Thilogen Thambirajah

Pose Estimation with The Invariant Extended Kalman Filter as a Stable Observer

Master's thesis in Mechanical engineering
Supervisor: Olav Egeland
June 2022

Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Mechanical and Industrial Engineering

NTNU
Norwegian University of
Science and Technology

Thilogen Thambirajah

Pose Estimation with The Invariant Extended Kalman Filter as a Stable Observer

Master’s thesis in Mechanical engineering
Supervisor: Olav Egeland
June 2022

Norwegian University of Science and Technology
Faculty of Engineering
Department of Mechanical and Industrial Engineering

NTNU
Norwegian University of
Science and Technology

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Olav Egeland, for the guidance throughout this master's thesis. The feedback and assistance provided played a significant role in completing this master thesis and is greatly appreciated.

Abstract

A typical application of an IMU is to measure the acceleration and angular velocity using accelerometers and gyroscopes, and is often found in localization problems within the robotics field. In this thesis, an extension of the extended Kalman filter (EKF), termed the invariant extended Kalman filter (IEKF), is derived from [2] in order to achieve convergence around any trajectory which is a coveted property for nonlinear observers.

Two cases from [2], simple car model and navigation on flat earth, are simulated to investigate the IEKF as a stable observer in comparison to the EKF. The simulations displays the superiority of the IEKF as it outperforms the EKF on every simulation performed. When it comes to more challenging cases, the EKF is seen to diverge, whereas the IEKF does not diverge as a result of the logarithm of the error obeying a linear differential equation, also referred to as a log-linear property, which gives local stability around any trajectory.

Sammendrag

En typisk anvendelse av en IMU er å måle akselerasjonen og vinkelhastigheten ved hjelp av akselerometre og gyroskoper, og finnes ofte i lokaliseringsproblemer innen robotikkfeltet. I denne oppgaven er en utvidelse av det utvidede Kalman-filteret (EKF), kalt det invariante utvidede Kalman-filteret (IEKF), utledet fra [2] for å oppnå konvergens rundt enhver bane som er en ettertraktet egenskap for ikke-lineære observatører.

To tilfeller fra [2], enkel bilmodell og navigasjon på flat jord, er simulert for å undersøke IEKF som en stabil observatør i forhold til EKF. Simuleringene viser IEKFs overlegenhet ettersom den overgår EKF på hver simulering som utføres. Når det gjelder mer utfordrende tilfeller, ser man at EKF divergerer, mens IEKF ikke divergerer som et resultat av at logaritmen til feilen følger en lineær differensialligning, også referert til som en log-lineær egenskap, som gir lokal stabilitet rundt hvilken som helst bane.

Contents

List of Figures

List of Tables

Chapter 1.

Introduction

Kalman filtering and nonlinear observers are important in pose estimation with IMUs. Pose estimation is the operation of estimating attitude and position of a robot, and plays a significant role for autonomous robots or vehicles in regards to decision making about future actions [9].

A typical application for pose estimation is navigation [4],[19] which requires accurate estimation of robot pose. For such problems, filter based methods like the extended Kalman filter (EKF) are widely used as a result of its simplicity and efficiency [16],[14],[5]. The goal of a filter estimator is to achieve convergence to zero of the state estimate [9]. However, the EKF does not guarantee optimality and its efficiency is spontaneous. To achieve this goal, the Invariant EKF built on the nonlinear observer theory [2] is introduced, where the theory of invariant observer design is based on the estimation error being invariant under the action of matrix Lie group [6]. Since the EKF uses Kalman equations to stabilize the estimation error, a general method is to attempt to derive local convergence properties around any trajectories using the EKF [2].

The next chapter in this thesis consists of the background, where theoretical background is provided such that the reader gains familiarity with the filter equations for the cases presented later in this thesis. In chapter 3, attitude filtering using quaternions for the multiplicative extended Kalman filter (MEKF) and the right invariant extended Kalman filter (RIEKF) is introduced. Further, chapter 4 presents two cases from [2] that are to be simulated using the nonlinear observer design as presented by Barrau and Bonnabel, in order to highlight the properties of the Invariant EKF. At last, the remaining chapters displays the results of the filters for the corresponding initialization parameters, where the results are discussed and concluded.

1.1. Notations

In this thesis, $\mathbb{R}^{n\times n}$ is used to denote a matrix with dimension $n \times n$ with real entities, and $\mathbb{R}^n$ is used to denote a vector with dimension n. Given a matrix M, the inverse is denoted M^{-1}, the transpose is denoted M^T and the skew-symmetric is denoted $(M)^\times$. At last, I_n is used to denote an identity matrix with dimension $n \times n$.

Chapter 2.

Background

This chapter serves the theoretical background for the implementations presented later in this thesis. Given this background information, one should be able to understand and implement the presented equations and algorithms in chapter 3 and chapter 4.

2.1. Linear and nonlinear systems

In the state-space model, the state is defined by the state variables given in Euclidean space as

$$x = (x_1, x_2, \cdots, x_n) \in \mathbb{R}^n \tag{2.1}$$

where $x \in \mathbb{R}^n$ indicates that the state x is a vector with n number of state variables. The input variable can also be written in the form

$$u = (u_1, u_2, \cdots, u_n) \in \mathbb{R}^p \tag{2.2}$$

indicating that the input vector consist of p number of inputs.

2.1.1. Linear systems

A linear model is a mathematical model of a process, where it is possible to control the process and also extract information. The tools needed for estimation and control of a process is easier to implement and understand for linear systems as opposed to nonlinear systems.

A linear system is defined in continuous-time using the equations in the state space

$$\dot{x} = Ax + Bu \tag{2.3}$$

$$y = Cx \tag{2.4}$$

where x is the state vector, u is the control/input vector and y is the output vector. Consequently, A is the system matrix, B is the input matrix and C is the output matrix. Even If the matrices A, B and C are time-varying, the system is still linear. Figure 2.1 shows a simplified explanation of a linear system dynamics.

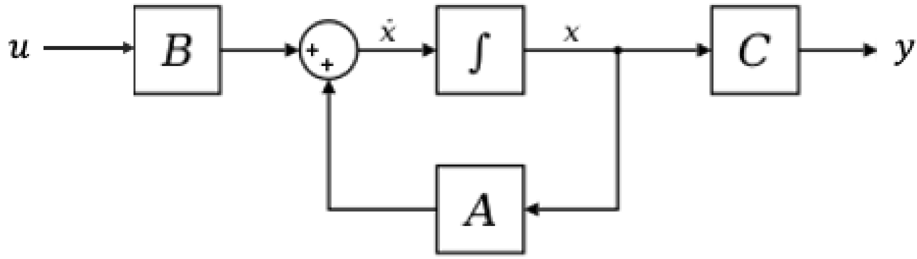

Figure 2.1.: Matrix block diagram of a linear system with state x, input u and output y

2.1.2. Nonlinear systems

A nonlinear system can be written in continuous-time as

$$\dot{x} = f(x, u, w) \tag{2.5}$$

$$y = h(x, v) \tag{2.6}$$

where $f(\cdot)$ and $h(\cdot)$ are nonlinear functions, and w and v are the process noise and measurement noise, respectively. In the case where $f(\cdot)$ and $h(\cdot)$ are explicit functions of t, then the system is time-varying. It is noted that the system is nonlinear, unless $f(x, u, w) = Ax + Bu + w$ and $h(x, v) = Hx + v$.

2.2. Lie Groups

From [17], a Lie Group is defined as a smooth manifold where the group operation and the inversion must be continous. The group operation must be associative, there must be an identity element $e \in G$ and there must be an inversion to satisfy the usual group axioms for all $g, g1, g2, g3 \in G$:

$$eg = ge = g \qquad (identity)$$
$$g^{-1}g = gg^{-1} = e \qquad (inverses)$$
$$g\left(g_2 g_3\right) = \left(g_1 g_2\right) g_3 = g_1 g_2 g_3 \qquad (associativity)$$

2.2.1. Matrix Lie Groups

A matrix lie group G is a closed subgroup of the defined set $GL(n;\mathbb{R})$ of invertible $n \times n$ matrices with real entries, where $M_n(\mathbb{R}) = \mathbb{R}^{n\times n}$ is defined as the set of all $n \times n$ matrices with real entities. Since G is subgroup of $GL(n;\mathbb{R})$, it has the following properties

$$I_n \in G, \quad \forall g \in G, g^{-1} \in G, \quad \forall a, b \in G, ab \in G \tag{2.7}$$

where I_n is the identity matrix of $\mathbb{R}^n$. The matrix lie group G is associated with a vector space, $\mathfrak{g}$, called the Lie algebra of G which is a real subspace of $M_n(\mathbb{R})$ with the same dimension as G. $\mathfrak{g}$ can be identified to $\mathbb{R}^{\dim \mathfrak{g}}$ using the linear invertible map $\mathscr{L}_{\mathfrak{g}} : \mathbb{R}^{\dim \mathfrak{g}} \to \mathfrak{g}$ and it can be mapped to the matrix lie group G through the matrix exponential $\exp_m$, yielding $\exp(\xi) = \exp_m\left(\mathscr{L}_{\mathfrak{g}}(\xi)\right)$ for $\xi \in \mathbb{R}^{\dim \mathfrak{g}}$. This map is invertible for small ξ, and we have $(\exp(\xi))^{-1} = \exp(-\xi)$. Also for any $g \in G$, the adjoint matrix $\mathrm{ad}(g) \in \mathbb{R}^{\dim \mathfrak{g} \times \dim \mathfrak{g}}$ is defined by $g \exp(\zeta) g^{-1} = \exp\left(\mathrm{ad}(g)\zeta\right)$ for all $\zeta \in \mathfrak{g}$ [3].

2.2.2. Group of rotation matrices, $SO(2)$

Elements of the rotation group in two dimensions are represented by 2D rotation matrices. G is defined as a group of rotation matrices, $SO(2)$, so that

$$G = SO(2) = \left\{ R \in \mathscr{M}_2(\mathbb{R}), RR^T = I, \det(R) = 1 \right\} \tag{2.8}$$

and $\mathfrak{g}$ the space of skew-symmetric matrices $\mathfrak{so}(2)$ is

$$\mathfrak{g} = \mathfrak{so}(2) = \left\{ A \in \mathscr{M}_2(\mathbb{R}), A = -A^T \right\} \tag{2.9}$$

recall from the introduction above that $\mathscr{M}_2(\mathbb{R}) = \mathbb{R}^{2\times 2}$. The logarithm is then defined as

$$\mathscr{L}_{\mathfrak{so}(2)}(\theta) = (\theta)^{\times} = \begin{pmatrix} 0 & -\theta \\ \theta & 0 \end{pmatrix} \tag{2.10}$$

and the exponential map is expressed as

$$\exp\left(\theta^{\times}\right) = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix} \tag{2.11}$$

which verifies $R = \exp\left(\theta^{\times}\right)$.

2.2.3. Group of rotation matrices, $SO(3)$

Elements of the rotation group in two dimensions are represented by 3D rotation matrices. G is defined as a group of rotation matrices, $SO(3)$, so that

$$G = SO(3) = \left\{ R \in \mathscr{M}_3(\mathbb{R}), RR^T = I, \det(R) = 1 \right\} \tag{2.12}$$

and $\mathfrak{g}$ the space of skew-symmetric matrices $\mathfrak{so}(3)$ is

$$\mathfrak{g} = \mathfrak{so}(3) = \left\{ A \in \mathscr{M}_3(\mathbb{R}), A = -A^T \right\} \tag{2.13}$$

The logarithm is then defined as

$$\mathscr{L}_{\mathfrak{so}(3)} \begin{pmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \end{pmatrix} = \begin{pmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \end{pmatrix}^{\times} = \begin{pmatrix} 0 & -\xi_3 & \xi_2 \\ \xi_3 & 0 & -\xi_1 \\ -\xi_2 & \xi_1 & 0 \end{pmatrix} \tag{2.14}$$

and the exponential map verifies $R = \exp(\xi)$ and is expressed as

$$\exp(\xi) = I + \left(\frac{\sin(\|\xi\|)}{\|\xi\|} \right) S + \left(\frac{1 - \cos(\|\xi\|)}{\|\xi\|^2} \right) S^2 \tag{2.15}$$

where $S = \mathscr{L}_{\mathfrak{so}(3)} \begin{pmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \end{pmatrix}$.

2.2.4. Group of direct planar isometries, $SE(2)$

The group $SE(2)$ has three dimensions corresponding to translation and rotation in the plane. G is defined as a group of direct planar isometries, $SE(2)$, so that it is represented in homogeneous form as

$$G = SE(2) = \left\{ \begin{pmatrix} R(\theta) & x \\ 0_{1\times 2} & 1 \end{pmatrix}; \theta \in \mathbb{R}, x \in \mathbb{R}^2 \right\} \tag{2.16}$$

where $R(\theta)$ is planar rotation matrix of angle θ. Then the lie algebra $\mathfrak{g}$ is defined as

$$\mathfrak{g} = \mathfrak{se}(2) = \left\{ \begin{pmatrix} 0 & -\alpha & u_1 \\ \alpha & 0 & u_2 \\ 0 & 0 & 0 \end{pmatrix}; \begin{pmatrix} \alpha \\ u_1 \\ u_2 \end{pmatrix} \in \mathbb{R}^3 \right\} \tag{2.17}$$

which gives the vector form of the logarithm and the logarithm

$$\zeta = \begin{pmatrix} \alpha \\ u_1 \\ u_2 \end{pmatrix} \tag{2.18}$$

$$\mathscr{L}_{\mathfrak{se}(2)}(\zeta) = \begin{pmatrix} 0 & -\alpha & u_1 \\ \alpha & 0 & u_2 \\ 0 & 0 & 0 \end{pmatrix} \tag{2.19}$$

Further, the Lie exponential map writes

$$\exp(\zeta) = \begin{pmatrix} R(\alpha) & E(\alpha)u \\ 0_{1\times 2} & 1 \end{pmatrix} \tag{2.20}$$

where $R(\alpha) \in SO(2)$, $u = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$ and the matrix $E(\alpha)$ is given as

$$E(\alpha) = \begin{pmatrix} \sin(\alpha)/\alpha & -(1-\cos(\alpha)/\alpha) \\ (1-\cos(\alpha))/\alpha & \sin(\alpha)/\alpha \end{pmatrix} \tag{2.21}$$

and the adjoint representation of the logarithm is defined as

$$ad(\zeta) = \begin{pmatrix} 0 & 0 & 0 \\ u_2 & 0 & -\omega_k \\ -u_1 & \omega_k & 0 \end{pmatrix} \tag{2.22}$$

where ω_k is the angular velocity at time step k.

2.2.5. Group of double direct planar isometries, $SE_2(3)$

G is defined as a group of double direct planar isometries, $SE_2(3)$, so that

$$G = SE_2(3) = \left\{ \begin{pmatrix} R & v & x \\ 0_{1\times 3} & 1 & 0 \\ 0_{1\times 3} & 0 & 1 \end{pmatrix}; R \in SO(3), v, x \in \mathbb{R}^3 \right\} \tag{2.23}$$

Then the lie algebra $\mathfrak{g}$ is defined as

$$\mathfrak{g} = \mathfrak{se}_2(3) = \left\{ \begin{pmatrix} (\xi)^\times & u & y \\ 0 & 0 & 0 \\ 0_{1\times 3} & 0 & 0 \end{pmatrix}; \xi, u, y \in \mathbb{R}^3 \right\} \tag{2.24}$$

which gives the vector form of the logarithm and the logarithm

$$\zeta = \begin{pmatrix} \xi \\ u \\ y \end{pmatrix} \tag{2.25}$$

$$\mathscr{L}_{\mathfrak{se}_2(3)} \begin{pmatrix} \xi \\ u \\ y \end{pmatrix} = \begin{pmatrix} (\xi)^{\times} & u & y \\ 0_{1\times 3} & 0 & 0 \\ 0_{1\times 3} & 0 & 0 \end{pmatrix} \tag{2.26}$$

Then the Lie exponential map writes

$$\exp(\zeta) = I_5 + S + \frac{1-\cos(\|\xi\|)}{\|\xi\|^2} S^2 + \frac{\|\xi\| - \sin(\|\xi\|)}{\|\xi\|^3} S^3 \tag{2.27}$$

where $S = \mathscr{L}_{\mathfrak{se}_2(3)}(\xi, u, y)^T$.

2.3. Quaternions

This section serves as an introduction to quaternions where the main focus is to present the expression for the mathematical computations using quaternions. A quaternion can be given in various forms, and the most common forms and their corresponding mathematical operations are presented in this section. The material in this section is inspired by [8] and [21].

2.3.1. Hamilton's representation

Quaternions were first invented by William Rowan Hamilton, a 19th-century Irish mathematician, and often appears in mathematics as an algebraic system [21].

A quaternion is a vector with one real and three imaginary parts, written in the form

$$q = q_s + q_1 i + q_2 j + q_3 k \in \mathbb{H} \tag{2.28}$$

where q_s, q_1, q_2 and q_3 are real coefficients and the complex units i, j and k satisfies

$$i^2 = j^2 = k^2 = -1$$
$$ij = -ji = k, \quad jk = -kj = i, \quad ki = -ik = j$$

It is noted that the notation $\mathbb{H}$ is to indicate that a quaternion is defined as in the fomulation of Hamilton, which can be seen from equation (2.28).

Then, multiplying a quaternion with a scalar α gives

$$\alpha q = \alpha (q_s + q_1 i + q_2 j + q_3 k) = \alpha q_s + \alpha q_1 i + \alpha q_2 j + \alpha q_3 k \tag{2.29}$$

and the inner product of a quaternion can be computed by defining the inner product of the complex units Two quaternions are defined as $q = q_s + q_1 i + q_2 j + q_3 k$ and $p = p_s + p_1 i + p_2 j + p_3 k$, in order to express the formulas for the mathematical operations using quaternions.

Addition and subtraction is computed element-wise and gives

$$q \pm p = q_s \pm p_s + (q_1 \pm p_1)\, i + (q_2 \pm p_2)\, j + (q_3 \pm p_3)\, k \tag{2.30}$$

and the multiplication of the two quaternions is computed exactly like the multiplication of complex numbers, which writes

$$\begin{aligned} qp &= (q_s + q_1 i + q_2 j + q_3 k)(p_s + p_1 i + p_2 j + p_3 k) \\ &= q_s p_s - q_1 p_1 - q_2 p_2 - q_3 p_3 \\ &+ (q_s p_1 + p_s q_1 + q_2 p_3 - q_3 p_2)\, i \\ &+ (q_s p_2 + p_s q_2 + q_3 p_1 - q_1 p_3)\, j \\ &+ (q_s p_3 + p_s q_3 + q_1 p_2 - q_2 p_1)\, k \end{aligned} \tag{2.31}$$

The conjugate of a quaternion is defined as a quaternion with opposite signs on the imaginary parts

$$q^* = q_s - iq_1 - jq_2 - kq_3 \tag{2.32}$$

and the magnitude of a quaternion $\|q\|$ is defined as

$$\|q\|^2 = q_s^2 + q_1^2 + q_2^2 + q_3^2 \tag{2.33}$$

This can also be seen for the quaternion product of a quaternion and its corresponding conjugate quaternion, which is given as

$$qq^* = q_s^2 + q_1^2 + q_2^2 + q_3^2 \tag{2.34}$$

which indicates that

$$\|q\|^2 = qq^* \tag{2.35}$$

From this, the inverse quaternion is given using the quaternion product by $qq^{-1} = 1$. The inverse quaternion is then defined as

$$q^{-1} = \frac{q^*}{\|q\|^2} \tag{2.36}$$

2.3.2. Quaternion represented by a scalar and a vector

As mentioned earlier in this chapter, a quaternion consist of a real part and imaginary part. Then a quaternion can be defined using a scalar and a vector where the scalar represents the real part of a quaternion and the vector is a three-dimensional vector that represents the imaginary part of a quaternion. Therefore, a quaternion can be formulated as

$$q = \alpha + \beta \in \mathbb{H} \tag{2.37}$$

where $\alpha \in \mathbb{R}$ and $\beta \in \mathbb{R}^3$. So multiplying with a scalar γ gives

$$\gamma q = \gamma\alpha + \gamma\beta \in \mathbb{H} \tag{2.38}$$

Further, two quaternions are defined as $q_1 = \alpha_1 + \beta_1$ and $q_2 = \alpha_2 + \beta_2$. Then addition and substraction is computed component wise as

$$q_1 \pm q_2 = (\alpha_1 \pm \alpha_2) + (\beta_1 \pm \beta_2) \in \mathbb{H} \tag{2.39}$$

The multiplication of the two quaternions is defined as

$$q_1 \circ q_2 = (\alpha_1\alpha_2 - \beta_1 \cdot \beta_2) + (\alpha_1\beta_2 + \alpha_2\beta_1 + \beta_1 \times \beta_2) \in \mathbb{H} \tag{2.40}$$

and the quaternion product is associative, so it satisfies

$$q_1 \circ (q_2 \circ q_3) = (q_1 \circ q_2) \circ q_3 = q_1 \circ q_2 \circ q_3 \in \mathbb{H} \tag{2.41}$$

The conjugate quaternion is formulated as

$$q^* = \alpha - \beta \tag{2.42}$$

and the magnitude of a quaternion $\|q\|$ is

$$\|q\|^2 = q^* \circ q \tag{2.43}$$

The quaternion product of a quaternion and its corresponding conjugate quaternion is defined as

$$q \circ q^* = q^* \circ q = \alpha^2 + \beta \cdot \beta \tag{2.44}$$

and the inverse quaternion is defined exactly as in equation (2.36).

2.3.3. Vector represented as a quaternion

A vector can be formulated as a quaternion with a zero scalar part. The quaternion product of a quaternion q and a vector v is computed as

$$q \circ v = -\beta \cdot v + \alpha v + \beta \times v \tag{2.45}$$

$$v \circ q = -\beta \cdot v + \alpha v - \beta \times v \tag{2.46}$$

and the quaternion product of a vector v_1 and a vector v_2 is defined as

$$v_1 \circ v_2 = -v_1 \cdot v_2 + v_1 \times v_2 \tag{2.47}$$

The conjugate of a vector is defined as $v^* = -v$ and the magnitude of vector $\|v\|$ is

$$\|v\|^2 = v \circ v^* = -v \circ v = v \cdot v \tag{2.48}$$

2.3.4. Quaternion represented as a four-dimensional vector

A commonly used representation of a quaternion is a four-dimensional vector, so that the quaternion is defined as

$$[q] = \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \in \mathbb{R}^4 \tag{2.49}$$

Considering two quaternions q_1 and q_2, the quaternion product of the two quaternions gives

$$\begin{bmatrix} q_1 \circ & q_2 \end{bmatrix} = \begin{bmatrix} \alpha_1 \alpha_2 - \beta_1^{\mathrm{T}} \beta_2 \\ \alpha_1 \beta_2 + \alpha_2 \beta_1 + \beta_1 \times \beta_2 \end{bmatrix} \tag{2.50}$$

and alternatively, the quaternion product can be computed using matrices as

$$[q_1 \circ q_2] = Q_L(q_1)[q_2] = Q_R(q_2)[q_1] \tag{2.51}$$

with

$$Q_L(q) = \begin{bmatrix} \alpha & -\beta^{\mathrm{T}} \\ \beta & \alpha I + \beta^{\times} \end{bmatrix} \tag{2.52}$$

$$Q_R(q) = \begin{bmatrix} \alpha & -\beta^{\mathrm{T}} \\ \beta & \alpha I - \beta^{\times} \end{bmatrix} \tag{2.53}$$

A vector $v \in \mathbb{H}$ is defined as a four-dimensional vector, which gives

$$[v] = \begin{bmatrix} 0 \\ v \end{bmatrix} \in \mathbb{R}^4 \tag{2.54}$$

and the quaternion product of a quaternion q and a vector v is then computed as

$$[q \circ v] = Q_L(q)[v] \tag{2.55}$$

$$[v \circ q] = Q_R(q)[v] \tag{2.56}$$

Further, the conjugate quaternion is defined as

$$[q^*] = \begin{bmatrix} \alpha \\ -\beta \end{bmatrix} \in \mathbb{R}^4 \tag{2.57}$$

and the magnitude of a quaternion $\|q\|$ is found from

$$\|q\|^2 = [q]^{\mathrm{T}}[q] \tag{2.58}$$

2.3.5. Unit quaternions

A unit quaternion has four parameters and can be expressed as a 3×3 rotation matrix, which is the reason to why unit quaternions plays an important role in applications involving rotations, i.e. aerospace, robotics, drones, etc.

A unit quaternion is defined as a quaternion with unity norm. The unit quaternion is then defined as

$$q = \eta + \epsilon \tag{2.59}$$

which satisfies

$$\|q\|^2 = q \cdot q = q \circ q^* = \eta^2 + \epsilon \cdot \epsilon = 1 \tag{2.60}$$

In the case where a quaternion is defined as four-dimensional vector, the norm $\|q\|^2$ is given as

$$\|q\|^2 = \begin{bmatrix} \eta \\ \epsilon \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \eta \\ \epsilon \end{bmatrix} = \eta^2 + \epsilon^{\mathrm{T}}\epsilon = 1 \tag{2.61}$$

It is then seen that the conjugate quaternion can be written as

$$q^* = q^{-1} \tag{2.62}$$

A unit quaternion can also be defined by the euler paramaters $\eta = \cos\frac{\theta}{2}$ and $\epsilon = k \sin\frac{\theta}{2}$, where k is a unit vector. Considering a rotation matrix $R \in SO(3)$ with angle and axis parameteres given by θ and k, respectively, the rotation matrix can then be expressed using the Euler parameters as

$$R = I + 2\eta\epsilon^{\times} + 2\epsilon^{\times}\epsilon^{\times} \tag{2.63}$$

2.3.6. Quaternion logarithm and exponential

From the rotaton defined earlier, an angle ϕ is defined as $\phi = \frac{\theta}{2}$, and a quaternion is represented as ϕk. Then the exponential function writes

$$\exp(\phi k) = \cos\phi + k\sin\phi \tag{2.64}$$

and consequently the unit quaternion can be expressed using the exponential function as

$$q = \exp\left(\frac{\theta}{2}k\right) \tag{2.65}$$

Following this, the logarithm can then be defined as

$$\log(q) = \frac{\theta}{2}k \tag{2.66}$$

Computation of the logarithm and the exponential

Two cases are considered where for the first case, a logarithm v is given by the angle ϕ and a unit vector k, which writes $v = \phi k$. This means that from equation (2.64) the exponential of the logarithm gives $\exp(v) = \exp(\phi k) = \cos\phi + k\sin\phi$. A quaternion is to be found from the exponential logarithm as $q = \exp(v)$.

For the second case, a quaternion is defined as $q = \eta + \epsilon$ and the logarithm is defined so that it satisfies $\exp(v) = q$. This implies that $\epsilon = k\sin\phi$ which gives the angle formulated as $\phi = \arcsin\|\epsilon\|$.

Then, the computation of a quaternion from the logarithm can be defined from the first case as

$$\exp(v) = \cos\|v\| + \sin\|v\|\frac{v}{\|v\|} \tag{2.67}$$

As can be seen, this equation is undefined when $\|v\| = 0$. An alternative formulation is then expressed as

$$\exp(v) = \cos\|v\| + \operatorname{sinc}(\|v\|)v \tag{2.68}$$

where

$$\operatorname{sinc}(x) = \frac{\sin x}{x} \tag{2.69}$$

With this, it is possible to estimate $\operatorname{sinc}(x)$ using the Taylor series expansion of $\frac{\sin x}{x}$, when x is close to zero.

Further, the computation of the logarithm from a quaternion can be defined from the second case as

$$v = \frac{\arcsin\|\epsilon\|}{\|\epsilon\|}\epsilon \tag{2.70}$$

Here it is also seen that the equation is undefined when $\|\epsilon\| = 0$. This is avoided by estimating $\frac{\arcsin x}{x}$ using the Taylor series expansion when x is close to zero.

2.4. Kalman Filters

In this section, the Kalman filter and the extended Kalman filter are presented as background information for the filters used in this thesis, the multiplicative extended Kalman filter and the left and right invariant extended Kalman filter, which is implemented for attitude estimation and 3D SLAM. It is noted that the mentioned filters are all a modification of the extended Kalman filter.

2.4.1. The Kalman Filter

Kalman filter is an algorithm that recursively estimates the state variables given the measurements observed over time. It is an important tool in control systems and have been demonstrating its usefulness in various applications, i.e., estimation of attitude and the combined estimation of attitude and position. The updating process for Kalman filters is fairly general and so the Kalman filter have relatively simple form and require small computational power, meaning that it can often be implemented in real time [12].

Continuous-time Kalman Filter

A continuous-time linear system is given by the state-space model as

$$\dot{x} = A_c x + B_c u + n_c \tag{2.71}$$
$$y = Cx + w_c \tag{2.72}$$

where A_c is the state transition matrix corresponding to the state vector $x \in \mathbb{R}^n$, B_c is the control-input matrix corresponding to the control vector $u \in \mathbb{R}^q$, n_c is the process noise and w_c is the measurement noise. The process noise and measurement noise has zero mean and covariance defined as, respectively [18]

$$E\left\{n_{c,t} n_{c,t+\tau}^{\mathrm{T}}\right\} = Q_c \delta_\tau \tag{2.73}$$
$$E\left\{w_{c,t} w_{c,t+\tau}^{\mathrm{T}}\right\} = R_c \delta_\tau \tag{2.74}$$

where $\delta(\cdot)$ is the Kroneker delta function.

Then the corresponding Kalman Filter for this system is defined as

$$\dot{\hat{x}} = A_c \hat{x} + B_c u + K_c(y - C\hat{x}) \tag{2.75}$$
$$\dot{P} = P A_c^{\mathrm{T}} + A_c P + Q_c - P C^{\mathrm{T}} R_c^{-1} C P \tag{2.76}$$

where $\hat{x}$ is the estimated state, $P = \mathrm{cov}(x - \hat{x})$ is the covariance of the state estimation error $\tilde{x} = x - \hat{x}$ and K_c is the Kalman gain matrix defined as

$$K_c = P C^{\mathrm{T}} R_c^{-1} \tag{2.77}$$

The measurement and the measurement estimation error is then defined as

$$\hat{y} = C\hat{x} \tag{2.78}$$
$$\tilde{y} = y - C\hat{x} = y - \hat{y} \tag{2.79}$$

Here it can be seen that the time propagation of the state for the Kalman filter given in equation (2.75) is identical to the first equation from the linear system described in equation (2.71), but with an additional correction term $K_c(y - C\hat{x})$. It is then obvious that the correction term is proportional to the measurement estimation error $\tilde{y}$. From the formula of the Kalman gain given in equation (2.77), the relationship between the Kalman gain K_c, the covariance of state estimation error P and the covariance of measurement noise R_c can be explained by the fact that K_c increases as P increases and that K_c decreases as R_c increases.

Discrete-time Kalman Filter

The linear system from equations (2.71), (2.72) in discrete-time is defined as

$$x_k = A_k x_{k-1} + B_k u_{k-1} + n_k \tag{2.80}$$
$$y_k = C_k x_k + w_k \tag{2.81}$$

Then the corresponding Kalman Filter for this system is defined as a propagation and update part, where the propagation is given as

$$\hat{x}_{k|k-1} = A_k \hat{x}_{k-1|k-1} + B_k u_{k-1} \tag{2.82}$$
$$P_{k|k-1} = A_k P_{k-1|k-1} A_k^{\mathrm{T}} + Q_k \tag{2.83}$$

It is noted that the estimated state vector $\hat{x}_{k|k-1}$ gives the estimate of x_k based on measurements up until time step $k-1$, with k being the current time step, whereas $\hat{x}_{k-1|k-1}$ gives the estimate of x_{k-1} based on measurements up until time step $k-1$. Using this logic, it is obvious that $\hat{x}_{k|k}$ would give the estimate based on measurements up until time step k. The timeline from Figure 2.2 provides a simple explanation of the time propagation of the state estimate in discrete-time. This type of notation is commonly used when presenting the equations of the algorithm for the different filters later in the thesis.

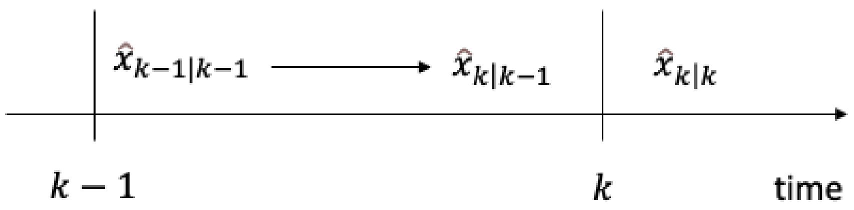

Figure 2.2.: Timeline illustrating the propagation and update notations used in relation to the state estimate [18]

.

Further, in order to calculate the update, the estimated measurement and measurement estimation error is defined as

$$\hat{y} = C_k \hat{x}_{k|k-1} \tag{2.84}$$
$$\tilde{y}_k = y_k - C_k \hat{x}_{k|k-1} \tag{2.85}$$

where the Kalman gain is computed using the covariance of the measurement estimation error, S, and gives

$$S_k = C_k P_{k|k-1} C_k^{\mathrm{T}} + R_k \tag{2.86}$$
$$K_k = P_{k|k-1} C_k^{\mathrm{T}} S_k^{-1} \tag{2.87}$$

Then the updated state estimate and the updated covariance is found using

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{y}_k \tag{2.88}$$
$$P_{k|k} = (I - K_k C_k) P_{k|k-1} \tag{2.89}$$

Continuous-discrete Kalman Filter

Assuming that the discrete-time system from equations (2.80), (2.81) is a discretization of the continous-time system from equations (2.71), (2.72), Euler´s first method is used yielding the system matrices

$$A_k = \exp\left(hA_c\right) \approx I + hA_c \tag{2.90}$$

$$B_k = \int_0^h \exp\left(\tau A_c\right) B \;\; d\tau \approx hB_c \tag{2.91}$$

$$Q_k = \int_0^h \exp\left(\tau A_c\right) Q_c \exp\left(\tau A_c\right)^{\mathrm{T}} \;\; d\tau \approx hQ_c \tag{2.92}$$

$$R_k = \frac{1}{h} R_c \tag{2.93}$$

and so the equation from (2.83) is now formulated as

$$\begin{aligned} P_{k|k-1} &= A_k P_{k-1|k-1} A_k^{\mathrm{T}} + Q_k \\ &= (I - A_c P_{k-1|k-1}) P_{k-1|k-1} (I - A_c P_{k-1|k-1})^{\mathrm{T}} \\ &\approx P_{k-1|k-1} + h\left(A_c P_{k-1|k-1} + P_{k-1|k-1} A_c^{\mathrm{T}} + Q_c\right) \end{aligned} \tag{2.94}$$

Here it can be seen that as h approaches zero, the euler discretization of the covariance propagation approches the covariance propagation from equation (2.83).

Then the time propagation from timestep $k-1$ to timestep k of the Kalman Filter is defined as

$$\dot{\hat{x}}_t = A_c \hat{x}_t + B_c u, \quad k-1 \leq t < k \tag{2.95}$$

$$\dot{P}_t = P_t A_c^{\mathrm{T}} + A_c^{\mathrm{T}} P_t + Q_c \tag{2.96}$$

and propagates $\hat{x}$ from $\hat{x}_{k-1|k-1}$ to $\hat{x}_{k|k-1}$ and P from $P_{k-1|k-1}$ to $P_{k|k-1}$. Further, the propagated state estimate and the propagated covariance is used in the update part of the system

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{y}_k \tag{2.97}$$

$$P_{k|k} = \left(I - K_k C_k\right) P_{k|k-1} \tag{2.98}$$

where $\tilde{y}, S_k$ and K_k is defined exactly the same as for the discrete-time system. When it comes to simulation of the Kalman filter, it is common that the equations for the filter are presented in contionus-discrete time, which is the case for the filters presented in this thesis later.

2.4.2. The Extended Kalman Filter

From the previous subsection, the Kalman filter was presented using a linear system dynamics. In practice all systems are ultimately nonlinear, so most applications where the Kalman filter is important involves a nonlinear system. The extended Kalman filter was then introduced as a nonlinear extension of the Kalman filter. The main idea of the filter is to use the nonlinear state-space model for time propagation of the state estimate and to use the linearized error dynamics to compute the time propagation of the state covariance matrix and the Kalman gain matrix [18]. The extended Kalman filter is used in a wide range of applications such as navigation, mobile robots, tracking of planes, etc.

Continuous-time Extended Kalman Filter

A continuous-time nonlinear system is given by the state-space model as

$$\dot{x}_t = f_c(x_t, u_t; t) + n_{c,t} \tag{2.99}$$
$$y_t = h(x_t; t) + w_t \tag{2.100}$$

Then the time propagation of the corresponding extended Kalman filter for this system is defined as

$$\dot{\hat{x}}_t = f_c(\hat{x}_t, u_t; t) + K_c(y_t - h(\hat{x}_t; t)) \tag{2.101}$$
$$\hat{y} = h(\hat{x}_t; t) \tag{2.102}$$

The state error $\tilde{x} = x - \hat{x}$ and innovation $\tilde{y} = y - \hat{y}$ is linearized in order to apply the Kalman filter to the nonlinear system. The linearized model is then used to define the covariance and to compute the Kalman gain.

The error model is defined as

$$\dot{\tilde{x}}_t = f_c(\underbrace{\hat{x}_t + \tilde{x}_t}_{x_t}, u_t; t) + n_{c,t} - f_c(\hat{x}_t, u_t; t) - K_c\tilde{y}_t \tag{2.103}$$
$$\tilde{y}_t = h(\underbrace{\hat{x}_t + \tilde{x}_t}_{x_t}; t) + w_t - h(\hat{x}_t; t) \tag{2.104}$$

and it is obvious to see that x_t is substituted with $\hat{x}_t + \tilde{x}_t$ by using the formula for the state error $\tilde{x} = x - \hat{x}$. Then the linearization of the estimate $\hat{x}_t$ is defined as

$$\left.\frac{\partial f_c}{\partial \tilde{x}}\right|_{\hat{x},u} \tilde{x} \approx f_c(\hat{x}_t + \tilde{x}_t, u_t; t) - f_c(\hat{x}_t, u_t; t) \tag{2.105}$$
$$\left.\frac{\partial h}{\partial \tilde{x}}\right|_{\hat{x}} \tilde{x} \approx h(\hat{x}_t + \tilde{x}_t; t) - h(\hat{x}_t; t) \tag{2.106}$$

where the partial derivatives are evaluated at $x = \hat{x}$ and the linearized model is computed, giving

$$\dot{\tilde{x}}_t = (A_c - KC)\,\tilde{x}_t + n_{c,t} \tag{2.107}$$
$$\tilde{y}_t = C\tilde{x}_t + w_t \tag{2.108}$$

with the linearized matrices A_c and C defined as

$$A_c = \left.\frac{\partial f_c}{\partial \tilde{x}}\right|_{\hat{x},u}, \quad C = \left.\frac{\partial h}{\partial \tilde{x}}\right|_{\hat{x}} \tag{2.109}$$

Further, the linearized model is used to compute the Kalman gain K_c and the time propagation of the covariance P, and the equations writes

$$\dot{P} = A_cP + PA^{\mathrm{T}} + Q - PC^{\mathrm{T}}R_c^{-1}CP \tag{2.110}$$
$$K_c = PC^{\mathrm{T}}R_c^{-1} \tag{2.111}$$

Discrete-time Extended Kalman Filter

The nonlinear system defined from equation (2.99), (2.100) in discrete-time is defined as

$$x_k = f\left(x_{k-1}, u_{k-1}; t_k\right) + n_{k-1} \tag{2.112}$$
$$y_k = h\left(x_k; t_k\right) + w_k \tag{2.113}$$

Then the corresponding extended Kalman filter for this system is defined as a propagation and update part, where the time propagation is given as

$$\hat{x}_{k|k-1} = f\left(\hat{x}_{k-1|k-1}, u_{k-1}; t_k\right) \tag{2.114}$$

The error model is defined from the state error $\tilde{x} = x - \hat{x}$ and innovation $\tilde{y} = y - \hat{y}$ as

$$\tilde{x}_k = f\left(x_{k-1}, u_{k-1}; t_k\right) + n_{k-1} - f\left(\hat{x}_{k-1|k-1}, u_{k-1}; t_k\right) \tag{2.115}$$

$$\tilde{y}_k = h\left(x_k; t_k\right) + w_k - h\left(\hat{x}_k; t_k\right) \tag{2.116}$$

and linearization gives the linearized error model

$$\tilde{x}_k = A\tilde{x}_{k-1} + n_{k-1} \tag{2.117}$$

$$\tilde{y}_k = C\tilde{x}_{k-1} + w_k \tag{2.118}$$

where the linearized matrices are defined as

$$A_k = \left.\frac{\partial f}{\partial \tilde{x}}\right|_{\hat{x}_{k-1|k-1}, u_{k-1}}, \quad C_k = \left.\frac{\partial h}{\partial \tilde{x}}\right|_{\hat{x}_{k|k-1}} \tag{2.119}$$

Further, the linearized model is used to compute and the Kalman gain K_c and the time propagation of the covariance P, where the covariance propagation writes

$$P_{k|k-1} = A_{k-1} P_{k-1|k-1} A_{k-1}^{\mathrm{T}} + Q_k \tag{2.120}$$

and the Kalman gain is found by

$$S_k = C_k P_{k|k-1} C_k^{\mathrm{T}} + R_k \tag{2.121}$$

$$K_k = P_{k|k-1} C_k^{\mathrm{T}} S_k^{-1} \tag{2.122}$$

Finally, the updated state estimate and the updated covariance is computed as

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{y}_k \tag{2.123}$$

$$P_{k|k} = \left(I - K_k C_k\right) P_{k|k-1} \tag{2.124}$$

Continuous-discrete Extended Kalman Filter

For the continuous-discrete extended Kalman filter, the time propagation of the state estimate $\hat{x}$ and the state covariance P is given in continous-time, while the update of the state estimate and the state covariance is given in discrete-time as the measurements are obtained at discrete instants of time.

The time propagation of the state estimate is defined as

$$\dot{\hat{x}}_t = f_c\left(\hat{x}_t, u_t; t\right) \tag{2.125}$$

and the innovation is defined as

$$\tilde{y}_k = y_k - h\left(\hat{x}_{k|k-1}; t_k\right) \tag{2.126}$$

The error model given by the state error $\tilde{x} = x - \hat{x}$ and the innovation $\tilde{y} = y - \hat{y}$ is used to present the linearized error model

$$\dot{\tilde{x}}_t = A_c \tilde{x}_t + n_{c,t} \tag{2.127}$$
$$\tilde{y}_t = C \tilde{x}_t + w_t \tag{2.128}$$

where the linearized matrices are defined as

$$A_c = \left. \frac{\partial f_c}{\partial \tilde{x}} \right|_{\hat{x},u}, \quad C = \left. \frac{\partial h}{\partial \tilde{x}} \right|_{\hat{x}} \tag{2.129}$$

The linearized model is used to compute and the Kalman gain K_c and the time propagation of the covariance P, where the covariance propagation writes

$$\dot{P}_t = P_t A_c^{\mathrm{T}} + A_c^{\mathrm{T}} P_t + Q_c \tag{2.130}$$

Finally, the updated state estimate and the updated covariance is computed as

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{y}_k \tag{2.131}$$
$$P_{k|k} = (I - K_k C_k) P_{k|k-1} \tag{2.132}$$

where K_c and S_k is defined exactly as for the discrete-time extended Kalman filter.

Chapter 3.

Nonlinear Attitude Filtering

This chapter is an introduction to the MEKF and RIEKF, where the necessary equations required to simulate nonlinear attitude filtering is presented. It is noted that this case is not simulated in this thesis, as this chapter is extracted from the specialization project prior to this thesis. This chapter only serves the purpose of introducing the reader to nonlinear attitude filtering and the MEKF and RIEKF filter. Although the results for this case is not included, the initialization parameters for this case can be found in chapter 5.

3.1. Multiplicative Extended Kalman Filter - MEKF

The MEKF is a modification of the Extended Kalman Filter (EKF) to estimate a three-component attitude error. The MEKF error quaternion results in an identity quaternion using an invariant output error termed left-invariant error instead of a linear error used for the EKF [15].

The presented equations and algorithm for the MEKF in this chapter are mainly based and inspired by [13], [15]. It is also worth to mention that these equations are the same as in [20], where the MEKF is simulated in a comparative study. The time propagation from [20] is calculated using Euler´s method, where the notations k and $k-1$ represents the current time step and the previous time step, respectively. The attitude is represented by an unit quaternion $q = \eta + \epsilon$, which is equivalent to the rotation matrix $R = I + 2\eta\epsilon + \epsilon^{\times}\epsilon^{\times}$.

Since q is a unit quaternion, the kinematic equation can be written in the form of

$$\dot{q} = \frac{1}{2} q \circ \omega \tag{3.1}$$

where ω corresponds to the angular velocity of the attitude. Following Farrenkopf´s model [10], the gyroscope measurement ω_m is expressed as

$$\omega_m(t) = \omega(t) + b(t) + n_1(t) \tag{3.2}$$

and the bias model as

$$\dot{b} = n_2 \tag{3.3}$$

where n_1 and n_2 is zero-mean white noise and b is the gyroscope bias. Combining equation (3.1) and (3.2) presents the system dynamics as

$$\dot{q} = \frac{1}{2} q \circ (\omega_m - b - n_1) \tag{3.4}$$
$$\dot{b} = n_2 \tag{3.5}$$

It is noted that in order to get an accurate estimation of the attitude R, it is important to determine the bias, b, by estimation.

3.1.1. Time propagation of state estimate

Given an angular velocity measured by a gyroscope, ω_m, the estimated angular velocity is expressed as

$$\hat{\omega} = \omega_m - \hat{b} \tag{3.6}$$

The estimate of the measurement i is expressed by the estimated rotation matrix, $\hat{R}$, and is given as

$$\hat{y}_i = \hat{R}^{\mathrm{T}} d_i^n \tag{3.7}$$

where d_i is a known vector in the spatial frame n, e.g the earth magnetic field in NED coordinates as it is described in [7], and is not to be confused with "d" further in this text. It is noted that $\hat{R}$ is defined by the components of $\hat{q}$.

The estimate of the gyroscope bias is updated by the following formula

$$\hat{b}_k = \hat{b}_{k-1} + h\left(K_b \tilde{y}\right) \tag{3.8}$$

where

$$K_b \tilde{y} = P_{c,k-1}^{\mathrm{T}} C_{ai}^{\mathrm{T}} R_c^{-1} \tilde{y} = P_{c,k-1}^{\mathrm{T}} \delta \tag{3.9}$$

and the estimated measurement error, $\tilde{y}$, is defined as $\tilde{y} = y - \hat{y}$. Further, the update of the state estimate is

$$\hat{q}_k = \hat{q}_{k-1} \circ \exp\left(0.5h\left(\hat{\omega} + P_{a,k-1}\delta\right)\right) \tag{3.10}$$

where h is the time step.

3.1.2. Error equations and linearization of error quaternions

The error equations for the MEKF is presented as

$$\tilde{q} = \hat{q}^{-1} \circ q \tag{3.11}$$
$$\tilde{b} = b - \hat{b} \tag{3.12}$$

Then the error dynamics of attitude can be described by the kinematic differential equation of the error quaternion, and is presented as

$$\dot{\tilde{q}} = \hat{q}^{-1} \circ \dot{q} + \frac{\mathrm{d}}{\mathrm{d}t}\left(\hat{q}^{-1}\right) \circ q \tag{3.13}$$

Following the article written by Silvère Bonnabel in 2009 [7] it is mentioned that if q depends on time, then $\dot{q}^{-1} = -q^{-1} * \dot{q} * q^{-1}$. This gives

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\hat{q}^{-1}\right) = -\hat{q}^{-1} \circ \dot{\hat{q}} \circ \hat{q}^{-1} \tag{3.14}$$

By combining equation (3.13) and (3.14), the differential equation of the error matrix is seen as

$$\begin{aligned}
\dot{\tilde{q}} &= \hat{q}^{-1} \circ \dot{q} - \hat{q}^{-1} \circ \dot{\hat{q}} \circ \hat{q}^{-1} \circ q \\
&= \hat{q}^{-1} \circ \frac{1}{2} q \circ (\omega_m - b - n_1) - \hat{q}^{-1} \circ \frac{1}{2}\hat{q} \circ \left(\omega_m - \hat{b} + K_q \tilde{y}\right) \circ \tilde{q} \\
&= \frac{1}{2}\tilde{q} \circ \left(\omega_m - (\hat{b} + \tilde{b}) - n_1\right) - \frac{1}{2}\left(\omega_m - \hat{b} + K_q \tilde{y}\right) \circ \tilde{q} \\
&= \tilde{\epsilon}^{\times}\hat{\omega} - \underbrace{\frac{1}{2}\left(\tilde{\epsilon}^{\mathrm{T}}\right)\left(-\tilde{b} + K_q \tilde{y} + n_1\right)}_{\text{Scalar}} + \frac{1}{2}\tilde{\eta}\left(-\tilde{b} - K_q \tilde{y} + n_1\right) + \frac{1}{2}\tilde{\epsilon}^{\times}\left(-\tilde{b} + K_q \tilde{y} + n_1\right)
\end{aligned} \tag{3.15}$$

From the resulting equation, the vector part of the error equation then presents the following kinematic differential equation

$$\dot{\tilde{\epsilon}} = \tilde{\epsilon}^{\times}\hat{\omega} + \frac{1}{2}\tilde{\eta}\left(-\tilde{b} - K_q \tilde{y} + n_1\right) + \frac{1}{2}\tilde{\epsilon}^{\times}\left(-\tilde{b} + K_q \tilde{y} + n_1\right) \tag{3.16}$$

It is noted that $\tilde{a} = 2\tilde{\epsilon}$ is used as the three parameter representation of the error rotation as long as $\hat{\theta} < \pi$, which is assumed to be the case for all practical implementations.

Given this information, the linearized error dynamics is expressed as

$$\begin{bmatrix} \dot{\tilde{a}} \\ \dot{\tilde{b}} \end{bmatrix} = \begin{bmatrix} -\hat{\omega}^{\times} & -I \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \tilde{a} \\ \tilde{b} \end{bmatrix} + \begin{bmatrix} K_q \\ K_b \end{bmatrix} \tilde{y} + \begin{bmatrix} n_1 \\ n_2 \end{bmatrix} \tag{3.17}$$

and the linearized measurement error equation is expressed as

$$\tilde{y}_i = C_i \begin{bmatrix} \tilde{a} \\ \tilde{b} \end{bmatrix}, \quad C_i = \begin{bmatrix} C_{ai} & 0 \end{bmatrix} = \begin{bmatrix} \hat{y}_i^{\times} & 0 \end{bmatrix} \tag{3.18}$$

Summarized, the A and C matrices are a result of the linearization of error quaternions and are given as

$$A = \begin{bmatrix} -\hat{\omega}^{\times} & -I \\ 0 & 0 \end{bmatrix} \tag{3.19}$$

$$C = \begin{bmatrix} \hat{y}_1^{\times} & 0 \\ \vdots & \\ \hat{y}_m^{\times} & 0 \end{bmatrix} \tag{3.20}$$

Further, the innovation is described as

$$\delta = \sum_{i=1} C_i^{\mathrm{T}} R_c^{-1} \tilde{y}_i = \sum_{i=1} \hat{y}_i^{\times} R_c^{-1} \left(\hat{y}_i - y_i\right) \tag{3.21}$$

and the covariance of the innovation S is

$$S = C^{\mathrm{T}} R_C^{-1} C = \sum_{i=1} \left(\hat{y}_i^{\times}\right)^{\mathrm{T}} R_c^{-1} \hat{y}_i^{\times} \tag{3.22}$$

3.1.3. Covariance propagation

The covariance matrix follows [20]

$$P_k = \begin{bmatrix} P_{a,k} & P_{c,k} \\ P_{c,k}^{\mathrm{T}} & P_{b,k} \end{bmatrix} \tag{3.23}$$

and consists of the gains P_a, P_b and P_c. These gains are updated from the following equations

$$P_{a,k} = P_{a,k-1} + h\left(\mathbb{P}\left(P_{a,k-1}\hat{\omega}^{\times} - P_{c,k-1}\right) + Q_a - P_{a,k-1} S P_{a,k-1}\right) \tag{3.24}$$
$$P_{b,k} = P_{b,k-1} + h\left(Q_b - P_{c,k-1}^{\mathrm{T}} S P_{c,k-1}\right) \tag{3.25}$$
$$P_{c,k} = P_{c,k-1} + h\left(-\hat{\omega}^{\times} P_{c,k-1} - P_{b,k-1} + Q_c - P_{a,k-1} S P_{c,k-1}\right) \tag{3.26}$$

3.1.4. MEKF Algorithm

Algorithm 1 MEKF signal Algorithm

Initialize Q_a, Q_b, Q_c and R_C
Initialize q_0, b_0 and P_{a0}, P_{b0}, P_{c0}
loop
$\hat{\omega} = \omega_m - \hat{b}_{k-1}$
$\hat{y}_i = \hat{R}^{\mathrm{T}} d_i^n$
$S = C^{\mathrm{T}} R_C^{-1} C = \sum_{i=1} \left(\hat{y}_i^{\times}\right)^{\mathrm{T}} R_c^{-1} \hat{y}_i^{\times}$
$\delta = \sum_{i=1} \hat{y}_i^{\times} R_c^{-1} \left(\hat{y}_i - y_i\right)$

$\hat{q}_k = \hat{q}_{k-1} \circ \exp\left(0.5h\left(\hat{\omega} + P_{a,k-1}\delta\right)\right)$
$\hat{b}_k = \hat{b}_{k-1} + h\left(P_{c,k-1}^{\mathrm{T}}\delta\right)$

$P_{a,k} = P_{a,k-1} + h\left(2\mathbb{P}\left(P_{a,k-1}\hat{\omega}^{\times} - P_{c,k-1}\right) + Q_a - P_{a,k-1} S P_{a,k-1}\right)$
$P_{b,k} = P_{b,k-1} + h\left(Q_b - P_{c,k-1}^{\mathrm{T}} S P_{c,k-1}\right)$
$P_{c,k} = P_{c,k-1} + h\left(-\hat{\omega}^{\times} P_{c,k-1} - P_{b,k-1} + Q_c - P_{a,k-1} S P_{c,k-1}\right)$

$P_k = \begin{bmatrix} P_{a,k} & P_{c,k} \\ P_{c,k}^{\mathrm{T}} & P_{b,k} \end{bmatrix}$
end loop

3.2. Right Invariant Extended Kalman Filter - RIEKF

The RIEKF is closely related to the MEKF in that they both use an invariant output error. Unlike the MEKF, the RIEKF uses the right-invariant error, hence its name. The main benefit of this modification is that the matrices A and C are constant on a much more extensive set of trajectories [7], which may lead to better accuracy and less computational power. The presented equations and algorithm for the RIEKF in this chapter are inspired by [7].

Given that the RIEKF and the MEKF both use an invariant output error, the implementation of the RIEKF is to some degree similar to the MEKF. Therefore, the identical formulas for both filters are not included in this section, as they have already been specified in section 3.1.

The state propagation of the RIEKF is given by

$$\dot{q} = \frac{1}{2} q \circ (\omega_m - b) + n_q \circ q \tag{3.27}$$

$$\dot{b} = q^{-1} \circ n_b \circ q \tag{3.28}$$

where n_q and n_b are white noise vectors in the spatial frame, whereas the noise from the system dynamics of the MEKF are in the body frame.

3.2.1. Time propagation of state estimate

The estimate of measurement i is

$$\hat{y}_i = \hat{q}^{-1} \circ d_i \circ \hat{q} \tag{3.29}$$

Similar to the MEKF, the estimate of the gyroscope bias is defined as

$$\hat{b}_k = \hat{b}_{k-1} + h \left(\hat{q}^{-1} \circ (K_b E) \circ \hat{q} \right) \tag{3.30}$$

where

$$K_b E = P_{c,k-1}^{\mathrm{T}} \delta \tag{3.31}$$

and unlike the MEKF, the estimated measurement error for the RIEKF, E, is defined as

$$E = \hat{q} \circ (y - \hat{y}) \circ \hat{q}^{-1} = \tilde{q}^{-1} \circ (d + w_y) \circ \tilde{q} - d \tag{3.32}$$

where d and w_y are a known vector and white noise in the spatial frame n, respectively.

3.2.2. Eror equations and linearization of error quaternions

Considering unit quaternions, the error equation for the left invariant, LIEKF, is

$$\tilde{q}_l = \hat{q}^{-1} \circ q \tag{3.33}$$

which is the same error equation used in the MEKF. Whereas the right invariant, RIEKF, the error equation is given as

$$\tilde{q}_r = q \circ \hat{q}^{-1} \tag{3.34}$$

Then the error equations for the RIEKF is

$$\tilde{q} = q \circ \hat{q}^{-1} \tag{3.35}$$
$$\tilde{b} = q \circ (b - \hat{b}) \circ q^{-1} \tag{3.36}$$

Next, the kinematic differential equation of the error quaternion is

$$\begin{aligned}
\dot{\tilde{q}} &= \dot{q} \circ \hat{q}^{-1} - q \circ \hat{q}^{-1} \circ \dot{\hat{q}} \circ \hat{q}^{-1} \\
&= \frac{1}{2} q \circ (\omega_m - b) \circ \hat{q}^{-1} - q \circ \hat{q}^{-1} \circ \frac{1}{2} \hat{q} \circ \left(\omega_m - \hat{b}\right) \circ \hat{q}^{-1} - q \circ \hat{q}^{-1} \circ K_q E \circ \hat{q} \circ \hat{q}^{-1} \\
&= \frac{1}{2} q \circ (\omega_m - b)\, \hat{q}^{-1} - \frac{1}{2} q \circ \left(\omega_m - \hat{b}\right) \hat{q}^{-1} - \tilde{q}^{-1} \circ K_q E \\
&= \underbrace{\frac{1}{2} \tilde{b}^{\mathrm{T}} \tilde{\epsilon}}_{\text{Scalar}} - \frac{1}{2} \tilde{\eta} \tilde{b} - \frac{1}{2} \tilde{b}^{\times} \tilde{\epsilon} + \underbrace{\tilde{q}^{\mathrm{T}} K_q E}_{\text{Scalar}} - \tilde{\eta} K_q E - \tilde{\epsilon}^{\times} K_q E
\end{aligned} \tag{3.37}$$

where the vector part of the equation is

$$\dot{\tilde{\epsilon}} = -\frac{1}{2} \tilde{\eta} \tilde{b} - \frac{1}{2} \tilde{b}^{\times} \tilde{\epsilon} - \tilde{\eta} K_q E - \tilde{\epsilon}^{\times} K_q E \tag{3.38}$$

The kinematic differential equation of the bias error is

$$\begin{aligned}
\dot{\tilde{b}} &= \dot{q} \circ (b - \hat{b}) \circ q^{-1} + q \circ (\dot{b} - \dot{\hat{b}}) \circ q^{-1} - q \circ (b - \hat{b}) \circ q^{-1} \circ \dot{q} \circ q^{-1} \\
&= \frac{1}{2} q \circ \hat{\omega} \circ (b - \hat{b}) \circ q^{-1} + q \circ \left(q^{-1} M_\omega w_\omega \circ q - \hat{q}^{-1} \circ K_\omega E \circ \hat{q}\right) \circ q^{-1} \\
&- \frac{1}{2} q \circ (b - \hat{b}) \circ q^{-1} \circ q \circ \hat{\omega} \circ q^{-1} \\
&= \frac{1}{2} \left(\tilde{q} \circ \left(\hat{q} \circ \hat{\omega} \circ \hat{q}^{-1}\right) \circ \tilde{q}^{-1}\right)^{\times} \tilde{b} + M_\omega w - \tilde{q} \circ K_\omega E \circ \tilde{q}^{-1}
\end{aligned} \tag{3.39}$$

using the common state variable $\tilde{a} = 2\tilde{\epsilon}$ the linearization of the error equations leads to the linearized model

$$\frac{\mathrm{d}}{\mathrm{d}t} \begin{bmatrix} \tilde{a} \\ \tilde{b} \end{bmatrix} = \begin{bmatrix} 0 & -I \\ 0 & \hat{\Omega}^{\times} \end{bmatrix} \begin{bmatrix} \tilde{a} \\ \tilde{b} \end{bmatrix} \tag{3.40}$$
$$E_i = \begin{bmatrix} d^{\times} & 0 \end{bmatrix} \begin{bmatrix} \tilde{a} \\ \tilde{b} \end{bmatrix} \tag{3.41}$$

which gives the linearized matrices

$$A = \begin{bmatrix} 0 & -I \\ 0 & \hat{\Omega}^{\times} \end{bmatrix} \tag{3.42}$$
$$C_i = \begin{bmatrix} d_i^{\times} & 0 \end{bmatrix} \tag{3.43}$$

It is noted that C is constant and that A consists only of $\hat{\Omega}^{\times}$ which is not constant, as $\hat{\Omega} = \hat{q} \circ \hat{\omega} \circ \hat{q}^{-1}$. Next, the innovation is

$$\delta = \sum_{i=1} C_i^{\mathrm{T}} R_c^{-1} E_i = \sum_{i=1} \left(d_i^{\times}\right)^{\mathrm{T}} R_c^{-1} \left(d_i - \hat{q} \circ y_i \circ \hat{q}^{-1}\right) \tag{3.44}$$

where the covariance of the innovation is

$$S = C^{\mathrm{T}} R_C^{-1} C = \sum_{i=1} \left(d_i^{\times}\right)^{\mathrm{T}} R_c^{-1} d_i^{\times} \tag{3.45}$$

3.2.3. Covariance propagation

Finally the time propagation of the gains are expressed as

$$P_{a,k} = P_{a,k-1} + h\left(-2\mathbb{P}\left(P_{c,k-1}\right) + Q_a - P_{a,k-1} S P_{a,k-1}\right) \tag{3.46}$$

$$P_{b,k} = P_{b,k-1} + h\left(2\mathbb{P}\left(\hat{\Omega}^{\times} P_{b,k-1}\right) + Q_b - P_{c,k-1}^{\mathrm{T}} S P_{c,k-1}\right) \tag{3.47}$$

$$P_{c,k} = P_{c,k-1} + h\left(-P_{c,k-1}\hat{\Omega}^{\times} - P_{b,k-1} + Q_c - P_{a,k-1} S P_{c,k-1}\right) \tag{3.48}$$

3.2.4. RIEKF Algorithm

Algorithm 2 RIEKF signal Algorithm

Initialize Q_a, Q_b, Q_c and R_C
Initialize q_0, b_0 and P_{a0}, P_{b0}, P_{c0}
loop

$\hat{\omega} = \omega_m - \hat{b}_{k-1}$
$\hat{y}_i = \hat{q}^{-1} \circ d_i \circ \hat{q}$
$\delta = \sum_{i=1} \left(d_i^{\times}\right)^{\mathrm{T}} R_c^{-1} \left(d_i - \hat{q} \circ y_i \circ \hat{q}^{-1}\right)$
$S = \sum_{i=1} \left(d_i^{\times}\right)^{\mathrm{T}} R_c^{-1} d_i^{\times}$

$\hat{q}_k = \hat{q}_{k-1} \circ \exp\left(0.5h\left(\hat{\omega} + P_{a,k-1}\delta\right)\right)$
$\hat{b}_k = \hat{b}_{k-1} + h\left(\hat{q}^{-1} \circ \left(P_{c,k-1}^{\mathrm{T}}\delta\right) \circ \hat{q}\right)$

$P_{a,k} = P_{a,k-1} + h\left(-2\mathbb{P}\left(P_{c,k-1}\right) + Q_a - P_{a,k-1} S P_{a,k-1}\right)$
$P_{b,k} = P_{b,k-1} + h\left(2\mathbb{P}\left(\hat{\Omega}^{\times} P_{b,k-1}\right) + Q_b - P_{c,k-1}^{\mathrm{T}} S P_{c,k-1}\right)$
$P_{c,k} = P_{c,k-1} + h\left(-P_{c,k-1}\hat{\Omega}^{\times} - P_{b,k-1} + Q_c - P_{a,k-1} S P_{c,k-1}\right)$

$P_k = \begin{bmatrix} P_{a,k} & P_{c,k} \\ P_{c,k}^{\mathrm{T}} & P_{b,k} \end{bmatrix}$

end loop

Chapter 4.

Pose Estimation

This chapter is based on [2], and the autonomous dynamics of the errors is examined to determine the convergence properties of the IEKF around any trajectory. The logarithm of the left- and right-invariant is introduced, where the error dynamics are linearized in terms of the logarithm. Then, the general structure of the IEKF is provided.

Using this theoretical information, two different cases from [2], simple car model and navigation on flat earth, are presented where the left invariant extended Kalman filter (LIEKF) and the right invariant extended Kalman filter (RIEKF) are compared against the EKF and its modification MEKF, respectively.

4.1. Autonomous error dynamics

Following Definition 1, this section serves as a proof of the autonomous error dynamics for the IEKF, as presented in [2].

A matrix lie group is defined as $G \in \mathbb{R}^{N \times N}$ so that the lie algebra is denoted as $\mathfrak{g}$ where $\mathfrak{g} \in \mathbb{R}^{\dim \mathfrak{g}}$. A noise-free dynamics is considered

$$\dot{\chi} = f_u(\chi) \tag{4.1}$$

where χ is the state which lies in the lie group G and u is an input variable so that $f_u(\chi) = f(\chi, u)$.

Two distinct trajectories χ and $\bar{\chi}$ are considered based on Equation 4.1, which means that $\dot{\bar{\chi}} = f_u(\bar{\chi})$. Then the error between the trajectories can be be defined with a left-invariant and right-invariant error as

$$\tilde{\chi}^L = \chi^{-1}\bar{\chi} \qquad \text{(Left-invariant)} \tag{4.2}$$

$$\tilde{\chi}^R = \bar{\chi}\chi^{-1} \qquad \text{(Right-invariant)} \tag{4.3}$$

where the left-invariant error satisfies $\bar{\chi} = \chi\tilde{\chi}^L$ and the right-invariant satisfies $\bar{\chi} = \tilde{\chi}^R\chi$.

Definition 1 *The left-invariant and right-invariant errors are said to have a state-trajectory independent propagation if they satisfy a differential equation of the form $\dot{\tilde{\chi}} = g_u(\tilde{\chi})$[2].*

A class of dynamic system is considered, which satisfies

$$f_u(ab) = f_u(a)b + af_u(b) - af_u(\mathrm{id})b \tag{4.4}$$

for all $a, b \in G$, where id denotes the identity matrix. Then the error dynamics from Equation 4.2 and Equation 4.3 are autonomous. This means that the system has state trajectory independent error propagation property and the left- and right-invariant error dynamics writes

$$\dot{\tilde{\chi}}^L = g_u^L\left(\tilde{\chi}^L\right) = f_u\left(\tilde{\chi}^L\right) - f_u(\mathrm{id})\tilde{\chi}^L \tag{4.5}$$

$$\dot{\tilde{\chi}}^R = g_u^R\left(\tilde{\chi}^R\right) = f_u\left(\tilde{\chi}^R\right) - \tilde{\chi}^R f_u(\mathrm{id}) \tag{4.6}$$

The proof can be derived by considering the left-invariant error dynamic

$$\begin{aligned}\dot{\tilde{\chi}}^L = g_u^L\left(\tilde{\chi}^L\right) &= \frac{d}{dt}\left(\chi^{-1}\bar{\chi}\right)\\ &= -\chi^{-1}\dot{\chi}\chi^{-1}\bar{\chi} + \chi^{-1}\dot{\bar{\chi}}\\ &= -\chi^{-1}f_u(\chi)\tilde{\chi}^L + \chi^{-1}f_u(\chi\tilde{\chi}^L)\end{aligned} \tag{4.7}$$

which has to hold for any χ and $\tilde{\chi}^L$, particularly for $\chi = \mathrm{id}$. This gives

$$g_u^L\left(\tilde{\chi}^L\right) = f_u\left(\tilde{\chi}^L\right) - f_u(\mathrm{id})\tilde{\chi}^L \tag{4.8}$$

Finally, reinjecting Equation 4.8 into Equation 4.7 gives

$$\begin{aligned}f_u\left(\tilde{\chi}^L\right) - f_u(\mathrm{id})\tilde{\chi}^L &= -\chi^{-1}f_u(\chi)\tilde{\chi}^L + \chi^{-1}f_u(\chi\tilde{\chi}^L)\\ f_u(\chi\tilde{\chi}^L) &= \chi f_u\left(\tilde{\chi}^L\right) - \chi f_u(\mathrm{id})\tilde{\chi}^L + f_u(\chi)\tilde{\chi}^L\\ &= f_u(\chi)\tilde{\chi}^L + \chi f_u(\tilde{\chi}^L) - \chi f_u(\mathrm{id})\tilde{\chi}^L\end{aligned} \tag{4.9}$$

and it is concluded that $f_u(\chi\tilde{\chi}^L) = f_u(\chi)\tilde{\chi}^L + \chi f_u(\tilde{\chi}^L) - \chi f_u(\mathrm{id})\tilde{\chi}^L$ satisfies Equation 4.4. The proof for right-invariant errors is analogous and straightforward.

4.1.1. Autonomous error dynamics formulated in $SE(2)$

Two trajectories in $SE(2)$ are considered as χ and $\bar{\chi}$ for the dynamics

$$\dot{\chi} = \chi\nu \tag{4.10}$$

where χ is the state in $SE(2)$ and ν is the logarithm of $\mu = (\omega, v, 0)^T$

$$\chi = \begin{pmatrix} R & x \\ 0_{1\times 2} & 1 \end{pmatrix} \tag{4.11}$$

$$\nu = \mathscr{L}_{\mathfrak{se}(2)}(\mu) = \begin{pmatrix} 0 & -\omega & v \\ \omega & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{4.12}$$

The dynamics of the left-invariant error, $\tilde{\chi}^L = \chi^{-1}\bar{\chi}$, is given as

$$\dot{\tilde{\chi}}^L = \tilde{\chi}^L\nu - \nu\tilde{\chi}^L \tag{4.13}$$

An error variable ξ is defined representing the left-invariant error, so that it satisfies $\tilde{\chi}^L = \exp(\xi)$. Then the error dynamics gives

$$\begin{aligned}\dot{\xi} &= J_R^{-1}(\mathrm{ad}(\xi))\mu - J_L^{-1}(\mathrm{ad}(\xi))\mu\\ &= \left(J_R^{-1}(\mathrm{ad}(\xi)) - J_L^{-1}(\mathrm{ad}(\xi))\right)\mu\end{aligned} \tag{4.14}$$

where $J_L^{-1}(\cdot)$ and $J_R^{-1}(\cdot)$ denotes the inverse of the left and right Jacobian, respectively, and are defined as

$$J_L(\mathrm{ad}(\mu))^{-1} = \sum_{k=0}^{\infty} \frac{B_k}{k!}(\mathrm{ad}(\mu))^k \tag{4.15}$$

$$J_R(\mathrm{ad}(\mu))^{-1} = \sum_{k=0}^{\infty} \frac{(-1)^k B_k}{k!}(\mathrm{ad}(\mu))^k \tag{4.16}$$

with B_k being the Bernoulli numbers.

It is observed that the terms of series for the left and right Jacobian are identical, except for the first order term, which gives the formulation

$$J_R^{-1}(\mathrm{ad}(\xi)) - J_L^{-1}(\mathrm{ad}(\xi)) = \mathrm{ad}(\xi) \tag{4.17}$$

and using this formulation in Equation 4.14 gives

$$\dot{\xi} = \mathrm{ad}(\xi)\mu \tag{4.18}$$

which yields the linear dynamics

$$\dot{\xi} = -\,\mathrm{ad}(\mu)\xi = A\xi \tag{4.19}$$

4.2. Log-linear property

This chapter serves as a proof of the fact that the time propagation of the logarithm of the error will be linear, provided that the left-invariant error or the right-invariant error satisfies Equation 4.5 or Equation 4.6. The presented proof in this chapter is based on [2], where a more detailed version can be found.

The system dynamic from Equation 4.1 and the condition

$$g_u(ab) = a g_u(b) + g_u(a) b, \quad a, b \in G \tag{4.20}$$

is considered. Then the functions g_u, which governs the errors propagation, has the following properties

$$g_u^L\left(\tilde{\chi}^L\right) = f_u\left(\tilde{\chi}^L\right) - f_u(\mathrm{id})\tilde{\chi}^L \tag{4.21}$$

$$g_u^R\left(\tilde{\chi}^R\right) = f_u\left(\tilde{\chi}^R\right) - \tilde{\chi}^R f_u(\mathrm{id}) \tag{4.22}$$

where the left-invariant error satisfies $\dot{\tilde{\chi}}^L = g_u^L\left(\tilde{\chi}^L\right)$ and the right-invariant error satisfies $\dot{\tilde{\chi}}^R = g_u^R\left(\tilde{\chi}^R\right)$. The verification of these properties are straightforward.

To show that the error has log-linear property, a variable which defines a solution at time t corresponding to a given initial condition is defined as Φ_t. This is the flow associated with the system $(d/dt)\tilde{\chi}_t = g_u(\tilde{\chi}_t)$ which satisfies the condition from Equation 4.20.

Two variables denoted $\Phi_t(X_0)$ and $\Phi_t(Y_0)$ with initial conditions X_0 and Y_0 respectively, are defined as the solutions of

$$\dot{X} = g_u(X) \tag{4.23}$$

$$\dot{Y} = g_u(Y) \tag{4.24}$$

and a variable $\Phi_t(Z_0)$ with initial condition $Z_0 = X_0Y_0$ is defined as a solution of

$$\dot{Z} = g_u(Z) \tag{4.25}$$

The goal is to see if $\Phi_t(X_0)\Phi_t(Y_0)$ is a solution of the system $(d/dt)\tilde{\chi}_t = g_u(\tilde{\chi}_t)$, which gives

$$\begin{aligned} \frac{d}{dt}\Phi_t(X_0)\Phi_t(Y_0) &= \Phi_t(X_0)\frac{d}{dt}\Phi_t(Y_0) + \frac{d}{dt}\ \Phi_t(X_0)\Phi_t(Y_0) \\ &= \Phi_t(X_0)g_u\left(\Phi_t(Y_0)\right) + g_u\left(\Phi_t(X_0)\right)\Phi_t(Y_0) \\ &= g_u\left(\Phi_t(X_0)\Phi_t(Y_0)\right) \end{aligned} \tag{4.26}$$

Here it is seen that $\Phi_t(X_0)\Phi_t(Y_0)$ is a solution of $\frac{d}{dt}\Phi_t(X_0)\Phi_t(Y_0)$ with initial condition X_0Y_0. This also suggests that

$$\Phi_t(Z_0) = \Phi_t(X_0Y_0) = \Phi_t(X_0)\Phi_t(Y_0) \tag{4.27}$$

Further, a case is considered where $\Phi_t(Y_0) = \Phi_t(X_0)$ which gives $\Phi_t(Z_0) = \Phi_t(X_0)^2$ with an initial condition $Z_0 = X_0^2$. This means that $\Phi_t(X_0)^2$ is a solution $(d/dt)X^2 = g_u(X^2)$.

Two variables $\mathscr{L}_{\mathfrak{g}}(\xi)$ and $\mathscr{L}_{\mathfrak{g}}(\zeta)$ are defined as the logarithm of X and Z, respectively, and are denoted as $\xi^\wedge$ and $\zeta^\wedge$. This gives

$$\exp(\zeta^\wedge) = \exp(\xi^\wedge)\exp(\xi^\wedge) \tag{4.28}$$

and the Baker-Campbell-Hausdorff formula [11] gives

$$\zeta = 2\xi \tag{4.29}$$

Since $\Phi_t(Z_0)$ and $\Phi_t(X_0)$ satisfies $\dot{Z} = g_u(Z)$ and $\dot{X} = g_u(X)$, then $\Phi_t(\zeta_0) = 2\Phi_t(\xi_0)$ will satisfy the same dynamics for $\Phi_t(\xi_0)$. This means that the initial conditions are $\xi_0 = \log(X_0)$, $\zeta_0 = \log(Z_0)$ where $\zeta_0 = 2\xi_0$. Further, this can be generalized by defining $\Phi_t(Y_0) = \exp((1-\alpha)\xi^\wedge)$ which gives $\zeta = \alpha\xi$ for any rational α. This suggests that the dynamics of $\Phi_t(\xi_0)$ is homogeneous, and given that the system is autonomous, it is concluded that the dynamics of the logarithm is linear and can be expressed as

$$\frac{d}{dt}\Phi_t(\xi_0) = A_t\Phi_t(\xi_0) \tag{4.30}$$

4.3. Invariant EKF for right and left observations

This section provides an introduction to the IEKF general structure in continuous-discrete time and shows that the IEKF is a nonlinear observer with local convergence properties around any trajectory, which is a rare feature when it comes to nonlinear observers.

From [2] a system is considered as

$$\dot{\chi}_t = f_u(\chi_t) \tag{4.31}$$

where $\chi_t \in G$ and

$$f_u(ab) = f_u(a)b + af_u(b) - af_u(\mathrm{id})b \tag{4.32}$$

Further, two kinds of observations corresponding to this system, left-invariant observations and right-invariant observations, are derived in the next sections.

4.3.1. Left-invariant observations

For the left-invariant observations, the measurements are considered as

$$y_k^n = \chi_k d^n \tag{4.33}$$

where d^n are known vectors.

Then the left-invariant EKF is given by

$$\dot{\hat{\chi}}_t = f_u(\hat{\chi}_t), \quad k-1 \leq t < k \tag{4.34}$$

$$\hat{\chi}_{k|k} = \hat{\chi}_{k|k-1} \exp\left(L_n\left(\hat{\chi}_{k|k-1}^{-1} y_k^n - d^n\right)\right) \tag{4.35}$$

The left invariant error is defined as

$$\tilde{\chi}_t^L = \chi_t^{-1}\hat{\chi}_t \tag{4.36}$$

where χ and $\hat{\chi}$ are trajectories of the system Equation 4.31 in the interval $k-1 \leq t < k$ of the propagation. This suggests that the left-invariant error dynamics is autonomous in this time interval, and is defined as

$$\dot{\tilde{\chi}}_t^L = g_u^L(\tilde{\chi}_t^L) \tag{4.37}$$

The logarithm of the left-invariant error $\xi_t^\wedge$ satisfies the linear dynamics

$$\frac{d}{dt}\xi_t = A_t \xi_t \tag{4.38}$$

where A_t is to be computed.

Further, the update of the left-invariant error is defined by $\tilde{\chi}_{k|k}^L = \chi_k^{-1}\hat{\chi}_{k|k}$, which writes

$$\tilde{\chi}_{k|k}^L = \tilde{\chi}_{k|k-1}^L \exp\left(L_n\left((\tilde{\chi}_{k|k-1}^L)^{-1} d^n - d^n\right)\right) \tag{4.39}$$

and it is seen that the update of the error does not depend on the trajectory of the system.

4.3.2. Right-invariant observations

The measurements for right-invariant observations are considered as

$$y_k^n = \chi_k^{-1} d^n \tag{4.40}$$

Then the right-invariant EKF is given by

$$\dot{\hat{\chi}}_t = f_u(\hat{\chi}_t), \quad k-1 \leq t < k \tag{4.41}$$

$$\hat{\chi}_{k|k} = \exp\left(L_n\left(\hat{\chi}_{k|k-1} y_k^n - d^n\right)\right)\hat{\chi}_{k|k-1} \tag{4.42}$$

The right-invariant error is defined as

$$\tilde{\chi}_t^R = \hat{\chi}_t \chi_t^{-1} \tag{4.43}$$

which has the autonomous dynamics

$$\tilde{\chi}_t^R = g_u(\tilde{\chi}_t^R) \tag{4.44}$$

and exactly like the left-invariant observations, the logarithm of the right-invariant error $\xi_t^\wedge$ has linear dynamics.

Further, the update of the right-invariant error is defined by $\tilde{\chi}_{k|k}^R = \hat{\chi}_{k|k}\chi_k^{-1}$, which writes

$$\tilde{\chi}_{k|k}^R = \exp\left(L_n\left(\tilde{\chi}_{k|k-1}^R d^n - d^n\right)\right)\tilde{\chi}_{k|k-1}^R \tag{4.45}$$

and it is seen that the update of the error does not depend on the trajectory of the system.

The next sections in this chapter presents two different cases where the LIEKF and the RIEKF are built upon the theoretical information and proof provided from the current and previous sections.

4.4. Simple car model

In this section, the Extended Kalman Filter (EKF) and the Left Invariant Extended Kalman Filter (LIEKF) are presented in relation to pose estimation using a simple car model in the Euclidean space.

The system dynamics for the simple car model is given as

$$\dot{\theta} = \omega_k \tag{4.46}$$
$$\dot{x}^{(1)} = \cos(\theta)v_k \tag{4.47}$$
$$\dot{x}^{(2)} = \sin(\theta)v_k \tag{4.48}$$

where θ and x is the heading and position of the robot, respectively, v_k is the velocity in along the x-axis and w_k is the angular velocity. In discrete time the model can be written as

$$\theta_k = \theta_{k-1} + h\omega_k \tag{4.49}$$
$$x_k^{(1)} = x_{k-1}^{(1)} + h\cos\left(\theta_{k-1}\right)v_k \tag{4.50}$$
$$x_k^{(2)} = x_{k-1}^{(2)} + h\sin\left(\theta_{k-1}\right)v_k \tag{4.51}$$

4.4.1. Extended Kalman Filter - EKF

The equations and algorithm for the EKF is mainly based on [1] and inspired by [3]. It is noted that the EKF error system is linear, and is defined as $\tilde{X} = X - \hat{X}$ as in [1].

The state vector for the EKF is defined as

$$X = (\theta, x)\,, \quad \theta \in \mathbb{R}, x \in \mathbb{R}^2 \tag{4.52}$$

The propagation of the estimates is defined as

$$\hat{\theta}_{k|k-1} = \hat{\theta}_{k-1|k-1} + h\omega_k \tag{4.53}$$
$$\hat{x}_{k|k-1}^{(1)} = \hat{x}_{k-1|k-1}^{(1)} + h\cos\left(\hat{\theta}_{k-1|k-1}\right)v_k \tag{4.54}$$
$$\hat{x}_{k|k-1}^{(2)} = \hat{x}_{k-1|k-1}^{(2)} + h\sin\left(\hat{\theta}_{k-1|k-1}\right)v_k \tag{4.55}$$

and the measurement and the estimated measurement is defined as

$$y_k = x_k \tag{4.56}$$
$$\hat{y}_k = \hat{x}_{k|k-1} \tag{4.57}$$

The propagated state in vector form is

$$\hat{X}_{k|k-1} = \left(\hat{\theta}_{k|k-1}, \hat{x}^1_{k|k-1}, \hat{x}^2_{k|k-1}\right) \tag{4.58}$$

Error equations and linearization

The estimation errors are given as

$$\tilde{\theta} = \theta - \hat{\theta} \tag{4.59}$$
$$\tilde{x}^{(1)} = x^{(1)} - \hat{x}^{(1)} \tag{4.60}$$
$$\tilde{x}^{(2)} = x^{(2)} - \hat{x}^{(2)} \tag{4.61}$$

so using Equation 4.50 and Equation 4.54 gives

$$\begin{aligned}
\tilde{x}^{(1)}_{k|k-1} &= x^{(1)}_k - \hat{x}^{(1)}_{k|k-1} \\
&= x^{(1)}_{k-1} + h\cos\left(\theta_{k-1}\right) v_k - \left(\hat{x}^{(1)}_{k-1|k-1} + h\cos\left(\hat{\theta}_{k-1|k-1}\right) v_k\right) \\
&= \tilde{x}^{(1)}_{k-1|k-1} + h\left(\cos\left(\theta_{k-1}\right) - \cos\left(\hat{\theta}_{k-1|k-1}\right)\right) v_k \\
&= \tilde{x}^{(1)}_{k-1|k-1} + h\left(\cos\left(\tilde{\theta}_{k-1|k-1} + \hat{\theta}_{k-1|k-1}\right) - \cos\left(\hat{\theta}_{k-1|k-1}\right)\right) v_k \\
&\approx \tilde{x}^{(1)}_{k-1|k-1} + h\tilde{\theta}_{k-1|k-1}\frac{\mathrm{d}}{\mathrm{d}\theta}\cos(\theta)\Big|_{\hat{\theta}_{k-1|k-1}} v_k \\
&= \tilde{x}^{(1)}_{k-1|k-1} + h\tilde{\theta}_{k-1|k-1}\left(-\sin\left(\hat{\theta}_{k-1|k-1}\right)\right) v_k
\end{aligned} \tag{4.62}$$

and using Equation 4.51 and Equation 4.55 gives

$$\begin{aligned}
\tilde{x}^{(2)}_{k|k-1} &= x^{(2)}_k - \hat{x}^{(2)}_{k|k-1} \\
&= x^{(2)}_{k-1} + h\sin\left(\theta_{k-1}\right) v_k - \left(\hat{x}^{(2)}_{k-1|k-1} + h\sin\left(\hat{\theta}_{k-1|k-1}\right) v_k\right) \\
&= \tilde{x}^{(2)}_{k-1|k-1} + h\left(\sin\left(\theta_{k-1}\right) - \sin\left(\hat{\theta}_{k-1|k-1}\right)\right) v_k \\
&= \tilde{x}^{(2)}_{k-1|k-1} + h\left(\sin\left(\tilde{\theta}_{k-1|k-1} + \hat{\theta}_{k-1|k-1}\right) - \sin\left(\hat{\theta}_{k-1|k-1}\right)\right) v_k \\
&\approx \tilde{x}^{(2)}_{k-1|k-1} + h\tilde{\theta}_{k-1|k-1}\frac{\mathrm{d}}{\mathrm{d}\theta}\sin(\theta)\Big|_{\hat{\theta}_{k-1|k-1}} v_k \\
&= \tilde{x}^{(2)}_{k-1|k-1} + h\tilde{\theta}_{k-1|k-1}\cos\left(\hat{\theta}_{k-1|k-1}\right) v_k
\end{aligned} \tag{4.63}$$

Further, the error equations are summarized as

$$\tilde{\theta}_{k|k-1} = \tilde{\theta}_{k-1|k-1} \tag{4.64}$$
$$\tilde{x}^{(1)}_{k|k-1} = \tilde{x}^{(1)}_{k-1|k-1} + h\tilde{\theta}_{k-1|k-1}\left(-\sin\left(\hat{\theta}_{k-1|k-1}\right)\right) v_k \tag{4.65}$$
$$\tilde{x}^{(2)}_{k|k-1} = \tilde{x}^{(2)}_{k-1|k-1} + h\tilde{\theta}_{k-1|k-1}\cos\left(\hat{\theta}_{k-1|k-1}\right) v_k \tag{4.66}$$

and the measurement estimation error is

$$\tilde{y}_k = y_k - \hat{y}_k \tag{4.67}$$

Then the linearized matrices at the estimates are obtained as

$$A_k = \begin{pmatrix} 0 & 0 & 0 \\ -\sin\left(\hat{\theta}_{k|k-1}\right) v_k & 0 & 0 \\ \cos\left(\hat{\theta}_{k|k-1}\right) v_k & 0 & 0 \end{pmatrix} \tag{4.68}$$

$$C_k = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{4.69}$$

Kalman gain and estimates

With the linearized error system, the covariance propagation is defined as

$$P_{k|k-1} = P_{k-1|k-1} + h\left(A_k P_{k-1|k-1} + P_{k-1|k-1} A_k^{\mathrm{T}} + \hat{Q}_k\right) \tag{4.70}$$

where the noise matrix $\hat{Q}_k$ given as:

$$\hat{Q}_k = Cov\left[\left(w_k^{\theta}, w_k^{l}, w_k^{tr}\right)^T\right] \tag{4.71}$$

with

$$Cov\left[\left(w_k^{\theta}, w_k^{l}, w_k^{tr}\right)^T\right] = Q_k = diag\left((\pi/180)^2, 10^{-4}, 10^{-4}\right) \tag{4.72}$$

Kalman gain is computed by

$$S = C_k P_{k|k-1} C_k^T + \hat{N}_k \tag{4.73}$$

$$L_k = P_{k|k-1} C_k^T S^{-1} \tag{4.74}$$

where the noise matrix $\hat{N}_k$ is given as

$$\hat{N}_k = R\left(\hat{\theta}_{k|k-1}\right) N_k R\left(\hat{\theta}_{k|k-1}\right)^T \tag{4.75}$$

with

$$N_k = I_2 \tag{4.76}$$

Then the covariance update is defined as

$$P_{k|k} = \left(I_3 - L_k C_k\right) P_{k|k-1} \tag{4.77}$$

and the state update

$$\hat{X}_{k|k} = \hat{X}_{k|k-1} + L_k\left(\tilde{y}_k\right) = \hat{X}_{k|k-1} + L_k\left(y_k - \hat{y}_k\right) \tag{4.78}$$

EKF algorithm

From the formulas presented in this chapter, the algorithm for the EKF can be described by a propagation and an update.

Algorithm 3 EKF - Simple car model

Initialize P_0 and $\hat{X}_0$
loop
Define A_k, C_k, as in Equation 4.68, Equation 4.69
Define Q_k, N_k as in Equation 4.72, Equation 4.76
Propagation
$\hat{\theta}_{k|k-1} = \hat{\theta}_{k-1|k-1} + h\omega_k$
$\hat{x}^{(1)}_{k|k-1} = \hat{x}^{(1)}_{k-1|k-1} + h\cos\left(\hat{\theta}_{k-1|k-1}\right) v_k$
$\hat{x}^{(2)}_{k|k-1} = \hat{x}^{(2)}_{k-1|k-1} + h\sin\left(\hat{\theta}_{k-1|k-1}\right) v_k$

$P_{k|k-1} = P_{k-1|k-1} + h\left(A_k P_{k-1|k-1} + P_{k-1|k-1} A_k^{\mathrm{T}} + Q_k\right)$
Update
$\hat{y}_k = \hat{x}_{k|k-1}$

$S = C_k P_{k|k-1} C_k^T + \hat{N}_k$
$L_k = P_{k|k-1} C_k^T S^{-1}$

$P_{k|k} = \left(I_3 - L_k C_k\right) P_{k|k-1}$
$\hat{X}_{k|k} = \hat{X}_{k|k-1} + L_k\left(y_k - \hat{y}_k\right)$
end loop

4.4.2. Left Invariant Extended Kalman Filter - LIEKF

The LIEKF equations and algorithm in this chapter are based on [2]. The EKF and the LIEKF are very similar to each other as the LIEKF is a modification of the EKF. The main difference between the mentioned filters is that for the LIEKF, a nonlinear error variable is chosen, which causes some of the error equations to differ. The lie group in $SE(2)$ will be implemented for this filter.

The state vector for the LIEKF is defined as

$$X = (\theta, x)\,, \quad \theta \in \mathbb{R}, x \in \mathbb{R}^2 \tag{4.79}$$

Recall from earlier in this chapter that the system dynamics in discrete time is defined as in Equation 4.49 - Equation 4.51 and that the time propagation of the state estimates is defined as in Equation 4.53 - Equation 4.55. Further, this system can be embedded in the matrix Lie group $SE(2)$ using the matrices introduced earlier in this thesis, from subsection 2.2.4. This gives

$$\begin{aligned}\hat{\chi}_{k-1|k-1} &= \begin{pmatrix} R\left(\hat{\theta}_{k-1|k-1}\right) & \hat{x}_{k-1|k-1} \\ 0_{1\times 2} & 1 \end{pmatrix} \\ &= \begin{pmatrix} \cos\left(\hat{\theta}_{k-1|k-1}\right) & -\sin\left(\hat{\theta}_{k-1|k-1}\right) & \hat{x}^{(1)}_{k-1|k-1} \\ \sin\left(\hat{\theta}_{k-1|k-1}\right) & \cos\left(\hat{\theta}_{k-1|k-1}\right) & \hat{x}^{(2)}_{k-1|k-1} \\ 0 & 0 & 1 \end{pmatrix}\end{aligned} \tag{4.80}$$

$$\nu_k = \mathscr{L}_{\mathfrak{se}(2)}(\mu_k) = \begin{pmatrix} 0 & -\omega_k & v_k \\ \omega_k & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{4.81}$$

where ν_k is the logarithm of $\mu_k = (w_k, v_k, 0)^T$.

The measurement is given in homogeneous form as

$$Y_k = \chi_{k|k-1} \begin{pmatrix} 0_{2\times 1} \\ 1 \end{pmatrix} \tag{4.82}$$

and the propagated state is defined as

$$\hat{\chi}_{k|k-1} = \hat{\chi}_{k-1|k-1} + h\left(\hat{\chi}_{k-1|k-1}\nu_k\right) \tag{4.83}$$

Error equations and linearization

The left invariant estimation error is

$$\tilde{\chi} = \chi^{-1}\hat{\chi} \in SE(2) \tag{4.84}$$

and the measurement estimation error is defined in homogeneous form as

$$\tilde{Y} = \chi^{-1}\hat{\chi} \begin{pmatrix} 0_{2\times 1} \\ 1 \end{pmatrix} = \tilde{\chi}^{-1} \begin{pmatrix} 0_{2\times 1} \\ 1 \end{pmatrix} \tag{4.85}$$

The propagation and update of the continuous-discrete LIEKF is

$$\dot{\hat{\chi}}_t = \hat{\chi}_t \nu_t \tag{4.86}$$

$$\hat{\chi}_{k|k} = \hat{\chi}_{k|k-1} \exp\left(\tilde{L}_k \tilde{Y}\right) \tag{4.87}$$

where $\tilde{L}_k$ is a reduced-dimension gain matrix defined by $\tilde{L}_k = L_k \tilde{p}$ with $\tilde{p} = (I_2, 0_{2,1})$. This results in the reduced-dimension gain matrix

$$\tilde{L}_k = (L_k, 0_{2,1}) \tag{4.88}$$

Then the error system dynamics is found by the time derivative of the corresponding dynamics of $\tilde{\chi}$, as

$$\begin{aligned} \dot{\tilde{\chi}} &= \chi^{-1}\dot{\hat{\chi}} + \dot{\chi}^{-1}\hat{\chi} \\ &= \chi^{-1}\dot{\hat{\chi}} - \chi^{-1}\dot{\chi}\chi^{-1}\hat{\chi} \\ &= \tilde{\chi}\nu^{\wedge} - \nu^{\wedge}\tilde{\chi} \end{aligned} \tag{4.89}$$

$$\tilde{\chi}_{k|k} = \chi_k^{-1}\hat{\chi}_{k|k} = \tilde{\chi}_{k|k-1} \exp\left(\tilde{L}_k \tilde{\chi}_{k|k-1}^{-1} \begin{pmatrix} 0_{2\times 1} \\ 1 \end{pmatrix}\right) \tag{4.90}$$

To linearize the error system, an error variable, ξ, is established representing the left invariant estimation error $\tilde{\chi}$ as

$$\xi = (\theta, \rho), \quad \theta \in \mathbb{R}, \rho \in \mathbb{R}^2 \tag{4.91}$$

which satisfies

$$\tilde{\chi} = \exp(\xi) = \begin{pmatrix} R(\theta) & E(\theta)\rho \\ 0_{1\times 2} & 0 \end{pmatrix} \tag{4.92}$$

From Equation 4.89, $\boldsymbol{\tilde{\chi}\nu - \nu\tilde{\chi}}$ implies that

$$\dot{\xi} = J_R(\xi)^{-1}\mu - J_L(\xi)^{-1}\mu = \mathrm{ad}(\xi)\mu = -\,\mathrm{ad}(\mu)\xi \tag{4.93}$$

yielding the first linearized equation.

Using $(\exp(u)^{-1} = \exp(-u))$, it is observed that

$$\tilde{\chi}^{-1} = \exp(-\xi) = \begin{pmatrix} R(-\theta) & -E(-\theta)\rho \\ 0_{1\times 2} & 1 \end{pmatrix} \tag{4.94}$$

which gives

$$\tilde{\chi}^{-1}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} -E(-\theta)\rho \\ 1 \end{pmatrix} \tag{4.95}$$

Since $E(\theta) \to I$ when $\theta \to 0$, the latter equation can be estimated by (for small θ)

$$\xi_k = \xi_{k-1} - L_n\rho \tag{4.96}$$

yielding the second linearization equation. It is noted that $\rho = C_k x = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} x$. The linearized error system is then summarized as

$$\dot{\xi}_t = -\operatorname{ad}(\mu)\xi_t = -\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -\omega_k \\ -v_k & \omega_k & 0 \end{pmatrix}\xi_t \tag{4.97}$$

$$\xi_{k|k} = \xi_{k|k-1} - L_n\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} x \tag{4.98}$$

which presents the linearized matrices

$$A_k = -\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -\omega_k \\ -v_k & \omega_k & 0 \end{pmatrix} \tag{4.99}$$

$$C_k = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{4.100}$$

Kalman gain, covariance matrix and state update

The kalman gain and the propagation and update of the covariance matrix for the LIEKF are computed exactly as for the EKF, and the equations are found from subsubsection 4.4.1. Unlike the EKF, the state update for the LIEKF is defined as

$$\hat{\chi}_{k|k} = \hat{\chi}_{k|k-1}\exp\left(\tilde{L}_k\left[\hat{\chi}_{k|k-1}^{-1}Y_k - \begin{pmatrix} 0_{2\times 1} \\ 1 \end{pmatrix}\right]\right) \tag{4.101}$$

It is noted that since the bottom element of $\hat{\chi}_{k|k-1}^{-1}Y_k - \begin{pmatrix} 0_{2\times 1} \\ 1 \end{pmatrix}$ is always zero, the reduced-dimension gain matrix $\tilde{L}_k$ is used. At last, the state update in vector form is defined as

$$\hat{X}_{k|k} = \left(\operatorname{Atan2}\left(\sin(\hat{\theta}_{k|k}), \cos(\hat{\theta}_{k|k})\right), \hat{x}_{k|k}^{(1)}, \hat{x}_{k|k}^{(2)}\right) \tag{4.102}$$

where $\hat{x}_{k|k}^{(1)}, \hat{x}_{k|k}^{(2)}$ is the updated estimated position of the robot and $\hat{\theta}_{k|k}$ is the updated estimated heading of the robot.

LIEKF algorithm

From the formulas presented in this chapter, the algorithm for the LIEKF can be described by a propagation and an update.

Algorithm 4 LIEKF - Simple car model

Initialize P_0 and $\hat{X}_0$

loop

Define A_k, C_k, as in Equation 4.99, Equation 4.100

Define Q_k, N_k as in Equation 4.72, Equation 4.76

Propagation

$\hat{\chi}_{k|k-1} = \hat{\chi}_{k-1|k-1} + h\left(\hat{\chi}_{k-1|k-1}\nu_k\right)$

$P_{k|k-1} = P_{k-1|k-1} + h\left(A_k P_{k-1|k-1} + P_{k-1|k-1} A_k^{\mathrm{T}} + Q_k\right)$

Update

$Y_k = \chi_{k|k-1}\begin{pmatrix} 0_{2\times 1} \\ 1 \end{pmatrix}$

$S = C_k P_{k|k-1} C_k^T + \hat{N}_k$

$L_k = P_{k|k-1} C_k^T S^{-1}$

$P_{k|k} = (I_3 - L_k C_k)\, P_{k|k-1}$

$\hat{\chi}_{k|k} = \hat{\chi}_{k|k-1} \exp\left(\tilde{L}_k\left[\hat{\chi}_{k|k-1}^{-1} Y_k - \begin{pmatrix} 0_{2\times 1} \\ 1 \end{pmatrix}\right]\right)$

end loop

4.5. Navigation on flat earth

This section presents the necessary equations and algorithms to simulate navigation on flat earth, which can be considered as a vehicle evolving in the 3D space. In addition, there are known features (landmarks) in the 3D space being observed relative to the vehicle position. The presented equations and algorithms in this chapter are based on [2].

In chapter 3 the MEKF and RIEKF was introduced in relation to nonlinear attitude filtering using quaternions, whereas for this chapter the MEKF and RIEKF is designed for estimation of attitude, position and linear velocity. The lie group in $SE_2(3)$ will be implemented for the RIEKF.

The system dynamics for the vehicle evolving in the 3D space is given in continuous time as

$$\dot{R} = R(\omega_k)^{\times} \tag{4.103}$$

$$\dot{v} = g + Ru_k \tag{4.104}$$

$$\dot{x} = v \tag{4.105}$$

where R, v and x is the attitude, velocity and position of the robot, respectively, w_k is the three-dimensional angular velocity as measured by the gyroscope, u_k is the three-dimensional measured acceleration, and g is the gravitational vector defined $g = (0, 0, -9.81)^T$.

The system dynamics is given in discrete time as

$$R_k = R_{k-1} + h(R_{k-1}(\omega_k)^{\times}) \tag{4.106}$$

$$v_k = v_{k-1} + h(g + R_{k-1}\hat{u}_k) \tag{4.107}$$

$$x_k = x_{k-1} + h(v_{k-1}) \tag{4.108}$$

Further, the corresponding noisy model is defined as

$$\dot{R} = R\left(\omega_k + w_\omega\right)^\times \tag{4.109}$$
$$\dot{v} = g + R\left(u_k + w_u\right) \tag{4.110}$$
$$\dot{x} = v \tag{4.111}$$

where w_ω and w_u are noise in the gyroscope and the accelerometer, respectively.

4.5.1. Multiplicative Extendend Kalman Filter - MEKF

The state vector for the MEKF is given as

$$X = (R, v, x) \tag{4.112}$$

The propagation of the state estimates in discrete time is defined as

$$\hat{R}_{k|k-1} = \hat{R}_{k-1|k-1} + h(\hat{R}_{k-1|k-1}\left(\hat{\omega}_k\right)^\times) \tag{4.113}$$
$$\hat{v}_{k|k-1} = \hat{v}_{k-1|k-1} + h\left(g + \hat{R}_{k-1|k-1}\hat{u}_k\right) \tag{4.114}$$
$$\hat{x}_{k|k-1} = \hat{x}_{k-1|k-1} + h(\hat{v}_{k-1|k-1}) \tag{4.115}$$

and the measurement and the estimated measurement is defined in vector form as

$$Y_k = \left(Y_k^1, \ldots, Y_k^n\right) = \left(R_k^T\left(p_1 - x_k\right), \ldots, R_k^T\left(p_n - x_k\right)\right) \tag{4.116}$$
$$\hat{Y}_k = \left(\hat{Y}_k^1, \ldots, \hat{Y}_k^n\right) = \left(\hat{R}_{k|k-1}^T\left(p_1 - \hat{x}_{k|k-1}\right), \ldots, \hat{R}_{k|k-1}^T\left(p_n - \hat{x}_{k|k-1}\right)\right) \tag{4.117}$$

where $Y_k, \hat{Y}_k \in \mathbb{R}^{3n}$, with n being the notation for the total number of landmarks.

Error equations and linearization

The estimation errors are given as

$$\tilde{R} = \hat{R}R^{\mathrm{T}} \tag{4.118}$$
$$\tilde{v} = \hat{v} - v \tag{4.119}$$
$$\tilde{x} = \hat{x} - x \tag{4.120}$$

so using Equation 4.106 and Equation 4.113 gives

$$\begin{aligned}\dot{\tilde{R}}_{k-1|k-1} &= \dot{\hat{R}}_{k-1|k-1}R_{k-1}^{\mathrm{T}} - \hat{R}_{k-1|k-1}\dot{R}_{k-1}^{\mathrm{T}} \\ &= \hat{R}_{k-1|k-1}(\omega_k)^\times R_{k-1}^{\mathrm{T}} - \hat{R}_{k-1|k-1}\left(\omega_k{}^\times\right)^{\mathrm{T}} R_{k-1}^{\mathrm{T}} \\ &= 0\end{aligned} \tag{4.121}$$

and using Equation 4.107 and Equation 4.114 gives

$$\begin{aligned}\dot{\tilde{v}}_{k-1|k-1} &= \dot{\hat{v}}_{k-1|k-1} - \dot{v}_{k-1} \\ &= g + \hat{R}_{k-1|k-1}u_k - (g + R_{k-1}u_k) \\ &= \hat{R}_{k-1|k-1}u_k - \tilde{R}_{k-1|k-1}^{\mathrm{T}}\hat{R}_{k-1|k-1}u_k \\ &= (I - \tilde{R}_{k-1|k-1}^{\mathrm{T}})\hat{R}_{k-1|k-1}u_k\end{aligned} \tag{4.122}$$

where R_{k-1} is substituted with $\tilde{R}_{k-1|k-1}^{\mathrm{T}}\hat{R}_{k-1|k-1}$.

Using Equation 4.108 and Equation 4.115 gives

$$\begin{aligned}\dot{\tilde{x}}_{k-1|k-1} &= \dot{\hat{x}}_{k-1|k-1} - \dot{x}_{k-1} \\ &= \hat{v}_{k-1|k-1} - v_{k-1} \\ &= \tilde{v}_{k-1|k-1}\end{aligned} \tag{4.123}$$

and finally, the measurement error is given as

$$\begin{aligned}\tilde{y}_k &= \hat{y}_k - y_k \\ &= \hat{R}^{\mathrm{T}}_{k|k-1}\left(p_n - \hat{x}_{k|k-1}\right) - R^{\mathrm{T}}_k\left(p_n - x_k\right) \\ &= \hat{R}^{\mathrm{T}}_{k|k-1}\left(p_n - \hat{x}_{k|k-1}\right) - \hat{R}^{\mathrm{T}}_{k|k-1}\tilde{R}_{k|k-1}\left(p_n - x_k\right) \\ &= \hat{R}^{\mathrm{T}}_{k|k-1}(I - \tilde{R}_{k|k-1})p_n - \hat{R}^{\mathrm{T}}_{k|k-1}(I - \tilde{R}_{k|k-1})\hat{x}_{k|k-1} - \hat{R}^{\mathrm{T}}_{k|k-1}\tilde{R}_{k|k-1}\tilde{x}_{k|k-1}\end{aligned} \tag{4.124}$$

The error model is summarized as

$$\begin{aligned}\dot{\tilde{R}}_{k-1|k-1} &= 0 \\ \dot{\tilde{v}}_{k-1|k-1} &= (I - \tilde{R}^{\mathrm{T}}_{k-1|k-1})\hat{R}_{k-1|k-1}u_k \\ \dot{\tilde{x}}_{k-1|k-1} &= \tilde{v}_{k-1|k-1}\end{aligned} \tag{4.125}$$

with the measurement error

$$\tilde{y}_k = \hat{R}^{\mathrm{T}}_{k|k-1}(I - \tilde{R}_{k|k-1})p_n - \hat{R}^{\mathrm{T}}_{k|k-1}(I - \tilde{R}_{k|k-1})\hat{x}_{k|k-1} - \hat{R}^{\mathrm{T}}_{k|k-1}\tilde{R}_{k|k-1}\tilde{x}_{k|k-1} \tag{4.126}$$

Since the error variable $\tilde{R}$ is not a vector variable, it is linearized using the first-order expansion $\tilde{R} \approx I + \tilde{\theta}^{\times}$, where $\tilde{\theta} \in \mathbb{R}^3$. Then, linearization at $\tilde{R} = I, \tilde{v} = 0$ and $\tilde{x} = 0$ using $\tilde{R} \approx I + \tilde{\theta}^{\times}$ gives

$$\begin{aligned}\dot{\tilde{v}}_{k-1|k-1} &= \left(I - \tilde{R}^{\mathrm{T}}_{k-1|k-1}\right)\hat{R}_{k-1|k-1}u_k \\ &\approx -\left(\tilde{\theta}^{\times}_{k-1|k-1}\right)^{\mathrm{T}}\hat{R}_{k-1|k-1}u_k \\ &= \tilde{\theta}^{\times}_{k-1|k-1}\hat{R}_{k-1|k-1}u_k \\ &= -(\hat{R}_{k-1|k-1}u_k)^{\times}\tilde{\theta}_{k-1|k-1}\end{aligned} \tag{4.127}$$

and the linearized measurement error is

$$\begin{aligned}\tilde{y}_k &= -\hat{R}^{\mathrm{T}}_{k|k-1}\tilde{\theta}^{\times}_{k|k-1}p_n + \hat{R}^{\mathrm{T}}_{k|k-1}\tilde{\theta}^{\times}_{k|k-1}\hat{x}_{k|k-1} - \hat{R}^{\mathrm{T}}_{k|k-1}\tilde{x}_{k|k-1} \\ &= -\hat{R}^{\mathrm{T}}_{k|k-1}\tilde{\theta}^{\times}_{k|k-1}\left(p_n - \hat{x}_{k|k-1}\right) - \hat{R}^{\mathrm{T}}_{k|k-1}\tilde{x}_{k|k-1} \\ &= \hat{R}^{\mathrm{T}}_{k|k-1}\left(p_n - \hat{x}_{k|k-1}\right)^{\times}\tilde{\theta}_{k|k-1} - \hat{R}^{\mathrm{T}}_{k|k-1}\tilde{x}_{k|k-1}\end{aligned} \tag{4.128}$$

The linearized matrices are then defined as

$$A_k = \begin{pmatrix} 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ -\left(\hat{R}_{k-1|k-1}u_k\right)^{\times} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & I_3 & 0_{3\times3}\end{pmatrix} \tag{4.129}$$

$$C_k = \begin{pmatrix} \hat{R}^T_{k|k-1}\left(p_1 - \hat{x}_{k|k-1}\right)^{\times} & 0_{3\times3} & -\hat{R}^T_{k|k-1} \\ & \cdots & \\ \hat{R}^T_{k|k-1}\left(p_n - \hat{x}_{k|k-1}\right)^{\times} & 0_{3\times3} & -\hat{R}^T_{k|k-1}\end{pmatrix} \tag{4.130}$$

Linearization of noisy model

Linearization of the noisy model defined in Equation 4.109-Equation 4.111 in continuous-time gives

$$
\begin{aligned}
\dot{\tilde{R}} &= \left(\frac{\mathrm{d}}{\mathrm{d}t}\hat{R}\right) R^{\mathrm{T}} - \hat{R}\dot{R}^{\mathrm{T}} \\
&= \hat{R}\omega_k^{\times} R^{\mathrm{T}} - \hat{R}\left(\omega_k^{\times} + w_\omega^{\times}\right)^{\mathrm{T}} R^{\mathrm{T}} \\
&= \hat{R} w_\omega^{\times} \hat{R}^{\mathrm{T}} \tilde{R}
\end{aligned} \tag{4.131}
$$

$$
\begin{aligned}
\dot{\tilde{v}} &= \dot{\hat{v}} - \dot{v} \\
&= g + \hat{R}u_k - g - Ru_k - Rw_u \\
&= \left(I - \tilde{R}^{\mathrm{T}}\right)\hat{R}u_k - \tilde{R}^{\mathrm{T}}\hat{R}w_u
\end{aligned} \tag{4.132}
$$

$$
\dot{\tilde{x}} = \dot{\hat{x}} - \dot{x} = \hat{v} \tag{4.133}
$$

Given that $\tilde{R} \approx I + \tilde{\theta}^{\times}$, the propagation of the error rotation is defined as $\dot{\tilde{R}} = \dot{\tilde{\theta}}^{\times}$, which gives

$$
\dot{\tilde{\theta}} = \hat{R}w_\omega \tag{4.134}
$$

$$
\begin{aligned}
\dot{\tilde{v}} &= \left(I - \tilde{R}^{\mathrm{T}}\right)\hat{R}u_k - \tilde{R}^{\mathrm{T}}\hat{R}w_u \\
&\approx \tilde{\theta}^{\times}\hat{R}u_k - \hat{R}w_u \\
&= -(\hat{R}u_k)^{\times}\tilde{\theta} - \hat{R}w_u
\end{aligned} \tag{4.135}
$$

$$
\dot{\tilde{x}} = \tilde{v} \tag{4.136}
$$

yielding the noise matrix G_k defined as

$$
G_k = \begin{pmatrix} \hat{R} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & -\hat{R} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \end{pmatrix} \tag{4.137}
$$

and in discrete time it is given as

$$
G_k = \begin{pmatrix} \hat{R}_{k-1|k-1} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & -\hat{R}_{k-1|k-1} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \end{pmatrix} \tag{4.138}
$$

Kalman gain and estimates

Using the linearized matrices, the covariance propagation is defined as

$$
P_{k|k-1} = P_{k-1|k-1} + h\left(A_k P_{k-1|k-1} + P_{k-1|k-1}A_k^{\mathrm{T}} + \hat{Q}_k\right) \tag{4.139}
$$

where

$$
\hat{Q}_k = G_k \operatorname{Cov}(w_k)\, G_k^T \tag{4.140}
$$

$$
\operatorname{Cov}(w_k) = Q \tag{4.141}
$$

It is noted that for the simulation of navigation on flat earth, two different noise matrices Q_1 and Q_2 are initialized as Q in order to study the behaviour of the MEKF and RIEKF filter. The noise matrices are defined in chapter 5.

It is noted that previously in chapter 3 the Kalman gain was denoted K_k, but for the current chapter, the Kalman gain is denoted L_k as in [2].

The Kalman gain is defined as

$$L_k = P_{k|k-1} H^T S^{-1} \tag{4.142}$$
$$S = H P_{k|k-1} H^T + \hat{N}_k \tag{4.143}$$

where S is defined using the noise matrix $\hat{N}_k$ given as

$$\hat{N}_k = \begin{pmatrix} \hat{R}_{k|k-1} \operatorname{Cov}\left(V_k^1\right) \hat{R}_{k|k-1}^T & & \\ & \ddots & \\ & & \hat{R}_{k|k-1} \operatorname{Cov}\left(V_k^n\right) \hat{R}_{k|k-1}^T \end{pmatrix} \tag{4.144}$$

and

$$N_k = \operatorname{Cov}\left(V_k\right) = Cov\left[\left(V_k^1, \cdots, V_k^n\right)^T\right] = \begin{pmatrix} 10^{-2} I_3 & 0_{3\times 3} & 0_{3\times 3} \\ 0_{3\times 3} & 10^{-2} I_3 & 0_{3\times 3} \\ 0_{3\times 3} & 0_{3\times 3} & 10^{-2} I_3 \end{pmatrix} \tag{4.145}$$

Then the covariance update is written as

$$P_{k|k} = \left(I_9 - L_k C_k\right) P_{k|k-1} \tag{4.146}$$

For the MEKF, the state is updated by defining a vector variable representing the innovation

$$\xi = \left(\xi_\theta, \xi_v, \xi_x\right)^{\mathrm{T}} = -L_k\left(\hat{Y}_k - Y_k\right), \quad \xi_\theta, \xi_v, \xi_x \in \mathbb{R}^3 \tag{4.147}$$

and then the updated state is defined as

$$\hat{X}_{k|k} = \left(\hat{R}_{k|k-1} \exp(\xi_\theta), \left(\hat{v}_{k|k-1} + \xi_v\right)^{\mathrm{T}}, \left(\hat{x}_{k|k-1} + \xi_x\right)^{\mathrm{T}}\right) \tag{4.148}$$

where $\exp(\cdot)$ is the exponential map corresponding to the exponential map defined for $SO(3)$ in Equation 2.15, so that it satisfies $\exp(\xi_\theta) = R(\xi_\theta)$.

MEKF algorithm

From the formulas presented in this chapter, the algorithm for the MEKF can be described by a propagation and an update.

Algorithm 5 MEKF - Navigation on flat earth

Initialize P_0 and $\hat{X}_0$
loop
Define A_k, C_k, as in Equation 4.130, Equation 4.129
Define $G_k, \hat{N}_k$ as in Equation 4.138, Equation 4.144
Propagation
$\hat{R}_{k|k-1} = \hat{R}_{k-1|k-1} + h(\hat{R}_{k-1|k-1}(\hat{\omega}_k)^\times)$
$\hat{v}_{k|k-1} = \hat{v}_{k-1|k-1} + h\left(g + \hat{R}_{k-1|k-1}\hat{u}_k\right)$
$\hat{x}_{k|k-1} = \hat{x}_{k-1|k-1} + h(\hat{v}_{k-1|k-1})$
$P_{k|k-1} = P_{k-1|k-1} + h\left(A_k P_{k-1|k-1} + P_{k-1|k-1}A_k^{\mathrm{T}} + \hat{Q}_k\right)$
Update
$Y_k = (Y_k^1, \ldots, Y_k^n) = \left(R_k^T(p_1 - x_k), \ldots, R_k^T(p_n - x_k)\right)$

$S = C_k P_{k|k-1} C_k^T + \hat{N}_k$
$L_k = P_{k|k-1} C_k^T S^{-1}$

$P_{k|k} = (I_9 - L_k C_k) P_{k|k-1}$

$\xi = (\xi_\theta, \xi_v, \xi_x)^{\mathrm{T}} = -L_k\left(Y_k - \hat{Y}_k\right), \quad \xi_\theta, \xi_v, \xi_x \in \mathbb{R}^3$
$\hat{X}_{k|k} = \left(\hat{R}_{k|k-1}R(\xi_\theta), \left(\hat{v}_{k|k-1} + \xi_v\right)^{\mathrm{T}}, \left(\hat{x}_{k|k-1} + \xi_x\right)^{\mathrm{T}}\right)$
end loop

4.5.2. Right Invariant Extended Kalman Filter - RIEKF

The state vector for the RIEKF is given as

$$X = (R, v, x) \tag{4.149}$$

Recall from earlier in this chapter that the system dynamics in discrete time is defined as in Equation 4.106 - Equation 4.108 and that the time propagation of the state estimates is defined as in Equation 4.113 - Equation 4.115.

This system can be embedded in the group of double homogeneous matrices $SE_2(3)$, introduced in subsection 2.2.5, which gives

$$\hat{\chi}_{k-1|k-1} = \begin{pmatrix} \hat{R}_{k-1|k-1} & \hat{v}_{k-1|k-1} & \hat{x}_{k-1|k-1} \\ 0_{1\times 3} & 1 & 0 \\ 0_{1\times 3} & 0 & 1 \end{pmatrix} \tag{4.150}$$

$$f_{\omega_k,u_k}(\hat{\chi}_{k-1|k-1}) = \begin{pmatrix} \hat{R}_{k-1|k-1}(\hat{\omega}_k)_\times & g + \hat{R}_{k-1|k-1}\hat{u}_k & \hat{v}_{k-1|k-1} \\ 0_{1\times 3} & 0 & 0 \\ 0_{1\times 3} & 0 & 0 \end{pmatrix} \tag{4.151}$$

Then the time propagation of the state estimate is defined as

$$\hat{\chi}_{k|k-1} = \hat{\chi}_{k-1|k-1} + h(f_{\omega_k,u_k}(\hat{\chi}_{k-1|k-1})) \tag{4.152}$$

and the measurement and the estimated measurement is given in homogeneous form as

$$Y_k = \left(Y_k^1, \ldots, Y_k^n\right) = \left(\chi_k^{-1}\begin{pmatrix} p_1 \\ 0 \\ 1 \end{pmatrix}, \ldots, \chi_k^{-1}\begin{pmatrix} p_n \\ 0 \\ 1 \end{pmatrix}\right) \tag{4.153}$$

$$\hat{Y}_k = \left(\hat{Y}_k^1, \ldots, \hat{Y}_k^n\right) = \left(\hat{\chi}_{k|k-1}^{-1}\begin{pmatrix} p_1 \\ 0 \\ 1 \end{pmatrix}, \ldots, \hat{\chi}_{k|k-1}^{-1}\begin{pmatrix} p_n \\ 0 \\ 1 \end{pmatrix}\right) \tag{4.154}$$

It is noted that Y_k and $\hat{Y}_k$ are both given in homogeneous form as mentioned above, and can be formulated as

$$Y_k^n = \begin{pmatrix} y_k^n \\ 0 \\ 1 \end{pmatrix}, \quad \hat{Y}_k^n = \begin{pmatrix} \hat{y}_k^n \\ 0 \\ 1 \end{pmatrix} \tag{4.155}$$

where

$$y_k^n = R_k\left(p_n - x_k\right), \quad \hat{y}_k^n = \hat{R}_{k|k-1}\left(p_n - \hat{x}_{k|k-1}\right) \tag{4.156}$$

Condition for autonomous error dynamics

Using the state embedded in $SE_2(3)$ from Equation 4.150, two states are defined in order to determine if the condition for autonomous error dynamics from Equation 4.4 is satisfied. The states are defined as

$$A = \begin{pmatrix} R_a & v_a & x_a \\ 0_{1\times 3} & 1 & 0 \\ 0_{1\times 3} & 0 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} R_b & v_b & x_b \\ 0_{1\times 3} & 1 & 0 \\ 0_{1\times 3} & 0 & 1 \end{pmatrix} \tag{4.157}$$

and from Equation 4.151, the corresponding functions are given as

$$f(A) = \begin{pmatrix} R_a\omega^\times & g + R_a u & v_a \\ 0_{1\times 3} & 0 & 0 \\ 0_{1\times 3} & 0 & 0 \end{pmatrix}, \quad f(B) = \begin{pmatrix} R_b\omega^\times & g + R_b u & v_b \\ 0_{1\times 3} & 0 & 0 \\ 0_{1\times 3} & 0 & 0 \end{pmatrix} \tag{4.158}$$

and lastly, the identity function is defined as

$$f(I) = \begin{pmatrix} I\omega^\times & g + u & 0_{3\times 1} \\ 0_{1\times 3} & 0 & 0 \\ 0_{1\times 3} & 0 & 0 \end{pmatrix} \tag{4.159}$$

The next step is to compute the matrices $f(AB), Af(B), f(A)B$ and $Af(I)B$. Then

$$AB = \begin{pmatrix} R_aR_b & R_av_b + v_a & R_ax_b + x_a \\ 0_{1\times 3} & 1 & 0 \\ 0_{1\times 3} & 0 & 1 \end{pmatrix} \tag{4.160}$$

$$f(AB) = \begin{pmatrix} R_aR_b\omega^\times & g + R_aR_bu & R_av_b + v_a \\ 0_{1\times 3} & 0 & 0 \\ 0_{1\times 3} & 0 & 0 \end{pmatrix} \tag{4.161}$$

The matrices $Af(B), f(A)B$ are defined as

$$Af(B) = \begin{pmatrix} R_a R_b \omega^\times & R_a g + R_a R_b u & R_a v_b \\ 0_{1\times3} & 0 & 0 \\ 0_{1\times3} & 0 & 0 \end{pmatrix} \tag{4.162}$$

$$f(A)B = \begin{pmatrix} R_a \omega^\times R_b & R_a \omega^\times v_b + g + R_a u & R_a \omega^\times x_b + v_a \\ 0_{1\times3} & 0 & 0 \\ 0_{1\times3} & 0 & 0 \end{pmatrix} \tag{4.163}$$

and lastly

$$\begin{aligned} Af(I)B &= \begin{pmatrix} R_a \omega^\times & R_a g + R_a u & 0_{3\times1} \\ 0_{1\times3} & 0 & 0 \\ 0_{1\times3} & 0 & 0 \end{pmatrix} \begin{pmatrix} R_b & v_b & x_b \\ 0_{1\times3} & 1 & 0 \\ 0_{1\times3} & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} R_a \omega^\times R_b & R_a \omega^\times v_b + R_a g + R_a u & R_a \omega^\times x_b \\ 0_{1\times3} & 0 & 0 \\ 0_{1\times3} & 0 & 0 \end{pmatrix} \end{aligned} \tag{4.164}$$

From these matrices, it is seen that

$$f(AB) = Af(B) + f(A)B - Af(I)B \tag{4.165}$$

which satisfies the condition Equation 4.4, and it is concluded that the error dynamics are autonomous.

Error equations and linearization

The estimation errors are given as

$$\tilde{R} = \hat{R}R^{\mathrm{T}} \tag{4.166}$$

$$\tilde{v} = \hat{v} - \tilde{R}v \tag{4.167}$$

$$\tilde{x} = \hat{x} - \tilde{R}x \tag{4.168}$$

and the estimated measurement error is

$$\tilde{y}^n = \hat{R}\left(\hat{y}^n - y^n\right) \tag{4.169}$$

Then, using Equation 4.106 and Equation 4.113 gives

$$\begin{aligned} \dot{\tilde{R}}_{k-1|k-1} &= \dot{\hat{R}}_{k-1|k-1} R_{k-1}^{\mathrm{T}} - \hat{R}_{k-1|k-1} \dot{R}_{k-1}^{\mathrm{T}} \\ &= \hat{R}_{k-1|k-1} (\omega_k)^\times R_{k-1}^{\mathrm{T}} - \hat{R}_{k-1|k-1} \left(\omega_k{}^\times\right)^{\mathrm{T}} R_{k-1}^{\mathrm{T}} \\ &= 0 \end{aligned} \tag{4.170}$$

and using Equation 4.107 and Equation 4.114 gives

$$\begin{aligned} \dot{\tilde{v}}_{k-1|k-1} &= \dot{\hat{v}}_{k-1|k-1} - \dot{\tilde{R}}_{k-1|k-1} v_{k-1} - \tilde{R}_{k-1|k-1} \dot{v}_{k-1} \\ &= g + \hat{R}_{k-1|k-1} u_k - \tilde{R}_{k-1|k-1} (g + R_{k-1} u_k) \\ &= (I - \tilde{R}_{k-1|k-1}) g \end{aligned} \tag{4.171}$$

where R_{k-1} is substituted with $\tilde{R}_{k-1|k-1}^{\mathrm{T}} \hat{R}_{k-1|k-1}$.

Using Equation 4.108 and Equation 4.115 gives

$$\begin{aligned}
\dot{\tilde{x}}_{k-1|k-1} &= \dot{\hat{x}}_{k-1|k-1} - \dot{\tilde{R}}_{k-1|k-1}x_{k-1} - \tilde{R}_{k-1|k-1}\dot{x}_{k-1} \\
&= \hat{v}_{k-1|k-1} - \tilde{R}_{k-1|k-1}v_{k-1}
\end{aligned} \tag{4.172}$$

and finally, the measurement error is given as

$$\begin{aligned}
\tilde{y}^n_{k|k-1} &= \hat{R}_{k|k-1}\left(\hat{y}^n_k - y^n_k\right) \\
&= \hat{R}_{k|k-1}\hat{R}^{\mathrm{T}}_{k|k-1}\left(p_n - \hat{x}_{k|k-1}\right) - \hat{R}_{k|k-1}R^{\mathrm{T}}_{k|k-1}\left(p_n - x_k\right) \\
&= (I - \tilde{R}_{k|k-1})p_n - (\hat{x}_{k|k-1} - \tilde{R}_{k|k-1}x_k) \\
&= (I - \tilde{R}_{k|k-1})p_n - \tilde{x}_{k|k-1}
\end{aligned} \tag{4.173}$$

The error model is summarized as

$$\dot{\tilde{R}}_{k-1|k-1} = 0 \tag{4.174}$$

$$\dot{\tilde{v}}_{k-1|k-1} = (I - \tilde{R}_{k-1|k-1})g \tag{4.175}$$

$$\dot{\tilde{x}}_{k-1|k-1} = \hat{v}_{k-1|k-1} - \tilde{R}_{k-1|k-1}v_{k-1} \tag{4.176}$$

with the measurement error

$$\tilde{y}_k = (I - \tilde{R}_{k|k-1})p_n - \tilde{x}_{k|k-1} \tag{4.177}$$

Recall from the previous subsection that since the error variable $\tilde{R}$ is not a vector variable, it is linearized using the first-order expansion $\tilde{R} \approx I + \tilde{\theta}^{\times}$, where $\tilde{\theta} \in \mathbb{R}^3$. Then, linearization at $\tilde{R} = I, \tilde{v} = 0$ and $\tilde{x} = 0$ using $\tilde{R} \approx I + \tilde{\theta}^{\times}$ gives

$$\begin{aligned}
\dot{\tilde{v}}_{k-1|k-1} &= (I - \tilde{R}_{k-1|k-1})g \\
&\approx -\tilde{\theta}^{\times}_{k-1|k-1}g = g^{\times}\tilde{\theta}_{k-1|k-1}
\end{aligned} \tag{4.178}$$

$$\begin{aligned}
\dot{\tilde{x}}_{k-1|k-1} &= \hat{v}_{k-1|k-1} - \tilde{R}_{k-1|k-1}v_{k-1} \\
&= \hat{v}_{k-1|k-1} - \tilde{R}_{k-1|k-1}\tilde{R}^{\mathrm{T}}_{k-1|k-1}\left(\hat{v}_{k-1|k-1} - \tilde{v}_{k-1|k-1}\right) \\
&= \tilde{v}_{k-1|k-1}
\end{aligned} \tag{4.179}$$

and the linearized measurement error is

$$\begin{aligned}
\tilde{y}_k &= (I - \tilde{R}_{k|k-1})p_n - \tilde{x}_{k|k-1} \\
&= -\tilde{\theta}^{\times}_{k|k-1}p_n - \tilde{x}_{k|k-1} \\
&= (p_n)^{\times}\tilde{\theta}_{k|k-1} - \tilde{x}_{k|k-1}
\end{aligned} \tag{4.180}$$

The linearized matrices are then defined as

$$A_k = \begin{pmatrix} 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ (g)^{\times} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & I_3 & 0_{3\times3} \end{pmatrix} \tag{4.181}$$

$$C_k = \begin{pmatrix} (p_1)^{\times} & 0_{3\times3} & -I_3 \\ & \cdots & \\ (p_n)^{\times} & 0_{3\times3} & -I_3 \end{pmatrix} \tag{4.182}$$

Linearization of noisy model

Linearization of the noisy model defined in Equation 4.109-Equation 4.111 in continuous-time gives

$$\begin{aligned}\dot{\tilde{R}} &= \left(\frac{\mathrm{d}}{\mathrm{d}t}\hat{R}\right)R^{\mathrm{T}} - \hat{R}\dot{R}^{\mathrm{T}} \\ &= \hat{R}\omega_k^{\times}R^{\mathrm{T}} - \hat{R}\left(\omega_k^{\times} + w_\omega^{\times}\right)^{\mathrm{T}}R^{\mathrm{T}} \\ &= \hat{R}w_\omega^{\times}\hat{R}^{\mathrm{T}}\tilde{R}\end{aligned} \tag{4.183}$$

$$\begin{aligned}\dot{\tilde{v}} &= \dot{\hat{v}} - \dot{\tilde{R}}v - \tilde{R}\dot{v} \\ &= g + \hat{R}u_k - \left(\hat{R}w_\omega^{\times}\hat{R}^{\mathrm{T}}\tilde{R}\right)v - \tilde{R}g + \tilde{R}Ru_k + \tilde{R}Rw_u \\ &= (I - \tilde{R})g - \left(\hat{R}w_\omega\right)^{\times}\tilde{R}v + \hat{R}w_u\end{aligned} \tag{4.184}$$

$$\begin{aligned}\dot{\tilde{x}} &= \dot{\hat{x}} - \dot{\tilde{R}}x - \tilde{R}\dot{x} \\ &= \hat{v} - \left(\hat{R}w_\omega^{\times}\hat{R}^{\mathrm{T}}\tilde{R}\right)x - \tilde{R}v\end{aligned} \tag{4.185}$$

Given that $\tilde{R} \approx I + \tilde{\theta}^{\times}$, the propagation of the error rotation is defined as $\dot{\tilde{R}} = \dot{\tilde{\theta}}^{\times}$, which gives

$$\dot{\tilde{\theta}}^{\times} = \hat{R}w_\omega^{\times}\hat{R}^{\mathrm{T}} = \left(\hat{R}w_\omega\right)^{\times} \tag{4.186}$$

Then the linearized noisy model is defined as

$$\dot{\tilde{\theta}} = \hat{R}w_\omega \tag{4.187}$$

$$\dot{\tilde{v}} = g^{\times}\tilde{\theta} + v^{\times}\hat{R}w_\omega + \hat{R}w_u \tag{4.188}$$

$$\begin{aligned}\dot{\tilde{x}} &= \hat{v} - \left(\hat{R}w_\omega^{\times}\hat{R}^{\mathrm{T}}\tilde{R}\right)x - \tilde{R}v \\ &= \tilde{v} + x^{\times}\hat{R}w_\omega\end{aligned} \tag{4.189}$$

yielding the noise matrix G_k defined as

$$G_k = \begin{pmatrix} \hat{R} & 0_{3\times3} & 0_{3\times3} \\ \hat{v}^{\times}\hat{R} & \hat{R} & 0_{3\times3} \\ \hat{x}^{\times}\hat{R} & 0_{3\times3} & 0_{3\times3} \end{pmatrix} \tag{4.190}$$

and in discrete time it is given as

$$G_k = \begin{pmatrix} \hat{R}_{k-1|k-1} & 0_{3\times3} & 0_{3\times3} \\ \left(\hat{v}_{k-1|k-1}\right)^{\times}\hat{R}_{k-1|k-1} & \hat{R}_{k-1|k-1} & 0_{3\times3} \\ \left(\hat{x}_{k-1|k-1}\right)^{\times}\hat{R}_{k-1|k-1} & 0_{3\times3} & 0_{3\times3} \end{pmatrix} \tag{4.191}$$

Kalman gain and estimates

Using the linearized matrices, the covariance propagation is defined as in Equation 4.139 using G_k derived above from Equation 4.191. As mentioned earlier, $\mathrm{Cov}\left(w_k\right)$ is defined by two different noise matrices Q_1 and Q_2 depending on which matrix is initiated, and can be found from chapter 5.

Exactly like the MEKF, the Kalman gain L_k is defined by using Equation 4.143-Equation 4.145, and the covariance update is defined as in Equation 4.146.

The innovation in homogeneous form is given as

$$\tilde{Y}_k = \hat{\chi}_{k|k-1}\left(\hat{Y}_k - Y_k\right) \tag{4.192}$$

and can be formulated as

$$\tilde{Y}_k^n = \begin{pmatrix} \tilde{y}_k^n \\ 0 \\ 0 \end{pmatrix}, \quad \tilde{y}_k^n = \hat{R}_{k|k-1}(\hat{y}_k^n - y_k^n) \tag{4.193}$$

Since the last two elements of each measurement, the state update is defined as

$$\hat{\chi}_{k|k} = \exp\left(-L_k \tilde{y}_k\right)\hat{\chi}_{k|k-1} \tag{4.194}$$

where $\tilde{y}_k = (\tilde{y}_k^1, \ldots, \tilde{y}_k^n) \in \mathbb{R}^{3n}$ and $\exp(\cdot)$ is the exponential map corresponding to the exponential map defined for $SE_2(3)$ in Equation 2.27.

RIEKF algorithm

From the formulas presented in this chapter, the algorithm for the RIEKF can be described by a propagation and an update.

Algorithm 6 RIEKF - Navigation on flat earth

Initialize P_0 and $\hat{X}_0$
loop
  Define A_k, C_k, as in Equation 4.181, Equation 4.182
  Define $G_k, \hat{N}_k$ as in Equation 4.191, Equation 4.144
  Propagation
  $\hat{\chi}_{k|k-1} = \hat{\chi}_{k-1|k-1} + h(f_{\omega_k,u_k}(\hat{\chi}_{k-1|k-1}))$
  $P_{k|k-1} = P_{k-1|k-1} + h\left(A_k P_{k-1|k-1} + P_{k-1|k-1}A_k^{\mathrm{T}} + \hat{Q}_k\right)$
  Update
  $y_k^n = R_k^T\left(p_n - x_k\right)$
  $\tilde{y}_k = (\tilde{y}_k^1, \ldots, \tilde{y}_k^n) = (\hat{R}_{k|k-1}(\hat{y}_k^1 - y_k^1), \ldots, \hat{R}_{k|k-1}(\hat{y}_k^n - y_k^n))$

  $S = C_k P_{k|k-1} C_k^T + \hat{N}_k$
  $L_k = P_{k|k-1} C_k^T S^{-1}$

  $P_{k|k} = \left(I_9 - L_k C_k\right) P_{k|k-1}$
  $\hat{\chi}_{k|k} = \exp\left(-L_k \tilde{y}_k\right)\hat{\chi}_{k|k-1}$
end loop

Chapter 5.

Simulation

The simulations performed in this thesis are simulated by python, and the codes are presented in appendix A.1 and A.2.

5.1. Nonlinear attitude filtering

This subsection presents the simulation steps and parameters of the MEKF and the RIEKF for nonlinear attitude filtering using quaternions. As mentioned earlier, the nonlinear attitude filtering case will not be simulated and displayed in this thesis as it is extracted from the specialization project prior to the master thesis.

The simulation is performed with 0.001 s time step with a total time of 30s, where the true trajectory is defined by the sinusoidal input $\Omega = \frac{1}{2}\sin(\frac{2\pi}{5}t)\,[0,0,1]\,\frac{\circ}{\mathrm{s}}$.

The initial rotation defined by the unit quaternion and the initial bias is initialized with a standard deviation $std_{q0} = 60^{\circ}$ and $std_{b0} = 20\frac{\circ}{s}$, respectively. The coefficient matrix Q_a is defined with a standard deviation $std_{\Omega} = 25\frac{\circ}{s}$, and Q_b with a standard deviation $std_b = 0.1\frac{\circ}{s}$ squared. Next, the coefficient matrix R_c is defined with a standard deviation $std_y = 30^{\circ}$. It is important to note that for the noise initialization, the standard deviations are converted from degrees to radians before they are implemented.

A complete overview of the system initialization parameters can be found in Table 5.1 and the noise initialization parameters can be found in Table 5.2. It is noted that the initialization parameters used for this case are identical for both the MEKF and RIEKF.

System initialization	
Parameter	Value
Ω	$\left(0, 0, \frac{1}{2}\sin(\frac{2\pi}{5}t)\right)\frac{\circ}{\mathrm{s}}$
h	$0.001s$
t	$30s$
d_i	$d_1 = (1,0,0)^T, d_2 = (0,0,0)^T, d_2 = (0,0,1)^T$
q_0	$(0,0,0,1)^T$
b_0	$(0,-0.5,0.01)^T$

Table 5.1.: Nonlinear attitude filtering - Quaternion: Parameters and their respective values for the system initialization for the MEKF and RIEKF

Noise initializaition	
Parameter	Value
std_{q0}	$60°$
std_{b0}	$20\frac{°}{s}$
std_{Ω}	$25\frac{°}{s}$
std_b	$0.1\frac{°}{s}$
std_y	$30°$
Pa_0	$\frac{1}{std_{q0}^2}I_3$
Pb_0	$\frac{1}{std_{b0}^2}I_3$
Pc_0	I_3
Qa_0	$diag((std_{\Omega}^2, std_{\Omega}^2, std_{\Omega}^2))$
Qb_0	$diag((std_b^2, std_b^2, std_b^2))$
Qc_0	$0_{3\times3}$
R_c	$diag((std_y^2, std_y^2, std_y^2))$

Table 5.2.: Nonlinear attitude filtering - Quaternion: Parameters and their respective values for the noise initialization and coefficient matrices for the MEKF and RIEKF

5.2. Pose estimation - Simple car model

This subsection presents the simulation steps and parameters of the EKF and the LIEKF in relation to pose estimation. The true trajectory moves in a circle with radius $r \approx 6.283m$, with a speed of $v = 1\frac{m}{s}$ and an angular velocity of $\omega = \frac{2\pi}{40}s^{-1}$ so that it takes $40s$ for the robot to complete the circle.

The simulation is simulated using $h = 0.1s$ for a duration of $32s$ as the remaining part of the trajectory is irrelevant due the filters convergence properties. Two initial cases are performed for the filters using two different initial headings, $\hat{\theta}_0 = 1°$ and $\hat{\theta}_0 = 45°$, corresponding to the initial heading error of the robot. The initialization of the covariance matrix P_0 depends on the heading error and the initial position of the robot is always assumed known.

A complete overview of the system initialization parameters can be found in Table 5.3 and the noise initialization parameters can be found in Table 5.4. It is noted that the initialization parameters used for this simulation are identical for both the EKF and LIEKF.

System initializaition	
Parameter	Value
h	$0.1s$
t	$32s$
v	$1\frac{m}{s}$
ω	$\frac{2\pi}{40}s^{-1}$
$\hat{\theta}_0$	$1°$ and $45°$
$\hat{x}_0$	$(0,0)^T$

Table 5.3.: Pose estimation - Simple car model: Parameters and their respective values for the system initialization of the EKF and LIEKF

Noise initializaition	
Parameter	Value
$P_0 \ \ (\hat{\theta}_0 = 1°)$	$\mathrm{diag}\left((\pi/180)^2, 0, 0\right)$
$P_0 \ \ (\hat{\theta}_0 = 45°)$	$\mathrm{diag}\left((15\pi/180)^2, 0, 0\right)$
Q_k	$\mathrm{diag}\left((\pi/180)^2, 10^{-4}, 10^{-4}\right)$
N_k	I_2

Table 5.4.: Pose estimation - Simple car model: Noise and covariance initialization parameters and their respective values for the EKF and LIEKF

5.3. Pose estimation - Navigation on flat earth

This section presents the simulation steps and parameters of the MEKF and the RIEKF for pose estimation. It is noted that for the simulation of navigation on flat earth, the MEKF uses the noise matrix G_k as defined in Equation 4.191, which is actually designed for the RIEKF. Even though G_k is defined as in Equation 4.138 for the MEKF (given its error equations), this did not produce the same result as in [2] for an unknown reason. This could be something to investigate as a future work of what has been done in this thesis.

The true trajectory moves in a circle with radius $r = 5m$ with a speed of $v = 1\frac{m}{s}$ and an angular velocity of $\omega = \frac{2\pi}{30}s^{-1}$ around the z-axis so that it takes $30s$ for the robot to complete the circle.

The simulation is simulated using $h = 0.1s$ for a duration of $30s$. Two cases are simulated using two different noise matrices, Q_1 and Q_2, where the attitude error of the robot is $\hat{R}_0 = \exp(15°) \in SO(3)$ and the initial position error is $\hat{x}_0 = (1, 0, 1)^T$, corresponding to a standard deviation of $1m$ in the x-direction and z-direction.

A complete overview of the system initialization parameters can be found in Table 5.5 and the noise initialization parameters can be found in Table 5.6. It is noted that the initialization parameters used for this simulation are identical for both the MEKF and RIEKF.

System initializaition	
Parameter	Value
h	$0.1s$
t	$30s$
r	$5m$
ω	$\frac{2\pi}{t}s^{-1}$
g	$(0, 0, -9.81)^T$
$\hat{R}_0$	$\exp(15°) \in SO(3)$
$\hat{v}_0$	$(\omega r, 0, 0)^T$
$\hat{x}_0$	$(1, 0, 1)^T$

Table 5.5.: Pose estimation - Navigation on flat earth: Parameters and their respective values for the system initialization of the MEKF and RIEKF

Noise initializaition	
Parameter	Value
$P_0 \;\; (Q = Q_1)$	$\begin{pmatrix} (\frac{5\pi}{180})^2 I_3 & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 10^{-2} I_3 & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & I_3 \end{pmatrix}$
$P_0 \;\; (Q = Q_2)$	$\begin{pmatrix} (\frac{15\pi}{180})^2 I_3 & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 10^{-2} I_3 & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & I_3 \end{pmatrix}$
Q_1	$\begin{pmatrix} 10^{-8} I_3 & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 10^{-8} I_3 & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \end{pmatrix}$
Q_2	$\begin{pmatrix} 10^{-4} I_3 & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 10^{-4} I_3 & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \end{pmatrix}$
N_k	$10^{-2} I_9$

Table 5.6.: Pose estimation - Navigation on flat earth: Noise and covariance initialization parameters and their respective values for the MEKF and RIEKF

Chapter 6.

Results & Discussions

6.1. Simple car model

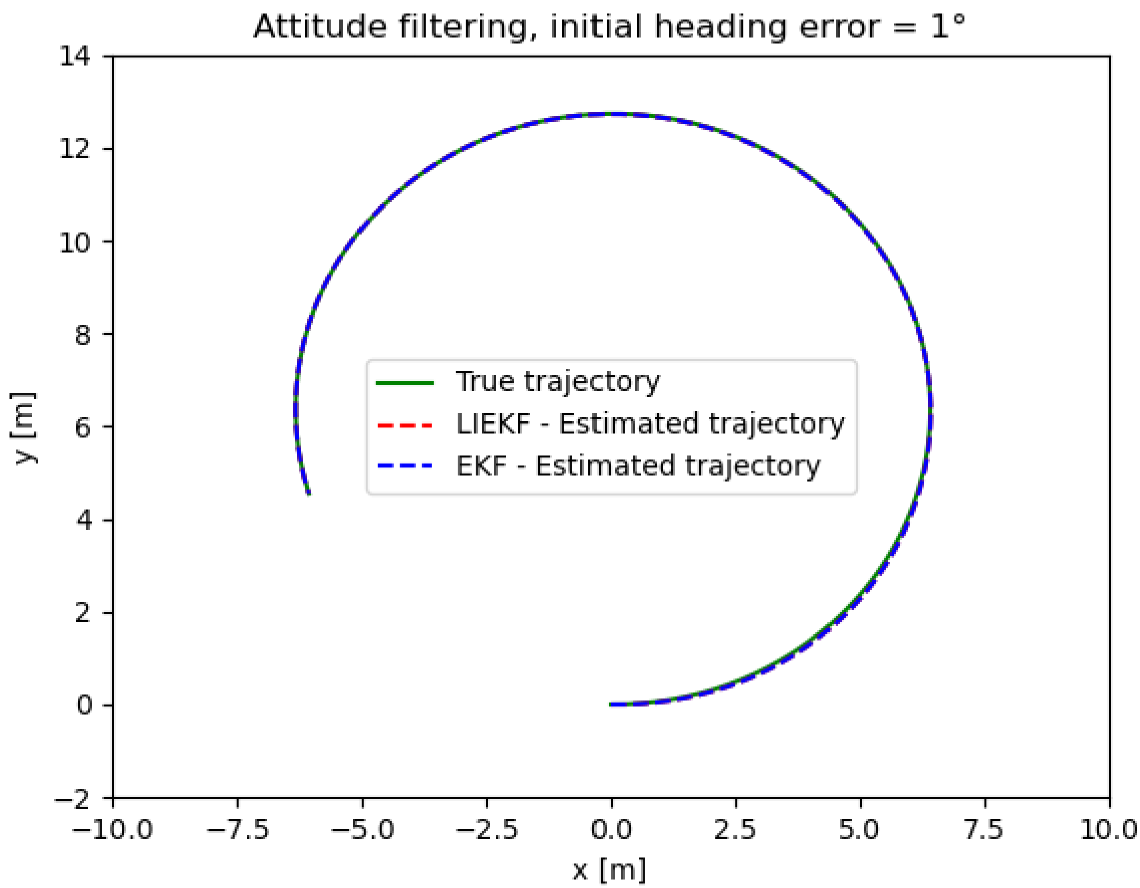

Figure 6.1.: Simulated results for the EKF and LIEKF pose estimation with an initial heading error $\hat{\theta}_0 = 1°$.

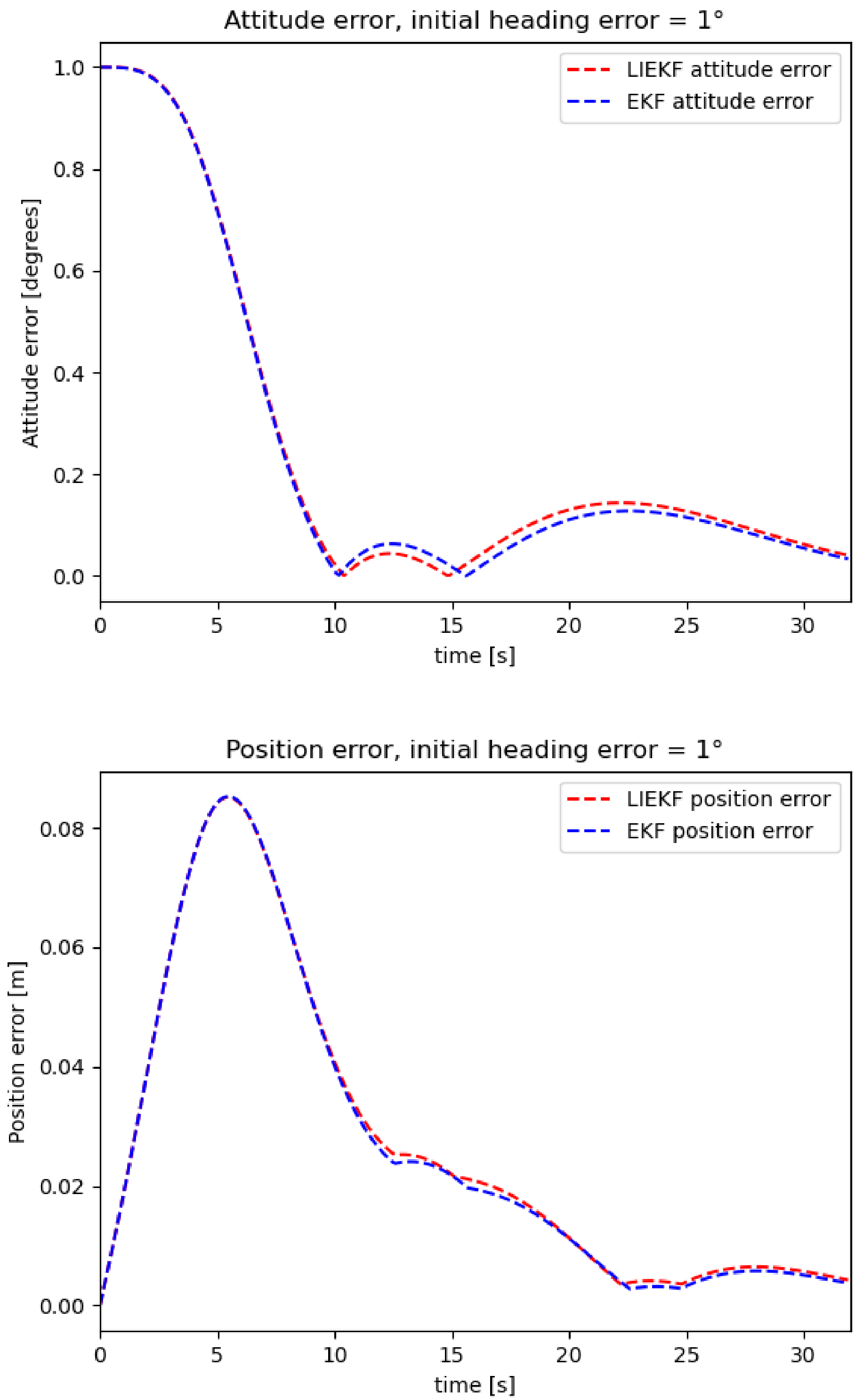

Figure 6.2.: Comparison of the attitude and position error for the EKF and LIEKF with initial heading error $\hat{\theta}_0 = 1°$, plotted against the time.

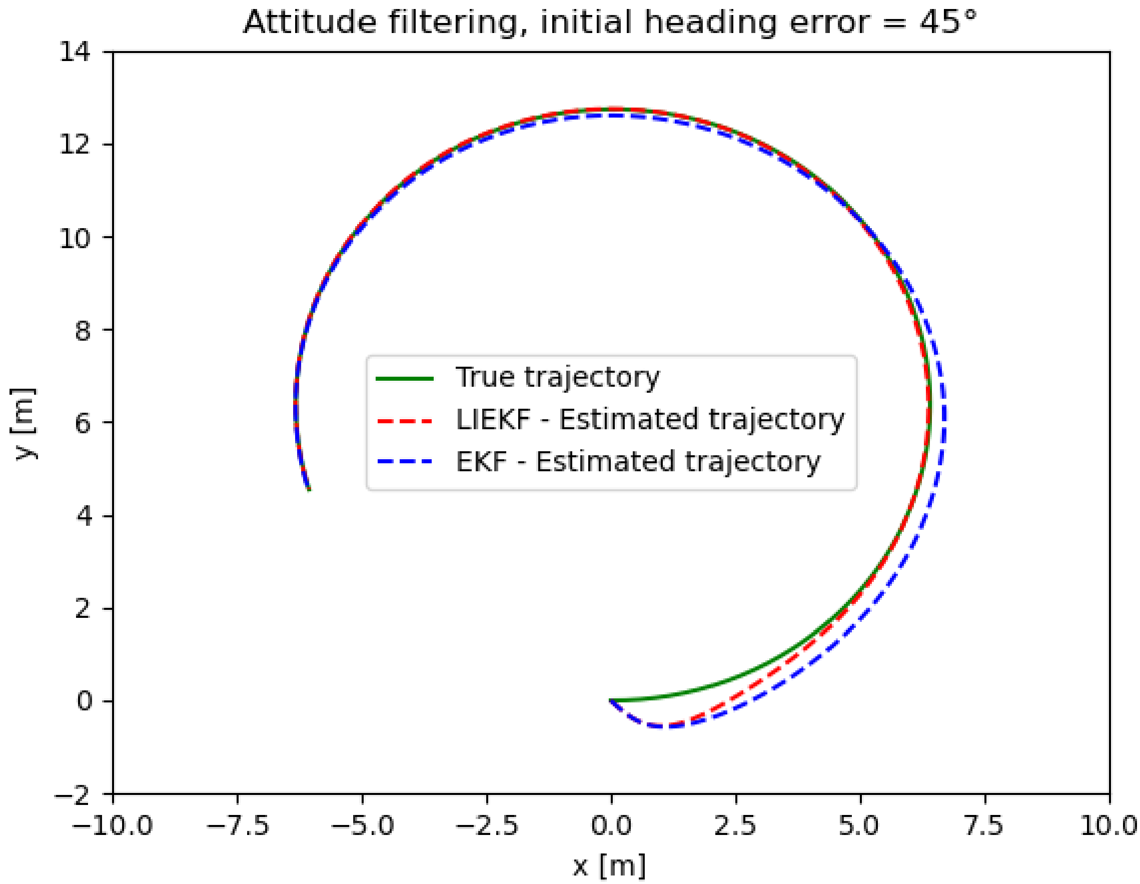

Figure 6.3.: Simulated results for the EKF and LIEKF pose estimation with an initial heading error $\hat{\theta}_0 = 45°$.

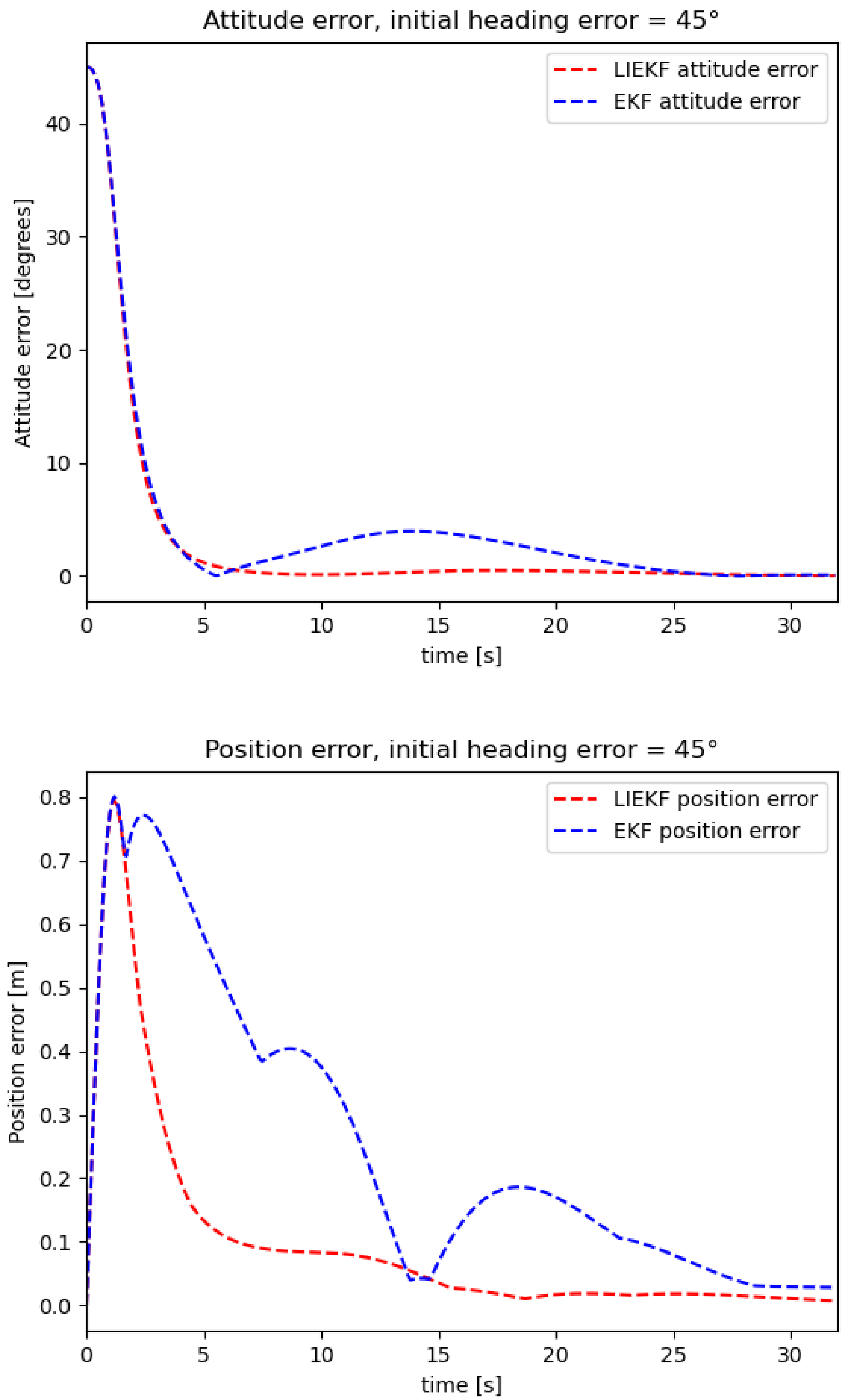

Figure 6.4.: Comparison of the attitude and position error for the EKF and LIEKF with initial heading error $\hat{\theta}_0 = 45°$, plotted against the time.

6.1.1. Discussion

From Figure 6.1 it is seen that a heading error of $1°$ has very little effect on the performance of the filters, and both filters behave similarly. Figure 6.3 shows that a heading error of $45°$ has a greater effect on the performance of the filters. The robot starts at $(0, 0)$, but as the heading error is now $45°$, the robot starts moving outwards. It is obvious to see that the LIEKF outperforms the EKF as it consists of autonomous error properties.

The error plots from Figure 6.2, shows that the EKF is clearly outperformed as the LIEKF error shows rapid decrease in position estimation error, whereas the EKF stuggles to converge up until approximately $t \approx 28s$. From the plot of the attitude estimation error it is seen that the EKF corrects itself within the first five seconds, but is seen to increase in error as the time increases, before converging at approximately $t \approx 25s$. The LIEKF on the other hand, completely outperforms the EKF as it converges within five seconds.

6.2. Navigation on flat earth

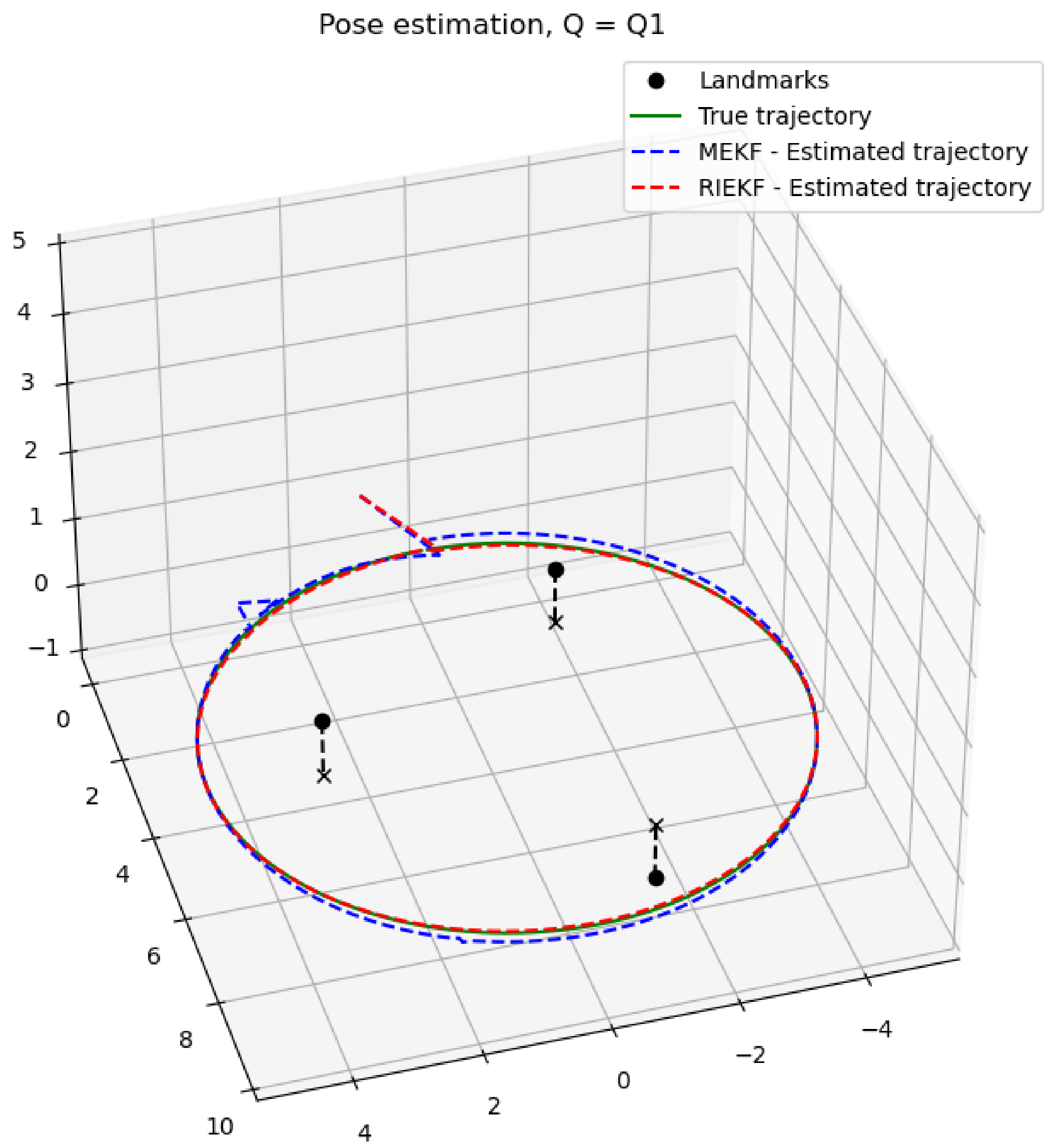

Figure 6.5.: Simulated results for the MEKF and RIEKF pose estimation with $Q = Q_1$.

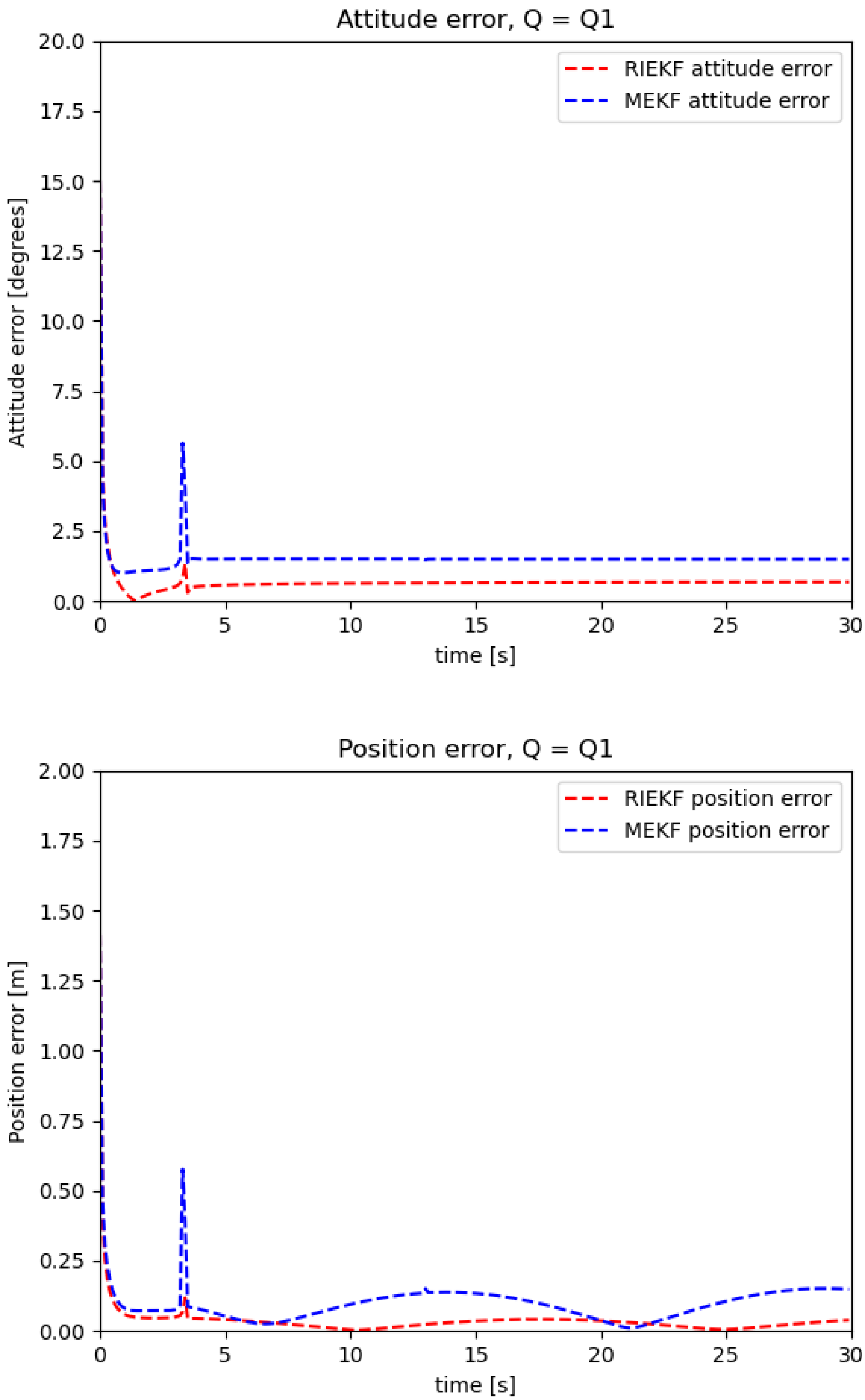

Figure 6.6.: Comparison of the attitude and position error for the MEKF and RIEKF with $Q = Q_1$, plotted against the time.

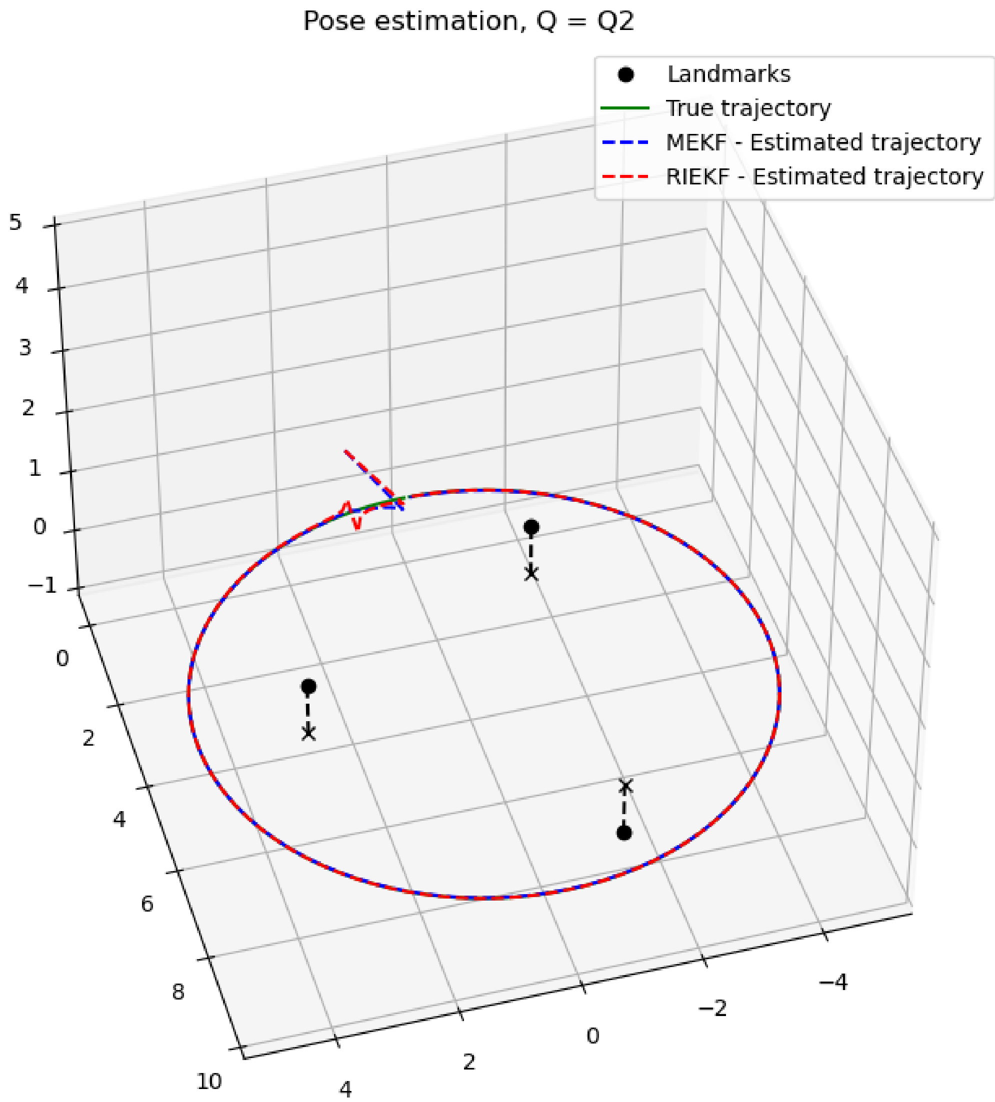

Figure 6.7.: Simulated results for the MEKF and RIEKF pose estimation with $Q = Q_2$.

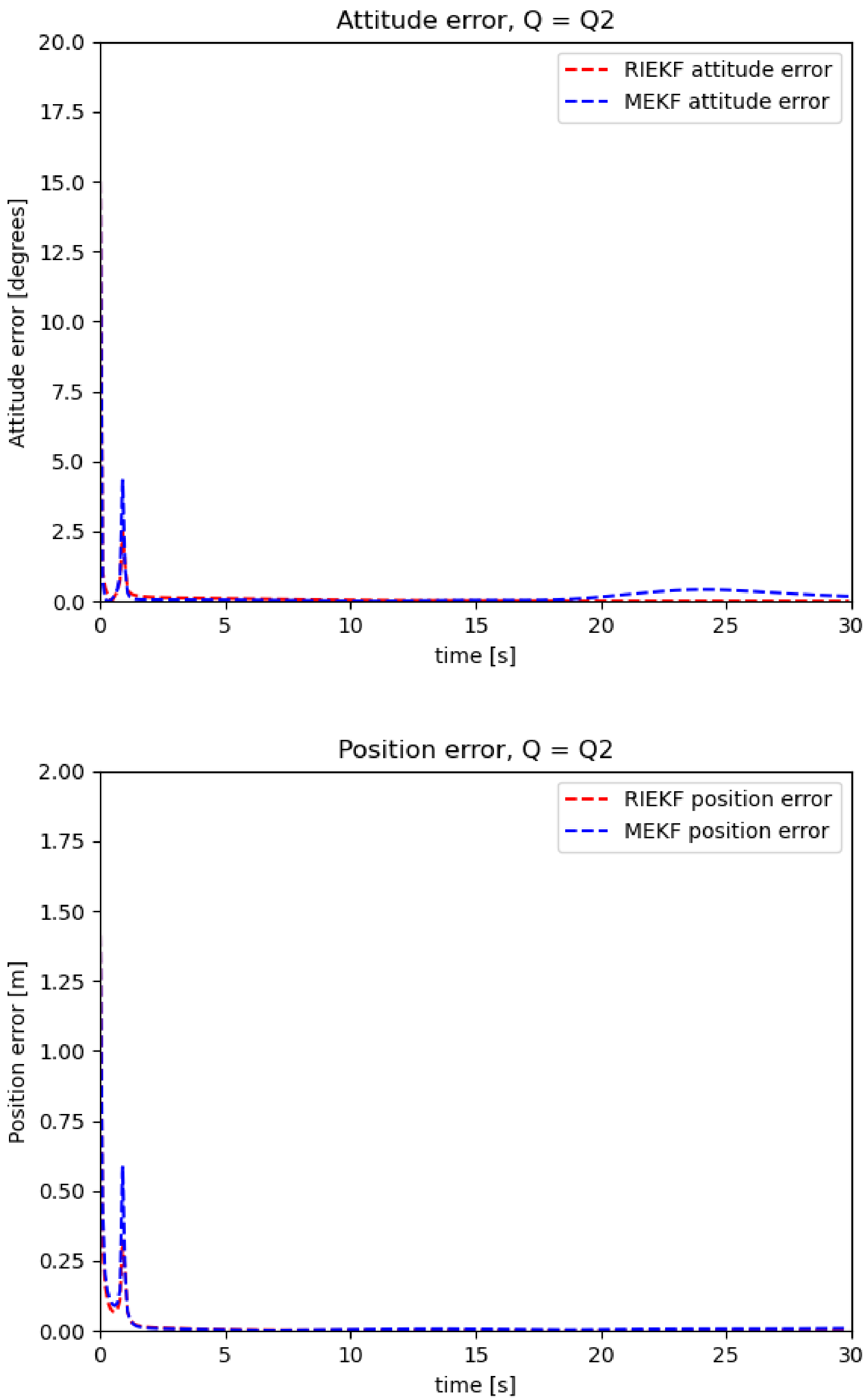

Figure 6.8.: Comparison of the attitude and position error for the MEKF and RIEKF with $Q = Q_2$, plotted against the time.

6.2.1. Discussion

It is noted that the dashed line in Figure 6.5 and Figure 6.7 shows the distance from the landmark (feature) to the xy-plane (height in z-direction) and are the same for both figures.

The error plots from Figure 6.6 shows the error of the filters for the tightly tuned process noise matrix $Q = Q1$, to represent highly precise inertial sensors. From the attitude error plot, it is seen that both the MEKF and RIEKF correct themselves immediately which is caused by the gains decreasing. However, the gains are now too small to be corrected for the MEKF, leading to its convergence. This is clearly illustrated in Figure 6.5 as well. The RIEKF is seen to rapidly decrease to zero in attitude estimation error, but then increases before stabilizing over the next couple of seconds. The position error plot also shows that the RIEKF converges to zero, although it does fluctuate a very small amount after, and the MEKF is clearly outperformed.

The error plots from Figure 6.8 shows the error of the filters for the enlarged process noise matrix $Q = Q2$. The tuning of the process noise matrix Q_1 is called robust tuning, which is a way of improving the convergence properties of the (M)EKF [2], as can be seen from the plots. Both filters converge to zero in attitude and position estimation error.

Comparing the results provided in this thesis with that of [2], it is clear to see that the error plots does not look exactly the same. This could be a result of the difference between the initial covariance matrix P_0 defined for this thesis, and the initial covariance matrix used in [2] which was not specified.

Chapter 7.

Conclusion

The simulations presented in this thesis shows the superiority of the IEKF in comparison to the EKF as the latter filter is outperformed on every case that was simulated, even for challenging Q. This is a result of the invariance of the IEKF, which causes the estimation error to satisfy the log-linear autonomous differential equation on the Lie algebra corresponding to the Lie algebra of the system dynamic [9].

In the simulation of the simple car model, a small heading error results in both filters converging but as the heading error increases, the IEKF outperfomrs the EKF due to the use of the system's nonlinearities. In the simulation of navigation on flat earth, the differently tuned process noise matrices Q_1 and Q_2 highlights the advantage of the IEKF over the EKF. It is concluded that the IEKF possesses theoretical stability guarantees around any trajectory, which EKF does not, using the same tuning implementation and computational load [2]. Although the EKF does not guarantee convergence around any trajectory, it is still possible to improve the EKF using the trick where the process noise matrix is tuned properly, e.g. Q_2.

References

[1] Axel Barrau. "Non-linear state error based extended Kalman filters with applications to navigation". PhD thesis. Mines Paristech, 2015.

[2] Axel Barrau and Silvère Bonnabel. "The invariant extended Kalman filter as a stable observer". In: *IEEE Transactions on Automatic Control* 62.4 (2016), pp. 1797–1812.

[3] Axel Barrau and Silvere Bonnabel. "An EKF-SLAM algorithm with consistency properties". In: *arXiv preprint arXiv:1510.06263* (2015).

[4] Billur Barshan and Hugh F Durrant-Whyte. "Inertial navigation systems for mobile robots". In: *IEEE transactions on robotics and automation* 11.3 (1995), pp. 328–342.

[5] Michael Bloesch, Sammy Omari, Marco Hutter, and Roland Siegwart. "Robust visual inertial odometry using a direct EKF-based approach". In: *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE. 2015, pp. 298–304.

[6] Silvere Bonnabel, Philippe Martin, and Pierre Rouchon. "Non-linear symmetry-preserving observers on Lie groups". In: *IEEE Transactions on Automatic Control* 54.7 (2009), pp. 1709–1713.

[7] Silvère Bonnable, Philippe Martin, and Erwan Salaün. "Invariant extended Kalman filter: theory and application to a velocity-aided attitude estimation problem". In: *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*. IEEE. 2009, pp. 1297–1304.

[8] Jack CK Chou. "Quaternion kinematic and dynamic differential equations". In: *IEEE Transactions on robotics and automation* 8.1 (1992), pp. 53–64.

[9] Saptadeep Debnath, Anthony Liang, Gaurav Manda, Sunbochen Tang, and Hao Zhou. "Invariant Extended Kalman Filtering for Robot Localization using IMU and GPS". In: ().

[10] RL Farrenkopf. "Analytic steady-state accuracy solutions for two common spacecraft attitude estimators". In: *Journal of Guidance and Control* 1.4 (1978), pp. 282–284.

[11] Brian C Hall et al. *Lie groups, Lie algebras, and representations: an elementary introduction*. Vol. 10. Springer, 2003.

[12] Jeffrey Humpherys, Preston Redd, and Jeremy West. "A Fresh Look at the Kalman Filter". In: *SIAM Review* 54 (Nov. 2012). DOI: 10.1137/100799666.

[13] Ern J Lefferts, F Landis Markley, and Malcolm D Shuster. "Kalman filtering for spacecraft attitude estimation". In: *Journal of Guidance, Control, and Dynamics* 5.5 (1982), pp. 417–429.

[14] Mingyang Li and Anastasios I Mourikis. "High-precision, consistent EKF-based visual-inertial odometry". In: *The International Journal of Robotics Research* 32.6 (2013), pp. 690–711.

[15] F Landis Markley. "Attitude error representations for Kalman filtering". In: *Journal of guidance, control, and dynamics* 26.2 (2003), pp. 311–317.

[16] Anastasios I Mourikis, Stergios I Roumeliotis, et al. "A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation." In: *ICRA*. Vol. 2. 2007, p. 6.

[17] Jon M Selig. *Geometric fundamentals of robotics*. Vol. 128. Springer, 2005.

[18] Dan Simon. *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons, 2006.

[19] David Titterton, John L Weston, and John Weston. *Strapdown inertial navigation technology*. Vol. 17. IET, 2004.

[20] Mohammad Zamani, Jochen Trumpf, and R Mahony. "Nonlinear attitude filtering: A comparison study". In: *arXiv preprint arXiv:1502.03990* (2015).

[21] Fuzhen Zhang. "Quaternions and matrices of quaternions". In: *Linear algebra and its applications* 251 (1997), pp. 21–57.

Appendix A.

Code listing

A.1. Simple car model - EKF & LIEKF python code

```
"""
@author: Thilogen
"""
#-------------------------------------------------------------------

import numpy as np
import math
import matplotlib.pyplot as plt

#-------------------------------------------------------------------

# Define some functions needed to calculate the necessary equations

def skew2(a):
    return a*np.array([[0, -1],[1, 0]])
def exp2(theta):
    #theta = theta.item()
    return np.array([[np.cos(theta), -np.sin(theta)],
                      [np.sin(theta), np.cos(theta)]])
def Ese2(theta):
    a = np.sinc(theta/np.pi)
    b = (theta/2)*(np.square(np.sinc(theta/(2*np.pi))))
    return np.array([[a, -b], [b, a]])

#-------------------------------------------------------------------

# Define the function that creates the real trajectory

def create_dynamics(n_time, h, om, v):
    X = np.zeros((3,n_time))
    for i in range(1,n_time):
        theta = X[0,i-1]; x = X[1:3,i-1]
        X[:,i] = np.array([theta + om * h,
                           x[0] + np.cos(theta)* v * h,
                           x[1] + np.sin(theta)* v * h])
    return X

#-------------------------------------------------------------------

#Define the propagate function for both filter

def propagate(h, Xh, P, v, om, Q, config):
    theta_h = Xh[0]; x_h = Xh[1:3]; R_h = exp2(theta_h)

    if config == "ekf":
```

```
        A = np.zeros((3,3))
        A[1,0] = -np.sin(theta_h)*v; A[2,0] = np.cos(theta_h)*v

        P_p = P + h*(A @ P + P @ A.T + Q)


        Xh_p = np.array([theta_h + om * h,
                         x_h[0] + np.cos(theta_h)* v * h,
                         x_h[1] + np.sin(theta_h)* v * h])


    elif config == "liekf":
        A = np.zeros((3,3))
        A[2,0] = -v; A[1:3,1:3] = skew2(om)
        A = -A

        P_p = P + h*(A @ P + P @ A.T + Q)



        Xhi = np.block([[R_h, x_h.reshape(2,1)],
                        [0,0,1]])

        v_lie = np.zeros((3,3))
        v_lie[0:2, 0:2] = skew2(om); v_lie[0,2] = v
        Xhi_p = Xhi + h * (Xhi @ v_lie)
        Xh_p = Xhi_p


    return Xh_p, P_p

#Define the update function for both filter

def update(Xh_p, P_p, Y, N, config, step):

    if config == "ekf":
        Rh_p = exp2(Xh_p[0])
    elif config == "liekf":
        Rh_p = Xh_p[0:2,0:2]


    N_hat = Rh_p @ N @ Rh_p.T
    H = np.hstack((np.zeros((2,1)), np.eye(2)))

    S = H @ P_p @ H.T + N_hat
    L_n = P_p @ H.T @ np.linalg.inv(S)

    if config == "ekf":
        Yn = Y - Xh_p[1:3]
        Xh_u = Xh_p + (L_n @ Yn)

    elif config == "liekf":
        Xhi_p = Xh_p

        Yn = np.block([Y, 1])

        inno = (np.linalg.inv(Xhi_p) @ Yn) - np.array([0,0,1])
        Ln_tilde = np.hstack((L_n, np.zeros((3,1))))
        delta = Ln_tilde @ inno

        E_delta = Ese2(delta[0]); R_delta = exp2(delta[0])
        exp_delta = np.block([[np.hstack((R_delta,np.array([E_delta @ delta[1:3]]).T)
    )],
```

```
                              [np.array([0,0,1])]])
        Xhi_u = Xhi_p @ exp_delta

        # Using Atan(y,x) to find theta, where y = sin(theta) and x = cos(theta)
        thetah_u = math.atan2(Xhi_u[1,0], Xhi_u[0,0]); xh_u = Xhi_u[0:2,2]
        if (thetah_u < 0 and k > n_time/2):
            thetah_u += 2*np.pi
        Xh_u = np.hstack((thetah_u, xh_u))

    P_u = (np.eye(3) - L_n @ H) @ P_p

    return Xh_u, P_u

#-------------------------------------------------------------------

#Define the system initialization parameters for the simple car model

v = 1; t_circ = 40; om = 2*np.pi/t_circ; h = 0.1
n_rounds = 4/5
n_time = int(n_rounds * t_circ/h)


X = create_dynamics(n_time, h, om, v)
#-------------------------------------------------------------------

# Define the noise intialization parameters

N = np.eye(2)
Q = np.diag(((np.pi/180)**2,1e-4,1e-4))

#-------------------------------------------------------------------

filters = ["liekf","ekf"]

Xu_ekf = np.zeros((3, n_time))
Pu_ekf = np.zeros((3, 3, n_time))
Xu_liekf = np.zeros((3, n_time))
Pu_liekf = np.zeros((3, 3, n_time))

# Define heading error (degrees)

heading_error = 45

if heading_error == 45:
    Pu_ekf[:,:,0] = np.diag([(15*np.pi/180)**2, 0, 0])
    Pu_liekf[:,:,0] = np.diag([(15*np.pi/180)**2, 0, 0])
elif heading_error == 1:
    Pu_ekf[:,:,0] = np.diag([(np.pi/180)**2, 0, 0])
    Pu_liekf[:,:,0] = np.diag([(np.pi/180)**2, 0, 0])


# Creating initial theta given the heading_error

Xu_ekf[0,0] = - heading_error * (np.pi/180)
Xu_liekf[0,0] = - heading_error * (np.pi/180)

Xu_ekf[0,0] = -heading_error * (np.pi/180)
Xu_liekf[0,0] = -heading_error * (np.pi/180)

#-------------------------------------------------------------------

plt.figure(1)
plt.figure(1).clear()
```

```
plt.plot(X[1], X[2],"g", label = "True trajectory")

#Iterate through the filter list and calculate and plot trajectory for each filter

for config in filters:

    for k in range(1, n_time):

        Y = X[1:3,k]

        if config == "ekf":
            Xp, Pp = propagate(h, Xu_ekf[:,k-1].copy(), Pu_ekf[:,:,k-1].copy(), v, om, Q, config)
            Xu_ekf[:,k], Pu_ekf[:,:,k] = update(Xp, Pp, Y, N, config, k)
        elif config == "liekf":
            Xp, Pp = propagate(h, Xu_liekf[:,k-1].copy(), Pu_liekf[:,:,k-1].copy(), v, om, Q, config)
            Xu_liekf[:,k], Pu_liekf[:,:,k] = update(Xp, Pp, Y, N, config, k)


    if config == "ekf":
        plt.plot(Xu_ekf[1,:], Xu_ekf[2,:], "b--", label = "EKF - Estimated trajectory")

    elif config == "liekf":
        plt.plot(Xu_liekf[1,:], Xu_liekf[2,:], "r--", label = "LIEKF - Estimated trajectory")

plt.xlabel("x [m]")
plt.ylabel("y [m]")
plt.xlim([-10, 10])
plt.ylim([-2, 14])

plt.title(f"Attitude filtering, initial heading error = {heading_error}\N{DEGREE SIGN}")

plt.legend()

plt.savefig(f"attitude_filtering_{heading_error}.png")
plt.show()

#-------------------------------------------------------------------

#Define the error of the estimated trajectories

err_ekf = abs(X - Xu_ekf)
err_liekf = abs(X - Xu_liekf)

t = h*np.arange(n_time)

pos_err_liekf = np.zeros(n_time)
pos_err_ekf = np.zeros(n_time)
for i in range(n_time):
    pos_err_liekf[i] = err_liekf[1:3,i].sum()
    pos_err_ekf[i] = err_ekf[1:3,i].sum()

#-------------------------------------------------------------------

#Plot the attitude error

plt.figure(2)
plt.figure(2).clear()
```

```
plt.plot(t, err_liekf[0,:] / (np.pi/180), "r--", label = "LIEKF attitude error")
plt.plot(t, err_ekf[0,:] / (np.pi/180), "b--", label = "EKF attitude error")

plt.xlabel("time [s]")
plt.ylabel("Attitude error [degrees]")

plt.title(f"Attitude error, initial heading error = {heading_error}\N{DEGREE SIGN}")


plt.legend()

plt.xlim([0, n_time*h])
plt.savefig(f"attitude_filtering_attitude_error_{heading_error}.png")
plt.show()

#-------------------------------------------------------------------

#Plot the position error

plt.figure(3)
plt.figure(3).clear()

plt.plot(t, pos_err_liekf, "r--", label = "LIEKF position error")
plt.plot(t, pos_err_ekf, "b--", label = "EKF position error")

plt.xlabel("time [s]")
plt.ylabel("Position error [m]")
plt.title(f"Position error, initial heading error = {heading_error}\N{DEGREE SIGN}")

plt.legend()

plt.xlim([0, n_time*h])
plt.savefig(f"attitude_filtering_position_error_{heading_error}.png")
plt.show()
```

A.2. Navigation on flat earth - MEKF & RIEKF python code

```
"""
@author: Thilogen
"""

import numpy as np
import matplotlib.pyplot as plt

#------------------------------------------------------------------

# Define some functions needed to calculate the necessary equations

def skewm(r):
    return np.array([[0,-r[2],r[1]], [r[2],0,-r[0]], [-r[1],r[0],0]])
def vex(u):
    return np.array([u[2,1], u[0,2], u[1,0]])
def expso3(u):
    S = skewm(u); un = np.linalg.norm(u)
    return np.identity(3) + np.sinc(un/np.pi)*S + 0.5*(np.sinc(un/(2*np.pi)))**2 *
    S@S

def expse2_3(u):
    un = np.linalg.norm(u[0:3])
    if un > 0.000001:
        a = ((1-np.cos(un)) / un**2)
        b = (un - np.sin(un))/(un**3)
    else:
        a = 1/2 + (un**2)/24
        b = 1/6 + (un**2)/120
    S = np.block([[skewm(u[0:3]), u[3:6].reshape(3,1), u[6:9].reshape(3,1)],
                  [np.zeros((2,5))]])
    return np.identity(5) + S + a*S@S + b*S@S@S

def integrate_se3(h, X, om, u):
    R = X[0:3,0:3]; v = X[0:3,3]; x = X[0:3,4]
    R_int = R @ expso3(h*om)
    v = np.array([1.0, 0.0, 0.0])
    x_int = x + h * R @ JLso3(h*om) @ v
    return np.hstack((R_int, v.reshape(3,1), x_int.reshape(3,1)))

def JLso3(u):
    theta = np.linalg.norm(u); S = skewm(u)
    a = 0.5 * (np.sinc(theta/(2*np.pi)))**2
    if theta > 0.000001:
        b = (theta - np.sin(theta))/(theta**3)
    else:
        b = 1/6 + theta**2/120
    return np.eye(3) + a*S + b*S@S

def logSO3(R):
    # The vector form of the logarithm in SO(3) (Robotic style)
    theta = np.arccos(0.5 * (np.trace(R) - 1))
    ct = np.cos(theta); vt = 1-ct
    if theta < 0.000001:
        f = 1 + (theta**2)/6
        u = vex(f * 0.5 * (R-R.T))
    if np.pi - theta < 0.00001:
        ct = np.cos(theta); vt = 1-ct
        if R[0,0] - ct > 0.5:
            kx = np.sqrt((R[0,0] - ct)/vt)
            ky = (R[0,1] + R[1,0]) / (2*kx*vt)
            kz = (R[2,0] + R[0,2]) / (2*kx*vt)
        elif R[1,1] - ct > 0.5:
```

```
            ky = np.sqrt((R[1,1] - ct)/vt)
            kz = (R[1,2] + R[2,1]) / (2*ky*vt)
            kx = (R[0,1] + R[1,0]) / (2*ky*vt)
        else:
            kz = np.sqrt((R[2,2] - ct)/vt)
            kx = (R[2,0] + R[0,2]) / (2*kz*vt)
            ky = (R[1,2] + R[2,1]) / (2*kz*vt)
            u = theta * np.array([kx, ky, kz])
    else:
        f = theta / np.sin(theta)
        u = vex(f * 0.5 * (R-R.T))
    return u

#--------------------------------------------------------------------

# Define the function that creates the real trajectory

def create_dynamics(n_time, h, om, v, u):
    X = np.zeros((3,5,n_time))
    X[:,0:3, 0] = np.eye(3)
    X[:,3, 0] = v
    for i in range(1,n_time):
        X[:,:,i] = integrate_se3(h, X[:,:,i-1].copy(), om, u)
    return X

def generate_map(v, omega, delta_r, n_L):
    p_L = np.zeros((3,n_L)); r = v/omega
    for i in range(n_L):
        rho = r - 2*delta_r
        theta = 2*np.pi*i/n_L
        p_L[:,i] = [rho*np.cos(theta), rho*np.sin(theta)+5.0, 0.8*((-1)**i)]
    return p_L

#--------------------------------------------------------------------

#Define the propagate function for both filter

def propagate(Xh, P, h, om, u, g, Q, config):
    Rh = Xh[:, 0:3]; vh = Xh[:,3]; xh = Xh[:,4]

    zero_3 = np.zeros((3,3))
    A = np.zeros((9,9))
    if config == "mekf":
        A[3:6,0:3] = -skewm(Rh @ u); A[6:9,3:6] = np.eye(3)
    elif config == "riekf":
        A[3:6,0:3] = skewm(g); A[6:9,3:6] = np.eye(3)


    G = np.block([[Rh, zero_3, zero_3],
          [skewm(vh), Rh, zero_3],
          [skewm(xh), zero_3, Rh]])

    Q_hat = G @ Q @ G.T

    P_p = P + h * (A @ P + P @ A.T + Q_hat)


    if config == "mekf":
        Xh_p = integrate_se3(h, Xh, om, u)
    elif config == "riekf":
        Xh_p = np.block([[integrate_se3(h, Xh, om, u)],
                         [np.zeros((2,3)), np.eye(2)]])

```

```
    return Xh_p, P_p

#Define the update function for both filter

def update(Xh_p, P_p, Y, N, config):
    Rh_p = Xh_p[0:3,0:3]; vh_p = Xh_p[0:3,3]; xh_p = Xh_p[0:3,4]
    R = Y[0:3,0:3]; x = Y[0:3,4]

    yn = np.zeros(3 * n_L) # 1x9 - vec
    yn_h = np.zeros(3 * n_L) # 1x9 - vec

    for i in range(n_L):
        yn[3*i:3*i+3] = R.T @ (p_L[:,i] - x)
        yn_h[3*i:3*i+3] = Rh_p.T @ (p_L[:,i] - xh_p)

    N_hat = np.zeros((3*n_L,3*n_L))
    H = np.zeros((3*n_L,3*n_L))

    for i in range(n_L):
        N_hat[3*i:3*i+3, 3*i:3*i+3] = Rh_p @ N[3*i:3*i+3, 3*i:3*i+3] @ Rh_p.T


    if config == "mekf":
        for i in range(n_L):
            H[3*i:3*i+3, 0:3] = Rh_p.T @ skewm(p_L[:,i] - xh_p)
            H[3*i:3*i+3, 6:9] = -Rh_p.T

    elif config == "riekf":
        for i in range(n_L):
            H[3*i:3*i+3, 0:3] = skewm(p_L[:,i])
            H[3*i:3*i+3, 6:9] = -np.eye(3)

    S = H @ P_p @ H.T + N_hat # 9x9 - mat
    L_n = P_p @ H.T @ np.linalg.inv(S) # 9x9 - mat

    if config == "mekf":
        inno =  -L_n @ (yn_h - yn) # 9x1 vec

        Xh_u = np.zeros((3,5))
        Xh_u[:,0:3] = Xh_p[:,0:3] @ expso3(inno[0:3])
        Xh_u[:,3] = Xh_p[:,3] + inno[3:6]
        Xh_u[:,4] = Xh_p[:,4] + inno[6:9]

    elif config == "riekf":
        yn_tilde = np.zeros(3 * n_L) # 1x9 - vec
        for i in range(n_L):
            yn_tilde[3*i:3*i+3] = Rh_p @ (yn_h[3*i:3*i+3] - yn[3*i:3*i+3]) # 1x3 - vec

        delta = -L_n @ yn_tilde # 9x1 - mat

        Xhi_u = expse2_3(delta) @ Xh_p
        Xh_u = Xhi_u[0:3,:]



    P_u = (np.eye(9) - L_n @ H) @ P_p


    return Xh_u, P_u

#----------------------------------------------*---------------------
```

```

# Define the system initialization parameters

radius = 5.0; t_circ = 30.0; omega = 2*np.pi/t_circ; v_t = omega * radius
h = 0.1
n_time = 1*int(t_circ/h)

n_L = 3

ex = np.array([1,0,0]); ey = np.array([0,1,0]); ez = np.array([0,0,1])
g =  -9.81 * ez
v_3d = v_t * ex; om_3d = omega * ez; u_3d = omega**2 * radius * ey + g

#-------------------------------------------------------------------

init_heading_est_error =  -15 * np.pi/180
theta_t_i = init_heading_est_error

# Define the noise intialization parameters

I_3 = np.eye(3)
zero_3 = np.zeros((3,3))

N = (1/h) * 1.0e-2 * np.block([[I_3, zero_3, zero_3],
                               [zero_3, I_3, zero_3],
                               [zero_3, zero_3, I_3]])

Q1 = 1.0e-8 * np.block([[I_3, zero_3, zero_3],
                     [zero_3, I_3, zero_3],
                     [zero_3, zero_3, zero_3]])


Q2 = 1.0e-4 * np.block([[I_3, zero_3, zero_3],
                     [zero_3, I_3, zero_3],
                     [zero_3, zero_3, zero_3]])


Q10 =  np.block([[((5*np.pi/180)**2)*I_3, zero_3, zero_3],
                 [zero_3, 1e-2*I_3, zero_3],
                 [zero_3, zero_3, I_3]])


Q20 =  np.block([[((15*np.pi/180)**2)*I_3, zero_3, zero_3],
                 [zero_3, 1e-2*I_3, zero_3],
                 [zero_3, zero_3, I_3]])



#-------------------------------------------------------------------

# Create the true trajectory and true landmark positions

X = create_dynamics(n_time, h, om_3d, v_3d, u_3d)
p_L = generate_map(v_t, omega, 1, n_L)

filters = ["mekf","riekf"]

cov_om = Q2

Xu_mekf = np.zeros((3,5, n_time))
Xu_riekf = np.zeros((3,5, n_time))

Rt_init = expso3(np.array([0,0, init_heading_est_error]))
```

```

#Attitude
Xu_mekf[:,0:3,0] = Rt_init
Xu_riekf[:,0:3,0] = Rt_init

#Velocity
Xu_mekf[:,3,0] = Rt_init @ v_3d
Xu_riekf[:,3,0] = Rt_init @ v_3d

#Position
Xu_mekf[:,4,0] = np.array([1.0, 0.0, 1.0])
Xu_riekf[:,4,0] = np.array([1.0, 0.0, 1.0])

Pu_riekf = np.zeros((9, 9, n_time))
Pu_mekf = np.zeros((9, 9, n_time))

if np.array_equal(cov_om, Q1):
    Pu_riekf[:,:,0] = Q10
    Pu_mekf[:,:,0] = Q10

elif np.array_equal(cov_om, Q2):
    Pu_riekf[:,:,0] = Q20
    Pu_mekf[:,:,0] = Q20

#-------------------------------------------------------------------

plt.figure(1)
plt.figure(1).clear()
ax = plt.axes(projection='3d')

ax.plot3D(p_L[0,:], p_L[1,:], p_L[2,:],"ko", label = "Landmarks")
ax.plot3D(p_L[0,:], p_L[1,:], np.zeros(3),"kx")
for i in range(n_L):
    px = np.hstack((p_L[0,i], p_L[0,i]))
    py = np.hstack((p_L[1,i], p_L[1,i]))
    pz = np.hstack((p_L[2,i], 0))
    ax.plot3D(px, py, pz, "k--")

ax.plot3D(X[0,4,:], X[1,4,:], X[2,4,:],"g", label = "True trajectory")

#Iterate through the filter list and calculate and plot trajectory for each filter

for config in filters:
    for k in range(1, n_time):
        Y = X[:,:,k]
        if config == "mekf":
            Xp, Pp = propagate(Xu_mekf[:,:,k-1].copy(), Pu_mekf[:,:,k-1].copy(),
                                                                      h, om_3d, u_3d, g, cov_om, config)
            Xu_mekf[:,:,k], Pu_mekf[:,:,k] = update(Xp, Pp, Y, N, config)
        elif config == "riekf":
            Xp, Pp = propagate(Xu_riekf[:,:,k-1].copy(), Pu_riekf[:,:,k-1].copy(),
                                                                      h, om_3d, u_3d, g, cov_om, config)
            Xu_riekf[:,:,k], Pu_riekf[:,:,k] = update(Xp, Pp, Y, N, config)

    if config == "mekf":
        ax.plot3D(Xu_mekf[0,4,:], Xu_mekf[1,4,:], Xu_mekf[2,4,:], "b--", label = "MEKF - Estimated trajectory")


    elif config == "riekf":
        ax.plot3D(Xu_riekf[0,4,:], Xu_riekf[1,4,:], Xu_riekf[2,4,:],"r--", label = "
```

```
    RIEKF - Estimated trajectory")


ax.set_zlim(-1,5)
plt.legend()


if np.array_equal(cov_om, Q1):
    plt.title("Pose estimation, Q = Q1")
    plt.savefig("navigation_pose_estimation_Q1.png")
elif np.array_equal(cov_om, Q2):
    plt.title("Pose estimation, Q = Q2")
    plt.savefig("navigation_pose_estimation_Q2.png")


plt.show()

#-------------------------------------------------------------------

#Plot the attitude error

plt.figure(2)
plt.figure(2).clear()

t = h*np.arange(n_time)


riekf_attitude_error = np.zeros(n_time)
mekf_attitude_error = np.zeros(n_time)

for i in range(n_time):
    riekf_attitude_error[i] = abs(logSO3(X[0:3,0:3,i] @ Xu_riekf[0:3,0:3,i].T).sum())
    mekf_attitude_error[i] = abs(logSO3(X[0:3,0:3,i] @ Xu_mekf[0:3,0:3,i].T).sum())



plt.plot(t, riekf_attitude_error / (np.pi/180), "r--", label = "RIEKF attitude error"
    )
plt.plot(t, mekf_attitude_error / (np.pi/180), "b--", label = "MEKF attitude error")



plt.xlim([0, n_time*h])
plt.ylim([0, 20])
plt.legend()
plt.xlabel("time [s]")
plt.ylabel("Attitude error [degrees]")

if np.array_equal(cov_om, Q1):
    plt.title("Attitude error, Q = Q1")
    plt.savefig("navigation_attitude_error_Q1.png")
elif np.array_equal(cov_om, Q2):
    plt.title("Attitude error, Q = Q2")
    plt.savefig("navigation_attitude_error_Q2.png")

plt.show()

#-------------------------------------------------------------------

#Plot the position error

plt.figure(3)
plt.figure(3).clear()
```

```

riekf_position_error = np.zeros(n_time)
mekf_position_error = np.zeros(n_time)

for i in range(n_time):
    riekf_position_error[i] = np.linalg.norm(X[0:3,4,i] - Xu_riekf[0:3,4,i])
    mekf_position_error[i] = np.linalg.norm(X[0:3,4,i] - Xu_mekf[0:3,4,i])


plt.plot(t, riekf_position_error, "r--", label = "RIEKF position error")
plt.plot(t, mekf_position_error, "b--", label = "MEKF position error")
plt.xlim([0, n_time*h])
plt.ylim([0, 2])
plt.xlabel("time [s]")
plt.ylabel("Position error [m]")
plt.legend()
if np.array_equal(cov_om, Q1):
    plt.title("Position error, Q = Q1")
    plt.savefig("navigation_position_error_Q1.png")
elif np.array_equal(cov_om, Q2):
    plt.title("Position error, Q = Q2")
    plt.savefig("navigation_position_error_Q2.png")
plt.show()
```

NTNU

Norwegian University of
Science and Technology