FACULTY OF INFORMATION TECHNOLOGY CTU IN PRAGUE

Assignment of bachelor's thesis

Title:	Kubaturní Kalmanův filtr v inženýrské praxi
Student:	Petr Fiedler
Supervisor:	doc. Ing. Kamil Dedecius, Ph.D.
Study program:	Informatics
Branch / specialization:	Artificial Intelligence 2021
Department:	Department of Applied Mathematics
Validity:	until the end of summer semester 2024/2025

Instructions

Abstract: The class of state-space models abounds in many real-time domains of the engineering practice. If the models are linear, the celebrated Kalman filter is mainly used as an optimal linear sequential estimator. However, if the assumption of linearity is not met, an alternative approach is required. For instance, the model is linearized using Taylor-type approximations. This gives rise to the extended Kalman filters. Another option is to proceed with the original model and perform approximate inference, e.g., using unscented transform or numerical methods. The thesis focuses on nonlinear filtering and the cubature Kalman filter in particular. This filter exploits the quadrature rule of integration. The goal is to provide a clear exposition of the filter derivation and to demonstrate its properties, such as its connection to the unscented Kalman filter. The work should contain some interesting simulations or real data examples.

Goals:
1) Describe the problematics of the linear and nonlinear state-space models.
2) Outline the Kalman filter, describe its properties and weaknesses.
3) Carefully derive the cubature Kalman filter. Give all details in a comprehensible exposition. Discuss its properties and connections with other filters.
4) Provide simulation and/or real-world examples demonstrating the properties of the filter.

References:
[1] I. Arasaratnam and S. Haykin, "Cubature Kalman Filters," IEEE Trans. Automat. Contr., vol.

Electronically approved by Ing. Magda Friedjungová, Ph.D. on 18 January 2024 in Prague.

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

54, no. 6, pp. 1254–1269, Jun. 2009, doi: 10.1109/TAC.2009.2019800.
[2] D. Simon, Optimal state estimation: Kalman, H-infinity, and nonlinear approaches. Wiley-Interscience, 2006.
[3] A. H. Stroud, Approximate calculation of multiple integrals. Prentice-Hall series in automatic computation. Englewood Cliffs, NJ: Prentice-Hall, 1971.

Bachelor's thesis

CUBATURE KALMAN FILTERING IN ENGINEERING PRACTICE

Petr Fiedler

Faculty of Information Technology
Department of Applied Mathematics
Supervisor: doc. Ing. Kamil Dedecius, Ph.D.
April 30, 2024

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis: Fiedler Petr. *Cubature Kalman filtering in engineering practice.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2024.

Contents

List of Figures

List of Tables

I would like to express my deepest gratitude to my supervisor, doc. Ing. Kamil Dedecius, Ph.D., for his unwavering support, guidance, and insightful feedback throughout the course of my bachelor's thesis. Additionally, I would like to thank my family for their endless understanding and encouragement.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended. In accordance with Section 2373(2) of Act No. 89/2012 Coll., the Civil Code, as amended, I hereby grant a non-exclusive authorization (licence) to utilize this thesis, including all computer programs that are part of it or attached to it and all documentation thereof (hereinafter collectively referred to as the "Work"), to any and all persons who wish to use the Work. Such persons are entitled to use the Work in any manner that does not diminish the value of the Work and for any purpose (including use for profit). This authorisation is unlimited in time, territory and quantity.

In Prague on April 30, 2024

Abstract

This thesis explores the cubature Kalman filter, an algorithm designed for estimating system states in real-time within nonlinear models. While the filter possesses solid theoretical foundations, the original exposition of this algorithm may present accessibility challenges due to its reliance on advanced mathematical concepts. This thesis aims to present its derivation comprehensively, making it more accessible while explaining its core mathematical principles. Additionally, the cubature Kalman filter is compared with other algorithms for nonlinear filtering to highlight its unique features. Through practical examples, this thesis demonstrates the filter's effectiveness and operation in various scenarios.

Keywords Kalman filter, state estimation, nonlinear filtering, spherical-radial coordinates, numerical integration, cubature rules, Gaussian quadrature

Abstrakt

Tato práce zkoumá kubaturní Kalmanův filtr, algoritmus určený pro odhad stavu systému v reálném čase v rámci nelineárních modelů. Tento filtr má silné teoretické základy, avšak originální představení tohoto algoritmu může být hůře přístupné kvůli jeho závislosti na pokročilých matematických konceptech. Práce si klade za cíl pečlivě představit jeho odvození, tak, aby bylo přístupnější. Součástí toho jsou v této práci vysvětleny matematické principy klíčové pro odvození tohoto algoritmu. Kubaturní Kalmanův filtr je dále porovnán s jinými nelineárními filtračními algoritmy, aby byly zdůrazněny jeho jedinečné vlastnosti. Práce také demonstruje účinnost a fungování filtru na praktických příkladech.

Klíčová slova Kalmanův filtr, stavový odhad, nelineární filtrování, sféricko-radiální souřadnice, numerická integrace, kubaturní pravidla, Gaussova kvadratura

Abbreviations

CAM	constant acceleration model
CKF	cubature Kalman filter
CTM	coordinated turn model
CVM	constant velocity model
EKF	extended Kalman filter
KF	Kalman filter
MSE	mean squared error
UAV	unmanned aerial vehicle
UKF	unscented Kalman filter

Introduction

A frequent challenge in engineering practice is to estimate the true state of a system on the basis of limited and noisy data about the system. For instance, we may want to estimate the precise location of a vehicle using GPS measurements that are not entirely accurate. Another example would be trying to get an accurate temperature using a thermostat, which sometimes gives inaccurate readings.

A typical approach involves constructing a state-space model for the system and attempting to estimate the current state of the system based on the measurement history. The problem, however, is that in practice the model cannot accurately describe the dynamics of the system. We are always dealing with noise, either in the state itself or in the measurements. The solution to this problem is the use of a filter such as the Kalman filter (KF).

The limitation of the Kalman filter is that it is only applicable to linear models. Nonlinear models require alternative approaches, often still based on the Kalman filter. One of the methods for nonlinear state estimation is the cubature Kalman filter. In comparison to other nonlinear state estimation methods, this filter remains relatively underexplored.

In this thesis, we aim to:

1. Introduce the state-space model, both linear and nonlinear.
2. Derive the Kalman filter, an algorithm for linear state estimation, and describe its properties and weaknesses.
3. Derive the cubature Kalman filter, an algorithm for nonlinear state estimation, along with introducing the mathematical principles necessary for its derivation.
4. Describe the properties of the cubature Kalman filter and compare it to other nonlinear filters.
5. Demonstrate the cubature Kalman filter on simulations and/or real-world examples suitable for expressing its properties.

Chapter 1

State-space model

A vast number of processes in the real world form a system that evolves over time. At each point in time, the system has a state. If a model for such a system were to be created, various mathematical tools could be applied to analyze the system. This model is called a state-space model. By knowing the current state of the system, we can describe the future behavior of the system based on future inputs using the state-space model.

Although processes in nature usually evolve in continuous time, we restrict ourselves to discrete-time state-space models in this thesis. This allows us, among other things, to simulate the processes on a computer. Moreover, the discrete-time description removes the need for tediously solved differential equations. Thus, the state-space model will depict the system's state at time k, considering its state at time $k-1$ and the input parameters.

1.1 Linear state-space model

Even though most processes in the real world are nonlinear, linear systems are more convenient to handle. There are a variety of mathematical tools that are readily available and well-understood for working with linear systems. Therefore, we often approximate nonlinear systems by linear systems [1].

Let us represent the state of the system at time k by the vector x_k. Thus, we can form the linear system equation [1] describing the transition from state x_{k-1} to state x_k as

$$x_k = F_{k-1}x_{k-1} + G_{k-1}u_{k-1} + w_{k-1}, \tag{1.1}$$

where u_{k-1} is the control input at time $k-1$ in the form of a vector. The F_{k-1} matrix, often called the system matrix, and the G_{k-1} matrix, called the input matrix, both of appropriate dimensions, represent the transition based on the previous state and the control input. The vector w_{k-1} represents the process

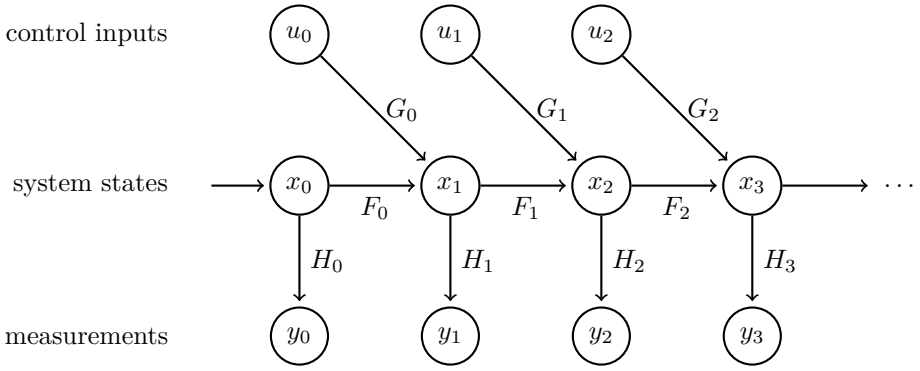

Figure 1.1 Linear state-space model representation of the system dynamics.

noise, which is assumed to be zero-mean and uncorrelated, with a covariance matrix Q_{k-1}.

The system equation 1.1 is a hidden part of the state-space model, therefore the true value of the vector x_k is never known. It can only be estimated using observations in the form of measurements. The values obtained from measuring the system reflect its current state. Let us represent the measured values at time k by the output vector y_k. Thus, we can formulate the linear measurement equation [1]

$$y_k = H_k x_k + v_k, \tag{1.2}$$

where H_k, often called the output matrix, is a matrix of appropriate dimensions and the vector v_k represents the measurement noise, also assumed to be zero-mean and uncorrelated, with a covariance matrix R_k. Unlike the vector x_k, the vector y_k is known to us because it is the result of a measurement.

The linear system equation, along with the linear measurement equation, constitutes a linear state-space model.

In Figure 1.1, we can observe the evolution of the state vector over time and its influence on the measurements. However, it is important to note that the depicted relations are not exact due to the presence of noise.

▶ **Example 1.1** (Constant acceleration model)**.** Consider an object moving in one-dimensional space maintaining a nearly constant acceleration, except for changes due to noise. The motion of this object can be described by the equations of the Newtonian system. The state of the object at time k will include its position p_k, velocity r_k, and acceleration a_k. As mentioned earlier, we operate within discrete time, therefore, we obtain equations

$$p_k = p_{k-1} + \Delta_t r_{k-1} + \frac{1}{2}\Delta_t^2 a_{k-1} + w_{p,k-1}, \tag{1.3}$$

$$r_k = r_{k-1} + \Delta_t a_{k-1} + w_{r,k-1}, \tag{1.4}$$

$$a_k = a_{k-1} + w_{a,k-1}, \tag{1.5}$$

where Δ_t is the constant time difference between time $k-1$ and k. Then we can write the system equation in matrix form

$$x_k = F_{k-1}x_{k-1} + w_{k-1}, \tag{1.6}$$

for

$$x_k = \begin{pmatrix} p_k \\ r_k \\ a_k \end{pmatrix}, \tag{1.7}$$

$$F_k = \begin{pmatrix} 1 & \Delta_t & \frac{1}{2}\Delta_t^2 \\ 0 & 1 & \Delta_t \\ 0 & 0 & 1 \end{pmatrix}, \tag{1.8}$$

$$w_k = \begin{pmatrix} w_{p,k} \\ w_{r,k} \\ w_{a,k} \end{pmatrix}. \tag{1.9}$$

For completeness, the vector u_k is always a zero vector as there is no control input. Therefore, the matrix G_k is no longer relevant.

Assume the presence of a measuring device that determines the position of the object. The measurement equation will have the form

$$y_k = H_k x_k + v_k, \tag{1.10}$$

where y_k is a one-dimensional vector containing the measurement, v_k is also a one-dimensional vector of measurement noise, and H_k is the output matrix defined as

$$H_k = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}. \tag{1.11}$$

The elements in H_k indicate that the measuring device extracts the position information along the first dimension from the state vector x_k. ◀

1.2 Nonlinear state-space model

In practice, we usually encounter a nonlinear system. The nonlinear state-space model contains a system and a measurement equation as well. However, it does not place restrictions on the functions used in these equations. Thus we receive the nonlinear state-space model [1] constituting equations

$$x_k = f(x_{k-1}, u_{k-1}, w_{k-1}), \tag{1.12}$$

$$y_k = h(x_k, v_k), \tag{1.13}$$

where x_k is the state vector at time k, u_{k-1} is the known control input, y_k is the output vector, w_{k-1} and v_k are zero-mean, uncorrelated noises[1] with covariance matrices Q_k and R_k, respectively. The functions f and h are arbitrary vector-valued functions. A linear state-space system is, therefore, a special case of a nonlinear one.

[1] A variant of the state-space model with correlated noises exists as well [1].

▶ **Example 1.2** (Coordinated turn model)**.** Consider an object moving in two-dimensional space with a nearly constant turn rate, affected only by noise. The state of the object at time k will consist of the two-dimensional coordinates $x_{1,k}$ and $x_{2,k}$, the velocities in both dimensions $r_{1,k}$ and $r_{2,k}$ and the turn rate ω_k. According to [2], we get the discretized state-space model system equation

$$\begin{aligned} x_k &= f(x_{k-1}, u_{k-1}, w_{k-1}) \\ &= \begin{pmatrix} 1 & 0 & \frac{\sin(\Delta_t \omega_{k-1})}{\omega_{k-1}} & \frac{-1+\cos(\Delta_t \omega_{k-1})}{\omega_{k-1}} & 0 \\ 0 & 1 & \frac{1-\cos(\Delta_t \omega_{k-1})}{\omega_{k-1}} & \frac{\sin(\Delta_t \omega_{k-1})}{\omega_{k-1}} & 0 \\ 0 & 0 & \cos(\Delta_t \omega_{k-1}) & -\sin(\Delta_t \omega_{k-1}) & 0 \\ 0 & 0 & \sin(\Delta_t \omega_{k-1}) & \cos(\Delta_t \omega_{k-1}) & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} x_{k-1} + w_{k-1}, \end{aligned} \tag{1.14}$$

where Δ_t is the constant discretisation time interval between time $k-1$ and k, w_{k-1} is the process noise vector, and

$$x_k = \begin{pmatrix} x_{1,k} \\ x_{2,k} \\ r_{1,k} \\ r_{2,k} \\ \omega_k \end{pmatrix}. \tag{1.15}$$

At first glance it may appear that f is still just a linear function, but the matrix we are multiplying the state vector x_{k-1} by contains ω_{k-1}, which is also included in the state vector x_{k-1}, causing nonlinearity.

Let only the coordinates of the object position be measured. The resulting measurement equation will be linear in this case. We get

$$y_k = h(x_k, v_k) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix} x_k + v_k. \tag{1.16}$$

Thus, the vector y_k will contain the measured coordinates at time k affected by the measurement noise v_k. ◀

Chapter 2

Kalman filter

Given a linear state-space model, we can use the Kalman filter to estimate the internal state parameters of the system. The linearity of the state-space model is an important requirement because the Kalman filter relies on linear equations to predict the state of the system and update it based on measurements.

2.1 Overview of the Kalman filter

The Kalman filter works iteratively, with each iteration corresponding to a discretization time interval. In each iteration, it produces an estimate of the current state vector, denoted by $\hat{x}_k$, influenced by a measurement y_k.

The covariance matrix of x_k, denoted by P_k, plays a pivotal role in the Kalman filtering process. It holds that

$$P_k = \mathbb{E}\left[(x_k - \mathbb{E}\left[x_k\right])\left(x_k - \mathbb{E}\left[x_k\right]\right)^\intercal\right]. \tag{2.1}$$

The covariance P_k indicates the level of uncertainty regarding the estimate.

Each iteration of the Kalman filter algorithm comprises two phases: the time update and the measurement update. In the time update, it is assumed that a discretization time interval has passed. The system equation 1.1 of the state-space model is used to form an estimate of the state vector x_k based on the estimate of x_{k-1}. The estimate formed during the time update is referred to as the *a priori* estimate and is denoted by $\hat{x}_k^-$. The P_k covariance estimate at this phase is known as the *a priori* covariance, denoted as P_k^-.

In the measurement update, we aim to improve the *a priori* estimate by utilizing the information about the measurement y_k and the measurement equation 1.2 of the state-space model, which lets us form an estimate of y_k, denoted as $\hat{y}_k$. Thus we get a new estimate of x_k, which is referred to as the *a posteriori* estimate and is denoted by $\hat{x}_k^+$. The P_k covariance estimate at this phase is known as the *a posteriori* covariance, denoted as P_k^+.

Figure 2.1 illustrates the evolution of estimates throughout the Kalman filter algorithm cycle.

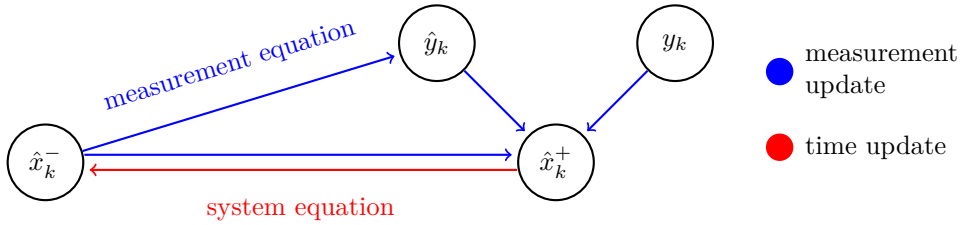

Figure 2.1 The state estimate evolution cycle in the Kalman filter and its use of the state-space model.

2.2 Derivation of the Kalman filter

This derivation of the Kalman filter draws heavily from [1]. We will present the derivation in two distinct parts: the time update and the measurement update.

2.2.1 The time update

In the time update, we assume that we have the *a posteriori* state estimate $\hat{x}_{k-1}^+$ and its covariance P_{k-1}^+ from the previous iteration.

Let us define D_k as the history of all control inputs and measurements up to time k. Therefore

$$D_k = \{u_i, y_i\}_{i=0}^k. \tag{2.2}$$

Another way of thinking about the *a priori* estimate $\hat{x}_k^-$ is as the expected value of x_k conditional on D_{k-1}. Therefore

$$\hat{x}_k^- = \mathbb{E}\left[x_k \mid D_{k-1}\right]. \tag{2.3}$$

Similarly for the *a posteriori* estimate, we get

$$\hat{x}_k^+ = \mathbb{E}\left[x_k \mid D_k\right]. \tag{2.4}$$

From the system equation 1.1 we get

$$\hat{x}_k^- = \mathbb{E}\left[F_{k-1}x_{k-1} + G_{k-1}u_{k-1} + w_{k-1} \mid D_{k-1}\right]. \tag{2.5}$$

Since the noise w_{k-1} is assumed to be zero-mean, we have

$$\hat{x}_k^- = F_{k-1}\mathbb{E}\left[x_{k-1} \mid D_{k-1}\right] + G_{k-1}u_{k-1}. \tag{2.6}$$

And finally, according to Equation 2.4, we get the desired equation

$$\hat{x}_k^- = F_{k-1}\hat{x}_{k-1}^+ + G_{k-1}u_{k-1}. \tag{2.7}$$

We now aim to derive an equation for P_k^-, the *a priori* covariance. By leveraging Equation 2.1 and utilizing the *a priori* state estimate $\hat{x}_k^-$, we can estimate P_k as

$$P_k^- = \mathbb{E}\left[(x_k - \hat{x}_k^-)(x_k - \hat{x}_k^-)^\intercal\right]. \tag{2.8}$$

First, let us examine the expression within the expectation operator. Thus

$$\begin{aligned}
(x_k - \hat{x}_k^-)(x_k - \hat{x}_k^-)^\intercal &= (F_{k-1}x_{k-1} + G_{k-1}u_{k-1} + w_{k-1} - \hat{x}_k^-)(\dots)^\intercal && (2.9)\\
&= (F_{k-1}(x_{k-1} - \hat{x}_{k-1}^+) + w_{k-1})(\dots)^\intercal && (2.10)\\
&= F_{k-1}(x_{k-1} - \hat{x}_{k-1}^+)(x_{k-1} - \hat{x}_{k-1}^+)^\intercal F_{k-1}^\intercal + w_{k-1}w_{k-1}^\intercal \\
&\quad + F_{k-1}(x_{k-1} - \hat{x}_{k-1}^+)w_{k-1}^\intercal \\
&\quad + w_{k-1}(x_{k-1} - \hat{x}_{k-1}^+)^\intercal F_{k-1}^\intercal. && (2.11)
\end{aligned}$$

Since $(x_{k-1} - \hat{x}_{k-1}^+)$ is uncorrelated with w_{k-1}, using Equation 2.8 we obtain

$$\begin{aligned}
P_k^- &= F_{k-1}\mathbb{E}\left[(x_{k-1} - \hat{x}_{k-1}^+)(x_{k-1} - \hat{x}_{k-1}^+)^\intercal\right] F_{k-1}^\intercal + \mathbb{E}\left[w_{k-1}w_{k-1}^\intercal\right] && (2.12)\\
&= F_{k-1}P_{k-1}^+F_{k-1}^\intercal + Q_{k-1}, && (2.13)
\end{aligned}$$

where Q_{k-1} represents the covariance matrix of the zero-mean process noise, as previously defined.

2.2.2 The measurement update

During the measurement update, the aim is to refine the estimate of the state vector using the measurement y_k. The update equation should be linear for mathematical tractability. To achieve this, we need to find a matrix K_k and a vector b_k that satisfy

$$\hat{x}_k^+ = K_k y_k + b_k. \tag{2.14}$$

The matrix K_k and the vector b_k should be determined based on the *a priori* state estimate $\hat{x}_k^-$ and the *a priori* covariance P_k^- obtained in the previous phase.

Our requirement for the *a posteriori* estimate $\hat{x}_k^+$ is unbiasedness, meaning $\mathbb{E}\left[x_k - \hat{x}_k^+\right] = 0$. Taking the mean of Equation 2.14, we get

$$\overline{\hat{x}_k^+} = K_k\overline{y_k} + b_k. \tag{2.15}$$

Unbiasedness of the *a posteriori* estimate implies that $\overline{\hat{x}_k^+} = \overline{x_k}$, resulting in the constraint

$$b_k = \overline{x_k} - K_k\overline{y_k}. \tag{2.16}$$

By substituting $\overline{x_k}$ with the *a priori* estimate and $\overline{y_k}$ with the measurement vector estimate $\hat{y}_k$ given by $H_k\hat{x}_k^-$, recalling the measurement equation 1.2, we get

$$b_k = \hat{x}_k^- - K_k\hat{y}_k, \tag{2.17}$$

therefore

$$\begin{aligned}\hat{x}_k^+ &= K_k y_k + \hat{x}_k^- - K_k H_k \hat{x}_k^- \\ &= \hat{x}_k^- + K_k(y_k - H_k \hat{x}_k^-).\end{aligned} \tag{2.18}$$

Now the objective is to find a matrix K_k that minimizes the optimality criterion J_k, defined as the trace of the *a posteriori* covariance. Therefore

$$J_k = \mathrm{Tr}(P_k^+). \tag{2.19}$$

The criterion is defined as it is because the covariance matrix P_k^+ represents the uncertainty associated with the state estimate $\hat{x}_k^+$. Specifically, the trace of P_k^+ captures this uncertainty by summing the variances of the components of the state vector along its diagonal. Minimizing this criterion ensures that the estimate $\hat{x}_k^+$ is as accurate as possible while accounting for the inherent uncertainty in the estimation process.

For an arbitrary random vector z, it holds that

$$P_z = \mathbb{E}\left[(z - \overline{z})(z - \overline{z})^\intercal\right]. \tag{2.20}$$

Let us set $z = x_k - \hat{x}_k^+$. The property of unbiasedness ensures that $\overline{z} = 0$. This gives us

$$P_z = \mathbb{E}\left[zz^\intercal\right] = \mathbb{E}\left[(x_k - \hat{x}_k^+)(x_k - \hat{x}_k^+)^T\right] = P_k^+. \tag{2.21}$$

Using Equation 2.20 and our definition of z, we can compute P_z as

$$P_k^+ = \mathbb{E}\left[(x_k - \hat{x}_k^+ - \mathbb{E}\left[x_k - \hat{x}_k^+\right])(\dots)^\intercal\right] \tag{2.22}$$

$$= \mathbb{E}\left[(x_k - (K_k y_k + b_k) - \overline{x_k} - (K_k \overline{y_k} + b_k))(\dots)^\intercal\right] \tag{2.23}$$

$$= \mathbb{E}\left[((x_k - \overline{x_k}) - K_k(y_k - \overline{y_k}))(\dots)^\intercal\right] \tag{2.24}$$

$$= P_k^- - K_k P_{xy,k}^\intercal - P_{xy,k} K_k^\intercal + K_k P_{y,k} K_k^\intercal, \tag{2.25}$$

where $P_{xy,k}$ denotes the cross-covariance between x_k and y_k and $P_{y,k}$ denotes the covariance of y_k. The matrix $P_{xy,k}$ will be referred to as the cross-covariance matrix. The matrix $P_{y,k}$ will be referred to as the innovation covariance matrix. Straightforward calculations [1] show that we can obtain the cross-covariance and the innovation covariance matrices as

$$P_{xy,k} = P_k^- H_k^\intercal, \tag{2.26}$$

$$P_{y,k} = H_k P_k^- H_k^\intercal + R_k, \tag{2.27}$$

where R_k represents the covariance matrix of the measurement noise, as previously defined. We have also used P_k^- instead of P_k for the covariance of x_k as it represents our best estimate of P_k.

Given the fact that

$$\begin{aligned}&K_k P_{y,k} K_k^\intercal - K_k P_{xy,k}^\intercal - P_{xy,k} K_k^\intercal \\ &= (K_k - P_{xy,k} P_{y,k}^{-1}) P_{y,k} (K_k - P_{xy,k} P_{y,k}^{-1})^\intercal - P_{xy,k} P_{y,k}^{-1} P_{xy,k}^\intercal,\end{aligned} \tag{2.28}$$

and recalling Equation 2.25, we can express the optimality criterion J_k as

$$\begin{aligned} J_k &= \mathrm{Tr}(P_k^+) && (2.29)\\ &= \mathrm{Tr}((K_k - P_{xy,k}P_{y,k}^{-1})P_{y,k}(K_k - P_{xy,k}P_{y,k}^{-1})^\intercal) && (2.30)\\ &\quad + \mathrm{Tr}(P_k^- - P_{xy,k}P_{y,k}^{-1}P_{xy,k}^\intercal). \end{aligned}$$

The first term in Equation 2.30 is always nonnegative. Therefore, to minimize the first term to zero, we set

$$K_k = P_{xy,k}P_{y,k}^{-1}. \tag{2.31}$$

Since the second term in Equation 2.30 is independent of K_k, the optimality criterion J_k is minimized. The matrix K_k is commonly referred to as the Kalman gain.

Finally, combining Equations 2.25, 2.28, and 2.31, we obtain

$$\begin{aligned} P_k^+ &= P_k^- - P_{xy,k}P_{y,k}^{-1}P_{xy,k}^\intercal && (2.32)\\ &= P_k^- - K_k P_{y,k} K_k^T. \end{aligned}$$

2.3 The Kalman filter algorithm

To ensure clarity, this section provides a summary of the Kalman filter algorithm derived in the previous section.

2.3.1 Initialization

1. Initialize the *a posteriori* state estimate with the most accurate estimation of the initial state, ideally
$$\hat{x}_0^+ = \mathbb{E}\left[x_0\right]. \tag{2.33}$$

2. Initialize the *a posteriori* covariance using a covariance matrix that reflects the level of confidence in the initial estimate. If we are sure that our initial estimate is ideal, we set
$$P_0^+ = \mathbb{E}\left[(x_0 - \hat{x}_0^+)(x_0 - \hat{x}_0^+)^\intercal\right]. \tag{2.34}$$

2.3.2 The time update

1. Evaluate the *a priori* state estimate
$$\hat{x}_k^- = F_{k-1}\hat{x}_{k-1}^+ + G_{k-1}u_{k-1}. \tag{2.35}$$

2. Evaluate the *a priori* covariance
$$P_k^- = F_{k-1}P_{k-1}^+F_{k-1}^\intercal + Q_{k-1}. \tag{2.36}$$

2.3.3 The measurement update

1. Evaluate the cross-covariance matrix

$$P_{xy,k} = P_k^- H_k^\intercal. \tag{2.37}$$

2. Evaluate the innovation covariance matrix

$$P_{y,k} = H_k P_k^- H_k^\intercal + R_k. \tag{2.38}$$

3. Evaluate the Kalman gain

$$K_k = P_{xy,k} P_{y,k}^{-1}. \tag{2.39}$$

4. Evaluate the measurement estimate

$$\hat{y}_k = H_k \hat{x}_k^-. \tag{2.40}$$

5. Evaluate the *a posteriori* state estimate

$$\hat{x}_k^+ = \hat{x}_k^- + K_k(y_k - \hat{y}_k). \tag{2.41}$$

6. Evaluate the *a posteriori* covariance

$$P_k^+ = P_k^- - K_k P_{y,k} K_k^\intercal. \tag{2.42}$$

2.4 Interpretation of the resulting equations

The derived Kalman filter equations have the advantageous characteristic of being relatively intuitive in their interpretation. This allows us to have a deeper comprehension of the functioning principles of the Kalman filter algorithm.

The measurement update aims to update the state estimate by the value of a measurement. This can be seen in Equation 2.41, where we shift our previous state estimate by $(y_k - \hat{y}_k)$ multiplied by the Kalman gain. The term $(y_k - \hat{y}_k)$ represents the difference between our prediction of y_k and the actual y_k, i.e., the prediction error. We use this prediction error to correct our state estimate, hence, it represents the innovation.

The higher the prediction error, the more we want our state estimate values to change. However, if we are uncertain about the measurements and their predictions, we do not want to emphasize the data as much. We rather want to emphasize the model that forms our state estimate in the time update. This is achieved by the Kalman gain. Observing Equation 2.39, we can see that the Kalman gain takes lower values as the measurement data are less reliable, which is represented by higher values of the innovation covariance $P_{y,k}$.

Conversely, if we are uncertain about the state estimate, the values of $P_{xy,k}$ are higher. This results in higher Kalman gain, thus, placing more emphasis

on the measurements. The Kalman gain then essentially represents the state estimate uncertainty relative to the total uncertainty of the prediction.

We can interpret the equation for the *a posteriori* covariance as well. To make the interpretation clearer, we can express it by

$$P_k^+ = (I - K_k H_k) P_k^-, \tag{2.43}$$

which is equivalent to Equation 2.42. The equivalence can be shown easily using the symmetry of the covariance matrices. It is evident from this form that we obtain P_k^+ from P_k^- by multiplying it by $(I - K_k H_k)$. Since the Kalman gain K_k represents the state estimate uncertainty relative to the uncertainty of the prediction, the state estimate uncertainty decreases the more we emphasize the measurement data in this time step.

2.5 Properties of the Kalman filter

The Kalman filter is a linear filter, that is, it calculates a state estimate based on a measurement using a linear equation[1], recall Equation 2.14. Its use of linear equations enables efficient estimation, especially for systems with Gaussian noise. The Kalman filter is also unbiased, meaning that on average, its estimates are equal to the true values of the quantities being estimated. This property is crucial for ensuring the accuracy and reliability of the filter's predictions over time. We have derived the Kalman filter in a way that preserves these properties.

To state the following property, let us define the mean squared error.

▶ **Definiton 2.1** (Mean squared error)**.** *Let x be a random vector and $\hat{x}$ its estimate. Then the mean squared error (MSE) is defined as*

$$\mathrm{MSE}(\hat{x}, x) = \mathbb{E}\left[\|\hat{x} - x\|^2\right]. \tag{2.44}$$

◀

The MSE allows us to measure the accuracy of an estimate. The higher the value, the worse the estimate. When we use the MSE to evaluate a Kalman filter process, for each time step k we get the value $\mathrm{MSE}(\hat{x}_k^+, x_k)$. In practice, we can estimate the MSE by averaging $\|\hat{x}_k^+ - x_k\|^2$ over multiple runs of a simulation (each with a random outcome).

Among the class of linear filters, the Kalman filter is the optimal filter in terms of minimizing the mean squared error, which is proved, e.g., in [3]. Furthermore, if the process and measurement noise are Gaussian, it is the optimal filter among all possible filters. It should be added that to achieve optimality, the assumptions for noise must also be satisfied, i.e., it must be zero-mean and uncorrelated [3].

[1] Not to be confused with a linear transformation. It is an affine transformation.

The Kalman filter is a consistent estimator, meaning it provides increasingly accurate estimates as the amount of measurement data increases. Thus, the estimator converges in mean square criterion[2] to the true state value [4]. However, this property is only achieved if all assumptions are met. It is assumed that the noise is zero-mean and uncorrelated and that we can describe the dynamic system perfectly by a linear state-space model for which we know the exact values of the matrices F_k, Q_k, H_k, and R_k, along with the initial state and covariance [1]. In practice, the Kalman filter estimator may not converge. Divergence can occur, mainly due to modeling errors and finite precision arithmetic on a computer [5].

2.6 Nonlinear filtering

Although the Kalman filter is a powerful tool, it can only operate with a linear state-space model. In the real world, however, virtually all systems are nonlinear [1]. In order to handle a nonlinear state-space model, the Kalman filter needs to be modified.

One possible modification of the Kalman filter for nonlinear models is the extended Kalman filter (EKF). The key idea of EKF is to linearize the nonlinear state-space model functions using Taylor expansion around the current estimate. Thus, the functions we linearize must be differentiable. The linearized functions are then applicable in the standard Kalman filter. Since the EKF uses first-order linearization, in the case of highly nonlinear functions, the estimation may be too imprecise, which may lead to divergence. Despite its limitations, the EKF remains a suitable option for many nonlinear estimation problems. There are also versions of EKF working with higher orders, which are more computationally demanding [1].

Another way to deal with nonlinearity is to use the unscented Kalman filter (UKF). Unlike the EKF, the UKF does not perform linearization. Instead, it selects a few representative points, called sigma points, which are used to capture the statistical properties of the nonlinear system. It further propagates the sigma points through the nonlinear functions of the state-space model, which allows it to compute the necessary means and covariances for use in the standard Kalman filter. The UKF becomes particularly useful for nonlinear systems where linearization may lead to inaccuracies [6].

An approach to nonlinear filtering that is similar to the UKF is the cubature Kalman filter (CKF). The CKF uses a different method to calculate representative points, which are called cubature points. We will introduce and derive the CKF thoroughly in the following chapter.

[2] The MSE converges to 0 as time k increases.

Chapter 3

Cubature Kalman filter

The method of nonlinear state estimation we focus on in this thesis is the cubature Kalman filter, as proposed by Arasaratnam and Haykin in their work [7]. The key idea behind this filter is a transformation to spherical-radial coordinates, allowing for the use of symmetry to significantly reduce the amount of computation required. In this chapter, we will carefully derive the CKF and describe its properties.

3.1 Preliminary mathematical principles

To comprehend the derivation of the CKF, it is necessary to introduce certain mathematical principles that the CKF heavily relies on.

First, we will introduce the transformation to a spherical-radial coordinate system with an emphasis on integration. Then we will show how to integrate certain functions over the surface of the unit n-sphere. Last, we will introduce some methods of numerical integration, namely, the Gaussian quadrature.

In this section, using the introduced mathematical principles, we will also obtain several results that will be crucial for the derivation of the CKF.

3.1.1 Transformation to spherical-radial coordinates

It is not always necessary to use the standard Cartesian coordinate system. Sometimes it is convenient to use another coordinate system, such as the spherical-radial coordinate system. The advantage of the spherical-radial coordinate system is that we can leverage its symmetry, which makes it easier to solve problems that are difficult to solve in the Cartesian coordinate system.

As the name suggests, the spherical-radial coordinate system uses the unit n-sphere[1] to represent the direction of the vector and a radius to represent

[1] Generalization of the sphere for n dimensions, since we are not bound to a specific dimension.

the magnitude of the vector. These two pieces of information are sufficient to express any vector. Specifically, an arbitrary vector $x \in \mathbb{R}^n$, represented in Cartesian coordinates, can be expressed as

$$x = ry, \tag{3.1}$$

where $\|y\| = 1$ and $r = \|x\|$.

Since the vector y always lies on the unit n-sphere, a vector of dimension $n-1$ is sufficient to express it. For example, in two dimensions, we only need one angle to accurately express the y vector.

3.1.1.1 Two-dimensional spherical-radial coordinates: The polar coordinates

The spherical-radial coordinate system is often used in two dimensions, where it is called the polar coordinate system. Although we will not limit ourselves to two dimensions when deriving the CKF, the polar coordinate system will serve as the simplest example to understand the spherical-radial coordinate system.

As we stated earlier, we can express any vector $x \in \mathbb{R}^2$ with the angle φ denoting its direction and a radius r which is the magnitude of the vector x. Thus, given the polar coordinates $(r, \varphi)^\intercal$, we can express the vector x in the Cartesian coordinates as

$$x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = r \begin{pmatrix} \cos\varphi \\ \sin\varphi \end{pmatrix}. \tag{3.2}$$

If we have a vector x given in the Cartesian coordinates $(x_1, x_2)^\intercal$, we can express it in the polar coordinates as

$$x = \begin{pmatrix} r \\ \varphi \end{pmatrix}, \tag{3.3}$$

for

$$r = \|x\| = \sqrt{x_1^2 + x_2^2}, \tag{3.4}$$

$$\varphi = \begin{cases} \arccos(\frac{x_1}{r}) & \text{if } x_2 \geq 0 \text{ and } r \neq 0, \\ -\arccos(\frac{x_1}{r}) & \text{if } x_2 < 0 \text{ and } r \neq 0, \\ 0 & \text{if } r = 0. \end{cases} \tag{3.5}$$

In Figure 3.1, we can see an example of polar coordinates being used to represent the vector $x = \left(\sqrt{2}, \sqrt{2}\right)^\intercal$ with $r = 2$ and $\varphi = \frac{\pi}{4}$.

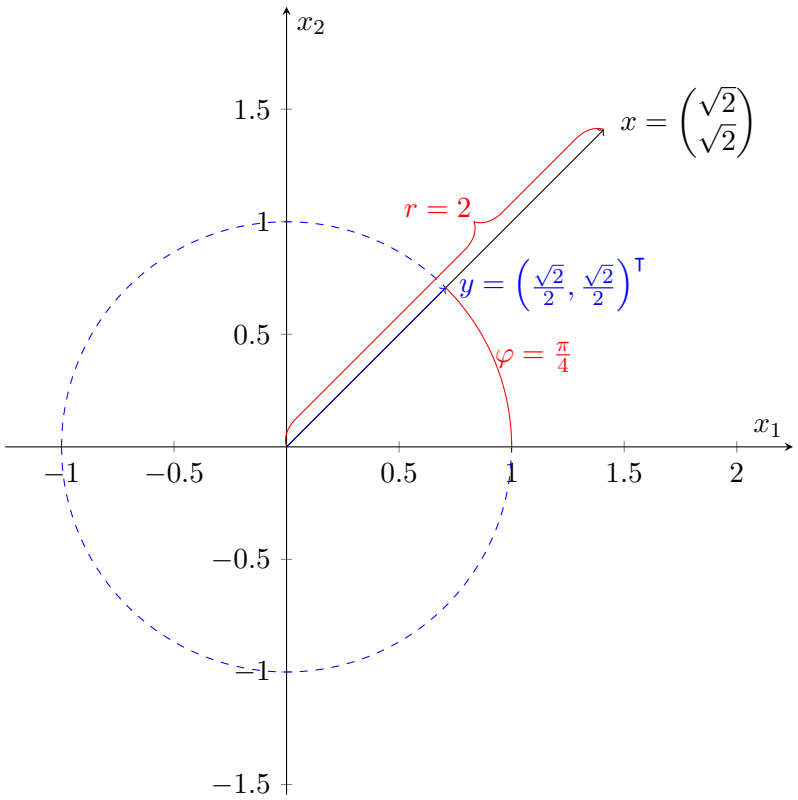

Figure 3.1 Transformation of the vector $x = \left(\sqrt{2}, \sqrt{2}\right)^\intercal$ to spherical-radial (polar) coordinates.

3.1.1.2 Integration in spherical-radial coordinates

It is often useful to transform an integral from Cartesian coordinates to spherical-radial coordinates. This can be achieved by the change of variables theorem.

▶ **Theorem 3.1** (Change of variables theorem)**.** *Let $\Omega \subseteq \mathbb{R}^n$ be a region and let U be an open set containing Ω so that $g\colon U \to \mathbb{R}^n$ is a bijective function with continuous partial derivatives at each point in U. Suppose $f\colon g(\Omega) \to \mathbb{R}$ and $f(g)|\det(\mathrm{D}g)|\colon \Omega \to \mathbb{R}$ are both integrable. Then*

$$\int_{g(\Omega)} f(y)\mathrm{d}y = \int_{\Omega} f(g(x))|\det(\mathrm{D}g)(x)|\mathrm{d}x. \tag{3.6}$$

◀

Proof of this theorem can be found, e.g., in [8].

For now, let us consider a two-dimensional case. Recalling Equation 3.2, we perform the change of variables using

$$g(r,\varphi) = \begin{pmatrix} r\cos\varphi \\ r\sin\varphi \end{pmatrix}. \tag{3.7}$$

Thus, for an arbitrary function $f\colon \mathbb{R}^2 \to \mathbb{R}$ defined in the Cartesian coordinate system, we get

$$\iint_{\mathbb{R}^2} f(x_1,x_2)\mathrm{d}x_1\mathrm{d}x_2 = \int_0^{\infty}\int_0^{2\pi} f(r\cos\varphi, r\sin\varphi)|\det(\mathrm{D}g)(r,\varphi)|\mathrm{d}\varphi\mathrm{d}r, \tag{3.8}$$

essentially integrating over every direction and radius, covering the whole $\mathbb{R}^2$. Now we just have to evaluate $|\det(\mathrm{D}g)(r,\varphi)|$. We get

$$\det(\mathrm{D}g)(r,\varphi) = \begin{vmatrix} \frac{\partial r\cos\varphi}{\partial r} & \frac{\partial r\cos\varphi}{\partial\varphi} \\ \frac{\partial r\sin\varphi}{\partial r} & \frac{\partial r\sin\varphi}{\partial\varphi} \end{vmatrix} = \begin{vmatrix} \cos\varphi & -r\sin\varphi \\ \sin\varphi & r\cos\varphi \end{vmatrix} = r\sin^2\varphi + r\cos^2\varphi = r. \tag{3.9}$$

Since r is always nonnegative, we have

$$\iint_{\mathbb{R}^2} f(x_1,x_2)\mathrm{d}x_1\mathrm{d}x_2 = \int_0^{\infty}\int_0^{2\pi} f(r\cos\varphi, r\sin\varphi)r\mathrm{d}\varphi\mathrm{d}r. \tag{3.10}$$

We can take advantage of the spherical-radial form to solve integrals that would be difficult to solve in the Cartesian form.

▶ **Example 3.2** (The Gaussian integral)**.** Let us consider an integral I in a form

$$I = \int_{\mathbb{R}} e^{-x^2}\mathrm{d}x. \tag{3.11}$$

Although this integral is only a single-variable integral, we can still utilize the spherical-radial coordinate system to solve it. By taking the square of this integral, we can introduce another dimension. This gives us

$$I^2 = \left(\int_{\mathbb{R}} e^{-x^2} \mathrm{d}x \right)^2 = \int_{\mathbb{R}} e^{-x^2} \mathrm{d}x \int_{\mathbb{R}} e^{-y^2} \mathrm{d}y = \iint_{\mathbb{R}^2} e^{-x^2-y^2} \mathrm{d}x\mathrm{d}y. \tag{3.12}$$

Now we can use Equation 3.10 to transform the integral to spherical-radial coordinates, so we get

$$I^2 = \int_0^{\infty} \int_0^{2\pi} e^{-r^2\cos^2\varphi - r^2\sin^2\varphi} r \mathrm{d}\varphi \mathrm{d}r \tag{3.13}$$

$$= \int_0^{\infty} \int_0^{2\pi} e^{-r^2} r \mathrm{d}\varphi \mathrm{d}r \tag{3.14}$$

$$= 2\pi \int_0^{\infty} e^{-r^2} r \mathrm{d}r \tag{3.15}$$

$$= 2\pi \int_{-\infty}^{0} \frac{1}{2} e^{s} \mathrm{d}s \tag{3.16}$$

$$= \pi. \tag{3.17}$$

Since e^{-x^2} is a nonnegative function, we obtain the solution

$$I = \sqrt{\pi}. \tag{3.18}$$

◀

Let us now define the surface of the unit n-sphere as

$$U_n = \{ y \in \mathbb{R}^n | \ \|y\| = 1 \}. \tag{3.19}$$

Similarly to the two-dimensional case, for an arbitrary dimension n it holds [9]

$$\int_{\mathbb{R}^n} f(x) \mathrm{d}x = \int_0^{\infty} \int_{U_n} f(ry) r^{n-1} \mathrm{d}\sigma(y) \mathrm{d}r, \tag{3.20}$$

where $\sigma(y)$ denotes the spherical surface measure for $y \in U_n$. The spherical surface measure $\sigma(y)$ essentially represents the area of the surface element at each point y on the surface of the unit n-sphere U_n. Thus we integrate over each such point, which represents the direction, and over each radius. This gives us the complete $\mathbb{R}^n$.

▶ Note 3.3. To satisfy the conditions of Theorem 3.1 we have so far only considered f as a scalar-valued function. Equation 3.20 holds for vector-valued functions as well, since the integration of a vector-valued function is simply an integration of its compounds. When deriving the CKF we will encounter the need to integrate a vector-valued function using the spherical-radial transformation. ◀

3.1.2 Integration of monomials over the surface of the unit n-sphere

In Equation 3.20 we introduced an integral over the surface of the unit n-sphere, which we also call the spherical integral. The question is how to solve such an integral. For our purposes, we are satisfied with the case where our integrand is just a monomial.

▶ **Definiton 3.4** (Monomial)**.** *For $x \in \mathbb{R}^n$, we call the function $\mathcal{M}\colon \mathbb{R}^n \to \mathbb{R}$ a monomial if*

$$\mathcal{M}(x) = \prod_{i=1}^{n} x_i^{d_i}, \tag{3.21}$$

where $d_1, \ldots, d_n$ are nonnegative integers. ◀

An important property of a monomial is its degree. Often, we want to have some control over the monomial, which we can express by limiting the degree.

▶ **Definiton 3.5** (Degree of a monomial)**.** *Let $\mathcal{M}(x) = \prod_{i=1}^{n} x_i^{d_i}$ be a monomial for any $x \in \mathbb{R}^n$. We define the function* deg *as*

$$\deg(\mathcal{M}) = \sum_{i=1}^{n} d_i. \tag{3.22}$$

We call $\deg(\mathcal{M})$ *the degree of the monomial $\mathcal{M}$.* ◀

▶ **Example 3.6.** Let us have a monomial $\mathcal{M}(x) = x_1^3 x_2$ for $x = (x_1, x_2)^\intercal \in \mathbb{R}^2$. Then, since $d_1 = 3$ and $d_2 = 1$, it holds

$$\deg(\mathcal{M}) = d_1 + d_2 = 3 + 1 = 4. \tag{3.23}$$

◀

To solve an integral in a form

$$\int_{U_n} \mathcal{M}(x) \mathrm{d}\sigma(x), \tag{3.24}$$

we use an approach that transforms it to the product of n single-variable integrals. Stroud [10] uses the spherical-radial coordinate system to come up with the following product formula for such integrals.

▶ **Theorem 3.7** (Product formula for spherical integrals of monomials [10])**.** *Let us have an integral I in the form*

$$I = \int_{U_n} \prod_{i=1}^{n} x_i^{d_i} \mathrm{d}\sigma(x). \tag{3.25}$$

Then the value of I is equal to the product of integrals

$$\int_{-\pi}^{\pi} (\cos\varphi_1)^{c_1} (\sin\varphi_1)^{d_2} \mathrm{d}\varphi_1, \tag{3.26}$$

$$\int_{-\pi/2}^{\pi/2} (\cos\varphi_2)(\cos\varphi_2)^{c_2} (\sin\varphi_2)^{d_3} \mathrm{d}\varphi_2, \tag{3.27}$$

$$\vdots$$

$$\int_{-\pi/2}^{\pi/2} (\cos\varphi_{n-1})^{n-2} (\cos\varphi_{n-1})^{c_{n-1}} (\sin\varphi_{n-1})^{d_n} \mathrm{d}\varphi_{n-1}, \tag{3.28}$$

where

$$c_k = \sum_{i=1}^{k} d_i, \quad k \in \{1, \ldots, n\}. \tag{3.29}$$

◀

We will demonstrate the use of Theorem 3.7 with two examples whose results will be useful for deriving the CKF. Before we proceed, let us first define the gamma function that we will use to express the result we obtain.

▶ **Definiton 3.8** (Gamma function [9])**.** *Let $\Gamma : \mathbb{R} \to \mathbb{R}$ be a function defined as*

$$\Gamma(x) = \int_0^{\infty} t^{x-1} e^{-t} \mathrm{d}t, \tag{3.30}$$

for $x > 0$. We call the function Γ the gamma function. ◀

The gamma function extends the factorial function to non-integers since it holds that $\Gamma(n+1) = n!$, for $n \in \mathbb{N}_0$ [9]. Let us formulate a property of the gamma function that will be useful for our later calculations. This property also partially proves the factorial behavior of the gamma function.

▶ **Theorem 3.9.** *For the gamma function Γ and $x > 0$ it holds*

$$\Gamma(x+1) = x\,\Gamma(x). \tag{3.31}$$

◀

Proof. From Definition 3.8 we get

$$\Gamma(x+1) = \int_0^\infty t^x e^{-t} \mathrm{d}t. \tag{3.32}$$

Using integration by parts, we obtain

$$\Gamma(x+1) = \left[t^x(-e^{-t})\right]_{t=0}^{t=\infty} - \int_0^\infty x t^x(-e^{-t})\mathrm{d}t \tag{3.33}$$

$$= x \int_0^\infty t^x e^{-t} \mathrm{d}t \tag{3.34}$$

$$= x\,\Gamma(x), \tag{3.35}$$

where we used the fact that $\lim_{t\to\infty} t^x e^{-t} = 0$. □

The gamma function is used to express the surface area of the unit n-sphere.

▶ **Theorem 3.10** (Surface area of the unit n-sphere)**.** *For the surface area A_n of the unit n-sphere it holds*

$$A_n = \frac{2\sqrt{\pi^n}}{\Gamma(\frac{n}{2})}. \tag{3.36}$$

◀

Proof of this theorem can be found, e.g., in [9].

▶ **Example 3.11.** Let us consider an integral I in a form

$$I = \int_{U_n} 1 \mathrm{d}\sigma(x). \tag{3.37}$$

This integral is as simple as it can get. The term 1 is in fact a monomial with degree 0, therefore $d_1 = \cdots = d_n = 0$. In Figure 3.2, we can see the visualization of this integral for a two-dimensional case where U_n is a circle.

Obviously, the integral I is equal to the surface area of the unit n-sphere, since we are integrating 1 over every point on its surface. Therefore

$$I = U_n. \tag{3.38}$$

Let us now use Theorem 3.7 to express this integral in an alternative form that may be useful later. Since $d_1 = \cdots = d_n = 0$, we also get $c_1 = \cdots = c_n =$

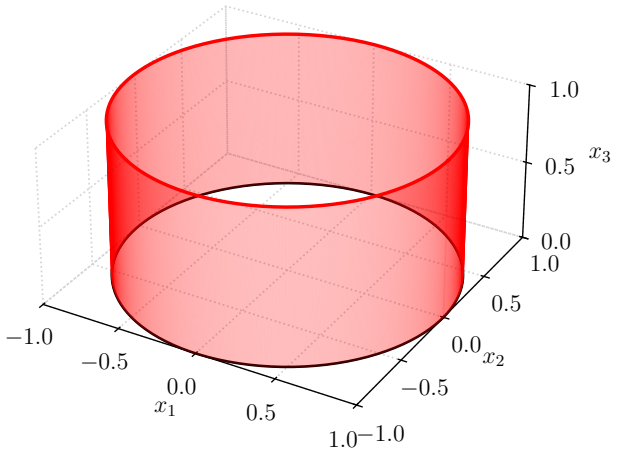

Figure 3.2 Integral of 1 over the surface of the unit 2-sphere.

0. Thus Theorem 3.7 yields

$$I = \int_{-\pi}^{\pi} 1 \mathrm{d}\varphi_1 \int_{-\pi/2}^{\pi/2} (\cos\varphi_2)\mathrm{d}\varphi_2 \cdots \int_{-\pi/2}^{\pi/2} (\cos\varphi_{n-1})^{n-2}\mathrm{d}\varphi_{n-1} \tag{3.39}$$

$$= 2\pi \prod_{i=1}^{n-2} \int_{-\pi/2}^{\pi/2} (\cos\varphi)^i \mathrm{d}\varphi. \tag{3.40}$$

◀

Before we continue with the next example, let us define the beta function that we will use in the example to solve the spherical integral.

▶ **Definiton 3.12** (Beta function [11])**.** *Let $a > 0$ and $b > 0$. The function* $\mathrm{B} : \mathbb{R}^2 \to \mathbb{R}$*, defined as*

$$\mathrm{B}(a, b) = \int_0^1 t^{a-1}(1-t)^{b-1}\mathrm{d}t, \tag{3.41}$$

is called the beta function. ◀

The beta function can be expressed in an alternative form involving the gamma function. Both forms will be useful for our purposes.

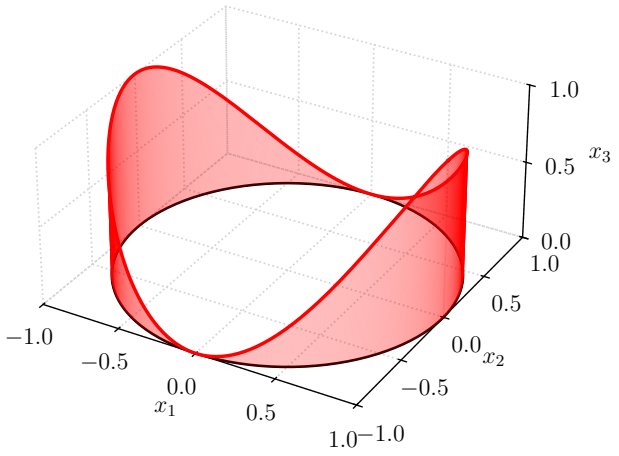

Figure 3.3 Integral of x_1^2 over the surface of the unit 2-sphere.

▶ **Theorem 3.13** (Alternative form of the beta function)**.** *For the beta function B, $a > 0$, and $b > 0$ it holds*

$$\mathrm{B}(a,b) = \frac{\Gamma(a)\,\Gamma(b)}{\Gamma(a+b)}. \tag{3.42}$$

◀

Proof of this theorem can be found, e.g., in [11].

▶ **Example 3.14.** Let us now consider an integral I in a form

$$I = \int_{U_n} x_1^2 \mathrm{d}\sigma(x). \tag{3.43}$$

In Figure 3.3, we can see the visualization of this integral for a two-dimensional case where U_n is a circle.

By applying the product formula according to Theorem 3.7, we get

$$I = \int_{-\pi}^{\pi} (\cos\varphi_1)^2 \mathrm{d}\varphi_1 \tag{3.44}$$

$$\times \int_{-\pi/2}^{\pi/2} (\cos\varphi_2)(\cos\varphi_2)^2 \mathrm{d}\varphi_2 \tag{3.45}$$

$$\vdots$$

$$\times \int_{-\pi/2}^{\pi/2} (\cos\varphi_{n-1})^{n-2}(\cos\varphi_{n-1})^2 \mathrm{d}\varphi_{n-1} \tag{3.46}$$

$$= \pi \prod_{i=3}^{n} \int_{-\pi/2}^{\pi/2} (\cos\varphi)^i \mathrm{d}\varphi, \tag{3.47}$$

Which we can express involving the expression we obtained in Equation 3.40 in the previous example. We get

$$I = \underbrace{2\pi \prod_{i=1}^{n-2} \int_{-\pi/2}^{\pi/2} (\cos\varphi)^i \mathrm{d}\varphi}_{A_n} \frac{\int_{-\pi/2}^{\pi/2} (\cos\varphi)^{n-1}\mathrm{d}\varphi \int_{-\pi/2}^{\pi/2} (\cos\varphi)^n \mathrm{d}\varphi}{2\int_{-\pi/2}^{\pi/2} \cos\varphi \mathrm{d}\varphi \int_{-\pi/2}^{\pi/2} \cos^2\varphi \mathrm{d}\varphi} \tag{3.48}$$

$$= \frac{A_n}{2\pi} \int_{-\pi/2}^{\pi/2} (\cos\varphi)^{n-1}\mathrm{d}\varphi \int_{-\pi/2}^{\pi/2} (\cos\varphi)^n \mathrm{d}\varphi. \tag{3.49}$$

Using the substitution $t = (\cos u)^2$ in the integral that defines the beta function by Definition 3.12, we get

$$\mathrm{B}(a,b) = 2\int_0^{\pi/2} (\cos u)^{2a-1}(\sin u)^{2b-1}\mathrm{d}u. \tag{3.50}$$

Therefore, using the fact that the cosine function is an even function, we get

$$\int_{-\pi/2}^{\pi/2} (\cos\varphi)^n \mathrm{d}\varphi = 2\int_0^{\pi/2} (\cos\varphi)^n \mathrm{d}\varphi$$

$$= \mathrm{B}\left(\frac{n+1}{2}, \frac{1}{2}\right) = \frac{\Gamma(\frac{n+1}{2})\,\Gamma(\frac{1}{2})}{\Gamma(\frac{n}{2}+1)} = \frac{\sqrt{\pi}\,\Gamma(\frac{n+1}{2})}{\Gamma(\frac{n}{2}+1)}, \tag{3.51}$$

where we used Theorem 3.13 and the fact that $\Gamma(\frac{1}{2}) = \sqrt{\pi}$, which can be easily shown by a simple substitution giving the Gaussian integral we solved in Example 3.2. Thus, substituting into Equation 3.49, we obtain

$$I = \frac{A_n}{2\pi} \frac{\sqrt{\pi}\,\Gamma(\frac{n}{2})}{\Gamma(\frac{n+1}{2})} \frac{\sqrt{\pi}\,\Gamma(\frac{n+1}{2})}{\Gamma(\frac{n}{2}+1)} \tag{3.52}$$

$$= \frac{A_n}{2} \frac{\Gamma(\frac{n}{2})}{\Gamma(\frac{n}{2}+1)} \tag{3.53}$$

and using Theorem 3.9, we finally get

$$I = \frac{A_n}{n}. \tag{3.54}$$

◀

The symmetry of the n-sphere gives us an important property concerning spherical integrals of monomials with odd degrees.

▶ **Theorem 3.15** (Spherical integrals of odd-degree monomials)**.** *Let* $\mathcal{M}(x)$ *be a monomial for any* $x \in \mathbb{R}^n$*. Suppose* $\deg(\mathcal{M})$ *is an odd integer. Then*

$$\int_{U_n} \mathcal{M}(x) \mathrm{d}\sigma(x) = 0. \tag{3.55}$$

◀

Proof. Since $\mathcal{M}(x) = \prod_{i=1}^{n} x_i^{d_i}$ and $\deg(\mathcal{M}) = \sum_{i=1}^{n} d_i$, we see that in order for $\deg(\mathcal{M})$ to be an odd integer, at least one d_i must be an odd integer for $i \in \{1, \ldots, n\}$. Assume, without loss of generality, that d_1 is an odd integer. Therefore, $\mathcal{M}(x)$ contains the term $x_1^{d_1}$, which is an odd function of x_1. An odd function f has the property $f(-y) = -f(y)$. That implies

$$(-x_1)^{d_1} \prod_{i=2}^{n} x_i^{d_i} = -x_1^{d_1} \prod_{i=2}^{n} x_i^{d_i}, \tag{3.56}$$

which results in

$$\mathcal{M}\left(\begin{pmatrix} -x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}\right) = -\mathcal{M}\left(\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}\right). \tag{3.57}$$

Thus, when integrating over a symmetric domain, in our case the surface of the unit n-sphere, for every $x = (x_1, x_2, \ldots, x_n)^\intercal \in U_n$, there exists exactly one $x' = (-x_1, x_2, \ldots, x_n)^\intercal \in U_n$. Since $\mathcal{M}(x)$ and $\mathcal{M}(x')$ cancel each other out, the resulting integral is zero. □

3.1.3 Numerical integration

Numerous definite integrals encountered in practice can be exceedingly challenging to solve analytically. Thus, we often need to use numerical methods to obtain an approximation of the resulting value. In many practical applications, if the approximation is accurate enough, it is sufficient to use it instead of the actual analytical value.

Typically, the goal is to approximate a nonnegatively weighted integral. That is, an integral of the form

$$I = \int_{\mathcal{D}} w(x) f(x) \mathrm{d}x, \tag{3.58}$$

where $\mathcal{D} \subseteq \mathbb{R}^n$, $f\colon \mathbb{R}^n \to \mathbb{R}^{n'}$ is an arbitrary function, and $w\colon \mathbb{R}^n \to \mathbb{R}_0^+$ is a weighting function for $n, n' \in \mathbb{N}$. Note that in the special case where $w(x) = 1$, we obtain the integral of f.

The standard and well-researched [12] approach to approximating such integrals is to find the set of points $\{x_1, \ldots, x_m\} \subset \mathbb{R}^n$ and the set of corresponding weights $\{\omega_1, \ldots, \omega_m\} \subset \mathbb{R}$ so that the sum

$$I \approx \sum_{i=1}^{m} \omega_i f(x_i) \tag{3.59}$$

is the best approximation by a certain criterion. The advantage of this method is that once we obtain the points $x_1, \ldots, x_m$ and the weights $\omega_1, \ldots, \omega_m$, we only need to evaluate the function f m times to get the resulting approximation of I. For many standard forms of I, usually defined by the weighting function and the region of integration, the points and weights are already precomputed.

Formulas given by Equation 3.59 are called m-point cubature rules. If I is just a single-variable integral, i.e., $n = 1$, we call them quadrature rules. In our case, where w is a nonnegative function, we talk about the Gaussian cubature [12].

The question remains what criterion to choose for the accuracy of the approximation. Typically, we want Equation 3.59 to be exact for all polynomials of degree up to k. This criterion is equivalent to being exact for all monomials of degree up to k. This is because polynomials are sums of monomials, each multiplied by a constant coefficient, and the integral has the property of linearity. Hence, we aim to find the cubature points and weights so that

$$I = \sum_{i=1}^{m} \omega_i \mathcal{M}(x_i), \tag{3.60}$$

for monomials $\mathcal{M}$ such that $\deg(\mathcal{M}) \leq k$.

Essentially, the higher m we choose, the more accurate the approximation will be for an arbitrary function. The price for this is higher computational

complexity. For an m-point Gaussian quadrature rule, it holds that the m points and m weights can be found to make Equation 3.59 exact for all polynomials of degree up to $2m-1$ [12].

We will now present two special cases of Gaussian quadrature. In a similar way to the quadrature rules, we will derive a higher-dimensional cubature rule later in the derivation of the CKF.

3.1.3.1 Gauss–Legendre quadrature

When using the Gauss–Legendre quadrature rules, we consider the integral I over the region $\mathcal{D} = [-1, 1]$ with the weighting function $w(x) = 1$ [13]. So we approximate the integral

$$I = \int_{-1}^{1} f(x)\mathrm{d}x, \tag{3.61}$$

where $x \in \mathbb{R}$. This form of the integral I may not be as restrictive as it appears. By using a simple substitution, we can transform any integral over a finite region into this form.

The Gauss–Legendre quadrature is not essential for our purposes. However, it will serve as a clear demonstration of the concept of cubature rules. We will not present the Gauss–Legendre quadrature in its full generality. Instead, we will derive a quadrature rule for a specific simple example. This will provide us with an insight into this method of numerical integration.

▶ **Example 3.16.** Let us find a Gauss–Legendre quadrature rule that is exact for all the polynomials of degree up to 3. That is, it needs to be exact for the monomials x^3, x^2, x, and 1. Since an m-point Gaussian quadrature rule is sufficient to solve integrals of polynomials of degree up to $2m-1$ exactly, we will derive a two-point quadrature rule.

Our goal now is to find the quadrature points x_1 and x_2 and the weights ω_1 and ω_2. This can be achieved by solving the set of equations obtained by using the monomials x^3, x^2, x, and 1 in Equation 3.60. Thus, we get

$$\int_{-1}^{1} 1\mathrm{d}x = 2 = \omega_1 \cdot 1 + \omega_2 \cdot 1, \tag{3.62}$$

$$\int_{-1}^{1} x\mathrm{d}x = 0 = \omega_1 x_1 + \omega_2 x_2, \tag{3.63}$$

$$\int_{-1}^{1} x^2\mathrm{d}x = \frac{2}{3} = \omega_1 x_1^2 + \omega_2 x_2^2, \tag{3.64}$$

$$\int_{-1}^{1} x^3\mathrm{d}x = 0 = \omega_1 x_1^3 + \omega_2 x_2^3, \tag{3.65}$$

which has a solution

$$x_1 = \frac{1}{\sqrt{3}}, \tag{3.66}$$

$$x_2 = -\frac{1}{\sqrt{3}}, \tag{3.67}$$

$$\omega_1 = 1, \tag{3.68}$$

$$\omega_2 = 1, \tag{3.69}$$

that defines the desired two-point Gauss–Legendre quadrature rule.

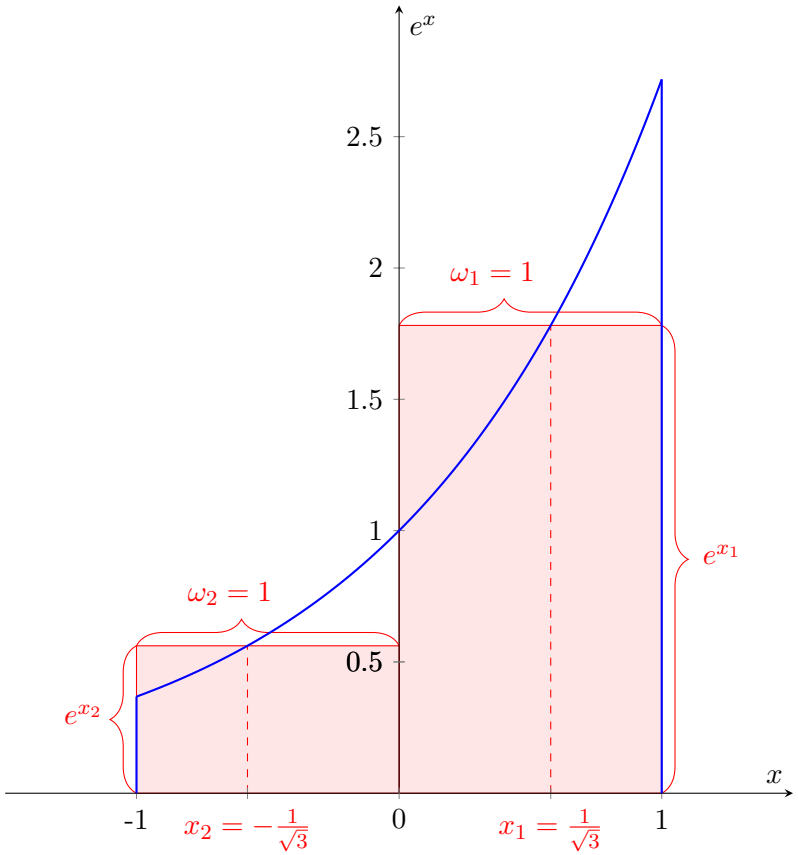

Figure 3.4 Approximation of the integral of e^x over $[-1, 1]$ using a two-point Gauss–Legendre quadrature rule.

We can now use the derived rule to approximate the integral of an arbitrary function defined on $[-1, 1]$. For demonstration purposes, let us approximate the integral of e^x as

$$\int_{-1}^{1} e^x \mathrm{d}x \approx \omega_1 e^{x_1} + \omega_2 e^{x_2} = 1 \cdot e^{\frac{1}{\sqrt{3}}} + 1 \cdot e^{-\frac{1}{\sqrt{3}}} \approx 2.3427. \tag{3.70}$$

The true value of this integral is approximately 2.3504. The approximation of this integral using the derived two-point Gauss–Legendre quadrature rule is illustrated in Figure 3.4.

◀

As we already know, an m-point Gaussian quadrature rule can accurately compute integrals of polynomials of degree up to $2m-1$. Let us now explore the effectiveness of quadrature rules with a smaller number of points. In particular, let us consider a fifth-degree polynomial

$$\mathcal{P}_5(x) = 1 + x - x^2 - 3x^3 + x^4 + 2x^5 \tag{3.71}$$

Our objective now is to approximate its integral

$$\int_{-1}^{1} \mathcal{P}_5(x)\mathrm{d}x = \frac{26}{15} = 1.7\bar{3} \tag{3.72}$$

using the Gauss–Legendre quadrature. We will use a one-point, a two-point, and a three-point Gauss–Legendre quadrature.

points	approximation	error
1	2	$0.2\bar{6}$
2	$1.\bar{5}$	$0.1\bar{7}$
3	$1.7\bar{3}$	0

Table 3.1 Performance comparison of Gauss–Legendre quadrature rules for approximating the integral of a fifth-degree polynomial based on the number of quadrature points.

In Table 3.1, we can see that the higher the number of points, the better the approximation given by the quadrature rule. The three-point quadrature rule gave an exact result, as expected. The approximations are visualized in Figures 3.5, 3.6, and 3.7.

▶ Note 3.17. In practice, the points and weights are not obtained by solving a set of equations. Instead, they are obtained by finding the roots of the Legendre polynomial [14]. Fortunately, these values have already been precomputed and are available in publicly accessible tables [12]. ◀

3.1.3.2 Generalized Gauss–Laguerre quadrature

The generalized Gauss–Laguerre quadrature rules apply to integrals over the region $\mathcal{D} = [0, \infty]$ with the weighting function $w(x) = x^\gamma e^{-x}$, where $\gamma \in \mathbb{R}$ is a parameter [13]. Unlike the Gauss–Legendre quadrature, the generalized Gauss–Laguerre quadrature allows us to approximate integrals over an infinite region. The integral I then takes the form

$$\int_0^\infty x^\gamma e^{-x} f(x)\mathrm{d}x, \tag{3.73}$$

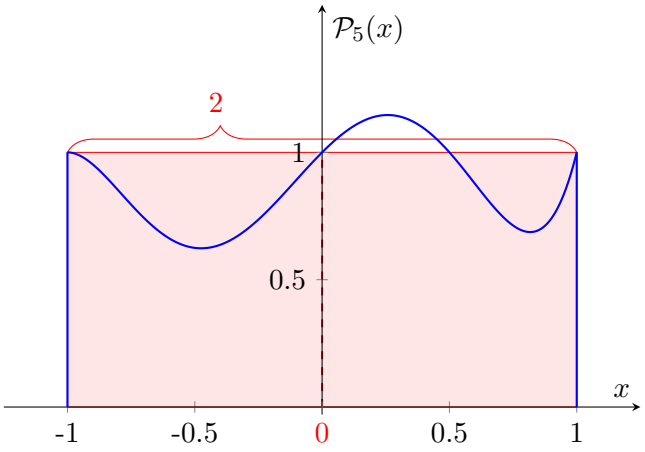

Figure 3.5 Approximation of the integral of a fifth-degree polynomial over $[-1, 1]$ using a one-point Gauss–Legendre quadrature rule.

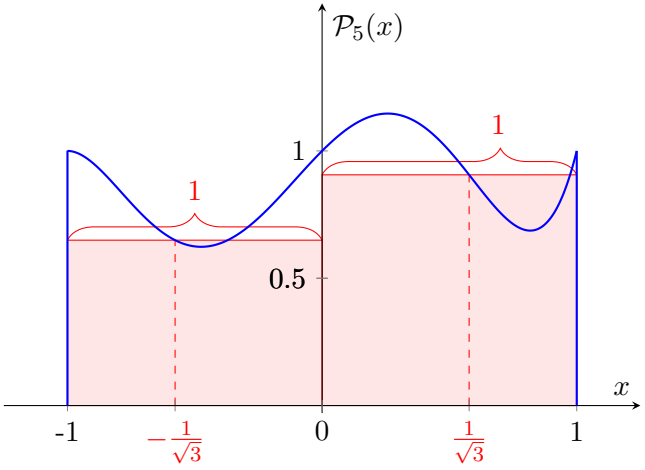

Figure 3.6 Approximation of the integral of a fifth-degree polynomial over $[-1, 1]$ using a two-point Gauss–Legendre quadrature rule.

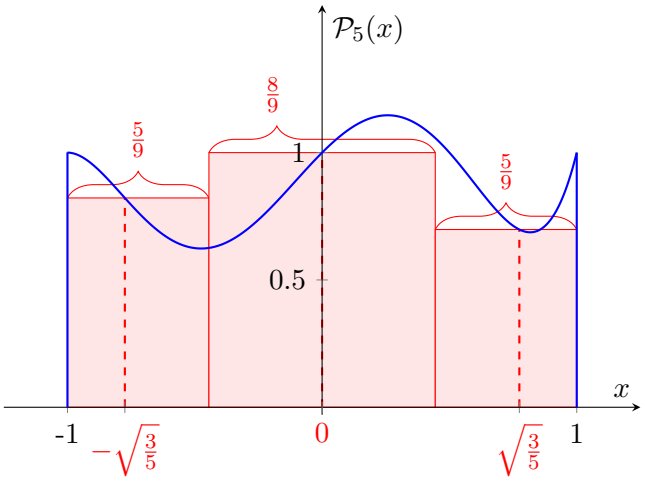

Figure 3.7 Approximation of the integral of a fifth-degree polynomial over $[-1, 1]$ using a three-point Gauss–Legendre quadrature rule.

where $x, \gamma \in \mathbb{R}$.

The points and weights for an m-order Gauss–Laguerre quadrature rule can be obtained using the generalized Laguerre polynomial [13]. However, for our later use, it is sufficient to find the weight and point for the first-order Gauss-Laguerre quadrature rule only. Thus, we can obtain the point and weight by solving a pair of equations. The following result is essential for the derivation of the CKF.

▶ **Example 3.18.** Let us find the first-order Gauss–Laguerre quadrature rule for the integral

$$\int_0^\infty f(x) x^{\frac{n}{2}-1} e^{-x} \mathrm{d}x, \tag{3.74}$$

where $n \in \mathbb{N}$.

We can use monomials of the zeroth and first degree, i.e., 1 and x, as the function f. This gives us the set of equations

$$\int_0^\infty 1 \cdot x^{\frac{n}{2}-1} e^{-x} \mathrm{d}x = \omega_1 \cdot 1, \tag{3.75}$$

$$\int_0^\infty x \cdot x^{\frac{n}{2}-1} e^{-x} \mathrm{d}x = \omega_1 x_1. \tag{3.76}$$

By recalling Definition 3.8 of the gamma function, we easily obtain

$$\Gamma\left(\frac{n}{2}\right) = \omega_1, \tag{3.77}$$

$$\Gamma\left(\frac{n}{2}+1\right) = \omega_1 x_1. \tag{3.78}$$

$$\tag{3.79}$$

Using Theorem 3.9, we get the final values

$$\omega_1 = \Gamma\left(\frac{n}{2}\right), \tag{3.80}$$

$$x_1 = \frac{n}{2}. \tag{3.81}$$

◀

3.2 Derivation of the cubature Kalman filter

In this section, we will derive the CKF based on the already introduced Kalman filter. This derivation heavily relies on the original article [7] that introduced the CKF.

3.2.1 Filtering with a nonlinear model

To construct a nonlinear filter, we need to work with arbitrary functions that make up the nonlinear state-space model. The nonlinear state-space model uses the function f to model the state and the function h to model the measurements (see Equations 1.12, 1.13).

It is often the case that the model noises are additive. This is our first assumption. That is, the state-space model becomes

$$x_k = f(x_{k-1}, u_{k-1}) + w_{k-1}, \tag{3.82}$$

$$y_k = h(x_k) + v_k, \tag{3.83}$$

where, at time k, x_k represents the state vector, u_k represents the control input vector, w_k represents the process noise vector, y_k represents the measurement vector, and v_k represents the measurement noise vector, as introduced in Chapter 1.

Our other assumption is that the noises are Gaussian. Although no real-world processes behave in an exact Gaussian manner, the Gaussian distribution still provides a good approximation and has many advantageous mathematical properties [15]. That is, we assume

$$w_k \sim \mathcal{N}(0, Q_k), \tag{3.84}$$

$$v_k \sim \mathcal{N}(0, R_k). \tag{3.85}$$

Therefore, the system states and measurements are also Gaussian due to the transformational properties of the Gaussian distribution.

Similar to the Kalman filter, the CKF operates by alternating between the time update and the measurement update. The difference lies in the method used to obtain the estimates and covariances, as linear equations cannot be used.

At each update, we obtain an estimate of the state and its covariance. Since the Gaussian distribution is defined by mean and covariance, we have enough information to form the current probability density of the state estimate, which is Gaussian.

3.2.1.1 The time update

In the time update, we have the *a posteriori* estimate $\hat{x}_{k-1}^{+}$ and the *a posteriori* covariance P_{k-1}^{+} from the previous time step. Thus, we have the *a posteriori* density

$$p(x_{k-1}|D_{k-1}) = \mathcal{N}(\hat{x}_{k-1}^{+}, P_{k-1}^{+}), \tag{3.86}$$

where D_{k-1} denotes the history of all control inputs and measurements up to time $(k-1)$, as defined in Equation 2.2.

The aim now is to compute the *a priori* density $p(x_k|D_{k-1})$. That is, to compute the *a priori* estimate $\hat{x}_k^-$ and the *a priori* covariance P_k^-. The *a priori* estimate is the expected value of the *a priori* density, that is

$$\hat{x}_k^- = \mathbb{E}\left[x_k|D_{k-1}\right]. \tag{3.87}$$

Using Equation 3.82, we can substitute x_k as

$$\hat{x}_k^- = \mathbb{E}\left[f(x_{k-1}, u_{k-1}) + w_{k-1}|D_{k-1}\right]. \tag{3.88}$$

Since the noise w_{k-1} is assumed to be zero-mean and uncorrelated, it holds

$$\begin{aligned} \hat{x}_k^- &= \mathbb{E}\left[f(x_{k-1}, u_{k-1})|D_{k-1}\right] \\ &= \int_{\mathbb{R}^n} f(x_{k-1}, u_{k-1}) p(x_{k-1}|D_{k-1}) \mathrm{d}x_{k-1} \\ &= \int_{\mathbb{R}^n} f(x_{k-1}, u_{k-1}) \mathcal{N}(\hat{x}_{k-1}^{+}, P_{k-1}^{+}) \mathrm{d}x_{k-1}. \end{aligned} \tag{3.89}$$

Thus, to obtain the estimate, it is sufficient to compute this integral. For any given function f we cannot rely on analytical solutions. A numerical integration method will be required to compute such an integral.

Similarly, for the *a priori* covariance, we get

$$\begin{aligned} P_k^- &= \mathbb{E}\left[(x_k - \hat{x}_k^-)(x_k - \hat{x}_k^-)^\intercal|D_{k-1}\right] \\ &= \mathbb{E}\left[f(x_{k-1}, u_{k-1}) f(x_{k-1}, u_{k-1})^\intercal|D_{k-1}\right] - \hat{x}_k^-(\hat{x}_k^-)^\intercal + Q_{k-1}, \end{aligned} \tag{3.90}$$

where

$$\begin{aligned}&\mathbb{E}\left[f(x_{k-1},u_{k-1})f(x_{k-1},u_{k-1})^\intercal|D_{k-1}\right]\\&=\int_{\mathbb{R}^n} f(x_{k-1},u_{k-1})f(x_{k-1},u_{k-1})^\intercal\mathcal{N}(\hat{x}_{k-1}^+,P_{k-1}^+)\mathrm{d}x_{k-1}.\end{aligned}\tag{3.91}$$

3.2.1.2 The measurement update

In the measurement update, we aim to compute the *a posteriori* density from the *a priori* density. Certain equations of the measurement update in the Kalman filter are not dependent on the state-space model functions. Thus, for a nonlinear case, we can utilize them as well. We need to compute the other variables differently. The approach will be very similar to the time update of the CKF.

From the Kalman filter, we get the following set of equations

$$K_k = P_{xy,k}P_{y,k}^{-1},\tag{3.92}$$

$$\hat{x}_k^+ = \hat{x}_k^- + K_k(y_k-\hat{y}_k),\tag{3.93}$$

$$P_k^+ = P_k^- - K_kP_{y,k}K_k^\intercal.\tag{3.94}$$

The remaining unknowns are $\hat{y}_k$, $P_{y,k}$, and $P_{xy,k}$.

The noise v_k is Gaussian, zero-mean, and uncorrelated as well. We assume the same for the prediction errors. The unknown variables can be then computed in the same way as in the time update. We get

$$\begin{aligned}\hat{y}_k &= \mathbb{E}\left[y_k|D_{k-1}\right]\\&=\int_{\mathbb{R}^n} h(x_k)\mathcal{N}(\hat{x}_k^-,P_k^-)\mathrm{d}x_k,\end{aligned}\tag{3.95}$$

$$\begin{aligned}P_{y,k} &= \mathbb{E}\left[(y_k-\hat{y}_k)(y_k-\hat{y}_k)^\intercal|D_{k-1}\right]\\&=\int_{\mathbb{R}^n} h(x_k)h(x_k)^\intercal\mathcal{N}(\hat{x}_k^-,P_k^-)\mathrm{d}x_k-\hat{y}_k\hat{y}_k^\intercal+R_k,\end{aligned}\tag{3.96}$$

$$\begin{aligned}P_{xy,k} &= \mathbb{E}\left[(x_k-\hat{x}_k^-)(y_k-\hat{y}_k)^\intercal|D_{k-1}\right]\\&=\int_{\mathbb{R}^n} x_kh(x_k)^\intercal\mathcal{N}(\hat{x}_k^-,P_k^-)\mathrm{d}x_k-\hat{x}_k^-\hat{y}_k^\intercal.\end{aligned}\tag{3.97}$$

Thus the CKF reduces to computing integrals of a certain form. The rest of the algorithm is the same as the Kalman filter.

3.2.2 Transformation to spherical-radial coordinates

The remaining problem to solve is how to compute integrals of the form

$$\int_{\mathbb{R}^n} f(x)\mathcal{N}(\mu,\Sigma)\mathrm{d}x,\tag{3.98}$$

with f being an arbitrary (nonlinear) function.

Let us, for now, consider an integral in a simpler but similar form

$$I = \int_{\mathbb{R}^n} f(x)\exp(-xx^\intercal)\mathrm{d}x. \tag{3.99}$$

Since f is an arbitrary function, this integral is not tractable in general. Thus, we need to use numerical methods of integration. We will derive our own cubature rule for this integral. For the cubature rule to be efficient, we exploit the symmetry by transforming the integral I to the spherical-radial coordinate system.

Using Equation 3.20, we obtain

$$I = \int_0^\infty \int_{U_n} f(ry)r^{n-1}\exp(-r^2)\mathrm{d}\sigma(y)\mathrm{d}r, \tag{3.100}$$

where U_n is the surface of the unit n-sphere. We can express this spherical-radial integral as two integrals: the spherical and the radial integral. The spherical integral integrates over the surface of the unit n-sphere and takes the form

$$S(r) = \int_{U_n} f(ry)\mathrm{d}\sigma(y). \tag{3.101}$$

The radial integral is the integral I utilizing the spherical integral as a function of r. It integrates over every radius and takes the form

$$I = \int_0^\infty S(r)r^{n-1}\exp(-r^2)\mathrm{d}r. \tag{3.102}$$

We will introduce a cubature rule to compute the spherical integral. Then we will use the Gaussian quadrature to compute the radial integral, as it is just a single-variable integral.

3.2.3 Spherical cubature rule

Before we construct our cubature rule for the spherical integral, let us define a fully symmetric set.

▶ **Definiton 3.19** (Symmetric set)**.** *Let $u_1 \geq u_2 \geq \ldots \geq u_r > 0$ for $r \leq n$. We call the set $U \subset \mathbb{R}^n$ of all vectors obtained by permutating and changing the sign of the compounds of the vector $(u_1, u_2, \ldots, u_r, 0, \ldots, 0)^\intercal \in \mathbb{R}^n$ a fully symmetric set.*

We denote the set U by $[u_1, u_2, \ldots, u_r]$ and use $[u_1, u_2, \ldots, u_r]_i$ to denote the i-th point of this set. ◀

▶ **Example 3.20.** For the set $[1] \subset \mathbb{R}^2$ it holds

$$[1] = \left\{ \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} -1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ -1 \end{pmatrix} \right\}. \tag{3.103}$$

◀

We can see that a fully symmetric set $[u] \subset \mathbb{R}^n$ contains $2n$ elements.

Let us now construct a third-degree cubature rule that takes the form

$$\int_{U_n} f(y) \mathrm{d}\sigma(y) \approx \omega \sum_{i=1}^{2n} f\left([u]_i\right), \tag{3.104}$$

that is, we want to find ω and u such that this cubature rule is exact for all monomials of degree up to 3.

It has been observed that in practice, higher than third-degree cubature rules yield no improvement or even make the performance worse. Furthermore, the third-degree cubature rule provides efficient computation and helps mitigate the curse of dimensionality [7].

Thanks to the symmetry of the integration domain and the fact that the integral is unweighted, meaning it is associated with a weighting function $w(y) = 1$, which is constant for each point of integration, it is sufficient to consider the same weight ω for every cubature point.

For all monomials of an odd degree, the left side of Equation 3.104 is zero due to Theorem 3.15. Following the same reasoning, the right side is also zero, since for every vector x in a fully symmetric set U, there exists $-x \in U$. Thus, the spherical cubature rule is exact for all odd-degree monomials.

It remains to fulfill the requirement that the spherical cubature rule is exact for all monomials $\mathcal{M}$ such that $\deg(\mathcal{M}) \in \{0, 2\}$. Because the cubature points form a symmetric set, it suffices to consider the monomials 1 and y_1^2 as a function f. Hence, using the same method as in Example 3.16, we get the

set of equations

$$\int_{U_n} 1 \mathrm{d}\sigma(y) = \omega \sum_{i=1}^{2n} 1 = 2\omega n, \tag{3.105}$$

$$\begin{aligned}\int_{U_n} y_1^2 \mathrm{d}\sigma(y) &= \omega \sum_{i=1}^{2n} ([u]_i)_1^2 \\ &= \omega\Big(((u,0,\ldots,0)^\intercal)_1^2 \\ &\qquad + ((0,u,\ldots,0)^\intercal)_1^2 \\ &\qquad \vdots \\ &\qquad + ((0,0,\ldots,u)^\intercal)_1^2 \\ &\qquad + ((-u,0,\ldots,0)^\intercal)_1^2 \\ &\qquad + ((0,-u,\ldots,0)^\intercal)_1^2 \\ &\qquad \vdots \\ &\qquad + ((0,0,\ldots,-u)^\intercal)_1^2\Big) \\ &= \omega\big(u^2 + 0^2 + \ldots + 0^2 \\ &\qquad +(-u)^2 + 0^2 + \ldots + 0^2\big) \\ &= 2\omega u^2.\end{aligned} \tag{3.106}$$

In Example 3.11, we obtained the result

$$\int_{U_n} 1 \mathrm{d}\sigma(y) = A_n, \tag{3.107}$$

where A_n is the surface area of the unit n-sphere. Further, in Example 3.14, we obtained the result

$$\int_{U_n} y_1^2 \mathrm{d}\sigma(y) = \frac{A_n}{n}. \tag{3.108}$$

Hence, the set of equations becomes

$$A_n = 2\omega n, \tag{3.109}$$

$$\frac{A_n}{n} = 2\omega u^2, \tag{3.110}$$

which has the solution

$$\omega = \frac{A_n}{2n}, \tag{3.111}$$

$$u = 1, \tag{3.112}$$

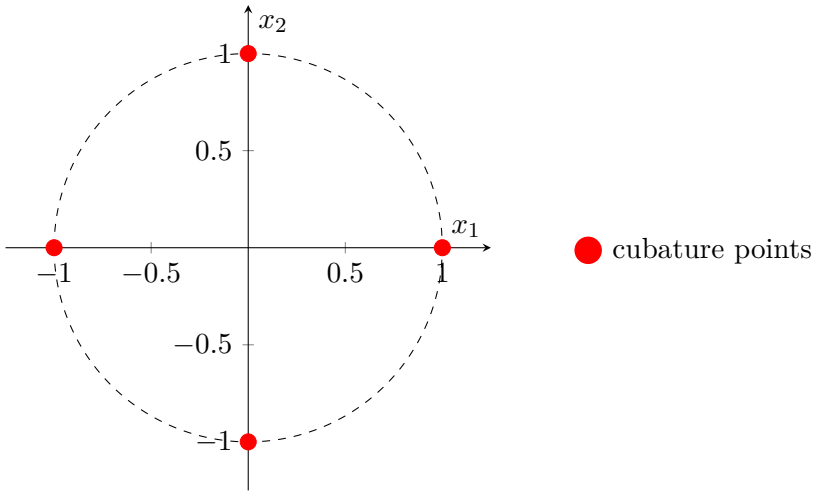

Figure 3.8 Cubature points for the third-degree spherical cubature rule in two dimensions.

since $u > 1$ from the definition of the symmetric set. The third-degree spherical cubature rule hence takes the form

$$\int_{U_n} f(y)\mathrm{d}\sigma(y) \approx \frac{A_n}{2n} \sum_{i=1}^{2n} f\left([1]_i\right). \tag{3.113}$$

Since the cubature points form the set $[1] \subset U_n$, they are located at the intersection of the surface of the unit n-sphere and its axes, which is illustrated in Figure 3.8 for two dimensions.

Thanks to the symmetry of the n-sphere, the spherical cubature rule consists of only $2n$ cubature points. This means that the number of evaluations of the function f scales linearly with the dimension. Furthermore, we obtain the $2n$ cubature points by solving the set of only two equations.

3.2.4 Radial Gaussian quadrature rule

To approximate the radial integral

$$I = \int_0^\infty S(r) r^{n-1} \exp(-r^2)\mathrm{d}r, \tag{3.114}$$

we use the generalized Gauss-Laguerre quadrature. In order to use the generalized Gauss-Laguerre quadrature, the integral needs to be in a slightly different form (see Equation 3.73). Thus, we perform a change of variable by $t = r^2$ that gives us

$$I = \frac{1}{2} \int_0^\infty \tilde{S}(t) t^{\frac{n}{2}-1} \exp(-t)\mathrm{d}t, \tag{3.115}$$

where $\tilde{S}(t) = S(\sqrt{t})$. We want the quadrature rule to be exact for $S(r)$ being all monomials of degree up to 3. Since $S(r) = 0$ for all monomials of an odd degree, it suffices to consider $S(r)$ being 1 and r^2.

Let us now construct a first-order generalized Gauss-Laguerre quadrature rule that takes the form

$$\int_0^\infty \tilde{S}(t) t^{\frac{n}{2}-1} \exp(-t) \mathrm{d}t \approx \omega_1 \tilde{S}(x_1). \tag{3.116}$$

In Example 3.18, we obtained the result

$$\omega_1 = \Gamma\left(\frac{n}{2}\right), \tag{3.117}$$

$$x_1 = \frac{n}{2}. \tag{3.118}$$

Since this rule is a first-order rule, it is exact for $\tilde{S}(t)$ being 1 and t, thus it is exact for $S(r)$ being 1 and r^2.

We can now change the variables back to obtain the final radial quadrature rule

$$\int_0^\infty S(r) r^{n-1} \exp(-r^2) \mathrm{d}r \approx \frac{1}{2}\Gamma\left(\frac{n}{2}\right) S\left(\sqrt{\frac{n}{2}}\right). \tag{3.119}$$

3.2.5 Combined spherical-radial rule for Gaussian-weighted integrals

We can combine the spherical rule in Equation 3.113 and the radial rule in Equation 3.119 to approximate an integral weighted by $\exp\left(-x^\intercal x\right)$ as

$$\int_0^\infty \int_{U_n} f(ry) r^{n-1} \exp(-r^2) \mathrm{d}\sigma(y) \mathrm{d}r \approx \frac{1}{2}\Gamma\left(\frac{n}{2}\right) \frac{A_n}{2n} \sum_{i=1}^{2n} f\left(\sqrt{\frac{n}{2}}[1]_i\right). \tag{3.120}$$

Using the formula for the surface area of the unit n-sphere, as stated in Theorem 3.10, we can simplify the approximation term. Thus, we get

$$\int_{\mathbb{R}^n} f(x) \exp(-xx^\intercal) \mathrm{d}x \approx \frac{\sqrt{\pi^n}}{2n} \sum_{i=1}^{2n} f\left(\sqrt{\frac{n}{2}}[1]_i\right), \tag{3.121}$$

where we expressed the integral in the Cartesian form again since the spherical-radial cubature rule is already derived.

Our initial desire, however, was to approximate the Gaussian-weighted integral

$$I = \int_{\mathbb{R}^n} f(x) \mathcal{N}(\mu, \Sigma) \mathrm{d}x. \tag{3.122}$$

It is possible to transform this integral into a form for which we have obtained the spherical-radial cubature rule in Equation 3.121. For the integral I, it holds

$$I = \int_{\mathbb{R}^n} f(x) \frac{1}{\sqrt{(2\pi)^n \det(\Sigma)}} \exp\left(-\frac{1}{2}(x-\mu)^\intercal \Sigma^{-1}(x-\mu)\right) \mathrm{d}x. \tag{3.123}$$

We can now perform a change of variables using Theorem 3.1 with $x = \sqrt{2\Sigma}y + \mu$, where we define $\sqrt{\Sigma}$ as $\sqrt{\Sigma}\sqrt{\Sigma}^\intercal = \Sigma$ which always holds since Σ is a positive definite matrix. Since

$$\det\left(\mathrm{D}\left(\sqrt{2\Sigma}y + \mu\right)\right) = \det\left(\sqrt{2\Sigma}\right) = \sqrt{2^n \det(\Sigma)}, \tag{3.124}$$

we get

$$\begin{aligned} I &= \int_{\mathbb{R}^n} f\left(\sqrt{2\Sigma}y + \mu\right) \frac{\sqrt{2^n \det(\Sigma)}}{\sqrt{(2\pi)^n \det(\Sigma)}} \exp\left(-\frac{1}{2}\left(\sqrt{2\Sigma}y\right)^\intercal \Sigma^{-1}\left(\sqrt{2\Sigma}y\right)\right) \mathrm{d}y \\ &= \frac{1}{\sqrt{\pi^n}} \int_{\mathbb{R}^n} f\left(\sqrt{2\Sigma}y + \mu\right) \exp\left(-y^\intercal y\right) \mathrm{d}y. \end{aligned} \tag{3.125}$$

Combining this result with Equation 3.121, we obtain the spherical-radial cubature rule for arbitrary mean and covariance μ and Σ

$$\int_{\mathbb{R}^n} f(x)\mathcal{N}(\mu, \Sigma)\mathrm{d}x \approx \frac{1}{2n} \sum_{i=1}^{2n} f\left(\xi_i\right), \tag{3.126}$$

where ξ_i is the i-th cubature point defined as

$$\xi_i = \sqrt{n\Sigma}[1]_i + \mu. \tag{3.127}$$

▶ Note 3.21. In practice, we can obtain the matrix $\sqrt{\Sigma}$ using the Cholesky decomposition [16]. ◀

3.3 The cubature Kalman filter algorithm

In this section, we present the CKF algorithm derived in the previous section to ensure clarity.

3.3.1 Initialization

1. Initialize the *a posteriori* state estimate with the most accurate estimation of the initial state, ideally

$$\hat{x}_0^+ = \mathbb{E}\left[x_0\right]. \tag{3.128}$$

2. Initialize the *a posteriori* covariance using a covariance matrix that reflects the level of confidence in the initial estimate. If we are sure that our initial estimate is ideal, we set

$$P_0^+ = \mathbb{E}\left[(x_0 - \hat{x}_0^+)(x_0 - \hat{x}_0^+)^\intercal\right]. \tag{3.129}$$

3.3.2 The time update

1. Factorize the *a posteriori* covariance to obtain the matrix S_{k-1}^+ as follows

$$P_{k-1}^+ = S_{k-1}^+ \left(S_{k-1}^+\right)^\intercal. \tag{3.130}$$

Let us represent the i-th column of the matrix S_{k-1}^+ by s_i.

2. Calculate cubature points

$$\xi_i = \begin{cases} \hat{x}_{k-1}^+ + \sqrt{n}s_i, & i \in \{1, \ldots, n\}, \\ \hat{x}_{k-1}^+ - \sqrt{n}s_{i-n}, & i \in \{n+1, \ldots, 2n\}, \end{cases} \tag{3.131}$$

for n representing the dimension of the state vector.

3. Propagate the cubature points through the system function

$$\tilde{x}_i = f(\xi_i, u_{k-1}), \quad i \in \{1, \ldots, 2n\}. \tag{3.132}$$

4. Evaluate the *a priori* state estimate

$$\hat{x}_k^- = \frac{1}{2n}\sum_{i=1}^{2n} \tilde{x}_i. \tag{3.133}$$

5. Evaluate the *a priori* covariance

$$P_k^- = \frac{1}{2n}\sum_{i=1}^{2n} (\tilde{x}_i \tilde{x}_i^\intercal) - \hat{x}_k^- \left(\hat{x}_k^-\right)^\intercal + Q_{k-1}. \tag{3.134}$$

3.3.3 The measurement update

1. Factorize the *a priori* covariance to obtain the matrix S_k^- as follows

$$P_{k-1}^- = S_k^- \left(S_k^-\right)^\intercal. \tag{3.135}$$

Let us represent the i-th column of the matrix S_k^- by s_i.

2. Calculate cubature points

$$\xi_i = \begin{cases} \hat{x}_k^- + \sqrt{n}s_i, & i \in \{1, \ldots, n\}, \\ \hat{x}_k^- - \sqrt{n}s_{i-n}, & i \in \{n+1, \ldots, 2n\}. \end{cases} \tag{3.136}$$

3. Propagate cubature points through the measurement function

$$\tilde{y}_i = h(\xi_i), \quad i \in \{1, \ldots, 2n\}. \tag{3.137}$$

4. Evaluate the measurement estimate

$$\hat{y}_k = \frac{1}{2n} \sum_{i=1}^{2n} \tilde{y}_i. \tag{3.138}$$

5. Evaluate the cross-covariance matrix

$$P_{xy,k} = \frac{1}{2n} \sum_{i=1}^{2n} (\xi_i \tilde{y}_i^\intercal) - \hat{x}_k^- \hat{y}_k^\intercal. \tag{3.139}$$

6. Evaluate the innovation covariance matrix

$$P_{y,k} = \frac{1}{2n} \sum_{i=1}^{2n} (\tilde{y}_i \tilde{y}_i^\intercal) - \hat{y}_k \hat{y}_k^\intercal + R_k. \tag{3.140}$$

7. Evaluate the Kalman gain

$$K_k = P_{xy,k} P_{y,k}^{-1}. \tag{3.141}$$

8. Evaluate the *a posteriori* state estimate

$$\hat{x}_k^+ = \hat{x}_k^- + K_k(y_k - \hat{y}_k). \tag{3.142}$$

9. Evaluate the *a posteriori* covariance

$$P_k^+ = P_k^- - K_k P_{y,k} K_k^\intercal. \tag{3.143}$$

3.4 Properties of the cubature Kalman filter

The main, obvious, property of the CKF, which was the motivation for its very creation, is that the CKF is a nonlinear filter. That is, the CKF can handle a state-space model that contains nonlinear functions, unlike the Kalman filter which handles only a linear state-space model.

Another property that makes the CKF stand out among other nonlinear filters is that the CKF eases the curse of dimensionality [7]. This is because the cubature rule entails only $2n$ cubature points, where n is the dimension of the state vector. This has the additional positive consequence that the computational complexity in terms of the number of evaluations of the state-space model functions increases linearly with the dimension.

The CKF algorithm can be suitably modified to be more numerically stable. The modification is that in each update the square root of the covariance is

propagated instead of the covariance itself. Thus, there is no need to compute a square root of a matrix, which is a numerically very sensitive operation. This solution is called the square-root cubature Kalman filter [7]. This offers other benefits, in particular preservation of symmetry and positive definiteness of the covariance and doubled-order precision [17].

It is important to note that the filter was derived for systems with Gaussian noise. Thus, if the noise is not Gaussian, the filter may not retain its positive properties. However, the reality is often that the Gaussian distribution is a satisfactory approximation of real-world processes. If the Gaussianity assumption is satisfied, the CKF, especially its square-root version, is robust in terms of divergence compared to other nonlinear filters [7].

3.5 Comparison with other nonlinear filters

Additional unique features of the CKF will be highlighted in comparison to other nonlinear filtering methods. We will compare the CKF with the two widely used Kalman-based nonlinear filters, the unscented Kalman filter and the extended Kalman filter.

3.5.1 Comparison with the unscented Kalman filter

The unscented Kalman filter [6], [18] is currently one of the most widely used nonlinear state estimation methods. However, the fact is that the UKF and the CKF are very similar. As we will see later, in one particular case, their algorithms are even identical.

The difference between the CKF and the UKF lies in the choice of cubature points, in the case of the UKF called sigma points. The rest of the algorithm remains the same. The sigma points in the UKF are parameterized by κ. The UKF also introduces an additional sigma point in the center, which has a different weight than the rest of the sigma points. The UKF then selects the $(2n+1)$ sigma points so that their sample mean and covariance are the same as the mean and covariance of the distribution of the state vector estimate. The aim is to approximate the entire distribution using only a few points, as it would be infeasible to propagate the entire distribution in a nonlinear function. Specifically, it selects the following sigma points:

$$\xi_i = \begin{cases} \hat{x}_k, & i = 0, \\ \hat{x}_k + \left(\sqrt{(n+\kappa)P_k}\right)_{\bullet i}, & i \in \{1, \dots, n\}, \\ \hat{x}_k - \left(\sqrt{(n+\kappa)P_k}\right)_{\bullet (i-n)}, & i \in \{n+1, \dots, 2n\}, \end{cases} \tag{3.144}$$

where the subscript $\bullet i$ denotes the i-th column of a matrix. The UKF then

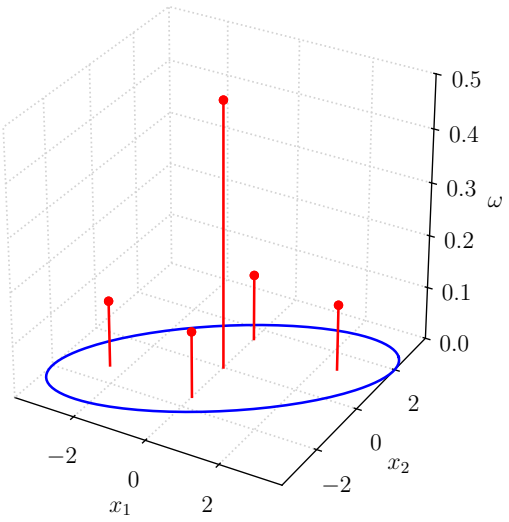

Figure 3.9 Sigma point set with weights ω for $\kappa = 2$ in two-dimensional space, highlighting the 3-sigma P_k covariance ellipse.

selects the corresponding weights

$$\omega_i = \begin{cases} \frac{\kappa}{n+\kappa}, & i = 0, \\ \frac{1}{2(n+\kappa)}, & i \in \{1, \dots, 2n\}. \end{cases} \tag{3.145}$$

The parameter κ controls the behavior at higher orders. For the Gaussian distribution, this parameter is typically set to $\kappa = 3 - n$ to capture the kurtosis [18]. In Figures 3.9 and 3.10, the selection of points and weights by the CKF and the UKF is demonstrated.

An interesting scenario occurs when we set $\kappa = 0$. In this case, the center point vanishes due to $\omega_0 = 0$. Also, the weights for the rest of the sigma points are equal to $\frac{1}{2n}$. Thus, for $\kappa = 0$, the CKF and the UKF result in the exact same algorithm.

Although the algorithms of the CKF and the UKF are very similar, these filters are mainly very different in their philosophy and approach to derivation. The UKF focuses on selecting the sigma points to have the same sample mean and covariance as the mean and covariance of the distribution of the state vector estimate so that their propagation approximates the propagation of the entire distribution. In contrast, the cubature points in the CKF are implicitly given by the third-order spherical-radial cubature rule. The CKF assumes the states and measurements to be Gaussian instead of a more general symmetric density. Furthermore, the CKF focuses on computing the first two-order moments of the measurement vector exactly [7].

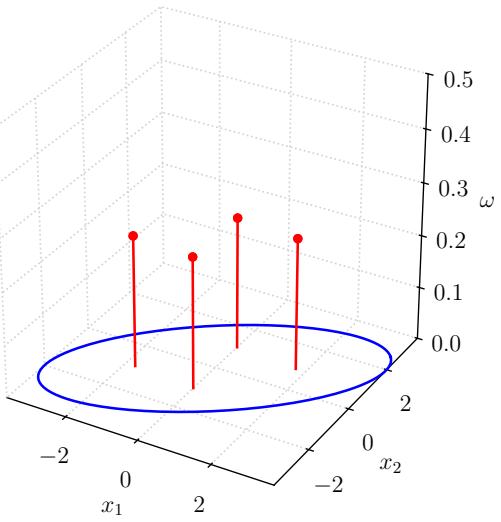

Figure 3.10 Cubature point set with weights ω in two-dimensional space, highlighting the 3-sigma P_k covariance ellipse.

The freedom in the choice of the κ parameter also brings disadvantages. Due to the need to account for different weights for different sigma points, rounding errors occur, leading to increased numerical inaccuracy. For arbitrary κ there is no square-root solution available for the UKF. Although there exists a pseudo square-root version of the UKF [19], this solution may produce a non-positive definite matrix, due to which the algorithm may end up with an error [7].

3.5.2 Comparison with the extended Kalman filter

The extended Kalman filter [1] takes a completely different approach than the CKF or the UKF. It relies on linearizing the system and measurement functions to obtain the matrices for later use in the classical linear Kalman filter. Namely, it uses a first-order Taylor series expansion of the state-space model functions around the current estimate. The first-order Taylor series expansion is illustrated on e^x in Figure 3.11.

In the time update [1], the system function is approximated by linearization around the *a posteriori* estimate $\hat{x}_{k-1}^{+}$ as

$$f(x_{k-1}, u_{k-1}) \approx f\left(\hat{x}_{k-1}^{+}, u_{k-1}\right) + F_{k-1}\left(x_{k-1} - \hat{x}_{k-1}^{+}\right) \tag{3.146}$$

$$= F_{k-1}x_{k-1} + f\left(\hat{x}_{k-1}^{+}, u_{k-1}\right) - F_{k-1}\hat{x}_{k-1}^{+} \tag{3.147}$$

$$= F_{k-1}x_{k-1} + \tilde{u}_{k-1}, \tag{3.148}$$

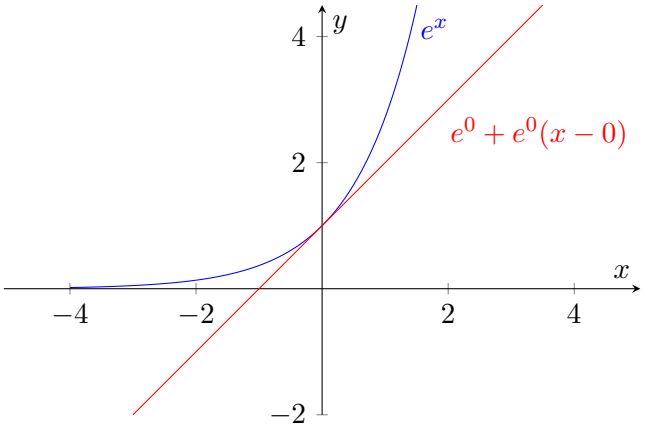

Figure 3.11 Linearization of e^x using the first-order Taylor series expansion around $x = 0$.

where

$$F_k = \left.\frac{\partial f}{\partial x}\right|_{\hat{x}_{k-1}^+}, \tag{3.149}$$

$$\tilde{u}_{k-1} = f\left(\hat{x}_{k-1}^+, u_{k-1}\right) - F_{k-1}\hat{x}_{k-1}^+. \tag{3.150}$$

Thus, we obtain a linear function with F_k being the system matrix and $\tilde{u}_{k-1}$ being the new known control input. We can then perform the Kalman filter time update as usual.

For the *a priori* estimate, it holds

$$\begin{aligned}
\hat{x}_k^- &= F_{k-1}\hat{x}_{k-1}^+ + \tilde{u}_{k-1} && (3.151)\\
&= F_{k-1}\hat{x}_{k-1}^+ + f\left(\hat{x}_{k-1}^+, u_{k-1}\right) - F_{k-1}\hat{x}_{k-1}^+ && (3.152)\\
&= f\left(\hat{x}_{k-1}^+, u_{k-1}\right). && (3.153)
\end{aligned}$$

Therefore, the estimate is updated using the nonlinear system function and the linearization serves only to form the *a priori* covariance.

The measurement update [1] is similar. We approximate the measurement function by linearizing it around the *a priori* estimate $\hat{x}_k^-$ as

$$\begin{aligned}
h(x_{k-1}) &\approx h\left(\hat{x}_k^-\right) + H_k\left(x_k - \hat{x}_k^-\right) && (3.154)\\
&= H_k x_k + h\left(\hat{x}_k^-\right) - H_k\hat{x}_k^- && (3.155)\\
&= H_k x_k + z_k && (3.156)
\end{aligned}$$

where

$$H_k = \mathrm{D}h\big|_{\hat{x}_k^-}, \tag{3.157}$$

$$z_k = h\left(\hat{x}_k^-\right) - H_k\hat{x}_k^-. \tag{3.158}$$

Thus, we get a linear function with the measurement matrix H_k and the input z_k. We can then proceed with the Kalman filter measurement update.

Similarly to the time update, for the measurement prediction, it holds

$$\begin{aligned} \hat{y}_k &= H_k \hat{x}_k^- + z_k && (3.159)\\ &= H_k \hat{x}_k^- + h\left(\hat{x}_k^-\right) - H_k \hat{x}_k^- && (3.160)\\ &= h\left(\hat{x}_k^-\right). && (3.161) \end{aligned}$$

Therefore, the linearization again only affects the covariance.

The major weakness of the EKF is that it requires derivatives of the state-space model functions to operate. However, some functions are difficult to differentiate, or even not differentiable at all. In this case, the CKF is a suitable option because it works without derivatives. This has another advantage that it suffices to pass the state-space model functions to the filtering algorithm making the CKF a prepackaged solution.

The strength of the EKF, in comparison to the CKF, is its computational efficiency. The EKF essentially boils down to the KF with only a few additional function evaluations, whereas the CKF requires $4n$ function evaluations and 2 Cholesky decompositions per iteration. In addition, the EKF is more straightforward to implement assuming that the Jacobian matrices are provided. Nevertheless, since the function approximations are only linear, the EKF may produce unreliable estimates for higher-nonlinear systems. Consequently, it is more prone to divergence [1].

Chapter 4
Examples

In this chapter, we will present the use of the CKF through several examples. The examples are chosen to demonstrate the properties of the CKF. They also serve as a comparison to other filtering methods.

4.1 Example 1: Linear state-space model

Although the CKF is a nonlinear filter, it can work with arbitrary model functions, including linear functions. Therefore, the CKF can be used with a linear state-space model. For the linear state-space model matrices F and H, we define the functions of a nonlinear state-space model

$$f(x_k) = Fx_k, \tag{4.1}$$
$$h(x_k) = Hx_k. \tag{4.2}$$

Our desire now is to use the CKF with a linear state-space model and compare its performance with the KF.

We will simulate the tracking of a target moving in one-dimensional space using the constant acceleration model (CAM) from Example 1.1 with the state vector

$$x_k = \begin{pmatrix} p_k \\ r_k \\ a_k \end{pmatrix}, \tag{4.3}$$

where p_k represents the position, r_k represents the velocity, and a_k represents the acceleration of the target. The CAM consists of the system matrix

$$F = \begin{pmatrix} 1 & \Delta_t & \frac{1}{2}\Delta_t^2 \\ 0 & 1 & \Delta_t \\ 0 & 0 & 1 \end{pmatrix}, \tag{4.4}$$

and the output matrix

$$H = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}, \tag{4.5}$$

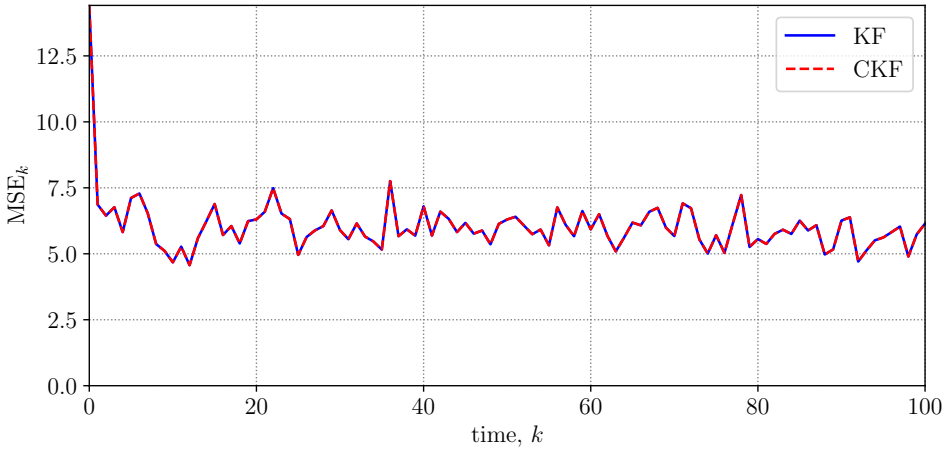

Figure 4.1 Comparison of MSE at each time step between the KF and the CKF with a linear state-space model.

meaning that we only measure the position of the target.

In our case, we consider $\Delta_t = 1$. We set the process noise covariance

$$Q = \text{diag}\left((0.3, 0.5, 1)^\intercal\right) \tag{4.6}$$

and the measurement noise covariance

$$R = I_2. \tag{4.7}$$

We set the initial state

$$x_0 = \big(0, 1, 0\big)^\intercal \tag{4.8}$$

and the initial covariance

$$P_0 = \text{diag}\left((9, 4, 1)^\intercal\right). \tag{4.9}$$

We will perform 100 independent simulation runs. For each run, a true target state history and the measurements will be simulated for 100 time steps using the CAM with the set parameters and Gaussian noise. The CKF and KF will then be used to filter the measurements, i.e., to obtain an estimate of the true state at each time step k.

We will measure the performance of the filters using the MSE estimate at each time k

$$\text{MSE}_k = \frac{1}{N}\sum_{i=1}^{N} \|x_k^{(i)} - \hat{x}_k^{(i)}\|^2, \tag{4.10}$$

where N is the number of simulation runs, in our case $N = 100$.

In Figure 4.1, we can see that the CKF performed with the same MSE as the KF. Given that the KF is an optimal filter in terms of minimizing the MSE

if the model is linear with Gaussian noise, the CKF performed optimally as well. This is in fact an expected result. Since the cubature rule is exact up to third-degree monomials, it propagates the first two-order moments exactly, if the model is linear.

On the other hand, the CKF requires more computing resources to achieve the same result as the KF. Notably, it executes the Cholesky decomposition twice in each iteration, which might have an additional negative impact on numerical stability. Moreover, it computes the required vectors and matrices using redundant operations. For instance, when computing the *a priori* estimate, the CKF first evaluates the $2n$ symmetrically distributed cubature points by propagating the *a posteriori* estimate through the linear model $2n$-times, and then it takes their mean, instead of propagating the *a posteriori* estimate once, as the KF does.

In conclusion, it is generally better to use the KF instead of the CKF when the state-space system is linear. On the other hand, the KF cannot handle nonlinear systems at all. Therefore, the CKF provides a more flexible solution at a potentially higher cost for certain models.

4.2 Example 2: Nonlinear filters comparison

In this example, we will compare the performance of the three nonlinear filtering algorithms presented in this thesis: the CKF, the UKF, and the EKF. The comparison will be conducted by simulating the tracking of an object with a constant velocity model (CVM) [2] describing its movement. Although the CVM is a linear model, the nonlinearity will arise from the measurements being in the polar coordinates. This may happen, for example, if the measurements are obtained using a radar. Thus, the measurement equation will be nonlinear in this scenario.

The CVM keeps the state vector

$$x_k = \begin{pmatrix} p_{1,k} \\ p_{2,k} \\ r_{1,k} \\ r_{2,k} \end{pmatrix}, \tag{4.11}$$

where $x_{1,k}$ and $x_{2,k}$ denote the location of the object in Cartesian coordinates with $v_{1,k}$ and $v_{2,k}$ denoting the respectful velocities at time k. The CVM uses the system function

$$f(x_k) = Fx_k, \tag{4.12}$$

where

$$F = \begin{pmatrix} 1 & 0 & \Delta_t & 0 \\ 0 & 1 & 0 & \Delta_t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{4.13}$$

where Δ_t is a constant time interval between time steps. Throughout this example, we consider $\Delta_t = 1$. The Jacobian of the system function f is the constant matrix F, which we will need for the EKF.

The measurement function then transforms the Cartesian coordinates into polar coordinates, which express the location using a distance and an angle. That is,

$$h(x_k) = \begin{pmatrix} \sqrt{p_{1,k}^2 + p_{2,k}^2} \\ \operatorname{atan2}(p_{2,k}, p_{1,k}) \end{pmatrix}, \tag{4.14}$$

where the function $\operatorname{atan2}(y, x)$ is an extension of the function $\operatorname{arctg}\left(\frac{y}{x}\right)$ for angles in the range of the whole circle. This is an equivalent method for computing the angle to the Equation 3.5. At the points where this function is differentiable, the function $\operatorname{atan2}(y, x)$ is equal to the function $\operatorname{arctg}\left(\frac{y}{x}\right)$, with the exception of a constant, and thus has an equal derivative. Hence, we obtain the Jacobian

$$H(x_k) = \begin{pmatrix} \frac{p_{1,k}}{\sqrt{p_{1,k}^2+p_{2,k}^2}} & \frac{p_{2,k}}{\sqrt{p_{1,k}^2+p_{2,k}^2}} & 0 & 0 \\ -\frac{p_{2,k}}{p_{1,k}^2+p_{2,k}^2} & \frac{p_{1,k}}{p_{1,k}^2+p_{2,k}^2} & 0 & 0 \end{pmatrix}. \tag{4.15}$$

We set the system noise covariance matrix to

$$Q = \sigma_a^2 \begin{pmatrix} \frac{\Delta_t^3}{3} & 0 & \frac{\Delta_t^2}{2} & 0 \\ 0 & \frac{\Delta_t^3}{3} & 0 & \frac{\Delta_t^2}{2} \\ \frac{\Delta_t^2}{2} & 0 & \Delta_t & 0 \\ 0 & \frac{\Delta_t^2}{2} & 0 & \Delta_t \end{pmatrix}, \tag{4.16}$$

where

$$\sigma_a^2 = 4^2. \tag{4.17}$$

Then we set the measurement noise covariance matrix

$$R = \operatorname{diag}\left(\sigma_d^2, \sigma_\varphi^2\right), \tag{4.18}$$

for

$$\sigma_d^2 = 2^2, \quad \sigma_\varphi^2 = 0.1^2, \tag{4.19}$$

where σ_d^2 is the variance of the measured distance and σ_φ^2 is the variance of the measured angle. This is why we use the transition to polar coordinates as the measurement function instead of transforming the coordinates in advance. If we did the latter, we wouldn't be able to describe the noise correctly since the angle noise is more prominent the further the object is from the radar.

We initialize the simulation with

$$x_0 = (0, 0, 1, 1)^\intercal, \tag{4.20}$$

$$P_0^+ = I_4, \tag{4.21}$$

$$\hat{x}_0^+ \sim \mathcal{N}\left(x_0, P_0^+\right). \tag{4.22}$$

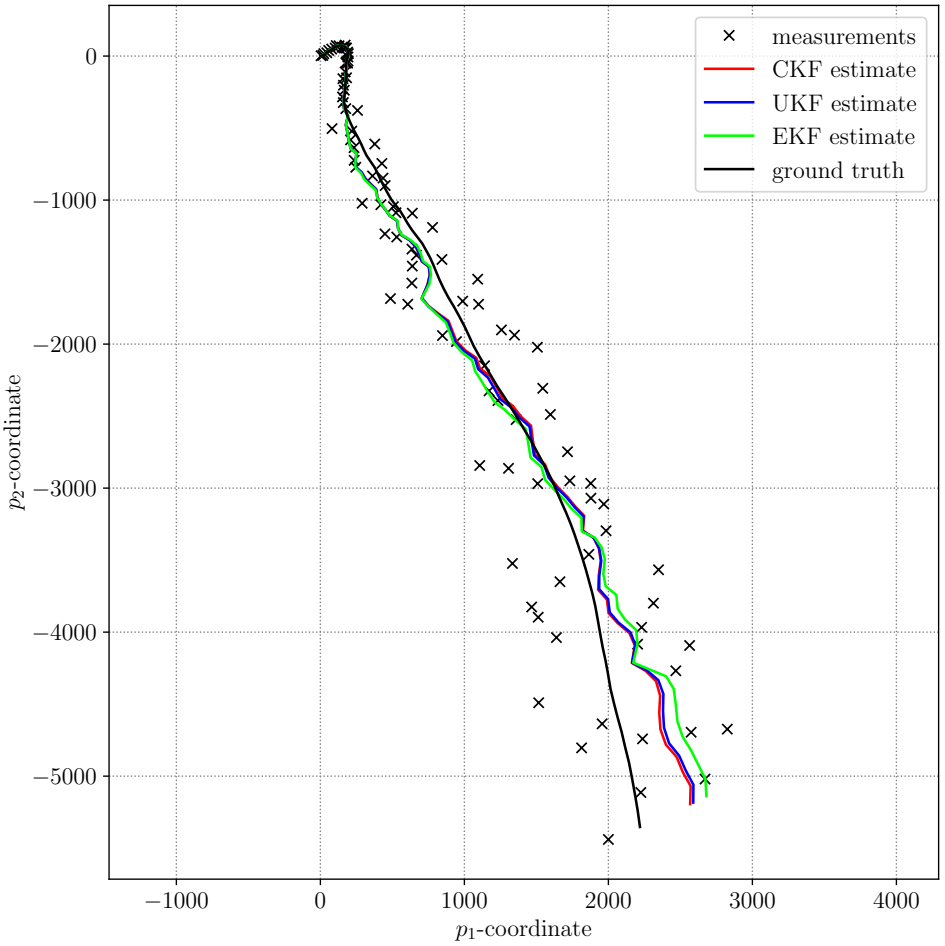

Figure 4.2 Comparison of trajectory estimates with the ground truth and measurements.

We will simulate the ground truth and the measurements using the CVM with Gaussian noise from the initial state x_0. Then we will use the CKF, the UKF, and the EKF to filter the measurements and compare their errors. We will initialize the UKF with the established [18] κ value of $3 - n$, that is $\kappa = 1$. Recall that for $\kappa = 0$, we get the CKF. Therefore, comparing the UKF and the CKF is equivalent to comparing two different values of the UKF parameter initialization.

In Figure 4.2, we can see how the filters estimated the trajectory in a single simulation instance. The trajectories are presented in the Cartesian coordinate system. It is also visible that the measurements are noisier further from the center due to the angle noise.

In Figures 4.3 and 4.4, we can see a comparison of the errors, which are calculated as a distance from the true state at time k. We split the errors

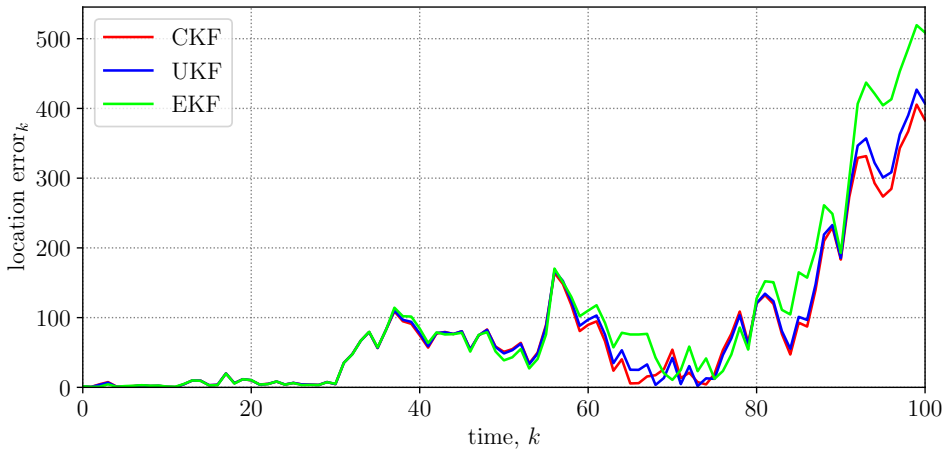

Figure 4.3 Error of the estimated location over time.

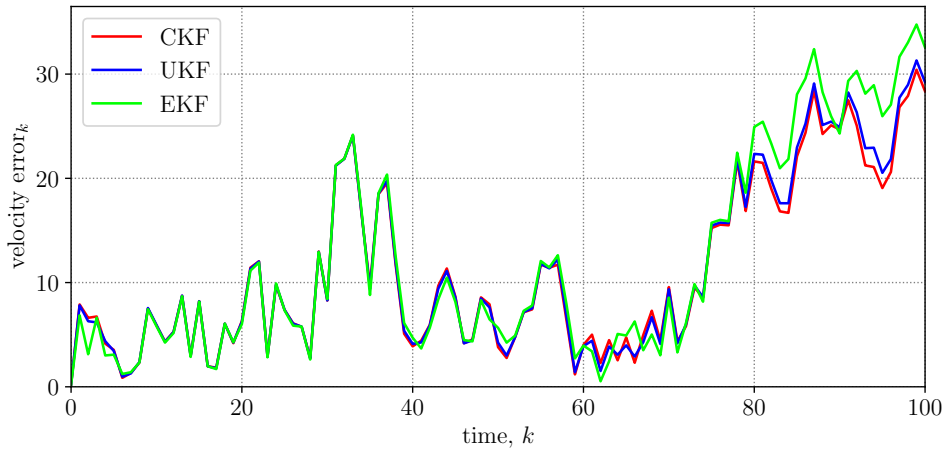

Figure 4.4 Error of the estimated velocity over time.

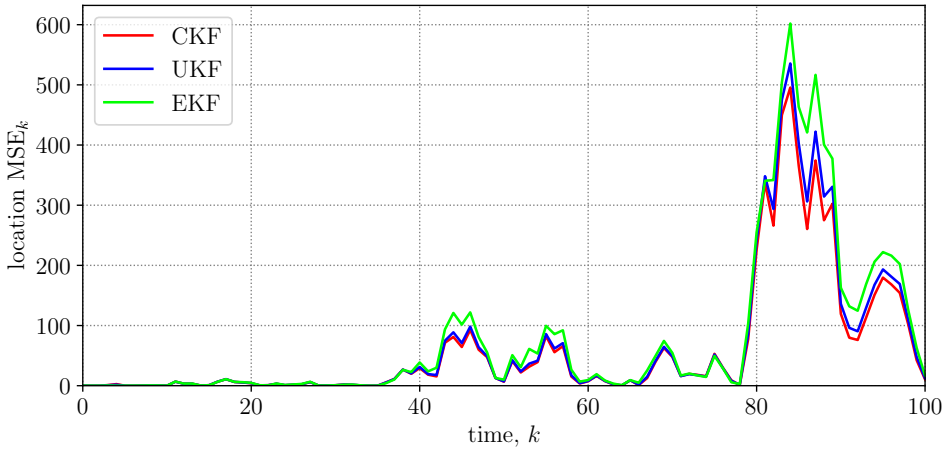

Figure 4.5 MSE of the estimated location over time.

into location and velocity errors due to their different magnitude. If we are interested in knowing the location only, we might dismiss the velocity error since it is just the inner state of the filter. However, we might be interested in knowing the estimate of the velocity as well.

In this particular instance, the CKF and the UKF performed a little better than the EKF. The CKF provided slightly more accurate estimates than the UKF. However, this is just a result of a single simulation run. To compare the filters in a more consistent manner, we perform 100 independent simulation runs to calculate their MSE estimate.

In Figures 4.5 and 4.6, we can observe the estimated MSE of the location and velocity estimates over time. In both cases, the results indicate that the CKF yielded the most favorable results, closely followed by the UKF, which demonstrated comparable performance, and the EKF, showing slightly inferior accuracy. However, it is crucial to note that these findings are obtained from a single scenario. The results might significantly differ in other scenarios, such as a different model or initial values.

4.3 Example 3: Localization of an unmanned aerial vehicle

This final example will serve to illustrate the application of cubature Kalman filtering in a real-world scenario. The objective is to identify the precise location of a flying unmanned aerial vehicle (UAV), commonly referred to as a drone. The UAV is equipped with a GPS receiver that provides measurements every 100 milliseconds. The CKF will be used to filter the noisy measurements, i.e., to produce more accurate estimates of the exact location.

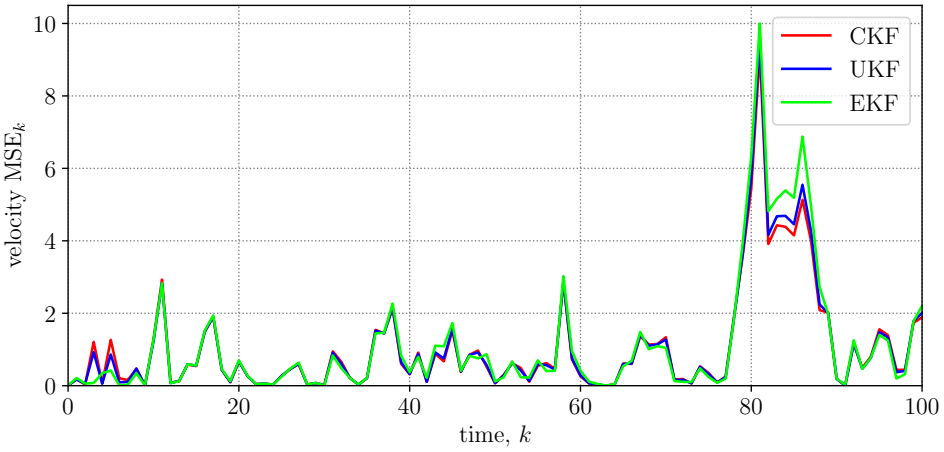

Figure 4.6 MSE of the estimated velocity over time.

In contrast to simulations, real-world scenarios present unique challenges. The fundamental problem is that we are limited in our knowledge of the dynamic system we are observing. That is, we do not possess an exact state-space model that describes the behavior of the system. In addition, once we select the model, we do not have access to the noise covariance matrices, so we have to estimate and tune them. However, it is challenging to tune the parameters because we do not know the ground truth and, therefore, cannot evaluate the filtered estimates with MSE.

In our case, we have prior information that the UAV is moving in circular motions over a small area, a few meters in size. The height of flight is known and constant. This leads us to use the coordinated turn model (CTM) presented in Example 1.2. The CTM captures the turn rate of the UAV in its state. The state is thus represented by the vector

$$x_k = \begin{pmatrix} c_{1,k} \\ c_{2,k} \\ r_{1,k} \\ r_{2,k} \\ \omega_k \end{pmatrix}, \tag{4.23}$$

where $c_{1,k}$ and $c_{2,k}$ represent the location coordinates, $r_{1,k}$ and $r_{2,k}$ represent the velocities in corresponding directions, and ω_k represents the turn rate.

The CTM consists of the nonlinear system function

$$f(x_k) = \begin{pmatrix} 1 & 0 & \frac{\sin(\Delta_t \omega_k)}{\omega_k} & \frac{-1+\cos(\Delta_t \omega_k)}{\omega_k} & 0 \\ 0 & 1 & \frac{1-\cos(\Delta_t \omega_k)}{\omega_k} & \frac{\sin(\Delta_t \omega_k)}{\omega_k} & 0 \\ 0 & 0 & \cos(\Delta_t \omega_k) & -\sin(\Delta_t \omega_k) & 0 \\ 0 & 0 & \sin(\Delta_t \omega_k) & \cos(\Delta_t \omega_k) & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} x_k, \tag{4.24}$$

where $\Delta_t = 0.1$ since the time interval between measurements is 100 ms and we operate in seconds.

The measurement function then becomes

$$h(x_k) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix} x_k, \tag{4.25}$$

as the measurements consist only of the GPS coordinates.

▶ Note 4.1. The GPS measurements consist of latitude and longitude in degrees. However, for ease of manipulation, we convert these values to coordinates in meters relative to the first measured location using the universal transverse Mercator projection. Given the limited spatial extent of the flight area, this transformation can be approximated as predominantly linear. As a result, the measurement noise, which operates within the latitude and longitude dimensions, will scale proportionally. ◀

An appropriate [2] system noise covariance matrix for the CTM has the form

$$Q = \begin{pmatrix} \frac{\Delta_t^3}{3}\sigma_a^2 & 0 & \frac{\Delta_t^2}{2}\sigma_a^2 & 0 & 0 \\ 0 & \frac{\Delta_t^3}{3}\sigma_a^2 & 0 & \frac{\Delta_t^2}{2}\sigma_a^2 & 0 \\ \frac{\Delta_t^2}{2}\sigma_a^2 & 0 & \Delta_t\sigma_a^2 & 0 & 0 \\ 0 & \frac{\Delta_t^2}{2}\sigma_a^2 & 0 & \Delta_t\sigma_a & 0 \\ 0 & 0 & 0 & 0 & \Delta_t^2\sigma_\omega^2 \end{pmatrix}, \tag{4.26}$$

where σ_a^2 and σ_ω^2 are parameters to determine. Thus, it is sufficient to specify two variables to obtain the entire covariance matrix. The σ_a represents the standard deviation of the acceleration along a coordinate and σ_ω represents the standard deviation of the turn rate. It is reasonable to set these values as high as is necessary to account for the most extreme maneuvers that are likely to occur [2]. In our example, we expect the UAV to change its velocity by at most 0.5 m/s per second and its turn rate by at most 0.2 units per second. Therefore, we set the parameters

$$\sigma_a^2 = 0.5^2, \tag{4.27}$$

$$\sigma_\omega^2 = 0.2^2. \tag{4.28}$$

The measurement noise covariance matrix will have the form

$$R = \sigma_y^2 I_2, \tag{4.29}$$

where σ_y^2, being the variance of the measurement along one coordinate, is a parameter to determine. We assume that the standard deviation σ_y is 20 cm, so we set

$$\sigma_y^2 = 0.2^2. \tag{4.30}$$

The remaining task is to set the initial state estimate and covariance. A common approach is to use the first measurement as the initial estimate. However, the measurement consists only of the location, therefore we need to determine the initial velocities and the initial turn rate separately. In our case, we have no prior information about the initial velocities and turn rate, so we set them to zero. Thus, the initial state estimate will take the form

$$\hat{x}_0^+ = \begin{pmatrix} y_{0,0} \\ y_{0,1} \\ 0 \\ 0 \\ 0 \end{pmatrix}, \tag{4.31}$$

where y_0 is the first measurement vector.

▶ Note 4.2. The CTM is not defined for $\omega = 0$. However, it can be shown that as ω approaches 0, the CTM converges to the constant velocity model [2], which we can switch to in this particular case. ◀

For the initial covariance estimate, we will use a diagonal matrix with variances of each variable. Since we obtained the initial location from the measurement, we can use σ_y^2 for the initial location variances. For the remaining variables, we will attempt to use a higher value than the actual variance, as the covariance will be corrected after a few iterations. Thus, we set the initial covariance

$$P_0^+ = \operatorname{diag}\left(\left(\sigma_y^2, \sigma_y^2, 1, 1, 2\right)^\intercal\right). \tag{4.32}$$

We can observe the filtered trajectory in Figure 4.7. Since we have no knowledge of the ground truth, we cannot evaluate the filtering performance. Without further knowledge, it is not possible to determine whether the estimated trajectory aligns with the measurements to a greater or lesser extent.

The inner state of the filter also enables us to obtain estimates of all the variables within the UAV's state along with their covariance. We can see the estimates of each variable with their 95% confidence intervals in Figures 4.8–4.12.

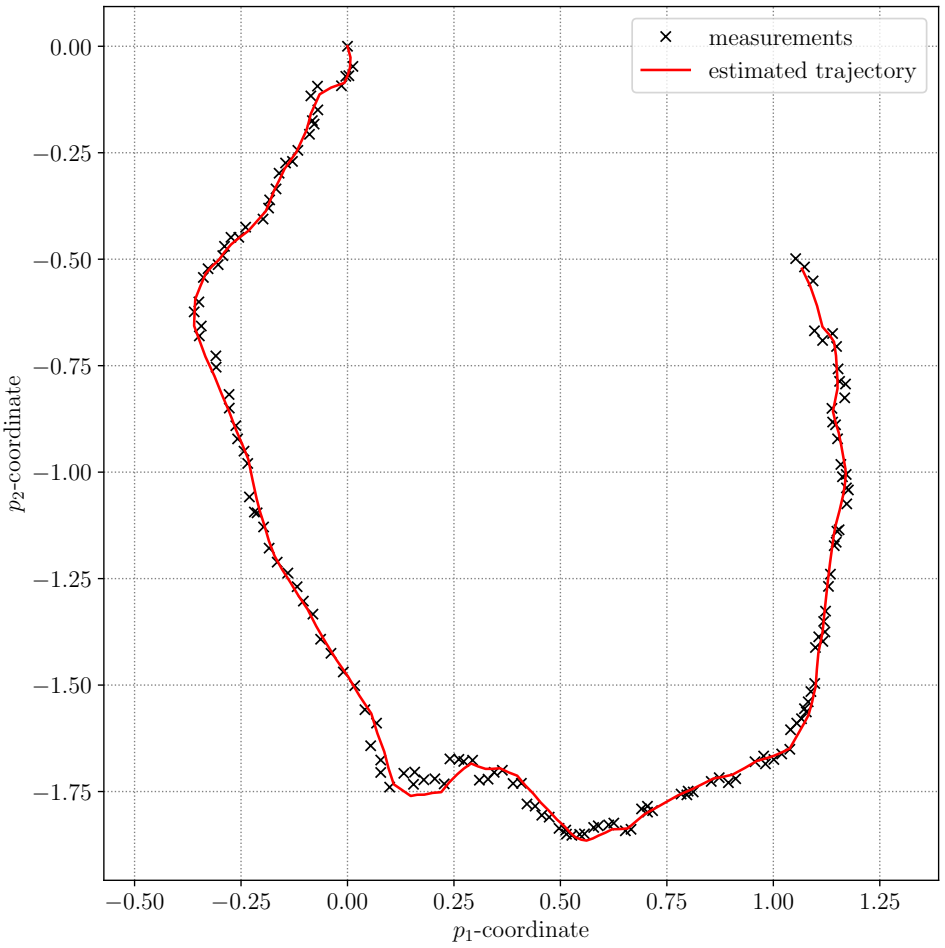

Figure 4.7 Estimated trajectory of the UAV.

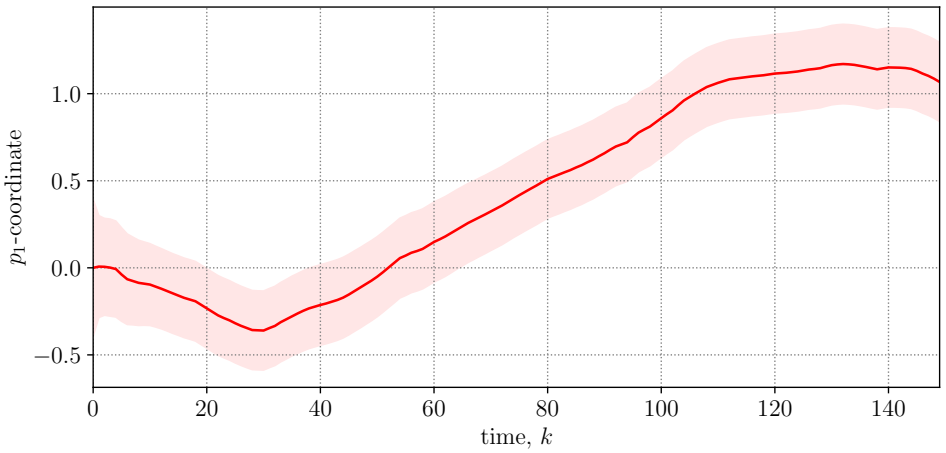

Figure 4.8 The p_1-coordinate estimation with 95% confidence interval.

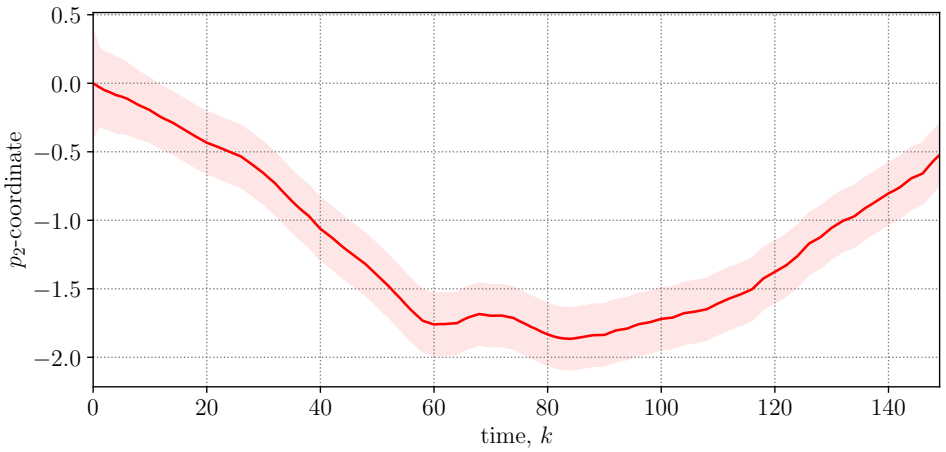

Figure 4.9 The p_2-coordinate estimation with 95% confidence interval.

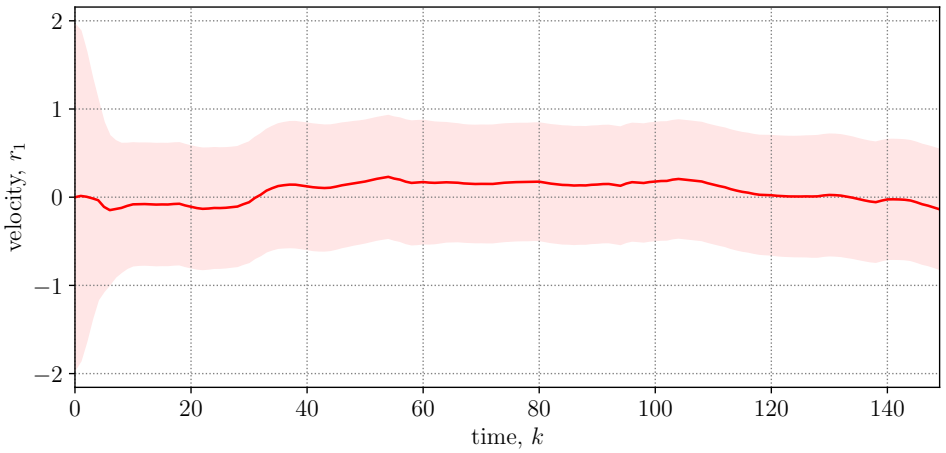

Figure 4.10 The velocity r_1 estimation with 95% confidence interval.

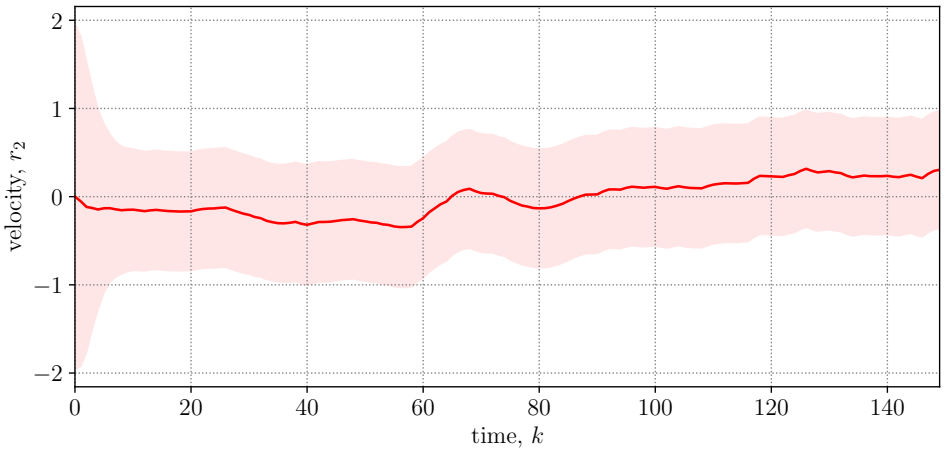

Figure 4.11 The velocity r_2 estimation with 95% confidence interval.

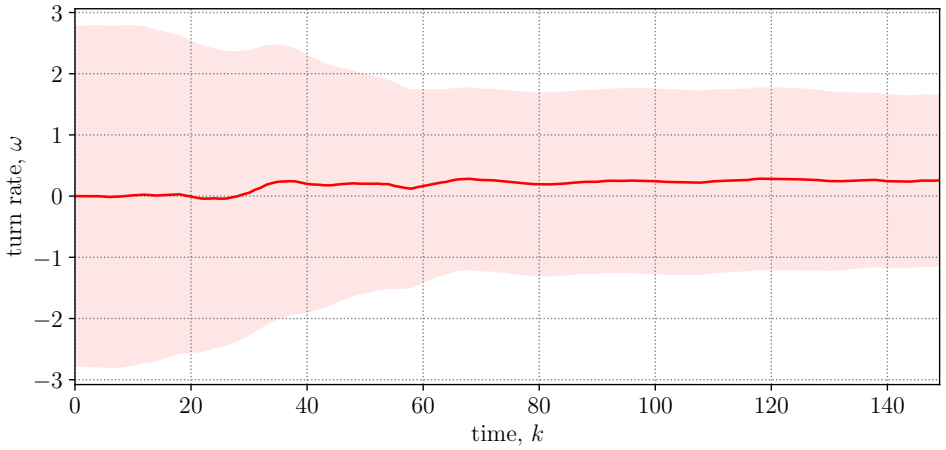

Figure 4.12 The turn rate ω estimation with 95% confidence interval.

Chapter 5
Conclusion

In this final chapter, we will present a summary of the thesis. Furthermore, we will propose potential future work on the topic of cubature Kalman filtering.

5.1 Summary

The thesis first introduces the state-space model in Chapter 1, as this model type is crucial for the state estimation approaches described later. This chapter also provides an explanation of the differences between linear and nonlinear state-space models. In order to illustrate their practical use, the CVM and CTM models are presented.

The Kalman filter, an essential algorithm for state estimation involving a linear state-space model, is derived in Chapter 2. In addition, this chapter discusses the properties and weaknesses of the Kalman filter. Subsequently, an overview of several approaches to Kalman filtering with a nonlinear state-space model is presented.

In Chapter 3, the cubature Kalman filter, an algorithm for nonlinear state estimation that relies on the Kalman filter, is derived. For this purpose, a third-degree cubature rule for numerical integration is constructed. An explanation of all preliminary mathematical principles is presented to make the derivation comprehensive, supported by appropriate examples. Furthermore, the properties of the CKF are discussed. The chapter concludes with a comparison of the CKF with other nonlinear filtering approaches, the UKF and the EKF.

Finally, in Chapter 4, the unique features of the CKF are demonstrated through a number of different examples. The examples also serve as a comparison with other filtering algorithms. It was examined that in a linear case, the CKF provides exactly the same estimates as the traditional Kalman filter, but with an additional computational cost. The second example displays a comparison of the CKF, the UKF, and the EKF on a nonlinear filtering problem

involving measurements in the polar coordinate system. The final example demonstrates the use of the CKF in a real-world scenario using data from an unmanned aerial vehicle.

5.2 Future work

Future work in cubature Kalman filtering could delve into further comparisons with other nonlinear filters. In this thesis, we focused on assessing filters primarily by their estimation accuracy. There remains a gap to explore how these filters perform in terms of computational efficiency and numerical stability. Particularly, the CKF is derived in a way that reduces computational demands and enables implementation that enhances numerical stability. This provides an opportunity to examine its performance alongside other filters in these respects.

While the CKF serves as a versatile state estimation method, future research could explore its application in specialized areas such as target tracking. Extending the CKF to handle cluttered measurements or to track multiple targets would be a valuable direction for further investigation.

Bibliography

[1] D. Simon, *Optimal State Estimation: Kalman, H_∞, and Nonlinear Approaches*. Wiley-Interscience, 2006, ISBN: 978-0471708582.

[2] E. Brekke, *Fundamentals of Sensor Fusion*. Trondheim: Norwegian University of Science and Technology, 2020.

[3] B. Anderson and J. Moore, *Optimal Filtering*. Prentice-Hall, 1979, ISBN: 978-0136381228.

[4] Y. Bar-Shalom, X. Rong Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation*. Wiley-Interscience, 2001, ISBN: 978-0471416555.

[5] R. Fitzgerald, "Divergence of the Kalman filter," *IEEE Transactions on Automatic Control*, vol. 16, no. 6, pp. 736–747, 1971. DOI: 10.1109/TAC.1971.1099836.

[6] S. Julier and J. Uhlmann, "Unscented filtering and nonlinear estimation," *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401–422, 2004. DOI: 10.1109/JPROC.2003.823141.

[7] I. Arasaratnam and S. Haykin, "Cubature Kalman filters," *IEEE Transactions on Automatic Control*, vol. 54, no. 6, pp. 1254–1269, 2009. DOI: 10.1109/TAC.2009.2019800.

[8] T. Shifrin, *Multivariable Mathematics: Linear Algebra, Multivariable Calculus, and Manifolds*. John Wiley and Sons, 2005, ISBN: 978-0471526384.

[9] G. Folland, *Real Analysis: Modern Techniques and Their Applications*, second. John Wiley and Sons, 1999, ISBN: 978-0471317166.

[10] A. H. Stroud, *Approximate calculation of multiple integrals*. Englewood Cliffs, NJ: Prentice Hall, 1971.

[11] F. Olver, *Asymptotics and Special Functions*. Natick, MA: A K Peters, 1997, ISBN: 978-1568810690.

[12] A. H. Stroud and D. Secrest, *Gaussian Quadrature Formulas*. Prentice Hall, 1966.

[13] F. B. Hildebrand, *Introduction to Numerical Analysis*, Second. Dover Publications, 1987, ISBN: 978-0486653631.

[14] A. Ralston and P. Rabinowitz, *A First Course in Numerical Analysis*, Second. Dover Publications, 2001, ISBN: 978-0486414546.

[15] C. Stone, *A Course in Probability and Statistics*. Boston, MA: Duxbury Press, 1996, ISBN: 978-0534233280.

[16] L. N. Trefethen and I. D. Bau, *Numerical Linear Algebra*, Twenty-Fifth Anniversary. Philadelphia, PA: IAM-Society for Industrial and Applied Mathematics, 2022, ISBN: 978-1611977158.

[17] I. Arasaratnam and S. Haykin, "Square-root quadrature Kalman filtering," *IEEE Transactions on Signal Processing*, vol. 56, no. 6, pp. 2589–2593, 2008. DOI: 10.1109/TSP.2007.914964.

[18] S. Julier, J. Uhlmann, and H. F. Durrant-Whyte, "A new method for the nonlinear transformation of means and covariances in filters and estimators," *IEEE Transactions on Automatic Control*, vol. 45, no. 3, pp. 477–482, 2000. DOI: 10.1109/9.847726.

[19] R. Van der Merwe and E. Wan, "The square-root unscented Kalman filter for state and parameter-estimation," in *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)*, vol. 6, 2001, pp. 3461–3464. DOI: 10.1109/ICASSP.2001.940586.

Contents of the attachment

- `readme.md` ..... description of the contents of the attachment
- `filtering` ..... directory containing a filtering library
- `examples` ..... directory containing notebooks with examples
- `text` ..... directory containing the text of the thesis
  - `source` ..... directory containing the source code of the text
  - `thesis.pdf` ..... text of the thesis in PDF format