k = 0, E \,|\, Z_{1:k}\}$ to $\Pr\{a_k = 0 \,|\, Z_{1:k}\}$. The details are as follows:

$$\begin{aligned}
\Pr\{a_k = 0, E \,|\, Z_{1:k}\} =& \Pr\{E \,|\, a_k = 0, Z_{1:k}\} \Pr\{a_k = 0 \,|\, Z_{1:k}\} \\
=& \Pr\{E \,|\, a_k = 0, Z_{1:k-1}\} \Pr\{a_k = 0 \,|\, Z_{1:k}\} \\
=& \frac{\Pr\{E, a_k = 0 \,|\, Z_{1:k-1}\}}{\Pr\{a_k = 0 \,|\, Z_{1:k-1}\}} \Pr\{a_k = 0 \,|\, Z_{1:k}\} \\
=& \frac{r_{k|k-1}(1 - P_{\mathrm{D}})}{1 - P_{\mathrm{D}} r_{k|k-1}} \Pr\{a_k = 0 \,|\, Z_{1:k}\}
\end{aligned} \tag{7.38}$$

The first and third equalities of (7.38) are both obtained from the definition of conditional probability. In between these, we utilize the fact that if we know that no detection has happened, then we also know that all measurements are clutter, and if we know this then the current measurement set does not provide any information about the target existence. The fourth and last equality follows from noting that, conditional on only the previous measurements, we can express both the numerator and the denominator in terms of $r_{k|k-1}$ and P_{D} only. It follows that the association probabilities are given by

$$\begin{aligned}
\beta_k^{a_k} &\propto \Pr\{a_k, E \,|\, Z_{1:k}\} \\
&\propto \begin{cases} \dfrac{r_{k|k-1}(1 - P_{\mathrm{D}})}{1 - P_{\mathrm{D}} r_{k|k-1}} \Pr\{a_k \,|\, Z_{1:k}\} & \text{if } a_k = 0 \\ \Pr\{a_k \,|\, Z_{1:k}\} & \text{if } a_k > 0 \end{cases} \\
&\propto \begin{cases} \dfrac{r_{k|k-1}(1 - P_{\mathrm{D}})}{1 - P_{\mathrm{D}} r_{k|k-1}} (1 - P_{\mathrm{D}} r_{k|k-1}) m_k \dfrac{\mu(m_k)}{\mu(m_k - 1)} & \text{if } a_k = 0 \\ P_{\mathrm{D}} r_{k|k-1} l^{a_k} / c(\mathbf{z}_k^{a_k}) & \text{if } a_k > 0 \end{cases} \\
&\propto \begin{cases} (1 - P_{\mathrm{D}}) m_k \dfrac{\mu(m_k)}{\mu(m_k - 1)} & \text{if } a_k = 0 \\ P_{\mathrm{D}} l^{a_k} / c(\mathbf{z}_k^{a_k}) & \text{if } a_k > 0 \end{cases}
\end{aligned} \tag{7.39}$$

The final expressions in (7.39) are identical to those in Theorem 7.4.1, and thus the IPDA uses exactly the same association probabilities as the PDAF.

It remains to derive the proposed formula for the posterior existence probability. From the total probability theorem we obtain

$$\begin{aligned}
\Pr\{E \,|\, Z_{1:k}\} =& \Pr\{E \,|\, a_k = 0, Z_{1:k}\} \Pr\{a_k = 0 \,|\, Z_{1:k}\} + \sum_{i=1}^{m_k} \Pr\{E \,|\, a_k = i, Z_{1:k}\} \Pr\{a_k = i \,|\, Z_{1:k}\} \\
=& \Pr\{E \,|\, a_k = 0, Z_{1:k}\} \Pr\{a_k = 0 \,|\, Z_{1:k}\} + \sum_{i=1}^{m_k} \Pr\{a_k = i \,|\, Z_{1:k}\}
\end{aligned} \tag{7.40}$$

We will then insert the details into this expression from (7.34) and (7.36). Before we do that, let us simplify matters by assuming that the clutter is a Poisson process with constant intensity. That is, we assume that $c(\mathbf{z}_k^i) = 1/V$ for all measurements $\mathbf{z}_k^i$ and that $\mu(m_k) = \exp(-\lambda V)(\lambda V)^{m_k}/m_k!$. Then we obtain the following simplification:

$$\frac{\mu(m_k) m}{\mu(m_k - 1) V} = \lambda. \tag{7.41}$$

Inserting the details into (7.40) then yields

$$\begin{aligned}\Pr\{E \mid Z_{1:k}\} &= \frac{\dfrac{r_{k|k-1}(1-P_{\mathrm{D}})}{1-P_{\mathrm{D}}r_{k|k-1}}(1-P_{\mathrm{D}}r_{k|k-1})\lambda + P_{\mathrm{D}}r_{k|k-1}\sum_{i=1}^{m_k} l^{a_k}}{(1-P_{\mathrm{D}}r_{k|k-1})\lambda + P_{\mathrm{D}}r_{k|k-1}\sum_{i=1}^{m_k} l^{a_k}} \\ &= \frac{(1-P_{\mathrm{D}}) + \dfrac{P_{\mathrm{D}}}{\lambda}\sum_{i=1}^{m_k} l^{a_k}}{1-P_{\mathrm{D}}r_{k|k-1} + r_{k|k-1}\dfrac{P_{\mathrm{D}}}{\lambda}\sum_{i=1}^{m_k} l^{a_k}} r_{k|k-1}\end{aligned} \tag{7.42}$$

We can now recognize the expression for $\mathscr{L}$ in the numerator. Manipulating the denominator so that we also can find $\mathscr{L}$ is a small exercise in algebra, and concludes the proof. ■

The main reason for using an IPDA instead of a plain-vanilla PDAF is that it can make track confirmation significantly more efficient. To quantify this, we need some core concepts from detection theory, because track initialization is fundamentally a detection problem. We want to make a decision whether or not we believe that a target is present based on the data and statistical models of the data. Tuning parameters of this decision procedure will affect its track probability of false alarm (TPFA) and its track probability of detection (TPD).[2] We want TPD to be high while TPFA should be low. Unfortunately, increasing TPD will generally tend to cause a corresponding increase in TPFA, while decreasing TPFA will cause a corresponding decrease in TPD. In many cases, a general detector will come with a free tuning parameter that can be adjusted to decide how this trade-off should be balanced. By plotting all possible pairs of TPFA and TPD in the TPFA-TPD plane, a curve known as the Receiving Operating Characteristic (ROC) results. The ROC curve is monotonously increasing, typically starting in $(0,0)$ and ending in $(1,1)$, reflecting the impossibility of a perfect detector.

For the IPDA, this free tuning parameter will be the threshold that the existence probability r_k must exceed for a track to be confirmed. For the M/N rule, it will not be possible to trace out a continuous ROC curve due to the discrete nature of the tuning options. However, it is obvious that a low value of M relative to N will lead to a more trigger-happy track detector (i.e., high TPFA and high TPD) while a high value of M relative to N will lead to a more conservative track detector (i.e., low TPFA and low TPD). It is not possible to predict these performance indicators analytically for something as complex as track initialization, but we can use empirical results from simulations to investigate different initialization procedures.

The track confirmation problem has a special sequential nature that makes it differ from many other detection problems. We would like to establish a track on the target as fast as possible. This can obviously be achieved by setting the threshold on the IPDA existence probability very low, but again the result will be that we also confirm many false tracks. This trade-off can also be analyzed by means of simulations.

The two plots in Figure 7.4 provide this kind of analysis for the M/N logic and the IPDA. On the left-hand side we see a logarithmic plot displaying one minus the probability that a target is detected as a function of the probability that a false track is established. The vertical axis is flipped, so that higher values means higher probability of detection. The main take-home message is that M/N can achieve the same detection probability as the IPDA, but for this to happen we must accept a 100-fold increase of the false track probability. On the right-hand side we see the detection time, again as a function of the false track probability. In these simulations, the IPDA needed about 5 to 6 scans on average to confirm a target when the overall false track probability was 0.001. For the M/N rule to achieve the same false track probability its tuning parameters would have to set quite

[2] We use the terms TPFA and TPD to distinguish these *track-level* detection performance indicators from the *sensor-level* detection performance indicators P_{FA} and P_{D} that were introduced on page 103.

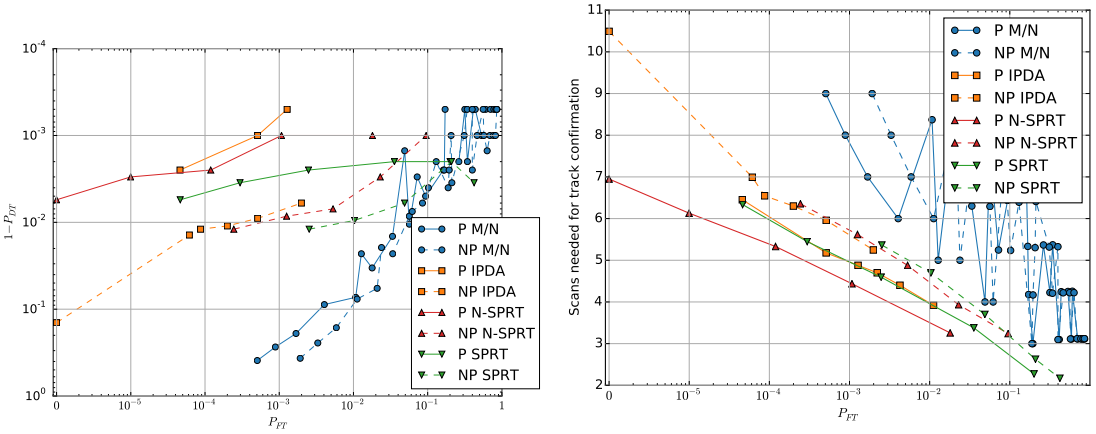


Figure 7.4: Detection performance of different versions of IPDA, SPRT (not a topic here) and M/N. Courtesy of [119].

conservatively (i.e., high values of both M and N), with the result that 50% more time steps (e.g., 9) were needed to make a confirmation on average.

The existence probability of the IPDA can also be used to control track maintainance and termination after a track has been confirmed. In this phase, the merits of the IPDA are less obvious. The Markov chain for the existence state will quickly converge towards non-existence if detections for any reason do not arrive during a prolonged period. This could easily happen, due to temporary occlusions, SNR fluctuations, etc., but the simple Markov model that we used for the IPDA does not contain any model of this. One solution that was suggested in the original paper [78] is to include a visibility state in addition to the existence state in the Markov model.

Another tuning parameter that has impact on how the IPDA behaves is the initial existence probability r_0. A systematic approach to determine this quantity can be derived from knowledge about how many undetected targets one expects there to be in the surveillance scene. We defer this discussion to Chapter 13 because it is based on the concept of random finite sets.

7.7 Extension to maneuvering targets: IMM-PDAF

We saw in Chapter 6 that it could be useful to employ two or more kinematic models in parallel to get good state estimates for maneuvering targets. Having moved from pure state estimation in that chapter to target tracking in this chapter, the question naturally arises whether multiple model methods, and the IMM method in particular, can be integrated in the PDAF framework. We shall see that this is indeed fairly straightforward.

This can be addressed from two viewpoints which eventually are equivalent. On the one hand, we can replace the conventional kinematic state vector in the PDAF with a hybrid state vector that both contains the kinematic state and the model state (mode). We can then repeat all the derivations that led to the PDAF with this modified assumption, and include the mixing step of the IMM method. The so-called IMM-PDAF then results. On the other hand, we can use the IMM as a starting point and replace its measurement update with a PDAF update. The posterior model probabilities can be obtained as a by-product, leading to exactly the same IMM-PDAF.

By taking the hybrid route here, we summarize the workflow of the IMM-PDAF in the following 4 steps.

1) Calculation of IMM mixing probabilities: This is done exactly as in the IMM method.

2) IMM mixing: This is also done exactly as in the IMM method.

3) Filtering: Here we predict and update the mixture over models so that we get a new mixture

over models for each measurement, including misdetection. The prediction is done for each mode independently, just as in Section 6.4. Thus, before we are ready to process the measurement scan Z_k our predicted pdf of the state vector is

$$p_{k|k-1}(\mathbf{x}) = \sum_{s_k} \Pr\{s_k \,|\, Z_{1:k-1}\} p(\mathbf{x}_k \,|\, s_k, Z_{1:k-1}) = \sum_{s_k} p_{k|k-1}^{(s_k)} \mathcal{N}(\mathbf{x}_k; \hat{\mathbf{x}}_{k|k-1}^{(s_k)}, \mathbf{P}_{k|k-1}^{(s_k)}). \tag{7.43}$$

To address the measurement update, we first describe how the IMM-update is conducted conditional on each association hypothesis, before we explain how the association probabilities are evaluated. To be able to re-use our previous derivations of the PDAF with minimal modifications, we introduce the hybrid state vector

$$\mathbf{y}_k = \begin{bmatrix} \mathbf{x}_k \\ s_k \end{bmatrix} \tag{7.44}$$

which both contains the kinematic state and the model state. The motivation behind this is that key results for the PDAF such as Theorem 7.4.1 also are valid for such an extended state vector. With the hybrid state vector we can then rewrite (7.43) as

$$p_{k|k-1}(\mathbf{y}) = \Pr\{s_k \,|\, Z_{1:k-1}\} p(\mathbf{x}_k \,|\, s_k, Z_{1:k-1}). \tag{7.45}$$

The posterior pdf of $\mathbf{y}_k$ is in its most general form

$$p_k(\mathbf{y}_k) = \sum_{a_k=0}^{m_k} p(\mathbf{y}_k \,|\, a_k, Z_{1:k}) \Pr\{a_k \,|\, Z_{1:k}\}. \tag{7.46}$$

We want to find the hypothesis-conditional posteriors in (7.46). This is done by adapting the PDAF formulas from (7.7) to the hybrid state vector. For the case that $a_k = 0$ we have

$$p(\mathbf{y}_k \,|\, a_k = 0, Z_{1:k}) = p(\mathbf{x}_k, s_k \,|\, Z_{1:k-1}) \tag{7.47}$$

which, we recall, is the same as

$$p(\mathbf{x}_k \,|\, a_k = 0, Z_{1:k}) = \sum_{s_k} \Pr\{s_k \,|\, Z_{1:k-1}\} p(\mathbf{x}_k \,|\, s_k, Z_{1:k-1}). \tag{7.48}$$

The probability $\Pr\{s_k \,|\, Z_{1:k-1}\}$ is simply the predicted mode probability that the IMM prediction step provides, as given by (6.5). The pdf $p(\mathbf{x}_k \,|\, s_k, Z_{1:k-1})$ is the mode-conditional pdf of the state, also provided by the IMM method. For the case that $a_k > 0$, again by straightforward adaptation of the PDAF formulas in (7.7), we have

$$p(\mathbf{y}_k \,|\, a_k = i, Z_{1:k}) \propto f_{\mathbf{z}}(\mathbf{z}_k^{a_k} \,|\, \mathbf{x}_k, s_k) p(\mathbf{x}_k \,|\, s_k, Z_{1:k-1}) \Pr\{s_k \,|\, Z_{1:k-1}\}. \tag{7.49}$$

This can also be written as

$$p(\mathbf{x}_k \,|\, a_k, Z_{1:k}) \propto \sum_{s_k} \Lambda^{s_k, a_k} \Pr\{s_k \,|\, Z_{1:k-1}\} p(\mathbf{x}_k \,|\, s_k, a_k, Z_{1:k}) \tag{7.50}$$

where

$$p(\mathbf{x}_k \,|\, s_k, a_k, Z_{1:k}) = \frac{f_{\mathbf{z}}(\mathbf{z}_k^{a_k} \,|\, \mathbf{x}_k, s_k) p(\mathbf{x}_k \,|\, s_k, Z_{1:k-1})}{\Lambda^{s_k, a_k}} \tag{7.51}$$

$$\Pr\{s_k \,|\, a_k, Z_{1:k}\} = \frac{\Lambda^{s_k, a_k} \Pr\{s_k \,|\, Z_{1:k-1}\}}{\sum_{s_k} \Lambda^{s_k, a_k} \Pr\{s_k \,|\, Z_{1:k-1}\}} \tag{7.52}$$

$$\Lambda^{s_k, a_k} = \int f_{\mathbf{z}}(\mathbf{z}_k^{a_k} \,|\, \mathbf{x}_k, s_k) p(\mathbf{x}_k \,|\, s_k, Z_{1:k-1}) \mathrm{d}\mathbf{x}_k. \tag{7.53}$$

Thus, for association hypotheses that include a target detection we evaluate $p(\mathbf{x}_k \,|\, s_k, a_k, Z_{1:k})$ using a standard Bayes update for each mode and measurement, and we evaluate the conditional mode probability $\Pr\{s_k \,|\, a_k, Z_{1:k}\}$ for each measurement, by means of the "mode-measurement likelihood" The underlying argument that was used for splitting up the general product in (7.49) into the three terms in (7.51)-(7.53) goes as follows: The pdf of $\mathbf{x}_k$ itself must clearly normalize, forcing us to introduce Λ^{s_k,a_k} as a normalization constant. However, since Λ^{s_k,a_k} depends on s_k we cannot simply put it aside in the *common* normalization constant for $\mathbf{x}_k$ and s_k. Instead, we must introduce it in the numerator, which is done in (7.52), in order to compensate for its presence in the denominator of (7.51).

To summarize, as the output of the filtering step we have a double mixture

$$p_k(\mathbf{x}_k) = \sum_{a_k=0}^{m_k} \Pr\{a_k \,|\, Z_{1:k}\} \sum_{s_k} \Pr\{s_k \,|\, a_k, Z_{1:k}\} p(\mathbf{x}_k \,|\, s_k, a_k, Z_{1:k}) \tag{7.54}$$

It remains to calculate the association probabilities $\Pr\{a_k \,|\, Z_{1:k}\}$. These were given by Theorem 7.4.1 for the PDAF, and we can still use the same result with the $\mathbf{x}_k$ replaced by the hybrid state vector $\mathbf{y}_k$. However, the likelihood l^{a_k} must then be redefined as a marginalization integral over $\mathbf{y}_k$, and not only over $\mathbf{x}_k$, as we did on page 107. Thus, its expression must include a sum over s_k in addition to the integral over $\mathbf{x}_k$:

$$\begin{aligned} l^{a_k} &= \int p(\mathbf{z}_k^{a_k} \,|\, \mathbf{y}_k) p_{k|k-1}(\mathbf{y}_k) \mathrm{d}\mathbf{y}_k \\ &= \sum_{s_k} \Pr\{s_k \,|\, Z_{1:k-1}\} \int p(\mathbf{z}_k^{a_k} \,|\, \mathbf{x}_k, s_k) p(\mathbf{x}_k \,|\, s_k, Z_{1:k-1}) \mathrm{d}\mathbf{x}_k \\ &= \sum_{s_k} \Lambda^{s_k,a_k} \Pr\{s_k \,|\, Z_{1:k-1}\}. \end{aligned} \tag{7.55}$$

We recognize the expression for l^{a_k} as the denominator or normalization constant in (7.52). By furthermore imposing the Poisson-uniform clutter assumption, the association probabilities are given by

$$\Pr\{a_k \,|\, Z_{1:k}\} \propto \begin{cases} \lambda(1-P_\mathrm{D}) & \text{if } a_k = 0 \\ P_\mathrm{D} l^{a_k} & \text{if } a_k > 0. \end{cases} \tag{7.56}$$

in agreement with Corollary 7.4.3.

4) PDAF mixture reduction: We have now obtained our posterior density, but it is a mixture (over the models) of a mixture (over the association hypotheses). We reduce the mixture over association hypotheses to a single Gaussian for each model by means of the same mixture reduction scheme that was used for the PDAF in Section 7.4.6. Since the mixture reduction is done independently per model, we need the mode-conditional association probabilities, which according to Bayes' rule are given by

$$\Pr\{a_k \,|\, s_k, Z_{1:k}\} = \frac{\Pr\{s_k \,|\, a_k, Z_{1:k}\} \Pr\{a_k \,|\, Z_{1:k}\}}{\Pr\{s_k \, Z_{1:k}\}} = \frac{\Pr\{s_k \,|\, a_k, Z_{1:k}\} \Pr\{a_k \,|\, Z_{1:k}\}}{\sum_{a_k} \Pr\{s_k \,|\, a_k, Z_{1:k}\} \Pr\{a_k \,|\, Z_{1:k}\}}. \tag{7.57}$$

To obtain the mode-conditional densities needed at the next estimation cycle we perform mixture reduction on the following mixture

$$p(\mathbf{x}_k \,|\, s_k, Z_{1:k}) = \sum \Pr\{a_k \,|\, s_k, Z_{1:k}\} p(\mathbf{x}_k \,|\, s_k, a_k, Z_{1:k}). \tag{7.58}$$

After this we are left with a mixture over modes only, where the mixture weights $\Pr\{s_k \,|\, Z_{1:k}\}$ are given by the expression in the denominator of (7.57). This concludes our treatment of the IMM-PDAF in general terms.

7.7.1 Gaussian-linear formulas

Under Gaussian-linear assumptions all the quantities involved will of course be expressed in terms of Gaussians. Before the measurement update, the posterior is then given by

$$p_{k|k-1}(\mathbf{x}) = \sum_{s_k} p_{k|k-1}^{(s_k)} \mathcal{N}(\mathbf{x}_k; \hat{\mathbf{x}}_{k|k-1}^{(s_k)}, \mathbf{P}_{k|k-1}^{(s_k)}). \tag{7.59}$$

where we recognize $p_{k|k-1}^{(s_k)}$ as the shorthand notation for the predicted mode probabilities. For every element in this mixture and for every measurement we run a Kalman filter (or EKF) whose output is given by

$$[\hat{\mathbf{x}}_k^{(s_k)}, \mathbf{P}_k^{(s_k)}, \hat{\mathbf{x}}_{k|k-1}^{(s_k)}, \mathbf{P}_{k|k-1}^{(s_k)}, \mathbf{S}_k^{(s_k)}] = \mathrm{KF}(\hat{\mathbf{x}}_{k-1}^{0,(s_k)}, \mathbf{P}_{k-1}^{0,(s_k)}, \mathbf{z}_k). \tag{7.60}$$

The hypothesis-conditional mode likelihood is given by

$$\Lambda^{s_k,a_k} = \mathcal{N}(\mathbf{z}_k^{a_k}; \hat{\mathbf{z}}_{k|k-1}^{(s_k)}, \mathbf{S}_k^{(s_k)}) \tag{7.61}$$

and the hypothesis likelihood l^{a_k} is then given by

$$l^{a_k} = \sum_{s_k} p_{k|k-1}^{(s_k)} \Lambda^{s_k,a_k}. \tag{7.62}$$

The mode- and hypothesis-conditional posteriors are

$$p(\mathbf{x}_k \,|\, s_k, a_k, Z_{1:k}) = \begin{cases} \mathcal{N}(\mathbf{x}; \hat{\mathbf{x}}_{k|k-1}^{(s_k)}, \mathbf{P}_{k|k-1}^{(s_k)}) & \text{if } a_k = 0 \\ \mathcal{N}(\mathbf{x}; \hat{\mathbf{x}}_k^{(s_k)}, \mathbf{P}_k^{(s_k)}) & \text{if } a_k > 0. \end{cases} \tag{7.63}$$

The actual implementation of an IMM-PDAF will then include inserting (7.61) - (7.63) into the formulas (7.56), (7.52) and (7.57), that provide the association probabilities, posterior mode probabilities and mode-conditional association probabilities, respectively. The mixture reduction over association hypotheses in Step 4 (i.e., mode-conditional mixture reduction) is done by the techniques in Section 7.4.6 just as for the ordinary PDAF.

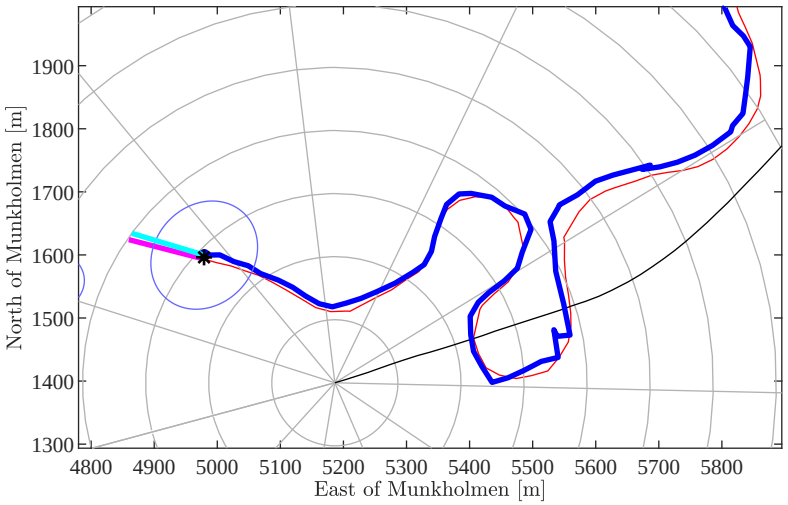


Figure 7.5: Trajectory from an IMM-PDAF tracker on the Joyride data at $k = 195$, together with ground truth from GPS and the GPS data from Telemetron, on which the radar was mounted. The cyan and magenta lines indicate estimated and true course of the motorboat.

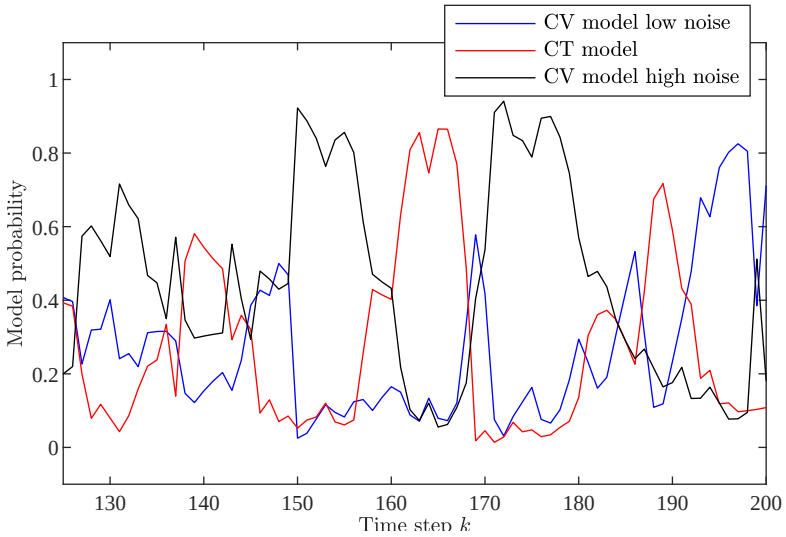


Figure 7.6: Posterior model probabilities over time steps 125-200. Sometimes strong turns are caught by the CT model, and sometimes they are caught by the large noise CV model.

7.7.2 Case study: The joyride data

Here we take a look at the IMM-PDAF implemented on a radar data set that was recorded as part of the Autosea project. In this particular data set, a small motorboat is performing several very strong maneuvers. Some of these have been plotted in Figure 7.5. Due to its small size, misdetections frequently occur. This is therefore a challenging data set for a radar tracker, and any attempt at maintaining a track on the motorboat during the entire sequence without multiple models has so far failed.

Figures 7.6 and 7.7 give some impressions of what you can expect when working with this data set, which will be given as the first compulsory programming assignment in TTK4250. In particular, notice that we never encounter a situation where only one model takes all the probability mass. Beyond simply maintaining the track, the main challenge in this data set is to estimate the course of the motorboat with acceptable accuracy. From Figure 7.7 we can judge that the estimates tend to be in the order of 10° away from the ground truth, but often much larger deviations, up to 90° are observed. This would be some reason for concern in a real-world collision avoidance system. It seems unlikely that this problem could be solved by anything else than including additional sensors (e.g., cameras) with a faster update rate.

7.8 References and chapter remarks

The derivation of the PDAF is first and foremost inspired by [3], which is a standard reference on PDAF and JPDA. At the same time, a notation more similar to [115] has been used, to narrow the gap between the classical tracking methods discussed in [3] and the modern state of the art. The IPDA was originally presented in [78]. The formulation presented here is equivalent, but uses a likelihood formulation that is adapted from [118]. A plethora of related methods have since been developed (IMM-IPDA, multi-target versions of the IPDA, etc.). Many of these are treated in [21] and [80]. The topic of clutter estimation has been addressed in references such as [67], [65], [79] and [119]. While validation gates typically are elliptical, other shapes are considered in [38]. Our treatment of the IMM-PDAF is to some extent inspired by [106], and differs significantly from how it is presented in [3]. The seminal article for the IMM-PDAF is [63], but it was used in a rather different context there; namely track-initialization. However, the consensus has since then become

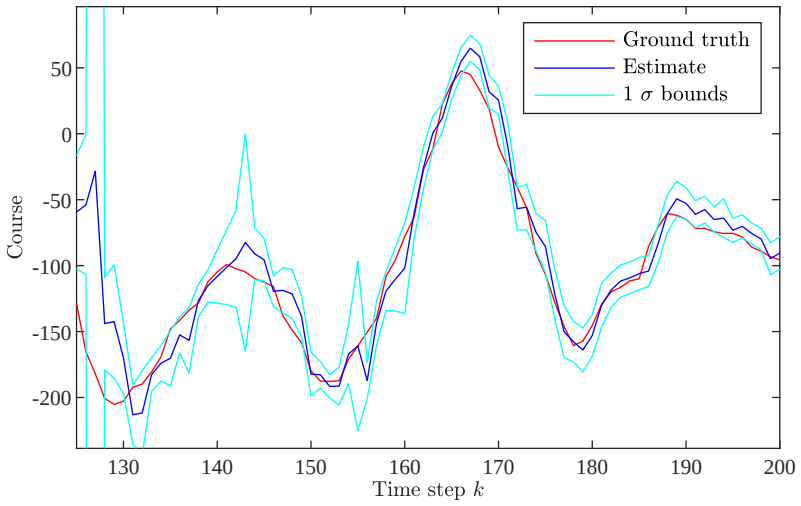


Figure 7.7: Estimates and ground truth for the motorboat's course. While the estimates generally seem fairly accurate, large deviations do frequently occur, as one can see on the far left.

clear that IPDA and related methods are preferable for this task. Other references on IMM-PDAF include [60]. For readers who want to explore more sophisticated mixture reduction schemes, that hopefully can lead to more robust tracking methods than the PDAF and its relatives, good starting points are [92] and [116]. As mentioned earlier in this chapter, feature information such as amplitudes can also sometimes be used to improve the robustness of tracking methods, including variations of the PDAF [63] [18][22]. Another issue that can be of practical importance, possibly also in the joyride data, is state-dependent clutter. For example, false detections are often observed in the wakes of boats. This has been addressed in [89], [20] and [111].

7.9 Exercises

Exercise 7.1 In the beginning of this chapter we encountered both the false alarm rate P_{FA} and the rate of the Poisson clutter process Λ. What is the relationship between these entities for a $2D$ range-bearing sensor covering $1\,\text{km} \times 180^\circ$ where each resolution cell is $1\,\text{m} \times 3^\circ$? How can you express the clutter intensity in terms of the false alarm rate? ■

Exercise 7.2 We have implicitly specified the details of the mixture (7.5) that describes the posterior density of single-target tracking under the general assumptions S1-S7. The building blocks are the expressions in (7.7) and Theorem 7.4.1. Let us now keep the assumptions S1-S5 on their general form, but make S5 and S6 more precise by assuming that the spatial density of false alarms is uniform over the validation gate, and that the cardinality distribution of false alarms is Poisson with rate λV_k where V_k is the volume of the validation gate. Provide an expression for $p(\mathbf{x}_k \,|\, Z_{1:k})$ under these assumptions (i.e., without invoking Gaussian-linear assumptions). ■

Exercise 7.3 Prove Corollary 7.4.3 using Theorem 7.4.1 as a starting point. ■

8. Multi-target tracking: JPDA

In this chapter we take the big step from single-target tracking to multi-target tracking. This leads to a considerable increase in complexity, because we now must deal with several possible combinations associations between the n_k targets and the m_k measurements, as opposed to the $m_k + 1$ associations that the PDAF had to consider.

In a typical tracking system single-target tracking may be sufficient 99% of the time. If targets are well separated, so that there is no way to confuse their measurements, there is no need to consider whether a measurements comes from another target. However, when targets get sufficiently close to each other, we have to consider joint data association hypotheses. In this chapter we will become familiar the multi-target tracking problem, and how it is solved by the Joint Probablistic Data Association (JPDA), which is a generalization of the PDAF [36][3].

8.1 The multi-target tracking problem

Let us first look at a typical multi-target scenario to motive the discussion to come. In the leftmost part of Figure 8.1 we see three tracks with validation gates and a total of four validated measurements. We see that the measurement $\mathbf{z}_k^1$ could have been caused by both target 1 and target 2, while the measurement $\mathbf{z}_k^2$ could have been caused by both target 2 and target 3. If we are inclined to believe that $\mathbf{z}_k^1$ indeed is caused by target 2, then we should also be inclined to believe that $\mathbf{z}_k^2$ and $\mathbf{z}_k^4$ are not due to target 2. This would again be related to how plausible it is that $\mathbf{z}_k^2$ is generated by target 3, and how plausible it is $\mathbf{z}_k^4$ is a false alarm, and so on.

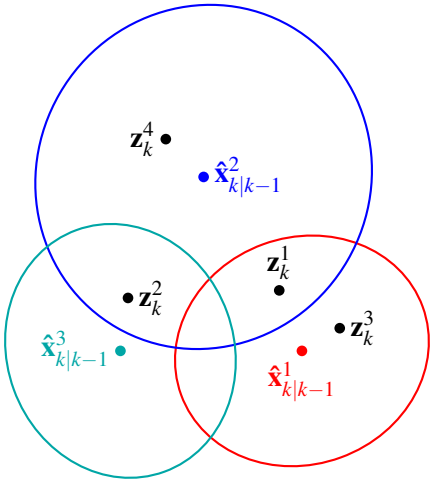
In order to conduct this kind of reasoning in a structured manner we need assumptions which are similar to, but also more general than the PDAF assumptions from page 105. In the sequel we first discuss two assumptions which constitute a cornerstone for standard tracking methods, before we proceed to outlining the full standard model for multi-target tracking.

8.1.1 The at-most-one-assumptions and the validation matrix

In multi-target tracking, the following two assumptions are almost always made:

O1 *Point target assumption*: Any target generates at most one measurement.

O2 *No merged measurement assumption*: Any measurement comes from at most one target.



$t\backslash j$	0	1	2	3	4
1	1	1	0	1	0
2	1	1	1	0	1
3	1	0	1	0	0

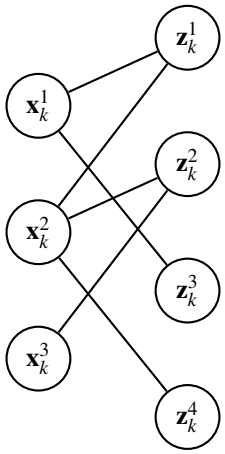


Figure 8.1: Illustration of multi-target tracking scenario with three targets and four measurements, corresponding target-oriented validation matrix, and bipartite association graph.

We recognize the first of these at-most-one assumptions from our treatment of the PDAF on page 106. This assumption is physically dubious, not only because it rules out extended targets, but also because it rules out blurring effects which makes even point targets affect several resolution cells. However, its validity can to a large degree be satisfied by clustering methods which ensure that only the centroid of a potential object image is returned from the sensor image. It is a useful assumption for two reasons. First, it means that for a single target, we only need to consider the possibility that individual measurements come from that target, and not the possibility that combinations of measurements come from that target. This significantly reduces computational complexity. Second, without this assumption a tracking method could be tempted to attribute all measurements in a sensor scan to a single target, depending on prior estimates and uncertainties of clutter intensity, target extent and so on. This phenomenon, known as hospitality to clutter [113], is much less likely to occur when the point target assumption is enforced.

The second at-most-one assumption rules out the possibility that a single resolution cell is affected by more than one target. Again, the assumption is physically dubious. No sensor has perfect resolution, and if two targets are close enough they will affect the same resolution cells. As another example, from tracking in camera images, if two objects overlap, a feature extractor may generate one common detection. The most important reason why this assumption is so prevalent is perhaps that it helps ensuring the well-posedness of the multi-target tracking problem. If two targets are allowed to affect the same resolution cell, and thus generate the same measurement, could it not also be that it was generated by a hundred targets?

Armed with the at-most-one assumptions, we return to the scenario in Figure 8.1. It is now clear that $\mathbf{z}_k^1$ must either originate from target 1, from target 2 or from clutter. Similar statements can be made for the other measurements. Target 1 must either have generated $\mathbf{z}_k^1$, $\mathbf{z}_k^3$ or no measurement, and similar statements can be made for the other targets. The information from the gating process can be represented in a validation matrix as shown in the middle part of Figure 8.1. In this matrix, we have a 1 whenever an assignment between a track (row) and a measurement (column) is possible. The matrix includes a zeroth column representing misdetection, which can be assigned to an arbitrary number of tracks. An valid association hypothesis must then involve a 1 for each row, in such a manner that no column except the zeroth column has more than one 1. The association hypotheses can be written in several different ways, for example as vectors of the same length as the number of targets, whose elements are the corresponding assigned measurements. Possible hypotheses for this scenario can then be $[1,2,0]$ and $[0,0,2]$. On the other hand, hypotheses such as

$[1,1,0]$ or $[2,3,1]$ are not possible.

The constraints imposed by the validation matrix can also be represented in other manners. For example, the constraints can be represented as an undirected graph as shown in the rightmost part of Figure 8.1. This graph is an example of what is known as a bipartite graph: Its vertices can be decomposed into two disjoint sets (i.e., tracks and measurements) such that no vertices in the same set are adjacent. In some sense, the graph gives a more economical and visual representation of the possible associations. The task of finding a good association hypothesis can be related to graph-theoretical operations, such as finding a maximal matching between the two sets of vertices.

8.1.2 The standard model for multi-target tracking

Before we delve into the details of the JPDA, we shall take a brief look ahead towards the dominating paradigm in multi-target tracking. Below we present the assumptions of what is known as the standard model of multi-target tracking, and which is more general that the assumptions underlying the JPDA.

- M1 New targets are born according to a Poisson process with intensity $\mu(\mathbf{x})$.
- M2 Existing targets survive from time step $k-1$ to k with probability $P_{\mathrm{S}}(\mathbf{x}_{k-1})$.
- M3 The motion of a surviving target is given by $f_{\mathbf{x}}(\mathbf{x}_k|\mathbf{x}_{k-1})$.
- M4 A target with state $\mathbf{x}_k$ generates a measurement $\mathbf{z}_k$ with probability $P_{\mathrm{D}}(\mathbf{x}_k)$.
- M5 Clutter measurements occur according to a Poisson process with intensity $\lambda(\mathbf{z})$.
- M6 The measurement of a detected target is related to the state according to $f_{\mathbf{z}}(\mathbf{z}_k|\mathbf{x}_k)$.

The obvious independence assumptions do of course also apply. This includes the standard Markov assumption: Once $\mathbf{x}_{k-1}$ is known, then $\mathbf{x}_k$ is independent of $\mathbf{x}_{k-2}$, $\mathbf{x}_{k-3}$ etc.

It follows from assumptions M1 and M2 that several targets eventually will be present. Furthermore, these assumptions imply that there will be uncertainty regarding the number of targets present. In the previous chapter we discussed how this kind of uncertainty could be dealt with in a single-target context by the M/N rule and the IPDA. In Chapters 9 and 13, we will on the other hand address this uncertainty in a formal manner according to assumptions M1 and M2.

8.2 The JPDA: Moment-based mixture reduction

The JPDA conforms to the standard model as a special case, where

- M7 The number n of targets is constant and known (i.e., zero birth intensity and death probability).
- M8 The single target motion model and the likelihood are Gaussian-linear:

$$\begin{aligned} f_{\mathbf{x}}(\mathbf{x}_k \,|\, \mathbf{x}_{k-1}) &= \mathcal{N}(\mathbf{x}_k \,;\, \mathbf{F}\mathbf{x}_{k-1}, Q) \\ f_{\mathbf{z}}(\mathbf{z}_k \,;\, \mathbf{x}_k) &= \mathcal{N}(\mathbf{z}_k \,|\, \mathbf{H}\mathbf{x}_k, \mathbf{R}) \end{aligned} \tag{8.1}$$

- M9 At the end of the previous estimation cycle, the posterior densities of the targets are independent and Gaussian

$$p^t_{k-1}(\mathbf{x}^t_{k-1}) = \mathcal{N}(\mathbf{x}^t_{k-1} \,;\, \hat{\mathbf{x}}^t_{k-1}, \mathbf{P}^t_{k-1}). \tag{8.2}$$

- M10 Target t has constant detection probability P^t_{D}.
- M11 The clutter Poisson process has constant intensity λ.

As the JPDA is supposed to generalize the PDAF, it builds on the same strategy of merging association hypotheses together, so that only a single Gaussian remains for each target. This is of course necessary to ensure that M9 remains satisfied. The main complication when compared to the PDAF is that an exponential number of association hypotheses now are involved, as sketched in Section 8.1.1.

8.2.1 Prediction in the JPDA

The prediction from time step $k-1$ to time step k follows the same Chapman-Kolmogorov equation as in (7.6) and (7.15):

$$p^t_{k|k-1}(\mathbf{x}^t_k) = \int f_{\mathbf{x}}(\mathbf{x}^t_k|\mathbf{x}^t_{k-1}) p^t_k(\mathbf{x}^t_{k-1})\,\mathrm{d}\mathbf{x}^t_{k-1} = \mathcal{N}(\mathbf{x}^t_k; \hat{\mathbf{x}}^t_{k|k-1}, \mathbf{P}^t_{k|k-1}). \tag{8.3}$$

The expectation $\hat{\mathbf{x}}^t_{k|k-1}$ and covariance $\mathbf{P}^t_{k|k-1}$ are given by (7.16) and (7.17).

8.2.2 Association events and their posterior conditional densities

Building upon the informal discussion of the scenario in Figure 8.1 we shall now introduce the association hypotheses of the JPDA in a more concrete manner. We can define an association hypothesis a_k as a vector $a_k = [a^1_k, a^2_k, \ldots, a^n_k]$ such that

$$a^t_k = \begin{cases} j & \text{if measurement } j \text{ is claimed by target } t \\ 0 & \text{if no measurement is claimed by target } t. \end{cases} \tag{8.4}$$

For future reference, let us point out that the value 0 (i.e., a misdetection) in an association hypothesis often is referred to as a *dummy measurement*. This is in contrast to a *proper measurement* whenever a^t_k is a natural number greater than 0.

In the literature, association hypotheses defined as in (8.4) are sometimes described as mappings.[1] The set of possible association hypotheses for a scenario with n tracks and m_k measurements is then defined as the set of mappings $a_k : \{1,\ldots,n\} \to \{0,1,\ldots,m_k\}$ such that if $a^s_k = a^t_k \neq 0$, then $s = t$. This more abstract definition does not take gating into account. However, in practice all association hypotheses that violate the gating constraints should get negligible probability, so this is not an important discrepancy. It is not hard to see that any such mapping can be represented as a vector of length n, and vice versa.

The association hypotheses are mutually exclusive and exhaustive. The total probability theorem therefore yields

$$p(\mathbf{x}^1_k, \ldots, \mathbf{x}^n_k \,|\, Z_{1:k}) = \sum_{a_k} p(\mathbf{x}^1_k, \ldots, \mathbf{x}^n_k \,|\, a_k, Z_{1:k}) \Pr\{a_k \,|\, Z_{1:k}\} \tag{8.5}$$

where the sum goes over all feasible association hypotheses. It is straightforward to obtain the event-conditional densities $p(\mathbf{x}^1_k, \ldots, \mathbf{x}^n_k \,|\, a_k, Z_{1:k})$. We use a KF whenever a measurement $\mathbf{z}^{a^t_k}_k$ is assigned to target t, and we use the prediction otherwise. In terms of pdfs we have

$$p(\mathbf{x}^1_k, \ldots, \mathbf{x}^n_k \,|\, a_k, Z_{1:k}) \propto \prod_{t\,:\,a^t_k = 0} p^t_{k|k-1}(\mathbf{x}^t_k) \prod_{t\,:\,a^t_k > 0} f_{\mathbf{z}}(\mathbf{z}^{a^t_k}_k \,|\, \mathbf{x}^t_k) p^t_{k|k-1}(\mathbf{x}^t_k). \tag{8.6}$$

To make this more concrete under the Gaussian-linear assumptions M8 and M9, we denote the Kalman gain for track number t by $\mathbf{W}^t_k = \mathbf{P}^t_{k|k-1}\mathbf{H}^\mathsf{T}(\mathbf{H}\mathbf{P}^t_{k|k-1}\mathbf{H}^\mathsf{T} + \mathbf{R})^{-1}$ and express the posterior event-conditional expectations and covariances by

$$\hat{\mathbf{x}}^{t,a^t_k}_k = \begin{cases} \hat{\mathbf{x}}^t_{k|k-1} & \text{if } a^t_k = 0 \\ \hat{\mathbf{x}}^t_{k|k-1} + \mathbf{W}^t_k(\mathbf{z}^{a^t_k}_k - \mathbf{H}\hat{\mathbf{x}}^t_{k|k-1}) & \text{if } a^t_k > 0 \end{cases} \tag{8.7}$$

$$\mathbf{P}^{t,a^t_k}_k = \begin{cases} \mathbf{P}^t_{k|k-1} & \text{if } a^t_k = 0 \\ (\mathbf{I} - \mathbf{W}^t_k\mathbf{H})\mathbf{P}^t_{k|k-1} & \text{if } a^t_k > 0. \end{cases} \tag{8.8}$$

[1] A mapping is a generalized function which, in contrast to regular functions, can have several output values for a given argument. However, the mappings considered in the context of association hypotheses *are* functions.

We can then express the joint event-conditional posterior as

$$p(\mathbf{x}_k^1,\dots,\mathbf{x}_k^n \,|\, a_k, Z_{1:k}) = \prod_{t=1}^{n} \mathcal{N}(\mathbf{x}_k^t; \hat{\mathbf{x}}_k^{t,a_k^t}, \mathbf{P}_k^{t,a_k^t}). \tag{8.9}$$

As for the PDAF, fairly straightforward amendments must be made if KFs are replaced by EKFs or if the measurement noise, clutter intensity or other quantities depend on the state or the measurement.

8.2.3 Association events and their posterior probabilities

To obtain the hypothesis probabilities $\Pr\{a_k \,|\, Z_{1:k}\}$ we start by defining the track-to-measurement likelihood

$$l^{t,a_k^t} = \int f_{\mathbf{z}}(\mathbf{z}_k^{a_k^t} \,|\, \mathbf{x}_k^t) p_{k|k-1}^t(\mathbf{x}_k^t)\, \mathrm{d}\mathbf{x}_k^t = \mathcal{N}(\mathbf{z}_k^{a_k^t}; \mathbf{H}\hat{\mathbf{x}}_k^{t,a_k^t}, \mathbf{H}\mathbf{P}_k^{t,a_k^t}\mathbf{H}^\mathsf{T} + \mathbf{R}). \tag{8.10}$$

Also, as in Chapter 7, let φ_k denote the number of clutter measurement under the current association hypothesis.

Theorem 8.2.1 — Association probabilities for the JPDA. The probabilities for the association hypotheses in the JPDA are given by

$$\Pr\{a_k | Z_{1:k}\} \propto \lambda^{\varphi_k} \prod_{t\,:\,a_k^t=0} (1 - P_{\mathrm{D}}^t) \prod_{t\,:\,a_k^t>0} P_{\mathrm{D}}^t\, l^{t,a_k^t}. \tag{8.11}$$

Proof. In the same way as we did for the PDAF, we separate the current measurement and its cardinality from the historical measurement set:

$$\Pr\{a_k \,|\, Z_{1:k}\} = \Pr\{a_k \,|\, Z_{1:k-1}, Z_k, m_k\} \propto p(Z_k \,|\, m_k, a_k, Z_{1:k-1}) \Pr\{a_k \,|\, m_k, Z_{1:k-1}\} \tag{8.12}$$

The first term in (8.12) is then

$$\begin{aligned} p(Z_k \,|\, m_k, a_k, Z_{1:k-1}) &= \int p(Z_k \,|\, m_k, a_k, \mathbf{x}_k^1, \dots, \mathbf{x}_k^n)\, p(\mathbf{x}_k^1, \dots, \mathbf{x}_k^n \,|\, a_k, Z_{1:k-1})\, \mathrm{d}\mathbf{x}_k^1 \cdots \mathrm{d}\mathbf{x}_k^n \\ &= \frac{1}{V^{\varphi_k}} \int \cdots \int \prod_{t\,:\,a_k^t>0} \left[f_{\mathbf{z}}(\mathbf{z}_k^{a_k^t} \,|\, \mathbf{x}_k^t) p_{k|k-1}(\mathbf{x}_k^t \,|\, Z_{1:k-1}) \right] \\ &\qquad \prod_{t\,:\,a_k^t=0} \left[p_{k|k-1}(\mathbf{x}_k^t \,|\, Z_{1:k-1}) \right] \mathrm{d}\mathbf{x}_k^1 \cdots \mathrm{d}\mathbf{x}_k^n \\ &= \frac{1}{V^{\varphi_k}} \prod_{t\,:\,a_k^t>0} l^{t,a_k^t}. \end{aligned} \tag{8.13}$$

To find the second term in (8.12) we start by defining the individual track-wise detection events $\tau(t)$ defined as follows: If target t is detected then $\tau^t = 1$, otherwise it is zero. Furthermore, we define the *track-oriented configuration* as the vector $\boldsymbol{\tau} = [\tau^1, \dots, \tau^n]^\mathsf{T}$. In other words, $\boldsymbol{\tau}$ contains part of the information contained in a_k. It contains the information of which targets are detected, but it does not contain any information about which measurements are those detections. The dependencies between a_k and m_k can by means of $\boldsymbol{\tau}$ be visualized as in Figure 8.2.

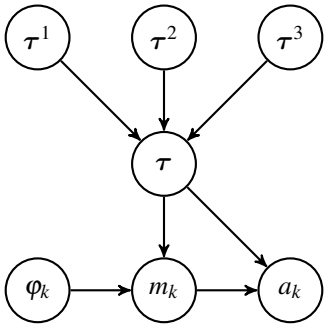


Figure 8.2: Relationships involved in the calculation of prior association probabilities for the JPDA.

The unconditional probability of the event τ is equal to the product of the individual detection events:

$$\Pr\{\tau\} = \prod_{t:a_k^t=0} (1-P_{\mathrm{D}}^t) \prod_{t:a_k^t>0} P_{\mathrm{D}}^t. \tag{8.14}$$

Conditional on τ we have the following probabilities

$$\Pr\{m_k \,|\, \tau\} = e^{-V\lambda} \frac{(V\lambda)^{\varphi_k}}{\varphi_k!} \tag{8.15}$$

$$\Pr\{a_k \,|\, \tau, m_k\} = \frac{\varphi_k!}{m_k!} \tag{8.16}$$

The first probability follows because the number of measurements not accounted for by τ, that is φ measurements, are Poisson distributed with rate $V\lambda$, and the total number of measurements m_k are uniquely given by τ and φ_k. The second probability follows because given τ, we have $m_k!/(m_k - \sum_t \tau^t)! = m_k!/\varphi_k!$ equally likely permutations of the detected measurements, which each constitute a separate association hypothesis. We can stitch these results together as follows:

$$\begin{aligned}
\Pr\{a_k \,|\, m_k\} &= \Pr\{a_k, \tau \,|\, m_k\} \\
&\propto \Pr\{a_k \,|\, \tau, m_k\} \Pr\{m_k \,|\, \tau\} \Pr\{\tau\} \\
&\propto (V\lambda)^{\varphi_k} \prod_{t:a_k^t=0} (1-P_{\mathrm{D}}^t) \prod_{t:a_k^t>0} P_{\mathrm{D}}^t.
\end{aligned} \tag{8.17}$$

If we now multiply the expressions for the likelihood $p(Z_k \,|\, m_k, a_k, Z_{1:k-1})$ and the prior probability $\Pr\{a_k \,|\, m_k\}$ we see that the V terms cancel. Furthermore, the total power of λ is always m_k, since one λ is contributed by every detected target in the likelihood,and one λ is contributed by every false alarm in the prior probability. Thus, the λ terms can be moved into the proportionality constant, and we are left with (8.11). ■

8.2.4 Marginal association probabilities and mixture reduction

In the same spirit as the PDAF, the key principle of the JPDA is to merge together different hypotheses, so that one in the end is left with one Gaussian for each track. Thus, we are not content with having found the probabilities $\Pr\{a_k \,|\, Z_{1:k}\}$. Rather, we want the marginal probabilities

$$\beta_k^{t,j} = \Pr\{\mathbf{z}_k^j \text{ is assigned to target } t \,|\, Z_{1:k}\} = \sum_{a_k \text{ s.t. } a_k^t=j} \Pr\{a_k \,|\, Z_{1:k}\}. \tag{8.18}$$

Once the marginal probabilities have been calculated, we can for each track find its posterior single-Gaussian approximation according to

$$p_k^t(\mathbf{x}_k^t) = \sum_{j=0}^{m_k} \beta_k^{t,j} \mathcal{N}(\mathbf{x}_k^t; \hat{\mathbf{x}}_k^{t,j}, \mathbf{P}_k^{t,j}) \approx \mathcal{N}(\mathbf{x}_k^t; \hat{\mathbf{x}}_k^t, \mathbf{P}_k^t). \tag{8.19}$$

That is, for each track the posterior density is a Gaussian mixture. Notice that the mixture ranges over all measurements plus misdetection, and not over all association hypotheses. The moments $\hat{\mathbf{x}}_k^{t,j}$ and $\mathbf{P}_k^{t,j}$ are indeed the same as the hypothesis-conditional moments in (8.7) and (8.8). However, it is important to keep in mind that $\hat{\mathbf{x}}_k^{t,a_k^t}$ is entirely given by the track t and the measurement selected by the hypothesis a_k for track t. The notation a_k^t signifies after all the index of this measurement. The merged expectation $\hat{\mathbf{x}}_k^t$ and covariance $\mathbf{P}_k^t$ are found using the same mixture-reduction formulas as for the single-target PDAF. For brevity, we do not repeat the details but refer the reader to (7.23) and (7.25). This concludes our development of the JPDA.

8.3 How to implement the JPDA

Among multi-target tracking algorithms, the JPDA is far from the most complex. Nevertheless, the required summation over association hypotheses is in general of exponential complexity. Any JPDA implementation that does not take steps to mitigate this will soon run into difficulties.

Some of the techniques that should be used to mitigate complexity are fairly obvious. We have already looked at validation gating, which ensures that for any track we only consider plausible measurements. The process of validation gating can be further enhanced by spatial indexing using k-d trees [84]. In this approach, the measurement space is partitioned into a hirearchy of cells, so that each track and measurement can be labeled by the cells in which they belong. It is then only necessary to check measurements against tracks in the same or nearby cells. For this to work, the tree structure describing the hirearchy of cells must be build, but this can be done in $O(m \log m)$ time (assuming the number of measurements m is larger than the number of tracks n), as opposed to the $O(nm)$ time that is required to check all measurements against all tracks. However, while spatial clustering may be worthwhile in large-scale real-world implementations, the gains are hardly significant in smaller scenarios such as those typically simulated in research papers.

More important techniques for complexity mitigation are *track clustering* and approximations of marginal association probabilities within each track cluster. Dividing the tracks into clusters is fairly straightforward in the JPDA. In section 8.3.1 we will describe on possible approach using *single linkage*. Having decoupled the multi-target tracking problem in this way, we may be able to run single-target PDAF modules for most of the targets. Only when two or more tracks become so close to each other that their validation gates start to share measurements do we actually need to switch to the JPDA. If such a cluster contains something like 2 tracks and 2 shared measurements, we do not need to worry about computational complexity; enumerating all hypotheses is perfectly feasible. But if we have a situation with, say, 5 tracks and 10 shared measurements, then approximations are absolutely necessary.

Several possibilities for approximative evaluation of association probabilities exist, and we mention three of these here:

1. We may only generate the most probable a_k's. This is by far the easiest approach. It works well for typical scenarios with a moderate number of targets per cluster. We will discuss this approach in detail below, building upon works on similar strategies for multiple hypothesis tracking (MHT) [25], which will be covered in the next chapter.
2. Exploit the fact that a lot of structure is repeated among the different a_k's. This is done in for example the method of [51], which provides a significantly accelerated version of exact JPDA. The method is, however, difficult to implement and it is also patented by Qinetiq. The method does not seem to be compatible with solutions to some known weaknesses of JPDA such as track coalescence.
3. A third and more recent approach is to replace the enumeration of hypotheses by loopy belief propagation [117]. This is also an approximative technique. We will study graphical model theory in more detail in the context of SLAM in Chapter 12.

8.3.1 Clustering of tracks

In any case, the first step towards complexity mitigation is clustering of tracks. Only tracks that share measurements within their validation gates should be treated together. A direct solution to this clustering problem can be implemented by means of the *single linkage* clustering algorithm. It merges clusters whenever it finds that two clusters contain tracks which share measurements. It can be implemented in $O(n^2)$ time as shown in Algorithm 4.

Algorithm 4 Single linkage track clustering

procedure CLUSTER-TRACKS(Ω, n)
  $\texttt{assoc} \leftarrow \mathbf{0}_{2\times n}$ ▷ Master label array
  for $p \in 1:N$ **do**
    if $\texttt{assoc}(2,p) = 0$ **then**
      $\texttt{assoc}(1,p) \leftarrow 1$ ▷ Make track p a master
      $\texttt{assoc}(2,p) \leftarrow p$ ▷ Label track p with its own index
    end if
    for $i \in 1:N \setminus p$ **do**
      if tracks p and i gate common measurements **then**
        if $\texttt{assoc}(2,i) > \texttt{assoc}(2,p)$ **then**
          $\texttt{slaves} \leftarrow$ tracks t such that $\texttt{assoc}(2,t) = \texttt{assoc}(2,i)$
          $\texttt{assoc}(1,\texttt{slaves}) \leftarrow 0$
          $\texttt{assoc}(2,\texttt{slaves}) \leftarrow \texttt{assoc}(2,p)$
        else if $\texttt{assoc}(2,i) < \texttt{assoc}(2,p)$ **then**
          $\texttt{slaves} \leftarrow$ tracks t such that $\texttt{assoc}(2,t) = \texttt{assoc}(2,p)$
          $\texttt{assoc}(1,\texttt{slaves}) \leftarrow 0$
          $\texttt{assoc}(2,\texttt{slaves}) \leftarrow \texttt{assoc}(2,p)$
        else
          $\texttt{assoc}(2,i) \leftarrow \texttt{assoc}(2,p)$
        end if
      end if
    end for
  end for
  $\texttt{masters} \leftarrow \{t$ such that $\texttt{assoc}(1,t) = 1\}$
  for $m \in 1:c$ **do**
    $C_m \leftarrow$ tracks t such that $\texttt{assoc}(2,t) = \texttt{masters}(m)$
  end for
  return $\{C_m\}_1^c$
end procedure

The algorithm uses a $2 \times n$ array called `assoc` which contains all temporarily stored information during the procedure. The first row contains marks for every track that is considered a master of its cluster. There can be maximally one master in each cluster. The second row contains the label of the master for each track that is part of a cluster. The algorithm loops through all tracks p, and consists of potentially two stages for each track. During the first stage, it checks whether the track is claimed by any cluster. If not, then it generates a new singleton cluster, and it becomes the master of that cluster. During the second stage, it identifies all other tracks that share common measurements. Every time such a track i is found, it identifies the master of track i, and thus the cluster that track i belongs to. The clusters of tracks p and i are merged, and the master of the new cluster becomes the one among the masters of the original clusters that came first in the original sorting of the tracks.

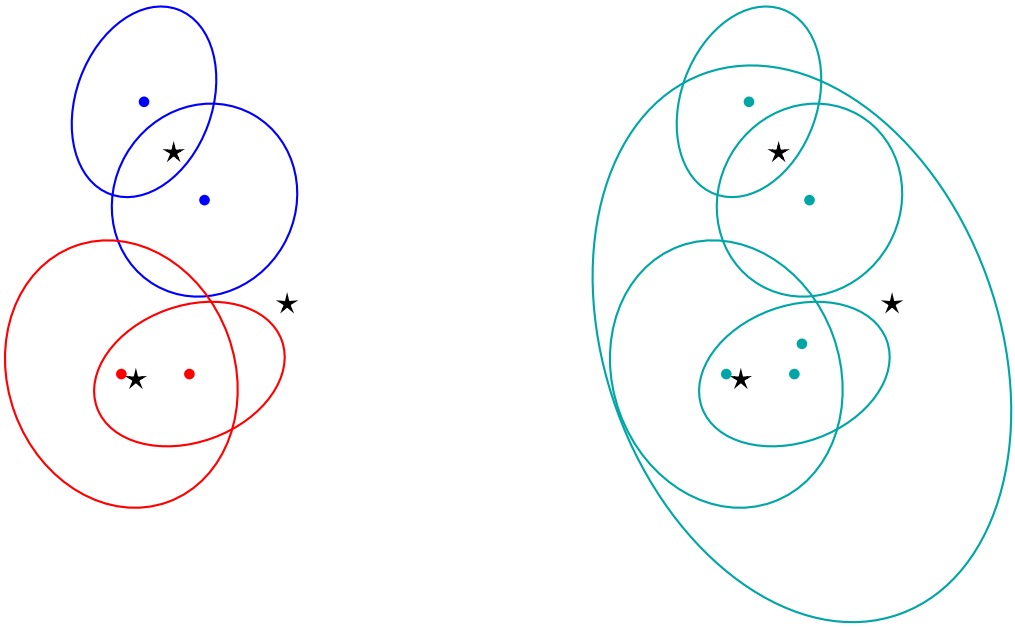


Figure 8.3: Illustration of how a single poor-quality track can sabotage the track clustering.

Single linkage is an appropriate algorithm for track clustering because it works in a local manner. Either two tracks share common measurements, or they don't. The main disadvantage of single linkage, which is a direct consequence of its local mode of operation, is that it may create long clusters that connect tracks far away through *chaining*. In particularly dense cases, single linkage may end clustering all the tracks together in one cluster, even if most of the tracks have no pairwise interactions. If one still wants to keep tracks separated in such cases, one would either have to decrease the sizes of the validation gates, or use alternative clustering algorithms which don't guarantee that all tracks that share measurements are kept in the same cluster. The employment of such clustering methods for track clustering appears to be an open research topic.

8.3.2 The 2D assignment problem and the Auction method

Even in fairly modest clusters, the number of possible hypotheses can easily reach hundreds and thousands and beyond. Most often, only a handful of these will contain something like 99% of the probability mass. It is therefore useful to have methods capable of searching for such good hypotheses. It is of particular interest to find the best hypothesis. Indeed, as opposed to the merging approach of the PDAF and the JPDA, there exist many tracking methods which only attempt to find the best hypothesis. In particular, if we modify the JPDA to only provide the best hypothesis and to discard all other hypotheses, we get what is known as *global nearest neighbor* (GNN). We will become familiar with more advanced approaches building on such an MAP philosophy in the next chapter.

The problem of finding the best association between a set of tracks and a set of measurements is an example of an *assignment problem*. More precisely, it is a 2-dimensional assignment problem. The mathematical structure of such problem is not only relevant to multi-target tracking, but also to a wealth of other problems in optimization and operations analysis. Many efficient solutions have therefore been developed. Here we will restrict attention to the *auction algorithm*, which tends to be very efficient (although its worst-case complexity can be quite bad), and which can be interpreted in a very intuitive manner.

The auction algorithm treats the tracks as *customers*, and the measurements as *items* that the

customers try to buy. The customers make bids for the items, which consequently increase in price. The algorithm terminates when all customers are content with their choices, i.e., when no customer is willing to bid more than what his current item is worth. The the sum of *rewards* that all the customers gain from their respective items will reach its maximal value.

From where do we get such a sum of rewards in the setting of the JPDA? We recall that the association probabilities in (8.11) were given by a product over the tracks. If we take the logarithm, this becomes a sum over the tracks. Instead of attempting to maximize the association probabilities we can aim to maximize the logarithmic hypothesis score given by

$$s(a_k) = \sum_{t \text{ such that } a_k^t=0} \ln(1-P_{\mathrm{D}}^t) \; + \sum_{t \text{ such that } a_k^t>0} \left(\ln P_{\mathrm{D}}^t + \ln l^{t,a_k^t} - \ln \lambda \right). \tag{8.20}$$

Every association hypothesis a_k choses a measurement a_k^t for each track t. If it is a proper measurement, then reward of that particular track-measurement combination (customer-item-combination) is $\ln P_{\mathrm{D}}^t + \ln l^{t,a_k^t}$. If it is a dummy measurement, then it is $\ln(1-P_{\mathrm{D}}^t)$. Let us now recall the at-most-one assumptions. Every proper measurement can be claimed by at most one track, and every track can only claim one proper measurement. This fits neatly with the auction concept. However, we have so far only used a single dummy measurement for tracks t which are not detected according to a_k. To make this conform with the auction framework, we must define a separate dummy measurement for each track. We can then arrange all the customer-item pairs in a reward matrix. For the example illustrated in Figure 8.1 (with 3 tracks and 4 measurements) it may look like this

$$\begin{bmatrix} \ln P_{\mathrm{D}}^1 + \ln l^{1,1}/\lambda & \ln P_{\mathrm{D}}^2 + \ln l^{2,1}/\lambda & -\infty \\ -\infty & \ln P_{\mathrm{D}}^2 + \ln l^{2,2}/\lambda & \ln P_{\mathrm{D}}^3 + \ln l^{3,2}/\lambda \\ \ln P_{\mathrm{D}}^1 + \ln l^{1,3}/\lambda & -\infty & -\infty \\ -\infty & \ln P_{\mathrm{D}}^2 + \ln l^{2,4}/\lambda & -\infty \\ \ln(1-P_{\mathrm{D}}^1) & -\infty & -\infty \\ -\infty & \ln(1-P_{\mathrm{D}}^2) & -\infty \\ -\infty & -\infty & \ln(1-P_{\mathrm{D}}^3) \end{bmatrix} = \begin{bmatrix} -5.69 & 5.37 & -\infty \\ -\infty & -3.80 & 6.58 \\ 4.78 & -\infty & -\infty \\ -\infty & 5.36 & -\infty \\ -0.46 & -\infty & -\infty \\ -\infty & -0.52 & -\infty \\ -\infty & -\infty & -0.60 \end{bmatrix}$$

Notice that the value $-\infty$ is used for all customer-item combinations that are impossible due to validation gating. However, for practical implementation it may be preferable to use a large negative number, say -10000, to ensure that the problem remains solvable even if no proper solution exists. Also notice that the last 3 rows constitute a 3×3 diagonal matrix: Among the dummy measurements, each track can only claim its own dummy measurement. The optimal assignment problem can then be formulated as chosing one row in each column of the reward matrix so that the sum of the elements is maximized.

Pseudocode for the Auction method is given in Algorithm 5. Notice that the price increase includes an additional parameter ε. This is typically a fairly small number larger than zero. Its inclusion is necessary to guarantee that the algorithm will converge. A larger ε may lead to faster convergence, but will also increase the risk that the optimal solution is overlooked. While it is possible to update ε in an adaptive manner, the author's experience is that it is safe to use a constant value of ε, say 0.001. In the vast majority of cases, the auction method is capable of converging without relying on ε.

8.3.3 Extracting the M best hypotheses using Murty's method

Recall that our goal in this chapter is to study efficient JPDA implementations, and not to merely find the best association hypothesis. Since the JPDA in principle involves marginalization over all association hypotheses, we want a method capable of discovering all association hypotheses with significant probabilities. The full marginalization can then be approximated by the marginalization

Algorithm 5 The Auction algorithm

1: **procedure** AUCTION($\mathbf{A}$) ▷ Reward matrix $\mathbf{A}$ is of size $(m+n) \times n$
2: `unassigned queue` $\leftarrow$ customers $1:n$
3: `prices` $\leftarrow \mathbf{0}_{1\times m}$
4: **while** unassigned customers exist **do**
5: $t^* \leftarrow$ pop first unassigned customer in `unassigned queue`
6: **for** each customer **do**
7: Register his preferred item i that maximizes value, i.e., reward minus price
8: **end for**
9: $i^* \leftarrow$ preferred item of customer t^*
10: Take item i^* from its current customer and give it to t^*
11: Add the customer who had i^* to `unassigned queue`
12: Find the values, i.e., rewards minus prices, for all items available for customer t^*
13: $y \leftarrow$ how much customer t^* gains from chosing best item over next best item
14: Increase the price of item i^* corrrespondingly $+\varepsilon$
15: **end while**
16: **end procedure**

over this selected set of association hypotheses. Such a set can be found by searching for the M best association hypotheses. The standard method for making this search is known as Murty's method. Murty's method starts with the original assignment problem and its optimal solution as found by, e.g., auction. This problem-solution pair becomes the first element in a list of problem-solution pairs. During each iteration, Murty's method removes the best problem-solution pair from the list, and makes its individual track assignments impossible one by one, so that new assignment problems and corresponding solutions are generated and added to the list. The method continues working until the list is empty or M solutions have been generated.

Pseudocode for Murty's algorithm is given in Algorithm 6. Notice that on line 14 the method attempts to solve the modified assignment problem whether it has a solution or not. For this to work, infeasible track-measurement pairs must be given a large negative reward value and not the value $-\infty$, as discussed in the previous section. We can then easily check whether the resulting solution is valid or not by checking its score. An alternative could be to check whether the modified problem is solvable before solving it. This could be done using, e.g., the Hopcroft-Karp method for maximal matchings. However, this is likely to incur a greater computational cost than the direct solution approach.

8.4 Strenghts and weaknesses of the JPDA

The JPDA is a natural first choice among multi-target methods because it makes a sensible compromise between efficiency and robustness. On the one hand, it is of modest complexity, both with regard to implementation and computational demands. A JPDA that uses the runtime enhancements from Sections 8.3.2 and 8.3.3 should be capable of running in real time on any modern laptop. For more advanced tracking methods (e.g., the multi-scan methods that we shall get to know in the next chapter) dedicated servers or cloud computing, or very drastic simplifications, may be required to reach real-time performance. On the other hand, it provides reliable uncertainty estimates. In particular, if we compare the JPDA with a basic GNN approach, that only retains a single association hypothesis, it is clear that the track covariances of that single hypothesis will under-represent the actual uncertainty, while the JPDA will provide the correct covariances since it builds on moment-matching.

However, while the JPDA is the closest one gets to a standard tracking method, it should

Algorithm 6 Murty's method

procedure MURTY($\mathscr{A}_p$, N) ▷ $\mathscr{A}_p$ is the original assignment problem
  $\mathscr{A}_s \leftarrow$ solve the original assignment problem $\mathscr{A}_p$ using Auction
  $\mathscr{L} \leftarrow [\mathscr{A}]$
  $\mathscr{R} \leftarrow \emptyset$
  while $i \leq N$ **and** $\mathscr{L}$ is non-empty **do**
    $\mathscr{M} \leftarrow$ Best problem-solution pair in $\mathscr{L}$
    $\mathscr{R} \leftarrow \{\mathscr{R}, \mathscr{M}_s\}$
    Remove $\mathscr{M}$ from $\mathscr{L}$
    if We have N solutions in $\mathscr{R}$ **then**
      break
    end if
    for each track **do**
      $\mathscr{Q}_p \leftarrow$ The problem $\mathscr{M}_p$ with current track's assignment made impossible
      $\mathscr{Q}_s \leftarrow$ solve $\mathscr{Q}_p$ using Auction
      if $\mathscr{Q}_s$ is a valid solution **then**
        $\mathscr{L} \leftarrow [\mathscr{L}, \mathscr{Q}]$
      end if
    end for
  end while
end procedure

never be thought of as a gold standard. Like any other practical tracking method, it suffers from weaknesses that may or may not have a serious impact on performance. The most important weaknesses are perhaps its single-scan limitations, track coalescence and lack of track initiation facilities.

We recall that moment matching is used after each estimation cycle in the JPDA to enforce the one-Gaussian-per-track assumption M9 from page 129. This process causes an inevitable loss of information. This means that the JPDA is a *single-scan* method which only processes one sensor scan at a time. One would intuitively expect that *multi-scan* methods that process several sensor scans simultaneously could attain better performance than the JPDA. The single-scan nature of the JPDA means that misassociations cannot be amended at a later time. While the JPDA attempts to mitigate this by hedging over different association hypotheses, the hedging probabilities may nevertheless be misleading. Sooner or later, any tracker is bound to lose track due to such occurences, and this may happen earlier for the JPDA than for a more optimal multi-scan tracker. How fast it will happen depends on system parameters such as clutter intensity and process noise. The safe region for the JPDA in this space will be larger than for GNN, but smaller than for multi-scan trackers.

The fact that the JPDA is based on averaging leads to a problem known as track coalescence. Consider the scenario depicted in Figure 8.4. Here, two kayaks are paddling together in the Trondheim City Canal. In the left-hand scan, there are separate tracks on the two kayaks. In the right-hand scan, the tracks on the two kayaks have merged so that the state estimate ends up between the two kayaks. This is clearly undesirable, but it is understandable that the JPDA returns this as its output. When the kayaks are close, both tracks will hedge equally much on both measurements, and both state estimates end up in between.

Versions of the JPDA that mitigate or avoid track coalescence have been developed by several authors. A common approach, which is used in [13], is to introduce elements of hypothesis pruning so that the modified JPDA is forced to make a decision. More advanced mixture reduction

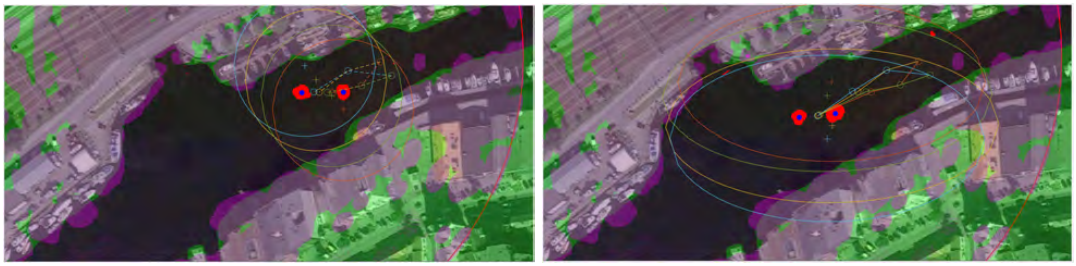


Figure 8.4: Example of track coalescence in radar tracking of kayaks in the Trondheim City Canal. Courtesy of [86].

techniques, such as methods based on variational inference [114], have been proposed to resolve track coalescence in more recent years. Finally, it may also be possible to mitigate track coalescence by sacrificing track continuity: Such a tracking method will not care about which kayak is which kayak, it only cares about where the kayaks are in the state space.

In the author's experience, track coalescence is of limited importance in the real world. The scenario from Figure 8.4 is somewhat artificial: The radar had a sampling time of 20 s, so that the uncertainty contributions from the process noise became rather excessive, as reflected by the large covariance ellipses in Figure 8.4. If the sampling time had been shorter, track coalescence would probably have resolved much faster, caused by the assymetry of noises and perturbations. However, a similar phenomena can often be seen in real world tracking systems, including Figure 8.4: Several tracks do eventually end up following the same target. This is something that often happens when a parallellized single-target PDAF-based system is used. Using a multi-target JPDA instead tends to mitigate this problem. Also, in situations where track coalesence hypothetically could happen, the biggest problem tends to be track swaps: The JPDA manages to maintain tracks on both targets, but they are the wrong tracks. Whether the swapping of identities is problematic or not will depend on the application (e.g., confusing a passenger jet with a bomber in anti-aircraft system). However, also in systems where identities are unimportant, a track swap is likely to come together with poor state estimates, at least temporarily, which could be problematic (e.g., in a collision avoidance system).

We have already discussed the lack of track initiation capability of the PDAF and the ad-hoc solution of M/N rules in Section 7.5.3. We also studied the more sophisticated approach of establishing and terminating tracks based on existence probabilities in the form of the IPDA in Section 7.6. It is also fairly straightforward to include existence probabilities in the JPDA. This leads to the joint integrated probabilistic data association (JIPDA). The derivation of the standard JIPDA method is left to the reader as Exercise 8.2.

8.5 Performance measures

To verify a tracking system one needs performance measures that can be used to assess whether the system has sufficient reliability and accuracy. For pure state estimation, which was studied in Chapters 4 and 5, we typically use variations of RMSE and normalized error measures (consistency measures) to assess performance. For multi-target tracking, these measures remain important, but additional measures are needed to provide a meaningful picture of performance.

In a tracking system, the biggest concern is track loss, followed by track swaps. Thus, the two most important performance measures are rates of track loss and track swaps. Any tracking system will inevitably lose all tracks after a long enough time, but the question of how fast it happens is important. In most applications, it is better to have a tracking method with high RMSE and low track loss probability, than the opposite situation. Of course track-loss and high RMSE are related,

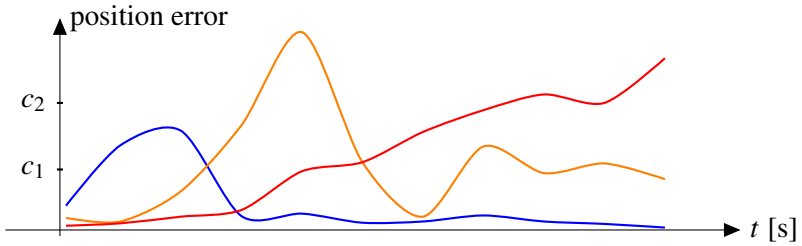


Figure 8.5: Examples of the track-loss concept from [18]. The red and orange tracks are considered lost at the end of the scenario, but the blue track is not considered lost.

but this also raises the question of whether RMSE should be evaluated over lost tracks at all. The presence of lost tracks in a set of Monte-Carlo simulations will blow up the RMSE so much that one only ends up with an indirect measure of track loss, and not a measure of how accurate the method is when it actually works.

A standard definition of track loss has never been established, but different definitions have been used in the literature. The author has found that a description of track-loss as a two-stage process can be quite intuitive [18] [19] [20]. According to this definition, we consider the track tentatively lost at time step k when its position error exceeds some threshold c_1. If the error later goes below c_1, the lost label is removed. On the other hand, if the error never manages to go below c_1 again, we consider it lost at time k. If the error exceeds a higher threshold $c_2 > c_1$, we immediately consider it lost at time k, irrespectively of whether the error later goes below c_1.

For complete tracking systems, including track initialization and termination, many more performance measures can be used, and many of these can be found in Matlab's sensor fusion toolbox. These are not directly relevant to the JPDA, but we nevertheless take an overview over the most important measures here, as awareness of these measures is important in order to understand the considerations involved in designing a complete multi-target tracking system. The performance of the track initialization process is a tradeoff between initialization time and track probability of detection (TPD) on the one hand, and track false alarm rate (TFAR) on the other hand. TPD is defined as the number of targets tracked divided by the total number of targets, while TFAR is defined as the average number of false tracks per time step [82]. Another measurement of track loss is the track fragmentation rate (TFR), defined as the average number of unique tracks required to estimate the trajectory of one specific target [82]. Just as important as these performance measures is the execution time of the tracking method. The requirement to execution time can be very different depending on whether the tracking method is to be used for post processing or for real-time implementation. Also, if a Matlab prototype has a given execution time, one would expect that an optimized C++ implementation of the same method can achieve a much better execution time.

8.6 References and chapter remarks

The JPDA dates from 1983, when it was proposed by Fortmann, Bar-Shalom & Scheffe in [36]. The description found in Bar-Shalom's "yellow book" [3] from 1995 is highly recommended, and a similar description can be found in the more recent version of the book [6]. In the same way as for the PDAF, extensions towards multiple models and existence probabilities exist, leading to methods known as IMM-JPDA and JIPDA [77]. These can of course again be combined into an IMM-JIPDA [80]. Several variations of IMM-JPDA have been published. The reader is recommended to consult [80] or [106] before other references.

JPDA-type methods have been employed and studied in a variety of applications, such as

maritime collision avoidance [96], space situational awareness [91], sonar tracking of divers [89], tracking in infrared images [98] and autonomous navigation [30]. The reader who digs into these papers will soon find out that most of the verifications have been made on simulated data. This reflects a typical challenge for the research community in target tracking. Few researchers are actually working with real data. This has three reasons: Real data are expensive to record, real data (when recorded as part of defence-related activities) are typically classified, and simulations are often needed anyway to generate the challenging scenarios for which multi-target trackers actually are needed.

Techniques to speed up and bypass the exponential complexity of hypothesis enumeration include the graphical model solutions of [51] and [117]. Many simpler and more heuristic solutions have been proposed, see [44], [2]

8.7 Exercises

Exercise 8.1 How many association hypotheses are there in total in the scenario depicted in Figure 8.1? How many of these do not involve any misdetections? ■

Exercise 8.2 The JIPDA is a generalization of the JPDA where each target may or may not exist with some probability [80]. It is the multi-target version of the IPDA that we encountered in Section 7.7. In this exercise we shall establish the key formulas for the JIPDA and study how it differs from the JPDA.

a) Assume that each track t has a predicted existence probability $r^t_{k|k-1}$. In addition, the assumptions used in Theorem 8.2.1 apply. Based on this, the JIPDA calculates association probabilities for the same set of association events as the JPDA, but the probabilities will be slightly different. Show that the association probabilities for the JIPDA are

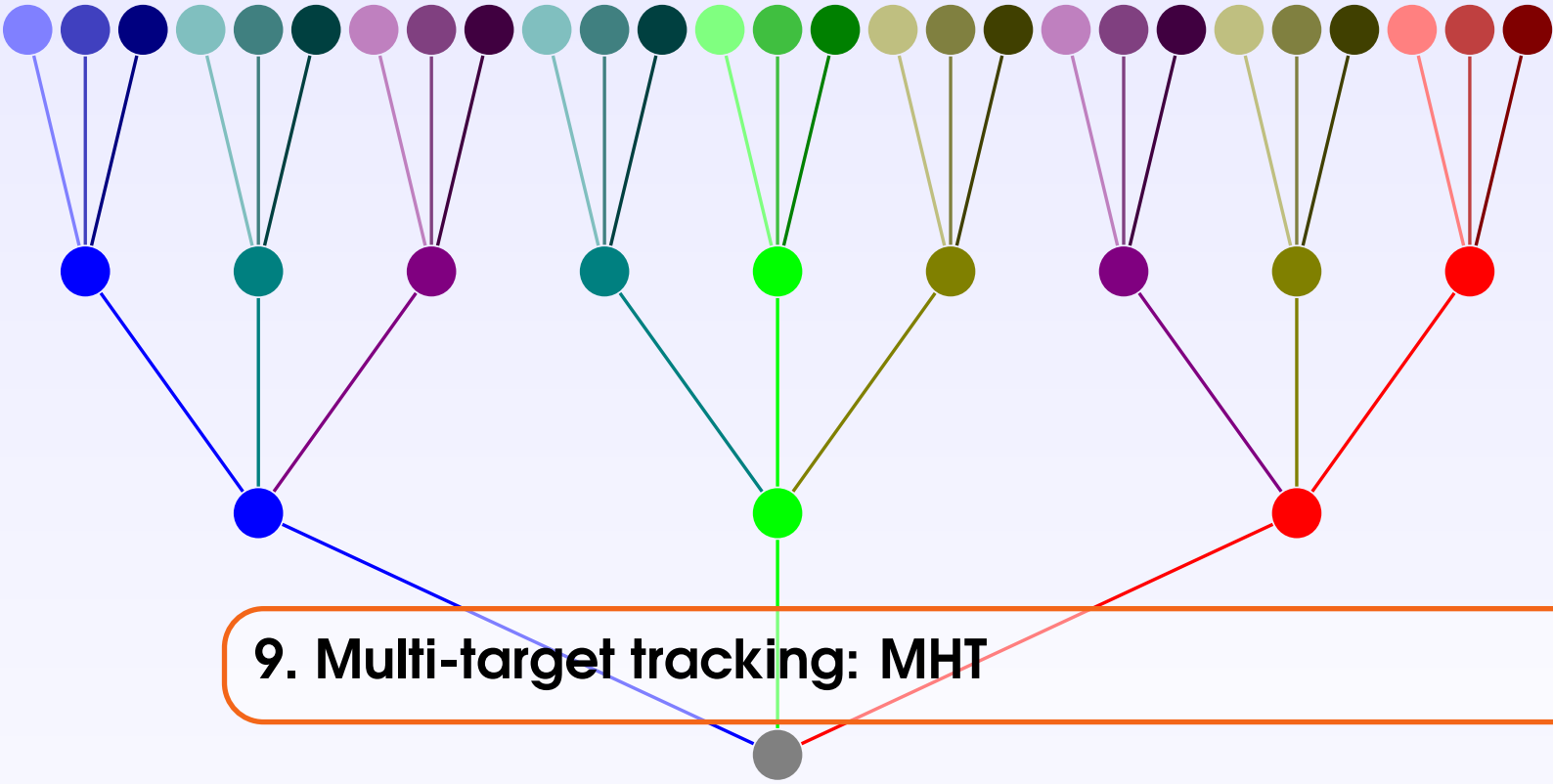
$$\Pr\{a_k|Z_{1:k}\} \propto \prod_{t:a_k(t)=0} (1 - r^t_{k|k-1} P^t_{\mathrm{D}}) \prod_{t:a_k(t)>0} r^t_{k|k-1} P^t_{\mathrm{D}} \, l^{t,a_k(t)}. \tag{8.21}$$

Hint: You only need to change one of the equations in the proof of Theorem 8.2.1. Explain which equation, and justify the change that you make.

b) Show that the results from **a)** together with (7.34) yield the the IPDA/PDAF association probabilities given in Corollary 7.4.3 when maximally one target is present.

c) Write expressions for the posterior existence probability and kinematic density for each target (before mixture reduction).

■



9. Multi-target tracking: MHT

In the JPDA we studied 2-dimensional assignments between tracks and measurements. The key principle underlying the JPDA was to approximate the joint density of the target states as a product of Gaussians during every filtering cycle. In other words, the JPDA restricts its memory to a single scan, and we call it a single-scan method.

To improve performance a reasonable strategy is therefore to process multiple scans simultaneously. The archetypical multi-scan method is the multiple hypothesis tracker (MHT). The key idea of the MHT is to evaluate the probabilities of all feasible data association hypotheses over multiple scans. While the underlying mathematics is very similar to the JPDA, the practical impact of this principle is that the MHT behaves in a manner very different from the JPDA. While the JPDA merges different hypotheses together, MHT splits its parent hypotheses into several children hypotheses every time a new scan is received. Obviously, the number of hypotheses will expand very rapidly. It is therefore necessary to prune hypotheses with low probabilities rather aggressively. The goal in an MHT is typically to find the best hypothesis, and only present this to the operator as the output of the tracking method. Thus, one can think of MHT as representing MAP estimation, while JPDA is more related to MMSE estimation.

MHT implementations are complex beasts. The mathematical evaluation of hypothesis probabilities relies on subtle combinatorial arguments. The potential for errors is absolutely present, and careful definitions of all the building stones are needed to ensure correctness. To make the implementations efficient, sophisticated search techniques and data structures are needed.

This chapter is devided into four sections. In the first section, we discuss the original version of the MHT, which was developed by Reid in [87]. This is known as Reid's MHT or hypothesis-oriented MHT (HO-MHT). We will not merely repeat Reid's derivations, but instead view Reid's 40 year old MHT through the lens of modern tracking theory. In the second section, we will look at an alternative version of MHT, known as track-oriented MHT (TO-MHT). This approach builds on realizing that we do not need to enumerate all possible association hypotheses in order to find the most likely hypothesis. It can be shown that the hypothesis search problem is equivalent to classical NP-complete problems, and established integer linear programming techniques can therefore be used in the hypothesis search. We also refer to this as multi-dimensional assignment (MDA). In

the third section, we will study a tracking method known as the Poisson Multi-Bernoulli Mixture (PMBM) filter, which currently is emerging as the *de facto* gold standard for multi-target tracking. The PMBM filter is closely related to Reid's MHT, but it is both more general and more efficient. Finally, the fourth section of this chapter is devoted to a brief survey of the many other multi-scan tracking methods that exist.

9.1 Reid's MHT

9.1.1 Multi-scan association hypotheses

The key building stones in Reid's MHT are multi-scan association hypotheses. Before delving into the assumptions and mathematical derivations of the MHT it is useful to have firm concept of what these entities account for. An important pitfall here is the distinction between a target and a track. While we generally want to distinguish different targets from each other (one could be friend and one could be foe), the targets do not come with *a priori* identities. We can only say something about their identities after having observed the targets. For example, after receiving two measurement scans, both with two measurements, we can hypothesize that the target that caused the first the measurement at $k = 1$, also caused the first measurement at $k = 2$, or we can make the alternative hypothesis that the target that caused the first the measurement at $k = 1$, also caused the second measurement at $k = 2$. Based on this way of thinking, we can define a track as a sequence of measurements.

Definition 9.1.1 — Track. Assume that we are given k consecutive scans of measurements: $Z_1 = \{\mathbf{z}_1^1, \dots, \mathbf{z}_1^{m_1}\}$, ..., $Z_k = \{\mathbf{z}_k^1, \dots, \mathbf{z}_k^{m_k}\}$. A track b^t is then a vector $b^t = [i_1, \dots, i_k]^\mathsf{T}$ where $i_l \in \{0, \dots, m_l\}$ for each $l \in \{1, \dots, k\}$.

The association hypotheses of the MHT must then for each measurement account for whether that measurement is part of a particular track, or perhaps not part of any track. We shall look at four different structures that can be used to represent association hypotheses. To make this discussion more concrete, let us take a look at the simple example in Figure 9.1.

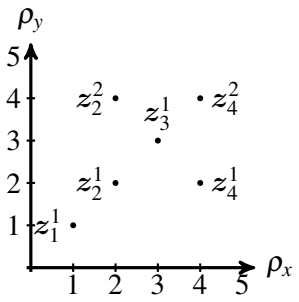


Figure 9.1: A simple scenario for illustration of the MHT.

Here we have observed a total of 6 measurements over 4 scans. It seems fairly convincing that a target has been moving upwards towards the right. It also seems plausible that another target has been moving downwards towards the right. However, it is not clear which of these targets have generated the measurement $\mathbf{z}_3^1$. Furthermore, if this measurement indeed was generated by the first target, then we have only two measurements left for the second target, and perhaps these could just as well have been generated clutter. The collection of association hypotheses must account for all such possibilities.

Association hypotheses as sets of tracks

The most direct and modern representation of an association hypothesis is as a *set of tracks*. Thus, for the scenario illustrated in Figure 9.1, some plausible association hypotheses are

$$b^{(1)} = \left\{ \begin{bmatrix} 1 \\ 1 \\ 1 \\ 2 \end{bmatrix} \right\}, \; b^{(2)} = \left\{ \begin{bmatrix} 1 \\ 1 \\ 0 \\ 2 \end{bmatrix}, \begin{bmatrix} \text{ø} \\ 2 \\ 0 \\ 1 \end{bmatrix} \right\}, \text{ etc} \tag{9.1}$$

According to the hypothesis $b^{(1)}$ only one target is present in Figure 9.1. That target generated $\mathbf{z}_1^1$ at $k = 1$, it generated $\mathbf{z}_2^1$ at $k = 2$, and so on. According to the hypothesis $b^{(2)}$ two targets are present in Figure 9.1. The first track in $b^{(2)}$ is identical to the track in $b^{(1)}$, except from the a misdetection (represented by a 0) at $k = 3$. The second track in $b^{(2)}$ represents a target which generated the measurements $\mathbf{z}_2^2$ and $\mathbf{z}_4^1$ at $k = 2$ and $k = 4$, respectively. We use the letter ø to signify that this target did not exist, or more precisely, was not considered to exist, at $k = 1$.

Association hypotheses as matrices

It is straightforward to construct matrices consisting of the tracks in $b^{(1)}$ or $b^{(2)}$ as column vectors. This leads to the alternative representation

$$b^{(1)} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 2 \end{bmatrix}, \; b^{(2)} = \begin{bmatrix} 1 & \text{ø} \\ 1 & 2 \\ 0 & 0 \\ 2 & 1 \end{bmatrix}, \text{ etc .} \tag{9.2}$$

When we do this, it is important to be aware that all *permutations of the columns vectors in a given hypothesis constitute the same hypothesis*. In more mathematical terms, an association hypothesis is an equivalence class of such matrices under the equivalence relation of permutation-invariance.

It is often convenient to view such matrices as mappings. That is, we can view an association hypothesis b as a mapping

$$b : \{1, \ldots, n\} \to \{\text{ø}, 0, \ldots, m_1\} \times \ldots, \times \{\text{ø}, 0, \ldots, m_k\}. \tag{9.3}$$

Here b is a function which takes the index of a given target as argument, and outputs a column vector consisting of the its measurements at the time steps 1 to k. Here n is the number of targets assumed to exist by b. This enables several useful notations. For example, we can use b_k^t to denote the measurement claimed by track t according to b at time k, and b^t to denote the entire track. Again, it must be emphasized most strongly that all permutations of this mapping constitute the same hypothesis.

Association hypotheses as track-oriented trees

The standard way of implementing MHT is by means of track trees. To build the track tree, or more correctly track forest, we establish root nodes in each and every measurement. For every root node, we generate a new branch for each feasible measurement, including misdetection, in the next scan. And so we proceed through the subsequent scans. At a given scan, an association hypothesis can be generated by picking maximally one leaf node from each track tree, in such a way that no measurement index except misdetection is repeated within any of the scans. This construction is illustrated in Figure 9.2. It is worth noticing that a new track tree is initialized for every measurement, and also that the zeroth measurement (misdetection) generates a new branch for all parent branches at all time steps. Both observations are reasons to be concerned over computational complexity.

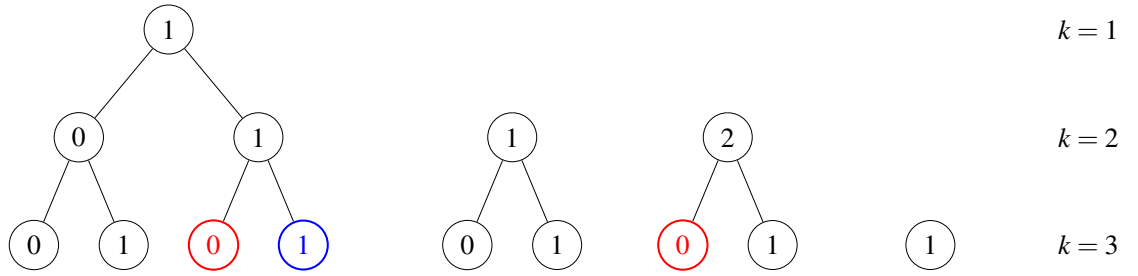


Figure 9.2: The track forest of the example in Figure 9.1 at $k = 3$, under the assumption that $\mathbf{z}_2^2$ does not gate with the track initialized in $\mathbf{z}_1^1$. The two example hypotheses from (9.1) are marked by blue and red leaf nodes, respectively.

9.1.2 The assumptions of Reid's MHT

The basic principle of the HO-MHT is to calculate hypothesis probabilities in a recursive manner. For a given parent hypothesis at time step $k-1$, we identify all possible children hypotheses at time step k, and calculate their probabilities. This calculation is based on the probability of the parent hypothesis, the likelihood of the measurements, and prior knowledge about births, clutter, detection probabilities etc. The calculation is based on a Bayesian model, which means that assumptions need to be specified. Reid's original MHT used the following assumptions, loosely translated to our notation:

- R1 The state vector is $\mathbf{x} = [x, y, v_x, v_y]^\mathsf{T}$.
- R2 New targets appear according to a homogeneous Poisson process with intensity $P_\mathrm{D} v$, uniform over $x, y \in \Omega \subset \mathbb{R}^2$.
- R3 The motion of a target is given by $f_\mathbf{x}(\mathbf{x}_k|\mathbf{x}_{k-1}) = \mathcal{N}(\mathbf{x}_k; \mathbf{F}\mathbf{x}_{k-1}, Q)$.
- R4 A target with state $\mathbf{x}_k$ generates a measurement $\mathbf{z}_k$ with probability P_D.
- R5 The measurement of a detected target is related to the state according to $f_\mathbf{z}(\mathbf{z}_k|\mathbf{x}_k) = \mathcal{N}(\mathbf{z}_k; \mathbf{H}\mathbf{x}_k, R)$ where $\mathbf{H} = [\mathbf{I}_2, \mathbf{0}_{2\times 2}]$.
- R6 Clutter measurements occur according to a homogeneous Poisson process with intensity λ.
- R7 The velocity of a newborn target is distributed according to $\mathcal{N}(\mathbf{0}, \mathbf{P}_v)$.

Comparison with the general multi-target assumptions M1-M6 on page 129 reveal that Reid's model involves several simplifications in comparison to the standard model. Most striking is perhaps the restriction to Gaussian linear models in assumptions R4 and R5. Furthermore, the Poisson processes in R2 and R6 are assumed homogeneous, i.e, with constant intensities v and λ. The particular choice state vector in R1 is necessary to follow Reid's derivations as close as possible. All these restrictions can of course be relaxed, leading to more general formulations of the MHT.

Another subtle distinction between the standard model and Reid's model is that the birth intensity μ from M1 has been replaced with the new target intensity $P_\mathrm{D} v$ in R2. The different wording is intentional. In R2, the model describes targets which physically enter the surveillance region, e.g., a boat crossing the perimeter of the radar FOV, an airplane taking off from an airport, etc. In M1, the model describes targets which are observed for the first time. The fact that such a target necessarily is detected, is emphasized by using the notation $P_\mathrm{D} v$ as opposed to simply v. In Chapter 13, we shall see that v is the intensity of a Poisson process of so-called unknown targets, i.e., targets which have never been seen. This will also provide a mathematical justification for including P_D in the new target intensity.[1] This material is postponed to Chapter 13 because it depends on the modern theory of random finite sets, which is beyond the scope of TTK4250, and instead will be covered in TK8102.

[1] Reid wrote in [87] that P_D had been already been included in the new target density. Factoring Reid's version into P_D and v makes it possible to equate v with the unknown target intensity of the PMBM filter [14].

Finally, we also notice that there is no death process in Reid's MHT. In other words, any target that exists at some time, is supposed to exist indefinitely. This does not necessarily mean that no tracks will be terminated: Any hypothesis which fails to accumulate measurements over time will get very low posterior probabilities, and eventually it will only make sense to prune away such hypotheses.

9.1.3 State estimation in the MHT

As was the case for the JPDA, the workflow of the MHT also consists of state estimation and data association. The state estimation is more straightforward than in the JPDA, at least in a plain-vanilla MHT, since there is no hypothesis merging involved. Under the Gaussian-linear assumptions in R2 and R5 the state estimation boils down to Kalman filtering. Two concerns need to be adressed, however. First, it is important to organize the state estimation in an orderly fashion due to the large number of candidate tracks. Second, track initialization is now included and needs special treatment, especially because the new target assumption R1 violates Gaussianity.

It can be useful to classify tracks in the following classes, since their Kalman filter update formulas will differ:

$$\begin{aligned} B(b) &= \{t \text{ such that } b^t_{1:k-1} = \emptyset_{k-1\times 1} \text{ and } b^t_k > 0\} \\ M(b) &= \{t \text{ such that } b^t_k = 0 \text{ and } b^t_l > 0 \text{ for some } l < k\} \\ D(b) &= \{t \text{ such that } b^t_k > 0 \text{ and } b^t_l > 0 \text{ for some } l < k\}. \end{aligned} \tag{9.4}$$

The set $D(b)$ consists of all previously existing tracks whose target *is* detected at k according to b. The set $M(b)$ consists of all previously existing tracks whose target *is not* detected at k according to b. The set $B(b)$ consists of all new tracks according to b. All such targets are by definition detected at k.

For tracks in $D(b)$ we use the standard Kalman filter formulas. By referring to Algorithm 1, track t is in this case given by

$$[\hat{\mathbf{x}}^{b^t}_k, \mathbf{P}^{b^t}_k, \hat{\mathbf{x}}^{b^t}_{k|k-1}, \mathbf{P}^{b^t}_{k|k-1}, \mathbf{S}^{b^t}_k] = \mathrm{KF}(\hat{\mathbf{x}}^{b^t_{1:k-1}}_{k-1}, \mathbf{P}^{b^t_{1:k-1}}_{k-1}, \mathbf{z}^{b^t_k}_k) \tag{9.5}$$

Of course, only $\hat{\mathbf{x}}^{b^t}_k$ and $\mathbf{P}^{b^t}_k$ are required to construct the hypothesis-conditional kinematic pdf, but $\mathbf{S}^{b^t}_k$ also plays a role in the evaluation of hypothesis probabilities.

For tracks in $M(b)$ we have not received any measurement, and we are thus left with $\hat{\mathbf{x}}^{b^t}_{k|k-1}$ and $\mathbf{P}^{b^t}_{k|k-1}$, which are given by standard Kalman filter prediction formulas just as for tracks in $D(b)$.

For tracks in $B(b)$ things become tricky because the homogeneous Poisson process in assumption R1 implies that the prior positional pdf of new targets is uniform, thus violating Gaussianity. However, if the measurement covariance is sufficient small, then the kinematic pdf of a newborn target will still be approximately Gaussian. More precisely it will have expectation and covariance given by

$$\hat{\mathbf{x}}^{b^t}_k = \begin{bmatrix} \mathbf{z}^{b^t_k}_k \\ \mathbf{0} \end{bmatrix} \quad \text{and} \quad \mathbf{P}^{b^t}_k = \begin{bmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{0} & \mathbf{P}_v \end{bmatrix}. \tag{9.6}$$

The derivation is left to the reader as Exercise 9.1. This concludes our treatment of state estimation in Reid's MHT.

9.1.4 Data association in the MHT

The probability of a data association hypothesis depends on how well the hypothesis describes the given data. This depends on how well the tracks from the parent hypothesis fit the measurements

assigned to them by the current hypothesis. It also depends on how many measurements are believed to be false alarms or from new targets, compared to the expected amount of false alarms and new targets. The elegance of the MHT largely lies in the fact that these considerations can be put together in one simple formula. Let us first define the cardinality numbers δ, β, φ and N_{TGT} according to

$$\begin{aligned}\delta &= |D(b)| \\ \beta &= |B(b)| \\ \varphi &= m - \delta - \beta \\ N_{\text{TGT}} &= |D(b)| + |M(b)|.\end{aligned} \tag{9.7}$$

Here N_{TGT} is the number of targets existing according to the parent hypothesis b^*. The number of these that are detected at time k according to the current hypothesis b is denoted δ. Furthermore, β is the number of new targets, which all are detected by definition, and φ is the number of false alarms. We can then state the main result of this chapter.

Theorem 9.1.1 Assume R1-R6. Let $b_{1:k-1}$ denote the parent hypothesis of b. The posterior probability of b is then given by

$$\Pr\{b \,|\, Z_{1:k}\} \propto \Pr\{b_{1:k-1} \,|\, Z_{1:k-1}\} \lambda^{\varphi} P_{\text{D}}^{\delta} (1 - P_{\text{D}})^{N_{\text{TGT}} - \delta} (P_{\text{D}} v)^{\beta} \prod_{t \in D(b)} \mathcal{N}(\mathbf{z}_k^{b^t}; \mathbf{H}\hat{\mathbf{x}}_{k|k-1}^{b^t}, \mathbf{S}_k^{b^t}). \tag{9.8}$$

Proof. In the proof, let b^* denote the parent hypothesis from time $k-1$. For notational simplicity, we drop the time indices for the remainder of this section. We want to calculate the posterior probabilities of all current hypotheses b at time k that build on b^*. To achieve this, we define ψ as the assignment between tracks in b^* and measurements in Z_k. Then ψ and b^* together specify b, and Bayes' rule yields

$$\Pr\{b\} \propto p(Z_k \,|\, b^*, \psi) \Pr\{\psi | b^*\} \Pr\{b^*\} \tag{9.9}$$

The difficult part here is the probability $\Pr\{\psi | b^*\}$ which must be found using combinatorial arguments. The reasoning that Reid used to obtain $\Pr\{\psi | b^*\}$ was based on three successive partitions of the outcome space conditional on b^* until ψ was found. These refinements are known as the number event, the configuration event and the assignment event.

We begin with the number event, which we denote $\gamma = (\delta, \beta, \varphi)$. It specifies that δ measurements come from previously existing targets, β measurements come from new targets, and φ measurements come from clutter. The numbers are independent conditional on b^*, and so their probabilities can be multiplied together. Let N_{TGT} denote the number of previously existing targets according to b^*. The conditional probability of the number event $\gamma = (\delta, \beta, \varphi)$ is

$$\begin{aligned}\Pr\{\gamma | b^*\} &= \text{Binomial}(\delta; N_{\text{TGT}}, P_{\text{D}}) \, \text{Poiss}(\beta; vV) \, \text{Poiss}(\varphi; \lambda V) \\ &= \frac{N_{\text{TGT}}!}{\delta!(N_{\text{TGT}} - \delta)!} P_{\text{D}}^{\delta} (1 - P_{\text{D}})^{N_{\text{TGT}} - \delta} e^{-P_{\text{D}} vV} \frac{(P_{\text{D}} vV)^{\beta}}{\beta!} e^{-\lambda V} \frac{(\lambda V)^{\varphi}}{\varphi!}.\end{aligned} \tag{9.10}$$

The next refinement is the configuration event τ, which partitions the measurements into a particular set of δ measurements from existing targets, a particular set of β measurements from new targets, and a particular set of φ measurements from clutter. This partitioning of measurements is about counting the number of combinations. Conditional on γ, the number of ways that we can separate the m measurements into detections from existing targets and other measurements is

$$\binom{m}{\delta} \tag{9.11}$$

The remaining measurements can be separated into measurements from new targets and clutter measurements in

$$\binom{m-\delta}{\varphi}. \tag{9.12}$$

different ways. Since all such combinations are *a priori* equally likely, the configuration has the following probability

$$\Pr\{\tau|\gamma\} = \left[\binom{m}{\delta}\binom{m-\delta}{\varphi}\right]^{-1} = \frac{\delta!\beta!\varphi!}{m!}. \tag{9.13}$$

The third refinement, the assignment event, specifies a unique association between previously established tracks and the δ measurements that according to τ were generated by established tracks. This is about counting the number of permutations, and the conditional probability is

$$\Pr\{\psi\,|\,\tau\} = \frac{(N_{\text{TGT}}-\delta)!}{N_{\text{TGT}}!} \tag{9.14}$$

By combining (9.10), (9.13) and (9.14) we find the prior hypothesis probability according to

$$\begin{aligned}\Pr\{\psi|b^*\} &= \Pr\{\psi,\tau,\gamma|\,b^*\}\\ &= \Pr\{\psi\,|\,\tau\}\Pr\{\tau\,|\,\gamma\}\Pr\{\gamma|\,b^*\}\\ &\propto P_{\text{D}}^{\delta}(1-P_{\text{D}})^{N_{\text{TGT}}-\delta}(P_{\text{D}}vV)^{\beta}(\lambda V)^{\varphi}\end{aligned} \tag{9.15}$$

In the last line of (9.15) we have made several cancellations based on the reciprocity between (9.10), (9.13) and (9.14). Furthermore, we have removed terms, such as e^{-vV}, which are common for all association hypotheses.

Then we look at the hypothesis-conditional likelihood. Recall from assumptions R2 and R6 that the Poisson processes of new targets (i.e., targets in $B(b)$) and false alarms are homogeneous, meaning that their spatial densities over $\mathbb{R}^2$ are simply $1/V$. The *misdetected* targets in $M(b)$ do not generate measurements and are therefore of no relevance in the hypothesis-conditional likelihood. The *detected* targets in $D(b)$ give rise to Gaussian terms in accordance with the Kalman cycle (9.5). By reasoning similar to what we did for the JPDA in Section 8.2.3 (i.e., by integration over the kinematic state), we arrive at the hypothesis-conditional likelihood

$$p(Z_k\,|\,b,Z_{1:k-1}) = \frac{1}{V^{\varphi+\beta}}\prod_{t\text{ s.t. }b_k^t>0}\mathcal{N}(\mathbf{z}_k^{b_k^t};\mathbf{H}\hat{\mathbf{x}}_{k|k-1}^{b^t},\mathbf{P}_{k|k-1}^{b^t}). \tag{9.16}$$

When we multiply this by the prior probability in (9.15) the volume terms cancel. When we furthermore multiply by the parent hypothesis probability the desired result follows. ■

9.1.5 Efficient implementation of the MHT

In Section 8.3 we discussed how the auction method combined with Murtys M-best method could be used to speed up an implementation of the JPDA. The same principles can be used to enable real-time implementation of Reid's MHT. In this case, we generate maximally M hypotheses for each parent hypothesis. Pruning is then used on the complete collection of parent hypotheses to ensure that the total number of hypotheses remains manageable.

Recall that the auction method works in terms of customers and items. It is possible to design the assignment problem both with tracks as customers and the measurements as items, or the other way around. If the measurements are used as customers, then the 2D assignment problem can be formulated in terms of the following $m\times(N_{\text{TGT}}+2m)$ reward matrix

$$\begin{bmatrix}\tilde{L} & (\ln v+\ln(1-P_{\text{D}}))\,\mathbf{I}_m^* & (\ln\lambda-\ln P_{\text{D}}+\ln(1-P_{\text{D}}))\,\mathbf{I}_m^*\end{bmatrix} \tag{9.17}$$

where the element belonging to measurement i and track t in the $m \times N_{\text{TGT}}$ matrix $\tilde{L}$ is given by

$$\tilde{l}_{it} = \ln \mathcal{N}\left(\mathbf{z}_k^{b_k^t}; \mathbf{H}\hat{\mathbf{x}}_{k|k-1}^{b^t}, \mathbf{S}_k^{b^t}\right) \tag{9.18}$$

and where the notation $\mathbf{I}_m^*$ signifies a matrix which is unity for all its diagonal elements and $-\infty$ for all off-diagonal elements. This construction is similar to the JPDA reward matrix that we constructed on page 136, but different due to the choice of measurements instead of tracks as customers, and because we in Reid's MHT account for both false alarms and new targets.[2]

In a Murty-Auction implementation of MHT the auction calls are likely to be the most expensive phase of the algorithm. Having an efficient 2D assignment solver is therefore the most obvious step one can make to mitigate complexity. Following this, calculations of the likelihoods in (9.18) may also demand a fair share of computation resources. This is where track trees become useful. To avoid that redundant resources are spent on likelihood calculations, one calculates the likelihoods for tracks in the track tree. The association hypotheses should then contain points to the leaf nodes in the track tree, as opposed to copies of the individual tracks.

Finally, gating and clustering is just as important to mitigate complexity for MHT as for the JPDA. Efficient clustering for MHT is, however, not nearly as trivial as it is for JPDA. Clustering may again be based on single linkage connecting nearby tracks. All hypotheses that contain pointers to connected tracks should then be assigned to the same hypothesis cluster. Two challenges in particular arise for MHT clustering. First, it will happen now and then that two previous clusters have to merged into a single cluster. This merging means that each hypothesis in the first cluster must be combined with each hypothesis in the second cluster. This can cause serious bottlenecks. Second, this agglomeration of clusters may actually happen quite fast due to the expanding covariances and validation gates of tracks with misdetections. To counteract this it is necessary with methods for cluster splitting. However, for cluster splitting to work efficiently one has to enforce approximations, i.e., assuming that two clusters are independent when they actually are not. This is a topic of active research, which we will revisit in Chapter 13.

■ **Example 9.1 — M-best Murty hypothesis enumeration.** Let us return to the scenario depicted in Figure 9.1. In addition to the measurement locations specified by the figure, we assume the following model parameters: The surveillance region is given by $[0,5] \times [0,5]$. False alarms are generated by a Poisson process with intensity $0.04\,\text{m}^{-2}$. The targets are assumed to follow a CV model (see page 59) with process noise standard deviation $0.1\,\text{m/s}^2$. The sampling time is 1 s and the position measurements have measurement noise with standard deviation 0.1 m. We assume a constant new target intensity of $0.06\,\text{m}^{-2}$. For the sake of demonstration, we do not use any pruning or validation gating in this example.

Hypotheses at $k = 1$: Since we only have received one measurement at $k = 1$, we have two possible hypotheses

$$b_1^{(1)} = \begin{bmatrix} 1 \end{bmatrix}, \; b_1^{(2)} = \begin{bmatrix} \end{bmatrix}.$$

These have probabilities 0.51 and 0.49. Notice that the new target intensity and false alarm intensity are quite similar, so it should not come as a surprise that the hypothesis of one target and the empty hypothesis score similar probabilities.

Hypotheses at $k = 2$: At the next time step we get a total of 12 hypotheses. These include 8 hypotheses generated from $b_1^{(1)}$, namely

$$b_2^{(1)} = \begin{bmatrix} 1 & \text{ø} \\ 1 & 2 \end{bmatrix}, \; b_2^{(2)} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \; b_2^{(3)} = \begin{bmatrix} 1 & \text{ø} & \text{ø} \\ 0 & 1 & 2 \end{bmatrix}, \; b_2^{(4)} = \begin{bmatrix} 1 & \text{ø} \\ 0 & 2 \end{bmatrix},$$

[2] The reader may notice that this makes the assignment problem more complicated, and wonder if this complication can be avoided. Again the answer depends on concepts that will be introduced in Chapter 13.

$$b_2^{(5)} = \begin{bmatrix} 1 & \text{ø} \\ 0 & 1 \end{bmatrix},\ b_2^{(6)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix},\ b_2^{(7)} = \begin{bmatrix} 1 & \text{ø} \\ 2 & 1 \end{bmatrix},\ b_2^{(8)} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

and 4 hypotheses generated from $b_1^{(2)}$, namely

$$b_2^{(9)} = \begin{bmatrix} \text{ø} & \text{ø} \\ 1 & 2 \end{bmatrix},\ b_2^{(10)} = \begin{bmatrix} \text{ø} \\ 2 \end{bmatrix},\ b_2^{(11)} = \begin{bmatrix} \text{ø} \\ 1 \end{bmatrix},\ b_2^{(12)} = \begin{bmatrix} \end{bmatrix}.$$

The ancestors of $b_1^{(2)}$ get a total probability mass of 0.59 (which is larger than 0.51), fairly equally distributed among $b_2^{(9)}, \ldots, b_2^{(12)}$. Thus, after two iterations, the MHT shows some reluctance to accept that any target has generated two consecutive measurements. Nevertheless, the empty hypothesis $b_2^{(12)}$ get only a probability of 0.14, so the MHT is fairly convinced that at least some of the 3 measurements hitherto observed must be from targets. It is evident that only after 2 time steps and 3 measurements the growth in the number of hypothesis formidable.

Hypotheses at $k = 3$: The number of hypotheses has now grown to 42. As it is meaningless to enumerate all of these here we will restrict attention to the children of the hypotheses that from an intuitive perspective perhaps is most plausible at $k = 2$, namely $b_2^{(1)}$. The reward matrix, as defined in (9.17), becomes

$$\begin{bmatrix} 0.87 & -3.48 & -4.01 & -4.062 \end{bmatrix}. \tag{9.19}$$

Recall that a single measurement, $\mathbf{z}_3^1$, is observed at $k = 3$. The first element in the reward matrix is the reward of assigning the first track in $b_2^{(1)}$ to the measurement. The second element is the reward of assigning the second track in $b_2^{(1)}$ to the measurement. The third element is the reward of assigning a newborn track to the measurement. The fourth element is the reward of assigning the label false alarm to the measurement. Since there is only one row in the reward matrix these possibilities correspond to 4 hypotheses at $k = 3$, namely

$$b_3^{(1)} = \begin{bmatrix} 1 & \text{ø} \\ 1 & 2 \\ 1 & 0 \end{bmatrix},\ b_3^{(2)} = \begin{bmatrix} 1 & \text{ø} \\ 1 & 2 \\ 0 & 1 \end{bmatrix},\ b_3^{(3)} = \begin{bmatrix} 1 & \text{ø} & \text{ø} \\ 1 & 2 & \text{ø} \\ 0 & 0 & 1 \end{bmatrix},\ b_3^{(4)} = \begin{bmatrix} 1 & \text{ø} \\ 1 & 2 \\ 0 & 0 \end{bmatrix}.$$

What can we say about hypothesis probabilities? Successors of b_1^1 have now massively increased their proportion of the probability mass to 0.86. Thus, after observing 3 measurements on a row the MHT is suddenly much more convinced that a target did actually show at $k = 1$ and generating $\mathbf{z}_1^1$.

Hypotheses at $k = 4$: The number of hypotheses has now increased to 602. Of course, these are not equally probable. 99% of the probability mass is concentrated in the topmost 19 hypotheses, and 90% of the probability mass is claimed by the top 4 hypotheses. Let us take a look at the top 5 hypotheses:

$$b_4^{(1)} = \begin{bmatrix} 1 & \text{ø} \\ 1 & \text{ø} \\ 1 & \text{ø} \\ 2 & 1 \end{bmatrix},\ b_4^{(2)} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 2 \end{bmatrix},\ b_4^{(3)} = \begin{bmatrix} 1 & \text{ø} & \text{ø} \\ 1 & 2 & \text{ø} \\ 1 & 0 & \text{ø} \\ 2 & 0 & 1 \end{bmatrix},\ b_4^{(4)} = \begin{bmatrix} 1 & \text{ø} \\ 1 & 2 \\ 1 & 0 \\ 2 & 0 \end{bmatrix},\ b_4^{(5)} = \begin{bmatrix} 1 & \text{ø} \\ 1 & 2 \\ 1 & 0 \\ 2 & 1 \end{bmatrix}.$$

So only with the 5th hypothesis (and many less probable hypotheses) is the MHT considering the possibility of a downwards moving target, despite its intuitive plausibility. The top 4 hypotheses are all very similar: None of them contain any tracks that are not present in $b_4^{(3)}$. This is a hint that there may be inefficiencies in Reid's MHT. The measurements depicted in Figure 9.1 could possibly have been the result of two targets moving in V-shaped patterns. The first hypothesis that considers this possibility is hypothesis number 533 when ranked according to probability. The

probability of this hypothesis is only $3 \cdot 10^{-20}$, and the total probability mass claimed by this one and less probable hypotheses is roughly twice of that. It follows that we have basically no margins for hypothesis pruning if we want to keep this option open. Of course, that is not practical, and validation gating using e.g. 4 standard deviations would anyway have eliminated the possibility of such V-hypotheses with the given model parameters. If we prune all hypotheses with probability less than 10^{-4} and use 4-sigma validation gating the number of hypotheses at the fourth time step is reduced to 68. ∎

Reid's MHT is, just as the PDAF, JPDA etc., a Bayesian method. Similarly to the IPDA, the MHT also models target existence in a Bayesian framework, although this is done in a more indirect and implicit manner than in the IPDA. There is no track-wise existence probability function, but by summing the probabilities of all the hypotheses that contain a given track, one can find the probability that a target with that track exists. This probability is governed by the false alarm intensity and the new target intensity.

∎ **Example 9.2 — Impact of new target density.** We continue with the previous example, but now we allow the new target intensity v to vary between 0.005 and 0.5. We want to study the total probability of hypotheses that contain the track $[1,1,1,2]^{\mathsf{T}}$ as a function of v. We also study the probability of the empty hypothesis, i.e., the probability that no targets are present. The results of this analysis are displayed in Figure 9.3. We see that for the given scenario we would have to

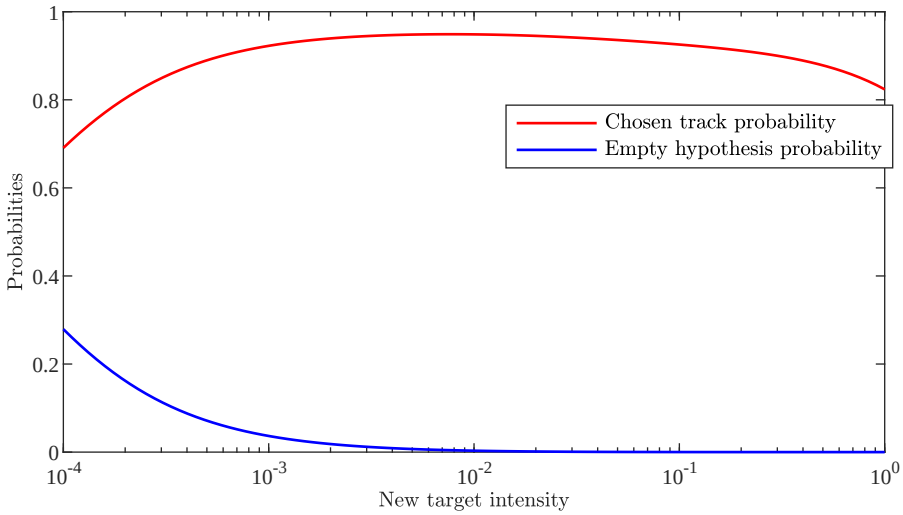


Figure 9.3: bla bla.

set v as low as 10^3 to make the MHT consider the possibility that no targets are present with any significant probability. This would correspond to a belief that new targets appear on average in every 40th scan. As for the chosen track, its accumulated posterior probability is largely insensitive to v. This demonstrates that one should not be overly concerned about having a very precise value of v. A sensible ballpark estimate (within, say, a decade) may suffice. Over 4 scans, the information contained in the data is so strong that *a priori* information is of limited importance. ∎

9.2 Track-oriented MHT

In Example 9.1 we saw indications that the hypothesis search by Murty may struggle to explore the actual variability of association hypotheses. However, it is not really necessary to enumerate all association hypotheses to explore this variability. If we are able to enumerate all the tracks that occur in the association hypotheses we have at least some representation of the possibilities.

Returning to Example 9.1, we had 602 hypotheses after 4 time steps, but the number of tracks is only 35. Based on this, versions of MHT known as track-oriented MHT (TO-MHT) aims to discover the single best association hypothesis directly from the collection of tracks.

This is fundamentally different from how Reid's MHT works, although the underlying mathematical model is more or less the same. Reid's MHT can both be viewed as recursive algorithm and as a multi-scan method. The TO-MHT is not recursive at all. It is entirely a multi-scan method that processes all the data simultaneously, or at least all the data within a sliding window, in its search for the optimal association hypothesis.

To focus on the core concepts, we shall skip the issue of new targets in this section. Thus, N_{TGT} is henceforth a constant, and Reid's formula simplifies to

$$\Pr\{b \,|\, Z_{1:k}\} \propto \Pr\{b_{1:k-1} \,|\, Z_{1:k-1}\} \lambda^{\varphi} P_{\text{D}}^{\delta} (1-P_{\text{D}})^{N_{\text{TGT}}-\delta} \prod_{t \in D(b)} \mathcal{N}(\mathbf{z}_k^{b_k^t}; \mathbf{H}\hat{\mathbf{x}}_{k|k-1}^{b^t}, \mathbf{S}_k^{b^t}). \tag{9.20}$$

9.2.1 The track split filter and the track file

To begin our discussion of the TO-MHT, let us first take a look at the concept of multi-scan *single-target* tracking. This boils down to the construction of a track tree such as in Figure 9.2, and the specification of an appropriate track score function that can be evaluated for each leaf node.

In Chapter 7 we sketched the optimal solution to single-target tracking, but only in very cursorial terms, since our main focus was to derive the PDAF. The key result was Theorem 7.4.1, where we spelled out a formula for the association probabilities without making any assumptions on the clutter density $c(\mathbf{z}_k)$, the measurement likelihood $f_{\mathbf{z}}(\mathbf{z}_k \,|\, \mathbf{x}_k)$, the predicted density $p_{k|k-1}(\mathbf{x}_k)$ or the clutter cardinality model $\mu(\varphi_k)$. To derive the PDAF, we had to assume that $c(\mathbf{z}_k)$ was uniform, that $f_{\mathbf{z}}(\mathbf{z}_k \,|\, \mathbf{x}_k)$ and $p_{k|k-1}(\mathbf{x}_k)$ were Gaussian, and that $\mu(\varphi_k)$ was diffuse or Poisson. Let us now, instead, think of $p_{k|k-1}(\mathbf{x}_k)$ as a mixture of the form

$$p_{k|k-1}(\mathbf{x}_k) = \sum_{d_{1:k-1}} \beta_{k-1}^{d_{1:k-1}} p_{k|k-1}^{d_{1:k-1}}(\mathbf{x}_k).$$

The summation variable $d_{1:k-1}$ ranges over possible descents through the track tree, and can be represented as a sequence of measurement indices, just as the columns of the Reid-style hypotheses b. Thus, each mixture component is represented by a parent node in the track tree. Making state estimates for the m_k measurements plus the possibility of misdetection leads to $m_k + 1$ children nodes for each parent, or possibly fewer if validation gating is used. Clearly, the posterior pdf at time step k will be a new mixture

$$p_k(\mathbf{x}_k) = \sum_{d_{1:k}} \beta_k^{d_{1:k}} p_k^{d_{1:k}}(\mathbf{x}_k).$$

where the conditional pdfs are found according to the usual Bayesian filtering formulas

$$p_{k|k}^{d_{1:k}}(\mathbf{x}_k) \propto \begin{cases} p_{k|k-1}^{d_{1:k-1}}(\mathbf{x}_k) & \text{if } d_k = 0 \\ f_{\mathbf{z}}(\mathbf{z}_k \,|\, \mathbf{x}_k) p_{k|k-1}^{d_{1:k-1}}(\mathbf{x}_k) & \text{if } d_k > 0 \end{cases}$$

which boil down to Kalman filter formulas under the appropriate assumptions. The update formula for the mixture weights can be taken directly from Theorem 7.4.1. Inserting the Poisson clutter model that we have used in the last two chapters gives

$$\beta_k^{d_{1:k}} \propto \begin{cases} (1-P_{\text{D}}) \beta_{k-1}^{d_{1:k-1}} & \text{if } d_k = 0 \\ l^{d_{1:k}} \beta_{k-1}^{d_{1:k-1}} & \text{if } d_k > 0 \end{cases}$$

where $l^{d_{1:k}}$ is very similar to the likelihood ratio that we used in the theorem:

$$l^{d_{1:k}} = \frac{1}{\lambda} \int f_{\mathbf{z}}(\mathbf{z}_k \,|\, \mathbf{x}_k) p_{k|k-1}^{d_{1:k-1}}(\mathbf{x}_k) \mathrm{d}\mathbf{x}_k. \tag{9.21}$$

What is missing in the above is the development from the previous posterior $p_{k-1}(\mathbf{x}_{k-1})$ to the current prediction. This is of course given by a usual Chapman-Kolmogorov formula

$$p_{k|k-1}(\mathbf{x}_k) = \sum_{d_{1:k-1}} \beta_{k-1}^{d_{1:k-1}} \int f_{\mathbf{x}}(\mathbf{x}_k \,|\, \mathbf{x}_{k-1}) p_{k-1}^{d_{1:k-1}}(\mathbf{x}_{k-1}) \mathrm{d}\mathbf{x}_{k-1}.$$

Again it is straightforward to obtain concrete expression in Gaussian-linear cases using product identity and Kalman filtering framework. If

$$p_{k-1}^{d_{1:k-1}}(\mathbf{x}_k) = \mathcal{N}(\mathbf{x}_k \,;\, \hat{\mathbf{x}}_k^{d_{1:k-1}}, \mathbf{P}_k^{d_{1:k-1}})$$

then we find the expectation and covariance of $f_{\mathbf{z}}(\mathbf{z}_k \,|\, \mathbf{x}_k) p_{k|k-1}^{d_{1:k-1}}(\mathbf{x}_k)$ by means of Algorithm 1:

$$[\hat{\mathbf{x}}_k^{d_{1:k}}, \mathbf{P}_k^{d_{1:k}}, \hat{\mathbf{x}}_{k|k-1}^{d_{1:k}}, \mathbf{P}_{k|k-1}^{d_{1:k}}, \mathbf{S}_k^{d_{1:k}}] = \mathrm{KF}(\hat{\mathbf{x}}_{k-1}^{d_{1:k-1}}, \mathbf{P}_{k-1}^{d_{1:k-1}}, \mathbf{z}_k). \tag{9.22}$$

Furthermore, we can use the product identity to turn the general expression for $l^{d_{1:k}}$ in (9.21) into a concrete expressions. Instead of working with the mixture weights, we define a *track score function* which corresponds to the logarithm of the weights, i.e, $s_k^{d_{1:k}} = \ln \beta_k^{d_{1:k}} + \text{constant}$. The track score function can then be updated as follows:

$$s_k^{d_{1:k}} = \begin{cases} \ln \lambda + \ln(1 - P_{\mathrm{D}}) + s_{k-1}^{d_{1:k-1}} & \text{if } d_k = 0 \\ \ln \mathcal{N}(\mathbf{z}_k \,;\, \mathbf{H}\hat{\mathbf{x}}_{k|k-1}^{d_{1:k}}, \mathbf{S}_k^{d_{1:k}}) + s_{k-1}^{d_{1:k-1}} & \text{if } d_k > 0. \end{cases} \tag{9.23}$$

A single-target tracker that uses (9.22) and (9.23) to progressively expand a track tree is known as a track split filter. As for the more general MHT, pruning methods are essential for this to work. The most common approach to pruning is N-scan pruning. In this approach, all branches except the best one is pruned at N time steps behind the current time step. Typical values for N may be between 3 and 7. It is also possible to terminate arbitrary branches with very low track score. However, for reasons that will be explained in Section 9.2.2, this can be dangerous when the track-split framework is used within a TO-MHT. Finally, a wide variety of mixture reduction techniques can be used within a track split filter. This is not any more complicated than the mixture reduction methods themselves, because we have a simple and well-defined state space in single-target tracking (i.e., the Euclidean space where $\mathbf{x}_k$ resides). The potential of advanced mixture reduction techniques in the context of TO-MHT is on the other hand highly non-trivial.

9.2.2 Multi-dimensional assignment

Let us now return to the case of a fixed number of N_{TGT} targets. Consider the case where we are provided with N_{TGT} track trees for these targets over the last N time steps. We will often refer to the combined information in the track trees as the *track file*. With this information we can discover the optimal association hypothesis without recursive enumeration. As already mentioned, this can be a huge mitigator of complexity, because the track trees typically contain much fewer leaf nodes than the number of association hypotheses. Furthermore, we only have to solve one search problem, instead of the multiple 2-dimensional assignment problems that we had to solve for every parent hypothesis in our implementation of Reid's MHT.

However, this single search problem is considerably more complex than the 2D assignment problem. Bypassing the recursive formulation, we have to deal with assignments between N

collections of measurements. This is known as N-dimensional assignment or multi-dimensional assignment (MDA). The MDA problem is intimately linked with several well known combinatorial optimization problems. A popular and systematic framework for solving such problems is found in the theory of integer linear programming (ILP). In ILP, the MDA problem can be neatly written as

$$\max_{\boldsymbol{\tau}} \mathbf{s}^{\mathsf{T}}\boldsymbol{\tau} \text{ subject to } \begin{cases} \mathbf{A}_1\boldsymbol{\tau} \leq & \mathbf{b}_1 \\ \mathbf{A}_2\boldsymbol{\tau} = & \mathbf{b}_2 \\ \boldsymbol{\tau} \in & \{0,1\}^n. \end{cases} \tag{9.24}$$

The terms involved in (9.24) have the following meanings. The vector $\mathbf{s}$ contains the score values for all the n leaf nodes in the track forest. The binary vector $\boldsymbol{\tau}$ is our representation of the chosen association hypothesis. It is of the same dimension as $\mathbf{s}$. It contains a 1 for those leaf nodes that are present in the association hypothesis, and it contains zeroes for all other leaf nodes. Thus, the product $\mathbf{c}^{\mathsf{T}}\boldsymbol{\tau}$ provides the total score of the association hypothesis given by $\boldsymbol{\tau}$. The ILP formulation involves two constraint matrices. The first of these, $\mathbf{A}_1$, encodes the constraint that maximally one target can be the origin of any measurement. This matrix contains one row for each measurement in the k scans, and one column for each leaf node. Its element in row j and column t is one if measurement j occurs in the descent leading to leaf node t, otherwise it is zero. Every column of $\mathbf{A}_1$ can therefore be viewed as an encoding of a possible descent, also known as track, in the track forest. The second constraint matrix, $\mathbf{A}_2$, encodes the assumption that exactly N_{TGT} targets exist. It has N_{TGT} rows. The first row is one for all leaf nodes in the first track tree, the second row is one for leaf nodes in the second track tree, etc. Finally, the vectors $\mathbf{b}_1$ and $\mathbf{b}_2$ are filled with ones, and are of dimension M and n. To make this machinery more concrete, let us look at a simple example.

■ **Example 9.3 — Construction of ILP problem.** Consider the track forest in Figure 9.4 . Here

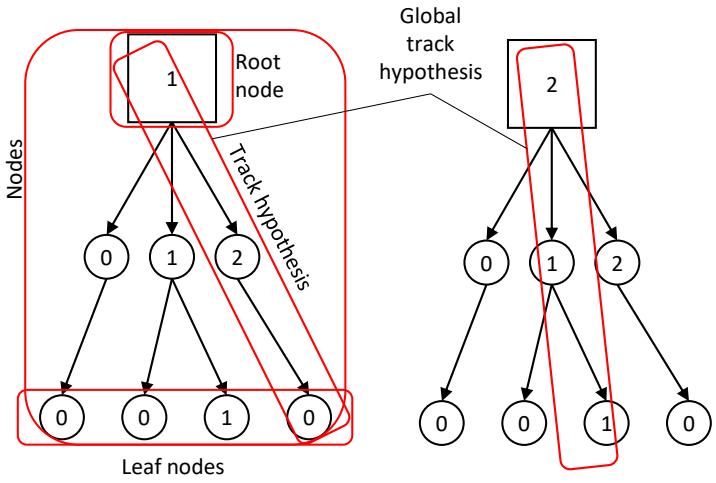


Figure 9.4: The track forest generated by a scenario with 2 measurements in the first scan, and 1 measurement in the second scan. Courtesy of [69].

the entities involved in (9.24) can be written

$$\begin{aligned} \mathbf{A}_1 &= \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}, \quad \mathbf{b}_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \\ \mathbf{A}_2 &= \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}, \quad \mathbf{b}_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\ \mathbf{s} &= \begin{bmatrix} s_1 & s_2 & s_3 & s_4 & s_5 & s_6 & s_7 & s_8 \end{bmatrix}^{\mathsf{T}}. \end{aligned} \tag{9.25}$$

How is this to be interpreted? We see for example that there are two pure zero vectors in $\mathbf{A}_1$. Both of these represent tracks consisting purely of misdetections. If we look at the corresponding columns in $\mathbf{A}_2$, we find that the first one has a one in the first row of $\mathbf{A}_2$, while other one (column number 5) has a one in the second row of $\mathbf{A}_2$. This encodes that the first misdetection track belongs to target 1, while the second misdetection track belongs to target 2.

Furthermore, let us look at the global hypothesis illustrated in Figure 9.4 . With $\mathbf{A}_1$, etc. as given in (9.25) this hypothesis can be represented by

$$\boldsymbol{\tau} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}^\mathsf{T}. \tag{9.26}$$

That is, the marked hypothesis contains tracks number 3 and 8. ∎

To solve the multidimensional assignment problem we can choose between two philosophies. In the first philosophy, we take as our starting point that a a lot of work has been done on solving problems of the form (9.24). Several powerful solvers, such as CBC, CPLEX, GLPK and Gurobi exist (see [68] for an investigation of their suitability), and the assignment problem as given by the matrices and vectors in (9.24) can be fed directly to these solvers.

In the other philosophy, we take as our starting point that the matrices $\mathbf{A}_1$ and $\mathbf{A}_2$ may contain structure particular to the assignment problem, and thus attempt to design solution methods tailor-made for such problems. Three different approaches are dominating the research literature as of mid 2019: Linear programming relaxation [101], Lagrangian relaxation [29] and dual decomposition [120]. Dual decomposition is currently very popular, but builds on Lagrangian relaxation. We will only focus on Lagrangian relaxation here.

9.2.3 Hypothesis search by Lagrangian relaxation

The core idea of Lagrangian relaxation is to replace difficult constraints with additional costs, so that one can solve a simpler optimization problem. To strive for some coherence with standard literature on Lagrangian relaxation such as [47], we will henceforth view the TO-MHT optimization problem as a minimization problem, where the costs of the leaf nodes of the track tree are collected in the vector $\mathbf{c} = -\mathbf{s}$. Consider the two cost functions

$$J(\boldsymbol{\tau}) = \mathbf{c}^\mathsf{T}\boldsymbol{\tau} \tag{9.27}$$

$$J^*(\boldsymbol{\tau}, \mathbf{u}) = \mathbf{c}^\mathsf{T}\boldsymbol{\tau} + \mathbf{u}^\mathsf{T}(\mathbf{A}_1\boldsymbol{\tau} - \mathbf{b}_1). \tag{9.28}$$

The vector $\mathbf{u}$ consists of Lagrange multipliers, also known as slack variables. The original optimization problem, also known as the *primal problem*, which we denote (P), is then

$$v(P) = \min_{\boldsymbol{\tau}} J(\boldsymbol{\tau}) \text{ subject to } \begin{cases} \mathbf{A}_1\boldsymbol{\tau} \leq & \mathbf{b}_1 \\ \mathbf{A}_2\boldsymbol{\tau} = & \mathbf{b}_2 \\ \boldsymbol{\tau} \in & \{0,1\}^n. \end{cases} \tag{9.29}$$

The *relaxed problem*, which we denote (LR_u) is

$$v(LR_\mathbf{u}) = \min_{\boldsymbol{\tau}} J^*(\boldsymbol{\tau}, \mathbf{u}) \text{ subject to } \begin{cases} \mathbf{A}_2\boldsymbol{\tau} = & \mathbf{b}_2 \\ \boldsymbol{\tau} \in & \{0,1\}^n. \end{cases} \tag{9.30}$$

The key observation that underlies Lagrangian relaxation is that $v(LR_\mathbf{u})$ is a lower bound for $v(P)$, no matter what $\mathbf{u}$ is. This can be seen from the fact that $\mathbf{u} \geq \mathbf{0}$ while $\mathbf{A}_1\boldsymbol{\tau} - \mathbf{b}_1 \leq \mathbf{0}$. Thus, if we are able to find a Lagrange vector $\mathbf{u}$ and a candidate solution $\boldsymbol{\tau}$ which satisfies the constraints of the primal problem so that $v(LR_\mathbf{u}) = J(\boldsymbol{\tau})$, then we know that $\boldsymbol{\tau}$ is the optimal solution of the primal problem. This motivates us to define the *dual problem*, denoted (D), as

$$v(D) = \max_{\mathbf{u} \geq 0} v(LR_\mathbf{u}). \tag{9.31}$$

In general, it may not be possible to reach equality between $v(P)$ and $v(D)$. The difference

$$d = v(P) - v(D) \tag{9.32}$$

is known as the *duality gap*. The exact duality gap only can be known after the primal and dual problems both are solved. However, in an iterative scheme that tries out a new solution $\boldsymbol{\tau}_i$ to (P) and a new solution-pair $(\boldsymbol{\tau}_i^*, \mathbf{u}_i)$ to (D) in each iteration, we estimate it from above by updating the following quantities in each iteration i:

$$\begin{aligned} f_0 &= \infty \\ f_0^* &= -\infty \\ f_i &= \min(J(\boldsymbol{\tau}_i), f_{i-1}) \\ f_i^* &= \max(J^*(\boldsymbol{\tau}_i^*, \mathbf{u}_i), f_{i-1}^*). \end{aligned} \tag{9.33}$$

If the difference $f_i - f_i^*$ is small enough one may conclude that $\boldsymbol{\tau}_i$ either is optimal with a high likelihood, or that it is sufficiently close to optimality to be good enough. In such an iterative scheme, we must be able to solve the problems $(LR_{\mathbf{u}})$ and (D) easily. Furthermore there must be a mechanism for constructing a candidate $\boldsymbol{\tau}_i$ for the solution feasible for (P) from the solution $\boldsymbol{\tau}_i^*$ of $(LR_{\mathbf{u}})$, and there must be a mechanism for transferring information from one iteration to the next one. Returning to the multi-dimensional assignment problem of TO-MHT, there are different possibilities for how these challenges can be addressed. In the sequel we sketch a version of TO-MHT based upon the Lagrangian relaxation method that was proposed by Deb, Yeddanapudi, Pattipati and Bar-Shalom in [29]. Each of the 4 mentioned challenges is addressed by a separate phase of the algorithm.

The relaxation phase: In this phase we solve the relaxed problem $(LR_{\mathbf{u}})$. All the inequalities $\mathbf{A}_1\boldsymbol{\tau} \leq \mathbf{b}_1$ are lifted simultaneously, so that we are free to assign any measurement to multiple tracks if it gives a lower overall cost. This is a straightforward optimization problem. In practice, this is done via an upwards pass through the track forest. For each parent node, we choose the children node that gives the lowest cost, taking the Lagrange multipliers of the children nodes into account. We also store the relaxed costs of each node. In mathematical terms we can write this as

$$\begin{aligned} d_{i_0,\ldots,i_L} &= c_{i_0,\ldots,i_L} \\ d_{i_0,\ldots,i_k} &= \min_{i_{k+1}} \left(d_{i_0,\ldots,i_{k+1}} + u_{i_{k+1}} \right) \text{ for } k = 1,\ldots,L-1 \end{aligned} \tag{9.34}$$

where $c_{i_1,\ldots,i_L}$ is the cost of a track consisting of target number i_0 and measurements $i_1,\ldots,i_L$ and $u_{i_{k+1}}$ is the Lagrange multiplier corresponding to measurement number i_{k+1} in scan number $k+1$. Notice that dummy measurements (misdetections) do not impose an constraints. Their Lagrange multipliers are therefore always zero.

The enforcement phase: The goal of this phase is to construct a primal-feasible solution for (P) which hopefully is optimal or near-optimal. Recall that if we only had one scan we could solve the assignment problem by means of the auction method. When several scans are present we can still obtain a feasible solution by employing the auction method recursively, between the track list and scan 1, between scan 1 and scan 2, and so on. However, the reward matrices are constructed by means of the relaxed costs $d_{i_0,\ldots,i_k}$ so that the information from the Lagrange multipliers is utilized. To make this more concrete, the reward matrix for matching tracks with scan number 1 is constructed as follows

$$\begin{bmatrix} -d_{11} & \cdots & -d_{1m_1} & -d_{10} & \cdots & 0 \\ \vdots & & \vdots & \vdots & \ddots & \vdots \\ -d_{n1} & \ldots & -d_{nm_1} & 0 & \cdots & -d_{n0} \end{bmatrix} \tag{9.35}$$

where the values d_{ti_1} have been obtained from the recursive application of (9.34) upwards through the track tree. By penalizing contended measurements, the Lagrange multipliers helps steer the enforcement phase towards to optimal solution.

Solving the dual problem: In this phase, we solve the dual problem (D). The goal is to find a set of Lagrange multipliers so that our estimated duality gap becomes as small as possible. If we are able to reach the ideal situation of zero duality gap, then the solution obtained through the relaxation phase will be identical to the solution obtained through the enforcement phase. This would mean that there is no measurement contention. Thus, we have at least two information sources that can be used to steer the update of the Lagrange multipliers: Reduction of the duality gap and penalties on contended measurements. How do we exploit this information? The good news is that the dual cost $v(LR_{\mathbf{u}})$ is a concave function. This follows from the fact that $J^*(\boldsymbol{\tau},\mathbf{u})$ is a linear function in $\mathbf{u}$, and that $v(LR_{\mathbf{u}})$ is the minimum of a family of such linear functions, parameterized by different possible candidates of $\boldsymbol{\tau}$. The bad news is that the dual cost is not everywhere differentiable. Since we have a finite number of possible $\boldsymbol{\tau}$, there will be boundaries between different regions on the $\mathbf{u}$-space where the derivative of $v(LR_{\mathbf{u}})$ with regard to $\mathbf{u}$ is discontinuous. Furthermore, for all other $\mathbf{u}$ than along these boundaries, the second derivative of $v(LR_{\mathbf{u}})$ will be zero. Standard optimization methods such as Newton's optimization method cannot be used.

Instead, this optimization problem is typically solved using a *subgradient method*. A subgradient of $v(LR_{\mathbf{u}})$ in $\mathbf{u}_0$ is a vector $\mathbf{g}$ so that

$$v(LR_{\mathbf{u}}) - v(LR_{\mathbf{u_0}}) \geq \mathbf{g}(\mathbf{u})(\mathbf{u}-\mathbf{u}_0). \tag{9.36}$$

The subgradient can be understood as the slope of any linear function that touches $v(LR_{\mathbf{u}})$ at $\mathbf{u}_0$ without crossing the surface of $v(LR_{\mathbf{u}})$. The subgradient method is basically a steepest descent (or ascent for the dual problem) method with the ordinary gradient replaced by the subgradient. We have an infinitude of possible choices for the subgradient. Recalling that we want to penalize measurement contention, we can use the following vector as our subgradient: $\mathbf{g} = [g_1,\ldots,g_M]$ where

$$g_i = \begin{cases} 0 & \text{if measurement number } i \text{ is a misdetection.} \\ 0 & \text{if measurement number } i \text{ is claimed by 1 or 0 tracks} \\ \kappa & \text{if measurement number } i \text{ is claimed by } \kappa \text{ tracks.} \end{cases} \tag{9.37}$$

As for any steepest descent approach, the subgradient method will work most efficiently if a good step size is chosen. It is possible to estimate somewhat artificial measures of the overall curvature of $v(LR_{\mathbf{u}})$ and utilize this information in the subgradient method. It was, however, found in [29] that simpler and more heuristic approach worked just as well. First, it obviously makes sense to let the step size depend on the estimated duality gap. Furthermore, the prices from the auction method provides a measure of how serious the measurement contention is. Recall from Algorithm (5) that the prices are increased every time a measurement is taken from one track to another track. This leads to the following update step for the Lagrange multipliers

$$\mathbf{u}_{(i+1)} = \mathbf{u}_{(i)} + \frac{f_{(i)} - f^*_{(i)}}{\|\mathbf{g}\|_2} \frac{\mathbf{p}}{\|\mathbf{p}\|_1}\mathbf{g} \tag{9.38}$$

where $\mathbf{p}$ is the vector of length M containing the prices obtained from the auctions of the enforcement phase.

Two weaknesses of TO-MHT and the Lagrangian relaxation approach in particular are worth mentioning. First, for Lagrangian relaxation to work it is important that the duality gap never is underestimated. This means that the hypothesis cost should never decrease during the enforcement phase. It would of course be absurd if it did, since enforcing more constraints only can limit the

available search space. However, if the auction method fails to solve the 2D assignment problems optimally, such a situation could nevertheless occur. It is therefore important to have a reliable 2D assignment solver.

Second, the track forest (see Figure 9.2 or Figure 9.4) will contain a large number of branches caused primarily by the possibility of misdetections. The state estimates for many of these tracks are likely to be quite similar, and it would therefore seem like a waste to maintain all these branches. Instead, it could be tempting to prune these away, either when they have low track scores or when they are very similar to tracks with higher scores. Unfortunately, this could be rather dangerous. Misdetection nodes provide the necessary flexibility to ensure that the assignment problem has solutions. Removing misdetection nodes could therefore lead to a situation where the Lagrangian relaxation is unable to decrease f_i below ∞, and the algorithm will never converge. In the HO-MHT, on the other hand, we can freely delete any hypothesis without suffering similar consequences.

Our treatment of the TO-MHT did not include track initialization. Conceptually, it is fairly straightforward to include, but it leads to increased complexity, both with regard to bookkeeping and with regard to computational demands. To mitigate the computational demands it has been recommended to deliberately confuse the clutter Poisson process with the new target Poisson process [5]. We shall not pursue this any further in this chapter, but similar ideas will play a central role when we introduce the modern approach to multi-hypothesis tracking in Section 13.7.

9.3 Alternative multi-scan methods

While our focus is on the classical JPDA-MHT paradigm and its generalizations and special cases, it is important to be aware that many other approaches to multi-target tracking exist, each with its own merits and shortcomings. In this section we will cursorially explore three of the most well-known alternatives.[3]

The most radical tracking method that has ever been proposed is perhaps the Recursive RANSAC method. RANSAC stands for Random Sample and Consensus. It can be viewed as a technique for estimating parameters in the presence of outliers. The goal of a RANSAC method is to include as much of the data as possible without including any outliers. Its workflow is to sample a minimal set of data points at random, and then add all additional data points that fits sufficiently well together with the minimal sample set. In multi-target tracking we can employ this approach by treating clutter measurements as outliers and target measurements as inliers. The R-RANSAC method differs from a conventional RANSAC method in that it performs recursive state estimation instead of estimating a single parameter vector. For details, the reader is referred to works by Niedfeldt, Ingersoll and Beard, including [82] and [83]. R-RANSAC is radically different from JPDA and MHT because it does not rely on association probabilities. It only counts inliers. This can be beneficial for robustness, since it to a much lesser extent relies on models, which unfortunately always suffer from serious or not so serious flaws. On the other hand, it means that some of the very precise reasoning capabilities of the JPDA-MHT paradigm are lost.

The Probabilistic Multiple Hypothesis Tracker (PMHT) has been around for 25 years without ever getting the same kind of breakthrough as JPDA and MHT have enjoyed. The PMHT is essentially an attempt at combining the strenghts of JPDA and MHT. It is multi-scan, but in contrast to MHT it is based on soft association by means of association probabilities in the same way as the JPDA. To avoid the combinatorial explosion of enumerating association hypotheses, the PMHT abandons the first at-most-one assumption. In other words, any number of measurements can come from any target. The PMHT treats the data association relationships as *missing data*. The missing data are joint estimated together with the state variables by means of the well-known

[3] It is often thought that random finite sets provide a radically different paradigm than JPDA/MHT. That is a flawed impression. The methods discussed in this section have much less in common with JPDA/MHT.

Expectation-Maximization (EM) method. The EM method is an example of a *variational* estimation technique. Such methods attempt to find a good approximation of the true posterior by iteratively adapting an approximating density of a prescribed general form. The topic of variational inference is currently a very hot topic in sensor fusion, and the PMHT was the first example of such an approach to multi-target tracking. In standard tracking scenarios the performance of the PMHT has generally been disappointing, and it serves perhaps more as a warning against relaxing the at-most-one assumptions than anything else [113]. However, a modified version of the PMHT has show very promising performance in so-called track-before-detect, where a tracking method aims to track very dim targets by working directly on the sensor images [28].

Another approach to the tracking of very dim targets is found in methods based on maximum-likelihood estimation. The rationale behind these methods can perhaps be summarized as follows: If a target is really difficult to spot, then we may only be able to track it if it follows a straight line or some other easily parameterized path. These methods aim to estimate the parameters describing such a path. Thus, the process noise is discarded and the state estimation problem is turned into a parameter estimation problem.

9.4 References and chapter remarks

The MHT was originally proposed by Reid [87] in 1979. However, may of the central ideas, including the integer linear programming (ILP) formulation that we touched upon in Section 9.2.2, can be traced back to an earlier paper by Morefield [75] in 1977. The ILP formulation was later "rediscovered" by Storms & Spieksma in [101] and also by Coraluppi & Carthel [23]. The concept of track-oriented MHT is often credited to Kurien's paper [62] from 1990. However, what is normally understood as a track-oriented MHT is better explained by [75] , [101] and [23]. The main strengths of [62] are a more general formulation than the one used in [87], and the introduction of track trees, which is an enabler for more efficient track score calculations. A systematic derivation of the track score function used in track-oriented MHT is found in [5].

The standard textbook on MHT methods is [11]. In particular, the auction method is very well described in [11]. For the general theory of the MHT the reader should instead consult [3]. The Murty-auction approach to hypothesis-oriented MHT is described in [25].

9.5 Exercises

Exercise 9.1 Derive the initialization formulas for the kinematic pdf of new tracks in Reid's MHT. That is, derive (9.6). Recall that you have to assume that the measurement noise is sufficiently small. ∎

Exercise 9.2 In (9.17) and (9.18) we constructed the reward matrix used in Murty-based hypothesis generation for Reid's MHT when measurements are used as customers. But what if tracks are used as customers and measurements are used as items? Construct a suitable reward matrix and show that assignments based on this reward matrix agrees with Reid's formula from Theorem (9.1.1). What will the dimensions of the reward matrix be? ∎

10. The error-state Kalman filter

Until now, the main focus has been on estimating the motion of objects observed by exteroceptive sensors. The next three chapters will be devoted to a somewhat opposite problem, which is to estimate one's own motion using interoceptive sensors (this chapter) and exteroceptive sensors (the subsequent two chapters). This kind of capability, which we refer to as *navigation* in this book, is essential for all kinds of vehicles and robots.

Interoceptive sensors used for navigation include an inertial measurement unit (IMU), which typically includes an accelerometer and a gyroscope, magnetic or gyroscopic compasses, GPS receivers and possibly also several other sensors. Combining IMU data with any other kinds of sensors is done by an *inertial navigation system* (INS). With inertial navigation we are therefore entering bona-fide sensor fusion. This sensor fusion problem is typically solved by a variation of Kalman filtering known as the error-state Kalman filter (ESKF) or by nonlinear observers. We shall focus on the first approach in this chapter.

For fundamental reasons, a plain-vanilla KF or EKF is not suitable for inertial navigation. IMU measurements generally arrive in 6 degrees of freedom (accelerations along each coordinate axis as well as rotation around the axes). Therefore, an INS is forced to estimate the full 3-dimensional position of the vehicle, as well as its attitude. The attitude is also a 3-dimensional entity, which for example can be represented by the Euler angles yaw, pitch and roll. However, Euler angles are ill fitted to be used in the state vector for inertial navigation. Any 3-dimensional attitude representation will inevitably suffer from singularities, where an infinite number of possible Euler angles will be valid attitude representations.

Another and more subtle argument against Euler angles is that composition of Euler angles is mathematically cumbersome. If we rotate a rigid body according to a triple of Euler angles, and then rotate it according to another triple of Euler angles, then the total rotation will not be anything as simple as just the sums of Euler angles. However, if we use quaternions or rotation matrices, the composition boils down to taking products of these entities. Furthermore, both are singularity-free representations, in respectively 4 and 9 dimensions.

The choice between quaternions and rotation matrices is largely one of personal preference. Other useful representations also exist [35] [24]. Quaternions are in some sense the simplest

singularity-free representation, since quaternions only need 4 dimensions, and since normalization of quaternions is straightforward. More importantly, it is straigthforward to convert quaternions into Euler angles or rotation matrices, while it is more cumbersome to convert a rotation matrix into a quaternion or Euler angles. We will therefore restrict our attention to the quaternion representation of inertial navigation in this chapter.

10.1 Attitude representations and quaternions

Since quaternions are a somewhat abstract concept, we shall in this section recapitulate the more tangible representations of rotation matrices and Euler angles, and relate quaternions to these representations, before we develop state-space models for quaternions in Section 10.2.

10.1.1 Axis-angle representation and rotation matrices

Euler's rotation theorem states that any rotation or sequence of rotations of a rigid body in a three-dimensional space is equivalent to a pure rotation about a single fixed axis. It follows from this that the orientation of a vehicle can be described by a 3-dimensional vector $\mathbf{n}$ (the axis) plus a scalar angle α.

Let $\mathbf{v}$ be a vector in the body frame of the vehicle, let $\mathbf{v}'$ be the same vector in the world frame, and assume that the origin of the two frames coincide. The axis-angle representation of the vehicle's attitude can then be summarized by Rodrigues' rotation formula

$$\mathbf{v}' = (1-\cos\alpha)(\mathbf{n}\cdot\mathbf{v})\mathbf{n} + \cos\alpha\mathbf{v} - \sin\alpha(\mathbf{v}\times\mathbf{n}). \tag{10.1}$$

At this stage, the reader is encouraged to make two observations. First, the formulation of the relationship between $\mathbf{v}$, α and $\mathbf{v}'$ implies a *convention* choice for what we mean by the concept of attitude. The angle α is used to describe how a vector originally decomposed in the body frame, alternatively can be decomposed in the world frame. This is known as the **the passive body-to-world convention**, and will be elaborated below. Second, you may verify that the rotation described by (10.1) boils down to the rotation described by a 2-dimensional rotation matrix

$$\mathsf{R}(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \tag{10.2}$$

around the z-axis if $\mathbf{n} = [0,0,1]^\mathsf{T}$. Indeed, we have used this observation to derive (10.1) in Appendix P.3. This means that Rodrigues' formula follows a right-hand convention: The rotations will appear counter-clockwise if $\mathbf{n}$ is aligned with your right hand thumb.

What happens to $\mathbf{v}$ in (10.1) consists purely of linear operations. We can therefore replace the complicated expression in (10.1) with a single matrix multiplication

$$\mathbf{v}' = \mathsf{R}_{\mathbf{n},\alpha}\mathbf{v} \tag{10.3}$$

where the 3-dimensional rotation matrix is found as

$$\begin{aligned}\mathsf{R}_{\mathbf{n},\alpha} &= \cos\alpha\mathbf{I} + (1-\cos\alpha)\mathbf{n}\mathbf{n}^\mathsf{T} + \sin\alpha\mathsf{S}(\mathbf{n}) \\ &= \begin{bmatrix} c\alpha + (1-c\alpha)n_x^2 & (1-c\alpha)n_xn_y - s\alpha n_z & (1-c(\alpha)n_xn_z + s\alpha n_y \\ (1-c\alpha)n_xn_y + s\alpha n_z & c\alpha + (1-c\alpha)n_y^2 & (1-c\alpha)n_yn_z - s\alpha n_x \\ (1-c\alpha)n_zn_x - s\alpha n_y & (1-c\alpha)n_zn_y + s\alpha n_x & c\alpha + (1-c\alpha)n_z^2. \end{bmatrix}\end{aligned} \tag{10.4}$$

Here cos has been abbreviated c, while sin has been abbreviated s. By S we denote the skew-symmetric matrix operator, which is defined such that $\mathbf{n}\times\boldsymbol{a} = \mathsf{S}(\mathbf{n})\boldsymbol{a}$, or equivalently:

$$\mathsf{S}(\mathbf{n}) = \begin{bmatrix} 0 & -n_3 & n_2 \\ n_3 & 0 & -n_1 \\ -n_2 & n_1 & 0 \end{bmatrix}. \tag{10.5}$$

The axis-angle representation is worth keeping in mind because it is so intuitive. Representations using Euler angles or quaternions may quickly lead to complicated expressions which are prone to programming errors. An implementation of the axis-angle representation is therefore essential for debugging purposes.

Rotation matrices are elements of the manifold $SO(3)$, which consist of all orthogonal matrices with determinant 1. Thus, any rotation matrix satisfies

$$\det(\mathsf{R}) = 1 \tag{10.6}$$

$$\mathsf{R}^\mathsf{T} = \mathsf{R}^{-1}. \tag{10.7}$$

Also notice that for the axis-angle representation the following applies:

$$\mathsf{R}_{\mathbf{n},-\alpha} = \mathsf{R}_{-\mathbf{n},\alpha} = \mathsf{R}_{\mathbf{n},\alpha}^\mathsf{T}. \tag{10.8}$$

As we now have introduced the concept of rotation matrices, it is time to give these rotations meaning in a physical sense, and for this we shall return to the discussion of the many convention choices that are available.

Rotations can be interpreted both as active or as passive transformations. In the active convention, the rotation matrix describes how the vehicle is rotated from being aligned with the world frame to its current attitude. In the passive convention, the rotation matrix describes a change of coordinates between the body frame and the world frame. Thus, there are two possible passive conventions: Body-to-world and world-to-body. As mentioned, we use the passive body-to-world convention. It is fairly evident that the mathematics of this convention become exactly the same as in the active convention. On the other hand, the rotation matrix of the the passive world-to-body convention will be the transpose of the rotation matrix of the active convention. The active convention is perhaps easier to understand from an intuitive perspective. After all, it tells us *how the vehicle is rotated*. Nevertheless, at least the author feels that a passive convention is preferable in order to define attitude precisely. It reduces everything to a question of coordinate transformations, which is purely a matter of information representation, which is what estimation after all is about. Another reason why we use the passive body-to-world convention is simply that this is the convention used in [100], which this chapter is based on. To summarize this paragraph, in the passive body-to-world convention we can think of the rotation matrix as a transformation matrix which places objects seen by the vehicle in its body frame into the world frame.

Another important convention choice that must be clarified is what we mean by the terms *world frame* and *body frame*. For simplicity, we assume in this chapter that the Earth is flat, and we place the origin of the world frame at an arbitrary but fixed geographical location, e.g., the Greenwhich observatory, Munkholmen, a corner in the snake robot lab, etc. To specify its basis vectors, the most common choices are East (x) - North (y) - Up (z), also known as ENU, and North (x) - East (y) - Down (z), also known as NED. Notice that both follow a right-hand convention. While ENU may seem like the most natural choice for most people outside of cybernetics, it is the NED convention that dominates in nautical and aerospace engineering.

Both conventions have some pros and cons with regard to how well they fit our intuition. The ENU convention allows the xy plane to be visualized with the y-axis above the x-axis when seen from above, and it does not force us to include a minus to describe how high above the ground an airplane is. On the other hand, the NED convention has at least two important benefits. First, it becomes natural in the NED convention to let the x-axis point forward, as opposed to sideways. This makes it more natural to align the x-direction with the direction of motion. Second, when we introduce Euler angles, we will see that for the NED convention positive pitch means accelerating upwards, while for the ENU convention it means accelerating downwards, which is less inuitive.

Once the world frame has been chosen according to ENU or NED, then the body frame is chosen so that the x-axis points along the forward direction of the vehicle, the z-axis points up for

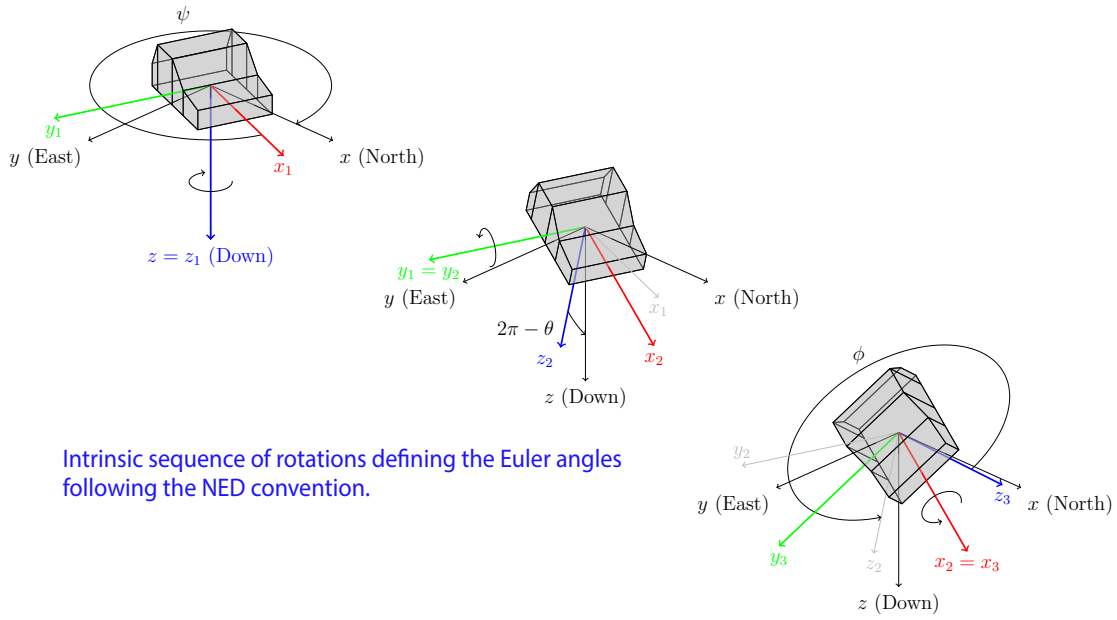


Figure 10.1: Illustration of Euler angles.

ENU or down for NED, and the y axis then follows from the right-hand rule: Portside for ENU or starboard for NED. In the sequel, we shall only use the NED convention.

10.1.2 Euler angles and intrinsic-extrinsic equivalence

Any orientation can be decomposed into 3 successive rotations around axes of the coordinate system. Such a decomposition is in general referred to as Euler angles. Since several such decompositions are possible, we again have to discuss the different convention choices.

The rotations can be performed around each axis exactly once (e.g. $X \to Y \to Z$), or with one axis repeated once (e.g. $X \to Y \to X$). The former kind of decompositions are often referred to as *Tait-Bryan angles*, while the latter kind of decompositions are referred to as *proper Euler angles*. The order of decomposition matters. Therefore, there are 6 different ways in which the Tait-Bryan angles can be specified, and 6 different ways in which the proper Euler angles can be specified. Tait-Bryan angles can intuitively be interpreted as roll, pitch and yaw, and are therefore by far the most popular choice in navigation applications.

The rotations of the Euler angles can be carried out in an *intrinsic* manner or in an *extrinsic* manner. Intrinsic rotation means that the axes are mobile: During each rotation the axes are also rotated, so that the next rotation are carried out around its corresponding axis as rotated by previous rotations. The entire process of intrinsic rotation can be written as follows:

$$\begin{aligned}\{\boldsymbol{e}'_x, \boldsymbol{e}'_y, \boldsymbol{e}'_z\} &= \mathsf{R}_{\boldsymbol{e}_z,\psi}\{\boldsymbol{e}_x, \boldsymbol{e}_y, \boldsymbol{e}_z\} \\ \{\boldsymbol{e}''_x, \boldsymbol{e}''_y, \boldsymbol{e}''_z\} &= \mathsf{R}_{\boldsymbol{e}'_y,\theta}\{\boldsymbol{e}'_x, \boldsymbol{e}'_y, \boldsymbol{e}'_z\} \\ \{\boldsymbol{e}'''_x, \boldsymbol{e}'''_y, \boldsymbol{e}'''_z\} &= \mathsf{R}_{\boldsymbol{e}''_x,\phi}\{\boldsymbol{e}''_x, \boldsymbol{e}''_y, \boldsymbol{e}''_z\}.\end{aligned} \tag{10.9}$$

Notice that we have used the notation for the axis angle representation to define the Euler angles. Extrinsic rotation means that all the rotations are applied around the original axes of the original frame:

$$\{\boldsymbol{e}'''_x, \boldsymbol{e}'''_y, \boldsymbol{e}'''_z\} = \mathsf{R}_{\boldsymbol{e}_z,\psi}\mathsf{R}_{\boldsymbol{e}_y,\theta}\mathsf{R}_{\boldsymbol{e}_x,\phi}\{\boldsymbol{e}_x, \boldsymbol{e}_y, \boldsymbol{e}_z\}. \tag{10.10}$$

Intrinsic rotation is easier to visualize, and more directly applicable to navigation problems, but extrinsic rotation is neater from a mathematical perspective. Fortunately, intrinsic rotation can be replaced by extrinsic rotation and vice versa:

Theorem 10.1.1 — Intrinsic-extrinsic equivalence. Intrinsic rotations yield the same result as extrinsic rotations carried out in the opposite sequence:

$$\begin{aligned}\{\boldsymbol{e}_x''', \boldsymbol{e}_y''', \boldsymbol{e}_z'''\} &= \mathsf{R}_{\boldsymbol{e}_x'',\phi}\mathsf{R}_{\boldsymbol{e}_y',\theta}\mathsf{R}_{\boldsymbol{e}_z,\psi}\{\boldsymbol{e}_x, \boldsymbol{e}_y, \boldsymbol{e}_z\} \\ &= \mathsf{R}_{\boldsymbol{e}_z,\psi}\mathsf{R}_{\boldsymbol{e}_y,\theta}\mathsf{R}_{\boldsymbol{e}_x,\phi}\{\boldsymbol{e}_x, \boldsymbol{e}_y, \boldsymbol{e}_z\}. \end{aligned} \tag{10.11}$$

Proof. The matrices in the first line of (10.11) can be written

$$\begin{aligned}\mathsf{R}_{\boldsymbol{e}_y',\theta} =& \mathsf{R}_{\boldsymbol{e}_z,\psi}\mathsf{R}_{\boldsymbol{e}_y,\theta}\mathsf{R}_{\boldsymbol{e}_z,\psi}^{\mathsf{T}} \\ =& \mathsf{R}_{\boldsymbol{e}_z,\psi}\mathsf{R}_{\boldsymbol{e}_y,\theta}\mathsf{R}_{\boldsymbol{e}_z,\psi}^{-1} \\ \mathsf{R}_{\boldsymbol{e}_x'',\phi} =& [\mathsf{R}_{\boldsymbol{e}_y',\theta}\mathsf{R}_{\boldsymbol{e}_z,\psi}]\mathsf{R}_{\boldsymbol{e}_x,\phi}[\mathsf{R}_{\boldsymbol{e}_z,\psi}^{\mathsf{T}}\mathsf{R}_{\boldsymbol{e}_y',\theta}^{\mathsf{T}}] \\ =& \left[\left[\mathsf{R}_{\boldsymbol{e}_z,\psi}\mathsf{R}_{\boldsymbol{e}_y,\theta}\mathsf{R}_{\boldsymbol{e}_z,\psi}^{-1}\right]\mathsf{R}_{\boldsymbol{e}_z,\psi}\right]\mathsf{R}_{\boldsymbol{e}_x,\phi} \\ & \cdot\left[\mathsf{R}_{\boldsymbol{e}_z,\psi}^{-1}\left[\mathsf{R}_{\boldsymbol{e}_z,\psi}\mathsf{R}_{\boldsymbol{e}_y,\theta}^{-1}\mathsf{R}_{\boldsymbol{e}_z,\psi}^{-1}\right]\right] \\ =& \mathsf{R}_{\boldsymbol{e}_z,\psi}\mathsf{R}_{\boldsymbol{e}_y,\theta}\mathsf{R}_{\phi,\boldsymbol{e}_x}\mathsf{R}_{\boldsymbol{e},\theta}^{-1}\mathsf{R}_{\boldsymbol{e}_z,\psi}^{-1}. \end{aligned} \tag{10.12}$$

Therefore the combined intrinsic rotation is given by

$$\begin{aligned}\{\boldsymbol{e}_x''', \boldsymbol{e}_y''', \boldsymbol{e}_z'''\} =& \left[\mathsf{R}_{\boldsymbol{e}_z,\psi}\mathsf{R}_{\boldsymbol{e}_y,\theta}\mathsf{R}_{\boldsymbol{e}_x,\phi}\mathsf{R}_{\boldsymbol{e}_y,\theta}^{-1}\mathsf{R}_{\boldsymbol{e}_z,\psi}^{-1}\right] \\ & \cdot\left[\mathsf{R}_{\boldsymbol{e}_z,\psi}\mathsf{R}_{\boldsymbol{e}_y,\theta}\mathsf{R}_{\boldsymbol{e}_z,\psi}^{-1}\right]\mathsf{R}_{\boldsymbol{e}_z,\psi}\{\boldsymbol{e}_x, \boldsymbol{e}_y, \boldsymbol{e}_z\} \\ =& \mathsf{R}_{\boldsymbol{e}_z,\psi}\mathsf{R}_{\boldsymbol{e}_y,\theta}\mathsf{R}_{\boldsymbol{e}_x,\phi}\{\boldsymbol{e}_x, \boldsymbol{e}_y, \boldsymbol{e}_z\}. \end{aligned} \tag{10.13}$$

■

This motivates us to define the standard Euler-angle based rotation matrix as

$$\mathsf{R}(\phi, \theta, \psi) \triangleq \mathsf{R}_{\boldsymbol{e}_z,\psi}\mathsf{R}_{\boldsymbol{e}_y,\theta}\mathsf{R}_{\boldsymbol{e}_x,\phi}. \tag{10.14}$$

To specify our convention choice explicitly we use the extrinsic sequence yaw - pitch - roll ($Z \to Y \to X$), which is equivalent to the intrinsic sequence roll - pitch - yaw ($X \to Y \to Z$). According to the right hand rule (and this would also hold if we had used the ENU convention), the matrices involved in (10.14) are

$$\mathsf{R}_{\boldsymbol{e}_x,\phi} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\phi & -s\phi \\ 0 & s\phi & c\phi \end{bmatrix} \tag{10.15}$$

$$\mathsf{R}_{\boldsymbol{e}_y,\theta} = \begin{bmatrix} c\theta & 0 & s\theta \\ 0 & 1 & 0 \\ -s\theta & 0 & c\theta \end{bmatrix} \tag{10.16}$$

$$\mathsf{R}_{\boldsymbol{e}_z,\psi} = \begin{bmatrix} c\psi & -s\psi & 0 \\ s\psi & c\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{10.17}$$

$$\mathsf{R}(\phi, \theta, \psi) = \begin{bmatrix} c\psi c\theta & -s\psi c\phi + c\psi s\theta s\phi & s\psi s\phi + c\psi s\theta c\phi \\ s\psi c\theta & c\psi c\phi + s\psi s\theta s\phi & -c\psi s\phi + s\psi s\theta c\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix}. \tag{10.18}$$

10.1.3 Quaternions

Recall that complex numbers can be used to encode rotations in the complex plane. If we multiply a complex number c with a complex number whose absolute value is 1 and phase angle is θ, then the result is to rotate c by θ radians counterclockwise in the complex plane. Quaternions were invented to generalize this concept to rotations in 3-dimensional space. To achieve this, the complex numbers had to be generalized, not only to become 3-dimensional entities, but actually 4-dimensional entities. A complex number is a sum of a real-valued entry and an imaginary-valued entry. A quaternion, in contrast, is a sum of a real-valued entry and three "hyperimaginary" entries. We write a quaternion as

$$\boldsymbol{q} = \begin{bmatrix} \eta \\ \boldsymbol{\epsilon} \end{bmatrix} \tag{10.19}$$

where $\eta \in \mathbb{R}$ is called the scalar part and $\boldsymbol{\epsilon} \in \mathbb{R}^3$ is called the vector part. In a manner similar to complex numbers, quaternions are not merely such 4-vectors, but come with certain rules for addition and multiplication, which are used as part of the definition of what a quaternions is. The sum of two quaternions is, as for complex numbers, trivial:

$$\boldsymbol{q}_a + \boldsymbol{q}_b = \begin{bmatrix} \eta_a \\ \boldsymbol{\epsilon}_b \end{bmatrix} + \begin{bmatrix} \eta_a \\ \boldsymbol{\epsilon}_b \end{bmatrix} = \begin{bmatrix} \eta_a + \eta_p \\ \boldsymbol{\epsilon}_a + \boldsymbol{\epsilon}_p \end{bmatrix}. \tag{10.20}$$

The multiplication, on the other hand, is where the interesting stuff happens:

$$\boldsymbol{q}_a \otimes \boldsymbol{q}_b = \begin{bmatrix} \eta_a \eta_b - \boldsymbol{\epsilon}_a^\mathsf{T} \boldsymbol{\epsilon}_b \\ \eta_b \boldsymbol{\epsilon}_a + \eta_a \boldsymbol{\epsilon}_b + \boldsymbol{\epsilon}_a \times \boldsymbol{\epsilon}_b \end{bmatrix} \tag{10.21}$$

All the well-known or more obscure properties of quaternions that one finds in sources such as [31] or [100] can be derived from (10.21). See Exercise 10.2 for the most fundamental properties. Thus, we emphasize that (10.21) really can be used as a definition for the quaternions.

To strengthen our understanding of what the quaternion product does, let us first notice that it covers complex multiplication as a special case. If $\boldsymbol{\epsilon}_a = [r_a, 0, 0]^\mathsf{T}$ and $\boldsymbol{\epsilon}_b = [r_b, 0, 0]^\mathsf{T}$, then the cross product is zero and what comes out of the quaternion product is

$$\begin{bmatrix} \eta_a \eta_b - r_a r_b \\ \eta_b r_a + \eta_a r_b \\ 0 \\ 0 \end{bmatrix} \tag{10.22}$$

which is nothing else than the standard formula for complex multiplication. Also as a generalization of complex numbers, the identity quaternion is given by

$$\boldsymbol{q}_\mathbf{I} = \begin{bmatrix} 1 \\ \mathbf{0} \end{bmatrix}. \tag{10.23}$$

The identity quaternion satisfies $\boldsymbol{q} \otimes \boldsymbol{q}_\mathbf{I} = \boldsymbol{q}_\mathbf{I} \otimes \boldsymbol{q} = \boldsymbol{q}$ for any $\boldsymbol{q}$. The main complicating element in (10.21) is the cross product. From the assumption that quaternions should be capable of representing spatial rotation, it should not come as a surprise that a cross product must be present. After all, a similar cross product is present in the angle-axis rotation formula (10.1). The presence of a cross product means that the quaternion product is non-commutative. Again, this should not come as a surprise, since rotation matrices also behave in a non-commutative manner.

To lay the groundwork for attitude representation in terms of quaternions we have to introduce some additional concepts, in particular *unit quaternions* and *conjugate quaternions*. The conjugate

of the quaternion $\boldsymbol{q} = [\eta, \boldsymbol{\epsilon}^\mathsf{T}]^\mathsf{T}$ is

$$\boldsymbol{q}^* = \begin{bmatrix} \eta \\ -\boldsymbol{\epsilon} \end{bmatrix}. \tag{10.24}$$

The norm of a quaternion is

$$\|\boldsymbol{q}\| = \sqrt{\eta^2 + \varepsilon_1^2 + \varepsilon_2^2 + \varepsilon_3^2}. \tag{10.25}$$

Let us also mention that the inverse of the quaternion $\boldsymbol{q}$ is the quaternion $\boldsymbol{q}^{-1}$ which is given by

$$\boldsymbol{q} \otimes \boldsymbol{q}^{-1} = \boldsymbol{q}^{-1} \otimes \boldsymbol{q} = \boldsymbol{q}_\mathbf{I}. \tag{10.26}$$

The inverse of a quaternion can be found by means of the conjugate according to

$$\boldsymbol{q}^{-1} = \boldsymbol{q}^* / \|\boldsymbol{q}\|^2. \tag{10.27}$$

A unit quaternion $\boldsymbol{q}$ is a quaternion that satisfies $\|\boldsymbol{q}\| = 1$. For unit quaternions the inverse equals the conjugate according to $\boldsymbol{q}^{-1} = \boldsymbol{q}^*$. We prove that every unit quaternion represents a spatial rotation, and we also show how this is done, in the following theorem.

Theorem 10.1.2 Let the unit quaternion be given by

$$\boldsymbol{q} = \begin{bmatrix} \eta \\ \boldsymbol{\epsilon} \end{bmatrix} = \begin{bmatrix} \cos\frac{\alpha}{2} \\ \mathbf{n}\sin\frac{\alpha}{2} \end{bmatrix}, \tag{10.28}$$

and assume that $\|\mathbf{n}\| = 1$. Let the vector $\mathbf{v}'$ be related to the vector $\mathbf{v}$ according to $\mathbf{v}' = \mathsf{R}_{\mathbf{n},\alpha}\mathbf{v}$. In terms of quaternions, the same rotation is given by

$$\begin{bmatrix} 0 \\ \mathbf{v}' \end{bmatrix} = \boldsymbol{q} \otimes \begin{bmatrix} 0 \\ \mathbf{v} \end{bmatrix} \otimes \boldsymbol{q}^* \tag{10.29}$$

Proof. The second product in (10.29) becomes

$$\begin{bmatrix} 0 \\ \mathbf{v} \end{bmatrix} \otimes \begin{bmatrix} \eta \\ -\boldsymbol{\epsilon} \end{bmatrix} = \begin{bmatrix} \mathbf{v}^\mathsf{T}\boldsymbol{\epsilon} \\ \eta\mathbf{v} - \mathbf{v}\times\boldsymbol{\epsilon} \end{bmatrix}. \tag{10.30}$$

Performing the first product as well yields

$$\begin{aligned} \begin{bmatrix} \eta \\ \boldsymbol{\epsilon} \end{bmatrix} \begin{bmatrix} \mathbf{v}^\mathsf{T}\boldsymbol{\epsilon} \\ \eta\mathbf{v} - \mathbf{v}\times\boldsymbol{\epsilon} \end{bmatrix} &= \begin{bmatrix} \eta\mathbf{v}^\mathsf{T}\boldsymbol{\epsilon} - \boldsymbol{\epsilon}^\mathsf{T}(\eta\mathbf{v} - \mathbf{v}\times\boldsymbol{\epsilon}) \\ (\mathbf{v}^\mathsf{T}\boldsymbol{\epsilon})\boldsymbol{\epsilon} + \eta(\eta\mathbf{v} - \mathbf{v}\times\boldsymbol{\epsilon}) + \boldsymbol{\epsilon}\times(\eta\mathbf{v} - \mathbf{v}\times\boldsymbol{\epsilon}) \end{bmatrix} \\ &= \begin{bmatrix} 0 \\ (\mathbf{v}^\mathsf{T}\boldsymbol{\epsilon})\boldsymbol{\epsilon} + \eta^2\mathbf{v} - 2\eta(\mathbf{v}\times\boldsymbol{\epsilon}) - \big((\boldsymbol{\epsilon}^\mathsf{T}\boldsymbol{\epsilon})\mathbf{v} - (\mathbf{v}^\mathsf{T}\boldsymbol{\epsilon})\boldsymbol{\epsilon}\big) \end{bmatrix} \\ &= \begin{bmatrix} 0 \\ 2(\mathbf{v}^\mathsf{T}\boldsymbol{\epsilon})\boldsymbol{\epsilon} + 2\eta(\boldsymbol{\epsilon}\times\mathbf{v}) + (\eta^2 - \|\boldsymbol{\epsilon}\|^2)\mathbf{v} \end{bmatrix} \end{aligned} \tag{10.31}$$

We see from (10.31) that the proposed rotation formula indeed gives zero contribution in the scalar slot of the resulting quaternion, as it is supposed to. In the vector part of (10.31) there are three terms, and we shall show that these correspond to the terms in the angle-axis formula (10.1). With the proposed substitutions we can use standard trigonometric identities to obtain

$$\begin{aligned} \eta^2 - \|\boldsymbol{\epsilon}\|^2 &= \cos^2\frac{\alpha}{2} - \sin^2\frac{\alpha}{2} = \cos\alpha \\ 2(\mathbf{v}^\mathsf{T}\boldsymbol{\epsilon})\boldsymbol{\epsilon} &= 2\sin^2\frac{\alpha}{2}(\mathbf{v}^\mathsf{T}\mathbf{n})\mathbf{n} = (1-\cos\alpha)(\mathbf{v}^\mathsf{T}\mathbf{n})\mathbf{n} \\ 2\eta(\boldsymbol{\epsilon}\times\mathbf{v}) &= 2\cos\frac{\alpha}{2}\sin\frac{\alpha}{2}(\boldsymbol{\epsilon}\times\mathbf{v}) = \sin\alpha(\boldsymbol{\epsilon}\times\mathbf{v}). \end{aligned} \tag{10.32}$$

Let us recall that the angle-axis formula was

$$\mathbf{v}' = (1-\cos\alpha)(\mathbf{n}\cdot\mathbf{v})\mathbf{n} + \cos\alpha\mathbf{v} + \sin\alpha(\mathbf{n}\times\mathbf{v}). \tag{10.33}$$

We see that the terms in (10.32) are the same as in the angle-axis formula, and this concludes the proof. ■

Returning to general properties of the quaternion product, it should be noted that it can be expressed in terms of conventional matrix-vector products as follows:

$$\boldsymbol{q}_a \otimes \boldsymbol{q}_b = \left(\eta_a \mathbf{I} + \begin{bmatrix} 0 & -\boldsymbol{\epsilon}_a^\mathsf{T} \\ \boldsymbol{\epsilon}_a & \mathsf{S}(\boldsymbol{\epsilon}_a) \end{bmatrix}\right) \begin{bmatrix} \eta_b \\ \boldsymbol{\epsilon}_b \end{bmatrix} \tag{10.34}$$

$$\boldsymbol{q}_a \otimes \boldsymbol{q}_b = \left(\eta_b \mathbf{I} + \begin{bmatrix} 0 & -\boldsymbol{\epsilon}_b^\mathsf{T} \\ \boldsymbol{\epsilon}_b & -\mathsf{S}(\boldsymbol{\epsilon}_b) \end{bmatrix}\right) \begin{bmatrix} \eta_a \\ \boldsymbol{\epsilon}_a \end{bmatrix}. \tag{10.35}$$

For convenience we will sometimes use a short-hand notation for quaternion-vector-products such as (10.29). When a vector is inserted in the vector slot of a quaternion, and the scalar slot is zero, we may simply write this as a quaternion product with the vector. With this notation, (10.29) becomes

$$\mathbf{v}' = \boldsymbol{q} \otimes \mathbf{v} \otimes \boldsymbol{q}^*. \tag{10.36}$$

Unfortunately, the quaternion representation of rotations is not unique. This can be seen from its close relationship with the angle-axis representation. If we always replace $\mathbf{n}$ with $-\mathbf{n}$, and simultaneously replace α with $2\pi - \alpha$, the we obtain another representation of exactly the same rotation. The consequence of this is that $\boldsymbol{q}$ and $-\boldsymbol{q}$ represent the same rotation. This is not an issue of big practical importance, but it means that care must be exercise if one for example wants to evaluate the distance between two quaternions or wants to calculate derivatives of quaternions by means of finite differences.

10.1.4 Conversion formulas

From Theorem 10.1.2 we have established formulas for conversion between quaternions and the angle-axis formula. Based on this, we can also derive formulas for conversion between quaternions and other attitude representations such as Euler angles or rotation matrices. Since we have a formula for conversion from angle-axis to rotation matrix, we also have a formula for conversion from quaternion to rotation matrix:

$$\mathsf{R} = \mathbf{I} + 2\eta\mathsf{S}(\boldsymbol{\epsilon}) + 2\mathsf{S}(\boldsymbol{\epsilon})\mathsf{S}(\boldsymbol{\epsilon}). \tag{10.37}$$

The opposite conversion is less straightforward, because it involves extracting the 4 quaternion parameters from the 9 parameters in the rotation matrix. By spelling out the details in (10.37) it is possible to construct several nonlinear equations in η, ε_1, ε_2 and ε_3 which are fairly straightforward to solve. However, numerical stability can be an issue: One must choose the right equations to solve to avoid division by zero. As an alternative one can make a conversion from rotation matrix to Euler angles, based on the expression for $\mathsf{R}(\phi,\theta,\psi)$ in (10.18) and then from Euler angles to quaternions. Again, the equations must be chosen so that singularities are avoided. We shall not delve further into concrete expressions for these conversions here, but refer the reader to sources such as [12] and [109].

The conversion from quaternions to (Tait-Bryan) Euler angles is given by

$$\begin{aligned} \phi &= \operatorname{atan2}\left(2(\varepsilon_3\varepsilon_2 + \eta\varepsilon_1), \eta^2 - \varepsilon_1^2 - \varepsilon_2^2 + \varepsilon_3^2\right) \\ \theta &= \operatorname{asin}\left(2(\eta\varepsilon_2 - \varepsilon_1\varepsilon_3)\right) \\ \psi &= \operatorname{atan2}\left(2(\varepsilon_1\varepsilon_2 + \eta\varepsilon_3), \eta^2 + \varepsilon_1^2 - \varepsilon_2^2 - \varepsilon_3^2\right) \end{aligned} \tag{10.38}$$

Notice that our usage of the atan2$(\cdot)$ function relieves us from worry about singularities. This conversion is especially important for visualization purposes. The inverse conversion is given by

$$\boldsymbol{q} = \begin{bmatrix} \cos\frac{\phi}{2}\cos\frac{\theta}{2}\cos\frac{\psi}{2} + \sin\frac{\phi}{2}\sin\frac{\theta}{2}\sin\frac{\psi}{2} \\ \sin\frac{\phi}{2}\cos\frac{\theta}{2}\cos\frac{\psi}{2} - \cos\frac{\phi}{2}\sin\frac{\theta}{2}\sin\frac{\psi}{2} \\ \cos\frac{\phi}{2}\sin\frac{\theta}{2}\cos\frac{\psi}{2} + \sin\frac{\phi}{2}\cos\frac{\theta}{2}\sin\frac{\psi}{2} \\ \cos\frac{\phi}{2}\cos\frac{\theta}{2}\sin\frac{\psi}{2} - \sin\frac{\phi}{2}\sin\frac{\theta}{2}\cos\frac{\psi}{2} \end{bmatrix} \tag{10.39}$$

This conversion is for example useful when one wants to specify some initial conditions in Euler angles. The results in (10.38) and (10.39) can be derived by comparing the elements in the quaternion-based rotation matrix from (10.37) with the elements in the Euler angle based rotation matrix $\mathsf{R}(\phi, \theta, \psi)$ from (10.18).

10.1.5 Frame transformations

Transformations of vectors between different frames is of central importance in inertial navigation. The reasons for this are manifold. Some applications, e.g., underwater submarine navigation, have extremely high requirements for long-term accuracy, which means that the rotation of the Earth must be taken into account, which again means that the location on the Earth ellipsoid is of importance. In other applications, e.g., ship or airplane navigation, different sensors that are used in the navigation system may be located a significant distance away from the center of gravity (COG). In such cases lever arm compensation is necessary. Finally, in SLAM we typically observe landmarks in different consecutive body frames, and from these observations we attempt to estimate the transformations between the body frames.

Different notations exist for ensuring precision when working with frame transformations. In most of this chapter we shall only have to deal with two frames: Body and World, and their relationships will be obvious. We therefore adopt the minimalist notation used by [100], where frame indices are omitted entirely. This allows us to use the subscript slot for other indices indices instead (truth, nominal, error, etc., as elaborated on page 172). When explicit frame notations are needed, we attempt to follow the same principles as in Fossen's book [37]. We denote the world frame by $\{w\}$ and the body frame by $\{b\}$. We use the notation $\boldsymbol{\rho}^w_{b|w}$ to denote the distance from the world frame to the body frame expressed in world coordinates. The passive body-to-world rotation matrix is written R^w_b when the frame relationships are to be emphasized. Also, let $\boldsymbol{\rho}^b$ denote a point expressed in the body frame.[1] This leads to the transformation formula

$$\boldsymbol{\rho}^w = \boldsymbol{\rho}^w_{b|w} + \mathsf{R}^w_b \boldsymbol{\rho}^b.$$

10.2 Kinematics and process model for the ESKF

Assume that the vehicle at a given time is rotating with angular velocity $\boldsymbol{\omega}$ decomposed in the body frame. It will continue to rotate with the same angular velocity, decomposed in the body frame, for as long as no forces are applied to it. Its attitude will obviously change. This change is more complicated than simply adding together the rotation contributions over different time segments. Instead, a differential equation involving both the attitude and the angular velocity must be solved. For this reason, the continuous-time perspective is ubiquitous in inertial navigation.

10.2.1 The quaternion differential equation

To develop a formula for the time derivative of the attitude quaternion we start by studying *perturbations* of quaternions. Recall from Theorem 10.1.2 that rotation using quaternions is done

[1] A distinction is sometimes made between vectors and points, also known as affine points. The difference is perhaps best understood through an analogy: Times are points in time, while durations are displacements (i.e., vectors) in time.

by means of the quaternion product. Also recall that quaternion multiplication is non-commutative. Perturbing a quaternion by a perturbation $\Delta\boldsymbol{q}$ can therefore be expressed in two different ways:

$$\tilde{\boldsymbol{q}} = \boldsymbol{q} \otimes \Delta\boldsymbol{q} \quad \text{or} \quad \tilde{\boldsymbol{q}} = \Delta\boldsymbol{q} \otimes \boldsymbol{q}. \tag{10.40}$$

These two expressions do not yield the same result. The former is known as a local perturbation while the latter is known as a global perturbation. Some justification for this terminology can be seen from (10.29): With this choice the local vector $\mathbf{v}$ is first rotated according to the local perturbation $\Delta\boldsymbol{q}$, before it is rotated from local to global frame by $\boldsymbol{q}$. In contrast, the former choice implies that we rotate according to the perturbation $\Delta\boldsymbol{q}$ before we rotate from local to global by means of $\boldsymbol{q}$.

The perturbation is of course assumed to be small, and we utilize this to relate it to the angular velocity. Let the angle-axis representation of the perturbation be $\boldsymbol{v}$ and α, more compactly written as $\Delta\boldsymbol{\theta} = \alpha\boldsymbol{v}$. A first-order expansion of the fundamental rotation formula (10.28) is then

$$\Delta\boldsymbol{q} \approx \begin{bmatrix} 1 \\ \frac{1}{2}\Delta\boldsymbol{\theta} \end{bmatrix}. \tag{10.41}$$

Over an infinitesimal duration we can treat rotation as an additive quantity, leading to the relationship

$$\boldsymbol{\omega} = \lim_{\Delta t \to 0} \frac{\Delta\boldsymbol{\theta}}{\Delta t}. \tag{10.42}$$

We use this to express the time derivative of the quaternion as follows

$$\begin{aligned} \dot{\boldsymbol{q}} &= \lim_{\Delta t \to 0} \frac{\boldsymbol{q}(t+\Delta t) - \boldsymbol{q}(t)}{\Delta t} = \lim_{\Delta t \to 0} \frac{\boldsymbol{q} \otimes \Delta\boldsymbol{q} - \boldsymbol{q}}{\Delta t} \\ &= \lim_{\Delta t \to 0} \frac{\boldsymbol{q} \otimes \left(\begin{bmatrix} 1 \\ \Delta\boldsymbol{\theta}/2 \end{bmatrix} - \begin{bmatrix} 1 \\ \mathbf{0} \end{bmatrix} \right)}{\Delta t} = \lim_{\Delta t \to 0} \frac{\boldsymbol{q} \otimes \begin{bmatrix} 0 \\ \Delta\boldsymbol{\theta}/2 \end{bmatrix}}{\Delta t} = \frac{1}{2}\boldsymbol{q} \otimes \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix}. \end{aligned} \tag{10.43}$$

In our development of the ESKF we wil need to work with time derivatives of quaternion products such as those appearing in (10.40). Luckily, this is quite straightforward because the usual product rule still applies:

$$\frac{\mathrm{d}}{\mathrm{d}t}(\boldsymbol{q} \otimes \Delta\boldsymbol{q}) = \dot{\boldsymbol{q}} \otimes \Delta\boldsymbol{q} + \boldsymbol{q} \otimes \Delta\dot{\boldsymbol{q}}. \tag{10.44}$$

We must remember that the quaterion product is non-commutative, so that the two terms in (10.44) *cannot* be combined into, e.g., $2\dot{\boldsymbol{q}} \otimes \Delta\boldsymbol{q}$.

Let us also delve briefly into the alternative formulation of attitude kinematics by means of rotation matrices. The equivalent of the relationship $\tilde{\boldsymbol{q}} = \boldsymbol{q} \otimes \Delta\boldsymbol{q}$ is with rotation matrices expressed as

$$\tilde{\mathsf{R}} = \mathsf{R}\Delta\mathsf{R}$$

where $\Delta\mathsf{R}$ is a rotation matrix representing the attitude perturbation. From (10.37) and (10.41) we obtain a first-order approximation of $\Delta\mathsf{R}$ as

$$\Delta\mathsf{R} \approx (\mathbf{I} + \mathsf{S}(\Delta\boldsymbol{\theta})) \tag{10.45}$$

It is then possible, by steps very similar to those in (10.43) to find the time derivative of the rotation matrix as

$$\dot{\mathsf{R}} = \mathsf{R}\mathsf{S}(\boldsymbol{\omega}). \tag{10.46}$$

In the sequel we shall make use of the relationship (10.45). The differential equation (10.46) is, on the other hand, mentioned mainly for completeness, since it plays a central role in rotation matrix versions of the ESKF such as Gade's formulation [39]. See also Section 2.2.1 in [37].

10.2.2 Elements of the process model for inertial navigation

In this section we shall spell out our process model for inertial navigation. Our state vector will consist of position $\boldsymbol{\rho}$, velocity $\mathbf{v}$, attitude $\boldsymbol{q}$, accelerometer bias $\mathbf{a}_b$ and gyro bias $\boldsymbol{\omega}_b$. Since attitude is represented by quaternions, the state vector has a total of 13 states. Other parametrizations are also possible. This may include additional biases related to e.g. GNSS or compass. A particular peculiarity of the parametrization used in [100] is the inclusion of gravity in the state vector. This will not be pursued further here, since accurate models of gravity at any given latitude are easily available [46].

The differential equation for position is straightforward, given that the velocity vector also is decomposed in the world frame:

$$\dot{\boldsymbol{\rho}} = \mathbf{v}. \tag{10.47}$$

The differential equation for the velocity is equally simple if the acceleration $\mathbf{a}$ is available:

$$\dot{\mathbf{v}} = \mathbf{a}. \tag{10.48}$$

The differential equation for the attitude is more complicated, but has already been derived in (10.43). We proceed to look at models for the biases. For notational simplicity we will talk about a generic scalar bias b in this discussion. We will here consider three options. The first option is to model the bias as a Wiener process. This is the model used in [100], and represents a bias that is drifting indefinitely. It is given by the differential equation

$$\dot{b} = w \quad \text{where} \quad w \sim \mathcal{N}(0, \sigma^2 \delta(t-\tau)). \tag{10.49}$$

The second option is to model the bias as a Gauss-Markov process. This is reasonable if we can specify a mean square value σ^2_{ms} and a time constant T for the bias. If we define $p = 1/T$ the Gauss-Markov model is given by the differential equation

$$\dot{b} = -pb + w \quad \text{where} \quad w \sim \mathcal{N}(0, 2p\sigma^2_{\text{ms}} \delta(t-\tau)). \tag{10.50}$$

We shall use this model in the sequel. The reader may scroll back to Examples 4.5, 4.7 and 4.9 for further details and derivations concerning this model. The third option is to model the bias as an unknown constant. In this case the differential equation becomes

$$\dot{b} = 0. \tag{10.51}$$

10.3 True, nominal and error kinematics

We are now ready to become familiar with the two main ideas of the ESKF. The first idea can be summarized as propagation of a covariance that corresponds to the error state instead of the actual state.

In any kind of KF, our aim is to estimate the *true* state vector. However, since the truth is fundamentally unknown, we are only able to estimate and predict a *nominal* state vector. For the models that we have studied in previous chapters, the *error* state is simply the difference between these two state vectors. To quantify the performance of an estimator, such as the MMSE estimator that lies inherent in a Kalman filter, we use the covariance of the error state. So far, we have always been able to find this as the covariance of the nominal state, because the distribution of the nominal state minus the true state was the same as the distribution of the error state.

When we work with attitude, the relationship between true state, nominal state and error state is more complicated than a simple subtraction. The relationship involves a composition of rotations, which can be represented as a quaternion product such as in (10.40). Furthermore, it is possible and

Table 10.1: Elements in Sola's formulation of the ESKF

Magnitude	**True**	**Nominal**	**Error**	**Composition**	**Measured**	**Noise**
Position	$\boldsymbol{\rho}_t$	$\boldsymbol{\rho}$	$\delta\boldsymbol{\rho}$	$\boldsymbol{\rho}_t = \boldsymbol{\rho} + \delta\boldsymbol{\rho}$		
Velocity	$\mathbf{v}_t$	$\mathbf{v}$	$\delta\mathbf{v}$	$\mathbf{v}_t = \mathbf{v} + \delta\mathbf{v}$		
Orientation	$\boldsymbol{q}_t$	$\boldsymbol{q}$	$\delta\boldsymbol{q}$	$\boldsymbol{q}_t = \boldsymbol{q} \otimes \delta\boldsymbol{q}$		
Angles vector			$\delta\boldsymbol{\theta}$	$\delta\boldsymbol{q} \approx \begin{bmatrix} 1 \\ \delta\boldsymbol{\theta}/2 \end{bmatrix}$		
Accelerometer bias	$\mathbf{a}_{bt}$	$\mathbf{a}_b$	$\delta\mathbf{a}_b$	$\mathbf{a}_{bt} = \mathbf{a}_b + \delta\mathbf{a}_b$		$\mathbf{a}_w$
Gyro bias	$\boldsymbol{\omega}_{bt}$	$\boldsymbol{\omega}_b$	$\delta\boldsymbol{\omega}_b$	$\boldsymbol{\omega}_{bt} = \boldsymbol{\omega}_b + \delta\boldsymbol{\omega}_b$		$\boldsymbol{\omega}_w$
Acceleration	$\mathbf{a}_t$				$\mathbf{a}_m$	$\mathbf{a}_n$
Angular ratre	$\boldsymbol{\omega}_t$				$\boldsymbol{\omega}_m$	$\boldsymbol{\omega}_n$

also advisable to represent the nominal attitude and the error attitude using representations with different dimensions. On the one hand, we need a representation of the attitude itself with minimum 4 dimensions, both for reasons of mathematical convenience and to avoid singularities, even though attitude fundamentally is a 3-dimensional quantity. On the other hand, for the attitude error we want to avoid such an overparameterization, because the covariance would be difficult to handle. If we used a 4×4 covariance matrix, we would quickly find ourselves in a situation where the covariance matrix implies some uncertainty perpendicular to the surface of $SO(3)$. This is meaningless, as we know with absolute certainty that the attitude lies exactly on $SO(3)$. We could solve this by artificially enforcing zero convariance perpendicular to $SO(3)$, but we would then get a singular covariance matrix. Therefore, we always use a 3-dimensional representation of the attitude error. With a 13-dimensional nominal state vector, the covariance matrix gets dimension 12×12.

The propagation of the error state itself is actually never implemented. However, its process model governs how the covariance is propagated. During the measurement update, the error state is estimated, but this is immediately followed by a reset operation, which adjusts the nominal state accordingly. Thus, the expectation of the error state is zero all the time.

The second key idea of the ESKF is that the IMU measurements can be treated as control inputs instead of bona-fide measurements. This is preferable for two reasons. First, the IMU does typically provide measurements much more frequently than other sensors (100Hz or more) and running a full KF-update so often would be very expensive. Second, if the IMU data were to be processed according to a measurements model of the form $\mathbf{z} = \mathbf{h}(\mathbf{x}) + \mathbf{w}$ we would need to include accelerations and angular rates in the state vector. In addition to making the state vector unnecessarily large, we would have to provide a model for the time evolution of acceleration and angular rates, and such a model would most likely become notoriously unreliable. On the other hand, when we treat IMU data as control inputs, we can simply let the process noise be given by their sensor accuracies, and thus we avoid, at least in principle, the need for tuning of the process noise as we did in Section 4.5.

Following [100] we shall use the following subscripts in the mathematical developments that follow: m means measured value t means true value, b means bias, n means noise. Furthermore, we let δ signify an error state. If a state symbol is neither preceded by δ nor equipped with subscripts t, m or n, then it is part of the nominal state vector. For notational simplicity we skip any reference to time, except when we talk about discretization, for the remainder of this chapter. To summarize,

the true, nominal and error states that we consider here are

$$\mathbf{x}_t = \begin{bmatrix} \rho_t \\ \mathbf{v}_t \\ q_t \\ \mathbf{a}_{bt} \\ \omega_{bt} \end{bmatrix}, \ \mathbf{x} = \begin{bmatrix} \rho \\ \mathbf{v} \\ q \\ \mathbf{a}_b \\ \omega_b \end{bmatrix}, \ \delta\mathbf{x} = \begin{bmatrix} \delta\rho \\ \delta\mathbf{v} \\ \delta\theta \\ \delta\mathbf{a}_b \\ \delta\omega_b \end{bmatrix}. \tag{10.52}$$

10.3.1 The true state kinematics

To specify the true state kinematics mathematically we shall restate the equations from Section 10.2.2 in such a manner that the IMU measurements clearly appear as control inputs. The IMU data are given by

$$\mathbf{a}_m = \mathsf{R}^{\mathsf{T}}(q_t)(\mathbf{a}_t - \mathbf{g}) + \mathbf{a}_{bt} + \mathbf{a}_n \tag{10.53}$$
$$\omega_m = \omega_t + \omega_{bt} + \omega_n. \tag{10.54}$$

Here $\mathbf{g}$ is the gravity vector which is assumed known. The true rotation matrix R_t is given by the corresponding true attitude quaternion q_t. The disturbance of the accelerometer measurements has two components: The true bias $\mathbf{a}_{bt}$ and a white noise component $\mathbf{a}_n$. Similar comments apply to the angular velocity equation. The white noise processes are assumed Gaussian, and we denote their continuous-time covariances by $\mathbf{V}$ and $\boldsymbol{\Theta}$:

$$\begin{aligned} \mathbf{a}_n &\sim \mathcal{N}(\mathbf{0}, \mathbf{V}\delta(t-\tau)) \\ \omega_n &\sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Theta}\delta(t-\tau)). \end{aligned} \tag{10.55}$$

By rearranging (10.53) and (10.54) we find the following expressions for the kinematics of the true acceleration and angular velocity:

$$\begin{aligned} \mathbf{a}_t &= \mathbf{g} + \mathsf{R}(q_t)(\mathbf{a}_m + \mathbf{a}_{bt} + \mathbf{a}_n) \\ \omega_t &= \omega_m - \omega_{bt} - \omega_n. \end{aligned} \tag{10.56}$$

If we combine this with the process models from Section 10.2.2 we arrive at

$$\begin{aligned} \dot{\rho}_t &= \mathbf{v}_t \\ \dot{\mathbf{v}}_t &= \mathsf{R}(q_t)(\mathbf{a}_m - \mathbf{a}_{bt} - \mathbf{a}_n) + \mathbf{g} \\ \dot{q}_t &= \frac{1}{2} q_t \otimes (\omega_m - \omega_{bt} - \omega_n) \\ \dot{\mathbf{a}}_{bt} &= -p_{\mathbf{a}b}\mathbf{I}\,\mathbf{a}_{bt} + \mathbf{a}_w \\ \dot{\omega}_{bt} &= -p_{\omega b}\mathbf{I}\,\omega_{bt} + \omega_w \end{aligned} \tag{10.57}$$

where $p_{\mathbf{a}b}$ and $p_{\omega b}$ are parameters (inverse time constants) for the bias processes. The process model (10.57) is obviously a nonlinear model due to the many cross-terms in the equations for $\dot{\mathbf{v}}_t$ and $\dot{q}_t$.

10.3.2 The nominal state kinematics

The nominal state kinematics models how the system would behave if all the noise processes were zero. By simplifying the true state kinematics we immediately arrive at

$$\begin{aligned}\dot{\boldsymbol{\rho}} =& \mathbf{v} \\ \dot{\mathbf{v}} =& \mathsf{R}(\boldsymbol{q})(\mathbf{a}_m - \mathbf{a}_b) + \mathbf{g} \\ \dot{\boldsymbol{q}} =& \frac{1}{2}\boldsymbol{q} \otimes (\boldsymbol{\omega}_m - \boldsymbol{\omega}_b) \\ \dot{\mathbf{a}}_b =& -p_{\mathbf{a}b}\mathbf{I}\,\mathbf{a}_b \\ \dot{\boldsymbol{\omega}}_b =& -p_{\omega b}\mathbf{I}\,\boldsymbol{\omega}_b \end{aligned} \tag{10.58}$$

10.3.3 The error state kinematics

The error-state kinematics, both the equations and their derivations, are more complicated. It is sometimes believed that the error state automatically has linear kinematics. This is not true. Since the underlying system is nonlinear, approximations are needed to obtain a linear model for the error state. Different approximations are possible. In the subsequent theorem and its proof we study the approximation used by Sola in [100].

Theorem 10.3.1 The kinematics of the error state can by means of first-order approximations be expressed as

$$\begin{aligned}\delta\dot{\boldsymbol{\rho}} &= \delta\mathbf{v} \\ \delta\dot{\mathbf{v}} &= -\mathsf{R}(\boldsymbol{q})\mathsf{S}(\mathbf{a}_m - \mathbf{a}_b)\delta\boldsymbol{\theta} - \mathsf{R}(\boldsymbol{q})\delta\mathbf{a}_b - \mathsf{R}(\boldsymbol{q})\mathbf{a}_n \\ \delta\dot{\boldsymbol{\theta}} &= -\mathsf{S}(\boldsymbol{\omega}_m - \boldsymbol{\omega}_b)\delta\boldsymbol{\theta} - \delta\boldsymbol{\omega}_b - \boldsymbol{\omega}_n \\ \delta\dot{\mathbf{a}}_b &= -p_{\mathbf{a}b}\mathbf{I}\,\delta\mathbf{a}_b + \mathbf{a}_w \\ \delta\dot{\boldsymbol{\omega}}_b &= -p_{\omega b}\mathbf{I}\,\delta\boldsymbol{\omega}_b + \boldsymbol{\omega}_w. \end{aligned} \tag{10.59}$$

Proof. The expressions for $\delta\dot{\boldsymbol{\rho}}$, $\delta\dot{\mathbf{a}}_b$ and $\delta\dot{\boldsymbol{\omega}}_b$ are straightforward because the corresponding differential equations are linear both for the true state and the nominal state. On the other hand, the nonlinearities present in the differential equations for velocity and position mean that we will have to use nontrivial approximations to relate $\delta\dot{\mathbf{v}}$ and $\delta\dot{\boldsymbol{\theta}}$ to $\delta\mathbf{v}$ and $\delta\boldsymbol{\theta}$, respectively.
The linear velocity error: The true velocity derivative is given by

$$\dot{\mathbf{v}}_t = \mathbf{a}_t = \mathsf{R}_t(\mathbf{a}_m - \mathbf{a}_{bt} - \mathbf{a}_n) + \mathbf{g}_t = \mathsf{R}_t(\underbrace{\mathbf{a}_m - \mathbf{a}_b}_{\mathbf{a}_\beta} \underbrace{-\delta\mathbf{a}_b - \mathbf{a}_n}_{\delta\mathbf{a}_\beta}) + \mathbf{g}. \tag{10.60}$$

We have introduced the new quantity $\mathbf{a}_\beta$ which is the difference between the measured acceleration and the nominal accelerometer bias. We also combine the bias error and the accelerometer noise in the quantity $\delta\mathbf{a}_\beta$, where the δ-notation signifies an assumption that this quantity remains small also after the noise is added in. The nominal velocity derivative is given by

$$\dot{\mathbf{v}} = \mathsf{R}(\mathbf{a}_m - \mathbf{a}_n) + \mathbf{g} = \mathsf{R}\mathbf{a}_\beta + \mathbf{g}. \tag{10.61}$$

From the approximation that we introduced in (10.45) we have that $\mathsf{R}_t \approx \mathsf{R}(\mathbf{I} + \mathsf{S}(\delta\boldsymbol{\theta}))$. Inserting this in (10.60) and expanding into individual terms yields

$$\dot{\mathbf{v}}_t \approx \mathsf{R}(\mathbf{I} + \mathsf{S}(\delta\boldsymbol{\theta}))(\mathbf{a}_\beta + \delta\mathbf{a}_\beta) + \mathbf{g} = \mathsf{R}\mathbf{a}_\beta + \mathsf{R}\delta\mathbf{a}_\beta + \mathsf{RS}(\delta\boldsymbol{\theta})\mathbf{a}_\beta + \mathsf{RS}(\delta\boldsymbol{\theta})\delta\mathbf{a}_\beta + \mathbf{g}. \tag{10.62}$$

We invoke another approximation by assuming that the term $\mathsf{RS}(\delta\boldsymbol{\theta})\delta\mathbf{a}_\beta$ is negligible. Then it follows that

$$\begin{aligned}\delta\dot{\mathbf{v}} =&\dot{\mathbf{v}}_t - \dot{\mathbf{v}} \approx \mathsf{R}\left(\delta\mathbf{a}_\beta + \mathsf{S}(\delta\boldsymbol{\theta})\mathbf{a}_\beta\right) = \mathsf{R}\left(-\delta\mathbf{a}_b - \mathbf{a}_n - \mathsf{S}(\mathbf{a}_\beta)\delta\boldsymbol{\theta}\right)\\ =&-\mathsf{RS}(\mathbf{a}_m - \mathbf{a}_b)\delta\boldsymbol{\theta} - \mathsf{R}\delta\mathbf{a}_b - \mathsf{R}\mathbf{a}_n.\end{aligned} \tag{10.63}$$

The orientation error: Similarly to what we did for the linear velocity error, we define the new quantities $\boldsymbol{\omega} = \boldsymbol{\omega}_m - \boldsymbol{\omega}_b$ and $\delta\boldsymbol{\omega} = -\delta\boldsymbol{\omega}_b - \boldsymbol{\omega}_n$. These are somewhat artificial quantities, as they do not really belong in the nominal state and the error state, as the notation would seem to indicate. If we recall from (10.56) that $\boldsymbol{\omega}_t = \boldsymbol{\omega}_m - \boldsymbol{\omega}_{bt} - \boldsymbol{\omega}_n$ and expand $\boldsymbol{\omega}_{bt}$ into nominal and error terms, it can be seen that $\boldsymbol{\omega}_t = \boldsymbol{\omega} + \delta\boldsymbol{\omega}$. By using the fundamental quaternion time derivative formula (10.43) and the product rule (10.44) we can write the time derivative of the true attitude in two ways:

$$\dot{\boldsymbol{q}}_t = \frac{1}{2}\boldsymbol{q}_t \otimes \boldsymbol{\omega}_t = \frac{1}{2}\boldsymbol{q}\otimes\delta\boldsymbol{q}\otimes\boldsymbol{\omega}_t \tag{10.64}$$

$$= \dot{\boldsymbol{q}}\otimes\delta\boldsymbol{q} + \boldsymbol{q}\otimes\delta\dot{\boldsymbol{q}} = \frac{1}{2}\boldsymbol{q}\otimes\boldsymbol{\omega}\otimes\delta\boldsymbol{q} + \boldsymbol{q}\otimes\delta\dot{\boldsymbol{q}} \tag{10.65}$$

Notice that the short-hand notation from (10.36) for quaternion-vector products has been used here. Equating the expressions in (10.64) and (10.65) we can cancel the left-hand $\boldsymbol{q}$'s. From the last term in (10.65) we are then left with only $\delta\dot{\boldsymbol{q}}$, which accordingly must be given by

$$\begin{aligned}2\delta\dot{\boldsymbol{q}} =&\delta\boldsymbol{q}\otimes\boldsymbol{\omega}_t - \boldsymbol{\omega}\otimes\delta\boldsymbol{q} = \begin{bmatrix}0 & -\boldsymbol{\omega}_t^\mathsf{T}\\ \boldsymbol{\omega}_t & -\mathsf{S}(\boldsymbol{\omega}_t)\end{bmatrix}\delta\boldsymbol{q} - \begin{bmatrix}0 & -\boldsymbol{\omega}^\mathsf{T}\\ \boldsymbol{\omega} & \mathsf{S}(\boldsymbol{\omega})\end{bmatrix}\delta\boldsymbol{q}\\ \approx&\begin{bmatrix}0 & -(\boldsymbol{\omega}_t - \boldsymbol{\omega})^\mathsf{T}\\ (\boldsymbol{\omega}_t - \boldsymbol{\omega}) & -\mathsf{S}(\boldsymbol{\omega}_t + \boldsymbol{\omega})\end{bmatrix}\begin{bmatrix}1\\ \delta\boldsymbol{\theta}/2\end{bmatrix} = \begin{bmatrix}0 & -\delta\boldsymbol{\omega}^\mathsf{T}\\ \delta\boldsymbol{\omega} & -\mathsf{S}(2\boldsymbol{\omega} + \delta\boldsymbol{\omega})\end{bmatrix}\begin{bmatrix}1\\ \delta\boldsymbol{\theta}/2\end{bmatrix}\end{aligned} \tag{10.66}$$

If we equate the vector slot of $2\delta\dot{\boldsymbol{q}}$ with $\delta\dot{\boldsymbol{\theta}}$ we arrive at

$$\delta\dot{\boldsymbol{\theta}} \approx \delta\boldsymbol{\omega} - \mathsf{S}(2\boldsymbol{\omega})\delta\boldsymbol{\theta}/2 = -\mathsf{S}(\boldsymbol{\omega}_m - \boldsymbol{\omega}_b)\delta\boldsymbol{\theta} - \delta\boldsymbol{\omega}_b - \boldsymbol{\omega}_n. \tag{10.67}$$

■

10.4 Time discretization in the ESKF

To implement an ESKF, we must of course translate the continuous time kinematics into discrete-time kinematics. We reiterate that the only state vector that is actually predicted is the nominal state, while the model for the error state only is needed for prediction of the covariance.

10.4.1 The nominal state

The continuous-time nominal state model (10.58) contains nonlinearities due to the presence of the rotation matrix $\mathsf{R}(\boldsymbol{q})$, the product $\mathsf{R}(\boldsymbol{q})(\mathbf{a}_m - \mathbf{a}_b)$ between the rotation matrix and measurements and biases of acceleration, and because of the product $\boldsymbol{q}\otimes(\boldsymbol{\omega}_m - \boldsymbol{\omega}_b)$ between the quaternion and the measurements and biases of angular velocity. We have no closed-form solution to this discretization, and we must choose a numerical integration technique. This can be done by your favourite ordinary differential equation (ODE) solver. If noises are high, typically due to a budget-price IMU, then there is hardly any reason to feel guilty for resorting to a basic Euler method. For a general treatment of numerical solution of differential equations the reader is referred to [54].

10.4.2 The error state

The error state kinematics (10.59) constitute a linear time-varying (LTV) system, subject to the approximations utilized in the subsequent proof. To see this, notice that all the δ-terms in (10.59) always occur as a vector in a matrix-vector product, where the matrices depend on the nominal state vector through $\boldsymbol{q}$ and $\boldsymbol{\omega}_b$. As an LTV system, we can rewrite (10.59) by means of a system matrix that depends on the nominal state:

$$\delta\dot{\mathbf{x}} = \underbrace{\begin{bmatrix} \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & -\mathsf{R}(\boldsymbol{q})\mathsf{S}(\mathbf{a}_m - \mathbf{a}_b) & -\mathsf{R}(\boldsymbol{q}) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & -\mathsf{S}(\boldsymbol{\omega}_m - \boldsymbol{\omega}_b) & \mathbf{0} & -\mathbf{I} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & -p_{\mathbf{a}b}\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -p_{\boldsymbol{\omega}b}\mathbf{I} \end{bmatrix}}_{\mathbf{A}(\mathbf{x})} \delta\mathbf{x} + \underbrace{\begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ -\mathsf{R}(\boldsymbol{q}) & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix}}_{\mathbf{G}(\mathbf{x})} \mathbf{n} \tag{10.68}$$

We recall that $\delta\mathbf{x} = [\delta\boldsymbol{\rho}, \delta\boldsymbol{\theta}, \delta\mathbf{v}, \delta\mathbf{a}_b, \delta\boldsymbol{\omega}_b]^\mathsf{T}$. The process noise vector $\mathbf{n}$ consists of the noises on the IMU inputs, known as $\mathbf{a}_n$ and $\boldsymbol{\omega}_n$, and the driving noises of the bias processes, known as $\mathbf{a}_w$ and $\boldsymbol{\omega}_w$. We denote their continuous-time covariance matrices $\tilde{\mathbf{V}}$, $\tilde{\boldsymbol{\Theta}}$, $\tilde{\mathbf{W}}$ and $\tilde{\boldsymbol{\Omega}}$, and we assume that all these noise processes are white:

$$\mathbf{n} \sim \mathcal{N}\left(\mathbf{0}_{12\times 1}, \tilde{\mathbf{Q}}\right) \quad \text{where} \quad \tilde{\mathbf{Q}} = \text{blkdiag}\left(\tilde{\mathbf{V}}, \tilde{\boldsymbol{\Theta}}, \tilde{\mathbf{A}}, \tilde{\boldsymbol{\Omega}}\right). \tag{10.69}$$

If the quantities $\boldsymbol{q}$, $\mathbf{a}_m - \mathbf{a}_b$ and $\boldsymbol{\omega}_m - \boldsymbol{\omega}_b$ are constant over the time interval, then we can discretize (10.68) exactly using standard techniques from Section 4.5. If we predict the covariance using the discrete-time version of (10.68) with the same frequency as the IMU measurements arrive, then this will most likely be a reasonable approximation. However, it is often desirable to do this less frequently, perhaps only when a new position, velocity or attitude measurements is received. Then one may replace $\mathbf{a}_m - \mathbf{a}_b$ and $\boldsymbol{\omega}_m - \boldsymbol{\omega}_b$ with averages of these quantities over the time interval [90].

The most straightforward discretization approach is to use Van Loan's formula (4.63). This presupposes that a sufficiently fast and robust implementation of the matrix exponential is available. If that is not the case, one can evaluate the transition matrix corresponding to $\mathbf{A}(\mathbf{x})$ as a Taylor series truncated after a handful of terms. For details the reader is referred to [100].

10.4.3 Tuning of the noise processes

It is important to understand that the noise processes involved in (10.68) are of two very different natures, because it affects how their covariances should be tuned.

On the one hand, $\mathbf{a}_n$ and $\boldsymbol{\omega}_n$ are continuous-time representations of the IMU measurement noises. Sola refers to these as the "noise levels in the control signals". If we know the actual noise characteristics of the discrete time IMU measurement noises, we can find the covariances of $\mathbf{a}_n$ and $\boldsymbol{\omega}_n$ according to

$$\tilde{\mathbf{V}} = \sigma_{an}^2\mathbf{I}/\Delta t \quad \text{and} \quad \tilde{\boldsymbol{\Theta}} = \sigma_{\omega n}^2\mathbf{I}/\Delta t \tag{10.70}$$

where Δt is the IMU's sampling time and σ_{an} and σ_{an} are the discrete-time standard deviations of accelerometer and gyro (assumed isotropic). The division by Δt is a consequence of the properties of the Wiener process and can be traced back to (4.15). More precisely, if we want to model the discrete-time IMU noise over a given time span by means of a continuous-time white noise process, then then every discrete-time noise value should be the integral of the continuous-time noise process over that time interval, and, as we showed in Example 4.2, that variance of that integral is a factor Δt times the continuous-time noise variance.

On the other hand, $\mathbf{a}_{bn}$ and $\boldsymbol{\omega}_{bn}$ are white noise processes driving the IMU biases. For the Gauss-Markov bias model, we have already established the tuning formula in Example 4.9: The

variance of the continuous time white noise process should be 2 times the mean square bias divided by the time constant of the linear filter that is used to model the bias.

10.5 Measurement update in the ESKF

The filter that we so far have described performs dead reckoning. It will have an inevitable drift in position and velocity estimates, which only can be counteracted by position measurements (from e.g., GNSS), possibly supplemented by velocity measurements (from e.g., DVL or odometry). As for the attitude, the presence of gravity means that the filter solely based on the IMU will have some idea about roll and pitch. The yaw angle, on the other hand, can only be estimated by means of additional measurement (from e.g., compass). The IMU biases can be estimated without further measurements by means of maneuvering, but are unobservable if the vehicle is at rest or moving with a constant velocity.

Measurements such as GNSS, odometry or compass can be related to the state vector using models of the form

$$\mathbf{z} = \mathbf{h}(\mathbf{x}_t) + \mathbf{w} \ , \ \mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}) \tag{10.71}$$

under the convenient assumption of Gaussian measurement noise. Notice that the measurement model specifies how the measurement relates to the *true state*, and not to the nominal state or the error state. However, our prior statistical description, in terms of a covariance matrix, is of the *error state*. What we want is a way of using the model (10.71) to estimate the error state. To achieve this, we spell out the relationship between the true state and the error state, which can be expressed as

$$\mathbf{x}_t = \mathbf{x} \oplus \delta\mathbf{x} = \begin{bmatrix} \rho + \delta\rho \\ \mathbf{v} + \delta\mathbf{v} \\ q \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\delta\theta \end{bmatrix} \\ \mathbf{a}_b + \delta\mathbf{a}_b \\ \boldsymbol{\omega}_b + \delta\boldsymbol{\omega}_b \end{bmatrix}. \tag{10.72}$$

In this context, the notation $\oplus$ signifies an operation that is a conventional addition for the the state components except the attitude, for which it instead is a quaternion product with a term of the same form as (10.41). Using this notation, we can rewrite (10.71) as

$$\mathbf{z} = \mathbf{h}(\mathbf{x} \oplus \delta\mathbf{x}) + \mathbf{w} \ , \ \mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}). \tag{10.73}$$

Since the error state is supposed to be small we linearize this around $\delta\mathbf{x} = \mathbf{0}$. This yields

$$\mathbf{z} \approx \mathbf{h}(\mathbf{x} \oplus \mathbf{0}) + \left.\frac{\partial \mathbf{h}(\mathbf{x} \oplus \delta\mathbf{x})}{\partial \delta\mathbf{x}}\right|_{\delta\mathbf{x}=\mathbf{0}} \delta\mathbf{x} + \mathbf{w} = \mathbf{h}(\mathbf{x}) + \mathbf{H}\delta\mathbf{x} + \mathbf{w} \tag{10.74}$$

where we have introduced to conventional short-hand notation $\mathbf{H}$ for the Jacobian in the rightmost expression. This is a measurement model that fits straight into an EKF framework. We can write the EFK update in terms of the familiar equations

$$\begin{aligned} \mathbf{W} &= \mathbf{P}\mathbf{H}^\mathsf{T}(\mathbf{H}\mathbf{P}\mathbf{H}^\mathsf{T} + \mathbf{R})^{-1} \\ \delta\hat{\mathbf{x}} &\leftarrow \mathbf{W}(\mathbf{z} - \mathbf{h}(\mathbf{x})) \\ \mathbf{P} &\leftarrow (\mathbf{I} - \mathbf{W}\mathbf{H})\mathbf{P}. \end{aligned} \tag{10.75}$$

To say something more about the Jacobian $\mathbf{H}$, we notice that $\mathbf{z}$ depends on $\delta\mathbf{x}$ through a composition of two nonlinear functions, namely $\mathbf{h}(\cdot)$ and $\mathbf{x} \oplus \delta\mathbf{x}$. Consequently, we can use the Chain Rule to

express it as a product of Jacobians of these two functions:

$$\mathbf{H} = \frac{\partial}{\partial \mathbf{x}_t}\mathbf{h}\bigg|_{\mathbf{x}_t=\mathbf{x}} \cdot \frac{\partial}{\partial \delta\mathbf{x}}\mathbf{x}_t\bigg|_{\mathbf{x}_t=\mathbf{x}} \triangleq \mathbf{H}_\mathbf{x}\mathbf{X}_{\delta\mathbf{x}}. \tag{10.76}$$

The matrix $\mathbf{H}_\mathbf{x}$ depends on the measurement model. The matrix $\mathbf{X}_{\delta\mathbf{x}}$ has on the other hand a fixed expression that only depends on the choice of the state vector. For our choice it is

$$\mathbf{X}_{\delta\mathbf{x}} = \begin{bmatrix} \mathbf{I}_6 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_{\delta\boldsymbol{\theta}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_6 \end{bmatrix} \tag{10.77}$$

where

$$\begin{aligned} \mathbf{Q}_{\delta\boldsymbol{\theta}} &\approx \frac{\partial}{\partial \delta\boldsymbol{\theta}}(\boldsymbol{q}\otimes\delta\boldsymbol{q})\bigg|_{\boldsymbol{q}} \\ &= \frac{\partial}{\partial \delta\boldsymbol{q}}(\boldsymbol{q}\otimes\delta\boldsymbol{q})\bigg|_{\boldsymbol{q}} \frac{\partial}{\partial \delta\boldsymbol{\theta}}(\delta\boldsymbol{q})\bigg|_{\delta\boldsymbol{\theta}=0} \\ &= \frac{1}{2}\begin{bmatrix} -\varepsilon_1 & -\varepsilon_2 & -\varepsilon_3 \\ \eta & -\varepsilon_3 & \varepsilon_2 \\ \varepsilon_3 & \eta & -\varepsilon_1 \\ -\varepsilon_2 & \varepsilon_1 & \eta \end{bmatrix}. \end{aligned} \tag{10.78}$$

We see that a Chain Rule is also present in the evaluation of $\mathbf{Q}_{\delta\boldsymbol{\theta}}$. This is because we have to link the error quaternion to its corresponding error angle. The final expression for $\mathbf{Q}_{\delta\boldsymbol{\theta}}$ is written in terms of the components of the nominal attitude quaternion $\boldsymbol{q}$. The details of the derivation are left as an exercise to the reader. In the following couple of examples we take a look at what the sensor-dependent Jacobian $\mathbf{H}_\mathbf{x}$ can be for different sensor models.

■ **Example 10.1 — Jacobian for standard GNSS measurements.** Standard off-the-shelf GNSS receivers provide a direct measurement of position. The measurement model the becomes

$$\mathbf{z} = \mathbf{H}_\mathbf{x}\mathbf{x}_t + \mathbf{w} \tag{10.79}$$

where

$$\mathbf{H}_\mathbf{x} = \begin{bmatrix} \mathbf{I}_3 & \mathbf{0}_{3\times 13} \end{bmatrix}. \tag{10.80}$$

■

We remark that this measurement model is often known as loose integration in the GNSS literature, in contrast to so-called tight integration. In reality, a GNSS receiver calculates so-called pseudoranges which indicate the distances to different satellites. The are called pseudoranges, and not merely ranges, because they area contaminated with an unknown clock offset, which also must be estimated. The position $\mathbf{z}$ is calculated by means of trilateration using the pseudoranges. Thus, the pseudoranges are not used directly in the measurement model. Using them as part of the measurement model is known as a tight integration. For further details the reader is recommended to follow the course TTK5.

■ **Example 10.2 — Jacobian for a magnetic compass.** In this example, we consider a compass that only measures the direction of the magnetic field lines. In other words, the compass measures how the unit vector pointing towards north is decomposed in the body frame. This measurement model can be written as $\mathbf{z} = \mathbf{h}(\boldsymbol{q}) + \mathbf{w}$ where

$$\mathbf{h}(\boldsymbol{q}) = \boldsymbol{q}^*\mathbf{e}_x\boldsymbol{q} = \eta^2\mathbf{e}_x + 2\eta(\mathbf{e}_x\times\boldsymbol{\epsilon}) + (\mathbf{e}_x^\mathsf{T}\boldsymbol{\epsilon})\boldsymbol{\epsilon} - \boldsymbol{\epsilon}\times\mathbf{e}_x\times\boldsymbol{\epsilon} \tag{10.81}$$

and where $\mathbf{e}_x = [1,0,0]^\mathsf{T}$. Notice that in the quaternion rotation formula in (10.81) we premultiply with the conjugate attitude quaternion, which is opposite of the standard rotation formula from Theorem 10.1.2, where the conjugate quaternion appeared in the postmultiplication. This is because we here take a vector from the world frame and decompose it in the body frame, while the quaternion according to the passive body-to-world convention is defined as something that takes a vector from the body frame and decomposes it in the world frame.

We immediately see that the measurement Jacobian $\mathbf{H}_\mathbf{x}$ only will include derivatives with respect to the attitude quaternion. It is straightforward to find the derivative with respect to its scalar component:

$$\frac{\partial \mathbf{z}}{\partial \eta} = 2\eta \mathbf{e}_x + 2\mathbf{e}_x \times \boldsymbol{\epsilon}. \tag{10.82}$$

The derivative with respect to $\boldsymbol{\epsilon}$ is a bit more cumbersome to evaluate. Eventually we arrive at

$$\frac{\partial \mathbf{z}}{\partial \boldsymbol{q}} = \begin{bmatrix} 2\eta & 2\varepsilon_1 & -2\varepsilon_2 & -2\varepsilon_3 \\ -2\varepsilon_2 & 2\varepsilon_2 & 2\varepsilon_1 & -2\eta \\ 2\varepsilon_2 & 2\varepsilon_3 & 2\eta & 2\varepsilon_1 \end{bmatrix}. \tag{10.83}$$

Again, the detailed calculations are left as an exercise for the reader. The complete expression for the Jacobian $\mathbf{H}_\mathbf{x}$ becomes

$$\mathbf{H}_\mathbf{x} = \begin{bmatrix} \mathbf{0}_{3\times 6} & \dfrac{\partial \mathbf{z}}{\partial \boldsymbol{q}} & \mathbf{0}_{3\times 6} \end{bmatrix}. \tag{10.84}$$

■

This last example may perhaps be a bit artificial. Compasses typically provide complete attitude measurements. This could for example be in terms of Euler angles. However, also for such a measurement a Jacobian must be calculated, e.g., for the mapping (10.38) from quaternions to Euler angles.

10.5.1 Injection and reset

Obtaining an estimate $\delta\hat{\mathbf{x}}$ of the error state, does not in itself leave us where we want to be at the end of the estimation cycle. After all, we have no intention of propagating the error state in addition to the nominal state. It makes little sense to say that the error is so and so large, without taking the consequence of this and adjust the nominal state accordingly. This leads to the injection step

$$\mathbf{x} \leftarrow \mathbf{x} \oplus \delta\hat{\mathbf{x}}. \tag{10.85}$$

After this is done, we have moved all the information contained in $\delta\hat{\mathbf{x}}$ to the nominal state, and the expectation of the error state once again becomes zero. It remains zero until we perform a new measurement update.

Since the nominal orientation $\boldsymbol{q}$ has changed due to the injection, the covariance of $\delta\boldsymbol{\theta}$ must be modified accordingly. This is so because $\delta\boldsymbol{\theta}$ is expressed locally relative to $\boldsymbol{q}$. This is the last major result in Sola's development of the ESKF, and we state it as a theorem with subsequent proof.

Theorem 10.5.1 — The ESKF covariance reset. After the injection step (10.85) the error state covariance must be updated according to

$$\mathbf{P} \leftarrow \mathbf{G}\mathbf{P}\mathbf{G}^\mathsf{T} \text{ where } \mathbf{G} = \begin{bmatrix} \mathbf{I}_6 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} - \mathsf{S}\left(\frac{1}{2}\delta\hat{\boldsymbol{\theta}}\right) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_6 \end{bmatrix}. \tag{10.86}$$

Proof. Let us denote the states after injection and reset by a plus sign, e.g., $\boldsymbol{q}^+$. We have the following equalities

$$\boldsymbol{q}^+ \otimes \delta\boldsymbol{q}^+ = \boldsymbol{q} \otimes \delta\boldsymbol{q} \tag{10.87}$$
$$\boldsymbol{q}^+ = \boldsymbol{q} + \delta\hat{\boldsymbol{q}} \tag{10.88}$$

The first equality holds because both sides equal the true attitude, which is unaffected by the injection. The second equality is just a restatement of the attitude part of the injection step. From these equations it follows that

$$\begin{aligned}\delta\boldsymbol{q}^+ &= \left(\boldsymbol{q}^+\right)^* \otimes \boldsymbol{q} \otimes \delta\boldsymbol{q} \\ &= (\boldsymbol{q} \otimes \delta\hat{\boldsymbol{q}})^* \otimes \boldsymbol{q} \otimes \delta\boldsymbol{q} \\ &= \delta\hat{\boldsymbol{q}}^* \otimes \boldsymbol{q}^* \otimes \boldsymbol{q} \otimes \delta\boldsymbol{q} \\ &= \delta\hat{\boldsymbol{q}}^* \otimes \delta\boldsymbol{q}\end{aligned} \tag{10.89}$$

In accordance with (10.41) we use the approximation

$$\delta\hat{\boldsymbol{q}}^* \approx \begin{bmatrix} 1 \\ -\frac{1}{2}\delta\hat{\boldsymbol{\theta}} \end{bmatrix}. \tag{10.90}$$

We insert this into (10.89), and spell out the details by means of the matrix representation of the quaternion product from (10.34):

$$\begin{bmatrix} 1 \\ \frac{1}{2}\delta\boldsymbol{\theta}^+ \end{bmatrix} \approx \begin{bmatrix} 1 & \frac{1}{2}\delta\boldsymbol{\theta}^\mathsf{T} \\ -\frac{1}{2}\delta\boldsymbol{\theta} & \mathbf{I} - \mathsf{S}(\frac{1}{2}\delta\hat{\boldsymbol{\theta}}) \end{bmatrix} \begin{bmatrix} 1 \\ \frac{1}{2}\delta\boldsymbol{\theta} \end{bmatrix}. \tag{10.91}$$

We do not pay any attention to the scalar part of this equation: It only states that one is approximately equal to one plus a second-order term, which we neglect. The vector part is where the interesting stuff happens:

$$\delta\boldsymbol{\theta}^+ \approx -\frac{1}{2}\delta\hat{\boldsymbol{\theta}} + \left(\mathbf{I} - \mathsf{S}\left(\frac{1}{2}\delta\hat{\boldsymbol{\theta}}\right)\right)\delta\boldsymbol{\theta}. \tag{10.92}$$

There are three error angles present in (10.92). Of these, both $\delta\boldsymbol{\theta}^+$ and $\delta\boldsymbol{\theta}$ are random quantities, conditional on the observed data, because both describe the residual between the nominal attitude and the unknown true attitude in (10.87). On the other hand, $\delta\hat{\boldsymbol{\theta}}$ is not random, because it is an estimate that is entirely given by the data and the initial conditions. Before the reset, $\delta\boldsymbol{\theta}$ had the expectation $\delta\hat{\boldsymbol{\theta}}$ and the covariance $\mathbf{P}$ as given by (10.75). From (10.92) it the follows that the covariance of $\delta\boldsymbol{\theta}^+$ can be found by pre- and postmultiplication with

$$\mathbf{I} - \mathsf{S}\left(\frac{1}{2}\delta\hat{\boldsymbol{\theta}}\right). \tag{10.93}$$

This leads directly to (10.86). ■

10.6 References and chapter remarks

This introduction to the quaternion formulation of the ESKF is based on Sola's eminent text on the topic [100]. If the reader instead would prefer a formulation of the ESKF that uses rotation matrices, a good starting point would be Gade's MSc thesis [39], which is the key reference for most work on inertial navigation in maritime Norwegian industry. Students who plans to specialize

in inertial navigation may want to read both [39] and [100]. The ESKF is also extensively studied by Roumeliotis and coworkers, see e.g. [90] and [107]. It should be noted that these three main sources, Sola, Gade and Roumeliotis, use rather different notations. In particular, Sola uses a minimalist notation, with the goal of avoiding any clutter that could confuse the reader. Gade follows the opposite principle, by striving for a complete, unambiguous and consistent notation, which is further developed in [41]. Roumeliotis is a middle ground between these two extremes. Sola uses the passive body-to-world convention. Gade uses the active convention, which is mathematically equivalent to Sola's convention. Roumeliotis uses the passive world-to-body convention. This is also the convention being used at NASA JPL. For fundamentals on attitude representations the reader may also consult [37], which uses the same rotation convention as [100] and [39]. The proof of the quaternion rotation formula in Theorem 10.1.2 is inspired by [45].

The goal of this chapter has been to introduce the fundamental framework of the ESKF. Several important topics for practical application have not been covered. Such topics include observability analysis, filter tuning and lever arm compensation. Observability analysis is important to understand what can be be estimated and what cannot be estimated in a navigation system. For example, thanks to the observability properties of an inertial navigation system operating on the surface of the earth heading can be estimated in several different ways [40]. Observability analysis for INS is extensively treated in the research literature, see e.g., [50]. The reader is referred to [105] for the latter two topics.

10.7 Exercises

Exercise 10.1 Derive the rotation matrix formula (10.4) from Rodrigues' formula (10.1). ■

Exercise 10.2 As an alternative to the vector form in (10.19) quaternions are sometimes written in the form $\boldsymbol{q} = a + bi + cj + dk$ where i, j and k are "hyperimaginary" numbers which together with the real unit 1 span a 4-dimensional space. We related this formulation to the vectorial formulation by setting $\eta = a$ and $\boldsymbol{\epsilon} = [b,c,d]^\mathsf{T}$.

1. Show that the definition of the quaternion product in (10.21) yields Hamilton's famous relationship

$$i^2 = j^2 = k^2 = ijk = -1.$$

2. Show that this again leads to the relationship

$$ij = -ji = k.$$

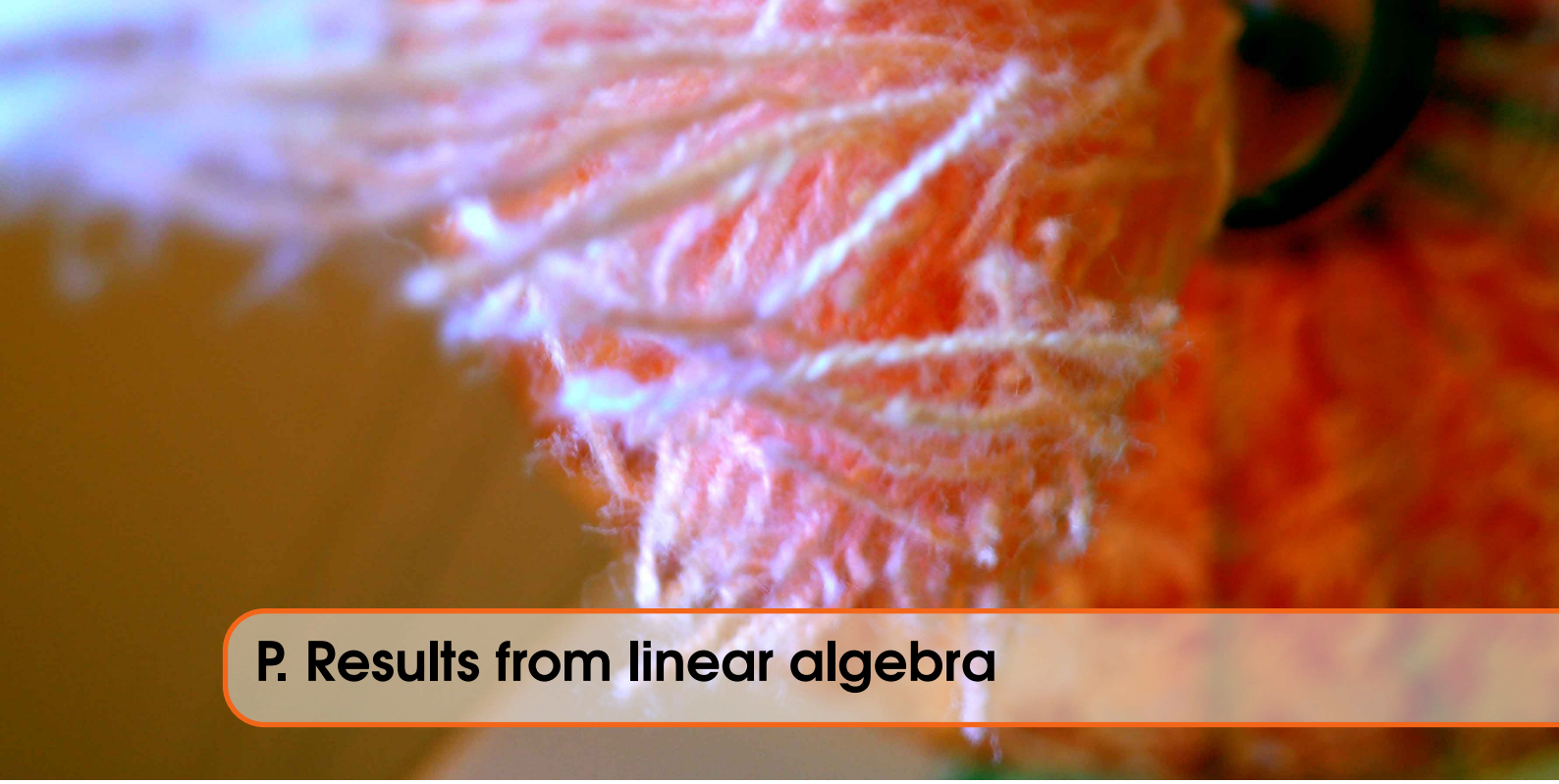
■

Exercise 10.3 Derive the rotation matrix differential equation $\dot{\mathsf{R}} = \mathsf{R}\mathsf{S}(\boldsymbol{\omega})$ from the first-order approximation $\Delta\mathsf{R} \approx (\mathbf{I} + \mathsf{S}(\Delta\boldsymbol{\theta}))$ in (10.45). ■

Exercise 10.4 Derive the formula for the error-angle-to-true-quaternion Jacobian $\mathbf{X}_{\delta\mathbf{x}}$ in (10.78). ■

Exercise 10.5 Derive the formula for the magnetic compass Jacobian (10.83). ■

Exercise 10.6 Find the Jacobian of the quaternion-to-Euler-angle mapping (10.38). ■



P. Results from linear algebra

P.1 The Schur complement and Boltz' inversion rule

Let the matrix $\mathbf{M}$ be given by

$$\mathbf{M} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}. \tag{P.1}$$

The Schur complement of the block $\mathbf{D}$ in $\mathbf{M}$ is $\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C}$, and the Schur complement of the block $\mathbf{A}$ in $\mathbf{M}$ is $\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B}$. The Schur complement plays a central role when we invert matrices of the form (P.1). With the purpose of obtaining a general formula for inverting such matrices, let us also introduce the matrices

$$\mathbf{L} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{D}^{-1}\mathbf{C} & \mathbf{I} \end{bmatrix}, \quad \mathbf{U} = \begin{bmatrix} \mathbf{I} & -\mathbf{B}\mathbf{D}^{-1} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \quad \text{and} \quad \mathbf{N} = \begin{bmatrix} \mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C} & \mathbf{0} \\ \mathbf{0} & \mathbf{D} \end{bmatrix}. \tag{P.2}$$

Obviously, the goal here is to replace the inversion of a complicated matrix $\mathbf{M}$ with inversion of a simpler block-diagonal matrix $\mathbf{N}$ which involves the Schur complement.

Theorem P.1.1 — Blockwise matrix inversion. With $\mathbf{M}$, $\mathbf{L}$, $\mathbf{U}$ and $\mathbf{N}$ defined in (P.1) and (P.2), we have the identity $\mathbf{M}^{-1} = \mathbf{L}\mathbf{N}^{-1}\mathbf{U}$.

Proof. By reverse engineering of the desired result, we find that $\mathbf{L}^{-1}\mathbf{M}^{-1} = \mathbf{N}^{-1}\mathbf{U}$ should hold, and furthermore that $\mathbf{M}\mathbf{L} = \mathbf{U}^{-1}\mathbf{N}$ should hold. The left-hand-side of this identity becomes

$$\mathbf{M}\mathbf{L} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{D}^{-1}\mathbf{C} & \mathbf{I} \end{bmatrix} = \begin{bmatrix} \mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C} & \mathbf{B} \\ \mathbf{0} & \mathbf{D} \end{bmatrix}.$$

For the right-hand-side, we need to invert $\mathbf{U}$. We leave it as an exercise to the reader to show that

$$\mathbf{U}^{-1} = \begin{bmatrix} \mathbf{I} & \mathbf{B}\mathbf{D}^{-1} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}.$$

Thus, the right-hand-side becomes

$$\mathbf{U}^{-1}\mathbf{N} = \begin{bmatrix} \mathbf{I} & \mathbf{B}\mathbf{D}^{-1} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{A}-\mathbf{B}\mathbf{D}^{-1}\mathbf{C} & \mathbf{0} \\ \mathbf{0} & \mathbf{D} \end{bmatrix} = \begin{bmatrix} \mathbf{A}-\mathbf{B}\mathbf{D}^{-1}\mathbf{C} & \mathbf{B} \\ \mathbf{0} & \mathbf{D} \end{bmatrix}.$$

This concludes the proof. ■

Having established this important result, we can express the inverse of a block matrix in two ways which both are useful:

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{D}^{-1}\mathbf{C} & \mathbf{I} \end{bmatrix} \begin{bmatrix} (\mathbf{A}-\mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{I} & -\mathbf{B}\mathbf{D}^{-1} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \tag{P.3}$$

$$= \begin{bmatrix} (\mathbf{A}-\mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1} & -(\mathbf{A}-\mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1}\mathbf{B}\mathbf{D}^{-1} \\ -\mathbf{D}^{-1}\mathbf{C}(\mathbf{A}-\mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1} & \mathbf{D}^{-1}+\mathbf{D}^{-1}\mathbf{C}(\mathbf{A}-\mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1}\mathbf{B}\mathbf{D}^{-1} \end{bmatrix} \tag{P.4}$$

The first of these two equations (P.3) is known as Aitken's block diagonalization formula, while the last equation (P.4) is known as Boltz' rule for matrix inversion.

P.2 The matrix inversion lemma

In Boltz' rule (P.4) it can be noted that $\mathbf{A}$ and $\mathbf{D}$ are treated somewhat differently. This is because we chose to express $\mathbf{N}$ in terms of the Schur complement of $\mathbf{D}$. If we instead choose to work with the Schur complement of $\mathbf{A}$, symmetry dictates that we must arrive at

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{A}^{-1}+\mathbf{A}^{-1}\mathbf{B}(\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{C}\mathbf{A}^{-1} & -\mathbf{A}^{-1}\mathbf{B}(\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1} \\ -(\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{C}\mathbf{A}^{-1} & (\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1} \end{bmatrix}. \tag{P.5}$$

The matrix inversion lemma, also known as the Woodbury identity, follows from equating the upper diagonal blocks in (P.4) and (P.5):

$$(\mathbf{A}-\mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1} = \mathbf{A}^{-1}+\mathbf{A}^{-1}\mathbf{B}(\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{C}\mathbf{A}^{-1}. \tag{P.6}$$

The matrix inversion lemma is often invoked in derivations of the Kalman filter.

P.3 Rodrigues' rotation formula

Rodrigues' formula was given in (10.1) and formed the basis for our development of rotation matrices and Euler angles. We repeat it here:

$$\mathbf{v}' = (1-\cos\alpha)(\mathbf{n}\cdot\mathbf{v})\mathbf{n}+\cos\alpha\mathbf{v}-\sin\alpha(\mathbf{v}\times\mathbf{n}). \tag{P.7}$$

To derive this formula with a minimal amount of effort, let us first convince ourselves that it is invariant to the choice of coordinate system. Let $\mathbf{T}$ be a matrix whose rows are mutually orthogonal, all with Euclidean norm 1. Such a matrix is of course a 3-dimensional rotation matrix, but let us for now only think of it as a transformation matrix that changes the basis of our vectors. We can rest assured that

$$\mathbf{T}\mathbf{v}' = (1-\cos\alpha)((\mathbf{T}\mathbf{n})\cdot(\mathbf{T}\mathbf{v}))(\mathbf{T}\mathbf{n})+\cos\alpha(\mathbf{T}\mathbf{v})-\sin\alpha((\mathbf{T}\mathbf{v})\times(\mathbf{T}\mathbf{n})). \tag{P.8}$$

This holds because it generally holds that

$$\begin{aligned} (\mathbf{T}\mathbf{n})\cdot(\mathbf{T}\mathbf{v}) &= \mathbf{n}\cdot\mathbf{v} \\ (\mathbf{T}\mathbf{v})\times(\mathbf{T}\mathbf{n}) &= \mathbf{T}(\mathbf{v}\times\mathbf{n}) \end{aligned} \tag{P.9}$$

insofar as $\mathbf{T}$ is an orthogonal matrix. As a consequence of this, if Rodrigues' formula holds for the transformation between $\mathbf{v}$ and $\mathbf{v}'$, then it must also hold for the transformation between $\mathbf{Tv}$ and $\mathbf{Tv}'$ insofar as $\mathbf{T}$ is an orthogonal matrix. Let us then choose our orthogonal matrix such that

$$\mathbf{T} = \begin{bmatrix} - \\ - \\ \mathbf{n}^\mathsf{T} \end{bmatrix}$$

where the first two rows can be any orthonormal row vectors that also are orthogonal to $\mathbf{n}^\mathsf{T}$. Let us then define $\mathbf{w} = \mathbf{Tv}$, $\mathbf{w}' = \mathbf{Tv}'$ and

$$\mathbf{m} = \mathbf{Tn} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}.$$

With this choice of $\mathbf{T}$, and expanding $\mathbf{w} = [w_x, w_y, w_z]^\mathsf{T}$, (P.9) turns into

$$\begin{aligned} \mathbf{w}' =& (1-\cos\alpha)\left(\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}\mathbf{w}\right) + \cos\alpha\mathbf{w} - \sin\alpha\left(\mathbf{w} \times \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}\right) \\ =& (1-\cos\alpha)\begin{bmatrix} 0 \\ 0 \\ w_z \end{bmatrix} + \cos\alpha\begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix} - \sin\alpha\begin{bmatrix} w_y \\ -w_x \\ 0 \end{bmatrix} \\ =& \begin{bmatrix} \cos\alpha w_x - \sin\alpha w_y \\ \cos\alpha w_y + \sin\alpha w_x \\ w_z \end{bmatrix}. \end{aligned} \tag{P.10}$$

We recognize the last expression in (P.10) as a rotation of $\mathbf{w}$ by α radians around $\mathbf{m}$. Thus, Rodrigues' formula holds in the transformed coordinates, and it must also hold in the original coordinates, i.e., for $\mathbf{v}$ and $\mathbf{v}'$.

Bibliography

Books

[3] Y. Bar-Shalom and X. R. Li, *Multitarget-Multisensor Tracking: Principles and Techniques*. Storrs, CT, USA: YBS Publishing, 1995 (cited on pages 14, 65, 110, 112, 114, 124, 127, 140, 160).

[4] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with Application to Tracking and Navigation*. Wiley, 2001 (cited on pages 13, 34, 46, 60, 63, 69, 91, 98, 102).

[6] Y. Bar-Shalom, P. K. Willett, and X. Tian, *Tracking and Data Fusion: A Handbook of Algorithms*. Storrs, CT, USA: YBS Publishing, 2011 (cited on pages 14, 140).

[7] T. D. Barfoot, *State estimation for robotics*. Cambridge University Press, 2018 (cited on page 14).

[10] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006 (cited on pages 46, 102).

[11] S. Blackman and R. Popoli, *Design and Analysis of Modern Tracking Systems*. Norwood, MA, USA: Artech House, 1999 (cited on pages 14, 160).

[21] S. Challa, M. R. Morelande, D. Musicki, and R. J. Evans, *Fundamentals of object tracking*. Cambridge University Press, 2011 (cited on pages 14, 124).

[31] O. Egeland and T. Gravdahl, *Modeling and Simulation for Automatic Control*. Trondheim, Norway: Marine Cybernetics, 2002 (cited on page 166).

[34] G. B. Folland, *Real Analysis: Modern Techniques and Their Applications*. Wiley, 1999 (cited on page 34).

[37] T. I. Fossen, *Handbook of Marine Craft Hydrodynamics and Motion Control*. Wiley, 2011 (cited on pages 169, 170, 181).

[43] A. Gelb, *Applied Optimal Estimation*. MIT Press, 1973 (cited on page 69).

[46] P. D. Groves, *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems*, 2nd edition. Artech House, 2013 (cited on pages 14, 171).

[48] F. Gustafsson, *Statistical Sensor Fusion*. Lund, Sweden: Studentlitteratur, 2010 (cited on pages 13, 14, 46, 91, 102).

[54] A. Iserles, *A first course in the Numerical Analysis of Differential Equations*, 1st edition. Cambridge University Press, 1996 (cited on page 175).

[56] R. A. Johnson and D. W. Wichern, *Applied Multivariate Statistical Analysis*, 5th edition. Prentice Hall, 2002 (cited on page 46).

[59] S. M. Kay, *Fundamentals of Statistical Signal Processing Volume I Estimation Theory*. Prentice-Hall, 1993 (cited on page 14).

[71] R. Mahler, *Statistical Multisource-Multitarget Information Fusion*. Norwood, MA, USA: Artech House, 2007 (cited on pages 14, 46, 115, 199, 202, 206, 214).

[76] K. P. Murphy, *Machine Learning - A Probabilistic Perspective*. Cambridge, MA, USA: MIT Press, 2012 (cited on pages 46, 102).

[85] A. Papoulis and S. U. Pillai, *Probability, Random Variables and Stochastic Processes*. McGraw-Hill, 2002 (cited on pages 14, 27, 34, 55, 69).

[88] B. Ristic, S. Arulampalam, and N. Gordon, *Beyond the Kalman Filter: Particle Filters for Tracking Applications*. Artech House, 2004 (cited on pages 68, 89, 91).

[93] S. Särkkä, *Bayesian Filtering and Smooting*. Cambridge University Press, 2013 (cited on page 68).

[104] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2006 (cited on page 14).

[105] D. Titterton and J. Weston, *Strapdown Inertial Navigation Technology*, 2nd edition, series Radar, Sonar and Navigation. IET, 2004 (cited on pages 14, 181).

[109] B. Vik, *Integrated Satellite and Inertial Navigation Systems*. Department of Engineering Cybernetics, NTNU, 2014 (cited on pages 14, 168).

[112] R. E. Walpole, R. H. Myers, and S. L. Myers, *Probability and Statistics for Engineers and Scientists*. Prentice Hall, 1998 (cited on page 34).

Articles

[1] K. B. Anonsen, "Advances in terrain aided navigation for underwater vehicles", PhD thesis, NTNU, 2010 (cited on page 91).

[2] E. H. Aoki and K. H. Kienitz, "Suboptimal JPDA for tracking in the presence of clutter and missed detections", in *IProceedings of Fusion*, Seattle, WA, USA, Jul. 2009, pages 818–825 (cited on page 141).

[5] Y. Bar-Shalom, S. S. Blackman, and R. J. Fitzgerald, "Dimensionless score function for multiple hypothesis tracking", *IEEE Transactions on Aerospace and Electronic Systems*, volume 43, number 1, pages 392–400, Jan. 2007. DOI: 10.1109/TAES.2007.357141 (cited on pages 159, 160).

[8] Y. Barniv, "Dynamic Programming Algorithm for Detecting Dim Moving Targets", in *Multitarget-Multisensor Tracking: Advanced Applications*, Y. Bar-Shalom, Ed., Artech House, 1990, ch. 4, pages 85–155 (cited on page 91).

[9] N. Bergman, "Recursive bayesian estimation", PhD thesis, Linköping University, 1999 (cited on page 91).

[12] J.-L. Blanco, "A tutorial on se(3) transformation parameterizations and on-manifold optimization", MAPIR: Grupo de Percepción y Robótica, Universidad de Málaga, Tech. Rep., Oct. 2014 (cited on page 168).

[13] H. A. P. Blom and E. A. Bloem, "Probabilistic Data Association Avoiding Track Coalescence", *IEEE Transactions on Automatic Control*, volume 45, number 2, pages 247–259, Feb. 2000, ISSN: 0018-9286. DOI: 10.1109/9.839947 (cited on page 138).

[14] E. Brekke and M. Chitre, "The multiple hypothesis tracker derived from finite set statistics", in *20th International Conference on Information Fusion*, Xi'An, China, Jul. 2017, pages 1–8 (cited on pages 146, 211).

[15] E. Brekke, "Clutter Mitigation for Target Tracking", PhD thesis, Norwegian University of Science and Technology (NTNU), Trondheim, Norway, Jun. 2010 (cited on page 66).

[16] E. Brekke and M. Chitre, "A multi-hypothesis solution to data association for the two-frame SLAM problem", *The International Journal of Robotics Research*, volume 34, number 1, pages 43–63, Jan. 2015. DOI: 10.1177/0278364914545674 (cited on pages 91, 189).

[17] E. Brekke, O. Hallingstad, and J. Glattetre, "The signal-to-noise ratio of human divers", in *Proceedings of OCEANS'10*, Sydney, Australia, May 2010 (cited on pages 62, 66, 215).

[18] ——, "Tracking small targets in heavy-tailed clutter using amplitude information", *IEEE Journal of Oceanic Engineering*, volume 35, number 2, pages 314–329, May 2010 (cited on pages 114, 125, 140).

[19] ——, "The Modified Riccati Equation for Amplitude-Aided Target Tracking in Heavy-Tailed Clutter", *IEEE Transactions on Aerospace and Electronic Systems*, volume 47, number 4, pages 2874–2886, Oct. 2011. DOI: 10.1109/TAES.2011.6034670 (cited on page 140).

[20] ——, "Improved target tracking in the presence of wakes", *IEEE Transactions on Aerospace and Electronic Systems*, volume 48, number 2, pages 1005–1017, Apr. 2012 (cited on pages 125, 140).

[22] D. Clark, B. Ristic, and B.-N. Vo, "PHD filtering with target amplitude feature", in *Information Fusion, 2008 11th International Conference on*, Cologne, Jun. 2008, pages 1–7 (cited on pages 114, 125).

[23] S. Coraluppi and C. Carthel, "Recursive track fusion for multi-sensor surveillance", *Information Fusion*, volume 5, number 1, pages 23–33, 2004, ISSN: 1566-2535 (cited on page 160).

[24] J. L. Crassidis and F. L. Markley, "Attitude estimation using modified rodrigues parameters", in *Proceedings of the Flight Mechanics/Estimation Theory Symposium*, Greenbelt, MD, USA, 1996, pages 71–83 (cited on page 161).

[25] R. Danchick and G. E. Newnam, "Reformulating Reid's MHT method with generalised Murty K-best ranked linear assignment algorithm", *IEE Proceedings - Radar, Sonar and Navigation*, volume 153, pages 13–22, Feb. 2006 (cited on pages 133, 160).

[26] F. Daum, "Solution of the zakai equation by separation of variables", *IEEE Transactions on Automatic Control*, volume 32, number 10, pages 941–943, Oct. 1987, ISSN: 0018-9286 (cited on page 91).

[27] F. Daum and J. Huang, "Nonlinear filters with log-homotopy", in *Proc. Signal and Data Processing of Small Targets (SPIE)*, San Diego, CA, USA, Sep. 2007 (cited on page 91).

[28] S. Davey, M. G. Rutten, and B. Cheung, "A Comparison of Detection Performance for Several Track-before-Detect Algorithms", *EURASIP Journal on Advances in Signal Processing*, volume 2008, 2008. DOI: 10.1155/2008/428036 (cited on page 160).

[29] S. Deb, M. Yeddanapudi, K. Pattipati, and Y. Bar-Shalom, "A generalized S-D assignment algorithm for multisensor-multitarget state estimation", *IEEE Transactions on Aerospace and Electronic Systems*, volume 33, number 2, pages 523–538, Apr. 1997, ISSN: 0018-9251. DOI: 10.1109/7.575891 (cited on pages 156–158).

[30] J. Dezert and Y. Bar-Shalom, "Joint probabilistic data association for autonomous navigation", *Aerospace and Electronic Systems, IEEE Transactions on*, volume 29, number 4, pages 1275–1286, 1993, ISSN: 0018-9251. DOI: 10.1109/7.259531 (cited on page 141).

[33] A. L. Flåten and E. Brekke, "Rao-blackwellized particle filter for turn rate estimation", in *Proceedings of IEEE Aerospace Conference*, Big Sky, MT, USA, Mar. 2017 (cited on pages 91, 114).

[35] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual–inertial odometry", *IEEE Transactions on Robotics*, volume 33, number 1, pages 1–21, Feb. 2017, ISSN: 1552-3098. DOI: 10.1109/TRO.2016.2597321 (cited on page 161).

[36] T. E. Fortmann, Y. Bar-Shalom, and M. Scheffe, "Sonar tracking of multiple targets using joint probabilistic data association", *IEEE Journal of Ocean Engineering*, volume 3, pages 173–184, 1983 (cited on pages 127, 140).

[38] B. Gade, M. Kloster, and M. Aronsen, "Non-elliptical validation gate for maritime target tracking", in *2018 21st International Conference on Information Fusion (FUSION)*, Jul. 2018, pages 1301–1308. DOI: 10.23919/ICIF.2018.8455282 (cited on page 124).

[39] K. Gade, "Integrering av treghetsnavigasjon i en autonom undervannsfarkost", FFI, Tech. Rep. 97/03179, 1997 (cited on pages 170, 180, 181).

[40] ——, "The seven ways to find heading", *The Journal of Navigation*, volume 69, pages 955–970, 2016 (cited on page 181).

[41] ——, "Inertial navigation - theory and applications", PhD thesis, NTNU, 2018 (cited on page 181).

[42] Á. F. García-Fernández, L. Svensson, M. R. Morelande, and S. Särkkä, "Posterior linearization filter: Principles and implementation using sigma points", *IEEE Transactions on Signal Processing*, volume 63, number 20, pages 5561–5573, Oct. 2015, ISSN: 1053-587X. DOI: 10.1109/TSP.2015.2454485 (cited on page 91).

[44] M. Goosen, B. van Wyk, and M. van Wyk, "Survey of jpda algorithms for possible real-time implementation", in *Proceedings of the Fifteenth Annual Symposium of the Pattern Recognition Association of South Africa (PRASA 2004)*, 2004 (cited on page 141).

[45] B. Graf, "Quaternions and dynamics", *ArXiv e-prints*, Nov. 2008. arXiv: 0811.2889 [math.DS] (cited on page 181).

[47] M. Guignard, "Lagrangean relaxation", *Top*, volume 11, number 2, 2003 (cited on page 156).

[49] G. Hendeby, "Performance and implementation aspects of nonlinear filtering", PhD thesis, Linköping University, 2008 (cited on page 91).

[50] S. Hong, M. H. Lee, M. H. Lee, S.-H. Kwon, and J. L. Speyer, "Observability of error states in GPS/INS integration", *IEEE Transactions on Vehicular Technology*, volume 54, number 2, pages 731–742, Mar. 2005 (cited on page 181).

[51] P. Horridge and S. Maskell, "Real-time tracking of hundreds of targets with efficient exact JPDAF implementation", in *9th International Conference on Information Fusion*, Florence, Jul. 2006 (cited on pages 133, 141).

[53] G. P. Huang, A. I. Mourikis, and S. I. Roumeliotis, "Observability-based rules for designing consistent EKF SLAM estimators", *The international Journal of Robotics Research*, volume 29, number 5, pages 502–528, Apr. 2010, ISSN: 02783649. DOI: 10.1177/0278364909353640 (cited on pages 191, 192).

[57] S. J. Julier and J. K. Uhlmann, "Unscented filtering and nonlinear estimation", *Proceedings of the IEEE*, volume 92, number 3, pages 401–422, Mar. 2004, ISSN: 0018-9219. DOI: 10.1109/JPROC.2003.823141 (cited on page 91).

[58] M. Katzfuss, J. R. Stroud, and C. K. Wikle, "Understanding the ensemble kalman filter", *The American Statistician*, volume 70, number 4, pages 350–357, 2016 (cited on page 91).

[60] T. Kirubarajan, Y. Bar-Shalom, W. D. Blair, and G. A. Watson, "IMMPDAF for radar management and tracking benchmark with ECM", volume 34, number 4, pages 1115–1134, Oct. 1998, ISSN: 00189251. DOI: 10.1109/7.722696 (cited on page 125).

[61] A. Kong, J. S. Liu, and W. H. Wong, "Sequential imputations and bayesian missing data problems", *Journal of the American Statistical Association*, volume 89, number 425, 1994 (cited on page 83).

[62] T. Kurien, "Issues in the design of practical multitarget tracking algorithms", in *Multitarget-Multisensor Tracking: Applications and Advances*, Y. Bar-Shalom and W. D. Blair, Eds., Artech House, 1990, ch. 3, pages 219–245 (cited on page 160).

[63] D. Lerro and Y. Bar-Shalom, "Interacting Multiple Model Tracking with Target Amplitude Feature", *IEEE Transactions on Aerospace and Electronic Systems*, volume 29, number 2, pages 494–509, Apr. 1993 (cited on pages 114, 124, 125).

[64] ——, "Tracking with debiased consistent converted measurements versus EKF", *IEEE Transactions on Aerospace and Electronic Systems*, volume 29, number 3, pages 1015–1022, Jul. 1993, ISSN: 0018-9251 (cited on page 65).

[65] N. Li and X. R. Li, "Target Perceivability and its Applications", *IEEE Transactions on Signal Processing*, volume 49, number 11, pages 2588–2604, 2001, ISSN: 1053-587X. DOI: 10.1109/78.960406 (cited on page 124).

[66] X. R. Li, "Tracking in clutter with strongest neighbor measurements. i. theoretical analysis", *IEEE Transactions on Automatic Control*, volume 43, number 11, pages 1560–1578, Nov. 1998 (cited on page 107).

[67] X. R. Li and N. Li, "Integrated Real-Time Estimation of Clutter Density for Tracking", *IEEE Transactions on Signal Processing*, volume 48, number 10, pages 2797–2805, Oct. 2000, ISSN: 1053-587X (cited on page 124).

[69] E. Liland, "AIS aided multi hypothesis tracker", Master's thesis, NTNU, Jun. 2017 (cited on page 155).

[70] M. Longbin, S. Xiaoquan, Z. Yiyu, S. Z. Kang, and Y. Bar-Shalom, "Unbiased Converted Measurements for Tracking", *IEEE Transactions on Aerospace and Electronic Systems*, volume 34, number 3, pages 1023–1027, 1998 (cited on page 65).

[72] M. McDonald and B. Balaji, "Continuous-discrete filtering for dim manoeuvring maritime targets", in *Proceedings of the 10th International Conference on Information Fusion*, Quebec, Canada, Jul. 2007, pages 777–782 (cited on page 91).

[73] M. Montemerlo, "FastSLAM: A factored solution to the simultaneous localization and mapping problem with unknown data association", PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA, Jun. 2003 (cited on page 193).

[74] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: A factored solution to the simultaneous localization and mapping problem", in *Proceedings of AAAI-02*, Edmonton, AL, Canada, 2002, pages 593–598 (cited on page 193).

[75] C. L. Morefield, "Application of 0-1 integer programming to multitarget tracking problems", *IEEE Transactions on Automatic Control*, volume 22, number 3, pages 300–10, Jan. 1977, ISSN: 0018-9286 (cited on page 160).

[77] D. Musicki and R. Evans, "Joint integrated probabilistic data association - jipda", in *Proceedings of Fusion*, volume 2, 2002, 1120–1125 vol.2. DOI: 10.1109/ICIF.2002.1020938 (cited on page 140).

[78] D. Musicki, R. Evans, and S. Stankovic, "Integrated Probabilistic Data Association", *IEEE Transactions on Automatic Control*, volume 39, number 6, pages 1237–1241, 1994 (cited on pages 115, 116, 120, 124).

[79] D. Musicki, S. Suvorova, M. Morelande, and B. Moran, "Clutter Map and Target Tracking", in *Proceedings of Fusion*, Philadelphia, PA, USA, Jul. 2005. DOI: 10.1109/ICIF.2005.1591838 (cited on page 124).

[80] D. Musicki and S. Suvorova, "Tracking in clutter using IMM-IPDA-based algorithms", volume 44, number 1, pages 111–126, Jan. 2008, ISSN: 00189251 (cited on pages 124, 140, 141).

[81] J. Neira and J. D. Tardós, "Data association in stochastic mapping using the joint compatibility test", *IEEE Transactions on Robotics and Automation*, volume 17, number 6, pages 890–897, Dec. 2001, ISSN: 1042296X. DOI: 10.1109/70.976019 (cited on page 189).

[82] P. C. Niedfeldt, K. Ingersoll, and R. W. Beard, "Comparison and analysis of Recursive-RANSAC for multiple target tracking", *IEEE Transactions on Aerospace and Electronic Systems*, volume 53, number 1, pages 461–476, Feb. 2017, ISSN: 0018-9251 (cited on pages 140, 159).

[83] P. C. Niedfeldt, "Recursive-ransac: A novel algorithm for tracking multiple targets in clutte", PhD thesis, Brigham Young University, 2014 (cited on page 159).

[84] J. Olofsson, E. Brekke, T. I. Fossen, and T. A. Johansen, "Spatially indexed clustering for scalable tracking of remotely sensed drift ice", in *Proceedings of IEEE Aerospace Conference*, Big Sky, MT, USA, Mar. 2017 (cited on page 133).

[87] D. Reid, "An algorithm for tracking multiple targets", *IEEE Transactions on Automatic Control*, volume 24, number 6, pages 843–854, Dec. 1979 (cited on pages 143, 146, 160).

[89] A. Rødningsby, "Multitarget multisensor tracking in the presence of wakes", PhD thesis, Norwegian University of Science and Technology, May 2010 (cited on pages 125, 141).

[90] S. Roumeliotis, G. Sukhatme, and G. A. Bekey, "Circumventing dynamic modeling: Evaluation of the error-state Kalman filter applied to mobile robot localization", in *Proceedings of ICRA*, volume 2, 1999, 1656–1663 vol.2 (cited on pages 176, 181).

[91] M. Rutten, J. Williams, N. Gordon, J. Stauch, J. Baldwin, and M. Jah, "A comparison of jpda and belief propagation for data association in ssa", in *Proceedings of Advanced Maui Optical and Space Surveillance Technologies Conference*, 2014 (cited on page 141).

[92] D. J. Salmond, "Tracking in uncertain environments", Royal Aerospace Establishment, UK, Tech. Rep. AW 121, Sep. 1989 (cited on pages 46, 99, 125).

[94] T. Schön, "On computational methods for nonlinear estimation", PhD thesis, Linköping University, 2003 (cited on page 91).

[96] M. Schuster, M. Blaich, and J. Reuter, "Collision avoidance for vessels using a low-cost radar sensor", English, in *Proceedings of IFAC World Congress*, volume 19, Cape Town, South africa, 2014, pages 9673–9678 (cited on page 141).

[97] F. Schweppe, "System identification", in *Uncertain Dynamic Systems*, Prentice Hall, 1979, ch. 14 (cited on page 69).

[98] H. M. Shertukde and Y. Bar-Shalom, "Tracking of crossing targets with imaging sensors", *IEEE Transactions on Aerospace and Electronic Systems*, volume 27, number 4, pages 582–592, Jul. 1991, ISSN: 0018-9251. DOI: 10.1109/7.85031 (cited on page 141).

[99] R. Smith, M. Self, and P. Cheeseman, "Estimating uncertain spatial relationships in robotics", *Autonomous robot vehicles*, pages 167–193, 1990 (cited on page 184).

[101] P. Storms and F. Spieksma, "An LP-based algorithm for the data association problem in multitarget tracking", *Computers & Operations Research*, volume 30, number 7, pages 1067–1085, 2003, ISSN: 0305-0548. DOI: https://doi.org/10.1016/S0305-0548(02)00057-6 (cited on pages 156, 160).

[102] E. F. Wilthil, A. L. Flaten, and E. F. Brekke, "A target tracking system for asv collision avoidance based on the pdaf", English, in *Sensing and Control for Autonomous Vehicles*, P. Fossen and Nijmeijer, Eds., volume 474, Alesund, Norway: Springer, 2017, pages 269–288 (cited on pages 63, 64, 66).

[103] T. Tengesdal, "Uncertainty management in a scenario-based mpc for collision avoidance", Master's thesis, NTNU, Jun. 2019 (cited on page 80).

[106] L.-C. N. Tokle, "Multi target tracking using random finite sets with a hybrid state space and approximations", Master's thesis, NTNU, Sep. 2018 (cited on pages 46, 124, 140).

[107] N. Trawny and S. I. Roumeliotis, "Indirect kalman filter for 3d attitude estimation - a tutorial for quaternion algebra", MARS-LAB, Tech. Rep., 2005 (cited on page 181).

[108] C. Van Loan, "Computing integrals involving the matrix exponential", *IEEE Transactions on Automatic Control*, volume 23, number 3, pages 395–404, 1978 (cited on page 61).

[110] B.-N. Vo, S. Singh, and A. Doucet, "Sequential Monte Carlo methods for multitarget filtering with random finite sets", *IEEE Transactions on Aerospace and Electronic Systems*, volume 41, number 4, pages 1224–1245, Oct. 2005, ISSN: 0018-9251. DOI: 10.1109/TAES.2005.1561884 (cited on page 205).

[111] B.-T. Vo, B.-N. Vo, and A. Cantoni, "Bayesian Filtering With Random Finite Set Observations", *IEEE Transactions on Signal Processing*, volume 56, number 4, pages 1313–1326, Apr. 2008. DOI: 10.1109/TSP.2007.908968 (cited on page 125).

[113] P. Willett, Y. Ruan, and R. Streit, "PMHT: Problems and Some Solutions", *IEEE Transactions on Aerospace and Electronic Systems*, volume 38, number 3, pages 738–754, Jul. 2002, ISSN: 0018-9251. DOI: 10.1109/TAES.2002.1039396 (cited on pages 128, 160).

[114] J. Williams, "An efficient, variational approximation of the best fitting multi-Bernoulli filter", *IEEE Transactions on Signal Processing*, volume 63, number 1, pages 258–273, Jan. 2015, ISSN: 1053-587X. DOI: 10.1109/TSP.2014.2370946 (cited on page 139).

[115] ——, "Marginal multi-Bernoulli filters: RFS derivation of MHT, JIPDA, and association-based MeMBer", *IEEE Transactions on Aerospace and Electronic Systems*, volume 51, number 3, pages 1664–1687, Jul. 2015 (cited on pages 124, 210).

[116] J. Williams, "Gaussian Mixture Reduction for Tracking Multiple Maneuvering Targets in Clutter", Master's thesis, Air Force Institute of Technology, Air University, Mar. 2003 (cited on pages 99, 125).

[117] J. Williams and R. Lau, "Approximate evaluation of marginal association probabilities with belief propagation", *IEEE Transactions on Aerospace and Electronic Systems*, volume 50, number 4, pages 2942–2959, Oct. 2014 (cited on pages 133, 141).

[118] E. F. Wilthil, Y. Bar-Shalom, P. Willett, and E. Brekke, "Estimation of target detectability for maritime target tracking in the pda framework", in *Proceedings of FUSION*, Ottawa, Canada, Jul. 2019 (cited on page 124).

[119] E. F. Wilthil, E. Brekke, and O. B. Asplin, "Track initiation for maritime radar tracking with and without prior information", in *Proc. Fusion*, Cambridge, UK, Jul. 2018 (cited on pages 120, 124).

[120] Y. Xia, K. Granström, L. Svensson, and Á. F. García-Fernández, "An implementation of the Poisson Multi-Bernoulli mixture trajectory filter via dual decomposition", in *2018 21st International Conference on Information Fusion (FUSION)*, Jul. 2018, pages 1–8. DOI: 10.23919/ICIF.2018.8455236 (cited on page 156).

Sources and resources from the web

[32] M. A. T. Figueiredo, "Lecture notes on Bayesian estimation and classification", Instituto de Telecomunicacoes, Portugal., Oct. 2004 (cited on page 34).

[52] G. B. Huang, "Conditional and marginal distributions of a multivariate Gaussian", Accessed 17th of June 2017, Feb. 2010, [Online]. Available: https://gbhqed.wordpress.com/2010/02/21/conditional-and-marginal-distributions-of-a-multivariate-gaussian (cited on pages 46, 194).

[55] T. A. Johansen and T. I. Fossen, "The eXogenous Kalman filter (XKF)", Submitted to International Journal of Control, 2015 (cited on page 192).

[68] E. Liland, "An ILP approach to multi hypothesis tracking", Specialization project at NTNU, Dec. 2016 (cited on page 156).

[86] J. Pedersen, "Surveillance of the channel", Specialization Project at NTNU, Dec. 2017 (cited on page 139).

[95] T. B. Schön and F. Lindsten, "Manipulating the Multivariate Gaussian Density", Accessed 17th of June 2017, Jan. 2011, [Online]. Available: user.it.uu.se/~thosc112/pubpdf/schonl2011.pdf (cited on page 46).

[100] J. Solà, "Quaternion kinematics for the error-state KF", Mar. 2015, [Online]. Available: http://www.iri.upc.edu/people/jsola/JoanSola/objectes/notes/kinematics.pdf (cited on pages 14, 163, 166, 169, 171, 172, 174, 176, 180, 181).