# KALMAN FILTER

## from the Ground Up

Alex Becker

First Edition

Revision history for the first edition
2023-05-01 First Release
2023-05-08 Minor Typo Updates

**ISBN 978-965-598-439-2**



KALMANFILTER.NET



*First edition, May 2023*

The road to learning by precept is
long, by example short and effective.

*Lucius Annaeus Seneca*
A philosopher of Ancient Rome

# Preface

## Introduction

The Kalman Filter algorithm is a powerful tool for estimating and predicting system states in the presence of uncertainty and is widely used as a fundamental component in applications such as target tracking, navigation, and control.

Although the Kalman Filter is a straightforward concept, many resources on the subject require extensive mathematical background and fail to provide practical examples and illustrations, making it more complicated than necessary.

Back in 2017, I created an online tutorial based on numerical examples and intuitive explanations to make the topic more accessible and understandable. The online tutorial provides introductory material covering the univariate (one-dimensional) and multivariate (multidimensional) Kalman Filters.

Over time, I have received many requests to include more advanced topics, such as non-linear Kalman Filters (Extended Kalman Filter and Unscented Kalman Filter), sensors fusion, and practical implementation guidelines.

Based on the material covered in the online tutorial, I authored the "***Kalman Filter from the Ground Up***" e-book.

The original online tutorial will remain available for free access on the KALMANFILTER.NET website. The e-book "Kalman Filter from the Ground Up" and the source code for the numerical examples can be purchased online.

The book takes the reader from the basics to the advanced topics, covering both theoretical concepts and practical applications. The writing style is intuitive, prioritizing clarity of ideas over mathematical rigor, and it approaches the topic from a philosophical perspective before delving into quantification.

The book contains many illustrative examples, including 14 fully solved numerical examples with performance plots and tables. Examples progress in a paced, logical manner and build upon each other.

The book also includes the necessary mathematical background, providing a solid foundation to expand your knowledge and help to overcome your math fears.

This book is the solution for those facing challenges with the Kalman Filter and the underlying math.

Upon finishing this book, you will be able to design, simulate, and evaluate the performance of the Kalman Filter.

The book includes four parts:

- **Part 1** serves as an introduction to the Kalman Filter, using eight numerical examples, and doesn't require any prior mathematical knowledge. You can call it "The Kalman Filter for Dummies," as it aims to provide an intuitive understanding and develop "Kalman Filter intuition." Upon completing Part 1, readers will thoroughly understand the Kalman Filter's concept and be able to design a univariate (one-dimensional) Kalman Filter.
  **This part is available for free access!**
- **Part 2** presents the Kalman Filter in matrix notation, covering the multivariate (multidimensional) Kalman Filter. It includes a mathematical derivation of Kalman Filter equations, dynamic systems modeling, and two numerical examples. This section is more advanced and requires basic knowledge of Linear Algebra (only matrix operations). Upon completion, readers will understand the math behind the Kalman Filter and be able to design a multivariate Kalman Filter.
  **Most of this part is available for free access!**
- **Part 3** is dedicated to the non-linear Kalman Filter, which is essential for mastering the Kalman Filter since most real-life systems are non-linear. This part begins with a problem statement and describes the differences between linear and non-linear systems. It includes derivation and examples of the most common non-linear filters: the Extended Kalman Filter and the Unscented Kalman Filter.
- **Part 4** contains practical guidelines for Kalman Filter implementation, including sensor fusion, variable measurement uncertainty, treatment of missing measurements, treatment of outliers, and the Kalman Filter design process.

# About the author

My name is Alex Becker. I am from Israel. I am an engineer with over 20 years of experience in the wireless technologies field. As a part of my work, I had to deal with Kalman Filters, mainly for tracking applications.

Constructive criticism is always welcome. I would greatly appreciate your comments and suggestions. Please drop me an email (alex@kalmanfilter.net).

**The numerical examples in this book do not exemplify any modes, methodologies, techniques, or parameters employed by any operational system known to the author**.

# About the Kalman Filter

Many modern systems utilize multiple sensors to estimate hidden (unknown) states through a series of measurements. For instance, a GPS receiver can estimate location and velocity, where location and velocity represent the hidden states, while the differential time of the arrival of signals from satellites serves as measurements.

One of the biggest challenges of tracking and control systems is providing an accurate and precise estimation of the hidden states in the presence of uncertainty. For example, GPS receivers are subject to measurement uncertainties influenced by external factors, such as thermal noise, atmospheric effects, slight changes in satellite positions, receiver clock precision, and more.

The Kalman Filter is a widely used estimation algorithm that plays a critical role in many fields. It is designed to estimate the hidden states of the system, even when the measurements are imprecise and uncertain. Also, the Kalman Filter predicts the future system state based on past estimations.

The filter is named after Rudolf E. Kálmán (May 19, 1930 – July 2, 2016). In 1960, Kálmán published his famous paper describing a recursive solution to the discrete-data linear filtering problem [1].

# Contents

# List of Figures

# List of Tables

# Acronyms

**CPU** Central Processing Unit
**EKF** Extended Kalman Filter
**GNSS** Global Navigation Satellite System
**GPS** Global Positioning System
**INS** Inertial Navigation System
**KF** Kalman Filter
**LiDAR** Light Detection and Ranging
**LKF** Linear Kalman Filter
**LTI** Linear Time Invariant
**NASA** National Aeronautics and Space Administration
**PDF** Probability Density Function
**RMS** Root Mean Square
**RMSE** Root Mean Square Error
**SNR** Signal to Noise Ratio
**SPKF** Sigma-point Kalman Filter
**UAV** Unmanned Air Vehicle
**UKF** Unscented Kalman Filter
**UT** Unscented Transform

# Introduction to Kalman Filter

# 1. The Necessity of Prediction

Before delving into the Kalman Filter explanation, let us first understand the necessity of a tracking and prediction algorithm.

To illustrate this point, let's take the example of a tracking radar.

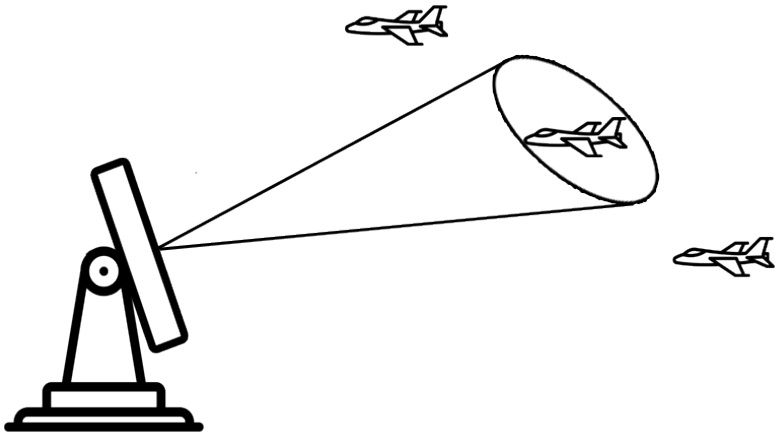

**Figure 1.1:** *Tracking radar.*

Suppose we have a track cycle of 5 seconds. At intervals of 5 seconds, the radar samples the target by directing a dedicated pencil beam.

Once the radar "visits" the target, it proceeds to estimate the current position and velocity of the target. The radar also estimates (or predicts) the target's position at the time of the next track beam.

The future target position can be easily calculated using Newton's motion equations:

$$x = x_0 + v_0 \Delta t + \frac{1}{2} a \Delta t^2 \tag{1.1}$$

Where:

- $x$ is the target position
- $x_0$ is the initial target position
- $v_0$ is the initial target velocity
- $a$ is the target acceleration
- $\Delta t$ is the time interval (5 seconds in our example)

When dealing with three dimensions, Newton's motion equations can be expressed as a system of equations:

$$\begin{cases} x = x_0 + v_{x0}\Delta t + \dfrac{1}{2}\, a_x \Delta t^2 \\ y = y_0 + v_{y0}\Delta t + \dfrac{1}{2}\, a_y \Delta t^2 \\ z = z_0 + v_{z0}\Delta t + \dfrac{1}{2}\, a_z \Delta t^2 \end{cases} \tag{1.2}$$

The set of target parameters $[x, y, z, v_x, v_y, v_z, a_x, a_y, a_z]$ is known as the **System State**. The current state serves as the input for the prediction algorithm, while the algorithm's output is the future state, which includes the target parameters for the subsequent time interval.

The system of equations mentioned above is known as a **Dynamic Model** or **State Space Model**. The dynamic model describes the relationship between the input and output of the system.

Apparently, if the target's current state and dynamic model are known, predicting the target's subsequent state can be easily accomplished.

In reality, the radar measurement is not entirely accurate. It contains random errors or uncertainties that can affect the accuracy of the predicted target state. The magnitude of the errors depends on various factors, such as radar calibration, beam width, and signal-to-noise ratio of the returned echo. The random errors or uncertainties in the radar measurement are known as **Measurement Noise**.

In addition, the target motion is not always aligned with the motion equations due to external factors like wind, air turbulence, and pilot maneuvers. This misalignment between the motion equations and the actual target motion results in an error or uncertainty in the dynamic model, which is called **Process Noise**.

Due to the Measurement Noise and the Process Noise, the estimated target position can be far away from the actual target position. In this case, the radar might send the track beam in the wrong direction and miss the target.

In order to improve the radar's tracking accuracy, it is essential to employ a prediction algorithm that accounts for both process and measurement uncertainty.

The most common tracking and prediction algorithm is the **Kalman Filter**.

# 2. Essential background I

Before we start, I would like to explain several fundamental terms such as variance, standard deviation, normal distribution, estimate, accuracy, precision, mean, expected value, and random variable.

I expect that many readers of this book are familiar with introductory statistics. However, at the beginning of this book, I promised to supply the necessary background that is required to understand how the Kalman Filter works. If you are familiar with this topic, feel free to skip this chapter and jump to chapter 3.

## 2.1 Mean and Expected Value

**Mean** and **Expected Value** are closely related terms. However, there is a difference.

For example, given five coins – two 5-cent coins and three 10-cent coins, we can easily calculate the mean value by averaging the values of the coins.

**Figure 2.1:** *Coins.*

$$V_{mean} = \frac{1}{N}\sum_{n=1}^{N} V_n = \frac{1}{5}\left(5+5+10+10+10\right) = 8cent \tag{2.1}$$

The above outcome cannot be defined as the expected value because the system states (the coin values) are not hidden, and we've used the entire population (all 5 coins) for the mean value calculation.

Now assume five different weight measurements of the same person: 79.8kg, 80kg, 80.1kg, 79.8kg, and 80.2kg. The person is a system, and the person's weight is a system state.

**Figure 2.2:** *Man on scales.*

The measurements are different due to the random measurement error of the scales. We do not know the true value of the weight since it is a **Hidden State**. However, we can estimate the weight by averaging the scales' measurements.

$$W = \frac{1}{N}\sum_{n=1}^{N} W_n = \frac{1}{5}\left(79.8 + 80 + 80.1 + 79.8 + 80.2\right) = 79.98kg \tag{2.2}$$

The outcome of the estimate is the expected value of the weight.

The expected value is the value you would expect your hidden variable to have over a long time or many trials.

The mean is usually denoted by the Greek letter $\mu$.

The letter E usually denotes the expected value.

## 2.2 Variance and Standard deviation

The **Variance** is a measure of the spreading of the data set from its mean.

The **Standard Deviation** is the square root of the variance.

The standard deviation is denoted by the Greek letter $\sigma$ (sigma). Accordingly, the variance is denoted by $\sigma^2$.

Suppose we want to compare the heights of two high school basketball teams. The following table provides the players' heights and the mean height of each team.

| | Player 1 | Player 2 | Player 3 | Player 4 | Player 5 | Mean |
|---|---|---|---|---|---|---|
| **Team A** | 1.89m | 2.1m | 1.75m | 1.98m | 1.85m | **1.914m** |
| **Team B** | 1.94m | 1.9m | 1.97m | 1.89m | 1.87m | **1.914m** |

**Table 2.1:** *Players' heights.*

As we can see, the mean height of both teams is the same. Let us examine the height variance.

Since the variance measures the spreading of the data set, we would like to know the data set deviation from its mean. We can calculate the distance from the mean for each variable by subtracting the mean from each variable.

The height is denoted by $x$, and the heights mean by the Greek letter $\mu$. The distance from the mean for each variable would be:

$$x_n - \mu = x_n - 1.914m \tag{2.3}$$

The following table presents the distance from the mean for each variable.

| | Player 1 | Player 2 | Player 3 | Player 4 | Player 5 |
|---|---|---|---|---|---|
| **Team A** | -0.024m | 0.186m | -0.164m | 0.066m | -0.064m |
| **Team B** | 0.026m | -0.014m | 0.056m | -0.024m | -0.044m |

**Table 2.2:** *Distance from the mean.*

Some of the values are negative. To get rid of the negative values, let us square the distance from the mean:

$$(x_n - \mu)^2 = (x_n - 1.914m)^2 \tag{2.4}$$

The following table presents the squared distance from the mean for each variable.

| | Player 1 | Player 2 | Player 3 | Player 4 | Player 5 |
|---|---|---|---|---|---|
| **Team A** | $0.000576\text{m}^2$ | $0.034596\text{m}^2$ | $0.026896\text{m}^2$ | $0.004356\text{m}^2$ | $0.004096\text{m}^2$ |
| **Team B** | $0.000676\text{m}^2$ | $0.000196\text{m}^2$ | $0.003136\text{m}^2$ | $0.000576\text{m}^2$ | $0.001936\text{m}^2$ |

**Table 2.3:** *Squared distance from the mean.*

To calculate the variance of the data set, we need to find the average value of all squared distances from the mean:

$$\sigma^2 = \frac{1}{N}\sum_{n=1}^{N}(x_n - \mu)^2 \tag{2.5}$$

For team A, the variance would be:

$$\sigma_A^2 = \frac{1}{N}\sum_{n=1}^{N}(x_n - \mu)^2$$

$$= \frac{1}{5}(0.000576 + 0.034596 + 0.026896 + 0.004356 + 0.004096) = 0.014m^2$$

For team B, the variance would be:

$$\sigma_B^2 = \frac{1}{N}\sum_{n=1}^{N}(x_n - \mu)^2$$

$$= \frac{1}{5}(0.000676 + 0.000196 + 0.003136 + 0.000576 + 0.001936) = 0.0013m^2$$

We can see that although the mean of both teams is the same, the measure of the height spreading of Team A is higher than the measure of the height spreading of Team B. Therefore, the Team A players are more diverse than the Team B players. There are players for different positions like ball handler, center, and guards, while the Team B players are not versatile.

The units of the variance are meters squared; it is more convenient to look at the standard deviation, which is a square root of the variance.

$$\sigma = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(x_n - \mu)^2} \tag{2.6}$$

- The standard deviation of Team A players' heights would be 0.12m.
- The standard deviation of Team B players' heights would be 0.036m.

Now, assume that we would like to calculate the mean and variance of all basketball players in all high schools. That would be an arduous task - we would need to collect data on every player from every high school.

On the other hand, we can estimate the players' mean and variance by picking a big data set and making the calculations on this data set.

The data set of 100 randomly selected players should be sufficient for an accurate estimation.

However, when we estimate the variance, the equation for the variance calculation is slightly different. Instead of normalizing by the factor $N$, we shall normalize by the factor $N - 1$:

$$\sigma^2 = \frac{1}{N-1}\sum_{n=1}^{N}(x_n - \mu)^2 \tag{2.7}$$

The factor of $N - 1$ is called Bessel's correction.

You can see the mathematical proof of the above equation on visiondummy or Wikipedia.

## 2.3 Normal Distribution

It turns out that many natural phenomena follow the **Normal Distribution**. The normal distribution, also known as the **Gaussian** (named after the mathematician Carl Friedrich Gauss), is described by the following equation:

$$f\left(x; \mu, \sigma^2\right) = \frac{1}{\sqrt{2\pi\sigma^2}}\, exp\left(\frac{-\left(x-\mu\right)^2}{2\sigma^2}\right) \tag{2.8}$$

The Gaussian curve is also called the **PDF (Probability Density Function)** for the normal distribution.

The following chart describes PDFs of the pizza delivery time in three cities: city 'A,' city 'B,' and city 'C.'

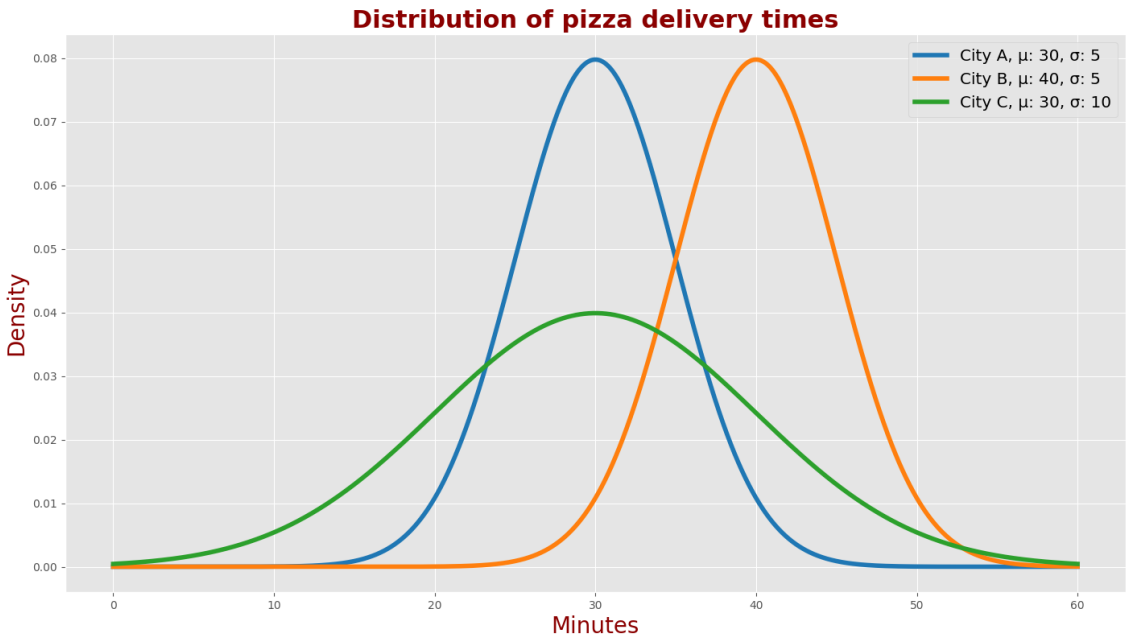


**Figure 2.3:** *Normal distribution PDFs.*

- In city 'A,' the mean delivery time is 30 minutes, and the standard deviation is 5 minutes.
- In city 'B,' the mean delivery time is 40 minutes, and the standard deviation is 5 minutes.
- In city 'C,' the mean delivery time is 30 minutes, and the standard deviation is 10 minutes.

We can see that the Gaussian shapes of the city 'A' and city 'B' pizza delivery times are identical; however, their centers are different. That means that in city 'A,' you wait for pizza for 10 minutes less on average, while the measure of spread in pizza delivery time is the same.

We can also see that the centers of Gaussians in the city 'A' and city 'C' are the same; however, their shapes are different. Therefore the average pizza delivery time in both cities is the same, but the measure of spread is different.

The following chart describes the proportions of the normal distribution.

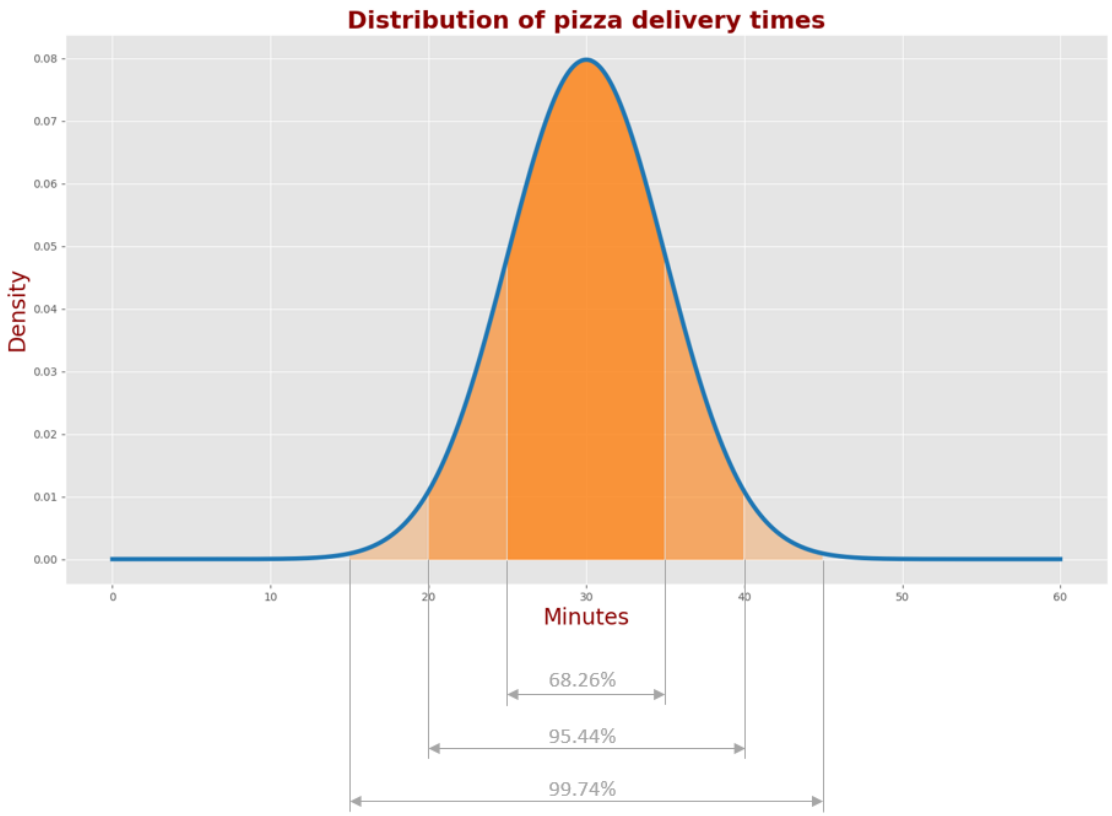


**Figure 2.4:** *Proportions of the normal distribution.*

- 68.26% of the pizza delivery times in City A lie within $\mu \pm \sigma$ range (25-35 minutes)
- 95.44% of the pizza delivery times in City A lie within $\mu \pm 2\sigma$ range (20-40 minutes)
- 99.74% of the pizza delivery times in City A lie within $\mu \pm 3\sigma$ range (15-45 minutes)

Usually, measurement errors are distributed normally. The Kalman Filter design assumes a normal distribution of the measurement errors.

## 2.4 Random Variables

A **random variable** describes the hidden state of the system. A random variable is a set of possible values from a random experiment.

The random variable can be continuous or discrete:

- A continuous random variable can take any value within a specific range, such as battery charge time or marathon race time.
- A discrete random variable is countable, such as the number of website visitors or the number of students in the class.

The random variable is described by the probability density function. The probability density function is characterized by **moments**.

The moments of the random value are expected values of powers of the random variable. We are interested in two types of moments:

- The $k^{th}$ raw moment is the expected value of the $k^{th}$ power of the random variable: $E\left(X^k\right)$.
- The $k^{th}$ central moment is the expected value of the $k^{th}$ power of the random variable distribution about its mean: $E\left(\left(X-\mu_X\right)^k\right)$.

In this book, the random variables are characterized by the following:

- The first raw moment $E\left(X\right)$ – the mean of the sequence of measurements.
- The second central moment $E\left(\left(X-\mu_X\right)^2\right)$ – the variance of the sequence of measurements.

## 2.5 Estimate, Accuracy and Precision

An **Estimate** is about evaluating the hidden state of the system. The true position of the aircraft is hidden from the observer. We can estimate the aircraft position using sensors, such as radar. The estimate can be significantly improved by using multiple sensors and applying advanced estimation and tracking algorithms (such as the Kalman Filter). Every measured or computed parameter is an estimate.

**Accuracy** indicates how close the measurement is to the true value.

**Precision** describes the variability in a series of measurements of the same parameter. Accuracy and precision form the basis of the estimate.

The following figure illustrates accuracy and precision.

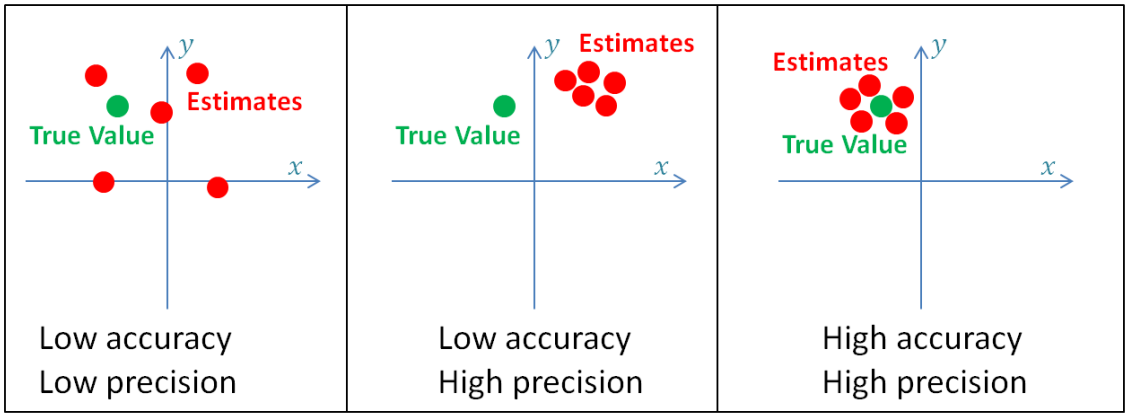


**Figure 2.5:** *Accuracy and Precision.*

High-precision systems have low variance in their measurements (i.e., low uncertainty), while low-precision systems have high variance in their measurements (i.e., high uncertainty). The random measurement error produces the variance.

Low-accuracy systems are called **biased** systems since their measurements have a built-in systematic error (bias).

The influence of the variance can be significantly reduced by averaging or smoothing measurements. For example, if we measure temperature using a thermometer with a random measurement error, we can make multiple measurements and average them. Since the error is random, some measurements would be above the true value and others below the true value. The estimate would be close to the true value. The more measurements we make, the closer the estimate will be.

On the other hand, a biased thermometer produces a constant systematic error in the estimate.

All examples in this book assume **unbiased** systems.

## 2.6 Summary

The following figure represents a statistical view of measurement.

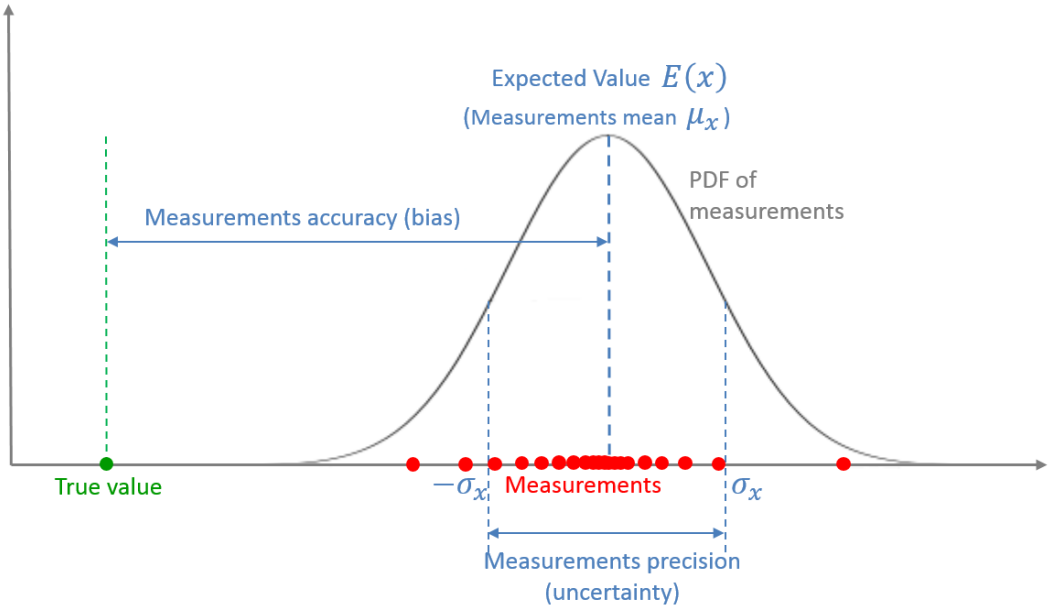


**Figure 2.6:** *Statistical view of measurement.*

A measurement is a **random variable** described by the **PDF**.

The mean of the measurements is the **Expected Value** of the random variable.

The offset between the mean of the measurements and the true value is the **accuracy of the measurements**, also known as **bias** or **systematic measurement error**.

The dispersion of the distribution is the measurement **precision**, also known as the **measurement noise**, **random measurement error**, or **measurement uncertainty**.

# 3. The $\alpha - \beta - \gamma$ filter

This chapter is introductory, and it describes the $\alpha - \beta$ and $\alpha - \beta - \gamma$ filters. These filters are frequently used for time series data smoothing. The principles of the $\alpha - \beta(-\gamma)$ filter are closely related to the Kalman Filter principles.

## 3.1 Example 1 – Weighting the gold

Now we are ready for the first simple example. In this example, we estimate the state of the static system. A static system is a system that doesn't change its state over a reasonable period. For instance, the static system could be a tower, and the state would be its height.

In this example, we estimate the weight of the gold bar. We have unbiased scales, i.e., the measurements don't have a systematic error, but the measurements do include random noise.


**Figure 3.1:** *Gold Bars.*

The system is the gold bar, and the system state is the weight of the gold bar. The dynamic model of the system is constant since we assume that the weight doesn't change over short periods.

To estimate the system state (i.e., the weight value), we can make multiple measurements and average them.

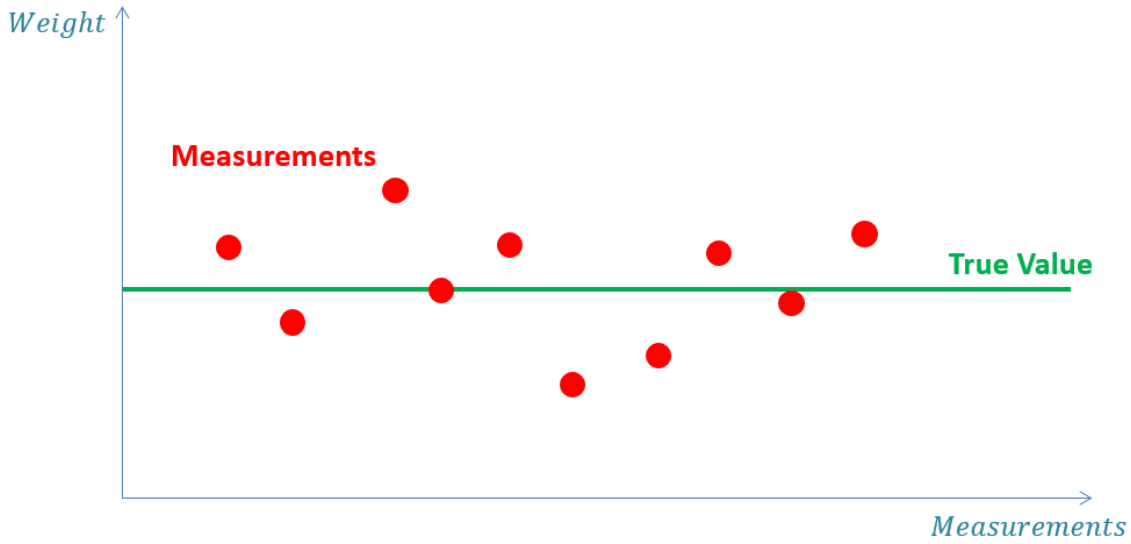


**Figure 3.2:** *Measurements vs. True value.*

At the time $n$, the estimate $\hat{x}_{n,n}$ would be the average of all previous measurements:

$$\hat{x}_{n,n} = \frac{1}{n}(z_1 + z_2 + \ldots + z_{n-1} + z_n) = \frac{1}{n}\sum_{i=1}^{n}(z_i) \tag{3.1}$$

Example Notation:

| | |
|---|---|
| $x$ | is the true value of the weight |
| $z_n$ | is the measured value of the weight at time $n$ |
| $\hat{x}_{n,n}$ | is the estimate of $x$ at time $n$ (the estimate is made after taking the measurement $z_n$) |
| $\hat{x}_{n+1,n}$ | is the estimate of the future state $(n+1)$ of $x$. The estimate is made at the time $n$. In other words, $\hat{x}_{n+1,n}$ is a predicted state or extrapolated state |
| $\hat{x}_{n-1,n-1}$ | is the estimate of $x$ at time $n-1$ (the estimate is made after taking the measurement $z_{n-1}$) |
| $\hat{x}_{n,n-1}$ | is a prior prediction - the estimate of the state at time $n$. The prediction is made at the time $n-1$ |

R In the literature, a caret (or hat) over a variable indicates an estimated value.

The dynamic model in this example is static (or constant) since the weight of gold doesn't change over time, therefore $\hat{x}_{n+1,n} = \hat{x}_{n,n}$.

Although the Equation 3.1 is mathematically correct, it is not practical for implementation. In order to estimate $\hat{x}_{n,n}$ we need to remember all historical measurements; therefore, we need a large memory. We also need to recalculate the average repeatedly

if we want to update the estimated value after every new measurement. Thus, we need a more powerful Central Processing Unit (CPU).

It would be more practical to keep the last estimate only ($\hat{x}_{n-1,n-1}$) and update it after every new measurement. The following figure exemplifies the required algorithm:

- Estimate the current state based on the measurement and prior prediction.
- Predict the next state based on the current state estimate using the Dynamic Model.

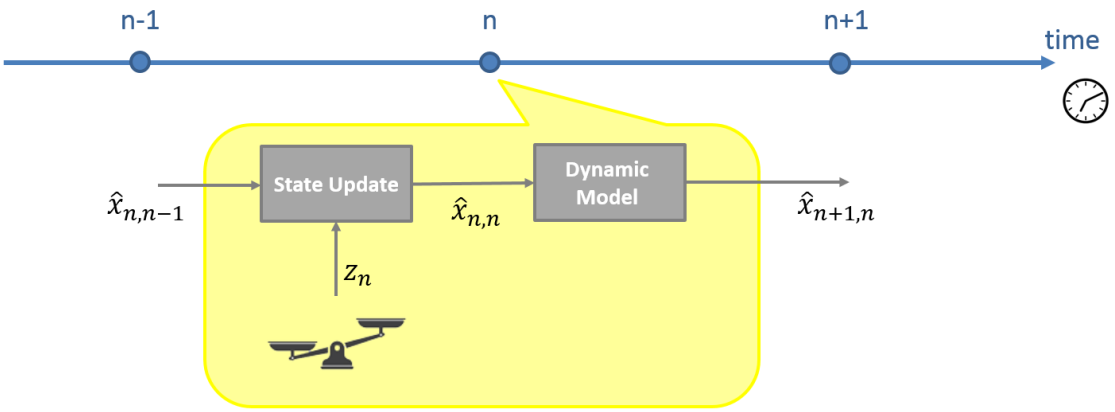


**Figure 3.3:** *Example Notation.*

We can modify the averaging equation for our needs using a small mathematical trick:

| Equation | Notes |
|---|---|
| $\hat{x}_{n,n} = \frac{1}{n}\sum_{i=1}^{n}(z_i)$ | Average formula: sum of $n$ measurements divided by $n$ |
| $= \frac{1}{n}\left(\sum_{i=1}^{n-1}(z_i) + z_n\right)$ | Sum of the $n-1$ measurements plus the last measurement divided by $n$ |
| $= \frac{1}{n}\sum_{i=1}^{n-1}(z_i) + \frac{1}{n}z_n$ | Expand |
| $= \frac{1}{n}\frac{n-1}{n-1}\sum_{i=1}^{n-1}(z_i) + \frac{1}{n}z_n$ | Multiply and divide by term $n-1$ |
| $= \frac{n-1}{n}\frac{1}{n-1}\sum_{i=1}^{n-1}(z_i) + \frac{1}{n}z_n$ | Reorder. The 'orange' term is the prior estimate |
| $= \frac{n-1}{n}\hat{x}_{n-1,n-1} + \frac{1}{n}z_n$ | Rewriting the sum |
| $= \hat{x}_{n-1,n-1} - \frac{1}{n}\hat{x}_{n-1,n-1} + \frac{1}{n}z_n$ | Distribute the term $\frac{n-1}{n}$ |
| $= \hat{x}_{n-1,n-1} + \frac{1}{n}(z_n - \hat{x}_{n-1,n-1})$ | Reorder |

**Table 3.1:** *Averaging equation.*

$\hat{x}_{n-1,n-1}$ is the estimated state of $x$ at the time $n-1$, based on the measurement at the time $n-1$.

Let's find $\hat{x}_{n,n-1}$ (the predicted state of $x$ at the time $n$), based on $\hat{x}_{n-1,n-1}$ (the estimation at the time $n-1$). In other words, we would like to extrapolate $\hat{x}_{n-1,n-1}$ to the time $n$.

Since the dynamic model in this example is static, the predicted state of $x$ equals the estimated state of $x$: $\hat{x}_{n,n-1} = \hat{x}_{n-1,n-1}$.

Based on the above, we can write the **State Update Equation**:

**State Update Equation**

$$\hat{x}_{n,n} = \hat{x}_{n,n-1} + \frac{1}{n}\left(z_n - \hat{x}_{n,n-1}\right) \tag{3.2}$$

The State Update Equation is one of the five Kalman filter equations. It means the following:

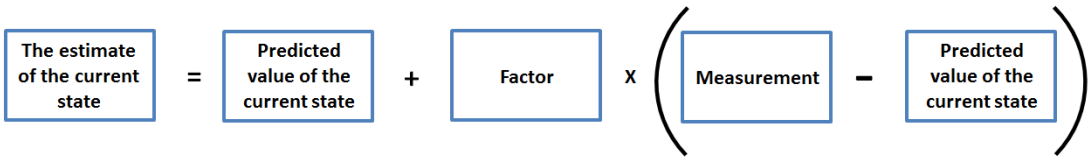


**Figure 3.4:** *State Update Equation.*

The factor $1/n$ is specific to our example. We will discuss the vital role of this factor later, but right now, I would like to note that in "Kalman Filter language," this factor is called the **Kalman Gain**. It is denoted by $K_n$. The subscript $n$ indicates that the Kalman Gain can change with every iteration.

The discovery of $K_n$ was one of Rudolf Kalman's significant contributions.

Before we get into the guts of the Kalman Filter, we use the Greek letter $\alpha_n$ instead of $K_n$.

So, the State Update Equation looks as follows:

$$\hat{x}_{n,n} = \hat{x}_{n,n-1} + \alpha_n\left(z_n - \hat{x}_{n,n-1}\right) \tag{3.3}$$

The term $(z_n - \hat{x}_{n,n-1})$ is the "measurement residual," also called **innovation**. The innovation contains new information.

In this example, $1/n$ decreases as $n$ increases. In the beginning, we don't have enough information about the current state; thus, the first estimation is based on the first measurement $\frac{1}{n}|_{n=1} = 1$ . As we continue, each successive measurement has less weight in the estimation process, since $1/n$ decreases. At some point, the contribution of the new measurements will become negligible.

Let's continue with the example. Before we make the first measurement, we can guess (or rough estimate) the gold bar weight simply by reading the stamp on the gold bar. It is called the **Initial Guess**, and it is our first estimate.

The Kalman Filter requires the initial guess as a preset, which can be very rough.

### 3.1.1 Estimation algorithm

The following chart depicts the estimation algorithm that is used in this example.

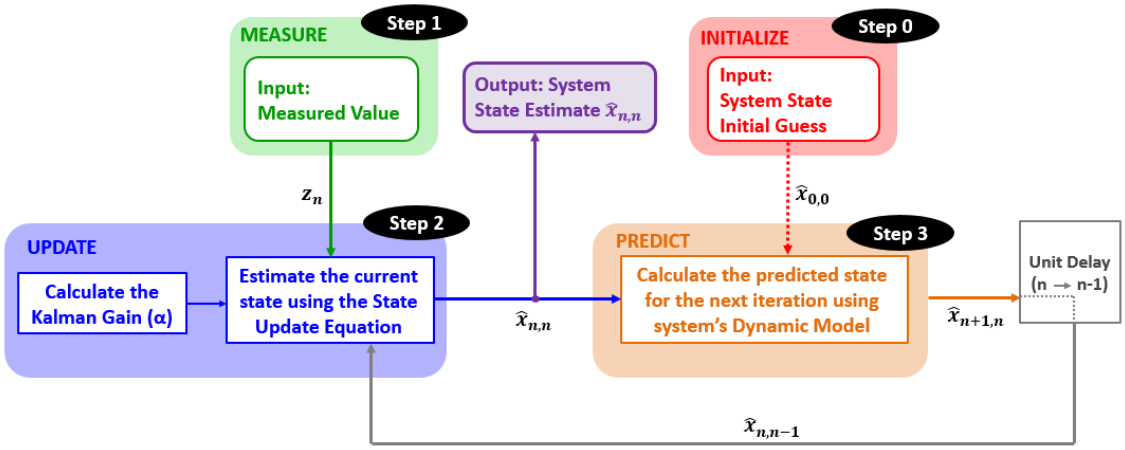


**Figure 3.5:** *State Update Equation.*

Now we are ready to start the measurement and estimation process.

### 3.1.2 The numerical example

#### 3.1.2.1 Iteration Zero

##### Initialization

Our initial guess of the gold bar weight is 1000 grams. The initial guess is used only once for the filter initiation. Thus, it won't be required for successive iterations.

$\hat{x}_{0,0} = 1000g$

##### Prediction

The weight of the gold bar is not supposed to change. Therefore, the dynamic model of the system is static. Our next state estimate (prediction) equals the initialization:

$\hat{x}_{1,0} = \hat{x}_{0,0} = 1000g$

#### 3.1.2.2 First Iteration

##### Step 1

Making the weight measurement with the scales:

$z_1 = 996g$

#### Step 2

Calculating the gain. In our example $\alpha_n = 1/n$, thus:

$$\alpha_1 = \frac{1}{1} = 1$$

Calculating the current estimate using the State Update Equation:

$$\hat{x}_{1,1} = \hat{x}_{1,0} + \alpha_1 (z_1 - \hat{x}_{1,0}) = 1000 + 1\,(996 - 1000) = 996g$$

R The initial guess could be any number in this specific example. Since $\alpha_1 = 1$, the initial guess is eliminated in the first iteration.

#### Step 3

The dynamic model of the system is static; thus, the weight of the gold bar is not supposed to change. Our next state estimate (prediction) equals to current state estimate:

$$\hat{x}_{2,1} = \hat{x}_{1,1} = 996g$$

### 3.1.2.3 Second Iteration

After a unit time delay, the **predicted estimate** from the previous iteration becomes the **prior estimate** in the current iteration:

$$\hat{x}_{2,1} = 996g$$

#### Step 1

Making the second measurement of the weight:

$$z_2 = 994g$$

#### Step 2

Calculating the gain:

$$\alpha_2 = \frac{1}{2}$$

Calculating the current estimate:

$$\hat{x}_{2,2} = \hat{x}_{2,1} + \alpha_2 (z_2 - \hat{x}_{2,1}) = 996 + \frac{1}{2}(994 - 996) = 995g$$

#### Step 3

$$\hat{x}_{3,2} = \hat{x}_{2,2} = 995g$$

#### 3.1.2.4 Third Iteration

$$z_3 = 1021g$$

$$\alpha_3 = \frac{1}{3}$$

$$\hat{x}_{3,3} = 995 + \frac{1}{3}(1021 - 995) = 1003.67g$$

$$\hat{x}_{4,3} = 1003.67g$$

#### 3.1.2.5 Fourth Iteration

$$z_4 = 1000g$$

$$\alpha_4 = \frac{1}{4}$$

$$\hat{x}_{4,4} = 1003.67 + \frac{1}{4}(1000 - 1003.67) = 1002.75g$$

$$\hat{x}_{5,4} = 1002.75g$$

#### 3.1.2.6 Fifth Iteration

$$z_5 = 1002g$$

$$\alpha_5 = \frac{1}{5}$$

$$\hat{x}_{5,5} = 1002.75 + \frac{1}{5}(1002 - 1002.75) = 1002.6g$$

$$\hat{x}_{6,5} = 1002.6g$$

#### 3.1.2.7 Sixth Iteration

$$z_6 = 1010g$$

$$\alpha_6 = \frac{1}{6}$$

$$\hat{x}_{6,6} = 1002.6 + \frac{1}{6}(1010 - 1002.6) = 1003.83$$

$$\hat{x}_{7,6} = 1003.83g$$

#### 3.1.2.8 Seventh Iteration

$$z_7 = 983g$$

$$\alpha_7 = \frac{1}{7}$$

$$\hat{x}_{7,7} = 1003.83 + \frac{1}{7}(983 - 1003.83) = 1000.86g$$

$$\hat{x}_{8,7} = 1000.86g$$

#### 3.1.2.9 Eighth Iteration

$$z_8 = 971g$$

$$\alpha_8 = \frac{1}{8}$$

$$\hat{x}_{8,8} = 1000.86 + \frac{1}{8}(971 - 1000.86) = 997.125g$$

$$\hat{x}_{9,8} = 997.125g$$

#### 3.1.2.10 Ninth Iteration

$$z_9 = 993g$$

$$\alpha_9 = \frac{1}{9}$$

$$\hat{x}_{9,9} = 997.125 + \frac{1}{9}(993 - 997.125) = 996.67g$$

$$\hat{x}_{10,9} = 996.67g$$

#### 3.1.2.11 Tenth Iteration

$$z_{10} = 1023g$$

$$\alpha_{10} = \frac{1}{10}$$

$$\hat{x}_{10,10} = 996.67 + \frac{1}{10}(1023 - 996.67) = 999.3g$$

$$\hat{x}_{11,10} = 999.3g$$

We can stop here. The gain decreases with each measurement. Therefore, the contribution of each successive measurement is lower than the contribution of the previous measurement. We get pretty close to the true weight, which is 1000g. If we were making more measurements, we would get closer to the true value.

The following table summarizes our measurements and estimates, and the chart compares the measured values, the estimates, and the true value.

| $n$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| $\alpha_n$ | 1 | $\alpha_2$ | $\alpha_3$ | $\alpha_4$ | $\alpha_5$ | $\alpha_6$ | $\alpha_7$ | $\alpha_8$ | $\alpha_9$ | $\alpha_{10}$ |
| $z_n$ | 996 | 994 | 1021 | 1000 | 1002 | 1010 | 983 | 971 | 993 | 1023 |
| $\hat{x}_{n,n}$ | 996 | 995 | 1003.67 | 1002.75 | 1002.6 | 1003.83 | 1000.86 | 997.125 | 996.67 | 999.3 |
| $\hat{x}_{n+1,n}$ | 996 | 995 | 1003.67 | 1002.75 | 1002.6 | 1003.83 | 1000.86 | 997.125 | 996.67 | 999.3 |

**Table 3.2:** *Example 1 summary.*

### 3.1.3 Results analysis

The following chart compares the true, measured, and estimated values.

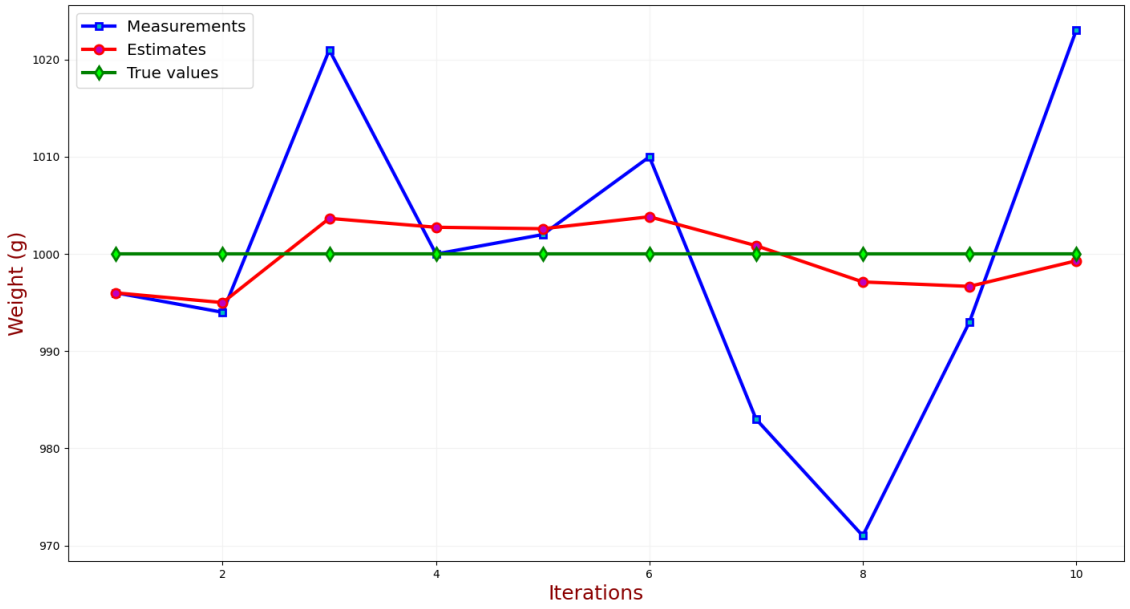


**Figure 3.6:** *Example 1: Measurements vs. True value vs. Estimates.*

The estimation algorithm has a smoothing effect on the measurements and converges toward the true value.

### 3.1.4 Example summary

In this example, we've developed a simple estimation algorithm for a static system. We have also derived the state update equation, one of the five Kalman Filter equations. We will revise the state update equation in subsection 4.1.4.

## 3.2 Example 2 – Tracking the constant velocity aircraft

It is time to examine a dynamic system that changes its state over time. In this example, we track a constant-velocity aircraft in one dimension using the $\alpha - \beta$ **filter**. Let us assume a one-dimensional world. We assume an aircraft that is moving radially away from the radar (or towards the radar). In the one-dimensional world, the angle to the radar is constant, and the aircraft altitude is constant, as shown in the following figure. The following chart compares the true, measured, and estimated values.

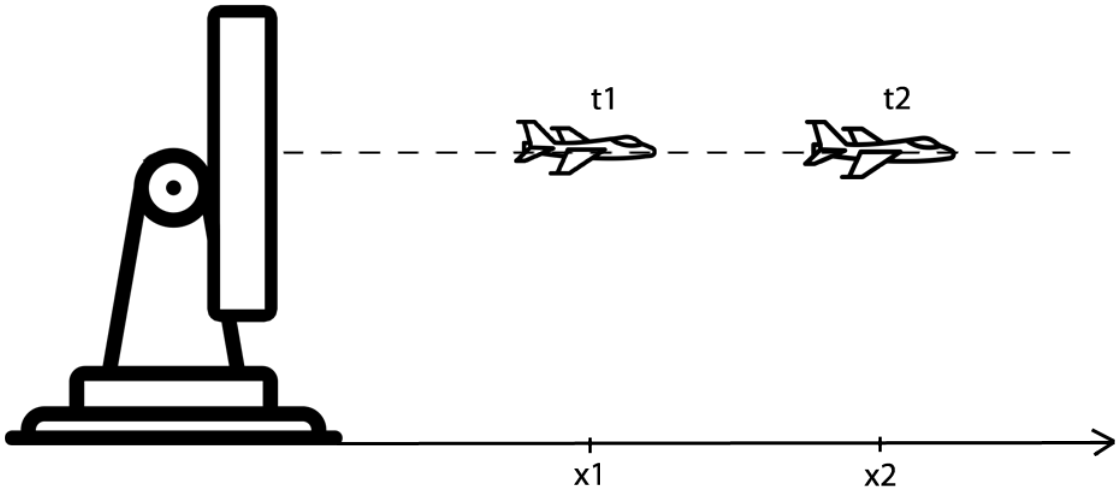


**Figure 3.7:** *1D scenario.*

$x_n$ represents the range to the aircraft at time $n$. The aircraft velocity can be approximated using the range differentiation method - the change in the measured range with time.

Thus, the velocity is a derivative of the range:

$$\dot{x} = v = \frac{dx}{dt} \tag{3.4}$$

The radar sends a track beam in the direction of the target at a constant rate. The track-to-track interval is $\Delta t$.

Two motion equations describe the system dynamic model for constant velocity motion:

$$\begin{aligned} x_{n+1} &= x_n + \Delta t \dot{x}_n \\ \dot{x}_{n+1} &= \dot{x}_n \end{aligned} \tag{3.5}$$

According to Equation 3.5, the aircraft range at the next track cycle equals the range at the current track cycle plus the target velocity multiplied by the track-to-track interval. Since we assume constant velocity in this example, the velocity at the next cycle equals the velocity at the current cycle.

The above system of equations is called a **State Extrapolation Equation** (also called a **Transition Equation** or a **Prediction Equation**) and is also one of the five Kalman filter equations. This system of equations extrapolates the current state to the next state (prediction).

We have already used the State Extrapolation Equation in the previous example, where we assumed that the weight at the next state equals the weight at the current state.

The State Extrapolation Equations depend on the system dynamics and differ from example to example.

There is a general form of the State Extrapolation Equation in matrix notation. We learn it later.

In this example, we use the above equations specific to our case.

R We have already learned two of the five Kalman Filter equations:
- State Update Equation
- State Extrapolation Equation

Now we are going to modify the State Update Equation for our example.

### 3.2.1 The $\alpha - \beta$ filter

Let the radar track-to-track ($\Delta t$) period be 5 seconds. Assume that at time $n-1$, the estimated range of the Unmanned Air Vehicle (UAV) is 30,000m, and the estimated UAV velocity is 40m/s.

Using the State Extrapolation Equations, we can predict the target position at time $n$:

$$\hat{x}_{n,n-1} = \hat{x}_{n-1,n-1} + \Delta t \hat{\dot{x}}_{n-1,n-1} = 30000 + 5 \times 40 = 30200m$$

The target velocity prediction for time $n$:

$$\hat{\dot{x}}_{n,n-1} = \hat{\dot{x}}_{n-1,n-1} = 40m/s$$

However, at time $n$, the radar measures range ($z_n$) of 30,110m and not 30,200m

as expected. There is a 90m gap between the expected (predicted) range and the measured range. There are two possible reasons for this gap:

- The radar measurements are not precise.
- The aircraft velocity has changed. The new aircraft velocity is: $\frac{30{,}110-30{,}000}{5} = 22m/s$.

Which of the two statements is true?

Let us write down the State Update Equation for the velocity:

$$\hat{\dot{x}}_{n,n} = \hat{\dot{x}}_{n,n-1} + \beta\left(\frac{z_n - \hat{x}_{n,n-1}}{\Delta t}\right) \tag{3.6}$$

The value of the factor $\beta$ depends on the precision level of the radar. Suppose that the $1\sigma$ precision of the radar is 20m. The 90 meters gap between the predicted and measured ranges most likely results from a change in the aircraft velocity. We should set the $\beta$ factor to a high value in this case. If we set $\beta = 0.9$, then the estimated velocity would be:

$$\hat{\dot{x}}_{n,n} = \hat{\dot{x}}_{n,n-1} + \beta\left(\frac{z_n - \hat{x}_{n,n-1}}{\Delta t}\right) = 40 + 0.9\left(\frac{30110 - 30200}{5}\right) = 23.8m/s$$

On the other hand, suppose that the $1\sigma$ precision of the radar is 150m. Then the 90 meters gap probably results from the radar measurement error. We should set the $\beta$ factor to a low value in this case. If we set $\beta = 0.1$, then the estimated velocity would be:

$$\hat{\dot{x}}_{n,n} = \hat{\dot{x}}_{n,n-1} + \beta\left(\frac{z_n - \hat{x}_{n,n-1}}{\Delta t}\right) = 40 + 0.1\left(\frac{30110 - 30200}{5}\right) = 38.2m/s$$

If the aircraft velocity has changed from 40m/s to 22m/s, we see this after 10 track cycles (running the above equation 10 times with $\beta = 0.1$). If the gap has been caused by measurement error, then the successive measurements would be in front or behind the predicted positions. Thus on average, the target velocity would not change.

The State Update Equation for the aircraft position is similar to the equation that was derived in the previous example:

$$\hat{x}_{n,n} = \hat{x}_{n,n-1} + \alpha\left(z_n - \hat{x}_{n,n-1}\right) \tag{3.7}$$

Unlike the previous example, where the $\alpha$ factor is recalculated in each iteration

$\left(\alpha_n = \frac{1}{n}\right)$ , the $\alpha$ factor is constant in this example.

The magnitude of the $\alpha$ factor depends on the radar measurement precision. For high precision-radar, we should choose high $\alpha$, giving high weight to the measurements. If $\alpha = 1$, then the estimated range equals the measured range:

$$\hat{x}_{n,n} = \hat{x}_{n,n-1} + 1\,(z_n - \hat{x}_{n,n-1}) = z_n \tag{3.8}$$

If $\alpha = 0$ , then the measurement has no meaning:

$$\hat{x}_{n,n} = \hat{x}_{n,n-1} + 0\,(z_n - \hat{x}_{n,n-1}) = \hat{x}_{n,n-1} \tag{3.9}$$

So, we have derived a system of equations that composes the State Update Equation for the radar tracker. They are also called **$\alpha - \beta$ track update equations** or **$\alpha - \beta$ track filtering equations**.

### The State Update Equation for position and velocity

$$\begin{aligned} \hat{x}_{n,n} &= \hat{x}_{n,n-1} + \alpha\,(z_n - \hat{x}_{n,n-1}) \\ \hat{\dot{x}}_{n,n} &= \hat{\dot{x}}_{n,n-1} + \beta\left(\frac{z_n - \hat{x}_{n,n-1}}{\Delta t}\right) \end{aligned} \tag{3.10}$$

R In some books, the $\alpha - \beta$ filter is called the **g-h filter**, where the Greek letter $\alpha$ is replaced by the English letter g, and the English letter h replaces the Greek letter $\beta$.

R In this example, we are deriving the aircraft velocity from the range measurements ( $\dot{x} = \frac{\Delta x}{\Delta t}$ ). Modern radars can measure radial velocity directly using the Doppler Effect. However, my goal is to explain the Kalman Filter, not the radar operation. So, for the sake of generality, I will keep deriving the velocity from the range measurements in our examples.

### 3.2.2 Estimation Algorithm

The following chart depicts the estimation algorithm that is used in this example.

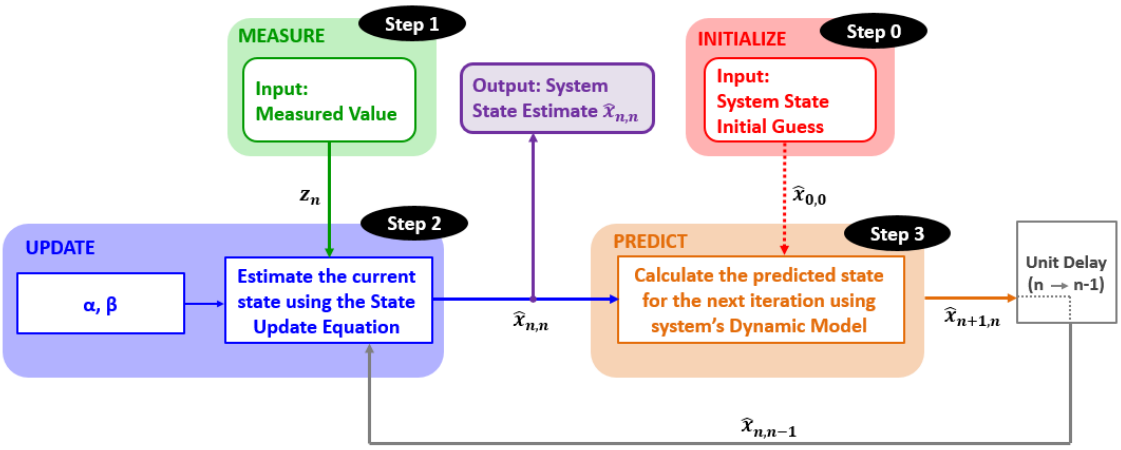


**Figure 3.8:** *$\alpha - \beta$ filter estimation algorithm.*

Unlike the previous example, the Gain values $\alpha$ and $\beta$ are given for this example. For the Kalman Filter, the $\alpha$ and $\beta$ are replaced by Kalman Gain, which is calculated at each iteration, but we learn it later.

Now we are ready to start a numerical example.

### 3.2.3 The numerical example

Consider an aircraft moving radially toward (or away from) a radar in a one-dimensional world.

The $\alpha - \beta$ filter parameters are:

- $\alpha = 0.2$
- $\beta = 0.1$

The track-to-track interval is 5 seconds.

R Note: In this example, we use an imprecise radar and a low-speed target (UAV) for better graphical representation. The radars are usually more precise in real life, and the targets can be much faster.

#### 3.2.3.1 Iteration Zero

##### Initialization

The initial conditions for the time $n = 0$ are given:

$\hat{x}_{0,0} = 30000m$

$\hat{\dot{x}}_{0,0} = 40m/s$

R The **Track Initiation** (or how we get the initial conditions) is an important topic that will be discussed in chapter 21. Right now, our goal is to understand the basic $\alpha - \beta$ filter operation, so let's assume that the initial conditions are given by somebody else.

#### Prediction

The initial guess should be extrapolated to the first cycle using the State Extrapolation Equations:

$\hat{x}_{n+1,n} = \hat{x}_{n,n} + \Delta t \hat{\dot{x}}_{n,n} \rightarrow \hat{x}_{1,0} = \hat{x}_{0,0} + \Delta t \hat{\dot{x}}_{0,0} = 30000 + 5 \times 40 = 30200m$

$\hat{\dot{x}}_{n+1,n} = \hat{\dot{x}}_{n,n} \rightarrow \hat{\dot{x}}_{1,0} = \hat{\dot{x}}_{0,0} = 40m/s$

### 3.2.3.2 First Iteration

In the first cycle ($n = 1$), the initial guess is the prior estimate:

$\hat{x}_{n,n-1} = \hat{x}_{1,0} = 30200m$

$\hat{\dot{x}}_{n,n-1} = \hat{\dot{x}}_{1,0} = 40m/s$

#### Step 1

The radar measures the aircraft range:

$z_1 = 30171m$

#### Step 2

Calculating the current estimate using the State Update Equation:

$\hat{x}_{1,1} = \hat{x}_{1,0} + \alpha\left(z_1 - \hat{x}_{1,0}\right) = 30200 + 0.2\left(30171 - 30200\right) = 30194.2m$

$$\hat{\dot{x}}_{1,1} = \hat{\dot{x}}_{1,0} + \beta\left(\frac{z_1 - \hat{x}_{1,0}}{\Delta t}\right) = 40 + 0.1\left(\frac{30171 - 30200}{5}\right) = 39.42m/s$$

#### Step 3

Calculating the next state estimate using the State Extrapolation Equations:

$\hat{x}_{2,1} = \hat{x}_{1,1} + \Delta t \hat{\dot{x}}_{1,1} = 30194.2 + 5 \times 39.42 = 30391.3m$

$\hat{\dot{x}}_{2,1} = \hat{\dot{x}}_{1,1} = 39.42m/s$

#### 3.2.3.3 Second Iteration

After a unit time delay, the predicted estimate from the previous iteration becomes the prior estimate in the current iteration:

$\hat{x}_{2,1} = 30391.3m$

$\hat{\dot{x}}_{2,1} = 39.42m/s$

##### Step 1

The radar measures the aircraft range:

$z_2 = 30353m$

##### Step 2

Calculating the current estimate using the State Update Equation:

$\hat{x}_{2,2} = \hat{x}_{2,1} + \alpha\left(z_2 - \hat{x}_{2,1}\right) = 30391.3 + 0.2\left(30353 - 30391.3\right) = 30383.64m$

$\hat{\dot{x}}_{2,2} = \hat{\dot{x}}_{2,1} + \beta\left(\frac{z_2 - \hat{x}_{2,1}}{\Delta t}\right) = 39.42 + 0.1\left(\frac{30353 - 30391.3}{5}\right) = 38.65m/s$

##### Step 3

Calculating the next state estimate using the State Extrapolation Equations:

$\hat{x}_{3,2} = \hat{x}_{2,2} + \Delta t\hat{\dot{x}}_{2,2} = 30383.64 + 5 \times 38.65 = 30576.9m$

$\hat{\dot{x}}_{3,2} = \hat{\dot{x}}_{2,2} = 38.65m/s$

#### 3.2.3.4 Third Iteration

$z_3 = 30756m$

$\hat{x}_{3,3} = 30576.9 + 0.2\left(30756 - 30576.9\right) = 30612.73m$

$\hat{\dot{x}}_{3,3} = 38.65 + 0.1\left(\frac{30756 - 30576.9}{5}\right) = 42.2m/s$

$\hat{x}_{4,3} = 30612.73 + 5 \times 42.2 = 30823.9m$

$\hat{\dot{x}}_{4,3} = 42.2m/s$

#### 3.2.3.5 Fourth Iteration

$$z_4 = 30799m$$
$$\hat{x}_{4,4} = 30823.9 + 0.2\,(30799 - 30823.9) = 30818.93m$$
$$\hat{\dot{x}}_{4,4} = 42.2 + 0.1\left(\frac{30799 - 30823.9}{5}\right) = 41.7m/s$$
$$\hat{x}_{5,4} = 30818.93 + 5 \times 41.7 = 31027.6m$$
$$\hat{\dot{x}}_{5,4} = 41.7m/s$$

#### 3.2.3.6 Fifth Iteration

$$z_5 = 31018m$$
$$\hat{x}_{5,5} = 31027.6 + 0.2\,(31018 - 31027.6) = 31025.7m$$
$$\hat{\dot{x}}_{5,5} = 41.7 + 0.1\left(\frac{31018 - 31027.6}{5}\right) = 41.55m/s$$
$$\hat{x}_{6,5} = 31025.7 + 5 \times 41.55 = 31233.4m$$
$$\hat{\dot{x}}_{6,5} = 41.55m/s$$

#### 3.2.3.7 Sixth Iteration

$$z_6 = 31278m$$
$$\hat{x}_{6,6} = 31233.4 + 0.2\,(31278 - 31233.4) = 31242.3m$$
$$\hat{\dot{x}}_{6,6} = 41.55 + 0.1\left(\frac{31278 - 31233.4}{5}\right) = 42.44m/s$$
$$\hat{x}_{7,6} = 31242.3 + 5 \times 42.44 = 31454.5m$$
$$\hat{\dot{x}}_{7,6} = 42.44m/s$$

#### 3.2.3.8 Seventh Iteration

$$z_7 = 31276m$$
$$\hat{x}_{7,7} = 31454.5 + 0.2\,(31276 - 31454.5) = 31418.8m$$
$$\hat{\dot{x}}_{7,7} = 42.44 + 0.1\left(\frac{31276 - 31454.5}{5}\right) = 38.9m/s$$
$$\hat{x}_{8,7} = 31418.8 + 5 \times 38.9 = 31613.15m$$
$$\hat{\dot{x}}_{8,7} = 38.9m/s$$

#### 3.2.3.9 Eighth Iteration

$$z_8 = 31379m$$
$$\hat{x}_{8,8} = 31613.15 + 0.2\,(31379 - 31613.15) = 31566.3m$$
$$\hat{\dot{x}}_{8,8} = 38.9 + 0.1\left(\frac{31379 - 31613.15}{5}\right) = 34.2m/s$$
$$\hat{x}_{9,8} = 31566.3 + 5 \times 34.2 = 31737.24m$$
$$\hat{\dot{x}}_{9,8} = 34.2m/s$$

#### 3.2.3.10 Ninth Iteration

$$z_9 = 31748m$$
$$\hat{x}_{9,9} = 31737.24 + 0.2\,(31748 - 31737.24) = 31739.4m$$
$$\hat{\dot{x}}_{9,9} = 34.2 + 0.1\left(\frac{31748 - 31737.24}{5}\right) = 34.4m/s$$
$$\hat{x}_{10,9} = 31739.4 + 5 \times 34.4 = 31911.4m$$
$$\hat{\dot{x}}_{10,9} = 34.4m/s$$

#### 3.2.3.11 Tenth Iteration

$$z_{10} = 32175m$$
$$\hat{x}_{10,10} = 31911.4 + 0.2\,(32175 - 31911.4) = 31964.1m$$
$$\hat{\dot{x}}_{10,10} = 34.4 + 0.1\left(\frac{32175 - 31911.4}{5}\right) = 39.67m/s$$
$$\hat{x}_{11,10} = 31964.1 + 5 \times 39.67 = 32162.45m$$
$$\hat{\dot{x}}_{11,10} = 39.67m/s$$

The following table summarizes our measurements and estimates.

| $n$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| $z_n$ | 30171 | 30353 | 30756 | 30799 | 31018 | 31278 | 31276 | 31379 | 31748 | 32175 |
| $\hat{x}_{n,n}$ | 30194.2 | 30383.64 | 30612.73 | 30818.93 | 31025.7 | 31242.3 | 31418.8 | 31566.3 | 31739.4 | 31964.1 |
| $\hat{\dot{x}}_{n,n}$ | 39.42 | 38.65 | 42.2 | 41.7 | 41.55 | 42.44 | 38.9 | 34.2 | 34.4 | 39.67 |
| $\hat{x}_{n+1,n}$ | 30391.3 | 30576.9 | 30823.9 | 31027.6 | 31233.4 | 31454.5 | 31613.15 | 31737.24 | 31911.4 | 32162.45 |
| $\hat{\dot{x}}_{n+1,n}$ | 39.42 | 38.65 | 42.2 | 41.7 | 41.55 | 42.44 | 38.9 | 34.2 | 34.4 | 39.67 |

**Table 3.3:** *Example 2 summary.*

### 3.2.4 Results analysis

The following chart compares the true values, measured values, and estimates.

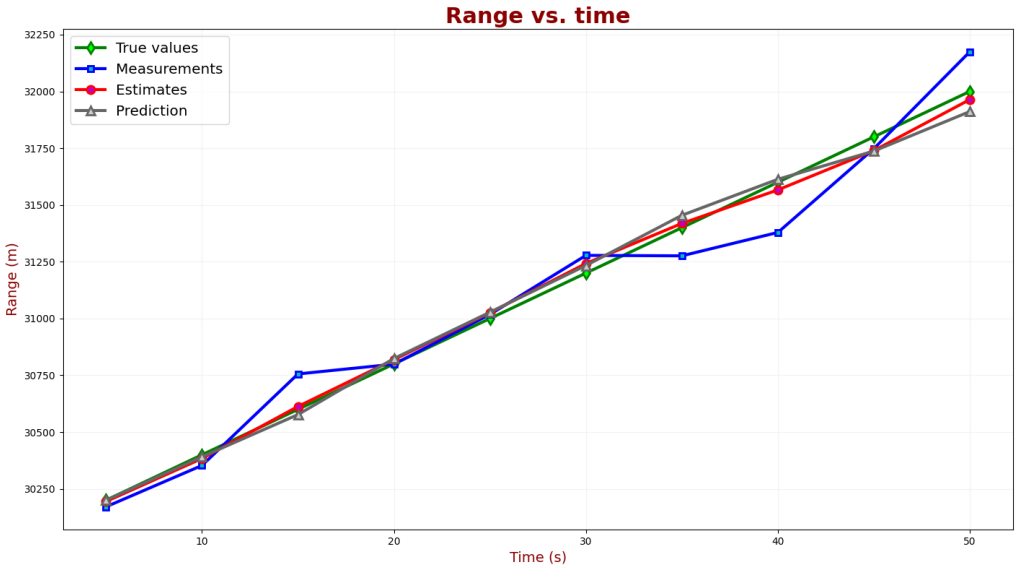


**Figure 3.9:** *Measurements vs. True value vs. Estimates - low Alpha and Beta.*

Our estimation algorithm has a smoothing effect on the measurements and converges toward the true value.

#### 3.2.4.1 Using high $\alpha$ and $\beta$

The following chart depicts the true, measured, and estimated values for $\alpha = 0.8$ and $\beta = 0.5$.

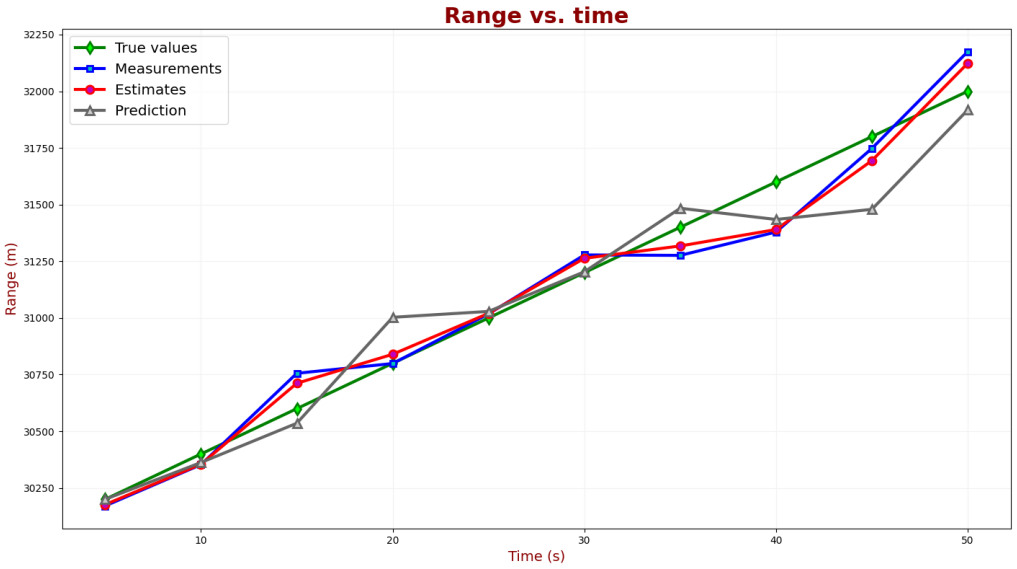


**Figure 3.10:** *Measurements vs. True value vs. Estimates - high Alpha and Beta.*

The "smoothing" degree of this filter is much lower. The "current estimate" is very

close to the measured values, and predicted estimate errors are relatively high.

So, shall we always choose low values for $\alpha$ and $\beta$?

The answer is NO. The value of $\alpha$ and $\beta$ should depend on the measurement precision. If we use high-precision equipment, like laser radar, we would prefer a high $\alpha$ and $\beta$ that follow measurements. In this case, the filter would quickly respond to a velocity change of the target. On the other hand, if measurement precision is low, we prefer low $\alpha$ and $\beta$. In this case, the filter smoothes the uncertainty (errors) in the measurements. However, the filter reaction to target velocity changes would be much slower.

### 3.2.5 Example summary

We've derived the $\alpha - \beta$ filter state update equation. We've also learned the State Extrapolation Equation. We've developed an estimation algorithm for a one-dimensional dynamic system based on the $\alpha - \beta$ filter and solved a numerical example for a constant velocity target.

## 3.3 Example 3 – Tracking accelerating aircraft

In this example, we track an aircraft moving with constant acceleration with the $\alpha - \beta$ filter.

In the previous example, we tracked a UAV moving at a constant velocity of 40m/s. The following chart depicts the target range and velocity vs. time.

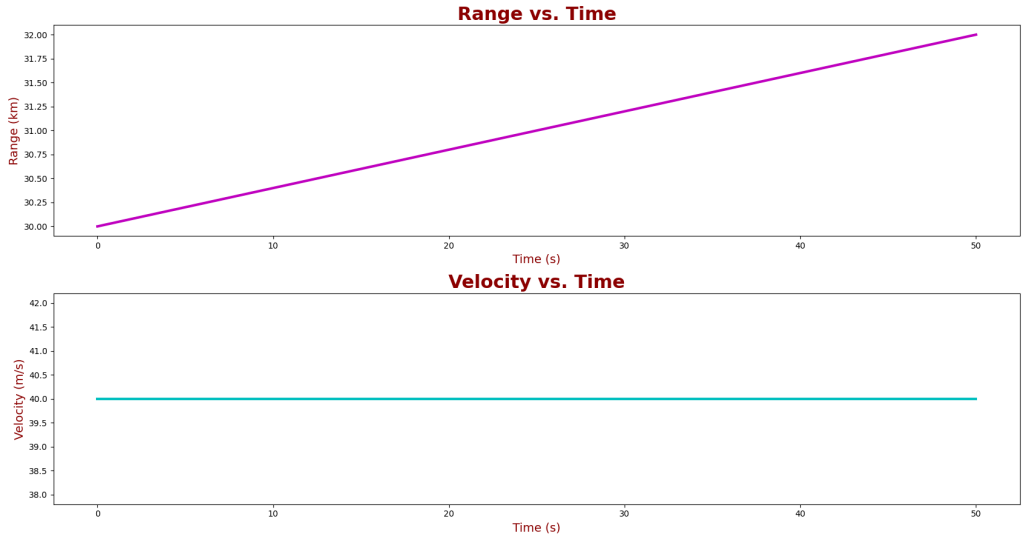


**Figure 3.11:** *Constant Velocity Movement.*

As you can see, the range function is linear.

Now let's examine a fighter aircraft. This aircraft moves at a constant velocity of 50m/s for 20 seconds. Then the aircraft accelerates with a constant acceleration of $8\text{m/s}^2$ for another 35 seconds.

The following chart depicts the target range, velocity and acceleration vs. time.

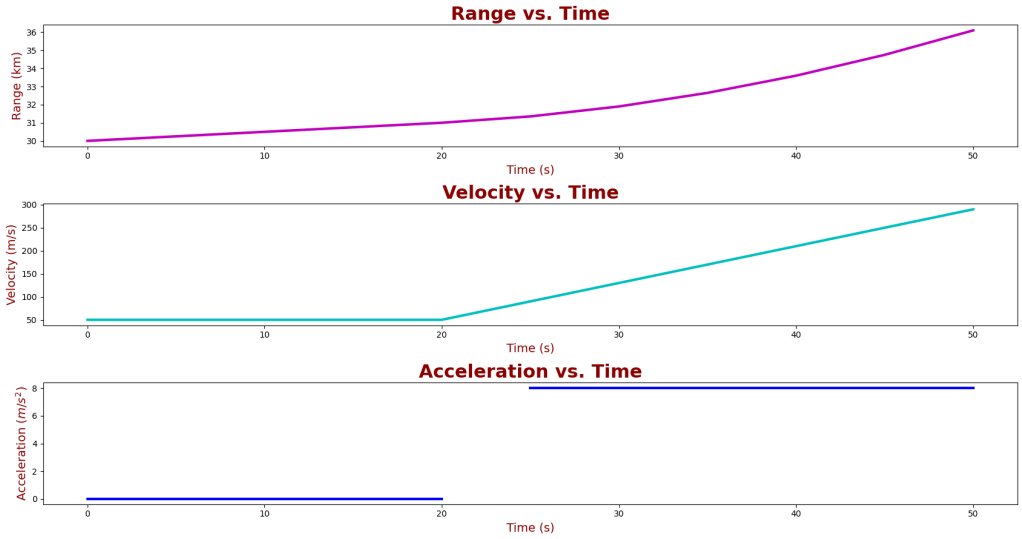


**Figure 3.12:** *Accelerated Movement.*

As you can see from the chart, the aircraft velocity is constant for the first 20 seconds and then grows linearly. The range grows linearly for the first 20 seconds and then grows quadratically. We are going to track this aircraft with the $\alpha - \beta$ filter that was used in the previous example.

## 3.3.1 The numerical example

Consider an aircraft moving radially toward (or away from) a radar in a one-dimensional world. The $\alpha - \beta$ filter parameters are:

- $\alpha = 0.2$.
- $\beta = 0.1$.

The track-to-track interval is 5 seconds.

### 3.3.1.1 Iteration Zero

#### Initialization

The initial conditions for the time $n = 0$ are given:

$\hat{x}_{0,0} = 30000m$

$\hat{\dot{x}}_{0,0} = 50m/s$

R The **Track Initiation** (or how we get the initial conditions) is an important topic that will be discussed in chapter 21. Right now, our goal is to understand the basic $\alpha - \beta$ filter operation, so let's assume that the initial conditions are given by somebody else.

#### Prediction

The initial guess should be extrapolated to the first cycle using the State Extrapolation Equations:

$\hat{x}_{n+1,n} = \hat{x}_{n,n} + \Delta t \hat{\dot{x}}_{n,n} \rightarrow \hat{x}_{1,0} = \hat{x}_{0,0} + \Delta t \hat{\dot{x}}_{0,0} = 30000 + 5 \times 50 = 30250m$

$\hat{\dot{x}}_{n+1,n} = \hat{\dot{x}}_{n,n} \rightarrow \hat{\dot{x}}_{1,0} = \hat{\dot{x}}_{0,0} = 50m/s$

### 3.3.1.2 Iterations 1-10

All filter iterations are summarized in the following table:

Table 3.4: Example 3 filter iterations.

| $n$ | $z_n$ | Current state estimates $(\hat{x}_{n,n}, \hat{\dot{x}}_{n,n})$ | Prediction $(\hat{x}_{n+1,n}, \hat{\dot{x}}_{n+1,n})$ |
|---|---|---|---|
| 1 | 30221m | $\hat{x}_{1,1} = 30250 + 0.2\,(30221 - 30250) = 30244.2m$<br><br>$\hat{\dot{x}}_{1,1} = 50 + 0.1\left(\frac{30221-30250}{5}\right) = 49.42m/s$ | $\hat{x}_{2,1} = 30244.2 + 5 \times 49.42 = 30491.3m$<br><br>$\hat{\dot{x}}_{2,1} = 49.42m/s$ |
| 2 | 30453m | $\hat{x}_{2,2} = 30491.3 + 0.2\,(30453 - 30491.3) = 30483.64m$<br><br>$\hat{\dot{x}}_{2,2} = 49.42 + 0.1\left(\frac{30453-30491.3}{5}\right) = 48.65m/s$ | $\hat{x}_{3,2} = 30483.64 + 5 \times 48.65 = 30726.9m$<br><br>$\hat{\dot{x}}_{3,2} = 48.65m/s$ |
| 3 | 30906m | $\hat{x}_{3,3} = 30726.9 + 0.2\,(30906 - 30726.9) = 30762.7m$<br><br>$\hat{\dot{x}}_{3,3} = 48.65 + 0.1\left(\frac{30906-30726.9}{5}\right) = 52.24m/s$ | $\hat{x}_{4,3} = 30762.7 + 5 \times 52.24 = 31023.9m$<br><br>$\hat{\dot{x}}_{4,3} = 52.24m/s$ |

Continued on next page

Table 3.4: Example 3 filter iterations. (Continued)

| | | | |
|---|---|---|---|
| 4 | 30999m | $\hat{x}_{4,4} = 31023.9 + 0.2\,(30999 - 31023.9) = 31018.93m$<br><br>$\hat{\dot{x}}_{4,4} = 52.24 + 0.1\left(\frac{30999-31023.9}{5}\right) = 51.74m/s$ | $\hat{x}_{5,4} = 31018.93 + 5 \times 51.74 = 31277.6m$<br><br>$\hat{\dot{x}}_{5,4} = 51.74m/s$ |
| 5 | 31368m | $\hat{x}_{5,5} = 31277.6 + 0.2\,(31368 - 31277.6) = 31295.7m$<br><br>$\hat{\dot{x}}_{5,5} = 51.74 + 0.1\left(\frac{31368-31277.6}{5}\right) = 53.55m/s$ | $\hat{x}_{6,5} = 31295.7 + 5 \times 53.55 = 31563.4m$<br><br>$\hat{\dot{x}}_{6,5} = 53.55m/s$ |
| 6 | 31978m | $\hat{x}_{6,6} = 31563.4 + 0.2\,(31978 - 31563.4) = 31646.3m$<br><br>$\hat{\dot{x}}_{6,6} = 53.55 + 0.1\left(\frac{31978-31563.4}{5}\right) = 61.84m/s$ | $\hat{x}_{7,6} = 31646.3 + 5 \times 61.84 = 31955.5m$<br><br>$\hat{\dot{x}}_{7,6} = 61.84m/s$ |
| 7 | 32526m | $\hat{x}_{7,7} = 31955.5 + 0.2\,(32526 - 31955.5) = 32069.6m$<br><br>$\hat{\dot{x}}_{7,7} = 61.84 + 0.1\left(\frac{32526-31955.5}{5}\right) = 73.25m/s$ | $\hat{x}_{8,7} = 32069.6 + 5 \times 73.25 = 32435.85m$<br><br>$\hat{\dot{x}}_{8,7} = 73.25m/s$ |
| 8 | 33379m | $\hat{x}_{8,8} = 32435.85 + 0.2\,(33379 - 32435.85) = 32624.5m$<br><br>$\hat{x}_{8,8} = 32435.85 + 0.2\,(33379 - 32435.85) = 32624.5m$ | $\hat{x}_{9,8} = 32624.5 + 5 \times 92.1 = 33085m$<br><br>$\hat{\dot{x}}_{9,8} = 92.1m/s$ |

Continued on next page

Table 3.4: Example 3 filter iterations. (Continued)

| | | | |
|---|---|---|---|
| 9 | 34698m | $\hat{x}_{9,9} = 33085 + 0.2\,(34698 - 33085) = 33407.6m$<br><br>$\hat{\dot{x}}_{9,9} = 92.1 + 0.1\left(\frac{34698-33085}{5}\right) = 124.37m/s$ | $\hat{x}_{10,9} = 33407.6 + 5 \times 124.37 = 34029.5m$<br><br>$\hat{\dot{x}}_{10,9} = 124.37m/s$ |
| 10 | 36275m | $\hat{x}_{10,10} = 34029.5 + 0.2\,(36275 - 34029.5) = 34478.6m$<br><br>$\hat{\dot{x}}_{10,10} = 124.37 + 0.1\left(\frac{36275-34029.5}{5}\right) = 169.28m/s$ | $\hat{x}_{11,10} = 34478.6 + 5 \times 169.28 = 35325m$<br><br>$\hat{\dot{x}}_{11,10} = 169.28m/s$ |

### 3.3.2 Results analysis

The following charts compare the true, measured, and estimated values for the range and velocity for the first 75 seconds.

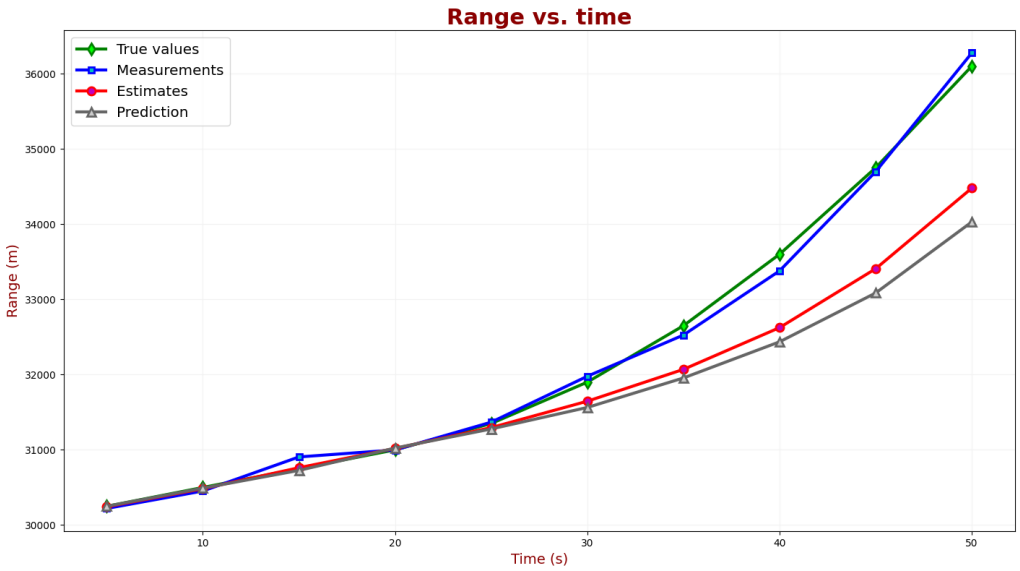


**Figure 3.13:** *Example 3 - range vs. time.*

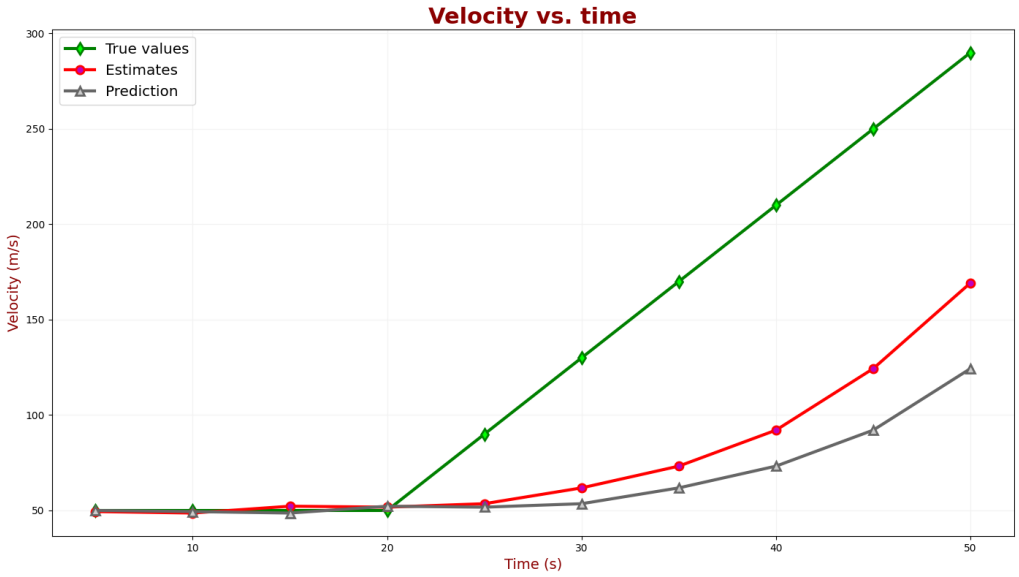


**Figure 3.14:** *Example 3 - velocity vs. time.*

You can see a constant gap between true or measured values and estimates. The gap is called a **lag error**. Other common names for the lag error are:

- **Dynamic error**
- **Systematic error**
- **Bias error**
- **Truncation error**

The lag error appears during the acceleration period. After the acceleration period, the filter closes the gap and converges toward the true value. However, a significant lag error can result in the target loss. The lag error is unacceptable in certain applications, such as missile guidance or air defense.

### 3.3.3 Example summary

We've examined the lag error caused by target acceleration.

## 3.4 Example 4 – Tracking accelerating aircraft using the $\alpha - \beta - \gamma$ filter

In this example, we track an aircraft using the $\alpha - \beta - \gamma$ filter. The aircraft is moving with a constant acceleration.

### 3.4.1 The $\alpha - \beta - \gamma$ filter

The $\alpha - \beta - \gamma$ **filter** (sometimes called **g-h-k filter**) considers a target acceleration. Thus, the State Extrapolation Equations become:

**The State Extrapolation Equations for position, velocity, and acceleration**

$$\begin{aligned}\hat{x}_{n+1,n} &= \hat{x}_{n,n} + \hat{\dot{x}}_{n,n}\Delta t + \hat{\ddot{x}}_{n,n}\frac{\Delta t^2}{2} \\ \hat{\dot{x}}_{n+1,n} &= \hat{\dot{x}}_{n,n} + \hat{\ddot{x}}_{n,n}\Delta t \\ \hat{\ddot{x}}_{n+1,n} &= \hat{\ddot{x}}_{n,n}\end{aligned} \tag{3.11}$$

Where $\ddot{x}_n$ is acceleration (the second derivative of $x$ ).

The State Update Equations become:

**The State Update Equations for position, velocity, and acceleration**

$$\begin{aligned}\hat{x}_{n,n} &= \hat{x}_{n,n-1} + \alpha\left(z_n - \hat{x}_{n,n-1}\right) \\ \hat{\dot{x}}_{n,n} &= \hat{\dot{x}}_{n,n-1} + \beta\left(\frac{z_n - \hat{x}_{n,n-1}}{\Delta t}\right) \\ \hat{\ddot{x}}_{n,n} &= \hat{\ddot{x}}_{n,n-1} + \gamma\left(\frac{z_n - \hat{x}_{n,n-1}}{0.5\Delta t^2}\right)\end{aligned} \tag{3.12}$$

### 3.4.2 The numerical example

Let's take the scenario from the previous example: an aircraft that moves with a constant velocity of 50m/s for 20 seconds and then accelerates with a constant acceleration of $8\text{m/s}^2$ for another 35 seconds.

The $\alpha$ - $\beta$ -$\gamma$ filter parameters are:

- $\alpha = 0.5$.

- $\beta = 0.4$.
- $\gamma = 0.1$.

The track-to-track interval is 5 seconds.

R We use an imprecise radar and a low-speed target for better graphical representation. The radars are usually more precise in real life, and the targets can be much faster.

#### 3.4.2.1 Iteration Zero

##### Initialization

The initial conditions for the time $n = 0$ are given:

$\hat{x}_{0,0} = 30000m$

$\hat{\dot{x}}_{0,0} = 50m/s$

$\hat{\ddot{x}}_{0,0} = 0m/s^2$

##### Prediction

The initial guess should be extrapolated to the first cycle using the State Extrapolation Equations:

$\hat{x}_{n+1,n} = \hat{x}_{n,n} + \hat{\dot{x}}_{n,n}\Delta t + 0.5\,\hat{\ddot{x}}_{n,n}\Delta t^2 \quad \rightarrow \hat{x}_{1,0} = \hat{x}_{0,0} + \hat{\dot{x}}_{0,0}\Delta t + 0.5\,\hat{\ddot{x}}_{0,0}\Delta t^2$

$= 30000 + 50 \times 5 + 0.5 \times 0 \times 5^2 = 30250m$

$\hat{\dot{x}}_{n+1,n} = \hat{\dot{x}}_{n,n} + \hat{\ddot{x}}_{n,n}\Delta t \;\rightarrow \hat{\dot{x}}_{1,0} = \hat{\dot{x}}_{0,0} + \hat{\ddot{x}}_{0,0}\Delta t = 50 + 0 \times 5 = 50m/s$

$\hat{\ddot{x}}_{n+1,n} = \hat{\ddot{x}}_{n,n} \rightarrow \;\hat{\ddot{x}}_{1,0} = \hat{\ddot{x}}_{0,0} = 0m/s^2$

#### 3.4.2.2 First Iteration

In the first cycle ($n = 1$), the initial guess is the prior estimate:

$\hat{x}_{n,n-1} = \hat{x}_{1,0} = 30250m$

$\hat{\dot{x}}_{n,n-1} = \hat{\dot{x}}_{1,0} = 50m/s$

$\hat{\ddot{x}}_{n,n-1} = \hat{\ddot{x}}_{1,0} = 0m/s^2$

##### Step 1

The radar measures the aircraft range:

$z_1 = 30221m$

#### Step 2

Calculating the current estimate using the State Update Equation:

$\hat{x}_{1,1} = \hat{x}_{1,0} + \alpha\left(z_1 - \hat{x}_{1,0}\right) = 30250 + 0.5\left(30221 - 30250\right) = 30235.5m$

$\hat{\dot{x}}_{1,1} = \hat{\dot{x}}_{1,0} + \beta\left(\frac{z_1 - \hat{x}_{1,0}}{\Delta t}\right) = 50 + 0.4\left(\frac{30221 - 30250}{5}\right) = 47.68m/s$

$\hat{\ddot{x}}_{1,1} = \hat{\ddot{x}}_{1,0} + \gamma\left(\frac{z_1 - \hat{x}_{1,0}}{0.5\Delta t^2}\right) = 0 + 0.1\left(\frac{30221 - 30250}{0.5 \times 5^2}\right) = -0.23m/s^2$

#### Step 3

Calculating the next state estimate using the State Extrapolation Equations:

$\hat{x}_{2,1} = \hat{x}_{1,1} + \hat{\dot{x}}_{1,1}\Delta t + 0.5\,\hat{\ddot{x}}_{1,1}\Delta t^2 = 30235.5 + 47.68 \times 5 + 0.5 \times (-0.23) \times 5^2 = 30471m$

$\hat{\dot{x}}_{2,1} = \hat{\dot{x}}_{1,1} + \hat{\ddot{x}}_{1,1}\Delta t = 47.68 + (-0.23) \times 5 = 46.52m/s$

$\hat{\ddot{x}}_{2,1} = \hat{\ddot{x}}_{1,1} = -0.23m/s^2$

### 3.4.2.3 Second Iteration

After a unit time delay, the predicted estimate from the foregoing iteration becomes the prior estimate in the current iteration.

$\hat{x}_{2,1} = 30471m$

$\hat{\dot{x}}_{2,1} = 46.52m/s$

$\hat{\ddot{x}}_{2,1} = -0.23m/s^2$

#### Step 1

The radar measures the aircraft range:

$z_2 = 30453m$

#### Step 2

Calculating the current estimate using the State Update Equation:

$\hat{x}_{2,2} = \hat{x}_{2,1} + \alpha\left(z_2 - \hat{x}_{2,1}\right) = 30471 + 0.5\left(30453 - 30471\right) = 30462m$

$\hat{\dot{x}}_{2,2} = \hat{\dot{x}}_{2,1} + \beta\left(\frac{z_2 - \hat{x}_{2,1}}{\Delta t}\right) = 46.52 + 0.4\left(\frac{30453 - 30471}{5}\right) = 45.08m/s$

$\hat{\ddot{x}}_{2,2} = \hat{\ddot{x}}_{2,1} + \gamma\left(\frac{z_2 - \hat{x}_{2,1}}{0.5\Delta t^2}\right) = -0.23 + 0.1\left(\frac{30453 - 30471}{0.5 \times 5^2}\right) = -0.38m/s^2$

#### Step 3

Calculating the next state estimate using the State Extrapolation Equations:

$$\hat{x}_{3,2} = \hat{x}_{2,2} + \hat{\dot{x}}_{2,2}\Delta t + 0.5\,\hat{\ddot{x}}_{2,2}\Delta t^2 = 30462 + 45.08 \times 5 + 0.5 \times (-0.38) \times 5^2 = 30682.7m$$
$$\hat{\dot{x}}_{3,2} = \hat{\dot{x}}_{2,2} + \hat{\ddot{x}}_{2,2}\Delta t = 45.08 + (-0.38) \times 5 = 43.2m/s$$
$$\hat{\ddot{x}}_{3,2} = \hat{\ddot{x}}_{2,2} = -0.38m/s^2$$

### 3.4.2.4 Iterations 3-10

The calculations for the successive iterations are summarized in the following table:

Table 3.5: Example 4 filter iterations.

| $n$ | $z_n$ | Current state estimates $(\hat{x}_{n,n}, \hat{\dot{x}}_{n,n}, \hat{\ddot{x}}_{n,n})$ | Prediction $(\hat{x}_{n+1,n}, \hat{\dot{x}}_{n+1,n}, \hat{\ddot{x}}_{n+1,n})$ |
|---|---|---|---|
| 3 | 30906m | $\hat{x}_{3,3} = 30682.7 + 0.5\,(30906 - 30682.7) = 30794.35m$<br>$\hat{\dot{x}}_{3,3} = 43.2 + 0.4\left(\frac{30906-30682.7}{5}\right) = 61.06m/s$<br>$\hat{\ddot{x}}_{3,3} = -0.38 + 0.1\left(\frac{30906-30682.7}{0.5\times5^2}\right) = 1.41m/s^2$ | $\hat{x}_{4,3} = 30794.35 + 5 \times 61.06 + 1.41 \times \frac{5^2}{2} = 31117.3m$<br>$\hat{\dot{x}}_{4,3} = 61.06 + 1.41 \times 5 = 68.1m/s$<br>$\hat{\ddot{x}}_{4,3} = 1.41m/s^2$ |
| 4 | 30999m | $\hat{x}_{4,4} = 31117.3 + 0.5\,(30999 - 31117.3) = 31058.15m$<br>$\hat{\dot{x}}_{4,4} = 68.1 + 0.4\left(\frac{30999-31117.3}{5}\right) = 58.65m/s$<br>$\hat{\ddot{x}}_{4,4} = 1.41 + 0.1\left(\frac{30999-31117.3}{0.5\times5^2}\right) = 0.46m/s^2$ | $\hat{x}_{5,4} = 31058.15 + 5 \times 58.65 + 0.46 \times \frac{5^2}{2} = 31357.2m$<br>$\hat{\dot{x}}_{5,4} = 58.65 + 0.46 \times 5 = 61m/s$<br>$\hat{\ddot{x}}_{5,4} = 0.46m/s^2$ |

Continued on next page

Table 3.5: Example 4 filter iterations. (Continued)

| | | | |
|---|---|---|---|
| 5 | 31368m | $\hat{x}_{5,5} = 31357.2 + 0.5\,(31368 - 31357.2) = 31362.6m$<br><br>$\hat{\dot{x}}_{5,5} = 61 + 0.4\left(\frac{31368-31357.2}{5}\right) = 61.8m/s$<br><br>$\hat{\ddot{x}}_{5,5} = 0.46 + 0.1\left(\frac{31368-31357.2}{0.5\times5^2}\right) = 0.55m/s^2$ | $\hat{x}_{6,5} = 31362.6 + 5 \times 61.8 + 0.55 \times \frac{5^2}{2} = 31678.7m$<br><br>$\hat{\dot{x}}_{6,5} = 61.8+0.55\times5 = 64.6m/s$<br><br>$\hat{\ddot{x}}_{6,5} = 0.55m/s^2$ |
| 6 | 31978m | $\hat{x}_{6,6} = 31678.7 + 0.5\,(31978 - 31678.7) = 31828.3m$<br><br>$\hat{\dot{x}}_{6,6} = 64.6 + 0.4\left(\frac{31978-31678.7}{5}\right) = 88.5m/s$<br><br>$\hat{\ddot{x}}_{6,6} = 0.55 + 0.1\left(\frac{31978-31678.7}{0.5\times5^2}\right) = 2.95m/s^2$ | $\hat{x}_{7,6} = 31828.3 + 5 \times 88.5 + 2.95 \times \frac{5^2}{2} = 32307.8m$<br><br>$\hat{\dot{x}}_{7,6} = 88.5 + 2.95 \times 5 = 103.26m/s$<br><br>$\hat{\ddot{x}}_{7,6} = 2.95m/s^2$ |
| 7 | 32526m | $\hat{x}_{7,7} = 32307.8 + 0.5\,(32526 - 32307.8) = 32416.9m$<br><br>$\hat{\dot{x}}_{7,7} = 103.26 + 0.4\left(\frac{32526-32307.8}{5}\right) = 120.7m/s$<br><br>$\hat{\ddot{x}}_{7,7} = 2.95 + 0.1\left(\frac{32526-32307.8}{0.5\times5^2}\right) = 4.7m/s^2$ | $\hat{x}_{8,7} = 32416.9 + 5 \times 120.7 + 4.7 \times \frac{5^2}{2} = 33079.1m$<br><br>$\hat{\dot{x}}_{8,7} = 120.7 + 4.7 \times 5 = 144.17m/s$<br><br>$\hat{\ddot{x}}_{8,7} = 4.7m/s^2$ |

Continued on next page

Table 3.5: Example 4 filter iterations. (Continued)

| | | | |
|---|---|---|---|
| 8 | 33379m | $\hat{x}_{8,8} = 33079.1 + 0.5\,(33379 - 33079.1) = 33229.05m$<br><br>$\hat{\dot{x}}_{8,8} = 144.17 + 0.4\left(\frac{33379-33079.1}{5}\right) = 168.2m/s$<br><br>$\hat{\ddot{x}}_{8,8} = 4.7 + 0.1\left(\frac{33379-33079.1}{0.5\times5^2}\right) = 7.1m/s^2$ | $\hat{x}_{9,8} = 33229.05 + 5 \times 168.2 + 7.1 \times \frac{5^2}{2} = 34158.5m$<br><br>$\hat{\dot{x}}_{9,8} = 168.2 + 7.1 \times 5 = 203.6m/s$<br><br>$\hat{\ddot{x}}_{9,8} = 7.1m/s^2$ |
| 9 | 34698m | $\hat{x}_{9,9} = 34158.5 + 0.5\,(34698 - 34158.5) = 34428.2m$<br><br>$\hat{\dot{x}}_{9,9} = 203.6 + 0.4\left(\frac{34698-34158.5}{5}\right) = 246.8m/s$<br><br>$\hat{\ddot{x}}_{9,9} = 7.1 + 0.1\left(\frac{34698-34158.5}{0.5\times5^2}\right) = 11.4m/s^2$ | $\hat{x}_{10,9} = 34428.2 + 5 \times 246.8 + 11.4 \times \frac{5^2}{2} = 35804.7m$<br><br>$\hat{\dot{x}}_{10,9} = 246.8 + 11.4 \times 5 = 303.8m/s$<br><br>$\hat{\ddot{x}}_{10,9} = 11.4m/s^2$ |
| 10 | 36275m | $\hat{x}_{10,10} = 35804.7 + 0.5\,(36275 - 35804.7) = 36039.8m$<br><br>$\hat{\dot{x}}_{10,10} = 303.8 + 0.4\left(\frac{36275-35804.7}{5}\right) = 341.4m/s$<br><br>$\hat{\ddot{x}}_{10,10} = 11.4 + 0.1\left(\frac{36275-35804.7}{0.5\times5^2}\right) = 15.2m/s^2$ | $\hat{x}_{11,10} = 36039.8 + 5 \times 341.4 + 15.2 \times \frac{5^2}{2} = 37936.6m$<br><br>$\hat{\dot{x}}_{11,10} = 341.4 + 15.2 \times 5 = 417.3m/s$<br><br>$\hat{\ddot{x}}_{11,10} = 15.2m/s^2$ |

### 3.4.3 Results analysis

The following charts compare the true, measured, and estimated values for the range, velocity, and acceleration for the first 50 seconds.

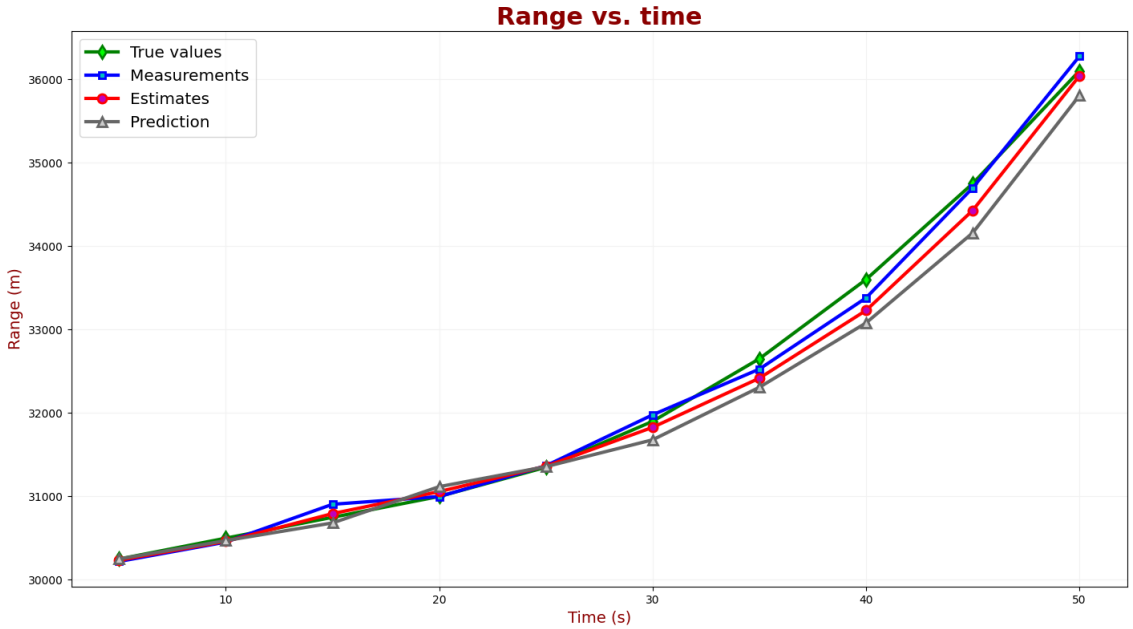


**Figure 3.15:** *Example 4: Range vs. Time.*

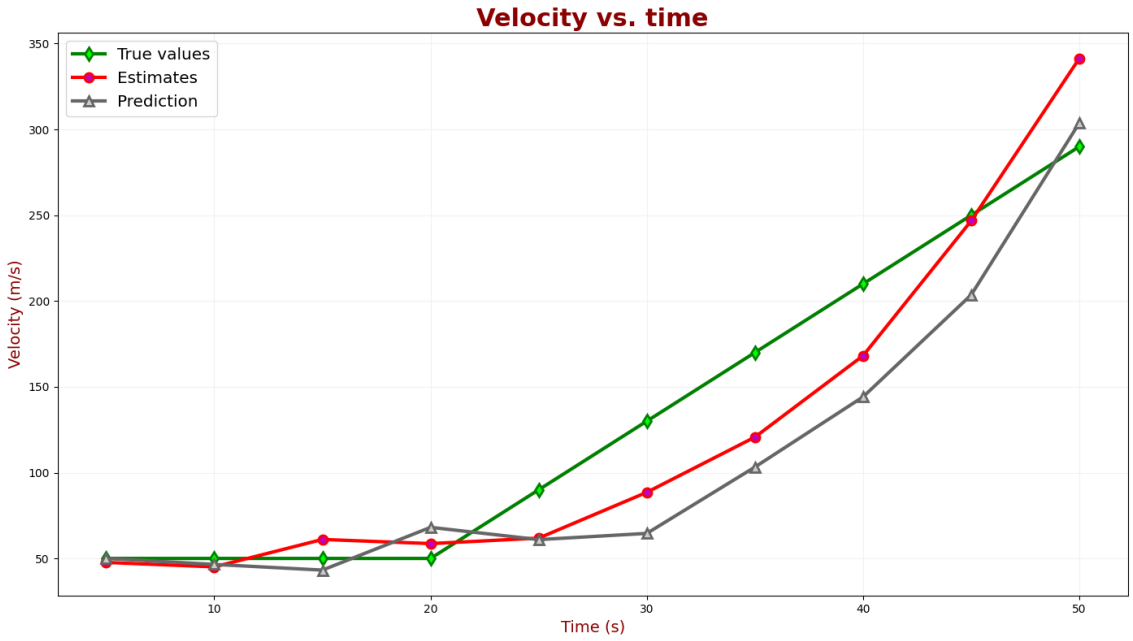


**Figure 3.16:** *Example 4: Velocity vs. Time.*

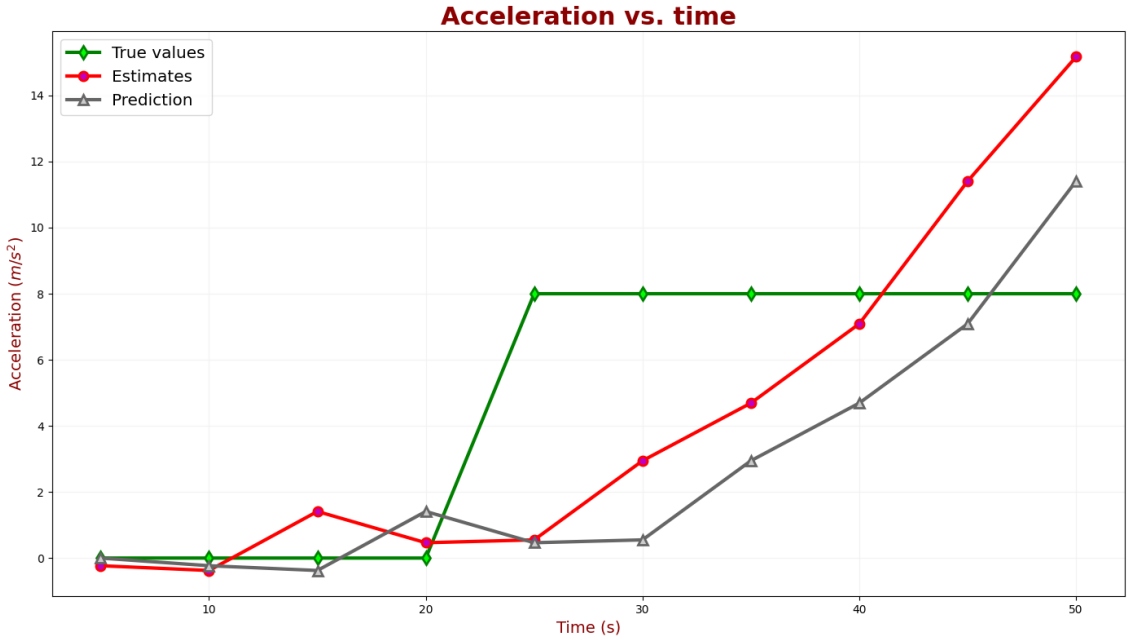


**Figure 3.17:** *Example 4: Acceleration vs. Time.*

## 3.5 Summary of the $\alpha - \beta - (\gamma)$ filter

There are many types of $\alpha - \beta - (\gamma)$ filters, and they are based on the same principle:

- The current state estimation is based on the state update equations.
- The following state estimation (prediction) is based on the dynamic model equations.

The main difference between these filters is the selection of weighting coefficients $\alpha - \beta - (\gamma)$. Some filter types use constant weighting coefficients; others compute weighting coefficients for every filter iteration (cycle).

The choice of the $\alpha$, $\beta$ and $\gamma$ is crucial for proper functionality of the estimation algorithm.

What should be the $\alpha - \beta - (\gamma)$ parameters?

I described the $\alpha - \beta - (\gamma)$ filter as an introductory to the Kalman Filter, thus, I won't cover this topic. The curious reader can find many books and papers on this topic [2], [3].

Another important issue is the initiation of the filter (chapter 21), i.e., providing the initial value for the first filter iteration.

The following list includes the most popular $\alpha - \beta - (\gamma)$ filters:

- Wiener Filter
- Bayes Filter
- Fading-memory polynomial Filter
- Expanding-memory (or growing-memory) polynomial Filter
- Least-squares Filter
- Benedict–Bordner Filter
- Discounted least-squares $\alpha - \beta$ Filter
- Critically damped $\alpha - \beta$ Filter
- Growing-memory Filter
- Kalman Filter
- Extended Complex Kalman Filter
- Gauss-Hermite Kalman Filter
- Cubature Kalman Filter
- Particle Filter

I hope to write about some of these filters. But this book is about the Kalman Filter, which is the topic of the following examples.

# 4. Kalman Filter in one dimension

In this chapter, we derive the Kalman Filter in one dimension. The main goal of this chapter is to explain the Kalman Filter concept simply and intuitively without using math tools that may seem complex and confusing.

We are going to advance toward the Kalman Filter equations step by step.

In this chapter, we derive the Kalman Filter equations without process noise. In the following chapter, we add process noise.

## 4.1 One-dimensional Kalman Filter without process noise

As I mentioned earlier, the Kalman Filter is based on five equations. We are already familiar with two of them:

- The state update equation
- The dynamic model equation

In this chapter, we derive another three Kalman Filter Equations and revise the state update equation.

Like the $\alpha - \beta - (\gamma)$ filter, the Kalman filter utilizes the "Measure, Update, Predict" algorithm.

Contrary to the $\alpha - \beta - (\gamma)$ filter, the Kalman Filter treats measurements, current state estimation, and next state estimation (predictions) as normally distributed **random variables**. The random variable is described by mean and variance.

The following chart provides a low-level schematic description of the Kalman Filter algorithm:

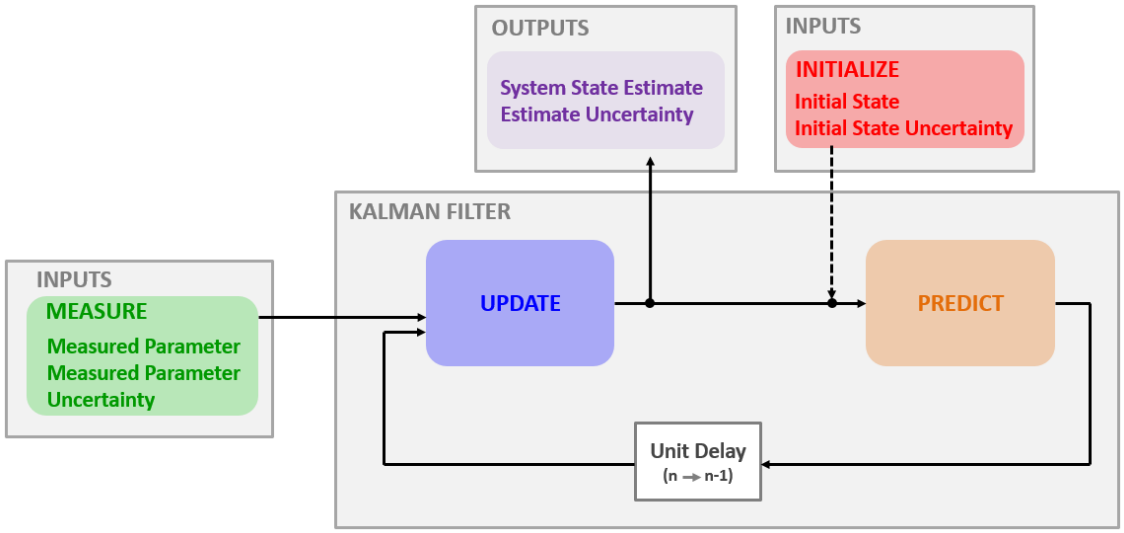


**Figure 4.1:** *Schematic description of the Kalman Filter algorithm.*

Let's recall our first example (Example 1 – Weighting the gold); We made multiple measurements and computed the estimate by averaging.

We obtained the following result:

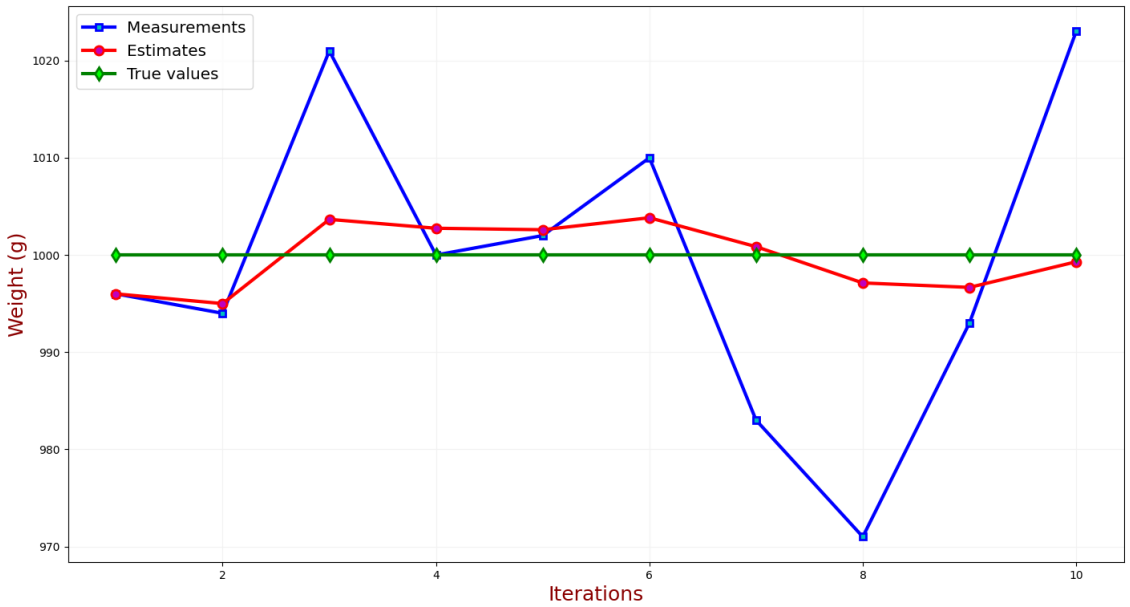


**Figure 4.2:** *Example 1: Measurements vs. True value vs. Estimates.*

Figure 4.2 shows the true, measured, and estimated values vs. the number of measurements.

### 4.1.1 Estimate as a random variable

The difference between the estimates (the red line) and the true values (the green line) is the **estimate error**. As you can see, the estimate error becomes lower as we make additional measurements, and it converges towards zero, while the estimated value converges towards the true value. We don't know the estimate error, but we can estimate the state **uncertainty**.

We denote the state estimate variance by $p$.

### 4.1.2 Measurement as a random variable

The measurement errors are the differences between the measurements (blue samples) and the true values (green line). Since the measurement errors are random, we can describe them by variance ($\sigma^2$). The standard deviation ($\sigma$) of the measurement errors is the **measurement uncertainty**.

R In some literature, the measurement uncertainty is also called the **measurement error**.

We denote the measurement variance by $r$.

The variance of the measurement errors could be provided by the measurement equipment vendor, calculated, or derived empirically by a calibration procedure.

For example, when using scales, we can calibrate the scales by making multiple measurements of an item with a known weight and empirically derive the standard deviation. The scales vendor can also supply the measurement uncertainty parameter.

For advanced sensors like radar, the measurement uncertainty depends on several parameters such as Signal to Noise Ratio (SNR), beam width, bandwidth, time on target, clock stability, and more. Every radar measurement has a different SNR, beam width, and time on target. Therefore, the radar calculates the uncertainty of each measurement and reports it to the tracker.

Let's look at the weight measurements Probability Density Function (PDF). The following plot shows ten measurements of the gold bar weight.

- The blue circles describe the measurements.
- The true values are described by the red dashed line.
- The green line describes the probability density function of the measurement.
- The bold green area is the standard deviation ($\sigma$) of the measurement, i.e., there is a probability of 68.26% that the measurement value lies within this area.

As you can see, 7 out of 10 measurements are within $1\sigma$ boundaries.

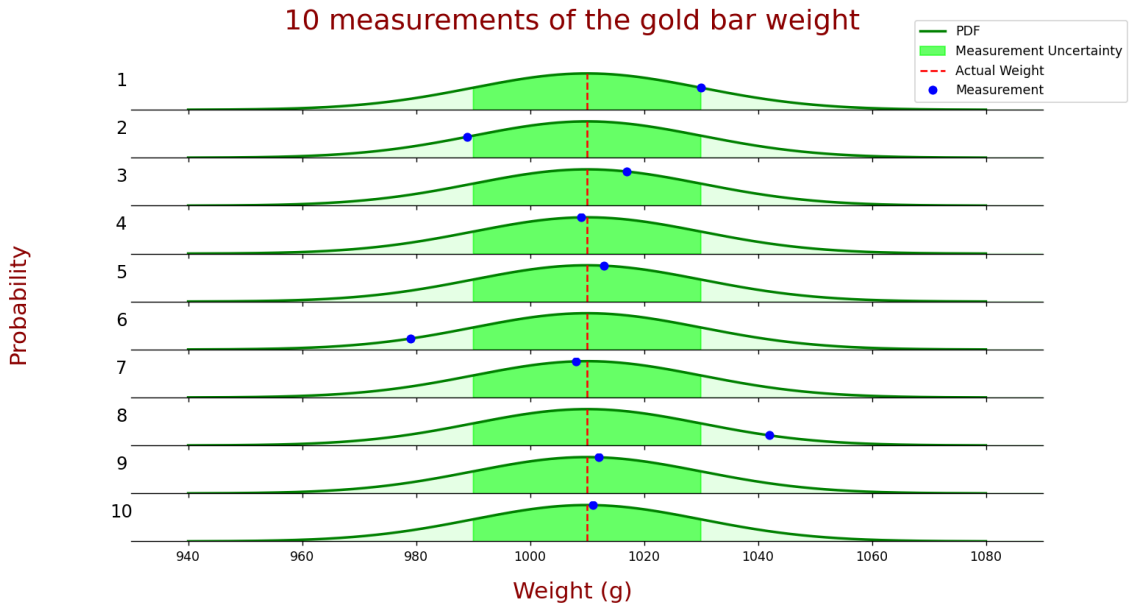


**Figure 4.3:** *Measurements Probability Density Function.*

### 4.1.3 State prediction

In the first example - "The measurement of the gold bar weight" (section 3.1), the dynamic model is constant:

$$\hat{x}_{n+1,n} = \hat{x}_{n,n} \tag{4.1}$$

In the second example - "Tracking the constant velocity aircraft" (section 3.2), we extrapolated the current state (target position and velocity) to the next state using motion equations:

$$\begin{aligned} \hat{x}_{n+1,n} &= \hat{x}_{n,n} + \Delta t \hat{\dot{x}}_{n,n} \\ \hat{\dot{x}}_{n+1,n} &= \hat{\dot{x}}_{n,n} \end{aligned} \tag{4.2}$$

i.e., the predicted position equals the current estimated position plus the currently estimated velocity multiplied by the time. The predicted velocity equals the current velocity estimate (assuming a constant velocity model).

The dynamic model equation depends on the system.

Since Kalman Filter treats the estimate as a random variable, we must also extrapolate the estimation variance ($p_{n,n}$) to the next state.

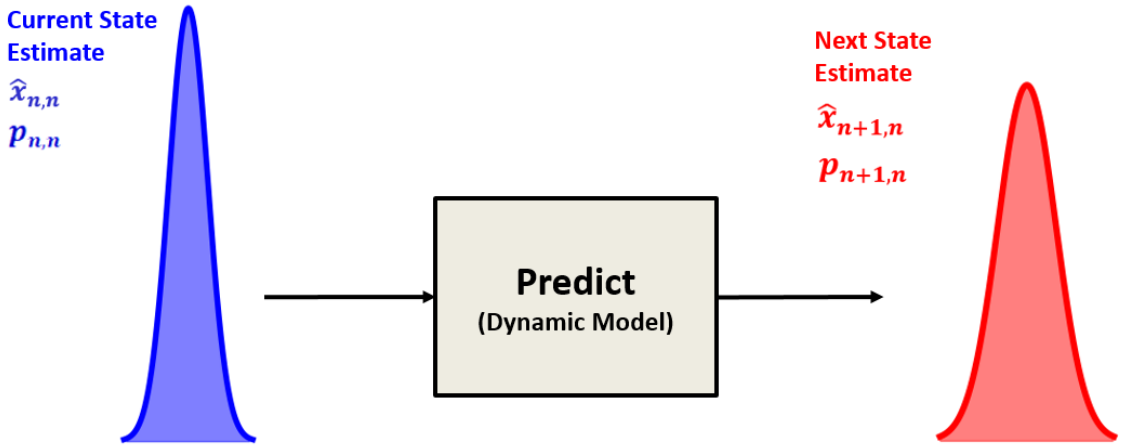


**Figure 4.4:** *State Prediction Illustration.*

In the first example - "The measurement of the gold bar weight" (section 3.1), the dynamic model of the system is constant. Thus, the estimate uncertainty extrapolation would be:

**The estimate uncertainty extrapolation for a constant dynamics**

$$\hat{p}_{n+1,n} = \hat{p}_{n,n} \tag{4.3}$$

Where $p$ is the estimate variance of the gold bar weight.

In the second example - "Tracking the constant velocity aircraft" (section 3.2), the estimate uncertainty extrapolation would be:

**The estimate uncertainty extrapolation for a constant velocity dynamics**

$$\begin{aligned} p^x_{n+1,n} &= p^x_{n,n} + \Delta t^2 \cdot p^v_{n,n} \\ p^v_{n+1,n} &= p^v_{n,n} \end{aligned} \tag{4.4}$$

Where:
$p^x$ is the position estimate variance.
$p^v$ is the velocity estimate variance.

i.e., the predicted position estimate variance equals the current position estimate variance plus the current velocity estimate variance multiplied by the time squared. The predicted velocity estimate variance equals the current velocity estimate variance

(assuming a constant velocity model).

Note that for any normally distributed random variable $x$ with variance $\sigma^2$, $kx$ is distributed normally with variance $k^2\sigma^2$, therefore the time term in the uncertainty extrapolation equation is squared. You can find a detailed explanation in Appendix A.

The estimate uncertainty extrapolation equation is called the Covariance Extrapolation Equation, which is the third Kalman Filter equation. Why covariance? We will see this in Part II - Multivariate Kalman Filter.

### 4.1.4 State update

To estimate the current state of the system, we combine two random variables:

- The prior state estimate (the current state estimate predicted at the previous state).
- The measurement.

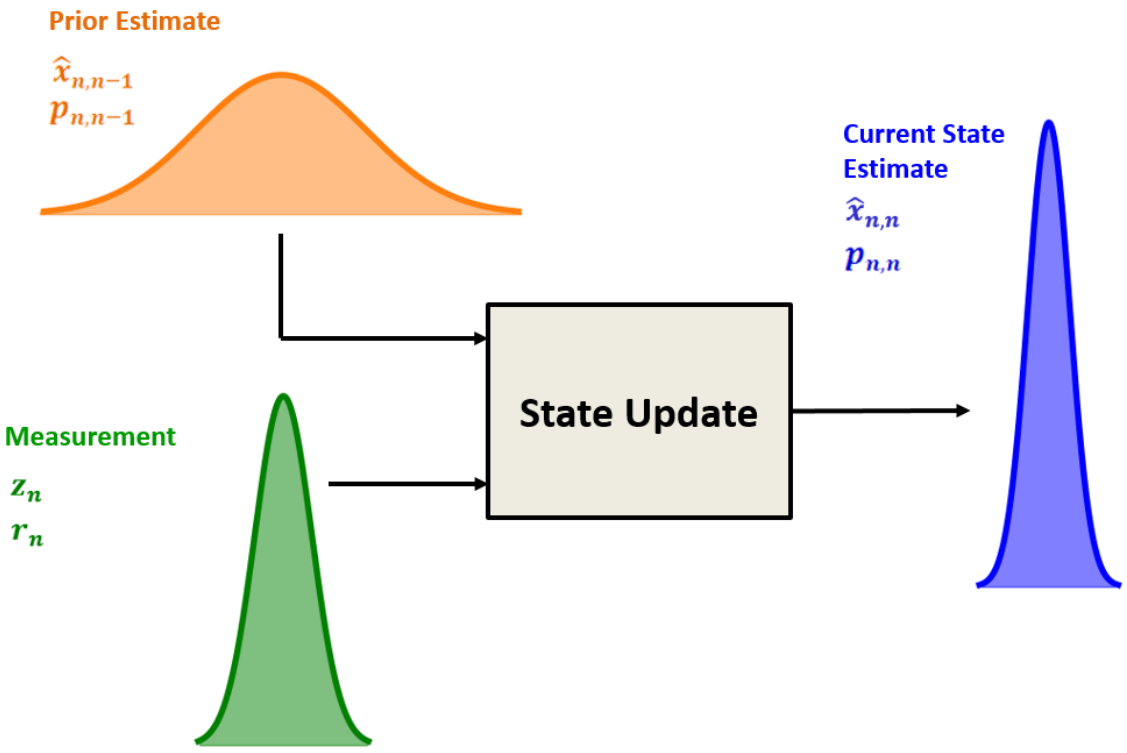


**Figure 4.5:** *State Update Illustration.*

The Kalman Filter is an **optimal filter**. It combines the prior state estimate with the measurement in a way that minimizes the uncertainty of the current state estimate.

The current state estimate is a weighted mean of the measurement and the prior

state estimate:

$$\begin{aligned} \hat{x}_{n,n} &= w_1 z_n + w_2 \hat{x}_{n,n-1} \\ w_1 + w_2 &= 1 \end{aligned} \tag{4.5}$$

Where $w_1$ and $w_2$ are the weights of the measurement and the prior state estimate.

We can write $\hat{x}_{n,n}$ as follows:

$$\hat{x}_{n,n} = w_1 z_n + (1 - w_1)\hat{x}_{n,n-1} \tag{4.6}$$

The relation between variances is given by:

$$p_{n,n} = w_1^2 r_n + (1 - w_1)^2 p_{n,n-1} \tag{4.7}$$

Where:

| | |
|---|---|
| $p_{n,n}$ | is the variance of the optimum combined estimate |
| $p_{n,n-1}$ | is the variance of the prior estimate $\hat{x}_{n,n-1}$ |
| $r_n$ | is the variance of the measurement $z_n$ |

Remember that for any normally distributed random variable $x$ with variance $\sigma^2$, $kx$ is distributed normally with variance $k^2\sigma^2$. You can find a detailed explanation in Appendix A.

Since we are looking for an **optimum** estimate, we want to minimize $p_{n,n}$.

To find $w_1$ that minimizes $p_{n,n}$, we differentiate $p_{n,n}$ with respect to $w_1$ and set the result to zero.

$$\frac{dp_{n,n}}{dw_1} = 2w_1 r_n - 2(1 - w_1)p_{n,n-1} = 0 \tag{4.8}$$

Hence

$$\begin{aligned} w_1 r_n &= p_{n,n-1} - w_1 p_{n,n-1} \\ w_1 p_{n,n-1} + w_1 r_n &= p_{n,n-1} \\ w_1 &= \frac{p_{n,n-1}}{p_{n,n-1} + r_n} \end{aligned} \tag{4.9}$$

Let us substitute the result into the current state estimation $\hat{x}_{n,n}$

$$\hat{x}_{n,n} = w_1 z_n + (1 - w_1)\hat{x}_{n,n-1} = w_1 z_n + \hat{x}_{n,n-1} - w_1 \hat{x}_{n,n-1} = \hat{x}_{n,n-1} + w_1 \left(z_n - \hat{x}_{n,n-1}\right) \tag{4.10}$$

**State Update Equation**

$$\hat{x}_{n,n} = \hat{x}_{n,n-1} + \frac{p_{n,n-1}}{p_{n,n-1} + r_n}\left(z_n - \hat{x}_{n,n-1}\right) \tag{4.11}$$

We have derived an equation that looks similar to the $\alpha - \beta - (\gamma)$ filter state update equation (Equation 3.2). The weight of the innovation is called the **Kalman Gain** (denoted by $K_n$).

The **Kalman Gain Equation** is the fourth Kalman Filter equation. In one dimension, the Kalman Gain Equation is the following:

**State Update Equation**

$$K_n = \frac{Variance \quad in \quad Estimate}{Variance \quad in \quad Estimate \quad + \quad Variance \quad in \quad Measurement}$$

$$K_n = \frac{p_{n,n-1}}{p_{n,n-1} + r_n} \tag{4.12}$$

Where:

$p_{n,n-1}$ is the extrapolated estimate variance
$r_n$ is the measurement variance

The Kalman Gain is a number between zero and one:

$$0 \leq K_n \leq 1 \tag{4.13}$$

Finally, we need to find the variance of the current state estimate. We've seen that the relation between variances is given by:

$$p_{n,n} = w_1^2 r_n + (1 - w_1)^2 \, p_{n,n-1} \tag{4.14}$$

The weight $w_1$ is a Kalman Gain:

$$p_{n,n} = K_n^2 r_n + (1 - K_n)^2 \, p_{n,n-1} \tag{4.15}$$

Let us find the $(1 - K_n)$ term:

$$(1 - K_n) = \left(1 - \frac{p_{n,n-1}}{p_{n,n-1} + r_n}\right) = \left(\frac{p_{n,n-1} + r_n - p_{n,n-1}}{p_{n,n-1} + r_n}\right) = \left(\frac{r_n}{p_{n,n-1} + r_n}\right) \tag{4.16}$$

| **Equation** | **Notes** |
|---|---|
| $p_{n,n} = \left(\frac{p_{n,n-1}}{p_{n,n-1} + r_n}\right)^2 r_n + \left(\frac{r_n}{p_{n,n-1} + r_n}\right)^2 p_{n,n-1}$ | Substitute $K_n$ and $(1 - K_n)$ |
| $= \frac{p_{n,n-1}^2 r_n}{(p_{n,n-1} + r_n)^2} + \frac{r_n^2 p_{n,n-1}}{(p_{n,n-1} + r_n)^2}$ | Expand |
| $= \frac{p_{n,n-1} r_n}{p_{n,n-1} + r_n}\left(\frac{p_{n,n-1}}{p_{n,n-1} + r_n} + \frac{r_n}{p_{n,n-1} + r_n}\right)$ | |
| $= (1 - K_n) \, p_{n,n-1}\Big(K_n + (1 - K_n)\Big)$ | Substitute $K_n$ and $(1 - K_n)$ |
| $= (1 - K_n) \, p_{n,n-1}$ | |

**Table 4.1:** *Covariance update equation derivation.*

This equation updates the estimate variance of the current state. It is called the **Covariance Update Equation**.

**Covariance Update Equation**

$$p_{n,n} = (1 - K_n)\, p_{n,n-1} \tag{4.17}$$

It is clear from the equation that the estimate uncertainty is constantly decreasing with each filter iteration, since $(1 - K_n) \leq 1$. When the measurement uncertainty is high, the Kalman gain is low. Therefore, the convergence of the estimate uncertainty would be slow. On the other hand, the Kalman gain is high when the measurement uncertainty is low. Therefore, the estimate uncertainty would quickly converge toward zero.

So, it is up to us to decide how many measurements to make. If we are measuring a building height, and we are interested in a precision of 3 centimeters ($\sigma$), we should make measurements until the state estimate variance ($\sigma^2$) is less than 9 $centimeters^2$.

### 4.1.5 Putting all together

This section combines all of these pieces into a single algorithm.

The filter inputs are:

- Initialization
  The initialization is performed only once, and it provides two parameters:
  - Initial System State ($\hat{x}_{0,0}$)
  - Initial State Variance ($p_{0,0}$)

  The initialization parameters can be provided by another system, another process (for instance, a search process in radar), or an educated guess based on experience or theoretical knowledge. Even if the initialization parameters are not precise, the Kalman filter can converge close to the true value.

- Measurement
  The measurement is performed for every filter cycle, and it provides two parameters:
  - Measured System State ($z_n$)
  - Measurement Variance ($r_n$)

The filter outputs are:

- System State Estimate ($\hat{x}_{n,n}$)
- Estimate Variance ($p_{n,n}$)

The following table summarizes the five Kalman Filter equations.

| | **Equation** | **Equation Name** | **Alternative names used in the literature** |
|---|---|---|---|
| State Update | $\hat{x}_{n,n} = \hat{x}_{n,n-1} + K_n(z_n - \hat{x}_{n,n})$ | State Update | Filtering Equation |
| | $p_{n,n} = (1 - K_n)p_{n,n-1}$ | Covariance Update | Corrector Equation |
| | $K_n = \frac{p_{n,n-1}}{p_{n,n-1} + r_n}$ | Kalman Gain | Weight Equation |
| State Predict | $\hat{x}_{n+1,n} = \hat{x}_{n,n}$ (for constant dynamics)<br>$\hat{x}_{n+1,n} = \hat{x}_{n,n} + \Delta t\hat{\dot{x}}_{n,n}$<br>$\hat{\dot{x}}_{n+1,n} = \hat{\dot{x}}_{n,n}$ (for constant velocity dynamics) | State Extrapolation | Predictor Equation<br>Transition Equation<br>Prediction Equation<br>Dynamic Model<br>State Space Model |
| | $p^x_{n+1,n} = p^x_{n,n}$ (for constant dynamics)<br>$p^x_{n+1,n} = p^x_{n,n} + \Delta t^2 p^v_{n,n}$<br>$p^v_{n+1,n} = p^v_{n,n}$ (for constant velocity dynamics) | Covariance Extrapolation | Predictor Covariance Equation |

**Table 4.2:** *Kalman Filter equations in one dimension.*

R The equations above don't include the process noise. In the following chapter, we add process noise.

R The State Extrapolation Equation and the Covariance Extrapolation Equation depend on the system dynamics.

R The table above demonstrates a special form of the Kalman Filter equations tailored for a specific case. The general form of the equation in matrix notation is presented in Part II.

The following figure provides a detailed description of the Kalman Filter's block diagram.

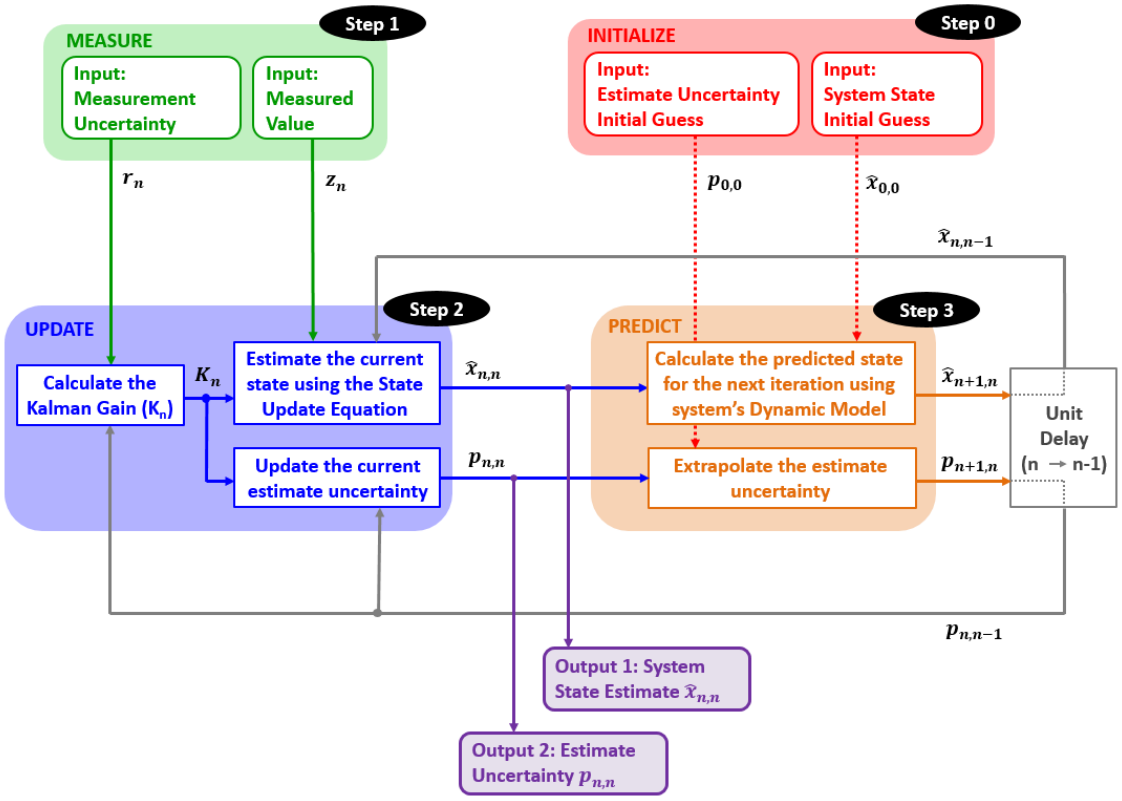


**Figure 4.6:** *Detailed description of the Kalman Filter algorithm.*

- Step 0: Initialization

  As mentioned above, the initialization is performed only once, and it provides two parameters:
  - Initial System State ($\hat{x}_{0,0}$)
  - Initial State Variance ($p_{0,0}$)

  The initialization is followed by prediction.
- Step 1: Measurement

  The measurement process provides two parameters:
  - Measured System State ($z_n$)
  - Measurement Variance ($r_n$)
- Step 2: State Update

  The state update process is responsible for the state estimation of the current state of the system.

  The state update process inputs are:
  - Measured Value ($z_n$)
  - A Measurement Variance ($r_n$)
  - A prior Predicted System State Estimate ($\hat{x}_{n,n-1}$)
  - A prior Predicted System State Estimate Variance ($p_{n,n-1}$)

  Based on the inputs, the state update process calculates the Kalman Gain and provides two outputs:

  - Current System State Estimate ($\hat{x}_{n,n}$)
  - Current State Estimate Variance ($p_{n,n}$)

  These parameters are the Kalman Filter outputs.
- Step 3: Prediction

  The prediction process extrapolates the current system state estimate and its variance to the next system state based on the dynamic model of the system. At the first filter iteration, the initialization is treated as the Prior State Estimate and Variance. The prediction outputs are used as the Prior (predicted) State Estimate and Variance on the following filter iterations.

### 4.1.6 Kalman Gain intuition

Let's rewrite the state update equation:

$$\hat{x}_{n,n} = \hat{x}_{n,n-1} + K_n \left( z_n - \hat{x}_{n,n-1} \right) = \left( 1 - K_n \right) \hat{x}_{n,n-1} + K_n z_n \tag{4.18}$$

As you can see, the Kalman Gain ($K_n$) is the measurement weight, and the ($1 - K_n$) term is the weight of the current state estimate.

The Kalman Gain is close to zero when the measurement uncertainty is high and the estimate uncertainty is low. Hence we give a significant weight to the estimate and a small weight to the measurement.

On the other hand, when the measurement uncertainty is low, and the estimate uncertainty is high, the Kalman Gain is close to one. Hence we give a low weight to the estimate and a significant weight to the measurement.

If the measurement uncertainty equals the estimate uncertainty, then the Kalman gain equals 0.5.

The Kalman Gain Defines the measurement's weight and the prior estimate's weight when forming a new estimate. It tells us how much the measurement changes the estimate.

#### 4.1.6.1 High Kalman Gain

A low measurement uncertainty relative to the estimate uncertainty would result in a high Kalman Gain (close to 1). Therefore the new estimate would be close to the measurement. The following figure illustrates the influence of a high Kalman Gain on the estimate in an aircraft tracking application.

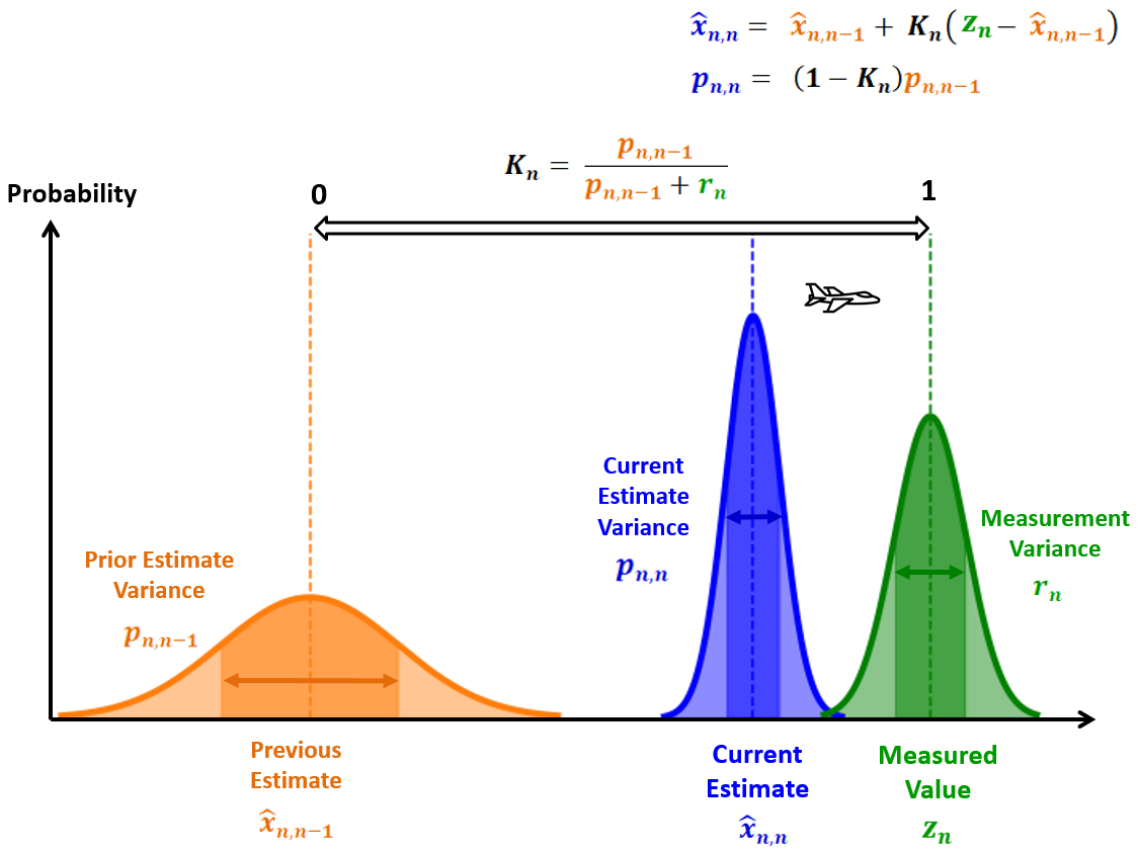


**Figure 4.7:** *High Kalman Gain.*

#### 4.1.6.2 Low Kalman Gain

A high measurement uncertainty relative to the estimate uncertainty would result in a low Kalman Gain (close to 0). Therefore the new estimate would be close to the prior estimate. The following figure illustrates the influence of a low Kalman Gain on the estimate in an aircraft tracking application.

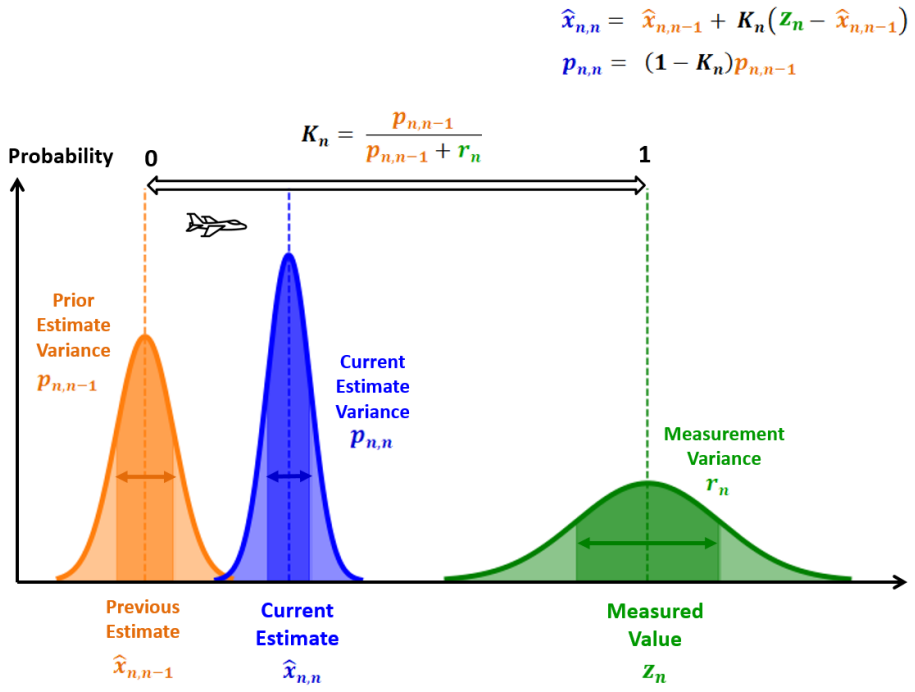


**Figure 4.8:** *Low Kalman Gain.*

Now we understand the Kalman Filter algorithm and are ready for the first numerical example.

## 4.2 Example 5 – Estimating the height of a building

Assume that we would like to estimate the height of a building using an imprecise altimeter.

We know that building height doesn't change over time, at least during the short measurement process.

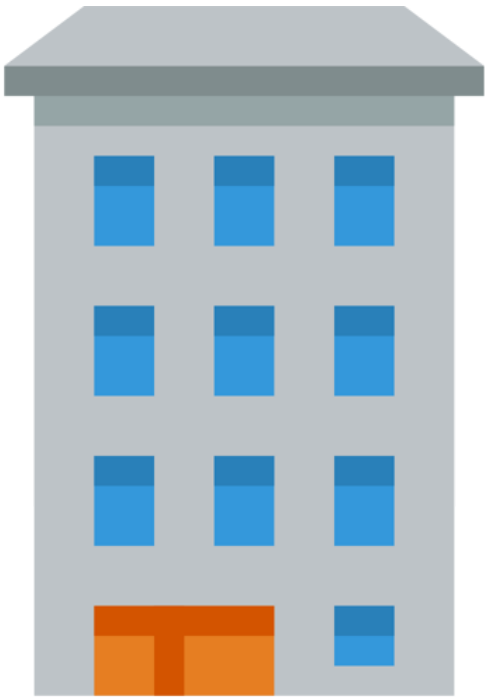

**Figure 4.9:** *Estimating the building height.*

### 4.2.1 The numerical example

- The true building height is 50 meters.
- The altimeter measurement error (standard deviation) is 5 meters.
- The ten measurements are: 49.03m, 48.44m, 55.21m, 49.98m, 50.6m, 52.61m, 45.87m, 42.64m, 48.26m, 55.84m.

#### 4.2.1.1 Iteration Zero

##### Initialization

One can estimate the height of the building simply by looking at it.

The estimated height of the building for the initialization purpose is:
$\hat{x}_{0,0} = 60m$

Now we shall initialize the estimate variance. A human estimation error (standard deviation) is about 15 meters: $\sigma = 15$ . Consequently the variance is 225: $\sigma^2 = 225$.

$p_{0,0} = 225$

#### Prediction

Now, we shall predict the next state based on the initialization values.

Since our system's Dynamic Model is constant, i.e., the building doesn't change its height:

$\hat{x}_{1,0} = \hat{x}_{0,0} = 60m$

The extrapolated estimate variance also doesn't change:

$p_{1,0} = p_{0,0} = 225$

### 4.2.1.2 First Iteration

#### Step 1 - Measure

The first measurement is:

$z_1 = 49.03m$

Since the standard deviation ($\sigma$) of the altimeter measurement error is 5, the variance ($\sigma^2$) would be 25, thus, the measurement uncertainty is: $r_2 = 25$

#### Step 2 - Update

Kalman Gain calculation:

$$K_1 = \frac{p_{1,0}}{p_{1,0} + r_1} = \frac{225}{225 + 25} = 0.9$$

Estimating the current state:

$$\hat{x}_{1,1} = \hat{x}_{1,0} + K_1 (z_1 - \hat{x}_{1,0}) = 60 + 0.9 (49.03 - 60) = 50.13m$$

Update the current estimate variance:

$$p_{1,1} = (1 - K_1) p_{1,0} = (1 - 0.9) 225 = 22.5$$

#### Step 3 - Predict

Since the dynamic model of our system is constant, i.e., the building doesn't change its height:

$\hat{x}_{2,1} = \hat{x}_{1,1} = 50.13m$

The extrapolated estimate variance also doesn't change:

$p_{2,1} = p_{1,1} = 22.5$

#### 4.2.1.3 Second Iteration

After a unit time delay, the predicted estimate from the previous iteration becomes the prior estimate in the current iteration:

$\hat{x}_{2,1} = 50.13m$

The extrapolated estimate variance becomes the prior estimate variance:

$p_{2,1} = 22.5$

##### Step 1 - Measure

The second measurement is:

$z_2 = 48.44m$

The measurement variance is:

$r_2 = 25$

##### Step 2 - Update

Kalman Gain calculation:

$$K_2 = \frac{p_{2,1}}{p_{2,1} + r_2} = \frac{22.5}{22.5 + 25} = 0.47$$

Estimating the current state:

$\hat{x}_{2,2} = \hat{x}_{2,1} + K_2 (z_2 - x_{2,1}) = 50.13 + 0.47 (48.44 - 50.13) = 49.33m$

Update the current estimate variance:

$p_{2,2} = (1 - K_2) p_{2,1} = (1 - 0.47) 22.5 = 11.84$

##### Step 3 - Predict

Since the dynamic model of our system is constant, i.e., the building doesn't change its height:

$\hat{x}_{3,2} = \hat{x}_{2,2} = 49.33m$

The extrapolated estimate variance also doesn't change:

$p_{3,2} = p_{2,2} = 11.84$

#### 4.2.1.4 Iterations 3-10

The calculations for the subsequent iterations are summarized in the following table:

Table 4.3: Example 5 filter iterations.

| $n$ | $z_n$ | Current state estimates $(K_n, \hat{x}_{n,n}, p_{n,n})$ | Prediction $(\hat{x}_{n+1,n}, p_{n+1,n})$ |
|---|---|---|---|
| 3 | 55.21m | $K_3 = \frac{11.84}{11.84+25} = 0.32$<br>$\hat{x}_{3,3} = 49.33 + 0.32\,(55.21 - 49.33) = 51.22m$<br>$p_{3,3} = (1 - 0.32)\,11.84 = 8.04$ | $\hat{x}_{4,3} = \hat{x}_{3,3} = 51.22m$<br>$p_{4,3} = p_{3,3} = 8.04$ |
| 4 | 49.98m | $K_4 = \frac{8.04}{8.04+25} = 0.24$<br>$\hat{x}_{4,4} = 51.22 + 0.24\,(49.98 - 51.22) = 50.92m$<br>$p_{4,4} = (1 - 0.24)\,8.04 = 6.08$ | $\hat{x}_{5,4} = \hat{x}_{4,4} = 50.92m$<br>$p_{5,4} = p_{4,4} = 6.08$ |
| 5 | 50.6m | $K_5 = \frac{6.08}{6.08+25} = 0.2$<br>$\hat{x}_{5,5} = 50.92 + 0.2\,(50.6 - 50.92) = 50.855m$<br>$p_{5,5} = (1 - 0.2)\,6.08 = 4.89$ | $\hat{x}_{6,5} = \hat{x}_{5,5} = 50.855m$<br>$p_{6,5} = p_{5,5} = 4.89$ |
| 6 | 52.61m | $K_6 = \frac{4.89}{4.89+25} = 0.16$<br>$\hat{x}_{6,6} = 50.855 + 0.16\,(52.61 - 50.855) = 51.14m$<br>$p_{6,6} = (1 - 0.16)\,4.89 = 4.09$ | $\hat{x}_{7,6} = \hat{x}_{6,6} = 51.14m$<br>$p_{7,6} = p_{6,6} = 4.09$ |

Continued on next page

Table 4.3: Example 5 filter iterations. (Continued)

| | | | |
|---|---|---|---|
| 7 | 45.87m | $K_7 = \frac{4.09}{4.09 + 25} = 0.14$<br>$\hat{x}_{7,7} = 51.14 + 0.14\,(45.87 - 51.14) = 50.4m$<br>$p_{7,7} = (1 - 0.14)\,4.09 = 3.52$ | $\hat{x}_{8,7} = \hat{x}_{7,7} = 50.4m$<br>$p_{8,7} = p_{7,7} = 3.52$ |
| 8 | 42.64m | $K_8 = \frac{3.52}{3.52 + 25} = 0.12$<br>$\hat{x}_{8,8} = 50.4 + 0.12\,(42.64 - 50.4) = 49.44m$<br>$p_{8,8} = (1 - 0.12)\,3.52 = 3.08$ | $\hat{x}_{9,8} = \hat{x}_{8,8} = 49.44m$<br>$p_{9,8} = p_{8,8} = 3.08$ |
| 9 | 48.26m | $K_9 = \frac{3.08}{3.08 + 25} = 0.11$<br>$\hat{x}_{9,9} = 49.44 + 0.11\,(48.26 - 49.44) = 49.31m$<br>$p_{9,9} = (1 - 0.11)\,3.08 = 2.74$ | $\hat{x}_{10,9} = \hat{x}_{9,9} = 49.31m$<br>$p_{10,9} = p_{9,9} = 2.74$ |
| 10 | 55.84m | $K_{10} = \frac{2.74}{2.74 + 25} = 0.1$<br>$\hat{x}_{10,10} = 49.31 + 0.1\,(55.84 - 49.31) = 49.96m$<br>$p_{10,10} = (1 - 0.1)\,2.74 = 2.47$ | $\hat{x}_{11,10} = \hat{x}_{10,10} = 49.96m$<br>$p_{11,10} = p_{10,10} = 2.47$ |

### 4.2.2 Results analysis

First of all, we want to ensure Kalman Filter convergence. The Kalman Gain should gradually decrease until it reaches a steady state. When Kalman Gain is low, the weight of the noisy measurements is also low. The following plot describes the Kalman Gain for the first one hundred iterations of the Kalman Filter.

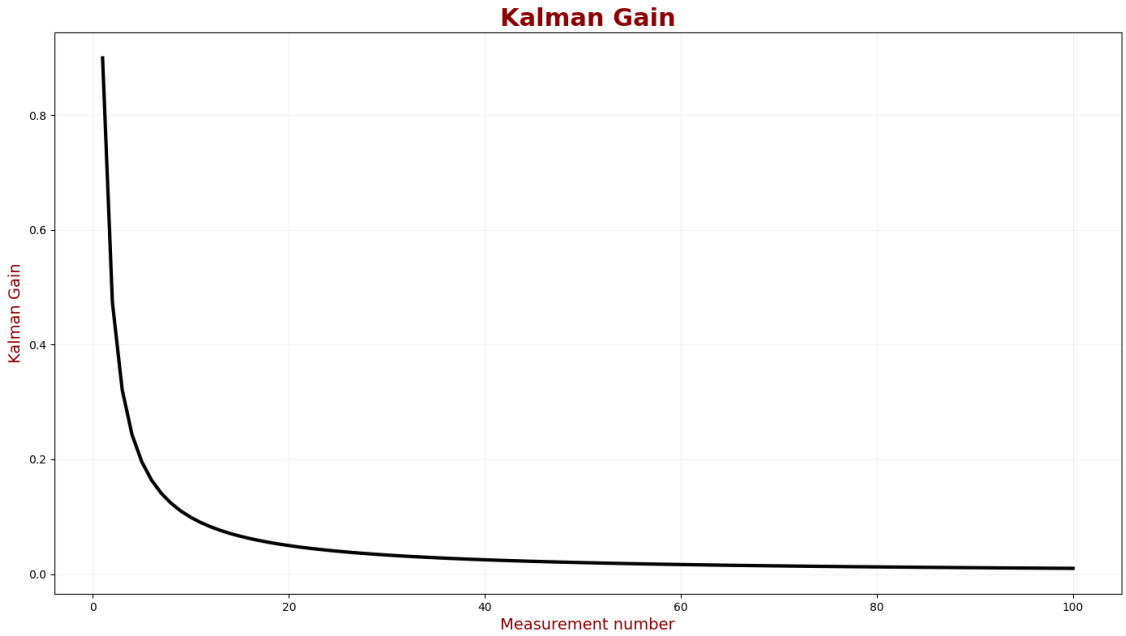


**Figure 4.10:** *Example 5: the Kalman Gain.*

We can see a significant reduction in the Kalman Gain during the first ten iterations. The Kalman Gain enters a steady state after approximately fifty iterations.

We also want to examine accuracy. Accuracy indicates how close the measurement is to the true value. Figure 4.11 compares the true value, measured values, and estimates for the first 50 iterations.

An estimation error is a difference between the true values (the green line) and the KF estimates (the red line). We can see that the estimation errors of our KF decrease in the filter convergence region.

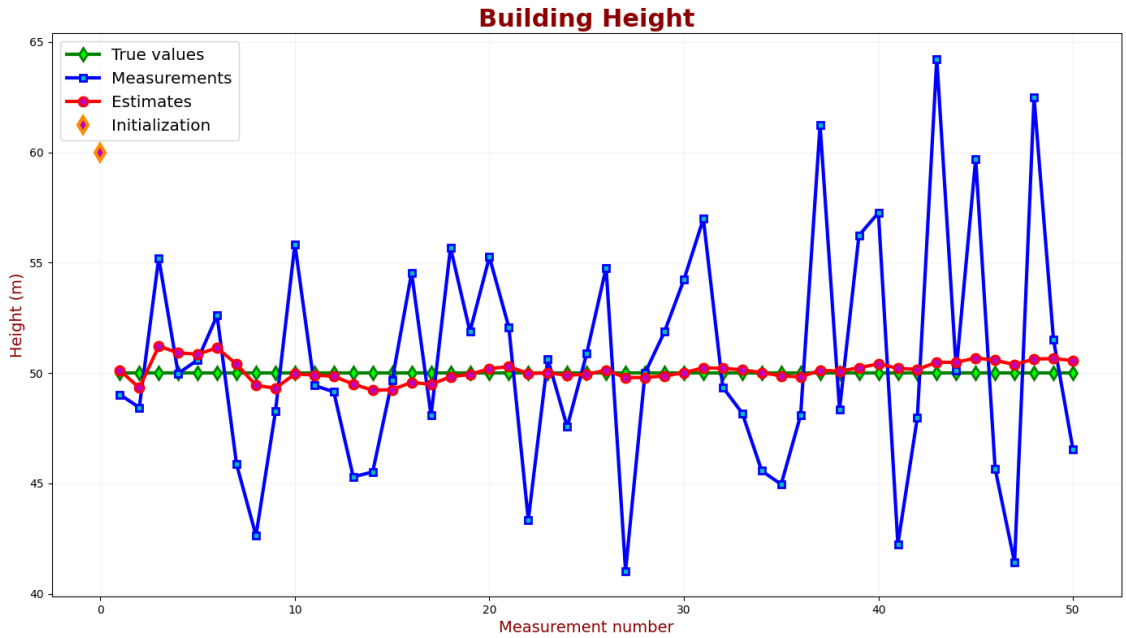


**Figure 4.11:** *Example 5: True value, measured values and estimates.*

One can define accuracy criteria based on the specific application requirements. The typical accuracy criteria are:

- Maximum error
- Mean error
- Root Mean Square Error (RMSE)

Another important parameter is estimation uncertainty. We want the Kalman Filter (KF) estimates to be precise; therefore, we are interested in low estimation uncertainty.

Assume that for a building height measurement application, there is a requirement for 95% confidence. The following chart shows the KF estimates and the true values with 95% confidence intervals.

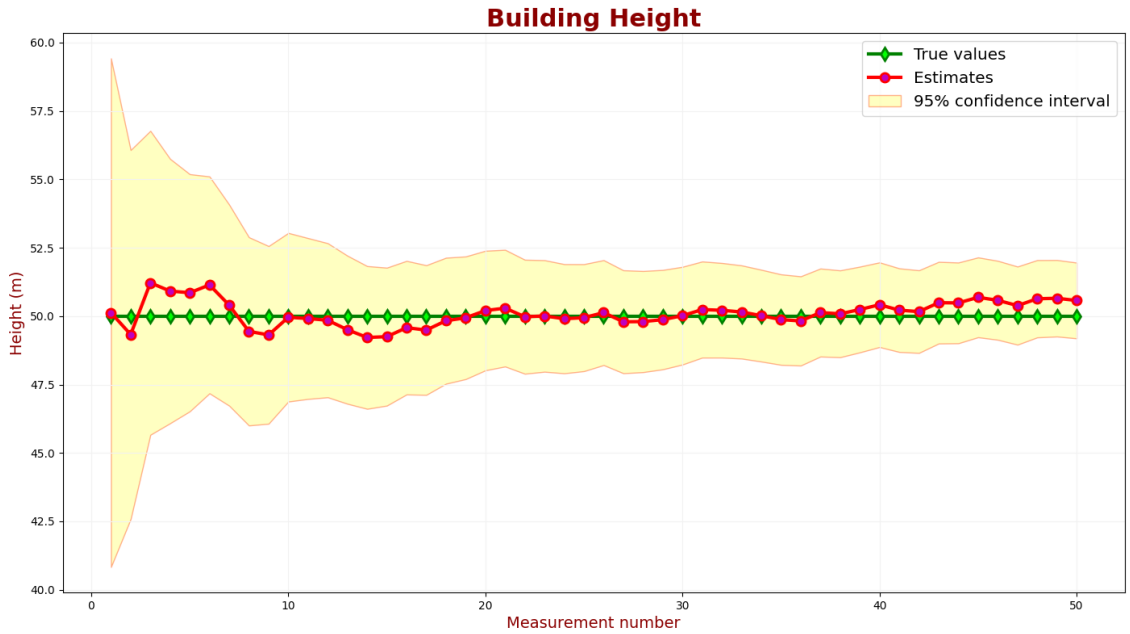


**Figure 4.12:** *High uncertainty.*

You can find the guidelines for a confidence interval calculation in Appendix B.

In the above chart, the confidence intervals are added to the estimates (the red line). 95% of the green samples should be within the 95% confidence region.

We can see that the uncertainty is too high. Let us decrease the measurement uncertainty. The following chart describes the KF output for a low measurement uncertainty parameter.

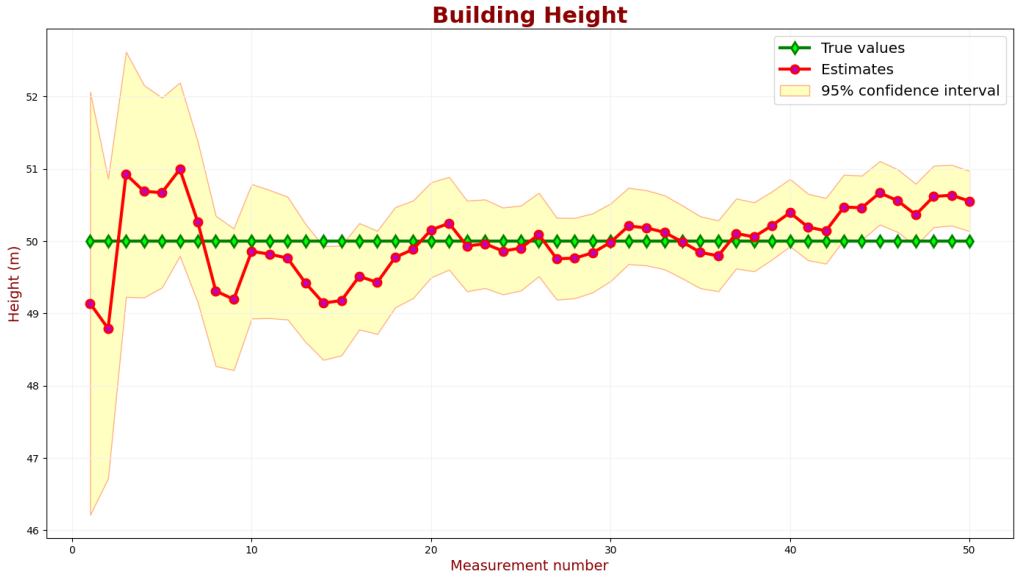


**Figure 4.13:** *Low uncertainty.*

Although we've decreased the uncertainty of the estimates, many green samples

are outside the 95% confidence region. The Kalman Filter is overconfident and too optimistic about its accuracy.

Let us find the measurement uncertainty that yields the desired estimate uncertainty.

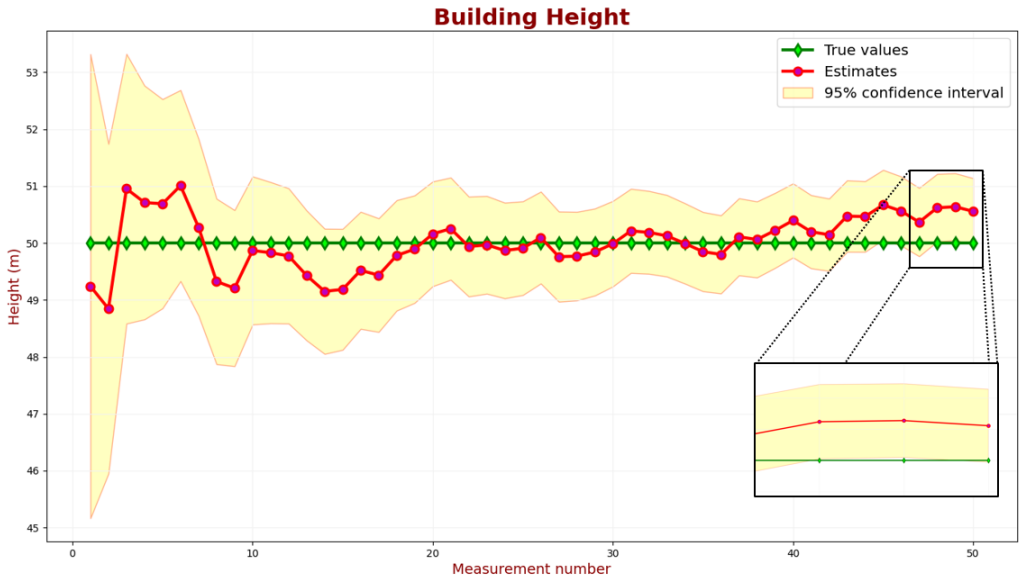


**Figure 4.14:** *Normal uncertainty.*

The above chart shows that 2 out of 50 samples slightly exceed the 95% confidence region. This performance satisfies our requirements.

### 4.2.3 Example summary

We measured the building height using the one-dimensional Kalman Filter in this example. Unlike the $\alpha - \beta - (\gamma)$ filter, the Kalman Gain is dynamic and depends on the precision of the measurement device.

The initial value used by the Kalman Filter is not precise. Therefore, the measurement weight in the State Update Equation is high, and the estimate uncertainty is high.

With each iteration, the measurement weight is lower; therefore, the estimate uncertainty is lower.

The Kalman Filter output includes the estimate and the estimate uncertainty.

# 5. Adding process noise

In this chapter, we add process noise to the one-dimensional Kalman Filter model.

## 5.1 The complete model of the one-dimensional Kalman Filter

### 5.1.1 The Process Noise

In the real world, there are uncertainties in the system dynamic model. For example, when we want to estimate the resistance value of the resistor, we assume a constant dynamic model, i.e., the resistance doesn't change between the measurements. However, the resistance can change slightly due to the fluctuation of the environment temperature. When tracking ballistic missiles with radar, the uncertainty of the dynamic model includes random changes in the target acceleration. The uncertainties are much more significant for an aircraft due to possible aircraft maneuvers.

On the other hand, when we estimate the location of a static object using a Global Positioning System (GPS) receiver, the uncertainty of the dynamic model is zero since the static object doesn't move. The uncertainty of the dynamic model is called the **Process Noise**. In the literature, it is also called plant noise, driving noise, dynamics noise, model noise, and system noise. The process noise produces estimation errors.

In the previous example, we estimated the height of the building. Since the building height doesn't change, we didn't consider the process noise.

The **Process Noise Variance** is denoted by the letter $q$.

The **Covariance Extrapolation Equation** shall include the **Process Noise Variance**.

The Covariance Extrapolation Equation for constant dynamics is:

$$p_{n+1,n} = p_{n,n} + q_n \tag{5.1}$$

These are the updated Kalman Filter equations in one dimension:

| | Equation | Equation Name | Alternative names used in the literature |
|---|---|---|---|
| State Update | $\hat{x}_{n,n} = \hat{x}_{n,n-1} + K_n(z_n - \hat{x}_{n,n})$ | State Update | Filtering Equation |
| | $p_{n,n} = (1 - K_n)p_{n,n-1}$ | Covariance Update | Corrector Equation |
| | $K_n = \frac{p_{n,n-1}}{p_{n,n-1} + r_n}$ | Kalman Gain | Weight Equation |
| State Predict | $\hat{x}_{n+1,n} = \hat{x}_{n,n}$ (for constant dynamics)<br>$\hat{x}_{n+1,n} = \hat{x}_{n,n} + \Delta t\hat{\dot{x}}_{n,n}$<br>$\hat{\dot{x}}_{n+1,n} = \hat{\dot{x}}_{n,n}$ (for constant velocity dynamics) | State Extrapolation | Predictor Equation<br>Transition Equation<br>Prediction Equation<br>Dynamic Model<br>State Space Model |
| | $p^x_{n+1,n} = p^x_{n,n} + q_n$ (for constant dynamics)<br>$p^x_{n+1,n} = p^x_{n,n} + \Delta t^2 p^v_{n,n}$<br>$p^v_{n+1,n} = p^v_{n,n} + q_n$ (for constant velocity dynamics) | Covariance Extrapolation | Predictor Covariance Equation |

**Table 5.1:** *Kalman Filter equations in one dimension with process noise.*

R The State Extrapolation Equation and the Covariance Extrapolation Equation depend on the system dynamics.

R The table above demonstrates the special form of the Kalman Filter equations tailored for the specific case. The general form of the equation is presented later in matrix notation. For now, our goal is to understand the concept of the Kalman Filter.

## 5.2 Example 6 – Estimating the temperature of the liquid in a tank

We want to estimate the temperature of the liquid in a tank.

**Figure 5.1:** *Estimating the liquid temperature.*

We assume that at a steady state, the liquid temperature is constant. However, some fluctuations in the true liquid temperature are possible. We can describe the system dynamics by the following equation:

$$x_n = T + w_n \tag{5.2}$$

Where:

$T$ is the constant temperature

$w_n$ is a random process noise with variance $q$

### 5.2.1 The numerical example

- Let us assume a true temperature of 50 degrees Celsius.
- We assume that the model is accurate. Thus we set the process noise variance ($q$) to 0.0001.
- The measurement error (standard deviation) is 0.1 degrees Celsius.
- The measurements are taken every 5 seconds.
- The true liquid temperature values at the measurement points are: $50.005^oC$, $49.994^oC$, $49.993^oC$, $50.001^oC$, $50.006^oC$, $49.998^oC$, $50.021^oC$, $50.005^oC$, $50^oC$, and $49.997^oC$.
- The measurements are: $49.986^oC$, $49.963^oC$, $50.09^oC$, $50.001^oC$, $50.018^oC$, $50.05^oC$, $49.938^oC$, $49.858^oC$, $49.965^oC$, and $50.114^oC$.

The following chart compares the true liquid temperature and the measurements.

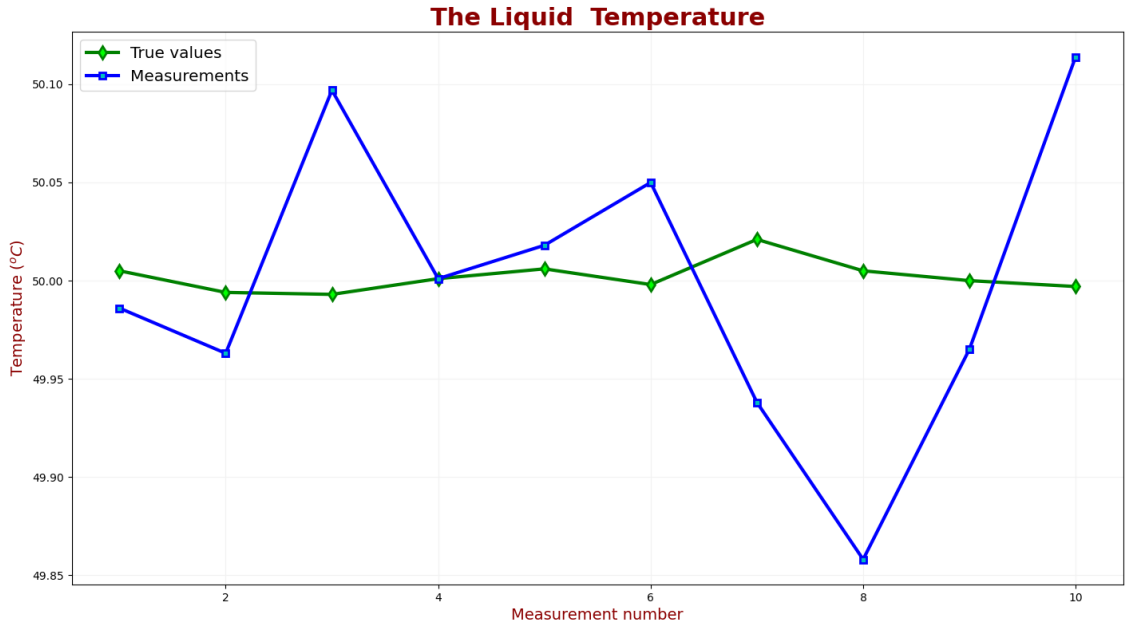


**Figure 5.2:** *Example 6 : true temperature vs. measurements*

#### 5.2.1.1 Iteration Zero

Before the first iteration, we must initialize the Kalman Filter and predict the following state (which is the first state).

##### Initialization

We don't know the true temperature of the liquid in a tank, and our guess is $60^oC$.

$\hat{x}_{0,0} = 60^oC$

Our guess is imprecise, so we set our initialization estimate error $\sigma$ to 100. The **Estimate Variance** of the initialization is the error variance ($\sigma^2$):

$p_{0,0} = 100^2 = 10,000$

This variance is very high. We get faster Kalman Filter convergence if we initialize with a more meaningful value.

##### Prediction

Since our model has constant dynamics, the predicted estimate is equal to the current estimate:

$\hat{x}_{1,0} = 60^oC$

The extrapolated estimate variance:

$p_{1,0} = p_{0,0} + q = 10000 + 0.0001 = 10000.0001$

#### 5.2.1.2 First Iteration

##### Step 1 - Measure

The measurement value:

$z_1 = 49.986^oC$

Since the measurement error is 0.1 ($\sigma$), the variance ($\sigma^2$) would be 0.01; thus, the measurement variance is:

$r_1 = 0.01$

##### Step 2 - Update

Kalman Gain calculation:

$$K_1 = \frac{p_{1,0}}{p_{1,0} + r_1} = \frac{10000.0001}{10000.0001 + 0.01} = 0.999999$$

The Kalman Gain is almost 1; thus, our estimate error is much bigger than the measurement error. Thus the weight of the estimate is negligible, while the measurement weight is almost 1.

Estimating the current state:

$\hat{x}_{1,1} = \hat{x}_{1,0} + K_1 (z_1 - \hat{x}_{1,0}) = 60 + 0.999999 (49.986 - 60) = 49.986^oC$

Update the current estimate variance:

$p_{1,1} = (1 - K_1) p_{1,0} = (1 - 0.999999) 10000.0001 = 0.01$

##### Step 3 - Predict

Since our system's Dynamic Model is constant, i.e., the liquid temperature doesn't change:

$\hat{x}_{2,1} = \hat{x}_{1,1} = 49.986^oC$

The extrapolated estimate variance is:

$p_{2,1} = p_{1,1} + q = 0.01 + 0.0001 = 0.0101$

#### 5.2.1.3 Second Iteration

##### Step 1 - Measure

The second measurement is:

$z_2 = 49.963^oC$

Since the measurement error is 0.1 ($\sigma$), the variance ($\sigma^2$) would be 0.01; thus, the measurement variance is:

$r_2 = 0.01$

#### Step 2 - Update

Kalman Gain calculation:

$$K_2 = \frac{p_{2,1}}{p_{2,1} + r_2} = \frac{0.0101}{0.0101 + 0.01} = 0.5$$

The Kalman Gain is 0.5, i.e., the weight of the estimate and the measurement weight are equal.

Estimating the current state:

$$\hat{x}_{2,2} = \hat{x}_{2,1} + K_2 \left(z_2 - x_{2,1}\right) = 50.13 + 0.47 \left(48.44 - 50.13\right) = 49.33m$$

Update the current estimate variance:

$$p_{2,2} = \left(1 - K_2\right) p_{2,1} = \left(1 - 0.5\right) 0.0101 = 0.005$$

#### Step 3 - Predict

Since the dynamic model of the system is constant, i.e., the liquid temperature doesn't change:

$$\hat{x}_{3,2} = \hat{x}_{2,2} = 49.974^oC$$

The extrapolated estimate variance is:

$$p_{3,2} = p_{2,2} + q = 0.005 + 0.0001 = 0.0051$$

### 5.2.1.4 Iterations 3-10

The calculations for the subsequent iterations are summarized in the following table:

Table 5.2: Example 6 filter iterations.

| $n$ | $z_n$ | Current state estimates $(K_n, \hat{x}_{n,n}, p_{n,n})$ | Prediction $(\hat{x}_{n+1,n}, p_{n+1,n})$ |
|---|---|---|---|
| 3 | $50.09^oC$ | $K_3 = \frac{0.0051}{0.0051 + 0.01} = 0.3388$<br><br>$\hat{x}_{3,3} = 49.974 + 0.3388 \left(50.09 - 49.974\right) = 50.016^oC$<br><br>$p_{3,3} = \left(1 - 0.3388\right) 0.0051 = 0.0034$ | $\hat{x}_{4,3} = \hat{x}_{3,3} = 50.016^oC$<br><br>$p_{4,3} = 0.0034 + 0.0001 = 0.0035$ |

Continued on next page

Table 5.2: Example 6 filter iterations. (Continued)

| | | | |
|---|---|---|---|
| 4 | $50.001^oC$ | $K_4 = \frac{0.0035}{0.0035 + 0.01} = 0.2586$<br><br>$\hat{x}_{4,4} = 50.016 + 0.2586\,(50.001 - 50.016) = 50.012^oC$<br><br>$p_{4,4} = (1 - 0.2586)\,0.0035 = 0.0026$ | $\hat{x}_{5,4} = \hat{x}_{4,4} = 50.012^oC$<br><br>$p_{5,4} = 0.0026 + 0.0001 = 0.0027$ |
| 5 | $50.018^oC$ | $K_5 = \frac{0.0027}{0.0027 + 0.01} = 0.2117$<br><br>$\hat{x}_{5,5} = 50.012 + 0.2117\,(50.018 - 50.012) = 50.013^oC$<br><br>$p_{5,5} = (1 - 0.2117)\,0.0027 = 0.0021$ | $\hat{x}_{6,5} = \hat{x}_{5,5} = 50.013^oC$<br><br>$p_{6,5} = 0.0021 + 0.0001 = 0.0022$ |
| 6 | $50.05^oC$ | $K_6 = \frac{0.0022}{0.0022 + 0.01} = 0.1815$<br><br>$\hat{x}_{6,6} = 50.013 + 0.1815\,(50.05 - 50.013) = 50.02^oC$<br><br>$p_{6,6} = (1 - 0.1815)\,0.0022 = 0.0018$ | $\hat{x}_{7,6} = \hat{x}_{6,6} = 50.02^oC$<br><br>$p_{7,6} = 0.0018 + 0.0001 = 0.0019$ |
| 7 | $49.938^oC$ | $K_7 = \frac{0.0019}{0.0019 + 0.01} = 0.1607$<br><br>$\hat{x}_{7,7} = 50.02 + 0.1607\,(49.938 - 50.02) = 50.007^oC$<br><br>$p_{7,7} = (1 - 0.1607)\,0.0019 = 0.0016$ | $\hat{x}_{8,7} = \hat{x}_{7,7} = 49.978^oC$<br><br>$p_{8,7} = 0.0016 + 0.0001 = 0.0017$ |

Continued on next page

Table 5.2: Example 6 filter iterations. (Continued)

| | | | |
|---|---|---|---|
| 8 | $49.858^oC$ | $K_8 = \frac{0.0017}{0.0017 + 0.01} = 0.1458$<br><br>$\hat{x}_{8,8} = 50.007 + 0.1458\,(49.858 - 50.007) = 49.985^oC$<br><br>$p_{8,8} = (1 - 0.1458)\,0.0017 = 0.0015$ | $\hat{x}_{9,8} = \hat{x}_{8,8} = 49.985^oC$<br><br>$p_{9,8} = 0.0015 + 0.0001 = 0.0016$ |
| 9 | $49.965^oC$ | $K_9 = \frac{0.0016}{0.0016 + 0.01} = 0.1348$<br><br>$\hat{x}_{9,9} = 49.985 + 0.1348\,(49.965 - 49.985) = 49.982^oC$<br><br>$p_{9,9} = (1 - 0.1348)\,0.0016 = 0.0014$ | $\hat{x}_{10,9} = \hat{x}_{9,9} = 49.982^oC$<br><br>$p_{10,9} = 0.0014 + 0.0001 = 0.0015$ |
| 10 | $50.114^oC$ | $K_{10} = \frac{0.0015}{0.0015 + 0.01} = 0.1265$<br><br>$\hat{x}_{10,10} = 49.982 + 0.1265\,(50.114 - 49.982) = 49.999^oC$<br><br>$p_{10,10} = (1 - 0.1265)\,0.0015 = 0.0013$ | $\hat{x}_{11,10} = \hat{x}_{10,10} = 49.999^oC$<br><br>$p_{11,10} = 0.0013 + 0.0001 = 0.0014$ |

### 5.2.2 Results analysis

The following chart describes the Kalman Gain.

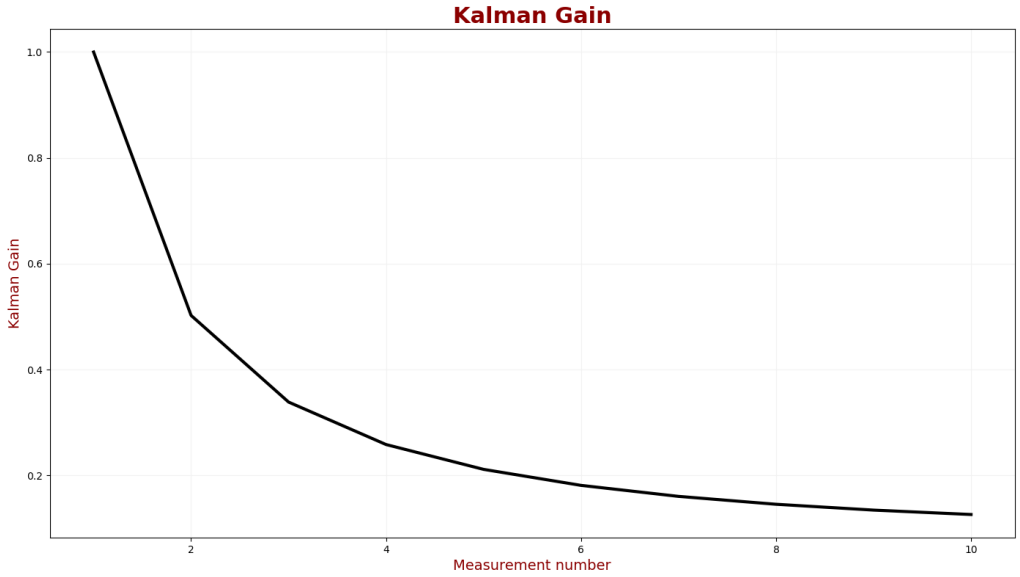


**Figure 5.3:** *Example 6: the Kalman Gain.*

As you can see, the Kalman Gain gradually decreases; therefore, the KF converges.

The following chart compares the true value, measured values, and estimates. The confidence interval is 95%.

You can find the guidelines for a confidence interval calculation in Appendix B.

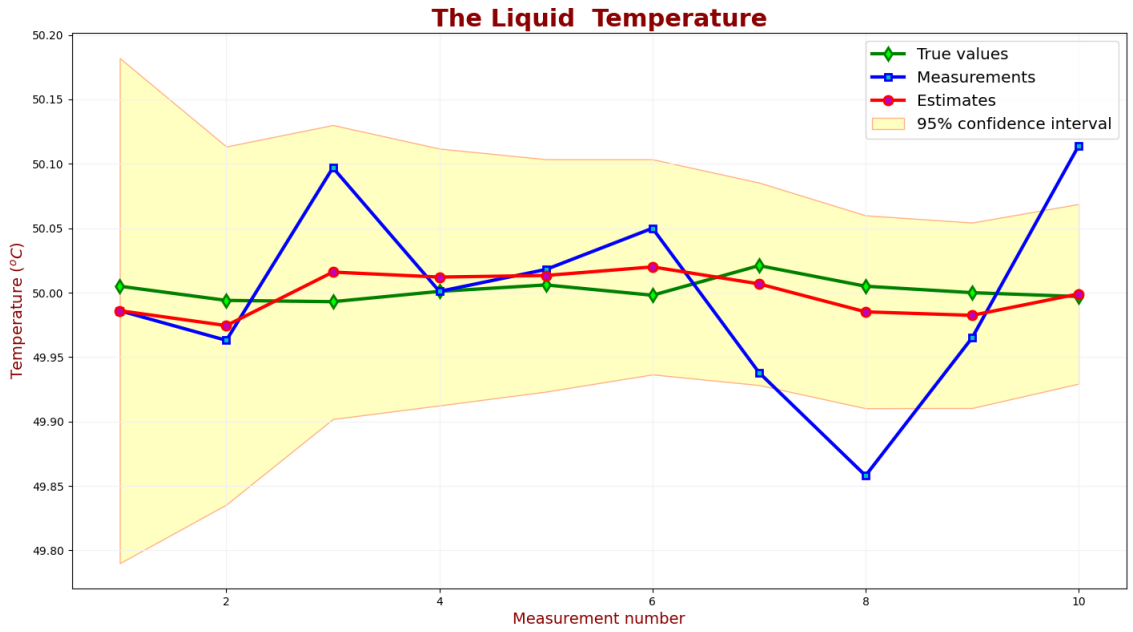


**Figure 5.4:** *Example 6: true value, measured values and estimates.*

As you can see, the estimated value converges toward the true value. The KF estimates uncertainties are too high for the 95% confidence level.

### 5.2.3 Example summary

We measured a liquid temperature using the one-dimensional Kalman Filter. Although the system dynamics include a random process noise, the Kalman Filter provides a good estimation.

## 5.3 Example 7 – Estimating the temperature of a heating liquid I

Like in the previous example, we estimate the temperature of a liquid in a tank. In this case, the dynamic model of the system is not constant - the liquid is heating at a rate of $0.1^oC$ every second.

### 5.3.1 The numerical example

The Kalman Filter parameters are similar to the previous example:

- We assume that the model is accurate. Thus we set the process noise variance ($q$) to 0.0001.
- The measurement error (standard deviation) is $0.1^oC$.
- The measurements are taken every 5 seconds.
- The dynamic model of the system is constant.

R Although the true dynamic model of the system is not constant (since the liquid is heating), we treat the system as a system with a constant dynamic model (the temperature doesn't change).

- The true liquid temperature values at the measurement points are: $50.505^oC$, $50.994^oC$, $51.493^oC$, $52.001^oC$, $52.506^oC$, $52.998^oC$, $53.521^oC$, $54.005^oC$, $54.5^oC$, and $54.997^oC$.
- The measurements are: $50.486^oC$, $50.963^oC$, $51.597^oC$, $52.001^oC$, $52.518^oC$, $53.05^oC$, $53.438^oC$, $53.858^oC$, $54.465^oC$, and $55.114^oC$.

The following chart compares the true liquid temperature and the measurements.

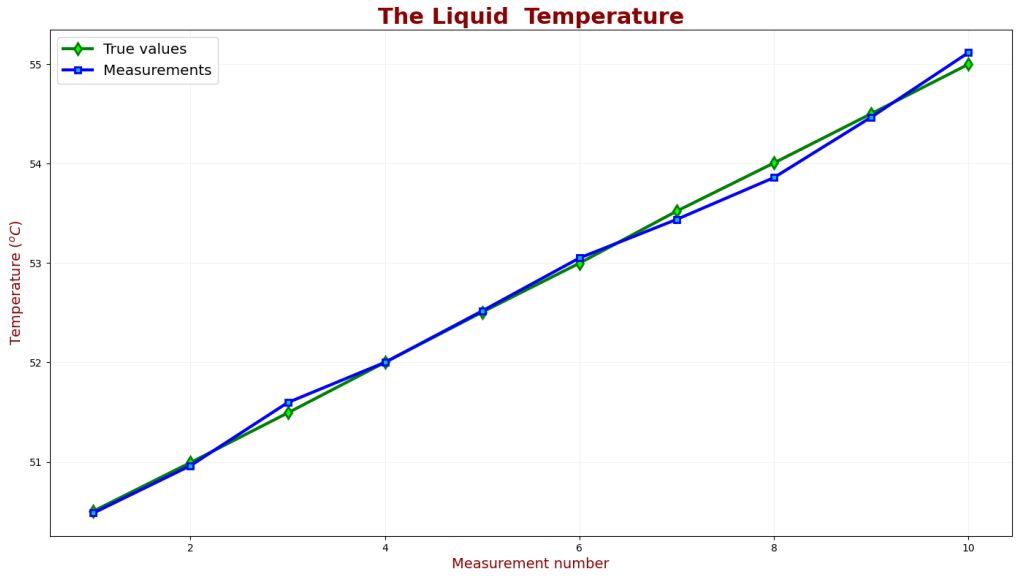


**Figure 5.5:** *Example 7 : true temperature vs. measurements*

#### 5.3.1.1 Iteration Zero

Iteration zero is similar to the previous example.

Before the first iteration, we must initialize the Kalman Filter and predict the following state (which is the first state).

##### Initialization

We don't know the true temperature of the liquid in a tank, and our guess is $10^oC$.

$\hat{x}_{0,0} = 10^oC$

Our guess is imprecise, so we set our initialization estimate error $\sigma$ to 100. The **Estimate Variance** of the initialization is the error variance ($\sigma^2$):

$p_{0,0} = 100^2 = 10,000$

This variance is very high. We get faster Kalman Filter convergence if we initialize with a more meaningful value.

##### Prediction

Now, we shall predict the next state based on the initialization values.

Since our model has constant dynamics, the predicted estimate is equal to the current estimate:

$\hat{x}_{1,0} = 10^oC$

The extrapolated estimate variance:

$p_{1,0} = p_{0,0} + q = 10000 + 0.0001 = 10000.0001$

#### 5.3.1.2 Iterations 1-10

The calculations for the subsequent iterations are summarized in the following table:

Table 5.3: Example 7 filter iterations.

| $n$ | $z_n$ | Current state estimates $(K_n, \hat{x}_{n,n}, p_{n,n})$ | Prediction $(\hat{x}_{n+1,n}, p_{n+1,n})$ |
|---|---|---|---|
| 1 | $50.486^oC$ | $K_1 = \frac{10000.0001}{10000.0001 + 0.01} = 0.999999$<br><br>$\hat{x}_{1,1} = 10 + 0.999999\,(50.486 - 10) = 50.486^oC$<br><br>$p_{1,1} = (1 - 0.999999)\,10000.0001 = 0.01$ | $\hat{x}_{2,1} = \hat{x}_{1,1} = 50.486^oC$<br><br>$p_{2,1} = 0.01 + 0.0001 = 0.0101$ |
| 2 | $50.963^oC$ | $K_2 = \frac{0.0101}{0.0101 + 0.01} = 0.5025$<br><br>$\hat{x}_{2,2} = 50.486 + 0.5025\,(50.963 - 50.486) = 50.726^oC$<br><br>$p_{2,2} = (1 - 0.5025)\,0.0101 = 0.005$ | $\hat{x}_{3,2} = \hat{x}_{2,2} = 50.726^oC$<br><br>$p_{3,2} = 0.005 + 0.0001 = 0.0051$ |
| 3 | $51.597^oC$ | $K_3 = \frac{0.0051}{0.0051 + 0.01} = 0.3388$<br><br>$\hat{x}_{3,3} = 50.726 + 0.3388\,(51.597 - 50.726) = 51.021^oC$<br><br>$p_{3,3} = (1 - 0.3388)\,0.0051 = 0.0034$ | $\hat{x}_{4,3} = \hat{x}_{3,3} = 51.021^oC$<br><br>$p_{4,3} = 0.0034 + 0.0001 = 0.0035$ |

Continued on next page

Table 5.3: Example 7 filter iterations. (Continued)

| | | | |
|---|---|---|---|
| 4 | $52.001^oC$ | $K_4 = \frac{0.0035}{0.0035 + 0.01} = 0.2586$<br><br>$\hat{x}_{4,4} = 51.021 + 0.2586\,(52.001 - 51.021) = 51.274^oC$<br><br>$p_{4,4} = (1 - 0.2586)\,0.0035 = 0.0026$ | $\hat{x}_{5,4} = \hat{x}_{4,4} = 51.274^oC$<br><br>$p_{5,4} = 0.0026 + 0.0001 = 0.0027$ |
| 5 | $52.518^oC$ | $K_5 = \frac{0.0027}{0.0027 + 0.01} = 0.2117$<br><br>$\hat{x}_{5,5} = 51.274 + 0.2117\,(52.518 - 51.274) = 51.538^oC$<br><br>$p_{5,5} = (1 - 0.2117)\,0.0027 = 0.0021$ | $\hat{x}_{6,5} = \hat{x}_{5,5} = 51.538^oC$<br><br>$p_{6,5} = 0.0021 + 0.0001 = 0.0022$ |
| 6 | $53.05^oC$ | $K_6 = \frac{0.0022}{0.0022 + 0.01} = 0.1815$<br><br>$\hat{x}_{6,6} = 51.538 + 0.1815\,(53.05 - 51.538) = 51.812^oC$<br><br>$p_{6,6} = (1 - 0.1815)\,0.0022 = 0.0018$ | $\hat{x}_{7,6} = \hat{x}_{6,6} = 51.812^oC$<br><br>$p_{7,6} = 0.0018 + 0.0001 = 0.0019$ |
| 7 | $53.438^oC$ | $K_7 = \frac{0.0019}{0.0019 + 0.01} = 0.1607$<br><br>$\hat{x}_{7,7} = 51.812 + 0.1607\,(53.438 - 51.812) = 52.0735^oC$<br><br>$p_{7,7} = (1 - 0.1607)\,0.0019 = 0.0016$ | $\hat{x}_{8,7} = \hat{x}_{7,7} = 52.0735^oC$<br><br>$p_{8,7} = 0.0016 + 0.0001 = 0.0017$ |

Continued on next page

Table 5.3: Example 7 filter iterations. (Continued)

| | | | |
|---|---|---|---|
| 8 | $53.858^oC$ | $K_8 = \frac{0.0017}{0.0017 + 0.01} = 0.1458$<br><br>$\hat{x}_{8,8} = 52.0735 + 0.1458\,(53.858 - 52.0735) = 52.334^oC$<br><br>$p_{8,8} = (1 - 0.1458)\,0.0017 = 0.0015$ | $\hat{x}_{9,8} = \hat{x}_{8,8} = 52.334^oC$<br><br>$p_{9,8} = 0.0015 + 0.0001 = 0.0016$ |
| 9 | $54.523^oC$ | $K_9 = \frac{0.0016}{0.0016 + 0.01} = 0.1348$<br><br>$\hat{x}_{9,9} = 52.334 + 0.1348\,(54.523 - 52.334) = 52.621^oC$<br><br>$p_{9,9} = (1 - 0.1348)\,0.0016 = 0.0014$ | $\hat{x}_{10,9} = \hat{x}_{9,9} = 52.621^oC$<br><br>$p_{10,9} = 0.0014 + 0.0001 = 0.0015$ |
| 10 | $55.114^oC$ | $K_{10} = \frac{0.0015}{0.0015 + 0.01} = 0.1265$<br><br>$\hat{x}_{10,10} = 2.621 + 0.1265\,(55.114 - 52.621) = 52.936^oC$<br><br>$p_{10,10} = (1 - 0.1265)\,0.0015 = 0.0013$ | $\hat{x}_{11,10} = \hat{x}_{10,10} = 52.936^oC$<br><br>$p_{11,10} = 0.0013 + 0.0001 = 0.0014$ |

### 5.3.2 Results analysis

The following chart compares the true value, measured values, and estimates.

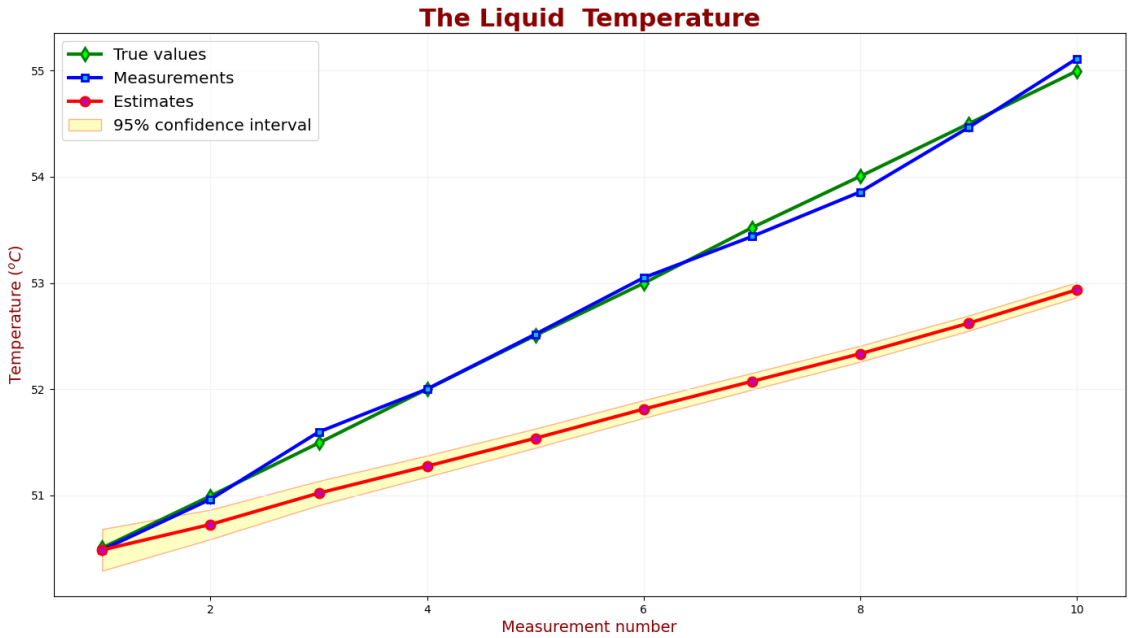


**Figure 5.6:** *Example 7: true value, measured values and estimates.*

As you can see, the Kalman Filter has failed to provide a reliable estimation. There is a **lag error** in the Kalman Filter estimation. We've already encountered the lag error in Example 3, where we estimated the position of an accelerating aircraft using the $\alpha - \beta$ filter that assumes constant aircraft velocity. We got rid of the lag error in Example 4, where we replaced the $\alpha - \beta$ filter with the $\alpha - \beta - \gamma$ filter that assumes acceleration.

There are two reasons for the lag error in our Kalman Filter example:

- The dynamic model doesn't fit the case.
- We have chosen very low process noise ($q = 0.0001$) while the true temperature fluctuations are much more significant.

R The lag error is constant. Therefore the estimate curve should have the same slope as the true value curve. Figure 5.6 presents only the 10 first measurements, which is not enough for convergence. Figure 5.7 presents the first 100 measurements with a constant lag error.

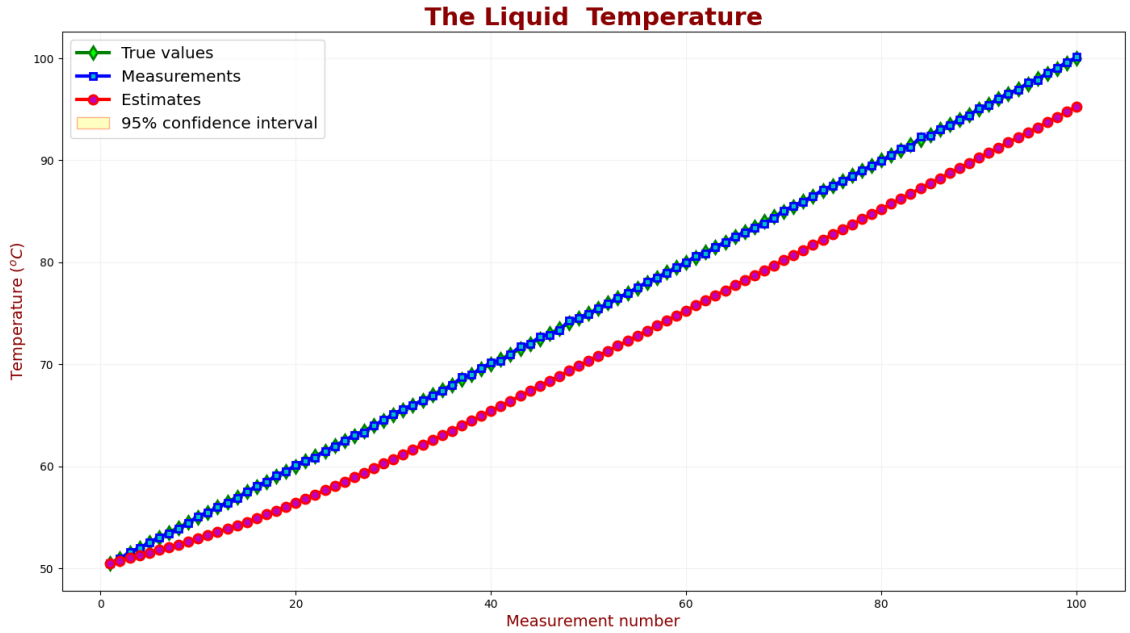


**Figure 5.7:** *Example 7: 100 measurements.*

There are two possible ways to fix the lag error:

- If we know that the liquid temperature can change linearly, we can define a new model that considers a possible linear change in the liquid temperature. We did this in Example 4. This method is preferred. However, this method won't improve the Kalman Filter performance if the temperature change can't be modeled.
- On the other hand, since our model is not well defined, we can adjust the process model reliability by increasing the process noise ($q$). See the next example for details.

Another problem is a low estimate uncertainty. The KF failed to provide accurate estimates and is also confident in its estimates. It is an example of a **bad KF design**.

### 5.3.3 Example summary

In this example, we measured the temperature of a heating liquid using a one-dimensional Kalman Filter with a constant dynamic model. We've observed the lag error in the Kalman Filter estimation. The wrong dynamic model and process model definitions cause the lag error.

An appropriate dynamic model or process model definition can fix the lag error.

## 5.4 Example 8 – Estimating the temperature of a heating liquid II

This example is similar to the previous example, with only one change. Since our process is not well-defined, we increase the process variance ($q$) from 0.0001 to 0.15.

### 5.4.1 The numerical example

#### 5.4.1.1 Iteration Zero

Iteration zero is similar to the previous example.

Before the first iteration, we must initialize the Kalman Filter and predict the following state (which is the first state).

**Initialization**

We don't know the true temperature of the liquid in a tank, and our guess is $10^oC$.

$\hat{x}_{0,0} = 10^oC$

Our guess is imprecise, so we set our initialization estimate error $\sigma$ to 100. The **Estimate Variance** of the initialization is the error variance ($\sigma^2$):

$p_{0,0} = 100^2 = 10,000$

This variance is very high. We get faster Kalman Filter convergence if we initialize with a more meaningful value.

**Prediction**

Now, we shall predict the next state based on the initialization values.

Since our model has constant dynamics, the predicted estimate is equal to the current estimate:

$\hat{x}_{1,0} = 10^oC$

The extrapolated estimate variance:

$p_{1,0} = p_{0,0} + q = 10000 + 0.15 = 10000.15$

#### 5.4.1.2 Iterations 1-10

The calculations for the subsequent iterations are summarized in the following table:

Table 5.4: Example 5 filter iterations.

| $n$ | $z_n$ | Current state estimates $(K_n, \hat{x}_{n,n}, p_{n,n})$ | Prediction $(\hat{x}_{n+1,n}, p_{n+1,n})$ |
|---|---|---|---|
| 1 | $50.486^oC$ | $K_1 = \frac{10000.15}{10000.15 + 0.01} = 0.999999$<br><br>$\hat{x}_{1,1} = 10 + 0.999999\,(50.486 - 10) = 50.486^oC$<br><br>$p_{1,1} = (1 - 0.999999)\,10000.15 = 0.01$ | $\hat{x}_{2,1} = \hat{x}_{1,1} = 50.486^oC$<br><br>$p_{2,1} = 0.01 + 0.15 = 0.16$ |
| 2 | $50.963^oC$ | $K_2 = \frac{0.16}{0.16 + 0.01} = 0.9412$<br><br>$\hat{x}_{2,2} = 50.486 + 0.9412\,(50.963 - 50.486) = 50.934^oC$<br><br>$p_{2,2} = (1 - 0.9412)\,0.16 = 0.0094$ | $\hat{x}_{3,2} = \hat{x}_{2,2} = 50.934^oC$<br><br>$p_{3,2} = 0.0094 + 0.15 = 0.1594$ |
| 3 | $51.597^oC$ | $K_3 = \frac{0.1594}{0.1594 + 0.01} = 0.941$<br><br>$\hat{x}_{3,3} = 50.934 + 0.941\,(51.597 - 50.934) = 51.556^oC$<br><br>$p_{3,3} = (1 - 0.941)\,0.1594 = 0.0094$ | $\hat{x}_{4,3} = \hat{x}_{3,3} = 51.556^oC$<br><br>$p_{4,3} = 0.0094 + 0.15 = 0.1594$ |

Continued on next page

Table 5.4: Example 5 filter iterations. (Continued)

| | | | |
|---|---|---|---|
| 4 | $52.001^oC$ | $K_4 = \frac{0.1594}{0.1594 + 0.01} = 0.941$<br>$\hat{x}_{4,4} = 51.556 + 0.941\,(52.001 - 51.556) = 51.975^oC$<br>$p_{4,4} = (1 - 0.941)\,0.1594 = 0.0094$ | $\hat{x}_{5,4} = \hat{x}_{4,4} = 51.975^oC$<br>$p_{5,4} = 0.0094 + 0.15 = 0.1594$ |
| 5 | $52.518^oC$ | $K_5 = \frac{0.1594}{0.1594 + 0.01} = 0.941$<br>$\hat{x}_{5,5} = 51.975 + 0.941\,(52.518 - 51.975) = 52.486^oC$<br>$p_{5,5} = (1 - 0.941)\,0.1594 = 0.0094$ | $\hat{x}_{6,5} = \hat{x}_{5,5} = 52.486^oC$<br>$p_{6,5} = 0.0094 + 0.15 = 0.1594$ |
| 6 | $53.05^oC$ | $K_6 = \frac{0.1594}{0.1594 + 0.01} = 0.941$<br>$\hat{x}_{6,6} = 52.486 + 0.941\,(53.05 - 52.486) = 53.017^oC$<br>$p_{6,6} = (1 - 0.941)\,0.1594 = 0.0094$ | $\hat{x}_{7,6} = \hat{x}_{6,6} = 53.017^oC$<br>$p_{7,6} = 0.0094 + 0.15 = 0.1594$ |
| 7 | $53.438^oC$ | $K_7 = \frac{0.1594}{0.1594 + 0.01} = 0.941$<br>$\hat{x}_{7,7} = 53.017 + 0.941\,(53.438 - 53.017) = 53.413^oC$<br>$p_{7,7} = (1 - 0.941)\,0.1594 = 0.0094$ | $\hat{x}_{8,7} = \hat{x}_{7,7} = 53.413^oC$<br>$p_{8,7} = 0.0094 + 0.15 = 0.1594$ |

Continued on next page

Table 5.4: Example 5 filter iterations. (Continued)

| | | | |
|---|---|---|---|
| 8 | $53.858^oC$ | $K_8 = \frac{0.1594}{0.1594 + 0.01} = 0.941$<br>$\hat{x}_{8,8} = 53.413 + 0.941\,(53.858 - 53.413) = 53.832^oC$<br>$p_{8,8} = (1 - 0.941)\,0.1594 = 0.0094$ | $\hat{x}_{9,8} = \hat{x}_{8,8} = 53.832^oC$<br>$p_{9,8} = 0.0094 + 0.15 = 0.1594$ |
| 9 | $54.523^oC$ | $K_9 = \frac{0.1594}{0.1594 + 0.01} = 0.941$<br>$\hat{x}_{9,9} = 53.832 + 0.941\,(54.523 - 53.832) = 54.428^oC$<br>$p_{9,9} = (1 - 0.941)\,0.1594 = 0.0094$ | $\hat{x}_{10,9} = \hat{x}_{9,9} = 54.428^oC$<br>$p_{10,9} = 0.0094 + 0.15 = 0.1594$ |
| 10 | $55.114^oC$ | $K_{10} = \frac{0.1594}{0.1594 + 0.01} = 0.941$<br>$\hat{x}_{10,10} = 54.428 + 0.941\,(55.114 - 54.428) = 55.074^oC$<br>$p_{10,10} = (1 - 0.941)\,0.1594 = 0.0094$ | $\hat{x}_{11,10} = \hat{x}_{10,10} = 55.074^oC$<br>$p_{11,10} = 0.0094 + 0.15 = 0.1594$ |

### 5.4.2 Results analysis

The following chart compares the true value, measured values, and estimates.

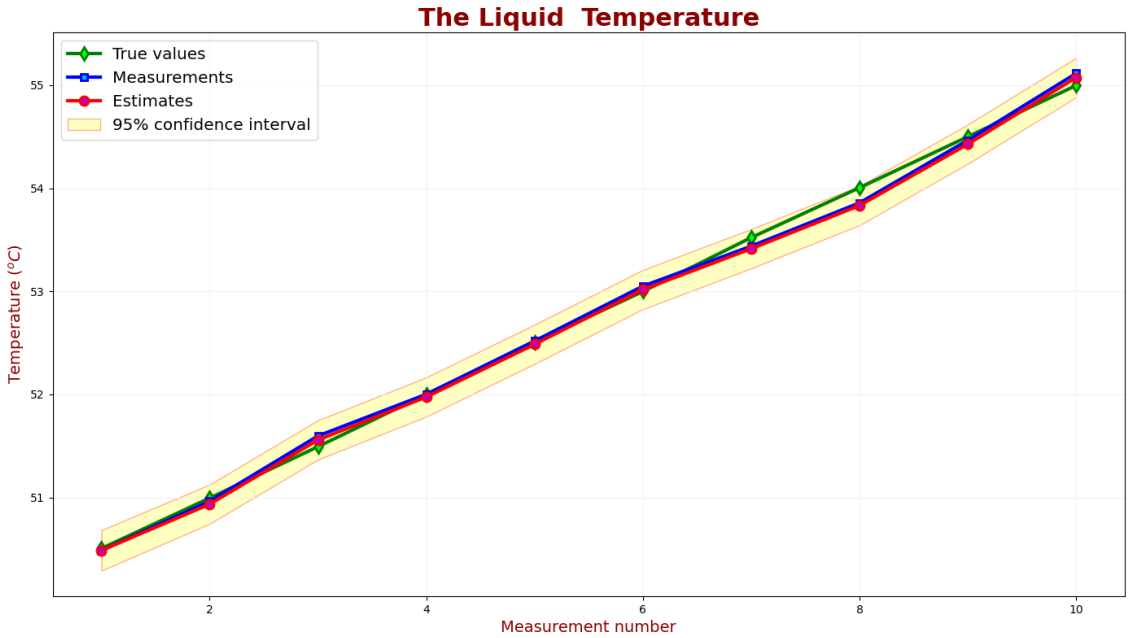


**Figure 5.8:** *Example 8: true value, measured values and estimates.*

As you can see, the estimates follow the measurements. There is no lag error.

Let us take a look at the Kalman Gain.

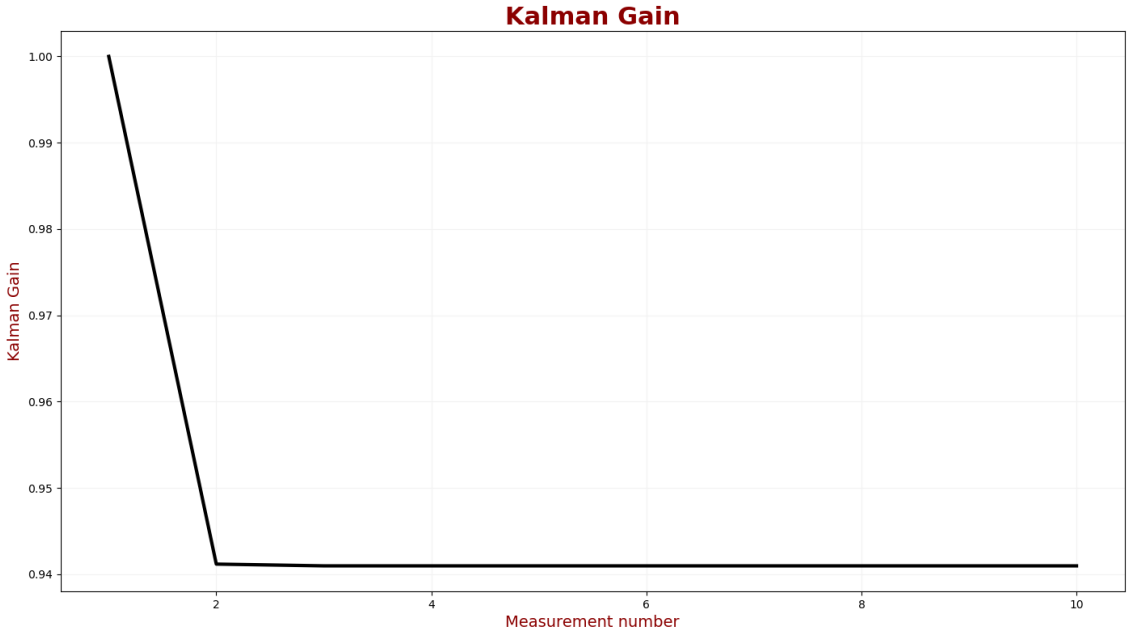


**Figure 5.9:** *Example 8: the Kalman Gain.*

Due to the high process uncertainty, the measurement weight is much higher than

the weight of the estimate. Thus, the Kalman Gain is high, and it converges to 0.94.

The good news is that we can trust the estimates of this KF. The true values (the green line) are within the 95% confidence region.

### 5.4.3 Example summary

The best Kalman Filter implementation involves a model that is very close to reality, leaving little room for process noise. However, a precise model is not always available. For example, an airplane pilot may decide to perform a sudden maneuver that changes the predicted airplane trajectory. In this case, the process noise would be increased.

# II Multivariate Kalman Filter

# 6. Foreword

After reading the "Kalman Filter in one dimension" part, you should be familiar with the concepts of the Kalman Filter. In this part, we derive the multidimensional (multivariate) Kalman Filter equations.

This part deals with a Linear Kalman Filter (LKF). The LKF assumes that the system dynamics are linear. The next part describes non-linear Kalman Filters.

Until now, we've dealt with one-dimensional processes, like estimating the liquid temperature. But many dynamic processes have two, three, or even more dimensions.

For instance, the **state vector** that describes the airplane's position in space is three-dimensional:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{6.1}$$

The state vector that describes the airplane position and velocity is six-dimensional:

$$\begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} \tag{6.2}$$

The state vector that describes the airplane position, velocity, and acceleration is nine-dimensional:

$$\begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \\ \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} \tag{6.3}$$

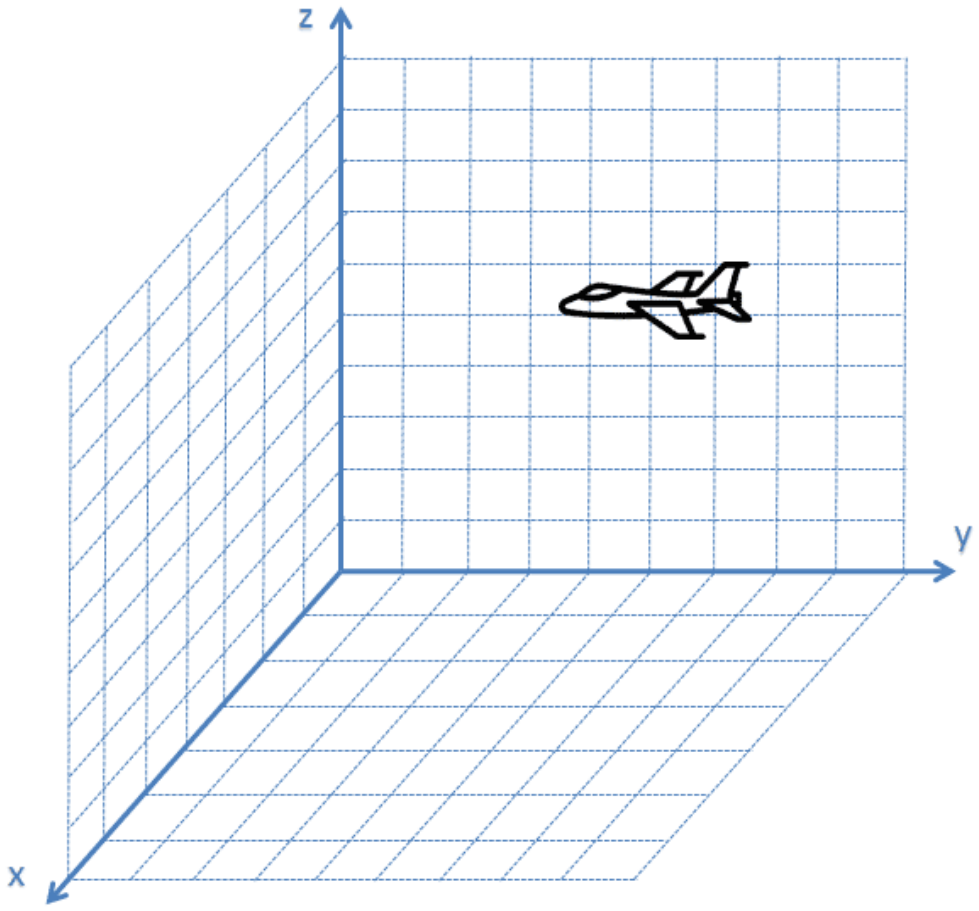


**Figure 6.1:** *Airplane in 3D.*

Assuming a constant acceleration dynamic model, we can describe the extrapolated airplane state at time $n$ by nine motion equations:

$$
\begin{cases}
x_n = x_{n-1} + \dot{x}_{n-1}\Delta t + \frac{1}{2}\ddot{x}_{n-1}\Delta t^2 \\
y_n = y_{n-1} + \dot{y}_{n-1}\Delta t + \frac{1}{2}\ddot{y}_{n-1}\Delta t^2 \\
z_n = z_{n-1} + \dot{z}_{n-1}\Delta t + \frac{1}{2}\ddot{z}_{n-1}\Delta t^2 \\
\dot{x}_n = \dot{x}_{n-1} + \ddot{x}_{n-1}\Delta t \\
\dot{y}_n = \dot{y}_{n-1} + \ddot{y}_{n-1}\Delta t \\
\dot{z}_n = \dot{z}_{n-1} + \ddot{z}_{n-1}\Delta t \\
\ddot{x}_n = \ddot{x}_{n-1} \\
\ddot{y}_n = \ddot{y}_{n-1} \\
\ddot{z}_n = \ddot{z}_{n-1}
\end{cases}
\tag{6.4}
$$

It is common practice to describe a multidimensional process with a single equation in matrix form.

First, it is very exhausting to write all these equations; representing them in matrix notation is much shorter and more elegant.

Second, computers are highly efficient at matrix calculations. Implementing the Kalman Filter in matrix form yields faster computation run time.

The following chapters describe the Kalman Filter equations in matrix form. And, of course, the theoretical part is followed by fully solved numerical examples.

The final chapter includes two numerical examples. In the first example, we design a six-dimensional Kalman Filter without control input. In the second example, we design a two-dimensional Kalman Filter with a control input.

# 7. Essential background II

Before we tackle the multidimensional Kalman Filter, we'll need to review some essential math topics:

- Matrix operations.
- Expectation algebra.
- Multivariate Normal Distribution.

You can jump to chapter 8 if you are familiar with these topics.

The notation used in this book:

- Bold-face, lower-case letters refer to vectors, such as $\boldsymbol{x}$.
- Bold-face, capital letters refer to matrices, such as $\boldsymbol{A}$.
- Normal-face lower-case letters refer to scalars or vector elements.
- Normal-face capital letters refer to matrix elements.

## 7.1 Matrix operations

All you need to know is basic terms and operations such as:

- Vector and matrix addition and multiplication.
- Matrix Transpose.
- Matrix Inverse (you don't need to invert matrices by yourself, you just need to know what the inverse of the matrix is).
- Symmetric Matrix.
- Eigenvalues and eigenvectors.

There are numerous Linear Algebra textbooks and web tutorials that cover these topics.

## 7.2 Expectation algebra

I extensively use the expectation algebra rules for Kalman Filter equations derivations. If you are interested in understanding the derivations, you need to master expectation algebra.

You already know what a random variable is and what an expected value (or expectation) is. If not, please read chapter 2 - "Essential background I."

### 7.2.1 Basic expectation rules

The expectation is denoted by the capital letter $E$.

The expectation $E(X)$ of the random variable $X$ equals the mean of the random variable:

**Expectation of the random variable**

$$E(X) = \mu_X \tag{7.1}$$

Where $\mu_X$ is the mean of the random variable.

Here are some basic expectation rules:

| | Rule | Notes |
|---|---|---|
| **1** | $E(X) = \mu_X = \Sigma x p(x)$ | $p(x)$ is the probability of $x$ (discrete case) |
| **2** | $E(a) = a$ | $a$ is constant |
| **3** | $E(aX) = aE(X)$ | $a$ is constant |
| **4** | $E(a \pm X) = a \pm E(X)$ | $a$ is constant |
| **5** | $E(a \pm bX) = a \pm bE(X)$ | $a$ and $b$ are constant |
| **6** | $E(X \pm Y) = E(X) \pm E(Y)$ | $Y$ is another random variable |
| **7** | $E(XY) = E(X)E(Y)$ | If $X$ and $Y$ are independent |

**Table 7.1:** *Expectation rules.*

### 7.2.2 Variance and Covariance expectation rules

The following table includes the variance and covariance expectation rules.

| | Rule | Notes |
|---|---|---|
| **8** | $V(a) = 0$ | $V(a)$ is the variance of $a$<br>$a$ is constant |
| **9** | $V(a \pm X) = V(X)$ | $V(X)$ is the variance of $X$<br>$a$ is constant |
| **10** | $V(X) = E(X^2) - \mu_X^2$ | $V(X)$ is the variance of $X$ |
| **11** | $COV(X, Y) = E(XY) - \mu_X \mu_Y$ | $COV(X, Y)$ is a covariance of $X$ and $Y$ |
| **12** | $COV(X, Y) = 0$ | if $X$ and $Y$ are independent |
| **13** | $V(aX) = a^2 V(X)$ | $a$ is constant |
| **14** | $V(X \pm Y) =$<br>$V(X) + V(Y) \pm 2COV(X, Y)$ | |
| **15** | $V(XY) \neq V(X)V(Y)$ | |

**Table 7.2:** *Variance and covariance expectation rules.*

The variance and covariance expectation rules are not straightforward. I prove some of them.

#### Rule 8

$$V(a) = 0 \tag{7.2}$$

A constant does not vary, so the variance of a constant is 0.

#### Rule 9

$$V(a \pm X) = V(X) \tag{7.3}$$

Adding a constant to the variable does not change its variance.

### Rule 10

$$V(X) = E(X^2) - \mu_X^2 \tag{7.4}$$

The proof

| Equation | Notes |
|---|---|
| $V(X) = \sigma_X^2 = E((X - \mu_X)^2)$ | |
| $= E(X^2 - 2X\mu_X + \mu_X^2)$ | |
| $= E(X^2) - E(2X\mu_X) + E(\mu_X^2)$ | Applied rule number 5: $E(a \pm bX) = a \pm bE(X)$ |
| $= E(X^2) - 2\mu_X E(X) + E(\mu_X^2)$ | Applied rule number 3: $E(aX) = aE(X)$ |
| $= E(X^2) - 2\mu_X E(X) + \mu_X^2$ | Applied rule number 2: $E(a) = a$ |
| $= E(X^2) - 2\mu_X \mu_X + \mu_X^2$ | Applied rule number 1: $E(X) = \mu_X$ |
| $= E(X^2) - \mu_X^2$ | |

**Table 7.3:** *Variance expectation rule.*

### Rule 11

$$COV(X,Y) = E(XY) - \mu_X \mu_Y \tag{7.5}$$

The proof

| Equation | Notes |
|---|---|
| $COV(X,Y) = E((X-\mu_X)(Y-\mu_Y)$ | |
| $= E(XY - X\mu_Y - Y\mu_X + \mu_X\mu_Y)$ | |
| $= E(XY) - E(X\mu_Y) - E(Y\mu_X) + E(\mu_X\mu_Y)$ | Applied rule number 6: $E(X \pm Y) = E(X) \pm E(Y)$ |
| $= E(XY) - \mu_Y E(X) - \mu_X E(Y) + E(\mu_X\mu_Y)$ | Applied rule number 3: $E(aX) = aE(X)$ |
| $= E(XY) - \mu_Y E(X) - \mu_X E(Y) + \mu_X\mu_Y$ | Applied rule number 2: $E(a) = a$ |
| $= E(XY) - \mu_Y\mu_X - \mu_X\mu_Y + \mu_X\mu_Y$ | Applied rule number 1: $E(X) = \mu_X$ |
| $= E(XY) - \mu_X\mu_Y$ | |

**Table 7.4:** *Covariance expectation rule.*

### Rule 13

$$V(aX) = a^2V(X) \tag{7.6}$$

The proof

| Equation | Notes |
|---|---|
| $V(K) = \sigma_K^2 = E(K^2) - \mu_K^2$ | |
| $K = aX$ | |
| $V(K) = V(aX) = E((aX)^2) - (a\mu_X)^2$ | Substitute $K$ with $aX$ |
| $= E(a^2X^2) - a^2\mu_X^2$ | |
| $= a^2E(X^2) - a^2\mu_X^2$ | Applied rule number 3: $E(aX) = aE(X)$ |
| $= a^2(E(X^2) - \mu_X^2)$ | |
| $= a^2V(X)$ | Applied rule number 10: $V(X) = E(X^2) - \mu_X^2$ |

**Table 7.5:** *Variance square expectation rule.*

For constant velocity motion:

$$V(x) = \Delta t^2 V(v) \tag{7.7}$$

or

$$\sigma_x^2 = \Delta t^2 \sigma_v^2 \tag{7.8}$$

Where:
$x$ is the displacement of the body
$v$ is the velocity of the body
$\Delta t$ is the time interval

### Rule 14

$$V(X \pm Y) = V(X) + V(Y) \pm 2COV(X, Y) \tag{7.9}$$

The proof

| Equation | Notes |
|---|---|
| $V(X \pm Y)$ | |
| $= E((X \pm Y)^2) - (\mu_X \pm \mu_Y)^2$ | Applied rule number 10: $V(X) = E(X^2) - \mu_X^2$ |
| $= E(X^2 \pm 2XY + Y^2) - (\mu_X^2 \pm 2\mu_X\mu_Y + \mu_y^2)$ | |
| $= E(X^2) - \mu_X^2 + E(Y^2) - \mu_Y^2 \pm 2(E(XY) - \mu_X\mu_Y)$ | Applied rule number 6: $E(X \pm Y) = E(X) \pm E(Y)$ |
| $= V(X) + V(Y) \pm 2(E(XY) - \mu_X\mu_Y)$ | Applied rule number 10: $V(X) = E(X^2) - \mu_X^2$ |
| $= V(X) + V(Y) \pm 2COV(X, Y)$ | Applied rule number 11: $COV(X, Y) = E(XY) - \mu_X\mu_Y$ |

**Table 7.6:** *Variance sum expectation rule.*

## 7.3 Multivariate Normal Distribution

### 7.3.1 Introduction

We've seen that the Kalman Filter output is a random variable. The mean of the random variable is the state estimate. The variance of the random variable represents the estimation uncertainty. The Kalman Filter provides us with the estimate and the level of confidence of its estimate.

The one-dimensional Kalman Filter equations include four uncertainty variables:

| | |
|---|---|
| $p_{n,n}$ | is the variance of an estimate (the current state) |
| $p_{n+1,n}$ | is the variance of a prediction (the next state) |
| $r_n$ | is the measurement variance |
| $q$ | is the process noise |

For a multivariate Kalman Filter, the system state is described by a vector with more than one variable. For example, the object's position on the plane can be described by two variables: x – position and y – position:

$$\boldsymbol{x} = \begin{bmatrix} x \\ y \end{bmatrix} \tag{7.10}$$

The Kalman Filter output is a **multivariate random variable**. A **covariance matrix** describes the squared uncertainty of the multivariate random variable.

The uncertainty variables of the multivariate Kalman Filter are:

| | |
|---|---|
| $\boldsymbol{P}_{n,n}$ | is a covariance matrix that describes the squared uncertainty of an estimate |
| $\boldsymbol{P}_{n+1,n}$ | is a covariance matrix that describes the squared uncertainty of a prediction |
| $\boldsymbol{R}_n$ | is a covariance matrix that describes the squared measurement uncertainty |
| $\boldsymbol{Q}$ | is a covariance matrix that describes the process noise |

This chapter is about the **multivariate normal distribution** and the covariance matrix.

### 7.3.2 Covariance

Covariance is a measure of the strength of the correlation between two or more sets of random variates.

Assume a set of object location measurements on the $x - y$ plane.

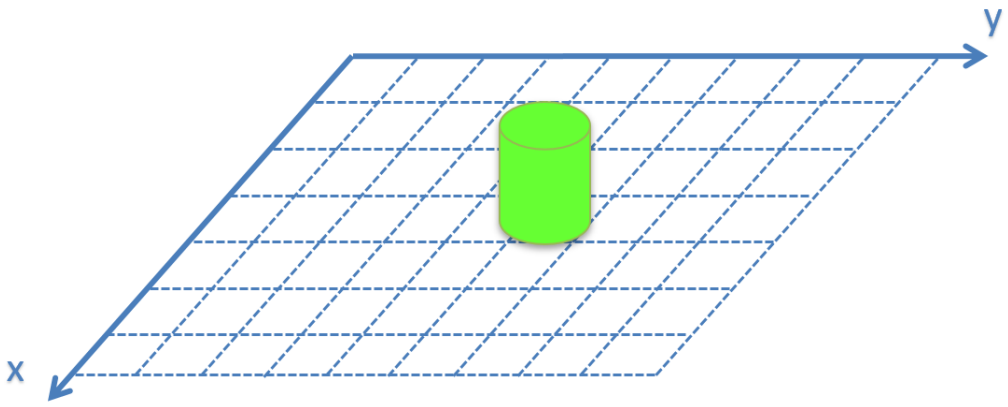


**Figure 7.1:** *Object on the x-y plane.*

Due to the random error, there is a variance in the measurements.

Let us see some examples of different measurement sets.

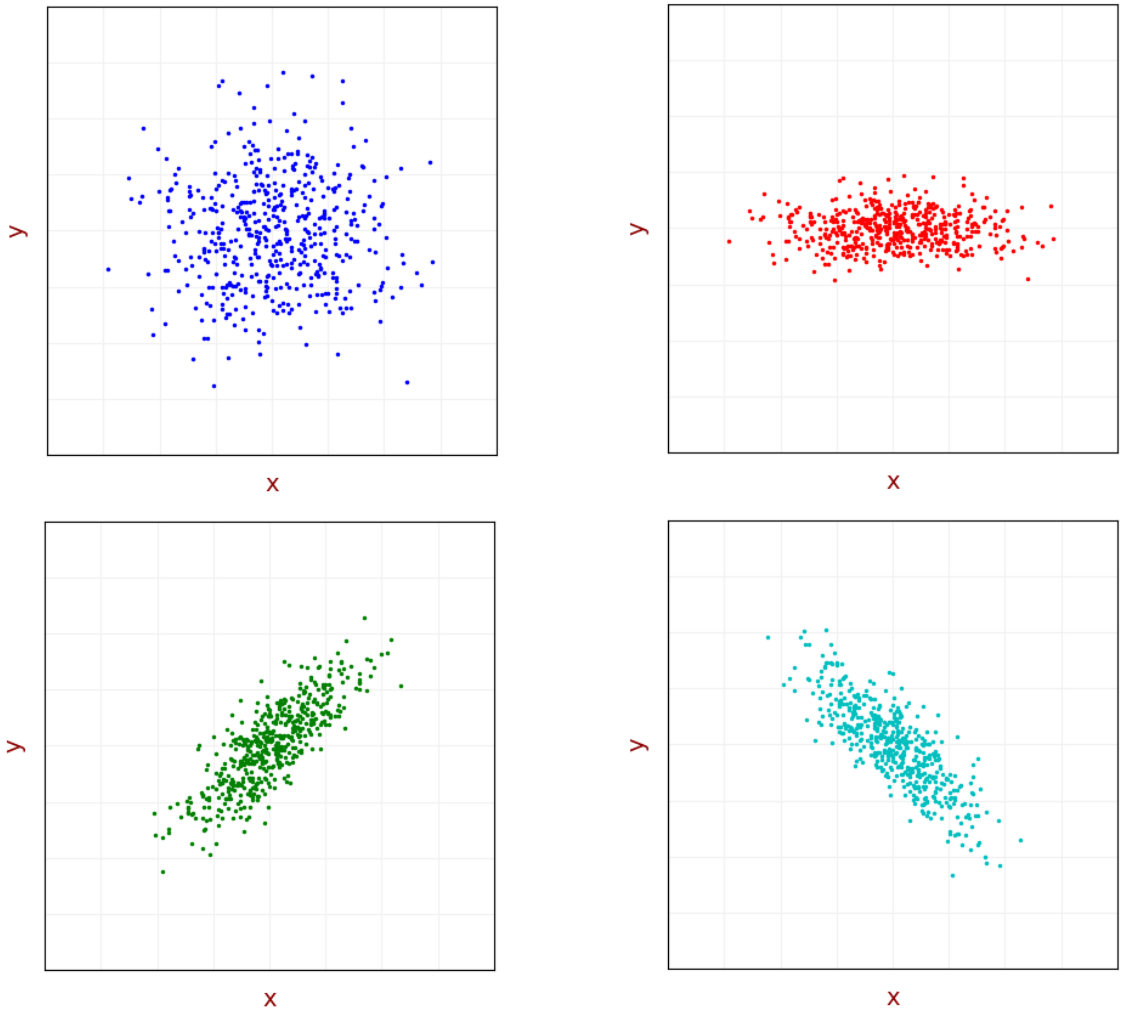


**Figure 7.2:** *Examples of different measurement sets.*

The two upper subplots demonstrate uncorrelated measurements. The $x$ and $y$ values

don't depend on each other. The $x$ and $y$ values of the blue data set have the same variance, and we can see a circular distribution shape. For the red data set, the variance of the $x$ values is greater than that of the $y$ values, and the distribution shape is elliptic. Since the measurements are uncorrelated, the covariance of $x$ and $y$ equals zero.

The two lower subplots demonstrate correlated measurements. There is a dependency between $x$ and $y$ values. For the green data set, an increase in $x$ results in an increase in $y$ and vice versa. The correlation is positive; therefore, the covariance is positive. For the cyan data set, an increase in $x$ results in a decrease in $y$ and vice versa. The correlation is negative; therefore, the covariance is negative.

The covariance between population $X$ and population $Y$ with size $N$ is given by:

$$COV(X,Y) = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_x)(y_i - \mu_y) \tag{7.11}$$

Let us rewrite the covariance equation:

| **Equation** | **Notes** |
|---|---|
| $COV(X,Y) = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_x)(y_i - \mu_y)$ | |
| $= \frac{1}{N}\sum_{i=1}^{N}(x_i y_i - x_i\mu_y - y_i\mu_x + \mu_x\mu_y)$ | Open parenthesis |
| $= \frac{1}{N}\sum_{i=1}^{N}(x_i y_i) - \frac{1}{N}\sum_{i=1}^{N}(x_i\mu_y) - \frac{1}{N}\sum_{i=1}^{N}(y_i\mu_x) + \frac{1}{N}\sum_{i=1}^{N}(\mu_x\mu_y)$ | Distribute |
| $= \frac{1}{N}\sum_{i=1}^{N}(x_i y_i) - \frac{\mu_y}{N}\sum_{i=1}^{N}(x_i) - \frac{\mu_x}{N}\sum_{i=1}^{N}(y_i) + \mu_x\mu_y$ | $\mu_x = \frac{1}{N}\sum_{i=1}^{N}(x_i)$<br>$\mu_y = \frac{1}{N}\sum_{i=1}^{N}(y_i)$ |
| $= \frac{1}{N}\sum_{i=1}^{N}(x_i y_i) - \mu_x\mu_y - \mu_x\mu_y + \mu_x\mu_y$ | |
| $= \frac{1}{N}\sum_{i=1}^{N}(x_i y_i) - \mu_x\mu_y$ | |

**Table 7.7:** *Covariance equation.*

The covariance of a sample with size $N$ is normalized by $N-1$:

$$COV(X,Y) = \frac{1}{N-1}\sum_{i=1}^{N}(x_i-\mu_x)(y_i-\mu_y) \tag{7.12}$$

| **Equation** | **Notes** |
|---|---|
| $COV(X,Y) = \frac{1}{N-1}\sum_{i=1}^{N}(x_i-\mu_x)(y_i-\mu_y)$ | |
| $= \frac{1}{N-1}\sum_{i=1}^{N}(x_iy_i - x_i\mu_y - y_i\mu_x + \mu_x\mu_y)$ | Open parenthesis |
| $= \frac{1}{N-1}\sum_{i=1}^{N}(x_iy_i) - \frac{1}{N-1}\sum_{i=1}^{N}(x_i\mu_y)$ $- \frac{1}{N-1}\sum_{i=1}^{N}(y_i\mu_x) + \frac{1}{N-1}\sum_{i=1}^{N}(\mu_x\mu_y)$ | Distribute |
| $= \frac{1}{N-1}\sum_{i=1}^{N}(x_iy_i) - \frac{\mu_y}{N-1}\sum_{i=1}^{N}(x_i)$ $- \frac{\mu_x}{N-1}\sum_{i=1}^{N}(y_i) + \frac{N}{N-1}\mu_x\mu_y$ | $\mu_x = \frac{1}{N}\sum_{i=1}^{N}(x_i)$ $\mu_y = \frac{1}{N}\sum_{i=1}^{N}(y_i)$ |
| $= \frac{1}{N-1}\sum_{i=1}^{N}(x_iy_i) - \frac{N}{N-1}\mu_x\mu_y - \frac{N}{N-1}\mu_x\mu_y + \frac{N}{N-1}\mu_x\mu_y$ | |
| $= \frac{1}{N-1}\sum_{i=1}^{N}(x_iy_i) - \frac{N}{N-1}\mu_x\mu_y$ | |

**Table 7.8:** *Sample covariance equation.*

### Example

Given samples:

$$\boldsymbol{x} = \begin{bmatrix} 2 \\ 3 \\ -1 \\ 4 \end{bmatrix} \quad \boldsymbol{y} = \begin{bmatrix} 8 \\ 7 \\ 9 \\ 6 \end{bmatrix}$$

$$COV(X,Y) = \frac{1}{N-1}\sum_{i=1}^{N}(x_i y_i) - \frac{N}{N-1}\mu_x\mu_y$$
$$= \frac{1}{3}(2\times 8 + 3\times 7 - 1\times 9 + 4\times 6) - \frac{4}{3}\left(\frac{(2+3-1+4)}{4}\frac{(8+7+9+6)}{4}\right) = -2.67$$

We can stack the samples in two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$. The covariance in vector notation is given by:

$$COV(X,Y) = \frac{1}{N-1}\boldsymbol{x}^T\boldsymbol{y} - \frac{N}{N-1}\mu_x\mu_y \tag{7.13}$$

For a zero-mean random variable, the covariance is given by:

$$COV(X,Y) = \frac{1}{N-1}\boldsymbol{x}^T\boldsymbol{y} \tag{7.14}$$

### 7.3.3 Covariance matrix

A covariance matrix is a square matrix that represents the covariance between each pair of elements in a given multivariate random variable.

For a two-dimensional random variable, the covariance matrix is:

$$\boldsymbol{\Sigma} = \begin{bmatrix}\sigma_{xx} & \sigma_{xy}\\ \sigma_{yx} & \sigma_{yy}\end{bmatrix} = \begin{bmatrix}\sigma_x^2 & \sigma_{xy}\\ \sigma_{yx} & \sigma_y^2\end{bmatrix} = \begin{bmatrix}VAR(\boldsymbol{x}) & COV(\boldsymbol{x},\boldsymbol{y})\\ COV(\boldsymbol{y},\boldsymbol{x}) & VAR(\boldsymbol{y})\end{bmatrix} \tag{7.15}$$

Note that the off-diagonal entries of the covariance matrix are equal since $COV(\boldsymbol{x},\boldsymbol{y}) = COV(\boldsymbol{y},\boldsymbol{x})$. If $x$ and $y$ are uncorrelated, the off-diagonal entries of the covariance matrix are zero.

For a $n$ - dimensional random variable, the covariance matrix is:

$$\boldsymbol{\Sigma} = \begin{bmatrix}\sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1n}\\ \sigma_{21} & \sigma_2^2 & \cdots & \sigma_{2n}\\ \vdots & \vdots & \ddots & \vdots\\ \sigma_{n1} & \sigma_{n2} & \cdots & \sigma_n^2\end{bmatrix} \tag{7.16}$$

Most scientific computer packages can compute the covariance matrix.

**Python example:**

```
import numpy as np

x = np.array([2, 3, -1, 4])
y = np.array([8, 7, 9, 6])

C = np.cov(x,y)
print(C)

[[ 4.66666667 -2.66666667]
 [-2.66666667  1.66666667]]
```

**MATLAB example:**

```
x = [2 3 -1 4];
y = [8 7 9 6];

C = cov(x,y)

C =

    4.6667   -2.6667
   -2.6667    1.6667
```

#### 7.3.3.1 Properties of the covariance matrix

1. The diagonal entries of this covariance matrix are the variances of the components of the multivariate random variable.

$$\Sigma_{ii} = \sigma_i^2 \tag{7.17}$$

2. Since the diagonal entries are all non-negative, the trace (the sum of diagonal entries) of this covariance matrix is non-negative:

$$tr(\boldsymbol{\Sigma}) = \sum_{i=1}^{n} \Sigma_{ii} \geq 0 \tag{7.18}$$

3. Since $\Sigma_{ij} = \sigma_{ij} = \sigma_{ji} = \Sigma_{ji}$, the covariance matrix is symmetric:

$$\boldsymbol{\Sigma} = \boldsymbol{\Sigma}^T \tag{7.19}$$

4. The covariance matrix is positive semidefinite.
   The matrix $\boldsymbol{A}$ is called positive semidefinite if $\boldsymbol{v}^T \boldsymbol{A} \boldsymbol{v} \geq 0$, for any vector $\boldsymbol{v}$.
   The eigenvalues of $\boldsymbol{A}$ are non-negative.

#### 7.3.3.2 Covariance matrix and expectation

Assume vector $\boldsymbol{x}$ with $k$ elements:

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{bmatrix} \tag{7.20}$$

**The covariance matrix of the vector $x$**

$$COV(\boldsymbol{x}) = E\left((\boldsymbol{x} - \boldsymbol{\mu}_x)(\boldsymbol{x} - \boldsymbol{\mu}_x)^T\right) \tag{7.21}$$

Where $\mu_X$ is the mean of the random variable.

The proof

$$COV(\boldsymbol{x}) = E\left(\begin{bmatrix} (x_1 - \mu_{x_1})^2 & (x_1 - \mu_{x_1})(x_2 - \mu_{x_2}) & \cdots & (x_1 - \mu_{x_1})(x_k - \mu_{x_k}) \\ (x_2 - \mu_{x_2})(x_1 - \mu_{x_1}) & (x_2 - \mu_{x_2})^2 & \cdots & (x_2 - \mu_{x_2})(x_k - \mu_{x_k}) \\ \vdots & \vdots & \ddots & \vdots \\ (x_k - \mu_{x_k})(x_1 - \mu_{x_1}) & (x_k - \mu_{x_k})(x_2 - \mu_{x_2}) & \cdots & (x_k - \mu_{x_k})^2 \end{bmatrix}\right)$$

$$= E\left(\begin{bmatrix} (x_1 - \mu_{x_1}) \\ (x_2 - \mu_{x_2}) \\ \vdots \\ (x_k - \mu_{x_k}) \end{bmatrix} \begin{bmatrix} (x_1 - \mu_{x_1}) & (x_2 - \mu_{x_2}) & \cdots & (x_k - \mu_{x_k}) \end{bmatrix}\right)$$

$$= E\left((\boldsymbol{x} - \boldsymbol{\mu}_x)(\boldsymbol{x} - \boldsymbol{\mu}_x)^T\right)$$

### 7.3.4 Multivariate normal distribution

We are already familiar with a univariate normal distribution. It is described by a bell-shaped Gaussian function:

$$p(x|\mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{7.22}$$

The normal distribution is denoted by:

$$\mathcal{N}(\mu, \sigma^2)$$

The multivariate normal distribution is a generalization of the univariate normal distribution for a multidimensional random variable.

The $n$ - dimensional multivariate normal distribution is described by:

$$p(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^n|\boldsymbol{\Sigma}|}} exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right) \tag{7.23}$$

Where:

- $\boldsymbol{x}$ is an $n$ - dimensional random vector
- $\boldsymbol{\mu}$ is an $n$ - dimensional mean vector
- $\boldsymbol{\Sigma}$ is a square $n \times n$ covariance matrix

### 7.3.5 Bivariate normal distribution

A bivariate (two-dimensional) normal distribution comprises two normally distributed random variables. I want to focus on the bivariate normal distribution since it is the highest dimension of the multivariate normal distribution that we can visualize.

The following plot describes the bivariate Gaussian function.

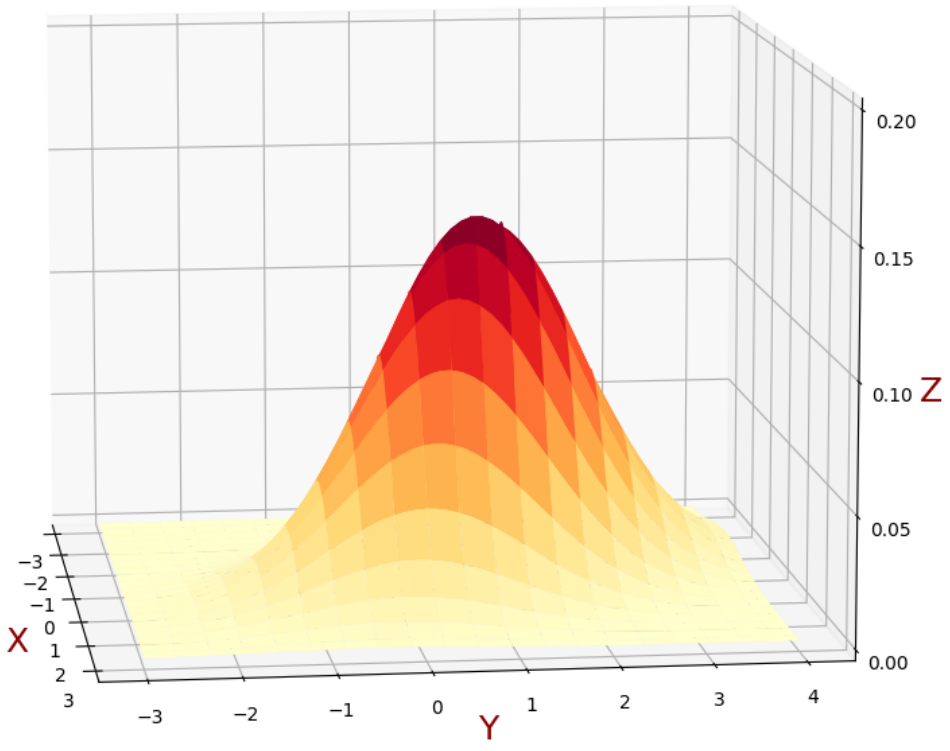


**Figure 7.3:** *Bivariate Gaussian.*

### 7.3.6 Confidence intervals

A confidence interval specifies the probability that a parameter falls between a pair of values around the mean for the univariate normal distribution.

For the univariate normal distribution, the area between the $1\sigma$ boundaries and the Gaussian function is 68.26% of the total area under the Gaussian function.

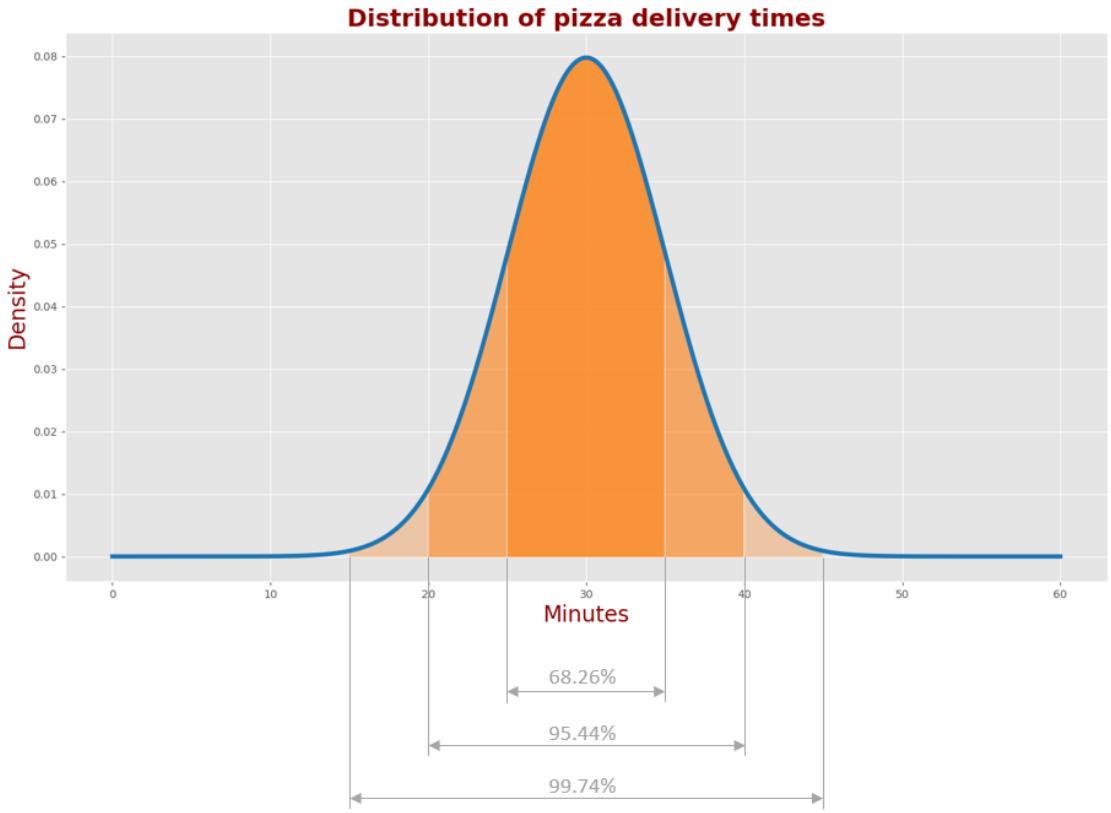


**Figure 7.4:** *Univariate Gaussian.*

For the univariate normal distribution, we can formulate the following statements:

- The confidence interval for the probability of 68.26% is $1\sigma$.
- The confidence interval for the probability of 95.44% is $2\sigma$.
- The confidence interval for the probability of 99.74% is $3\sigma$.

We can also find the confidence interval of any specified probability for the univariate normal distribution. You can see the explanation in Appendix B.

The probability of the bivariate normal distribution is a volume of the three-dimensional Gaussian function.

For example, the volume of the three-dimensional Gaussian function sliced at $1\sigma$ level is 39.35% of the total volume of the three-dimensional Gaussian function.

The projection of the three-dimensional Gaussian slice is an ellipse.

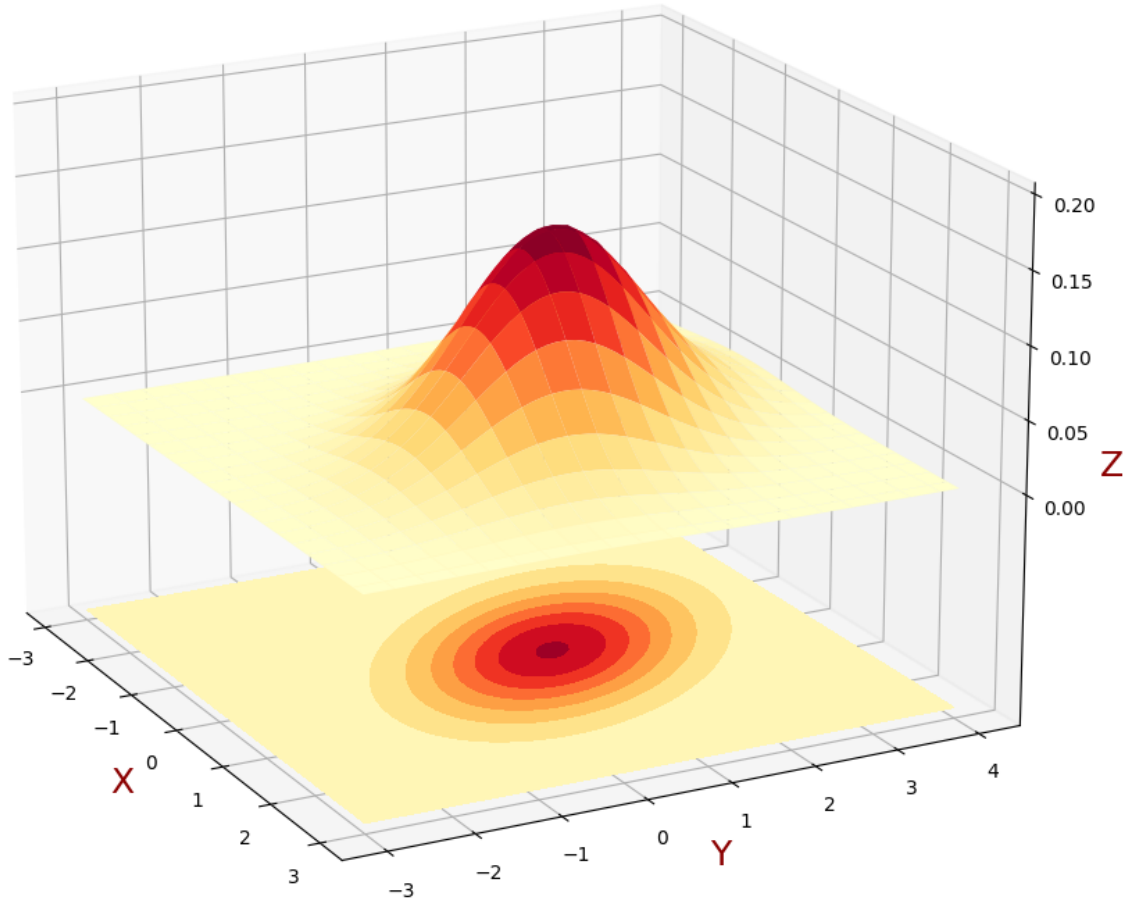


**Figure 7.5:** *Bivariate Gaussian with projection.*

### 7.3.7 Covariance ellipse

First, let us find the properties of the covariance ellipse. The covariance ellipse represents an iso-contour of the Gaussian distribution and allows visualization of a $1\sigma$ confidence interval in two dimensions. The covariance ellipse provides a geometric interpretation of the covariance matrix.

Any ellipse can be described by four parameters:

- Ellipse center $\mu_x, \mu_y$
- Half-major axis $a$
- Half-minor axis $b$
- Orientation angle $\theta$

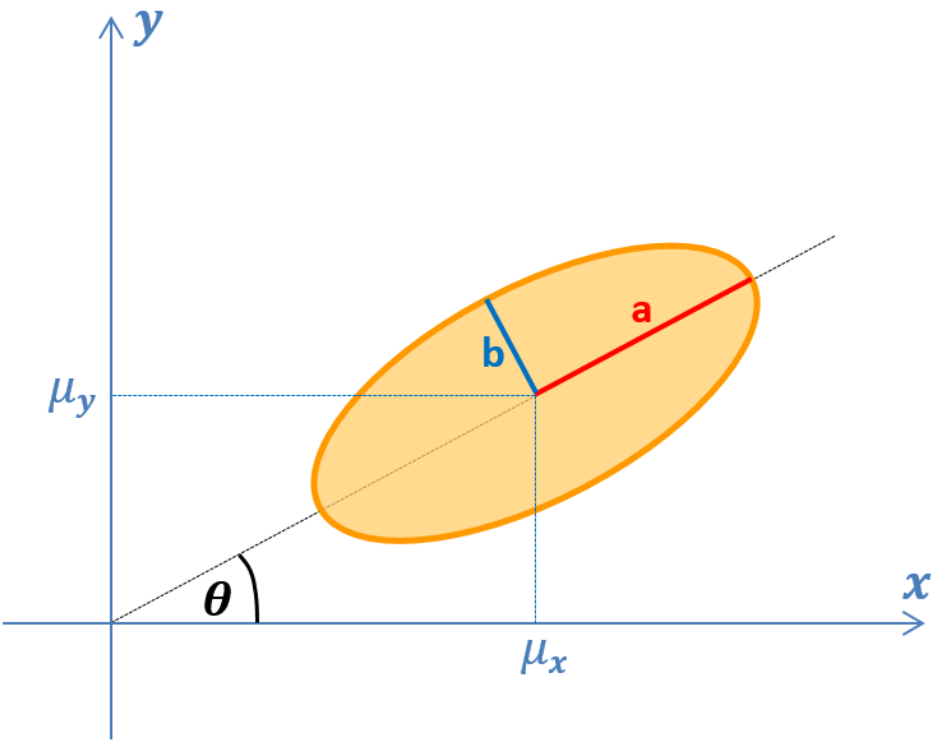


**Figure 7.6:** *Covariance ellipse.*

The ellipse center is a mean of the random variable:

$$\begin{aligned} \mu_x &= \frac{1}{N}\sum_{i=1}^{N} x_i \\ \mu_y &= \frac{1}{N}\sum_{i=1}^{N} y_i \end{aligned} \tag{7.24}$$

The lengths of the ellipse axes are the square roots of the eigenvalues of the random variable covariance matrix:

- The length of the half-major axis $a$ is given by the highest eigenvalue square root
- The length of the half-minor axis $b$ is given by the second eigenvalue square root

The orientation of the ellipse is an orientation of the covariance matrix eigenvector that corresponds to the highest eigenvalue.

$$\theta = arctan\left(\frac{v_y}{v_x}\right) \tag{7.25}$$

Where:

$v_x$ is the $x$ - coordinate of the eigenvector that corresponds to the highest eigenvalue

$v_y$ is the $y$ - coordinate of the eigenvector that corresponds to the highest eigenvalue

You can use a scientific computer package to compute the covariance ellipse parameters.

**Python example:**

```python
import numpy as np

C = np.array([[5, -2],[-2, 1]]) # define covariance matrix

eigVal, eigVec = np.linalg.eig(C) # find eigenvalues and
    eigenvectors

a = np.sqrt(eigVal[0]) # half-major axis length
b = np.sqrt(eigVal[1]) # half-minor axis length

# ellipse orientation angle
theta = np.arctan(eigVec[1, 0] / eigVec[0, 0])
```

**MATLAB example:**

```
C = [5 -2; -2 1]; % define covariance matrix

[eigVec, eigVal] = eig(C); % find eigenvalues and eigenvectors


if eigVal(1,1) > eigVal(2,2) % get the highest eigenvalue index

    a = sqrt(eigVal(1,1)); % half-major axis length
    b = sqrt(eigVal(2,2)); % half-minor axis length

    theta = atan(eigVec(2,1) / eigVec(1,1));  % ellipse angle (
    radians)
else

    a = sqrt(eigVal(2,2)); % half-major axis length
    b = sqrt(eigVal(1,1)); % half-minor axis length

    theta = atan(eigVec(2,2) / eigVec(2,1));  % ellipse angle (
    radians)
end
```

### 7.3.8 Confidence ellipse

In many applications, there is an interest in finding the boundaries of specific probability. For example, for 95% probability, we should find the boundary that includes 95% of the Gaussian function volume.

The projection of this boundary onto the $x - y$ plane is the confidence ellipse.

We want to find an **elliptical scale factor** $k$, that extends the covariance ellipse to the confidence ellipse associated with 95% probability.

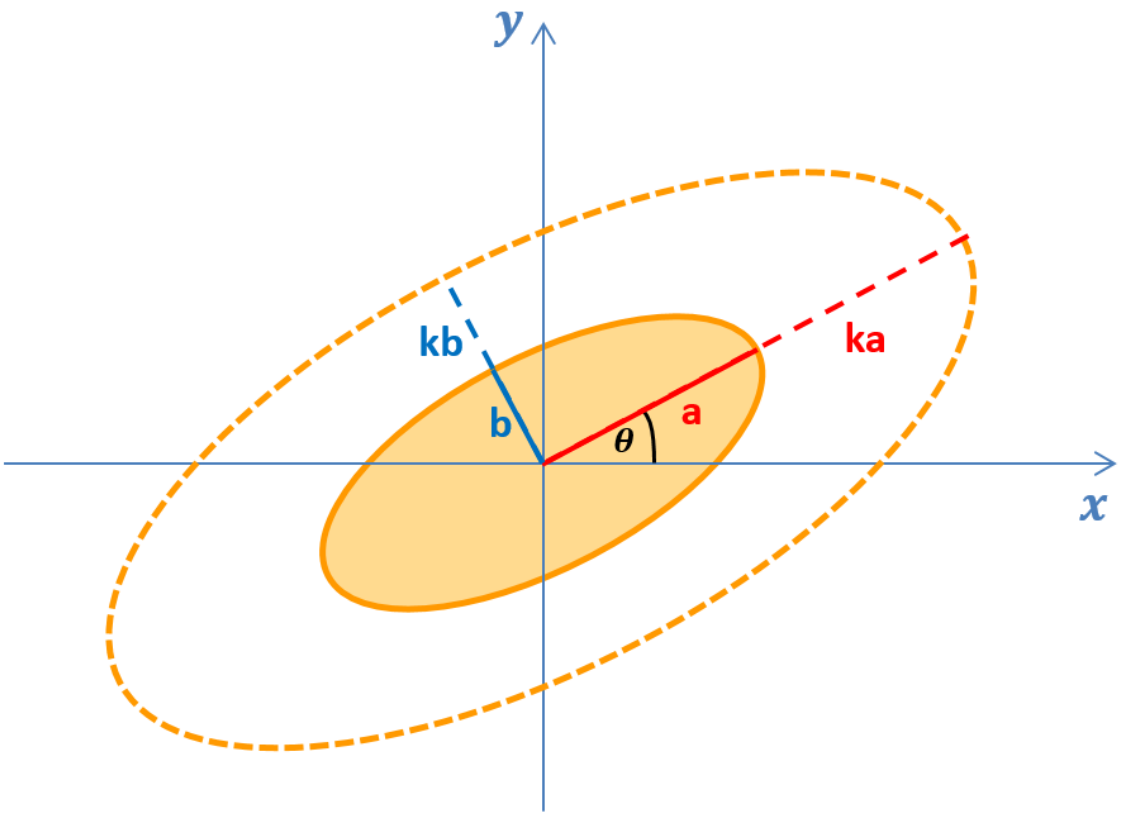


**Figure 7.7:** *Confidence ellipse.*

Since $\sigma_x$ and $\sigma_y$ represent the standard deviations of stochastically independent random variables, the addition theorem for the chi-square distribution may be used to show that the probability associated with a confidence ellipse is given by [4]:

$$p = 1 - exp\left(-\frac{1}{2}k^2\right) \tag{7.26}$$

For a covariance ellipse $k = 1$, therefore, the probability associated with a covariance

ellipse is:

$$p = 1 - exp\left(-\frac{1}{2}\right) = 39.35\% \tag{7.27}$$

For a given probability, we can find an elliptical scale factor:

$$k = \sqrt{-2ln(1-p)} \tag{7.28}$$

For the probability of 95%:

$$k = \sqrt{-2ln(1-0.95)} = 2.45 \tag{7.29}$$

The properties of the confidence ellipse associated with 95% probability are:

- Ellipse center $(\mu_x, \mu_y)$ is similar to the covariance ellipse.
- Orientation angle $\theta$ is similar to the covariance ellipse.
- Half-major axis length is $2.45a$ – a scaled half-major axis of the covariance ellipse.
- Half-minor axis length is $2.45b$ – a scaled half-minor axis of the covariance ellipse.

# 8. Kalman Filter Equations Derivation

## 8.1 State Extrapolation Equation

The first Kalman Filter equation that I would like to describe is the **state extrapolation equation**.

Using the state extrapolation equation, we can predict the next system state based on the knowledge of the current state. It extrapolates the state vector from the present (time step $n$) to the future (time step $n+1$).

The state extrapolation equation describes the model of the dynamic system.

The general form of the state extrapolation equation in a matrix notation is:

**State Extrapolation Equation**

$$\hat{\boldsymbol{x}}_{n+1,n} = \boldsymbol{F}\hat{\boldsymbol{x}}_{n,n} + \boldsymbol{G}\boldsymbol{u}_n + \boldsymbol{w}_n \tag{8.1}$$

Where:

| | |
|---|---|
| $\hat{\boldsymbol{x}}_{n+1,n}$ | is a predicted system state vector at time step $n+1$ |
| $\hat{\boldsymbol{x}}_{n,n}$ | is an estimated system state vector at time step $n$ |
| $\boldsymbol{u}_n$ | is a **control variable** or **input variable** - a measurable (deterministic) input to the system |
| $\boldsymbol{w}_n$ | is a **process noise** or disturbance - an unmeasurable input that affects the state |
| $\boldsymbol{F}$ | is a **state transition matrix** |
| $\boldsymbol{G}$ | is a **control matrix** or **input transition matrix** (mapping control to state variables) |

R In the literature, the state transition matrix $F$ is sometimes denoted by a Greek letter $\Phi$.

In the literature, state extrapolation equation is also called:

- Predictor Equation

- Transition Equation
- Prediction Equation
- Dynamic Model
- State Space Model

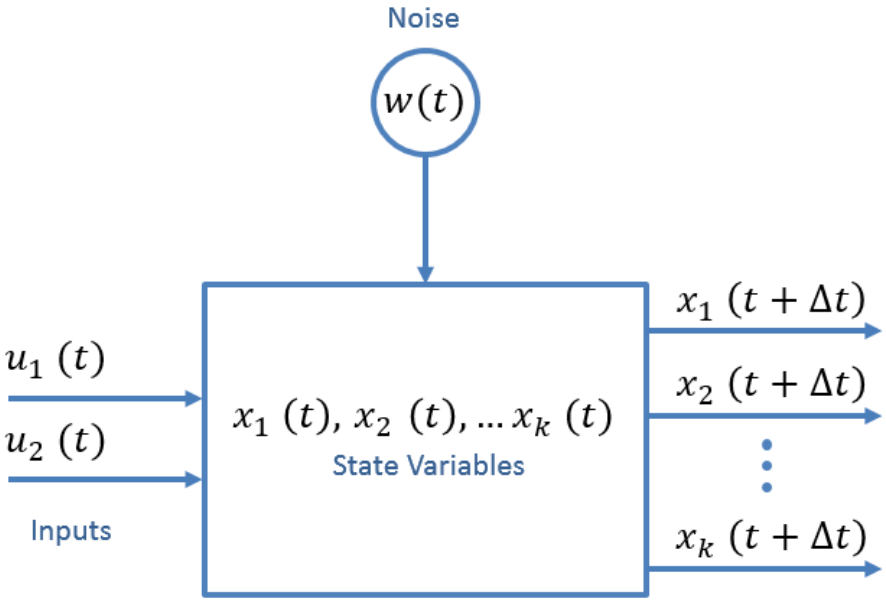


**Figure 8.1:** *Kalman Filter Extrapolation.*

The state variables may represent attributes of the system that we wish to know.

For example, a moving vehicle has three attributes: position, velocity, and acceleration.

You might ask yourself, which attributes are the state variables, and which attributes are the input to the system?

- Moving mechanical systems have attributes such as position, velocity, acceleration, and drag.
- A force that acts on a system should be considered an external forcing function, i.e., an input to the system that controls the state vector (position and velocity in the constant acceleration case).
- Newton's second law tells us that $F = ma$. Thus we can consider acceleration as an external input to the system.
- The position and the velocity are the primary state variables of interest.

For instance, in a spring system, the force applied to the spring $F(t)$ is an input $u(t)$, while the spring displacement $x(t)$ is the system state.

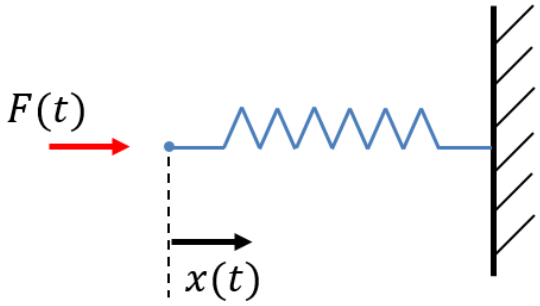


**Figure 8.2:** *Spring System.*

For a falling object, the inputs are the gravitational force $F_g$ and the drag force $F_{drag}(t)$, while the object height $h(t)$ and velocity $v(t)$ are the system states.

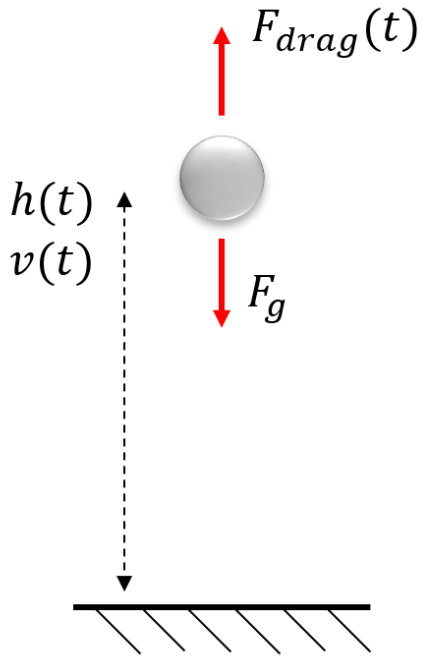


**Figure 8.3:** *Falling Object.*

R The process noise $w_n$ does not typically appear directly in the equations of interest. Instead, this term is used to model the uncertainty in the Covariance Extrapolation Equation.

Let's take a look at several examples of the state extrapolation equation.

### 8.1.1 Example - airplane - no control input

In this example, we define the State Extrapolation Equation for an airplane, assuming a constant acceleration model

In this example, there is no control input.

$$\boldsymbol{u}_n = 0 \tag{8.2}$$

Consider an airplane moving in three-dimensional space with constant acceleration. The state vector $\hat{\boldsymbol{x}}_{\boldsymbol{n}}$ that describes the estimated airplane position, velocity, and acceleration in a cartesian coordinate system $(x, y, z)$ is: In this example, there is no control input.

$$\hat{\boldsymbol{x}}_n = \begin{bmatrix} \hat{x}_n \\ \hat{y}_n \\ \hat{z}_n \\ \hat{\dot{x}}_n \\ \hat{\dot{y}}_n \\ \hat{\dot{z}}_n \\ \hat{\ddot{x}}_n \\ \hat{\ddot{y}}_n \\ \hat{\ddot{z}}_n \end{bmatrix} \tag{8.3}$$

R Don't confuse the estimated state vector $\hat{\boldsymbol{x}}_{\boldsymbol{n}}$ (bold-face font) and the estimated airplane position at $x$ - axis denoted by $\hat{x}_n$ (normal-face font). Both variables are denoted by the same letter in the literature, and I am trying to be consistent.

The state transition matrix $\boldsymbol{F}$ is:

$$\boldsymbol{F} = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 & 0.5\Delta t^2 & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 & 0 & 0.5\Delta t^2 & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t & 0 & 0 & 0.5\Delta t^2 \\ 0 & 0 & 0 & 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{8.4}$$

The state extrapolation equation is:

$$\begin{bmatrix} \hat{x}_{n+1,n} \\ \hat{y}_{n+1,n} \\ \hat{z}_{n+1,n} \\ \hat{\dot{x}}_{n+1,n} \\ \hat{\dot{y}}_{n+1,n} \\ \hat{\dot{z}}_{n+1,n} \\ \hat{\ddot{x}}_{n+1,n} \\ \hat{\ddot{y}}_{n+1,n} \\ \hat{\ddot{z}}_{n+1,n} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 & 0.5\Delta t^2 & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 & 0 & 0.5\Delta t^2 & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t & 0 & 0 & 0.5\Delta t^2 \\ 0 & 0 & 0 & 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{x}_{n,n} \\ \hat{y}_{n,n} \\ \hat{z}_{n,n} \\ \hat{\dot{x}}_{n,n} \\ \hat{\dot{y}}_{n,n} \\ \hat{\dot{z}}_{n,n} \\ \hat{\ddot{x}}_{n,n} \\ \hat{\ddot{y}}_{n,n} \\ \hat{\ddot{z}}_{n,n} \end{bmatrix} \tag{8.5}$$

The matrix multiplication results:

$$\begin{cases} \hat{x}_{n+1,n} = \hat{x}_{n,n} + \hat{\dot{x}}_{n,n}\Delta t + \frac{1}{2}\hat{\ddot{x}}_{n,n}\Delta t^2 \\ \hat{y}_{n+1,n} = \hat{y}_{n,n} + \hat{\dot{y}}_{n,n}\Delta t + \frac{1}{2}\hat{\ddot{y}}_{n,n}\Delta t^2 \\ \hat{z}_{n+1,n} = \hat{z}_{n,n} + \hat{\dot{z}}_{n,n}\Delta t + \frac{1}{2}\hat{\ddot{z}}_{n,n}\Delta t^2 \\ \hat{\dot{x}}_{n+1,n} = \hat{\dot{x}}_{n,n} + \hat{\ddot{x}}_{n,n}\Delta t \\ \hat{\dot{y}}_{n+1,n} = \hat{\dot{y}}_{n,n} + \hat{\ddot{y}}_{n,n}\Delta t \\ \hat{\dot{z}}_{n+1,n} = \hat{\dot{z}}_{n,n} + \hat{\ddot{z}}_{n,n}\Delta t \\ \hat{\ddot{x}}_{n+1,n} = \hat{\ddot{x}}_{n,n} \\ \hat{\ddot{y}}_{n+1,n} = \hat{\ddot{y}}_{n,n} \\ \hat{\ddot{z}}_{n+1,n} = \hat{\ddot{z}}_{n,n} \end{cases} \tag{8.6}$$

### 8.1.2 Example - airplane - with control input

This example is similar to the previous example, but now we have a sensor connected to the pilot's controls, so we have additional information about the airplane acceleration based on the pilot's commands.

The state vector $\hat{\boldsymbol{x}}_{\boldsymbol{n}}$ that describes the estimated airplane position and velocity in a cartesian coordinate system $(x, y, z)$ is:

$$\hat{\boldsymbol{x}}_n = \begin{bmatrix} \hat{x}_n \\ \hat{y}_n \\ \hat{z}_n \\ \hat{\dot{x}}_n \\ \hat{\dot{y}}_n \\ \hat{\dot{z}}_n \end{bmatrix} \tag{8.7}$$

The control vector $\boldsymbol{u_n}$ that describes the measured airplane acceleration in a cartesian coordinate system $(x, y, z)$ is:

$$\boldsymbol{u}_n = \begin{bmatrix} \ddot{x}_n \\ \ddot{y}_n \\ \ddot{z}_n \end{bmatrix} \tag{8.8}$$

The state transition matrix $\boldsymbol{F}$ is:

$$\boldsymbol{F} = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{8.9}$$

The control matrix $\boldsymbol{G}$ is:

$$\boldsymbol{G} = \begin{bmatrix} 0.5\Delta t^2 & 0 & 0 \\ 0 & 0.5\Delta t^2 & 0 \\ 0 & 0 & 0.5\Delta t^2 \\ \Delta t & 0 & 0 \\ 0 & \Delta t & 0 \\ 0 & 0 & \Delta t \end{bmatrix} \tag{8.10}$$

The state extrapolation equation is:

$$\hat{\boldsymbol{x}}_{n+1,n} = \boldsymbol{F}\hat{\boldsymbol{x}}_{n,n} + \boldsymbol{G}\boldsymbol{u}_{n,n} \tag{8.11}$$

$$\begin{bmatrix} \hat{x}_{n+1,n} \\ \hat{y}_{n+1,n} \\ \hat{z}_{n+1,n} \\ \hat{\dot{x}}_{n+1,n} \\ \hat{\dot{y}}_{n+1,n} \\ \hat{\dot{z}}_{n+1,n} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{x}_{n,n} \\ \hat{y}_{n,n} \\ \hat{z}_{n,n} \\ \hat{\dot{x}}_{n,n} \\ \hat{\dot{y}}_{n,n} \\ \hat{\dot{z}}_{n,n} \end{bmatrix} + \begin{bmatrix} 0.5\Delta t^2 & 0 & 0 \\ 0 & 0.5\Delta t^2 & 0 \\ 0 & 0 & 0.5\Delta t^2 \\ \Delta t & 0 & 0 \\ 0 & \Delta t & 0 \\ 0 & 0 & \Delta t \end{bmatrix} \begin{bmatrix} \ddot{x}_n \\ \ddot{y}_n \\ \ddot{z}_n \end{bmatrix} \tag{8.12}$$

### 8.1.3 Example – falling object

Consider a free-falling object. The state vector includes the altitude $h$ and the object's velocity $\dot{h}$:

$$\hat{\boldsymbol{x}}_n = \begin{bmatrix} \hat{h}_n \\ \hat{\dot{h}}_n \end{bmatrix} \tag{8.13}$$

The state transition matrix $\boldsymbol{F}$ is:

$$\boldsymbol{F} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \tag{8.14}$$

The control matrix $\boldsymbol{G}$ is:

$$\boldsymbol{G} = \begin{bmatrix} 0.5\Delta t^2 \\ \Delta t \end{bmatrix} \tag{8.15}$$

The input variable $\boldsymbol{u}_n$ is:

$$\boldsymbol{u}_n = \begin{bmatrix} g \end{bmatrix} \tag{8.16}$$

Where $g$ is the gravitational acceleration.

We don't have a sensor that measures acceleration, but we know that for a falling object, acceleration equals $g$.

The state extrapolation equation looks as follows:

$$\begin{bmatrix} \hat{h}_{n+1,n} \\ \hat{\dot{h}}_{n+1,n} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{h}_{n,n} \\ \hat{\dot{h}}_{n,n} \end{bmatrix} + \begin{bmatrix} 0.5\Delta t^2 \\ \Delta t \end{bmatrix} \begin{bmatrix} g \end{bmatrix} \tag{8.17}$$

The matrix multiplication results in the following:

$$\begin{cases} \hat{h}_{n+1,n} = \hat{h}_{n,n} + \hat{\dot{h}}_{n,n}\Delta t + 0.5\Delta t^2 g \\ \hat{\dot{h}}_{n+1,n} = \hat{\dot{h}}_{n,n} + \Delta t g \end{cases} \tag{8.18}$$

Well, that was easy to describe a dynamic model for an airplane or falling Object. I suppose that you are familiar with Newton's motion equations from high school. Appendix C ("Modeling linear dynamic systems") generalizes dynamic model derivation for any linear dynamic system.

### 8.1.4 State extrapolation equation dimensions

The following table specifies the matrix dimensions of the state extrapolation equation variables:

| Variable | Description | Dimension |
|---|---|---|
| $\boldsymbol{x}$ | state vector | $n_x \times 1$ |
| $\boldsymbol{F}$ | state transition matrix | $n_x \times n_x$ |
| $\boldsymbol{u}$ | input variable | $n_u \times 1$ |
| $\boldsymbol{G}$ | control matrix | $n_x \times n_u$ |
| $\boldsymbol{w}$ | process noise vector | $n_x \times 1$ |

**Table 8.1:** *Matrix dimensions of the state extrapolation equation.*

### 8.1.5 Linear time-invariant systems

The Linear Kalman Filter assumes the **Linear Time-Invariant** (LTI) system model.

So, what is "linear," and what is "time-invariant"?

**Linear systems** are described by systems of equations in which the variables are never multiplied with each other but only with constants and then summed up. Linear systems are used to describe both static and dynamic relationships between

variables.

A linear system is a system whose output function $y(t)$ satisfies the following equation:

$$y(t) = \mathcal{F}\left(a \times g\left(t\right) + b \times h\left(t\right)\right) = a \times \mathcal{F}\left(g\left(t\right)\right) + b \times \mathcal{F}\left(h\left(t\right)\right) \tag{8.19}$$

Where:
$a$ and $b$ are constant real numbers
$g$ and $h$ are any arbitrary functions of an independent variable $t$

A linear system follows two basic rules:

1. You can "factor out" constant multiplicative scale factors (the $a$ and $b$ above).
2. The system's response to a sum of inputs is the sum of the responses to each input separately.

A **time-invariant system** has a **system function** that is not a direct function of time.

Let's take an amplifier with gain $G = 10$ as an example.

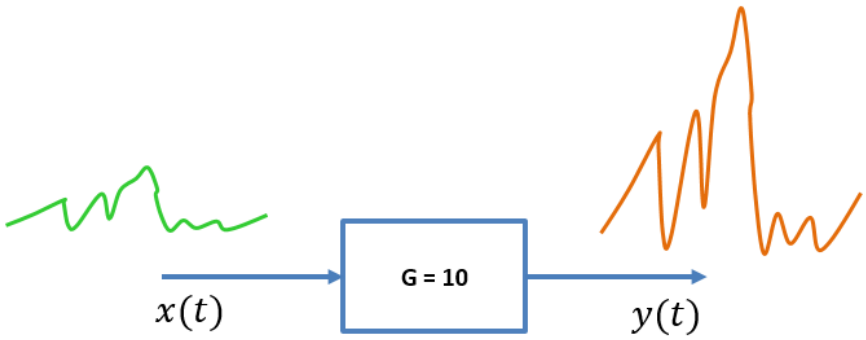


**Figure 8.4:** *Amplifier.*

This system is time-invariant. Although the system's output changes with time, the system function is not time-dependent.

A time-invariant system is one where a time delay (or shift) in the input sequence causes an equivalent time delay in the system's output sequence.

## 8.2 Covariance Extrapolation Equation

I assume the reader is already familiar with the concept of covariance extrapolation (prediction). We've already met the **Covariance Extrapolation Equation** (or **Predictor Covariance Equation**) in Part I. In this section, we derive the Kalman Filter Covariance Extrapolation Equation in matrix notation.

The general form of the Covariance Extrapolation Equation is given by:

**Covariance Extrapolation Equation**

$$\boldsymbol{P}_{n+1,n} = \boldsymbol{F}\boldsymbol{P}_{n,n}\boldsymbol{F}^T + \boldsymbol{Q} \tag{8.20}$$

Where:

- $\boldsymbol{P}_{n,n}$ is the squared uncertainty of an estimate (covariance matrix) of the current state
- $\boldsymbol{P}_{n+1,n}$ is the squared uncertainty of a prediction (covariance matrix) for the next state
- $\boldsymbol{F}$ is the state transition matrix that we derived in Appendix C ("Modeling linear dynamic systems")
- $\boldsymbol{Q}$ is the process noise matrix

### 8.2.1 The estimate covariance without process noise

Let's assume that the process noise is equal to zero ($Q = 0$), then:

$$\boldsymbol{P}_{n+1,n} = \boldsymbol{F}\boldsymbol{P}_{n,n}\boldsymbol{F}^T \tag{8.21}$$

The derivation is relatively straightforward. I've shown in chapter 7 - "Essential background II," that:

$$COV(\boldsymbol{x}) = E\left((\boldsymbol{x} - \boldsymbol{\mu}_{\boldsymbol{x}})(\boldsymbol{x} - \boldsymbol{\mu}_{\boldsymbol{x}})^T\right) \tag{8.22}$$

Where vector $x$ is a system state vector.

Therefore:

$$\boldsymbol{P}_{n,n} = E\left((\hat{\boldsymbol{x}}_{\boldsymbol{n},\boldsymbol{n}} - \boldsymbol{\mu}_{\boldsymbol{x}_{\boldsymbol{n},\boldsymbol{n}}})(\hat{\boldsymbol{x}}_{\boldsymbol{n},\boldsymbol{n}} - \boldsymbol{\mu}_{\boldsymbol{x}_{\boldsymbol{n},\boldsymbol{n}}})^T\right) \tag{8.23}$$

Therefore:

$$\boldsymbol{P}_{n+1,n} = E\left(\left(\hat{\boldsymbol{x}}_{n+1,n} - \boldsymbol{\mu}_{x_{n+1,n}}\right)\left(\hat{\boldsymbol{x}}_{n+1,n} - \boldsymbol{\mu}_{x_{n+1,n}}\right)^T\right) \tag{8.24}$$

According to the state extrapolation equation (Equation 8.1):

$$\hat{\boldsymbol{x}}_{n+1,n} = \boldsymbol{F}\hat{\boldsymbol{x}}_{n,n} + \boldsymbol{G}\hat{\boldsymbol{u}}_{n,n} \tag{8.25}$$

$$\begin{aligned}\boldsymbol{P}_{n+1,n} = E\Big(&\left(\boldsymbol{F}\hat{\boldsymbol{x}}_{n,n} + \boldsymbol{G}\hat{\boldsymbol{u}}_{n,n} - \boldsymbol{F}\boldsymbol{\mu}_{\boldsymbol{x_{n,n}}} - \boldsymbol{G}\hat{\boldsymbol{u}}_{\boldsymbol{n,n}}\right)\\ &\times\left(\boldsymbol{F}\hat{\boldsymbol{x}}_{n,n} + \boldsymbol{G}\hat{\boldsymbol{u}}_{n,n} - \boldsymbol{F}\boldsymbol{\mu}_{\boldsymbol{x_{n,n}}} - \boldsymbol{G}\hat{\boldsymbol{u}}_{\boldsymbol{n,n}}\right)^T\Big)\end{aligned} \tag{8.26}$$

$$\boldsymbol{P}_{n+1,n} = E\left(\boldsymbol{F}\left(\hat{\boldsymbol{x}}_{n,n} - \boldsymbol{\mu}_{x_{n,n}}\right)\left(\boldsymbol{F}\left(\hat{\boldsymbol{x}}_{n,n} - \boldsymbol{\mu}_{x_{n,n}}\right)\right)^T\right) \tag{8.27}$$

Apply the matrix transpose property: $(\boldsymbol{AB})^T = \boldsymbol{B}^T\boldsymbol{A}^T$

$$\boldsymbol{P}_{n+1,n} = E\left(\boldsymbol{F}\left(\hat{\boldsymbol{x}}_{n,n} - \boldsymbol{\mu}_{x_{n,n}}\right)\left(\hat{\boldsymbol{x}}_{n,n} - \boldsymbol{\mu}_{x_{n,n}}\right)^T\boldsymbol{F}^T\right) \tag{8.28}$$

$$\boldsymbol{P}_{n+1,n} = \boldsymbol{F}E\left(\left(\hat{\boldsymbol{x}}_{\boldsymbol{n,n}} - \boldsymbol{\mu}_{\boldsymbol{x_{n,n}}}\right)\left(\hat{\boldsymbol{x}}_{\boldsymbol{n,n}} - \boldsymbol{\mu}_{\boldsymbol{x_{n,n}}}\right)^T\right)\boldsymbol{F}^T \tag{8.29}$$

$$\boldsymbol{P}_{n+1,n} = \boldsymbol{F}\boldsymbol{P}_{n,n}\boldsymbol{F}^T \tag{8.30}$$

### 8.2.2 Constructing the process noise matrix $Q$

As you already know, the system dynamics is described by:

$$\hat{\boldsymbol{x}}_{n+1,n} = \boldsymbol{F}\hat{\boldsymbol{x}}_{n,n} + \boldsymbol{G}\hat{\boldsymbol{u}}_{n,n} + \boldsymbol{w}_n \tag{8.31}$$

Where $\boldsymbol{w}_n$ is the process noise at the time step $n$.

We've discussed the process noise and its influence on the Kalman Filter performance in the "One-dimensional Kalman Filter" section. In the one-dimensional Kalman Filter, the process noise variance is denoted by $q$.

In the multidimensional case, the process noise is a covariance matrix denoted by $\boldsymbol{Q}$.

We've seen that the process noise variance has a critical influence on the Kalman Filter performance. Too small $q$ causes a lag error (see section 5.3 - Example 7). If the $q$ value is too high, the Kalman Filter follows the measurements (see section 5.4 - Example 8) and produces noisy estimations.

The process noise can be independent between different state variables. In this case, the process noise covariance matrix $\boldsymbol{Q}$ is a diagonal matrix:

$$\boldsymbol{Q} = \begin{bmatrix} q_{11} & 0 & \cdots & 0 \\ 0 & q_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & q_{kk} \end{bmatrix} \tag{8.32}$$

The process noise can also be dependent. For example, the constant velocity model assumes zero acceleration ($a = 0$). However, a random variance in acceleration $\sigma_a^2$ causes a variance in velocity and position. In this case, the process noise is correlated with the state variables.

There are two models for the environmental process noise.

- Discrete noise model
- Continuous noise model

#### 8.2.2.1 Discrete noise model

The discrete noise model assumes that the noise is different at each period but is constant between periods.

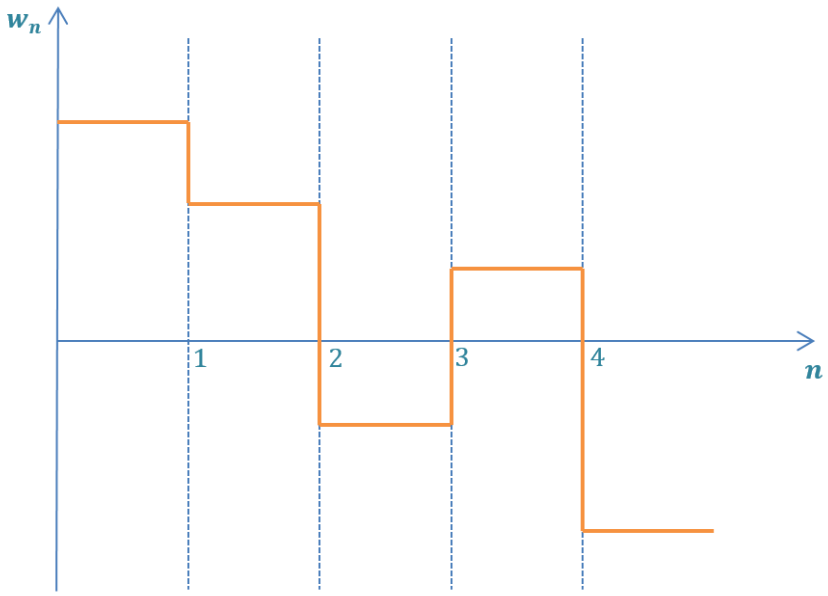


**Figure 8.5:** *Discrete Noise.*

For the constant velocity model, the process noise covariance matrix looks like the following:

$$\boldsymbol{Q} = \begin{bmatrix} V(x) & COV(x,v) \\ COV(v,x) & V(v) \end{bmatrix} \tag{8.33}$$

We express the position and velocity variance and covariance in terms of the random acceleration variance of the model: $\sigma_a^2$.

We derived the matrix elements using the expectation arithmetic rules in chapter 7 - "Essential background II."

$$\begin{aligned} V(v) = \sigma_v^2 = E\left(v^2\right) - \mu_v^2 = E\left((a\Delta t)^2\right) - (\mu_a \Delta t)^2 \\ = \Delta t^2 \left(E\left(a^2\right) - \mu_a^2\right) = \Delta t^2 \sigma_a^2 \end{aligned} \tag{8.34}$$

$$\begin{aligned} V(x) = \sigma_x^2 = E\left(x^2\right) - \mu_x^2 = E\left(\left(\frac{1}{2}\, a\Delta t^2\right)^2\right) - \left(\frac{1}{2}\, \mu_a \Delta t^2\right)^2 \\ = \frac{\Delta t^4}{4}\left(E\left(a^2\right) - \mu_a^2\right) = \frac{\Delta t^4}{4}\, \sigma_a^2 \end{aligned} \tag{8.35}$$

$$\begin{aligned} COV(x,v) = COV(v,x) = E\left(xv\right) - \mu_x \mu_v \\ = E\left(\frac{1}{2}\, a\Delta t^2 a\Delta t\right) - \left(\frac{1}{2}\, \mu_a \Delta t^2 \mu_a \Delta t\right) = \frac{\Delta t^3}{2}\left(E\left(a^2\right) - \mu_a^2\right) = \frac{\Delta t^3}{2}\, \sigma_a^2 \end{aligned} \tag{8.36}$$

Now we can substitute the results into $\boldsymbol{Q}$ matrix:

$$\boldsymbol{Q} = \sigma_a^2 \begin{bmatrix} \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} \\ \frac{\Delta t^3}{2} & \Delta t^2 \end{bmatrix} \tag{8.37}$$

There are two methods for a faster construction of the $\boldsymbol{Q}$ matrix.

### Projection using the state transition matrix

If the dynamic model doesn't include a control input, we can project the random variance in acceleration $\sigma_a^2$ on our dynamic model using the state transition matrix.

Let us define a matrix $\boldsymbol{Q}_a$:

$$\boldsymbol{Q}_a = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \sigma_a^2 \tag{8.38}$$

The process noise matrix is:

$$\boldsymbol{Q} = \boldsymbol{F}\boldsymbol{Q}_a\boldsymbol{F}^T \tag{8.39}$$

For the motion model, the $\boldsymbol{F}$ matrix is given by:

$$\boldsymbol{F} = \begin{bmatrix} 1 & \Delta t & \frac{\Delta t^2}{2} \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix} \tag{8.40}$$

$$\boldsymbol{Q} = \boldsymbol{F}\boldsymbol{Q}_a\boldsymbol{F}^T \tag{8.41}$$

$$\boldsymbol{Q} = \begin{bmatrix} 1 & \Delta t & \frac{\Delta t^2}{2} \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ \Delta t & 1 & 0 \\ \frac{\Delta t^2}{2} & \Delta t & 1 \end{bmatrix} \sigma_a^2 \tag{8.42}$$

$$\boldsymbol{Q} = \begin{bmatrix} 0 & 0 & \frac{\Delta t^2}{2} \\ 0 & 0 & \Delta t \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ \Delta t & 1 & 0 \\ \frac{\Delta t^2}{2} & \Delta t & 1 \end{bmatrix} \sigma_a^2 \tag{8.43}$$

$$\boldsymbol{Q} = \begin{bmatrix} \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} & \frac{\Delta t^2}{2} \\ \frac{\Delta t^3}{2} & \Delta t^2 & \Delta t \\ \frac{\Delta t^2}{2} & \Delta t & 1 \end{bmatrix} \sigma_a^2 \tag{8.44}$$

### Projection using the control matrix

If the dynamic model includes a control input, we can compute the $\boldsymbol{Q}$ matrix even faster. We can project the random variance in acceleration $\sigma_a^2$ on our dynamic model using the control matrix.

$$\boldsymbol{Q} = \boldsymbol{G}\sigma_a^2\boldsymbol{G}^{\boldsymbol{T}} \tag{8.45}$$

Where $\boldsymbol{G}$ is the control matrix (or input transition matrix).

For the motion model, the $\boldsymbol{G}$ matrix is given by:

$$\boldsymbol{G} = \begin{bmatrix} \frac{\Delta t^2}{2} \\ \Delta t \end{bmatrix} \tag{8.46}$$

$$\boldsymbol{Q} = \boldsymbol{G}\sigma_a^2\boldsymbol{G}^{\boldsymbol{T}} = \sigma_a^2\boldsymbol{G}\boldsymbol{G}^{\boldsymbol{T}} = \sigma_a^2 \begin{bmatrix} \frac{\Delta t^2}{2} \\ \Delta t \end{bmatrix} \begin{bmatrix} \frac{\Delta t^2}{2} & \Delta t \end{bmatrix} = \sigma_a^2 \begin{bmatrix} \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} \\ \frac{\Delta t^3}{2} & \Delta t^2 \end{bmatrix} \tag{8.47}$$

You can use the above methods to construct the discrete $\boldsymbol{Q}$ matrix.

#### 8.2.2.2 Continuous noise model

The continuous model assumes that the noise changes continuously over time.

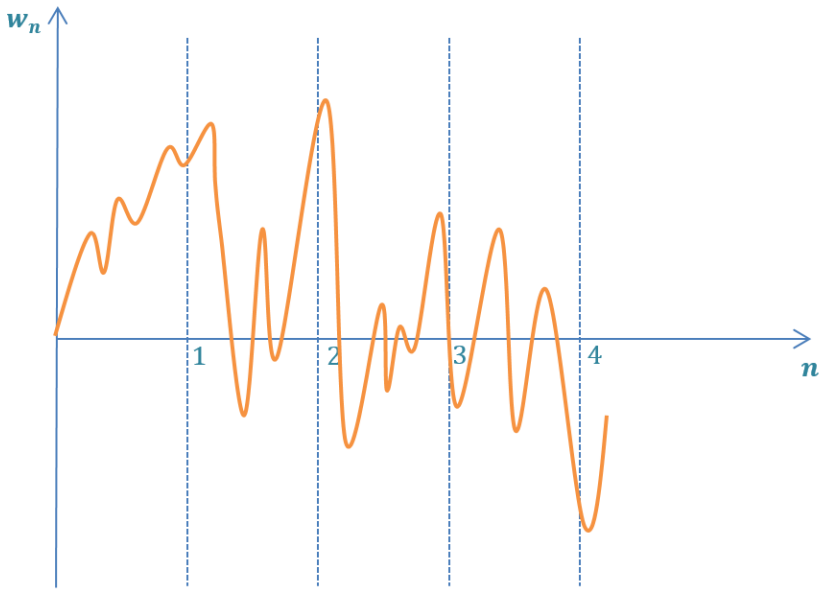


**Figure 8.6:** *Continuous Noise.*

To derive the process noise covariance matrix for the continuous model $\boldsymbol{Q}_c$, we need to integrate the discrete process noise covariance matrix $\boldsymbol{Q}$ over time.

$$\boldsymbol{Q}_c = \int_0^{\Delta t} \boldsymbol{Q} dt = \int_0^{\Delta t} \sigma_a^2 \begin{bmatrix} \frac{t^4}{4} & \frac{t^3}{2} \\ \frac{t^3}{2} & t^2 \end{bmatrix} dt = \sigma_a^2 \begin{bmatrix} \frac{\Delta t^5}{20} & \frac{\Delta t^4}{8} \\ \frac{\Delta t^4}{8} & \frac{\Delta t^3}{3} \end{bmatrix} \tag{8.48}$$

#### 8.2.2.3 Which model to choose?

Before answering this question, you need to select the right value for the process noise variance. You can calculate it using the stochastic statistics formulas or choose a reasonable value based on your engineering practice (which is preferable).

In the radar world, the $\sigma_a^2$ depends on the target characteristics and model completeness. For maneuvering targets, like airplanes, the $\sigma_a^2$ should be relatively high. For non-maneuvering targets, like rockets, you can use smaller $\sigma_a^2$. The model completeness is also a factor in selecting the process noise variance. If your model includes environmental influences like air drag, then the degree of the process noise randomness is smaller and vice versa.

Once you've selected a reasonable process noise variance value, you should choose the noise model. Should it be discrete or continuous?

There is no clear answer to this question. I recommend trying both models and checking which one performs better with your Kalman Filter. When $\Delta t$ is very small, you can use the discrete noise model. When $\Delta t$ is high, it is better to use the continuous noise model.

## 8.3 Measurement equation

Until now, we've dealt with the future. We've derived two Kalman Filter prediction equations:

- State Extrapolation Equation
- Covariance Extrapolation Equation

From now on, we are going to deal with the present. Let's start with the Measurement Equation.

In the "One-dimensional Kalman Filter" section, we denoted the measurement by $z_n$.

The measurement value represents a true system state in addition to the random measurement noise $v_n$, caused by the measurement device.

The measurement noise variance $r_n$ can be constant for each measurement - for example, scales with a precision of 0.5kg (standard deviation). On the other hand, the measurement noise variance $r_n$ can be different for each measurement - for example, a thermometer with a precision of 0.5% (standard deviation). In the latter case, the noise variance depends on the measured temperature.

The generalized measurement equation in matrix form is given by:

**Measurement Equation**

$$\boldsymbol{z}_n = \boldsymbol{H}\boldsymbol{x}_n + \boldsymbol{v}_n \tag{8.49}$$

Where:

- $\boldsymbol{z}_n$ is a measurement vector
- $\boldsymbol{x}_n$ is a true system state (hidden state)
- $\boldsymbol{v}_n$ is a random noise vector
- $\boldsymbol{H}$ is an **observation matrix**

### 8.3.1 The observation matrix

In many cases, the measured value is not the desired system state. For example, a digital electric thermometer measures an electric current, while the system state is the temperature. There is a need for a transformation of the system state (input) to the measurement (output).

The purpose of the observation matrix $\boldsymbol{H}$ is to convert the system state into outputs using linear transformations. The following chapters include examples of observation matrix usage.

#### 8.3.1.1 Scaling

A range meter sends a signal toward a destination and receives a reflected echo. The measurement is the time delay between the transmission and reception of the signal. The system state is the range.

In this case, we need to perform a scaling:

$$\boldsymbol{z}_n = \begin{bmatrix} \frac{2}{c} \end{bmatrix} \boldsymbol{x}_n + \boldsymbol{v}_n \tag{8.50}$$

$$\boldsymbol{H} = \begin{bmatrix} \frac{2}{c} \end{bmatrix} \tag{8.51}$$

Where:

- $c$ is the speed of light
- $\boldsymbol{x}_n$ is the range
- $\boldsymbol{z}_n$ is the measured time delay

#### 8.3.1.2 State selection

Sometimes certain states are measured while others are not. For example, the first, third, and fifth states of a five-dimensional state vector are measurable, while the second and fourth states are not measurable:

$$\boldsymbol{z}_n = \boldsymbol{H}\boldsymbol{x}_n + \boldsymbol{v}_n = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} + \boldsymbol{v}_n = \begin{bmatrix} x_1 \\ x_3 \\ x_5 \end{bmatrix} + \boldsymbol{v}_n \tag{8.52}$$

#### 8.3.1.3 Combination of states

Sometimes some combination of states can be measured instead of each separate state. For example, maybe the lengths of the sides of a triangle are the states, and only the total perimeter can be measured:

$$\boldsymbol{z}_n = \boldsymbol{H}\boldsymbol{x}_n + \boldsymbol{v}_n = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \boldsymbol{v}_n = (x_1 + x_2 + x_3) + \boldsymbol{v}_n \tag{8.53}$$

### 8.3.2 Measurement equation dimensions

The following table specifies the matrix dimensions of the measurement equation variables:

| Variable | Description | Dimension |
|---|---|---|
| $\boldsymbol{x}$ | state vector | $n_x \times 1$ |
| $\boldsymbol{z}$ | measurements vector | $n_z \times 1$ |
| $\boldsymbol{H}$ | observation matrix | $n_z \times n_x$ |
| $\boldsymbol{v}$ | measurement noise vector | $n_z \times 1$ |

**Table 8.2:** *Matrix dimensions of the measurement equation variables.*

## 8.4 Interim Summary

It is a good place to stop for a short summary. Before going further, I would like to summarize what we have learned so far.

As you remember from Part I (if you don't remember, please review it again), the Kalman Filter computations are based on five equations.

Two prediction equations:

- State Extrapolation Equation - predicting or estimating the future state based on the known present estimation.
- Covariance Extrapolation Equation - the measure of uncertainty in our prediction.

Two update equations:

- State Update Equation - estimating the current state based on the known past estimation and present measurement.
- Covariance Update Equation - the measure of uncertainty in our estimation.

Kalman Gain Equation - required for computation of the update equations. The Kalman Gain is a "weighting" parameter for the measurement and the past estimations. It defines the weight of the past estimation and the weight of the measurement in estimating the current state.

So far, we have learned the two prediction equations in matrix notation and several auxiliary equations required for computing the main equations.

## 8.4.1 Prediction equations

### 8.4.1.1 State Extrapolation Equation

The general form of the state extrapolation equation in a matrix notation is:

**State Extrapolation Equation**

$$\hat{\boldsymbol{x}}_{n+1,n} = \boldsymbol{F}\hat{\boldsymbol{x}}_{n,n} + \boldsymbol{G}\boldsymbol{u}_n + \boldsymbol{w}_n \tag{8.54}$$

Where:

| | |
|---|---|
| $\hat{\boldsymbol{x}}_{n+1,n}$ | is a predicted system state vector at time step $n+1$ |
| $\hat{\boldsymbol{x}}_{n,n}$ | is an estimated system state vector at time step $n$ |
| $\boldsymbol{u}_n$ | is a control variable or input variable - a measurable (deterministic) input to the system |
| $\boldsymbol{w}_n$ | is a process noise or disturbance - an unmeasurable input that affects the state |
| $\boldsymbol{F}$ | is a state transition matrix |
| $\boldsymbol{G}$ | is a control matrix or input transition matrix (mapping control to state variables) |

### 8.4.1.2 Covariance Extrapolation Equation

The general form of the Covariance Extrapolation Equation is given by:

**Covariance Extrapolation Equation**

$$\boldsymbol{P}_{n+1,n} = \boldsymbol{F}\boldsymbol{P}_{n,n}\boldsymbol{F}^T + \boldsymbol{Q} \tag{8.55}$$

Where:

| | |
|---|---|
| $\boldsymbol{P}_{n,n}$ | the covariance matrix of the current state estimation |
| $\boldsymbol{P}_{n+1,n}$ | is the covariance matrix of the next state estimation (prediction) |
| $\boldsymbol{F}$ | is the state transition matrix that we derived in Appendix C ("Modeling linear dynamic systems") |
| $\boldsymbol{Q}$ | is the process noise matrix |

## 8.4.2 Auxiliary equations

### 8.4.2.1 Measurement Equation

The generalized measurement equation in matrix form is given by:

**Measurement Equation**

$$\boldsymbol{z}_n = \boldsymbol{H}\boldsymbol{x}_n + \boldsymbol{v}_n \tag{8.56}$$

Where:

- $\boldsymbol{z}_n$ is a measurement vector
- $\boldsymbol{x}_n$ is a true system state (hidden state)
- $\boldsymbol{v}_n$ is a random noise vector
- $\boldsymbol{H}$ is an **observation matrix**

### 8.4.2.2 Covariance Equations

The terms $\boldsymbol{w}$ and $\boldsymbol{v}$, which correspond to the process and measurement noise, do not typically appear directly in the calculations since they are unknown.

Instead, these terms are used to model the uncertainty (or noise) in the equations themselves.

All covariance equations are covariance matrices in the form of:

$$E\left(\boldsymbol{e}\boldsymbol{e}^T\right) \tag{8.57}$$

i.e., an expectation of a squared error. See chapter 7 ("Essential background II") for more details.

#### Measurement uncertainty

The measurement covariance is given by:

### Measurement Uncertainty

$$\boldsymbol{R}_n = E\left(\boldsymbol{v}_n \boldsymbol{v}_n^T\right) \tag{8.58}$$

Where:

$\boldsymbol{R}_n$ is the covariance matrix of the measurement
$\boldsymbol{v}_n$ is the measurement error

### Process noise uncertainty

The process noise covariance is given by:

### Process Noise Uncertainty

$$\boldsymbol{Q_n} = E\left(\boldsymbol{w}_n \boldsymbol{w}_n^T\right) \tag{8.59}$$

Where:

$\boldsymbol{Q}_n$ is the covariance matrix of the process noise
$\boldsymbol{w}_n$ is the process noise

### Estimation uncertainty

The estimation covariance is given by:

### Estimation Uncertainty

$$\boldsymbol{P}_{n,n} = E\left(\boldsymbol{e}_n \boldsymbol{e}_n^T\right) = E\left(\left(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n}\right)\left(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n}\right)^T\right) \tag{8.60}$$

Where:

$\boldsymbol{P}_{n,n}$ is the covariance matrix of the estimation error
$\boldsymbol{e}_n$ is the estimation error
$\boldsymbol{x}_n$ is the true system state (hidden state)
$\hat{\boldsymbol{x}}_{n,n}$ is the estimated system state vector at time step $n$

## 8.5 State Update Equation

This section is the shortest section of this book. I've provided an extensive description of the State Update Equation in Part I.

The State Update Equation in the matrix form is given by:

**Estimation Uncertainty**

$$\hat{\boldsymbol{x}}_{n,n} = \hat{\boldsymbol{x}}_{n,n-1} + \boldsymbol{K}_n(\boldsymbol{z}_n - \boldsymbol{H}\hat{\boldsymbol{x}}_{n,n-1}) \tag{8.61}$$

Where:

| | |
|---|---|
| $\hat{\boldsymbol{x}}_{n,n}$ | is an estimated system state vector at time step $n$ |
| $\hat{\boldsymbol{x}}_{n,n-1}$ | is a predicted system state vector at time step $n-1$ |
| $\boldsymbol{K}_n$ | is a Kalman Gain |
| $\boldsymbol{z}_n$ | is a measurement |
| $\boldsymbol{H}$ | is an observation matrix |

You should be familiar with all components of the State Update Equation except the Kalman Gain in a matrix notation. We derive the Kalman Gain in section 8.7.

You should pay attention to the dimensions. If, for instance, the state vector has 5 dimensions, while only 3 dimensions are measurable (the first, third, and fifth states):

$$\boldsymbol{x}_n = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} \qquad \boldsymbol{z}_n = \begin{bmatrix} z_1 \\ z_3 \\ z_5 \end{bmatrix} \tag{8.62}$$

The observation matrix would be a $3 \times 5$ matrix:

$$\boldsymbol{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{8.63}$$

The **innovation** $(\boldsymbol{z}_n - \boldsymbol{H}\hat{\boldsymbol{x}}_{n,n-1})$ yields:

$$(\boldsymbol{z}_n - \boldsymbol{H}\hat{\boldsymbol{x}}_{n,n-1}) = \begin{bmatrix} z_1 \\ z_3 \\ z_5 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \\ \hat{x}_3 \\ \hat{x}_4 \\ \hat{x}_5 \end{bmatrix} = \begin{bmatrix} (z_1 - \hat{x}_1) \\ (z_3 - \hat{x}_3) \\ (z_5 - \hat{x}_5) \end{bmatrix} \tag{8.64}$$

The Kalman Gain dimensions shall be $5 \times 3$.

### 8.5.1 State Update Equation dimensions

The following table specifies the matrix dimensions of the State Update Equation variables:

| Variable | Description | Dimension |
|---|---|---|
| $\boldsymbol{x}$ | state vector | $n_x \times 1$ |
| $\boldsymbol{z}$ | measurements vector | $n_z \times 1$ |
| $\boldsymbol{H}$ | observation matrix | $n_z \times n_x$ |
| $\boldsymbol{K}$ | Kalman gain | $n_x \times n_z$ |

**Table 8.3:** *Matrix dimensions of the state update equation variables.*

## 8.6 Covariance Update Equation

The Covariance Update Equation is given by:

### Covariance Update Equation

$$\boldsymbol{P}_{n,n} = (\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})\,\boldsymbol{P}_{n,n-1}\,(\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})^T + \boldsymbol{K}_n\boldsymbol{R}_n\boldsymbol{K}_n^T \tag{8.65}$$

Where:

| | |
|---|---|
| $\boldsymbol{P}_{n,n}$ | is the covariance matrix of the current state estimation |
| $\boldsymbol{P}_{n,n-1}$ | is the prior estimate covariance matrix of the current state (predicted at the previous state) |
| $\boldsymbol{K}_n$ | is a Kalman Gain |
| $\boldsymbol{H}$ | is the observation matrix |
| $\boldsymbol{R}_n$ | is the measurement noise covariance matrix |
| $\boldsymbol{I}$ | is an Identity Matrix (the $n \times n$ square matrix with ones on the main diagonal and zeros elsewhere) |

### 8.6.1 Covariance Update Equation Derivation

This section includes the Covariance Update Equation derivation. Some of you may find it too detailed, but on the other hand, it will help others to understand better.

You can jump to the next section if you don't care about the derivation.

For the derivation, I use the following four equations:

| | Equation | Notes |
|---|---|---|
| **1** | $\hat{\boldsymbol{x}}_{n,n} = \hat{\boldsymbol{x}}_{n,n-1} + \boldsymbol{K}_n(\boldsymbol{z}_n - \boldsymbol{H}\hat{\boldsymbol{x}}_{n,n-1})$ | State Update Equation |
| **2** | $\boldsymbol{z}_n = \boldsymbol{H}\boldsymbol{x}_n + \boldsymbol{v}_n$ | Measurement Equation |
| **3** | $\boldsymbol{P}_{n,n} = E\left(\boldsymbol{e}_n\boldsymbol{e}_n^T\right)$ $= E\left((\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n})\,(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n})^T\right)$ | Estimate Covariance |
| **4** | $\boldsymbol{R}_n = E\left(\boldsymbol{v}_n\boldsymbol{v}_n^T\right)$ | Measurement Covariance |

**Table 8.4:** *Equations for the Covariance Update Equation derivation.*

We derive the Current Estimate Covariance ($\boldsymbol{P}_{n,n}$) as a function of the Kalman Gain $\boldsymbol{K}_n$.

Table 8.5: Covariance Update Equation derivation.

| **Equation** | **Notes** |
|---|---|
| $\hat{\boldsymbol{x}}_{n,n} = \hat{\boldsymbol{x}}_{n,n-1} + \boldsymbol{K}_n(\boldsymbol{z}_n - \boldsymbol{H}\hat{\boldsymbol{x}}_{n,n-1})$ | State Update Equation |
| $\hat{\boldsymbol{x}}_{n,n} = \hat{\boldsymbol{x}}_{n,n-1} + \boldsymbol{K}_n(\boldsymbol{H}\boldsymbol{x}_n + \boldsymbol{v}_n - \boldsymbol{H}\hat{\boldsymbol{x}}_{n,n-1})$ | Plug the Measurement Equation into the State Update Equation |
| $\boldsymbol{e}_n = \boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n}$ | Estimate error |
| $\boldsymbol{e}_n = \boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n-1} - \boldsymbol{K}_n(\boldsymbol{H}\boldsymbol{x}_n + \boldsymbol{v}_n - \boldsymbol{H}\hat{\boldsymbol{x}}_{n,n-1})$ | Plug $\hat{\boldsymbol{x}}_{n,n}$ |
| $= \boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n-1} - \boldsymbol{K}_n\boldsymbol{H}\boldsymbol{x}_n - \boldsymbol{K}_n\boldsymbol{v}_n + \boldsymbol{K}_n\boldsymbol{H}\hat{\boldsymbol{x}}_{n,n-1}$ | Expand |
| $= \boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n-1} - \boldsymbol{K}_n\boldsymbol{H}(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n-1}) - \boldsymbol{K}_n\boldsymbol{v}_n$ | Localize $(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n-1})$ |
| $= (\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n-1}) - \boldsymbol{K}_n\boldsymbol{v}_n$ | |
| $\boldsymbol{P}_{n,n} = E(\boldsymbol{e}_n\boldsymbol{e}_n^T) = E\left((\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n})\,(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n})^T\right)$ | Estimate Covariance |
| $\boldsymbol{P}_{n,n} = E\Big(\big((\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n-1}) - \boldsymbol{K}_n\boldsymbol{v}_n\big)$ $\times\big((\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n-1}) - \boldsymbol{K}_n\boldsymbol{v}_n\big)^T\Big)$ | Plug $\boldsymbol{e}_n$ |
| $\boldsymbol{P}_{n,n} = E\Bigg(\Big((\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n-1}) - \boldsymbol{K}_n\boldsymbol{v}_n\Big)$ $\times\Big(\big((\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n-1})\big)^T - (\boldsymbol{K}_n\boldsymbol{v}_n)^T\Big)\Bigg)$ | Expand |

Continued on next page

Table 8.5: Covariance Update Equation derivation. (Continued)

| | |
|---|---|
| $\boldsymbol{P}_{n,n} = E\Big(\big((\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n-1}) - \boldsymbol{K}_n\boldsymbol{v}_n\big)$ $\times\big((\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n-1})^T(\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})^T - (\boldsymbol{K}_n\boldsymbol{v}_n)^T\big)\Big)$ | Apply the matrix transpose property: $(\boldsymbol{AB})^T = \boldsymbol{B}^T\boldsymbol{A}^T$ |
| $\boldsymbol{P}_{n,n} = E\Big((\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n-1})$ $\times(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n-1})^T(\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})^T$ $-(\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n-1})(\boldsymbol{K}_n\boldsymbol{v}_n)^T$ $-\boldsymbol{K}_n\boldsymbol{v}_n(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n-1})^T(\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})^T$ $+\boldsymbol{K}_n\boldsymbol{v}_n(\boldsymbol{K}_n\boldsymbol{v}_n)^T\Big)$ | Expand |
| $\boldsymbol{P}_{n,n} = E\Big((\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n-1})$ $\times(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n-1})^T(\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})^T\Big)$ $-E\Big((\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n-1})(\boldsymbol{K}_n\boldsymbol{v}_n)^T\Big)$ $-E\Big(\boldsymbol{K}_n\boldsymbol{v}_n(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n-1})^T(\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})^T\Big)$ $+E\big(\boldsymbol{K}_n\boldsymbol{v}_n(\boldsymbol{K}_n\boldsymbol{v}_n)^T\big)$ | Apply the rule $E(X \pm Y) = E(X) \pm E(Y)$ |
| $E\left((\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})\,(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n-1})\,(\boldsymbol{K}_n\boldsymbol{v}_n)^T\right) = 0$ $E\left(\boldsymbol{K}_n\boldsymbol{v}_n\,(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n-1})^T\,(\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})^T\right) = 0$ | $(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n-1})$ is the error of the prior estimate. It is uncorrelated with the current measurement noise $\boldsymbol{v}_n$. The expectation value of the product of two uncorrelated variables is zero. |
| $\boldsymbol{P}_{n,n} = E\Big((\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n-1})$ $\times(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n-1})^T(\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})^T\Big)$ $+E\big(\boldsymbol{K}_n\boldsymbol{v}_n\boldsymbol{v}_n^T\boldsymbol{K}_n^T\big)$ | Apply the matrix transpose property: $(\boldsymbol{AB})^T = \boldsymbol{B}^T\boldsymbol{A}^T$ |
| $\boldsymbol{P}_{n,n} = (\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})E\Big((\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n-1})$ $\times(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n-1})^T\Big)(\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})^T$ $+\boldsymbol{K}_nE\big(\boldsymbol{v}_n\boldsymbol{v}_n^T\big)\boldsymbol{K}_n^T$ | Apply the rule $E(aX) = aE(X)$ |

Continued on next page

Table 8.5: Covariance Update Equation derivation. (Continued)

| | |
|---|---|
| $E\left(\left(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n-1}\right)\left(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n-1}\right)^T\right) = \boldsymbol{P}_{n,n-1}$<br>$E\left(\boldsymbol{v}_n \boldsymbol{v}_n^T\right) = \boldsymbol{R}_n$ | $\boldsymbol{P}_{n,n-1}$ is the prior estimate covariance<br>$\boldsymbol{R}_n$ is the measurement covariance |
| $\boldsymbol{P}_{n,n} = \left(\boldsymbol{I} - \boldsymbol{K}_n \boldsymbol{H}\right) \boldsymbol{P}_{n,n-1} \left(\boldsymbol{I} - \boldsymbol{K}_n \boldsymbol{H}\right)^T$<br>$+\boldsymbol{K}_n \boldsymbol{R}_n \boldsymbol{K}_n^T$ | Covariance Update Equation! |

## 8.7 The Kalman Gain

The final equation is the Kalman Gain Equation. The Kalman Gain in matrix notation is given by:

**Kalman Gain**

$$\boldsymbol{K}_n = \boldsymbol{P}_{n,n-1} \boldsymbol{H}^T \left(\boldsymbol{H} \boldsymbol{P}_{n,n-1} \boldsymbol{H}^T + \boldsymbol{R}_n\right)^{-1} \tag{8.66}$$

Where:

| | |
|---|---|
| $\boldsymbol{K}_n$ | is the Kalman Gain |
| $\boldsymbol{P}_{n,n-1}$ | is the prior estimate covariance matrix of the current state (predicted at the previous step) |
| $\boldsymbol{H}$ | is the observation matrix |
| $\boldsymbol{R}_n$ | is the measurement noise covariance matrix |

### 8.7.1 Kalman Gain Equation Derivation

This chapter includes the derivation of the Kalman Gain Equation. You can jump to the next section if you don't care about the derivation.

First, let's rearrange the Covariance Update Equation:

| Equation | Notes |
|---|---|
| $\boldsymbol{P}_{n,n} = (\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})\,\boldsymbol{P}_{n,n-1}(\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})^T + \boldsymbol{K}_n\boldsymbol{R}_n\boldsymbol{K}_n^T$ | Covariance Update Equation |
| $\boldsymbol{P}_{n,n} = (\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})\,\boldsymbol{P}_{n,n-1}\left(\boldsymbol{I} - (\boldsymbol{K}_n\boldsymbol{H})^T\right) + \boldsymbol{K}_n\boldsymbol{R}_n\boldsymbol{K}_n^T$ | $\boldsymbol{I}^T = \boldsymbol{I}$ |
| $\boldsymbol{P}_{n,n} = (\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})\,\boldsymbol{P}_{n,n-1}\left(\boldsymbol{I} - \boldsymbol{H}^T\boldsymbol{K}_n^T\right) + \boldsymbol{K}_n\boldsymbol{R}_n\boldsymbol{K}_n^T$ | Apply the matrix transpose property: $(\boldsymbol{AB})^T = \boldsymbol{B}^T\boldsymbol{A}^T$ |
| $\boldsymbol{P}_{n,n} = (\boldsymbol{P}_{n,n-1} - \boldsymbol{K}_n\boldsymbol{H}\boldsymbol{P}_{n,n-1})\left(\boldsymbol{I} - \boldsymbol{H}^T\boldsymbol{K}_n^T\right) + \boldsymbol{K}_n\boldsymbol{R}_n\boldsymbol{K}_n^T$ | |
| $\boldsymbol{P}_{n,n} = \boldsymbol{P}_{n,n-1} - \boldsymbol{P}_{n,n-1}\boldsymbol{H}^T\boldsymbol{K}_n^T - \boldsymbol{K}_n\boldsymbol{H}\boldsymbol{P}_{n,n-1} + \boldsymbol{K}_n\boldsymbol{H}\boldsymbol{P}_{n,n-1}\boldsymbol{H}^T\boldsymbol{K}_n^T + \boldsymbol{K}_n\boldsymbol{R}_n\boldsymbol{K}_n^T$ | Expand |
| $\boldsymbol{P}_{n,n} = \boldsymbol{P}_{n,n-1} - \boldsymbol{P}_{n,n-1}\boldsymbol{H}^T\boldsymbol{K}_n^T - \boldsymbol{K}_n\boldsymbol{H}\boldsymbol{P}_{n,n-1} + \boldsymbol{K}_n\left(\boldsymbol{H}\boldsymbol{P}_{n,n-1}\boldsymbol{H}^T + \boldsymbol{R}_n\right)\boldsymbol{K}_n^T$ | Group the last two terms |

**Table 8.6:** *Covariance Update Equation rearrange.*

The Kalman Filter is an **optimal filter**. Thus, we seek a Kalman Gain that minimizes the estimate variance.

To minimize the estimate variance, we need to minimize the main diagonal (from the upper left to the lower right) of the covariance matrix $\boldsymbol{P}_{n,n}$.

The sum of the main diagonal of the square matrix is the **trace** of the matrix. Thus, we need to minimize $tr(\boldsymbol{P}_{n,n})$. To find the conditions required to produce a minimum, we differentiate the trace of $\boldsymbol{P}_{n,n}$ with respect to $\boldsymbol{K}_n$ and set the result to zero.

| Equation | Notes |
|---|---|
| $tr\left(\boldsymbol{P}_{n,n}\right) = tr\left(\boldsymbol{P}_{n,n-1}\right)$<br>$-tr\left(\boldsymbol{P}_{n,n-1}\boldsymbol{H}^T\boldsymbol{K}_n^T\right) - tr\left(\boldsymbol{K}_n\boldsymbol{H}\boldsymbol{P}_{n,n-1}\right)$<br>$+tr\Big(\boldsymbol{K}_n\left(\boldsymbol{H}\boldsymbol{P}_{n,n-1}\boldsymbol{H}^T + \boldsymbol{R}_n\right)\boldsymbol{K}_n^T\Big)$ | Trace of the Covariance Update Equation |
| $tr\left(\boldsymbol{P}_{n,n}\right) = tr\left(\boldsymbol{P}_{n,n-1}\right) - 2tr\left(\boldsymbol{K}_n\boldsymbol{H}\boldsymbol{P}_{n,n-1}\right)$<br>$+tr\Big(\boldsymbol{K}_n\left(\boldsymbol{H}\boldsymbol{P}_{n,n-1}\boldsymbol{H}^T + \boldsymbol{R}_n\right)\boldsymbol{K}_n^T\Big)$ | The trace of the matrix is equal to the trace of its transpose (the same main diagonal) |
| $\frac{d\Big(tr\left(\boldsymbol{P}_{n,n}\right)\Big)}{d\boldsymbol{K}_n} = \frac{d\Big(tr\left(\boldsymbol{P}_{n,n-1}\right)\Big)}{d\boldsymbol{K}_n}$<br>$- \frac{d\Big(2tr\left(\boldsymbol{K}_n\boldsymbol{H}\boldsymbol{P}_{n,n-1}\right)\Big)}{d\boldsymbol{K}_n}$<br>$+ \frac{d\Big(tr\Big(\boldsymbol{K}_n\left(\boldsymbol{H}\boldsymbol{P}_{n,n-1}\boldsymbol{H}^T + \boldsymbol{R}_n\right)\boldsymbol{K}_n^T\Big)\Big)}{d\boldsymbol{K}_n} = 0$ | Differentiate the trace of $\boldsymbol{P}_{n,n}$ with respect to $\boldsymbol{K}_n$ |
| $\frac{d\Big(tr\left(\boldsymbol{P}_{n,n}\right)\Big)}{d\boldsymbol{K}_n} = 0 - 2\left(\boldsymbol{H}\boldsymbol{P}_{n,n-1}\right)^T$<br>$+2\boldsymbol{K}_n\left(\boldsymbol{H}\boldsymbol{P}_{n,n-1}\boldsymbol{H}^T + \boldsymbol{R}_n\right) = 0$ | $\frac{d}{d\boldsymbol{A}}\Big(tr\left(\boldsymbol{A}\boldsymbol{B}\right)\Big) = \boldsymbol{B}^T$<br>$\frac{d}{d\boldsymbol{A}}\Big(tr\left(\boldsymbol{A}\boldsymbol{B}\boldsymbol{A}^T\right)\Big) = 2\boldsymbol{A}\boldsymbol{B}$<br>See the proof in Appendix D |
| $\left(\boldsymbol{H}\boldsymbol{P}_{n,n-1}\right)^T = \boldsymbol{K}_n\left(\boldsymbol{H}\boldsymbol{P}_{n,n-1}\boldsymbol{H}^T + \boldsymbol{R}_n\right)$ | |
| $\boldsymbol{K}_n = \left(\boldsymbol{H}\boldsymbol{P}_{n,n-1}\right)^T\left(\boldsymbol{H}\boldsymbol{P}_{n,n-1}\boldsymbol{H}^T + \boldsymbol{R}_n\right)^{-1}$ | |
| $\boldsymbol{K}_n = \boldsymbol{P}_{n,n-1}^T\boldsymbol{H}^T\left(\boldsymbol{H}\boldsymbol{P}_{n,n-1}\boldsymbol{H}^T + \boldsymbol{R}_n\right)^{-1}$ | Apply the matrix transpose property: $(\boldsymbol{A}\boldsymbol{B})^T = \boldsymbol{B}^T\boldsymbol{A}^T$ |
| $\boldsymbol{K}_n = \boldsymbol{P}_{n,n-1}\boldsymbol{H}^T\left(\boldsymbol{H}\boldsymbol{P}_{n,n-1}\boldsymbol{H}^T + \boldsymbol{R}_n\right)^{-1}$ | The Covariance matrix is a symmetric matrix: $\boldsymbol{P}_{n,n-1}^T = \boldsymbol{P}_{n,n-1}$ |

**Table 8.7:** *Kalman Gain Equation Derivation.*

## 8.8 Simplified Covariance Update Equation

In many textbooks, you can find a simplified form of the Covariance Update Equation:

$$\boldsymbol{P}_{n,n} = (\boldsymbol{I} - \boldsymbol{K}_n \boldsymbol{H})\, \boldsymbol{P}_{n,n-1} \tag{8.67}$$

To derive a simplified form of the Covariance Update Equation, plug the Kalman Gain Equation into the Covariance Update Equation.

| Equation | Notes |
|---|---|
| $\boldsymbol{P}_{n,n} = \boldsymbol{P}_{n,n-1} - \boldsymbol{P}_{n,n-1}\boldsymbol{H}^T\boldsymbol{K}_n^T - \boldsymbol{K}_n\boldsymbol{H}\boldsymbol{P}_{n,n-1}$ $+\boldsymbol{K}_n \left(\boldsymbol{H}\boldsymbol{P}_{n,n-1}\boldsymbol{H}^T + \boldsymbol{R}_n\right)\boldsymbol{K}_n^T$ | Covariance Update Equation after expansion (see section 8.6) |
| $\boldsymbol{P}_{n,n} = \boldsymbol{P}_{n,n-1} - \boldsymbol{P}_{n,n-1}\boldsymbol{H}^T\boldsymbol{K}_n^T - \boldsymbol{K}_n\boldsymbol{H}\boldsymbol{P}_{n,n-1}$ $+\boldsymbol{P}_{n,n-1}\boldsymbol{H}^T \left(\boldsymbol{H}\boldsymbol{P}_{n,n-1}\boldsymbol{H}^T + \boldsymbol{R}_n\right)^{-1}$ $\times \left(\boldsymbol{H}\boldsymbol{P}_{n,n-1}\boldsymbol{H}^T + \boldsymbol{R}_n\right)\boldsymbol{K}_n^T$ | Substitute the Kalman Gain Equation |
| $\boldsymbol{P}_{n,n} = \boldsymbol{P}_{n,n-1} - \boldsymbol{P}_{n,n-1}\boldsymbol{H}^T\boldsymbol{K}_n^T - \boldsymbol{K}_n\boldsymbol{H}\boldsymbol{P}_{n,n-1}$ $+\boldsymbol{P}_{n,n-1}\boldsymbol{H}^T\boldsymbol{K}_n^T$ | $\left(\boldsymbol{H}\boldsymbol{P}_{n,n-1}\boldsymbol{H}^T + \boldsymbol{R}_n\right)^{-1} \times$ $\times \left(\boldsymbol{H}\boldsymbol{P}_{n,n-1}\boldsymbol{H}^T + \boldsymbol{R}_n\right) = 1$ |
| $\boldsymbol{P}_{n,n} = \boldsymbol{P}_{n,n-1} - \boldsymbol{K}_n\boldsymbol{H}\boldsymbol{P}_{n,n-1}$ | |
| $\boldsymbol{P}_{n,n} = (\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})\, \boldsymbol{P}_{n,n-1}$ | |

**Table 8.8:** *Equations for the Covariance Update Equation derivation.*

**Warning!** This equation is much more elegant and easier to remember and it performs well in many cases. However, **a minor error in computing the Kalman Gain (due to round-off) can lead to huge computation errors. The subtraction $(\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})$ can lead to nonsymmetric matrices due to floating-point errors. This equation is numerically unstable!**

For more details, see [17].

## 8.9 Summary

We have derived all five Kalman Filter equations in matrix notation. Let us put them all together on a single page. The Kalman Filter operates in a "predict-correct" loop, as shown in the diagram below.

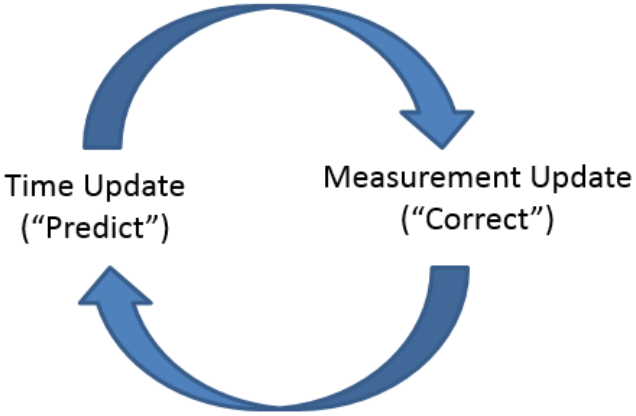


**Figure 8.7:** *Predict-Update Diagram.*

Once initialized, the Kalman Filter **predicts** the system state at the next step. It also provides the uncertainty of the prediction.

Once the measurement is received, the Kalman Filter **updates** (or **corrects**) the prediction and the uncertainty of the current state. As well the Kalman Filter predicts the following states, and so on. The following diagram provides a complete picture of the Kalman Filter operation.

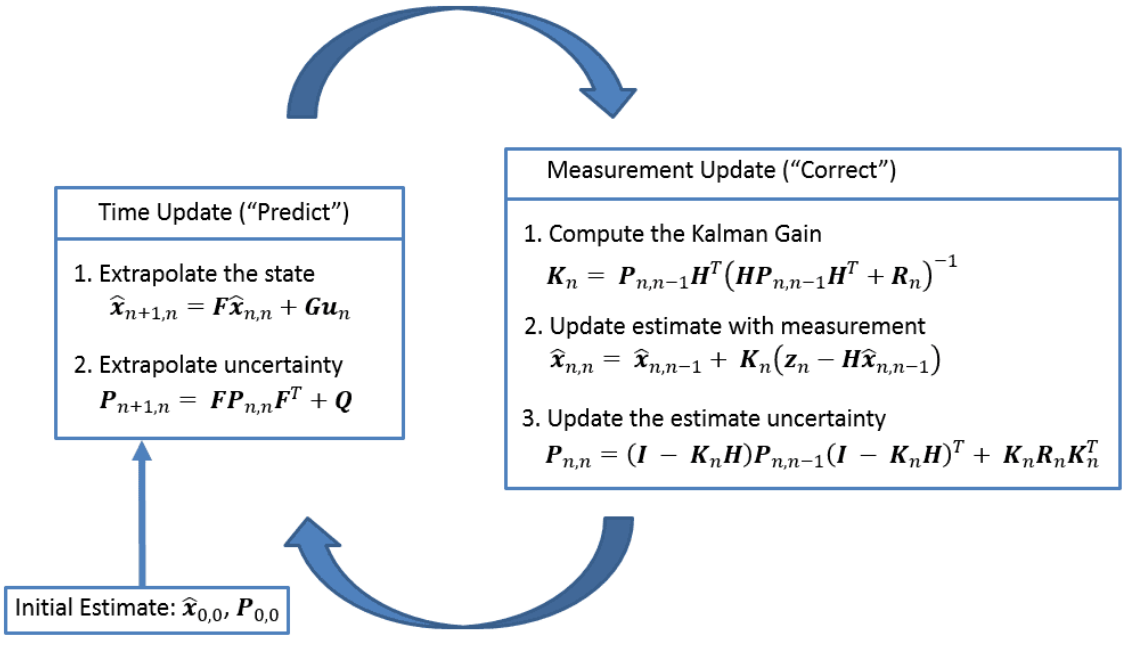


**Figure 8.8:** *The Kalman Filter Diagram.*

The following table describes all Kalman Filter Equations.

| | **Equation** | **Name** | **Alternative names in the literature** |
|---|---|---|---|
| Predict | $\hat{\boldsymbol{x}}_{n+1,n} = \boldsymbol{F}\hat{\boldsymbol{x}}_{n,n} + \boldsymbol{G}\boldsymbol{u}_n$ | State Extrapolation | Predictor Equation<br>Transition Equation<br>Prediction Equation<br>Dynamic Model<br>State Space Model |
| | $\boldsymbol{P}_{n+1,n} = \boldsymbol{F}\boldsymbol{P}_{n,n}\boldsymbol{F}^T + \boldsymbol{Q}$ | Covariance Extrapolation | Predictor Covariance Equation |
| Update | $\hat{\boldsymbol{x}}_{n,n} = \hat{\boldsymbol{x}}_{n,n-1} + \boldsymbol{K}_n$<br>$\times(\boldsymbol{z}_n - \boldsymbol{H}\hat{\boldsymbol{x}}_{n,n-1})$ | State Update | Filtering Equation |
| | $\boldsymbol{P}_{n,n} = (\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})\,\boldsymbol{P}_{n,n-1}$<br>$\times\,(\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})^T + \boldsymbol{K}_n\boldsymbol{R}_n\boldsymbol{K}_n^T$ | Covariance Update | Corrector Equation |
| | $\boldsymbol{K}_n = \boldsymbol{P}_{n,n-1}\boldsymbol{H}^T$<br>$\times\left(\boldsymbol{H}\boldsymbol{P}_{n,n-1}\boldsymbol{H}^T + \boldsymbol{R}_n\right)^{-1}$ | Kalman Gain | Weight Equation |
| Auxiliary | $\boldsymbol{z}_n = \boldsymbol{H}\boldsymbol{x}_n$ | Measurement Equation | |
| | $\boldsymbol{R}_n = E\left(\boldsymbol{v}_n\boldsymbol{v}_n^T\right)$ | Measurement Covariance | Measurement Error |
| | $\boldsymbol{Q}_n = E\left(\boldsymbol{w}_n\boldsymbol{w}_n^T\right)$ | Process Noise Covariance | Process Noise Error |
| | $\boldsymbol{P}_{n,n} = E\left(\boldsymbol{e}_n\boldsymbol{e}_n^T\right)$<br>$= E\left(\left(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n}\right)\left(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n}\right)^T\right)$ | Estimation Covariance | Estimation Error |

**Table 8.9:** *Kalman Filter equations.*

The following table summarizes notation (including differences found in the literature) and dimensions.

| Term | Name | Alternative term | Dimensions |
|---|---|---|---|
| $\boldsymbol{x}$ | State Vector | | $n_x \times 1$ |
| $\boldsymbol{z}$ | Measurements Vector | $\boldsymbol{y}$ | $n_z \times 1$ |
| $\boldsymbol{F}$ | State Transition Matrix | $\boldsymbol{\Phi}, \boldsymbol{A}$ | $n_x \times n_x$ |
| $\boldsymbol{u}$ | Input Variable | | $n_u \times 1$ |
| $\boldsymbol{G}$ | Control Matrix | $\boldsymbol{B}$ | $n_x \times n_u$ |
| $\boldsymbol{P}$ | Estimate Covariance | $\boldsymbol{\Sigma}$ | $n_x \times n_x$ |
| $\boldsymbol{Q}$ | Process Noise Covariance | | $n_x \times n_x$ |
| $\boldsymbol{R}$ | Measurement Covariance | | $n_z \times n_z$ |
| $\boldsymbol{w}$ | Process Noise Vector | $\boldsymbol{y}$ | $n_x \times 1$ |
| $\boldsymbol{v}$ | Measurement Noise Vector | | $n_z \times 1$ |
| $\boldsymbol{H}$ | Observation Matrix | $\boldsymbol{C}$ | $n_z \times n_x$ |
| $\boldsymbol{K}$ | Kalman Gain | | $n_x \times n_z$ |
| $\boldsymbol{n}$ | Discrete-Time Index | $k$ | |

**Table 8.10:** *Kalman Filter notation.*

Dimensions notation:

- $n_x$ is a number of states in a state vector
- $n_z$ is a number of measured states
- $n_u$ is a number of elements of the input variable

# 9. Multivariate KF Examples

It is the final part of the Multivariate Kalman Filter chapter. It includes two numerical examples. In the first example, we design a six-dimensional Kalman Filter without control input. In the second example, we design a two-dimensional Kalman Filter with a control input.

## 9.1 Example 9 – vehicle location estimation

In the following example, we implement the Multivariate Kalman Filter using the material we've learned.

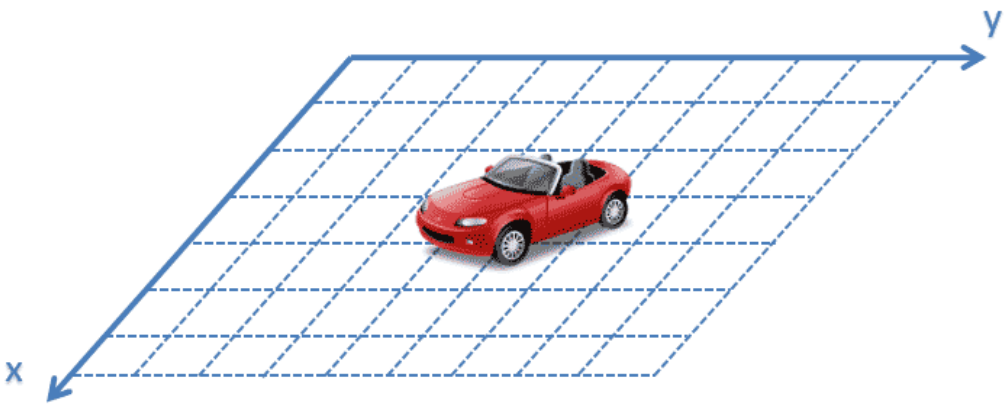


**Figure 9.1:** *Vehicle location estimation.*

In this example, we estimate the vehicle's location on the $XY$ plane. The vehicle has an onboard location sensor that reports $X$ and $Y$ coordinates of the system. We assume constant acceleration dynamics.

### 9.1.1 Kalman Filter equations

#### The state extrapolation equation

First, we derive the state extrapolation equation. As you remember, the general form of the state extrapolation equation in matrix notation is:

$$\hat{\boldsymbol{x}}_{n+1,n} = \boldsymbol{F}\hat{\boldsymbol{x}}_{n,n} + \boldsymbol{G}\boldsymbol{u}_n + \boldsymbol{w}_n \tag{9.1}$$

Where:

| | |
|---|---|
| $\hat{\boldsymbol{x}}_{n+1,n}$ | is a predicted system state vector at time step $n+1$ |
| $\hat{\boldsymbol{x}}_{n,n}$ | is an estimated system state vector at time step $n$ |
| $\boldsymbol{u}_n$ | is a **control variable** or **input variable** - a measurable (deterministic) input to the system |
| $\boldsymbol{w}_n$ | is a **process noise** or disturbance - an unmeasurable input that affects the state |
| $\boldsymbol{F}$ | is a **state transition matrix** |
| $\boldsymbol{G}$ | is a **control matrix** or **input transition matrix** (mapping control to state variables) |

In this example, there is no control variable $\boldsymbol{u}$ since there is no control input.

For this example, the state extrapolation equation can be simplified to:

$$\hat{\boldsymbol{x}}_{n+1,n} = \boldsymbol{F}\hat{\boldsymbol{x}}_{n,n} \tag{9.2}$$

The system state $\boldsymbol{x}_n$ is defined by:

$$\boldsymbol{x}_n = \begin{bmatrix} x_n \\ \dot{x}_n \\ \ddot{x}_n \\ y_n \\ \dot{y}_n \\ \ddot{y}_n \end{bmatrix} \tag{9.3}$$

The extrapolated vehicle state for time $n+1$ can be described by the following system of equations:

$$\begin{cases} \hat{x}_{n+1,n} = \hat{x}_{n,n} + \hat{\dot{x}}_{n,n}\Delta t + \frac{1}{2}\hat{\ddot{x}}_{n,n}\Delta t^2 \\ \hat{\dot{x}}_{n+1,n} = \hat{\dot{x}}_{n,n} + \hat{\ddot{x}}_{n,n}\Delta t \\ \hat{\ddot{x}}_{n+1,n} = \hat{\ddot{x}}_{n,n} \\ \hat{y}_{n+1,n} = \hat{y}_{n,n} + \hat{\dot{y}}_{n,n}\Delta t + \frac{1}{2}\hat{\ddot{y}}_{n,n}\Delta t^2 \\ \hat{\dot{y}}_{n+1,n} = \hat{\dot{y}}_{n,n} + \hat{\ddot{y}}_{n,n}\Delta t \\ \hat{\ddot{y}}_{n+1,n} = \hat{\ddot{y}}_{n,n} \end{cases} \tag{9.4}$$

In matrix form:

$$\begin{bmatrix} \hat{x}_{n+1,n} \\ \hat{\dot{x}}_{n+1,n} \\ \hat{\ddot{x}}_{n+1,n} \\ \hat{y}_{n+1,n} \\ \hat{\dot{y}}_{n+1,n} \\ \hat{\ddot{y}}_{n+1,n} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t & 0.5\Delta t^2 & 0 & 0 & 0 \\ 0 & 1 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta t & 0.5\Delta t^2 \\ 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{x}_{n,n} \\ \hat{\dot{x}}_{n,n} \\ \hat{\ddot{x}}_{n,n} \\ \hat{y}_{n,n} \\ \hat{\dot{y}}_{n,n} \\ \hat{\ddot{y}}_{n,n} \end{bmatrix} \tag{9.5}$$

## The covariance extrapolation equation

The general form of the Covariance Extrapolation Equation is given by:

$$\boldsymbol{P}_{n+1,n} = \boldsymbol{F}\boldsymbol{P}_{n,n}\boldsymbol{F}^T + \boldsymbol{Q} \tag{9.6}$$

Where:

| | |
|---|---|
| $\boldsymbol{P}_{n,n}$ | is the covariance matrix of the current state estimation |
| $\boldsymbol{P}_{n+1,n}$ | is the covariance matrix of the next state estimation (prediction) |
| $\boldsymbol{F}$ | is the state transition matrix that we derived in Appendix C ("Modeling linear dynamic systems") |
| $\boldsymbol{Q}$ | is the process noise matrix |

The estimate covariance matrix is:

$$\boldsymbol{P} = \begin{bmatrix} p_x & p_{x\dot{x}} & p_{x\ddot{x}} & p_{xy} & p_{x\dot{y}} & p_{x\ddot{y}} \\ p_{\dot{x}x} & p_{\dot{x}} & p_{\dot{x}\ddot{x}} & p_{\dot{x}y} & p_{\dot{x}\dot{y}} & p_{\dot{x}\ddot{y}} \\ p_{\ddot{x}x} & p_{\ddot{x}\dot{x}} & p_{\ddot{x}} & p_{\ddot{x}y} & p_{\ddot{x}\dot{y}} & p_{\ddot{x}\ddot{y}} \\ p_{yx} & p_{y\dot{x}} & p_{y\ddot{x}} & p_y & p_{y\dot{y}} & p_{y\ddot{y}} \\ p_{\dot{y}x} & p_{\dot{y}\dot{x}} & p_{\dot{y}\ddot{x}} & p_{\dot{y}y} & p_{\dot{y}} & p_{\dot{y}\ddot{y}} \\ p_{\ddot{y}x} & p_{\ddot{y}\dot{x}} & p_{\ddot{y}\ddot{x}} & p_{\ddot{y}y} & p_{\ddot{y}\dot{y}} & p_{\ddot{y}} \end{bmatrix} \tag{9.7}$$

The elements on the main diagonal of the matrix are the variances of the estimation:

- $p_x$ is the variance of the $X$ coordinate position estimation
- $p_{\dot{x}}$ is the variance of the $X$ coordinate velocity estimation
- $p_{\ddot{x}}$ is the variance of the $X$ coordinate acceleration estimation
- $p_y$ is the variance of the $Y$ coordinate position estimation
- $p_{\dot{y}}$ is the variance of the $Y$ coordinate velocity estimation
- $p_{\ddot{y}}$ is the variance of the $Y$ coordinate acceleration estimation
- The off-diagonal entries are covariances

We assume that the estimation errors in $X$ and $Y$ axes are not correlated so that the mutual terms can be set to zero.

$$\boldsymbol{P} = \begin{bmatrix} p_x & p_{x\dot{x}} & p_{x\ddot{x}} & 0 & 0 & 0 \\ p_{\dot{x}x} & p_{\dot{x}} & p_{\dot{x}\ddot{x}} & 0 & 0 & 0 \\ p_{\ddot{x}x} & p_{\ddot{x}\dot{x}} & p_{\ddot{x}} & 0 & 0 & 0 \\ 0 & 0 & 0 & p_y & p_{y\dot{y}} & p_{y\ddot{y}} \\ 0 & 0 & 0 & p_{\dot{y}y} & p_{\dot{y}} & p_{\dot{y}\ddot{y}} \\ 0 & 0 & 0 & p_{\ddot{y}y} & p_{\ddot{y}\dot{y}} & p_{\ddot{y}} \end{bmatrix} \tag{9.8}$$

We have already derived the state transition matrix $\boldsymbol{F}$. Now we shall derive the process noise $\boldsymbol{Q}$ matrix.

## The process noise matrix

We assume a discrete noise model - the noise is different at each time sample but is constant between time samples.

The process noise matrix for the two-dimensional constant acceleration model looks as follows:

$$\boldsymbol{Q} = \begin{bmatrix} \sigma_x^2 & \sigma_{x\dot{x}}^2 & \sigma_{x\ddot{x}}^2 & \sigma_{xy}^2 & \sigma_{x\dot{y}}^2 & \sigma_{x\ddot{y}}^2 \\ \sigma_{\dot{x}x}^2 & \sigma_{\dot{x}}^2 & \sigma_{\dot{x}\ddot{x}}^2 & \sigma_{\dot{x}y}^2 & \sigma_{\dot{x}\dot{y}}^2 & \sigma_{\dot{x}\ddot{y}}^2 \\ \sigma_{\ddot{x}x}^2 & \sigma_{\ddot{x}\dot{x}}^2 & \sigma_{\ddot{x}}^2 & \sigma_{\ddot{x}y}^2 & \sigma_{\ddot{x}\dot{y}}^2 & \sigma_{\ddot{x}\ddot{y}}^2 \\ \sigma_{yx}^2 & \sigma_{y\dot{x}}^2 & \sigma_{y\ddot{x}}^2 & \sigma_y^2 & \sigma_{y\dot{y}}^2 & \sigma_{y\ddot{y}}^2 \\ \sigma_{\dot{y}x}^2 & \sigma_{\dot{y}\dot{x}}^2 & \sigma_{\dot{y}\ddot{x}}^2 & \sigma_{\dot{y}y}^2 & \sigma_{\dot{y}}^2 & \sigma_{\dot{y}\ddot{y}}^2 \\ \sigma_{\ddot{y}x}^2 & \sigma_{\ddot{y}\dot{x}}^2 & \sigma_{\ddot{y}\ddot{x}}^2 & \sigma_{\ddot{y}y}^2 & \sigma_{\ddot{y}\dot{y}}^2 & \sigma_{\ddot{y}}^2 \end{bmatrix} \tag{9.9}$$

We assume that the process noise in $X$ and $Y$ axes is not correlated so that the mutual terms can be set to zero.

$$\boldsymbol{Q} = \begin{bmatrix} \sigma_x^2 & \sigma_{x\dot{x}}^2 & \sigma_{x\ddot{x}}^2 & 0 & 0 & 0 \\ \sigma_{\dot{x}x}^2 & \sigma_{\dot{x}}^2 & \sigma_{\dot{x}\ddot{x}}^2 & 0 & 0 & 0 \\ \sigma_{\ddot{x}x}^2 & \sigma_{\ddot{x}\dot{x}}^2 & \sigma_{\ddot{x}}^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_y^2 & \sigma_{y\dot{y}}^2 & \sigma_{y\ddot{y}}^2 \\ 0 & 0 & 0 & \sigma_{\dot{y}y}^2 & \sigma_{\dot{y}}^2 & \sigma_{\dot{y}\ddot{y}}^2 \\ 0 & 0 & 0 & \sigma_{\ddot{y}y}^2 & \sigma_{\ddot{y}\dot{y}}^2 & \sigma_{\ddot{y}}^2 \end{bmatrix} \tag{9.10}$$

We've already derived the $\boldsymbol{Q}$ matrix for the constant acceleration motion model (Equation 8.44). The $\boldsymbol{Q}$ matrix for our example is:

$$\boldsymbol{Q} = \begin{bmatrix} \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} & \frac{\Delta t^2}{2} & 0 & 0 & 0 \\ \frac{\Delta t^3}{2} & \Delta t^2 & \Delta t & 0 & 0 & 0 \\ \frac{\Delta t^2}{2} & \Delta t & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} & \frac{\Delta t^2}{2} \\ 0 & 0 & 0 & \frac{\Delta t^3}{2} & \Delta t^2 & \Delta t \\ 0 & 0 & 0 & \frac{\Delta t^2}{2} & \Delta t & 1 \end{bmatrix} \sigma_a^2 \tag{9.11}$$

Where:

- $\Delta t$ is the time between successive measurements
- $\sigma_a^2$ is a random variance in acceleration

Now we can write down the covariance extrapolation equation for our example:

$$\boldsymbol{P}_{n+1,n} = \boldsymbol{F}\boldsymbol{P}_{n,n}\boldsymbol{F}^T + \boldsymbol{Q} \tag{9.12}$$

$$
\begin{bmatrix}
p_{x_{n+1,n}} & p_{x\dot{x}_{n+1,n}} & p_{x\ddot{x}_{n+1,n}} & 0 & 0 & 0 \\
p_{\dot{x}x_{n+1,n}} & p_{\dot{x}_{n+1,n}} & p_{\dot{x}\ddot{x}_{n+1,n}} & 0 & 0 & 0 \\
p_{\ddot{x}x_{n+1,n}} & p_{\ddot{x}\dot{x}_{n+1,n}} & p_{\ddot{x}_{n+1,n}} & 0 & 0 & 0 \\
0 & 0 & 0 & p_{y_{n+1,n}} & p_{y\dot{y}_{n+1,n}} & p_{y\ddot{y}_{n+1,n}} \\
0 & 0 & 0 & p_{\dot{y}y_{n+1,n}} & p_{\dot{y}_{n+1,n}} & p_{\dot{y}\ddot{y}_{n+1,n}} \\
0 & 0 & 0 & p_{\ddot{y}y_{n+1,n}} & p_{\ddot{y}\dot{y}_{n+1,n}} & p_{\ddot{y}_{n+1,n}}
\end{bmatrix}
$$

$$
= \begin{bmatrix}
1 & \Delta t & 0.5\Delta t^2 & 0 & 0 & 0 \\
0 & 1 & \Delta t & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & \Delta t & 0.5\Delta t^2 \\
0 & 0 & 0 & 0 & 1 & \Delta t \\
0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
$$

$$
\times \begin{bmatrix}
p_{x_{n,n}} & p_{x\dot{x}_{n,n}} & p_{x\ddot{x}_{n,n}} & 0 & 0 & 0 \\
p_{\dot{x}x_{n,n}} & p_{\dot{x}_{n,n}} & p_{\dot{x}\ddot{x}_{n,n}} & 0 & 0 & 0 \\
p_{\ddot{x}x_{n,n}} & p_{\ddot{x}\dot{x}_{n,n}} & p_{\ddot{x}_{n,n}} & 0 & 0 & 0 \\
0 & 0 & 0 & p_{y_{n,n}} & p_{y\dot{y}_{n,n}} & p_{y\ddot{y}_{n,n}} \\
0 & 0 & 0 & p_{\dot{y}y_{n,n}} & p_{\dot{y}_{n,n}} & p_{\dot{y}\ddot{y}_{n,n}} \\
0 & 0 & 0 & p_{\ddot{y}y_{n,n}} & p_{\ddot{y}\dot{y}_{n,n}} & p_{\ddot{y}_{n,n}}
\end{bmatrix}
\times \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 \\
\Delta t & 1 & 0 & 0 & 0 & 0 \\
0.5\Delta t^2 & \Delta t & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & \Delta t & 1 & 0 \\
0 & 0 & 0 & 0.5\Delta t^2 & \Delta t & 1
\end{bmatrix}
$$

$$
+ \begin{bmatrix}
\frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} & \frac{\Delta t^2}{2} & 0 & 0 & 0 \\
\frac{\Delta t^3}{2} & \Delta t^2 & \Delta t & 0 & 0 & 0 \\
\frac{\Delta t^2}{2} & \Delta t & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} & \frac{\Delta t^2}{2} \\
0 & 0 & 0 & \frac{\Delta t^3}{2} & \Delta t^2 & \Delta t \\
0 & 0 & 0 & \frac{\Delta t^2}{2} & \Delta t & 1
\end{bmatrix} \sigma_a^2
$$

(9.13)

## The measurement equation

The generalized measurement equation in a matrix form is given by:

$$\boldsymbol{z}_n = \boldsymbol{H}\boldsymbol{x}_n + \boldsymbol{v}_n \tag{9.14}$$

Where:

- $\boldsymbol{z}_n$ is a measurement vector
- $\boldsymbol{x}_n$ is a true system state (hidden state)
- $\boldsymbol{v}_n$ is a random noise vector
- $\boldsymbol{H}$ is an **observation matrix**

The measurement provides us only $X$ and $Y$ coordinates of the vehicle.

So: $\boldsymbol{z}_n = \begin{bmatrix} x_{n,measured} \\ y_{n,measured} \end{bmatrix}$

$$\boldsymbol{z_n = H x_n} \tag{9.15}$$

$$\begin{bmatrix} x_{n,measured} \\ y_{n,measured} \end{bmatrix} = \boldsymbol{H} \begin{bmatrix} x_n \\ \dot{x}_n \\ \ddot{x}_n \\ y_n \\ \dot{y}_n \\ \ddot{y}_n \end{bmatrix} \tag{9.16}$$

The dimension of $\boldsymbol{z}_n$ is $2 \times 1$ and the dimension of $\boldsymbol{x}_n$ is $6 \times 1$. Therefore the dimension of the observation matrix $\boldsymbol{H}$ shall be $2 \times 6$.

$$\boldsymbol{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{9.17}$$

## The measurement uncertainty

The measurement covariance matrix is:

$$\boldsymbol{R}_n = \begin{bmatrix} \sigma^2_{x_m} & \sigma^2_{yx_m} \\ \sigma^2_{xy_m} & \sigma^2_{y_m} \end{bmatrix} \tag{9.18}$$

The subscript $m$ is for measurement uncertainty.

Assume that the $x$ and $y$ measurements are uncorrelated, i.e., the error in the $x$ coordinate measurement doesn't depend on the error in the $y$ coordinate measurement.

$$\boldsymbol{R}_n = \begin{bmatrix} \sigma^2_{x_m} & 0 \\ 0 & \sigma^2_{y_m} \end{bmatrix} \tag{9.19}$$

In real-life applications, the measurement uncertainty can differ between measurements. In many systems, the measurement uncertainty depends on the measurement SNR, the angle between the sensor (or sensors) and target, signal frequency, and many other parameters.

For the sake of the example simplicity, we assume a constant measurement uncertainty:

$$\boldsymbol{R}_1 = \boldsymbol{R}_2 ... \boldsymbol{R}_{n-1} = \boldsymbol{R}_n = \boldsymbol{R} \tag{9.20}$$

## The Kalman Gain

The Kalman Gain in matrix notation is given by:

$$\boldsymbol{K}_n = \boldsymbol{P}_{n,n-1}\boldsymbol{H}^T \left(\boldsymbol{H}\boldsymbol{P}_{n,n-1}\boldsymbol{H}^T + \boldsymbol{R}_n\right)^{-1} \tag{9.21}$$

Where:

| | |
|---|---|
| $\boldsymbol{K}_n$ | is the Kalman Gain |
| $\boldsymbol{P}_{n,n-1}$ | is the prior estimate covariance matrix of the current state (predicted at the previous step) |
| $\boldsymbol{H}$ | is the observation matrix |
| $\boldsymbol{R}_n$ | is the measurement noise covariance matrix |

We have already derived all the building blocks of the Kalman Gain:

$$\begin{bmatrix} K_{1,1_n} & K_{1,2_n} \\ K_{2,1_n} & K_{2,2_n} \\ K_{3,1_n} & K_{3,2_n} \\ K_{4,1_n} & K_{4,2_n} \\ K_{5,1_n} & K_{5,2_n} \\ K_{6,1_n} & K_{6,2_n} \end{bmatrix} = \begin{bmatrix} p_{x_{n,n-1}} & p_{x\dot{x}_{n,n-1}} & p_{x\ddot{x}_{n,n-1}} & 0 & 0 & 0 \\ p_{\dot{x}x_{n,n-1}} & p_{\dot{x}_{n,n-1}} & p_{\dot{x}\ddot{x}_{n,n-1}} & 0 & 0 & 0 \\ p_{\ddot{x}x_{n,n-1}} & p_{\ddot{x}\dot{x}_{n,n-1}} & p_{\ddot{x}_{n,n-1}} & 0 & 0 & 0 \\ 0 & 0 & 0 & p_{y_{n,n-1}} & p_{y\dot{y}_{n,n-1}} & p_{y\ddot{y}_{n,n-1}} \\ 0 & 0 & 0 & p_{\dot{y}y_{n,n-1}} & p_{\dot{y}_{n,n-1}} & p_{\dot{y}\ddot{y}_{n,n-1}} \\ 0 & 0 & 0 & p_{\ddot{y}y_{n,n-1}} & p_{\ddot{y}\dot{y}_{n,n-1}} & p_{\ddot{y}_{n,n-1}} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$\times \left( \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} p_{x_{n,n-1}} & p_{x\dot{x}_{n,n-1}} & p_{x\ddot{x}_{n,n-1}} & 0 & 0 & 0 \\ p_{\dot{x}x_{n,n-1}} & p_{\dot{x}_{n,n-1}} & p_{\dot{x}\ddot{x}_{n,n-1}} & 0 & 0 & 0 \\ p_{\ddot{x}x_{n,n-1}} & p_{\ddot{x}\dot{x}_{n,n-1}} & p_{\ddot{x}_{n,n-1}} & 0 & 0 & 0 \\ 0 & 0 & 0 & p_{y_{n,n-1}} & p_{y\dot{y}_{n,n-1}} & p_{y\ddot{y}_{n,n-1}} \\ 0 & 0 & 0 & p_{\dot{y}y_{n,n-1}} & p_{\dot{y}_{n,n-1}} & p_{\dot{y}\ddot{y}_{n,n-1}} \\ 0 & 0 & 0 & p_{\ddot{y}y_{n,n-1}} & p_{\ddot{y}\dot{y}_{n,n-1}} & p_{\ddot{y}_{n,n-1}} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} \sigma_{x_m}^2 & 0 \\ 0 & \sigma_{y_m}^2 \end{bmatrix} \right)^{-1} \tag{9.22}$$

## The state update equation

The State Update Equation in matrix form is given by:

$$\hat{\boldsymbol{x}}_{n,n} = \hat{\boldsymbol{x}}_{n,n-1} + \boldsymbol{K}_n(\boldsymbol{z}_n - \boldsymbol{H}\hat{\boldsymbol{x}}_{n,n-1}) \tag{9.23}$$

Where:

$\hat{\boldsymbol{x}}_{n,n}$ is an estimated system state vector at time step $n$
$\hat{\boldsymbol{x}}_{n,n-1}$ is a predicted system state vector at time step $n-1$
$\boldsymbol{K}_n$ is a Kalman Gain
$\boldsymbol{z}_n$ is a measurement
$\boldsymbol{H}$ is an observation matrix

We have already defined all the building blocks of the state update equation.

## The covariance update equation

The Covariance Update Equation in a matrix form is given by:

$$\boldsymbol{P}_{n,n} = (\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})\,\boldsymbol{P}_{n,n-1}\,(\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})^T + \boldsymbol{K}_n\boldsymbol{R}_n\boldsymbol{K}_n^T \tag{9.24}$$

Where:

$\boldsymbol{P}_{n,n}$ is the covariance matrix of the current state estimation

$\boldsymbol{P}_{n,n-1}$ is the prior estimate covariance matrix of the current state (predicted at the previous state)

$\boldsymbol{K}_n$ is a Kalman Gain

$\boldsymbol{H}$ is the observation matrix

$\boldsymbol{R}_n$ is the measurement noise covariance matrix

$\boldsymbol{I}$ is an Identity Matrix (the $n \times n$ square matrix with ones on the main diagonal and zeros elsewhere)

### 9.1.2 The numerical example

Now we are ready to solve the numerical example. Let us assume a vehicle moving in a straight line in the $X$ direction with a constant velocity. After traveling 400 meters the vehicle turns right, with a turning radius of 300 meters. During the turning maneuver, the vehicle experiences acceleration due to the circular motion (angular acceleration).

The following chart depicts the vehicle movement.

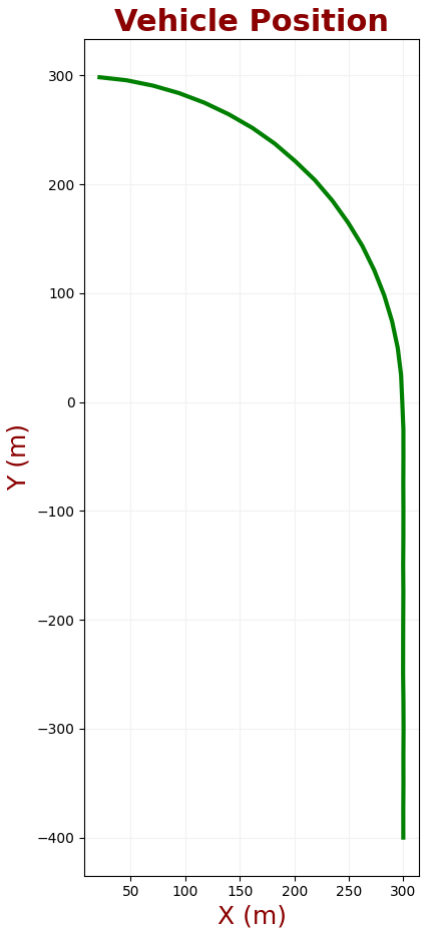


**Figure 9.2:** *Vehicle trajectory.*

- The measurements period: $\Delta t = 1s$
- The random acceleration standard deviation: $\sigma_a = 0.2\frac{m}{s^2}$
- The measurement error standard deviation: $\sigma_{x_m} = \sigma_{y_m} = 3m$
- The state transition matrix $\boldsymbol{F}$ would be:

$$\boldsymbol{F} = \begin{bmatrix} 1 & \Delta t & 0.5\Delta t^2 & 0 & 0 & 0 \\ 0 & 1 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta t & 0.5\Delta t^2 \\ 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0.5 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0.5 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

- The process noise matrix $\boldsymbol{Q}$ would be:

$$\boldsymbol{Q} = \begin{bmatrix} \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} & \frac{\Delta t^2}{2} & 0 & 0 & 0 \\ \frac{\Delta t^3}{2} & \Delta t^2 & \Delta t & 0 & 0 & 0 \\ \frac{\Delta t^2}{2} & \Delta t & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} & \frac{\Delta t^2}{2} \\ 0 & 0 & 0 & \frac{\Delta t^3}{2} & \Delta t^2 & \Delta t \\ 0 & 0 & 0 & \frac{\Delta t^2}{2} & \Delta t & 1 \end{bmatrix} \sigma_a^2 = \begin{bmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & 1 & 1 & 0 & 0 & 0 \\ \frac{1}{2} & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{4} & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & \frac{1}{2} & 1 & 1 \\ 0 & 0 & 0 & \frac{1}{2} & 1 & 1 \end{bmatrix} 0.2^2$$

- The measurement covariance $\boldsymbol{R}$ would be:

$$\boldsymbol{R}_n = \begin{bmatrix} \sigma_{x_m}^2 & 0 \\ 0 & \sigma_{y_m}^2 \end{bmatrix} = \begin{bmatrix} 9 & 0 \\ 0 & 9 \end{bmatrix}$$

The following table contains the set of 35 noisy measurements:

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| **x(m)** | 301.5 | 298.23 | 297.83 | 300.42 | 301.94 | 299.5 | 305.98 | 301.25 |
| **y(m)** | -401.46 | -375.44 | -346.15 | -320.2 | -300.08 | -274.12 | -253.45 | -226.4 |
| **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** | **17** |
| 299.73 | 299.2 | 298.62 | 301.84 | 299.6 | 295.3 | 299.3 | 301.95 | 296.3 |
| -200.65 | -171.62 | -152.11 | -125.19 | -93.4 | -74.79 | -49.12 | -28.73 | 2.99 |
| **18** | **19** | **20** | **21** | **22** | **23** | **24** | **25** | **26** |
| 295.11 | 295.12 | 289.9 | 283.51 | 276.42 | 264.22 | 250.25 | 236.66 | 217.47 |
| 25.65 | 49.86 | 72.87 | 96.34 | 120.4 | 144.69 | 168.06 | 184.99 | 205.11 |
| **27** | **28** | **29** | **30** | **31** | **32** | **33** | **34** | **35** |
| 199.75 | 179.7 | 160 | 140.92 | 113.53 | 93.68 | 69.71 | 45.93 | 20.87 |
| 221.82 | 238.3 | 253.02 | 267.19 | 270.71 | 285.86 | 288.48 | 292.9 | 298.77 |

**Table 9.1:** *Example 9 measurements.*

#### 9.1.2.1 Iteration Zero

##### Initialization

We don't know the vehicle location; we set the initial position, velocity, and acceleration to 0.

$$\hat{\boldsymbol{x}}_{0,0} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Since our initial state vector is a guess, we set a very high estimate uncertainty. The high estimate uncertainty results in a high Kalman Gain by giving a high weight to the measurement.

$$\boldsymbol{P}_{0,0} = \begin{bmatrix} 500 & 0 & 0 & 0 & 0 & 0 \\ 0 & 500 & 0 & 0 & 0 & 0 \\ 0 & 0 & 500 & 0 & 0 & 0 \\ 0 & 0 & 0 & 500 & 0 & 0 \\ 0 & 0 & 0 & 0 & 500 & 0 \\ 0 & 0 & 0 & 0 & 0 & 500 \end{bmatrix}$$

#### Prediction

$$\hat{\boldsymbol{x}}_{1,0} = \boldsymbol{F}\hat{\boldsymbol{x}}_{0,0} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\boldsymbol{P}_{1,0} = \boldsymbol{F}\boldsymbol{P}_{0,0}\boldsymbol{F}^T + \boldsymbol{Q} = \begin{bmatrix} 1125 & 750 & 250 & 0 & 0 & 0 \\ 750 & 1000 & 500 & 0 & 0 & 0 \\ 250 & 500 & 500 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1125 & 750 & 250 \\ 0 & 0 & 0 & 750 & 1000 & 500 \\ 0 & 0 & 0 & 250 & 500 & 500 \end{bmatrix}$$

### 9.1.2.2 First Iteration

#### Step 1 - Measure

The measurement values:

$$\boldsymbol{z}_1 = \begin{bmatrix} 301.5 \\ -401.46 \end{bmatrix}$$

#### Step 2 - Update

The Kalman Gain calculation:

$$\boldsymbol{K}_1 = \boldsymbol{P}_{1,0}\boldsymbol{H}^T\left(\boldsymbol{H}\boldsymbol{P}_{1,0}\boldsymbol{H}^T + \boldsymbol{R}\right)^{-1} = \begin{bmatrix} 0.9921 & 0 \\ 0.6614 & 0 \\ 0.2205 & 0 \\ 0 & 0.9921 \\ 0 & 0.6614 \\ 0 & 0.2205 \end{bmatrix}$$

As you can see, the Kalman Gain for a position is 0.9921, which means that the weight of the first measurement is significantly higher than the weight of the estimation. The weight of the estimation is negligible.

Estimate the current state:

$$\hat{\boldsymbol{x}}_{1,1} = \hat{\boldsymbol{x}}_{1,0} + \boldsymbol{K}_1(\boldsymbol{z}_1 - \boldsymbol{H}\hat{\boldsymbol{x}}_{1,0}) = \begin{bmatrix} 299.1 \\ 199.4 \\ 66.47 \\ -398.27 \\ -265.52 \\ -88.51 \end{bmatrix}$$

Update the current estimate uncertainty:

$$\boldsymbol{P}_{1,1} = (\boldsymbol{I} - \boldsymbol{K}_1\boldsymbol{H})\,\boldsymbol{P}_{1,0}\,(\boldsymbol{I} - \boldsymbol{K}_1\boldsymbol{H})^T + \boldsymbol{K}_1\boldsymbol{R}\boldsymbol{K}_1^T = \begin{bmatrix} 8.93 & 5.95 & 2 & 0 & 0 & 0 \\ 5.95 & 504 & 334.7 & 0 & 0 & 0 \\ 2 & 334.7 & 444.9 & 0 & 0 & 0 \\ 0 & 0 & 0 & 8.93 & 5.95 & 2 \\ 0 & 0 & 0 & 5.95 & 504 & 334.7 \\ 0 & 0 & 0 & 2 & 334.7 & 444.9 \end{bmatrix}$$

#### Step 3 - Predict

$$\hat{\boldsymbol{x}}_{2,1} = \boldsymbol{F}\hat{\boldsymbol{x}}_{1,1} = \begin{bmatrix} 531.75 \\ 265.88 \\ 66.47 \\ -708.05 \\ -354.03 \\ -88.51 \end{bmatrix}$$

$$\boldsymbol{P}_{2,1} = \boldsymbol{F}\boldsymbol{P}_{1,1}\boldsymbol{F}^T + \boldsymbol{Q} = \begin{bmatrix} 972 & 1236 & 559 & 0 & 0 & 0 \\ 1236 & 1618 & 780 & 0 & 0 & 0 \\ 559 & 780 & 445 & 0 & 0 & 0 \\ 0 & 0 & 0 & 972 & 1236 & 559 \\ 0 & 0 & 0 & 1236 & 1618 & 780 \\ 0 & 0 & 0 & 559 & 780 & 445 \end{bmatrix}$$

Our prediction uncertainty is still very high.

### 9.1.2.3 Second Iteration

#### Step 1 - Measure

The measurement values:

$$\boldsymbol{z}_2 = \begin{bmatrix} 298.23 \\ -375.44 \end{bmatrix}$$

### Step 2 - Update

The Kalman Gain calculation:

$$\boldsymbol{K}_2 = \boldsymbol{P}_{2,1}\boldsymbol{H}^T \left(\boldsymbol{H}\boldsymbol{P}_{2,1}\boldsymbol{H}^T + \boldsymbol{R}\right)^{-1} = \begin{bmatrix} 0.9908 & 0 \\ 1.26 & 0 \\ 0.57 & 0 \\ 0 & 0.9908 \\ 0 & 1.26 \\ 0 & 0.57 \end{bmatrix}$$

Estimate the current state:

$$\hat{\boldsymbol{x}}_{2,2} = \hat{\boldsymbol{x}}_{2,1} + \boldsymbol{K}_2(\boldsymbol{z}_2 - \boldsymbol{H}\hat{\boldsymbol{x}}_{2,1}) = \begin{bmatrix} 300.37 \\ -28.22 \\ -66.53 \\ -378.49 \\ 64.87 \\ 100.93 \end{bmatrix}$$

Update the current estimate uncertainty:

$$\boldsymbol{P}_{2,2} = (\boldsymbol{I} - \boldsymbol{K}_2\boldsymbol{H})\,\boldsymbol{P}_{2,1}\,(\boldsymbol{I} - \boldsymbol{K}_2\boldsymbol{H})^T + \boldsymbol{K}_2\boldsymbol{R}\boldsymbol{K}_2^T = \begin{bmatrix} 8.92 & 11.33 & 5.13 & 0 & 0 & 0 \\ 11.33 & 61.1 & 75.4 & 0 & 0 & 0 \\ 5.13 & 75.4 & 126.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 8.92 & 11.33 & 5.13 \\ 0 & 0 & 0 & 11.33 & 61.1 & 75.4 \\ 0 & 0 & 0 & 5.13 & 75.4 & 126.5 \end{bmatrix}$$

### Step 3 - Predict

$$\boldsymbol{P}_{3,2} = \boldsymbol{F}\boldsymbol{P}_{2,2}\boldsymbol{F}^T + \boldsymbol{Q} = \begin{bmatrix} 204.9 & 254 & 143.8 & 0 & 0 & 0 \\ 254 & 338.5 & 202 & 0 & 0 & 0 \\ 143.8 & 202 & 126.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 204.9 & 254 & 143.8 \\ 0 & 0 & 0 & 254 & 338.5 & 202 \\ 0 & 0 & 0 & 143.8 & 202 & 126.5 \end{bmatrix}$$

Our prediction uncertainty is still very high.

At this point, it would be reasonable to jump to the last Kalman Filter iteration.

#### 9.1.2.4 Thirty-Fifth Iteration

##### Step 1 - Measure

The measurement values:

$$\boldsymbol{z}_{35} = \begin{bmatrix} 20.87 \\ 298.77 \end{bmatrix}$$

##### Step 2 - Update

The Kalman Gain calculation:

$$\boldsymbol{K}_{35} = \boldsymbol{P}_{35,34}\boldsymbol{H}^T \left(\boldsymbol{H}\boldsymbol{P}_{35,34}\boldsymbol{H}^T + \boldsymbol{R}\right)^{-1} = \begin{bmatrix} 0.5556 & 0 \\ 0.2222 & 0 \\ 0.0444 & 0 \\ 0 & 0.5556 \\ 0 & 0.2222 \\ 0 & 0.0444 \end{bmatrix}$$

The Kalman Gain for the position has converged to 0.56, meaning that the measurement and estimation weights are almost equal.

Estimate the current state:

$$\hat{\boldsymbol{x}}_{35,35} = \hat{\boldsymbol{x}}_{35,34} + \boldsymbol{K}_{35}(\boldsymbol{z}_{35} - \boldsymbol{H}\hat{\boldsymbol{x}}_{35,34}) = \begin{bmatrix} 19.3 \\ -25.99 \\ -0.74 \\ 297.79 \\ 2.03 \\ -1.86 \end{bmatrix}$$

Update the current estimate uncertainty:

$$\boldsymbol{P}_{35,35} = (\boldsymbol{I} - \boldsymbol{K}_{35}\boldsymbol{H})\,\boldsymbol{P}_{35,34}\,(\boldsymbol{I} - \boldsymbol{K}_{35}\boldsymbol{H})^T + \boldsymbol{K}_{35}\boldsymbol{R}\boldsymbol{K}_{35}^T = \begin{bmatrix} 5 & 2 & 0.4 & 0 & 0 & 0 \\ 2 & 1.4 & 0.4 & 0 & 0 & 0 \\ 0.4 & 0.4 & 0.16 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 2 & 0.4 \\ 0 & 0 & 0 & 2 & 1.4 & 0.4 \\ 0 & 0 & 0 & 0.4 & 0.4 & 0.16 \end{bmatrix}$$

At this point, the position variance $p_x = p_y = 5$, which means that the standard deviation of the estimate is $\sqrt{5}m$.

### Step 3 - Predict

$$\hat{\boldsymbol{x}}_{36,35} = \boldsymbol{F}\hat{\boldsymbol{x}}_{35,35} = \begin{bmatrix} -7.05 \\ -26.73 \\ -0.74 \\ 298.89 \\ 0.17 \\ -1.87 \end{bmatrix}$$

$$\boldsymbol{P}_{36,35} = \boldsymbol{F}\boldsymbol{P}_{35,35}\boldsymbol{F}^T + \boldsymbol{Q} = \begin{bmatrix} 11.25 & 4.5 & 0.9 & 0 & 0 & 0 \\ 4.5 & 2.4 & 0.6 & 0 & 0 & 0 \\ 0.9 & 0.6 & 0.2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 11.25 & 4.5 & 0.9 \\ 0 & 0 & 0 & 4.5 & 2.4 & 0.6 \\ 0 & 0 & 0 & 0.9 & 0.6 & 0.2 \end{bmatrix}$$

## 9.1.3 Example analysis

The following chart demonstrates the KF location and velocity estimation performance.

The chart on the left compares the true, measured, and estimated values of the vehicle position. Two charts on the right compare the true, measured, and estimated values of $x$-axis velocity and $y$-axis velocity.

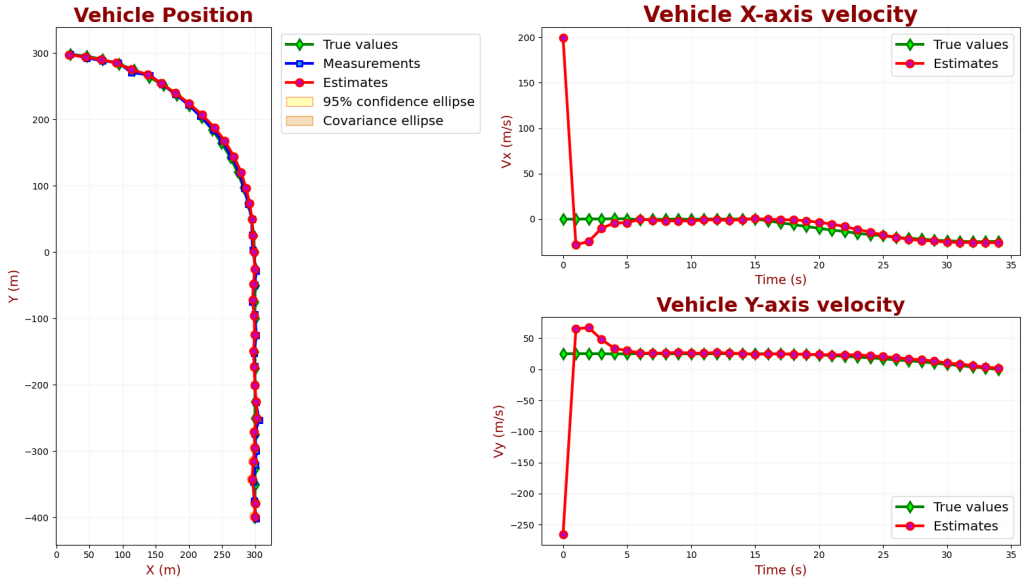


**Figure 9.3:** *Example 9: true value, measured values and estimates.*

As you can see, the Kalman Filter succeeds in tracking the vehicle.

Let us zoom the linear part of the vehicle motion and the turning maneuver part.

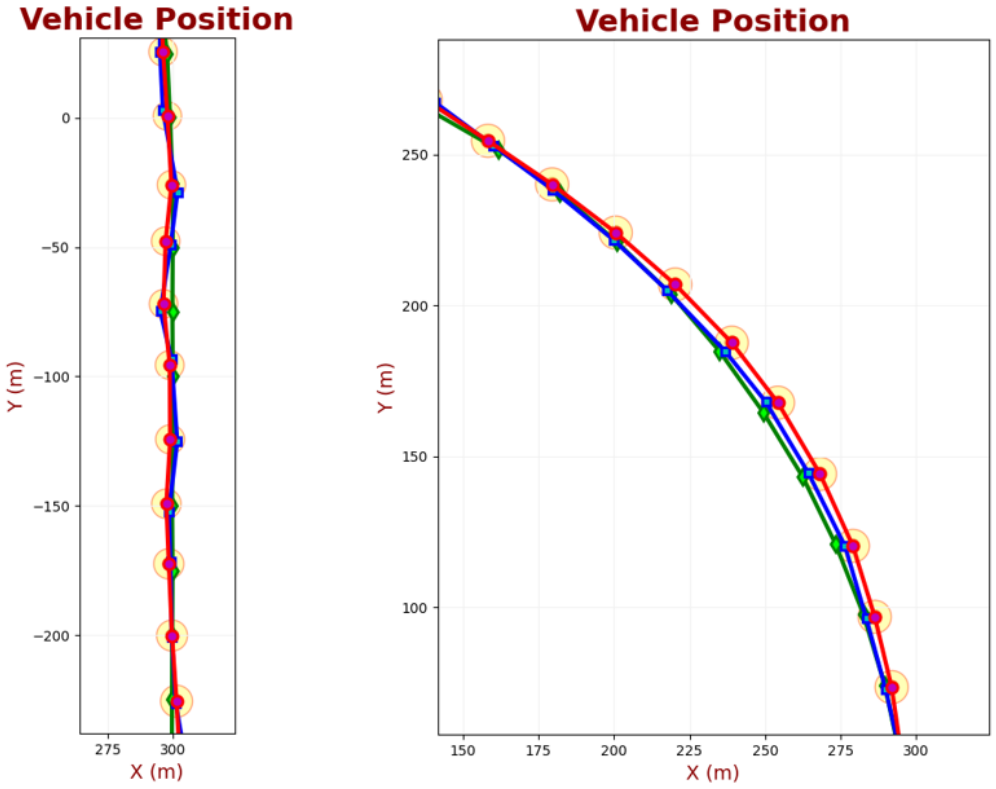


**Figure 9.4:** *Example 9: true value, measured values and estimates - zoom.*

The circles on the plot represent the 95% confidence ellipse. Since the $x$ and $y$ axes' measurement errors are equal, the confidence ellipse is a circle.

While the vehicle travels along a straight line, the acceleration is constant and equal to zero. However, during the turn maneuver, the vehicle experiences acceleration due to the circular motion - angular acceleration.

Recall from an introductory physics that angular acceleration is $\alpha = \frac{\Delta V}{R\Delta t}$ , where $\Delta t$ is the time interval, $\Delta V$ is the velocity difference within the time interval, and $R$ is the circle radius.

Our Kalman Filter is designed for a constant acceleration model. Nevertheless, it succeeds in tracking maneuvering vehicle due to a properly chosen $\sigma_a^2$ parameter.

I would like to encourage the readers to implement this example in software and see how different values of $\sigma_a^2$ or $\boldsymbol{R}$ influence the actual Kalman Filter accuracy, Kalman Gain convergence, and estimation uncertainty.

## 9.2 Example 10 – rocket altitude estimation

In this example, we estimate the altitude of a rocket. The rocket is equipped with an onboard altimeter that provides altitude measurements. In addition to an altimeter, the rocket is equipped with an accelerometer that measures the rocket's acceleration.

The accelerometer serves as a **control input** to the Kalman Filter.

We assume constant acceleration dynamics.

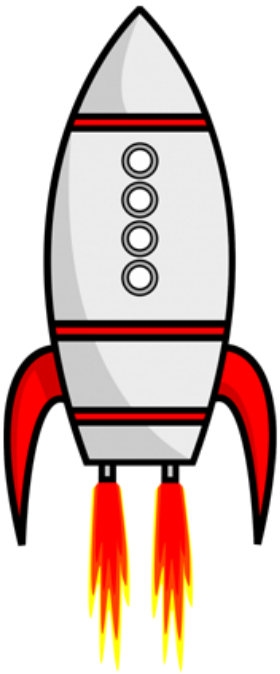

**Figure 9.5:** *Example 10: rocket altitude estimation.*

Accelerometers don't sense gravity. An accelerometer at rest on a table measures $1g$ upwards, while an accelerometer in free fall measures zero acceleration. Thus, we need to subtract the gravitational acceleration constant $g$ from each accelerometer measurement.

The accelerometer measurement at time step $n$ is:

$$a_n = \ddot{x} - g + \epsilon \tag{9.25}$$

Where:

$\ddot{x}$ is the actual acceleration of the object (the second derivative of the object position)

$g$ is the gravitational acceleration constant; $g = -9.8\frac{m}{s^2}$

$\epsilon$ is the accelerometer measurement error

### 9.2.1 Kalman Filter equations

#### The state extrapolation equation

The general form of the state extrapolation equation in matrix notation is:

$$\hat{\boldsymbol{x}}_{n+1,n} = \boldsymbol{F}\hat{\boldsymbol{x}}_{n,n} + \boldsymbol{G}\boldsymbol{u}_n + \boldsymbol{w}_n \tag{9.26}$$

Where:

$\hat{\boldsymbol{x}}_{n+1,n}$ is a predicted system state vector at time step $n+1$

$\hat{\boldsymbol{x}}_{n,n}$ is an estimated system state vector at time step $n$

$\boldsymbol{u}_n$ is a **control variable** or **input variable** - a measurable (deterministic) input to the system

$\boldsymbol{w}_n$ is a **process noise** or disturbance - an unmeasurable input that affects the state

$\boldsymbol{F}$ is a **state transition matrix**

$\boldsymbol{G}$ is a **control matrix** or **input transition matrix** (mapping control to state variables)

In this example, we have a control variable $\boldsymbol{u}$, which is based on the accelerometer measurement.

The system state $\boldsymbol{x}_n$ is defined by:

$$\boldsymbol{x}_n = \begin{bmatrix} x_n \\ \dot{x}_n \end{bmatrix} \tag{9.27}$$

Where:

$x_n$ is the rocket altitude at time $n$

$\dot{x}_n$ is the rocket velocity at time $n$

We can express the state extrapolation equation as follows:

$$\begin{bmatrix} \hat{x}_{n+1,n} \\ \hat{\dot{x}}_{n+1,n} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{x}_{n,n} \\ \hat{\dot{x}}_{n,n} \end{bmatrix} + \begin{bmatrix} 0.5\Delta t^2 \\ \Delta t \end{bmatrix} (a_n + g) \tag{9.28}$$

In the above equation:

$$\boldsymbol{F} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \tag{9.29}$$

$$\boldsymbol{G} = \begin{bmatrix} 0.5\Delta t^2 \\ \Delta t \end{bmatrix} \tag{9.30}$$

$$\boldsymbol{u}_n = (a_n + g) \tag{9.31}$$

## The covariance extrapolation equation

The general form of the Covariance Extrapolation Equation is:

$$\boldsymbol{P}_{n+1,n} = \boldsymbol{F}\boldsymbol{P}_{n,n}\boldsymbol{F}^T + \boldsymbol{Q} \tag{9.32}$$

Where:

| | |
|---|---|
| $\boldsymbol{P}_{n,n}$ | is the covariance matrix of the current state estimation |
| $\boldsymbol{P}_{n+1,n}$ | is the covariance matrix of the next state estimation (prediction) |
| $\boldsymbol{F}$ | is the state transition matrix that we derived in Appendix C ("Modeling linear dynamic systems") |
| $\boldsymbol{Q}$ | is the process noise matrix |

The estimate covariance in matrix form is:

$$\boldsymbol{P} = \begin{bmatrix} p_x & p_{x\dot{x}} \\ p_{\dot{x}x} & p_{\dot{x}} \end{bmatrix} \tag{9.33}$$

The elements of the main diagonal of the matrix are the variances of the estimation:

- $p_x$ is the variance of the altitude estimation
- $p_{\dot{x}}$ is the variance of the velocity estimation
- The off-diagonal entries are covariances

We have already derived the state transition matrix $\boldsymbol{F}$. Now we shall derive the process noise $\boldsymbol{Q}$ matrix.

## The process noise matrix

We assume a discrete noise model - the noise is different at each time sample but is constant between time samples.

The process noise matrix for a constant acceleration model looks like this:

$$\boldsymbol{Q} = \begin{bmatrix} \sigma_x^2 & \sigma_{x\dot{x}}^2 \\ \sigma_{\dot{x}x}^2 & \sigma_{\dot{x}}^2 \end{bmatrix} \tag{9.34}$$

We've already derived the Q matrix for the constant acceleration model (subsection 8.2.2). The Q matrix for our example is:

$$\boldsymbol{Q} = \begin{bmatrix} \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} \\ \frac{\Delta t^3}{2} & \Delta t^2 \end{bmatrix} \epsilon^2 \tag{9.35}$$

Where:

$\Delta t$ is the time between successive measurements

$\epsilon^2$ is the random variance in accelerometer measurement

In the previous example, we used the system's random variance in acceleration $\sigma_a^2$ as a multiplier of the process noise matrix. But here, we have an accelerometer that measures the system's random acceleration. The accelerometer error $v$ is much lower than the system's random acceleration; therefore, we use $\epsilon^2$ as a multiplier of the process noise matrix.

It makes our estimation uncertainty much lower!

Now we can write down the covariance extrapolation equation for our example:

$$\boldsymbol{P}_{n+1,n} = \boldsymbol{F}\boldsymbol{P}_{n,n}\boldsymbol{F}^T + \boldsymbol{Q} \tag{9.36}$$

$$\begin{bmatrix} p_{x_{n+1,n}} & p_{x\dot{x}_{n+1,n}} \\ p_{\dot{x}x_{n+1,n}} & p_{\dot{x}_{n+1,n}} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} p_{x_{n,n}} & p_{x\dot{x}_{n,n}} \\ p_{\dot{x}x_{n,n}} & p_{\dot{x}_{n,n}} \end{bmatrix} \times \begin{bmatrix} 1 & 0 \\ \Delta t & 1 \end{bmatrix} + \begin{bmatrix} \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} \\ \frac{\Delta t^3}{2} & \Delta t^2 \end{bmatrix} \epsilon^2 \tag{9.37}$$

R According to the "Constructing the process noise matrix" (subsection 8.2.2), the size of the $\boldsymbol{Q}$ matrix for constant velocity model should be $2 \times 2$, and the size of the $\boldsymbol{Q}$ matrix for constant acceleration model should be $3 \times 3$.

In this example, the acceleration is handled by control input (therefore, it is not part of $\boldsymbol{F}$ matrix, and the $\boldsymbol{Q}$ matrix is $2 \times 2$). Without an external control input, the process matrix would be $3 \times 3$.

## The measurement equation

The generalized measurement equation in matrix form is given by:

$$\boldsymbol{z}_n = \boldsymbol{H}\boldsymbol{x}_n + \boldsymbol{v}_n \tag{9.38}$$

Where:

$\boldsymbol{z}_n$ is a measurement vector

$\boldsymbol{x}_n$ is a true system state (hidden state)

$\boldsymbol{v}_n$ is a random noise vector

$\boldsymbol{H}$ is an **observation matrix**

The measurement provides only the altitude of the rocket:: $\boldsymbol{z}_n = [x_{n,measured}]$

$$\boldsymbol{z}_n = \boldsymbol{H}\boldsymbol{x}_n \tag{9.39}$$

$$[x_{n,measured}] = \boldsymbol{H} \begin{bmatrix} x_n \\ \dot{x}_n \end{bmatrix} \tag{9.40}$$

The dimension of $\boldsymbol{z}_n$ is $1 \times 1$ and the dimension of $\boldsymbol{x}_n$ is $2 \times 1$, so the dimension of the observation matrix $\boldsymbol{H}$ is $1 \times 2$.

$$\boldsymbol{H} = \begin{bmatrix} 1 & 0 \end{bmatrix} \tag{9.41}$$

## The measurement uncertainty

The measurement covariance matrix is:

$$\boldsymbol{R}_n = \begin{bmatrix} \sigma_{x_m}^2 \end{bmatrix} \tag{9.42}$$

The subscript $m$ means the "measurement".

For the sake of the example simplicity, we assume a constant measurement uncertainty:

$$\boldsymbol{R}_1 = \boldsymbol{R}_2 ... \boldsymbol{R}_{n-1} = \boldsymbol{R}_n = \boldsymbol{R} \tag{9.43}$$

## The Kalman Gain

The Kalman Gain in matrix notation is given by:

$$\boldsymbol{K}_n = \boldsymbol{P}_{n,n-1}\boldsymbol{H}^T \left(\boldsymbol{H}\boldsymbol{P}_{n,n-1}\boldsymbol{H}^T + \boldsymbol{R}_n\right)^{-1} \tag{9.44}$$

Where:

| | |
|---|---|
| $\boldsymbol{K}_n$ | is the Kalman Gain |
| $\boldsymbol{P}_{n,n-1}$ | is the prior estimate uncertainty (covariance) matrix of the current state (predicted at the previous step) |
| $\boldsymbol{H}$ | is the observation matrix |
| $\boldsymbol{R}_n$ | is the Measurement Uncertainty (measurement noise covariance matrix) |

We have already derived all the building blocks of the Kalman Gain:

$$\begin{bmatrix} K_{1,1_n} \\ K_{2,1_n} \end{bmatrix} = \begin{bmatrix} p_{x_{n,n-1}} & p_{x\dot{x}_{n,n-1}} \\ p_{\dot{x}x_{n,n-1}} & p_{\dot{x}_{n,n-1}} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \times \left( \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} p_{x_{n,n-1}} & p_{x\dot{x}_{n,n-1}} \\ p_{\dot{x}x_{n,n-1}} & p_{\dot{x}_{n,n-1}} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} \sigma^2_{x_m} \end{bmatrix} \right)^{-1} \tag{9.45}$$

## The state update equation

The State Update Equation in matrix form is given by:

$$\hat{\boldsymbol{x}}_{n,n} = \hat{\boldsymbol{x}}_{n,n-1} + \boldsymbol{K}_n(\boldsymbol{z}_n - \boldsymbol{H}\hat{\boldsymbol{x}}_{n,n-1}) \tag{9.46}$$

Where:

| | |
|---|---|
| $\hat{\boldsymbol{x}}_{n,n}$ | is an estimated system state vector at time step $n$ |
| $\hat{\boldsymbol{x}}_{n,n-1}$ | is a predicted system state vector at time step $n-1$ |
| $\boldsymbol{K}_n$ | is a Kalman Gain |
| $\boldsymbol{z}_n$ | is a measurement |
| $\boldsymbol{H}$ | is an observation matrix |

We have already defined all the building blocks of the state update equation.

### The covariance update equation

The Covariance Update Equation in a matrix form is given by:

$$\boldsymbol{P}_{n,n} = (\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})\,\boldsymbol{P}_{n,n-1}\,(\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})^T + \boldsymbol{K}_n\boldsymbol{R}_n\boldsymbol{K}_n^T \tag{9.47}$$

Where:

| | |
|---|---|
| $\boldsymbol{P}_{n,n}$ | is the uncertainty (covariance) matrix of the current state estimation |
| $\boldsymbol{P}_{n,n-1}$ | is the prior estimate uncertainty (covariance) matrix of the current state (predicted at the previous state) |
| $\boldsymbol{K}_n$ | is a Kalman Gain |
| $\boldsymbol{H}$ | is the observation matrix |
| $\boldsymbol{R}_n$ | is the Measurement Uncertainty (measurement noise covariance matrix) |
| $\boldsymbol{I}$ | is an Identity Matrix (the $n \times n$ square matrix with ones on the main diagonal and zeros elsewhere) |

## 9.2.2 The numerical example

Let us assume a vertically boosting rocket with constant acceleration. The rocket is equipped with an altimeter that provides altitude measurements and an accelerometer that serves as a control input.

- The measurements period: $\Delta t = 0.25s$
- The rocket acceleration: $\ddot{x} = 30\frac{m}{s^2}$
- The altimeter measurement error standard deviation: $\sigma_{x_m} = 20m$
- The accelerometer measurement error standard deviation: $\epsilon = 0.1\frac{m}{s^2}$
- The state transition matrix $\boldsymbol{F}$ would be:

$$\boldsymbol{F} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0.25 \\ 0 & 1 \end{bmatrix}$$

- The control matrix $\boldsymbol{G}$ would be:

$$\boldsymbol{G} = \begin{bmatrix} 0.5\Delta t^2 \\ \Delta t \end{bmatrix} = \begin{bmatrix} 0.0313 \\ 0.25 \end{bmatrix}$$

- The process noise matrix $\boldsymbol{Q}$ would be:

$$\boldsymbol{Q} = \begin{bmatrix} \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} \\ \frac{\Delta t^3}{2} & \Delta t^2 \end{bmatrix} \sigma_a^2 = \begin{bmatrix} \frac{0.25^4}{4} & \frac{0.25^3}{2} \\ \frac{0.25^3}{2} & 0.25^2 \end{bmatrix} 0.1^2$$

- The measurement uncertainty $\boldsymbol{R}$ would be:

$$\boldsymbol{R}_n = \boldsymbol{R} = \begin{bmatrix} \sigma_{x_m}^2 \end{bmatrix} = 400$$

The following table contains the set of 30 noisy measurements of the altitude $h_n$ and acceleration $a_n$:

| | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
|---|---|---|---|---|---|---|---|---|
| $\boldsymbol{h}_n(m)$ | 6.43 | 1.3 | 39.43 | 45.89 | 41.44 | 48.7 | 78.06 | 80.08 |
| $\boldsymbol{a}_n(m/s^2)$ | 39.81 | 39.67 | 39.81 | 39.84 | 40.05 | 39.85 | 39.78 | 39.65 |
| **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** | **17** |
| 61.77 | 75.15 | 110.39 | 127.83 | 158.75 | 156.55 | 213.32 | 229.82 | 262.8 |
| 39.67 | 39.78 | 39.59 | 39.87 | 39.85 | 39.59 | 39.84 | 39.9 | 39.63 |
| **18** | **19** | **20** | **21** | **22** | **23** | **24** | **25** | **26** |
| 297.57 | 335.69 | 367.92 | 377.19 | 411.18 | 460.7 | 468.39 | 553.9 | 583.97 |
| 39.59 | 39.76 | 39.79 | 39.73 | 39.93 | 39.83 | 39.85 | 39.94 | 39.86 |
| **27** | **28** | **29** | **30** | | | | | |
| 655.15 | 723.09 | 736.85 | 787.22 | | | | | |
| 39.76 | 39.86 | 39.74 | 39.94 | | | | | |

**Table 9.2:** *Example 10 measurements.*

#### 9.2.2.1 Iteration Zero

##### Initialization

We don't know the rocket's location; we set the initial position and velocity to 0.

$$\hat{\boldsymbol{x}}_{0,0} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

We also don't know the rocket's acceleration, but we can assume it's greater than zero. Let's assume:

$$\boldsymbol{u}_0 = 0$$

Since our initial state vector is a guess, we set a very high estimate uncertainty. The high estimate uncertainty results in high Kalman Gain, giving a high weight to the

measurement.

$$\boldsymbol{P}_{0,0} = \begin{bmatrix} 500 & 0 \\ 0 & 500 \end{bmatrix}$$

#### Prediction

Now we can predict the next state based on the initialization values.

$$\hat{\boldsymbol{x}}_{1,0} = \boldsymbol{F}\hat{\boldsymbol{x}}_{0,0} + \boldsymbol{G}\boldsymbol{u}_0 = \begin{bmatrix} 1 & 0.25 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0.0313 \\ 0.25 \end{bmatrix} 0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\boldsymbol{P}_{1,0} = \boldsymbol{F}\boldsymbol{P}_{0,0}\boldsymbol{F}^T + \boldsymbol{Q} = \begin{bmatrix} 1 & 0.25 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 500 & 0 \\ 0 & 500 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0.25 & 1 \end{bmatrix} + \begin{bmatrix} \frac{0.25^4}{4} & \frac{0.25^3}{2} \\ \frac{0.25^3}{2} & 0.25^2 \end{bmatrix} 0.1^2$$

$$= \begin{bmatrix} 531.25 & 125 \\ 125 & 500 \end{bmatrix}$$

### 9.2.2.2 First Iteration

#### Step 1 - Measure

The measurement values:

$\boldsymbol{z}_1 = 6.43$

$\boldsymbol{u}_1 = 39.81$

#### Step 2 - Update

The Kalman Gain calculation:

$$\boldsymbol{K}_1 = \boldsymbol{P}_{1,0}\boldsymbol{H}^T \left(\boldsymbol{H}\boldsymbol{P}_{1,0}\boldsymbol{H}^T + \boldsymbol{R}\right)^{-1} = \begin{bmatrix} 0.57 \\ 0.13 \end{bmatrix}$$

Estimate the current state:

$$\hat{\boldsymbol{x}}_{1,1} = \hat{\boldsymbol{x}}_{1,0} + \boldsymbol{K}_1(\boldsymbol{z}_1 - \boldsymbol{H}\hat{\boldsymbol{x}}_{1,0}) = \begin{bmatrix} 3.67 \\ 0.86 \end{bmatrix}$$

Update the current estimate uncertainty:

$$\boldsymbol{P}_{1,1} = (\boldsymbol{I} - \boldsymbol{K}_1\boldsymbol{H})\,\boldsymbol{P}_{1,0}\,(\boldsymbol{I} - \boldsymbol{K}_1\boldsymbol{H})^T + \boldsymbol{K}_1\boldsymbol{R}\boldsymbol{K}_1^T = \begin{bmatrix} 228.2 & 53.7 \\ 53.7 & 483.2 \end{bmatrix}$$

#### Step 3 - Predict

$$\hat{\boldsymbol{x}}_{2,1} = \boldsymbol{F}\hat{\boldsymbol{x}}_{1,1} + \boldsymbol{G}\boldsymbol{u}_1 = \begin{bmatrix} 1 & 0.25 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 3.67 \\ 0.86 \end{bmatrix} + \begin{bmatrix} 0.0313 \\ 0.25 \end{bmatrix} (39.81 + (-9.8)) = \begin{bmatrix} 4.82 \\ 8.36 \end{bmatrix}$$

$$\boldsymbol{P}_{2,1} = \boldsymbol{F}\boldsymbol{P}_{1,1}\boldsymbol{F}^T + \boldsymbol{Q} = \begin{bmatrix} 285.2 & 174.5 \\ 174.5 & 483.2 \end{bmatrix}$$

Our prediction uncertainty is still very high.

#### 9.2.2.3 Second Iteration

##### Step 1 - Measure

The measurement values:
$\boldsymbol{z}_2 = 1.3$
$\boldsymbol{u}_2 = 39.67$

##### Step 2 - Update

The Kalman Gain calculation:

$$\boldsymbol{K}_2 = \boldsymbol{P}_{2,1}\boldsymbol{H}^T \left(\boldsymbol{H}\boldsymbol{P}_{2,1}\boldsymbol{H}^T + \boldsymbol{R}\right)^{-1} = \begin{bmatrix} 0.42 \\ 0.26 \end{bmatrix}$$

Estimate the current state:

$$\hat{\boldsymbol{x}}_{2,2} = \hat{\boldsymbol{x}}_{2,1} + \boldsymbol{K}_2(\boldsymbol{z}_2 - \boldsymbol{H}\hat{\boldsymbol{x}}_{2,1}) = \begin{bmatrix} 3.36 \\ 7.47 \end{bmatrix}$$

Update the current estimate uncertainty:

$$\boldsymbol{P}_{2,2} = (\boldsymbol{I} - \boldsymbol{K}_2\boldsymbol{H})\,\boldsymbol{P}_{2,1}\,(\boldsymbol{I} - \boldsymbol{K}_2\boldsymbol{H})^T + \boldsymbol{K}_2\boldsymbol{R}\boldsymbol{K}_2^T = \begin{bmatrix} 166.5 & 101.9 \\ 101.9 & 438.8 \end{bmatrix}$$

##### Step 3 - Predict

$$\hat{\boldsymbol{x}}_{3,2} = \boldsymbol{F}\hat{\boldsymbol{x}}_{2,2} + \boldsymbol{G}\boldsymbol{u}_2 = \begin{bmatrix} 6.16 \\ 14.93 \end{bmatrix}$$

$$\boldsymbol{P}_{3,2} = \boldsymbol{F}\boldsymbol{P}_{2,2}\boldsymbol{F}^T + \boldsymbol{Q} = \begin{bmatrix} 244.9 & 211.6 \\ 211.6 & 438.8 \end{bmatrix}$$

At this point, it would be reasonable to jump to the last Kalman Filter iteration.

#### 9.2.2.4 Thirtieth Iteration

##### Step 1 - Measure

The measurement values:
$\boldsymbol{z}_{30} = 787.22$
$\boldsymbol{u}_{30} = 39.94$

##### Step 2 - Update

The Kalman Gain calculation:

$$\boldsymbol{K}_{30} = \boldsymbol{P}_{30,29}\boldsymbol{H}^T \left(\boldsymbol{H}\boldsymbol{P}_{30,29}\boldsymbol{H}^T + \boldsymbol{R}\right)^{-1} = \begin{bmatrix} 0.12 \\ 0.02 \end{bmatrix}$$

The Kalman Gain for altitude converged to 0.12, which means that the estimation weight is much higher than the measurement weight.

Estimate the current state:

$$\hat{\boldsymbol{x}}_{30,30} = \hat{\boldsymbol{x}}_{30,29} + \boldsymbol{K}_{30}(\boldsymbol{z}_{30} - \boldsymbol{H}\hat{\boldsymbol{x}}_{30,29}) = \begin{bmatrix} 797.07 \\ 215.7 \end{bmatrix}$$

Update the current estimate uncertainty:

$$\boldsymbol{P}_{30,30} = (\boldsymbol{I} - \boldsymbol{K}_{30}\boldsymbol{H})\,\boldsymbol{P}_{30,29}\,(\boldsymbol{I} - \boldsymbol{K}_{35}\boldsymbol{H})^T + \boldsymbol{K}_{30}\boldsymbol{R}\boldsymbol{K}_{30}^T = \begin{bmatrix} 49.3 & 9.7 \\ 9.7 & 2.6 \end{bmatrix}$$

At this point, the altitude variance $p_x = 49.3$, which means that the standard deviation of the estimate is $\sqrt{49.3}m = 7.02m$ (remember that the standard deviation of the measurement is $20m$).

### Step 3 - Predict

$$\hat{\boldsymbol{x}}_{31,30} = \boldsymbol{F}\hat{\boldsymbol{x}}_{30,30} + \boldsymbol{G}\boldsymbol{u}_{30} = \begin{bmatrix} 851.9 \\ 223.2 \end{bmatrix}$$

$$\boldsymbol{P}_{31,30} = \boldsymbol{F}\boldsymbol{P}_{30,30}\boldsymbol{F}^T + \boldsymbol{Q} = \begin{bmatrix} 54.3 & 10.4 \\ 10.4 & 2.6 \end{bmatrix}$$

### 9.2.3 Example analysis

The following chart compares the true, measured, and estimated values of the rocket altitude.

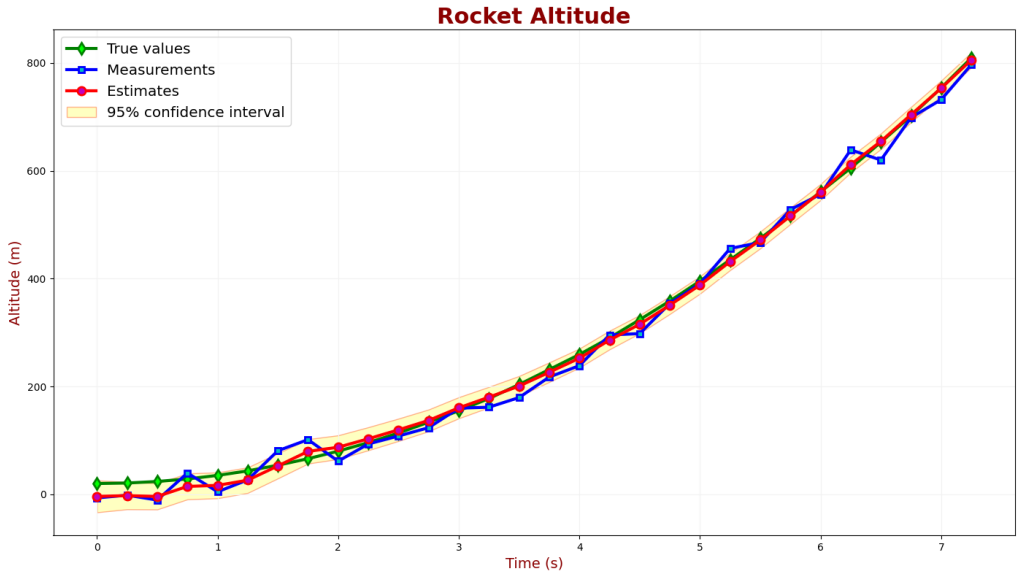


**Figure 9.6:** *Example 10: true value, measured values and estimates of the rocket altitude.*

We can see a good KF tracking performance and convergence.

The following chart compares the true value, measured values, and estimates of the rocket velocity. The confidence interval is 95%. (You can find the guidelines for a confidence interval calculation in Appendix B).

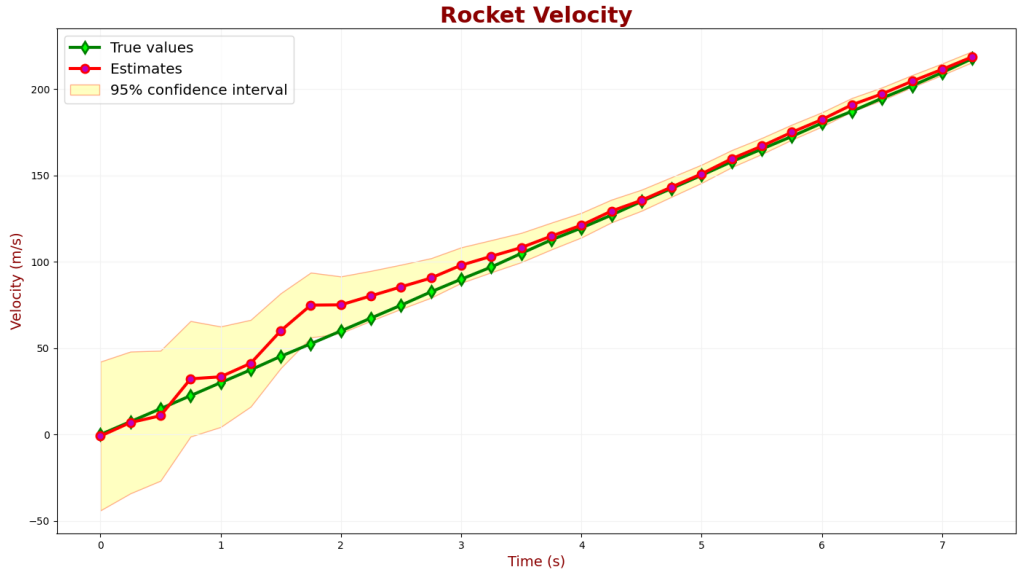


**Figure 9.7:** *Example 10: true value, measured values and estimates of the rocket velocity.*

It takes about 2.5 seconds to converge the estimates to the true rocket velocity.

In the beginning, the estimated altitude is influenced by measurements, and it is not aligned well with the true rocket altitude since the measurements are very noisy.

But as the KF converges, the noisy measurement has less influence, and the estimated altitude is well aligned with the true altitude.

In this example, we don't have any maneuvers that cause acceleration changes, but if we had, the control input (accelerometer) would update the state extrapolation equation.

# III Non-linear Kalman Filters

# 10. Foreword

Once you have mastered the multi-dimensional (or multivariate) Kalman Filter, you are ready to tackle the Non-linear Kalman Filters.

The Kalman Filter solves the estimation problem for linear systems. However, most real-life systems are non-linear.

For non-linear systems handling, the **Linear Approximation** techniques shall be applied. This part describes two common modifications of the Kalman Filter that perform Linear Approximation:

- Extended Kalman Filter (EKF)
- Unscented Kalman Filter (UKF) or Sigma-point Kalman Filter (SPKF)

The EKF performs **analytic linearization** of the model at each point in time. EKF is the most common non-linear Kalman Filter.

The UKF performs **statistical linearization** of the model at each point in time. The UKF is considered to be superior to the EKF.

While the standard Linear Kalman Filter (LKF) is an **optimal filter** since we minimize the estimate uncertainty (see subsection 8.7.1 - "Kalman Gain Equation Derivation"), all Kalman Filter modifications for non-linear systems are **sub-optimal** since we use **approximated models**.

This chapter describes the EKF and the UKF methods. We discuss the advantages and disadvantages of each method. Furthermore, each method is exemplified by numerical examples.

# 11. Essential background III

The non-linear Kalman Filter chapter requires prior knowledge of the following topics:

- Derivatives - required for mastering Extended Kalman Filter
- Matrix square root - required for mastering Unscented Kalman Filter

The author assumes that the readers are familiar with derivatives from an introductory Calculus course. Thus, this chapter deals with a matrix square root.

## 11.1 The square root of a matrix

A matrix $\boldsymbol{B}$ is a square root of a matrix $\boldsymbol{A}$ if:

$$\boldsymbol{A} = \boldsymbol{B}\boldsymbol{B} \tag{11.1}$$

The equation above can have several possible solutions, and there are different computation methods for finding the square root of a matrix.

The Uncentenced Kalman Filter employs an Unscented Transform that requires a square root computation of a covariance matrix.

Luckily, covariance matrices are positive and semi-definite. Thus we can use **Cholesky decomposition** (or **Cholesky factorization**) for the computation of a covariance matrix square root.

## 11.2 Cholesky decomposition

The Cholesky decomposition is a decomposition of a positive definite matrix into a product of a lower triangular matrix and its transpose.

If a matrix $\boldsymbol{A}$ is positive definite:

$$\boldsymbol{A} = \boldsymbol{L}^T\boldsymbol{L} \tag{11.2}$$

The lower triangular matrix is a square matrix where all the values above the diagonal

are zero:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} \end{bmatrix} \begin{bmatrix} l_{11} & l_{12} & l_{13} & l_{14} \\ 0 & l_{22} & l_{23} & l_{24} \\ 0 & 0 & l_{33} & l_{34} \\ 0 & 0 & 0 & l_{44} \end{bmatrix} \tag{11.3}$$

If matrix $\boldsymbol{A}$ is positive semi-definite, then the diagonal entries of $\boldsymbol{L}$ are allowed to be zero.

Cholesky decomposition algorithm:

- Diagonal elements of $\boldsymbol{L}$:

$$l_{vv} = \sqrt{a_{vv} - \sum_{u<v} l_{vu}^2} \tag{11.4}$$

- Off-diagonal elements of $\boldsymbol{L}$:

$$l_{tv} = \frac{1}{l_{vv}} \left( a_{tv} - \sum_{u<v} l_{tu} l_{vu} \right) \tag{11.5}$$

First, find the elements of the first row of $\boldsymbol{L}$, then find the elements of the second row of $\boldsymbol{L}$, and then find the elements of the third row of $\boldsymbol{L}$. Continue the process until you reach the final row.

Example:

$$\boldsymbol{A} = \begin{bmatrix} 9 & -6 & 3 & 24 \\ -6 & 20 & -22 & -4 \\ 3 & -22 & 30 & -9 \\ 24 & -4 & -9 & 99 \end{bmatrix}$$

Given the matrix $\boldsymbol{A}$, find the square root of $\boldsymbol{A}$ using Cholesky decomposition.

$$l_{11} = \sqrt{a_{11}} = 3$$

$$l_{21} = \frac{a_{21}}{l_{11}} = \frac{-6}{3} = -2$$

$$l_{22} = \sqrt{a_{22} - l_{11}^2} = \sqrt{20 - (-2)^2} = 4$$

$$l_{31} = \frac{a_{31}}{l_{11}} = \frac{3}{3} = 1$$

$$l_{32} = \frac{a_{32} - l_{31} \times l_{21}}{l_{22}} = \frac{-22 - 1 \times (-2)}{4} = -5$$

$$l_{33} = \sqrt{a_{33} - l_{31}^2 - l_{32}^2} = \sqrt{30 - 1^2 - (-5)^2} = 2$$

$$l_{41} = \frac{a_{41}}{l_{11}} = \frac{24}{3} = 8$$

$$l_{42} = \frac{a_{42} - l_{41} \times l_{21}}{l_{22}} = \frac{-4 - 8 \times (-2)}{4} = 3$$

$$l_{43} = \frac{a_{43} - l_{41} \times l_{31} - l_{42} \times l_{32}}{l_{33}} = \frac{-9 - 8 \times 1 - 3 \times (-5)}{2} = -1$$

$$l_{44} = \sqrt{a_{44} - l_{41}^2 - l_{42}^2 - l_{43}^2} = \sqrt{99 - 8^2 - 3^2 - (-1)^2} = 5$$

$$\boldsymbol{L} = \begin{bmatrix} 3 & 0 & 0 & 0 \\ -2 & 4 & 0 & 0 \\ 1 & -5 & 2 & 0 \\ 8 & 3 & -1 & 5 \end{bmatrix}$$

Some computer packages have built-in Cholesky decomposition functions.

**Python example:**

```
import numpy as np

A = np.array([[9, -6, 3, 24], [-6, 20, -22, -4], [3, -22, 30, -9],
    [24, -4, -9, 99]])

print(A)
[[  9  -6   3  24]
 [ -6  20 -22  -4]
 [  3 -22  30  -9]
 [ 24  -4  -9  99]]

L = np.linalg.cholesky(A)

print(L)
[[ 3.  0.  0.  0.]
 [-2.  4.  0.  0.]
 [ 1. -5.  2.  0.]
 [ 8.  3. -1.  5.]]
```

**MATLAB example:**

```
A = [9 -6 3 24; -6 20 -22 -4; 3 -22 30 -9; 24 -4 -9 99]

A =

     9    -6     3    24
    -6    20   -22    -4
     3   -22    30    -9
    24    -4    -9    99

L = chol(A)

L =

     3    -2     1     8
     0     4    -5     3
     0     0     2    -1
     0     0     0     5
```

MATLAB provided $\boldsymbol{L}^T$.

# 12. Non-linearity problem

Before diving into the problem solution, we must understand the problem itself. What are the non-linear systems, and why does the standard Linear Kalman Filter fail with non-linear systems?

We distinguish between two types of non-linearities:

- State-to-measurement non-linear relation
- Non-linear system dynamics

We will treat each type of non-linearity separately and then combine them.

First, let us start with an example of the linear system.

## 12.1 Example – linear system

Assume an air balloon that can move only upwards or downwards with constant acceleration dynamics. We are interested in estimating the balloon altitude.

The balloon system dynamic model is linear.

We can describe the balloon system dynamics as follows:

$$\hat{\boldsymbol{x}}_{n+1,n} = \boldsymbol{F}\hat{\boldsymbol{x}}_{n,n} \tag{12.1}$$

$$\begin{bmatrix} \hat{x}_{n+1,n} \\ \hat{\dot{x}}_{n+1,n} \\ \hat{\ddot{x}}_{n+1,n} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t & 0.5\Delta t^2 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{x}_{n,n} \\ \hat{\dot{x}}_{n,n} \\ \hat{\ddot{x}}_{n,n} \end{bmatrix} \tag{12.2}$$

Where:

$x$ is the balloon altitude
$\dot{x}$ is the balloon velocity
$\ddot{x}$ is the balloon acceleration
$\Delta t$ is the time interval

We are already familiar with linear motion system dynamics from Part II.

The balloon altitude is measured by the radar located beneath the balloon.

**Figure 12.1:** *Balloon altitude measurement using radar.*

The radar measurement error distribution is Gaussian.

The radar sends an electromagnetic pulse toward the balloon. The pulse is reflected from the balloon and received by the radar. The radar measures the time elapsed between pulse transmission and pulse reception. Since the pulse travels with the speed of light, we can easily calculate the target range (which is the balloon altitude):

$$x = t\frac{c}{2} \tag{12.3}$$

Where:

- $x$ is the balloon altitude
- $c$ is the speed of light

Let us construct the observation matrix $\boldsymbol{H}$.

$$z_n = \boldsymbol{H}x_n \tag{12.4}$$

Where:

$x_n$ is the balloon altitude
$\boldsymbol{z}_n$ is the measured time delay

$$\boldsymbol{H} = \begin{bmatrix} \frac{2}{c} \end{bmatrix} \tag{12.5}$$

The dependency between the measured value (elapsed time) and the estimated value (balloon altitude) is linear. However, what happens to the estimation uncertainty? Is it still Gaussian?

The following chart depicts the dependency between the elapsed time and the balloon altitude. We can also see the distribution of measurement error (uncertainty) on the bottom plot and the state estimation error (uncertainty) on the left plot.

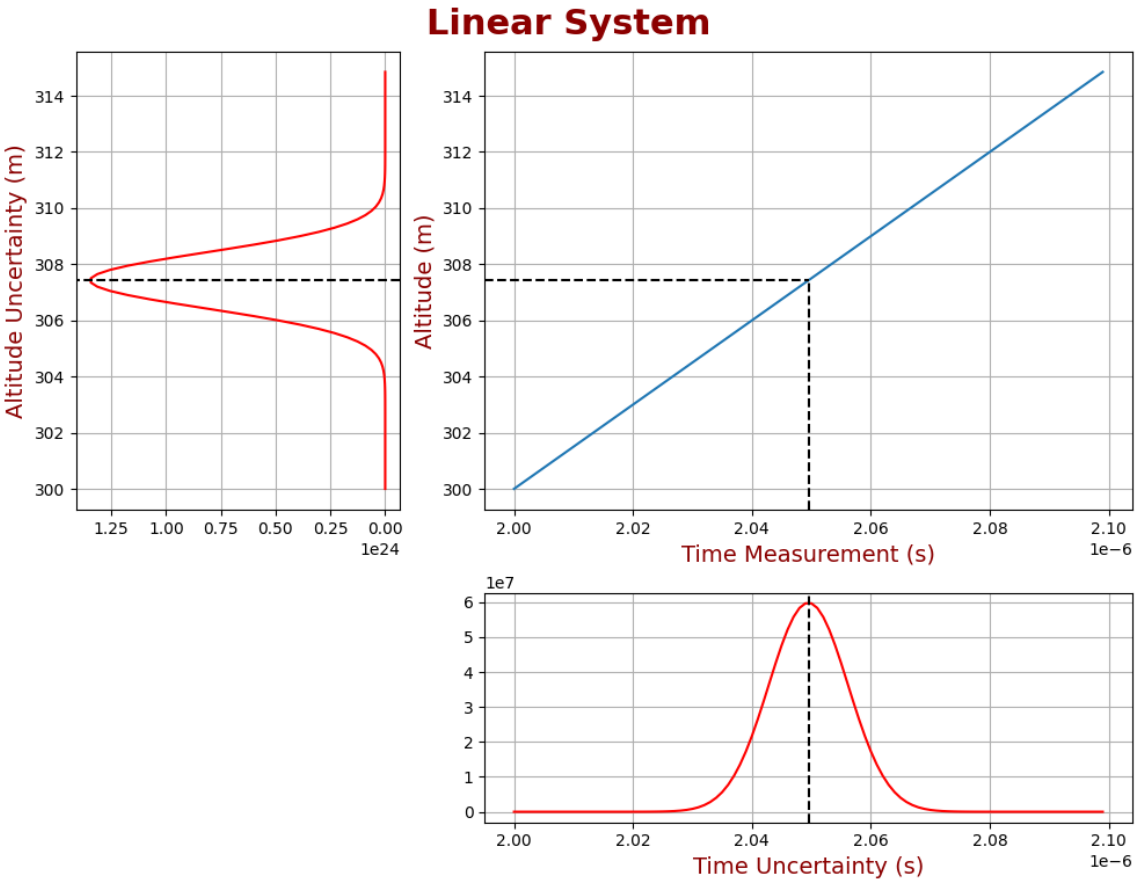


**Figure 12.2:** *Linear System.*

Since the dependency is between the measured value and the estimated value is linear. The estimated value uncertainty is also Gaussian!

## 12.2 Example – State-to-measurement non-linear relation

This example presents the first type of non-linearity: state-to-measurement relation. Let us see what happens to the estimated value uncertainty when the state-to-measurement relation is non-linear.

Like in the previous example, we are interested in measuring the balloon altitude. The balloon system dynamic model is linear and similar to the previous example. The balloon altitude is measured by the optical sensor that is located aside. The distance d between the sensor and the balloon nadir is known. The optical sensor can also measure the target angle. The sensor measurement error distribution is Gaussian.

R Nadir is the direction pointing directly below a particular location.

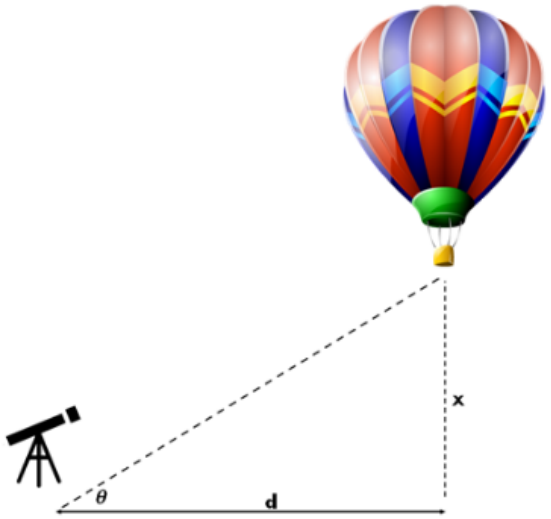


**Figure 12.3:** *Balloon altitude measurement using optical sensor.*

The balloon altitude can be calculated by using a trigonometric function:

$$x = d \cdot tan(\theta) \tag{12.6}$$

Where:

- $x_n$ is the balloon altitude
- $d$ is the distance between the sensor and the balloon nadir
- $\theta$ is the balloons elevation angle

The tangent function is non-linear. We can't construct the observation matrix $\boldsymbol{H}$!

For a linear system, the measurement equation for the linear system is given by:

$$\boldsymbol{z}_n = \boldsymbol{H}\boldsymbol{x}_n + \boldsymbol{v}_n \tag{12.7}$$

Where:

$\boldsymbol{z}_n$ is the measurement vector
$\boldsymbol{x}_n$ is the true system state (the hidden state)
$\boldsymbol{v}_n$ is a random noise vector
$\boldsymbol{H}$ is the observation matrix

For the system with non-linear state-to-measurement relation, the observation matrix $\boldsymbol{H}$ is a function of $\boldsymbol{x}$. Therefore, the measurement equation looks like:

$$\boldsymbol{z}_n = \boldsymbol{h}(\boldsymbol{x}_n, \boldsymbol{v}_n) \tag{12.8}$$

In this example:

$$\theta = tan^{-1}\frac{x}{d} \tag{12.9}$$

Let us see what happens to the estimated value uncertainty distribution.

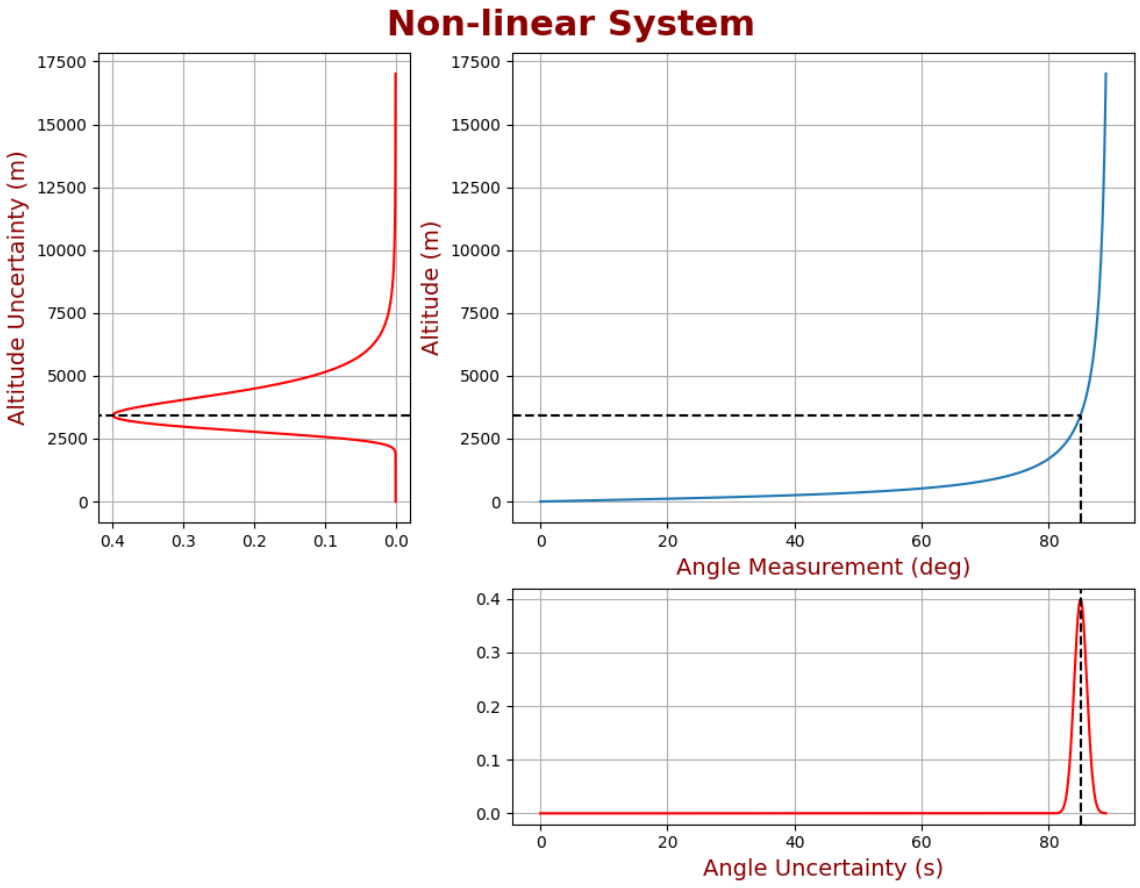


**Figure 12.4:** *Non-linear System.*

Figure 12.4 depicts the dependency between the measured angle and the balloon altitude. We can also see the distribution of measurement error (uncertainty) on the bottom plot and the state estimation error (uncertainty) on the left plot.

The altitude uncertainty distribution is not Gaussian! It also changes its shape at different measurement points.

The Kalman Filter algorithm assumes that the distribution of all random variables is Gaussian. For non-linear systems, this assumption does not hold anymore. The algorithm is not stable and yields significant estimation errors.

## 12.3 Example – Non-linear system dynamics

This example presents the second type of non-linearity: non-linear system dynamics.

Assume an ideal gravity pendulum that consists of a body with mass m hung by a string with length $L$ from fixed support - the pendulum swings back and forth at a constant amplitude.

We want to estimate the angle $\theta$ from the vertical to the pendulum.

The angle units are radians.

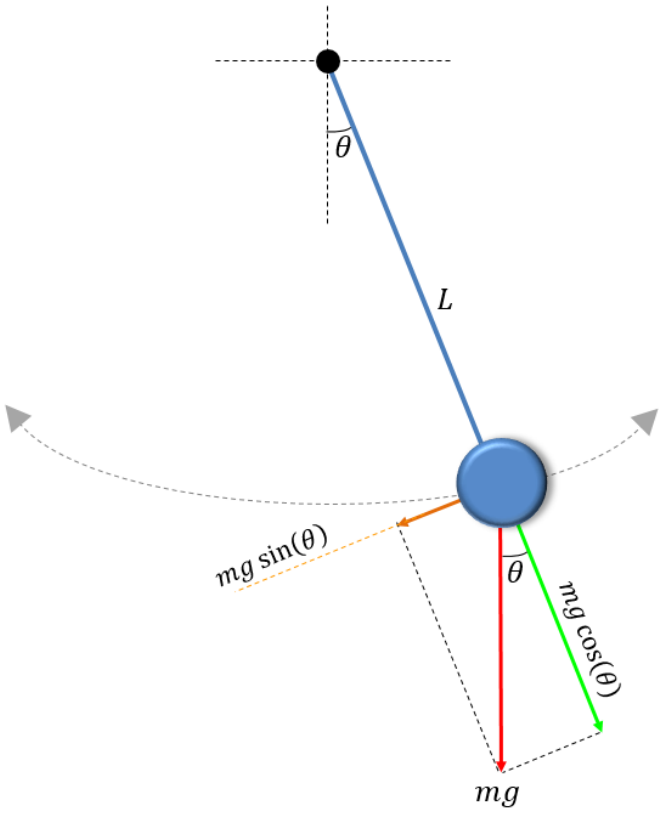


**Figure 12.5:** *Pendulum.*

First, we need to describe the pendulum motion.

According to Newton's second law, the sum of forces on the object equals:

$$F = ma \tag{12.10}$$

Where:

$m$ is the body mass

$a$ is acceleration

The force that is applied to the pendulum equals:

$$F = -mg\sin(\theta) = ma \tag{12.11}$$

The acceleration equals:

$$a = -g\sin(\theta) \tag{12.12}$$

The arc length $s$ that corresponds to angle $\theta$ is:

$$s = L\theta \tag{12.13}$$

Remember that $\theta$ units are radians. For degrees units: $s = L\theta\frac{\pi}{180}$

The pendulum velocity equals:

$$v = \frac{ds}{dt} = L\frac{d\theta}{dt} \tag{12.14}$$

The pendulum acceleration equals:

$$a = \frac{d^2s}{dt^2} = L\frac{d^2\theta}{dt^2} = -g\sin(\theta) \tag{12.15}$$

Thus, the differential equation that describes the pendulum movement is:

$$L\frac{d^2\theta}{dt^2} = -g\sin(\theta) \tag{12.16}$$

It is the second-order homogeneous differential equation.

Once we've derived the motion equation, we can define the dynamic model.

The state vector of the pendulum is in the form of the following:

$$\boldsymbol{x}_n = \begin{bmatrix} \theta_n \\ \dot{\theta}_n \end{bmatrix} \tag{12.17}$$

Where:

$\theta_n$ is the pendulum angle at time $n$

$\dot{\theta}_n$ is the pendulum angular velocity at time $n$

$$\begin{aligned} \hat{\theta}_{n+1,n} &= \hat{\theta}_{n,n} + \hat{\dot{\theta}}_{n,n}\Delta t \\ \hat{\dot{\theta}}_{n+1,n} &= \hat{\dot{\theta}}_{n,n} + \hat{\ddot{\theta}}_{n,n}\Delta t = \hat{\dot{\theta}}_{n,n} - \frac{g}{L} sin(\hat{\theta}_{n,n})\Delta t \end{aligned} \tag{12.18}$$

The dynamic model is not linear. For a linear system, the general form of the state extrapolation equation in a matrix notation is:

$$\hat{\boldsymbol{x}}_{n+1,n} = \boldsymbol{F}\hat{\boldsymbol{x}}_{n,n} + \boldsymbol{G}\boldsymbol{u}_n + \boldsymbol{w}_n \tag{12.19}$$

Where:

$\hat{\boldsymbol{x}}_{n+1,n}$ is a predicted system state vector at time step $n+1$

$\hat{\boldsymbol{x}}_{n,n}$ is an estimated system state vector at time step $n$

$\boldsymbol{u}_n$ is a control variable or input variable - a measurable (deterministic) input to the system

$\boldsymbol{w}_n$ is a process noise or disturbance - an unmeasurable input that affects the state

$\boldsymbol{F}$ is a state transition matrix

$\boldsymbol{G}$ is a control matrix or input transition matrix (mapping control to state variables)

For the non-linear system dynamics, the state transition matrix $\boldsymbol{F}$ is a function of $\boldsymbol{x}$ and $\boldsymbol{u}$. Therefore, the state extrapolation equation looks like:

$$\hat{\boldsymbol{x}}_{n+1,n} = \boldsymbol{f}(\hat{\boldsymbol{x}}_{n,n}, \boldsymbol{u}_n, \boldsymbol{w}_n) \tag{12.20}$$

In this example:

$$\boldsymbol{f}(\hat{\boldsymbol{x}}_{n+1,n}) = \begin{bmatrix} \hat{\theta}_{n,n} + \hat{\dot{\theta}}_{n,n}\Delta t \\ \hat{\dot{\theta}}_{n,n} - \frac{g}{L} sin(\hat{\theta}_{n,n})\Delta t \end{bmatrix} \tag{12.21}$$

The measurement equation depends on the measured parameter.

Let us review two cases:

**Case 1** - measured parameter is pendulum angle $\theta$

In this case, the measurement equation looks like:

$$\theta_{n,measured} = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \theta_n \\ \dot{\theta}_n \end{bmatrix} \tag{12.22}$$

The above equation is in the form of: $\boldsymbol{z}_n = \boldsymbol{H}\boldsymbol{x}_n$. Therefore, the state-to-measurement relation (the first type of non-linearity) is linear.

**Case 2** - measured parameter is the pendulum $x$ - position

In this case, the measurement equation looks like:

$$\theta_{n,measured} = Lsin(\theta_n) \tag{12.23}$$

The above equation is in the form of: $\boldsymbol{z}_n = \boldsymbol{h}(\boldsymbol{x}_n)$. Therefore, the state-to-measurement relation is non-linear.

Now we are ready to discuss different linearization methods.

# 13. Extended Kalman Filter (EKF)

The Extended Kalman Filter has emerged from National Aeronautics and Space Administration (NASA) Dynamic Analysis Branch research, led by Dr. Schmidt [5], [6].

The main idea behind the EKF is a linearization of the dynamic model at the working point.

This chapter includes a detailed explanation of the concept and two numerical examples.

## 13.1 Analytic linearization

The EKF performs analytic linearization of the model at each point in time. The following figure exemplifies a one-dimensional case. We want to find a tangent line for a point $x = x_0$. Using the tangent line, we can project the uncertainty to the y axis and keep its' shape Gaussian.

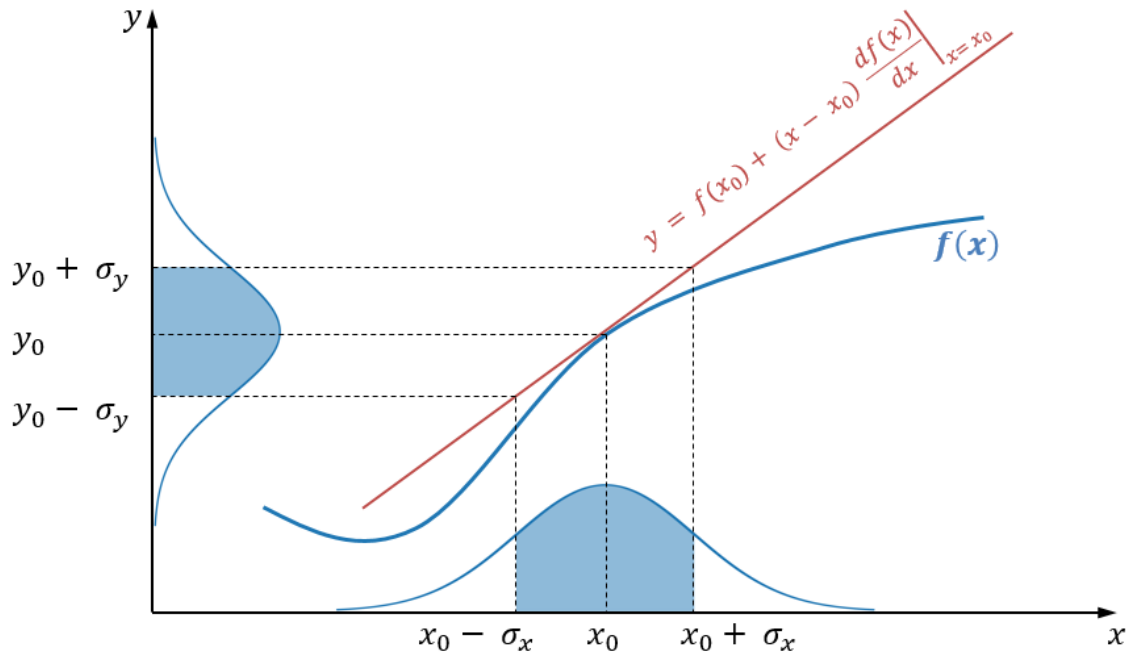


**Figure 13.1:** *Analytic Linearization.*

From basic calculus, we know that the slope of the tangent line to the function $f(x)$

at the point $x_0$ equals to the derivative of the function $f(x)$ at the point $x_0$:

$$m = \left.\frac{df(x)}{dx}\right|_{x=\ x_0} \tag{13.1}$$

Where $m$ is the slope of the tangent line.

The general straight-line equation is:

$$y = mx + b \tag{13.2}$$

Where:

$m$ is the slope of the line
$b$ is a $y$-axis intercept point

We can find the line equation given the slope $m$ and any point $x_0, y_0$ :

$$y - y_0 = m(x - x_0) \tag{13.3}$$

Expand and rearrange:

$$y = mx - mx_0 + y_0 \tag{13.4}$$

$$b = -mx_0 + y_0 \tag{13.5}$$

We can find $y_0$ using the original function $f(x)$:

$$y_0 = f(x_0) \tag{13.6}$$

So the tangent line equation would be:

$$y = mx - mx_0 + y_0 = y_0 + (x - x_0)m \tag{13.7}$$

$$y = f(x_0) + (x - x_0)\left.\frac{df(x)}{dx}\right|_{x=x_0} \tag{13.8}$$

## 13.2 First-order Taylor series expansion

In many Kalman Filter books, the process of analytic linearization is also called: "The approximation of $f(x)$ by a first-order Taylor series expansion about the point $x = x_0$".

According to Taylor's theorem, the function $f(x)$ equals an infinite sum of terms that are expressed in terms of the function derivatives at a single point $x = x_0$:

$$f(x) \approx f(x_0)+f^{'}(x_0)(x-x_0)+\frac{f^{''}(x_0)}{2!}(x-x_0)^2+\frac{f^{'''}(x_0)}{3!}(x-x_0)^3+\cdots+\frac{f^{(k)}(x_0)}{k!}(x-x_0)^k+\cdots \tag{13.9}$$

We can approximate the function $f(x)$ by calculating the first $k$ terms of the Taylor Series. For high approximation precision, we shall select a high $k$ value.

Taylor's series is named after Brook Taylor, who introduced it in 1715.

For the linear approximation, we should keep only the first two terms of the Taylor Series:

$$f(x) \approx f(x_0) + f^{'}(x_0)(x - x_0) \tag{13.10}$$

This equation is identical to Equation 13.8, which we derived in the previous chapter.

## 13.3 Uncertainty projection in one dimension

The EKF projects the uncertainty using the linearization technique.

As we will see soon, when projecting the uncertainty, there is no need to evaluate the observation function $h(x)$ and the state transition function $f(x)$. We only need to evaluate derivatives $\frac{dh(x)}{dx}, \frac{df(x)}{dx}$.

After finding the tangent line equation at the point $x_0$, we can project the uncertainty using the tangent line.

The line equation is:

$$y = mx + b \tag{13.11}$$

Where:

$$m = \left.\frac{df(x)}{dx}\right|_{x=\ x_0}$$
$$b = -mx_0 + y_0$$

The standard deviation (sigma) points are: $(x_0 + \sigma_x, \quad x_0 - \sigma_x)$.

$$\begin{aligned} y - \sigma_y &= m(x_0 - \sigma_x) + b \\ y + \sigma_y &= m(x_0 + \sigma_x) + b \end{aligned} \tag{13.12}$$

The measurement uncertainty is the difference between the sigma points:

$$(y + \sigma_y) - (y - \sigma_y) = m(x_0 + \sigma_x) + b - (m(x_0 - \sigma_x) + b) \tag{13.13}$$

$$\sigma_y = m\sigma_x \tag{13.14}$$

$$\sigma_y = \left(\frac{df(x)}{dx}\right)\sigma_x \tag{13.15}$$

The estimation variance: $p_x = \sigma_x^2$

$$p_y = \left(\frac{df(x)}{dx}\right)^2 p_x \tag{13.16}$$

### 13.3.1 Example – linearization in a single dimension

Let us return to the balloon altitude measurement example (section 12.2) with a state-to-measurement non-linear relation.

The balloon altitude is measured by the optical sensor that is located aside. The distance d between the sensor and the nadir of the balloon is known. The optical sensor can measure the target angle $\theta$.

Since the system state-to-measurement relation is not linear, the measurement equation has the following form:

$$\boldsymbol{z}_n = \boldsymbol{h}(\boldsymbol{x}_n) \tag{13.17}$$

In this example:

$$\boldsymbol{z}_n = \theta = tan^{-1}\frac{x_n}{d} \tag{13.18}$$

$$\boldsymbol{h}(\boldsymbol{x}_n) = \theta = tan^{-1}\frac{x_n}{d} \tag{13.19}$$

To propagate the measurement uncertainty from the angle domain to the altitude domain, we need to perform an analytic linearization, or in other words, differentiate $\boldsymbol{h}(\boldsymbol{x}_n)$.

$$\frac{d\boldsymbol{h}(\boldsymbol{x})}{d\boldsymbol{x}} = \frac{d\left(tan^{-1}\frac{x}{d}\right)}{dx} \tag{13.20}$$

We should calculate the derivative of $tan^{-1}\frac{x}{d}$. One can use computer software packages to calculate the derivative. I prefer to calculate derivatives myself, but checking yourself with a software package could be a good idea.

The derivative of the *arctan* is given by:

$$\frac{d}{dx}tan^{-1}(x) = \frac{1}{1+x^2} \tag{13.21}$$

Using chain rule:

$$\frac{d\left(tan^{-1}\frac{x}{d}\right)}{dx} = \frac{1}{1+\left(\frac{x}{d}\right)^2}\cdot\frac{1}{d} = \frac{d}{d^2+x^2} \tag{13.22}$$

The linearized observation matrix for this example is:

$$\frac{d\boldsymbol{h}(\boldsymbol{x})}{d\boldsymbol{x}} = \begin{bmatrix}\frac{d}{d^2+x^2}\end{bmatrix} \tag{13.23}$$

We can check the differentiation using the Python SymPy library or MATLAB symbolic toolbox.

**Python example:**

```
import sympy as sp  # import SymPy library

x, d = sp.symbols('x, d')  # define symbols

f = sp.atan(x/d)  # define function
dif = sp.diff(f, x)  # differentiate function

print(dif)  # print the result
```

The Result:

```
1/(d*(1 + x**2/d**2))
```

**MATLAB example:**

```
syms f(x) d;          % define symbols
f(x) = atan(x/d);     % define function
dif = diff(f,x);      % differentiate function

disp(dif);            % print the result
```

The Result:

```
1/(d*(x^2/d^2 + 1))
```

The following figure depicts the analytic linearization result for the "measuring balloon altitude" example.

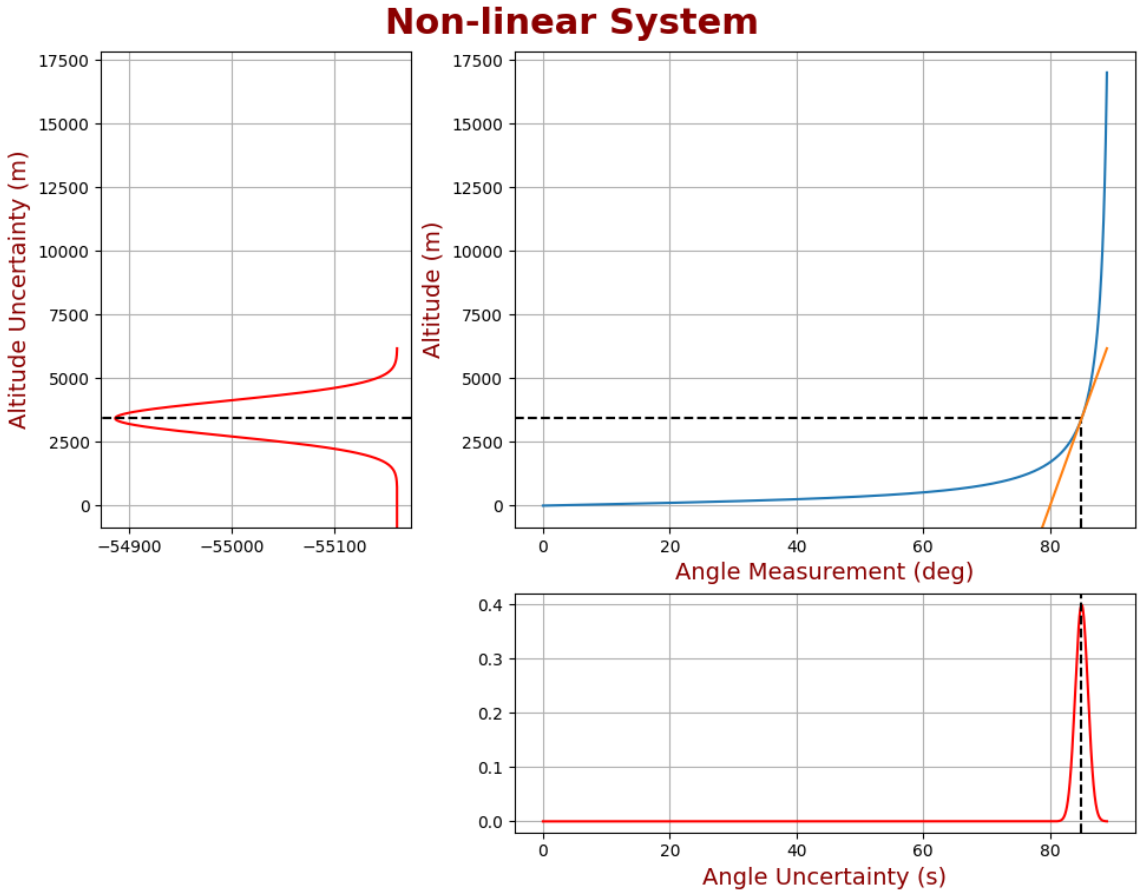


**Figure 13.2:** *Linearization .*

The orange line is the tangent line to the function $x = d \cdot tan\theta$ at the point $\theta = 85^0$.

## 13.4 Uncertainty projection in two dimensions

In two dimensions, the uncertainty is projected through a tangent plane. The following figure describes a tangent plane of the non-linear function $f(x, y)$ at a point $x_0, y_0$.

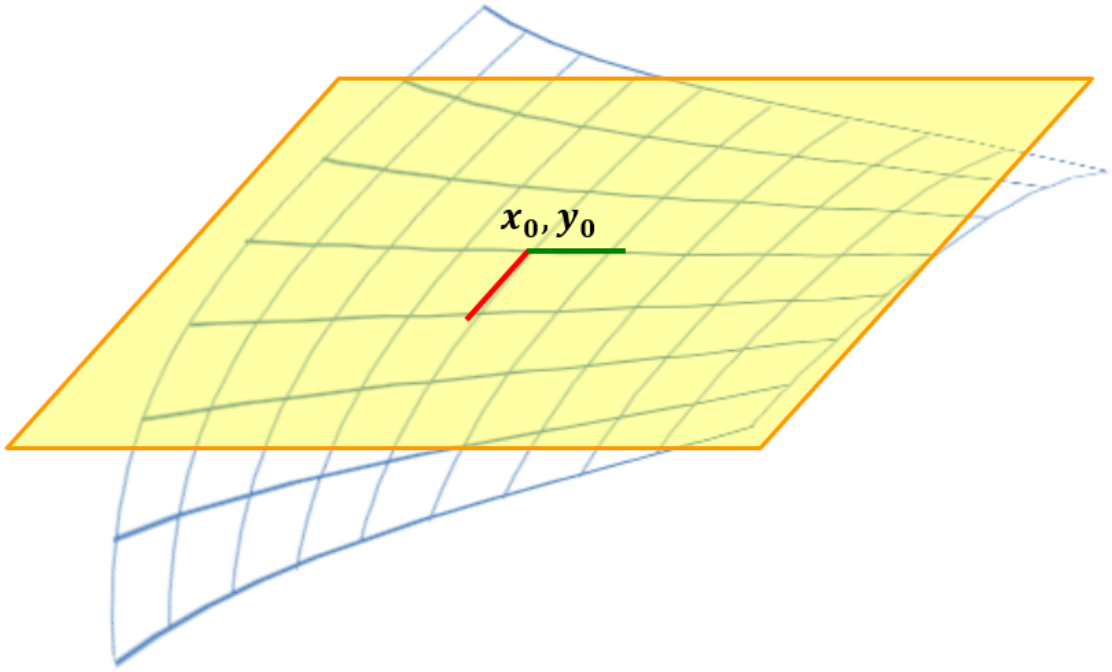


**Figure 13.3:** *The tangent plane.*

The tangent plane is characterized by two orthogonal slopes – the $x$ - axis slope and the $y$ - axis slope. The partial derivatives of $f(x, y)$ at a point $x_0, y_0$ are the slopes of the tangent plane:

$$\frac{\partial f(x,y)}{\partial x}, \frac{\partial f(x,y)}{\partial y} \tag{13.24}$$

As we've seen in the previous section, when projecting the uncertainty, there is no need to evaluate the observation function $\boldsymbol{h}(\boldsymbol{x})$ and the state transition function $\boldsymbol{f}(\boldsymbol{x})$. We only need to evaluate derivatives. In two dimensions, we should find two partial derivatives.

Let us recall the pendulum example (section 12.3). The state vector of the pendulum is in the form of the following:

$$\boldsymbol{x}_n = \begin{bmatrix} \theta_n \\ \dot{\theta}_n \end{bmatrix} \tag{13.25}$$

Where:

$\theta_n$ is the pendulum angle at time $n$
$\dot{\theta}_n$ is the pendulum angular velocity at time $n$

We measure the pendulum position: $L \cdot sin(\theta_n)$.

Since the state-to-measurement relation (the first type of non-linearity) is non-linear, the measurement equation is a type of:

$$\boldsymbol{z}_n = \boldsymbol{h}(\boldsymbol{x}_n) \tag{13.26}$$

$$\boldsymbol{h}(\boldsymbol{x}_n) = L \cdot sin(\theta_n) \tag{13.27}$$

The multivariate analytical linearization is given by:

$$\boldsymbol{P}_{out} = \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}} \boldsymbol{P}_{in} \left( \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}} \right)^T \tag{13.28}$$

We need to find the partial derivatives of $\boldsymbol{h}(\boldsymbol{x})$:

$$\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial \left( Lsin(\theta_n) \right)}{\partial \theta} & \frac{\partial \left( Lsin(\theta_n) \right)}{\partial \dot{\theta}} \end{bmatrix} = \begin{bmatrix} Lcos(\theta_n) & 0 \end{bmatrix} \tag{13.29}$$

$$\boldsymbol{P}_{out} = \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}} \boldsymbol{P}_{in} \left( \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}} \right)^T = \left( Lcos(\theta_n) \right)^2 \boldsymbol{P}_{in} \tag{13.30}$$

The dynamic model of the pendulum is also non-linear (the second type of non-linearity). It has the form of:

$$\hat{\boldsymbol{x}}_{n+1,n} = \boldsymbol{f}(\hat{\boldsymbol{x}}_{n,n}) \tag{13.31}$$

$$\hat{\boldsymbol{x}}_{n+1,n} = \begin{bmatrix} \hat{\theta}_{n+1,n} \\ \hat{\dot{\theta}}_{n+1,n} \end{bmatrix} = \begin{bmatrix} \hat{\theta}_{n,n} + \hat{\dot{\theta}}_{n,n} \Delta t \\ \hat{\dot{\theta}}_{n,n} - \frac{g}{L} sin(\hat{\theta}_{n,n}) \Delta t \end{bmatrix} \tag{13.32}$$

$$\boldsymbol{f}(\hat{\boldsymbol{x}}_{n,n}) = \begin{bmatrix} f_1(\hat{x}_{n,n}) \\ f_2(\hat{x}_{n,n}) \end{bmatrix} = \begin{bmatrix} \hat{\theta}_{n,n} + \hat{\dot{\theta}}_{n,n} \Delta t \\ \hat{\dot{\theta}}_{n,n} - \frac{g}{L} sin(\hat{\theta}_{n,n}) \Delta t \end{bmatrix} \tag{13.33}$$

The dynamic model function $\boldsymbol{f}(\hat{\boldsymbol{x}}_{n,n})$ is a matrix that contains two different sub-functions. We should find partial derivatives for each sub-function.

$$\frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial \hat{\theta}} & \frac{\partial f_1}{\partial \hat{\dot{\theta}}} \\ \frac{\partial f_2}{\partial \hat{\theta}} & \frac{\partial f_2}{\partial \hat{\dot{\theta}}} \end{bmatrix} = \begin{bmatrix} \frac{\partial \left(\hat{\theta}_{n,n} + \hat{\dot{\theta}}_{n,n}\Delta t\right)}{\partial \hat{\theta}} & \frac{\partial \left(\hat{\theta}_{n,n} + \hat{\dot{\theta}}_{n,n}\Delta t\right)}{\partial \hat{\dot{\theta}}} \\ \frac{\partial \left(\hat{\dot{\theta}}_{n,n} - \frac{g}{L} sin(\hat{\theta}_{n,n})\Delta t\right)}{\partial \hat{\theta}} & \frac{\partial \left(\hat{\dot{\theta}}_{n,n} - \frac{g}{L} sin(\hat{\theta}_{n,n})\Delta t\right)}{\partial \hat{\dot{\theta}}} \end{bmatrix}$$
$$= \begin{bmatrix} 1 & \Delta t \\ -\frac{g}{L} cos(\hat{\theta}_{n,n})\Delta t & 1 \end{bmatrix} \tag{13.34}$$

The multivariate analytical linearization is given by:

$$\boldsymbol{P}_{out} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} \boldsymbol{P}_{in} \left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right)^T \tag{13.35}$$

## 13.5 Multivariate uncertainty projection

We can generalize the two-dimensional case by extending it to an $N$ - dimensional case.

For multi-dimensional problems, we propagate the multivariate Gaussian random variable (represented by covariance matrix) using a linear approximation of the multi-dimensional function.

For the non-linear system dynamics, the multivariate analytical linearization is given by:

$$\boldsymbol{P}_{out} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} \boldsymbol{P}_{in} \left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right)^T \tag{13.36}$$

Where:

$\boldsymbol{P}_{in}$ is an input covariance
$\boldsymbol{P}_{out}$ is a projected covariance
$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}$ is a state transition matrix Jacobian

Jacobian matrix is a matrix of partial derivatives, named after Karl Gustav Jacob

Jacobi, a German mathematician.

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} \tag{13.37}$$

Similarly, for the state-to-measurement non-linear relation, the multivariate analytical linearization is given by:

$$\boldsymbol{P}_{out} = \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}} \boldsymbol{P}_{in} \left( \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}} \right)^T \tag{13.38}$$

Where:

$\boldsymbol{P}_{in}$ is an input covariance

$\boldsymbol{P}_{out}$ is a projected covariance

$\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}$ is an observation matrix Jacobian

For additional material on error propagation, I recommend reading [7].

### 13.5.1 Jacobian derivation example

Let us complicate the "Vehicle location estimation example" (section 9.1). We estimated the vehicle location in the $XY$ plane in that example. The vehicle had an onboard location sensor that reported $x$ and $y$ coordinates of the system.

Now, we want to track the vehicle using radar. The radar is located at the plane origin, and it measures the vehicle range ($r$) and bearing angle ($\varphi$).

The radar measurement error distribution is Gaussian.

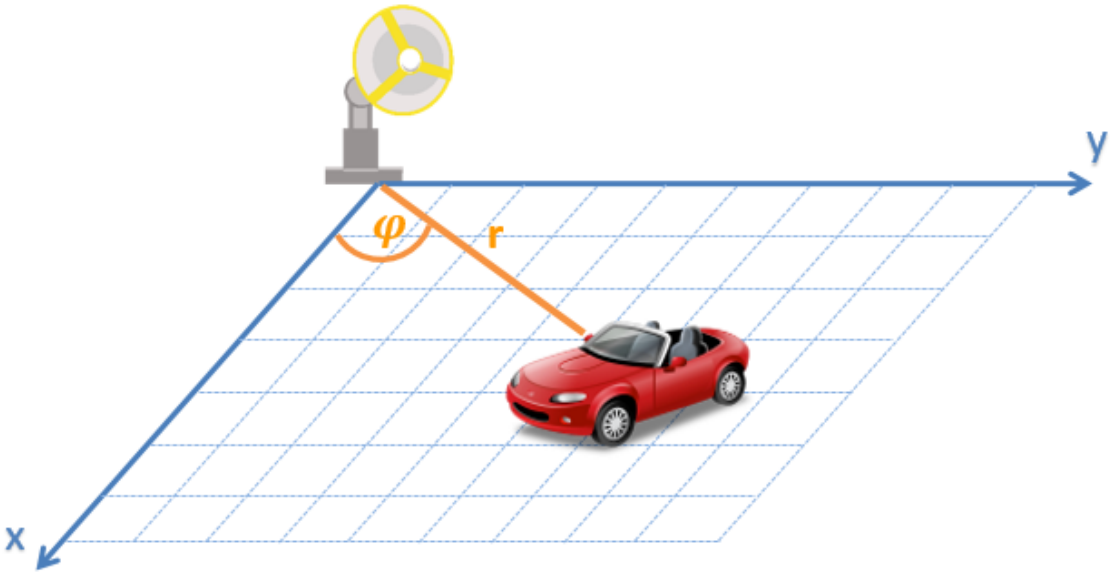


**Figure 13.4:** *Vehicle location estimation using radar.*

The measurement vector $\boldsymbol{z}_n$ is:

$$\boldsymbol{z}_n = \begin{bmatrix} r_n \\ \varphi_n \end{bmatrix} \tag{13.39}$$

The state vector $\boldsymbol{x}_n$ is:

$$\boldsymbol{x}_n = \begin{bmatrix} x_n \\ y_n \end{bmatrix} \tag{13.40}$$

Let us find the relation between the measurement vector and the state vector.

The vehicle range ($r$) can be expressed by $x$ and $y$ using the Pythagorean theorem:

$$r = \sqrt{x^2 + y^2} \tag{13.41}$$

The vehicle bearing angle ($\varphi$) can be expressed by $x$ and $y$ using a trigonometrical function:

$$\varphi = tan^{-1}\frac{y}{x} \tag{13.42}$$

Since the state-to-measurement relation is non-linear, the measurement equation is a type of $\boldsymbol{z}_n = \boldsymbol{h}(\boldsymbol{x}_n)$:

$$\begin{bmatrix} r \\ \varphi \end{bmatrix} = \begin{bmatrix} \sqrt{x^2 + y^2} \\ tan^{-1}\dfrac{y}{x} \end{bmatrix} \tag{13.43}$$

Jacobian derivation:

$$\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}} = \begin{bmatrix} \dfrac{\partial h_1}{\partial x_1} & \cdots & \dfrac{\partial h_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial h_m}{\partial x_1} & \cdots & \dfrac{\partial h_m}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial \left(\sqrt{x^2 + y^2}\right)}{\partial x} & \cdots & \dfrac{\partial \left(\sqrt{x^2 + y^2}\right)}{\partial y} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial \left(tan^{-1}\dfrac{y}{x}\right)}{\partial x} & \cdots & \dfrac{\partial \left(tan^{-1}\dfrac{y}{x}\right)}{\partial y} \end{bmatrix} \tag{13.44}$$

For this example, the Jacobian is a square 4 elements matrix of partial derivatives. Let us calculate each element separately.

$$\frac{\partial \left(\sqrt{x^2 + y^2}\right)}{\partial x} = \left(\frac{1}{2}\left(x^2 + y^2\right)^{-\frac{1}{2}}\right) 2x = \frac{x}{\sqrt{x^2 + y^2}} \tag{13.45}$$

$$\frac{\partial \left(\sqrt{x^2 + y^2}\right)}{\partial y} = \left(\frac{1}{2}\left(x^2 + y^2\right)^{-\frac{1}{2}}\right) 2y = \frac{y}{\sqrt{x^2 + y^2}} \tag{13.46}$$

$$\frac{\partial \left(tan^{-1}\dfrac{y}{x}\right)}{\partial x} = \left(\frac{1}{1 + \left(\frac{y}{x}\right)^2}\right)\left(-\frac{y}{x^2}\right) = -\frac{y}{x^2 + y^2} \tag{13.47}$$

$$\frac{\partial \left(tan^{-1}\frac{y}{x}\right)}{\partial y} = \left(\frac{1}{1 + \left(\frac{y}{x}\right)^2}\right)\left(\frac{1}{x}\right) = \frac{x}{x^2 + y^2} \tag{13.48}$$

$$\frac{\partial \boldsymbol{h}(\boldsymbol{x})}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{x}{\sqrt{x^2+y^2}} & \dfrac{y}{\sqrt{x^2 + y^2}} \\ -\dfrac{y}{x^2 + y^2} & \dfrac{x}{x^2 + y^2} \end{bmatrix} \tag{13.49}$$

We also can derive Jacobian using computer software packages.

**Python example:**

```
import sympy as sp  # import SymPy library

x, y = sp.symbols('x, y')  # define symbols

H = sp.Matrix([sp.sqrt(x**2 + y**2), sp.atan(y/x)])  # define H
    matrix
J = H.jacobian([x,y])  # calculate Jacobian

print(J)  # print the result
```

The Result:

```
Matrix([[x/sqrt(x**2 + y**2), y/sqrt(x**2 + y**2)],
[-y/(x**2*(1 + y**2/x**2)), 1/(x*(1 + y**2/x**2))]])
```

**MATLAB example:**

```
syms x y z                              % define symbols
H = [sqrt(x^2 + y^2) atan(y/x)];        % define H matrix
J = jacobian(H);                        % calculate Jacobian

disp(J);                                % print the result
```

The Result:

```
[    x/(x^2 + y^2)^(1/2), y/(x^2 + y^2)^(1/2)]
[ -y/(x^2*(y^2/x^2 + 1)), 1/(x*(y^2/x^2 + 1))]
```

## 13.6 EKF equations

The concept of the EKF is similar to the LKF (Linear Kalman Filter); however, some modifications should be made. The modifications are related to the observation matrix $\boldsymbol{H}$ and the state transition matrix $\boldsymbol{F}$.

### 13.6.1 The EKF observation matrix

If the state-to-measurement relation (the first type of non-linearity) of the system is non-linear, the observation matrix is of the type:

$$\boldsymbol{H} = \boldsymbol{h}(\boldsymbol{x}_n) \tag{13.50}$$

The State Update Equation looks like the following:

$$\hat{\boldsymbol{x}}_{n,n} = \hat{\boldsymbol{x}}_{n,n-1} + \boldsymbol{K}_n \left( \boldsymbol{z}_n - \boldsymbol{h}(\hat{\boldsymbol{x}}_{n,n-1}) \right) \tag{13.51}$$

For the uncertainty propagation, the observation matrix $\boldsymbol{H}$ should be linearized to keep the uncertainty PDF Gaussian.

The Covariance Update Equation looks like the following:

$$\boldsymbol{P}_{n,n} = \left( \boldsymbol{I} - \boldsymbol{K}_n \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}} \right) \boldsymbol{P}_{n,n-1} \left( \boldsymbol{I} - \boldsymbol{K}_n \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}} \right)^T + \boldsymbol{K}_n \boldsymbol{R}_n \boldsymbol{K}_n^T \tag{13.52}$$

The Kalman Gain Equation looks like the following:

$$\boldsymbol{K}_n = \boldsymbol{P}_{n,n-1} \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}^T \left( \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}} \boldsymbol{P}_{n,n-1} \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}^T + \boldsymbol{R}_n \right)^{-1} \tag{13.53}$$

### 13.6.2 The EKF state transition matrix

If the dynamic model (the second type of non-linearity) of the system is non-linear, the observation matrix is of the type:

$$\boldsymbol{F} = \boldsymbol{f}(\boldsymbol{x}_n) \tag{13.54}$$

For the uncertainty propagation, the state transition matrix $\boldsymbol{F}$ should be linearized to keep the uncertainty PDF Gaussian. The State Extrapolation Equation looks like the following:

$$\hat{\boldsymbol{x}}_{n+1,n} = \boldsymbol{f}(\hat{\boldsymbol{x}}_{n,n}) + \boldsymbol{G}\boldsymbol{u}_n \tag{13.55}$$

The Covariance Extrapolation Equation looks like the following:

$$\boldsymbol{P}_{n+1,n} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} \boldsymbol{P}_{n,n} \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}^T + \boldsymbol{Q} \tag{13.56}$$

### 13.6.3 EKF equations summary

The following table compares Extended Kalman Filter equations to the Linear Kalman Filter equations.

| | Equation | LKF Equation | EKF Equation |
|---|---|---|---|
| Predict | State Extrapolation | $\hat{\boldsymbol{x}}_{n+1,n} = \boldsymbol{F}\hat{\boldsymbol{x}}_{n,n} + \boldsymbol{G}\boldsymbol{u}_n$ | $\hat{\boldsymbol{x}}_{n+1,n} = \boldsymbol{f}(\hat{\boldsymbol{x}}_{n,n}) + \boldsymbol{G}\boldsymbol{u}_n$ |
| | Covariance Extrapolation | $\boldsymbol{P}_{n+1,n} = \boldsymbol{F}\boldsymbol{P}_{n,n}\boldsymbol{F}^T + \boldsymbol{Q}$ | $\boldsymbol{P}_{n+1,n} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\boldsymbol{P}_{n,n}\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right)^T + \boldsymbol{Q}$ |
| Update | State Update | $\hat{\boldsymbol{x}}_{n,n} = \hat{\boldsymbol{x}}_{n,n-1}$ $+\boldsymbol{K}_n\left(\boldsymbol{z}_n - \boldsymbol{H}\hat{\boldsymbol{x}}_{n,n-1}\right)$ | $\hat{\boldsymbol{x}}_{n,n} = \hat{\boldsymbol{x}}_{n,n-1}$ $+\boldsymbol{K}_n\left(\boldsymbol{z}_n - \boldsymbol{h}(\hat{\boldsymbol{x}}_{n,n-1})\right)$ |
| | Covariance Update | $\boldsymbol{P}_{n,n} = \left(\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H}\right)\boldsymbol{P}_{n,n-1}$ $\times\left(\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H}\right)^T + \boldsymbol{K}_n\boldsymbol{R}_n\boldsymbol{K}_n^T$ | $\boldsymbol{P}_{n,n} = \left(\boldsymbol{I} - \boldsymbol{K}_n\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}\right)\boldsymbol{P}_{n,n-1}$ $\times\left(\boldsymbol{I} - \boldsymbol{K}_n\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}\right)^T + \boldsymbol{K}_n\boldsymbol{R}_n\boldsymbol{K}_n^T$ |
| | Kalman Gain | $\boldsymbol{K}_n = \boldsymbol{P}_{n,n-1}\boldsymbol{H}^T$ $\times\left(\boldsymbol{H}\boldsymbol{P}_{n,n-1}\boldsymbol{H}^T + \boldsymbol{R}_n\right)^{-1}$ | $\boldsymbol{K}_n = \boldsymbol{P}_{n,n-1}\left(\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}\right)^T$ $\times\left(\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}\boldsymbol{P}_{n,n-1}\left(\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}\right)^T + \boldsymbol{R}_n\right)^{-1}$ |

**Table 13.1:** *Kalman Filter equations.*

Once we understand the concept, we can proceed to numerical examples.

## 13.7 Example 11 – vehicle location estimation using radar

I introduced this example earlier (subsection 13.5.1) when I explained the multivariate uncertainty projection concept.

We want to track the vehicle using radar. The radar is located at the plane origin, and it measures the vehicle range ($r$) and the bearing angle ($\varphi$).

The radar measurement error distribution is Gaussian. We assume constant acceleration dynamics.

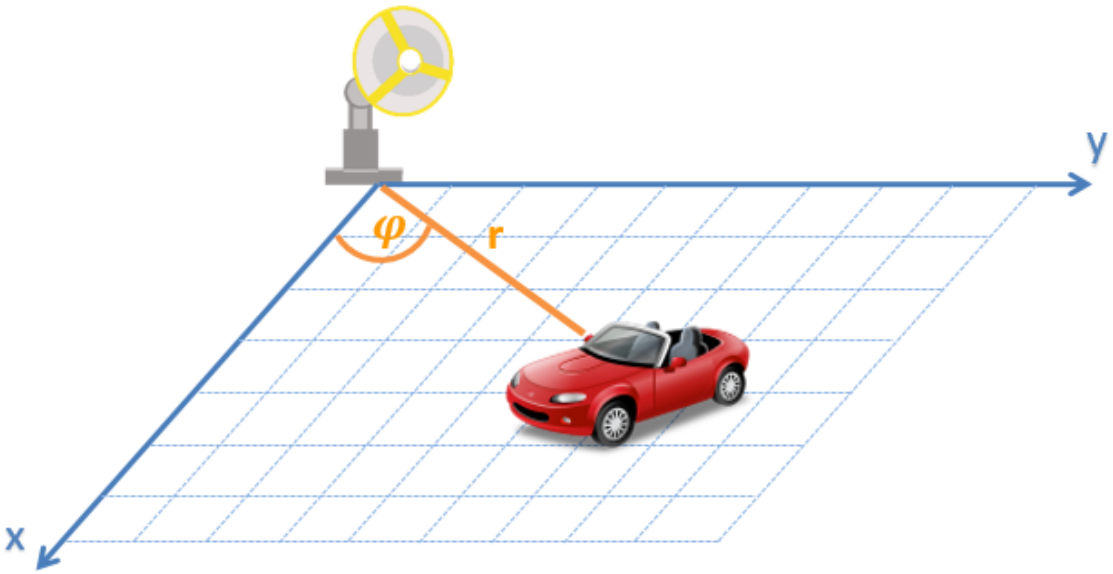


**Figure 13.5:** *Vehicle location estimation using radar.*

### 13.7.1 Kalman Filter equations

#### The state extrapolation equation

Similarly to example 9 (section 9.1), the state extrapolation equation is:

$$\hat{\boldsymbol{x}}_{n+1,n} = \boldsymbol{F}\hat{\boldsymbol{x}}_{n,n} \tag{13.57}$$

The system state $\boldsymbol{x}_n$ is defined by:

$$\boldsymbol{x}_n = \begin{bmatrix} x_n \\ \dot{x}_n \\ \ddot{x}_n \\ y_n \\ \dot{y}_n \\ \ddot{y}_n \end{bmatrix} \tag{13.58}$$

The extrapolated vehicle state for time $n+1$ can be described as follows:

$$\begin{bmatrix} \hat{x}_{n+1,n} \\ \hat{\dot{x}}_{n+1,n} \\ \hat{\ddot{x}}_{n+1,n} \\ \hat{y}_{n+1,n} \\ \hat{\dot{y}}_{n+1,n} \\ \hat{\ddot{y}}_{n+1,n} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t & 0.5\Delta t^2 & 0 & 0 & 0 \\ 0 & 1 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta t & 0.5\Delta t^2 \\ 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{x}_{n,n} \\ \hat{\dot{x}}_{n,n} \\ \hat{\ddot{x}}_{n,n} \\ \hat{y}_{n,n} \\ \hat{\dot{y}}_{n,n} \\ \hat{\ddot{y}}_{n,n} \end{bmatrix} \tag{13.59}$$

The dynamic model of the system (the second type of non-linearity) in this example is linear! There is no need to calculate the Jacobian $\dfrac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}$.

## The covariance extrapolation equation

The Covariance Extrapolation Equation is similar to example 9 (section 9.1):

$$\boldsymbol{P}_{n+1,n} = \boldsymbol{F}\boldsymbol{P}_{n,n}\boldsymbol{F}^T + \boldsymbol{Q} \tag{13.60}$$

The estimate covariance is:

$$\boldsymbol{P} = \begin{bmatrix} p_x & p_{x\dot{x}} & p_{x\ddot{x}} & 0 & 0 & 0 \\ p_{\dot{x}x} & p_{\dot{x}} & p_{\dot{x}\ddot{x}} & 0 & 0 & 0 \\ p_{\ddot{x}x} & p_{\ddot{x}\dot{x}} & p_{\ddot{x}} & 0 & 0 & 0 \\ 0 & 0 & 0 & p_y & p_{y\dot{y}} & p_{y\ddot{y}} \\ 0 & 0 & 0 & p_{\dot{y}y} & p_{\dot{y}} & p_{\dot{y}\ddot{y}} \\ 0 & 0 & 0 & p_{\ddot{y}y} & p_{\ddot{y}\dot{y}} & p_{\ddot{y}} \end{bmatrix} \tag{13.61}$$

The elements on the main diagonal of the matrix are the variances of the estimation:

- $p_x$ is the variance of the $X$ coordinate position estimation
- $p_{\dot{x}}$ is the variance of the $X$ coordinate velocity estimation
- $p_{\ddot{x}}$ is the variance of the $X$ coordinate acceleration estimation
- $p_y$ is the variance of the $Y$ coordinate position estimation
- $p_{\dot{y}}$ is the variance of the $Y$ coordinate velocity estimation
- $p_{\ddot{y}}$ is the variance of the $Y$ coordinate acceleration estimation
- The off-diagonal entries are covariances

## The process noise matrix

The process noise matrix is also similar to example 9 (section 9.1):

$$\boldsymbol{Q} = \begin{bmatrix} \sigma_x^2 & \sigma_{x\dot{x}}^2 & \sigma_{x\ddot{x}}^2 & 0 & 0 & 0 \\ \sigma_{\dot{x}x}^2 & \sigma_{\dot{x}}^2 & \sigma_{\dot{x}\ddot{x}}^2 & 0 & 0 & 0 \\ \sigma_{\ddot{x}x}^2 & \sigma_{\ddot{x}\dot{x}}^2 & \sigma_{\ddot{x}}^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_y^2 & \sigma_{y\dot{y}}^2 & \sigma_{y\ddot{y}}^2 \\ 0 & 0 & 0 & \sigma_{\dot{y}y}^2 & \sigma_{\dot{y}}^2 & \sigma_{\dot{y}\ddot{y}}^2 \\ 0 & 0 & 0 & \sigma_{\ddot{y}y}^2 & \sigma_{\ddot{y}\dot{y}}^2 & \sigma_{\ddot{y}}^2 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} & \frac{\Delta t^2}{2} & 0 & 0 & 0 \\ \frac{\Delta t^3}{2} & \Delta t^2 & \Delta t & 0 & 0 & 0 \\ \frac{\Delta t^2}{2} & \Delta t & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} & \frac{\Delta t^2}{2} \\ 0 & 0 & 0 & \frac{\Delta t^3}{2} & \Delta t^2 & \Delta t \\ 0 & 0 & 0 & \frac{\Delta t^2}{2} & \Delta t & 1 \end{bmatrix} \sigma_a^2 \tag{13.62}$$

Where:

- $\Delta t$ is the time between successive measurements
- $\sigma_a^2$ is a random variance in acceleration

## The measurement equation

The measurement equation is different from example 9.

The measurement vector $\boldsymbol{z}_n$ is:

$$\boldsymbol{z}_n = \begin{bmatrix} r_n \\ \varphi_n \end{bmatrix} \tag{13.63}$$

The system state vector $\boldsymbol{x}_n$ is defined by:

$$\boldsymbol{x}_n = \begin{bmatrix} x_n \\ \dot{x}_n \\ \ddot{x}_n \\ y_n \\ \dot{y}_n \\ \ddot{y}_n \end{bmatrix} \tag{13.64}$$

Let us find the relation between the measurement vector and the state vector. The vehicle range ($r$) can be expressed by $x$ and $y$ using the Pythagorean theorem:

$$r = \sqrt{x^2 + y^2} \tag{13.65}$$

The vehicle bearing angle ($\varphi$) can be expressed by x and y using a trigonometrical function:

$$\varphi = tan^{-1}\frac{y}{x} \tag{13.66}$$

Since the state-to-measurement relation (the first type of non-linearity) is non-linear, the measurement equation is a type of $\boldsymbol{z}_n = \boldsymbol{h}(\boldsymbol{x}_n)$:

$$\begin{bmatrix} r \\ \varphi \end{bmatrix} = \begin{bmatrix} \sqrt{x^2 + y^2} \\ tan^{-1}\frac{y}{x} \end{bmatrix} \tag{13.67}$$

Jacobian derivation:

$$\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \cdots & \frac{\partial h_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_m}{\partial x_1} & \cdots & \frac{\partial h_m}{\partial x_n} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\partial\left(\sqrt{x^2+y^2}\right)}{\partial x} & \frac{\partial\left(\sqrt{x^2+y^2}\right)}{\partial \dot{x}} & \frac{\partial\left(\sqrt{x^2+y^2}\right)}{\partial \ddot{x}} & \frac{\partial\left(\sqrt{x^2+y^2}\right)}{\partial y} & \frac{\partial\left(\sqrt{x^2+y^2}\right)}{\partial \dot{y}} & \frac{\partial\left(\sqrt{x^2+y^2}\right)}{\partial \ddot{y}} \\ \frac{\partial\left(tan^{-1}\frac{y}{x}\right)}{\partial x} & \frac{\partial\left(tan^{-1}\frac{y}{x}\right)}{\partial \dot{x}} & \frac{\partial\left(tan^{-1}\frac{y}{x}\right)}{\partial \ddot{x}} & \frac{\partial\left(tan^{-1}\frac{y}{x}\right)}{\partial y} & \frac{\partial\left(tan^{-1}\frac{y}{x}\right)}{\partial \dot{y}} & \frac{\partial\left(tan^{-1}\frac{y}{x}\right)}{\partial \ddot{y}} \end{bmatrix} \tag{13.68}$$

I derived the partial derivatives earlier when I explained the multivariate uncertainty projection concept (subsection 13.5.1).

$$\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{x}{\sqrt{x^2+y^2}} & 0 & 0 & \frac{y}{\sqrt{x^2+y^2}} & 0 & 0 \\ \frac{-y}{x^2+y^2} & 0 & 0 & \frac{x}{x^2+y^2} & 0 & 0 \end{bmatrix} \tag{13.69}$$

## The measurement uncertainty

The measurement covariance matrix is:

$$\boldsymbol{R}_n = \begin{bmatrix} \sigma_{r_m}^2 & 0 \\ 0 & \sigma_{\varphi_m}^2 \end{bmatrix} \tag{13.70}$$

## The Kalman Gain

The Kalman Gain in for EKF is given by:

$$\boldsymbol{K}_n = \boldsymbol{P}_{n,n-1} \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}^T \left( \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}} \boldsymbol{P}_{n,n-1} \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}^T + \boldsymbol{R}_n \right)^{-1} \tag{13.71}$$

Where:

| | |
|---|---|
| $\boldsymbol{K}_n$ | is the Kalman Gain |
| $\boldsymbol{P}_{n,n-1}$ | is a prior estimate covariance matrix of the current state (predicted at the previous state) |
| $\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}$ | is the linearized Observation Function |
| $\boldsymbol{R}_n$ | is the measurement noise covariance matrix |

## The state update equation

The State Update Equation is given by:

$$\hat{\boldsymbol{x}}_{n,n} = \hat{\boldsymbol{x}}_{n,n-1} + \boldsymbol{K}_n \left( \boldsymbol{z}_n - \boldsymbol{h}(\hat{\boldsymbol{x}}_{n,n-1}) \right) \tag{13.72}$$

Where:

| | |
|---|---|
| $\hat{\boldsymbol{x}}_{n,n}$ | is the estimated system state vector at time step $n$ |
| $\hat{\boldsymbol{x}}_{n,n-1}$ | is the predicted system state vector at time step $n-1$ |
| $\boldsymbol{K}_n$ | is the Kalman Gain |
| $\boldsymbol{h}(\hat{\boldsymbol{x}}_{n,n-1})$ | is the Observation Function |

### The covariance update equation

The Covariance Update Equation in a matrix form is given by:

$$\boldsymbol{P}_{n,n} = \left(\boldsymbol{I} - \boldsymbol{K}_n \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}\right) \boldsymbol{P}_{n,n-1} \left(\boldsymbol{I} - \boldsymbol{K}_n \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}\right)^T + \boldsymbol{K}_n \boldsymbol{R}_n \boldsymbol{K}_n^T \tag{13.73}$$

Where:

| | |
|---|---|
| $\boldsymbol{P}_{n,n}$ | is the estimate covariance matrix of the current state |
| $\boldsymbol{P}_{n,n-1}$ | is the prior estimate covariance matrix of the current state (predicted at the previous state) |
| $\boldsymbol{K}_n$ | is the Kalman Gain |
| $\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}$ | is the linearized Observation Function |
| $\boldsymbol{R}_n$ | is the measurement noise covariance matrix |

## 13.7.2 The numerical example

The vehicle trajectory is similar to example 9 (section 9.1). The vehicle moves straight in the Y direction with a constant velocity. After traveling 400 meters, the vehicle turns left with a turning radius of 300 meters. During the turning maneuver, the vehicle experiences acceleration due to the circular motion (angular acceleration).

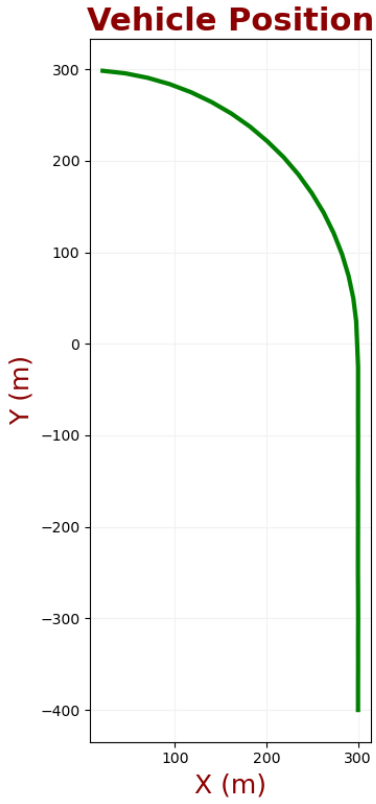


**Figure 13.6:** *Vehicle trajectory.*

- The measurements period: $\Delta t = 1s$
- The random acceleration standard deviation: $\sigma_a = 0.2\frac{m}{s^2}$
- The range measurement error standard deviation: $\sigma_{r_m} = 5m$
- The bearing angle measurement error standard deviation: $\sigma_{\varphi_m} = 0.0087rad$
- The state transition matrix $\boldsymbol{F}$ is:

$$\boldsymbol{F} = \begin{bmatrix} 1 & \Delta t & 0.5\Delta t^2 & 0 & 0 & 0 \\ 0 & 1 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta t & 0.5\Delta t^2 \\ 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0.5 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0.5 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

- The process noise matrix $\boldsymbol{Q}$ is:

$$\boldsymbol{Q} = \begin{bmatrix} \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} & \frac{\Delta t^2}{2} & 0 & 0 & 0 \\ \frac{\Delta t^3}{2} & \Delta t^2 & \Delta t & 0 & 0 & 0 \\ \frac{\Delta t^2}{2} & \Delta t & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} & \frac{\Delta t^2}{2} \\ 0 & 0 & 0 & \frac{\Delta t^3}{2} & \Delta t^2 & \Delta t \\ 0 & 0 & 0 & \frac{\Delta t^2}{2} & \Delta t & 1 \end{bmatrix} \sigma_a^2 = \begin{bmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & 1 & 1 & 0 & 0 & 0 \\ \frac{1}{2} & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{4} & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & \frac{1}{2} & 1 & 1 \\ 0 & 0 & 0 & \frac{1}{2} & 1 & 1 \end{bmatrix} 0.2^2$$

- The measurement variance $\boldsymbol{R}$ is:

$$\boldsymbol{R}_n = \begin{bmatrix} \sigma_{r_m}^2 & 0 \\ 0 & \sigma_{\varphi_m}^2 \end{bmatrix} = \begin{bmatrix} 5^2 & 0 \\ 0 & 0.0087^2 \end{bmatrix}$$

The following table contains the set of 35 noisy measurements:

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| **r(m)** | 502.55 | 477.34 | 457.21 | 442.94 | 427.27 | 406.05 | 400.73 | 377.32 |
| **$\varphi$(rad)** | -0.9316 | -0.8977 | -0.8512 | -0.8114 | -0.7853 | -0.7392 | -0.7052 | -0.6478 |
| **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** | **17** |
| 360.27 | 345.93 | 333.34 | 328.07 | 315.48 | 301.41 | 302.87 | 304.25 | 294.46 |
| -0.59 | -0.5183 | -0.4698 | -0.3952 | -0.3026 | -0.2445 | -0.1626 | -0.0937 | 0.0085 |
| **18** | **19** | **20** | **21** | **22** | **23** | **24** | **25** | **26** |
| 294.29 | 299.38 | 299.37 | 300.68 | 304.1 | 301.96 | 300.3 | 301.9 | 296.7 |
| 0.0856 | 0.1675 | 0.2467 | 0.329 | 0.4149 | 0.504 | 0.5934 | 0.667 | 0.7537 |
| **27** | **28** | **29** | **30** | **31** | **32** | **33** | **34** | **35** |
| 297.07 | 295.29 | 296.31 | 300.62 | 292.3 | 298.11 | 298.07 | 298.92 | 298.04 |
| 0.8354 | 0.9195 | 1.0039 | 1.0923 | 1.1546 | 1.2564 | 1.3274 | 1.409 | 1.5011 |

**Table 13.2:** *Example 11 measurements.*

#### 13.7.2.1 Iteration Zero

##### Initialization

We don't know the vehicle location, so we approximate the initial position at about 100m from the true vehicle position ($\hat{x}_{0,0} = 400m, \hat{y}_{0,0} = -300m$)

$$\hat{\boldsymbol{x}}_{0,0} = \begin{bmatrix} 400 \\ 0 \\ 0 \\ -300 \\ 0 \\ 0 \end{bmatrix}$$

Since our initial state vector is a guess, we set a very high estimate uncertainty. The high estimate uncertainty results in a high Kalman Gain by giving a high weight to the measurement.

$$\boldsymbol{P}_{0,0} = \begin{bmatrix} 500 & 0 & 0 & 0 & 0 & 0 \\ 0 & 500 & 0 & 0 & 0 & 0 \\ 0 & 0 & 500 & 0 & 0 & 0 \\ 0 & 0 & 0 & 500 & 0 & 0 \\ 0 & 0 & 0 & 0 & 500 & 0 \\ 0 & 0 & 0 & 0 & 0 & 500 \end{bmatrix}$$

#### Prediction

$$\hat{\boldsymbol{x}}_{1,0} = \boldsymbol{F}\hat{\boldsymbol{x}}_{0,0} = \begin{bmatrix} 400 \\ 0 \\ 0 \\ -300 \\ 0 \\ 0 \end{bmatrix}$$

$$\boldsymbol{P}_{1,0} = \boldsymbol{F}\boldsymbol{P}_{0,0}\boldsymbol{F}^T + \boldsymbol{Q} = \begin{bmatrix} 1125 & 750 & 250 & 0 & 0 & 0 \\ 750 & 1000 & 500 & 0 & 0 & 0 \\ 250 & 500 & 500 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1125 & 750 & 250 \\ 0 & 0 & 0 & 750 & 1000 & 500 \\ 0 & 0 & 0 & 250 & 500 & 500 \end{bmatrix}$$

### 13.7.2.2 First Iteration

#### Step 1 - Measure

The measurement values:

$$\boldsymbol{z}_1 = \begin{bmatrix} 502.55 \\ -0.9316 \end{bmatrix}$$

#### Step 2 - Update

Observation matrix ($\boldsymbol{h}(\hat{\boldsymbol{x}}_{1,0})$) calculation.

$$\boldsymbol{h}(\hat{\boldsymbol{x}}_{1,0}) = \begin{bmatrix} \sqrt{x_{1,0}^2 + y_{1,0}^2} \\ tan^{-1}\dfrac{y_{1,0}}{x_{1,0}} \end{bmatrix} = \begin{bmatrix} \sqrt{400^2 + (-300)^2} \\ tan^{-1}\dfrac{-300}{400} \end{bmatrix} = \begin{bmatrix} 500 \\ 0.644 \end{bmatrix}$$

Observation matrix Jacobian $\left(\dfrac{\partial \boldsymbol{h}(\hat{\boldsymbol{x}}_{1,0})}{\partial \boldsymbol{x}}\right)$ calculation:

$$\frac{\partial \boldsymbol{h}(\hat{\boldsymbol{x}}_{1,0})}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{x_{1,0}}{\sqrt{x_{1,0}^2 + y_{1,0}^2}} & 0 & 0 & \frac{y_{1,0}}{\sqrt{x_{1,0}^2 + y_{1,0}^2}} & 0 & 0 \\ \frac{-y_{1,0}}{x_{1,0}^2 + y_{1,0}^2} & 0 & 0 & \frac{x_{1,0}}{x_{1,0}^2 + y_{1,0}^2} & 0 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0.8 & 0 & 0 & -0.6 & 0 & 0 \\ 0.0012 & 0 & 0 & 0.0016 & 0 & 0 \end{bmatrix}$$

The Kalman Gain calculation:

$$\boldsymbol{K}_1 = \boldsymbol{P}_{1,0} \left( \frac{\partial \boldsymbol{h}(\hat{\boldsymbol{x}}_{1,0})}{\partial \boldsymbol{x}} \right)^T \left( \frac{\partial \boldsymbol{h}(\hat{\boldsymbol{x}}_{1,0})}{\partial \boldsymbol{x}} \boldsymbol{P}_{1,0} \left( \frac{\partial \boldsymbol{h}(\hat{\boldsymbol{x}}_{1,0})}{\partial \boldsymbol{x}} \right)^T + \boldsymbol{R} \right)^{-1} = \begin{bmatrix} 0.783 & 295 \\ 0.522 & 196.7 \\ 0.174 & 65.6 \\ -0.587 & 393.3 \\ -0.391 & 262.2 \\ -0.13 & 87.4 \end{bmatrix}$$

Estimate the current state:

$$\hat{\boldsymbol{x}}_{1,1} = \hat{\boldsymbol{x}}_{1,0} + \boldsymbol{K}_1 \left( \boldsymbol{z}_1 - \boldsymbol{h}(\hat{\boldsymbol{x}}_{1,0}) \right) = \begin{bmatrix} 317 \\ -55.3 \\ -18.4 \\ -414.8 \\ -76.5 \\ -25.5 \end{bmatrix}$$

Update the current estimate covariance:

$$\boldsymbol{P}_{1,1} = \left( \boldsymbol{I} - \boldsymbol{K}_1 \frac{\partial \boldsymbol{h}(\hat{\boldsymbol{x}}_{1,0})}{\partial \boldsymbol{x}} \right) \boldsymbol{P}_{1,0} \left( \boldsymbol{I} - \boldsymbol{K}_1 \frac{\partial \boldsymbol{h}(\hat{\boldsymbol{x}}_{1,0})}{\partial \boldsymbol{x}} \right)^T + \boldsymbol{K}_1 \boldsymbol{R} \boldsymbol{K}_1^T$$

$$= \begin{bmatrix} 22.39 & 14.93 & 4.98 & -2.75 & -1.84 & -0.61 \\ 14.93 & 509.97 & 336.67 & -1.84 & -1.22 & -0.41 \\ 4.98 & 336.67 & 445.58 & -0.61 & -0.41 & -0.14 \\ -2.75 & -1.84 & -0.61 & 22.39 & 14.93 & 4.98 \\ -1.84 & -1.22 & -0.41 & 14.93 & 509.97 & 336.67 \\ -0.61 & -0.41 & -0.14 & 4.98 & 336.67 & 445.58 \end{bmatrix}$$

#### Step 3 - Predict

$$\hat{\boldsymbol{x}}_{2,1} = \boldsymbol{F}\hat{\boldsymbol{x}}_{1,1} = \begin{bmatrix} 252.42 \\ -73.8 \\ -18.45 \\ -504.13 \\ -102.06 \\ -25.52 \end{bmatrix}$$

$$\boldsymbol{P}_{2,1} = \boldsymbol{F}\boldsymbol{P}_{1,1}\boldsymbol{F}^T + \boldsymbol{Q} = \begin{bmatrix} 1015.28 & 1257.7 & 564.46 & -8.7 & -4.35 & -1.09 \\ 1257.7 & 1628.94 & 782.3 & -4.35 & -2.18 & -0.54 \\ 564.46 & 782.3 & 445.62 & -1.09 & -0.54 & -0.14 \\ -8.7 & -4.35 & -1.09 & 1010.2 & 1255.16 & 563.83 \\ -4.35 & -2.18 & -0.54 & 1255.16 & 1627.67 & 781.98 \\ -1.09 & -0.54 & -0.14 & 563.83 & 781.98 & 445.54 \end{bmatrix}$$

### 13.7.2.3 Second Iteration

#### Step 1 - Measure

The measurement values:

$$\boldsymbol{z}_2 = \begin{bmatrix} 477.34 \\ -0.8977 \end{bmatrix}$$

#### Step 2 - Update

Observation matrix ($\boldsymbol{h}(\hat{\boldsymbol{x}}_{2,1})$) calculation.

$$\boldsymbol{h}(\hat{\boldsymbol{x}}_{2,1}) = \begin{bmatrix} \sqrt{x_{2,1}^2 + y_{2,1}^2} \\ tan^{-1}\dfrac{y_{2,1}}{x_{2,1}} \end{bmatrix} = \begin{bmatrix} \sqrt{252.42^2 + (-504.13)^2} \\ tan^{-1}\dfrac{-504.13}{252.42} \end{bmatrix} = \begin{bmatrix} 563.8 \\ -1.1 \end{bmatrix}$$

Observation matrix Jacobian $\left(\dfrac{\partial \boldsymbol{h}(\hat{\boldsymbol{x}}_{2,1})}{\partial \boldsymbol{x}}\right)$ calculation:

$$\frac{\partial \boldsymbol{h}(\hat{\boldsymbol{x}}_{2,1})}{\partial \boldsymbol{x}} = \begin{bmatrix} \dfrac{x_{2,1}}{\sqrt{x_{2,1}^2 + y_{2,1}^2}} & 0 & 0 & \dfrac{y_{2,1}}{\sqrt{x_{2,1}^2 + y_{2,1}^2}} & 0 & 0 \\ \dfrac{-y_{2,1}}{x_{2,1}^2 + y_{2,1}^2} & 0 & 0 & \dfrac{x_{2,1}}{x_{2,1}^2 + y_{2,1}^2} & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0.45 & 0 & 0 & -0.89 & 0 & 0 \\ 0.0016 & 0 & 0 & 0.0008 & 0 & 0 \end{bmatrix}$$

The Kalman Gain calculation:

$$\boldsymbol{K}_2 = \boldsymbol{P}_{2,1}\left(\frac{\partial \boldsymbol{h}(\hat{\boldsymbol{x}}_{2,1})}{\partial \boldsymbol{x}}\right)^T\left(\frac{\partial \boldsymbol{h}(\hat{\boldsymbol{x}}_{2,1})}{\partial \boldsymbol{x}}\boldsymbol{P}_{2,1}\left(\frac{\partial \boldsymbol{h}(\hat{\boldsymbol{x}}_{2,1})}{\partial \boldsymbol{x}}\right)^T + \boldsymbol{R}\right)^{-1} = \begin{bmatrix} 0.44 & 492.34 \\ 0.54 & 611.49 \\ 0.24 & 274.65 \\ -0.87 & 246.4 \\ -1.08 & 309.3 \\ -0.49 & 139.36 \end{bmatrix}$$

Estimate the current state:

$$\hat{\boldsymbol{x}}_{2,2} = \hat{\boldsymbol{x}}_{2,1} + \boldsymbol{K}_2\left(\boldsymbol{z}_2 - \boldsymbol{h}(\hat{\boldsymbol{x}}_{2,1})\right) = \begin{bmatrix} 317.47 \\ 7.6 \\ 18.19 \\ -377.14 \\ 56.13 \\ 45.6 \end{bmatrix}$$

Update the current estimate covariance:

$$\boldsymbol{P}_{2,2} = \left(\boldsymbol{I} - \boldsymbol{K}_2\frac{\partial \boldsymbol{h}(\hat{\boldsymbol{x}}_{2,1})}{\partial \boldsymbol{x}}\right)\boldsymbol{P}_{2,1}\left(\boldsymbol{I} - \boldsymbol{K}_2\frac{\partial \boldsymbol{h}(\hat{\boldsymbol{x}}_{2,1})}{\partial \boldsymbol{x}}\right)^T + \boldsymbol{K}_2\boldsymbol{R}\boldsymbol{K}_2^T$$

$$= \begin{bmatrix} 23.8 & 29.48 & 13.23 & -0.31 & -0.23 & -0.08 \\ 29.48 & 107.4 & 99.42 & -0.23 & -4.87 & -2.8 \\ 13.23 & 99.42 & 139.14 & -0.08 & -2.8 & -1.62 \\ -0.31 & -0.23 & -0.08 & 24.24 & 30.12 & 13.53 \\ -0.23 & -4.87 & -2.8 & 30.12 & 105.54 & 98.22 \\ -0.08 & -2.8 & -1.62 & 13.53 & 98.22 & 138.39 \end{bmatrix}$$

**Step 3 - Predict**

$$\hat{\boldsymbol{x}}_{3,2} = \boldsymbol{F}\hat{\boldsymbol{x}}_{2,2} = \begin{bmatrix} 334.17 \\ 25.8 \\ 18.19 \\ -298.21 \\ 101.73 \\ 45.6 \end{bmatrix}$$

$$\boldsymbol{P}_{3,2} = \boldsymbol{F}\boldsymbol{P}_{2,2}\boldsymbol{F}^T + \boldsymbol{Q} = \begin{bmatrix} 337.6 & 368.83 & 182.2 & -8.92 & -10.19 & -3.69 \\ 368.83 & 445.42 & 238.6 & -10.19 & -12.09 & -4.42 \\ 182.2 & 238.6 & 139.18 & -3.69 & -4.42 & -1.62 \\ -8.92 & -10.19 & -3.69 & 336.39 & 365.74 & 180.97 \\ -10.19 & -12.09 & -4.42 & 365.74 & 440.41 & 236.65 \\ -3.69 & -4.42 & -1.62 & 180.97 & 236.65 & 138.43 \end{bmatrix}$$

At this point, I think it would be reasonable to jump to the last Kalman Filter iteration.

#### 13.7.2.4 Thirty-Fifth Iteration

##### Step 1 - Measure

The measurement values:

$$\boldsymbol{z}_{35} = \begin{bmatrix} 298.04 \\ 1.5011 \end{bmatrix}$$

##### Step 2 - Update

Observation matrix ($\boldsymbol{h}(\hat{\boldsymbol{x}}_{35,34})$) calculation.

$$\boldsymbol{h}(\hat{\boldsymbol{x}}_{35,34}) = \begin{bmatrix} \sqrt{x_{35,34}^2 + y_{35,34}^2} \\ tan^{-1}\dfrac{y_{35,34}}{x_{35,34}} \end{bmatrix} = \begin{bmatrix} \sqrt{307.8^2 + (-277.04)^2} \\ tan^{-1}\dfrac{-277.04}{307.8} \end{bmatrix} = \begin{bmatrix} 300.07 \\ 1.5 \end{bmatrix}$$

Observation matrix Jacobian $\left(\dfrac{\partial \boldsymbol{h}(\hat{\boldsymbol{x}}_{35,34})}{\partial \boldsymbol{x}}\right)$ calculation:

$$\left(\frac{\partial \boldsymbol{h}(\hat{\boldsymbol{x}}_{35,34})}{\partial \boldsymbol{x}}\right) = \begin{bmatrix} \dfrac{x_{35,34}}{\sqrt{x_{35,34}^2 + y_{35,34}^2}} & 0 & 0 & \dfrac{y_{35,34}}{\sqrt{x_{35,34}^2 + y_{35,34}^2}} & 0 & 0 \\ \dfrac{-y_{35,34}}{x_{35,34}^2 + y_{35,34}^2} & 0 & 0 & \dfrac{x_{35,34}}{x_{35,34}^2 + y_{35,34}^2} & 0 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0.07 & 0 & 0 & 0.998 & 0 & 0 \\ -0.003 & 0 & 0 & 0.0002 & 0 & 0 \end{bmatrix}$$

The Kalman Gain calculation:

$$\boldsymbol{K}_{35} = \boldsymbol{P}_{35,34}\left(\frac{\partial \boldsymbol{h}(\hat{\boldsymbol{x}}_{35,34})}{\partial \boldsymbol{x}}\right)^T\left(\frac{\partial \boldsymbol{h}(\hat{\boldsymbol{x}}_{35,34})}{\partial \boldsymbol{x}}\boldsymbol{P}_{35,34}\left(\frac{\partial \boldsymbol{h}(\hat{\boldsymbol{x}}_{35,34})}{\partial \boldsymbol{x}}\right)^T + \boldsymbol{R}\right)^{-1} = \begin{bmatrix} 0.04 & -174.67 \\ 0.01 & -73.57 \\ 0 & -15.06 \\ 0.5 & -4.83 \\ 0.16 & -1.32 \\ 0.03 & -0.3 \end{bmatrix}$$

Estimate the current state:

$$\hat{\boldsymbol{x}}_{35,35} = \hat{\boldsymbol{x}}_{35,34} + \boldsymbol{K}_{35}\left(\boldsymbol{z}_{35} - \boldsymbol{h}(\hat{\boldsymbol{x}}_{35,34})\right) = \begin{bmatrix} 20.87 \\ -25.93 \\ -0.84 \\ 298.38 \\ 2.55 \\ -1.8 \end{bmatrix}$$

Update the current estimate covariance:

$$\boldsymbol{P}_{35,35} = \left(\boldsymbol{I} - \boldsymbol{K}_{35}\frac{\partial \boldsymbol{h}(\hat{\boldsymbol{x}}_{35,34})}{\partial \boldsymbol{x}}\right)\boldsymbol{P}_{35,34}\left(\boldsymbol{I} - \boldsymbol{K}_{35}\frac{\partial \boldsymbol{h}(\hat{\boldsymbol{x}}_{35,34})}{\partial \boldsymbol{x}}\right)^T + \boldsymbol{K}_{35}\boldsymbol{R}\boldsymbol{K}_{35}^T$$

$$= \begin{bmatrix} 4.07 & 1.7 & 0.34 & 0.95 & 0.31 & 0.04 \\ 1.7 & 1.3 & 0.38 & 0.16 & 0.2 & 0.04 \\ 0.34 & 0.38 & 0.16 & -0.01 & 0.03 & 0.01 \\ 0.95 & 0.16 & -0.01 & 12.02 & 4.05 & 0.7 \\ 0.31 & 0.2 & 0.03 & 4.05 & 2.29 & 0.56 \\ 0.04 & 0.04 & 0.01 & 0.7 & 0.56 & 0.19 \end{bmatrix}$$

### Step 3 - Predict

$$\hat{\boldsymbol{x}}_{36,35} = \boldsymbol{F}\hat{\boldsymbol{x}}_{35,35} = \begin{bmatrix} -5.49 \\ -26.77 \\ -0.84 \\ 300.02 \\ 0.74 \\ -1.8 \end{bmatrix}$$

$$\boldsymbol{P}_{36,35} = \boldsymbol{F}\boldsymbol{P}_{35,35}\boldsymbol{F}^T + \boldsymbol{Q} = \begin{bmatrix} 9.53 & 4 & 0.82 & 1.68 & 0.61 & 0.09 \\ 4 & 2.25 & 0.57 & 0.41 & 0.27 & 0.05 \\ 0.82 & 0.57 & 0.2 & 0.02 & 0.04 & 0.01 \\ 1.68 & 0.41 & 0.02 & 23.74 & 8 & 1.38 \\ 0.61 & 0.27 & 0.04 & 8 & 3.63 & 0.79 \\ 0.09 & 0.05 & 0.01 & 1.38 & 0.79 & 0.23 \end{bmatrix}$$

### 13.7.3 Example summary

The following chart demonstrates the EKF location and velocity estimation performance.

The chart on the left compares the true, measured, and estimated values of the vehicle position. Two charts on the right compare the true, measured, and estimated values of $x$ - axis velocity and $y$ - axis velocity.

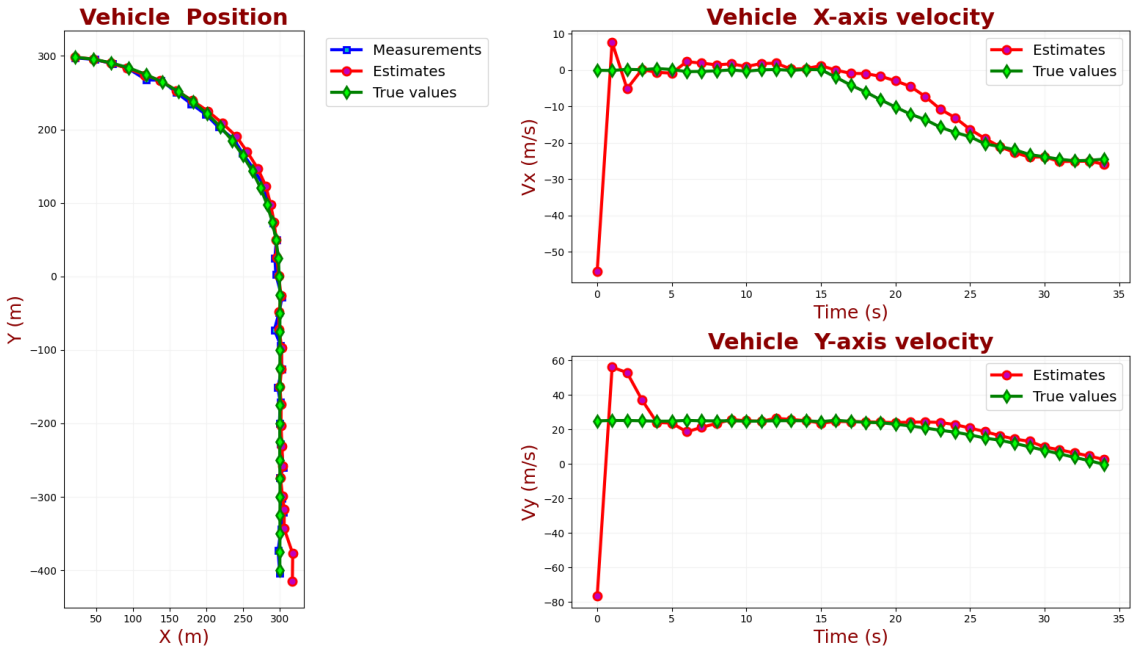


**Figure 13.7:** *Example 11: true value, measured values and estimates.*

We can see a satisfying performance of the EKF. Although the filter is roughly initiated at about 100 meters from the true position with zero initial velocity, it provides a good position estimation after taking two measurements and a good velocity estimation after taking four measurements.

Let us take a closer look at the vehicle position estimation performance. The following chart describes the true, measured, and estimated values of the vehicle position compared to the 95% confidence ellipses. We can see that the ellipses' size constantly decreases. That means that the EKF converges with time.

The following charts provide a zoom into the linear part of the vehicle motion and the turning maneuver part.

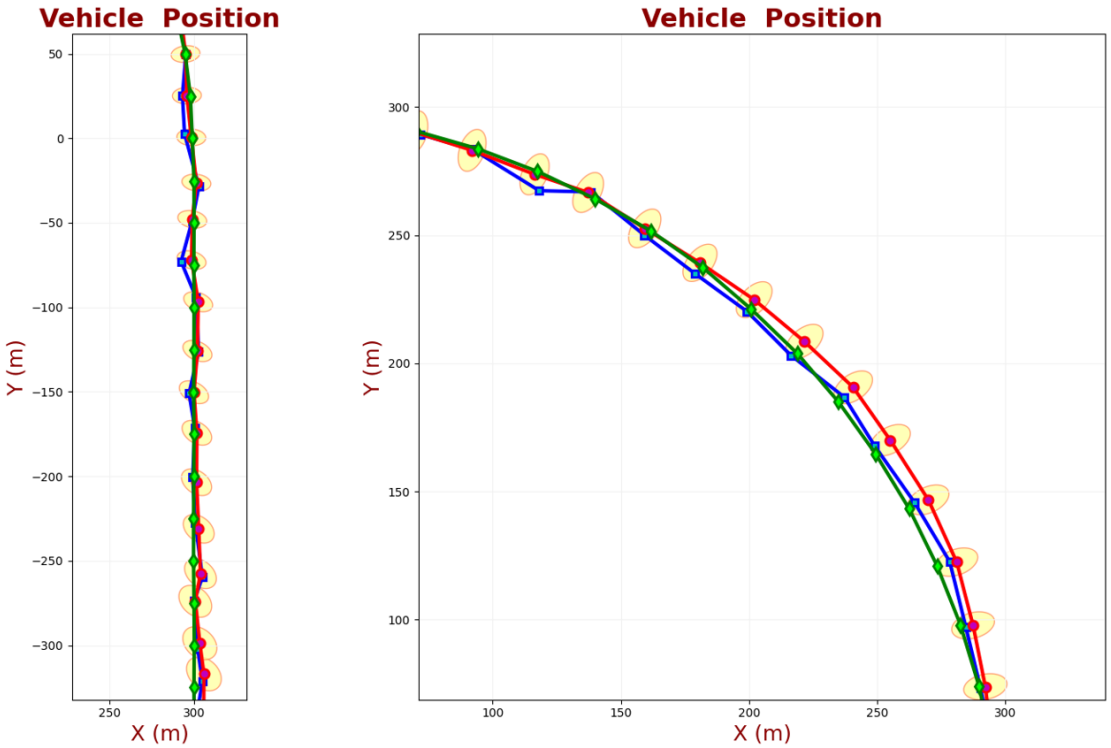


**Figure 13.8:** *Example 11: true value, measured values and estimates - zoom.*

We can see that at the linear part of the vehicle motion, the EKF copes with the noisy measurements and follows the true vehicle position. On the other hand, during the vehicle turning maneuver, the EKF estimates are quite away from the true vehicle position, although they are within the 90% confidence ellipse bounds.

## 13.8 Example 12 - estimating the pendulum angle

In this example, we estimate the pendulum angle $\theta$. The dynamic model of the pendulum was derived earlier (section 12.3) in this chapter. Assume an ideal gravity pendulum that consists of a body with mass m hung by a string with length $L$ from fixed support - the pendulum swings back and forth at a constant amplitude. We want to estimate the angle $\theta$ from the vertical to the pendulum. The angle units are radians.

We measure the pendulum position $z = Lsin(\theta_n)$.

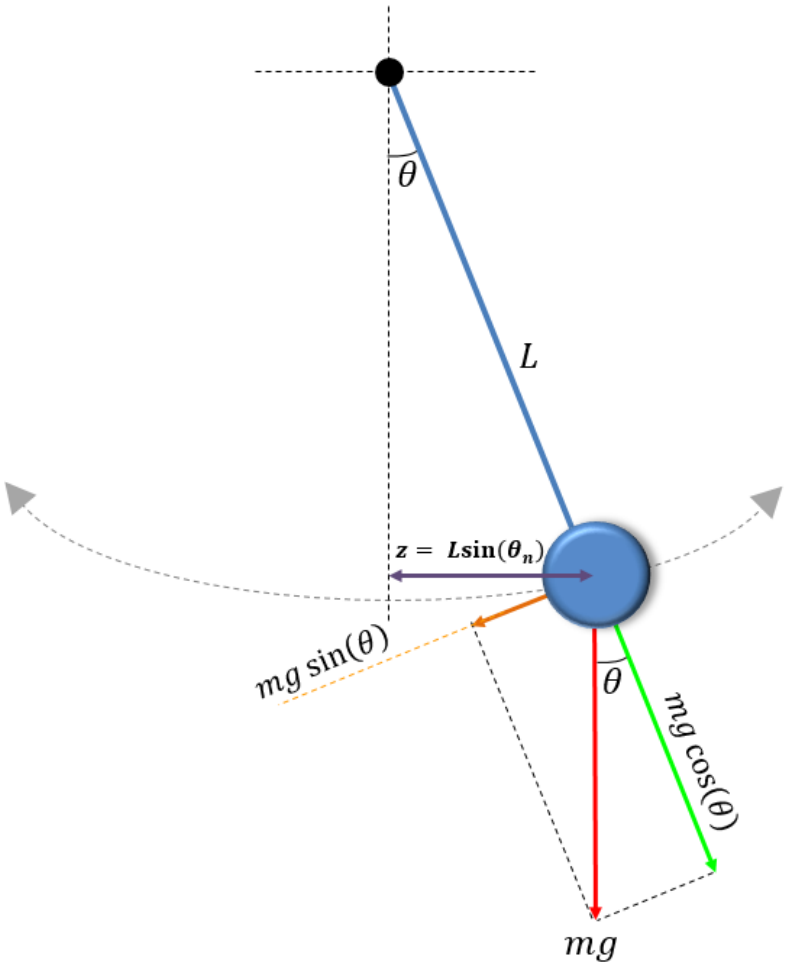


**Figure 13.9:** *Pendulum position measurement.*

### 13.8.1 Kalman Filter equations

#### The state extrapolation equation

The state vector of the pendulum is in the form of the following:

$$\boldsymbol{x}_n = \begin{bmatrix} \theta_n \\ \dot{\theta}_n \end{bmatrix} \tag{13.74}$$

Where:

- $\theta_n$ is the pendulum angle at time $n$
- $\dot{\theta}_n$ is the pendulum angular velocity at time $n$

As we've seen earlier, the dynamic model of the pendulum is non-linear (the second type of non-linearity). It has the form of:

$$\hat{\boldsymbol{x}}_{n+1,n} = \boldsymbol{f}(\hat{\boldsymbol{x}}_{n,n}) \tag{13.75}$$

$$\hat{\boldsymbol{x}}_{n+1,n} = \begin{bmatrix} \hat{\theta}_{n+1,n} \\ \hat{\dot{\theta}}_{n+1,n} \end{bmatrix} = \begin{bmatrix} \hat{\theta}_{n,n} + \hat{\dot{\theta}}_{n,n}\Delta t \\ \hat{\dot{\theta}}_{n,n} - \frac{g}{L} sin(\hat{\theta}_{n,n})\Delta t \end{bmatrix} \tag{13.76}$$

$$\boldsymbol{f}(\hat{\boldsymbol{x}}_{n,n}) = \begin{bmatrix} \hat{\theta}_{n,n} + \hat{\dot{\theta}}_{n,n}\Delta t \\ \hat{\dot{\theta}}_{n,n} - \frac{g}{L} sin(\hat{\theta}_{n,n})\Delta t \end{bmatrix} \tag{13.77}$$

## Jacobian derivation

We have already derived (section 12.3) the Jacobian for pendulum dynamic model:

$$\frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial \hat{\theta}} & \frac{\partial f_1}{\partial \hat{\dot{\theta}}} \\ \frac{\partial f_2}{\partial \hat{\theta}} & \frac{\partial f_2}{\partial \hat{\dot{\theta}}} \end{bmatrix} = \begin{bmatrix} \frac{\partial \left( \hat{\theta}_{n,n} + \hat{\dot{\theta}}_{n,n} \Delta t \right)}{\partial \hat{\theta}} & \frac{\partial \left( \hat{\theta}_{n,n} + \hat{\dot{\theta}}_{n,n} \Delta t \right)}{\partial \hat{\dot{\theta}}} \\ \frac{\partial \left( \hat{\dot{\theta}}_{n,n} - \frac{g}{L} sin(\hat{\theta}_{n,n}) \Delta t \right)}{\partial \hat{\theta}} & \frac{\partial \left( \hat{\dot{\theta}}_{n,n} - \frac{g}{L} sin(\hat{\theta}_{n,n}) \Delta t \right)}{\partial \hat{\dot{\theta}}} \end{bmatrix}$$

$$= \begin{bmatrix} 1 & \Delta t \\ -\frac{g}{L} cos(\hat{\theta}_{n,n}) \Delta t & 1 \end{bmatrix} \tag{13.78}$$

## The covariance extrapolation equation

The Covariance Extrapolation Equation is given by:

$$\boldsymbol{P}_{n+1,n} = \frac{\partial \boldsymbol{f}(\hat{\boldsymbol{x}}_{n,n})}{\partial \boldsymbol{x}} \boldsymbol{P}_{n,n} \frac{\partial \boldsymbol{f}(\hat{\boldsymbol{x}}_{n,n})}{\partial \boldsymbol{x}}^T + \boldsymbol{Q} \tag{13.79}$$

Where:

| | |
|---|---|
| $\boldsymbol{P}_{n,n}$ | is the estimate covariance matrix of the current state |
| $\boldsymbol{P}_{n+1,n}$ | is the predicted estimate covariance matrix for the next state |
| $\frac{\partial \boldsymbol{f}(\hat{\boldsymbol{x}}_{n,n})}{\partial \boldsymbol{x}}$ | is the linearized State Transition Matrix |
| $\boldsymbol{Q}$ | is the process noise matrix |

The estimate covariance is:

$$\boldsymbol{P} = \begin{bmatrix} p_x & p_{x\dot{x}} \\ p_{\dot{x}x} & p_{\dot{x}} \end{bmatrix} \tag{13.80}$$

The elements of the main diagonal of the matrix are the variances of the estimation:

- $p_x$ is the variance of the angle $\theta$ estimation
- $p_{\dot{x}}$ is the variance of the angular velocity $\dot{\theta}$ estimation

## The process noise matrix

$$\boldsymbol{Q} = \begin{bmatrix} \sigma_x^2 & \sigma_{x\dot{x}}^2 \\ \sigma_{\dot{x}x}^2 & \sigma_{\dot{x}}^2 \end{bmatrix} = \begin{bmatrix} \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} \\ \frac{\Delta t^3}{2} & \Delta t^2 \end{bmatrix} \sigma_a^2 \tag{13.81}$$

Where:

- $\Delta t$ is the time between successive measurements
- $\sigma_a^2$ is a random variance in acceleration

## The measurement equation

We measure the pendulum position: $Lsin\left(\theta_n\right)$.

Since the state-to-measurement relation (the first type of non-linearity) is non-linear, the measurement equation is a type of:

$$\boldsymbol{z}_n = \boldsymbol{h}\left(\boldsymbol{x}_n\right) \tag{13.82}$$

$$\boldsymbol{h}\left(\boldsymbol{x}_n\right) = Lsin\left(\theta_n\right) \tag{13.83}$$

## Jacobian derivation

We have already derived (section 12.3) the Jacobian for pendulum position measurement:

$$\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial\left(Lsin(\theta_n)\right)}{\partial \theta} & \frac{\partial\left(Lsin(\theta_n)\right)}{\partial \dot{\theta}} \end{bmatrix} = \begin{bmatrix} Lcos(\theta_n) & 0 \end{bmatrix} \tag{13.84}$$

## The measurement uncertainty

The measurement variance is:

$$\boldsymbol{R}_n = \begin{bmatrix} \sigma_{x_m}^2 \end{bmatrix} = r \tag{13.85}$$

## The Kalman Gain

The Kalman Gain in for EKF is given by:

$$\boldsymbol{K}_n = \boldsymbol{P}_{n,n-1}\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}^T \left(\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}\boldsymbol{P}_{n,n-1}\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}^T + \boldsymbol{R}_n\right)^{-1} \tag{13.86}$$

Where:

| | |
|---|---|
| $\boldsymbol{K}_n$ | is the Kalman Gain |
| $\boldsymbol{P}_{n,n-1}$ | is a prior estimate covariance matrix of the current state (predicted at the previous state) |
| $\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}$ | is the linearized Observation Matrix |
| $\boldsymbol{R}_n$ | is the measurement noise covariance matrix |

## The state update equation

The State Update Equation is given by:

$$\hat{\boldsymbol{x}}_{n,n} = \hat{\boldsymbol{x}}_{n,n-1} + \boldsymbol{K}_n\left(\boldsymbol{z}_n - \boldsymbol{h}(\hat{\boldsymbol{x}}_{n,n-1})\right) \tag{13.87}$$

Where:

| | |
|---|---|
| $\hat{\boldsymbol{x}}_{n,n}$ | is the estimated system state vector at time step $n$ |
| $\hat{\boldsymbol{x}}_{n,n-1}$ | is the predicted system state vector at time step $n-1$ |
| $\boldsymbol{K}_n$ | is the Kalman Gain |
| $\boldsymbol{h}(\hat{\boldsymbol{x}}_{n,n-1})$ | is the linearized Observation Matrix |

### The covariance update equation

The Covariance Update Equation in a matrix form is given by:

$$\boldsymbol{P}_{n,n} = \left(\boldsymbol{I} - \boldsymbol{K}_n \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}\right) \boldsymbol{P}_{n,n-1} \left(\boldsymbol{I} - \boldsymbol{K}_n \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}\right)^T + \boldsymbol{K}_n \boldsymbol{R}_n \boldsymbol{K}_n^T \tag{13.88}$$

Where:

| | |
|---|---|
| $\boldsymbol{P}_{n,n}$ | is the estimate covariance matrix of the current state |
| $\boldsymbol{P}_{n,n-1}$ | is the prior estimate covariance matrix of the current state (predicted at the previous state) |
| $\boldsymbol{K}_n$ | is the Kalman Gain |
| $\boldsymbol{h}(\hat{\boldsymbol{x}}_{n,n-1})$ | is the linearized Observation Matrix |
| $\boldsymbol{R}_n$ | is the measurement noise covariance matrix |

## 13.8.2 The numerical example

The example parameters:

- The Pendulum string length: $L = 0.5m$
- Gravitational acceleration constant: $g = 9.8\frac{m}{s^2}$
- Measurement Uncertainty (standard deviation): $\sigma_{x_m} = 0.01m$
- Process Noise Uncertainty (angular acceleration standard deviation): $\sigma_a = 1\frac{rad}{s^2}$

The following chart depicts the true angle and angular velocity vs. time (including the process noise).

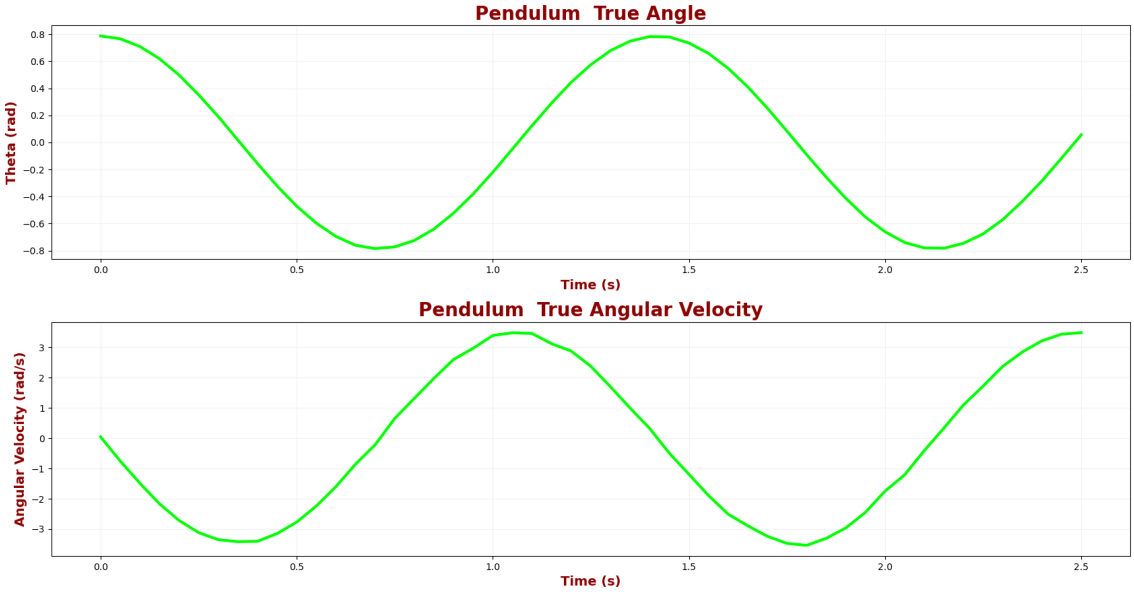


**Figure 13.10:** *Pendulum true position and velocity.*

If you are curious about pendulum motion simulation, you can find mathematical

derivations in Appendix E.

The process noise matrix $\boldsymbol{Q}$ is:

$$\boldsymbol{Q} = \begin{bmatrix} \sigma_x^2 & \sigma_{x\dot{x}}^2 \\ \sigma_{\dot{x}x}^2 & \sigma_{\dot{x}}^2 \end{bmatrix} = \begin{bmatrix} \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} \\ \frac{\Delta t^3}{2} & \Delta t^2 \end{bmatrix} \sigma_a^2 \tag{13.89}$$

The measurement variance $\boldsymbol{R}$ is:

$$\boldsymbol{R}_n = \sigma_{x_m}^2 = 0.01^2 \tag{13.90}$$

The following table contains the set of the first 10 noisy measurements:

| | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| $\boldsymbol{Lsin(\theta)}$ | 0.119 | 0.113 | 0.12 | 0.101 | 0.099 | 0.063 | 0.008 | -0.017 | -0.037 | -0.05 |

**Table 13.3:** *Example 12 measurements.*

#### 13.8.2.1 Iteration Zero

##### Initialization

We don't know the pendulum angle, so our initial angle approximation includes an error. We set the initial velocity to 0.

$$\hat{\boldsymbol{x}}_{0,0} = \begin{bmatrix} 0.0873 \\ 0 \end{bmatrix}$$

Since our initial state vector is a guess, we set a high estimate uncertainty. The high estimate uncertainty results in a high Kalman Gain by giving high weight to the measurement.

$$\boldsymbol{P}_{0,0} = \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix}$$

##### Prediction

In order to predict $\hat{\boldsymbol{x}}_{1,0}$ we need to find find the $\hat{\boldsymbol{x}}_{1,0} = \boldsymbol{f}(\hat{\boldsymbol{x}}_{0,0})$:

$$\hat{\boldsymbol{x}}_{1,0} = \boldsymbol{f}(\hat{\boldsymbol{x}}_{0,0}) = \begin{bmatrix} \hat{\theta}_{0,0} + \hat{\dot{\theta}}_{0,0}\Delta t \\ \hat{\dot{\theta}}_{0,0} - \frac{g}{L} sin(\hat{\theta}_{0,0})\Delta t \end{bmatrix} = \begin{bmatrix} 0.0873 + 0 \cdot 0.05 \\ 0 - \frac{9.8}{0.5} sin(0.0873) \cdot 0.05 \end{bmatrix} = \begin{bmatrix} 0.08735 \\ -0.0854 \end{bmatrix}$$

Jacobian $\frac{\partial \boldsymbol{f}(\hat{\boldsymbol{x}}_{0,0})}{\partial \boldsymbol{x}}$ calculation:

$$\frac{\partial \boldsymbol{f}(\hat{\boldsymbol{x}}_{0,0})}{\partial \boldsymbol{x}} = \begin{bmatrix} 1 & \Delta t \\ -\frac{g}{L} cos(\hat{\theta}_{n,n}) \Delta t & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0.05 \\ -\frac{9.8}{0.5} cos(0.0873) \cdot 0.05 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0.05 \\ -0.9763 & 1 \end{bmatrix}$$

Uncertainty propagation:

$$\boldsymbol{P}_{1,0} = \frac{\partial \boldsymbol{f}(\hat{\boldsymbol{x}}_{0,0})}{\partial \boldsymbol{x}} \boldsymbol{P}_{0,0} \left( \frac{\partial \boldsymbol{f}(\hat{\boldsymbol{x}}_{0,0})}{\partial \boldsymbol{x}} \right)^T + \boldsymbol{Q} = \begin{bmatrix} 5.013 & -4.631 \\ -4.631 & 9.768 \end{bmatrix}$$

#### 13.8.2.2 First Iteration

##### Step 1 - Measure

The measurement values:

$$\boldsymbol{z}_1 = 0.119$$

##### Step 2 - Update

Observation matrix ($\boldsymbol{h}(\hat{\boldsymbol{x}}_{1,0})$) calculation.

$$\boldsymbol{h}(\hat{\boldsymbol{x}}_{1,0}) = L sin(\theta_{1,0}) = 0.5 sin(0.0873) = 0.0436$$

Observation matrix Jacobian $\left( \frac{\partial \boldsymbol{h}(\hat{\boldsymbol{x}}_{1,0})}{\partial \boldsymbol{x}} \right)$ calculation:

$$\frac{\partial \boldsymbol{h}(\hat{\boldsymbol{x}}_{1,0})}{\partial \boldsymbol{x}} = \begin{bmatrix} L cos(\theta_{1,0}) & 0 \end{bmatrix} = \begin{bmatrix} 0.5 cos(0.0873) & 0 \end{bmatrix} = \begin{bmatrix} 0.4981 & 0 \end{bmatrix}$$

The Kalman Gain calculation:

$$\boldsymbol{K}_1 = \boldsymbol{P}_{1,0} \left( \frac{\partial \boldsymbol{h}(\hat{\boldsymbol{x}}_{1,0})}{\partial \boldsymbol{x}} \right)^T \left( \frac{\partial \boldsymbol{h}(\hat{\boldsymbol{x}}_{1,0})}{\partial \boldsymbol{x}} \boldsymbol{P}_{1,0} \left( \frac{\partial \boldsymbol{h}(\hat{\boldsymbol{x}}_{1,0})}{\partial \boldsymbol{x}} \right)^T + \boldsymbol{R} \right)^{-1} = \begin{bmatrix} 2.0075 \\ -1.8548 \end{bmatrix}$$

Estimate the current state:

$$\hat{\boldsymbol{x}}_{1,1} = \hat{\boldsymbol{x}}_{1,0} + \boldsymbol{K}_1 \left( \boldsymbol{z}_1 - \boldsymbol{h}(\hat{\boldsymbol{x}}_{1,0}) \right) = \begin{bmatrix} 0.2395 \\ -0.2261 \end{bmatrix}$$

Update the current estimate covariance:

$$\boldsymbol{P}_{1,1} = \left( \boldsymbol{I} - \boldsymbol{K}_1 \frac{\partial \boldsymbol{h}(\hat{\boldsymbol{x}}_{1,0})}{\partial \boldsymbol{x}} \right) \boldsymbol{P}_{1,0} \left( \boldsymbol{I} - \boldsymbol{K}_1 \frac{\partial \boldsymbol{h}(\hat{\boldsymbol{x}}_{1,0})}{\partial \boldsymbol{x}} \right)^T + \boldsymbol{K}_1 \boldsymbol{R} \boldsymbol{K}_1^T = \begin{bmatrix} 0.0004 & -0.00037 \\ -0.00037 & 5.489 \end{bmatrix}$$

#### Step 3 – Predict

$$\hat{\boldsymbol{x}}_{2,1} = \boldsymbol{f}(\hat{\boldsymbol{x}}_{1,1}) = \begin{bmatrix} \hat{\theta}_{1,1} + \hat{\dot{\theta}}_{1,1}\Delta t \\ \hat{\dot{\theta}}_{1,1} - \frac{g}{L} sin(\hat{\theta}_{1,1})\Delta t \end{bmatrix} = \begin{bmatrix} 0.2395 + (-0.2261) \cdot 0.05 \\ -0.2261 - \frac{9.8}{0.5} sin(0.2395) \cdot 0.05 \end{bmatrix}$$

$$= \begin{bmatrix} 0.2282 \\ -0.4586 \end{bmatrix}$$

Jacobian $\frac{\partial \boldsymbol{f}(\hat{\boldsymbol{x}}_{1,1})}{\partial \boldsymbol{x}}$ calculation:

$$\frac{\partial \boldsymbol{f}(\hat{\boldsymbol{x}}_{1,1})}{\partial \boldsymbol{x}} = \begin{bmatrix} 1 & \Delta t \\ -\frac{g}{L} cos(\hat{\theta}_{1,1})\Delta t & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0.05 \\ -\frac{9.8}{0.5} cos(0.2395) \cdot 0.05 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0.05 \\ -0.952 & 1 \end{bmatrix}$$

Uncertainty propagation:

$$\boldsymbol{P}_{2,1} = \frac{\partial \boldsymbol{f}(\hat{\boldsymbol{x}}_{1,1})}{\partial \boldsymbol{x}} \boldsymbol{P}_{1,1} \left( \frac{\partial \boldsymbol{f}(\hat{\boldsymbol{x}}_{1,1})}{\partial \boldsymbol{x}} \right)^T + \boldsymbol{Q} = \begin{bmatrix} 0.0141 & 0.274 \\ 0.2737 & 5.49 \end{bmatrix}$$

### 13.8.2.3 Second Iteration

#### Step 1 - Measure

The measurement values:

$\boldsymbol{z}_2 = 0.113$

#### Step 2 - Update

Observation matrix ($\boldsymbol{h}(\hat{\boldsymbol{x}}_{2,1})$) calculation.

$\boldsymbol{h}(\hat{\boldsymbol{x}}_{2,1}) = Lsin(\theta_{2,1}) = 0.5sin(0.2282) = 0.1131$

Observation matrix Jacobian $\left( \frac{\partial \boldsymbol{h}(\hat{\boldsymbol{x}}_{2,1})}{\partial \boldsymbol{x}} \right)$ calculation:

$$\frac{\partial \boldsymbol{h}(\hat{\boldsymbol{x}}_{2,1})}{\partial \boldsymbol{x}} = \begin{bmatrix} Lcos(\theta_{2,1}) & 0 \end{bmatrix} = \begin{bmatrix} 0.5cos(0.2282) & 0 \end{bmatrix} = \begin{bmatrix} 0.487 & 0 \end{bmatrix}$$

The Kalman Gain calculation:

$$\boldsymbol{K}_2 = \boldsymbol{P}_{2,1} \left( \frac{\partial \boldsymbol{h}(\hat{\boldsymbol{x}}_{2,1})}{\partial \boldsymbol{x}} \right)^T \left( \frac{\partial \boldsymbol{h}(\hat{\boldsymbol{x}}_{2,1})}{\partial \boldsymbol{x}} \boldsymbol{P}_{2,1} \left( \frac{\partial \boldsymbol{h}(\hat{\boldsymbol{x}}_{2,1})}{\partial \boldsymbol{x}} \right)^T + \boldsymbol{R} \right)^{-1} = \begin{bmatrix} 1.99 \\ 38.74 \end{bmatrix}$$

Estimate the current state:

$$\hat{\boldsymbol{x}}_{2,2} = \hat{\boldsymbol{x}}_{2,1} + \boldsymbol{K}_2 \left( \boldsymbol{z}_2 - \boldsymbol{h}(\hat{\boldsymbol{x}}_{2,1}) \right) = \begin{bmatrix} 0.228 \\ -0.463 \end{bmatrix}$$

Update the current estimate covariance:

$$\boldsymbol{P}_{2,2} = \left( \boldsymbol{I} - \boldsymbol{K}_2 \frac{\partial \boldsymbol{h}(\hat{\boldsymbol{x}}_{2,1})}{\partial \boldsymbol{x}} \right) \boldsymbol{P}_{2,1} \left( \boldsymbol{I} - \boldsymbol{K}_2 \frac{\partial \boldsymbol{h}(\hat{\boldsymbol{x}}_{2,1})}{\partial \boldsymbol{x}} \right)^T + \boldsymbol{K}_2 \boldsymbol{R} \boldsymbol{K}_2^T = \begin{bmatrix} 0.0004 & 0.008 \\ 0.008 & 0.328 \end{bmatrix}$$

#### Step 3 – Predict

$$\hat{\boldsymbol{x}}_{3,2} = \boldsymbol{f}(\hat{\boldsymbol{x}}_{2,2}) = \begin{bmatrix} \hat{\theta}_{2,2} + \hat{\dot{\theta}}_{2,2} \Delta t \\ \hat{\dot{\theta}}_{2,2} - \frac{g}{L} sin(\hat{\theta}_{2,2}) \Delta t \end{bmatrix} = \begin{bmatrix} 0.228 + (-0.463) \cdot 0.05 \\ -0.463 - \frac{9.8}{0.5} sin(0.228) \cdot 0.05 \end{bmatrix}$$

$$= \begin{bmatrix} 0.205 \\ -0.684 \end{bmatrix}$$

Jacobian $\frac{\partial \boldsymbol{f}(\hat{\boldsymbol{x}}_{2,2})}{\partial \boldsymbol{x}}$ calculation:

$$\frac{\partial \boldsymbol{f}(\hat{\boldsymbol{x}}_{2,2})}{\partial \boldsymbol{x}} = \begin{bmatrix} 1 & \Delta t \\ -\frac{g}{L} cos(\hat{\theta}_{2,2}) \Delta t & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0.05 \\ -\frac{9.8}{0.5} cos(0.463) \cdot 0.05 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0.05 \\ -0.955 & 1 \end{bmatrix}$$

Uncertainty propagation:

$$\boldsymbol{P}_{3,2} = \frac{\partial \boldsymbol{f}(\hat{\boldsymbol{x}}_{2,2})}{\partial \boldsymbol{x}} \boldsymbol{P}_{2,2} \left( \frac{\partial \boldsymbol{f}(\hat{\boldsymbol{x}}_{2,2})}{\partial \boldsymbol{x}} \right)^T + \boldsymbol{Q} = \begin{bmatrix} 0.002 & 0.0236 \\ 0.0236 & 0.315 \end{bmatrix}$$

At this point, I think it would be reasonable to jump to the tenth Kalman Filter iteration.

### 13.8.2.4 Tenth Iteration

#### Step 1 - Measure

The measurement values:

$$\boldsymbol{z}_{10} = -0.05$$

#### Step 2 - Update

Observation matrix ($\boldsymbol{h}(\hat{\boldsymbol{x}}_{10,9})$) calculation.

$$\boldsymbol{h}(\hat{\boldsymbol{x}}_{10,9}) = Lsin(\theta_{10,9}) = -0.075$$

Observation matrix Jacobian $\left(\frac{\partial \boldsymbol{h}(\hat{\boldsymbol{x}}_{10,9})}{\partial \boldsymbol{x}}\right)$ calculation:

$$\frac{\partial \boldsymbol{h}(\hat{\boldsymbol{x}}_{10,9})}{\partial \boldsymbol{x}} = \begin{bmatrix} Lcos(\theta_{10,9}) & 0 \end{bmatrix} = \begin{bmatrix} 0.494 & 0 \end{bmatrix}$$

The Kalman Gain calculation:

$$\boldsymbol{K}_{10} = \boldsymbol{P}_{10,9}\left(\frac{\partial \boldsymbol{h}(\hat{\boldsymbol{x}}_{10,9})}{\partial \boldsymbol{x}}\right)^T \left(\frac{\partial \boldsymbol{h}(\hat{\boldsymbol{x}}_{10,9})}{\partial \boldsymbol{x}}\boldsymbol{P}_{10,9}\left(\frac{\partial \boldsymbol{h}(\hat{\boldsymbol{x}}_{10,9})}{\partial \boldsymbol{x}}\right)^T + \boldsymbol{R}\right)^{-1} = \begin{bmatrix} 0.768 \\ 3.04 \end{bmatrix}$$

Estimate the current state:

$$\hat{\boldsymbol{x}}_{10,10} = \hat{\boldsymbol{x}}_{10,9} + \boldsymbol{K}_{10}\left(\boldsymbol{z}_{10} - \boldsymbol{h}(\hat{\boldsymbol{x}}_{10,9})\right) = \begin{bmatrix} -0.132 \\ -1.186 \end{bmatrix}$$

Update the current estimate covariance:

$$\boldsymbol{P}_{10,10} = \left(\boldsymbol{I} - \boldsymbol{K}_{10}\frac{\partial \boldsymbol{h}(\hat{\boldsymbol{x}}_{10,9})}{\partial \boldsymbol{x}}\right)\boldsymbol{P}_{2,1}\left(\boldsymbol{I} - \boldsymbol{K}_{10}\frac{\partial \boldsymbol{h}(\hat{\boldsymbol{x}}_{10,9})}{\partial \boldsymbol{x}}\right)^T + \boldsymbol{K}_{10}\boldsymbol{R}\boldsymbol{K}_{10}^T$$

$$= \begin{bmatrix} 0.155 & 0.616 \\ 0.616 & 9.58 \end{bmatrix} \times 10^3$$

### Step 3 – Predict

$$\hat{\boldsymbol{x}}_{11,10} = \boldsymbol{f}(\hat{\boldsymbol{x}}_{10,10}) = \begin{bmatrix} \hat{\theta}_{10,10} + \hat{\dot{\theta}}_{10,10}\Delta t \\ \hat{\dot{\theta}}_{2,2} - \frac{g}{L}sin(\hat{\theta}_{10,10})\Delta t \end{bmatrix} = \begin{bmatrix} -0.191 \\ -1.057 \end{bmatrix}$$

Jacobian $\frac{\partial \boldsymbol{f}(\hat{\boldsymbol{x}}_{10,10})}{\partial \boldsymbol{x}}$ calculation:

$$\frac{\partial \boldsymbol{f}(\hat{\boldsymbol{x}}_{10,10})}{\partial \boldsymbol{x}} = \begin{bmatrix} 1 & \Delta t \\ -\frac{g}{L}cos(\hat{\theta}_{10,10})\Delta t & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0.05 \\ -0.972 & 1 \end{bmatrix}$$

Uncertainty propagation:

$$\boldsymbol{P}_{11,10} = \frac{\partial \boldsymbol{f}(\hat{\boldsymbol{x}}_{10,10})}{\partial \boldsymbol{x}}\boldsymbol{P}_{10,10}\frac{\partial \boldsymbol{f}(\hat{\boldsymbol{x}}_{10,10})}{\partial \boldsymbol{x}}^T + \boldsymbol{Q} = \begin{bmatrix} 0.243 & 0.976 \\ 0.976 & 11.04 \end{bmatrix} \times 10^3$$

### 13.8.3 Example summary

The following chart compares the true, measured, and estimated pendulum angles for 50 measurements. (The measured angle is derived from the measured distance $\theta_n = \sin^{-1}\left(\frac{z}{L}\right)$). The chart also includes the 95% confidence interval.

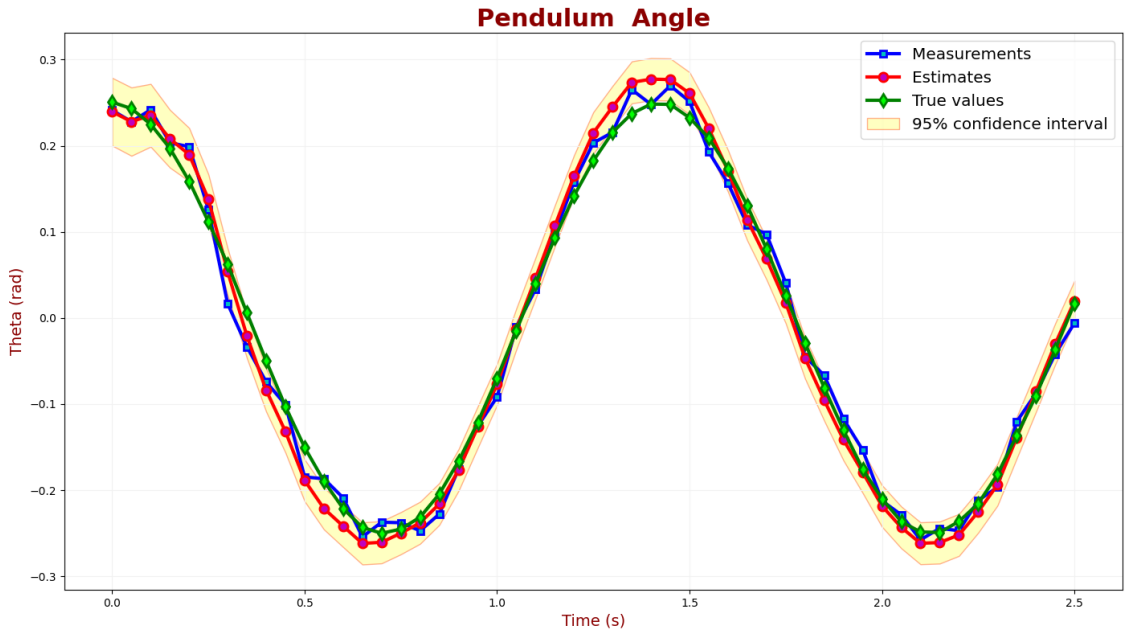


**Figure 13.11:** *Example 12: pendulum angle - true value, measured values and estimates.*

The next chart compares the true and estimated pendulum angular velocity.

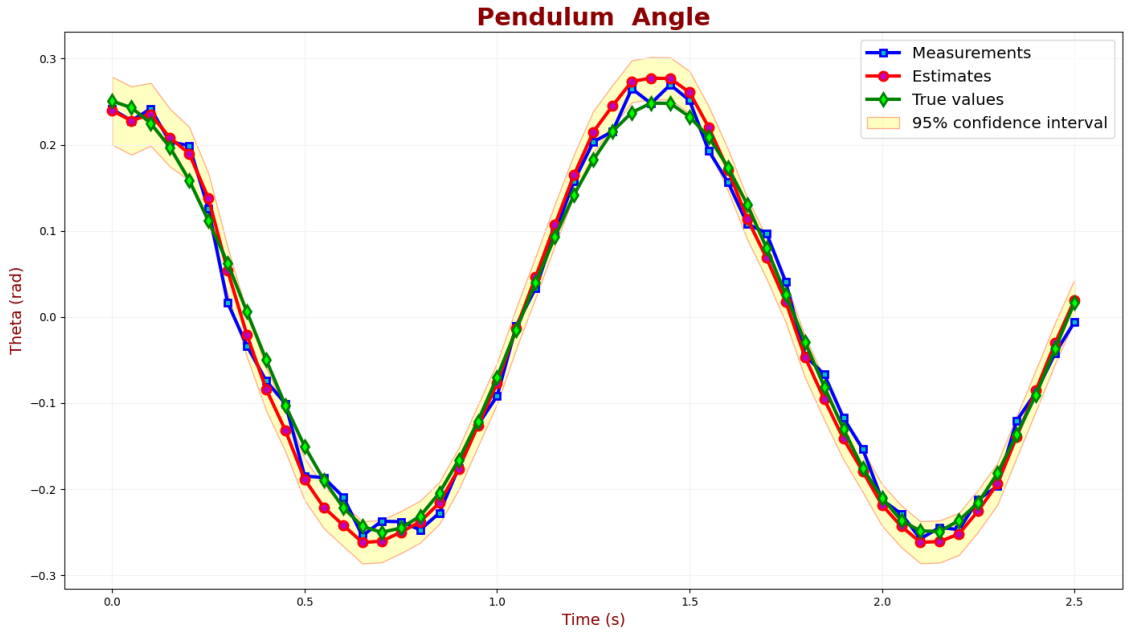


**Figure 13.12:** *Example 12: pendulum anglular velocity - true value, measured values and estimates.*

## 13.9 Limitations of EKF

EKF performs well for many practical problems when $\boldsymbol{f}(\boldsymbol{x})$ or $\boldsymbol{h}(\boldsymbol{x})$ are close to linear. However, it fails in highly non-linear regions.

The EKF concept is based on the linearization of the model. The EKF estimation includes the **linearization error**. The linearization error depends on the non-linearity degree of the function compared to the propagated uncertainty, as shown in the following figure.

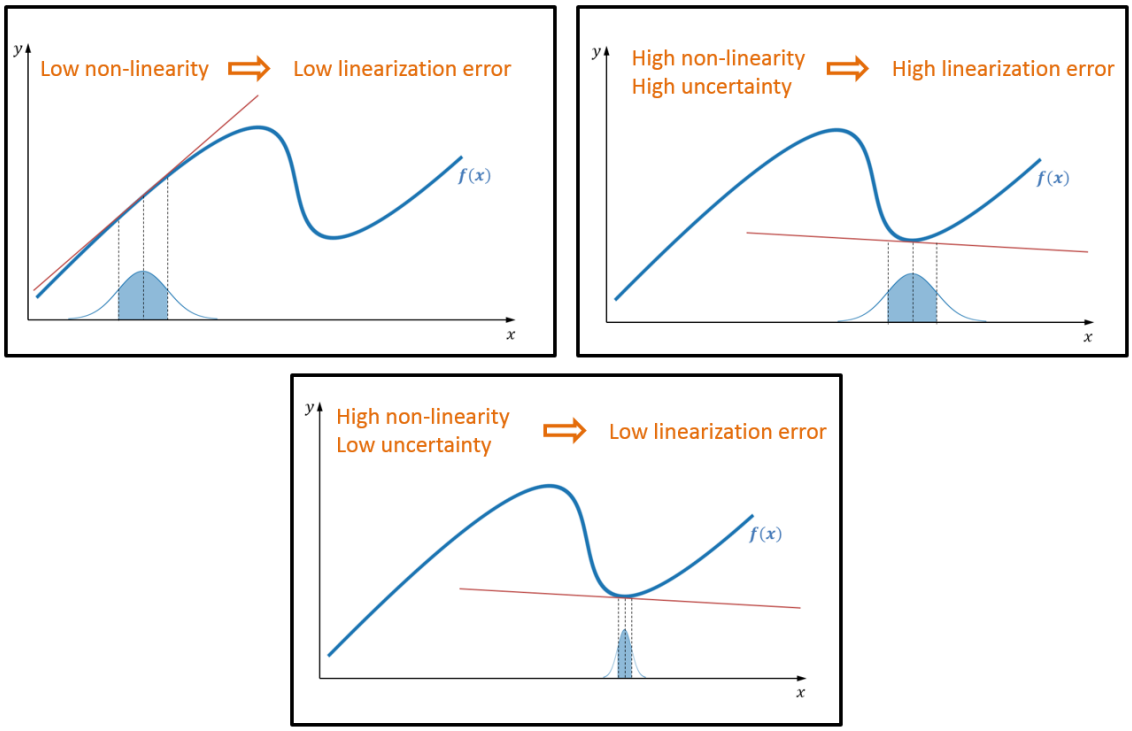


**Figure 13.13:** *Linearization Error.*

### 13.9.1 Linearization error - 2D example

Let us see the effect of the linearization error on polar to cartesian transformation. Assume a normally distributed random variable in polar coordinates. We want to estimate the random variable parameters in cartesian coordinates. A distance vector $r$ and an angle $\theta$ describe any value in the polar coordinates. In cartesian coordinates, the values are described by $x$ and $y$ coordinates. The dependency between $r$, $\theta$, and $x$, $y$ is non-linear:

$$\begin{aligned} x &= r \cdot cos(\theta) \\ y &= r \cdot sin(\theta) \end{aligned} \tag{13.91}$$

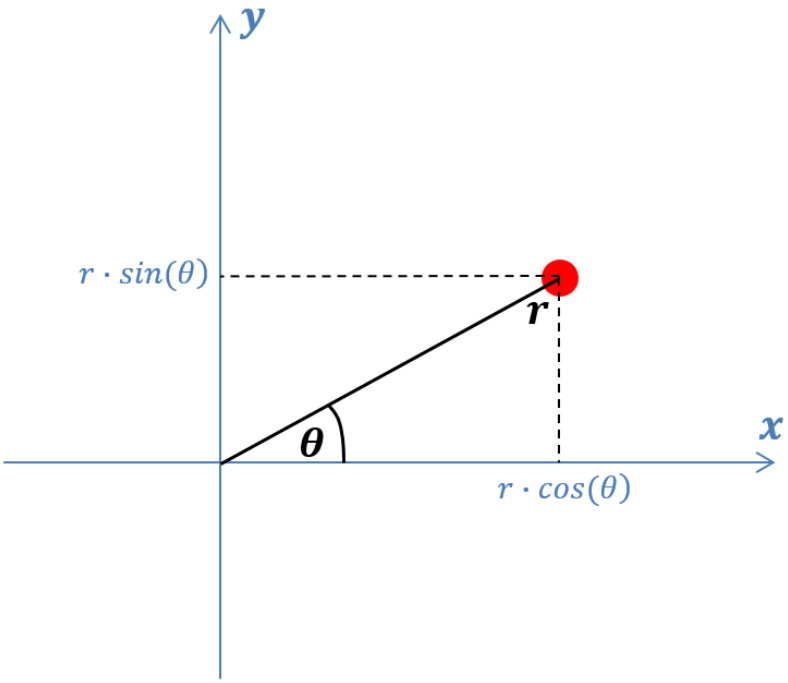


**Figure 13.14:** *2D example.*

The random variable parameters $(r, \theta)$ in polar coordinates are: $\mu = \begin{bmatrix} 1 & \frac{\pi}{2} \end{bmatrix}, \sigma = \begin{bmatrix} 0.05 & 0.5 \end{bmatrix}$.

We generate 1000 random points (samples) with normal distribution in polar coordinates. Each sample represents a possible variable value in polar coordinates. Then we transform all the samples from polar to cartesian coordinates. The random variable distribution in polar coordinates is normal.

The plot on the left describes the random samples of the random variable in polar coordinates. The right plot describes the random samples of the random variable in cartesian coordinates after the transformation.

The ellipses on the plots represent the covariance of the random variable.

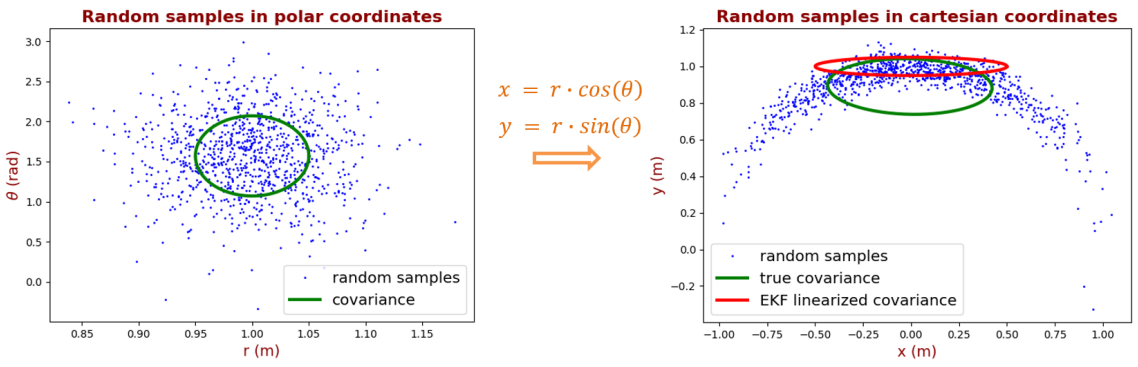


**Figure 13.15:** *EKF linearized covariance.*

We can see a significant difference between actual and EKF linearized covariance. The EKF linearized covariance includes a high linearization error.

The EKF yields a wrong estimation. The EKF estimation uncertainty is also relatively low (the error ellipse is relatively narrow). **The EKF is overconfident in a wrong estimation!**

A common alternative to the Extended Kalman Filter is the Unscented Kalman Filter.

The following plot compares EKF and UKF linearized covariance for our example.

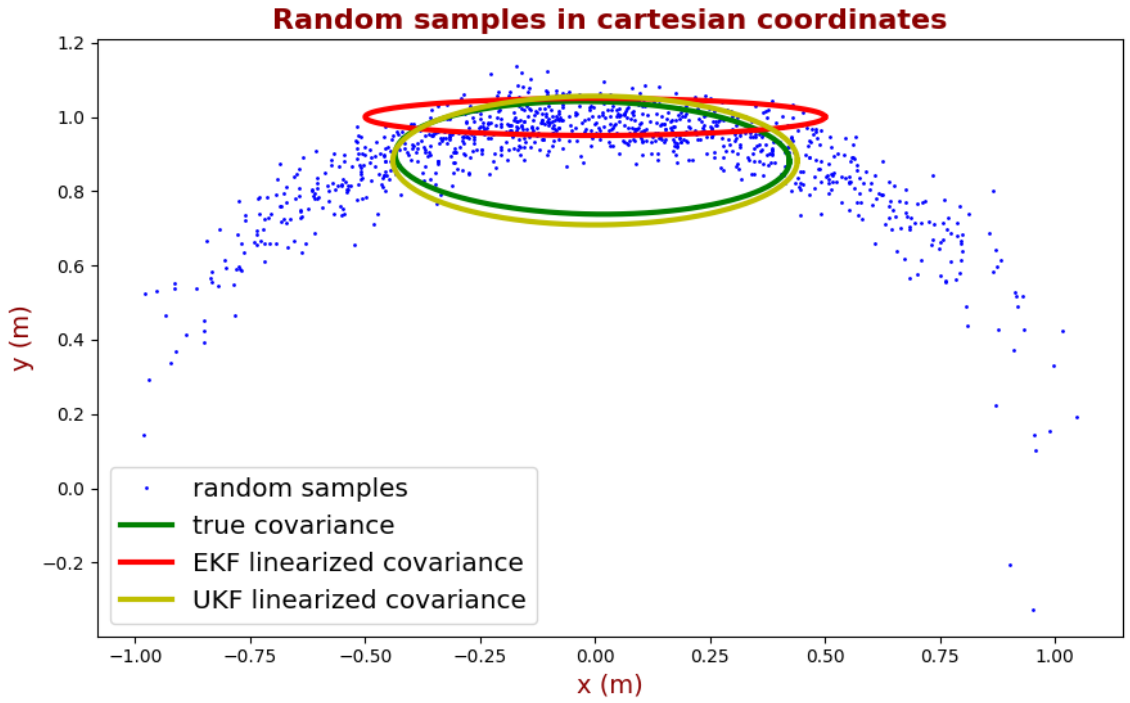


**Figure 13.16:** *EKF vs. UKF linearized covariance.*

We can see that the UKF linearized covariance is much closer to the actual covariance than the EKF linearized covariance.

# 14. Unscented Kalman Filter (UKF)

As we've seen in the previous chapter, when the State Transition model $\boldsymbol{f}(\boldsymbol{x})$ and Observation model $\boldsymbol{h}(\boldsymbol{x})$ are close to linear, the EKF performance is satisfying. However, when $\boldsymbol{f}(\boldsymbol{x})$ or $\boldsymbol{h}(\boldsymbol{x})$ models are highly non-linear, the linearization error can cause estimations that are significantly different from the true value of the state and estimation uncertainties that don't capture the true uncertainties in the state.

The Unscented Kalman Filter is an alternative approach to linearization. While Extended Kalman Filter treats the non-linearity using analytical linearization, the Unscented Kalman Filter performs Unscented Transform (UT) - statistical linearization based on a set of rules.

Jeffrey Uhlmann initially proposed the unscented transform (UT) as a component of his PhD thesis [18]; however, it is predominantly known from [8].

What is the meaning of the name "unscented"?

A running joke was made that "unscented" is a contrast to "scented," meaning the EKF performance is "stinky."

UKF creator Jeffrey Uhlmann explained that "unscented" was an arbitrary name he adopted to avoid being referred to as the "Uhlmann Filter."

*"Initially, I only referred to it as the "new filter." Needing a more specific name, people in my lab began referring to it as the "Uhlmann filter," which obviously isn't a name that I could use, so I had to come up with an official term. One evening everyone else in the lab was at the Royal Opera House, and as I was working, I noticed someone's deodorant on a desk. The word "unscented" caught my eye as the perfect technical term."*

Jeffrey Uhlmann also says:

*"What was most striking to people about the UT was not the accuracy so much as the ease with which it could be implemented. There was no longer a need to derive a linearized approximation which would then have to be coded up for use in the filter."*

## 14.1 The Unscented Transform (UT)

The **Unscented Transform** is a method for calculating the statistics of a random variable that undergoes a non-linear transformation [8].

The Unscented Transform includes three steps:

- Step 1 - Select a set of points from the input distribution. The points are selected according to a specific, deterministic algorithm.
- Step 2 - Propagate each selected point through the non-linear function, producing a new set of points belonging to the output distribution.
- Step 3 - Compute sigma points weights.
- Step 4 - Approximate the sample mean and covariance of the output distribution using the propagated set of points and carefully chosen weights.

### 14.1.1 Step 1 – sigma points selection

The set of sigma points includes the mean and a certain number of points located at a certain distance away from the mean.

#### 14.1.1.1 Number of selected points

The number of selected points depends on the input distribution. The $N$ – dimensional random variable is approximated by $2N+1$ points. For a one-dimensional distribution ($N = 1$), the number of points is 3. For a two-dimensional distribution ($N = 2$), the number of points is 5.

#### 14.1.1.2 Selected points location

The first point is the mean of the input distribution:

$$\mathcal{X}_{n,n}^{(0)} = \hat{\boldsymbol{x}}_{n,n} \tag{14.1}$$

The superscript in parentheses of $\mathcal{X}_{n,n}^{(0)}$ is a sigma point number.

Reminder: In "Kalman Filter language," the mean of the input distribution is the current estimate $\hat{\boldsymbol{x}}_{n,n}$, and the uncertainty of the input distribution represented by the covariance matrix of the current estimate $\boldsymbol{P}_{n,n}$. The main diagonal of the covariance matrix includes variances for each dimension ($\sigma_{xx}, \sigma_{yy}$).

The other points are located at a certain **statistical distance** from the mean. The statistical distance is a distance measure for probability distributions. In other words, the statistical distance is expressed in terms of standard deviation or sigma ($\sigma$). For this reason, the selected points are called the **Sigma Points**, and the Unscentenced

Transform is sometimes called the **Sigma point transform**.

The other sigma points location is:

$$\begin{aligned}\boldsymbol{\mathcal{X}}_{n,n}^{(i)} &= \hat{\boldsymbol{x}}_{n,n} + \left(\sqrt{(N+\kappa)\,\boldsymbol{P}_{n,n}}\right)_i, \qquad i = 1, \cdots, N \\ \boldsymbol{\mathcal{X}}_{n,n}^{(i)} &= \hat{\boldsymbol{x}}_{n,n} - \left(\sqrt{(N+\kappa)\,\boldsymbol{P}_{n,n}}\right)_{i-N}, \qquad i = N+1, \cdots, 2N\end{aligned} \tag{14.2}$$

Where:

$N$ is the number of dimensions

$\kappa$ (the Greek letter kappa) is a tuning parameter

$\left(\sqrt{(N+\kappa)\,\boldsymbol{P}_{n,n}}\right)$ is the $i^{th}$ row or column of the matrix square root of $\sqrt{(N+\kappa)\,\boldsymbol{P}_{n,n}}$

For Gaussian distribution, the rule of thumb is to set: $N + \kappa = 3$.

*The sigma points should be chosen so that they capture the most important statistical properties of the prior random variable* $\hat{\boldsymbol{x}}$ [9].

It is time for some numerical examples.

#### 14.1.1.3 Example: one-dimensional random variable

Assume a zero-mean normally distributed one-dimensional random variable with a standard deviation of 2: $\hat{\boldsymbol{x}}_{n,n} = 0, \boldsymbol{p}_{n,n} = 2$

- The number of dimensions: $N = 1$
- The number of sigma points: $2N + 1 = 3$
- Set: $N + \kappa = 3$
- The first point is the mean of the input distribution: $\boldsymbol{\mathcal{X}}_{n,n}^{(0)} = 0$
- The second point: $\boldsymbol{\mathcal{X}}_{n,n}^{(1)} = \hat{\boldsymbol{x}}_{n,n} + \left(\sqrt{(N+\kappa)\,\boldsymbol{P}_{n,n}}\right)_1 = 0 + \left(\sqrt{3 \cdot 2^2}\right) = 3.46$
- The third point: $\boldsymbol{\mathcal{X}}_{n,n}^{(2)} = \hat{\boldsymbol{x}}_{n,n} - \left(\sqrt{(N+\kappa)\,\boldsymbol{P}_{n,n}}\right)_2 = 0 - \left(\sqrt{3 \cdot 2^2}\right) = -3.46$
- The following figure describes the random variable PDF with sigma points (red circles)

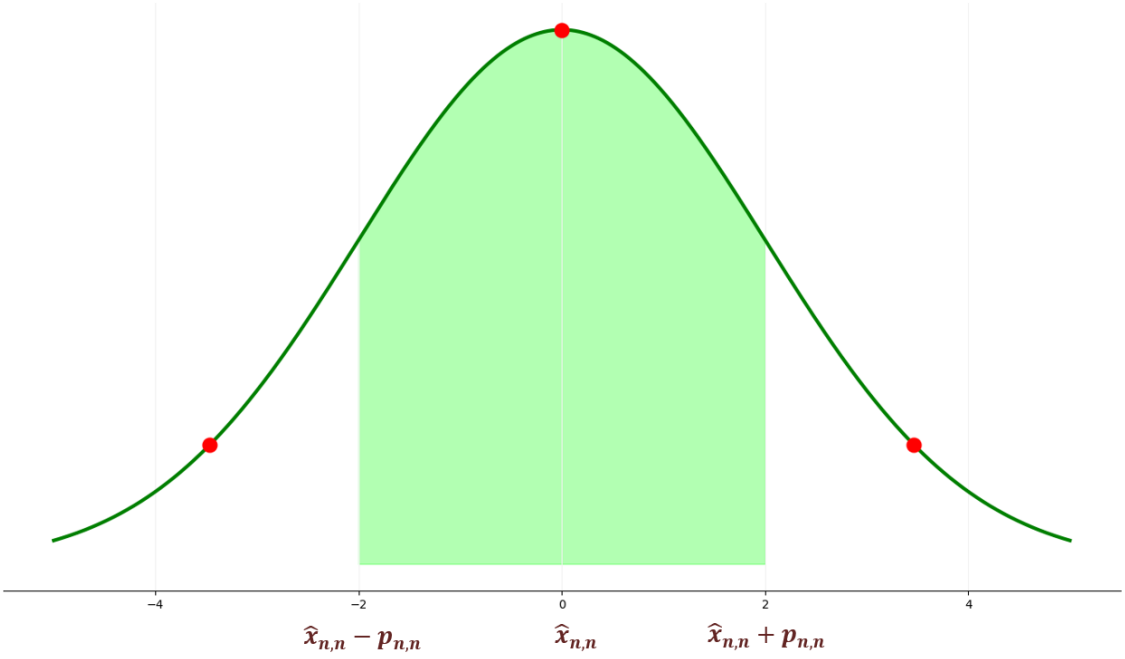


**Figure 14.1:** *1D RV Sigma Points.*

The Unscented Transform sigma points are not necessarily on the standard deviation boundaries.

#### 14.1.1.4 Example: two-dimensional random variable

Let us continue the polar to cartesian transformation example from section 13.9.

$$\hat{\boldsymbol{x}}_{n,n} = \begin{bmatrix} 1 \\ \frac{\pi}{2} \end{bmatrix} = \begin{bmatrix} 1 \\ 1.57 \end{bmatrix}$$

$$\boldsymbol{P}_{n,n} = \begin{bmatrix} 0.05^2 & 0 \\ 0 & 0.5^2 \end{bmatrix} = \begin{bmatrix} 0.0025 & 0 \\ 0 & 0.25 \end{bmatrix}$$

Finding the sigma points:

- The number of dimensions: $N = 2$
- The number of sigma points: $2N + 1 = 5$
- Set: $N + \kappa = 3$
- The first point is the mean of the input distribution: $\mathcal{X}_{n,n}^{(0)} = \begin{bmatrix} 1 \\ 1.57 \end{bmatrix}$
- In order to find the other points, we should compute: $\sqrt{(N + \kappa)\, \boldsymbol{P}_{n,n}}$

$$\sqrt{(N+\kappa)\,\boldsymbol{P}_{n,n}} = \sqrt{3\begin{bmatrix}0.0025 & 0\\ 0 & 0.25\end{bmatrix}} = \sqrt{\begin{bmatrix}0.0075 & 0\\ 0 & 0.75\end{bmatrix}}$$

We need to find the square root of the matrix. Luckily, the covariance matrix is positive and semi-definite; therefore, we can use Cholesky decomposition (section 11.2) to find the square root.

$$(N+\kappa)\,\boldsymbol{P}_{n,n} = LL^T$$

$L$ is a lower triangular matrix:

$$L = \begin{bmatrix}l_{11} & 0\\ l_{21} & l_{22}\end{bmatrix}$$

$$l_{11} = \sqrt{p_{11}} = \sqrt{0.0075} = 0.0866$$

$$l_{21} = \frac{p_{21}}{l_{11}} = \frac{0}{0.0866} = 0$$

$$l_{22} = \sqrt{p_{22} - l_{21}^2} = \sqrt{0.75 - 0^2} = 0.866$$

$$L = \begin{bmatrix}0.0866 & 0\\ 0 & 0.866\end{bmatrix}$$

We can compute it faster with computer software packages:

**Python example:**

```
import numpy as np
P = np.array([[0.0075, 0], [0, 0.75]])
print(P)
[[0.0075 0.    ]
 [0.     0.75  ]]

L = np.linalg.cholesky(P)
print(L)
[[0.08660254 0.        ]
 [0.         0.8660254 ]]
```

**MATLAB example:**

```
P = [0.0075 0; 0 0.75]
P =
    0.0075         0
         0    0.7500
L = chol(P)'
L =
    0.0866         0
         0    0.8660
```

- The second point: $\mathcal{X}_{n,n}^{(1)} = \hat{\boldsymbol{x}}_{n,n} + \left(\sqrt{(N+\kappa)\,\boldsymbol{P}_{n,n}}\right)_1 = \begin{bmatrix} 1 \\ 1.57 \end{bmatrix} + \begin{bmatrix} 0.0866 \\ 0 \end{bmatrix} = \begin{bmatrix} 1.0866 \\ 1.57 \end{bmatrix}$
- The third point: $\mathcal{X}_{n,n}^{(1)} = \hat{\boldsymbol{x}}_{n,n} + \left(\sqrt{(N+\kappa)\,\boldsymbol{P}_{n,n}}\right)_1 = \begin{bmatrix} 1 \\ 1.57 \end{bmatrix} + \begin{bmatrix} 0 \\ 0.866 \end{bmatrix} = \begin{bmatrix} 1 \\ 2.436 \end{bmatrix}$
- The fourth point: $\mathcal{X}_{n,n}^{(1)} = \hat{\boldsymbol{x}}_{n,n} + \left(\sqrt{(N+\kappa)\,\boldsymbol{P}_{n,n}}\right)_1 = \begin{bmatrix} 1 \\ 1.57 \end{bmatrix} - \begin{bmatrix} 0.0866 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.9134 \\ 1.57 \end{bmatrix}$
- The fifth point: $\mathcal{X}_{n,n}^{(1)} = \hat{\boldsymbol{x}}_{n,n} + \left(\sqrt{(N+\kappa)\,\boldsymbol{P}_{n,n}}\right)_1 = \begin{bmatrix} 1 \\ 1.57 \end{bmatrix} - \begin{bmatrix} 0 \\ 0.866 \end{bmatrix} = \begin{bmatrix} 1 \\ 0.704 \end{bmatrix}$

The following figure describes the covariance ellipse of the random variable PDF with sigma points (red circles).

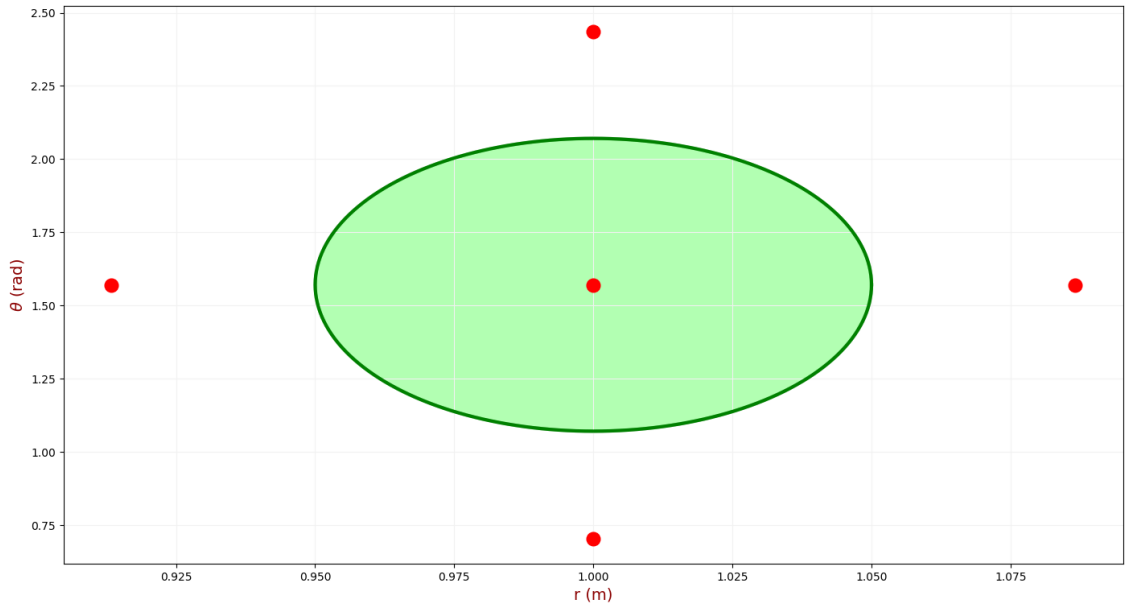


**Figure 14.2:** *2D RV Sigma Points.*

The Unscented Transform sigma points are not necessarily on the covariance ellipse boundaries.

### 14.1.2 Step 2 – points propagation

Propagate each selected point through the non-linear function, producing a new set of points belonging to the output distribution.

#### 14.1.2.1 Example: one-dimensional random variable - continued

The non-linear function is:

$$f(x) = sin(2x)sin(0.3x) + 2x$$

$$\mathcal{X}_{n+1,n} = \boldsymbol{f}\left(\mathcal{X}_{n,n}\right)$$

The propagated (or transformed) sigma points:

$$\mathcal{X}^{(0)}_{n+1,n} = sin(2 \cdot 0)sin(0.3 \cdot 0) + 2 \cdot 0 = 0$$

$$\mathcal{X}^{(1)}_{n+1,n} = sin(2 \cdot 3.46)sin(0.3 \cdot 3.46) + 2 \cdot 3.46 = 7.45$$

$$\mathcal{X}^{(1)}_{n+1,n} = sin(2 \cdot (-3.46))sin(0.3 \cdot (-3.46)) + 2 \cdot (-3.46) = -6.41$$

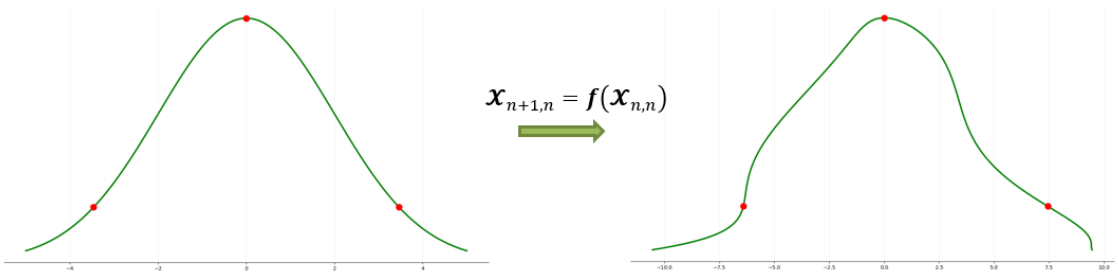


**Figure 14.3:** *1D RV Sigma Points propagation.*

The green line on the left plot is the PDF of the input random variable. The red circles on the left plot are the sigma points ($\mathcal{X}_{n,n}$) of the input random variable.

The green line on the right plot is the PDF of the input random variable after the non-linear transformation. The red circles on the right plot are the sigma points after the non-linear transformation ($\mathcal{X}_{n+1,n}$).

#### 14.1.2.2 Example: two-dimensional random variable - continued

The non-linear function is:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} r \cdot cos(\theta) \\ r \cdot sin(\theta) \end{bmatrix}$$

$$\mathcal{X}_{n+1,n} = \boldsymbol{f}\left(\mathcal{X}_{n,n}\right)$$

The propagated (or transformed) sigma points:

$$\mathcal{X}^{(0)}_{n+1,n} = \boldsymbol{f}\left(\mathcal{X}^{(0)}_{n,n}\right) = \begin{bmatrix} 1 \cdot cos(\frac{\pi}{2}) \\ 1 \cdot sin(\frac{\pi}{2}) \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\mathcal{X}^{(1)}_{n+1,n} = \boldsymbol{f}\left(\mathcal{X}^{(1)}_{n,n}\right) = \begin{bmatrix} 1.0866 \cdot cos(\frac{\pi}{2}) \\ 1.0866 \cdot sin(\frac{\pi}{2}) \end{bmatrix} = \begin{bmatrix} 0 \\ 1.0866 \end{bmatrix}$$

$$\mathcal{X}^{(2)}_{n+1,n} = \boldsymbol{f}\left(\mathcal{X}^{(2)}_{n,n}\right) = \begin{bmatrix} 1 \cdot cos(2.436) \\ 1 \cdot sin(2.436) \end{bmatrix} = \begin{bmatrix} -0.762 \\ 0.648 \end{bmatrix}$$

$$\mathcal{X}^{(3)}_{n+1,n} = \boldsymbol{f}\left(\mathcal{X}^{(3)}_{n,n}\right) = \begin{bmatrix} 0.9134 \cdot cos(\frac{\pi}{2}) \\ 0.9134 \cdot sin(\frac{\pi}{2}) \end{bmatrix} = \begin{bmatrix} 0 \\ 0.913 \end{bmatrix}$$

$$\mathcal{X}^{(2)}_{n+1,n} = \boldsymbol{f}\left(\mathcal{X}^{(2)}_{n,n}\right) = \begin{bmatrix} 1 \cdot cos(0.704) \\ 1 \cdot sin(0.704) \end{bmatrix} = \begin{bmatrix} 0.762 \\ 0.648 \end{bmatrix}$$

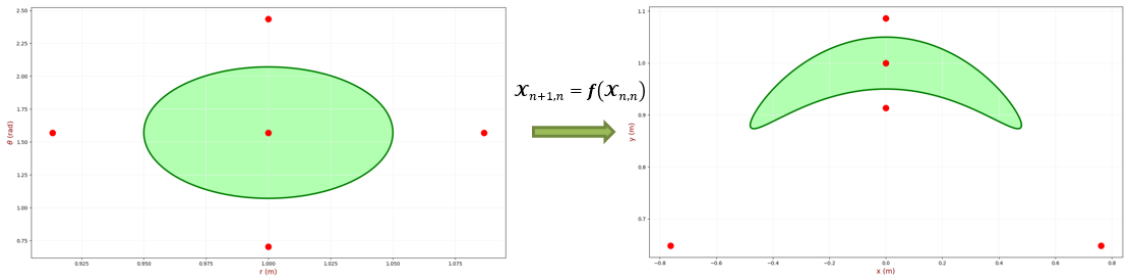


**Figure 14.4:** *2D RV Sigma Points propagation.*

The green shape on the left plot is the covariance ellipse of the input random variable. The red circles on the left plot are the sigma points ($\mathcal{X}_{n,n}$) of the input random variable.

The green shape on the right plot is the covariance ellipse of the input random variable after the non-linear transformation. The red circles on the right plot are the sigma points after the non-linear transformation ($\mathcal{X}_{n+1,n}$).

### 14.1.3 Step 3 – compute sigma points weights

We should compute two weights:

- $w_0$ - weight of the first sigma point $\left(\mathcal{X}_{n,n}^{(0)}\right)$
- $w_1$ - weight of the other sigma points $\left(\mathcal{X}_{n,n}^{(i)}\right), \qquad i > 0$

$$\begin{aligned} w_0 &= \kappa/(N+\kappa) \\ w_i &= 1/2(N+\kappa), \qquad i > 0 \end{aligned} \tag{14.3}$$

### 14.1.4 Step 4 - approximate the mean and covariance of the output distribution

In this step, we approximate the sample mean and covariance of the output distribution using the propagated set of points and carefully chosen weights.

The mean of the output distribution:

$$\hat{\boldsymbol{x}}_{n+1,n} = \sum_{i=0}^{2N} w_i \mathcal{X}_{n+1,n}^{(i)} \tag{14.4}$$

The covariance of the output is also computed with weights:

$$\boldsymbol{P}_{n+1,n} = \sum_{i=0}^{2N} w_i \left(\mathcal{X}_{n+1,n}^{(i)} - \hat{\boldsymbol{x}}_{n+1,n}\right) \left(\mathcal{X}_{n+1,n}^{(i)} - \hat{\boldsymbol{x}}_{n+1,n}\right)^T \tag{14.5}$$

Let us complete our examples.

#### 14.1.4.1 Example: one-dimensional random variable - continued

Weights computation:

- The number of dimensions: $N = 1$
- $N + \kappa = 3$
- $\kappa = 2$

$$w_0 = \kappa / (N + \kappa) = 2/3$$

$$w_i = 1/2 (N + \kappa) = 1/(2 \cdot 3) = 1/6$$

Mean computation:

$$\hat{\boldsymbol{x}}_{n+1,n} = \sum_{i=0}^{2N} w_i \mathcal{X}_{n+1,n}^{(i)} = 2/3 \cdot 0 + 1/6 \cdot 7.45 + 1/6 \cdot (-6.42) = 0.17$$

Covariance computation (in the one-dimensional case, it is variance):

$$\boldsymbol{P}_{n+1,n} = \sum_{i=0}^{2N} w_i \left(\mathcal{X}_{n+1,n}^{(i)} - \hat{\boldsymbol{x}}_{n+1,n}\right) \left(\mathcal{X}_{n+1,n}^{(i)} - \hat{\boldsymbol{x}}_{n+1,n}\right)^T$$

$$= 2/3 \cdot (0 - 0.17)^2 + 1/6 \cdot (7.45 - 0.17)^2 + 1/6 \cdot (-6.42 - 0.17)^2 = 16.06$$

The following plot depicts the output random variable after the Unscented Transform.

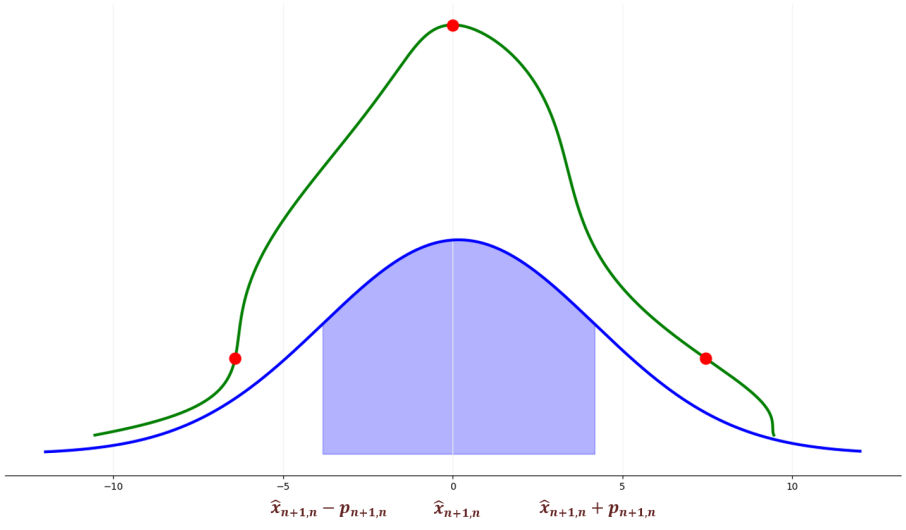


**Figure 14.5:** *1D RV Unscented Transform.*

The green line is the PDF of the input random variable after non-linear transformation. The red circles are the sigma points after the non-linear transformation $\left(\mathcal{X}_{n+1,n}^{(i)}\right)$.

The blue line is the PDF of the output random variable after the Unscented Transform.

#### 14.1.4.2 Example: two-dimensional random variable - continued

Weights computation:

- The number of dimensions: $N = 2$
- $N + \kappa = 3$
- $\kappa = 1$

$$w_0 = \kappa / (N + \kappa) = 1/3$$
$$w_i = 1/2 (N + \kappa) = 1/(2 \cdot 3) = 1/6$$

Mean computation:

$$\hat{\boldsymbol{x}}_{n+1,n} = \sum_{i=0}^{2N} w_i \mathcal{X}_{n+1,n}^{(i)} = 1/3 \begin{bmatrix} 0 \\ 1 \end{bmatrix} + 1/6 \begin{bmatrix} 0 \\ 1.0866 \end{bmatrix} + 1/6 \begin{bmatrix} -0.762 \\ 0.648 \end{bmatrix}$$
$$+1/6 \begin{bmatrix} 0 \\ 0.913 \end{bmatrix} + 1/6 \begin{bmatrix} 0.762 \\ 0.648 \end{bmatrix} = \begin{bmatrix} 0 \\ 0.8826 \end{bmatrix}$$

Covariance computation:

$$\boldsymbol{P}_{n+1,n} = \sum_{i=0}^{2N} w_i \left(\mathcal{X}_{n+1,n}^{(i)} - \hat{\boldsymbol{x}}_{n+1,n}\right) \left(\mathcal{X}_{n+1,n}^{(i)} - \hat{\boldsymbol{x}}_{n+1,n}\right)^T$$
$$= 1/3 \left(\begin{bmatrix} 0 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0.8826 \end{bmatrix}\right) \left(\begin{bmatrix} 0 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0.8826 \end{bmatrix}\right)^T$$
$$+1/6 \left(\begin{bmatrix} 0 \\ 1.0866 \end{bmatrix} - \begin{bmatrix} 0 \\ 0.8826 \end{bmatrix}\right) \left(\begin{bmatrix} 0 \\ 1.0866 \end{bmatrix} - \begin{bmatrix} 0 \\ 0.8826 \end{bmatrix}\right)^T$$
$$+1/6 \left(\begin{bmatrix} -0.762 \\ 0.648 \end{bmatrix} - \begin{bmatrix} 0 \\ 0.8826 \end{bmatrix}\right) \left(\begin{bmatrix} -0.762 \\ 0.648 \end{bmatrix} - \begin{bmatrix} 0 \\ 0.8826 \end{bmatrix}\right)^T$$
$$+1/6 \left(\begin{bmatrix} 0 \\ 0.913 \end{bmatrix} - \begin{bmatrix} 0 \\ 0.8826 \end{bmatrix}\right) \left(\begin{bmatrix} 0 \\ 0.913 \end{bmatrix} - \begin{bmatrix} 0 \\ 0.8826 \end{bmatrix}\right)^T$$
$$+1/6 \left(\begin{bmatrix} 0.762 \\ 0.648 \end{bmatrix} - \begin{bmatrix} 0 \\ 0.8826 \end{bmatrix}\right) \left(\begin{bmatrix} 0.762 \\ 0.648 \end{bmatrix} - \begin{bmatrix} 0 \\ 0.8826 \end{bmatrix}\right)^T$$
$$= \begin{bmatrix} 0.193 & 0 \\ 0 & 0.03 \end{bmatrix}$$

We can compute the mean and covariance more elegantly:

- Define a matrix (container) of transformed sigma points:
  $\mathcal{X}_{n+1,n} = \begin{bmatrix} \mathcal{X}_{n+1,n}^{(0)} & \mathcal{X}_{n+1,n}^{(1)} & \cdots & \mathcal{X}_{n+1,n}^{(2N)} \end{bmatrix}$
- Define a vector of weights:
  $\boldsymbol{w} = \begin{bmatrix} w_0 & w_1 & \cdots w_{2N} \end{bmatrix}$
- Mean computation:
  $\hat{\boldsymbol{x}}_{n+1,n} = \mathcal{X}_{n+1,n}\boldsymbol{w}$

$$\hat{\boldsymbol{x}}_{n+1,n} = \begin{bmatrix} 0 & 0 & -0.762 & 0 & 0.762 \\ 1 & 1.0866 & 0.648 & 0.913 & 0.648 \end{bmatrix} \begin{bmatrix} 1/3 \\ 1/6 \\ 1/6 \\ 1/6 \\ 1/6 \end{bmatrix} = \begin{bmatrix} 0 \\ 0.8826 \end{bmatrix}$$

- Covariance computation:
  $\boldsymbol{P}_{n+1,n} = (\mathcal{X}_{n+1,n} - \hat{\boldsymbol{x}}_{n+1,n}) \cdot diag(\boldsymbol{w}) \cdot (\mathcal{X}_{n+1,n} - \hat{\boldsymbol{x}}_{n+1,n})^T$

$$(\mathcal{X}_{n+1,n} - \hat{\boldsymbol{x}}_{n+1,n}) = \begin{bmatrix} 0 & 0 & -0.762 & 0 & 0.762 \\ 1 & 1.0866 & 0.648 & 0.913 & 0.648 \end{bmatrix} - \begin{bmatrix} 0 \\ 0.8826 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & -0.762 & 0 & 0.762 \\ 0.117 & 0.204 & -0.235 & 0.03 & -0.235 \end{bmatrix}$$

$$\boldsymbol{P}_{n+1,n} = \begin{bmatrix} 0 & 0 & -0.762 & 0 & 0.762 \\ 0.117 & 0.204 & -0.235 & 0.03 & -0.235 \end{bmatrix} \begin{bmatrix} 1/3 & 0 & 0 & 0 & 0 \\ 0 & 1/6 & 0 & 0 & 0 \\ 0 & 0 & 1/6 & 0 & 0 \\ 0 & 0 & 0 & 1/6 & 0 \\ 0 & 0 & 0 & 0 & 1/6 \end{bmatrix}$$

$$\times \begin{bmatrix} 0 & 0.117 \\ 0 & 0.204 \\ -0.762 & -0.235 \\ 0 & 0.03 \\ 0.762 & -0.235 \end{bmatrix} = \begin{bmatrix} 0.193 & 0 \\ 0 & 0.03 \end{bmatrix}$$

The following plot depicts the output random variable after the Unscented Transform.

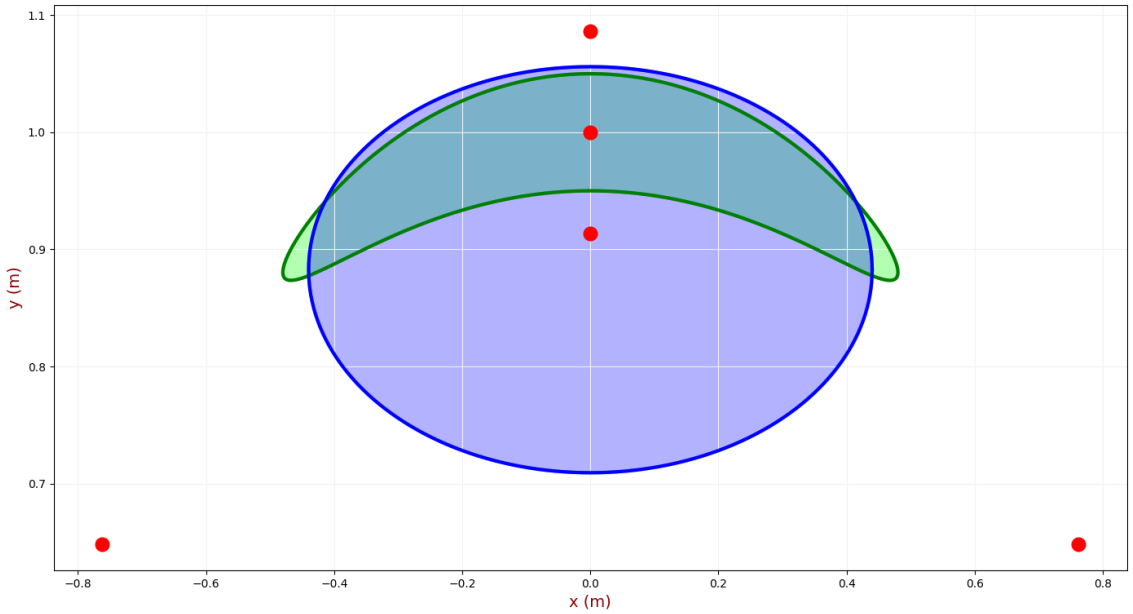


**Figure 14.6:** *2D RV Unscented Transform.*

The green shape is the covariance ellipse of the input random variable after the non-linear transformation. The red circles on the right plot are the sigma points after the non-linear transformation $\left(\mathcal{X}_{n+1,n}^{(i)}\right)$.

The blue ellipse is the covariance ellipse of the output random variable after the Unscented Transform.

### 14.1.5 Unscented Transform summary

The Unscented Transform is a method for calculating the statistics of a random variable that undergoes a non-linear transformation.

The Unscented Transform includes three steps:

- Step 1 - Select a set of points from the input distribution.
  - The set of sigma points includes the mean and a certain number of points located at a certain distance away from the mean.
  - The $N$ – dimensional random variable is approximated by $2N + 1$ points.

**Sigma Points Selection**

$$\begin{aligned}
\mathcal{X}_{n,n}^{(0)} &= \hat{\boldsymbol{x}}_{n,n} \\
\mathcal{X}_{n,n}^{(i)} &= \hat{\boldsymbol{x}}_{n,n} + \left(\sqrt{(N+\kappa)\,\boldsymbol{P}_{n,n}}\right)_i, \qquad i = 1, \cdots, N \\
\mathcal{X}_{n,n}^{(i)} &= \hat{\boldsymbol{x}}_{n,n} - \left(\sqrt{(N+\kappa)\,\boldsymbol{P}_{n,n}}\right)_{i-N}, \qquad i = N+1, \cdots, 2N
\end{aligned} \tag{14.6}$$

Where:

$N$ is the number of dimensions

$\kappa$ (the Greek letter kappa) is a tuning parameter

$\left(\sqrt{(N+\kappa)\,\boldsymbol{P}_{n,n}}\right)$ is the $i^{th}$ row or column of the matrix square root of $\sqrt{(N+\kappa)\,\boldsymbol{P}_{n,n}}$

For Gaussian distribution, the rule of thumb is to set: $N + \kappa = 3$.

- Step 2 - Propagate each selected point through the non-linear function, producing a new set of points belonging to the output distribution.

**Sigma Points Propagation**

$$\mathcal{X}_{n+1,n} = \boldsymbol{f}\left(\mathcal{X}_{n,n}\right) \tag{14.7}$$

- Step 3 - Compute sigma points weights.

**Sigma Points Weights**

$$\begin{aligned}
w_0 &= \kappa/(N+\kappa) \\
w_i &= 1/2(N+\kappa), \qquad i > 0
\end{aligned} \tag{14.8}$$

Where:

$w_0$ - weight of the first sigma point $\left(\mathcal{X}_{n,n}^{(0)}\right)$

$w_1$ - weight of the other sigma points $\left(\mathcal{X}_{n,n}^{(i)}\right)$, $\quad i > 0$

- Step 4 - Approximate mean and covariance of the output distribution:

**Mean and covariance of the output distribution**

$$\hat{\boldsymbol{x}}_{n+1,n} = \sum_{i=0}^{2N} w_i \mathcal{X}_{n+1,n}^{(i)}$$
$$\boldsymbol{P}_{n+1,n} = \sum_{i=0}^{2N} w_i \left(\mathcal{X}_{n+1,n}^{(i)} - \hat{\boldsymbol{x}}_{n+1,n}\right)\left(\mathcal{X}_{n+1,n}^{(i)} - \hat{\boldsymbol{x}}_{n+1,n}\right)^T \tag{14.9}$$

Where:

$$\mathcal{X}_{n+1,n} = \begin{bmatrix} \mathcal{X}_{n+1,n}^{(0)} & \mathcal{X}_{n+1,n}^{(1)} & \cdots & \mathcal{X}_{n+1,n}^{(2N)} \end{bmatrix}$$
$$\boldsymbol{w} = \begin{bmatrix} w_0 & w_1 & \cdots w_{2N} \end{bmatrix} \tag{14.10}$$

## 14.2 The UKF algorithm - Predict Stage

Like the Linear Kalman Filter, UKF operates in a "predict–correct" loop. We start with the prediction stage.

Extrapolate the current estimate to the text state using the Unscented Transform:

$$\hat{\boldsymbol{x}}_{n,n} \rightarrow \hat{\boldsymbol{x}}_{n+1,n}$$
$$\boldsymbol{P}_{n,n} \rightarrow \boldsymbol{P}_{n+1,n}$$

The following table compares the LKF and UKF predict stage equations:

| LKF | UKF |
|---|---|
| | $\mathcal{X}_{n,n}^{(0)} = \hat{\boldsymbol{x}}_{n,n}$ |
| | $\mathcal{X}_{n,n}^{(i)} = \hat{\boldsymbol{x}}_{n,n} + \left(\sqrt{(N+\kappa)\,\boldsymbol{P}_{n,n}}\right)_i, \quad i = 1, \cdots, N$ |
| | $\mathcal{X}_{n,n}^{(i)} = \hat{\boldsymbol{x}}_{n,n} - \left(\sqrt{(N+\kappa)\,\boldsymbol{P}_{n,n}}\right)_{i-N}, \quad i = N+1, \cdots, 2N$ |
| $\hat{\boldsymbol{x}}_{n+1,n} = \boldsymbol{F}\hat{\boldsymbol{x}}_{n,n}$ | $\mathcal{X}_{n+1,n} = \boldsymbol{f}\left(\mathcal{X}_{n,n}\right)$ |
| $\boldsymbol{P}_{n+1,n} = \boldsymbol{F}\boldsymbol{P}_{n,n}\boldsymbol{F}^T + \boldsymbol{Q}$ | $\hat{\boldsymbol{x}}_{n+1,n} = \sum_{i=0}^{2N} w_i \mathcal{X}_{n+1,n}^{(i)}$ |
| | $\boldsymbol{P}_{n+1,n} = \sum_{i=0}^{2N} w_i \left(\mathcal{X}_{n+1,n}^{(i)} - \hat{\boldsymbol{x}}_{n+1,n}\right)\left(\mathcal{X}_{n+1,n}^{(i)} - \hat{\boldsymbol{x}}_{n+1,n}\right)^T$ |

**Table 14.1:** *LKF and UKF predict stage equations.*

## 14.3 Statistical linear regression

Before proceeding to the update stage (section 14.4), we must understand the **statistical linear regression** concept [10], [11].

Consider a non-linear function $\boldsymbol{y} = \boldsymbol{g}(\boldsymbol{x})$ evaluated in $r$ points $\left(\mathcal{X}^{(i)}, \mathcal{Y}^{(i)}\right)$, where $\mathcal{Y}^{(i)} = \boldsymbol{g}\left(\mathcal{X}^{(i)}\right)$.

Define:

| | |
|---|---|
| $\overline{\boldsymbol{x}} = \frac{1}{r}\sum_{i=1}^{r}\mathcal{X}^{(i)}$ | Mean of $\mathcal{X}^{(i)}$ |
| $\overline{\boldsymbol{y}} = \frac{1}{r}\sum_{i=1}^{r}\mathcal{Y}^{(i)}$ | Mean of $\mathcal{Y}^{(i)}$ |
| $\boldsymbol{P}_{xx} = \frac{1}{r}\sum_{i=1}^{r}\left(\mathcal{X}^{(i)} - \overline{\boldsymbol{x}}\right)\left(\mathcal{X}^{(i)} - \overline{\boldsymbol{x}}\right)^T$ | Variance of $\mathcal{X}^{(i)}$ |
| $\boldsymbol{P}_{yy} = \frac{1}{r}\sum_{i=1}^{r}\left(\mathcal{Y}^{(i)} - \overline{\boldsymbol{y}}\right)\left(\mathcal{Y}^{(i)} - \overline{\boldsymbol{y}}\right)^T$ | Variance of $\mathcal{Y}^{(i)}$ |
| $\boldsymbol{P}_{xy} = \frac{1}{r}\sum_{i=1}^{r}\left(\mathcal{X}^{(i)} - \overline{\boldsymbol{x}}\right)\left(\mathcal{Y}^{(i)} - \overline{\boldsymbol{y}}\right)^T$ | Cross-variance of $\mathcal{X}^{(i)}$ and $\mathcal{Y}^{(i)}$ |
| $\boldsymbol{P}_{yx} = \frac{1}{r}\sum_{i=1}^{r}\left(\mathcal{Y}^{(i)} - \overline{\boldsymbol{y}}\right)\left(\mathcal{X}^{(i)} - \overline{\boldsymbol{x}}\right)^T$ | Cross-variance of $\mathcal{Y}^{(i)}$ and $\mathcal{X}^{(i)}$ |

**Table 14.2:** *Definitions.*

We want to approximate the non-linear function $\boldsymbol{y} = \boldsymbol{g}(\boldsymbol{x})$ by a linear function $\boldsymbol{y} = \boldsymbol{M}\boldsymbol{x} + \boldsymbol{b}$.

The linear approximation produces linearization error. For each point $\mathcal{X}^{(i)}$, the linearization error is given by:

$$\boldsymbol{e}^{(i)} = \mathcal{Y}^{(i)} - \left(\boldsymbol{M}\mathcal{X}^{(i)} + \boldsymbol{b}\right) \tag{14.11}$$

To minimize the linearization error, we should find $\boldsymbol{M}$ and $\boldsymbol{b}$ that minimize the sum of squared errors for all $\mathcal{X}^{(i)}$ points:

$$\min_{\boldsymbol{M},\boldsymbol{b}} \sum_{i=1}^{r}\left((\boldsymbol{e}^{(i)})^T\boldsymbol{e}^{(i)}\right) \tag{14.12}$$

The solution of Equation 14.12:

$$\begin{aligned}\boldsymbol{M} &= \boldsymbol{P}_{xy}^T\boldsymbol{P}_{xx}^{-1} = \boldsymbol{P}_{yx}\boldsymbol{P}_{xx}^{-1}\\ \boldsymbol{b} &= \overline{\boldsymbol{z}} - \boldsymbol{M}\overline{\boldsymbol{x}}\end{aligned} \tag{14.13}$$

The derivation of Equation 14.13 is shown in Appendix F.

## 14.4 The UKF algorithm - Update Stage

### 14.4.1 State update

After a unit delay the $\mathcal{X}_{n+1,n}$ becomes $\mathcal{X}_{n,n-1}$ , and $\hat{\boldsymbol{x}}_{n+1,n}$ becomes $\hat{\boldsymbol{x}}_{n,n-1}$.

Using the Unscented Transform, transfer the state sigma points ($\mathcal{X}_{n,n-1}$) to the measurement space ($\mathcal{Z}$) using the measurement function $\boldsymbol{h}(\boldsymbol{x})$:

$$\mathcal{Z}_n = \boldsymbol{h}\left(\mathcal{X}_{n,n-1}\right) \tag{14.14}$$

$$\overline{\boldsymbol{z}}_n = \sum_{i=0}^{2N} w_i \mathcal{Z}_n^{(i)} \tag{14.15}$$

Update estimate with measurement:

**State Update**

$$\hat{\boldsymbol{x}}_{n,n} = \hat{\boldsymbol{x}}_{n,n-1} + \boldsymbol{K}_n\left(\boldsymbol{z}_n - \overline{\boldsymbol{z}}_n\right) \tag{14.16}$$

### 14.4.2 Kalman gain derivation

The Kalman Gain ($\boldsymbol{K}_n$) transforms the innovation ($\boldsymbol{z}_n - \overline{\boldsymbol{z}}_n$) and the measurement covariance ($\boldsymbol{P}_{z_n}$) from the measurement space to the system space using linearization:

$$innovation = \left(\boldsymbol{M}\boldsymbol{z}_n + \boldsymbol{b}\right) - \left(\boldsymbol{M}\overline{\boldsymbol{z}}_n + \boldsymbol{b}\right) = \boldsymbol{M}\left(\boldsymbol{z}_n - \overline{\boldsymbol{z}}_n\right) \tag{14.17}$$

Since the Kalman Gain performs a linear transformation, it produces linearization errors. We want to minimize the linearization errors using the statistical linear regression method. Therefore, the Kalman gain is given by:

**Kalman gain**

$$\boldsymbol{K}_n = \boldsymbol{M} = \boldsymbol{P}_{xz_n}\left(\boldsymbol{P}_{z_n}\right)^{-1} \tag{14.18}$$

Compute the weighted variance of the measurement ($\boldsymbol{P}_{z_n}$) and cross-covariance of the state and the measurement ($\boldsymbol{P}_{xz_n}$):

**Weighted variance and cross-covariance**

$$\boldsymbol{P}_{z_n} = \sum_{i=0}^{2N} w_i \left(\mathcal{Z}_n^{(i)} - \overline{\boldsymbol{z}}_n\right)\left(\mathcal{Z}_n^{(i)} - \overline{\boldsymbol{z}}_n\right)^T + \boldsymbol{R}_n$$
$$\boldsymbol{P}_{xz_n} = \sum_{i=0}^{2N} w_i \left(\mathcal{X}_{n,n-1}^{(i)} - \hat{\boldsymbol{x}}_{n,n-1}\right)\left(\mathcal{Z}_n^{(i)} - \overline{\boldsymbol{z}}_n\right)^T \tag{14.19}$$

### 14.4.3 Covariance update equation

The covariance update equation is given by:

**Covariance update equation**

$$\boldsymbol{P}_{n,n} = \boldsymbol{P}_{n-1,n} - \boldsymbol{K}_n \boldsymbol{P}_{z_n} \boldsymbol{K}_n^T \tag{14.20}$$

The following is a derivation of the covariance update equation.

Table 14.3: Covariance Update Equation derivation.

| Equation | Notes |
|---|---|
| $\boldsymbol{P}_{n,n} = E(\boldsymbol{e}_n \boldsymbol{e}_n^T)$<br>$= E\Big((\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n})(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n})^T\Big)$ | Estimate Covariance |
| $= E\Big(\big(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n-1} - \boldsymbol{K}_n\left(\boldsymbol{z}_n - \overline{\boldsymbol{z}}_n\right)\big)$<br>$\times\big(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n-1} - \boldsymbol{K}_n\left(\boldsymbol{z}_n - \overline{\boldsymbol{z}}_n\right)\big)^T\Big)$ | Plug $\hat{\boldsymbol{x}}_{n,n} = \hat{\boldsymbol{x}}_{n,n-1} + \boldsymbol{K}_n\left(\boldsymbol{z}_n - \overline{\boldsymbol{z}}_n\right)$ |

Continued on next page

Table 14.3: Covariance Update Equation derivation. (Continued)

| | |
|---|---|
| $= E\Big( \left(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n-1}\right) \left(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n-1}\right)^T$<br>$- \left(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n-1}\right) \left(\boldsymbol{K}_n \left(\boldsymbol{z}_n - \overline{\boldsymbol{z}}_n\right)\right)^T$<br>$-\boldsymbol{K}_n \left(\boldsymbol{z}_n - \overline{\boldsymbol{z}}_n\right) \left(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n-1}\right)^T$<br>$+\boldsymbol{K}_n \left(\boldsymbol{z}_n - \overline{\boldsymbol{z}}_n\right) \left(\boldsymbol{K}_n \left(\boldsymbol{z}_n - \overline{\boldsymbol{z}}_n\right)\right)^T \Big)$ | Expand |
| $= \boldsymbol{P}_{n,n-1}$<br>$-E\Big( \left(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n-1}\right) \left(\boldsymbol{K}_n \left(\boldsymbol{z}_n - \overline{\boldsymbol{z}}_n\right)\right)^T$<br>$+\boldsymbol{K}_n \left(\boldsymbol{z}_n - \overline{\boldsymbol{z}}_n\right) \left(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n-1}\right)^T$<br>$-\boldsymbol{K}_n \left(\boldsymbol{z}_n - \overline{\boldsymbol{z}}_n\right) \left(\boldsymbol{K}_n \left(\boldsymbol{z}_n - \overline{\boldsymbol{z}}_n\right)\right)^T \Big)$ | $\boldsymbol{P}_{n,n-1} =$<br>$= E\Big( \left(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n-1}\right) \left(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n-1}\right)^T \Big)$ |
| $= \boldsymbol{P}_{n,n-1}$<br>$-E\Big( \left(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n-1}\right) \left(\boldsymbol{z}_n - \overline{\boldsymbol{z}}_n\right)^T \boldsymbol{K}_n^T$<br>$+\boldsymbol{K}_n \left(\boldsymbol{z}_n - \overline{\boldsymbol{z}}_n\right) \left(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n-1}\right)^T$<br>$-\boldsymbol{K}_n \left(\boldsymbol{z}_n - \overline{\boldsymbol{z}}_n\right) \left(\boldsymbol{K}_n \left(\boldsymbol{z}_n - \overline{\boldsymbol{z}}_n\right)\right)^T \Big)$ | Apply the matrix transpose property: $\left(\boldsymbol{AB}\right)^T = \boldsymbol{B}^T \boldsymbol{A}^T$ |
| $= \boldsymbol{P}_{n,n-1}$<br>$-E\Big( \left(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n-1}\right) \left(\boldsymbol{z}_n - \overline{\boldsymbol{z}}_n\right)^T \boldsymbol{K}_n^T$<br>$+\boldsymbol{K}_n \left(\boldsymbol{z}_n - \overline{\boldsymbol{z}}_n\right) \Big( \left(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n-1}\right)^T$<br>$- \left(\boldsymbol{K}_n \left(\boldsymbol{z}_n - \overline{\boldsymbol{z}}_n\right)\right)^T \Big) \Big)$ | Factor out: $\boldsymbol{K}_n \left(\boldsymbol{z}_n - \overline{\boldsymbol{z}}_n\right)$ |
| $= \boldsymbol{P}_{n,n-1}$<br>$-E\Big( \left(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n-1}\right) \left(\boldsymbol{z}_n - \overline{\boldsymbol{z}}_n\right)^T \boldsymbol{K}_n^T$<br>$+\boldsymbol{K}_n \left(\boldsymbol{z}_n - \overline{\boldsymbol{z}}_n\right) \Big( \left(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n-1}\right)^T$<br>$-\left(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n-1}\right)^T \Big) \Big)$ | Plug: $\boldsymbol{K}_n \left(\boldsymbol{z}_n - \overline{\boldsymbol{z}}_n\right) = \boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n-1}$ |

Continued on next page

Table 14.3: Covariance Update Equation derivation. (Continued)

| | |
|---|---|
| $= \boldsymbol{P}_{n,n-1}$<br>$-E\big( (\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n-1}) (\boldsymbol{z}_n - \overline{\boldsymbol{z}}_n)^T \boldsymbol{K}_n^T \big)$ | Remove zero term |
| $\boldsymbol{P}_{n,n} = \boldsymbol{P}_{n,n-1} - \boldsymbol{P}_{xz_n} \boldsymbol{K}_n^T$ | $\boldsymbol{P}_{xz} = E((\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n,n-1}) (\boldsymbol{z}_n - \overline{\boldsymbol{z}}_n)^T)$ |
| $\boldsymbol{P}_{n,n} = \boldsymbol{P}_{n,n-1} - \boldsymbol{K}_n \boldsymbol{P}_{z_n} \boldsymbol{K}_n^T$ | $\boldsymbol{K}_n = \boldsymbol{P}_{xz_n} (\boldsymbol{P}_{z_n})^{-1}$<br>$\boldsymbol{P}_{xz_n} = \boldsymbol{K}_n \boldsymbol{P}_{z_n}$ |

## 14.5 UKF update summary

The following table compares the LKF and UKF update stage equations:

| LKF | UKF |
|---|---|
| | $\mathcal{Z}_n = \boldsymbol{h}\left(\mathcal{X}_{n,n-1}\right)$ |
| | $\boldsymbol{\mu}_{z_n} = \sum_{i=0}^{2N} w_i \mathcal{Z}_n^{(i)}$ |
| $\boldsymbol{K}_n = \boldsymbol{P}_{n,n-1}\boldsymbol{H}^T \times \left(\boldsymbol{H}\boldsymbol{P}_{n,n-1}\boldsymbol{H}^T + \boldsymbol{R}_n\right)^{-1}$ | $\boldsymbol{P}_{z_n} = \sum_{i=0}^{2N} w_i \left(\mathcal{Z}_n^{(i)} - \overline{\boldsymbol{z}}_n\right)\left(\mathcal{Z}_n^{(i)} - \overline{\boldsymbol{z}}_n\right)^T + \boldsymbol{R}_n$ |
| $\hat{\boldsymbol{x}}_{n,n} = \hat{\boldsymbol{x}}_{n,n-1} + \boldsymbol{K}_n\left(\boldsymbol{z}_n - \boldsymbol{H}\hat{\boldsymbol{x}}_{n,n-1}\right)$ | $\boldsymbol{P}_{xz_n} = \sum_{i=0}^{2N} w_i \left(\mathcal{X}_{n,n-1}^{(i)} - \hat{\boldsymbol{x}}_{n,n-1}\right)\left(\mathcal{Z}_n^{(i)} - \overline{\boldsymbol{z}}_n\right)^T$ |
| $\boldsymbol{P}_{n,n} = \left(\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H}\right)\boldsymbol{P}_{n,n-1}\left(\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H}\right)^T + \boldsymbol{K}_n\boldsymbol{R}_n\boldsymbol{K}_n^T$ | $\boldsymbol{K}_n = \boldsymbol{P}_{xz_n}\left(\boldsymbol{P}_{z_n}\right)^{-1}$ |
| | $\hat{\boldsymbol{x}}_{n,n} = \hat{\boldsymbol{x}}_{n,n-1} + \boldsymbol{K}_n\left(\boldsymbol{z}_n - \overline{\boldsymbol{z}}_n\right)$ |
| | $\boldsymbol{P}_{n,n} = \boldsymbol{P}_{n-1,n} - \boldsymbol{K}_n\boldsymbol{P}_{z_n}\boldsymbol{K}_n^T$ |

**Table 14.4:** *LKF and UKF update stage equations.*

## 14.6 UKF algorithm summary

The following diagram describes the UKF algorithm.

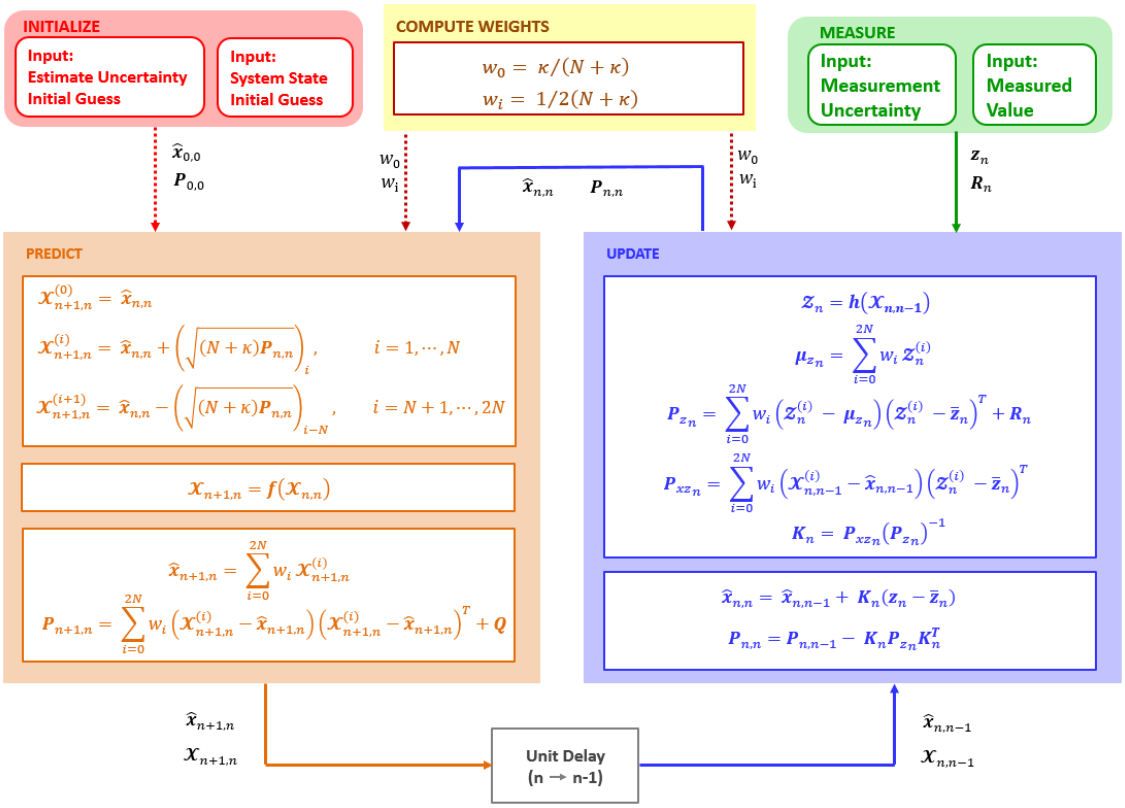


**Figure 14.7:** *UKF algorithm diagram.*

## 14.7 Example 13 – vehicle location estimation using radar

This example is identical to example 11 (section 13.7) with one difference. We use UKF instead of EKF.

The state transition matrix $\boldsymbol{F}$, the process noise matrix $\boldsymbol{Q}$, the measurement covariance $\boldsymbol{R}$, and the measurement model $\boldsymbol{h}(\boldsymbol{x})$ are similar to example 11 (section 13.7).

$$\boldsymbol{F} = \begin{bmatrix} 1 & \Delta t & 0.5\Delta t^2 & 0 & 0 & 0 \\ 0 & 1 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta t & 0.5\Delta t^2 \\ 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0.5 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0.5 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{Q} = \begin{bmatrix} \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} & \frac{\Delta t^2}{2} & 0 & 0 & 0 \\ \frac{\Delta t^3}{2} & \Delta t^2 & \Delta t & 0 & 0 & 0 \\ \frac{\Delta t^2}{2} & \Delta t & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} & \frac{\Delta t^2}{2} \\ 0 & 0 & 0 & \frac{\Delta t^3}{2} & \Delta t^2 & \Delta t \\ 0 & 0 & 0 & \frac{\Delta t^2}{2} & \Delta t & 1 \end{bmatrix} \sigma_a^2 = \begin{bmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & 1 & 1 & 0 & 0 & 0 \\ \frac{1}{2} & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{4} & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & \frac{1}{2} & 1 & 1 \\ 0 & 0 & 0 & \frac{1}{2} & 1 & 1 \end{bmatrix} 0.2^2$$

$$\boldsymbol{z}_n = \boldsymbol{h}(\boldsymbol{x}_n)$$

$$\begin{bmatrix} r \\ \varphi \end{bmatrix} = \begin{bmatrix} \sqrt{x^2 + y^2} \\ tan^{-1}\frac{y}{x} \end{bmatrix}$$

$$\boldsymbol{R}_n = \begin{bmatrix} \sigma_{r_m}^2 & 0 \\ 0 & \sigma_{\varphi_m}^2 \end{bmatrix} = \begin{bmatrix} 5^2 & 0 \\ 0 & 0.0087^2 \end{bmatrix}$$

### 14.7.1 The numerical example

The radar measurements are also identical to example 11 (section 13.7):

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| **r(m)** | 502.55 | 477.34 | 457.21 | 442.94 | 427.27 | 406.05 | 400.73 | 377.32 |
| **$\varphi$(rad)** | -0.9316 | -0.8977 | -0.8512 | -0.8114 | -0.7853 | -0.7392 | -0.7052 | -0.6478 |
| **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** | **17** |
| 360.27 | 345.93 | 333.34 | 328.07 | 315.48 | 301.41 | 302.87 | 304.25 | 294.46 |
| -0.59 | -0.5183 | -0.4698 | -0.3952 | -0.3026 | -0.2445 | -0.1626 | -0.0937 | 0.0085 |
| **18** | **19** | **20** | **21** | **22** | **23** | **24** | **25** | **26** |
| 294.29 | 299.38 | 299.37 | 300.68 | 304.1 | 301.96 | 300.3 | 301.9 | 296.7 |
| 0.0856 | 0.1675 | 0.2467 | 0.329 | 0.4149 | 0.504 | 0.5934 | 0.667 | 0.7537 |
| **27** | **28** | **29** | **30** | **31** | **32** | **33** | **34** | **35** |
| 297.07 | 295.29 | 296.31 | 300.62 | 292.3 | 298.11 | 298.07 | 298.92 | 298.04 |
| 0.8354 | 0.9195 | 1.0039 | 1.0923 | 1.1546 | 1.2564 | 1.3274 | 1.409 | 1.5011 |

**Table 14.5:** *Example 13 measurements.*

#### 14.7.1.1 UKF parameters computation

- The number of dimensions: $N = 6$
- The number of sigma points: $2N + 1 = 13$
- The input variable distribution is Gaussian: $N + \kappa = 3 \Longrightarrow \kappa = -3$

**Weights Calculation**

$w_0 = \kappa/(N + \kappa) = -1$
$w_1 = 1/2(N + \kappa) = 1/6, \quad i > 0$

$$\boldsymbol{w} = \begin{bmatrix} -1 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \end{bmatrix}$$

$$diag(\boldsymbol{w}) = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{6} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{6} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{6} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{6} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{6} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{6} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{6} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{6} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{6} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{6} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{6} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{6} \end{bmatrix}$$

#### 14.7.1.2 Iteration Zero

##### Initialization

The filter initialized to the same values: $(\hat{x}_{0,0} = 400m, \hat{y}_{0,0} = -300m)$. Initial velocity and acceleration are 0.

$$\hat{\boldsymbol{x}}_{0,0} = \begin{bmatrix} 400 \\ 0 \\ 0 \\ -300 \\ 0 \\ 0 \end{bmatrix}$$

$$\boldsymbol{P}_{0,0} = \begin{bmatrix} 500 & 0 & 0 & 0 & 0 & 0 \\ 0 & 500 & 0 & 0 & 0 & 0 \\ 0 & 0 & 500 & 0 & 0 & 0 \\ 0 & 0 & 0 & 500 & 0 & 0 \\ 0 & 0 & 0 & 0 & 500 & 0 \\ 0 & 0 & 0 & 0 & 0 & 500 \end{bmatrix}$$

### Prediction

#### Sigma points computation

$$\mathcal{X}_{0,0}^{(0)} = \hat{\boldsymbol{x}}_{0,0} = \begin{bmatrix} 400 \\ 0 \\ 0 \\ -300 \\ 0 \\ 0 \end{bmatrix}$$

To find other sigma points, we should compute the square root:

$$\sqrt{(N+\kappa)\,\boldsymbol{P}_{0,0}} = \sqrt{3 \begin{bmatrix} 500 & 0 & 0 & 0 & 0 & 0 \\ 0 & 500 & 0 & 0 & 0 & 0 \\ 0 & 0 & 500 & 0 & 0 & 0 \\ 0 & 0 & 0 & 500 & 0 & 0 \\ 0 & 0 & 0 & 0 & 500 & 0 \\ 0 & 0 & 0 & 0 & 0 & 500 \end{bmatrix}}$$

We can use the Cholesky decomposition:

The manual computation of Cholesky decomposition is quite tedious. I provide here Python and MATLAB examples.

**Python example:**

```
import numpy as np

P = 3*np.diag(np.tile(500,6))

print(P)

[[1500    0    0    0    0    0]
 [   0 1500    0    0    0    0]
 [   0    0 1500    0    0    0]
 [   0    0    0 1500    0    0]
 [   0    0    0    0 1500    0]
 [   0    0    0    0    0 1500]]

L = np.linalg.cholesky(P)

print(L)

[[38.72983346  0.          0.          0.          0.          0.        ]
 [ 0.         38.72983346  0.          0.          0.          0.        ]
 [ 0.          0.         38.72983346  0.          0.          0.        ]
 [ 0.          0.          0.         38.72983346  0.          0.        ]
 [ 0.          0.          0.          0.         38.72983346  0.        ]
 [ 0.          0.          0.          0.          0.         38.72983346]]
```

**MATLAB example:**

```
P = 3*diag(repmat(500,1,6))

P =

        1500           0           0           0           0
           0
           0        1500           0           0           0
           0
           0           0        1500           0           0
           0
           0           0           0        1500           0
           0
           0           0           0           0        1500
           0
           0           0           0           0           0
        1500

L = chol(P)'

L =

   38.7298         0         0         0         0         0
         0   38.7298         0         0         0         0
         0         0   38.7298         0         0         0
         0         0         0   38.7298         0         0
         0         0         0         0   38.7298         0
         0         0         0         0         0   38.7298
```

$$\mathcal{X}_{0,0}^{(1)} = \hat{\boldsymbol{x}}_{0,0} + \left(\sqrt{(N+\kappa)\,\boldsymbol{P}_{0,0}}\right)_1 = \begin{bmatrix} 400 \\ 0 \\ 0 \\ -300 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 38.73 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 438.73 \\ 0 \\ 0 \\ -300 \\ 0 \\ 0 \end{bmatrix}$$

$$\mathcal{X}_{0,0}^{(2)} = \hat{\boldsymbol{x}}_{0,0} + \left(\sqrt{(N+\kappa)\,\boldsymbol{P}_{0,0}}\right)_2 = \begin{bmatrix} 400 \\ 0 \\ 0 \\ -300 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 38.73 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 400 \\ 38.73 \\ 0 \\ -300 \\ 0 \\ 0 \end{bmatrix}$$

Similarly, we can compute $\mathcal{X}_{0,0}^{(3)}, \cdots, \mathcal{X}_{0,0}^{(6)}$.

$$\mathcal{X}_{0,0}^{(7)} = \hat{\boldsymbol{x}}_{0,0} - \left(\sqrt{(N+\kappa)\,\boldsymbol{P}_{0,0}}\right)_1 = \begin{bmatrix} 400 \\ 0 \\ 0 \\ -300 \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 38.73 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 361.27 \\ 0 \\ 0 \\ -300 \\ 0 \\ 0 \end{bmatrix}$$

$$\mathcal{X}_{0,0}^{(8)} = \hat{\boldsymbol{x}}_{0,0} - \left(\sqrt{(N+\kappa)\,\boldsymbol{P}_{0,0}}\right)_2 = \begin{bmatrix} 400 \\ 0 \\ 0 \\ -300 \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 38.73 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 400 \\ -38.73 \\ 0 \\ -300 \\ 0 \\ 0 \end{bmatrix}$$

Similarly, we can compute $\mathcal{X}_{0,0}^{(9)}, \cdots, \mathcal{X}_{0,0}^{(12)}$.

Now, stack all sigma points in a matrix:

$$\mathcal{X}_{0,0} = \begin{bmatrix} 400 & 438.73 & 400 & 400 & 400 & 400 & 400 & 361.27 & 400 & 400 & 400 & 400 & 400 \\ 0 & 0 & 38.73 & 0 & 0 & 0 & 0 & 0 & -38.73 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 38.73 & 0 & 0 & 0 & 0 & 0 & -38.73 & 0 & 0 & 0 \\ -300 & -300 & -300 & -300 & -261.27 & -300 & -300 & -300 & -300 & -300 & -338.73 & -300 & -300 \\ 0 & 0 & 0 & 0 & 0 & 38.73 & 0 & 0 & 0 & 0 & 0 & -38.73 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 38.73 & 0 & 0 & 0 & 0 & 0 & -38.73 \end{bmatrix}$$

### Points propagation

Propagate each selected point through the non-linear function, producing a new set of points belonging to the output distribution.

$$\mathcal{X}_{1,0} = \boldsymbol{f}\left(\mathcal{X}_{0,0}\right)$$

Since the dynamic model is linear, we can write:

$$\mathcal{X}_{1,0} = \boldsymbol{F}\mathcal{X}_{0,0}$$

$$\mathcal{X}_{1,0} = \begin{bmatrix} 400 & 438.73 & 438.73 & 419.36 & 400 & 400 & 400 & 361.27 & 361.27 & 380.64 & 400 & 400 & 400 \\ 0 & 0 & 38.73 & 38.73 & 0 & 0 & 0 & 0 & -38.73 & -38.73 & 0 & 0 & 0 \\ 0 & 0 & 0 & 38.73 & 0 & 0 & 0 & 0 & 0 & -38.73 & 0 & 0 & 0 \\ -300 & -300 & -300 & -300 & -261.27 & -261.27 & -280.64 & -300 & -300 & -300 & -338.73 & -338.73 & -319.36 \\ 0 & 0 & 0 & 0 & 0 & 38.73 & 38.73 & 0 & 0 & 0 & 0 & -38.73 & -38.73 \\ 0 & 0 & 0 & 0 & 0 & 0 & 38.73 & 0 & 0 & 0 & 0 & 0 & -38.73 \end{bmatrix}$$

### Mean and covariance computation

$$\hat{\boldsymbol{x}}_{1,0} = \mathcal{X}_{1,0}\boldsymbol{w} = \begin{bmatrix} 400 \\ 0 \\ 0 \\ -300 \\ 0 \\ 0 \end{bmatrix}$$

$$\boldsymbol{P}_{1,0} = (\mathcal{X}_{1,0} - \hat{\boldsymbol{x}}_{1,0}) \cdot diag(\boldsymbol{w}) \cdot (\mathcal{X}_{1,0} - \hat{\boldsymbol{x}}_{1,0})^T + \boldsymbol{Q}$$

$$= \begin{bmatrix} 1125.01 & 750.02 & 250.02 & 0 & 0 & 0 \\ 750.02 & 1000.04 & 500.04 & 0 & 0 & 0 \\ 250.02 & 500.04 & 500.04 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1125.01 & 750.02 & 250.02 \\ 0 & 0 & 0 & 750.02 & 1000.04 & 500.04 \\ 0 & 0 & 0 & 250.02 & 500.04 & 500.04 \end{bmatrix}$$

#### 14.7.1.3 First Iteration

##### Update

Using the Unscented Transform, transfer the state sigma points $\mathcal{X}_{1,0}$ to the measurement space $\mathcal{Z}$ using the measurement function $\boldsymbol{h}(\boldsymbol{x})$:

$$\mathcal{Z}_1 = \boldsymbol{h}\left(\mathcal{X}_{1,0}\right)$$

In order to compute $\mathcal{Z}_1$, we perform elementwise operations between the first row of $\mathcal{X}_{1,0}$ that represents the $x$ position ($\mathcal{X}_{1,0}\left(1,:\right)$) and the fourth row of $\mathcal{X}_{1,0}$ that represents the $y$ position ($\mathcal{X}_{1,0}\left(4,:\right)$):

$$\mathcal{Z}_1 = \boldsymbol{h}\left(\mathcal{X}_{1,0}\right) = \begin{bmatrix} \sqrt{\mathcal{X}_{1,0}\left(1,:\right)^2 + \mathcal{X}_{1,0}\left(4,:\right)^2} \\ tan^{-1}\dfrac{\mathcal{X}_{1,0}\left(4,:\right)}{\mathcal{X}_{1,0}\left(1,:\right)} \end{bmatrix}$$

$$\mathcal{Z}_1 = \begin{bmatrix} 500 & 531.49 & 531.49 & 515.62 & 477.77 & 477.77 & 488.63 & 469.59 & 469.59 & 484.65 & 524.15 & 524.15 & 511.85 \\ -0.64 & -0.6 & -0.6 & -0.62 & -0.58 & -0.58 & -0.61 & -0.69 & -0.69 & -0.67 & -0.7 & -0.7 & -0.67 \end{bmatrix}$$

Multiply $\mathcal{Z}_1$ by weights.

$$\overline{\boldsymbol{z}}_1 = \mathcal{Z}_1\boldsymbol{w} = \begin{bmatrix} 501.13 \\ -0.64 \end{bmatrix}$$

Compute covariance at the measurement space:

$$\boldsymbol{P}_{z_1} = (\mathcal{Z}_1 - \overline{\boldsymbol{z}}_1) \cdot diag(\boldsymbol{w}) \cdot (\mathcal{Z}_1 - \overline{\boldsymbol{z}}_1)^T + \boldsymbol{R} = \begin{bmatrix} 1146.7 & 0.0039 \\ 0.0039 & 0.0046 \end{bmatrix}$$

Compute the cross-covariance of the state and the measurement:

$$\boldsymbol{P}_{xz_1} = (\mathcal{X}_{1,0} - \hat{\boldsymbol{x}}_{1,0}) \cdot diag(\boldsymbol{w}) \cdot (\mathcal{Z}_1 - \overline{\boldsymbol{z}}_1)^T = \begin{bmatrix} 899.1 & 1.35 \\ 599.5 & 0.9 \\ 199.95 & 0.3 \\ -673.8 & 1.8 \\ -449.3 & 1.2 \\ -149.9 & 0.4 \end{bmatrix}$$

Compute the Kalman gain:

$$\boldsymbol{K}_1 = \boldsymbol{P}_{xz_1} \left(\boldsymbol{P}_{z_1}\right)^{-1} = \begin{bmatrix} 0.78 & 293.6 \\ 0.52 & 195.6 \\ 0.17 & 65.1 \\ -0.59 & 392 \\ -0.39 & 261.3 \\ -0.13 & 87 \end{bmatrix}$$

Update estimate with measurement:

$$\hat{\boldsymbol{x}}_{1,1} = \hat{\boldsymbol{x}}_{1,0} + \boldsymbol{K}_1 \left(\boldsymbol{z}_1 - \overline{\boldsymbol{z}}_1\right) = \begin{bmatrix} 316.5 \\ -55.6 \\ -18.5 \\ -413.8 \\ -75.8 \\ -25.23 \end{bmatrix}$$

Update covariance of the estimate:

$$\boldsymbol{P}_{1,1} = \boldsymbol{P}_{1,0} - \boldsymbol{K}_1 \boldsymbol{P}_{z_1} \boldsymbol{K}_1^T = \begin{bmatrix} 23.5 & 15.7 & 5.3 & -1.2 & -0.63 & -0.046 \\ 15.7 & 510.5 & 336.9 & -0.55 & -0.26 & 0.024 \\ 5.3 & 336.9 & 445.7 & 0.065 & 0.08 & 0.063 \\ -1.2 & -0.55 & 0.065 & 22.03 & 14.7 & 4.9 \\ -0.63 & -0.26 & 0.08 & 14.7 & 509.8 & 336.6 \\ -0.046 & 0.024 & 0.063 & 4.9 & 336.6 & 455.6 \end{bmatrix}$$

### Predict

**Sigma points computation**

$$\mathcal{X}_{1,1}^{(0)} = \hat{\boldsymbol{x}}_{1,1} = \begin{bmatrix} 316.5 \\ -55.6 \\ -18.5 \\ -413.8 \\ -75.8 \\ -25.23 \end{bmatrix}$$

To find other sigma points, we should compute the square root:

$$\sqrt{(N+\kappa)\,\boldsymbol{P}_{1,1}} = \sqrt{3 \begin{bmatrix} 23.5 & 15.7 & 5.3 & -1.2 & -0.63 & -0.046 \\ 15.7 & 510.5 & 336.9 & -0.55 & -0.26 & 0.024 \\ 5.3 & 336.9 & 445.7 & 0.065 & 0.08 & 0.063 \\ -1.2 & -0.55 & 0.065 & 22.03 & 14.7 & 4.9 \\ -0.63 & -0.26 & 0.08 & 14.7 & 509.8 & 336.6 \\ -0.046 & 0.024 & 0.063 & 4.9 & 336.6 & 455.6 \end{bmatrix}}$$

We can use the Cholesky decomposition:

**Python example:**

```
import numpy as np

L = np.linalg.cholesky(3*P)

print(L)

[[ 8.3955  0.       0.       0.       0.       0.     ]
 [ 5.6205  38.7305  0.       0.       0.       0.     ]
 [ 1.8971  25.8212  25.8203  0.       0.       0.     ]
 [-0.4253  0.0194   0.0194   8.1189   0.       0.     ]
 [-0.2251  0.0129   0.0129   5.4145   38.7305  0.     ]
 [-0.0165  0.0043   0.0043   1.8067   25.8212  25.8203]]
```

**MATLAB example:**

```
L = chol(3*P)'

L =

    8.3955         0         0         0         0         0
    5.6206   38.7305         0         0         0         0
    1.8971   25.8212   25.8203         0         0         0
   -0.4253    0.0194    0.0194    8.1189         0         0
   -0.2251    0.0129    0.0129    5.4145   38.7305         0
   -0.0165    0.0043    0.0043    1.8067   25.8212   25.8203
```

$$\mathcal{X}_{1,1}^{(1)} = \hat{\boldsymbol{x}}_{1,1} + \left(\sqrt{(N+\kappa)\,\boldsymbol{P}_{1,1}}\right)_1 = \begin{bmatrix} 316.5 \\ -55.6 \\ -18.5 \\ -413.8 \\ -75.8 \\ -25.23 \end{bmatrix} + \begin{bmatrix} 8.4 \\ 5.62 \\ 1.9 \\ -0.43 \\ -0.23 \\ -0.02 \end{bmatrix} = \begin{bmatrix} 324.92 \\ -50 \\ -16.6 \\ -414.2 \\ -76.06 \\ -25.3 \end{bmatrix}$$

$$\mathcal{X}_{1,1}^{(2)} = \hat{\boldsymbol{x}}_{1,1} + \left(\sqrt{(N+\kappa)\,\boldsymbol{P}_{1,1}}\right)_2 = \begin{bmatrix} 316.5 \\ -55.6 \\ -18.5 \\ -413.8 \\ -75.8 \\ -25.23 \end{bmatrix} + \begin{bmatrix} 0 \\ 38.7 \\ 25.8 \\ 0.02 \\ 0.012 \\ 0.004 \end{bmatrix} = \begin{bmatrix} 316.5 \\ -16.9 \\ 7.3 \\ -413.8 \\ -75.8 \\ -25.3 \end{bmatrix}$$

Similarly, we can compute $\mathcal{X}_{1,1}^{(3)}, \cdots, \mathcal{X}_{1,1}^{(6)}$.

$$\mathcal{X}_{1,1}^{(7)} = \hat{\boldsymbol{x}}_{1,1} - \left(\sqrt{(N+\kappa)\,\boldsymbol{P}_{1,1}}\right)_1 = \begin{bmatrix} 316.5 \\ -55.6 \\ -18.5 \\ -413.8 \\ -75.8 \\ -25.23 \end{bmatrix} - \begin{bmatrix} 8.4 \\ 5.62 \\ 1.9 \\ -0.43 \\ -0.23 \\ -0.02 \end{bmatrix} = \begin{bmatrix} 308.1 \\ -61.24 \\ -20.4 \\ -413.3 \\ -75.6 \\ -25.25 \end{bmatrix}$$

$$\mathcal{X}_{0,0}^{(8)} = \hat{\boldsymbol{x}}_{0,0} - \left(\sqrt{(N+\kappa)\,\boldsymbol{P}_{0,0}}\right)_2 = \begin{bmatrix} 316.5 \\ -55.6 \\ -18.5 \\ -413.8 \\ -75.8 \\ -25.23 \end{bmatrix} - \begin{bmatrix} 0 \\ 38.7 \\ 25.8 \\ 0.02 \\ 0.012 \\ 0.004 \end{bmatrix} = \begin{bmatrix} 316.52 \\ -94.35 \\ -44.3 \\ -413.8 \\ -75.9 \\ -25.3 \end{bmatrix}$$

Similarly, we can compute $\mathcal{X}_{1,1}^{(9)}, \cdots, \mathcal{X}_{1,1}^{(12)}$.

Now, stack all sigma points in a matrix:

$$\mathcal{X}_{1,1} = \begin{bmatrix} 316.52 & 324.9 & 316.52 & 316.52 & 316.52 & 316.52 & 316.52 & 308.13 & 316.52 & 316.52 & 316.52 & 316.52 & 316.52 \\ -55.63 & -50 & -16.9 & -55.63 & -55.63 & -55.63 & -55.63 & -61.24 & -94.35 & -55.63 & -55.63 & -55.63 & -55.63 \\ -18.5 & -16.6 & 7.31 & 7.31 & -18.5 & -18.5 & -18.5 & -20.4 & -44.3 & -44.3 & -18.5 & -18.5 & -18.5 \\ -413.8 & -414.2 & -413.8 & -413.8 & -405.7 & -413.8 & -413.8 & -413.8 & -413.8 & -413.8 & -421.9 & -413.8 & -413.8 \\ -75.8 & -76.1 & -75.8 & -75.8 & -70.4 & -37.1 & -75.8 & -75.6 & -75.8 & -75.8 & -81.3 & -114.6 & -75.8 \\ -25.3 & -25.3 & -25.3 & -25.3 & -23.5 & 0.55 & 0.55 & -25.3 & -25.3 & -25.3 & -27.1 & -51.1 & -51.1 \end{bmatrix}$$

### Points propagation

Propagate each selected point through the non-linear function, producing a new set

of points belonging to the output distribution.

$$\mathcal{X}_{2,1} = \boldsymbol{f}\left(\mathcal{X}_{1,1}\right)$$

Since the dynamic model is linear, we can write:

$$\mathcal{X}_{2,1} = \boldsymbol{F}\mathcal{X}_{1,1}$$

$$\mathcal{X}_{2,1} = \begin{bmatrix} 251.6 & 266.6 & 303.3 & 264.6 & 251.6 & 251.6 & 251.6 & 236.7 & 200 & 238.8 & 251.6 & 251.6 & 251.6 \\ -74.1 & -66.6 & -9.6 & -48.3 & -74.1 & -74.1 & -74.1 & -81.7 & -138.7 & -100 & -74.1 & -74.1 & -74.1 \\ -18.5 & -16.6 & 7.3 & 7.3 & -18.5 & -18.5 & -18.5 & -20.4 & -44.3 & -44.3 & -18.5 & -18.5 & -18.5 \\ -502.2 & -502.9 & -502.2 & -502.2 & -487.8 & -450.6 & -489.3 & -501.6 & -502.2 & -502.2 & -516.7 & -553.9 & -515.15 \\ -101.1 & -101.3 & -101.1 & -101.1 & -93.8 & -36.55 & -75.3 & -100.9 & -101.1 & -101.1 & -108.3 & -165.7 & -126.9 \\ -25.3 & -25.3 & -25.3 & -25.3 & -23.4 & 0.55 & 0.55 & -25.3 & -25.3 & -25.3 & -27.1 & -51.1 & -51.1 \end{bmatrix}$$

**Mean and covariance computation**

$$\hat{\boldsymbol{x}}_{2,1} = \mathcal{X}_{2,1}\boldsymbol{w} = \begin{bmatrix} 251.6 \\ -74.1 \\ -18.5 \\ -502.2 \\ -101.1 \\ -25.3 \end{bmatrix}$$

$$\boldsymbol{P}_{2,1} = \left(\mathcal{X}_{2,1} - \hat{\boldsymbol{x}}_{2,1}\right) \cdot diag(\boldsymbol{w}) \cdot \left(\mathcal{X}_{2,1} - \hat{\boldsymbol{x}}_{2,1}\right)^T + \boldsymbol{Q}$$

$$= \begin{bmatrix} 1019.1 & 1259.8 & 565 & -2.54 & -0.84 & 0.009 \\ 1259.8 & 1630 & 782.6 & -0.6 & -0.09 & 0.09 \\ 565.1 & 782.6 & 445.7 & 0.17 & 0.14 & 0.06 \\ -2.54 & -0.61 & 0.18 & 1014.1 & 1257.1 & 564.3 \\ -0.84 & -0.09 & 0.14 & 1257.1 & 1628.6 & 782.2 \\ 0.009 & -0.09 & 0.06 & 564.3 & 782.2 & 445.6 \end{bmatrix}$$

#### 14.7.1.4 Next Iterations

I want to encourage the readers to implement this example in software and compare results.

The results of the second iteration:

$$\hat{\boldsymbol{x}}_{2,2} = \begin{bmatrix} 316.2 \\ -5.99 \\ 17.48 \\ -376.84 \\ 54.5 \\ 44.6 \end{bmatrix}$$

$$\boldsymbol{P}_{2,2} = \begin{bmatrix} 24.97 & 30.9 & 13.8 & 0.8 & 1.02 & 0.6 \\ 30.9 & 111 & 101.2 & 1.05 & -0.85 & -0.56 \\ 13.8 & 101.2 & 140 & 0.54 & -0.68 & -0.4 \\ 0.8 & 1.05 & 0.54 & 25.7 & 31.9 & 14.35 \\ 1.02 & -0.85 & -0.68 & 31.9 & 109.8 & 100.5 \\ 0.6 & -0.56 & -0.4 & 14.35 & 100.5 & 139.6 \end{bmatrix}$$

The results of the last (thirty-fifth) iteration:

$$\hat{\boldsymbol{x}}_{35,35} = \begin{bmatrix} 20.87 \\ -25.94 \\ -0.84 \\ 298.4 \\ 2.56 \\ -1.8 \end{bmatrix}$$

$$\boldsymbol{P}_{35,35} = \begin{bmatrix} 4.1 & 1.72 & 0.36 & 0.95 & 0.31 & 0.04 \\ 1.72 & 1.3 & 0.38 & 0.17 & 0.2 & 0.04 \\ 0.36 & 0.38 & 0.16 & -0.01 & 0.03 & 0.008 \\ 0.95 & 0.17 & -0.01 & 12.05 & 4.01 & 0.72 \\ 0.31 & 0.2 & 0.03 & 4.01 & 2.28 & 0.56 \\ 0.04 & 0.04 & 0.008 & 0.72 & 0.56 & 0.19 \end{bmatrix}$$

### 14.7.2 Example summary

Figure 14.8 demonstrates the UKF location and velocity estimation performance.

The chart on the left compares the true, measured, and estimated values of the vehicle position. Two charts on the right compare the true, measured, and estimated values of $x$ and $y$ velocities.

We can see a satisfying performance of the UKF. Although the filter is roughly initiated at about 100 meters from the true position with zero initial velocity, it provides a good position estimation after taking two measurements and a good velocity estimation after taking four measurements.

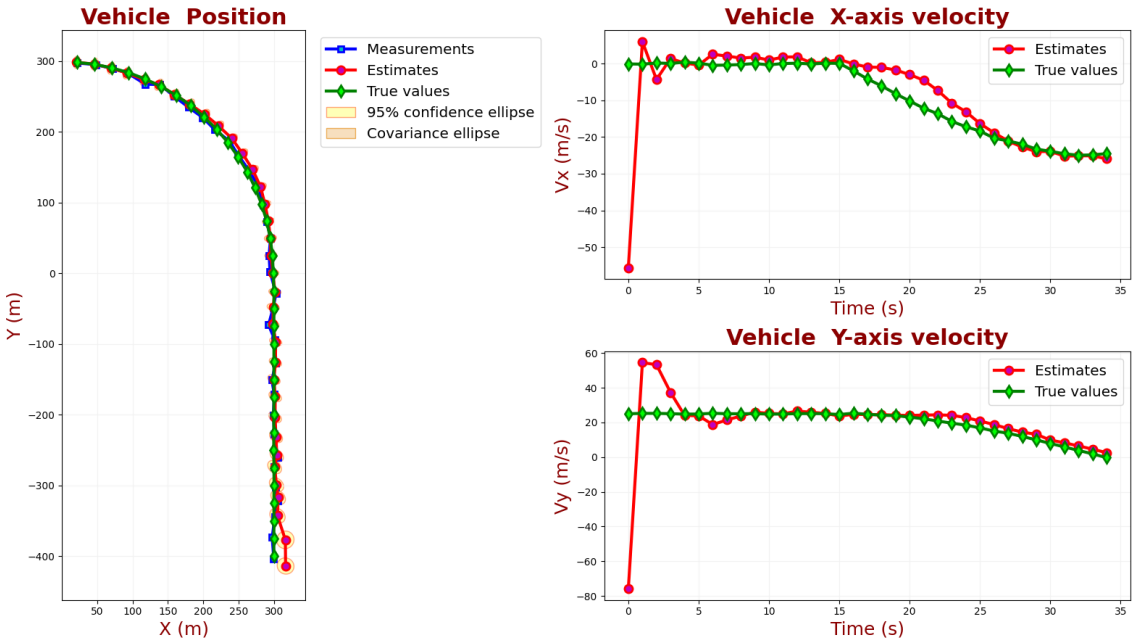


**Figure 14.8:** *Example 13: true value, measured values and estimates.*

Let us take a closer look at the vehicle position estimation performance. The following chart describes the true, measured, and estimated values of the vehicle position compared to the 95% confidence ellipses. We can see that the ellipses' size constantly decreases. That means that the UKF converges with time.

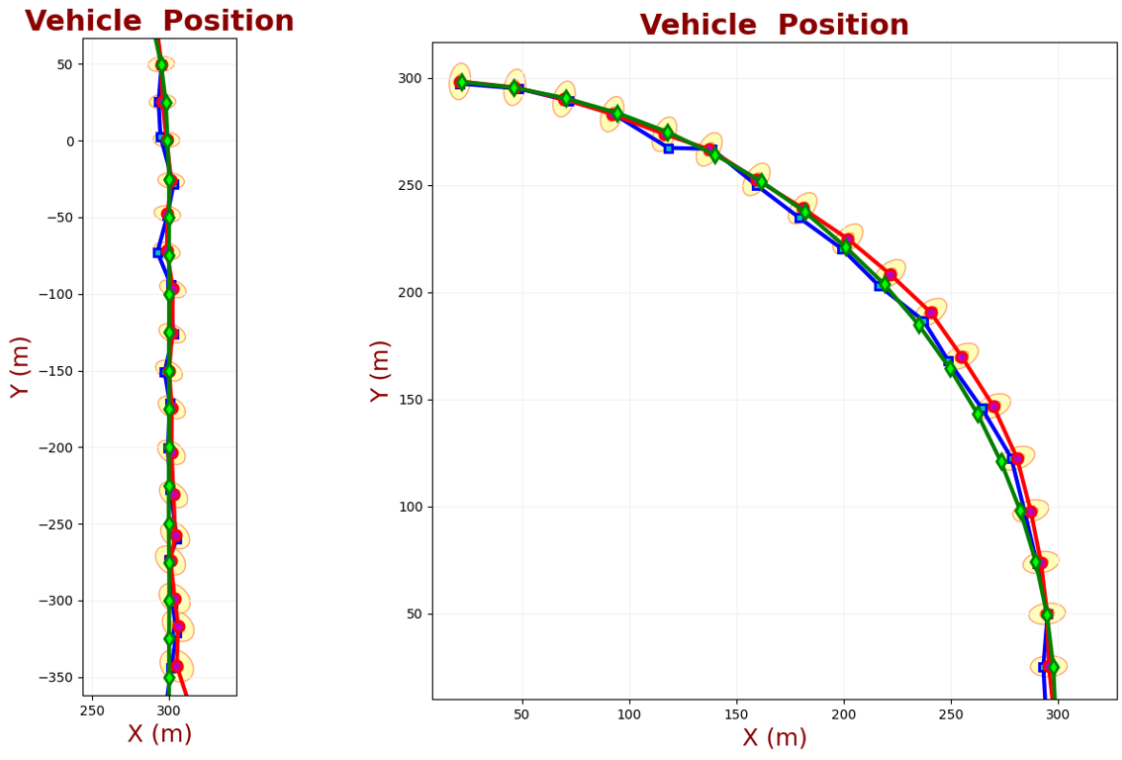


**Figure 14.9:** *Example 13: true value, measured values and estimates - zoom.*

The UKF results are similar to EKF.

We can see that at the linear part of the vehicle motion, the UKF copes with the noisy measurements and follows the true vehicle position. On the other hand, during the vehicle turning maneuver, the UKF estimates are quite away from the true vehicle position, although they are within the 90% confidence ellipse bounds.

## 14.8 Sigma Point Algorithm Modification

Since Jeffrey Uhlmann prosed the Unscented Transform [18], many alternative algorithms for sigma points computation appeared. Eric A. Wan and Rudolph van der Merwe presented the most common and accepted algorithm in their paper [12].

The sigma points computation is parametrized by $\alpha, \beta, \lambda$, and $\kappa$ parameters.

### Sigma Points parameters

$$\lambda = \alpha^2 (N + \kappa) - N \tag{14.21}$$

Where:

- $N$ is the number of dimensions
- $\alpha$ determines the spread of the sigma points around the mean and is usually set to a small positive value (e.g., $\alpha = 0.001$)
- $\kappa$ is a secondary scaling parameter that is usually set to 0
- $\beta$ is used to incorporate prior knowledge of the distribution of the input random variable (for Gaussian distributions, $\beta = 2$ is optimal)
- $\alpha, \beta, \lambda, \kappa$ are tuning parameters

$\alpha$ is a tuning parameter with a range of $0 < \alpha \leq 1$, while higher $\alpha$ provides a higher spread of the sigma points around the mean.

The sigma points are calculated as follows:

### Sigma Points

$$\mathcal{X}_{n,n}^{(0)} = \hat{\boldsymbol{x}}_{n,n} \tag{14.22}$$

$$\begin{aligned} \mathcal{X}_{n,n}^{(i)} &= \hat{\boldsymbol{x}}_{n,n} + \left(\sqrt{(N+\lambda)\,\boldsymbol{P}_{n,n}}\right)_i, \qquad i = 1, \cdots, N \\ \mathcal{X}_{n,n}^{(i)} &= \hat{\boldsymbol{x}}_{n,n} - \left(\sqrt{(N+\lambda)\,\boldsymbol{P}_{n,n}}\right)_{i-N}, \qquad i = N+1, \cdots, 2N \end{aligned} \tag{14.23}$$

$\left(\sqrt{(N+\lambda)\,\boldsymbol{P}_{n,n}}\right)_i$ is the $i^{th}$ row or column of the matrix square root of $(N+\kappa)\,\boldsymbol{P}_{n,n}$.

The sigma points weights:

### Sigma Points weights

$$\begin{aligned} w_0^{(m)} &= \lambda/(N+\kappa) \\ w_0^{(c)} &= \lambda/(N+\lambda) + (1-\alpha^2) + \beta \\ w_i &= 1/2(N+\lambda), \qquad i > 0 \end{aligned} \tag{14.24}$$

Where:

- $w_0^{(m)}$ is a weight for the first sigma point $\mathcal{X}_{n,n}^{(0)}$ when computing the weighted mean
- $w_0^{(c)}$ is a weight for the first sigma point $\mathcal{X}_{n,n}^{(0)}$ when computing the weighted covariance
- $w_i$ is a weight for the other sigma points $\mathcal{X}_{n,n}^{(i)}, i > 0$ when computing the weighted mean or covariance

The following figure describes the sigma points' position for different $\alpha$ values. The highest $\alpha$ provides a higher spread of the sigma points around the mean. The $\alpha$ range is $0 < \alpha \leq 1$.

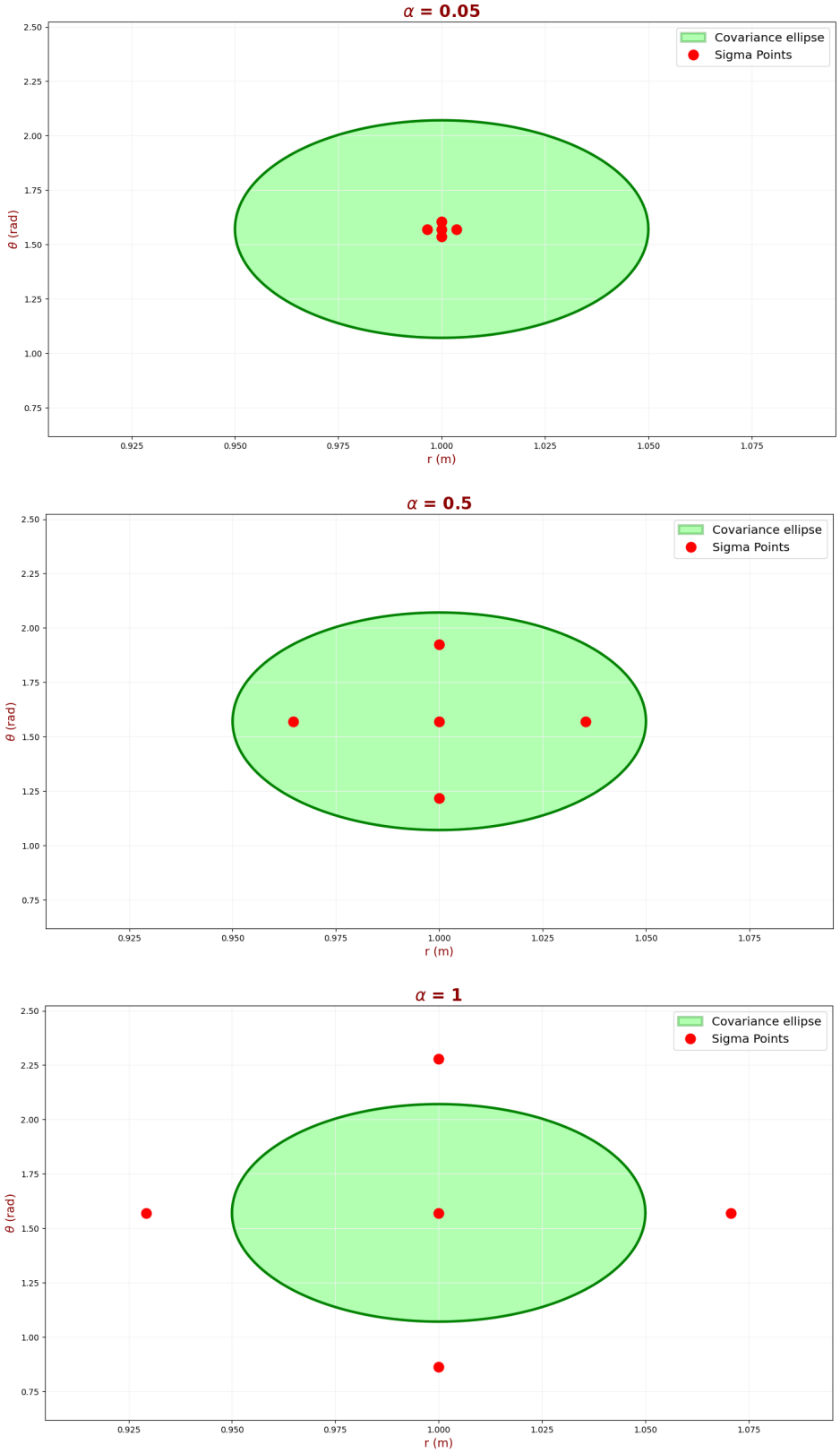


**Figure 14.10:** *$\alpha$ influence on the Sigma Points.*

You should play with $\alpha$ to find the right value for your problem.

## 14.9 Modified UKF algorithm summary

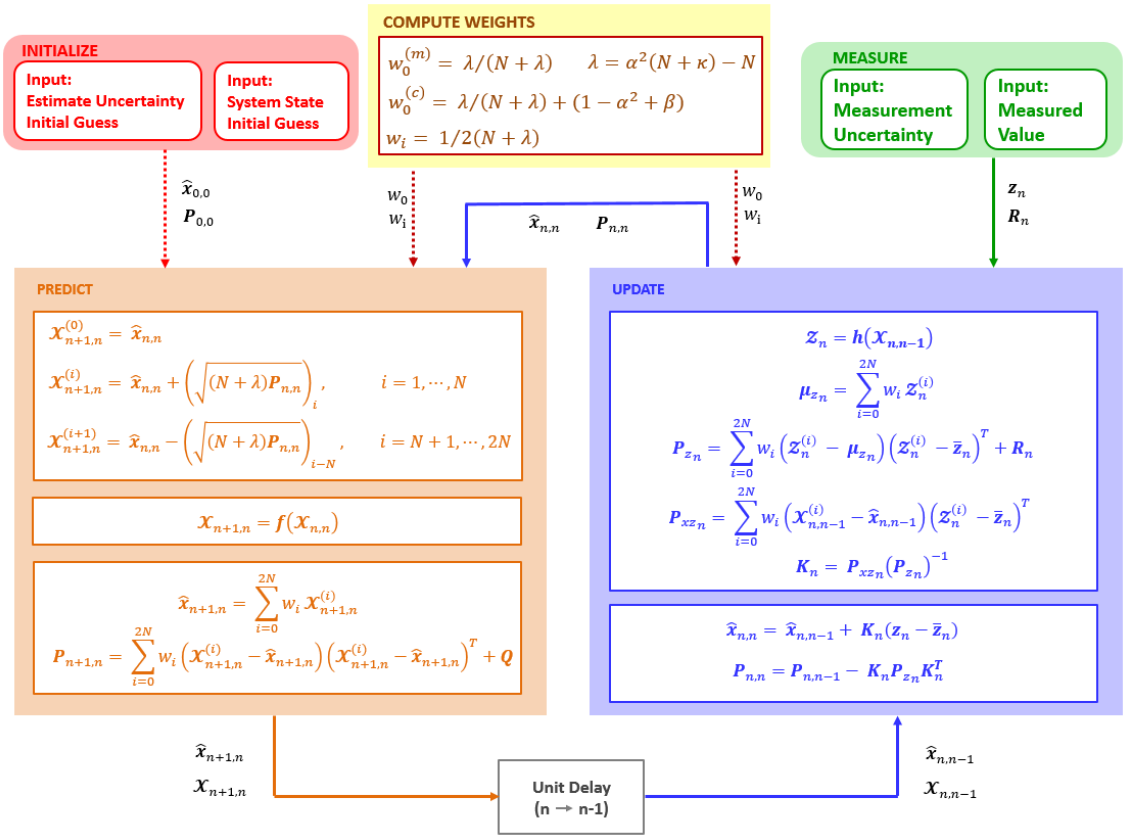


**Figure 14.11:** *Modified UKF algorithm diagram.*

Let us see an example.

## 14.10 Example 14 - estimating the pendulum angle

This example is identical to example 12 (section 13.8) with one difference. We use UKF instead of EKF. In this example, we estimate the angle $\theta$ of an ideal gravity pendulum.

We measure the pendulum position $z = Lsin(\theta_n)$.

The state vector of the pendulum is in the form of the following:

$$\boldsymbol{x}_n = \begin{bmatrix} \theta_n \\ \dot{\theta}_n \end{bmatrix} \tag{14.25}$$

Where:

- $\theta_n$ is the pendulum angle at time $n$
- $\dot{\theta}_n$ is the pendulum angular velocity at time $n$

The dynamic model of the pendulum is non-linear (the second type of non-linearity). It has the form of:

$$\hat{\boldsymbol{x}}_{n+1,n} = \boldsymbol{f}(\hat{\boldsymbol{x}}_{n,n}) \tag{14.26}$$

$$\hat{\boldsymbol{x}}_{n+1,n} = \begin{bmatrix} \hat{\theta}_{n+1,n} \\ \hat{\dot{\theta}}_{n+1,n} \end{bmatrix} = \begin{bmatrix} \hat{\theta}_{n,n} + \hat{\dot{\theta}}_{n,n}\Delta t \\ \hat{\dot{\theta}}_{n,n} - \frac{g}{L} sin(\hat{\theta}_{n,n})\Delta t \end{bmatrix} \tag{14.27}$$

$$\boldsymbol{f}(\hat{\boldsymbol{x}}_{n,n}) = \begin{bmatrix} \hat{\theta}_{n,n} + \hat{\dot{\theta}}_{n,n}\Delta t \\ \hat{\dot{\theta}}_{n,n} - \frac{g}{L} sin(\hat{\theta}_{n,n})\Delta t \end{bmatrix} \tag{14.28}$$

The estimate covariance is:

$$\boldsymbol{P} = \begin{bmatrix} p_x & p_{x\dot{x}} \\ p_{\dot{x}x} & p_{\dot{x}} \end{bmatrix} \tag{14.29}$$

The elements of the main diagonal of the matrix are the variances of the estimation:

- $p_x$ is the variance of the angle $\theta$ estimation
- $p_{\dot{x}}$ is the variance of the angular velocity $\dot{\theta}$ estimation

Since the state-to-measurement relation (the first type of non-linearity) is non-linear, the measurement equation is a type of:

$$\boldsymbol{z}_n = \boldsymbol{h}\left(\boldsymbol{x}_n\right) \tag{14.30}$$

$$\boldsymbol{h}\left(\boldsymbol{x}_n\right) = Lsin\left(\theta_n\right) \tag{14.31}$$

The example parameters:

- The Pendulum string length: $L = 0.5m$
- Gravitational acceleration constant: $g = 9.8\frac{m}{s^2}$
- Measurement Uncertainty (standard deviation): $\sigma_{x_m} = 0.01m$
- Process Noise Uncertainty (angular acceleration standard deviation): $\sigma_a = 1\frac{rad}{s^2}$

The process noise matrix $\boldsymbol{Q}$ is:

$$\boldsymbol{Q} = \begin{bmatrix} \sigma_x^2 & \sigma_{x\dot{x}}^2 \\ \sigma_{\dot{x}x}^2 & \sigma_{\dot{x}}^2 \end{bmatrix} = \begin{bmatrix} \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} \\ \frac{\Delta t^3}{2} & \Delta t^2 \end{bmatrix} \sigma_a^2 = \begin{bmatrix} 1.5625 \times 10^{-6} & 6.25 \times 10^{-5} \\ 6.25 \times 10^{-5} & 0.0025 \end{bmatrix} \tag{14.32}$$

The measurement variance $\boldsymbol{R}$ is:

$$\boldsymbol{R}_n = \begin{bmatrix} \sigma_{x_m}^2 \end{bmatrix} r = 0.01^2 \tag{14.33}$$

## 14.10.1 The numerical example

The following table contains the set of the first 10 noisy measurements. The measurements are identical to example 12 (section 13.8):

| | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| $\boldsymbol{Lsin(\theta)}$ | 0.119 | 0.113 | 0.12 | 0.101 | 0.099 | 0.063 | 0.008 | -0.017 | -0.037 | -0.05 |

**Table 14.6:** *Example 14 measurements.*

### 14.10.1.1 UKF parameters computation

- The number of dimensions: $N = 2$
- The number of sigma points: $2N + 1 = 5$

#### Weights Calculation

In this example, we use a modified algorithm for sigma points computation (section 14.9).

$$\lambda = \alpha^2 (N + \kappa) - N$$

Set:

- $\kappa = 0$
- $\alpha = 0.1$
- $\beta = 2$

$\lambda = 0.1^2 (2 + 0) - 2 = -1.98$

$w_0^{(m)} = \lambda / (N - \lambda) = -1.98 / (2 - 1.98) = -99$
$w_0^{(c)} = \lambda / (N - \lambda) + (1 - \alpha^2 + \beta) = -1.98 / (2 - 1.98) + (1 - 0.1^2 + 2) = -96$
$w_i = 1 / (2 (N + \lambda)) = 1 / (2 (2 - 1.98)) = 25, \qquad i > 0$

$$\boldsymbol{w}_{0,0}^{(m)} = \begin{bmatrix} -99 & 25 & 25 & 25 & 25 \end{bmatrix}$$

$$\boldsymbol{W}_{0,0}^{(c)} = \begin{bmatrix} -96 & 0 & 0 & 0 & 0 \\ 0 & 25 & 0 & 0 & 0 \\ 0 & 0 & 25 & 0 & 0 \\ 0 & 0 & 0 & 25 & 0 \\ 0 & 0 & 0 & 0 & 25 \end{bmatrix}$$

### 14.10.1.2 Iteration Zero

#### Initialization

We don't know the pendulum angle, so our initial angle approximation includes an error. We set the initial velocity to 0.

$$\hat{\boldsymbol{x}}_{0,0} = \begin{bmatrix} 0.0873 \\ 0 \end{bmatrix}$$

Since our initial state vector is a guess, we set a high estimate uncertainty. The high estimate uncertainty results in a high Kalman Gain by giving high weight to the measurement.

$$\boldsymbol{P}_{0,0} = \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix}$$

### Prediction

#### Sigma points computation

To find other sigma points, we should compute the square root:

$$\sqrt{(N+\lambda)\,\boldsymbol{P}_{0,0}} = \sqrt{0.02\begin{bmatrix}5 & 0\\ 0 & 5\end{bmatrix}}$$

We can use the Cholesky decomposition. The manual computation of Cholesky decomposition is quite tedious. I provide here Python and MATLAB examples.

**Python example:**

```
import numpy as np

P = 0.02*np.diag(np.tile(5,2))

print(P)

[[0.1  0. ]
 [0.   0.1]]

L = np.linalg.cholesky(P)

print(L)
[[0.31622777 0.        ]
 [0.         0.31622777]]
```

**MATLAB example:**

```
P = 0.02*diag(repmat(5,1,2))
L = chol(P)'

P =

    0.1                           0
      0                         0.1


L =

    0.316227766016838                           0
                      0         0.316227766016838
```

$$\mathcal{X}_{0,0}^{(1)} = \hat{\boldsymbol{x}}_{0,0} + \left(\sqrt{(N+\lambda)\,\boldsymbol{P}_{0,0}}\right)_1 = \begin{bmatrix} 0.0873 \\ 0 \end{bmatrix} + \begin{bmatrix} 0.3162 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.403 \\ 0 \end{bmatrix}$$

$$\mathcal{X}_{0,0}^{(2)} = \hat{\boldsymbol{x}}_{0,0} + \left(\sqrt{(N+\lambda)\,\boldsymbol{P}_{0,0}}\right)_2 = \begin{bmatrix} 0.0873 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0.3162 \end{bmatrix} = \begin{bmatrix} 0.0873 \\ 0.3162 \end{bmatrix}$$

$$\mathcal{X}_{0,0}^{(3)} = \hat{\boldsymbol{x}}_{0,0} - \left(\sqrt{(N+\lambda)\,\boldsymbol{P}_{0,0}}\right)_1 = \begin{bmatrix} 0.0873 \\ 0 \end{bmatrix} - \begin{bmatrix} 0.3162 \\ 0 \end{bmatrix} = \begin{bmatrix} -0.229 \\ 0 \end{bmatrix}$$

$$\mathcal{X}_{0,0}^{(4)} = \hat{\boldsymbol{x}}_{0,0} - \left(\sqrt{(N+\lambda)\,\boldsymbol{P}_{0,0}}\right)_2 = \begin{bmatrix} 0.0873 \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 0.3162 \end{bmatrix} = \begin{bmatrix} 0.0873 \\ -0.3162 \end{bmatrix}$$

Now, stack all sigma points in a matrix:

$$\mathcal{X}_{0,0} = \begin{bmatrix} 0.0873 & 0.403 & 0.0873 & -0.229 & 0.0873 \\ 0 & 0 & 0.3162 & 0 & -0.3162 \end{bmatrix}$$

### Points propagation

Propagate each selected point through the non-linear function, producing a new set of points belonging to the output distribution.

$$\mathcal{X}_{1,0} = \boldsymbol{f}\left(\mathcal{X}_{0,0}\right)$$

$$\mathcal{X}_{0,0} = \begin{bmatrix} 0.0873 & 0.4035 & 0.1031 & -0.229 & 0.0715 \\ -0.0854 & -0.3848 & 0.2308 & 0.2224 & -0.4016 \end{bmatrix}$$

**Mean and covariance computation**

$$\hat{\boldsymbol{x}}_{1,0} = \mathcal{X}_{1,0}\boldsymbol{w} = \begin{bmatrix} 0.0873 \\ 0.1263 \end{bmatrix}$$

$$\boldsymbol{P}_{1,0} = (\mathcal{X}_{1,0} - \hat{\boldsymbol{x}}_{1,0}) \cdot diag(\boldsymbol{w}) \cdot (\mathcal{X}_{1,0} - \hat{\boldsymbol{x}}_{1,0})^T + \boldsymbol{Q} = \begin{bmatrix} 5.01 & -4.55 \\ -4.55 & 9.7 \end{bmatrix}$$

#### 14.10.1.3 First Iteration

##### Update

Using the Unscented Transform, transfer the state sigma points $\mathcal{X}_{1,0}$ to the measurement space $\mathcal{Z}$ using the measurement function $\boldsymbol{h}(\boldsymbol{x})$:

$$\mathcal{Z}_1 = \boldsymbol{h}\left(\mathcal{X}_{1,0}\right)$$

In order to compute $\mathcal{Z}_1$, we perform elementwise operations between the first row of $\mathcal{X}_{1,0}$ that represents the angle $\theta$:

$$\mathcal{Z}_1 = \boldsymbol{h}\left(\mathcal{X}_{1,0}\right) = Lsin\left(\mathcal{X}_{1,0}\left(1,:\right)\right)$$

$$\mathcal{Z}_1 = \begin{bmatrix} 0.0436 & 0.1963 & 0.0514 & -0.1135 & 0.0357 \end{bmatrix}$$

Multiply $\mathcal{Z}_1$ by weights.

Compute covariance at the measurement space:

$$\boldsymbol{P}_{z_1} = (\mathcal{Z}_1 - \overline{\boldsymbol{z}}_1) \cdot diag(\boldsymbol{w}) \cdot (\mathcal{Z}_1 - \overline{\boldsymbol{z}}_1)^T + \boldsymbol{R} = 1.226$$

Compute the cross-covariance of the state and the measurement:

$$\boldsymbol{P}_{xz_1} = (\mathcal{X}_{1,0} - \hat{\boldsymbol{x}}_{1,0}) \cdot diag(\boldsymbol{w}) \cdot (\mathcal{Z}_1 - \overline{\boldsymbol{z}}_1)^T = \begin{bmatrix} 2.455 \\ -2.273 \end{bmatrix}$$

Compute the Kalman gain:

$$\boldsymbol{K}_1 = \boldsymbol{P}_{xz_1}\left(\boldsymbol{P}_{z_1}\right)^{-1} = \begin{bmatrix} 2 \\ -1.85 \end{bmatrix}$$

Update estimate with measurement:

$$\hat{\boldsymbol{x}}_{1,1} = \hat{\boldsymbol{x}}_{1,0} + \boldsymbol{K}_1\left(\boldsymbol{z}_1 - \overline{\boldsymbol{z}}_1\right) = \begin{bmatrix} 0.456 \\ -0.215 \end{bmatrix}$$

Update covariance of the estimate:

$$\boldsymbol{P}_{1,1} = \boldsymbol{P}_{1,0} - \boldsymbol{K}_1 \boldsymbol{P}_{z_1} \boldsymbol{K}_1^T = \begin{bmatrix} 0.097 & 0.0002 \\ 0.0002 & 5.489 \end{bmatrix}$$

### Predict

#### Sigma points computation

$$\mathcal{X}_{1,1}^{(0)} = \hat{\boldsymbol{x}}_{1,1} = \begin{bmatrix} 0.456 \\ -0.215 \end{bmatrix}$$

To find other sigma points, we should compute the square root:

$$\sqrt{(N+\lambda)\,\boldsymbol{P}_{1,1}} = \sqrt{0.02 \begin{bmatrix} 0.097 & 0.0002 \\ 0.0002 & 5.489 \end{bmatrix}}$$

We can use the Cholesky decomposition:

**Python example:**

```
import numpy as np

L = np.linalg.cholesky(0.02*P)

print(L)
[[4.39953012e-02 0.00000000e+00]
 [8.53887720e-05 3.31328834e-01]]
```

**MATLAB example:**

```
L = chol(0.02*P)'


L =

0.0439953011769878                   0
8.53887720392724e-05          0.331328833836111
```

$$\mathcal{X}_{1,1}^{(1)} = \hat{\boldsymbol{x}}_{1,1} + \left(\sqrt{(N+\lambda)\,\boldsymbol{P}_{1,1}}\right)_1 = \begin{bmatrix} 0.456 \\ -0.215 \end{bmatrix} + \begin{bmatrix} 0.043 \\ 8.5 \times 10^{-5} \end{bmatrix} = \begin{bmatrix} 0.5 \\ -0.215 \end{bmatrix}$$

$$\mathcal{X}_{1,1}^{(2)} = \hat{\boldsymbol{x}}_{1,1} + \left(\sqrt{(N+\lambda)\,\boldsymbol{P}_{1,1}}\right)_2 = \begin{bmatrix} 0.456 \\ -0.215 \end{bmatrix} + \begin{bmatrix} 0 \\ 0.3313 \end{bmatrix} = \begin{bmatrix} 0.456 \\ 0.1164 \end{bmatrix}$$

$$\mathcal{X}_{1,1}^{(3)} = \hat{\boldsymbol{x}}_{1,1} - \left(\sqrt{(N+\lambda)\,\boldsymbol{P}_{1,1}}\right)_1 = \begin{bmatrix} 0.456 \\ -0.215 \end{bmatrix} - \begin{bmatrix} 0.043 \\ 8.5 \times 10^{-5} \end{bmatrix} = \begin{bmatrix} 0.412 \\ -0.215 \end{bmatrix}$$

$$\mathcal{X}_{1,1}^{(4)} = \hat{\boldsymbol{x}}_{1,1} - \left(\sqrt{(N+\lambda)\,\boldsymbol{P}_{1,1}}\right)_2 = \begin{bmatrix} 0.456 \\ -0.215 \end{bmatrix} - \begin{bmatrix} 0 \\ 0.3313 \end{bmatrix} = \begin{bmatrix} 0.456 \\ -0.546 \end{bmatrix}$$

Stack all sigma points in a matrix:

$$\mathcal{X}_{1,1} = \begin{bmatrix} 0.456 & 0.5 & 0.456 & 0.412 & 0.456 \\ -0.215 & -0.215 & 0.1164 & -0.215 & -0.546 \end{bmatrix}$$

**Points propagation**

Propagate each selected point through the non-linear function, producing a new set of points belonging to the output distribution.

$$\mathcal{X}_{2,1} = \boldsymbol{f}\left(\mathcal{X}_{1,1}\right)$$

$$\mathcal{X}_{2,1} = \begin{bmatrix} 0.445 & 0.489 & 0.462 & 0.4 & 0.429 \\ -0.646 & -0.685 & -0.315 & -0.6 & -0.978 \end{bmatrix}$$

**Mean and covariance computation**

$$\hat{\boldsymbol{x}}_{2,1} = \mathcal{X}_{2,1}\boldsymbol{w} = \begin{bmatrix} 0.445 \\ -0.626 \end{bmatrix}$$

$$\boldsymbol{P}_{2,1} = \left(\mathcal{X}_{2,1} - \hat{\boldsymbol{x}}_{2,1}\right) \cdot diag(\boldsymbol{w}) \cdot \left(\mathcal{X}_{2,1} - \hat{\boldsymbol{x}}_{2,1}\right)^T + \boldsymbol{Q} = \begin{bmatrix} 0.11 & 0.19 \\ 0.19 & 5.57 \end{bmatrix}$$

#### 14.10.1.4 Next Iterations

The results of the second iteration:

$$\hat{\boldsymbol{x}}_{2,2} = \begin{bmatrix} 0.248 \\ -0.962 \end{bmatrix}$$

$$\boldsymbol{P}_{2,2} = \begin{bmatrix} 0.0018 & 0.0043 \\ 0.0043 & 5.25 \end{bmatrix}$$

The results of the last (tenth) iteration:

$$\hat{\boldsymbol{x}}_{10,10} = \begin{bmatrix} -0.136 \\ -1.21 \end{bmatrix}$$

$$\boldsymbol{P}_{10,10} = \begin{bmatrix} 0.00017 & 0.00074 \\ 0.00074 & 0.01 \end{bmatrix}$$

### 14.10.2 Example summary

The following chart compares the true, measured, and estimated pendulum angles for 50 measurements. (The measured angle is derived from the measured distance $\theta_n = sin^{-1}\left(\frac{z}{L}\right)$). The chart also depicts the 95% confidence interval.

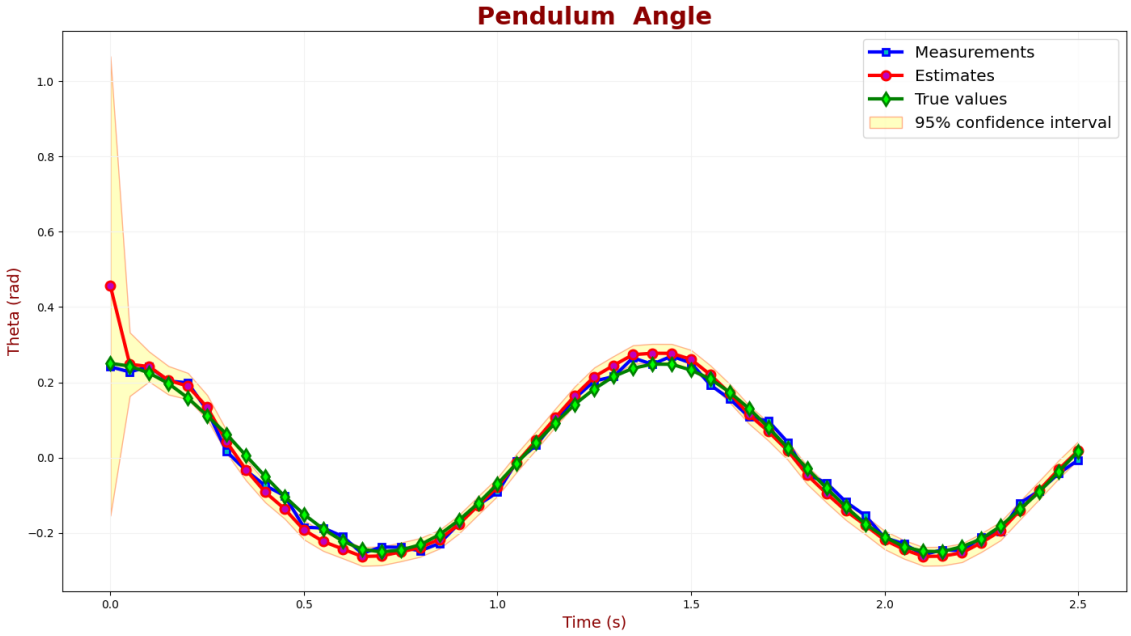


**Figure 14.12:** *Example 14: pendulum angle - true value, measured values and estimates.*

The next chart compares the true and estimated pendulum angular velocity.

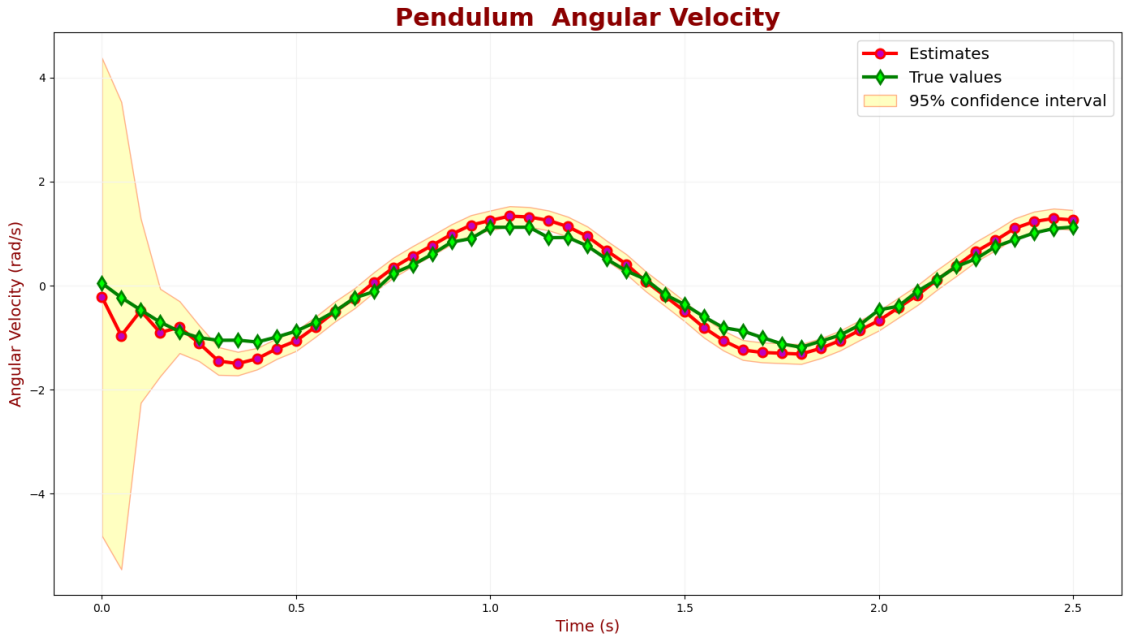


**Figure 14.13:** *Example 14: pendulum velocity - true value, measured values and estimates.*

# 15. Non-linear filters comparison

Let us compare the EKF and UKF performance for our examples.

Examples 11 and 13 deal with vehicle location estimation using radar. In example 11, we used EKF, and in example 13, we used UKF. Both examples have identical parameters – the same vehicle dynamics, the same radar, the same initialization, and the same measurements.

The following charts compare the EKF and UKF absolute error of the vehicle position on the X and Y axes.

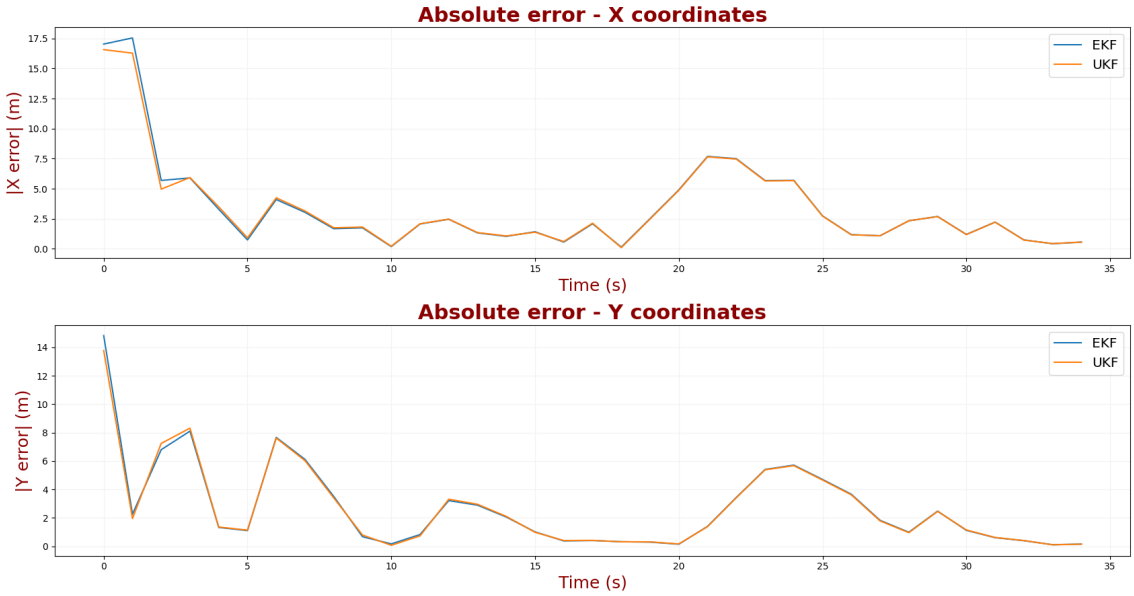


**Figure 15.1:** *EKF and UKF absolute error of the vehicle position.*

We can see that the UKF performance is slightly better for the first two iterations of the filter, but then the filters converge, and the performance of both filters is identical.

Let us take a look at the estimations uncertainty ($\sigma$) of both filters.

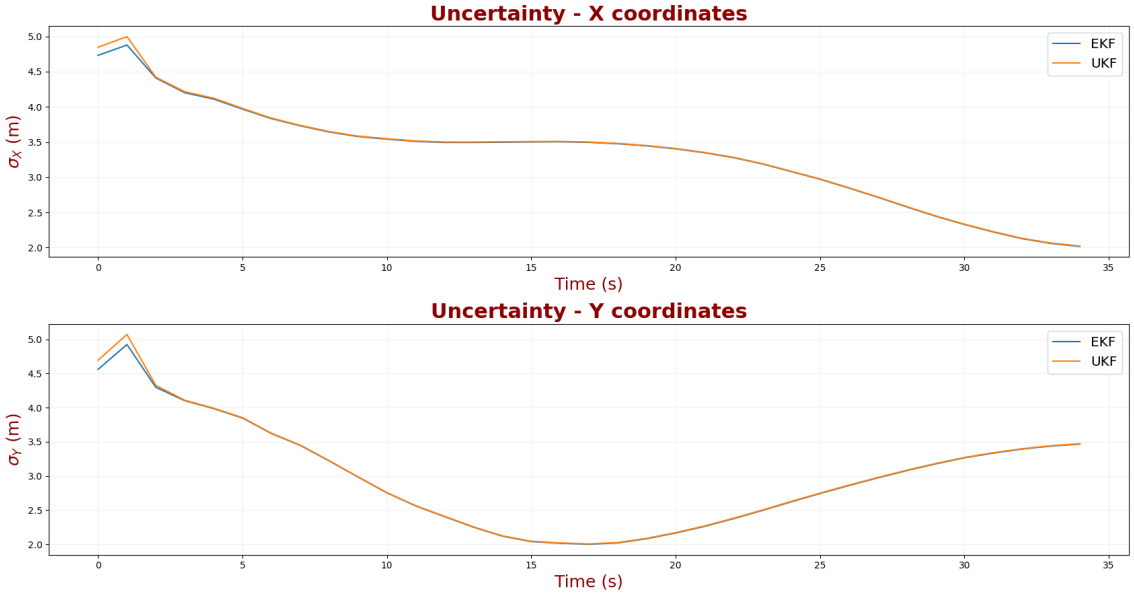


**Figure 15.2:** *EKF and UKF estimations uncertainty of the vehicle position.*

The estimation uncertainties are also identical for both filters.

Examples 12 and 14 deal with the estimation of the pendulum angle and angular velocity. In example 12, we used EKF, and in example 14, we used UKF. Both examples have identical parameters – the same pendulum dynamics, the same initialization, and the same measurements.

The following charts compare the EKF and UKF absolute error of the pendulum angle and angular velocity.

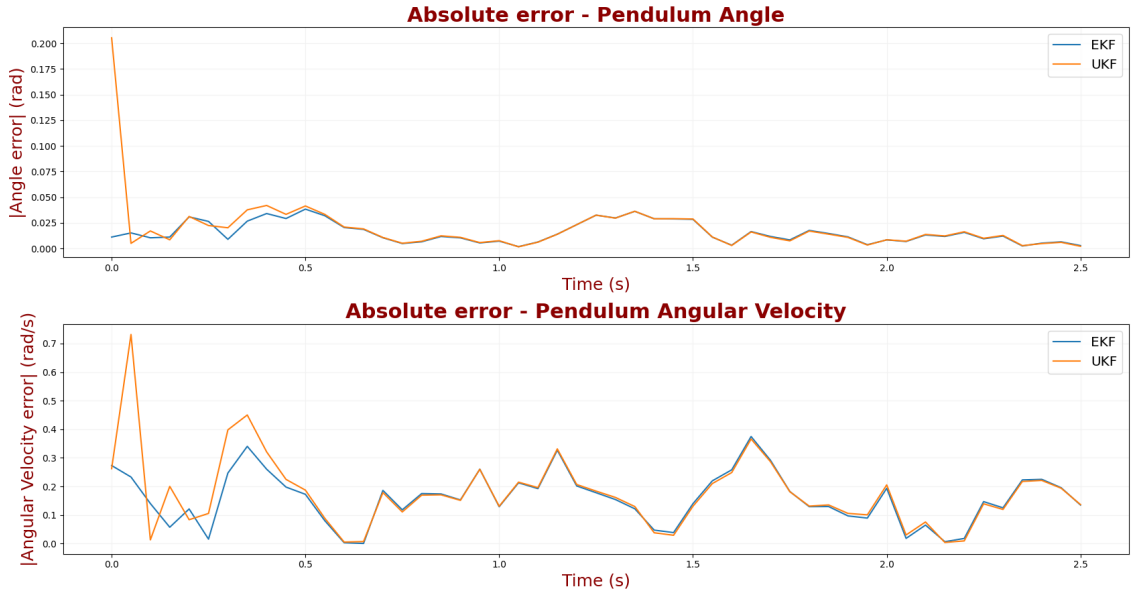


**Figure 15.3:** *EKF and UKF absolute error of the pendulum angle and angular velocity.*

We can see that during the first 0.5 seconds, the UKF error is significantly higher. Then the filters converge, and the performance of both filters is identical.

Let us take a look at the estimations uncertainty ($\sigma$) of both filters.

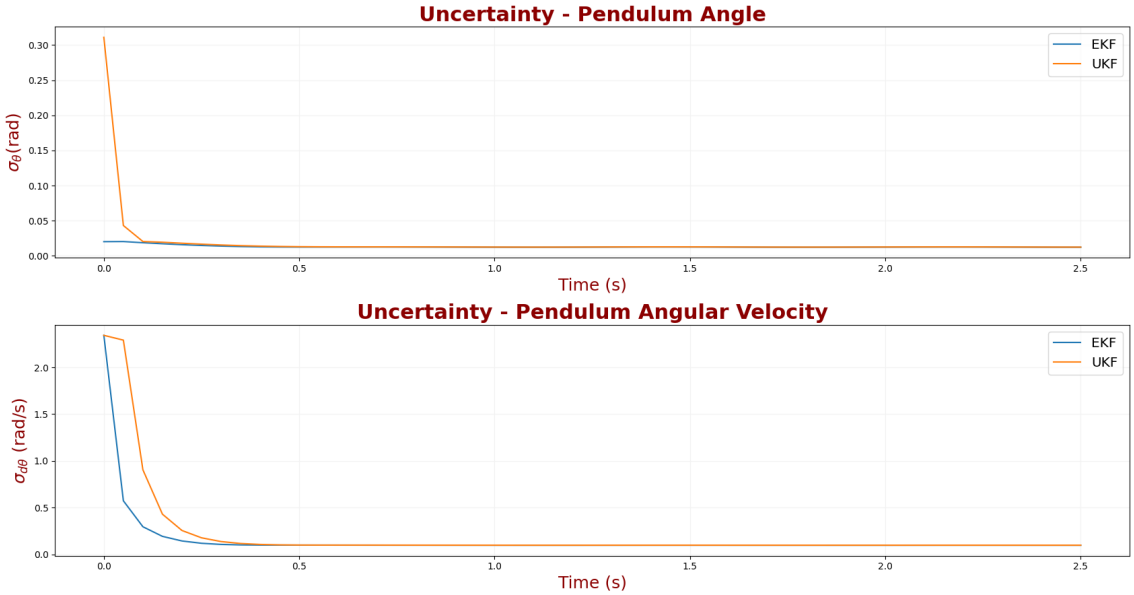


**Figure 15.4:** *EKF and UKF estimations uncertainty of the pendulum angle and angular velocity.*

During the filter convergence period, the UKF uncertainty is higher than the EKF uncertainty.

In this case, the EKF outperforms UKF.

# 16. Conclusion

Extended and Unscented Kalman Filters perform a linear approximation of the dynamic and state-to-measurement models.

The Extended Kalman Filter is a standard and most common technique used in non-linear estimation problems. Although the EKF became a standard for non-linear estimation, almost 60 years of EKF usage experience has led to a consensus that it is unreliable for highly non-linear models. The UKF is considered to be a better choice. [8] and [9] paper demonstrate examples of superior UKF performance compared to EKF.

On the other hand, in the vehicle location examples, we have seen a similar performance of UKF and EKF. In the pendulum measurement example, the UKF was even better during the filter convergence period.

Since both filters are sub-optimal, I recommend testing both filters for a specific problem. Then you can choose a better approach.

I would also recommend considering the **Particle Filter**. The particle filter uses using **Monte-Carlo method** technique for approximation. It is considered a precise technique. However, it often requires thousands of times more computations. The particle filter is out of the scope of this book. You can find a good tutorial in [13].

# IV Kalman Filter in practice

# 17. Sensors Fusion

Many practical systems are equipped with several complementary and sometimes interchangeable sensors that measure the same parameters.

A self-driving car has Light Detection and Ranging (LiDAR) and radar onboard. The LiDAR is much more precise than the radar. On the other hand, the radar measures velocity using the Doppler Effect, and its effective operational range is higher, especially in rain or fog conditions. The aircraft is equipped with Global Navigation Satellite System (GNSS) and Inertial Navigation System (INS) systems for navigation. Many surveillance systems include several radars for target tracking.

Using multiple sensors can significantly improve the state estimation precision in a process known as sensor fusion.

Sensor fusion refers to combining the measurements from multiple sensors resulting in joint information having less uncertainty than any of the sensors individually.

Let us examine the influence of the measurements fusion on the uncertainty.

## 17.1 Combining measurements in one dimension

Consider two range measurements of the same target performed by two independent radars simultaneously. The SNR of the first radar measurement is higher than the SNR of the second radar measurement. Thus the uncertainty of the first radar measurement is lower.

The following figure represents the measurements of both radars as normally distributed random variables (Gaussians).

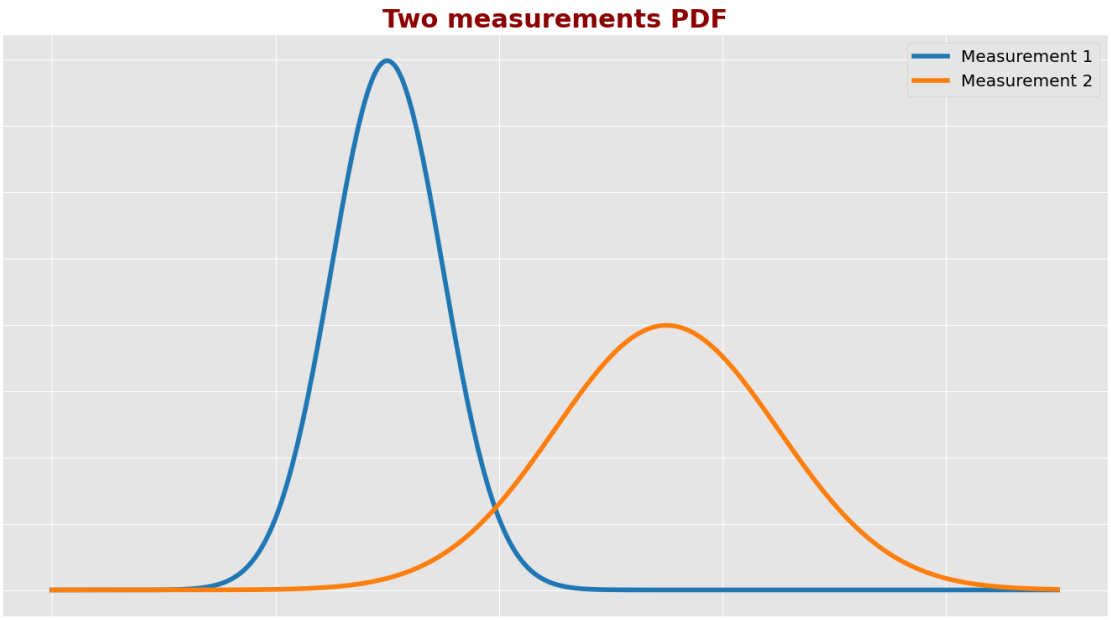


**Figure 17.1:** *Two measurements PDF.*

The PDFs are described by the following:

$$
\begin{aligned}
p(x)_1 &= \frac{1}{\sigma_1\sqrt{2\pi}}\, exp\left(-\frac{1}{2}\left(\frac{x-\mu_1}{\sigma_1}\right)^2\right) \\
p(x)_2 &= \frac{1}{\sigma_2\sqrt{2\pi}}\, exp\left(-\frac{1}{2}\left(\frac{x-\mu_2}{\sigma_2}\right)^2\right)
\end{aligned}
\tag{17.1}
$$

Where:

- $\mu_1$ is the measured value of the first radar
- $\sigma_1$ is the measurement standard deviation of the first radar
- $\mu_2$ is the measured value of the second radar
- $\sigma_2$ is the measurement standard deviation of the second radar

Since the measurements are independent, the joint PDF is a product of two PDFs. Let us see what happens when we multiply two Gaussians.

$$
p(x) = p(x)_1 p(x)_2 = \frac{1}{\sigma_1\sqrt{2\pi}}\, exp\left(-\frac{1}{2}\left(\frac{x-\mu_1}{\sigma_1}\right)^2\right) \frac{1}{\sigma_2\sqrt{2\pi}}\, exp\left(-\frac{1}{2}\left(\frac{x-\mu_2}{\sigma_2}\right)^2\right) \tag{17.2}
$$

The product of two Gaussian PDFs is proportional to Gaussian PDF with the

following properties:

$$\sigma_{12}^2 = \frac{1}{\frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2}} \tag{17.3}$$

$$\mu_{12} = \left( \frac{\mu_1}{\sigma_1^2} + \frac{\mu_2}{\sigma_2^2} \right) \sigma_{12}^2 \tag{17.4}$$

You can see the derivation in Appendix G.

Let us take a closer look at the Gaussian PDFs product properties.

Each measurement is weighted by the inverse of its variance. If the variance of the first measurement is lower than the variance of the second measurement ($\sigma_1^2 < \sigma_2^2$), then the weight of the first measurement is higher.

That means the Gaussian PDFs product is closer to the measurement with lower variance.

The variance of the Gaussian PDFs product is always lower than the variance of each measurement separately. Even a measurement with a very high variance contributes to the overall precision. If $\sigma_2^2 = \infty$ then $\sigma_{12}^2 = \sigma_1^2$.

The following plot exemplifies the fusion of two measurements. We can see that the joint PDF is closer to the measurement with a lower variance. The variance of the joint PDF is lower than the variance of each measurement.

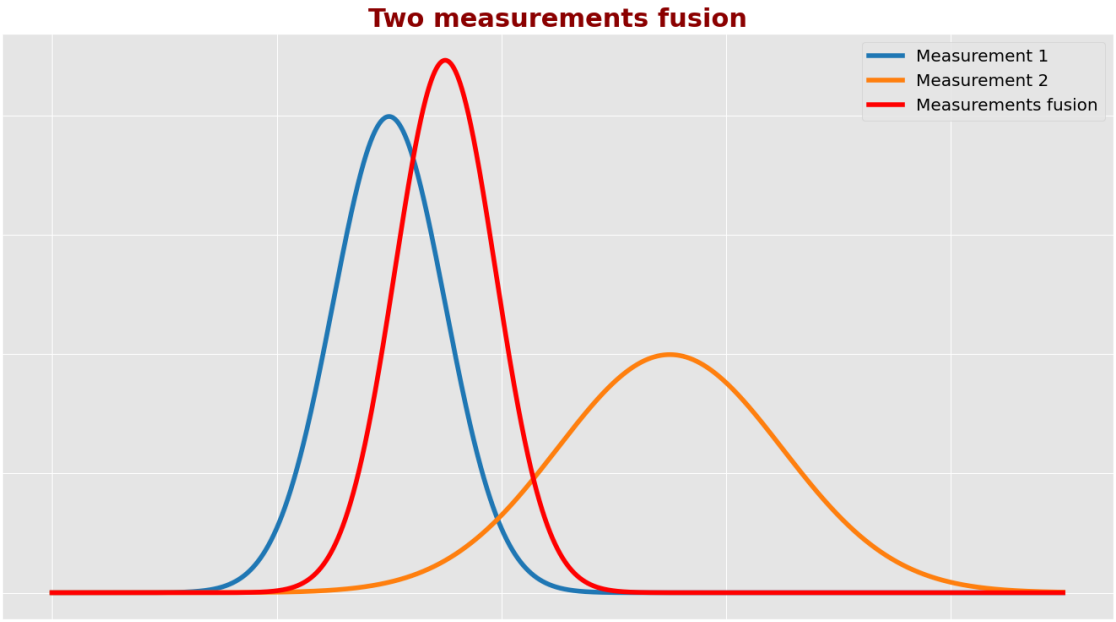


**Figure 17.2:** *Two measurements fusion.*

## 17.2 Combining $n$ measurements

The product univariate Gaussian PDFs is also a k-dimensional Gaussian PDF with the following properties:

$$\sigma_{1\cdots n}^2 = \frac{1}{\frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2} + \cdots + \frac{1}{\sigma_n^2}} = \frac{1}{\sum_{i=1}^{n} \sigma_i^2} \tag{17.5}$$

$$\mu_{1\cdots n} = \left( \sum_{i=1}^{n} \frac{\mu_i}{\sigma_i^2} \right) \sigma_{1\cdots n}^2 \tag{17.6}$$

You can see the derivation of the generalization to $n$ measurements in Appendix G.

Equation 17.5 shows that every additional measurement minimizes the overall uncertainty.

## 17.3 Combining measurements in $k$ dimensions

The multivariate $k$ - dimensional normal distribution is given by:

$$p(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^k |\boldsymbol{\Sigma}|}} exp\left( -\frac{1}{2} (\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu}) \right) \tag{17.7}$$

Where:

$\boldsymbol{\mu}$ is the measurement vector

$\boldsymbol{\Sigma}$ is the measurement (covariance matrix)

The product of $n$ $k$ - dimensional Gaussian PDFs is also a $k$ - dimensional Gaussian PDF with the following properties:

$$\boldsymbol{\Sigma}_{1\cdots n}^{-1} = \sum_{i=1}^{n} \boldsymbol{\Sigma}_i^{-1} \tag{17.8}$$

$$\boldsymbol{\mu}_{1\cdots n} = \boldsymbol{\Sigma} \sum_{i=1}^{n} \left(\boldsymbol{\Sigma}_i^{-1} \boldsymbol{\mu}_i\right) \tag{17.9}$$

You can see the derivation in Appendix H.

Like in the univariate Gaussian case, the mean of the PDFs product is the weighted mean of all PDFs, while the PDF with a lower variance has a higher weight. Each PDF contributes to the joint PDF and reduces the joint PDF covariance.

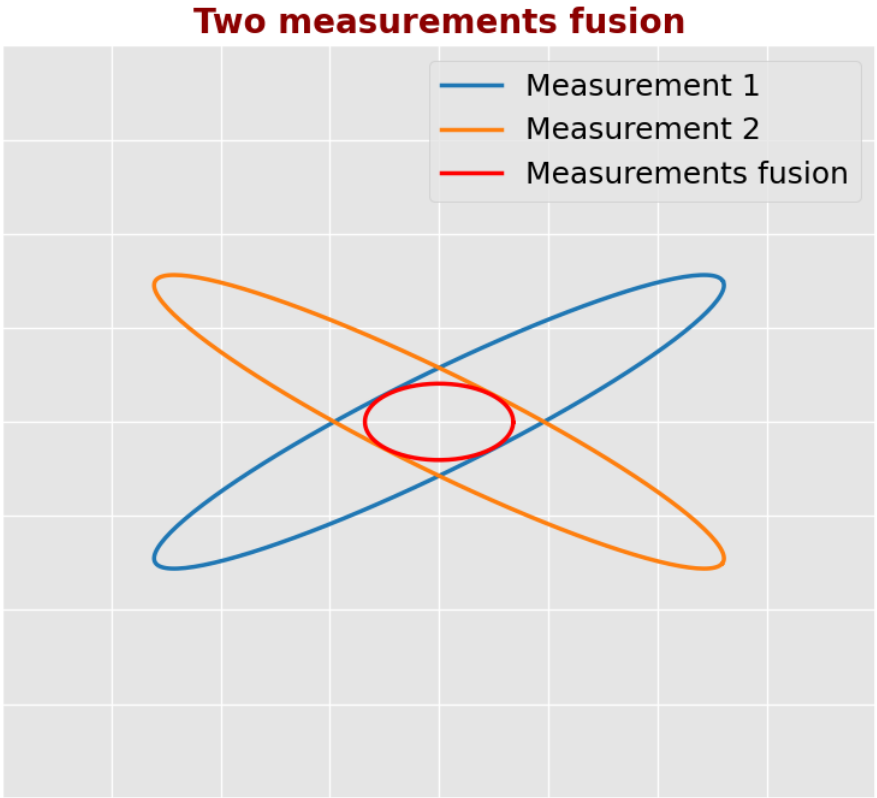


**Figure 17.3:** *Two 2D measurements fusion.*

Figure 17.3 depicts two two-dimensional measurements. The blue and orange ellipses

represent the covariance of two measurements, and the red ellipse represents the covariance of the joint PDF.

We can see that the multisensor measurement fusion results in a more precise estimation.

## 17.4 Sensor data fusion using Kalman filter

Kalman filter is the most widespread algorithm for multisensor data fusion. This chapter describes two different methods for multisensor data fusion using the Kalman filter. For simplicity, we assume that the sensors' sample rates are identical. We discuss the multi-rate Kalman Filter in section 17.5.

### 17.4.1 Method 1 – measurements fusion

Measurement fusion is the most common sensor data fusion method. Typically it is used by a system that is equipped with several sensors. The fusion is applied to the sensors' measurements.

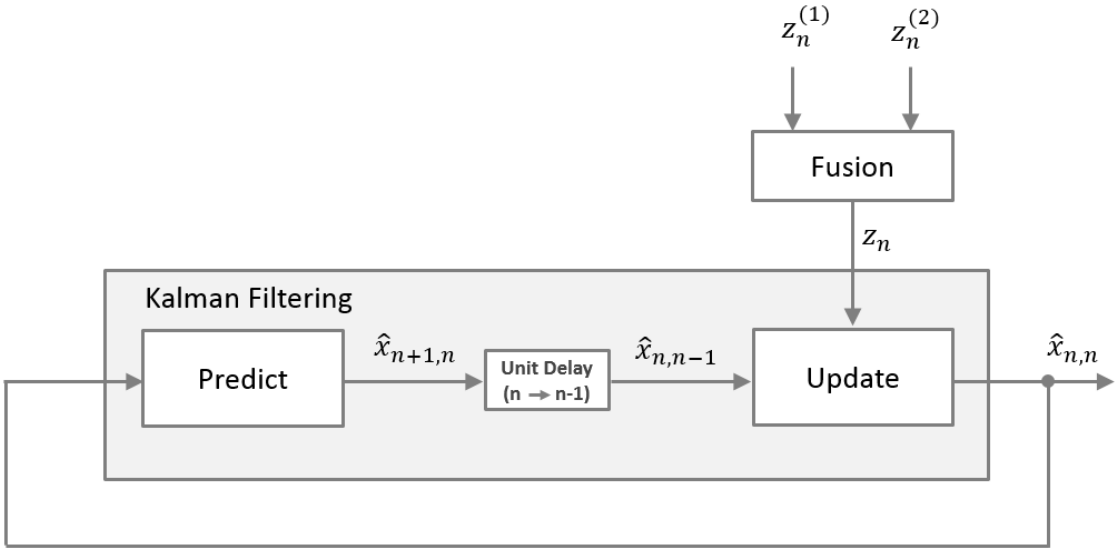


**Figure 17.4:** *2 Sensors measurements fusion.*

Two or more sensors provide measurements $\boldsymbol{z}_n^{(1)}$, $\boldsymbol{z}_n^{(2)}, \ldots, \boldsymbol{z}_n^{(k)}$ with measurement covariance $\boldsymbol{R}_n^{(1)}$, $\boldsymbol{R}_n^{(2)}, \ldots, \boldsymbol{R}_n^{(k)}$. Assuming normally distributed measurements PDFs, we can calculate a joint PDF:

$$\boldsymbol{R}_n^{-1} = \sum_{i=1}^{k} \boldsymbol{R}_i^{-1} \tag{17.10}$$

$$\boldsymbol{z}_n = \boldsymbol{R_n} \sum_{i=1}^{k} \left( \boldsymbol{R}_n^{(i)^{-1}} \boldsymbol{z}_n^{(i)} \right) \tag{17.11}$$

The conventional Kalman Filter receives the unified measurement $(\boldsymbol{z}_n, \boldsymbol{R}_n)$.

In the literature, you can also see that $\boldsymbol{z}_n$ for two sensors is calculated as follows:

$$\boldsymbol{z}_n = \boldsymbol{z}_n^{(1)} + \boldsymbol{R}_n^{(1)} \left( \boldsymbol{R}_n^{(1)} + \boldsymbol{R}_n^{(2)} \right)^{-1} \left( \boldsymbol{z}_n^{(2)} - \boldsymbol{z}_n^{(2)} \right) \tag{17.12}$$

First, it looks familiar. It is identical to the Kalman Filter state update equation, which also calculates the fusion of two normally distributed PDFs – the measurement and the prior estimate uncertainty.

Second, it is computationally effective. We perform only one matrix inversion.

You can find the derivation in Appendix H.

### 17.4.2 Method 2 – state fusion

Some applications involve different sensor systems that perform the same task. For example, several geographically distributed radars can track the same target. Each radar creates a separate target track using the Kalman filter. An external system can receive radars' tracks and compute a unified track with higher location precision. The state fusion method is also called the **track-to-track** fusion algorithm.

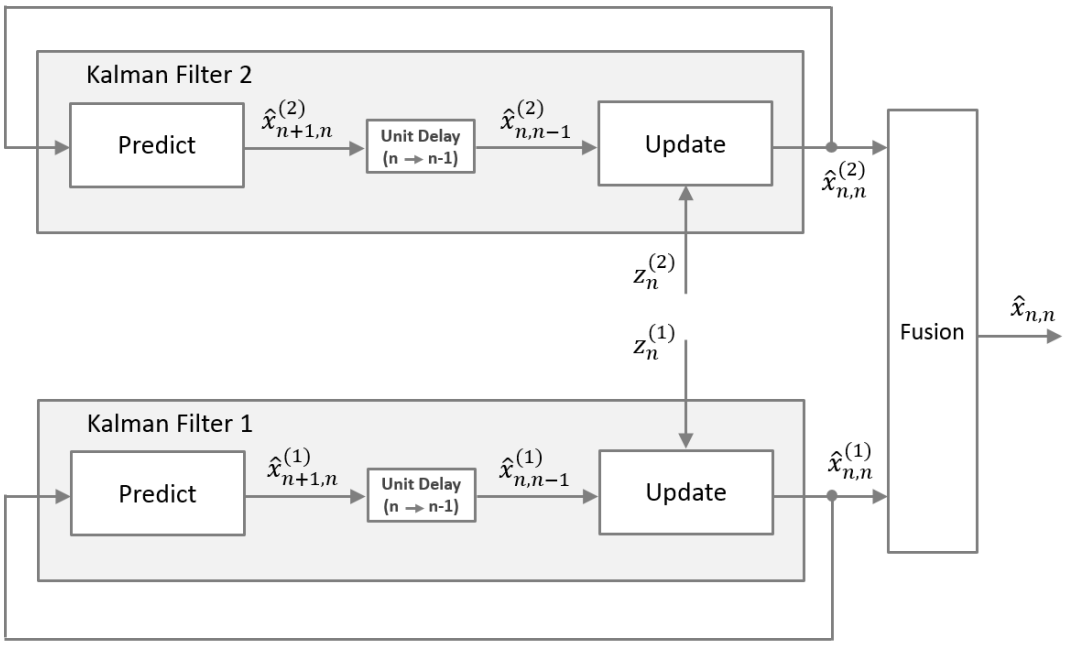


**Figure 17.5:** *Track-to-track fusion.*

Figure 17.5 describes the state fusion method.

When the cross-covariance between the two track estimates can be ignored, the state fusion involves a simple fusion of two random variables $\hat{\boldsymbol{x}}_{n,n}^{(1)}$ and $\hat{\boldsymbol{x}}_{n,n}^{(2)}$ with estimate uncertainties $\boldsymbol{P}_{n,n}^{(1)}$ and $\boldsymbol{P}_{n,n}^{(2)}$:

$$\hat{\boldsymbol{x}}_{n,n} = \hat{\boldsymbol{x}}_{n,n}^{(1)} + \boldsymbol{P}_{n,n}^{(1)} \left(\boldsymbol{P}_{n,n}^{(1)} + \boldsymbol{P}_{n,n}^{(2)}\right)^{-1} \left(\hat{\boldsymbol{x}}_{n,n}^{(2)} - \hat{\boldsymbol{x}}_{n,n}^{(2)}\right) \tag{17.13}$$

The estimate covariance of the joint state $\hat{\boldsymbol{x}}_{n,n}$ is:

$$\boldsymbol{P}_{n,n}^{-1} = \boldsymbol{P}_{n,n}^{(1)^{-1}} + \boldsymbol{P}_{n,n}^{(2)^{-1}} \tag{17.14}$$

The measurements from the two sensor tracks are not necessarily independent, and the two tracks can be correlated due to the common process noise (the target dynamics is not deterministic). The correlated estimation errors have to be considered in combining the state estimates. Otherwise, the target state estimates in the system tracks may degrade.

The method for correlated state fusion was shown by Bar-Shalom [14]. The fused state is given by:

$$\hat{\boldsymbol{x}}_{n,n} = \hat{\boldsymbol{x}}_{n,n}^{(1)} + \left(\boldsymbol{P}_{n,n}^{(1)} - \boldsymbol{P}_{n,n}^{(12)}\right) \left(\boldsymbol{P}_{n,n}^{(1)} + \boldsymbol{P}_{n,n}^{(2)} - \boldsymbol{P}_{n,n}^{(12)} - \boldsymbol{P}_{n,n}^{(21)}\right)^{-1} \left(\hat{\boldsymbol{x}}_{n,n}^{(2)} - \hat{\boldsymbol{x}}_{n,n}^{(2)}\right) \tag{17.15}$$

$\boldsymbol{P}_{n,n}^{(12)} = \left(\boldsymbol{P}_{n,n}^{(21)}\right)^T$ is the cross-covariance matrix between $\hat{\boldsymbol{x}}_{n,n}^{(1)}$ and $\hat{\boldsymbol{x}}_{n,n}^{(2)}$. The cross-covariance matrix is given by:

$$\begin{aligned} \boldsymbol{P}_{n,n}^{(12)} &= \left(\boldsymbol{I} - \boldsymbol{K}_n^{(1)} \boldsymbol{H}^{(1)}\right) \boldsymbol{F} \boldsymbol{P}_{n,n-1}^{(12)} \boldsymbol{F}^T \left(\boldsymbol{I} - \boldsymbol{K}_n^{(2)} \boldsymbol{H}^{(2)}\right)^T \\ &+ \left(\boldsymbol{I} - \boldsymbol{K}_n^{(1)} \boldsymbol{H}^{(1)}\right) \boldsymbol{G} \boldsymbol{Q} \boldsymbol{G}^T \left(\boldsymbol{I} - \boldsymbol{K}_n^{(2)} \boldsymbol{H}^{(2)}\right)^T \end{aligned} \tag{17.16}$$

Where:
$\boldsymbol{K}$ is a Kalman Gain
$\boldsymbol{H}$ is an Observation Matrix
$\boldsymbol{F}$ is a State Transition Matrix
$\boldsymbol{Q}$ is a Process Noise Covariance Matrix
$\boldsymbol{G}$ is a Control Matrix

Bar-Shalom and Campo showed [15] that when this dependence is taken into account, the area of the uncertainty ellipse is reduced by 70% percent instead of being cut by 50% as would be the case if the independent noise assumption were correct.

## 17.5 Multirate Kalman Filter

In many applications, the sensors' sample rates are not identical. For example, assume two sensors with a sampling rate of $\Delta t$ and $3\Delta t$. The Kalman Filter update scheme is described in the following figure.

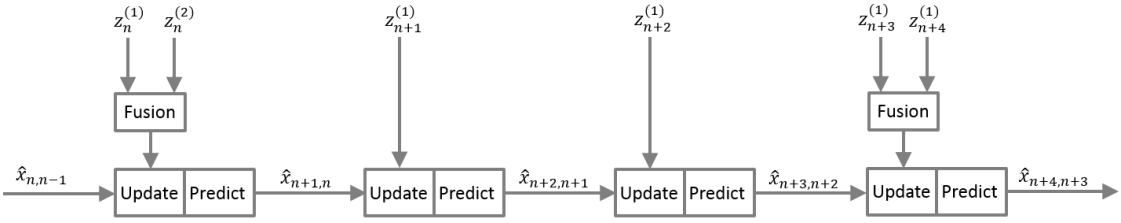


**Figure 17.6:** *Multirate Kalman Filter.*

The first sensor measurement updates the Kalman Filter state every $\Delta t$ period. Every $3\Delta t$ period, the measurements of the first and the second sensors are combined.

# 18. Variable measurement error

Until now, in all our examples, the measurement uncertainty was constant. For many measurement devices, the measurement uncertainty is a parameter given by the vendor. For example, the weight measurement accuracy of the scales or the power measurement accuracy of the power meter is a constant value.

However, in many systems, the measurement uncertainty varies between successive measurements. For example, in a radar system, the range accuracy (standard deviation) is given by:

$$\sigma_R = \frac{c}{2B\sqrt{SNR}} \tag{18.1}$$

Where:

$\sigma_R$ is a range measurement error (standard deviation)
$c$ is the speed of light
$B$ is a signal bandwidth
$SNR$ is a signal-to-noise ratio

The radar range measurement variance is:

$$r_n = \sigma_{R_n}^2 = \frac{c^2}{4B^2 SNR_n} \tag{18.2}$$

We can see that the range measurement uncertainty depends on $SNR$. The $SNR$ depends on many factors, such as interferences, target range, and perspective. For a radar system, the measurement uncertainty is not constant.

# 19. Treating missing measurements

Sometimes the sensor measurements are lost due to noisy communication channels, interferences, equipment malfunctions, or other anomalies. In such cases, you can set the measurement to an arbitrary value while setting the measurement variance to infinity.

The Kalman Gain would be zero:

$$K_n = \frac{p_{n,n-1}}{p_{n,n-1} + r_n} = \frac{p_{n,n-1}}{p_{n,n-1} + \infty} = 0 \tag{19.1}$$

The current state estimation $\hat{\boldsymbol{x}}_{n,n}$ would be equal to the prior estimate $\hat{\boldsymbol{x}}_{n,n-1}$:

$$\hat{\boldsymbol{x}}_{n,n} = \hat{\boldsymbol{x}}_{n,n-1} + 0\,(\boldsymbol{z}_n - \hat{\boldsymbol{x}}_{n,n-1}) = \hat{\boldsymbol{x}}_{n,n-1} \tag{19.2}$$

The current state estimation variance $p_{n,n}$ would be equal to the prior extrapolated state variance $p_{n,n-1}$:

$$p_{n,n} = (1 - 0)\, p_{n,n-1} = p_{n,n-1} \tag{19.3}$$

In case of missing measurement, the extrapolated state variance $p_{n+1,n}$ would increase due to the process noise:

$$p_{n+1,n} = p_{n,n} + q_n = p_{n,n-1} + q_n \tag{19.4}$$

Therefore, the missing measurement causes a temporary Kalman Filter divergence.

# 20. Treating outliers

Outliers are measurements outside an expected range or don't follow an expected pattern. Outliers can result from noisy communication channels, interferences, equipment malfunctions, recording errors, or other anomalies, such as radar signal reflection from a bypassing aircraft.

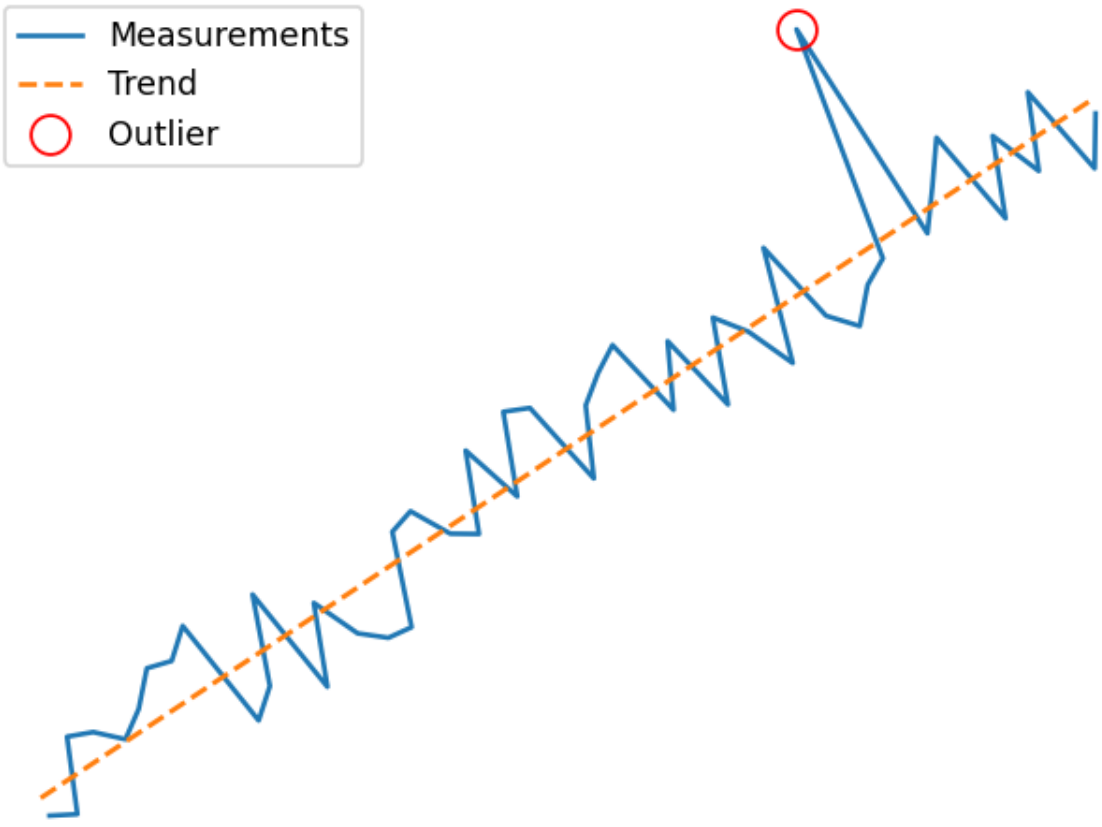


**Figure 20.1:** *Outlier example.*

We will learn how to identify and treat outliers.

## 20.1 Identifying outliers

We can divide outliers into two main categories:

- Unlikely or unusual measurements
- High statistical distance between the prior estimate and the measurement

Let us analyze each category separately.

### 20.1.1 Unlikely or unusual measurements

Sometimes you may decide that the measurement is unlikely based on your technical knowledge of the domain. For example:

- If the vehicle speed measurement is 400km/h, you can decide that the measurement is an outlier since the car can't travel at that speed.
- If the vehicle position measurement is far from the road, it is likely an outlier.
- The water temperature above $100^0C$ is an outlier.

### 20.1.2 High statistical distance

Assume a system that estimates the vehicle speed using the Kalman Filter. The prior estimate of the speed (prediction) $\hat{x}_{n,n-1}$ is 100km/h, and the measured speed $z_n$ is 120km/h. The difference between the predicted and measured values is 20km/h. Is it an outlier?

In order to answer this question, we should find the **Mahalanobis distance**.

The Mahalanobis distance is a measure of the distance between a point P and a distribution D or between two random variables, introduced by P. C. Mahalanobis in 1936.

In a single dimension, the Mahalanobis distance is given by:

$$d_M = \sqrt{\frac{\left(\hat{x}_{n,n-1} - z_n\right)^2}{p_{n,n-1}}} \tag{20.1}$$

Where:

$\hat{x}_{n,n-1}$ is the prior estimate
$p_{n,n-1}$ is the prior estimate variance
$z_n$ is the measurement

Let's return to the vehicle speed estimation example. If $p_{n,n-1} = 150km^2/h^2$, the Mahalanobis distance is:

$$d_M = \sqrt{\frac{\left(100 - 120\right)^2}{150}} = 1.63 \tag{20.2}$$

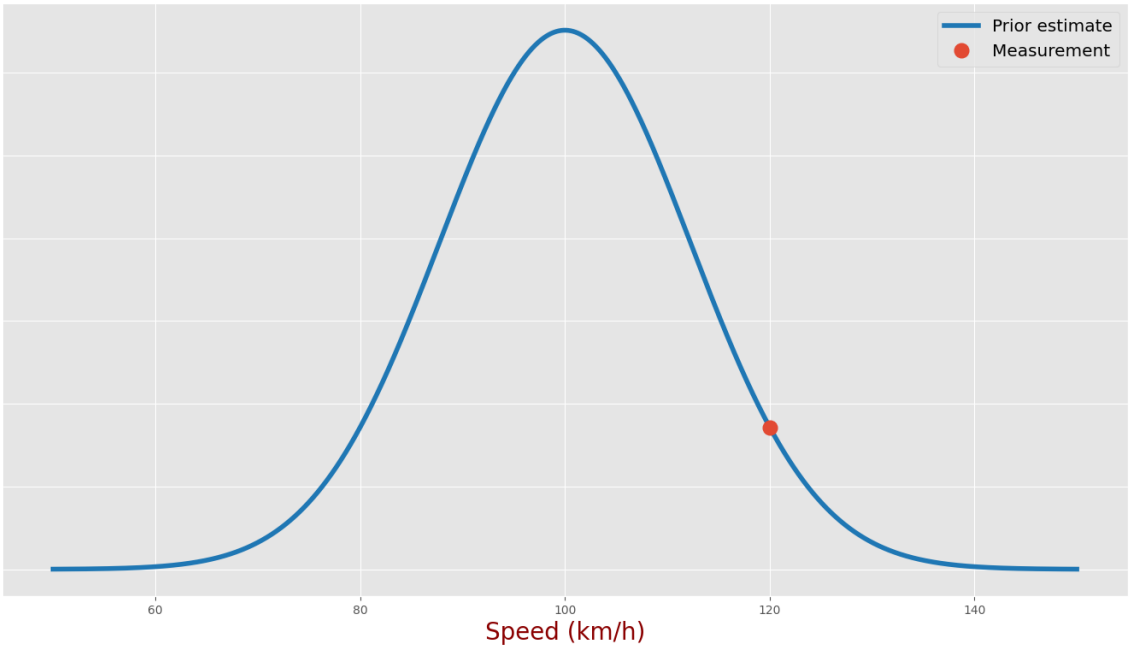


**Figure 20.2:** *Low Mahalanobis distance.*

If $p_{n,n-1} = 10km^2/h^2$, the Mahalanobis distance is:

$$d_M = \sqrt{\frac{(100-120)^2}{10}} = 6.32 \tag{20.3}$$

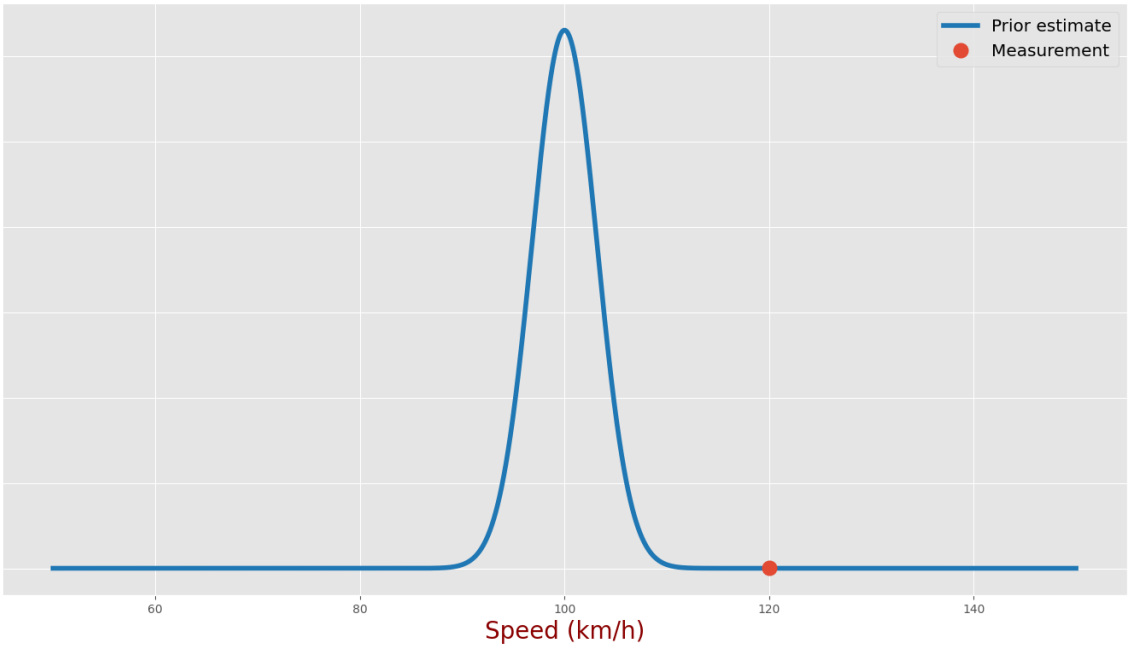


**Figure 20.3:** *High Mahalanobis distance.*

In the first case, the measurement is not an outlier since the distance of 1.63 standard

deviations is reasonable. In the second case, the measurement is definitely an outlier.

A Kalman Filter designer should define the Mahalanobis distance threshold based on the domain technical knowledge. Usually, it varies between 3 and 5.

For a multivariate Kalman Filter, the Mahalanobis distance is given by:

$$\boldsymbol{d}_M = \sqrt{(\hat{\boldsymbol{x}}_{n,n-1} - \boldsymbol{z}_n)^T \boldsymbol{P}_{n,n-1} (\hat{\boldsymbol{x}}_{n,n-1} - \boldsymbol{z}_n)} \tag{20.4}$$

Where:

$\hat{\boldsymbol{x}}_{n,n-1}$ is the prior estimate
$\boldsymbol{P}_{n,n-1}$ is the prior estimate covariance matrix
$\boldsymbol{z}_n$ is the measurement vector

## 20.2 Impact of outliers

The impact of an outlier depends on the variance of the outlier. Let us continue with the example of vehicle speed estimation. The prior estimate of the speed (prediction) $\hat{x}_{n,n-1}$ is 100km/h, and the measured speed $z_n$ is 120km/h.

We calculated the Mahalanobis distance for $p_{n,n-1} = 10km^2/h^2$:

$$d_M = \sqrt{\frac{(100 - 120)^2}{10}} = 6.32 \tag{20.5}$$

The distance between the prior estimate and the measurement is 6.32 standard deviations; therefore, the measurement is an outlier.

Let us take a look at the measurement uncertainty. The radar velocity measurement accuracy is given by:

$$\sigma_V = \frac{\lambda B}{2\sqrt{2SNR}} \tag{20.6}$$

Where:

$\sigma_V$ is a velocity measurement error (standard deviation)
$\lambda$ is the signal wavelength
$B$ is a signal bandwidth
$SNR$ is a Signal to Noise Ratio

The radar velocity measurement variance is:

$$r_n = \sigma_{V_n}^2 = \frac{\lambda^2 B^2}{8SNR_n} \tag{20.7}$$

Assume a noisy measurement with a low SNR that yields $\sigma_{V_n} = 30km/h$.

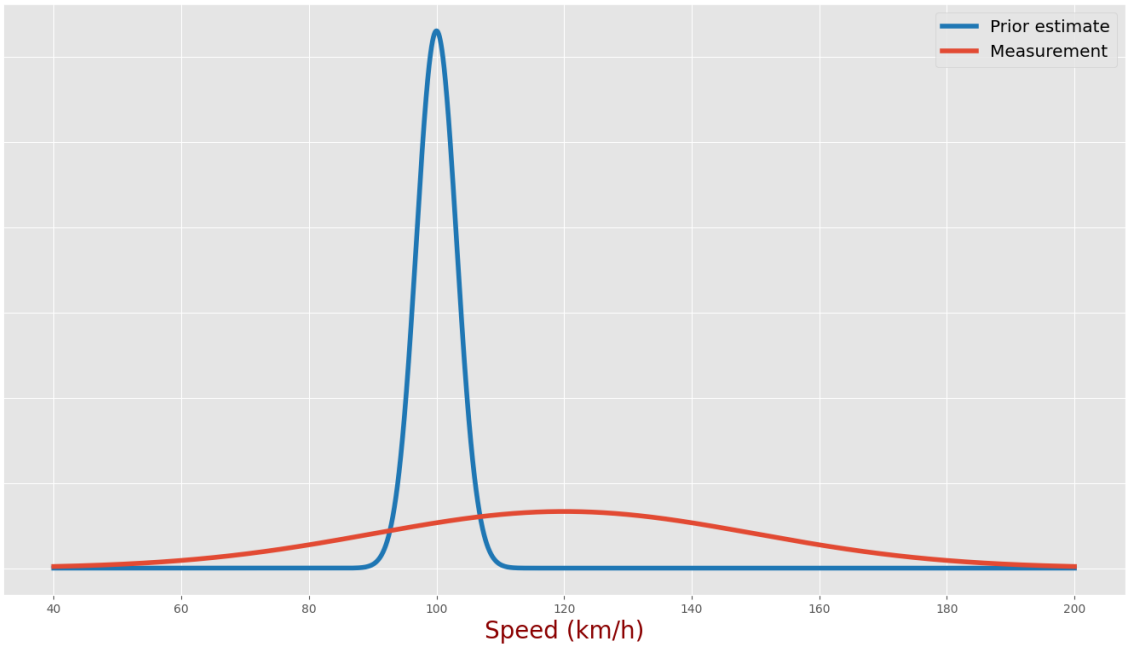


**Figure 20.4:** *Abnormal measurement with high uncertainty.*

The Kalman Gain is given by:

$$K_n = \frac{p_{n,n-1}}{p_{n,n-1} + r_n} = \frac{10}{10 + 30^2} = \frac{10}{910} = 0.01 \tag{20.8}$$

The state update equation is:

$$\hat{\boldsymbol{x}}_{n,n} = \hat{\boldsymbol{x}}_{n,n-1} + 0.01\left(\boldsymbol{z}_n - \hat{\boldsymbol{x}}_{n,n-1}\right) = 0.99\hat{\boldsymbol{x}}_{n,n-1} + 0.01\boldsymbol{z}_n \tag{20.9}$$

The Kalman Gain is very low; therefore, the impact of the noisy measurement with high uncertainty is very low.

Now assume an outlier measurement with high SNR that yields $\sigma_{V_n} = 2km/h$. Such a measurement can be caused by an anomaly.

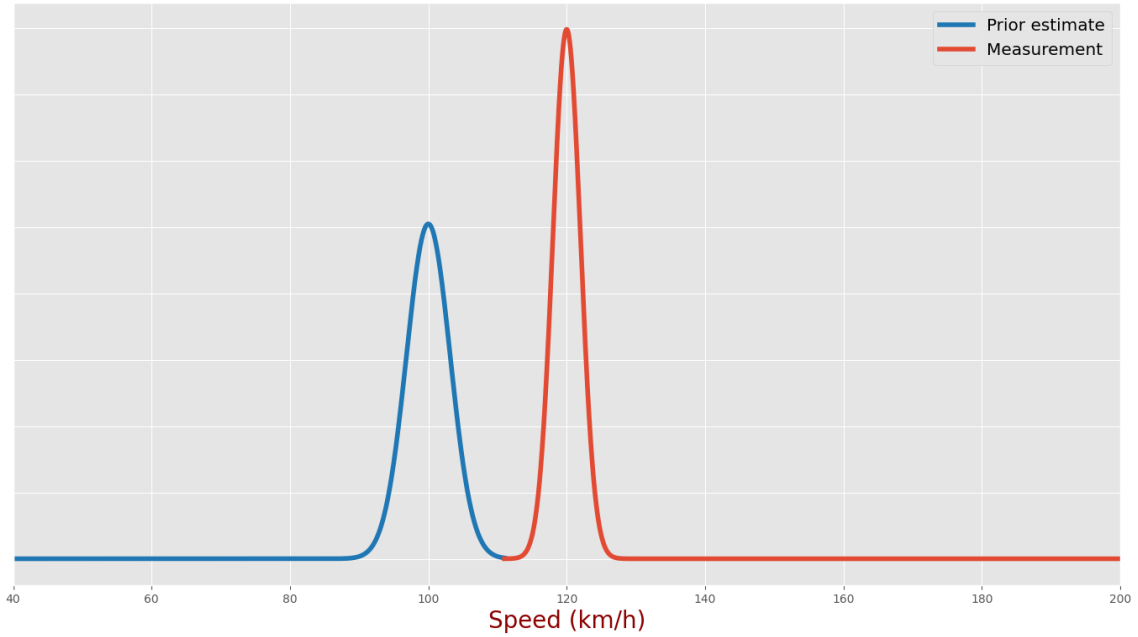


**Figure 20.5:** *Abnormal measurement with low uncertainty.*

The Kalman Gain is given by:

$$K_n = \frac{p_{n,n-1}}{p_{n,n-1} + r_n} = \frac{10}{10 + 2^2} = \frac{10}{14} = 0.71 \tag{20.10}$$

The state update equation is:

$$\hat{\boldsymbol{x}}_{n,n} = \hat{\boldsymbol{x}}_{n,n-1} + 0.71\left(\boldsymbol{z}_n - \hat{\boldsymbol{x}}_{n,n-1}\right) = 0.29\hat{\boldsymbol{x}}_{n,n-1} + 0.71\boldsymbol{z}_n \tag{20.11}$$

The Kalman Gain is high; therefore, the impact of the abnormal measurement with low uncertainty is very high.

## 20.3 Treating outliers

A high-impact outlier influences long-term filter stability. It influences the current estimation and the following estimations. Since the following estimations are based on the measurements and past estimations.

In case of an outlier with a high Mahalanobis distance, eliminate the outlier and treat it as a missing measurement.

If the outlier is unlikely or unusual, but the Mahalanobis distance is low, consider changing the outlier value to a reasonable value.

For example, when estimating the water temperature, you can set a lower temperature bound to $0^0C$ and an upper temperature bound to $100^0C$. If the measurement

temperature is $101^0C$, you can change the value to $100^0C$ (the upper temperature bound).

The outlier treatment algorithm includes the following steps:

- Identify an outlier by calculating the Mahalanobis Distance or based on the domain knowledge.
- Estimate the outlier impact.
- Eliminate the outlier or change its value to a lower or upper bound.

# 21. Kalman Filter Initialization

The Kalman Filter must be initiated with a prior estimation $\hat{\boldsymbol{x}}_{0,0}$. As well we should supply the initialization covariance $\boldsymbol{P}_{0,0}$.

## 21.1 Linear KF Initialization

Let us recall example 9 (section 9.1) - vehicle location estimation. The true vehicle location at the time $t = 0$ was $x = 300, y = 425$ .

We don't know the vehicle location, so we set the initial position, velocity, and acceleration to 0.

$$\hat{\boldsymbol{x}}_{0,0} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

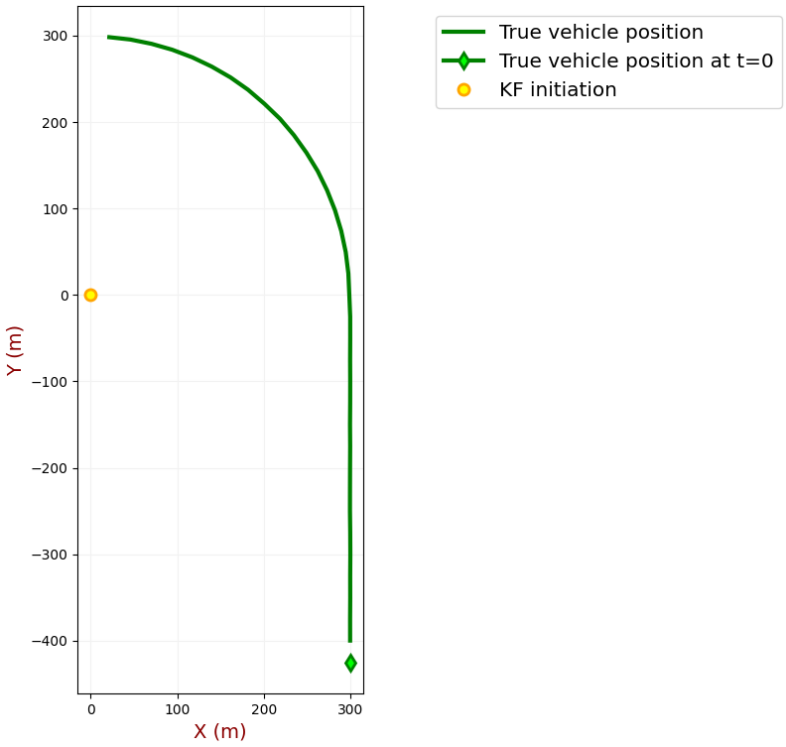


**Figure 21.1:** *LKF rough initiation.*

Figure 21.2 depicts the true vehicle position relative to initialization.

Since our initial state vector is a guess, we set a very high estimate uncertainty.

$$\hat{\boldsymbol{x}}_{0,0} = \boldsymbol{P}_{0,0} = \begin{bmatrix} 500 & 0 & 0 & 0 & 0 & 0 \\ 0 & 500 & 0 & 0 & 0 & 0 \\ 0 & 0 & 500 & 0 & 0 & 0 \\ 0 & 0 & 0 & 500 & 0 & 0 \\ 0 & 0 & 0 & 0 & 500 & 0 \\ 0 & 0 & 0 & 0 & 0 & 500 \end{bmatrix}$$

We have seen that the high estimate uncertainty results in a high Kalman Gain by giving a high weight to the measurement.

Finally, we could track the vehicle accurately, as shown in the next figure.

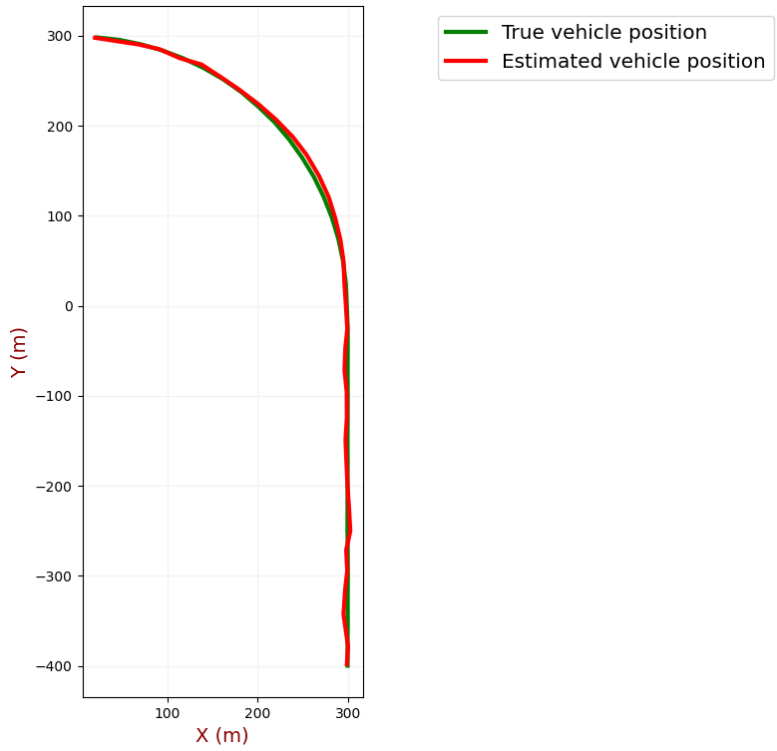


**Figure 21.2:** *LKF rough initiation: True vs. estimated position.*

Although KF initialization was a shot in the dark, it wasn't a problem since we set a high initial uncertainty.

Let us see what happens if we initiate the Kalman Filter close to the actual vehicle position. Assume that we had approximate information about the actual vehicle

position at the time $t = 0$. We set:

$$\hat{\boldsymbol{x}}_{0,0} = \begin{bmatrix} 303 \\ 0 \\ 0 \\ -428 \\ 0 \\ 0 \end{bmatrix}$$

In this case, the vehicle is also accurately tracked, as shown in the following figure.

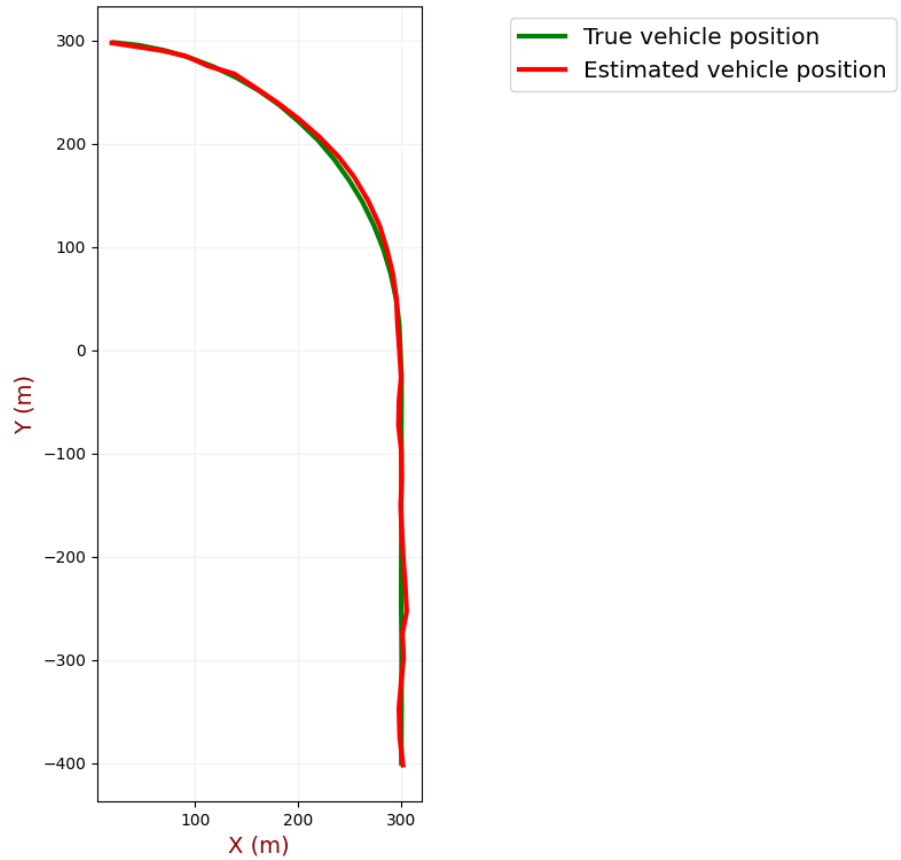


**Figure 21.3:** *LKF fine initiation: True vs. estimated position.*

So can we always initiate the Kalman Filter with a random value? Let us compare the estimation uncertainty for both cases.

In the following plot, we can see the estimation variance of the random initialization (the blue line) and the estimation variance of the fine initialization (the orange line).

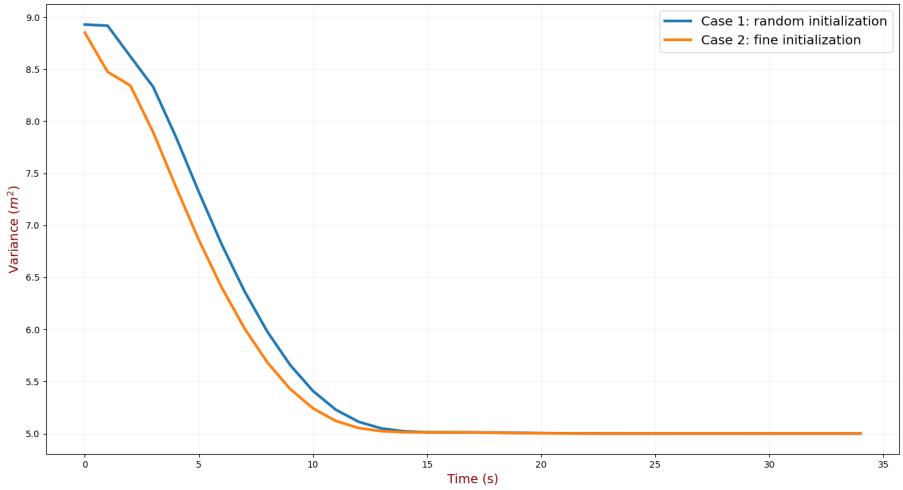


**Figure 21.4:** *LKF uncertainty: rough vs. fine initiation.*

We can see a lower uncertainty of the fine initialization. The KF converges faster if we initiate it with meaningful values.

## 21.2 Non-linear KF initialization

Let us recall example 11 (section 13.7) - "vehicle location estimation." The scenario in example 11 (section 13.7) is similar to example 9 (section 9.1). The measurement is performed by radar. If the EKF is initiated close to the true position, the EKF converges quickly and accurately tracks the target.

$$\hat{\boldsymbol{x}}_{0,0} = \begin{bmatrix} 100 \\ 0 \\ 0 \\ -100 \\ 0 \\ 0 \end{bmatrix} \quad \boldsymbol{P}_{0,0} = \begin{bmatrix} 500 & 0 & 0 & 0 & 0 & 0 \\ 0 & 500 & 0 & 0 & 0 & 0 \\ 0 & 0 & 500 & 0 & 0 & 0 \\ 0 & 0 & 0 & 500 & 0 & 0 \\ 0 & 0 & 0 & 0 & 500 & 0 \\ 0 & 0 & 0 & 0 & 0 & 500 \end{bmatrix}$$

The following figure depicts the true vehicle position and initialization point (on the left) and estimated position vs. true position (on the right).

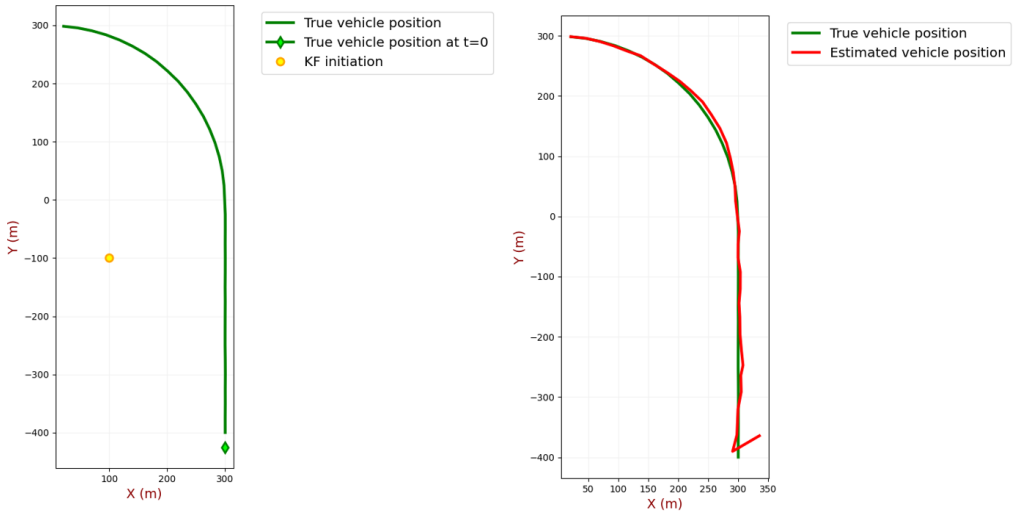


**Figure 21.5:** *Non-linear KF fine initiation.*

Let's see the EKF performance when the initialization point is moved farther.

$$\hat{\boldsymbol{x}}_{0,0} = \begin{bmatrix} 100 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \boldsymbol{P}_{0,0} = \begin{bmatrix} 500 & 0 & 0 & 0 & 0 & 0 \\ 0 & 500 & 0 & 0 & 0 & 0 \\ 0 & 0 & 500 & 0 & 0 & 0 \\ 0 & 0 & 0 & 500 & 0 & 0 \\ 0 & 0 & 0 & 0 & 500 & 0 \\ 0 & 0 & 0 & 0 & 0 & 500 \end{bmatrix}$$

The EKF provides wrong estimations. After some time, it converges and tracks the target, as shown in the following figure.

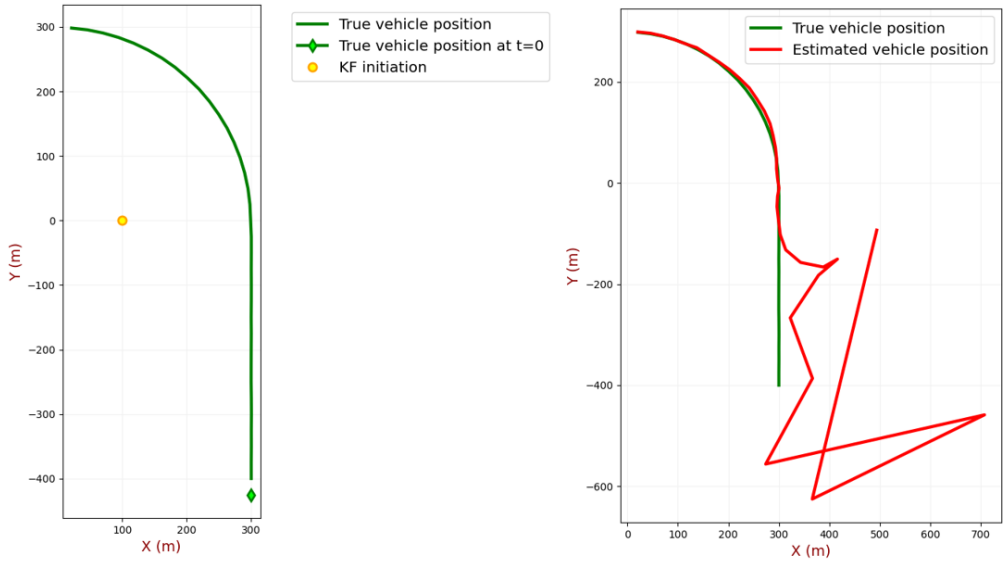


**Figure 21.6:** *Non-linear KF rough initiation.*

If we move the initialization point again, it takes more time for EKF to converge.

$$\hat{\boldsymbol{x}}_{0,0} = \begin{bmatrix} 300 \\ 0 \\ 0 \\ 300 \\ 0 \\ 0 \end{bmatrix} \quad \boldsymbol{P}_{0,0} = \begin{bmatrix} 500 & 0 & 0 & 0 & 0 & 0 \\ 0 & 500 & 0 & 0 & 0 & 0 \\ 0 & 0 & 500 & 0 & 0 & 0 \\ 0 & 0 & 0 & 500 & 0 & 0 \\ 0 & 0 & 0 & 0 & 500 & 0 \\ 0 & 0 & 0 & 0 & 0 & 500 \end{bmatrix}$$

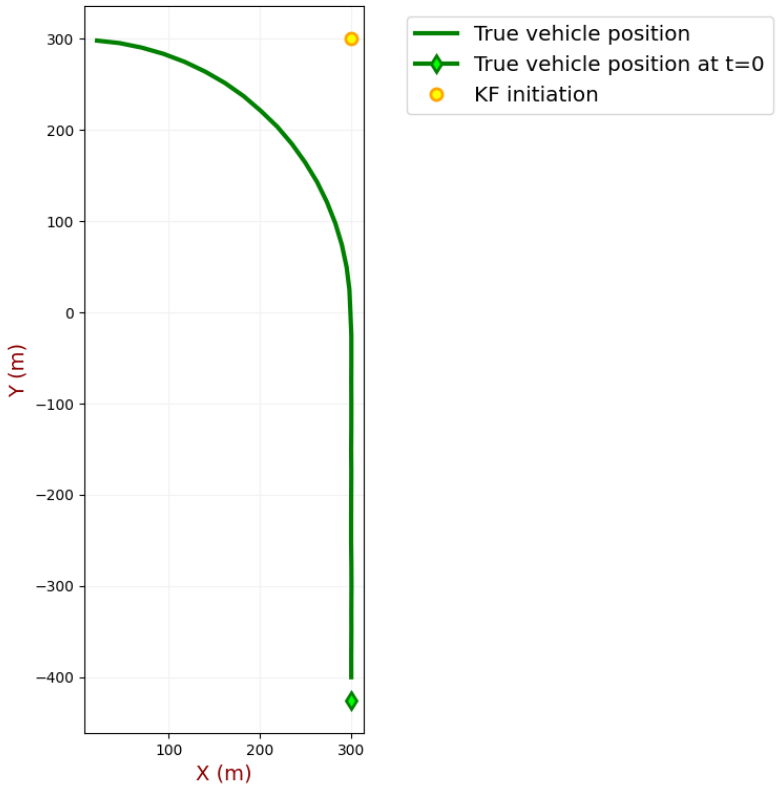


**Figure 21.7:** *Non-linear KF very rough initiation.*

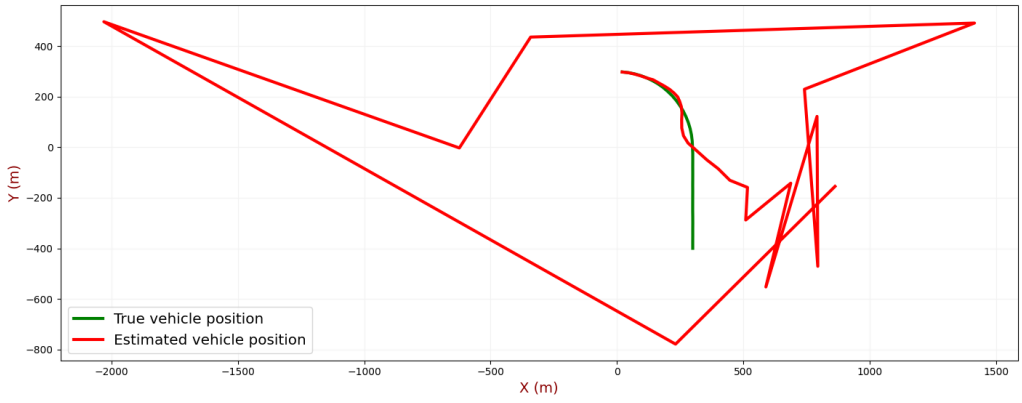


**Figure 21.8:** *Non-linear KF very rough initiation: filter performance.*

Unlike the LKF, the non-linear Kalman Filter requires fine initiation; otherwise, it wouldn't be able to provide satisfactory results.

## 21.3 KF initialization techniques

There is no generic initialization technique or method. It depends on the system.

For example, the radar search process provides coarse target measurement used as initialization for the tracker.

For other systems, an educated guess is sufficient for KF initialization.

You can also use the first measurement as initialization data and start the estimation process with the second measurement.

# 22. KF Development Process

The Kalman Filter development process includes four phases:

- Kalman Filter Design
- Simulation
- Performance Examination
- Parameters Tuning

The following diagram describes the KF development process.

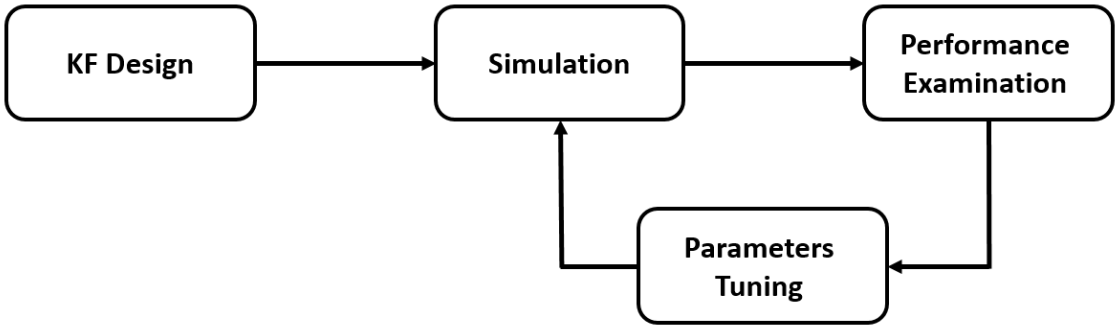


**Figure 22.1:** *KF development process.*

Let us explore each phase.

## 22.1 Kalman Filter Design

The KF design includes 6 steps:

**Step 1**: Define the Dynamic Model of the system.

- You are lucky if you know your system's Dynamic Model (state extrapolation equation).
- Otherwise, write down the differential equations that govern your system (the state space representation), as described in Appendix C ("Modeling linear dynamic systems"):

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) \\ \boldsymbol{y}(t) &= \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{D}\boldsymbol{u}(t) \end{aligned} \tag{22.1}$$

- Solve the equations to determine the state extrapolation equation.

$$\hat{\boldsymbol{x}}_{n+1,n} = \boldsymbol{F}\hat{\boldsymbol{x}}_{n,n} + \boldsymbol{G}\hat{\boldsymbol{u}}_{n,n} \tag{22.2}$$

- For a non-linear system, the differential equation is in the form of the following:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{g}\left(\boldsymbol{x}(t)\right) \tag{22.3}$$

- Solve the differential equation to determine the state extrapolation equation:

$$\hat{\boldsymbol{x}}_{n+1,n} = \boldsymbol{f}\left(\hat{\boldsymbol{x}}_{n,n}\right) + \boldsymbol{G}\hat{\boldsymbol{u}}_{n,n} \tag{22.4}$$

- If you can't solve the differential equation, perform the numerical integration using a computer for the following integral:

$$\boldsymbol{f}(\hat{\boldsymbol{x}}_{n,n}) = \hat{\boldsymbol{x}}_{n,n} + \int_{t_n}^{t_{n+1}} \boldsymbol{g}\left(\boldsymbol{x}(t)\right) dt \tag{22.5}$$

**Step 2**: Define the measurement equation.

- For LKF:

$$\boldsymbol{z}_n = \boldsymbol{H}\boldsymbol{x}_n \tag{22.6}$$

- For EKF:

$$\boldsymbol{z}_n = \boldsymbol{h}(\boldsymbol{x}_n) \tag{22.7}$$

**Step 3**: Process Noise error sources:

- Write down a list of all the error sources.
- Define the Process Noise matrix $\boldsymbol{Q}$ based on uncorrelated error sources (white noise).
- The correlated noise sources must be a part of the Dynamic Model.

**Step 4**: Decide on the KF initialization method:

- The initialization method depends on your system.
- It can be an educated guess, a coarse measurement, or the first measurement.
- Decide on the initialization covariance $\boldsymbol{P}_{0,0}$.

**Step 5**: Add treatment for missing measurements.

**Step 6**: Decide on the outliers treatment method:

- Which measurement should be considered an outlier?
- How to treat outliers?

## 22.2 Simulation

Simulation is a critical phase of the Kalman Filter development process. It allows us to observe the filter performance in a controlled environment. The following diagram describes the simulation block diagram.

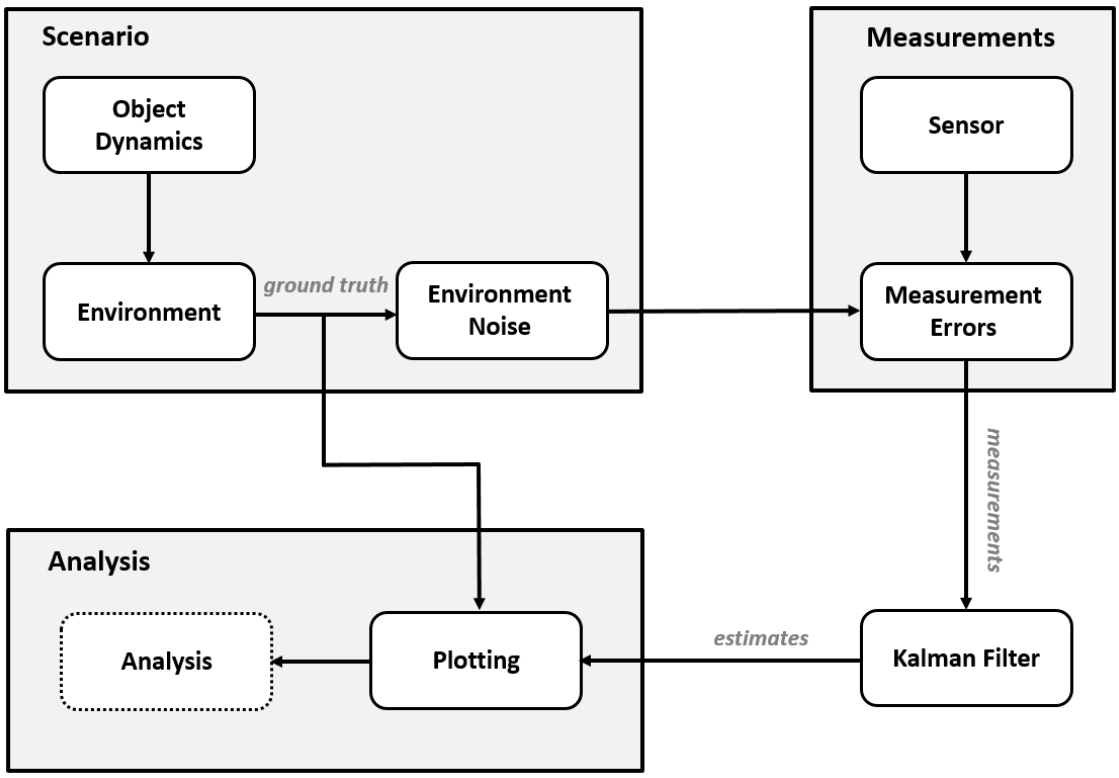


**Figure 22.2:** *KF simulation diagram.*

The simulation is a computer program that contains the following modules:

- Scenario
- Measurements
- Kalman Filter
- Analysis

### 22.2.1 Scenario Module

The scenario module simulates what is happening in the real world.

The object-dynamics module generates different dynamics scenarios of the object of interest. For example, the vehicle trajectory in example 9 (section 9.1), the pendulum position in example 12 (section 13.8), or the heating liquid temperature in example 7 (section 5.3). The object dynamics shall include possible extreme conditions, like sharp target maneuvers.

The environment module should simulate the environmental influence on the target dynamics, like air turbulence influence on the aircraft, the icy road influence on the vehicle, and the air-conditioner influence on the liquid temperature.

The generated scenario is a true state vector or the **ground truth**.

The environment noise module adds environmental noise to the ground truth - for example, an aircraft radar cross-section fluctuations, radio interferences, etc.

The Kalman Filter designer should create enough object dynamics scenarios to challenge his Kalman Filter design.

### 22.2.2 Measurements Module

The measurement model should create the sensor measurements, which are the input to Kalman Filter.

For simple sensor like scales with a measurement uncertainty $\sigma$, generate a normally distributed random noise with a standard deviation $\sigma$, and add the noise to the generated scenario.

For EKF, the ground truth should be transformed into the sensor coordinates. For example, the vehicle position in example 11 coordinates are $X$ and $Y$. The sensor coordinates are range ($r$) and the bearing angle ($\theta$). Then the range noise and the angle noise should be added to the transformed coordinates.

More sophisticated sensors require simulation of the sensor system to simulate phenomena like clock drift, imperfections due to sampling, and other system-specific aspects like beam broadening in radar.

### 22.2.3 Kalman Filter Module

The Kalman Filter simulation. See section 22.1 (Kalman Filter Design) for details.

### 22.2.4 Analysis

The analysis process includes the plotting module that creates results visualization. The interesting plots are:

- Estimates vs. measurements vs. ground truth.
- If possible, add estimation confidence intervals or confidence ellipses to the plot.
- Estimation uncertainty vs. time
- Kalman Gain vs. time

The analysis module can be the Kalman Filter designer that analyses plots or computer software that analyses results and makes conclusions.

### 22.2.5 Performance Examination

Before examining the KF performance, you should define the acceptable performance criteria:

- The Root Mean Square (RMS) error between the ground truth and estimates.
- The RMS error between the ground truth and predictions.
- The maximum error between the ground truth and estimates.
- The maximum error between the ground truth and predictions.
- The filter convergence time.
- Prediction and estimation uncertainties.
- The confidence levels.

The criteria must be reasonable. Don't expect low errors if your sensor is not accurate or environmental noise is too high.

If the KF errors are too high, examine your dynamic model. You can increase the process noise ($\boldsymbol{Q}$) value. However, as we've seen in example 8 (section 5.4), improving the dynamic model is a better approach.

Examine the filter convergence. The Kalman Gain should constantly decrease until it reaches a certain floor level. As a result, the uncertainty of the estimates should also decrease. If the convergence time is too high, examine the influence of the KF initialization on the convergence time. Find methods for finer initialization values. You can also decrease the process noise ($\boldsymbol{Q}$) value, but make sure it doesn't increase the errors.

Figure 22.3 describes KF filter performance in example 8 (section 5.4). We can see the 95% confidence intervals.

Assume that 100% of the ground truth is within the confidence intervals. Is it good? Not necessarily. Your KF estimates uncertainty is too high. You can achieve lower uncertainty estimates without compromising the confidence level.

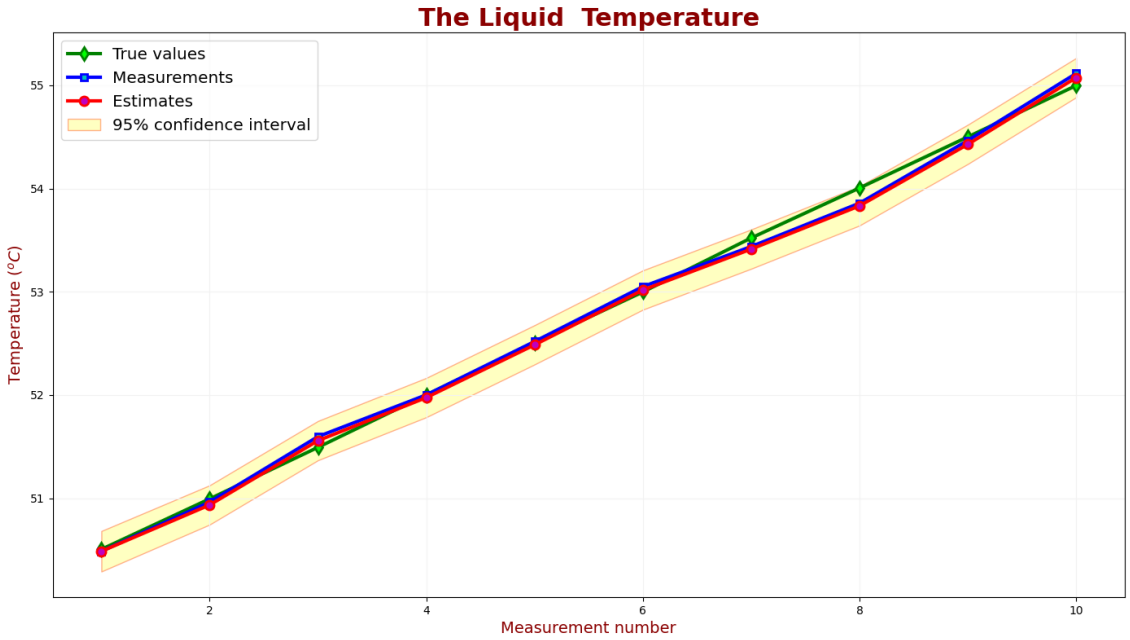


**Figure 22.3:** *KF simulation diagram.*

# V Appendices

# A. The expectation of variance

We already know what the random variable is and what the expected value (or expectation) is. If not, please read chapter 2 ("Essential background I").

## A.1 Expectation rules

The expectation is denoted by the capital letter $E$. The expectation of the random variable $E(X)$ equals the mean of the random variable:

**Expectation**

$$E(X) = \mu_X \tag{A.1}$$

where $\mu_X$ is the mean of the random variable.

Here are some basic expectation rules:

| | Rule | Notes |
|---|---|---|
| **1** | $E(X) = \mu_X = \Sigma x p(x)$ | $p(x)$ is the probability of $x$ (discrete case) |
| **2** | $E(a) = a$ | $a$ is constant |
| **3** | $E(aX) = aE(X)$ | $a$ is constant |
| **4** | $E(a \pm X) = a \pm E(X)$ | $a$ is constant |
| **5** | $E(a \pm bX) = a \pm bE(X)$ | $a$ and $b$ are constant |
| **6** | $E(X \pm Y) = E(X) \pm E(Y)$ | $Y$ is another random variable |
| **7** | $E(XY) = E(X)E(Y)$ | If $X$ and $Y$ are independent |

**Table A.1:** *Expectation rules.*

All the rules are quite straightforward and don't need proof.

## A.2 The expectation of the variance

The expectation of variance is given by:

**Expectation of the variance**

$$V(X) = \sigma_x^2 = E(X^2) - \mu_X^2 \tag{A.2}$$

Where $V(X)$ is the variance of $X$

The proof

| | Notes |
|---|---|
| $V(X) = \sigma_X^2 = E((X - \mu_X)^2)$ | |
| $= E(X^2 - 2X\mu_X + \mu_X^2)$ | |
| $= E(X^2) - E(2X\mu_X) + E(\mu_X^2)$ | Applied rule number 5: $E(a \pm bX) = a \pm bE(X)$ |
| $= E(X^2) - 2\mu_X E(X) + E(\mu_X^2)$ | Applied rule number 3: $E(aX) = aE(X)$ |
| $= E(X^2) - 2\mu_X E(X) + \mu_X^2$ | Applied rule number 2: $E(a) = a$ |
| $= E(X^2) - 2\mu_X\mu_X + \mu_X^2$ | Applied rule number 1: $E(X) = \mu_X$ |
| $= E(X^2) - \mu_X^2$ | |

**Table A.2:** *Variance expectation rule.*

## A.3 The expectation of the body displacement variance

The body position displacement variance in terms of time and velocity is given by:

$$V(x) = \Delta t^2 V(v) \tag{A.3}$$

or

$$\sigma_x^2 = \Delta t^2 \sigma_v^2 \tag{A.4}$$

Where:
$x$ is the displacement of the body
$v$ is the velocity of the body
$\Delta t$ is the time interval

The proof

| | Notes |
|---|---|
| $V(K) = \sigma_K^2 = E(K^2) - \mu_K^2$ | |
| $= E((v\Delta t)^2) - (\mu_v \Delta t)^2$ | Express the body position variance in terms of time and velocity: $x = \Delta t v$ |
| $= E(v^2 \Delta t^2) - \mu_v^2 \Delta t^2$ | |
| $= \Delta t^2 E(v^2) - \mu_v^2 \Delta t^2$ | Applied rule number 3: $E(aX) = aE(X)$ |
| $= \Delta t^2 (E(v^2) - \mu_v^2)$ | |
| $= \Delta t^2 V(v)$ | Applied expectation of variance rule: $V(X) = E(X^2) - \mu_X^2$ |

**Table A.3:** *Variance square expectation rule.*

# B. Confidence Interval

This appendix describes the method of confidence interval computation for a one-dimensional normal distribution.

## B.1 Cumulative Probability

The **cumulative probability** is the likelihood that the value of a random variable is within a specific range.

$$P\left(a \leq X \leq b\right) \tag{B.1}$$

Let us return to the pizza delivery distribution example (see chapter 2 - "Essential background I"). We want to find the likelihood that the pizza in city 'A' would be delivered within 33 minutes:

$$P\left(0 \leq X \leq 33\right)$$

Reminder, the pizza delivery time in the city 'A' is normally distributed with a mean of 30 minutes and a standard deviation of 5 minutes ($\mu = 30, \sigma = 5$). We need to find the area under the PDF curve between zero and 33 minutes:

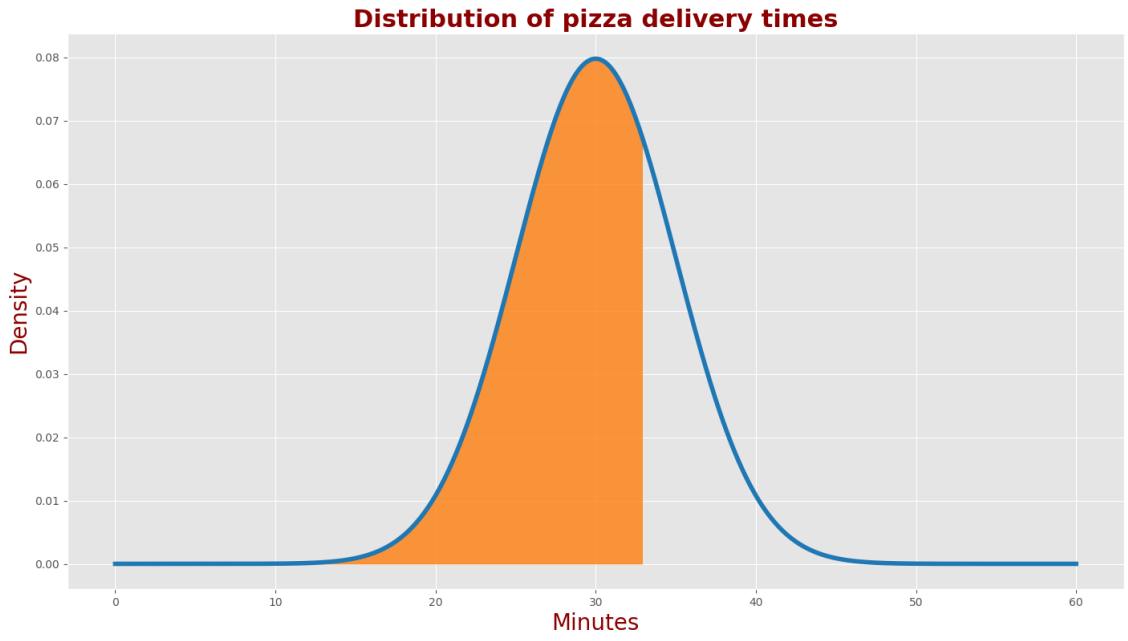


**Figure B.1:** *Cumulative Probability.*

The filled area under Gaussian is given by:

$$F\left(x;\mu,\sigma^2\right)=\frac{1}{\sqrt{2\pi\sigma^2}}\int_0^{33}exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right)dx \tag{B.2}$$

Don't worry. We won't need to compute this integral.

Let us define a **standardized score** (also called a **z-score**) to simplify the problem.

z-score is a standardized random variable with a mean of 0 and a standard deviation of 1 ($\mu=0,\sigma=1$).

$$z=\frac{x-\mu}{\sigma} \tag{B.3}$$

A z-score defines the distance of $x$ from the mean in units of standard deviations. For example:

- If $z-score=1$, the value of $z$ is one standard deviation above the mean.
- If $z-score=-2.5$, the value of $z$ is 2.5 standard deviations below the mean.
- If $z-score=0$, the value of $z$ equals the mean.

The pizza delivery time in city 'A' is a random variable with a mean of 30 and a standard deviation of 5 ($\mu=30,\sigma=5$).

z-score for 33 minutes is:

$$z=\frac{33-30}{5}=0.6$$

z-score for 0 minutes is:

$$z=\frac{0-30}{5}=-6 \tag{B.4}$$

The PDF of $z$ is a **standard normal distribution**:

$$F\left(z\right)=\frac{1}{\sqrt{2\pi}}exp\left(-0.5z^2\right) \tag{B.5}$$

The cumulative probability is the area under the PDF between $-\infty$ and $z$.

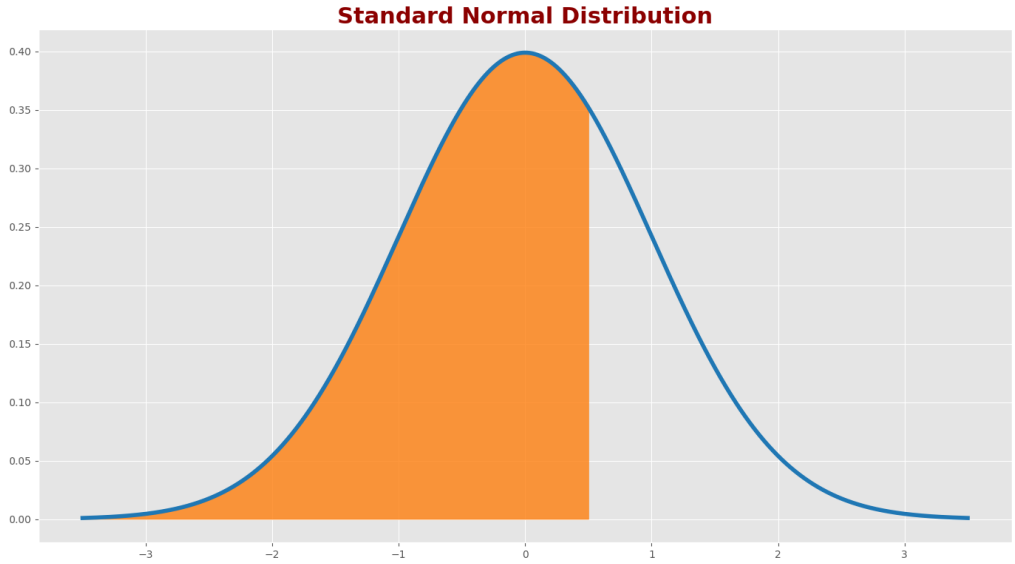


**Figure B.2:** *Standard Normal Distribution.*

The Cumulative Probability $CP$ of $z$ is given by:

$$CP\left(z\right) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} exp\left(-0.5z^2\right) dz \tag{B.6}$$

For our example, we need to find the following:

$$P(-6 \leq z \leq 0.6) = CP(z = 0.6) - CP(z = -6)$$

Calculating the PDF integral is not straightforward and requires much work. The faster method is to use statistical z-score tables or computer software packages.

z-score tables contain precalculated cumulative probabilities for different z-scores. Figure B.3 exemplifies the location of the cumulative probability for z-score (z=0.6).

| z | .00 | .01 | .02 | .03 | .04 | .05 | .06 | .07 | .08 | .09 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0.0 | .5000 | .5040 | .5080 | .5120 | .5160 | .5199 | .5239 | .5279 | .5319 | .5359 |
| 0.1 | .5398 | .5438 | .5478 | .5517 | .5557 | .5596 | .5636 | .5675 | .5714 | .5753 |
| 0.2 | .5793 | .5832 | .5871 | .5910 | .5948 | .5987 | .6026 | .6064 | .6103 | .6141 |
| 0.3 | .6179 | .6217 | .6255 | .6293 | .6331 | .6368 | .6406 | .6443 | .6480 | .6517 |
| 0.4 | .6554 | .6591 | .6628 | .6664 | .6700 | .6736 | .6772 | .6808 | .6844 | .6879 |
| 0.5 | .6915 | .6950 | .6985 | .7019 | .7054 | .7088 | .7123 | .7157 | .7190 | .7224 |
| 0.6 | .7257 | .7291 | .7324 | .7357 | .7389 | .7422 | .7454 | .7486 | .7517 | .7549 |
| 0.7 | .7580 | .7611 | .7642 | .7673 | .7704 | .7734 | .7764 | .7794 | .7823 | .7852 |
| 0.8 | .7881 | .7910 | .7939 | .7967 | .7995 | .8023 | .8051 | .8078 | .8106 | .8133 |
| 0.9 | .8159 | .8186 | .8212 | .8238 | .8264 | .8289 | .8315 | .8340 | .8365 | .8389 |
| 1.0 | .8413 | .8438 | .8461 | .8485 | .8508 | .8531 | .8554 | .8577 | .8599 | .8621 |
| 1.1 | .8643 | .8665 | .8686 | .8708 | .8729 | .8749 | .8770 | .8790 | .8810 | .8830 |
| 1.2 | .8849 | .8869 | .8888 | .8907 | .8925 | .8944 | .8962 | .8980 | .8997 | .9015 |
| 1.3 | .9032 | .9049 | .9066 | .9082 | .9099 | .9115 | .9131 | .9147 | .9162 | .9177 |

**Figure B.3:** *z-score table.*

$$CP\,(z = 0.6) = 0.7257$$

You can use scientific computer software packages for a z-score integral computation.

The following commands compute the z-score integral in different computer software packages:

| Computer Software Package | Command |
|---|---|
| **Python** | from scipy.stats import norm<br>norm.cdf(z) |
| **MATLAB** | normcdf(z) |
| **Excel** | NORM.DIST(z, 0, 1, TRUE) |

**Table B.1:** *Cumulative distribution from z-score.*

**Python example:**

```
from scipy.stats import norm

norm.cdf(0.6)
0.7257468822499265

norm.cdf(-6)
9.865876450376946e-10
```

**MATLAB example:**

```
normcdf(0.6)

0.7257

normcdf(-6)

9.8659e-10
```

$$P(-6 \leq z \leq 0.6) = 0.7257 - 0 = 0.7257$$

The likelihood of having a pizza in city 'A' within 33 minutes is 72.57%.

Or in other words, 72.57 percentile of the pizza delivery time in city 'A' is 33 minutes.

R When using computer software packages, you don't need to calculate the z-score. You can specify the mean and standard deviation as an argument of the software function.

The following commands compute the cumulative distribution in different computer software packages:

| Computer Software Package | Command |
|---|---|
| **Python** | from scipy.stats import norm<br>norm.cdf(x, mu, sigma) |
| **MATLAB** | normcdf(x, mu, sigma) |
| **Excel** | NORM.DIST(x, mu, sigma, TRUE) |

**Table B.2:** *Cumulative distribution from $\mu$ and $\sigma$.*

**Python example:**

```
from scipy.stats import norm

norm.cdf(33, 30, 5)
0.7257468822499265
```

**MATLAB example:**

```
normcdf(33, 30, 5)

0.7257
```

## B.2 Normal inverse cumulative distribution

In this chapter, we would like to answer a reverse question. What is the cumulative distribution for a given percentile?

For example, what is the $80^{\text{th}}$ percentile for the pizza delivery time in the city' A'?

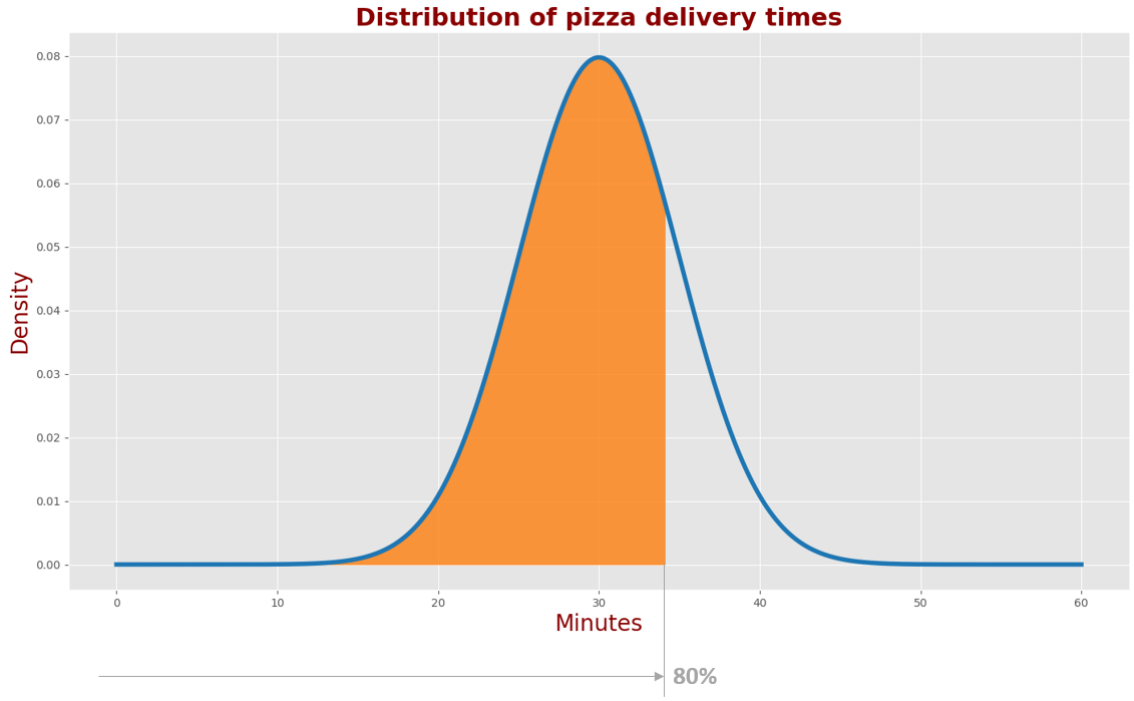


**Figure B.4:** *Normal Inverse Cumulative Distribution.*

One method is to use the z – score table:

- In the table below, find the cumulative distribution value closest to 0.8.
- The $z - score$ is a sum of the row and the column of $z - value$: $z = 0.84$.

| z | .00 | .01 | .02 | .03 | .04 | .05 | .06 | .07 | .08 | .09 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0.0 | .5000 | .5040 | .5080 | .5120 | .5160 | .5199 | .5239 | .5279 | .5319 | .5359 |
| 0.1 | .5398 | .5438 | .5478 | .5517 | .5557 | .5596 | .5636 | .5675 | .5714 | .5753 |
| 0.2 | .5793 | .5832 | .5871 | .5910 | .5948 | .5987 | .6026 | .6064 | .6103 | .6141 |
| 0.3 | .6179 | .6217 | .6255 | .6293 | .6331 | .6368 | .6406 | .6443 | .6480 | .6517 |
| 0.4 | .6554 | .6591 | .6628 | .6664 | .6700 | .6736 | .6772 | .6808 | .6844 | .6879 |
| 0.5 | .6915 | .6950 | .6985 | .7019 | .7054 | .7088 | .7123 | .7157 | .7190 | .7224 |
| 0.6 | .7257 | .7291 | .7324 | .7357 | .7389 | .7422 | .7454 | .7486 | .7517 | .7549 |
| 0.7 | .7580 | .7611 | .7642 | .7673 | .7704 | .7734 | .7764 | .7794 | .7823 | .7852 |
| 0.8 | .7881 | .7910 | .7939 | .7967 | .7995 | .8023 | .8051 | .8078 | .8106 | .8133 |
| 0.9 | .8159 | .8186 | .8212 | .8238 | .8264 | .8289 | .8315 | .8340 | .8365 | .8389 |
| 1.0 | .8413 | .8438 | .8461 | .8485 | .8508 | .8531 | .8554 | .8577 | .8599 | .8621 |
| 1.1 | .8643 | .8665 | .8686 | .8708 | .8729 | .8749 | .8770 | .8790 | .8810 | .8830 |
| 1.2 | .8849 | .8869 | .8888 | .8907 | .8925 | .8944 | .8962 | .8980 | .8997 | .9015 |
| 1.3 | .9032 | .9049 | .9066 | .9082 | .9099 | .9115 | .9131 | .9147 | .9162 | .9177 |

**Figure B.5:** *z-score table.*

Now, we must convert $z$ to $x$:

$$z = \frac{x - \mu}{\sigma} \tag{B.7}$$

$$x = z\sigma + \mu = 0.84 \times 5 + 30 = 34.2$$

The $80^{\text{th}}$ percentile for the pizza delivery time in the city 'A' is 34.2 minutes.

If you use computer software, you can use the following commands:

| Computer Software Package | Command |
|---|---|
| **Python** | from scipy.stats import norm<br>norm.ppf(x, mu, sigma) |
| **MATLAB** | norminv(p, mu, sigma) |
| **Excel** | NORMINV(x, mu, sigma) |

**Table B.3:** *Normal cumulative distribution.*

**Python example:**

```
from scipy.stats import norm

norm.ppf(0.8, 30, 5)
34.20810616786457
```

**MATLAB example:**

```
norminv(0.8, 30, 5)

34.2081
```

## B.3 Confidence interval

A normally distributed random variable is described by mean ($\mu$) and standard deviation ($\sigma$). A confidence interval is a probability that a parameter falls between a set of values for a certain proportion of times.

Assume a weight measurement of 80kg with a measurement standard deviation ($\sigma$) of 2kg. The probability that the true weight falls between 78kg and 82kg is 68.25%.

Usually, we are interested in higher confidence levels, such as 90% or 95%. Let us see how to find it.

The following plot describes the standard normal distribution ($\mu = 0, \sigma = 1$). We want to find a 90% confidence interval.

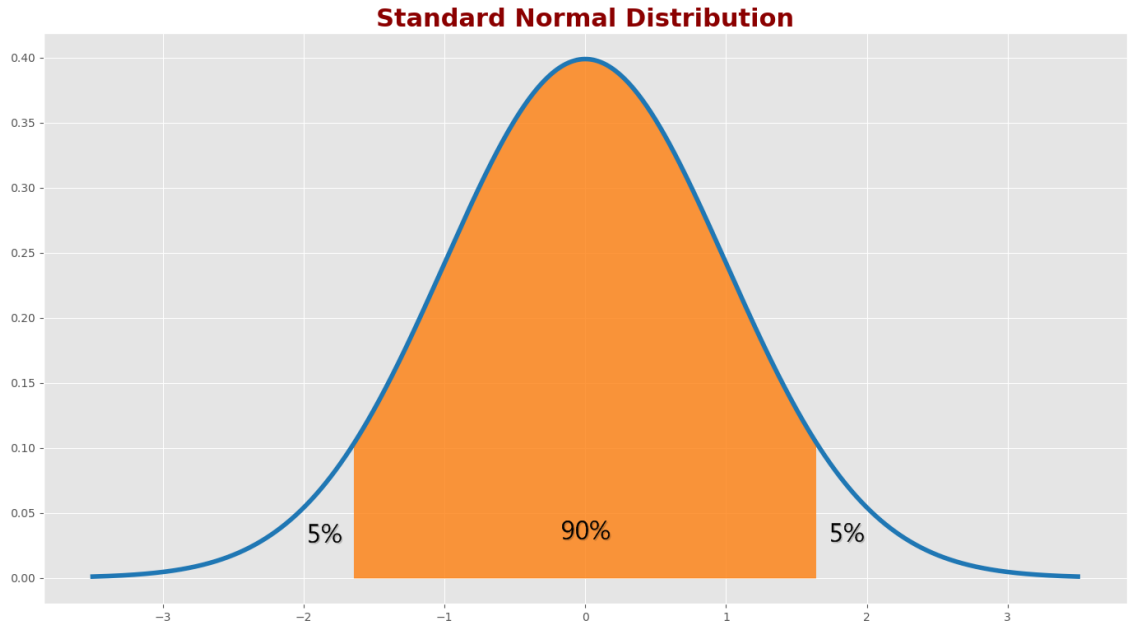


**Figure B.6:** *Confidence interval.*

The area of the filled region under the curve is 90% of the total area. The area of the unfilled region is 10% of the total area. The area of the unfilled region on the left is 5% of the total area. We can find a z-score for percentile 5 or percentile 95.

**Python example:**

```
from scipy.stats import norm

norm.ppf(0.05)
-1.6448536269514729

norm.ppf(0.95)
1.6448536269514722
```

**MATLAB example:**

```
norminv(0.05)
-1.6449

norminv(0.95)
1.6449
```

The 90% confidence interval is $(\pm 1.645\sigma)$.

For the weight measurement example, the 90% confidence interval is $\pm 3.29kg$. The probability that the true weight falls between 76.71kg and 83.29kg is 90%.

# C. Modeling linear dynamic systems

This chapter generalizes dynamic model derivation for any linear dynamic system. The following description includes integrals and differential equations.

**This chapter is the most challenging chapter of the book. It is not required for the understanding of the Kalman Filter principles. If you feel uncomfortable with this math – feel free to skip it.**

On my side, I tried to make my explanations as straightforward and easy to follow as possible, and, of course, I provided real-life examples.

## C.1 Derivation of the state extrapolation equation

Our goal is to derive the state extrapolation equation in the form:

$$\hat{\boldsymbol{x}}_{n+1,n} = \boldsymbol{F}\hat{\boldsymbol{x}}_{n,n} + \boldsymbol{G}\hat{\boldsymbol{u}}_{n,n} + \boldsymbol{w}_n \tag{C.1}$$

To do that, we need to model the dynamic system. In other words, to figure out the **state space representation** of the dynamic system. The following two equations are the state-space representation of the Linear Time Invariant (LTI) system:

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) \\ \boldsymbol{y}(t) &= \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{D}\boldsymbol{u}(t) \end{aligned} \tag{C.2}$$

Where:

- $\boldsymbol{x}$ is the state vector
- $\boldsymbol{y}$ is the output vector
- $\boldsymbol{A}$ is the system's dynamics matrix
- $\boldsymbol{B}$ is the input matrix
- $\boldsymbol{C}$ is the output matrix
- $\boldsymbol{D}$ is the feedthrough matrix

The following diagram summarizes the process of the state extrapolation equation derivation.

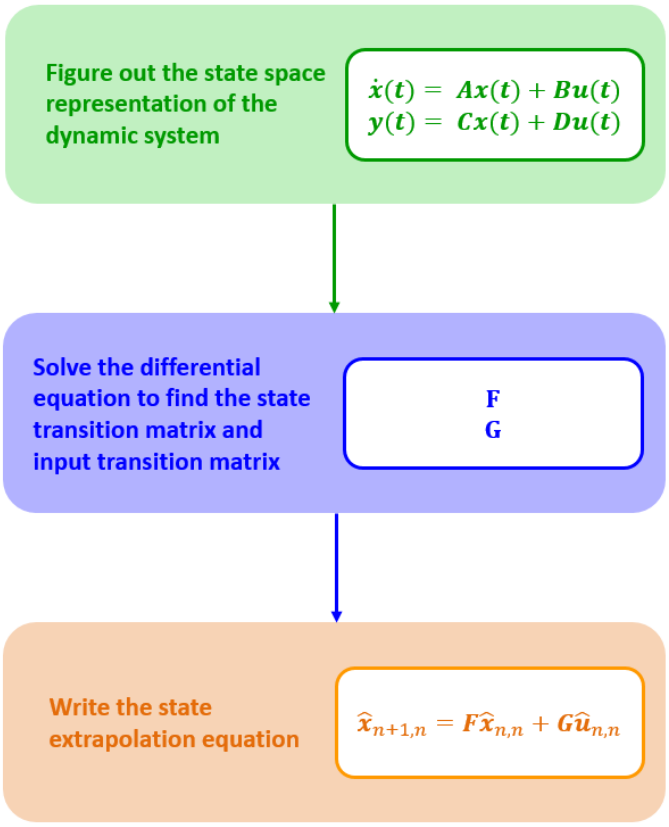


**Figure C.1:** *The process of the state extrapolation equation derivation.*

## C.2 The state space representation

You might be wondering why the state space representation must be in the following form:

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) \\ \boldsymbol{y}(t) &= \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{D}\boldsymbol{u}(t) \end{aligned} \tag{C.3}$$

Many computer software packages that solve differential equations require this representation.

The best way to describe the state space representation is by examples.

### C.2.1 Example - constant velocity moving body

Since there is no external force applied to the body, the system has no inputs:

$$\boldsymbol{u}(t) = 0 \tag{C.4}$$

The state space variable $\boldsymbol{x(t)}$ is the body's displacement $p(t)$ and the speed $v(t)$.

$$\boldsymbol{x(t)} = \begin{bmatrix} p \\ v \end{bmatrix} \tag{C.5}$$

R To avoid confusion between the state vector $\boldsymbol{x(t)}$ (bold-face font) and the body position along the $x$ - axis denoted by $x(t)$ (normal-face font), I've denoted the body position by $p(t)$.

$$\boldsymbol{\dot{x}(t) = Ax(t) + Bu(t)} \tag{C.6}$$

$$\begin{bmatrix} \dot{p} \\ \dot{v} \end{bmatrix} = \boldsymbol{A} \begin{bmatrix} p \\ v \end{bmatrix} + \boldsymbol{B} \times 0 \tag{C.7}$$

$$\begin{bmatrix} \dot{p} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} p \\ v \end{bmatrix} \tag{C.8}$$

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \tag{C.9}$$

We've got the first equation in a differential form.

The dynamic system output $\boldsymbol{y(t)}$ is the body displacement $\boldsymbol{p(t)}$ that is also the state variable.

$$\boldsymbol{y(t) = Cx(t) + Du(t)} \tag{C.10}$$

$$p = \boldsymbol{C} \begin{bmatrix} p \\ v \end{bmatrix} + \boldsymbol{D} \times 0 \tag{C.11}$$

$$p = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} p \\ v \end{bmatrix} \tag{C.12}$$

$$\boldsymbol{C} = \begin{bmatrix} 1 & 0 \end{bmatrix} \tag{C.13}$$

We are done.

## C.2.2 Modeling high-order dynamic systems

Many dynamic systems models are described by high-order differential equations. The order of the differential equation is the number of the highest derivative in a differential equation.

To tackle a high-order equation, we should reduce it to the first-order differential equation by defining new variables and substituting them into the highest-order terms.

### C.2.2.1 The algorithm for reducing the differential equation order

A general $n$-th order linear differential equation can be expressed as a system of $n$ first-order differential equations.

The high-order **governing equation** of the dynamic system looks as follows:

$$a_n \frac{d^n y}{dt^n} + a_{n-1} \frac{d^{n-1} y}{dt^{n-1}} + \ldots + a_2 \frac{d^2 y}{dt^2} + a_1 \frac{d^1 y}{dt^1} + a_0 y = u \tag{C.14}$$

The **governing equation** completely characterizes the dynamic state of the system.

Reducing the equation order

1. Isolate the highest-order derivative:

$$\frac{d^n y}{dt^n} = -\frac{a_0}{a_n} y - \frac{a_1}{a_n} \frac{d^1 y}{dt^1} - \frac{a_2}{a_n} \frac{d^2 y}{dt^2} - \ldots - \frac{a_{n-1}}{a_n} \frac{d^{n-1} y}{dt^{n-1}} + \frac{1}{a_n} u \tag{C.15}$$

   $y$ and its first $n-1$ derivatives are the states of this system.
2. Define new variables: Setting $x_1 = y$, we write:

$$\begin{aligned} x_1(t) &= y \\ x_2(t) &= \tfrac{dy}{dt} \\ x_3(t) &= \tfrac{d^2 y}{dt^2} \\ &\vdots \\ x_n(t) &= \tfrac{d^{n-1} y}{dt^{n-1}} \end{aligned} \tag{C.16}$$

Now, functions $x_i(t)$ are the state variables.

3. Take the derivatives of the state variables:

$$\begin{aligned}
\tfrac{dx_1}{dt} &= x_2\,(t) \\
\tfrac{dx_2}{dt} &= x_3\,(t) \\
&\vdots \\
\tfrac{dx_2}{dt} &= x_3\,(t) \\
\tfrac{dx_n}{dt} &= \tfrac{d^n y}{dt^n}
\end{aligned} \tag{C.17}$$

4. Plug the isolated $\frac{d^n y}{dt^n}$ term (see step 1) into the last equation:

$$\begin{aligned}
\tfrac{dx_1}{dt} &= x_2\,(t) \\
\tfrac{dx_2}{dt} &= x_3\,(t) \\
&\vdots \\
\tfrac{dx_{n-1}}{dt} &= x_n\,(t) \\
\tfrac{dx_n}{dt} &= -\tfrac{a_0}{a_n}x_1 - \tfrac{a_1}{a_n}x_2 - \tfrac{a_2}{a_n}x_3 - \ldots - \tfrac{a_{n-1}}{a_n}x_n + \tfrac{1}{a_n}u
\end{aligned} \tag{C.18}$$

5. Express the resulting system of equations using vector-matrix notation:

$$\begin{bmatrix} \frac{dx_1}{dt} \\ \frac{dx_2}{dt} \\ \vdots \\ \frac{dx_{n-1}}{dt} \\ \frac{dx_n}{dt} \end{bmatrix} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_{n-1} \\ \dot{x}_n \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 1 \\ -\frac{a_0}{a_n} & -\frac{a_1}{a_n} & -\frac{a_2}{a_n} & \cdots & -\frac{a_{n-2}}{a_n} & -\frac{a_{n-1}}{a_n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ \frac{1}{a_n} \end{bmatrix} u \tag{C.19}$$

That's it. We've got the state space equation in the form of the following:

$$\boldsymbol{\dot{x}(t) = Ax(t) + Bu(t)} \tag{C.20}$$

Where:

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 1 \\ -\frac{a_0}{a_n} & -\frac{a_1}{a_n} & -\frac{a_2}{a_n} & \cdots & -\frac{a_{n-2}}{a_n} & -\frac{a_{n-1}}{a_n} \end{bmatrix} \tag{C.21}$$

$$\boldsymbol{B} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ \hline \frac{1}{a_n} \end{bmatrix} \tag{C.22}$$

Let us see two examples.

### C.2.3 Example - constant acceleration moving body

In this example, there is an external force applied to the body.

The governing equation for a moving body with constant acceleration is Newton's second law:

$$m\ddot{p} = F \tag{C.23}$$

Where:

| | |
|---|---|
| $p$ | is the body position displacement |
| $m$ | is the body mass |
| $F$ | is the external force applied to the body |

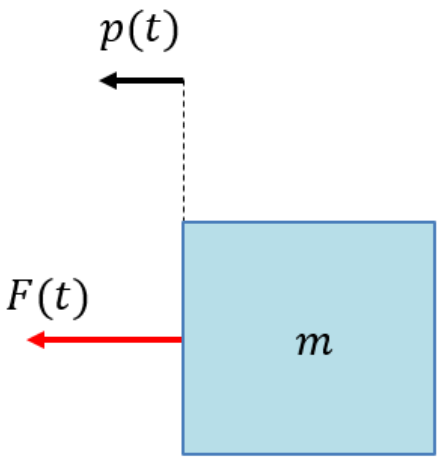


**Figure C.2:** *The constant acceleration model.*

We are going to apply an order reduction algorithm to the governing equation.

1. Isolate the highest-order derivative:

$$\ddot{p} = \frac{1}{m}F \tag{C.24}$$

2. Define new variables $x_1$ and $x_2$:

$$\begin{aligned} x_1 &= p \\ x_2 &= \dot{p} \end{aligned} \tag{C.25}$$

   $x_1$ and $x_2$ are the state variables.
3. Take the derivatives of the state variables:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= \ddot{p} \end{aligned} \tag{C.26}$$

4. Plug in the isolated $\ddot{p}$ term (see step 1) in the last equation:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= \tfrac{F}{m} \end{aligned} \tag{C.27}$$

5. Express the resulting system of equations using vector-matrix notation:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} F \tag{C.28}$$

Remember that $x_2 = \dot{p}$. It would be more meaningful to denote $x_2$ by $v$, since $x_2$ is the body velocity.

We can rewrite the above equation as follows:

$$\begin{bmatrix} \dot{p} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} p \\ v \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} F \tag{C.29}$$

Or:

$$\begin{bmatrix} \dot{p} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} p \\ v \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} a \tag{C.30}$$

Where $a$ is the body acceleration resulting from the applied force $F$.

We've got an equation in the form:

$$\dot{\boldsymbol{x}}(\boldsymbol{t}) = \boldsymbol{A}\boldsymbol{x}(\boldsymbol{t}) + \boldsymbol{B}\boldsymbol{u}(\boldsymbol{t}) \tag{C.31}$$

The state-space variable $\boldsymbol{x(t)}$ is the body's displacement $p(t)$ and the speed $v(t)$.

The dynamic system output $\boldsymbol{y(t)}$ is the body displacement $p(t)$ which is also an element of the state variable.

$$\boldsymbol{y(t)} = \boldsymbol{C}\boldsymbol{x(t)} + \boldsymbol{D}\boldsymbol{u(t)} \tag{C.32}$$

$$p = \boldsymbol{C}\begin{bmatrix} p \\ v \end{bmatrix} + \boldsymbol{D}a \tag{C.33}$$

$$p = \begin{bmatrix} 1 & 0 \end{bmatrix}\begin{bmatrix} p \\ v \end{bmatrix} + \begin{bmatrix} 0 \end{bmatrix} a \tag{C.34}$$

### C.2.4 Example - mass-spring-damper system

The mass-spring-damper system includes three basic elements:

- mass - inertia element
- spring - elastic element
- damper - frictional element

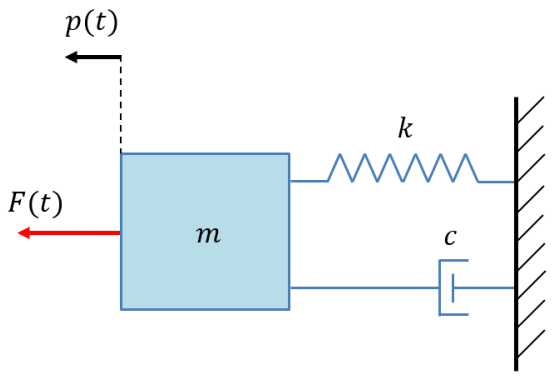


**Figure C.3:** *Mass-spring-damper model.*

Each of the elements has one of two possible energy behaviors:

- stores all the energy supplied to it
- dissipates all energy into heat by the "frictional" effect

The mass stores energy as kinetic energy.

When the spring is compressed from its original length, it stores the energy as potential energy.

The damper dissipates energy as a heat.

These three components: mass, spring, and damper, can model any dynamic response situation in a general sense.

The force diagram for this system is shown below.

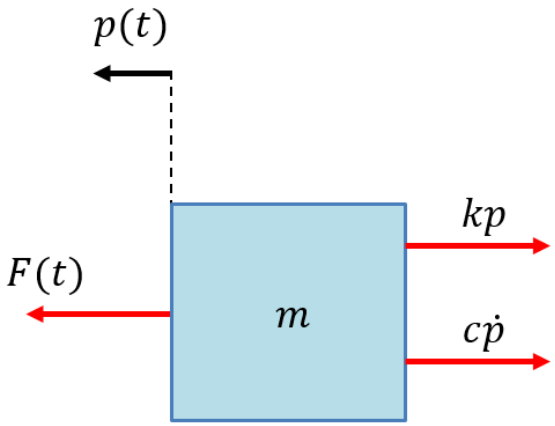


**Figure C.4:** *Mass-spring-damper forces.*

The spring force is proportional to the position displacement $p$ of the mass.

The viscous damping force is proportional to the velocity $\dot{p}$ of the mass.

Newton's second law states:

$$\sum F = ma = m\frac{d^2p}{dt^2} = m\ddot{p} \tag{C.35}$$

We proceed by summing the forces and applying Newton's second law:

$$\sum F = F(t) - c\dot{p} - kp = m\ddot{p} \tag{C.36}$$

Where:

$p$ is the body position displacement
$m$ is the body mass
$F$ is the external force applied to the body
$k$ is the spring constant
$c$ is the damping coefficient

This equation is a governing equation that completely characterizes the dynamic state of the system.

We are going to apply an order reduction algorithm to the governing equation.

1. Isolate the highest-order derivative:

$$\ddot{p} = -\frac{k}{m}p - \frac{c}{m}\dot{p} + \frac{1}{m}F \tag{C.37}$$

2. Define new variables $x_1$ and $x_2$:

$$\begin{aligned} x_1 &= p \\ x_2 &= \dot{p} \end{aligned} \tag{C.38}$$

   $x_1$ and $x_2$ are the state variables
3. Take the derivatives of the state variables:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= \ddot{p} \end{aligned} \tag{C.39}$$

4. Plug in the isolated $\ddot{p}$ term (see step 1) in the last equation:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -\tfrac{k}{m}p - \tfrac{c}{m}\dot{p} + \tfrac{1}{m}F \end{aligned} \tag{C.40}$$

5. Express the resulting system of equations using vector-matrix notation:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} F \tag{C.41}$$

Remember that: $x_2 = \dot{p}$, It would be more meaningful to denote $x_2$ by $v$, since $x_2$ is the body velocity.

We can rewrite the above equation as follows:

$$\begin{bmatrix} \dot{p} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix} \begin{bmatrix} p \\ v \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} F \tag{C.42}$$

We've got an equation in the form of the following:

$$\boldsymbol{\dot{x}(t) = Ax(t) + Bu(t)} \tag{C.43}$$

The state-space variable $\boldsymbol{x(t)}$ is the body's displacement $p(t)$ and the speed $v(t)$.

The dynamic system output $\boldsymbol{y(t)}$ is the body displacement $p(t)$ which is also an

element of the state variable.

$$\boldsymbol{y(t)} = \boldsymbol{Cx(t)} + \boldsymbol{Du(t)} \tag{C.44}$$

$$p = \boldsymbol{C}\begin{bmatrix} p \\ v \end{bmatrix} + \boldsymbol{D}\frac{F}{m} \tag{C.45}$$

### C.2.5 More examples

You can find many beautiful examples of state-space modeling on the ShareTechnote site.

## C.3 Solving the differential equation

Remember, for our Kalman Filter model, we need to determine the state extrapolation equation in the form of:

$$\hat{\boldsymbol{x}}_{n+1,n} = \boldsymbol{F}\hat{\boldsymbol{x}}_{n,n} + \boldsymbol{G}\hat{\boldsymbol{u}}_{n,n} \tag{C.46}$$

To get there, we shall solve the differential equation that describes the state space representation. We can use computer software to solve the differential equation or do it ourselves. Let us see how to solve the differential equation.

### C.3.1 Dynamic systems without input variable

LTI dynamic system without external input can be described by the first-order differential equation:

$$\dot{\boldsymbol{x}} = \boldsymbol{Ax} \tag{C.47}$$

Where $\boldsymbol{A}$ is a **system dynamics matrix**.

Our goal is to find the **state transition matrix** $\boldsymbol{F}$.

We need to solve the differential equation to find $\boldsymbol{F}$.

In a single dimension, the differential equation looks as follows:

$$\begin{aligned} \frac{dx}{dt} &= kx \\ \frac{dx}{x} &= kdt \end{aligned} \tag{C.48}$$

Integrating both sides results in the following:

$$\int_{x_0}^{x_1} \frac{1}{x} dx = \int_0^{\Delta t} k dt \tag{C.49}$$

Solving the integrals:

$$\begin{aligned} &ln\left(x_1\right) - ln\left(x_0\right) = k\Delta t \\ &ln\left(x_1\right) = ln\left(x_0\right) + k\Delta t \\ &x_1 = e^{ln(x_0)+k\Delta t} \\ &x_1 = e^{ln(x_0)} e^{k\Delta t} \\ &x_1 = x_0 e^{k\Delta t} \end{aligned} \tag{C.50}$$

Similarly, in the multidimensional case, for:

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} \tag{C.51}$$

The solution is:

$$x_{n+1} = x_n e^{\boldsymbol{A}\Delta t} \tag{C.52}$$

We've found the state transition matrix $\boldsymbol{F}$:

$$\boldsymbol{F} = e^{\boldsymbol{A}\Delta t} \tag{C.53}$$

$e^{\boldsymbol{A}t}$ is a matrix exponential.

The matrix exponential can be computed by Taylor series expansion:

$$e^{\boldsymbol{X}} = \sum_{k=0}^{\infty} \frac{1}{k!} \boldsymbol{X}^k \tag{C.54}$$

Therefore:

$$\boldsymbol{F} = e^{\boldsymbol{A}\Delta t} = \boldsymbol{I} + \boldsymbol{A}\Delta t + \frac{(\boldsymbol{A}\Delta t)^2}{2!} + \frac{(\boldsymbol{A}\Delta t)^3}{3!} + \frac{(\boldsymbol{A}\Delta t)^4}{4!} + \dots \tag{C.55}$$

Where $\boldsymbol{I}$ is an identity matrix.

### C.3.1.1 Example continued - constant velocity moving body

Now we can find the state transition matrix $\boldsymbol{F}$ for the constant velocity motion equations.

The following set of differential equations can describe the constant velocity dynamic model.

$$\begin{cases} \frac{dp}{dt} = v \\ \frac{dv}{dt} = 0 \end{cases} \tag{C.56}$$

In a matrix form:

$$\begin{bmatrix} \dot{p} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} p \\ v \end{bmatrix} \tag{C.57}$$

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \tag{C.58}$$

Let's calculate $\boldsymbol{A}^2$

$$\boldsymbol{A}^2 = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{C.59}$$

Since $\boldsymbol{A}^2 = 0$, higher powers of $\boldsymbol{A}$ are also equal to 0.

Now, we can find the state transition matrix $\boldsymbol{F}$ for the constant velocity model:

$$F = e^{\boldsymbol{A}\Delta t} = \boldsymbol{I} + \boldsymbol{A}\Delta t = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \Delta t = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \tag{C.60}$$

$$\boldsymbol{x}_{n+1} = \boldsymbol{F}\boldsymbol{x}_n = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \boldsymbol{x}_n \tag{C.61}$$

$$\begin{bmatrix} x_{n+1} \\ \dot{x}_{n+1} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_n \\ \dot{x}_n \end{bmatrix} \tag{C.62}$$

### C.3.2 Dynamic systems with an input variable

For zero-order hold sampling, assuming the input is piecewise constant, the general solution of the state space equation in the form of:

$$\dot{\boldsymbol{x}}(\boldsymbol{t}) = \boldsymbol{A}\boldsymbol{x}(\boldsymbol{t}) + \boldsymbol{B}\boldsymbol{u}(\boldsymbol{t}) \tag{C.63}$$

is given by:

$$\boldsymbol{x}(\boldsymbol{t}+\boldsymbol{\Delta t}) = \underbrace{e^{\boldsymbol{A}\Delta t}}_{\text{F}}\boldsymbol{x}(\boldsymbol{t}) + \underbrace{\int_0^{\Delta t} e^{\boldsymbol{A}t}dt\boldsymbol{B}}_{\text{G}}\boldsymbol{u}(\boldsymbol{t}) \tag{C.64}$$

If we remove the input variable ($\boldsymbol{u}(\boldsymbol{t}) = 0$), we get the solution derived in the previous chapter.

I am not going to prove this - you can find the proof in [19], or any other calculus textbook.

Now let's solve the state space equations for the constant acceleration moving body and the mass-spring-damper system examples.

#### C.3.2.1 Example continued - constant acceleration moving body

Recall that the state-space representation of the constant acceleration moving body is:

$$\begin{bmatrix}\dot{p}\\ \dot{v}\end{bmatrix} = \begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}\begin{bmatrix}p\\ v\end{bmatrix} + \begin{bmatrix}0\\ 1\end{bmatrix}a \tag{C.65}$$

Where:

$$\boldsymbol{A} = \begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix} \tag{C.66}$$

$$\boldsymbol{B} = \begin{bmatrix}0\\ 1\end{bmatrix} \tag{C.67}$$

Let's solve the equation:

$$\boldsymbol{x}(\boldsymbol{t}+\boldsymbol{\Delta t}) = \underbrace{e^{\boldsymbol{A}\Delta t}}_{\text{F}}\boldsymbol{x}(\boldsymbol{t}) + \underbrace{\int_0^{\Delta t} e^{\boldsymbol{A}t}dt\boldsymbol{B}}_{\text{G}}\boldsymbol{u}(\boldsymbol{t}) \tag{C.68}$$

Finding F

$$\boldsymbol{F} = e^{\boldsymbol{A}\Delta t} \tag{C.69}$$

We've already solved this for $A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$

$$\boldsymbol{F} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \tag{C.70}$$

Finding G

$$\boldsymbol{G} = \int_0^{\Delta t} e^{\boldsymbol{A}t} dt \boldsymbol{B} \tag{C.71}$$

The general formula for $\int_0^{\Delta t} e^{\boldsymbol{A}t} dt$ is the power series:

$$\int_0^{\Delta t} e^{\boldsymbol{A}t} dt = \Delta t \left( I + \frac{\boldsymbol{A}\Delta t}{2!} + \frac{(\boldsymbol{A}\Delta t)^2}{3!} + \frac{(\boldsymbol{A}\Delta t)^3}{4!} + \dots \right) \tag{C.72}$$

$$\boldsymbol{A}^2 = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{C.73}$$

The higher powers of $\boldsymbol{A}$ are also equal to 0.

$$\int_0^{\Delta t} e^{\boldsymbol{A}t} dt = \Delta t \left( \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \frac{\Delta t}{2} \right) = \begin{bmatrix} \Delta t & \frac{1}{2}\Delta t^2 \\ 0 & \Delta t \end{bmatrix} \tag{C.74}$$

$$\boldsymbol{G} = \int_0^{\Delta t} e^{\boldsymbol{A}t} dt \boldsymbol{B} = \begin{bmatrix} \Delta t & \frac{1}{2}\Delta t^2 \\ 0 & \Delta t \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\Delta t^2 \\ \Delta t \end{bmatrix} \tag{C.75}$$

Now, we can write the state extrapolation equation:

$$\begin{bmatrix} p_{n+1} \\ \dot{p}_{n+1} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p_n \\ \dot{p}_n \end{bmatrix} + \begin{bmatrix} \frac{1}{2}\Delta t^2 \\ \Delta t \end{bmatrix} a \tag{C.76}$$

### C.3.2.2 Example continued - mass-spring-damper system

Recall that the state-space representation of the mass-spring-damper system is:

$$\begin{bmatrix} \dot{p} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & \frac{c}{m} \end{bmatrix} \begin{bmatrix} p \\ v \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} F \tag{C.77}$$

Where:

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & \frac{c}{m} \end{bmatrix} \tag{C.78}$$

$$\boldsymbol{B} = \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} \tag{C.79}$$

In this example, computation of the matrix exponential is not so easy since the high powers of A are not zero. The Solution of this differential equation is beyond the scope of this book.

# D. Derivative of matrix product trace

In this appendix, I prove two statements:

| | | |
|---|---|---|
| **(1)** | $\frac{d}{d\boldsymbol{A}}\left(tr\left(\boldsymbol{AB}\right)\right)=\boldsymbol{B}^T$ | |
| **(2)** | $\frac{d}{d\boldsymbol{A}}\left(tr\left(\boldsymbol{ABA}^T\right)\right)=2\boldsymbol{AB}$ | (for symmetric $\boldsymbol{B}$ ) |

**Table D.1:** *Statements.*

## D.1 Statement 1

Given two matrices $\boldsymbol{A}(m\times n)$ and $\boldsymbol{B}(n\times m)$. The product of two matrices:

$$\boldsymbol{AB}=\begin{bmatrix}a_{11}&\cdots&a_{1n}\\ \vdots&\ddots&\vdots\\ a_{m1}&\cdots&a_{mn}\end{bmatrix}\begin{bmatrix}b_{11}&\cdots&b_{1m}\\ \vdots&\ddots&\vdots\\ b_{n1}&\cdots&b_{nm}\end{bmatrix}=\begin{bmatrix}\sum_{i=1}^{n}a_{1i}b_{i1}&\cdots&\sum_{i=1}^{n}a_{1i}b_{im}\\ \vdots&\ddots&\vdots\\ \sum_{i=1}^{n}a_{mi}b_{i1}&\cdots&\sum_{i=1}^{n}a_{mi}b_{im}\end{bmatrix} \tag{D.1}$$

The trace of $\boldsymbol{AB}$ is the sum of the main diagonal:

$$tr(\boldsymbol{AB})=\sum_{i=1}^{n}a_{1i}b_{i1}+\cdots+\sum_{i=1}^{n}a_{mi}b_{im}=\sum_{i=1}^{n}\sum_{j=1}^{m}a_{ji}b_{ij} \tag{D.2}$$

Differentiate using the function of gradient:

$$\frac{\partial f(\boldsymbol{X})}{\partial \boldsymbol{X}}=\begin{bmatrix}\frac{\partial f(\boldsymbol{X})}{\partial x_{11}}&\cdots&\frac{\partial f(\boldsymbol{X})}{\partial x_{1n}}\\ \vdots&\ddots&\vdots\\ \frac{\partial f(\boldsymbol{X})}{\partial x_{m1}}&\cdots&\frac{\partial f(\boldsymbol{X})}{\partial x_{mn}}\end{bmatrix} \tag{D.3}$$

$$\frac{\partial tr(\boldsymbol{AB})}{\partial \boldsymbol{A}}=\begin{bmatrix}\frac{\partial(\sum_{i=1}^{n}\sum_{j=1}^{m}a_{ji}b_{ij})}{\partial a_{11}}&\cdots&\frac{\partial(\sum_{i=1}^{n}\sum_{j=1}^{m}a_{ji}b_{ij})}{\partial a_{1n}}\\ \vdots&\ddots&\vdots\\ \frac{\partial(\sum_{i=1}^{n}\sum_{j=1}^{m}a_{ji}b_{ij})}{\partial a_{m1}}&\cdots&\frac{\partial(\sum_{i=1}^{n}\sum_{j=1}^{m}a_{ji}b_{ij})}{\partial a_{mn}}\end{bmatrix}=\begin{bmatrix}b_{11}&\cdots&b_{n1}\\ \vdots&\ddots&\vdots\\ b_{1m}&\cdots&b_{nm}\end{bmatrix}=\boldsymbol{B}^T \tag{D.4}$$

## D.2 Statement 2

Given two matrices $\boldsymbol{A}(m \times n)$ and $\boldsymbol{B}(n \times n)$.

$$\boldsymbol{ABA}^T = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} b_{11} & \cdots & b_{1n} \\ \vdots & \ddots & \vdots \\ b_{n1} & \cdots & b_{nn} \end{bmatrix} \begin{bmatrix} a_{11} & \cdots & a_{m1} \\ \vdots & \ddots & \vdots \\ a_{1n} & \cdots & a_{mn} \end{bmatrix}$$

$$= \begin{bmatrix} \sum_{i=1}^{n} a_{1i}b_{i1} & \cdots & \sum_{i=1}^{n} a_{1i}b_{in} \\ \vdots & \ddots & \vdots \\ \sum_{i=1}^{n} a_{mi}b_{i1} & \cdots & \sum_{i=1}^{n} a_{mi}b_{in} \end{bmatrix} \begin{bmatrix} a_{11} & \cdots & a_{m1} \\ \vdots & \ddots & \vdots \\ a_{1n} & \cdots & a_{mn} \end{bmatrix} \tag{D.5}$$

$$= \begin{bmatrix} \sum_{j=1}^{n}\sum_{i=1}^{n} a_{1i}b_{ij}a_{1j} & \cdots & \sum_{j=1}^{n}\sum_{i=1}^{n} a_{1i}b_{ij}a_{mj} \\ \vdots & \ddots & \vdots \\ \sum_{j=1}^{n}\sum_{i=1}^{n} a_{mi}b_{ij}a_{1j} & \cdots & \sum_{j=1}^{n}\sum_{i=1}^{n} a_{mi}b_{ij}a_{mj} \end{bmatrix}$$

The trace of $\boldsymbol{ABA}^T$ is the sum of the main diagonal:

$$tr(\boldsymbol{ABA}^T) = \sum_{j=1}^{n}\sum_{i=1}^{n} a_{1i}b_{ij}a_{1j} + \cdots + \sum_{j=1}^{n}\sum_{i=1}^{n} a_{mi}b_{ij}a_{mj} = \sum_{k=1}^{m}\sum_{j=1}^{n}\sum_{i=1}^{n} a_{ki}b_{ij}a_{kj} \tag{D.6}$$

$$\frac{\partial tr(\boldsymbol{ABA}^T)}{\partial \boldsymbol{A}} = \begin{bmatrix} \frac{\partial(\sum_{k=1}^n \sum_{j=1}^n \sum_{i=1}^n a_{ki} b_{ij} a_{kj})}{\partial a_{11}} & \cdots & \frac{\partial(\sum_{k=1}^n \sum_{j=1}^n \sum_{i=1}^n a_{ki} b_{ij} a_{kj})}{\partial a_{1n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial(\sum_{k=1}^n \sum_{j=1}^n \sum_{i=1}^n a_{ki} b_{ij} a_{kj})}{\partial a_{m1}} & \cdots & \frac{\partial(\sum_{k=1}^n \sum_{j=1}^n \sum_{i=1}^n a_{ki} b_{ij} a_{kj})}{\partial a_{mn}} \end{bmatrix}$$

$$= \begin{bmatrix} \sum_{j=1}^n b_{1j} a_{1j} + \sum_{i=1}^n a_{1i} b_{i1} & \cdots & \sum_{j=1}^n b_{nj} a_{1j} + \sum_{i=1}^n a_{1i} b_{in} \\ \vdots & \ddots & \vdots \\ \sum_{j=1}^n b_{1j} a_{mj} + \sum_{i=1}^n a_{mi} b_{i1} & \cdots & \sum_{j=1}^n b_{nj} a_{mj} + \sum_{i=1}^n a_{mi} b_{in} \end{bmatrix}$$

$$= \begin{bmatrix} \sum_{j=1}^n a_{1j} b_{1j} & \cdots & \sum_{j=1}^n a_{1j} b_{nj} \\ \vdots & \ddots & \vdots \\ \sum_{j=1}^n a_{mj} b_{1j} & \cdots & \sum_{j=1}^n a_{mj} b_{nj} \end{bmatrix} + \begin{bmatrix} \sum_{i=1}^n a_{1i} b_{i1} & \cdots & \sum_{i=1}^n a_{1i} b_{in} \\ \vdots & \ddots & \vdots \\ \sum_{i=1}^n a_{mi} b_{i1} & \cdots & \sum_{i=1}^n a_{mi} b_{in} \end{bmatrix} \tag{D.7}$$

$$= \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} b_{11} & \cdots & b_{n1} \\ \vdots & \ddots & \vdots \\ b_{1n} & \cdots & b_{nn} \end{bmatrix} + \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} b_{11} & \cdots & b_{1n} \\ \vdots & \ddots & \vdots \\ b_{n1} & \cdots & b_{nn} \end{bmatrix}$$

$$= \boldsymbol{AB}^T + \boldsymbol{AB}$$

If $\boldsymbol{B}$ is symmetric, $\boldsymbol{B} = \boldsymbol{B}^T$:

$$\frac{\partial tr(\boldsymbol{ABA}^T)}{\partial \boldsymbol{A}} = \boldsymbol{AB}^T + \boldsymbol{AB} = \boldsymbol{AB} + \boldsymbol{AB} = 2\boldsymbol{AB} \tag{D.8}$$

# E. Pendulum motion simulation

In this appendix we derive equations for pendulum motion simulation. Using these equations, we create the ground truth for examples 12 and 14.

In section 12.3, we derived the differential equation that describes the pendulum movement:

$$L\frac{d^2\theta}{dt^2} = -g\sin(\theta) \tag{E.1}$$

Where:

- $\theta$ is the pendulum's angle
- $L$ is the pendulum's string length
- $g$ is the gravitational acceleration constant
- $t$ is time

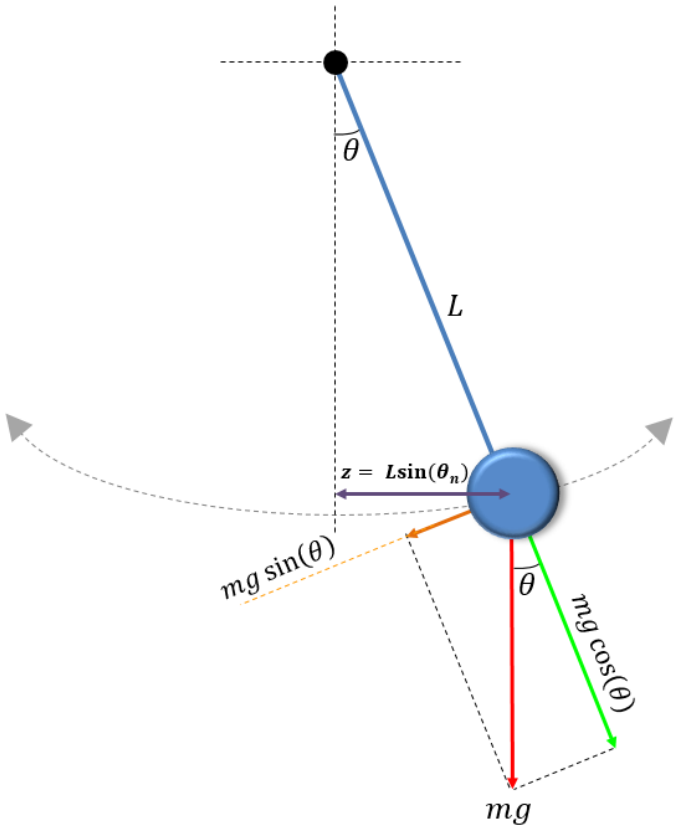


**Figure E.1:** *Pendulum motion.*

We re-write the equation as:

$$\ddot{\theta} + \frac{g}{L} sin\theta = 0 \tag{E.2}$$

This is a nonlinear equation, and we cannot solve it analytically.

Approximation: if $\theta$ is small, then $sin\theta \approx \theta$, and in this situation, we have an approximate equation given by:

$$\ddot{\theta} + \frac{g}{L}\theta = 0 \tag{E.3}$$

The method we shall employ for solving this differential equation is called the method of inspired guessing. Since the sine and cosine functions are periodic, we propose the following solution for the angle $\theta$ as a function of the time $t$:

$$\theta(t) = Acos(\omega t) + Bsin(\omega t) \tag{E.4}$$

At $t = 0$:

$$\theta(t = 0) = A \tag{E.5}$$

So, A is an initial angle, denoted by $\theta_0$.

Re-write Equation E.4:

$$\theta(t) = \theta_0 cos(\omega t) + Bsin(\omega t) \tag{E.6}$$

The derivative of $\theta(t)$ is the angular velocity $(\omega)$ of the pendulum:

$$\omega(t) = \frac{d\theta}{dt} = \dot{\theta} = -\omega\theta_0 sin(\omega t) + \omega B cos(\omega t) \tag{E.7}$$

To find $B$, evaluate $\omega(t)$ at $t = 0$:

$$\omega(t = 0) = \omega B \tag{E.8}$$

$$B = \frac{\omega_0}{\omega} \tag{E.9}$$

Thus, our proposed solution now has the following form:

$$\theta(t) = \theta_0 cos(\omega t) + \frac{\omega_0}{\omega} sin(\omega t) \tag{E.10}$$

Let us prove that Equation E.10 is a solution to Equation E.7.

$$\dot{\theta} = -\omega\theta_0 sin(\omega t) + \omega_0 cos(\omega t) \tag{E.11}$$

$$\ddot{\theta} = -\omega^2\theta_0 cos(\omega t) - \omega\omega_0 sin(\omega t) \tag{E.12}$$

$$\ddot{\theta} = -\omega^2 \left(\theta_0 cos(\omega t) - \frac{\omega_0}{\omega} sin(\omega t)\right) \tag{E.13}$$

$$\ddot{\theta} = -\omega^2\theta \tag{E.14}$$

Comparing with Equation E.3, we conclude that:

$$\omega^2 = \frac{g}{L} \tag{E.15}$$

$$\omega = \sqrt{\frac{g}{L}} \tag{E.16}$$

Now, we can simulate the pendulum movement for small $\theta$, using an approximation $sin\theta \approx \theta$.

# F. Statistical Linear Regression

**Note**: In this appendix, for more convenient notation, I am using a counter variable i in a subscript and not in a superscript.

Consider a nonlinear function $\boldsymbol{y} = \boldsymbol{g}(\boldsymbol{x})$ evaluated in $r$ points $(\mathcal{X}_i, \mathcal{Y}_i)$ , where $\mathcal{Y}_i = \boldsymbol{g}(\mathcal{X}_i)$. Define:

| **Equation** | **Notes** |
|---|---|
| $\overline{\boldsymbol{x}} = \frac{1}{r}\sum_{i=1}^{r} \mathcal{X}_i$ | Mean of $\mathcal{X}_i$ |
| $\overline{\boldsymbol{y}} = \frac{1}{r}\sum_{i=1}^{r} \mathcal{Y}_i$ | Mean of $\mathcal{Y}_i$ |
| $\boldsymbol{P}_{xx} = \frac{1}{r}\sum_{i=1}^{r} (\mathcal{X}_i - \overline{\boldsymbol{x}})(\mathcal{X}_i - \overline{\boldsymbol{x}})^T$ | Variance of $\mathcal{X}_i$ |
| $\boldsymbol{P}_{yy} = \frac{1}{r}\sum_{i=1}^{r} (\mathcal{Y}_i - \overline{\boldsymbol{y}})(\mathcal{Y}_i - \overline{\boldsymbol{y}})^T$ | Variance of $\mathcal{Y}_i$ |
| $\boldsymbol{P}_{xy} = \frac{1}{r}\sum_{i=1}^{r} (\mathcal{X}_i - \overline{\boldsymbol{x}})(\mathcal{Y}_i - \overline{\boldsymbol{y}})^T$ | Cross-variance of $\mathcal{X}_i$ and $\mathcal{Y}_i$ |
| $\boldsymbol{P}_{yx} = \frac{1}{r}\sum_{i=1}^{r} (\mathcal{Y}_i - \overline{\boldsymbol{y}})(\mathcal{X}_i - \overline{\boldsymbol{x}})^T$ | Cross-variance of $\mathcal{Y}_i$ and $\mathcal{X}_i$ |

**Table F.1:** *Definitions.*

We want to approximate the nonlinear function $\boldsymbol{y} = \boldsymbol{g}(\boldsymbol{x})$ by a linear function $\boldsymbol{y} = \boldsymbol{M}(\boldsymbol{x}) + \boldsymbol{b}$.

The linear approximation produces linearization error. For each point $\mathcal{X}_i$, the linearization error is given by:

$$\boldsymbol{e}_i = \mathcal{Y}_i - (\boldsymbol{M}\mathcal{X}_i + \boldsymbol{b}) \tag{F.1}$$

To minimize the linearization error, we should find $\boldsymbol{M}$ and $\boldsymbol{b}$ that minimize the sum of squared errors for all $\mathcal{X}_i$ points:

$$\min_{M,b} \sum_{i=1}^{r} \{\boldsymbol{e}_i^T \boldsymbol{e}_i\} \tag{F.2}$$

First, let us expand the above equation:

| Equation | Notes |
|---|---|
| $\sum_{i=1}^{r} \{e_i^T e_i\}$ | The sum of squared errors |
| $= \sum_{i=1}^{r} \left\{ (\mathcal{Y}_i - (M\mathcal{X}_i + b))^T (\mathcal{Y}_i - (M\mathcal{X}_i + b)) \right\}$ | Plug $e_i = \mathcal{Y}_i - (M\mathcal{X}_i + b)$ |
| $= \sum_{i=1}^{r} \Big\{ \mathcal{Y}_i^T \mathcal{Y}_i - \mathcal{Y}_i^T M\mathcal{X}_i - \mathcal{Y}_i^T b - (M\mathcal{X}_i)^T \mathcal{Y}_i$ $+ (M\mathcal{X}_i)^T M\mathcal{X}_i + (M\mathcal{X}_i)^T b - b^T \mathcal{Y}_i$ $+ b^T M\mathcal{X}_i + b^T b \Big\}$ | Expand |
| $= \sum_{i=1}^{r} \Big\{ \mathcal{Y}_i^T \mathcal{Y}_i - \mathcal{Y}_i^T M\mathcal{X}_i - \mathcal{Y}_i^T b - \left(\mathcal{Y}_i^T M\mathcal{X}_i\right)^T$ $+ \mathcal{X}_i^T M^T M\mathcal{X}_i + (M\mathcal{X}_i)^T b - \left(\mathcal{Y}_i^T b\right)^T$ $+ \left((M\mathcal{X}_i)^T b\right)^T + b^T b \Big\}$ | Apply the matrix transpose property: $(BA)^T = A^T B^T$ |
| $= \sum_{i=1}^{r} \Big\{ \mathcal{Y}_i^T \mathcal{Y}_i - 2\mathcal{Y}_i^T M\mathcal{X}_i - 2\mathcal{Y}_i^T b$ $+ \mathcal{X}_i^T M^T M\mathcal{X}_i + 2\,(M\mathcal{X}_i)^T b + b^T b \Big\}$ | $\mathcal{Y}_i^T M\mathcal{X}_i$ is a scalar (row vector $\times$ matrix $\times$ column vector): $\mathcal{Y}_i^T M\mathcal{X}_i = \left(\mathcal{Y}_i^T M\mathcal{X}_i\right)^T$ Similarly: $(M\mathcal{X}_i)^T b = \left((M\mathcal{X}_i)^T b\right)^T$ $\mathcal{Y}_i^T b = \left(\mathcal{Y}_i^T b\right)^T$ |

**Table F.2:** *Expand the error equation.*

To minimize, differentiate and equalize to 0.

First, find an optimal $b$ parameter.

| Equation | Notes |
|---|---|
| $\frac{\partial}{\partial \boldsymbol{b}}\Big(\sum_{i=1}^{r}\Big\{\mathcal{Y}_i^T\mathcal{Y}_i - 2\mathcal{Y}_i^T\boldsymbol{M}\mathcal{X}_i - 2\mathcal{Y}_i^T\boldsymbol{b}+$ $+\mathcal{X}_i^T\boldsymbol{M}^T\boldsymbol{M}\mathcal{X}_i + 2\left(\boldsymbol{M}\mathcal{X}_i\right)^T\boldsymbol{b} + \boldsymbol{b}^T\boldsymbol{b}\Big\}\Big) = 0$ | |
| $\sum_{i=1}^{r}\Big\{ - 2\mathcal{Y}_i^T + 2\left(\boldsymbol{M}\mathcal{X}_i\right)^T + 2\boldsymbol{b}^T\Big\} = 0$ | Differentiate |
| $\sum_{i=1}^{r}\mathcal{Y}_i = \boldsymbol{M}\sum_{i=1}^{r}\mathcal{X}_i + \sum_{i=1}^{r}\boldsymbol{b}$ | Simplify |
| $r\overline{\boldsymbol{y}} = r\boldsymbol{M}\overline{\boldsymbol{x}} + r\boldsymbol{b}$ | Compute the sum |
| $\boldsymbol{b} = \overline{\boldsymbol{y}} - \boldsymbol{M}\overline{\boldsymbol{x}}$ | Simplify |

**Table F.3:** *Finding an optimal* $\boldsymbol{b}$.

Find an optimal $\boldsymbol{M}$ parameter.

| Equation | Notes |
|---|---|
| $\frac{\partial}{\partial \boldsymbol{b}}\Big(\sum_{i=1}^{r}\Big\{\mathcal{Y}_i^T\mathcal{Y}_i - 2\mathcal{Y}_i^T\boldsymbol{M}\mathcal{X}_i - 2\mathcal{Y}_i^T\boldsymbol{b}+$ $+\mathcal{X}_i^T\boldsymbol{M}^T\boldsymbol{M}\mathcal{X}_i + 2\left(\boldsymbol{M}\mathcal{X}_i\right)^T\boldsymbol{b} + \boldsymbol{b}^T\boldsymbol{b}\Big\}\Big) = 0$ | |
| $\sum_{i=1}^{r}\Big\{ - 2\mathcal{Y}_i\mathcal{X}_i^T + 2\boldsymbol{M}\mathcal{X}_i\mathcal{X}_i^T + 2\boldsymbol{b}\mathcal{X}_i^T\Big\} = 0$ | Differentiate |
| $\sum_{i=1}^{r}\Big\{ - \mathcal{Y}_i\mathcal{X}_i^T + \boldsymbol{M}\mathcal{X}_i\mathcal{X}_i^T + \left(\overline{\boldsymbol{y}} - \boldsymbol{M}\overline{\boldsymbol{x}}\right)\mathcal{X}_i^T\Big\} = 0$ | Plug $\boldsymbol{b}$ |
| $\sum_{i=1}^{r}\Big\{ - \mathcal{Y}_i\mathcal{X}_i^T + \boldsymbol{M}\mathcal{X}_i\mathcal{X}_i^T + \overline{\boldsymbol{y}}\mathcal{X}_i^T - \boldsymbol{M}\overline{\boldsymbol{x}}\mathcal{X}_i^T\Big\} = 0$ | Expand |
| $\sum_{i=1}^{r}\Big\{\mathcal{Y}_i\mathcal{X}_i^T - \overline{\boldsymbol{y}}\mathcal{X}_i^T\Big\} = \boldsymbol{M}\sum_{i=1}^{r}\Big\{\mathcal{X}_i\mathcal{X}_i^T - \overline{\boldsymbol{x}}\mathcal{X}_i^T\Big\}$ | Reorder |
| $\boldsymbol{M} = \frac{\sum_{i=1}^{r}\Big\{\mathcal{Y}_i\mathcal{X}_i^T - \overline{\boldsymbol{y}}\mathcal{X}_i^T\Big\}}{\sum_{i=1}^{r}\Big\{\mathcal{X}_i\mathcal{X}_i^T - \overline{\boldsymbol{x}}\mathcal{X}_i^T\Big\}}$ | |

**Table F.4:** *Finding an optimal* $\boldsymbol{M}$.

Now we add two zero terms:

$$\sum_{i=1}^{r}\left\{\overline{\boldsymbol{x}}\ \overline{\boldsymbol{x}}^T-\mathcal{X}_i\overline{\boldsymbol{x}}^T\right\}=0$$

$$\sum_{i=1}^{r}\left\{\overline{\boldsymbol{y}}\ \overline{\boldsymbol{x}}^T-\mathcal{Y}_i\overline{\boldsymbol{x}}^T\right\}=0$$

| Equation | Notes |
|---|---|
| $\boldsymbol{M}=\frac{\sum_{i=1}^{r}\left\{\mathcal{Y}_i\mathcal{X}_i^T-\overline{\boldsymbol{y}}\mathcal{X}_i^T\right\}+\sum_{i=1}^{r}\left\{\overline{\boldsymbol{y}}\ \overline{\boldsymbol{x}}^T-\mathcal{Y}_i\overline{\boldsymbol{x}}^T\right\}}{\sum_{i=1}^{r}\left\{\mathcal{X}_i\mathcal{X}_i^T-\overline{\boldsymbol{x}}\mathcal{X}_i^T\right\}+\sum_{i=1}^{r}\left\{\overline{\boldsymbol{x}}\ \overline{\boldsymbol{x}}^T-\mathcal{X}_i\overline{\boldsymbol{x}}^T\right\}}$ | Add zero terms to the numerator and denominator |
| $\boldsymbol{M}=\frac{\sum_{i=1}^{r}\left\{\mathcal{Y}_i\mathcal{X}_i^T-\overline{\boldsymbol{y}}\mathcal{X}_i^T+\overline{\boldsymbol{y}}\ \overline{\boldsymbol{x}}^T-\mathcal{Y}_i\overline{\boldsymbol{x}}^T\right\}}{\sum_{i=1}^{r}\left\{\mathcal{X}_i\mathcal{X}_i^T-\overline{\boldsymbol{x}}\mathcal{X}_i^T+\overline{\boldsymbol{x}}\ \overline{\boldsymbol{x}}^T-\mathcal{X}_i\overline{\boldsymbol{x}}^T\right\}}$ | Reorder |
| $\boldsymbol{M}=\frac{\sum_{i=1}^{r}\left\{(\mathcal{Y}_i-\overline{\boldsymbol{y}})\mathcal{X}_i^T-(\mathcal{Y}_i-\overline{\boldsymbol{y}})\overline{\boldsymbol{x}}^T\right\}}{\sum_{i=1}^{r}\left\{(\mathcal{X}_i-\overline{\boldsymbol{x}})\mathcal{X}_i^T-(\mathcal{X}_i-\overline{\boldsymbol{x}})\overline{\boldsymbol{x}}^T\right\}}$ | |
| $\boldsymbol{M}=\frac{\sum_{i=1}^{r}\left\{(\mathcal{Y}_i-\overline{\boldsymbol{y}})\left(\mathcal{X}_i^T-\overline{\boldsymbol{x}}^T\right)\right\}}{\sum_{i=1}^{r}\left\{(\mathcal{X}_i-\overline{\boldsymbol{x}})\left(\mathcal{X}_i^T-\overline{\boldsymbol{x}}^T\right)\right\}}$ | |
| $\boldsymbol{M}=\frac{\sum_{i=1}^{r}\left\{\left((\mathcal{X}_i-\overline{\boldsymbol{x}})(\mathcal{Y}_i-\overline{\boldsymbol{y}})^T\right)^T\right\}}{\sum_{i=1}^{r}\left\{\left((\mathcal{X}_i-\overline{\boldsymbol{x}})(\mathcal{X}_i-\overline{\boldsymbol{x}})^T\right)^T\right\}}$ | Apply the matrix transpose property: $(\boldsymbol{BA})^T=\boldsymbol{A}^T\boldsymbol{B}^T$ |
| $\boldsymbol{M}=\frac{r\boldsymbol{P}_{xy}^T}{r\boldsymbol{P}_{xx}}$ | |
| $\boldsymbol{M}=\boldsymbol{P}_{xy}^T\boldsymbol{P}_{xx}^{-1}=\boldsymbol{P}_{yx}\boldsymbol{P}_{xx}^{-1}$ | |

**Table F.5:** *Finding an optimal $\boldsymbol{M}$ continued.*

## Summary

The linear approximation of a nonlinear function $\boldsymbol{y}=\boldsymbol{g}(\boldsymbol{x})$ evaluated in $r$ points $(\mathcal{X}_i,\mathcal{Y}_i)$ , where $\mathcal{Y}_i=\boldsymbol{g}\left(\mathcal{X}_i\right)$, described by the following:

$$\boldsymbol{y}=\boldsymbol{M}\boldsymbol{x}+\boldsymbol{b} \tag{F.3}$$

Mimimization of the linearization error yields the following solution:

$$\begin{aligned} \boldsymbol{M} &= \boldsymbol{P}_{xy}^{T}\boldsymbol{P}_{xx}^{-1} = \boldsymbol{P}_{yx}\boldsymbol{P}_{xx}^{-1} \\ \boldsymbol{b} &= \overline{\boldsymbol{y}} - \boldsymbol{M}\overline{\boldsymbol{x}} \end{aligned} \tag{F.4}$$

# G. The product of univariate Gaussian PDFs

## G.1 Product of two univariate Gaussian PDFs

In this section, we derive the product of two univariate (one-dimensional) Gaussian PDFs [16].

Let $p(x)_1$ and $p(x)_2$ be Gaussian PDFs:

$$p(x)_1 = \frac{1}{\sigma_1\sqrt{2\pi}}\, exp\left(-\frac{1}{2}\left(\frac{x-\mu_1}{\sigma_1}\right)^2\right)$$
$$p(x)_2 = \frac{1}{\sigma_2\sqrt{2\pi}}\, exp\left(-\frac{1}{2}\left(\frac{x-\mu_2}{\sigma_2}\right)^2\right) \tag{G.1}$$

Where:

$\mu_1$ is the mean of the first Gaussian PDF
$\mu_2$ is the mean of the second Gaussian PDF
$\sigma_1$ is the standard deviation of the first Gaussian PDF
$\sigma_2$ is the standard deviation of the second Gaussian PDF

The product of the two Gaussian PDFs is:

$$p(x)_{12} = p(x)_1 p(x)_2 = \frac{1}{\sigma_1\sqrt{2\pi}} exp\left(-\frac{1}{2}\left(\frac{x-\mu_1}{\sigma_1}\right)^2\right)\frac{1}{\sigma_2\sqrt{2\pi}} exp\left(-\frac{1}{2}\left(\frac{x-\mu_2}{\sigma_2}\right)^2\right) = \tag{G.2}$$
$$= \frac{1}{2\pi\sigma_1\sigma_2} exp\left(-\frac{1}{2}\frac{(x-\mu_1)^2}{\sigma_1^2} + \frac{(x-\mu_2)^2}{\sigma_2^2}\right)$$

Let us examine the term in the exponent:

$$K = \frac{1}{2}\left(\frac{(x-\mu_1)^2}{\sigma_1^2} + \frac{(x-\mu_2)^2}{\sigma_2^2}\right) \tag{G.3}$$

| Equation | Notes |
|---|---|
| $K = \frac{(x-\mu_1)^2}{2\sigma_1^2} + \frac{(x-\mu_2)^2}{2\sigma_2^2}$ | |
| $K = \frac{\sigma_2^2(x-\mu_1)^2+\sigma_1^2(x-\mu_2)^2}{2\sigma_1^2\sigma_2^2}$ | Common denominator |
| $K = \frac{\left(\sigma_1^2+\sigma_2^2\right)x^2-2\left(\mu_1\sigma_2^2+\mu_2\sigma_1^2\right)x+\mu_1^2\sigma_2^2+\mu_2^2\sigma_1^2}{2\sigma_1^2\sigma_2^2}$ | Expand |
| $K = \frac{x^2-2\frac{\mu_1\sigma_2^2+\mu_2\sigma_1^2}{\sigma_1^2+\sigma_2^2}x-\frac{\mu_1^2\sigma_2^2+\mu_2^2\sigma_1^2}{\sigma_1^2+\sigma_2^2}}{2\frac{\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2}}$ | Divide by $\sigma_1^2+\sigma_2^2$ |

**Table G.1:** *Exponent term.*

Define a zero term:

$$\epsilon = \frac{\left(\frac{\mu_1\sigma_2^2+\mu_2\sigma_1^2}{\sigma_1^2+\sigma_2^2}\right)^2 - \left(\frac{\mu_1\sigma_2^2+\mu_2\sigma_1^2}{\sigma_1^2+\sigma_2^2}\right)^2}{2\frac{\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2}} = 0 \tag{G.4}$$

Add the zero term to $K$:

$$K = K + \epsilon = \frac{x^2 - 2\frac{\mu_1\sigma_2^2+\mu_2\sigma_1^2}{\sigma_1^2+\sigma_2^2}x + \left(\frac{\mu_1\sigma_2^2+\mu_2\sigma_1^2}{\sigma_1^2+\sigma_2^2}\right)^2}{2\frac{\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2}} + \frac{\frac{\mu_1^2\sigma_2^2+\mu_2^2\sigma_1^2}{\sigma_1^2+\sigma_2^2} - \left(\frac{\mu_1\sigma_2^2+\mu_2\sigma_1^2}{\sigma_1^2+\sigma_2^2}\right)^2}{2\frac{\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2}} \tag{G.5}$$

The $K$ equation is too long. Let us try to simplify the red and the blue terms separately.

$$\frac{x^2 - 2\frac{\mu_1\sigma_2^2+\mu_2\sigma_1^2}{\sigma_1^2+\sigma_2^2}x + \left(\frac{\mu_1\sigma_2^2+\mu_2\sigma_1^2}{\sigma_1^2+\sigma_2^2}\right)^2}{2\frac{\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2}} = \frac{\left(x - \frac{\mu_1\sigma_2^2+\mu_2\sigma_1^2}{\sigma_1^2+\sigma_2^2}\right)^2}{2\frac{\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2}} \tag{G.6}$$

$$\begin{aligned}
\frac{\frac{\mu_1^2\sigma_2^2+\mu_2^2\sigma_1^2}{\sigma_1^2+\sigma_2^2} - \left(\frac{\mu_1\sigma_2^2+\mu_2\sigma_1^2}{\sigma_1^2+\sigma_2^2}\right)^2}{2\frac{\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2}} &= \frac{\frac{\mu_1^2\sigma_2^2+\mu_2^2\sigma_1^2}{\left(\sigma_1^2+\sigma_2^2\right)^2}\left(\sigma_1^2+\sigma_2^2\right) - \left(\frac{\mu_1\sigma_2^2+\mu_2\sigma_1^2}{\sigma_1^2+\sigma_2^2}\right)^2}{2\frac{\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2}} \\
&= \frac{\mu_1^2\sigma_2^2\sigma_1^2+\mu_1^2\sigma_2^4+\mu_2^2\sigma_1^4+\mu_2^2\sigma_1^2\sigma_2^2-\mu_1^2\sigma_2^4-2\mu_1\mu_2\sigma_1^2\sigma_2^2-\mu_2^2\sigma_1^4}{2\sigma_1^2\sigma_2^2\left(\sigma_1^2+\sigma_2^2\right)} \\
&= \frac{\mu_1^2+\mu_2^2-2\mu_1\mu_2}{2\left(\sigma_1^2+\sigma_2^2\right)} = \frac{(\mu_1-\mu_2)^2}{2\left(\sigma_1^2+\sigma_2^2\right)}
\end{aligned} \tag{G.7}$$

Combine the red term with the blue term:

$$K = \frac{\left(x - \frac{\mu_1\sigma_2^2+\mu_2\sigma_1^2}{\sigma_1^2+\sigma_2^2}\right)^2}{2\frac{\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2}} + \frac{(\mu_1-\mu_2)^2}{2(\sigma_1^2+\sigma_2^2)} \tag{G.8}$$

Multiply the nominator and denominator by: $1/\sqrt{\sigma_1^2+\sigma_2^2}$

$$p(x)_{12} = \frac{1/\sqrt{\sigma_1^2+\sigma_2^2}}{2\pi\sqrt{\frac{\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2}}} exp\left(-\frac{\left(x - \frac{\mu_1\sigma_2^2+\mu_2\sigma_1^2}{\sigma_1^2+\sigma_2^2}\right)^2}{2\frac{\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2}}\right) exp\left(-\frac{(\mu_1-\mu_2)^2}{2(\sigma_1^2+\sigma_2^2)}\right) \tag{G.9}$$

Rearrange:

$$p(x)_{12} = \frac{1}{\sqrt{2\pi}\sqrt{\frac{\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2}}} exp\left(-\frac{\left(x - \frac{\mu_1\sigma_2^2+\mu_2\sigma_1^2}{\sigma_1^2+\sigma_2^2}\right)^2}{2\frac{\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2}}\right) \frac{1}{\sqrt{2\pi(\sigma_1^2+\sigma_2^2)}} exp\left(-\frac{(\mu_1-\mu_2)^2}{2(\sigma_1^2+\sigma_2^2)}\right) \tag{G.10}$$

Let us define:

$$\mu_{12} = \frac{\mu_1\sigma_2^2+\mu_2\sigma_1^2}{\sigma_1^2+\sigma_2^2} \tag{G.11}$$

$$\sigma_{12}^2 = \frac{\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2} \tag{G.12}$$

Thus:

$$p(x)_{12} = \frac{1}{\sqrt{2\pi}\sigma_{12}} exp\left(-\frac{(x-\mu_{12})^2}{2\sigma_{12}^2}\right) \frac{1}{\sqrt{2\pi(\sigma_1^2+\sigma_2^2)}} exp\left(-\frac{(\mu_1-\mu_2)^2}{2(\sigma_1^2+\sigma_2^2)}\right) \tag{G.13}$$

The green term is actually a Gaussian PDF with mean $\mu_{12}$ and standard deviation $\sigma_{12}$.

The orange term is a scaling factor that doesn't depend on $x$:

$$S = \frac{1}{\sqrt{2\pi(\sigma_1^2+\sigma_2^2)}} exp\left(-\frac{(\mu_1-\mu_2)^2}{2(\sigma_1^2+\sigma_2^2)}\right) \tag{G.14}$$

$$p(x)_{12} = S\frac{1}{\sqrt{2\pi}\sigma_{12}} exp\left(-\frac{(x-\mu_{12})^2}{2\sigma_{12}^2}\right) \tag{G.15}$$

**Summary**

The product of two Gaussian PDFs is proportional to the Gaussian PDF:

$$\frac{1}{\sqrt{2\pi}\sigma_{12}} exp\left(-\frac{(x-\mu_{12})^2}{2\sigma_{12}^2}\right) \tag{G.16}$$

with mean:

$$\mu_{12} = \frac{\mu_1\sigma_2^2 + \mu_2\sigma_1^2}{\sigma_1^2 + \sigma_2^2} \tag{G.17}$$

and variance:

$$\sigma_{12}^2 = \frac{\sigma_1^2\sigma_2^2}{\sigma_1^2 + \sigma_2^2} \tag{G.18}$$

## G.2 Product of $n$ univariate Gaussian PDFs

Let us see the result of three Gaussian multiplication.

Derivation of variance:

| Equation | Notes |
|---|---|
| $\sigma_{123}^2 = \frac{\sigma_{12}^2\sigma_3^2}{\sigma_{12}^2+\sigma_3^2}$ | $\sigma_{123}^2$ - variance of three Gaussians product<br>$\sigma_{12}^2$ - variance of two Gaussians product<br>$\sigma_3^2$ - variance of the third Gaussian |
| $\sigma_{123}^2 = \frac{\frac{\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2}\sigma_3^2}{\frac{\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2}+\sigma_3^2}$ | Plug $\sigma_{12}^2 = \frac{\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2}$ |
| $\sigma_{123}^2 = \frac{\sigma_1^2\sigma_2^2\sigma_3^2}{\sigma_3^2(\sigma_1^2+\sigma_2^2)+\sigma_1^2\sigma_2^2}$ | Simplify |
| $\sigma_{123}^2 = \frac{\sigma_1^2\sigma_2^2\sigma_3^2}{\sigma_3^2\sigma_1^2+\sigma_3^2\sigma_2^2+\sigma_1^2\sigma_2^2}$ | Expand |
| $\sigma_{123}^2 = \frac{1}{\frac{1}{\sigma_1^2}+\frac{1}{\sigma_2^2}+\frac{1}{\sigma_3^2}}$ | Divide by $\sigma_1^2\sigma_2^2\sigma_3^2$ |

**Table G.2:** *Three Gaussian multiplication variance.*

Derivation of mean:

| **Equation** | **Notes** |
|---|---|
| $\mu_{123} = \frac{\sigma_3^2\mu_{12}+\sigma_{12}^2\mu_3}{\sigma_3^2+\sigma_{12}^2}$ | $\mu_{123}$ - mean of three Gaussians product<br>$\mu_{12}$ - mean of two Gaussians product<br>$\mu_3$ - mean of the third Gaussians<br>$\sigma_{12}^2$ - variance of two Gaussians product<br>$\sigma_3^2$ - variance of the third Gaussian |
| $\mu_{123} = \frac{\sigma_3^2\frac{\sigma_2^2\mu_1+\sigma_1^2\mu_2}{\sigma_1^2+\sigma_2^2}+\frac{\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2}\mu_3}{\sigma_3^2+\frac{\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2}}$ | Plug:<br>$\mu_{12} = \frac{\sigma_2^2\mu_1+\sigma_1^2\mu_2}{\sigma_1^2+\sigma_2^2}$<br>$\sigma_{12}^2 = \frac{\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2}$ |
| $\mu_{123} = \frac{\sigma_3^2\sigma_2^2\mu_1+\sigma_3^2\sigma_1^2\mu_2+\sigma_1^2\sigma_2^2\mu_3}{\sigma_3^2\sigma_2^2+\sigma_3^2\sigma_1^2+\sigma_1^2\sigma_2^2}$ | |
| $\mu_{123} = \frac{\sigma_3^2\sigma_2^2\mu_1+\sigma_3^2\sigma_1^2\mu_2+\sigma_1^2\sigma_2^2\mu_3}{\sigma_1^2\sigma_2^2\sigma_3^2}\sigma_{123}^2$ | Plug:<br>$\sigma_3^2\sigma_2^2+\sigma_3^2\sigma_1^2+\sigma_1^2\sigma_2^2 = \frac{\sigma_1^2\sigma_2^2\sigma_3^2}{\sigma_{123}^2}$ |
| $\mu_{123} = \left(\frac{\mu_1}{\sigma_1^2}+\frac{\mu_2}{\sigma_2^2}+\frac{\mu_3}{\sigma_3^2}\right)\sigma_{123}^2$ | |

**Table G.3:** *Three Gaussian multiplication variance.*

Using induction, we can show that the product of $n$ Gaussians has the following properties:

$$\mu_N = \left(\sum_{i=1}^{n}\frac{\mu_i}{\sigma_1^2}\right)\sigma_N^2 \tag{G.19}$$

$$\sigma_N^2 = \frac{1}{\frac{1}{\sigma_1^2}+\frac{1}{\sigma_2^2}+...+\frac{1}{\sigma_n^2}} = \frac{1}{\sum_{i=1}^{n}\sigma_i^2} \tag{G.20}$$

# H. Product of multivariate Gaussian PDFs

## H.1 Product of n multivariate Gaussian PDFs

In this section, we derive the product of $n$ multivariate (multi-dimensional) Gaussian PDFs [16].

Let $\mathcal{N}(\boldsymbol{\mu}; \boldsymbol{\Sigma})$ be a multivariate normal distribution, where $\boldsymbol{\mu}$ is the vector of means for variables $\boldsymbol{x}$ and $\boldsymbol{\Sigma}$ is the covariance matrix.

The PDF of $\mathcal{N}(\boldsymbol{\mu}; \boldsymbol{\Sigma})$ is given by:

$$\boldsymbol{p}(\boldsymbol{x}) = \frac{1}{(2\pi)^{k/2}|\boldsymbol{\Sigma}|^{1/2}} exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right) \tag{H.1}$$

Where $k$ is a dimension of $\boldsymbol{x}$.

For a more convenient notation, let us change the form of the equation, by moving the $\frac{1}{(2\pi)^{k/2}|\boldsymbol{\Sigma}|^{1/2}}$ term into the exponent.

Table H.1: Change the form of a multivariate Gaussian equation.

| **Equation** | **Notes** |
|---|---|
| $\frac{1}{(2\pi)^{k/2}\|\boldsymbol{\Sigma}\|^{1/2}} = exp\left(ln\left(\frac{1}{(2\pi)^{k/2}\|\boldsymbol{\Sigma}\|^{1/2}}\right)\right)$ $= exp\left(-ln\left((2\pi)^{k/2}\|\boldsymbol{\Sigma}\|^{1/2}\right)\right)$ | Move the $\frac{1}{(2\pi)^{k/2}\|\boldsymbol{\Sigma}\|^{1/2}}$ term into the exponent. |
| $\boldsymbol{p}(\boldsymbol{x}) = exp\Big(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})$ $-ln\left((2\pi)^{k/2}\|\boldsymbol{\Sigma}\|^{1/2}\right)\Big)$ | Rewrite $\boldsymbol{p}(\boldsymbol{x})$ |
| $(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})$ $= \left(\boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{x} - \boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} - \boldsymbol{\mu}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{x} + \boldsymbol{\mu}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\right)$ | Expand the term $(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})$ |

Continued on next page

Table H.1: Change the form of a multivariate Gaussian equation. (Continued)

| | |
|---|---|
| $(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}-\boldsymbol{\mu})$<br>$= \left(\boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{x} - 2\boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} + \boldsymbol{\mu}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\right)$ | $\boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} = \boldsymbol{\mu}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{x}$<br>(product of a vector, matrix, and transposed vector is a scaler) |
| $\boldsymbol{p}(\boldsymbol{x}) = exp\Big( -\frac{1}{2}\boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}$<br>$-\frac{1}{2}\boldsymbol{\mu}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} - ln\left((2\pi)^{k/2}\lvert\boldsymbol{\Sigma}\rvert^{1/2}\right)\Big)$ | Rewrite $\boldsymbol{p}(\boldsymbol{x})$ |
| $\boldsymbol{p}(\boldsymbol{x}) = exp\left(-\frac{1}{2}\boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} + \boldsymbol{g}\right)$ | Define the term $\boldsymbol{g}$ as follows:<br>$\boldsymbol{g} = -\frac{1}{2}\boldsymbol{\mu}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} - ln\left((2\pi)^{k/2}\lvert\boldsymbol{\Sigma}\rvert^{1/2}\right)$ |

Now, our multivariate normal PDF equation is much more elegant:

$$\boldsymbol{p}(\boldsymbol{x}) = exp\left(\boldsymbol{g} + \boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} - \frac{1}{2}\boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{x}\right) \tag{H.2}$$

The product of $n$ Gaussian PDFs is:

$$\boldsymbol{p}(\boldsymbol{x})_N = \prod_{i=1}^{n} \boldsymbol{p}(\boldsymbol{x})_i = exp\left(\sum_{i=1}^{n} \boldsymbol{g}_i + \boldsymbol{x}^T\left(\sum_{i=1}^{n} \boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i\right) - \frac{1}{2}\boldsymbol{x}^T\left(\sum_{i=1}^{n} \boldsymbol{\Sigma}_i^{-1}\right)\boldsymbol{x}\right) \tag{H.3}$$

Let us define:

$$\boldsymbol{\Sigma}_N^{-1} = \sum_{i=1}^{n} \boldsymbol{\Sigma}_i^{-1} \tag{H.4}$$

$$\boldsymbol{\Sigma}_N^{-1}\boldsymbol{\mu}_N = \sum_{i=1}^{n} \boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i \tag{H.5}$$

Rewrite $\boldsymbol{p}(\boldsymbol{x})_N$:

$$\boldsymbol{p}(\boldsymbol{x})_N = \prod_{i=1}^{n} \boldsymbol{p}(\boldsymbol{x})_i = exp\left(\sum_{i=1}^{n} \boldsymbol{g}_i + \boldsymbol{x}^T\boldsymbol{\Sigma}_N^{-1}\boldsymbol{\mu}_N - \frac{1}{2}\boldsymbol{x}^T\boldsymbol{\Sigma}_N^{-1}\boldsymbol{x}\right) \tag{H.6}$$

Note that $g$ is a constant that doesn't depend on $x$.

Define:

$$\boldsymbol{g}_N = -\frac{1}{2}\boldsymbol{\mu}_N^T \boldsymbol{\Sigma}_N^{-1} \boldsymbol{\mu}_N - ln\left((2\pi)^{k/2}|\boldsymbol{\Sigma}|_N^{1/2}\right) \tag{H.7}$$

We can write the following:

$$\boldsymbol{p}(\boldsymbol{x})_N = exp\left(\sum_{i=1}^{n} \boldsymbol{g}_i - \boldsymbol{g}_N + \boldsymbol{g}_N + \boldsymbol{x}^T \boldsymbol{\Sigma}_N^{-1} \boldsymbol{\mu}_N - \frac{1}{2}\boldsymbol{x}^T \boldsymbol{\Sigma}_N^{-1} \boldsymbol{x}\right) \tag{H.8}$$

Take the orange term to a different exponent:

$$\boldsymbol{p}(\boldsymbol{x})_N = exp\left(\sum_{i=1}^{n} \boldsymbol{g}_i - \boldsymbol{g}_N\right) exp\left(\boldsymbol{g}_N + \boldsymbol{x}^T \boldsymbol{\Sigma}_N^{-1} \boldsymbol{\mu}_N - \frac{1}{2}\boldsymbol{x}^T \boldsymbol{\Sigma}_N^{-1} \boldsymbol{x}\right) \tag{H.9}$$

The orange term is a scaling factor $S$:

$$S = exp\left(\sum_{i=1}^{n} \boldsymbol{g}_i - \boldsymbol{g}_N\right) \tag{H.10}$$

$$\boldsymbol{p}(\boldsymbol{x})_N = S \cdot exp\left(\boldsymbol{g}_N + \boldsymbol{x}^T \boldsymbol{\Sigma}_N^{-1} \boldsymbol{\mu}_N - \frac{1}{2}\boldsymbol{x}^T \boldsymbol{\Sigma}_N^{-1} \boldsymbol{x}\right) \tag{H.11}$$

We can see that the product of n multivariate Gaussian PDFs is a Gaussian PDF (the green term) multiplied by a scaling factor with the following properties:

$$\boldsymbol{\Sigma}_N^{-1} = \sum_{i=1}^{n} \boldsymbol{\Sigma}_i^{-1} \tag{H.12}$$

$$\boldsymbol{\mu}_N = \boldsymbol{\Sigma}_N \sum_{i=1}^{n} \boldsymbol{\Sigma}_i^{-1} \boldsymbol{\mu}_i \tag{H.13}$$

## H.2 Product of 2 multivariate Gaussian PDFs

In the literature, you can also see that $\mu_{12}$ for two sensors is calculated as follows:

$$\boldsymbol{\mu}_{12} = \boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_1 \left(\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2\right)^{-1} \left(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1\right) \tag{H.14}$$

First, it looks familiar. It is identical to the Kalman Filter state update equation, which also calculates the fusion of two normally distributed PDFs – the measurement and the prior estimate. Second, it is computationally effective. We perform only one matrix inversion.

In this section, we derive Equation H.14.

$$\boldsymbol{\Sigma}_{12}^{-1} = \sum_{i=1}^{2} \boldsymbol{\Sigma}_i^{-1} = \boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1} \tag{H.15}$$

$$\boldsymbol{\mu}_{12} = \boldsymbol{\Sigma}_{12} \sum_{i=1}^{2} \boldsymbol{\Sigma}_i^{-1} \boldsymbol{\mu}_i = \boldsymbol{\Sigma}_{12} \left( \boldsymbol{\Sigma}_1^{-1} \boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_2^{-1} \boldsymbol{\mu}_2 \right) \tag{H.16}$$

Substitute $\boldsymbol{\Sigma}_{12}$:

$$\begin{aligned} \boldsymbol{\mu}_{12} &= \left( \boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1} \right)^{-1} \left( \boldsymbol{\Sigma}_1^{-1} \boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_2^{-1} \boldsymbol{\mu}_2 \right) \\ &= \left( \boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1} \right)^{-1} \boldsymbol{\Sigma}_1^{-1} \boldsymbol{\mu}_1 + \left( \boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1} \right)^{-1} \boldsymbol{\Sigma}_2^{-1} \boldsymbol{\mu}_2 \end{aligned} \tag{H.17}$$

Define a zero term:

$$\boldsymbol{\epsilon} = \left( \boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1} \right)^{-1} \boldsymbol{\Sigma}_2^{-1} \boldsymbol{\mu}_1 - \left( \boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1} \right)^{-1} \boldsymbol{\Sigma}_2^{-1} \boldsymbol{\mu}_1 = 0 \tag{H.18}$$

Add $\boldsymbol{\epsilon}$ to $\boldsymbol{\mu}_{12}$:

$$\begin{aligned} \boldsymbol{\mu}_{12} = \boldsymbol{\mu}_{12} + \boldsymbol{\epsilon} = &\left( \boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1} \right)^{-1} \boldsymbol{\Sigma}_1^{-1} \boldsymbol{\mu}_1 + \left( \boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1} \right)^{-1} \boldsymbol{\Sigma}_2^{-1} \boldsymbol{\mu}_2 \\ &+ \left( \boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1} \right)^{-1} \boldsymbol{\Sigma}_2^{-1} \boldsymbol{\mu}_1 - \left( \boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1} \right)^{-1} \boldsymbol{\Sigma}_2^{-1} \boldsymbol{\mu}_1 \end{aligned} \tag{H.19}$$

Simplify the red term:

$$\left( \boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1} \right)^{-1} \left( \boldsymbol{\Sigma}_1^{-1} \boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_2^{-1} \boldsymbol{\mu}_1 \right) = \left( \boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1} \right)^{-1} \left( \boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1} \right) \boldsymbol{\mu}_1 = \boldsymbol{\mu}_1 \tag{H.20}$$

Simplify the blue term:

$$\left( \boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1} \right)^{-1} \boldsymbol{\Sigma}_2^{-1} \boldsymbol{\mu}_2 - \left( \boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1} \right)^{-1} \boldsymbol{\Sigma}_2^{-1} \boldsymbol{\mu}_1 = \left( \boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1} \right)^{-1} \boldsymbol{\Sigma}_2^{-1} \left( \boldsymbol{\mu}_2 - \boldsymbol{\mu}_1 \right) \tag{H.21}$$

$$\boldsymbol{\mu}_{12} = \boldsymbol{\mu}_1 + \left(\boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1}\right)^{-1} \boldsymbol{\Sigma}_2^{-1} \left(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1\right) \tag{H.22}$$

There are four matrix inversions in this equation.
Let us simplify the term $\left(\boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1}\right)^{-1} \boldsymbol{\Sigma}_2^{-1}$.

| **Equation** | **Notes** |
|---|---|
| $\left(\boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1}\right)^{-1} \boldsymbol{\Sigma}_2^{-1}$ | |
| $\left(\boldsymbol{\Sigma}_2 \left(\boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1}\right)\right)^{-1}$ | Apply the matrix inverse property: $(\boldsymbol{BA})^{-1} = \boldsymbol{A}^{-1}\boldsymbol{B}^{-1}$ |
| $\left(\boldsymbol{\Sigma}_2\boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2\boldsymbol{\Sigma}_2^{-1}\right)^{-1}$ | Expand |
| $\left(\boldsymbol{\Sigma}_2\boldsymbol{\Sigma}_1^{-1} + \boldsymbol{I}\right)^{-1}$ | |
| $\left(\boldsymbol{\Sigma}_2\boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_1\boldsymbol{\Sigma}_1^{-1}\right)^{-1}$ | Multiply and divide $\boldsymbol{I}$ by $\boldsymbol{\Sigma}_1$ |
| $\left(\left(\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2\right) \boldsymbol{\Sigma}_1^{-1}\right)^{-1}$ | |
| $\boldsymbol{\Sigma}_1 \left(\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2\right)^{-1}$ | Apply the matrix inverse property: $(\boldsymbol{BA})^{-1} = \boldsymbol{A}^{-1}\boldsymbol{B}^{-1}$ |

**Table H.2:** *Reducing the number of the matrix inversions.*

$$\boldsymbol{\mu}_{12} = \boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_1 \left(\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2\right)^{-1} \left(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1\right) \tag{H.23}$$

# Bibliography

## Articles


[1] R. E. Kalman. "A New Approach to Linear Filtering and Prediction Problems". In: *Trans. of the ASME - Journal of Basic Engineering* (1960), pages 35–45 (cited on page 5).

[2] Tarunraj Singh Dirk Tenne. "Optimal design of $\alpha - \beta - (\gamma)$ filters". In: *Proceedings of the American Control Conference 6* 6 (Feb. 2000), pages 4348–4352. DOI: 10.1109/ACC.2000.877043 (cited on page 73).

[3] P. R. Kalata. "A generalized parameter for $\alpha - \beta$ and $\alpha - \beta - \gamma$ target trackers". In: *The 22nd IEEE Conference on Decision and Control* (Dec. 1983). DOI: 10.1109/CDC.1983.269580 (cited on page 73).

[4] Wayne E. Hoover. "Algorithms For Confidence Circles and Ellipses". In: *NOAA Technical Report NOS 107 CGS 3* (Sept. 1984) (cited on page 148).

[5] L. Campo Y. Bar-Shalom. "The Effect of the Common Process Noise on the Two-Sensor Fused-Track Covariance". In: *Advances in Control Systems* 3 (1966), pages 293–340. DOI: https://doi.org/10.1016/B978-1-4831-6716-9.50011-4 (cited on page 237).

[6] S. F. Schmidt L. A. Mcgee. "Discovery of the Kalman filter as a practical tool for aerospace and industry". In: *ech. Rep., NASA-TM-86847* (1985). DOI: 10.1109/TAES.1986.310815 (cited on page 237).

[7] Kai O. Arras. "An Introduction To Error Propagation: Derivation, Meaning and Examples of Equation Cy=FxCxFxT". In: *EPFL-ASL-TR-98-01 R3* (1998). DOI: https://doi.org/10.3929/ethz-a-010113668 (cited on page 246).

[8] Jeffrey K. Uhlmann Simon J. Julier. "New extension of the Kalman filter to nonlinear systems". In: *Proc. SPIE 3068, Signal Processing, Sensor Fusion, and Target Recognition VI* (July 1997). DOI: https://doi.org/10.1117/12.280797 (cited on pages 283, 284, 337).

[9] Eric A. Wan Rudolph Van Der Merwe. "Sigma-point kalman filters for probabilistic inference in dynamic state-space models". In: *Oregon Health and Science University* (2004). DOI: https://doi.org/10.6083/M4Z60KZ5 (cited on pages 285, 337).

[10] H.F. Durrant-Whyte S. Julier J. Uhlmann. "A new method for the nonlinear transformation of means and covariances in filters and estimators". In: *IEEE*

*Transactions on Automatic Control* 45.3 (Mar. 2000), pages 477–482. DOI: 10.1109/9.847726 (cited on page 298).

[11] Herman Bruyninckx Tine Lefebvre and Joris De Schutter. "Comment on A new method for the nonlinear transformation of means and covariances in filters and estimators". In: *IEEE Transactions on Automatic Control* 47.8 (Aug. 2002), pages 1406–1408. DOI: 10.1109/TAC.2002.800742 (cited on page 298).

[12] Rudolph van der Merwe Eric A. Wan. "The Unscented Kalman Filter for Nonlinear Estimation". In: *IEEE Proceedings of IEEE 2000 adaptive systems for signal processing, communication and control symposium* (Oct. 2000). DOI: https://doi.org/10.3390/s21020438 (cited on page 320).

[13] René van de Molengraft Jos Elfring Elena Torta. "Particle Filters: A Hands-On Tutorial". In: *Sensors 2021, 21(2)* (2021), page 438 (cited on page 337).

[14] Y. Bar-Shalom K. C. Chang R. K. Saha. "On optimal track-to-track fusion". In: *IEEE Transactions on Aerospace and Electronic Systems* 33.8 (Oct. 1997), pages 1271–1276. DOI: doi:10.1109/7.625124 (cited on page 348).

[15] L. Campo Y. Bar-Shalom. "The Effect of the Common Process Noise on the Two-Sensor Fused-Track Covariance". In: *IEEE Transactions on Aerospace and Electronic Systems* AES-22.6 (Nov. 1986), pages 803–805. DOI: 10.1109/TAES.1986.310815 (cited on page 348).

[16] P.A. Bromiley. "Products and Convolutions of Gaussian Probability Density Functions". In: *Internal Report* (Aug. 2014) (cited on pages 421, 427).

## Books

[17] Joseph P.D. Bucy R.S. *Filtering for Stochastic Processes with Applications to Guidance, Chapter 16*. Interscience, New York, 1968 (cited on page 182).

[18] Uhlmann Jeffrey. *Dynamic map building and localization: new theoretical foundations, Thesis (Ph.D.)*. University of Oxford, 1995 (cited on pages 283, 320).

[19] Tom M. Apostol. *Calculus, theorem 8.3*. John Wiley and Sons, Inc, 1967 (cited on page 404).

# Index

## M



## N



## O



## P



## R



## S



## T



## U



## V



## Z