Technical University of Denmark

Advanced State Estimation

Particle Filtering Based Inertial Navigation System

Morten Nissov, s163962

January 2020

This page is intentionally left blank.

Contents

Abbreviations

DCM Direction Cosine Matrix. 6, 7, 39, 41

ECEF Earth-Centered, Earth-Fixed. ii, 4, 5

EKF Extended Kalman Filter. 2, 14, 20, 44

GNSS Global Navigation Satellite System. ii, 2, 4, 10, 13, 15, 20, 21, 26, 27, 34, 37, 44

IMU Inertial Measurement Unit. ii, 2, 4, 10, 12, 13, 20–23, 26, 29, 41, 44

INS Intertial Navigation System. ii, 2, 10, 11, 13–15, 44

KF Kalman Filter. 15

MC Monte Carlo. 15

NED North East Down. 4, 5

PF Particle Filter. ii, 2, 15–20, 31–38, 40–44

RMSE Root Mean Squared Error. 34, 37

SIS Sequential Importance Sampling. ii, 15, 16, 31–33, 35, 36, 38, 40, 44

SISR Sequential Importance Sampling with Resampling. 18

UKF Unscented Kalman Filter. 14

Introduction

This paper will be looking into solving the filtering problem for a strapdown Intertial Navigation System (INS) with an aiding sensor.

The filtering problem for a strapdown INS is the problem of generating reliable position, attitude, and velocity estimates by integrating the acceleration and angular velocity measurements from the Inertial Measurement Unit (IMU) and comparing with an aiding sensor. The aiding sensor in the context of this paper is a Global Navigation Satellite System (GNSS) unit, which returns position measurements, but this is not necessarily always the case. Note that "strapdown" refers to the fact that these sensors are mounted rigidly to the frame, i.e. not on an actuated platform. The INS in this paper is in the context of marine applications which carries some implicit assumptions, for example generally slow moving, slowly accelerating vehicles moving in relatively confined regions.

The traditional solution to this problem is framed in an Extended Kalman Filter (EKF) framework, this is not necessarily ideal given that the EKF works by applying a first order approximation on the nonlinear systems and assumes Gaussian processes for noise. The latter is especially problematic for IMU readings as the accelerometer and gyroscope measurements will not necessarily be Gaussian [1]. In the traditional approach the assumption of Gaussianity can be avoided by estimating the error states, instead of full, and creating a complementary filter. Formulating the dynamics equations for these error states involves some first order approximations as well [1], which is fine given the fact that the EKF is already a first order approximation.

This paper proposes and implements a Particle Filter (PF) approach in order to preserve the entire nonlinearity of the model as opposed to the first order approximation alternative. In addition the PF makes no inherent assumption of the noise distribution, e.g. Gaussian in the EKF case, and thereby removes the necessity for an error state model. This means that there should be potential for increased performance from a PF implementation, though at the cost of increased computational requirements.

Notation

The notation that will be used throughout this paper is summarized in table 1.

Symbol	Meaning
x	non-bold variable denotes scalar quantity
$\mathbf{x}$	bold variable denotes a vector/matrix quantity
$\mathbf{x}$	denotes true value of $\mathbf{x}$
$\dot{\mathbf{x}}$	denotes the time derivative of $\mathbf{x}$
$\hat{\mathbf{x}}$	denotes the calculated (estimated) value of $\mathbf{x}$
$\tilde{\mathbf{x}}$	denotes the measured value of $\mathbf{x}$
$\mathbf{x}^a$	denotes the value $\mathbf{x}$ represented with respect to frame a
$[\mathbf{x}\times]$	denotes the skew symmetric matrix formed from $\mathbf{x}$
$\|\|\mathbf{x}\|\|_2$	denotes the Euclidean norm of vector $\mathbf{x}$, i.e. $\sqrt{x_1^2 + x_2^2 + \dots x_n^2}$
$\mathbf{R}_a^b$	denotes the rotation matrix from frame a to b
$\mathbf{f}$	denotes the specific force, i.e. the accelerometer measurement
$\boldsymbol{\omega}$	denotes the angular velocity, i.e. the gyroscope measurement
$p(x\|y)$	denotes the probability distribution of x given y
$x \sim p$	denotes points x sampled from distribution p

Table 1: List of the most commonly used symbols and notations in the following sections as well as their meanings.

1. Reference Frames

In order to properly understand the following sections it is important to have a grasp on the relevant reference frames. Given that this paper is working in the context of marine vessels it is important to consider the assumptions that follow with that.

At minimum there will be two frames of importance: an inertial frame and a body frame. This is because the IMU sensor package measures effects on the body frame with respect to an inertial frame. In some circumstances it can be necessary to introduce a third frame for the GNSS measurements but here it is assumed that the GNSS measurements are with respect to the inertial frame as well.

The relevant frames are described below:

1.1 Earth-Centered, Earth-Fixed

The Earth-Centered, Earth-Fixed (ECEF) frame will not be used much throughout this paper, but it is useful when considering the other, more used frames. The ECEF frame has its origin at the center of the Earth and rotates with the planet, therefore it is not an inertial frame.

This frame uses coordinates of latitude λ, longitude ϕ, and height h to describe the position of a point on the Earth. Note that both the longitude and roll angle use ϕ. The usage should be clear based upon the context, but take care.

A possible definition of the ECEF frame axes can be seen in fig. 1.1.

1.2 Inertial Frame

Generally speaking an inertial frame is a non-rotating, non-accelerating frame that has no external forces acting on it. The inertial frame of interest here is the local tangent frame, also called the North East Down (NED) frame. This frame is constructed by taking the tangent plane at a point on the Earth's surface such as in fig. 1.2. Technically speaking the tangent frame is not an inertial frame, because it rotates with the Earth. But given a slow moving vessel which operates within a small, local region on the tangent plane the rotation is negligible and thereby the tangent frame is sufficiently inertial.

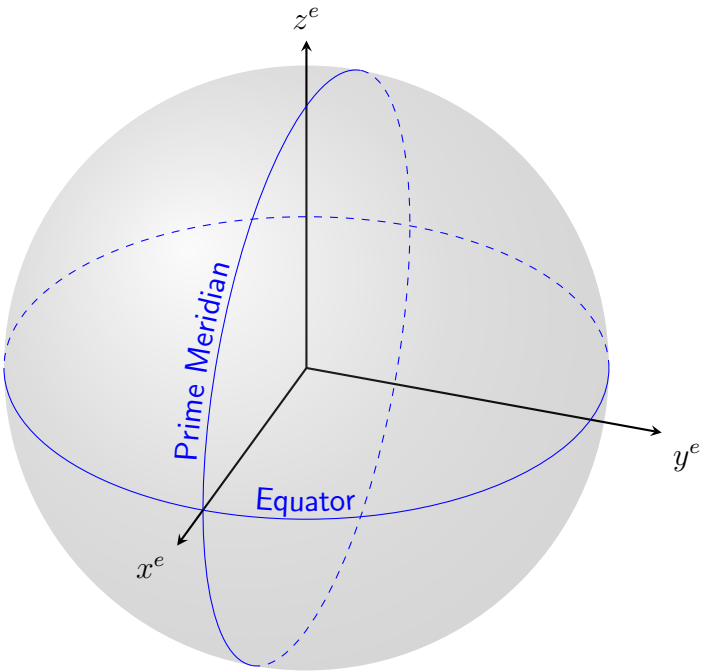


Figure 1.1: ECEF rectangular frame defined with x_e along the prime meridian.

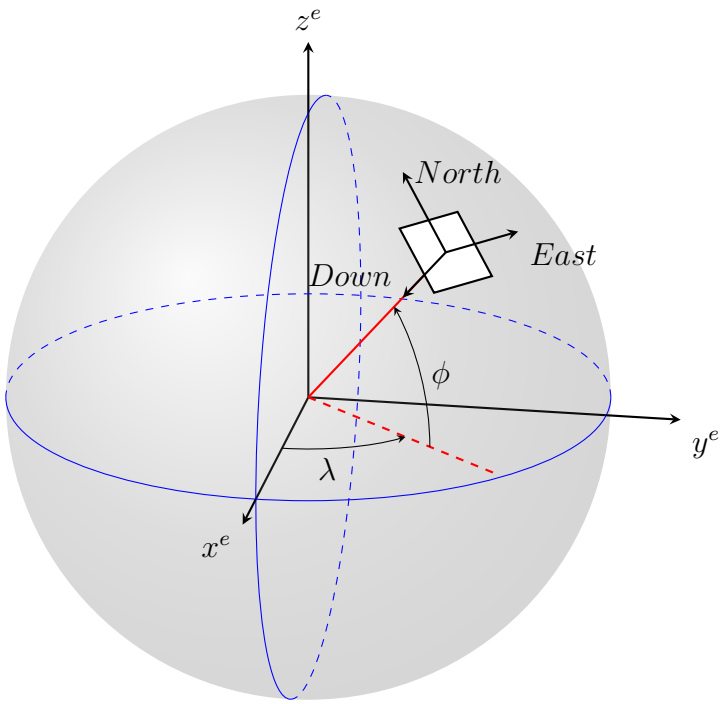


Figure 1.2: NED frame defined by a tangent plane at a point, (λ, ϕ, h), on the Earth's surface.

1.3 Body Frame

The body frame has been chosen to be as convenient as possible with respect to the tangent frame. The body frame is the same as the tangent frame, except it is fixed to the center of gravity of the vehicle with the north axis x pointing in the forward facing direction of the vehicle, refer to fig. 1.3.

The Euler angles are given as roll ϕ, pitch θ, and yaw ψ.

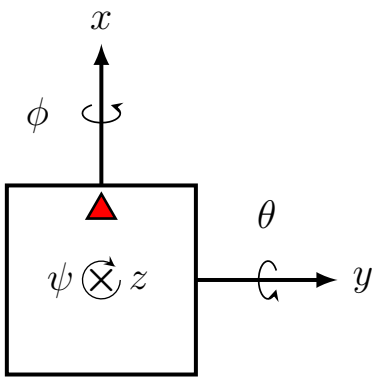


Figure 1.3: Body frame for the vehicle where the red arrow denotes the forward facing direction.

1.4 Frame Transformations

Given that one is working with several different frames it is vital to be able to rotate between frames. There are three main ways to represent such a rotation: Euler angles, Direction Cosine Matrix (DCM), and quaternions.

In order to avoid working with the rotation orders and singularities associated with Euler angles the DCM or quaternion approach can be pursued. This paper will pursue a DCM representation of the rotation.

However, regardless of the methodology to calculate the rotation matrix the usage remains the same, see appendix A.1.

1.4.1 Direction Cosine Matrix Formulation

Given two reference frames a and b which are defined by their unit vectors $\mathbf{i}$, $\mathbf{j}$, and $\mathbf{k}$ such that a point with respect to frame a is given by

$$\mathbf{v}^a = \begin{bmatrix} \mathbf{i}_a \cdot \mathbf{v} \\ \mathbf{j}_a \cdot \mathbf{v} \\ \mathbf{k}_a \cdot \mathbf{v} \end{bmatrix} \tag{1.1}$$

where $\mathbf{v}$ is the vector $\begin{bmatrix} x & y & z \end{bmatrix}^T$ defined with respect to no reference frame.

The rotation matrix is then made by creating the projections of each of the unit vectors in one coordinate frame onto the unit vectors in the other coordinate

frame. This will define a "mapping" where each unit vector in frame a is given by a combination of the unit vectors in frame b, for example:

$$\begin{aligned} \mathbf{v}_i &= \begin{bmatrix} \mathbf{i}_b \cdot \mathbf{i}_a \\ \mathbf{j}_b \cdot \mathbf{i}_a \\ \mathbf{k}_b \cdot \mathbf{i}_a \end{bmatrix} \\ \mathbf{v}_j &= \begin{bmatrix} \mathbf{i}_b \cdot \mathbf{j}_a \\ \mathbf{j}_b \cdot \mathbf{j}_a \\ \mathbf{k}_b \cdot \mathbf{j}_a \end{bmatrix} \\ \mathbf{v}_k &= \begin{bmatrix} \mathbf{i}_b \cdot \mathbf{k}_a \\ \mathbf{j}_b \cdot \mathbf{k}_a \\ \mathbf{k}_b \cdot \mathbf{k}_a \end{bmatrix} \end{aligned} \tag{1.2}$$

This represents the unit vectors of the a frame described by unit vectors in the b frame and with these the rotation matrix is

$$\mathbf{R}_a^b = \begin{bmatrix} \mathbf{v}_i & \mathbf{v}_j & \mathbf{v}_k \end{bmatrix} \tag{1.3}$$

and that is the DCM formulation of the rotation matrix from frame a to b.

This matrix can also be formulated using angles, which is clear to see if one inspects a single element of the rotation matrix:

$$\left[\mathbf{R}_a^b\right]_{1,2} = \mathbf{i}_b \cdot \mathbf{j}_a \tag{1.4}$$

which can be rewritten as

$$\begin{aligned} \left[\mathbf{R}_a^b\right]_{1,2} &= \mathbf{i}_b \cdot \mathbf{j}_a \\ &= ||\mathbf{i}_b||_2 \cdot ||\mathbf{j}_a||_2 \cos\beta_1 \end{aligned} \tag{1.5}$$

Since $\mathbf{i}_b$ and $\mathbf{j}_a$ are unit vectors this can be further simplified to

$$\left[\mathbf{R}_a^b\right]_{1,2} = \cos\beta_1 \tag{1.6}$$

where β_1 is the angle between $\mathbf{i}_b$ and $\mathbf{j}_a$. Given information regarding the angles between the unit vectors the rotation matrix can be written as

$$\mathbf{R}_a^b = \begin{bmatrix} \cos\alpha_1 & \cos\beta_1 & \cos\gamma_1 \\ \cos\alpha_2 & \cos\beta_2 & \cos\gamma_2 \\ \cos\alpha_3 & \cos\beta_3 & \cos\gamma_3 \end{bmatrix} \tag{1.7}$$

Figure 1.4 depicts how the α angles are defined, β and γ are defined similarly for their respective unit vectors.

1.4.2 Matrix Derivative

For estimation purposes it is necessary to have some kind of model for how the rotation matrix changes with respect to time given angular velocities. The derivative of the rotation matrix is defined as

$$\dot{\mathbf{R}}_a^b(t) = \lim_{\delta t \to 0} \frac{\mathbf{R}_a^b(t+\delta t) - \mathbf{R}_a^b(t)}{\delta t} \tag{1.8}$$

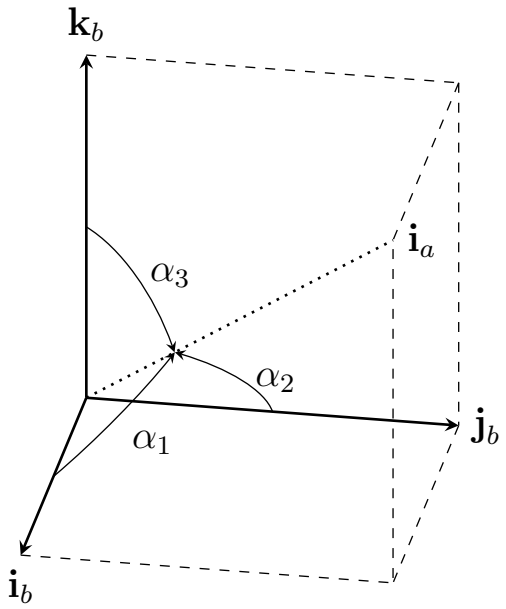


Figure 1.4: Representation of $\mathbf{i}_a$ with respect to unit vectors of frame b.

Assuming a small δt the rotation matrix given by $\mathbf{R}_a^b(t+\delta t)$ can be broken down into a rotation from a to b at time t, $\mathbf{R}_a^b(t)$ followed by a rotation from b at time t to b at time $t+\delta t$, $\mathbf{R}_{b(t)}^{b(t+\delta t)}$. Equivalently written as

$$\mathbf{R}_a^b(t+\delta t) = \mathbf{R}_{b(t)}^{b(t+\delta t)}\mathbf{R}_a^b(t) \tag{1.9}$$

Given that the angular velocity $\boldsymbol{\omega}_{ab}^b$ is finite and that δt approaches zero then the rotation matrix $\mathbf{R}_{b(t)}^{b(t+\delta t)}$ represents the small angle rotation $\delta\boldsymbol{\theta} = \boldsymbol{\omega}_{ab}^b\delta t$ such that

$$\mathbf{R}_{b(t)}^{b(t+\delta t)} = \mathbf{I} - \boldsymbol{\Omega}_{ab}^b\delta t \tag{1.10}$$

Where $\boldsymbol{\Omega}_{ab}^b = [\boldsymbol{\omega}_{ab}^b\times]$, i.e. the skew symmetric matrix formed from $\boldsymbol{\omega}_{ab}^b$, see appendix A.2. Note that it is assumed the angular velocity is represented in the following order

$$\boldsymbol{\omega} = \begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} \tag{1.11}$$

See appendix A.3 for more information regarding small angle rotations.

Equation (1.8) can then be rewritten and simplified

$$\begin{aligned}\dot{\mathbf{R}}_a^b(t) &= \lim_{\delta t\to 0}\frac{\left(\mathbf{I}-\boldsymbol{\Omega}_{ab}^b\delta t\right)\mathbf{R}_a^b(t) - \mathbf{R}_a^b(t)}{\delta t}\\ &= \lim_{\delta t\to 0}\frac{-\boldsymbol{\Omega}_{ab}^b\delta t\mathbf{R}_a^b(t)}{\delta t}\\ &= -\boldsymbol{\Omega}_{ab}^b\mathbf{R}_a^b(t)\end{aligned} \tag{1.12}$$

Using eq. (A.8) and the fact that $\boldsymbol{\Omega}_{ab}^a = -\boldsymbol{\Omega}_{ba}^a$ eq. (1.12) can also be expressed as

$$\dot{\mathbf{R}}_a^b = \mathbf{R}_a^b\boldsymbol{\Omega}_{ba}^a \tag{1.13}$$

Note this is not the only way to derive an expression for the derivative of a rotation matrix, see [2].

2. Mathematical Formulation

The following sections will document the mathematical formulation of the mechanization equations as well as the sensor models for the INS and GNSS systems, resulting in the complete model to be used for filtering.

Derivations and theory regarding the dynamics and sensor models introduced here are inspired by [1].

2.1 INS Subsystem

For this formulation the inertial frame is the earlier mentioned tangent frame and it is assumed that the GNSS position information is expressed in the tangent frame as well.

This is for a *strapdown* INS, this means that the sensors are rigidly mounted to the frame of the vessel. In addition it is assumed that the IMU measurements are with respect to the body frame, meaning that the IMU is mounted such that the accelerometer and gyroscope measurement axes mirror the body frame, as defined in fig. 1.3.

This means that the measurements are of the form

$$\begin{aligned} \mathbf{f}^b &= \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} \\ \boldsymbol{\omega}_{tb}^b &= \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \\ \mathbf{p}_{gnss}^t &= \begin{bmatrix} p_n \\ p_e \\ p_d \end{bmatrix} \end{aligned} \tag{2.1}$$

2.1.1 Tangent Frame Differential Equations

Assuming that the tangent frame is an inertial frame with respect to the motion of the body frame then the second derivative of $\mathbf{p}^t$ can be written as

$$\frac{d^2\mathbf{p}^t}{dt^2} = \mathbf{f}^t + \mathbf{g}^t \tag{2.2}$$

where $\mathbf{g}^t$ is the gravity vector in the tangent frame. Given that the specific force is measured with respect to the body frame it must be rotated to the inertial frame by

$$\mathbf{f}^t = \mathbf{R}_b^t \mathbf{f}^b \tag{2.3}$$

By integrating this equation once and taking the derivative of the rotation matrix, as in section 1.4.2, the kinematic equations are found to be

$$\begin{aligned} \dot{\mathbf{p}}^t &= \mathbf{v}^t \\ \dot{\mathbf{v}}^t &= \mathbf{R}_b^t \mathbf{f}^b + \mathbf{g}^t \\ \dot{\mathbf{R}}_b^t &= \mathbf{R}_b^t \boldsymbol{\Omega}_{tb}^b \end{aligned} \tag{2.4}$$

where the inputs to this system are $\mathbf{f}^b$ and $\boldsymbol{\omega}_{tb}^b$, quantities measured by the accelerometer and gyroscope respectively.

2.1.2 Mechanization Equations

Given the kinematic equations found in eq. (2.4) one can proceed in developing the mechanization equations for a strapdown INS system. It is assumed that the accelerometer and gyro measurements are modeled as

$$\begin{aligned} \tilde{\mathbf{f}}^b &= \mathbf{f}^b + \Delta\mathbf{f}^b \\ \tilde{\boldsymbol{\omega}}_{tb}^b &= \boldsymbol{\omega}_{tb}^b + \Delta\boldsymbol{\omega}_{tb}^b \end{aligned} \tag{2.5}$$

where $\Delta\mathbf{f}^b$ and $\Delta\boldsymbol{\omega}_{tb}^b$ represent the accelerometer and gyro measurement errors. The measurement errors include both white noise terms and instrument calibration factors, the latter of which can be estimated, estimates written as $\Delta\hat{\mathbf{f}}^b$ and $\Delta\hat{\boldsymbol{\omega}}_{tb}^b$.

Given both the measurements and the calibration estimates the true values for the specific force and angular rate can be estimated by

$$\begin{aligned} \hat{\mathbf{f}}^b &= \tilde{\mathbf{f}}^b - \Delta\hat{\mathbf{f}}^b \\ \hat{\boldsymbol{\omega}}_{tb}^b &= \tilde{\boldsymbol{\omega}}_{tb}^b - \Delta\hat{\boldsymbol{\omega}}_{tb}^b \end{aligned} \tag{2.6}$$

Definitions for $\Delta\hat{\mathbf{f}}^b$ and $\Delta\hat{\boldsymbol{\omega}}_{tb}^b$ are discussed in section 2.2. The position and velocity estimate dynamics for the tangent frame $\mathbf{p}^t = \begin{bmatrix} p_n & p_e & p_d \end{bmatrix}^T$ and $\mathbf{v}^t = \begin{bmatrix} v_n & v_e & v_d \end{bmatrix}^T$ are computed as

$$\begin{aligned} \dot{\hat{\mathbf{p}}}^t &= \hat{\mathbf{v}}^t \\ \dot{\hat{\mathbf{v}}}^t &= \hat{\mathbf{R}}_b^t \hat{\mathbf{f}}^b + \mathbf{g}^t \end{aligned} \tag{2.7}$$

where

$$\hat{\mathbf{R}}_b^t \hat{\mathbf{f}}^b = \hat{\mathbf{f}}^t = \begin{bmatrix} \hat{f}_n \\ \hat{f}_e \\ \hat{f}_d \end{bmatrix} \tag{2.8}$$

with $\hat{\mathbf{f}}^b$ defined in eq. (2.6). In addition the direction cosine matrix is computed by integrating

$$\dot{\hat{\mathbf{R}}}_b^t = \hat{\mathbf{R}}_b^t \hat{\boldsymbol{\Omega}}_{tb}^b \tag{2.9}$$

where $\hat{\boldsymbol{\Omega}} = [\hat{\boldsymbol{\omega}}\times]$ with the definition of $\hat{\boldsymbol{\omega}}_{tb}^{b}$ from eq. (2.6). In conclusion the full mechanization equations can be written in terms of all given and calculated terms as

$$\begin{aligned}\dot{\hat{\mathbf{p}}}^{t} &= \hat{\mathbf{v}}^{t} \\ \dot{\hat{\mathbf{v}}}^{t} &= \hat{\mathbf{R}}_{b}^{t}\left(\tilde{\mathbf{f}}^{b} - \Delta\hat{\mathbf{f}}^{b}\right) + \mathbf{g}^{t} \\ \dot{\hat{\mathbf{R}}}_{b}^{t} &= \hat{\mathbf{R}}_{b}^{t}\left[\left(\tilde{\boldsymbol{\omega}}_{tb}^{b} - \Delta\hat{\boldsymbol{\omega}}_{tb}^{b}\right)\times\right]\end{aligned} \tag{2.10}$$

2.2 IMU Error Models

The following sections will address the error models used for the accelerometer and gyroscope systems such that a definition for $\Delta\mathbf{f}^{b}$ and $\Delta\boldsymbol{\omega}_{tb}^{b}$ can be found.

Note that for the IMU it is assumed that the platform frame, defined by how the IMU is mounted, and the body frame coincide, otherwise additional transformation errors would arise while compensating for the mounting.

2.2.1 Accelerometer

The accelerometer is a sensor in the IMU package which measures the specific force with respect to the body frame, denoted as $\mathbf{f}^{b}$.

The term $\tilde{\mathbf{f}}^{b}$ in this case represents the measurement, expressed with respect to body frame, including all modeled inaccuracies; let the model for this measurement be

$$\tilde{\mathbf{f}}^{b} = \mathbf{f}^{b} + \mathbf{b}_{a} + \boldsymbol{\nu}_{acc} \tag{2.11}$$

where $\mathbf{b}_{a}$ represents the uncompensated bias and $\boldsymbol{\nu}_{acc}$ represents random measurement noise. Comparing with eq. (2.5) this means that the measurement error can be written as

$$\Delta\mathbf{f}^{b} = \mathbf{b}_{a} + \boldsymbol{\nu}_{acc} \tag{2.12}$$

which can be estimated as

$$\Delta\hat{\mathbf{f}}^{b} = \hat{\mathbf{b}}_{a} \tag{2.13}$$

Combining eqs. (2.6) and (2.13) results in

$$\hat{\mathbf{f}}^{b} = \tilde{\mathbf{f}}^{b} - \hat{\mathbf{b}}_{a} \tag{2.14}$$

In order to estimate the biases the state space variable will need to be augmented with

$$\mathbf{x}_{b_a} = \mathbf{b}_{a} = \begin{bmatrix} b_{a,x} & b_{a,y} & b_{a,z} \end{bmatrix}^{T} \tag{2.15}$$

These biases are assumed constant therefore their dynamic model is

$$\dot{\mathbf{x}}_{b_a} = \mathbf{0} + \boldsymbol{\xi}_{a} \tag{2.16}$$

where $\boldsymbol{\xi}_{a}$ is white, Gaussian noise driving the bias estimate.

2.2.2 Gyroscope

The gyroscope is a sensor in the IMU package which measures the angular velocity between body and tangent frame with respect to the body frame, denoted as $\boldsymbol{\omega}_{tb}^{b}$.

The term $\tilde{\boldsymbol{\omega}}_{tb}^{b}$ in this case represents the measurement, expressed with respect to body frame, including all modeled inaccuracies; let the model for this measurement be

$$\tilde{\boldsymbol{\omega}}_{tb}^{b} = \boldsymbol{\omega}_{tb}^{b} + \mathbf{b}_g + \boldsymbol{\nu}_{gyro} \tag{2.17}$$

where $\mathbf{b}_g$ represents the uncompensated bias and $\boldsymbol{\nu}_g$ represents random measurement noise. Comparing with eq. (2.5) this means that the measurement error can be written as

$$\Delta\boldsymbol{\omega}_{tb}^{b} = \mathbf{b}_g + \boldsymbol{\nu}_{gyro} \tag{2.18}$$

which can be estimated as

$$\Delta\hat{\boldsymbol{\omega}}_{tb}^{b} = \hat{\mathbf{b}}_g \tag{2.19}$$

Combining eqs. (2.6) and (2.19) results in

$$\hat{\boldsymbol{\omega}}_{tb}^{b} = \tilde{\boldsymbol{\omega}}_{tb}^{b} - \hat{\mathbf{b}}_g \tag{2.20}$$

In order to estimate the biases the state space variable will need to be augmented with

$$\mathbf{x}_{b_g} = \mathbf{b}_g = \begin{bmatrix} b_{g,x} & b_{g,y} & b_{g,z} \end{bmatrix}^T \tag{2.21}$$

These biases are assumed constant therefore their dynamic model is

$$\dot{\mathbf{x}}_{b_g} = \mathbf{0} + \boldsymbol{\xi}_g \tag{2.22}$$

where $\boldsymbol{\xi}_g$ is white, Gaussian noise driving the bias estimate.

2.3 GNSS Error Model

A more advanced analysis of the INS problem would make use of the pseudo-range calculation [1] of the GNSS data.

The implementation discussed in this paper, however, will be considering a much simpler version. As such the measurement model is

$$\tilde{\mathbf{p}}_{gnss}^{t} = \begin{bmatrix} p_n \\ p_e \\ p_d \end{bmatrix} + \boldsymbol{\nu}_{gnss} \tag{2.23}$$

where $\boldsymbol{\nu}_{gnss}$ is assumed to be white, Gaussian noise. Given that the GNSS sensor is independent from the IMU it is assumed that the two noises are uncorrelated.

3. Filtering Options

As described in the introduction the traditional solution for the INS filtering problem has been the EKF, though to be able to use the EKF some reformulations of the problem will have to be made. In order to maintain the assumption of Gaussianity the full state model described in chapter 2 cannot be used. Instead one must recast the problem in the error state and construct a complementary filter, process described at length in [1].

This results in a substantial increase in the mathematical complexity of the problem formulation, and in creating the error state model approximations are made accurate to the first order. This is not necessarily problematic given that the EKF itself takes first order approximations of the nonlinear system but an alternative which avoids such approximations would be of interest.

Another alternative could be the Unscented Kalman Filter (UKF), but this has similar failings to the EKF. The UKF manages to maintain precision better than a first order approximation by leveraging several sigma points to represent the true state's posterior mean and covariance [3] instead of using a Jacobian. But the posterior density is only fully described by a mean and variance if normally distributed, therefore there is still some preference for Gaussianity. The β parameter can be tuned to leverage prior knowledge regarding the distribution of the state variable but performance is optimal for Gaussian distributions, with $\beta = 2$ [3]. In addition, given the degree of nonlinearity it can often be difficult to dimension the sigma points narrow enough to give good estimation performance but simultaneously wide enough to be able to capture the nonlinearities.

The final alternative to be considered here is the particle filter. With a particle filter approach the degree of nonlinearity and Gaussianity is not necessarily a problem. The filter works on the full nonlinear mapping and allows one to develop a solution not limited by the assumption of Gaussianity. This also allows the designer to avoid the more complex math behind an error state model and proceed with the full state model.

The solution, however, is not perfect; the computational requirements are much greater, to accurately represent the actual distribution one needs a large particle cloud. In addition particle filters tend to suffer from various problems like sample depletion, i.e. after some time all but a couple of particles have negligible weights [4], or lack of sample diversity. These problems have a host of solutions but these modifications also come at the cost of increased computations. In addition to this sometimes the performance can be improved by artificially increasing the process and/or measurement noise greater than what is observed [4], so tuning can sometimes be complicated and unintuitive.

4. Particle Filter

The particle filter is a type of Bayesian filter, similar to the Kalman Filter (KF), which works on the full nonlinear system. Different to the KF variants, the particle filter does not make any assumption of Gaussian processes. This is convenient because it allows the designer to consider the model in its entirety.

In this paper we introduce the Sequential Importance Sampling (SIS) particle filter as a solution for the strapdown INS problem with GNSS as the aiding sensor.

4.1 Sequential Importance Sampling

Assume the following definition for a standard state space model

$$\begin{aligned} \mathbf{x}_{k+1} &= f(\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{\omega}_k) \\ \mathbf{z}_k &= h(\mathbf{x}_k, \boldsymbol{\upsilon}_k) \end{aligned} \tag{4.1}$$

where $\boldsymbol{\omega}_k$ and $\boldsymbol{\upsilon}_k$, i.e. process and measurement noise, are created by sampling from their respective probability density functions.

The working principle of the PF is to describe the a posteriori probability density, $p(\mathbf{x}_k|\mathbf{z}_k)$, with a set of particles and their respective weights, from which a single state estimate can also be calculated. As the number of particles goes towards infinity this Monte Carlo (MC) representation becomes equivalent to the true posterior probability density function.

The a posteriori can often be difficult to sample from which is where the concept of importance sampling, see appendix A.6, comes into play. Let the particles $\mathbf{x}_k^i$ be drawn from an importance density (also called proposal distribution), $q(\mathbf{x}_k|\mathbf{z}_k)$, and the weights be defined as

$$w_k^i \propto \frac{p(\mathbf{x}_k^i|\mathbf{z}_k)}{q(\mathbf{x}_k^i|\mathbf{z}_k)} \tag{4.2}$$

Let the importance density be chosen such that it factorizes as

$$q(\mathbf{x}_k|\mathbf{z}_k) \triangleq q(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{z}_k) q(\mathbf{x}_{k-1}|\mathbf{z}_{k-1}) \tag{4.3}$$

The weight update equations can thus be derived by applying Bayes' Theorem, see appendix A.5, to $p(\mathbf{x}_k|\mathbf{z}_k)$

$$p(\mathbf{x}_k|\mathbf{z}_k) = \frac{p(\mathbf{z}_k|\mathbf{x}_k, \mathbf{z}_{k-1}) p(\mathbf{x}_k|\mathbf{z}_{k-1})}{p(\mathbf{z}_k|\mathbf{z}_{k-1})} \tag{4.4}$$

applying a similar factorization this can be equivalently written such that

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

removing the denominator this can be rewritten

$$p(\mathbf{x}_k|\mathbf{z}_k) \propto p(\mathbf{z}_k|\mathbf{x}_k, \mathbf{z}_{k-1})p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{z}_k)p(\mathbf{x}_{k-1}|\mathbf{z}_{k-1}) \tag{4.6}$$

Combining eqs. (4.3), (4.6) and (4.7) leads to the weight update equation

$$w_k^i \propto w_{k-1}^i \frac{p(\mathbf{z}_k|\mathbf{x}_k^i)p(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i)}{q(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i, \mathbf{z}_k)} \tag{4.7}$$

which can be used to estimate the posterior

$$p(\mathbf{x}_k|\mathbf{z}_k) \approx \sum_{i=1}^{N} w_k^i \delta\left(\mathbf{x}_k - \mathbf{x}_k^i\right) \tag{4.8}$$

and as $N \to \infty$ this approximation will approach the true value [5]. This holds true when only a filtered estimate of $p(\mathbf{x}_k|\mathbf{z}_k)$ is necessary at each time step

Smoothing implementations will look into the path $\mathbf{x}_{0:k}$ and the observation history $\mathbf{z}_{0:k}$ but filtering it only concerned with the current state so this is not necessary. This is the basic theory for the SIS PF, for full derivation see [5, 4], and a pseudo code description can be found in algorithm 1 [4, 6].

Algorithm 1 SIS PF

Choose a proposal distribution $q(x_{k+1}|x_k, z_{k+1})$ and number of particles N
Initialize: Generate particles $x_1^i \sim x_0$ for $i \in \{1, 2, \ldots, N\}$ and weights $w_{1|0}^i = 1/N$

1: **for** $k \leftarrow 1, N$ **do**

2: *Measurement Update:* Evaluate weights

$$\hat{w}_k^i = w_{k-1}^i \frac{p(z_k|x_k^i)p(x_k^i|x_{k-1}^i)}{q(x_k^i|x_{k-1}^i, z_k))} \tag{4.9}$$

and normalize by the sum

$$w_k^i = \frac{\hat{w}_k^i}{\sum_{i=1}^{N} \hat{w}_k^i} \tag{4.10}$$

3: *Estimation:* Calculate approximate state estimate from particle cloud

$$\hat{x}_k \approx \sum_{i=1}^{N} w_{k|k}^i x_k^i \tag{4.11}$$

4: *Time Update:* Generate predictions from the proposal distribution

$$x_{k+1}^i \sim q(x_{k+1}|x_k^i, z_{k+1}) \tag{4.12}$$

5: **end for**

4.2 Selection of Importance Density

The choice of the importance density can have great impact on the filter performance and therefore deserves attention. Some typical choices of importance densities will be introduced in the following sections.

4.2.1 Optimal

Assuming that the measurement function and measurement noise from eq. (4.1) are linear and Gaussian then using the optimal importance density is simple. The optimal importance density, optimal in the sense of minimizing the variance of the importance weights, is given by [5]

$$q(\mathbf{x}_k|\mathbf{x}_{k-1}^i, \mathbf{z}_k) = p(\mathbf{x}_k|\mathbf{x}_{k-1}^i, \mathbf{z}_k) \tag{4.13}$$

Which can be equivalently written as

$$q(\mathbf{x}_k|\mathbf{x}_{k-1}^i, \mathbf{z}_k) = \frac{p(\mathbf{z}_k|\mathbf{x}_k, \mathbf{x}_{k-1}^i)p(\mathbf{x}_k|\mathbf{x}_{k-1}^i)}{p(\mathbf{z}_k|\mathbf{x}_k^i)} \tag{4.14}$$

through Bayes' Theorem. Combining this equality with the weight update equations, eq. (4.7), simplifies to

$$w_k^i \propto w_{k-1}^i p(\mathbf{z}_k|\mathbf{x}_{k-1}^i) \tag{4.15}$$

This means that the importance weights can be calculated before the particles are propagated to time k.

4.2.2 Suboptimal

The most popular choice of suboptimal importance densities is the transitional prior [5]

$$q(\mathbf{x}_k|\mathbf{x}_{k-1}^i, \mathbf{z}_k) = p(\mathbf{x}_k|\mathbf{x}_{k-1}^i) \tag{4.16}$$

Combining this importance density with the weight update equations, eq. (4.7), results in

$$w_k^i \propto w_{k-1}^i p(\mathbf{z}_k|\mathbf{x}_k^i) \tag{4.17}$$

The only difference between this and the optimal case is that the importance weight must be computed after the particles have been propagated to time k.

4.3 Problems

There are a few problems with this basic form of PF which each have proposed solutions.

4.3.1 Sample Degeneracy

The first problem is that the variance of the importance weights can only increase [5], this means that as time approaches infinity all but a single particle will have negligible weights [4]. This can be addressed by adding a resampling step to the algorithm in algorithm 1. A typical way to formalize this is to wait with resampling until the degeneracy of the particle cloud reaches blow a certain threshold which is measured by approximating the number of effective samples

$$N_{eff} = \frac{1}{\sum_{i=1}^{N}(w_k^i)^2} \tag{4.18}$$

There exist many different resampling strategies at this point but the most popular are multinomial, stratified, and systematic (described and compared in [7, 8]). Adding resampling leads to a slightly modified PF algorithm, see algorithm 2.

Algorithm 2 Sequential Importance Sampling with Resampling (SISR) PF

Choose a proposal distribution $q(x_{k+1}|x_k, z_{k+1})$, resampling strategy, resmapling threshold N_{th}, and number of particles N

Initialize: Generate particles $x_1^i \sim x_0$ for $i \in \{1, 2, \ldots, N\}$ and weights $w_{1|0}^i = 1/N$

1: **for** $k \leftarrow 1, N$ **do**

2: *Measurement Update:* Evaluate weights

$$\hat{w}_k^i = w_{k-1}^i \frac{p(z_k|x_k^i)p(x_k^i|x_{k-1}^i)}{q(x_k^i|x_{k-1}^i, z_k))} \tag{4.19}$$

and normalize by the sum

$$w_k^i = \frac{\hat{w}_k^i}{\sum_{i=1}^{N}\hat{w}_k^i} \tag{4.20}$$

3: *Estimation:* Calculate approximate state estimate from particle cloud

$$\hat{x}_k \approx \sum_{i=1}^{N} w_{k|k}^i x_k^i \tag{4.21}$$

4: *Resampling:* If $\hat{N}_{eff} < N_{th}$ then sample from set $\{x_{1:k}^i\}_{i=1}^N$ where probability to take each sample i is $w_{k|k}^i$. Afterwards normalize weights $w_{k|k}^i = 1/N$.

5: *Time Update:* Generate predictions from the proposal distribution

$$x_{k+1}^i \sim q(x_{k+1}|x_k^i, z_{k+1}) \tag{4.22}$$

6: **end for**

4.3.2 Loss of Sample Diversity

Implementing resampling adds another problem however, which is a loss of sample diversity. Since resampling selects relative to the weights particles with low weights will be less likely to be resampled. This means that as time approaches infinity

the particle values will begin to converge, this can be problematic if the point of convergence is skewed by corrupted measurement information.

As such proper dimensioning of the process noise is vital as the process noise is what allows the particles to drift over time, and when the process noise is too small dithering, increasing state noise, can be helpful in increasing the performance. In addition other parameters such as the number of particles and resampling threshold can impact convergence as well, obviously more particles and less resampling leads to slower convergence.

4.4 Dimensioning Noise

Selecting proper values for the process and measurement noise can have an enormous impact on the overall performance of the PF, however given high dimension state variables and complicated differential equations this can be quite nebulous. Gustafsson [4] suggests some different scenarios for process versus measurement noise magnitudes and how to address them in addition to introducing the concept of dithering, i.e. purposefully increasing the state noise in order to increase the area spanned by the particles.

5. Implementation

This chapter will address the more practical side of how the filtering problem will be addressed.

5.1 Sensor Parameters

To ensure some degree of realism in the simulation the parametrization of the necessary sensor constants have been taken from IMU and GNSS sensor data sheets [9, 10].

The sampling times are

$$\begin{aligned} T_{s,imu} &= 0.01\text{s} \\ T_{s,gnss} &= 1\text{s} \end{aligned} \tag{5.1}$$

and the noise variances are

$$\begin{aligned} R_{accelerometer} &= 0.1185 \cdot 10^{-3} \cdot \mathbf{I}\ [(\text{m/s}^2)^2] \\ R_{gyroscope} &= 0.6206 \cdot 10^{-4} \cdot \mathbf{I}\ [(\text{rad/s})^2] \\ R_{gnss} &= 25 \cdot \mathbf{I}\ [(\text{m})^2] \end{aligned} \tag{5.2}$$

where $\mathbf{I} \in \mathbb{R}^{3\times3}$. The effects of these noise parametrizations can be seen in fig. 5.7. In addition the true bias values, when relevant, are assumed to be

$$\begin{aligned} \mathbf{b}_a &= \begin{bmatrix} 0.05 \\ 0.05 \\ 0.05 \end{bmatrix}^T [\text{m}\,\text{s}^{-2}] \\ \mathbf{b}_g &= \begin{bmatrix} 0.00035 \\ 0.00035 \\ 0.00035 \end{bmatrix}^T [\text{rad}\,\text{s}^{-1}] \end{aligned} \tag{5.3}$$

For the simulation it has been assumed that the PF has already undergone initialization, as such this implementation assumes the same initialization expectations as the EKF implementation for the IMU package [9]. More can be read about initialization methods in [1].

5.2 Artificial Data

The particle filter will first be tested against an artificial data set. Given that this paper takes context in a maritime application it is assumed, for the artificial data,

that there is not significant amount of pitch or roll. Therefore the data set can be constructed by defining a thrust, i.e. forward acceleration, and a yaw rate and integrate these two to create position data.

This has been done with two different movement profiles, a simple and a complicated, both with the thrust profile in fig. 5.1. The difference between the data sets is that the simple is made with zero yaw rate throughout, whereas the complicated has the yaw profile shown in fig. 5.2.

Similarly to the sensor constants values for the acceleration and yaw rate have been chosen such that the velocities and turning rates are somewhat realistic according to [11]. The full set of data can then be found by integration and

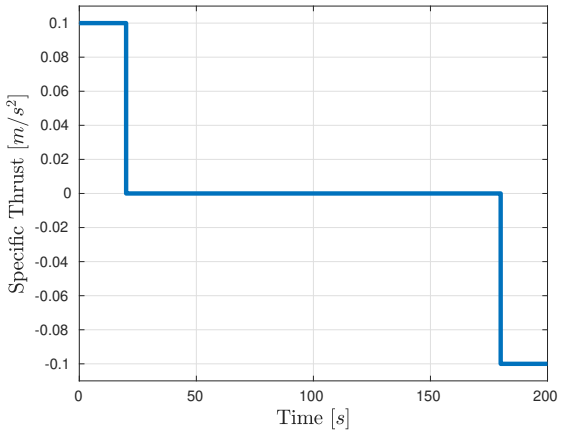


Figure 5.1: Thrust profile given for the artificial data sets.

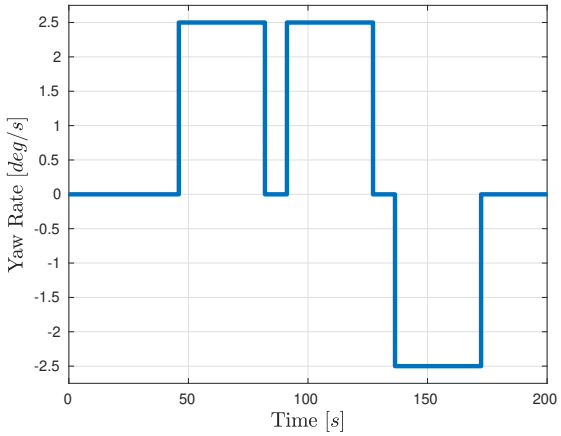


Figure 5.2: Yaw rate profile given for the complex data sets.

seen for the simple case in fig. 5.3 for variables pertaining to linear motion and in fig. 5.4 for angular. The same is true for the complex case in figs. 5.5 and 5.6.

This data can be artificially corrupted with noise and resampled to provide more realistic IMU and GNSS data, e.g. fig. 5.7. Take note, vertical motion and roll/pitch motion has not been shown in these visualizations. The data exists but for sake of clarity it is not shown as for the simple artificial case this motion has been assumed to be zero. It will still be corrupted and used when filtering.

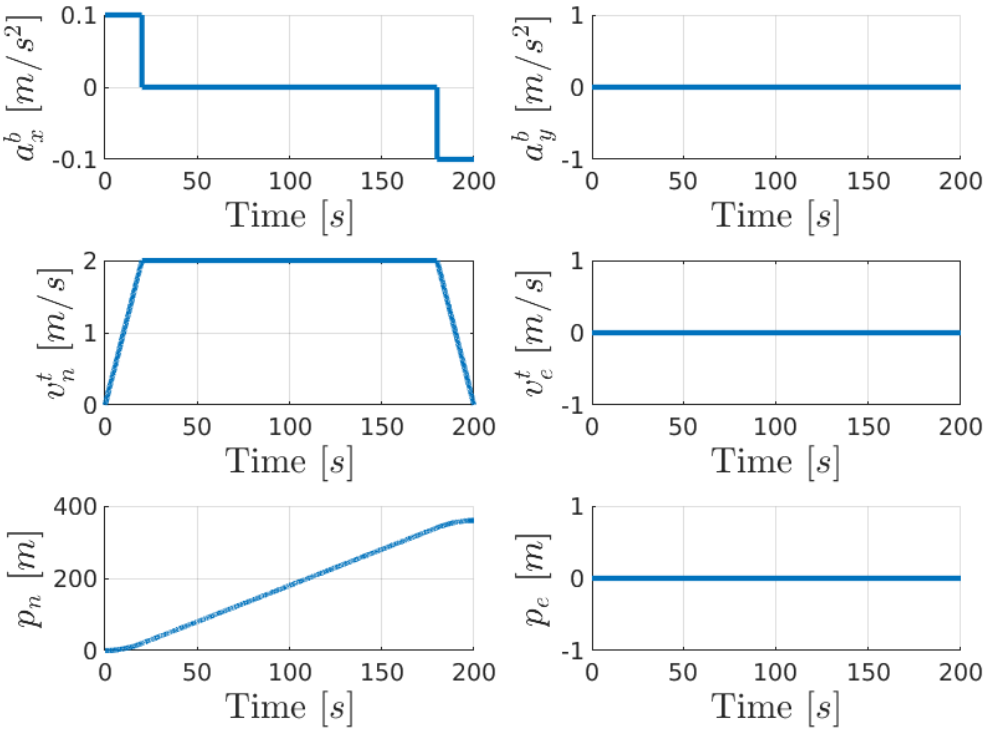


Figure 5.3: Linear data found by integration for the simple, no noise case.

5.3 State Transition Function

The state transition function to be used is a variant of the mechanization equations from section 2.1.2 augmented with relevant dynamic equations for the chosen noise model.

For example assuming the noise model given in sections 2.2 and 2.3 the augmented dynamic equations to be used would be

$$\begin{aligned}
\dot{\hat{\mathbf{p}}}^t &= \hat{\mathbf{v}}^t \\
\dot{\hat{\mathbf{v}}}^t &= \hat{\mathbf{R}}_b^t \left(\tilde{\mathbf{f}}^b - \hat{\mathbf{b}}_a\right) + \mathbf{g}^t \\
\dot{\hat{\mathbf{R}}}_b^t &= \hat{\mathbf{R}}_b^t \left[\left(\tilde{\boldsymbol{\omega}}_{tb}^b - \hat{\mathbf{b}}_g\right)\times\right] \\
\dot{\hat{\mathbf{b}}}_a &= \mathbf{0} \\
\dot{\hat{\mathbf{b}}}_g &= \mathbf{0}
\end{aligned} \tag{5.4}$$

The first three equations of eq. (5.4) are the core differential equations for describing the kinematics whereas the second two are there to mitigate problems with real life IMU sensors.

5.3.1 Discretizing

These equations are given in continuous time whereas the particle filter works on discrete time models. This can be easily overcome by applying a discretization method like backward Euler method, see appendix A.4.

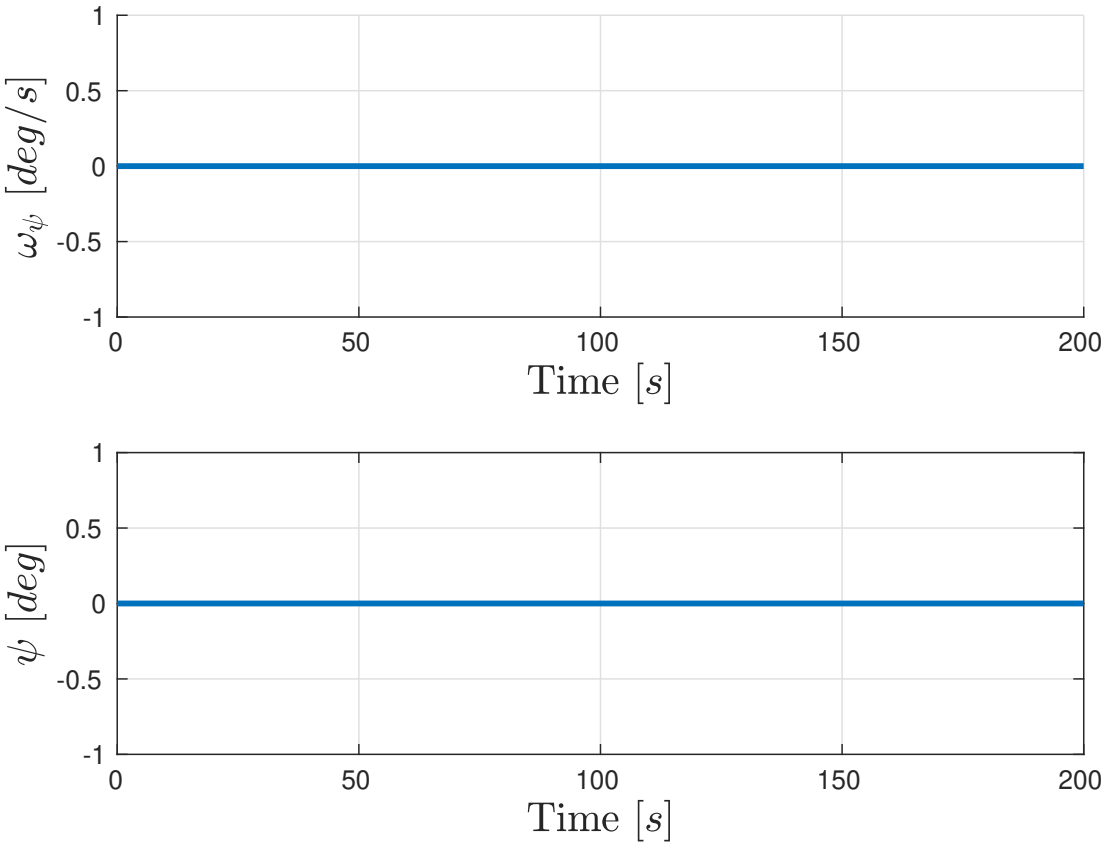


Figure 5.4: Angular data found by integration for the simple, no noise case.

The Euler method can sometimes provide a sub-standard approximation, but given the high sampling rate of the IMU this shouldn't be a problem. A basic error analysis has been conducted for the velocity and position in tangent frame given acceleration and yaw rate; see figs. 5.8a and 5.8b for the analysis applied to simple and complex data respectively. From the plots it is clear to see that the squared error is quite small so more complicated discretization methods will not be pursued.

5.3.2 State Variable

The most basic state variable will need to contain $\hat{\mathbf{p}}^t$, $\hat{\mathbf{v}}^t$, and $\hat{\mathbf{R}}_b^t$, this is problematic because $\hat{\mathbf{p}}^t$ and $\hat{\mathbf{v}}^t$ are column vectors whereas $\hat{\mathbf{R}}_b^t$ is a matrix. Assuming $\hat{\mathbf{R}}_b^t$ is of form

$$\hat{\mathbf{R}}_b^t = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{5.5}$$

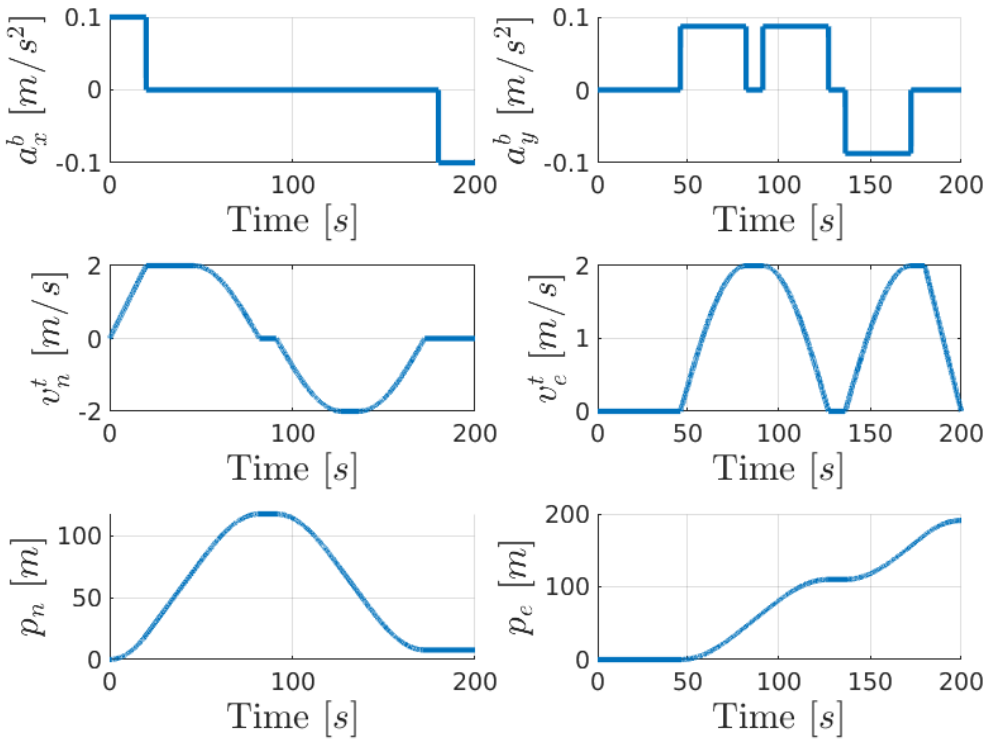


Figure 5.5: Linear data found by integration for the complex, no noise case.

the state vector can be formulated as a column by writing the columns of $\hat{\mathbf{R}}_b^t$ in one after another resulting in

$$\hat{\mathbf{x}} = \begin{bmatrix} \hat{\mathbf{p}}^t \\ \hat{\mathbf{v}}^t \\ r_{11} \\ r_{21} \\ r_{31} \\ r_{12} \\ \vdots \\ r_{33} \end{bmatrix} \tag{5.6}$$

this is the same operation performed in MATLAB by writing `R(:)` assuming that `R` is defined as a matrix. In later sections the state vector may be written as

$$\hat{\mathbf{x}} = \begin{bmatrix} \hat{\mathbf{p}}^t \\ \hat{\mathbf{v}}^t \\ \hat{\mathbf{R}}_b^t \end{bmatrix} \tag{5.7}$$

but this is only meant to serve as a shorthand for eq. (5.6).

Also note that bias states are still just appended after the rotation matrix, for

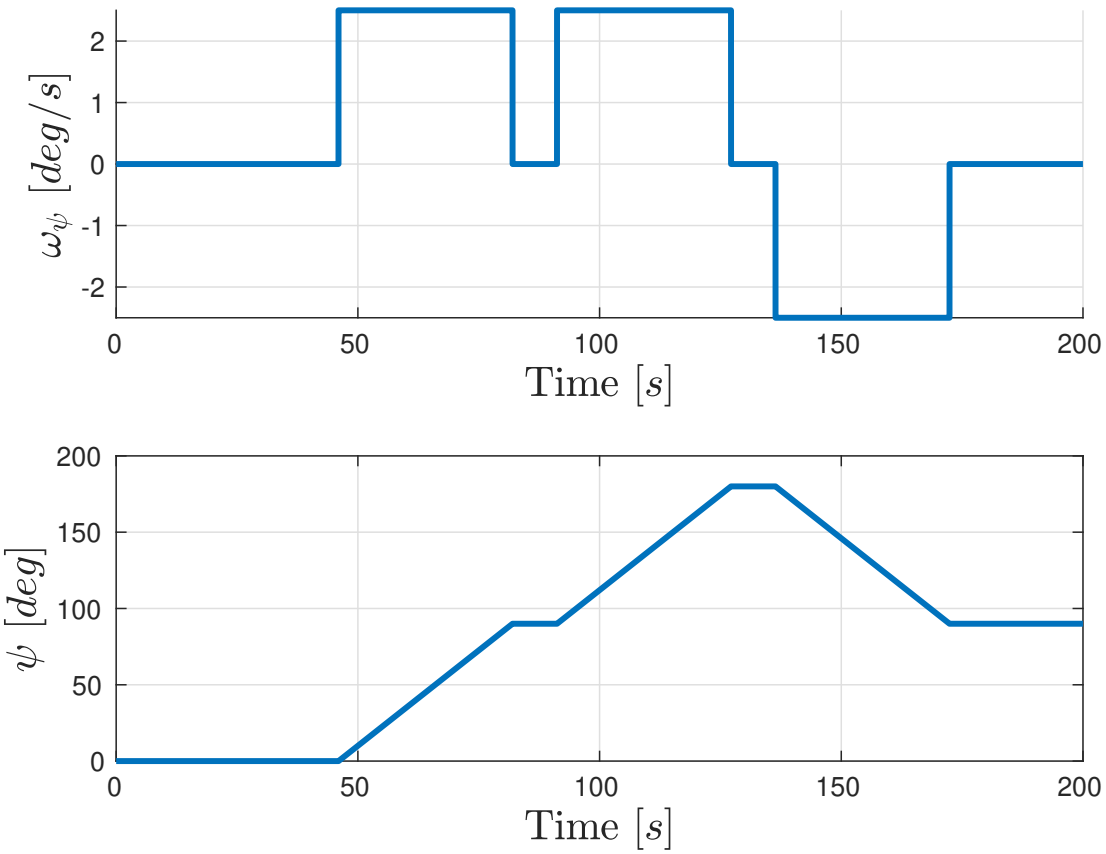


Figure 5.6: Angular data found by integration for the complex, no noise case.

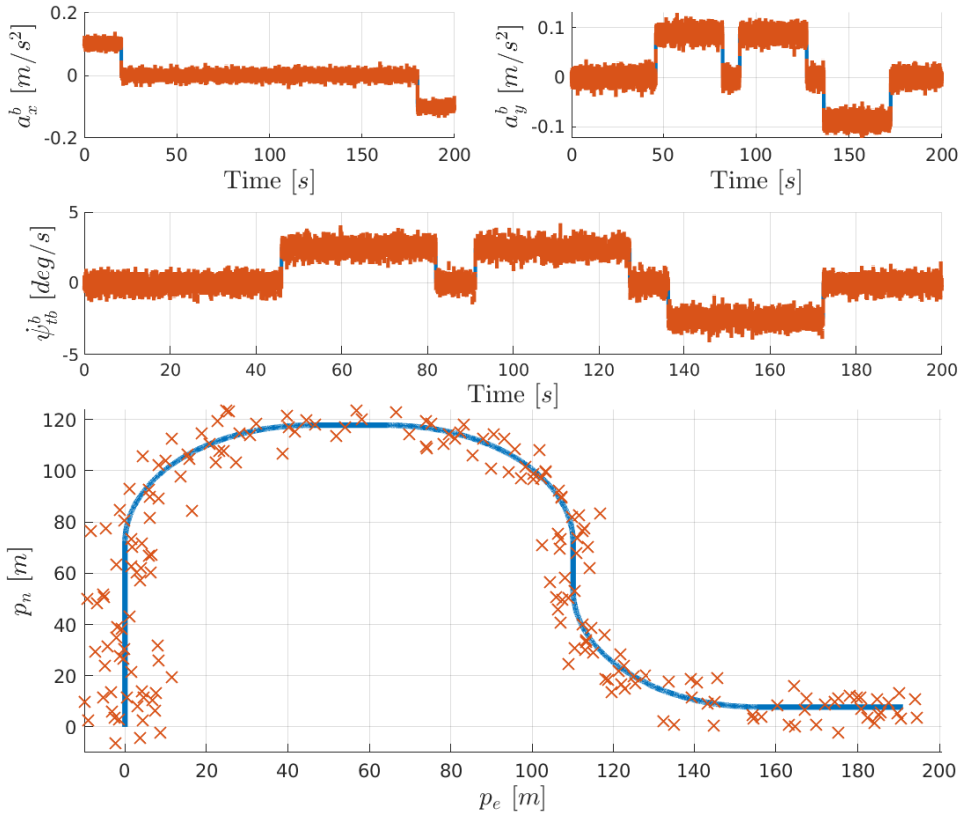
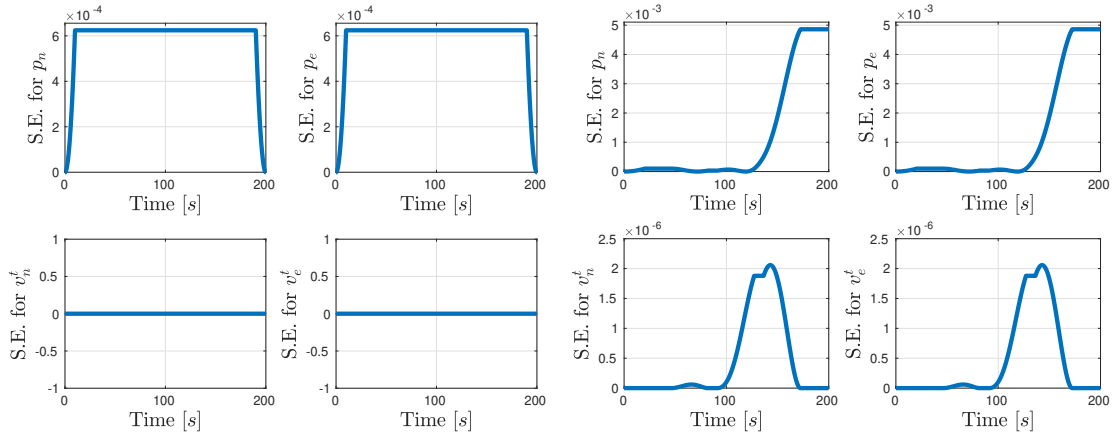


Figure 5.7: Measured quantities plotted with noise, blue lines correspond to true values and red lines corresponds to the true values corrupted with white, Gaussian noise. Top three plots are IMU measurements whereas bottom plot is GNSS measurement.



(a) Simple (b) Complex

Figure 5.8: Squared error for deviations in tangent frame velocity or position generated using the backward Euler method given body frame acceleration and angular velocity, i.e. IMU data.

example

$$\hat{\mathbf{x}} = \begin{bmatrix} \hat{\mathbf{p}}^t \\ \hat{\mathbf{v}}^t \\ r_{11} \\ r_{21} \\ r_{31} \\ r_{12} \\ \vdots \\ r_{33} \\ \hat{\mathbf{b}}_a \\ \hat{\mathbf{b}}_g \end{bmatrix} \tag{5.8}$$

5.4 Measurement Function

The measurement function is quite simple. It is assumed that the GNSS sensor returns position data given with respect to the tangent frame, so in this case the measurement function is actually a linear operation which corresponds to removing the first three elements of the state estimate, i.e. $\begin{bmatrix} \hat{p}_n & \hat{p}_e & \hat{p}_d \end{bmatrix}^T$.

5.5 Measurement Likelihood

The measurement likelihood is a calculation of $p(\tilde{\mathbf{z}}_k|\mathbf{x}_k^i)$, given that the noise model for the GNSS is purely Gaussian and that the measurement function is a linear operator one would imagine that the measurement estimate is normally distributed. This is the case as long as the state variable is also normally distributed which can be checked by numerical simulation. By examining the particle cloud after several iteration, fig. 5.9, it is clearly normally distributed, this can be further verified using the `dfittool` in MATLAB.

As such the measurement likelihood can be calculated as

$$p(\tilde{\mathbf{z}}_k|\hat{\mathbf{x}}_k^i) = \frac{1}{\sqrt{(2\pi)^n \det(\mathbf{R}_{gnss})}} e^{-0.5(\tilde{\mathbf{z}}_k - \hat{\mathbf{z}}_k^i)^T \mathbf{R}_{gnss}^{-1} (\tilde{\mathbf{z}}_k - \hat{\mathbf{z}}_k^i)} \tag{5.9}$$

or in other words as a multivariate normal distribution where n is the dimension of the measurements and $\hat{\mathbf{z}}_k^i$ is the output predictions from the particle cloud, defined as

$$\hat{\mathbf{z}}_k^i = \text{meas\_fcn}(\mathbf{x}_k^i) \tag{5.10}$$

5.6 Particle Filter Specifics

Additional parameters need specifying in order to implement the particle filter, in particular this would be the number of particles, resampling policy, and resampling strategy.

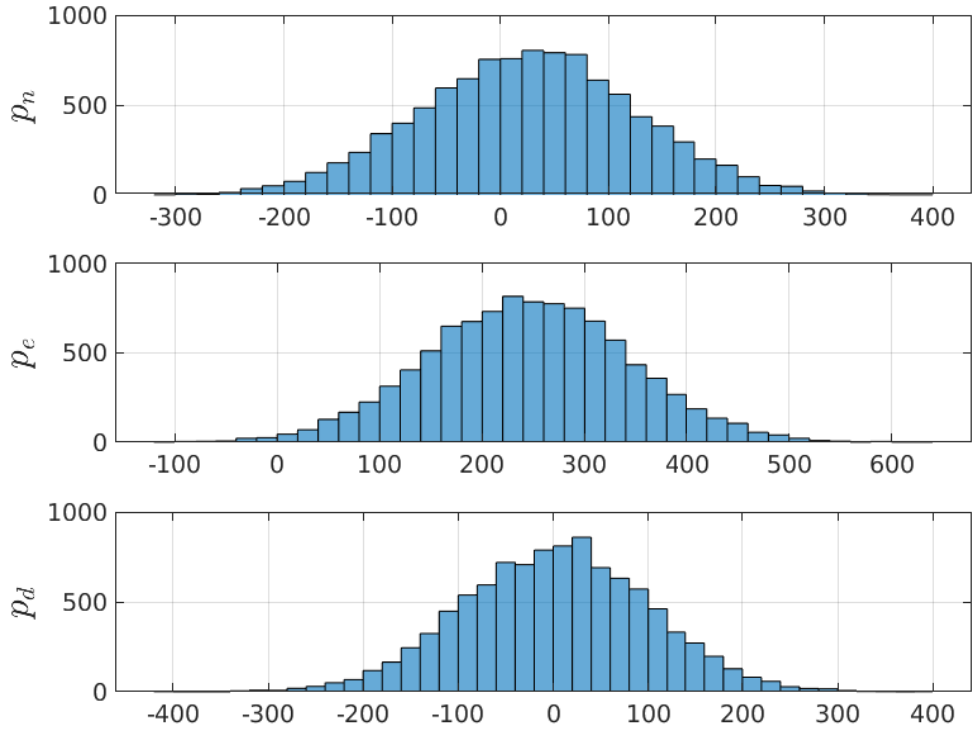


Figure 5.9: Histogram of particle cloud after many iterations, with inputs corresponding to complex data set, for each of the measured quantities, note that it remains normally distributed.

Number of Particles The number of particles is a main tuning knob for the particle filter, greater particles typically means better estimation accuracy as the number of particles linearly correlates with accuracy in portraying the state distribution but it also comes at the cost of higher computation power. Given the high sampling rate of the IMU it is critical that the number of particles does not become obstructive.

Importance Density The choice of importance density has been narrowed down to the two described in section 4.2. For the context of this implementation, where an IMU is considered, the optimal importance density will not necessarily be possible. This is because the IMU measurement noise is not necessarily Gaussian [1], in the artificial data it is convenient to model it as Gaussian but for future comparison with real data the assumption will not necessarily hold. As such the importance density is defined as

$$q(\mathbf{x}_k|\mathbf{x}_{k-1}^i, \mathbf{z}_k) = p(\mathbf{x}_k|\mathbf{x}_{k-1}) \tag{5.11}$$

or in other words the probability of the transitional prior.

Resampling Policy The resampling policy can also have great impact on the filter performance. As was discussed in chapter 4 there are several different resampling policies, but this implementation will be using importance sampling. Importance sampling is when sampling is conditional on the number of effective particles N_{eff} being less than the number of total particles, N_p, multiplied by some constant. This creates an additional tuning parameter: the effective particle ratio.

Gustafsson [4] recommends $\frac{2}{3}$ as a good rule-of-thumb minimum ratio of effective to total number of particles.

Resampling Strategy The resampling strategy can also impact the quality of the filtering performance as it has an impact on the variance and uniformity of the particle cloud post resample.

The simplest, and one of the most popular, resampling strategies is probably multinomial resampling [7] where one picks particles at random with a likelihood proportional to their weights. Other popular strategies are stratified and systematic which are essentially modifications on multinomial resampling. All three have been shown to have very similar performance in a practical scenario [8].

Multinomial is the resampling strategy used in this paper given that the differences are minor.

6. Filtering Performance

This chapter will discuss the filtering performance of the particle filter implementation discussed in chapter 5 for different data combinations, e.g. simple and complex.

6.1 Without Bias

First the performance of the filter will be examined using sensors corrupted only by white noise, i.e. without bias. Throughout the following simulations the initial estimates for the states is set as

$$\hat{\mathbf{x}}_0 = \begin{bmatrix} \hat{\mathbf{p}}^t \\ \hat{\mathbf{v}}^t \\ \hat{\mathbf{R}}_b^t \end{bmatrix} \tag{6.1}$$

where

$$\begin{aligned} \hat{\mathbf{p}}^t &= \mathbf{0} \\ \hat{\mathbf{v}}^t &= \mathbf{0} \\ \hat{\mathbf{R}}_b^t &= \mathbf{I} \end{aligned} \tag{6.2}$$

and the initial estimate covariance is given as

$$\mathbf{P}_0 = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & 10^{-3} \cdot \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & 10^{-3} \cdot \mathbf{I} \end{bmatrix} \tag{6.3}$$

this goes to say that after the initialization procedure one is relatively sure that the vessel is stationary and its heading, and relatively less sure about its exact position in the tangent frame.

6.1.1 Simple

This section will present a couple different tunings of the particle filter applied to the simple data set with a quantitative performance comparison as the end.

Tuning 1

First, in order to demonstrate the importance of process noise/dithering the filter has been run on the simple data set with no process noise and no signal roughening at all, see fig. 6.1. This is for a filter with the parameters shown in table 6.1.

	Value
Number of Particles	1000
Minimum Effective Particle Ratio	2/3
Resampling Strategy	multinomial
Measurement Likelihood	normal

Table 6.1: Particle Filter parameters

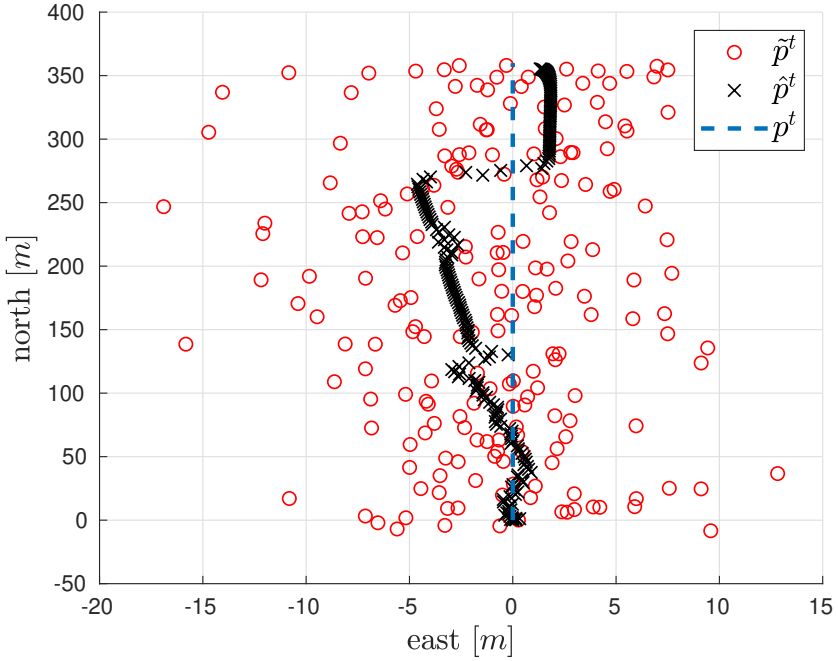


Figure 6.1: Filtering performance for an SIS particle filter with parameters in table 6.1 on the simple data set. This PF does not use process noise.

Notice how the estimate has diverged, this is in part due to convergence of the particle cloud to a small subset of state values which here are clearly incorrect. The convergence is evident when examining a plot of the particle cloud at the final iteration, see fig. 6.2.

Tuning 2

This effect can be mitigated by increasing the size of the particle cloud as this will delay the convergence at the cost of computation power. The performance after increasing the number of particles to 5000 can be seen in fig. 6.3. Clearly this performs better though convergence still happens as seen in fig. 6.4.

Tuning 3

By including process noise as well the filtering performance can be increased, see fig. 6.5, with a much broader particle cloud, see fig. 6.6, without having to increase the number of particles. The continuous time process noise used in this case is

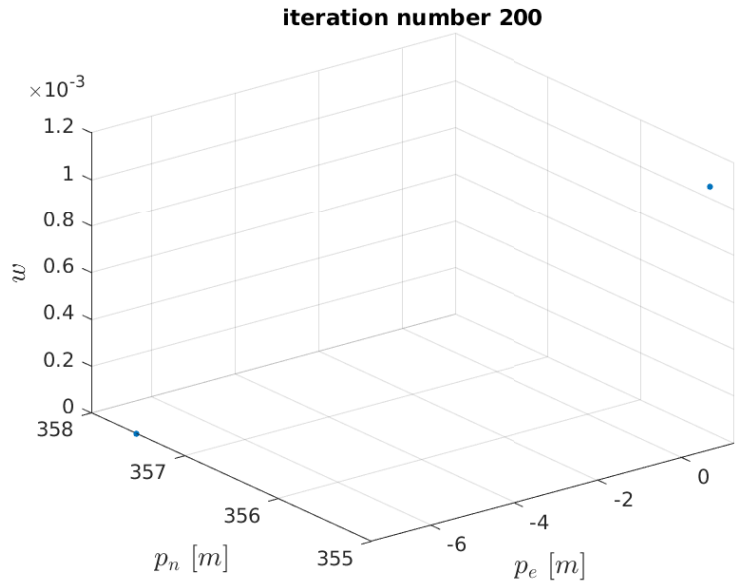


Figure 6.2: Particle cloud at the last iteration of filtering for filter shown in fig. 6.1.

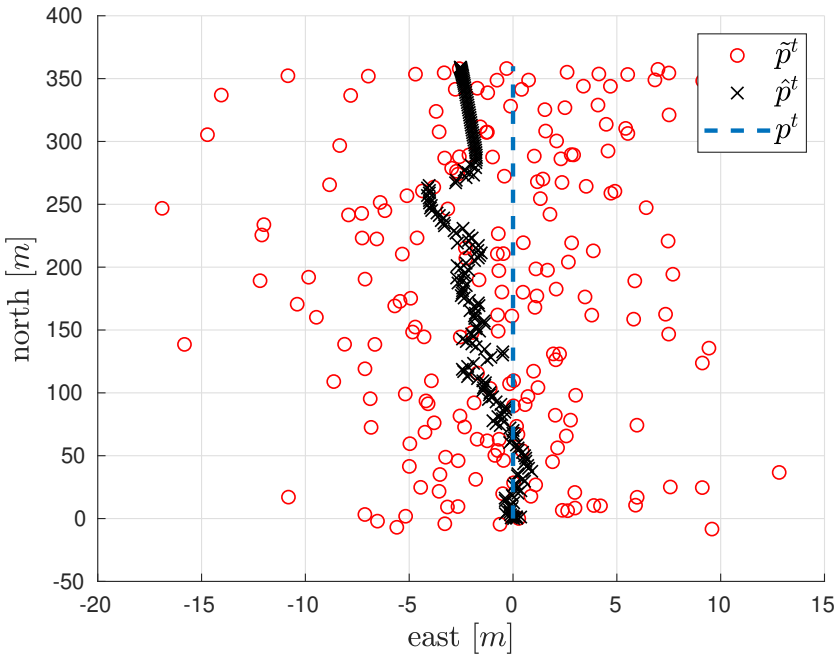


Figure 6.3: Filtering performance for an SIS particle filter with parameters in table 6.1, except 5000 particles, on the simple data set. This PF does not use process noise.

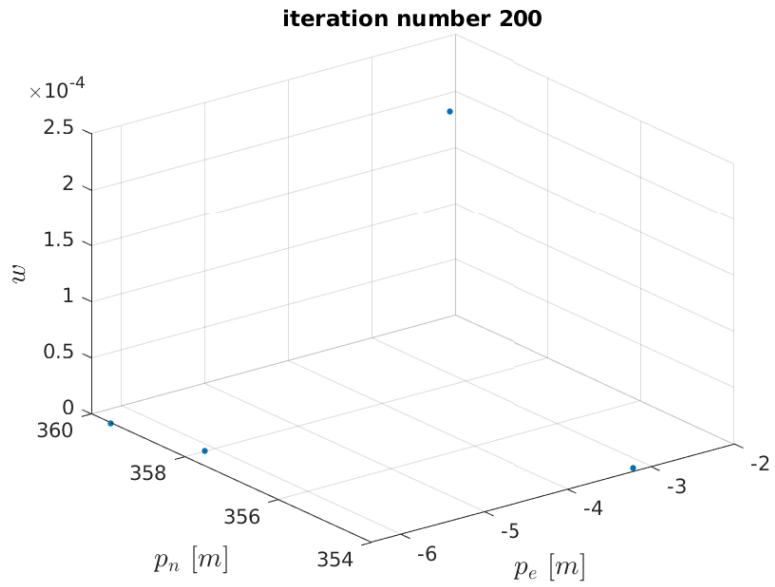


Figure 6.4: Particle cloud at the last iteration of filtering for filter shown in fig. 6.3.

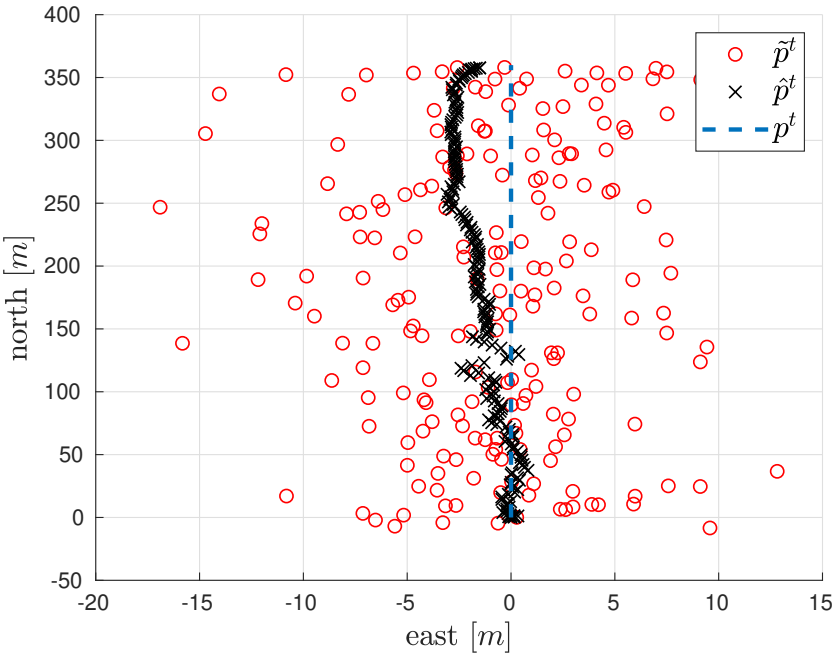


Figure 6.5: Filtering performance for an SIS particle filter with parameters in table 6.1 on the simple data set. This PF does use process noise.

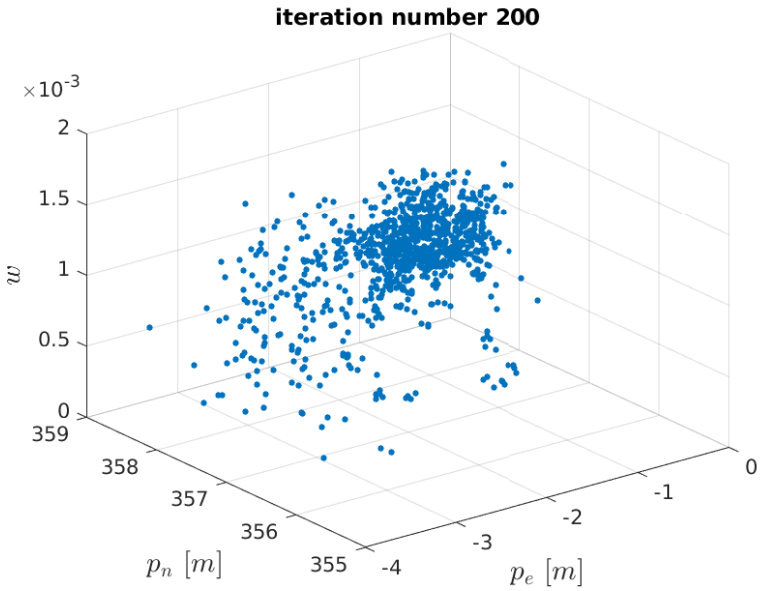


Figure 6.6: Particle cloud at the last iteration of filtering for filter shown in fig. 6.5.

$$\mathbf{Q} = \begin{bmatrix} 10^{-2} \cdot \mathbf{I}^{3\times 3} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_{accelerometer} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{R}_{gyroscope} \end{bmatrix} \tag{6.4}$$

Performance Comparison

The three tunings clearly have some differences in performance, evident from the plots, but here a slightly more quantitative comparison will be introduced.

In table 6.2 the Root Mean Squared Error (RMSE) of the noisy GNSS measurements as well as the filter estimates can be seen. Note that the RMSE is, by definition, averaged over the simulation duration and as such does not say much about the transient behavior. Regardless the numbers in the table show that all of the filters improve upon the noisy measurements and that, unsurprisingly, more particles and adding proces noisy increase the filtering performance.

	RMSE for Position	
Signal	North [m]	East [m]
Noisy GNSS Measurements	4.9345	5.2475
PF Tuning 1	2.1346	2.2284
PF Tuning 2	1.0002	1.9998
PF Tuning 3	1.2315	1.7866

Table 6.2: Comparison of the Root Mean Square Error for each of the filter tunings for the simple data set without bias.

This will end the use of the simple data set, now that it has been shown that the particle filter works to some degree further simulations will be carried out with the complex data set.

6.1.2 Complex

This section will present a couple different tunings of the particle filter applied to the simple data set with a quantitative performance comparison as the end.

Tuning 1

Just as before not taking advantage of the process noise and setting it to zero gives relatively poor performance, see fig. 6.7, with low particle diversity, see fig. 6.8. This is for a filter with the parameters shown in table 6.3.

	Value
Number of Particles	1000
Minimum Effective Particle Ratio	2/3
Resampling Strategy	multinomial
Measurement Likelihood	normal

Table 6.3: Particle Filter parameters

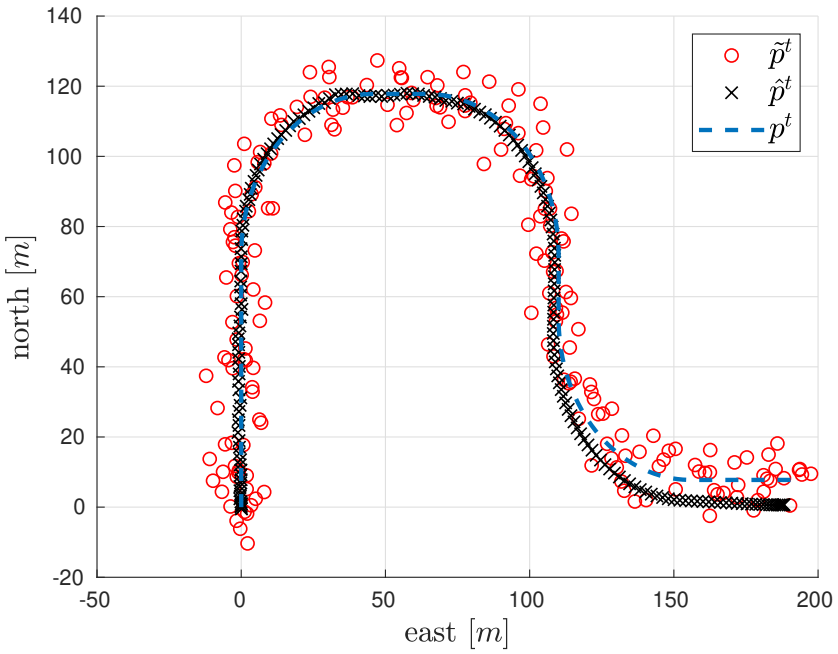


Figure 6.7: Filtering performance for an SIS particle filter with parameters in table 6.3 on the complex data set. This PF does not use process noise.

6.1.3 Tuning 2

This performance can be improved, see fig. 6.9, by applying the process noise, same as eq. (6.4), which unsurprisingly increases sample diversity as well, see fig. 6.10.

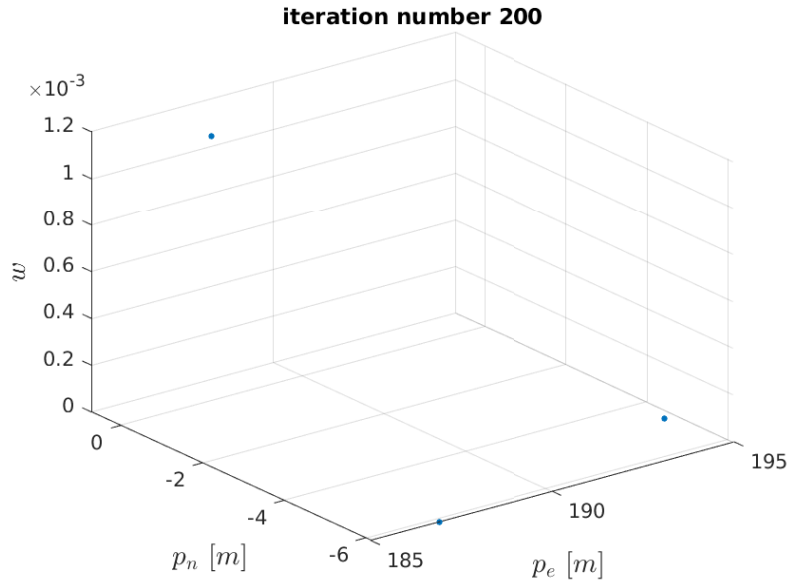


Figure 6.8: Particle cloud at the last iteration of filtering for filter shown in fig. 6.7.

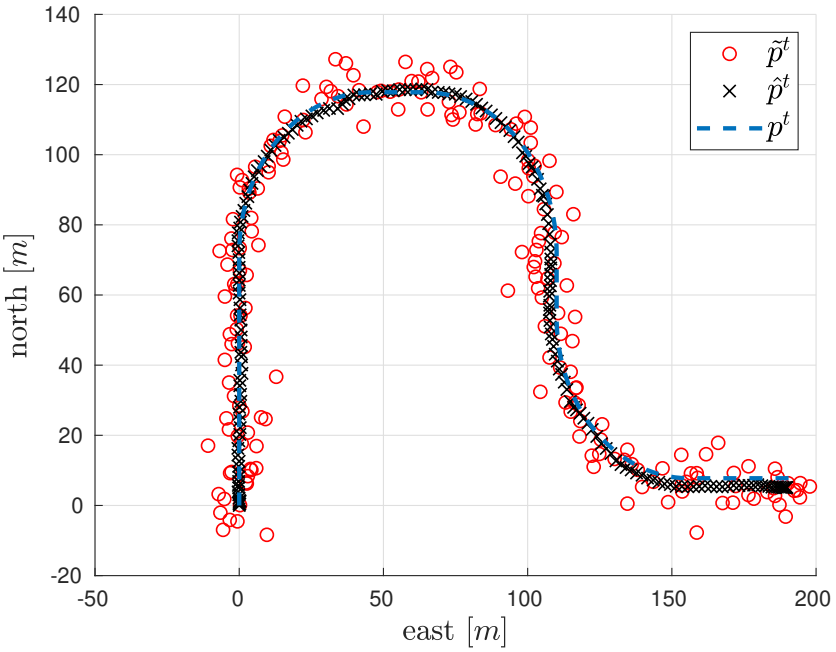


Figure 6.9: Filtering performance for an SIS particle filter with parameters in table 6.3 on the complex data set. This PF does use process noise.

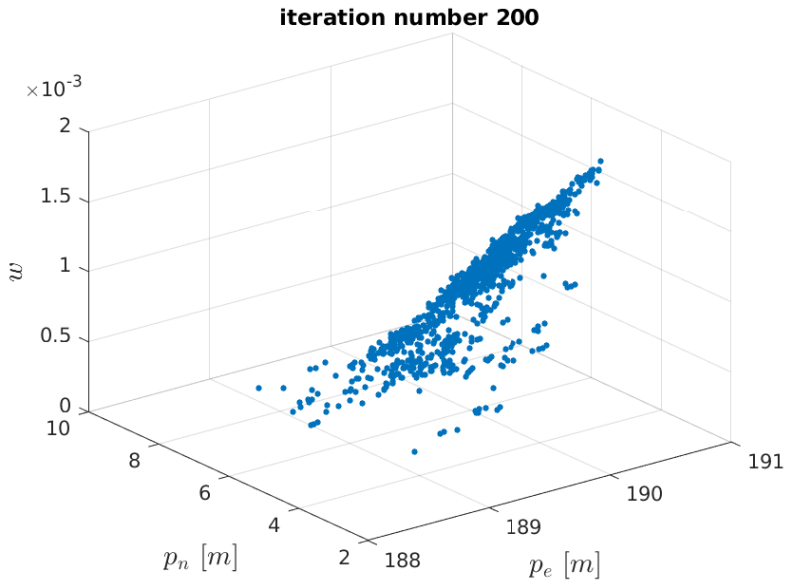


Figure 6.10: Particle cloud at the last iteration of filtering for filter shown in fig. 6.9.

6.1.4 Tuning 3

This performance can be further improved, slightly, by applying some dithering to the process noise, see fig. 6.11, such that the "process noise" is now

$$\mathbf{Q} = \begin{bmatrix} 10^{-2} \cdot \mathbf{I}^{3\times 3} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & 10 \cdot \mathbf{R}_{accelerometer} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & 10 \cdot \mathbf{R}_{gyroscope} \end{bmatrix} \tag{6.5}$$

6.1.5 Performance Comparison

The three tunings clearly have some differences in performance, evident from the plots, but here a slightly more quantitative comparison will be introduced.

In table 6.4 the RMSE of the noisy GNSS measurements as well as the filter estimates can be seen. Again, all of the filters improve upon the noisy measurements.

	RMSE for Position	
Signal	North [m]	East [m]
Noisy GNSS Measurements	4.9345	5.2475
PF Tuning 1	3.2757	1.6881
PF Tuning 2	1.5462	1.6730
PF Tuning 3	1.5349	1.2541

Table 6.4: Comparison of the Root Mean Square Error for each of the filter tunings for the complex data set without bias.

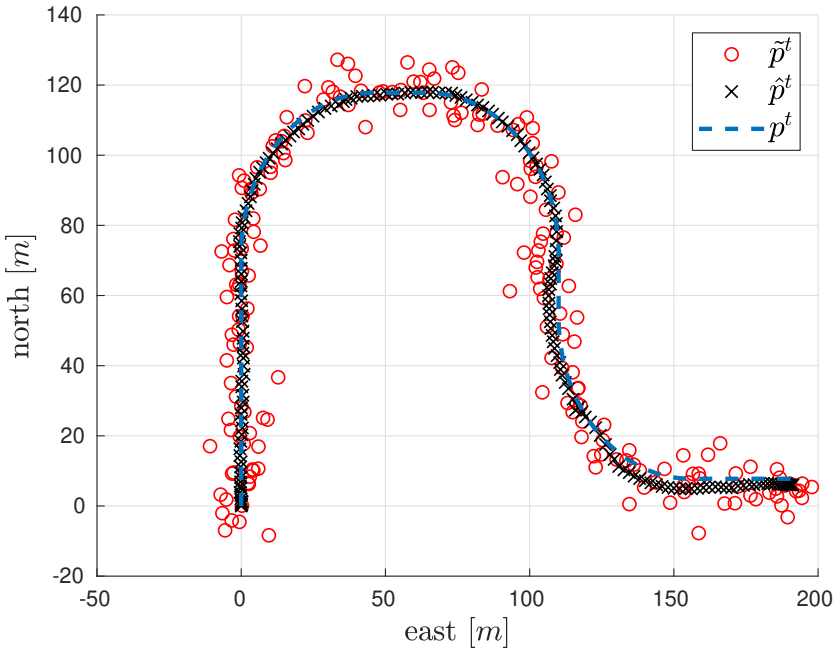


Figure 6.11: Filtering performance for an SIS particle filter with parameters in table 6.3 on the complex data set. This PF does use process noise and dithering.

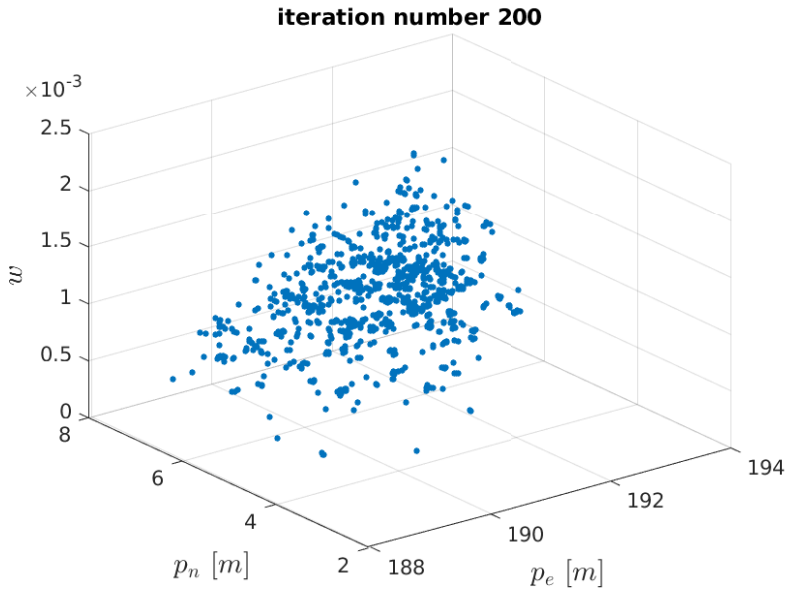


Figure 6.12: Particle cloud at the last iteration of filtering for filter shown in fig. 6.11.

6.2 With Bias

With bias everything about the simulation becomes a lot more complicated. Not only are there an additional six states to estimates and six process noises to dimension, but error due to the bias contributes greatly to the error of the overall filter.

Using the bias values from eq. (5.3), parameters shown in table 6.5, and the process noises

$$\mathbf{Q} = \begin{bmatrix} 10^{-2} \cdot \mathbf{I}^{3\times3} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_{accelerometer} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{R}_{gyroscope} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & 10^{-1} \cdot \mathbf{I}^{3\times3} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & 10^{-2} \cdot \mathbf{I}^{3\times3} \end{bmatrix} \tag{6.6}$$

the filtering performance is shown in fig. 6.13 with the bias estimates in fig. 6.14. The bias estimates are clearly not particularly promising and remain poor despite tuning.

Upon further reflection this is likely due to a shortcoming in the DCM method for attitude representation, this will be discussed in chapter 7.

	Value
Number of Particles	1000
Minimum Effective Particle Ratio	2/3
Resampling Strategy	multinomial
Measurement Likelihood	normal

Table 6.5: Particle Filter parameters.

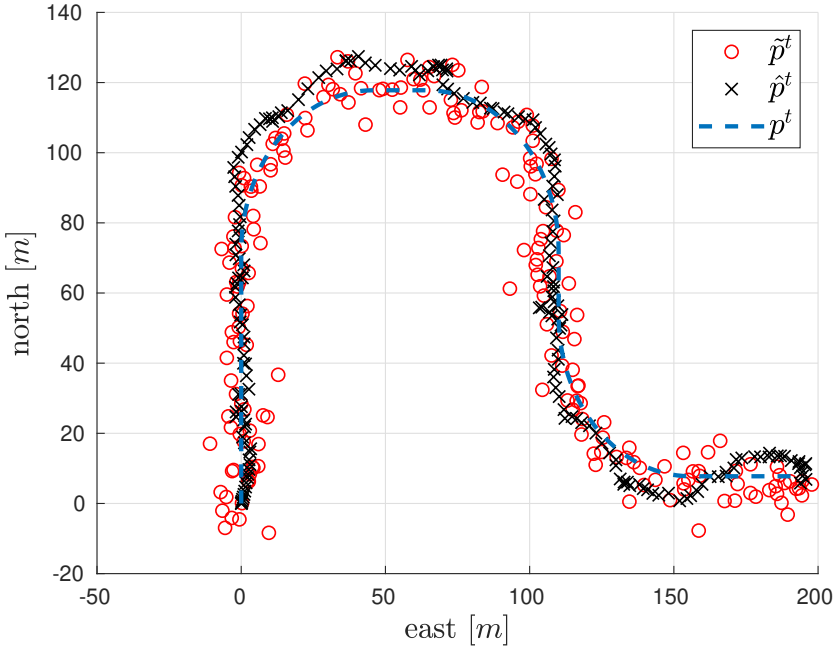


Figure 6.13: Filtering performance for an SIS particle filter with parameters in table 6.5 on the complex data set with biases.

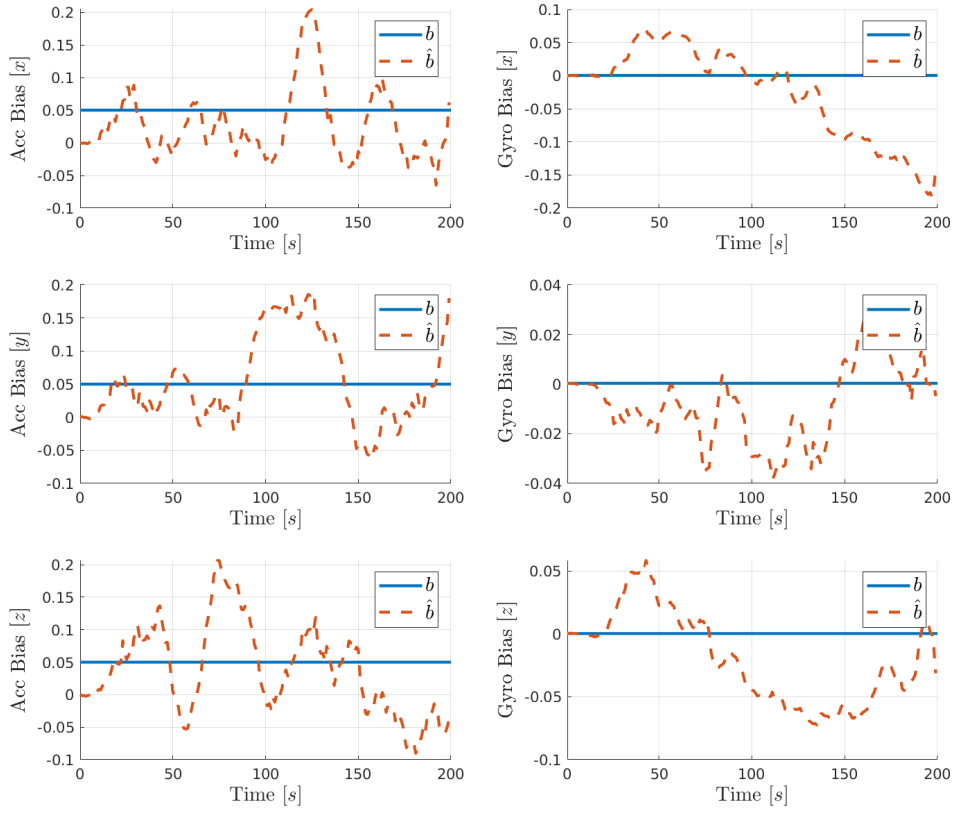


Figure 6.14: Bias estimates for the PF defined by table 6.5 and shown in fig. 6.13.

7. Discussion

This section will reintroduce and review the results introduced in sections 6.1 and 6.2.

7.1 Without Bias

In section 6.1 it can be clearly seen that the particle filter implementation is capable of creating state estimates which improve on the accuracy given by the noisy measurements. Quantitative data has also been given which supports this case both for the simple and complex artificial data sets. This serves as a demonstration that the implementation of the particle filter works so a more complex model can be approached.

This is in spite of the implementation errors to be reviewed in section 7.2. Upon correction of said errors even better performance could be expected because the orthonormality problem can be shown to exist even in the implementation of the PF without bias estimates, see fig. 7.1.

7.2 With Bias

The more complex model in question is with IMU biases included in the artificial data. Section 6.2 addresses and attempts to solve this problem by generating bias estimates according to the random walk model given in eq. (5.4). As can be seen in the plots, figs. 6.13 and 6.14, the filter performance is very poor; upon further reflection this is likely due to a failure in addressing one potential flaw with the DCM method of attitude representation.

Given the nature of the dynamic equation for a rotation matrix, see eq. (1.13), and the DCM representation the estimated values, $\hat{r}_{11} \dots \hat{r}_{33}$, are directly the matrix elements. As such any noise when evaluating

$$\dot{\mathbf{R}}_b^t = \mathbf{R}_b^t \boldsymbol{\Omega}_{tb}^b \tag{7.1}$$

will likely result in the matrix becoming non-orthonormal, which cannot happen as it violates one of the properties of a rotation matrix, see eq. (A.1). This theory was investigated by checking to see how often the simulation results in a non-orthonormal rotation matrix estimate, this is done by evaluating

$$||\mathbf{I} - (\hat{\mathbf{R}}_b^t)^T \hat{\mathbf{R}}_b^t|| \geq \epsilon \tag{7.2}$$

where ϵ is some tolerance to account for small rounding errors. The errors found by evaluating $||\mathbf{I}-(\hat{\mathbf{R}}_b^t)^T\hat{\mathbf{R}}_b^t||$ after every measurement update step for the complex data set with and without bias values can be seen in figs. 7.1 and 7.2. As can be

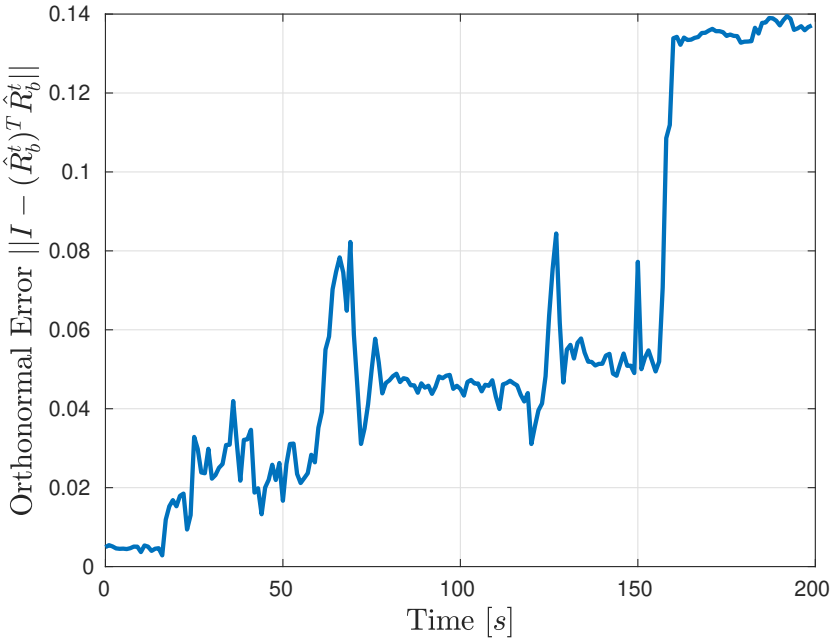


Figure 7.1: Orthonormal error over time evaluated after every measurement update step in the PF. This simulation is for a PF with the tunings described in section 6.1.4.

seen from the plot the deviation is quite large after some time and even more so for the simulation with biases.

A more developed navigation system implementation would make the effort to ensure orthonormality of the rotation matrix estimate, for which there are some established methods, e.g. [12]. Alternatively one could pursue a different methodology for attitude representation, for example Euler angles which are not susceptible to the same failing.

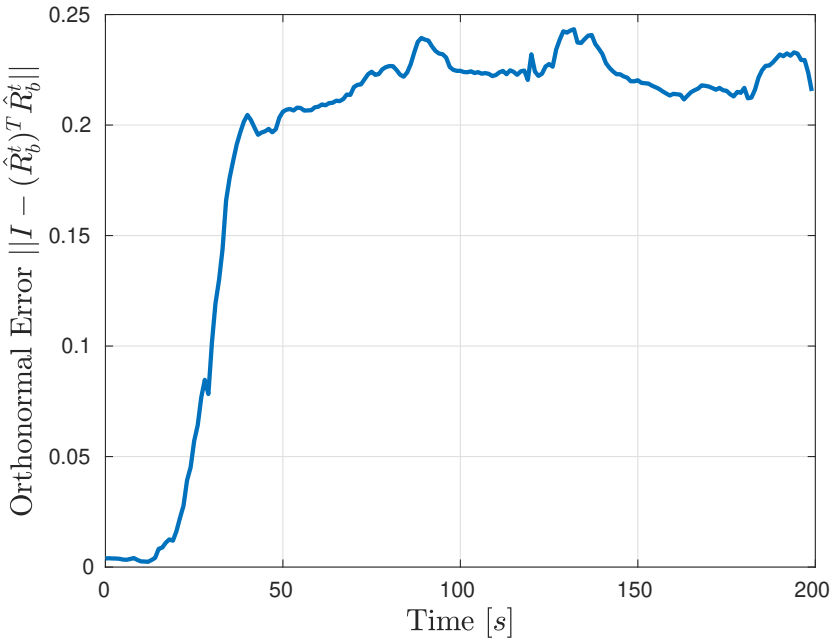


Figure 7.2: Orthonormal error over time evaluated after every measurement update step in the PF. This simulation is for a PF with the tunings described in section 6.2.

8. Conclusion

This paper introduces the strapdown INS with GNSS as the aiding sensor and attempts to solve the filtering problem using an SIS PF with resampling. The filtering problem that will be approached here is how to generate accurate position, attitude, and velocity estimates of a moving vehicle given noisy sensor data for the acceleration and angular velocity, from an IMU, as well as position, from a GNSS sensor, of this vehicle.

The PF implementation is able to address the INS problem insofar as it is able to reduce the noise of the GNSS measurements of the vehicle's position given an IMU sensor model without bias offsets. It is, however, in its current state unable to provide satisfactory filtering performance for the case of IMU sensor models with bias offsets. This is likely due to some problems with maintaining rotation matrix estimate orthonormality, discussed in section 6.2. This implementation utilizes a formulation of the full state model, see chapter 2, for the kinematic equations and has been made such as to accommodate for the possibility of considering non-Gaussian noise sources.

A more formal comparison of the particle filter implementation and an industry standard EKF solution would be interesting. In that case, for a more meaningful comparison, the implementation problems discussed in chapter 7 should be resolved. In addition one could leverage more advanced filtering techniques and more advanced sensor models than what is considered here for increased performance from the PF.

A. Further Explanations

A.1 Rotation Matrix

Properties Rotation matrices are defined to be orthogonal, meaning:

$$\left(\mathbf{R}_a^b\right)^{-1} = \left(\mathbf{R}_a^b\right)^T \tag{A.1}$$

Conveniently, changing the rotation order is the same calculation

$$\mathbf{R}_b^a = (\mathbf{R}_a^b)^T \tag{A.2}$$

Usage Rotation matrices are used to alter the reference frame data points are expressed with respect to. For example, given a vector expressed with respect to frame a $\mathbf{v}^a$, the same vector expressed with respect to frame b is found by

$$\mathbf{v}^b = \mathbf{R}_a^b \mathbf{v}^a \tag{A.3}$$

Rotations matrices can be applied to matrices as well, let $\mathbf{\Omega}^a$ be a matrix defined with respect to frame a and let $\mathbf{v}_1^a$ and $\mathbf{v}_2^a$, two vectors also defined in frame a, be related by

$$\mathbf{v}_1^a = \mathbf{\Omega}^a \mathbf{v}_2^a \tag{A.4}$$

By applying a rotation matrix this can be rewritten with vectors defined in frame b as

$$\mathbf{R}_b^a \mathbf{v}_1^b = \mathbf{\Omega}^a \mathbf{R}_b^a \mathbf{v}_2^b \tag{A.5}$$

Rearranging this equation gives

$$\mathbf{v}_1^b = \mathbf{R}_a^b \mathbf{\Omega}^a \mathbf{R}_b^a \mathbf{v}_2^b \tag{A.6}$$

which is equivalent to

$$\mathbf{v}_1^b = \mathbf{\Omega}^b \mathbf{v}_2^b \tag{A.7}$$

where the matrix defined with respect to frame b $\mathbf{\Omega}^b$ can be calculated using the rotation matrix $\mathbf{R}_b^a$ by

$$\mathbf{\Omega}^b = \mathbf{R}_a^b \mathbf{\Omega}^a \mathbf{R}_b^a \tag{A.8}$$

A.2 Skew-Symmetric Matrix

A skew-symmetric matrix is a square matrix whose transpose equals its negative. That is $\mathbf{A}$ is skew-symmetric if it satisfies

$$\mathbf{A}^T = -\mathbf{A} \tag{A.9}$$

Vectors can be used to define skew-symmetric matrices, give the vector $\mathbf{a} = \begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix}^T$ the skew-symmetric matrix $[\mathbf{a}\times]$ is

$$[\mathbf{a}\times] = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \tag{A.10}$$

This matrix is very useful because the cross product between two vectors can be expressed using skew-symmetric matrices. Given $\mathbf{a} = \begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix}^T$ and $\mathbf{b} = \begin{bmatrix} b_1 & b_2 & b_3 \end{bmatrix}^T$ the cross product

$$\mathbf{a} \times \mathbf{b} = [\mathbf{a}\times]\,\mathbf{b} \tag{A.11}$$

A.3 Small angle Rotation

The small angle rotation is an approximation for the rotation matrix between two coordinate systems with marginal differences in orientation. Given the rotations $\delta\boldsymbol{\theta} = \begin{bmatrix} \delta\theta_1 & \delta\theta_2 & \delta\theta_3 \end{bmatrix}^T$ the small angle rotation matrix can be written as

$$\mathbf{R}_a^b = \begin{bmatrix} 1 & \delta\theta_3 & -\delta\theta_2 \\ -\delta\theta_3 & 1 & \delta\theta_1 \\ \delta\theta_2 & -\delta\theta_1 & 1 \end{bmatrix} \tag{A.12}$$

or equivalently

$$\mathbf{R}_a^b = \mathbf{I} - \delta\boldsymbol{\Theta} \tag{A.13}$$

where $\delta\boldsymbol{\theta}$ is the skew symmetric representation of $\delta\boldsymbol{\theta}$.

A.4 Euler Method

The Euler method is a method of discretizing a continuous time model. It is a rather simple and computationally inexpensive discretization calculation, in addition it is also not extremely accurate and relatively costly in terms of accuracy versus step size.

The basic assumption of the Euler method is that the function is sufficiently linear for the chosen step size T_s. If this is the case one can approximate x_{k+1} by

$$x_{k+1} = x_k + T_s f(x_k) \tag{A.14}$$

where f is the continuous time differential equation of the form

$$\dot{x}(t) = f(x(t)) \tag{A.15}$$

This provides a relatively accurate approximation of x_{k+1} assuming a sufficiently small T_s. Should the approximation not be sufficiently accurate for a given sampling rate one could look into more advanced techniques, e.g. higher order Runge-Kutta [13].

A.5 Bayes' Theorem

Bayes' theorem describes the probability of an event based on prior conditions that may be relevant to the event. Mathematically it is written as

$$p(A|B) = \frac{p(B|A)p(A)}{p(B)} \tag{A.16}$$

where A and B are events in this context.

- $p(A|B)$ is the conditional probability, i.e. the likelihood of A given that B is true.
- $p(B|A)$ is also a conditiional probability, i.e. the likelihood of B given that A is true.
- $p(A)$ and $p(B)$ are the total probabilities of A and B. Also called the marginal probability.

A.6 Importance Sampling

This will be a brief introduction to importance sampling, a more thorough derivation is found in [5]. Importance sampling can be used to evaluate Monte Carlo integral estimates of the type

$$I = \int f(x)\pi(x)dx \tag{A.17}$$

for the case where $\pi(x)$ is a probability density. Suppose we cannot generate samples from $\pi(x)$ and can only sample from a similar probability density $q(x)$. Given that $\pi(x)$ and $q(x)$ have the same support the integral in eq. (A.17) can be rewritten as

$$I = \int f(x)\frac{\pi(x)}{q(x)}q(x)dx \tag{A.18}$$

An estimate of this integral can the be computed by generating $N >> 1$ samples, denoted $\{x^i\}_{i=1}^N$, from $q(x)$ and calculating

$$I_N = \sum_{i=1}^{N} f(x^i)w(x^i) \tag{A.19}$$

where

$$\tilde{w}(x) = \frac{\pi(x)}{q(x)} \tag{A.20}$$

are the un-normalized importance weights. The weights can then be normalized by

$$w(x^i) = \frac{\tilde{w}(x^i)}{\sum_i \tilde{w}(x^i)} \tag{A.21}$$

B. Bibliography

[1] J. Farrell, *Aided navigation: GPS with high rate sensors*. McGraw-Hill, Inc., 2008.

[2] S. Zhao, "Time derivative of rotation matrices: A tutorial," 2016.

[3] E. A. Wan and R. Van Der Merwe, "The unscented kalman filter for nonlinear estimation," in *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*, Oct 2000, pp. 153–158.

[4] F. Gustafsson, "Particle filter theory and practice with positioning applications," *IEEE Aerospace and Electronic Systems Magazine*, vol. 25, no. 7, pp. 53–82, 2010.

[5] B. Ristic, S. Arulampalam, and N. Gordon, *Beyond the Kalman filter: Particle filters for tracking applications*. Artech house, 2003.

[6] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, Feb 2002.

[7] R. Douc and O. Cappe, "Comparison of resampling schemes for particle filtering," in *ISPA 2005. Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis, 2005.*, Sep. 2005, pp. 64–69.

[8] J. Hol, T. Schön, and F. Gustafsson, "On resampling algorithms for particle filters," in *IEEE Nonlinear Statistical Signal Processing Workshop*, 10 2006, pp. 79–82.

[9] *MTi User Manual: MTi 10-series and MTi 100-series 5th generation*, Xsens Technologies B.V., 12 2017.

[10] *Satellite Compass Operator's Manual*, Furuno Electric, 1 2017.

[11] T. I. Fossen and T. Perez, "Marine Systems Simulator (MSS)," 2004. [Online]. Available: https://github.com/cybergalactic/MSS

[12] J. Mao, "Optimal orthonormalization of the strapdown matrix by using singular value decomposition," *Computers & Mathematics with Applications*, vol. 12, no. 3, Part 1, pp. 353 – 362, 1986. [Online]. Available: http://www.sciencedirect.com/science/article/pii/089812218690194X

[13] S. Gill, "A process for the step-by-step integration of differential equations in an automatic digital computing machine," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 47, no. 1, p. 96–108, 1951.