Formulation, development and testing of real-time algorithm for in-flight store IMU alignment

Himani Sinhmar

October 5, 2018

Contents

1 Synopsis

Stores are required to be dropped from a variety of flight vehicles, from airplanes to parachutes. Modern stores are 'guided— they carry an Inertial Measurement Unit (IMU), typically consisting of gyroscopes and accelerometers, which help in navigation and guidance. The IMU provides estimates of the store states (position, velocity, angular rates and attitude) in flight. This information is processed by an onboard controller which then appropriately varies the store control effectors in flight to guide the store to a pre-decided target. This information usually suffers from error, e.g., due to noise, drift, etc.

The standard way to filter out the noise is to use a Kalman filter. When preparing to launch the store from the mother ship, the store IMU must be correctly initialised along with the store Kalman filter. Once this is done, the store IMU plus Kalman filter will be able to provide good estimates of the store states to the onboard controller. Any drift in the IMU sensors will then be periodically corrected by the other sensors on the store (e.g., GPS, magnetometer) and accurate navigation and guidance can be achieved. Therefore, much hinges on how accurately the store IMU is initialised.

Week 1 report focuses on presenting the requirements and specification of the problem given by the client in an elaborated manner. A through literature survey was carried out which is narrowed down to a few well well written papers on the transfer alignment problem and alignment methods for the Strapdown Inertial Navigation System (SINS).

2 Requirements

This project is directed towards developing a real time algorithm for an in-flight guiding store IMU alignment. The Kalman filter need to be designed so as to keep the computing cost and time under check. The grade of IMU usually used in stores (as against those used in high-performance aircraft and launch vehicles) is prone to drift, which needs to be periodically set right. This is typically done by carrying another sensor/s which is relatively immune from the problem of drift, such as GPS or magnetometer. Hence it is also required to design these supporting sensors.

The standard to initialize the store IMU and Kalman filter may be to copy navigation information supplied by master IMU to the OnBoard Computer (OBC) of the slave directly. This approach is called one shot Transfer alignment. Although this method has the advantage of simplicity to implement, in real time environment this method may not give accurate estimation of navigation information. While copying information of master OBC to that of the slave, the angular displacement existing between master and slave axes of references (if exists at that particular instant when copying) will be retained as misalignment error in the slave and as a result of that, slave OBC will start with erroneous information.

On the other hand, under certain circumstances, it may not be possible to transfer the states from the parent ship IMU to the store IMU. In this case, the store IMU must pick up the correct states by itself with the help of the additional sensors it carries (e.g., GPS, magnetometer). This is the 'Self Alignment' problem. The major issue here is to make sure that the initial 'guess' state is close enough to the actual values so that the linear Kalman filter is able to converge and do so in a reasonably short time.

An algorithm for the store IMU self alignment problem should be developed and tested. The same algorithm with minimal changes is then to be applied to the Transfer Alignment problem. The work requires:

1. Modeling of the sensors, both IMU and other supporting sensors, of the store with standard error models
2. Formulating the Self Alignment problem by choosing the initial estimate of the states and the Kalman filter
3. Writing a linear Kalman filter and checking its convergence and stability.
4. Integrating the sensor models, initial estimates and Kalman filter to provide accurate and speedy estimates of the store states.
5. Integrating the other supporting sensors to correct for drift in-flight.

6. Testing the algorithm in all respects.

7. Modifying/ using the algorithm to address the Transfer Alignment problem, such that the final algorithm can be used to solve either problem — Self Alignment or Transfer Alignment.

8. Preparing a detailed report with formulation, development details and test cases, including convergence, stability, accuracy and real-time ability, accompanied by the final code

3 Specifications

When preparing to launch the store from the mother ship, its IMU is initially 'switched on.' At this time, the store IMU must be correctly initialized along with the store Kalman filter. Transfer alignment (TA) is the process of initializing the navigation sensors of launched from a platform. The navigation device needs to know its initial position, velocity and attitude information before entering the working condition. Through this method the misalignment between aligning and aligned body is estimated, which facilitates navigation of aligning system during its post ejection course of flight. The so-called alignment refers to the process of determining the coordinates of the inertial navigation system with respect to the reference coordinate system. Measurements provided by navigation sensors of aligning system are noisy compared to those of the launch platform. Objective of TA process is to estimate the misalignment as accurately as possible in order to improve dead reckoning of the store's IMU. It is easy to know the initial position and velocity information (such as with the help of GPS), but the initial attitude information needs to be aligned. Fig. 1 (1) summarizes the requirements of the given problem.

3.1 Design and Implementation

State space formulation of TA problem facilitates the use of stochastic estimator in designing TA algorithm. The process of TA may be depicted as in Fig. 2 (2), where differences in measurements between master INS and slave INS, or differences between resolved measurement values of aligning system along the fixed reference axes and expected values of those parameters along the same axes are processed by the algorithm to produce corrected estimates of navigation information. These differences become state variables in the state space representation of the system. General formulation of TA problem in state space representation takes the form of a linear discrete time system.

Inertial navigation system error models will be considered here. It is based on the INS error equations (error in velocity, position and attitude). The system error dynamics can be written in the matrix form as

$$\dot{X}(t) = F(t)X(t) + W$$

Where X is the error state vector, F is the system matrix, and W is the process noise vector. A fifteen state Kalman filter can be used for data fusion and error estimation, which includes nine navigation solution errors of three dimensional position, velocity and attitude, three accelerometer bias errors, and three gyro drifts. Measurements are taken from both : master and slave INS or slave INS and aiding sensors such as GPS (Self alignment), magnetometer, etc. Other techniques such as inertial measurement matching can be used with the help of an appropriate measurement matrix. Some of very popular inertial measurement matching methods are acceleration matching, velocity matching, position matching, angular rate matching or matching of some combination of these measurements. For this purpose measurements provided by sensors of both IMU (master and slave) may be compared with each other and resolved to estimate the alignment error. For example, in velocity matching the velocity errors are taken as observations for the filter.

3.1.1 Fusion with aiding sensors

One of the major drawbacks of INS is the error associated with sensor measurements and this error increases due to dead reckoning (integration scheme employed by INS for navigation). Although various filters are used to reduce or correct noisy measurements, aiding from one or more different sensors provides more accurate estimate. The method of updating states of a system with the help of two or more types of sensors, like GPS, odometer compass, astronavigation system, etc., is known as sensor fusion technique. In case of TA problem, INS measurement may be aided with barometric altimeter data, which provides measurement of height of the moving body.

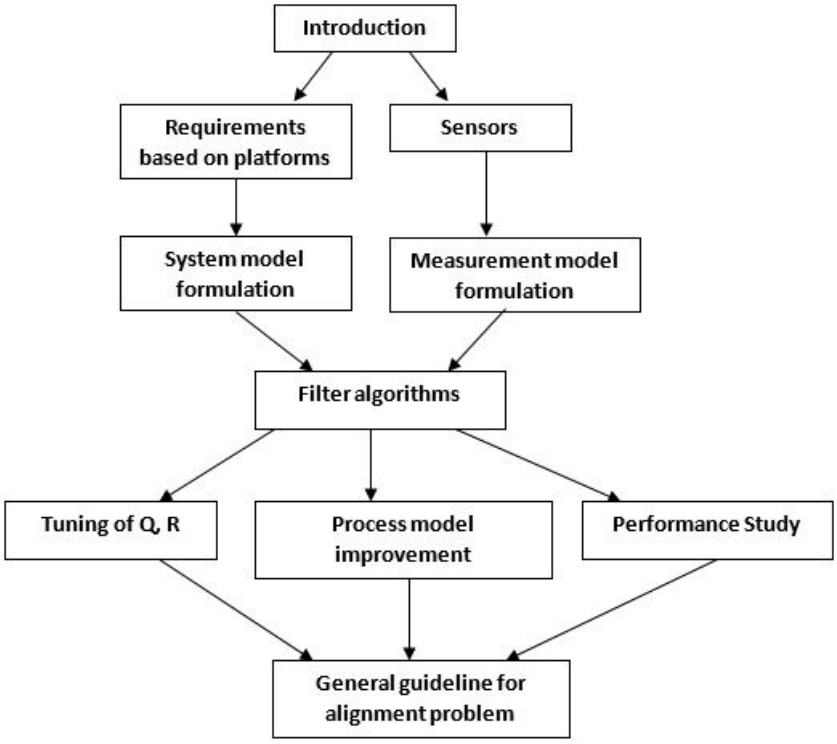


Figure 1: A Summary of the Requirements

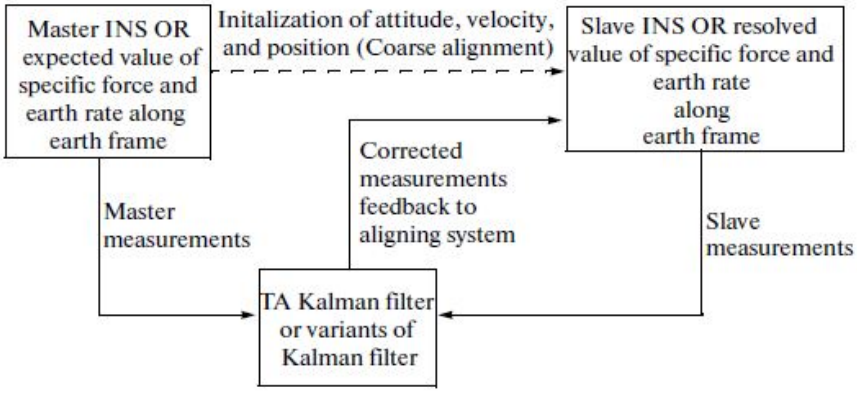


Figure 2: General form of TA problem

3.1.2 Parameter Tuning

Various factors like delay between master and slave measurement update, effect of flexure and vibration, quantization error, etc., may affect sensor measurement. To achieve accurate estimation of system states, a reasonable noise model needs to be associated with the measurement model of the algorithm. In simulation environment, white Gaussian noise is assumed to fit with KF requirement. Proper tuning of filter parameters, particularly Q (process noise covariance) and R (measurement noise covariance) may improve the performance of KF in terms of accuracy. It is difficult to set any value for the process noise covariance, Q, because of the lack of observation of the states to be estimated. It has been shown that filter performance will be slowed down if measurement noise is considered much higher than system noise, as gain of KF used will become too low. On the contrary, the system will become unstable and will produce biased estimates, if system noise is taken to be higher than measurement noise, because gain of KF will become large on this occasion. Generally, the measurement noise covariance R is constructed based on the maximum allowable differences in measurements of the mother and daughter, construction of Q is

based on sensor bias and initial noise covariance P_0 is constructed based on Q and R.

4 Master Schedule

Item	Work Scheduled	Period (in weeks)
1	Requirement and Specifications	T_0 —$T_0 + 1$
2	Formulation of Self Alignment Problem with sensor models, initial estimate and Kalman filter	$T_0 + 1$ —$T_0 + 3$
3	Development of integrated Self Alignment algorithm (coding)	$T_0 + 3$ —$T_0 + 6$
4	Testing for convergence, stability, accuracy, speed	$T_0 + 6$ —$T_0 + 8$
5	Test case (Self Alignment problem)	$T_0 + 8$ —$T_0 + 9$
6	Modification to Transfer Alignment problem — formulation, development, testing	$T_0 + 9$ —$T_0 + 12$
7	Transfer of technology	$T_0 + 12$ —$T_0 + 13$

5 Formulation of Self Alignment Problem

An INS is a navigation system which depends entirely on inertial measurements for navigation. An INS consists of accelerometers which measure the translatory acceleration and gyroscopes which measure the angular rotation of the system. This sensor array is called an Inertial Measurement Unit (IMU). Using the measurements from the IMU, the INS can calculate the current attitude, velocity and position of the system starting from some known initial point. An INS simply integrates measured acceleration and rotation rate to determine the position, velocity and orientation of the body frame.

Inertial navigation consists of three major parts: Alignment, navigation and aiding. During the alignment phase, the alignment algorithm estimates the initial value of the body frame with respect to the navigation frame, that is, the roll, pitch and yaw of the system. This is used by the navigation algorithm to transform the measurements from the body frame to the navigation frame. If the IMU is in-flight motion during alignment, this can be done by employing TRIAD algorithm. When the alignment phase has obtained a sufficiently accurate estimate of the roll, pitch and yaw, the navigation phase can begin. During navigation, a navigation algorithm continuously calculate the roll, pitch and yaw from inputs from the gyroscopes, so acceleration measurements can be resolved in the navigation frame and integrated to yield velocity and subsequently positions with respect to the earth. Ultimately, this results in the latitude, longitude, height, velocity and attitude outputs, which are the primary variables of interest in inertial navigation systems.

5.1 Mathematical Notations

In this report scalars are notated with no-bold lower case letters, vectors with bold lower case letters and matrices are capital bold letters. All frames used in this thesis span a three-dimensional Euclidean space. The basic vector of $n-$ frame will be denoted by n_x, n_y, n_z, where x, y and z are the principal axes of the frame. The position vector $\mathbf{r}$ referenced in the n-frame will be denoted by $\mathbf{r}^n$ and the components are as follows

$$\mathbf{r}^n = \begin{bmatrix} r_x^n & r_y^n & r_z^n \end{bmatrix}^T$$

Column matrices that are coordinatized in a particular reference frame can be transformed to another frame by the Direction Cosine Matrix (DCM)

$$\mathbf{r}^n = \mathbf{C}_b^n \mathbf{r}^b$$

where $\mathbf{C}_b^n$ = DCM , transforms a column vector from body frame coordinates, b to navigation frame coordinates n. The relative angular velocity of two frames is usually denoted as column

vectors with the subscript indication the rotational direction, where $\boldsymbol{\omega}_{in}^{n}$, is the angular velocity of the n-frame relative to the i frame coordinatized in the n-frame. Angular velocities follow the usual rules of vector addition. If rotations occur between a number of coordinate frames the subscript notation facilitates the statement of the mathematical relationship (Britting,1971)

$$\boldsymbol{\omega}_{ib}^{n} = \boldsymbol{\omega}_{ie}^{n} + \boldsymbol{\omega}_{eb}^{n}$$

In the matrix algebra of rotations it is often necessary to express the angular velocity in skew symmetric form. The skew symmetric form of $\boldsymbol{\omega}$ is denoted by its upper case form $\boldsymbol{\Omega}$ as

$$(\boldsymbol{\omega}_{in}^{n}\times) = \boldsymbol{\Omega}_{in}^{n} = \begin{bmatrix} 0 & -\omega_{in,z}^{n} & \omega_{in,y}^{n} \\ \omega_{in,z}^{n} & 0 & -\omega_{in,x}^{n} \\ -\omega_{in,y}^{n} & \omega_{in,x}^{n} & 0 \end{bmatrix}$$

5.1.1 ECIF

The Earth-Centered Inertial Frame (ECIF), denoted by the symbol i, is centered at the Earth's center of mass. The z-axis points along the Earth's axis of rotation from the center to the North Pole. The x-axis lies within the equatorial plane and the y-axis completes the right-hand set and does not rotate with the Earth. When the navigation solution is initialized the ECIF is aligned to the Earth-Centered Earth-Fixed Frame (ECEF)

5.1.2 ECEF

The Earth-Centered Earth-Fixed Frame, denoted by the symbol e, is similar to the ECIF, except that all the axes remain fixed with respect to the Earth. The x-axis points towards the mean meridian of Greenwich (0° longitude), z-axis points along the Earth's axis of rotation from the center to the North Pole and the y-axis completes the right-hand set.

5.1.3 Navigation Frame

The Navigation Frame (NF), denoted by the symbol n, is a local geodetic frame which has its origin at the location of the navigation system. In this report a common geodetic frame is used, the North, East, Down frame (NED), where the x-axis is pointing towards the geodetic north, z-axis is orthogonal to the reference ellipsoid, pointing down, and y-axis completes the right-hand set. The turn rate of the navigation frame, with respect to the Earth-fixed frame, $\boldsymbol{\omega}_{en}$ is governed by the motion of origin of the NF with respect to the earth. This is often referred to as the transport rate.

5.1.4 Body Frame

The body frame (BF), denoted by the symbol b, is rigidly attached to and defined within the vehicle carrying the navigation system. Definitions for this frame have the x-axis along the vehicle longitudinal axis, the z-axis downward, and the y-axis pointed outwards, completing the right-hand set. For angular motion the x-axis is the roll (ϕ), y-axis is the pitch (θ) and the z-axis is the yaw or heading (ψ).

5.2 Symbols

- ϵ : Eccentricity of the ellipsoid
- $\boldsymbol{\epsilon}$:Attitude error vector
- θ : Pitch
- λ : Longitude
- φ : Latitude
- ϕ : Roll
- $\boldsymbol{\Phi}$ State Transition Matrix

- ψ : Yaw
- $\boldsymbol{\omega}$: Angular rate vector
- $\boldsymbol{\Omega}$: Skew symmetric matrix form of $\boldsymbol{\omega}$
- $\mathbf{C}$: Direction Cosine Matrix
- $\mathbf{e}$: Measurement noise vector
- $\mathbf{f}$: Specific force vector
- $\mathbf{F}$: Dynamics matrix
- $\mathbf{G}$: Design matrix of system noise
- $\mathbf{g}$: Gravity vector
- h : height
- $\mathbf{H}$: Design matrix for measurement
- $\mathbf{I}$: Identity matrix
- $\mathbf{K}$: Kalman gain matrix
- M : Radius of curvature in meridian
- N : Radius of curvature in prime vertical
- $\mathbf{P}$: Covariance matrix
- $\mathbf{r}$: Position vector
- r_e : Semi-major axis of the ref. ellipsoid
- r_p : Semi-minor axis of the ref. ellipsoid
- $\mathbf{u}$: Continuous time system noise vector
- $\mathbf{v}$: Velocity vector
- $\mathbf{x}$: State vector
- $\mathbf{z}$: Measurement vector
- $\mathbf{w}$: Linearized system noise vector

The main reference for the subsequent sections is [3].

6 Inertial Navigation

6.1 Geometry of the earth

To develop an Inertial Navigation System (INS) error model it is necessary to model certain aspects of the Earth. The model of the Earth used in this report is based on the World Geodetic System (WGS-84), which defines many of the constants that are used in the INS error model, as well as the gravity model. In order to determine the position on Earth using inertial measurements, it is necessary to make some assumptions regarding the shape of the Earth. An ellipsoid model is used to reference the Earth geometry. In accordance with this model, the following parameters are defined : The length of the semi-major and semi-minor axes, and the eccentricity of the ellipsoid are defined as

$$
\begin{aligned}
r_e &= 6378137m \\
r_p &= 6356752.3m \\
\epsilon &= \left(1 - \left(\frac{r_p^2}{r_e^2}\right)^{1/2}\right)
\end{aligned}
$$

The meridian radius of curvature M and a transverse radius of curvature N may be derived in accordance with the following equations

$$M = \frac{r_e(1-\epsilon^2)}{(1-\epsilon^2 sin^2\varphi)^{3/2}}$$
$$N = \frac{r_e}{(1-\epsilon^2 sin^2\varphi)^{1/2}}$$

6.2 Transformation

The DCM from the e-frame to the n-frame can be expressed as

$$\mathbf{C}_e^n = \begin{bmatrix} -sin(\varphi)cos(\lambda) & -sin(\varphi) & cos(\varphi) \\ -sin(\lambda) & cos(\lambda) & 0 \\ -cos(\varphi)cos(\lambda) & -cos(\varphi)sin(\lambda) & -sin(\varphi) \end{bmatrix} \tag{1}$$

Due to the fact that the coordinate systems are orthogonal the DCM in opposite direction, from n-frame (NF) to e-frame (ECEF) is obtained by transposing the $\mathbf{C}_e^n$, $\mathbf{C}_n^e = (\mathbf{C}_e^n)^T$. The DCM from $b-$ frame to $n-$frame is given by

$$\mathbf{C}_n^b = \begin{bmatrix} cos(\theta)cos(\psi) & cos(\theta)sin(\psi) & -sin(\theta) \\ -cos(\phi)sin(\psi)+sin(\phi)sin(\theta)cos(\psi) & cos(\phi)cos(\psi)+sin(\phi)sin(\theta)sin(\psi) & sin(\phi)cos(\theta) \\ sin(\phi)sin(\psi)+cos(\phi)sin(\theta)cos(\psi) & -sin(\phi)cos(\psi)+cos(\phi)sin(\theta)sin(\psi) & cos(\phi)cos(\theta) \end{bmatrix} \tag{2}$$

The Euler angles can be determined from the DCM $\mathbf{C}_b^n$ by reverse transformations

$$\phi = arctan2(C_{32}, C_{33}) \tag{3a}$$
$$\theta = -arcsin(C_{31}) \tag{3b}$$
$$\psi = arctan2(C_{21}, C_{11}) \tag{3c}$$

where C_{ij} are the (i, j)-th elements of the DCM, $\mathbf{C}_b^n$ matrix.

6.3 Rotation Rate

The Earth's rotational rate is well known and can be assumed to be a constant

$$\omega_e = 7.29211510^{-5} rad/s \tag{4}$$

The rotational rate vector of the e-frame with respect to the i-frame is defined as

$$\boldsymbol{\omega}_{ie}^e = \begin{bmatrix} 0 & 0 & \omega_e \end{bmatrix}^T \tag{5}$$
$$\boldsymbol{\omega}_{ie}^n = \mathbf{C}_e^n \boldsymbol{\omega}_{ie}^e = \begin{bmatrix} \omega_e cos(\varphi) & 0 & -\omega_e sin(\varphi) \end{bmatrix}^T \tag{6}$$

The rate of change of latitude and longitude is expressed in terms of M and N as follows

$$\dot{\varphi} = \frac{v_N}{M+h} \tag{7a}$$
$$\dot{\lambda} = \frac{v_e}{(N+h)cos\varphi} \tag{7b}$$

where h is the ellipsoidal height, and the transport rate represent the turn rate of the n-frame in respect to the e-frame is expressed as follows

$$\boldsymbol{\omega}_{en}^n = \begin{bmatrix} \dot{\lambda}cos\varphi & -\dot{\varphi} & -\dot{\lambda}sin\varphi \end{bmatrix}^T = \begin{bmatrix} \frac{v_E}{(N+h)} \\ -\frac{v_N}{(M+h)} \\ -\frac{v_E tan\varphi}{(N+h)} \end{bmatrix} \tag{8}$$

where v_N, v_E and v_D are velocities in the NED frame. The equation for $\boldsymbol{\omega}_{in}^n$ can be obtained as $\boldsymbol{\omega}_{in}^n = \boldsymbol{\omega}_{ie}^n + \boldsymbol{\omega}_{en}^n$

$$\boldsymbol{\omega}_{in}^n = \begin{bmatrix} \omega_e cos\varphi + \frac{v_E}{N+h} \\ -\frac{v_N}{M+h} \\ -\omega_e sin\varphi - \frac{v_E tan\varphi}{N+h} \end{bmatrix} \tag{9}$$

6.4 Inertial Navigation Equations

Differential equations describing navigation states are developed in the following section. In developing the equations, the objective is to form an expression for navigation states in terms of sensed accelerations and angular rates available from accelerometers and gyros, respectively.

6.4.1 Position Equations

The position in the n-frame is expressed by curvilinear coordinates

$$\mathbf{r}^n = \begin{bmatrix}\varphi & \lambda & h\end{bmatrix}^T \tag{10}$$

The velocities in the n-frame are defined by

$$\mathbf{v}^n = \begin{bmatrix}v_N \\ v_E \\ v_D\end{bmatrix} = \begin{bmatrix}M+h & 0 & 0 \\ 0 & (N+h)cos\varphi & 0 \\ 0 & 0 & -1\end{bmatrix} \begin{bmatrix}\dot{\varphi} \\ \dot{\lambda} \\ \dot{h}\end{bmatrix} \tag{11}$$

The derivatives of the latitude, longitude, and height can be written as

$$\dot{\mathbf{r}} = \begin{bmatrix}\dot{\varphi} \\ \dot{\lambda} \\ \dot{h}\end{bmatrix} = \begin{bmatrix}\frac{1}{M+h} & 0 & 0 \\ 0 & \frac{1}{(N+h)cos\varphi} & 0 \\ 0 & 0 & -1\end{bmatrix} \begin{bmatrix}v_N \\ v_E \\ v_D\end{bmatrix} \tag{12}$$

6.4.2 Velocity Equations

The Earth's relative velocity vector, $\mathbf{v}^n$, in the rotating navigation n-frame is defined in terms of the rotating ECEF e-frame position as

$$\mathbf{v}^n = \mathbf{C}_e^n \dot{\mathbf{r}}^e$$

A detailed derivation of the velocity equation is given in [3]. The final velocity equation is defined as :

$$\dot{\mathbf{v}}^n = \mathbf{C}_b^n \mathbf{f}^b - (2\boldsymbol{\Omega}_{ie}^n + \boldsymbol{\Omega}_{en}^n)\mathbf{v}^n + \mathbf{g}^n \tag{13}$$

where the specific force vector, $\mathbf{f}$, is defined as the difference between the true acceleration in space and gravitational acceleration.

$$\mathbf{f}^n = \mathbf{C}_i^n \ddot{\mathbf{r}}^i - \bar{\mathbf{g}}^n \tag{14}$$

The $\bar{\mathbf{g}}^n$ is the gravitational acceleration and $\mathbf{g}$ is the gravity vector.

$$\mathbf{g}^n = \bar{\mathbf{g}}^n + \mathbf{C}_e^n(\boldsymbol{\Omega}_{ie}^e \boldsymbol{\Omega}_{ie}^e \mathbf{r}^e)$$

The n-frame velocity can also be expressed using the vector cross-product form for the rotation rates as

$$\dot{\mathbf{v}}^n = \mathbf{C}_b^n \mathbf{f}^b - (2\boldsymbol{\omega}_{ie}^n \times + \boldsymbol{\omega}_{en}^n \times)\mathbf{v}^n + \mathbf{g}^n$$

where $-2\boldsymbol{\omega}_{ie}^n \times \mathbf{v}^n$ is the acceleration caused by its velocity over the surface of a rotating Earth, often referred to as the Coriolis acceleration and $-\boldsymbol{\omega}_{en}^n \times [\boldsymbol{\omega}_{en}^n \times \mathbf{r}^n]$, $\mathbf{v}^n = \boldsymbol{\omega}_{en}^n \times \mathbf{r}^n$, is the centripetal acceleration experienced by the system owing to the rotation of the Earth, and is not separately distinguishable from the gravitational acceleration which arises through mass attraction, $\mathbf{g}$.

6.4.3 Attitude Equations

The attitude dynamics equation is represented by the DCM differential equation. In this form, once initialized, it can be integrated without specific expressions for each of its elements. The attitude dynamics is obtained from the following

$$\dot{\mathbf{C}}_b^n = -\boldsymbol{\Omega}_{bn}^n \mathbf{C}_b^n \tag{15}$$

where

$$\boldsymbol{\omega}_{bn}^n = \boldsymbol{\omega}_{in}^n - \mathbf{C}_b^n \boldsymbol{\omega}_{ib}^b \tag{16}$$

The rotation vector $\boldsymbol{\omega}_{ib}^b$ represents the output of the gyros.

6.5 Gravity Model

The gravity vector in the n-frame, $\mathbf{g}^n$, is approximated by the normal gravity vector $\begin{bmatrix} 0 & 0 & \gamma \end{bmatrix}^T$. Let us assume a spherical Earth model and the following simplified inverse square gravity model, where γ varies with altitude

$$\gamma = \gamma_0 (\frac{R}{R+h})^2 \tag{17}$$

where γ_0 is the normal gravity at $h = 0$ and $R = \sqrt{MN}$

6.6 Navigation Output

The latitude and longitude can be extracted from the $\mathbf{C}_e^n$ as :

$$\varphi = arccos(C_{13}) \tag{18a}$$
$$\lambda = arccos(C_{22}) \tag{18b}$$

7 Model for Simulator

The sensors that are needed : GNSS receiver (GPS measurements), magnetometer for initialization of orientation errors and IMU. Other sensors such as barometer can be fused later for further accuracy. In this section a simulation model is derived using the equations of motion. This model enables one to create repeatable test data as well as test the system under well-defined conditions such as erroneous sensor outputs. This simulation model should output the acceleration, rate and magnetic field sensed by the sensors when the vehicle is moving.

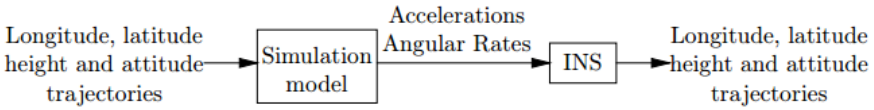


Figure 3: Simulator and INS

7.1 Modeling Accelerometer output

Rearranging and rotating Eq.(13):

$$\mathbf{f}^b = \mathbf{C}_n^b(\dot{\mathbf{v}}^n + (2\mathbf{\Omega}_{ie}^n + \mathbf{\Omega}_{en}^n)\mathbf{v}^n - \mathbf{g}^n) \tag{19}$$

The $\boldsymbol{\omega}_{ie}^n$ is given by Eq.(6). The $\mathbf{C}_n^b$ is calculated using the body rotations $\boldsymbol{\omega}_{bn}^n$ supplied by the user

$$\dot{\mathbf{C}_n^b} = \mathbf{C}_n^b \mathbf{\Omega}_{bn}^n \tag{20}$$

At start of the simulation, the $\mathbf{C}_n^b$ is initialized by the user. $\mathbf{v}^n$ and $\boldsymbol{\omega}_{en}^n$ are given by Eq.(11) and Eq.(8) respectively.

7.2 Modeling Gyro Output

The outputs from the gyros is

$$\boldsymbol{\omega}_{ib}^b = \mathbf{C}_n^b(\boldsymbol{\omega}_{ie}^n + \boldsymbol{\omega}_{en}^n + \boldsymbol{\omega}_{nb}^n) \tag{21}$$

which is calculated by inserting Eq.(6), Eq.(8) and $\boldsymbol{\omega}_{nb}^n$ given by the user.

7.3 Modeling Magnetometer output

A three-axis magnetometer measures the direction and the intensity of the magnetic field around the sensor. If this magnetic field is not perturbed, it corresponds to the Earth's magnetic field. The magnetometer measurements are the projection of this magnetic field $\mathbf{m}$ in the b- frame

$$\mathbf{m}^b = \mathbf{C}_n^b \mathbf{m}^n \tag{22}$$

The Earth's magnetic field is (in the Northern Hemisphere) directed toward the North Magnetic Pole and the inside of the Earth. With the aim of simulating all kinds of trajectories, an Earth's magnetic field model, called the World Magnetic Model, can be implemented in the simulator to provide the reference unperturbed magnetic value for any vehicle position. This model allows also simulating any movement between the two polar circles. The value of $\mathbf{m}^n$ is given by this model

This model was obtained by the interpolation of multiple measurement centers all on the Globe and is an empirical model. This is why it is only valid for five years and is constantly evolving. The implementation of this model also provides the value of the reference everywhere on the surface of the Earth. This allows simulations of trajectories everywhere, but also simulations of a long duration.

8 INS Error Model

In this section the nonlinear navigation state equations for position, velocity and attitude are linearized to obtain linear nine state error model, based on perturbation analysis. The purpose of the error model is to describe the propagation of the errors in the navigation equations. The errors are defined as the angle between the actual DCMs and the DCMs computed in the navigation computer and the velocity difference between the actual velocity of the INS and the velocity used by the navigation computer [5].

8.1 Perturbation Analysis

The error analysis in this thesis utilizes perturbation methods to linearize the nonlinear system differential equations. For example, the perturbation of the position, velocity, attitude DCM, and gravity can be expressed as

$$\hat{\mathbf{r}}^n = \mathbf{r}^n + \delta\mathbf{r}^n \tag{23a}$$

$$\hat{\mathbf{v}}^n = \mathbf{v}^n + \delta\mathbf{v}^n \tag{23b}$$

$$\hat{\mathbf{C}}_b^n = (\mathbf{I} - \mathbf{E}^n)\mathbf{C}_b^n + \delta\mathbf{r}^n \tag{23c}$$

$$\hat{\mathbf{g}}^n = \mathbf{g}^n + \delta\mathbf{g}^n \tag{23d}$$

where, e.g., $\hat{\mathbf{v}} = computed \quad velocity$, $\mathbf{v} = true \quad velocity$ and, $\delta v = computed \quad velocity \quad error$, g is the normal gravity vector and $\mathbf{E}^n$ is the skew symmetric form of the attitude errors

$$\mathbf{E}^n = (\boldsymbol{\epsilon}\times) = \begin{bmatrix} 0 & -\epsilon_D & \epsilon_E \\ \epsilon_D & 0 & -\epsilon_N \\ -\epsilon_E & \epsilon_N & 0 \end{bmatrix} \tag{24}$$

When substitutions of the type above are made for dependent variables in the nonlinear differential equations and products of error quantities are neglected, linear differential equations involving only the error quantities emerge. The derivation of Eq.(23c) is given in (Britting, 1971) The error state vector is defined as

$$\mathbf{x} = \begin{bmatrix} \delta\mathbf{r}^n & \delta\mathbf{v}^n & \boldsymbol{\epsilon}^n \end{bmatrix}^T \tag{25}$$

where $\delta\mathbf{r}^n$ is the position error vector, $\delta\mathbf{v}^n$ is the velocity error vector, and $\boldsymbol{\epsilon}^n$ is the attitude error vector, that defines $\mathbf{E}^n$.

8.2 Position Error Equation

Linearized position error dynamics can be obtained by perturbing Eq. (12). The position error dynamics equation can be obtained using the partial derivatives, because the position equations are a function of position and velocity

$$\delta\dot{\mathbf{r}}^n = \mathbf{F}_{rr}\delta\mathbf{r}^n + \mathbf{F}_{rv}\delta\mathbf{v}^n \tag{26}$$

where

$$\mathbf{F}_{rr} = \begin{bmatrix} \frac{\partial \dot{\varphi}}{\partial \varphi} & \frac{\partial \dot{\varphi}}{\partial \lambda} & \frac{\partial \dot{\varphi}}{\partial h} \\ \frac{\partial \dot{\lambda}}{\partial \varphi} & \frac{\partial \dot{\lambda}}{\partial \lambda} & \frac{\partial \dot{\lambda}}{\partial h} \\ \frac{\partial \dot{h}}{\partial \varphi} & \frac{\partial \dot{h}}{\partial \lambda} & \frac{\partial \dot{h}}{\partial h} \end{bmatrix} = \begin{bmatrix} 0 & 0 & -\frac{v_N}{(M+h)^2} \\ \frac{v_E sin\varphi}{(N+h)cos^2\varphi} & 0 & -\frac{v_E}{(N+h)^2 cos\varphi} \\ 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{F}_{rv} = \begin{bmatrix} \frac{\partial \dot{\varphi}}{\partial v_N} & \frac{\partial \dot{\varphi}}{\partial v_E} & \frac{\partial \dot{\varphi}}{\partial v_D} \\ \frac{\partial \dot{\lambda}}{\partial v_N} & \frac{\partial \dot{\lambda}}{\partial v_E} & \frac{\partial \dot{\lambda}}{\partial v_D} \\ \frac{\partial \dot{h}}{\partial v_N} & \frac{\partial \dot{h}}{\partial v_E} & \frac{\partial \dot{h}}{\partial v_D} \end{bmatrix} = \begin{bmatrix} \frac{1}{M+h} & 0 & 0 \\ 0 & \frac{1}{(N+h)cos\varphi} & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

8.3 Velocity Error Equation

The computed version of Eq.(13) can be written as

$$\dot{\hat{\mathbf{v}}}^n = \hat{\mathbf{C}}_b^n \tilde{\mathbf{f}}^b - (2\hat{\boldsymbol{\omega}}_{ie}^n + \hat{\boldsymbol{\omega}}_{en}^n) \times \hat{\mathbf{v}}^n + \mathbf{g}^n \tag{27}$$

The velocity error dynamics are given by

$$\delta\dot{\mathbf{v}}^n = \mathbf{F}_{vr}\delta\mathbf{r}^n + \mathbf{F}_{vv}\delta\mathbf{v}^n + (\mathbf{f}^n\times)\boldsymbol{\epsilon}^n + \mathbf{C}_b^n\delta\mathbf{f}^b \tag{28}$$

where

$$\mathbf{F}_{vr} = \begin{bmatrix} -2v_E\omega_e cos\varphi - \frac{v_e^2}{(N+h)cos^2\varphi} & 0 & -\frac{v_N v_D}{(M+h)^2} + \frac{v_E^2 tan\varphi}{(N+h)^2} \\ 2\omega_e(v_N cos\varphi - v_D sin\varphi) + \frac{v_E v_N}{(N+h)cos^2\varphi} & 0 & \frac{v_E v_D}{(N+h)^2} + \frac{v_E v_N tan\varphi}{(N+h)^2} \\ 2v_E\omega_e sin\varphi & 0 & \frac{v_E^2}{(N+h)^2} + \frac{v_N^2}{(M+h)^2} - \frac{2\gamma}{R+h} \end{bmatrix}$$

$$\mathbf{F}_{vv} = \begin{bmatrix} \frac{v_D}{M+h} & -2\omega_e sin\varphi - 2\frac{v_E tan\varphi}{N+h} & \frac{v_N}{M+h} \\ 2\omega_e sin\varphi + \frac{v_E tan\varphi}{N+h} & \frac{v_D + v_N tan\varphi}{N+h} & 2\omega_e cos\varphi + \frac{v_E}{N+h} \\ -2\frac{v_N}{M+h} & -2\omega_e cos\varphi - \frac{v_E}{N+h} & 0 \end{bmatrix}$$

- $\mathbf{f}^n$ is the specific force vector (obtained from IMU measurements) in n-frame
- $\delta\mathbf{f}^b$ is the noise in the accelerometer measurements which is modeled as white Gaussian noise.

8.4 Attitude Error Equation

The computed version from the INS mechanization output of Eq.(15) can be written as

$$\dot{\hat{\mathbf{C}}}_b^n = \hat{\mathbf{C}}_b^n(\hat{\boldsymbol{\Omega}}_{ib}^b - \hat{\boldsymbol{\Omega}}_{in}^b) \tag{29}$$

The attitude error dynamics equation can be rewritten as

$$\dot{\boldsymbol{\epsilon}}^n = \mathbf{F}_{er}\delta\mathbf{r}^n + \mathbf{F}_{ev}\delta\mathbf{v}^n - (\boldsymbol{\omega}_{in}^n\times)\boldsymbol{\epsilon}^n - \mathbf{C}_b^n\delta\boldsymbol{\omega}_{ib}^b \tag{30}$$

where

$$\mathbf{F}_{er} = \begin{bmatrix} -\omega_e sin\varphi & 0 & \frac{-v_E}{(N+h)^2} \\ 0 & 0 & \frac{v_N}{(M+h)^2} \\ -\omega_e cos\varphi - \frac{v_E}{(N+h)cos^2\varphi} & 0 & \frac{v_E tan\varphi}{(N+h)^2} \end{bmatrix}$$

$$\mathbf{F}_{ev} = \begin{bmatrix} 0 & \frac{1}{N+h} & 0 \\ -\frac{1}{M+h} & 0 & 0 \\ 0 & -\frac{tan\varphi}{N+h} & 0 \end{bmatrix}$$

- $\boldsymbol{\omega}_{in}^n$ is obtained from Eq.(9)
- $\delta\boldsymbol{\omega}_{ib}^b$ is the noise in the gyro measurements and is modeled as white Gaussian noise.

9 Continuous Error Model

The final, continuous, error model can be constructed using Eq.(26), Eq.(28) and Eq.(30) as follows

$$\dot{\mathbf{x}} = \mathbf{F}\mathbf{x} + \mathbf{G}\mathbf{u} \tag{32}$$

where $\mathbf{F}$ is the dynamic matrix, $\mathbf{x}$ is the state vector, $\mathbf{G}$ is the design matrix and $\mathbf{u}$ is the forcing vector function.

$$\mathbf{F} = \begin{bmatrix} \mathbf{F}_{rr} & \mathbf{F}_{rv} & 0 \\ \mathbf{F}_{vr} & \mathbf{F}_{vv} & (\mathbf{f}^n\times) \\ \mathbf{F}_{er} & \mathbf{F}_{ev} & -(\boldsymbol{\omega}^n_{in}\times) \end{bmatrix}$$
$$\mathbf{x} = \begin{bmatrix} \delta\mathbf{r}^n & \delta\mathbf{v}^n & \delta\boldsymbol{\epsilon}^n \end{bmatrix}$$
$$\mathbf{G} = \begin{bmatrix} 0 & 0 \\ \mathbf{C}^n_b & 0 \\ 0 & -\mathbf{C}^n_b \end{bmatrix}$$
$$\mathbf{u} = \begin{bmatrix} \delta\mathbf{f}^b \\ \delta\boldsymbol{\omega}^b_{ib} \end{bmatrix}$$

The specific force, $\mathbf{f}^n$, is the sensed output of the accelerometer transformed into the navigation frame as $\mathbf{C}^n_b\mathbf{f}^b$. The total angular velocity of the local-level navigation frame with respect to the inertial frame is given by Eq.(9).

10 INS/GPS Integration

A number of different INS/GPS integration schemes can be used. Few of those are [3]:

1. **Uncoupled system** in which GPS estimated position is used simply to reset the INS indicated position at regular intervals of time
2. **Loosely coupled system** in which the INS and GPS estimates of position and velocity are compared, the resulting differences forming the measurement inputs to a Kalman filter
3. **Tightly coupled system** in which the GPS measurements of pseudorange and rate are compared with estimates of these quantities generated by the internal system

10.1 Loosely Coupled Integration

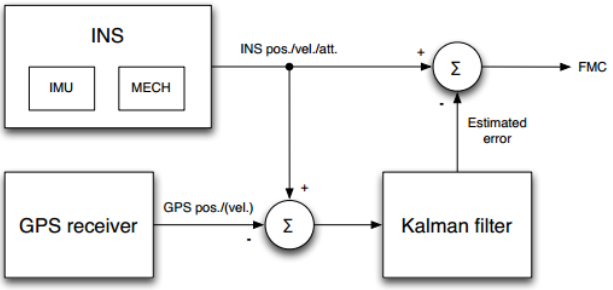
Two different approaches for this type of integration are feedforward and feedback methods as shown in 4.In this work, sensor model is assumed to be corrupted just with white Gaussian noise without any biases. In the feedforward method the estimated navigation errors are fed forward to the navigation output and subtracted. The main problem with the feedforward method is that the INS is not aware of the aiding and an unbounded error growth may occur.In the feedback method the errors are fed back to the INS mechanization to correct the equipment.

11 Indirect Kalman Filter Configuration [8]

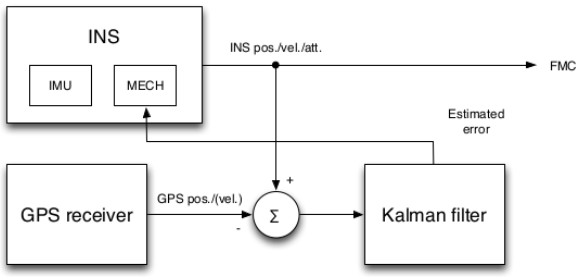
In the indirect integration approach, the set of variables linked to the error in the equations of motion is considered as the estimated variable in the Kalman filter.

11.1 Feedback Mode

In the feedback mode of the indirect integration, the corrected set of navigation variables replaces the previous value of the set of navigation variables. The set of error variables is then reset to zero for the next round of estimation. This configuration is the most robust one and is necessary when operating with low-cost sensors. This approach has been implemented in MATLAB.



(a) Feedforward method for Kalman filtering [3]. (FMC : Flight Management Computer



(b) Feedback method for Kalman filtering [3]. (FMC : Flight Management Computer)

Figure 4: Loosely coupled integration

11.2 Description of the fusion algorithm architecture

In this subsection a brief description of the fusion algorithm is presented. There are 2 multirate sensors and navigation filter operating with different frequencies:

- GPS : 8Hz
- IMU : 34 Hz
- Filter : 100 Hz

The hierarchical structure of the algorithm is presented in 5. The top level functions depicted in 6 are the equation of motion solutions and navigation filter. The navigation filter is composed of two main parts: the Kalman filter and the error control as shown in 7. The “error control” block manages the relation between the “extended Kalman filter” and the “integration of eq. of motion". The feedback mode implies the correction of the solution of the equations of motion and the reset of the estimated error variables. This is what "error control" function does. Here Kalman filter block has inputs from IMU and GPS (Kalman filter control) and the "correction" is measurement difference $z_k = x_k^{IMU} - x_k^{GPS}$. 8, 9 and 10 detail the Kalman filter operation and the explanation of the two modes (with or without external measurement) follows. Note that in 8 1,2,3,4 are inputs to the filter where

- 1 is from error control
- 2 is from Kalman filter control
- 3 is input form IMU/GPS (sensor readings)
- 4 is from integration of eq. of motion

The propagation of the estimates (state variables and covariance matrix) is done at each time step, whereas the event "correction" is driven by the reception of an external measurement. When there is no external measurement, the Kalman filter operates in the propagation mode. The “event correction” signal provided to the block “Correction of estimates” is FALSE. It drives a null output for the correction of the state variables and the covariance matrix. This correction (null for the propagation phase) is then sent to the “State vector propagation” and “Covariance matrix propagation” functions of 8. An inside look at the function “Covariance matrix propagation” is shown at 10. Since the correction is null, only the propagation of the covariance matrix is performed.

On the other hand, if there is an external signal available, the “event correction” signal is TRUE and the Kalman filter superposes the correction phase to the prediction phase. In fact, this event triggers the computation of a correction in the block “Correction of estimates” of 8, and detailed at 9. In this detailed figure, the difference between the external signal (GPS signal) and the INS solution is first computed, and the result is used in the “Computation of corrections to estimates” block. These corrections are sent to the “State vector propagation” and “Covariance matrix propagation” blocks of 8. It should be noted that a “system parameters” function updates the time varying parameters of the system model as depicted in 8

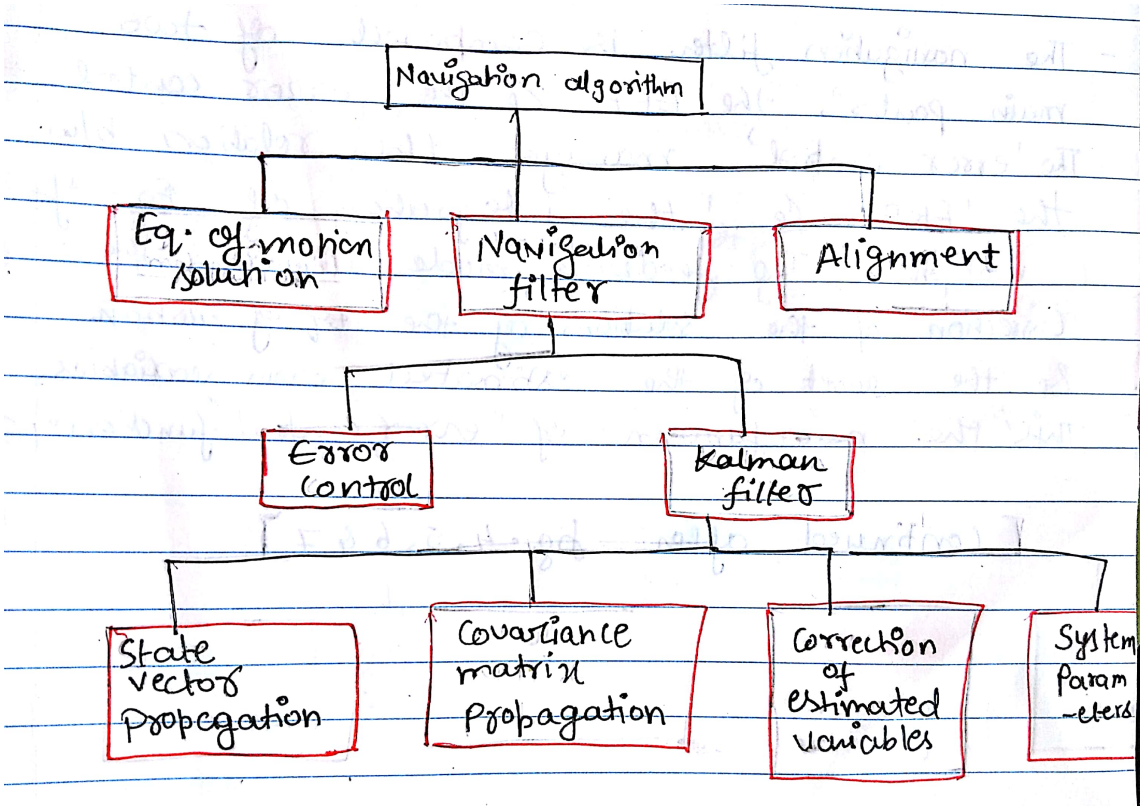


Figure 5: Hierarchical structure of the algorithm

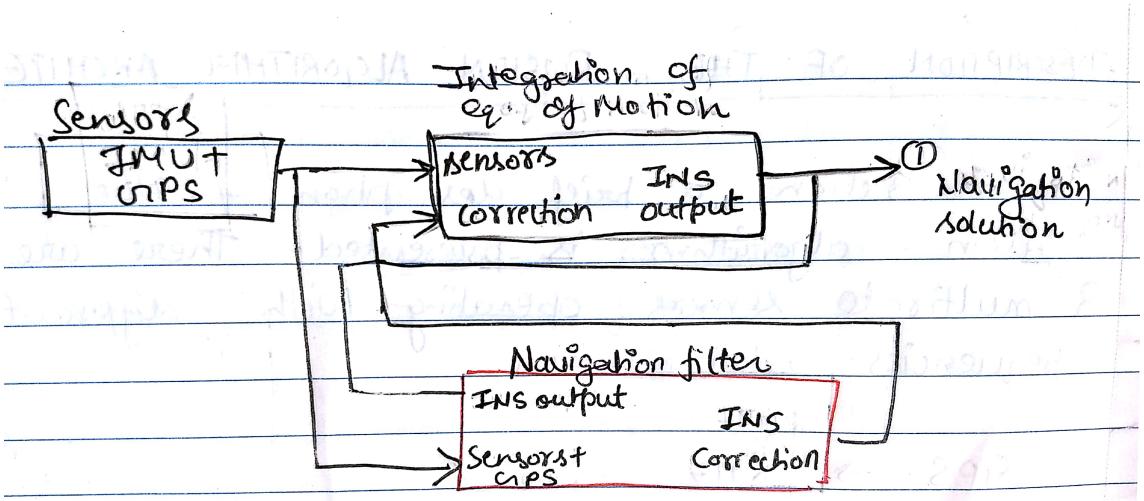


Figure 6: Top-level functions

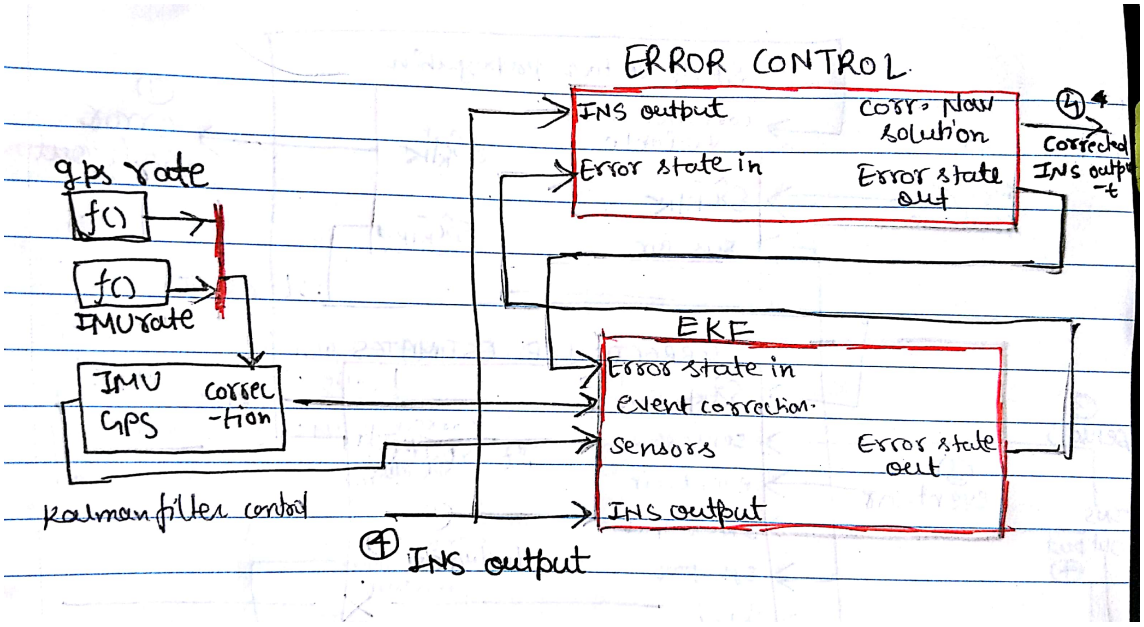


Figure 7: Navigation filter function

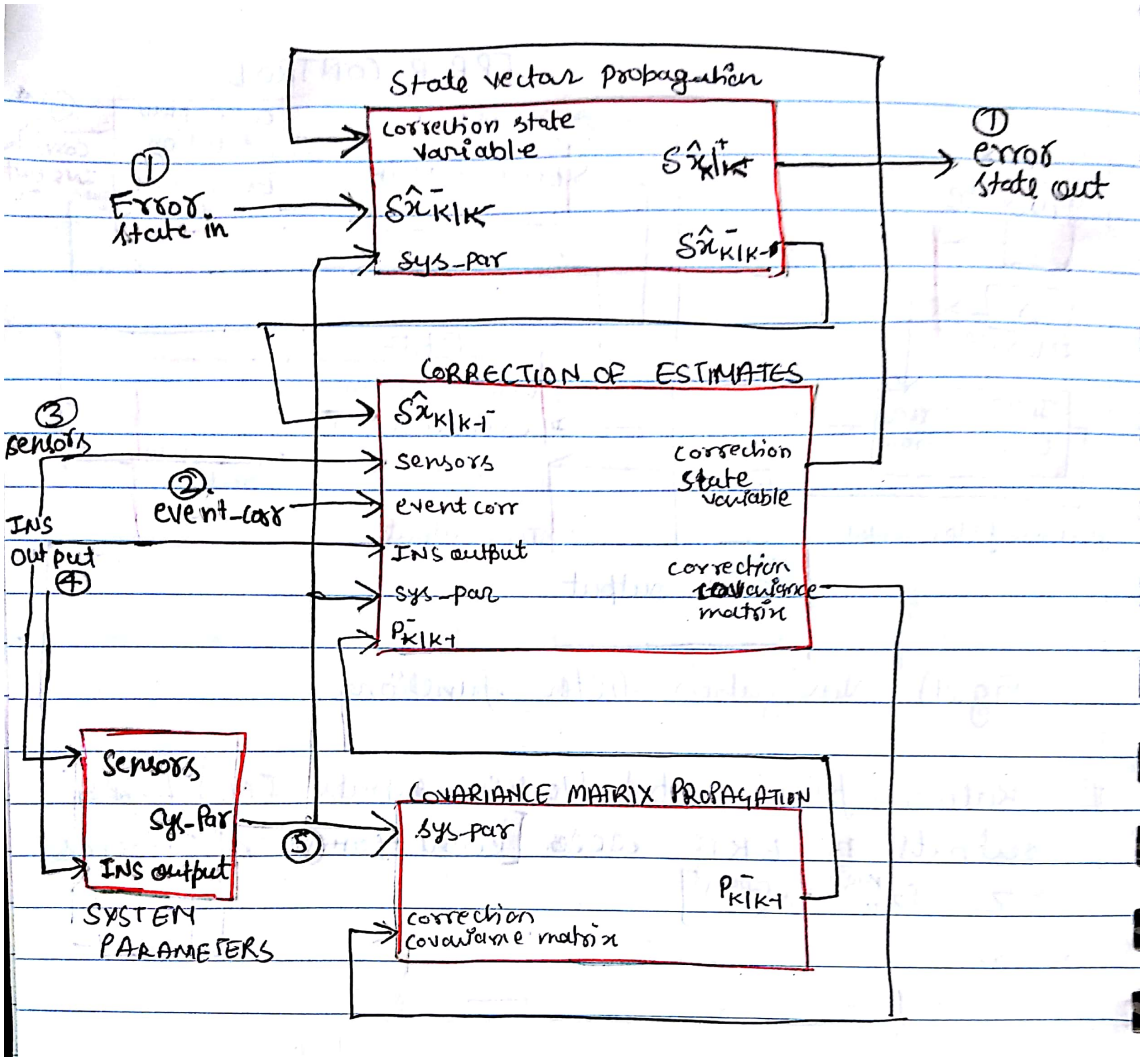


Figure 8: Kalman filter filter function

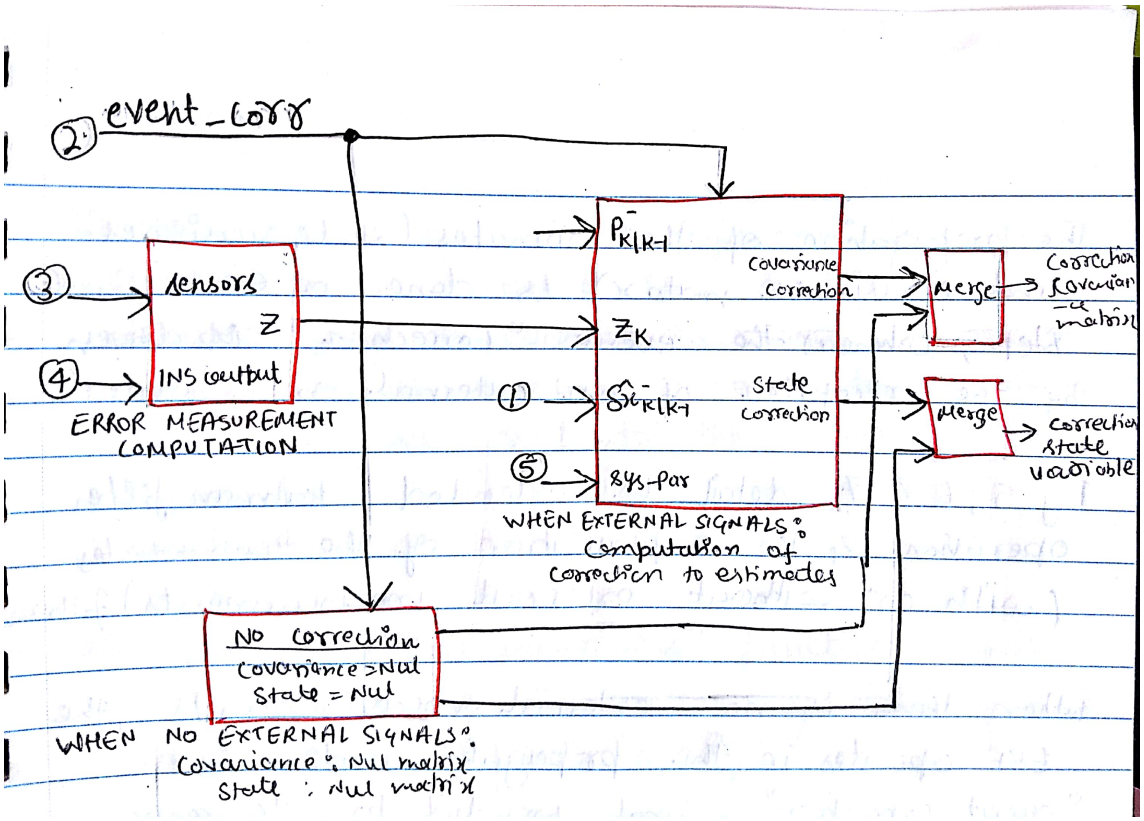


Figure 9: Correction of estimates function

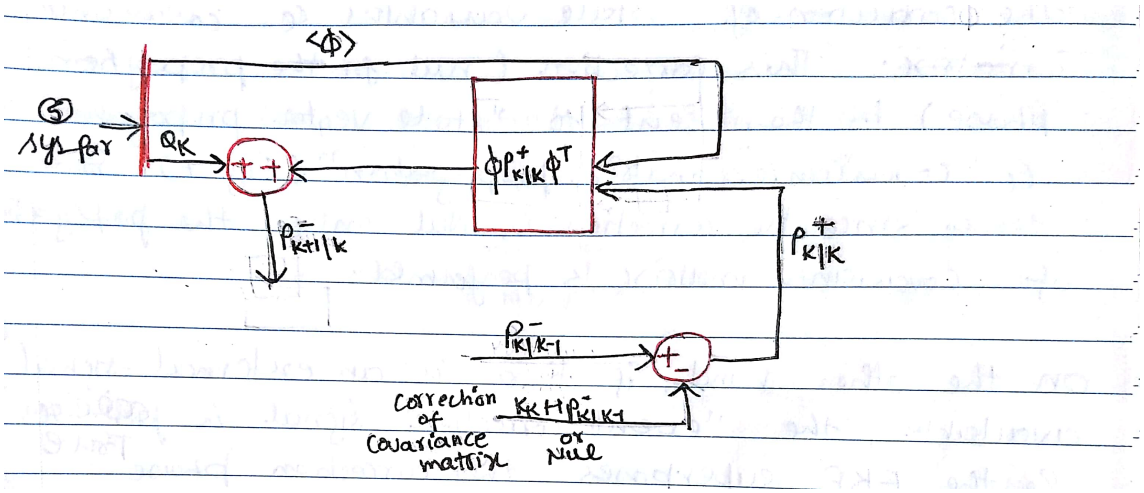


Figure 10: Propagation and correction of covariance matrix estimate

12 Initialization

At navigation system start-up the expected values of position, velocity and orientation as well as the corresponding error covariances are required to initialize the integrators of the inertial navigation equations and the navigation error filter. Initial values for position and velocity and the corresponding covariances are easily obtained from the GNSS receiver. Estimation of the initial orientation and $\mathbf{C}_b^n$ is expalined in the subsequent sections.

12.1 TRIAD : Orientation Angle

The Three-Axis Attitude Determination (TRIAD) method, which was originally presented by Black for satellites, will be used in the following for the in-flight orientation initialization. The roll angle, pitch angle and heading angle are estimated by means of two arbitrary vectors that are simultaneously observed in the n - and in the b-frame. Let these two vectors be given by

$$\mathbf{a}^n, \mathbf{a}^b$$

$$\mathbf{b}^n, \mathbf{b}^b$$

The vectors do not have to be orthogonal but must not be co-linear. Then, the transformation matrix $\mathbf{C}_b^n$ is given with

$$\mathbf{C}_b^n = \mathbf{A}\mathbf{B}^{-1}$$

where

$$\mathbf{A} = \begin{bmatrix} \mathbf{a}^n \\ \mathbf{b}^n \\ \mathbf{c}^n \end{bmatrix} \tag{33}$$

$$\mathbf{B} = \begin{bmatrix} \mathbf{a}^b \\ \mathbf{b}^b \\ \mathbf{c}^b \end{bmatrix} \tag{34}$$

$$\mathbf{c}^n = \mathbf{a}^n \times \mathbf{b}^n$$

$$\mathbf{c}^b = \mathbf{a}^b \times \mathbf{b}^b$$

The Euler angles ϕ_b^n, θ_b^n and ψ_b^n can be calculated from the entries of the transformation matrix $\mathbf{C}_b^n$.

12.1.1 Alignment method 1

The vectors are chosen to be integrated acceleration observations and magnetic fields with

- $\mathbf{a}^b$ = integrated acceleration observation obtained by integrating $\mathbf{f}^b$
- $\mathbf{a}^n$ = using velocity measurements from GPS which are in n-frame
- $\mathbf{b}^b$ = magnetometer readings which are in b-frame
- $\mathbf{b}^n$ = earth's magnetic field in n-frame

12.1.2 Alignment method 2

Assuming that the aircraft is in steady flight $\dot{\mathbf{v}}^n = 0$. Using Eq. 19 we have $\mathbf{f}^n = (2\boldsymbol{\Omega}_{ie}^n + \boldsymbol{\Omega}_{en}^n)\mathbf{v}^n - \mathbf{g}^n$. Then, the vectors are chosen to be acceleration and magnetic fields with

- $\mathbf{a}^b$ = accelerometer readings given by $\mathbf{f}^b$
- $\mathbf{a}^n$ = given by above eq. using gravity vector and $\mathbf{v}^n$ is obtained for GPS
- $\mathbf{b}^b$ = magnetometer readings which are in b-frame
- $\mathbf{b}^n$ = earth's magnetic field in n-frame

12.1.3 Alignment method 3

The vectors are chosen to be accelerations and integrated acceleration observations with

- $\mathbf{a}^b$ = accelerometer readings given by $\mathbf{f}^b$
- $\mathbf{a}^n$ = given by above eq. using gravity vector and $\mathbf{v}^n$ is obtained for GPS
- $\mathbf{b}^b$ = integrated acceleration observation obtained by integrating $\mathbf{f}^b$
- $\mathbf{b}^n$ = using velocity measurements from GPS which are in n-frame

This method eliminates the need of on-board magnetometer

12.2 Implementation

The idea behind alignment is to use the measurements to determine the orientation of the body frame with respect to a reference frame, here navigation frame. The alignment of an IMU is the determination of the initial DCM $\mathbf{C}_b^n$.

1. Initialization is carried out as explained in section 12. After initialization we have the knowledge of initial $\mathbf{C}_b^n$ and initial error state vector
2. With the continuous input of IMU measurements ($\mathbf{f}^b$ and $\boldsymbol{\omega}_{ib}^b$) Eq.(12), Eq.(13) and Eq.(15) are numerically integrated to obtain position $\mathbf{r}$, velocity $\mathbf{v}$ and DCM $\mathbf{C}_b^n$. Orientation angles are computed from $\mathbf{C}_b^n$ using Eq.(3).
3. A nine state Kalman filter is used for the navigation purpose which include nine navigation solution errors of three dimensional position, velocity and attitude. Filter is initialized by the output of the alignment/initialization process.
4. The measurement model fed to filter is the difference between IMU measurements (position and velocity) and GPS measurements (position and velocity).
5. Filter outputs error in the state (position, velocity and attitude). This error is subtracted from the INS obtained state giving us the corrected state to be passed on to the controller.
6. The corrected states are also used in the propagation equations of the filter which are given in the next section.

13 Documentation for the MATLAB Code

This section briefly explains the implementation of the self-alignment algorithm in MATLAB. Since the IMU unit is not available, a magnetometer-augmented IMU simulator is developed.

13.1 SIMULATOR (*simulator.m*)

Input to the simulator : Trajectory of the aircraft (traj) in CSV format and the discrete time step (dt). Trajectory comprises of latitude, longitude, altitude and Euler angles ($\phi, \theta \quad \& \quad \psi$) in n-frame for all the time steps during motion. Output of the simulator : acceleration $\mathbf{a}^b$, angular velocity $\boldsymbol{\omega}_{ib}^b$, magnetic field $\mathbf{m}^b$, velocity in n-frame $\mathbf{v}^\mathbf{n}$ (Velocity is given as output for modeling GPS) Following steps were implemented in order

1. $\mathbf{C}_b^n$ is calculated from input Euler angles and Eq.(2)
2. Smooth trajectory parameters are generated by fitting position and attitude sequence as piece-wise continuous cubic spline
3. $\dot{\mathbf{r}}^n$, $\dot{\mathbf{v}}^n$ and $\begin{bmatrix}\dot{\phi} & \dot{\theta} & \dot{\psi}\end{bmatrix}^T$ are computed using **fnder()** function of MATLAB
4. Then using Eq.(11) $\mathbf{v}^n$ is computed
5. $\boldsymbol{\omega}_{ie}^n$ and $\boldsymbol{\omega}_{en}^n$ is then computed using Eq.(6) and Eq.(8) respectively.

6. $\boldsymbol{\omega}_{nb}^{b}$ is given by

$$\boldsymbol{\omega}_{nb}^{b} = \begin{bmatrix} 1 & 0 & -sin\theta \\ 0 & cos\phi & sin\phi cos\theta \\ 0 & -sin\phi & cos\phi cos\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{35}$$

7. Finally $\mathbf{a}^b$ and $\boldsymbol{\omega}_{ib}^{b}$ are computed using Eq.(19) and Eq.(21)
8. It is assumed that no external magnetic field is present near the sensors except the earth's magnetic field. Then $\mathbf{m}^n$ is given by using MATLAB's inbuilt function *wrldmagn*.
9. Magnetometer output is given by coordinate transformation of $\mathbf{m}^n$, i.e. $\mathbf{m}^b = \mathbf{C}_n^b \mathbf{m}^n$

13.1.1 Function files used by SIMULATOR

- *omegas.m* : returns $\boldsymbol{\omega}_{ie}^{n}$ and $\boldsymbol{\omega}_{en}^{n}$ given φ, h, $\mathbf{v}^n$, $\dot{\lambda}$ and $\dot{\varphi}$
- *C_ nb.m* : returns DCM $\mathbf{C}_n^b$ for given Euler angles
- *gravity.m* : returns gravity vector, $\mathbf{g}^n$, for a given altitude and latitude
- *acc.m* : returns $\mathbf{a}^b$ given $\mathbf{v}^n$, $\dot{\mathbf{v}}^n$, $\boldsymbol{\omega}_{ie}^{n}$, $\boldsymbol{\omega}_{en}^{n}$, $\mathbf{C}_n^b$, $\mathbf{g}^n$
- *gyro.m* : returns $\boldsymbol{\omega}_{ib}^{b}$ given $\mathbf{C}_n^b$, $\boldsymbol{\omega}_{ie}^{n}$, $\boldsymbol{\omega}_{en}^{n}$ and $\boldsymbol{\omega}_{nb}^{b}$
- *crossm.m* : for a vector $\mathbf{v}$ returns $(\mathbf{v}\times)$

13.2 Main Code (*main.m*)

13.2.1 Measurement Model

GPS readings are used as measurements for the navigation filter. For now, the GPS readings ($[\mathbf{r}_{gps}^{n} \quad \mathbf{v}_{gps}^{n}]^T$) are simply given by corresponding true values corrupted by white Gaussian noise with the standard deviation $\sigma_{r_{gps}}$ and $\sigma_{v_{gps}}$ and zero mean.

$$\mathbf{r}_{gps}^{n} = \mathbf{r}_{true}^{n} + mvnrnd(0, \sigma_{r_{gps}}^2 \mathbf{I}_3) \tag{36a}$$

$$\mathbf{v}_{gps}^{n} = \mathbf{v}_{true}^{n}, + mvnrnd(0, \sigma_{v_{gps}}^2 \mathbf{I}_3) \tag{36b}$$

$\mathbf{r}_{true}$ is directly known from the trajectory file while $\mathbf{v}_{true}$ is given by the simulator (using forward difference scheme). *mvnrnd* is a MATLAB inbuilt function that returns an n-by-d matrix R of random vectors chosen from the multivariate normal distribution with mean MU, and covariance SIGMA, usage : $R = mvnrnd(MU, SIGMA)$

13.2.2 Alignment (*alignment_ 1.m*)

Initialization time (t_{ini}) and time step dt are given by the user as an input. Two vectors needed for TRIAD algorithm are integrated acceleration observation vectors *a_ b0*, *a_ n0* and magnetic field vectors *b_ b0*, *b_ n0*. The change of the DCM $\mathbf{C}_b^n$ with time due to orientation changes of the aircraft in the initialization phase is expressed on the one hand by the change of the n -frame with time, $\mathbf{C}_{n_0}^{n}$, and on the other hand by the change of the b-frame with time, $\mathbf{C}_{b}^{b_0}$, starting from the initial DCM $\mathbf{C}_{b_0}^{n_0}$. Full derivation can be found in [1]. Final equations used in the code are given as follows :

$$\mathbf{a}_{b_0} = \int_{t_0}^{t} \mathbf{C}_b^{b_0} \mathbf{f}^b dt \tag{37a}$$

$$\mathbf{C}_b^{b_0} = quat2rotm(q_{bb_0}^T) \tag{37b}$$

$$\dot{q}_{bb_0} = \frac{1}{2}(q_{bb_0} \breve{\omega}_{ib} - \breve{\omega}_{ib}(t_0) q_{bb_0}) \tag{37c}$$

$$\mathbf{a}^{n_0} = \mathbf{v}^n(t) - \mathbf{v}^{n_0}(t_0) - \mathbf{g}^n(t - t_0) \tag{37d}$$

where *quat2rotm* is a MATLAB function that converts a given quaternion to Rotation matrix, $\breve{\omega}_{ib} = [0; \omega_{ib}]$. Fourth order Runge Kutta is used to integrate Eq.(37a) and Eq.(37c). Final vectors *a_ n0*, *a_ b0* and magnetic field vectors *m_ b0*, *m_ n0* are computed at the final time step of the

initialization period and passed on to compute $\mathbf{C}_n^b(t_0)$ using TRIAD as explained in Eq.(33) and Eq.(34). Instead of using quaternion, DCM dynamics can be invoked for Eq.(37b) and Eq.(37c)

$$\dot{\mathbf{C}}_b^{b_0} = -\mathbf{\Omega}_{bb_0}^{b_0}\mathbf{C}_b^{b_0}$$

This equation is integrated using fourth order Runge Kutta to get $\mathbf{C}_b^{b_0}$ with initial value as $\mathbf{I}$.

Initialization of orientation error & error covariance matrix

In order to compute orientation error the error vectors of $\mathbf{a}^{b_0}$ and $\mathbf{a}^{n_0}$, $\delta\mathbf{a}^{b_0}$ and $\delta\mathbf{a}^{n_0}$ are required. As before, full derivations of error equations can be found in [1] and only final form of the equations are stated here.

$$\delta\mathbf{a}^{n_0} = \delta\mathbf{v}^n - \delta\mathbf{v}^n(t_0) \tag{38}$$

The discrete-time version of the state-space model with sample time dt for computing $\delta\mathbf{a}^{b_0}$ is given by

$$\begin{bmatrix}\delta\mathbf{a}_k^{b_0}\\ \boldsymbol{\epsilon}_{b_0\tilde{b}_0,k}\end{bmatrix} = \begin{bmatrix}\mathbf{I}_3 & -\mathbf{\Omega}(\mathbf{C}_{b,k-1}^{\tilde{b}_0}\tilde{\mathbf{a}}_{k-1}^b)dt\\ \mathbf{0}_3 & \mathbf{I}_3 + \tilde{\mathbf{\Omega}}_{ib,0}dt\end{bmatrix}\begin{bmatrix}\delta\mathbf{a}_{k-1}^{b_0}\\ \boldsymbol{\epsilon}_{b_0\tilde{b}_0,k-1}\end{bmatrix} + \begin{bmatrix}\mathbf{C}_{b,k-1}^{\tilde{b}_0}dt & \mathbf{0}_3\\ \mathbf{0}_3 & \mathbf{C}_{b,k-1}^{\tilde{b}_0}dt\end{bmatrix}\begin{bmatrix}\delta\mathbf{f}_{k-1}^b\\ \delta\boldsymbol{\omega}_{k-1}^b\end{bmatrix} - \begin{bmatrix}\mathbf{0}_3\\ \delta\boldsymbol{\omega}_{ib,0}dt\end{bmatrix} \tag{39}$$

where the variables with tilde represents error in measurement and $\mathbf{C}_{b_0}^{\tilde{b_0}}$ is the small angle rotation matrix and

$$\begin{bmatrix}\delta\mathbf{a}_0^{b_0}\\ \boldsymbol{\epsilon}_{b_0\tilde{b}_0,0}\end{bmatrix} = zeros(6,1) \tag{40}$$

Using Eq.(39) error vectors are calculated for each time step of the initialization period. Error vectors for magnetic field are given by:

$$\delta\mathbf{b}^{b_0} = -(\mathbf{C}_b^{\tilde{b}_0}\tilde{\mathbf{m}}^b\times)\boldsymbol{\epsilon}_{b_0\tilde{b}_0} + \mathbf{C}_b^{\tilde{b}_0}\delta\mathbf{m}^b \tag{41}$$

$$\delta\mathbf{b}^{n_0} = \delta\mathbf{m}^n \tag{42}$$

The perturbations in magnetic field and velocity are assumed to be white Gaussian and uncorrelated. Error in accelerometer and gyro measurements is modeled as white Gaussian noise. The initial orientation error and error covariance matrix for attitude is then a function of these error vectors.(Eq. 4.72-4.74 in [1]).

13.2.3 Alignment (*alignment_2.m*)

Two vectors (**a** and **b**) needed for TRIAD algorithms are accelerometer output, $\mathbf{f}^b$ and $\mathbf{f}^n$, and magnetic field. Here

$$\mathbf{f}^n = (2\mathbf{\Omega}_{ie}^n + \mathbf{\Omega}_{en}^n)\mathbf{v}^n - \mathbf{g}^n$$

Initialization of orientation error

Error vectors for acceleration is given as follows

$$\delta\mathbf{a}^{b_0} = -(\mathbf{C}_b^{\tilde{b}_0}\tilde{\mathbf{f}}^b\times)\boldsymbol{\epsilon}_{b_0\tilde{b}_0} + \mathbf{C}_b^{\tilde{b}_0}\delta\mathbf{f}^b \tag{43}$$

$$\delta\mathbf{a}^{n_0} = (2\delta\tilde{\mathbf{\Omega}}_{ie}^n + \delta\tilde{\mathbf{\Omega}}_{en}^n)\mathbf{v}^n + (2\tilde{\mathbf{\Omega}}_{ie}^n + \tilde{\mathbf{\Omega}}_{en}^n)\delta\mathbf{v}^n - \delta\mathbf{g}^n \tag{44}$$

13.2.4 Alignment (*alignment_ 3.m*)

Two vectors (**a** and **b**) needed for TRIAD algorithms are accelerometer output, $\mathbf{f}^b$ and $\mathbf{f}^n$, and integrated acceleration observation. Here

$$\begin{aligned}
\mathbf{a}^{b_0} &= \mathbf{C}_b^{b_0}\mathbf{f}^b \\
\mathbf{a}^{n_0} &= (2\mathbf{\Omega}_{ie}^n + \mathbf{\Omega}_{en}^n)\mathbf{v}^n - \mathbf{g}^n \\
\mathbf{b}_{b_0} &= \int_{t_0}^{t} \mathbf{C}_b^{b_0}\mathbf{f}^b dt \\
\dot{\mathbf{C}}_b^{b_0} &= -\mathbf{\Omega}_{bb_0}^{b_0}\mathbf{C}_b^{b_0} \\
\mathbf{b}^{n_0} &= \mathbf{v}^n(t) - \mathbf{v}^{n_0}(t_0) - \mathbf{g}^n(t - t_0)
\end{aligned}$$

Initialization of orientation error

Error vectors for acceleration is given as follows

$$\delta\mathbf{a}^{b_0} = -(\mathbf{C}_b^{\tilde{b}_0}\tilde{\mathbf{f}}^b\times)\boldsymbol{\epsilon}_{b_0\tilde{b}_0} + \mathbf{C}_b^{\tilde{b}_0}\delta\mathbf{f}^b \tag{45}$$

$$\delta\mathbf{a}^{n_0} = (2\delta\tilde{\mathbf{\Omega}}_{ie}^n + \delta\tilde{\mathbf{\Omega}}_{en}^n)\mathbf{v}^n + (2\tilde{\mathbf{\Omega}}_{ie}^n + \tilde{\mathbf{\Omega}}_{en}^n)\delta\mathbf{v}^n - \delta\mathbf{g}^n \tag{46}$$

$\delta\mathbf{b}^{n_0}$ and $\delta\mathbf{b}^{b_0}$ are given by Eq. 38 and Eq. 39 respectively

13.2.5 IMU Outputs

The outputs of the IMU (accelerometer and gyroscope) after the alignment are stored in *fb* and *gb* matrices. Fourth order Runge-Kutta method is used to integrate Eq.(12), Eq.(13) and Eq.(15).
Vertical Channel (altitude, *h*)
Relying only on an inertial measurement for calculation of the vertical channel will render it exponentially unstable (Rogers, 2007). The reason for the unstable behavior is that h is dependent on $\mathbf{g}$. As h increases, $\mathbf{g}$ decreases, creating upward acceleration. The vertical channel must be aided with an external measurement to keep it within an acceptable value, for e.g. a barometer as an aid. For this reason altitude values obtained from GPS are directly used in the integration of IMU measurements.

So this segment of the code gives position, velocity and DCM $\mathbf{C}_b^n$ (equivalently Euler angles) as computed by IMU/INS. *IMU is used to refer to sensor measurements (*$\mathbf{f}^b$ *and* $\boldsymbol{\omega}_{ib}^b$*) and INS(Inertial Navigation System) are the navigation equations that make use of IMU measurements to give position, velocity and attitude*

13.2.6 Extended Kalman Filter

A nine state Kalman filter is used for the alignment and navigation purpose which includes nine navigation solution errors of three dimensional position, velocity and attitude :

$$\dot{\mathbf{x}} = \mathbf{F}\mathbf{x} + \mathbf{G}\mathbf{u} \tag{47}$$

$\delta\mathbf{f}^b$ and $\delta\boldsymbol{\omega}^b$ are modeled as white Gaussian noise with standard deviation given by σ_{acc} and σ_{gyro}. The continuous system is discretized as :

$$\mathbf{x}_k = \mathbf{\Phi}_{k-1}\mathbf{x}_{k-1} + \mathbf{W}_{k-1} \tag{48}$$

The solution for discretized matrix $\mathbf{\Phi}_{k-1}$ and covariance matrix associated with $\mathbf{W}$, $\mathbf{Q}_k$, is given by VanLoan method. The velocity and position errors are taken as observations for the filter. It can be obtained from the velocity errors between the INS and the GPS (Eq. 36). It is expressed as follows:

$$\mathbf{z} = \begin{bmatrix} \mathbf{r}_{INS}^n - \mathbf{r}_{GPS}^n \\ \mathbf{v}_{INS}^n - \mathbf{v}_{GPS}^n \end{bmatrix} \tag{49}$$

Therefore, the measurement model is liner and can be written as:

$$\mathbf{z}_k = \mathbf{H}_k\mathbf{x}_k + \mathbf{V}_k \tag{50}$$

where $\mathbf{V}_k$ is the white noise with zero mean and covariance R_k. and

$$\mathbf{H}_k = \begin{bmatrix} \mathbf{I} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & \mathbf{I} & 0_{3\times3} \end{bmatrix}$$

13.2.7 Implementation of the Kalman Filter

Notations as used in the code:

- P0 ≡ Error covariance matrix of size 9×9
- K ≡ Kalman gain
- x_p ≡ state after update at time t_k
- r ≡ position vector in n-frame
- vel ≡ velocity vector in n-frame
- Q_sdm ≡ Process noise covariance matrix associated with **u**
- R ≡ Covariance matrix associated with measurement
- err ≡ Error in the state
- att_err0 ≡ Initial attitude error obtained from Initialization algorithm
- P0_att ≡ Initial attitude error covariance matrix obtained from Initialization algorithm
- σ ≡ standard deviation
- o_ibb ≡ $\boldsymbol{\omega}$ of b-frame wrt i-frame projected onto b-frame

Computation at time t_k
Initial state i.e. $x_{m,0} = [zeros(6,1); att\_err0]$
Kalman gain and measurements

$$K = P0 * H.' * (H * P0 * H.' + R) \tag{51}$$
$$z = [r_{imu} - r_{gps,k} \quad v_{imu} - v_{gps,k}] \tag{52}$$

Update Step, if measurements are available

$$x_{p,k} = x_{m,k} + K * (z - H * x_{m,k}) \tag{53}$$
$$P0 = (\mathbf{I}_9 - K * H) * P0 * (\mathbf{I}_9 - K * H).' + K * R * K.' \tag{54}$$
$$err_k = x_{p,k} \tag{55}$$
$$r_k = r_{imu,k} - x_{p,k}(1:3) \tag{56}$$
$$vel_k = v_{imu,k} - x_{p,k}(4:6) \tag{57}$$
$$att_k = att_{imu,k} - x_{p,k}(7:9) \tag{58}$$

Propagation

$$[Phi, Q] = VanLoan((F, dt, Q_s dm, G) \tag{59}$$
$$x_{m,k+1} = Phi * x_{p,k} \tag{60}$$
$$P0 = Phi * P0 * Phi.' + Q \tag{61}$$

where VanLoan() is a MATLAB function for obtaining discrete state transition matrix and covariance matrix. The propagation of the true states is then done by using Eq. 12, 13 and 15 using updated value at that time step (Eq. 56-58)

When only IMU measurements are available only propagation takes place and true states are corrected as given by INS at that time step. When neither IMU nor GPS values are available on propagation takes place with no correction of true states.

13.2.8 Tuning Parameters

$\sigma_{r_{gps}}$	[1e-3;1e-3;3] (rad;rad;m)
$\sigma_{v_{gps}}$	[1e-2;1e-2;1e-2] (m/s)
t_ini	15 s
dt	.1s
σ_{acc}	[15;55;48] (m/s^2)
σ_{gyro}	[3;.3;3]deg/h
σ_{mn}	5e-11 Tesla
σ_{mb}	4e-11 Tesla
σ_{vel_n}	[1e-3;9e-3;7e-3] (m/s)
σ_{r_n}	[1e-5;1e-5;3e-2] (rad;rad;m)

Subscript *mn*, *mb*, *acc*, *gyro*, *vel_n* denote magnetic field in n-frame, magnetic field in b-frame, accelerometer measurements, gyroscope measurements, velocity in n-frame. The standard deviation for the velocity and position in n-frame is used during the alignment, Eq.(38), for obtaining error in acceleration in n-frame [1].
The Q and R matrix for Kalman filter are given as follows

$$Q = \begin{bmatrix} 225 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3025 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2304 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{62}$$

$$R = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 9 & 0 & 0 & 0 \\ 0 & 0 & 0 & .0001 & 0 & 0 \\ 0 & 0 & 0 & 0 & .0001 & 0 \\ 0 & 0 & 0 & 0 & 0 & .0001 \end{bmatrix} \tag{63}$$

14 Simulation results

Sensor data from a commercial small Uninhabited Aerial Vehicle flown over an agriculture field on the morning of October 22, 2014 have been used here as documented in [9]. The sampling rate of the IMU is 34 Hz, GPS is 8Hz and the filter is run at 100 Hz.

14.1 Simulator plots

11, 12, 13 and 14 are the plots as obtained by IMU simulator vs true values

14.2 Alignment method 1

16, 17 and 18 shows the results using alignment method 1. 15 shows the evolution of close loop (using Kalman filter) and open loop (only INS solution) eigenvalues of the system. As can be seen Kalman filter stabilizes the system.

14.3 Alignment method 2

20, 21 and 22 shows the results using alignment method 1. 19 shows the evolution of close loop (using Kalman filter) and open loop (only INS solution) eigenvalues of the system. As can be seen Kalman filter stabilizes the system.

14.4 Alignment method 3

24, 25 and 26 shows the results using alignment method 1. 23 shows the evolution of close loop (using Kalman filter) and open loop (only INS solution) eigenvalues of the system. As can be seen Kalman filter stabilizes the system.

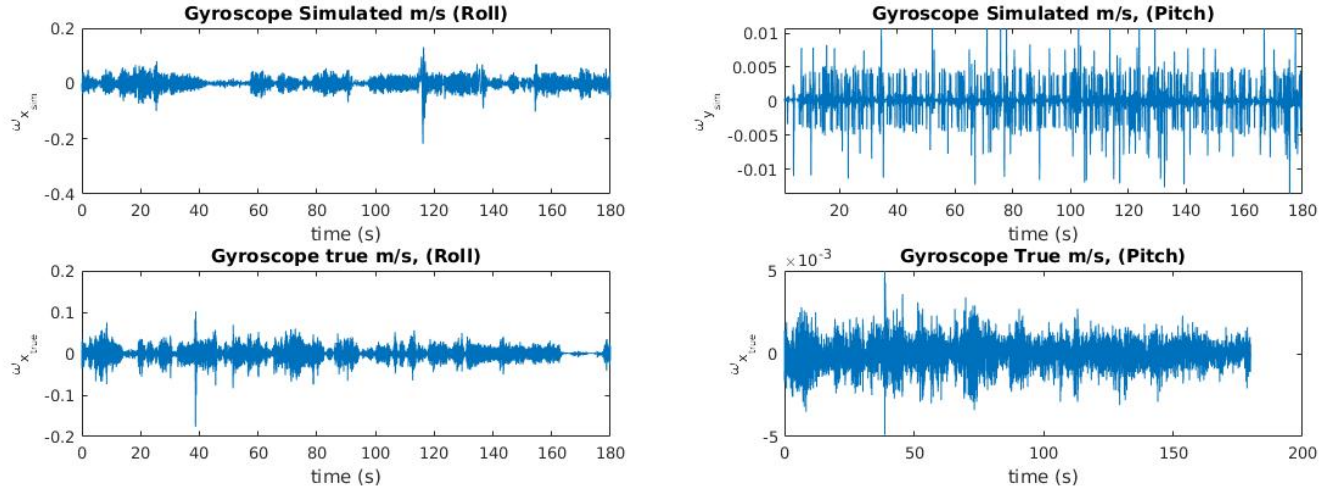


Figure 11: Gyroscope simulation, *m/s stands for measurements*

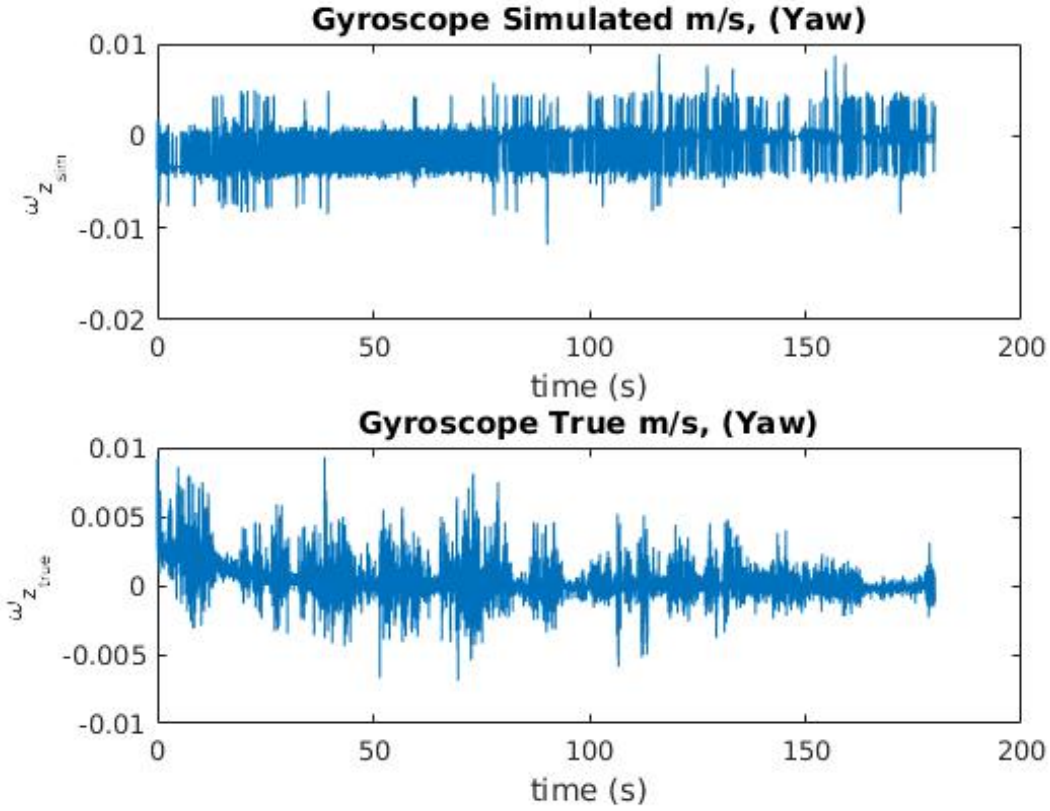


Figure 12: Gyroscope simulation for yaw axis

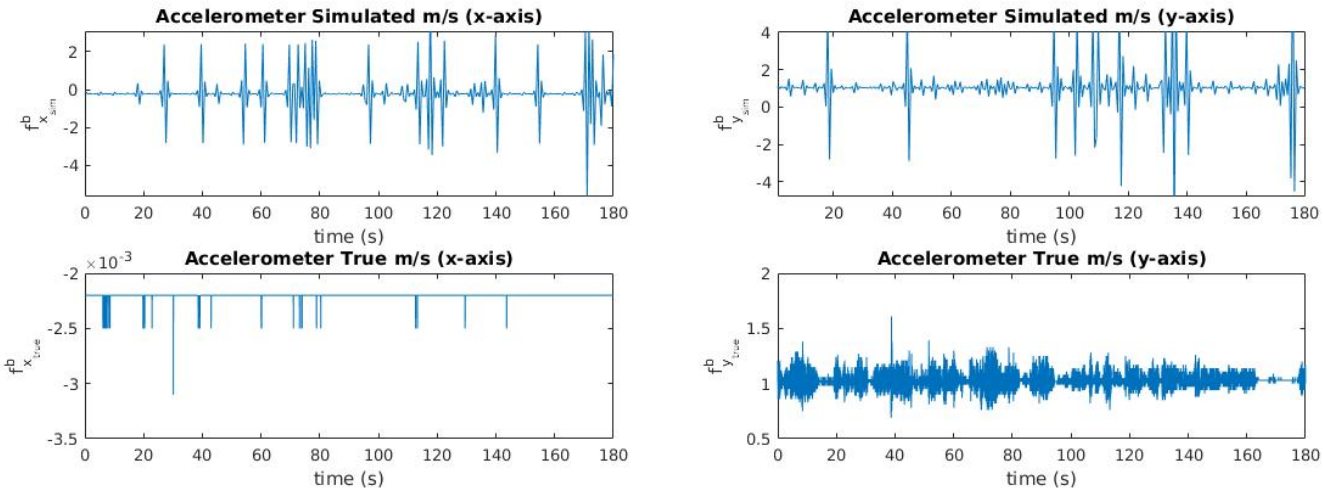


Figure 13: Accelerometer simulation

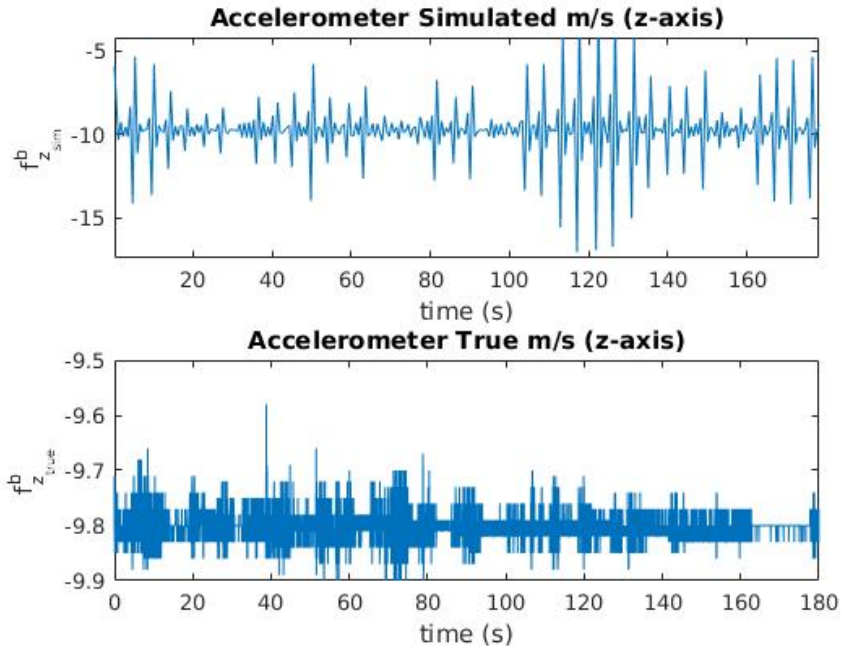


Figure 14: Accelerometer simulation for yaw axis

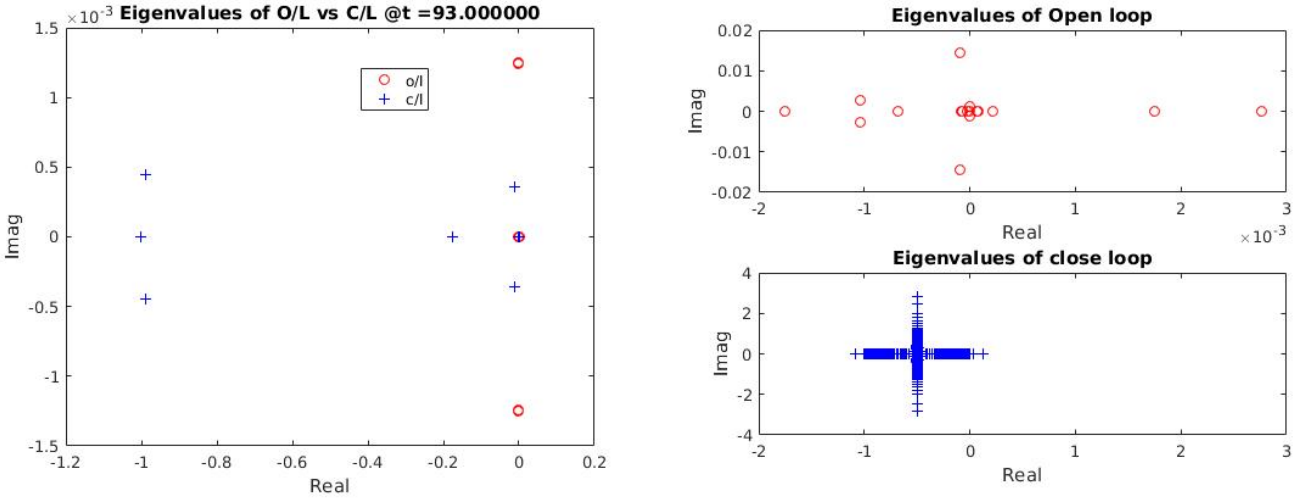


Figure 15: Alignment Method 1 : Eigenvalues

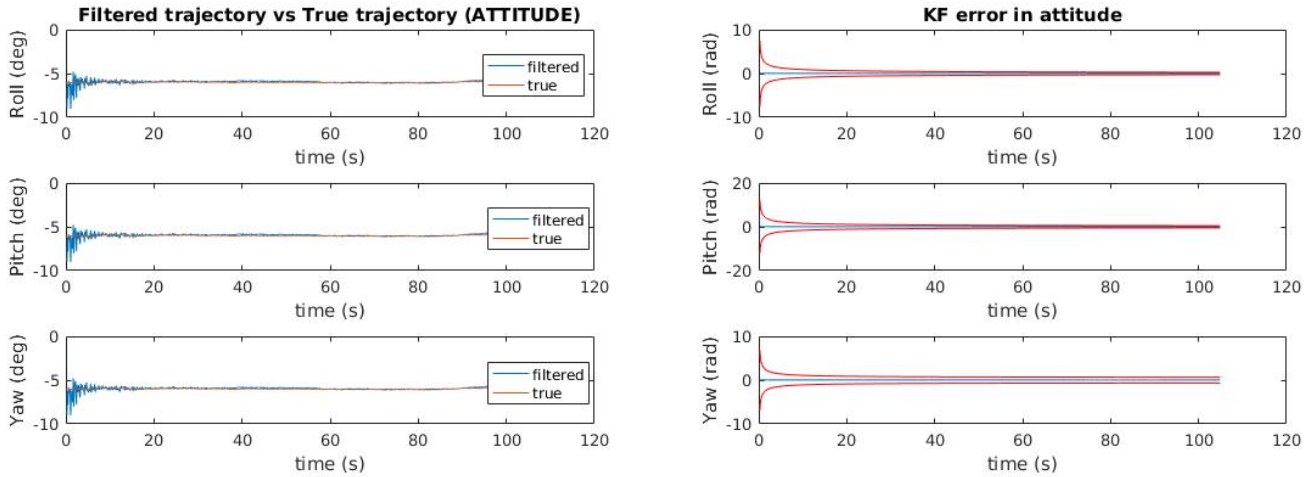


Figure 16: Alignment Method 1 : Attitude

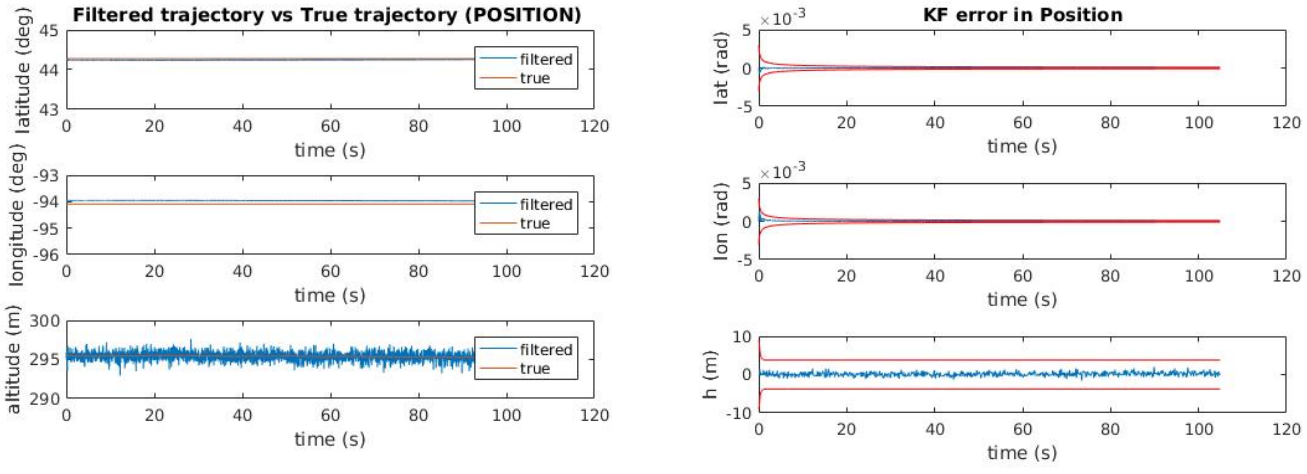


Figure 17: Alignment Method 1 : Position

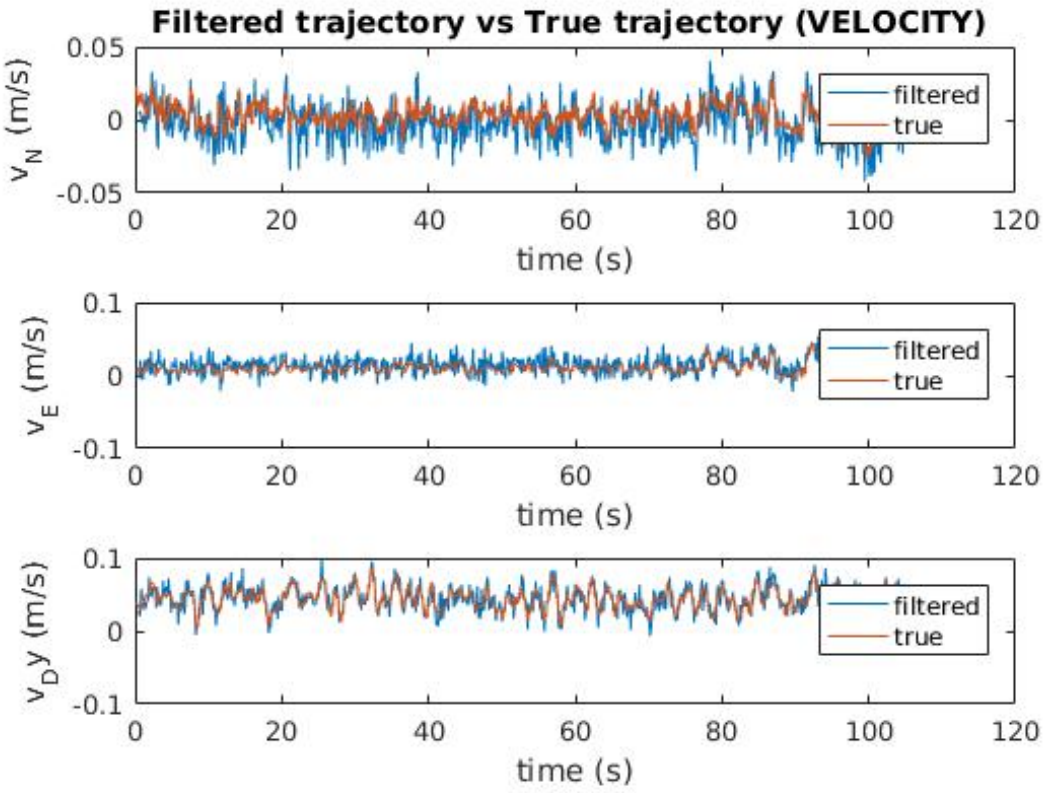


Figure 18: Alignment Method 1 : Velocity

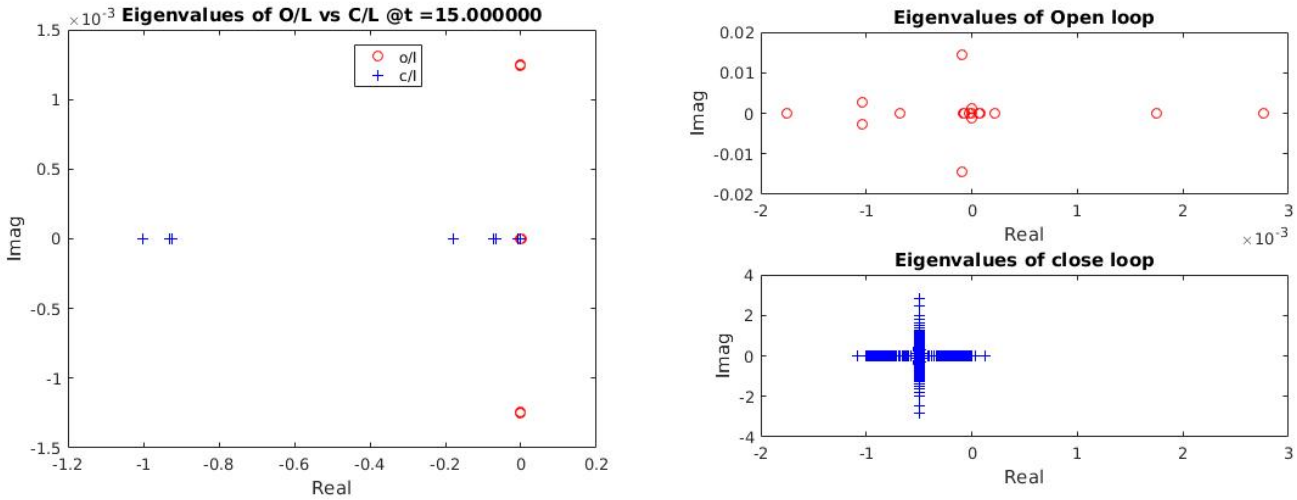


Figure 19: Alignment Method 2 : Eigenvalues

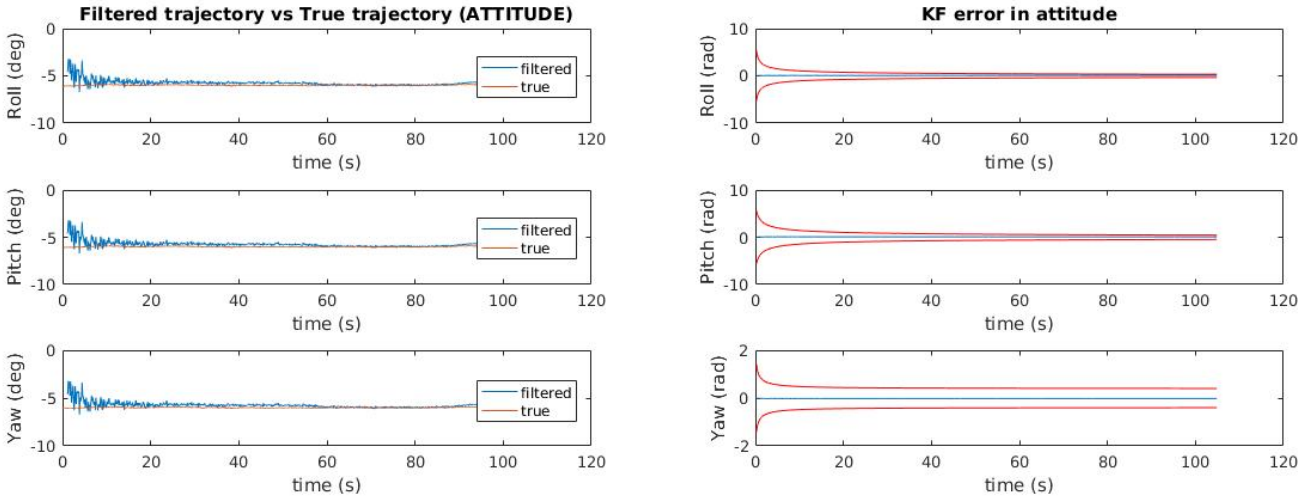


Figure 20: Alignment Method 2 : Attitude

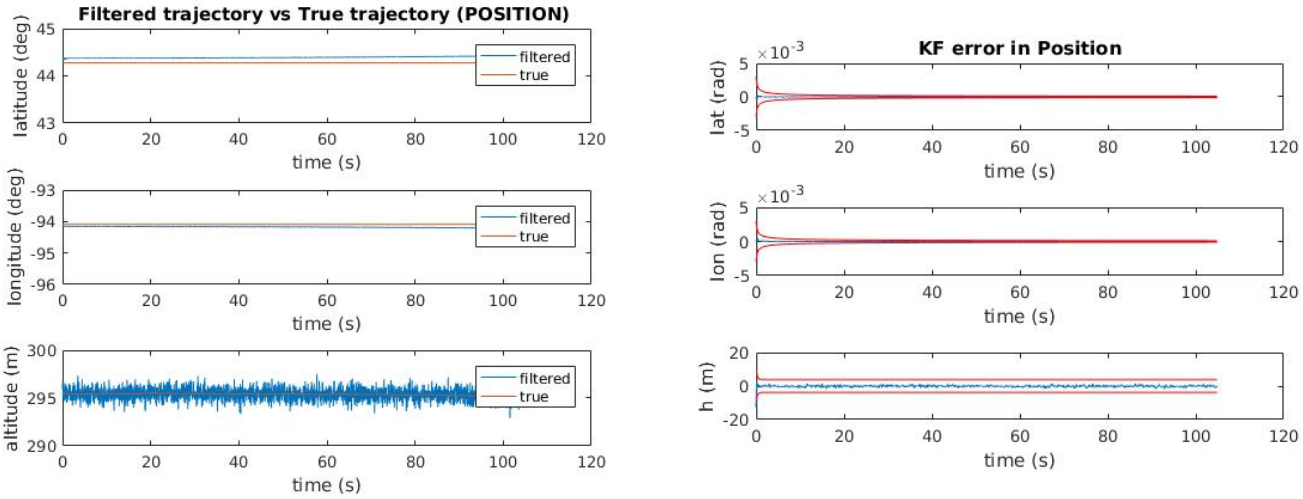


Figure 21: Alignment Method 2 : Position

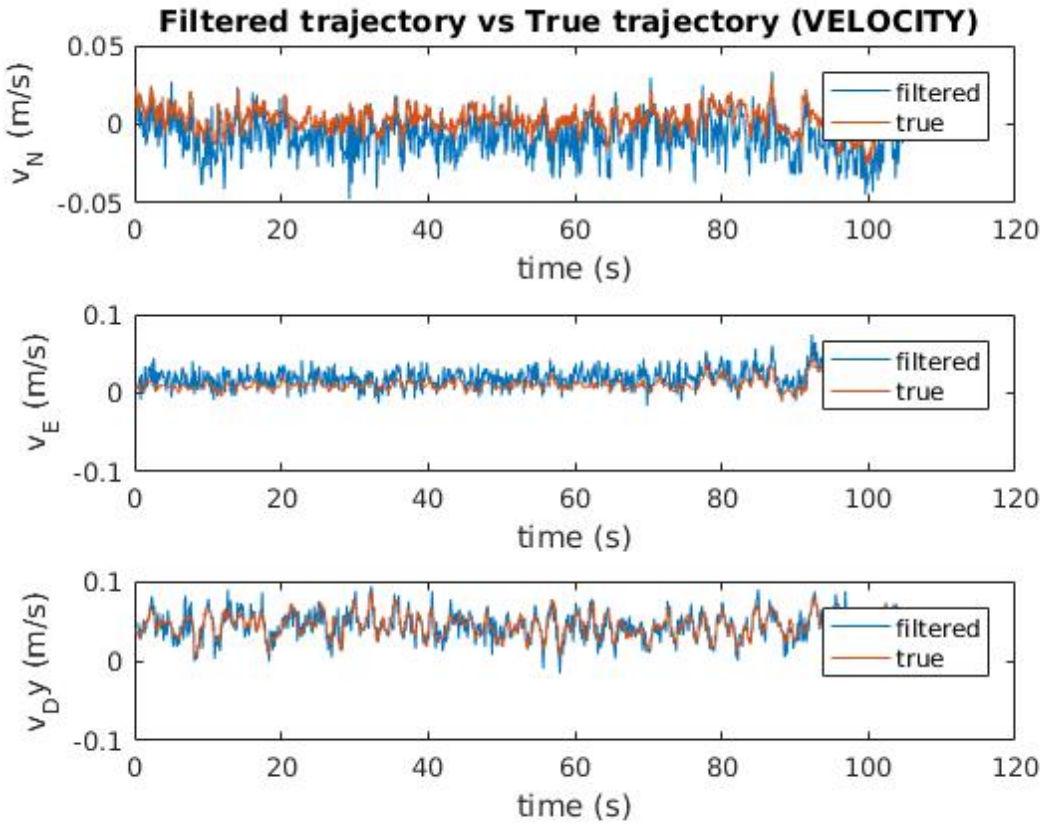


Figure 22: Alignment Method 2 : Velocity

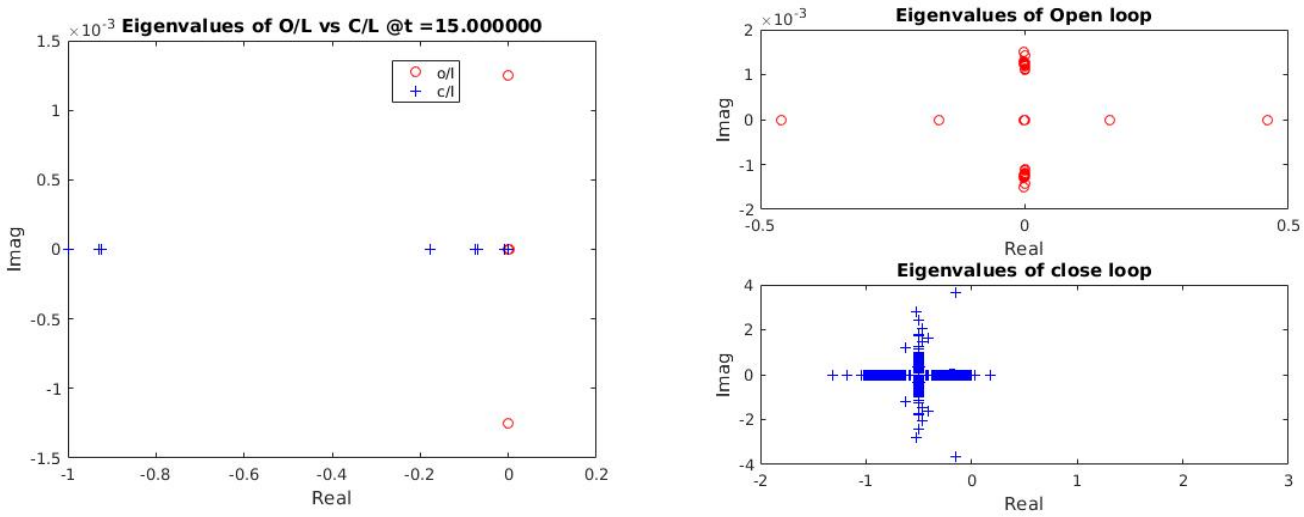


Figure 23: Alignment Method 3 : Eigenvalues

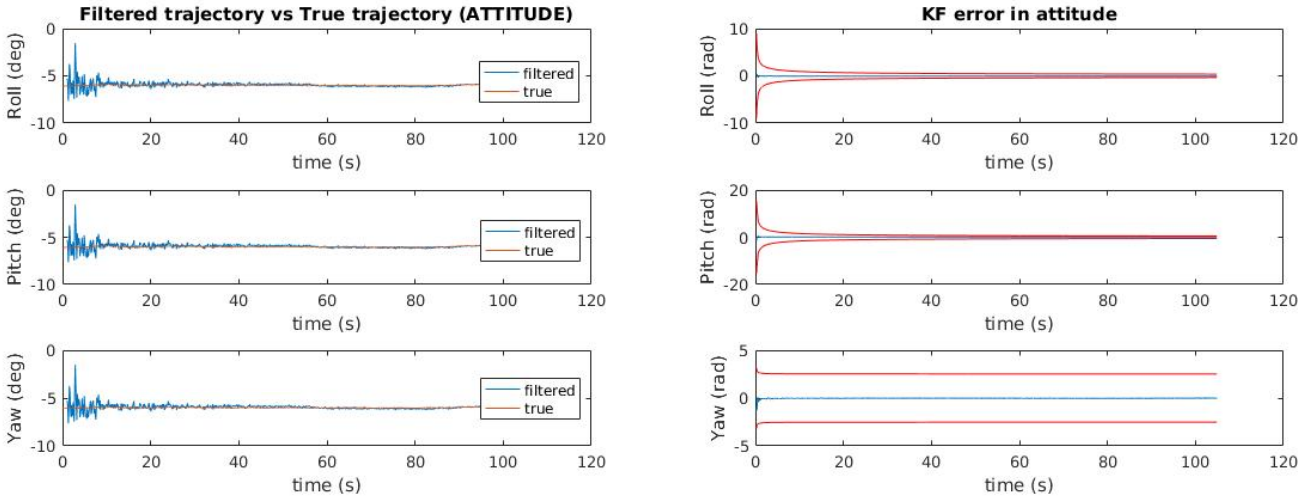


Figure 24: Alignment Method 3 : Attitude

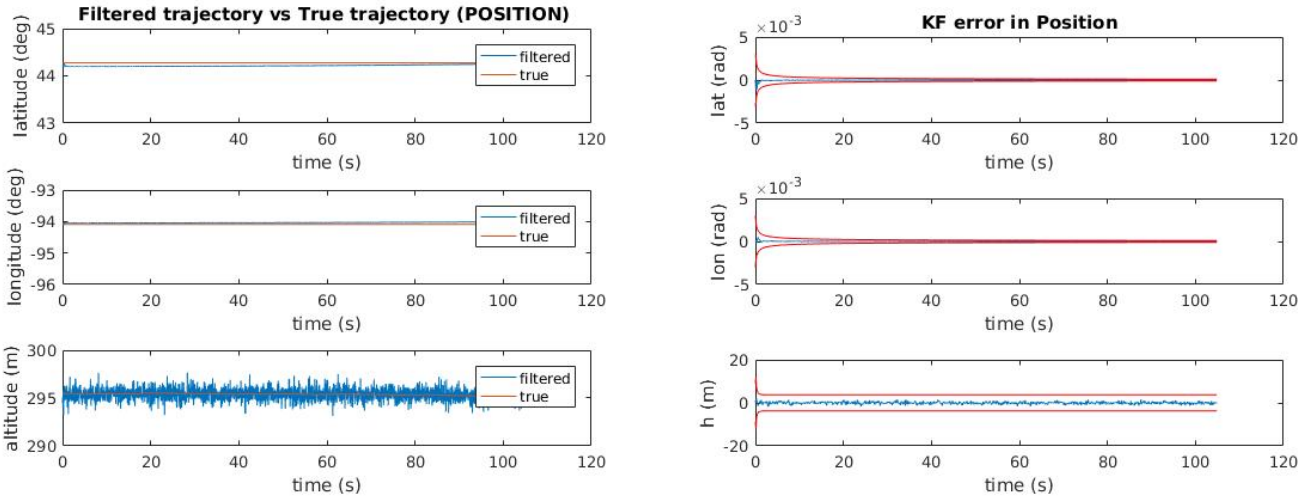


Figure 25: Alignment Method 3 : Position

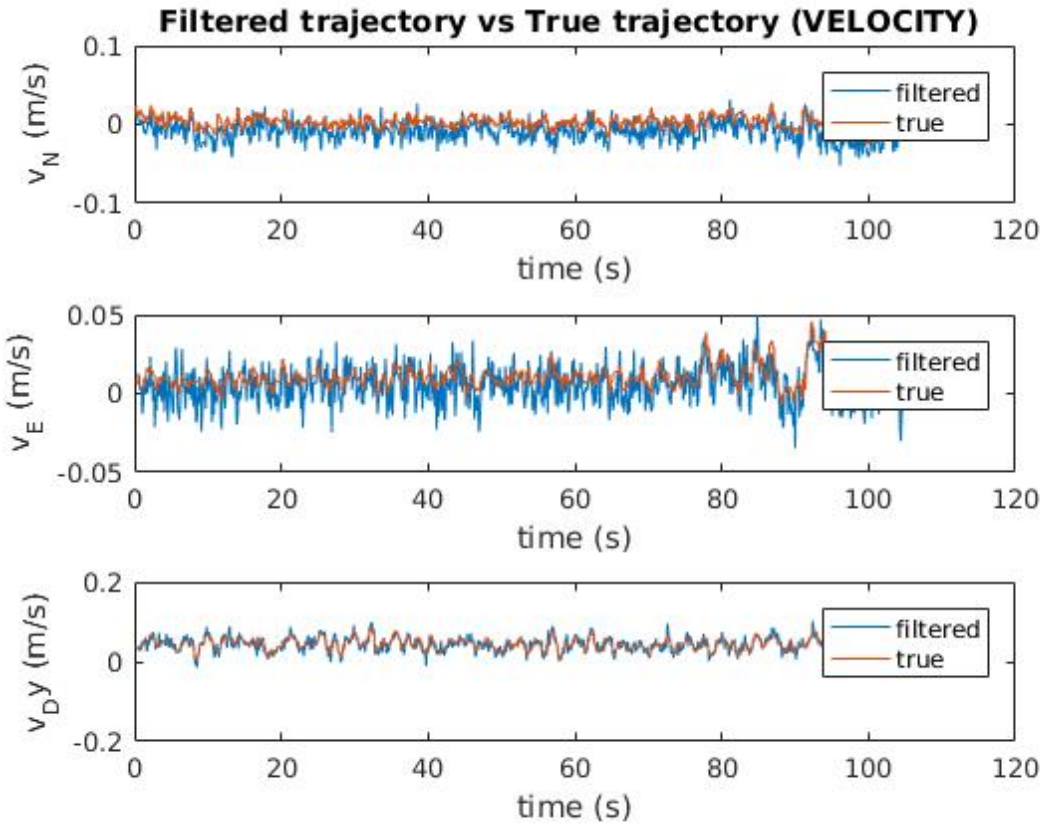


Figure 26: Alignment Method 3 : Velocity

References

[1] Braun, B.(216) *High Performance Kalman Filter Tuning for Integrated Navigation Systems*. Thesis dissertation, TU Munich. http://www.fsd.mw.tum.de/research/sensors-data-fusion-and-navigation/

[2] S. Chattaraj, A. Mukherjee, and S. K. Chaudhuri (2013). *Transfer Alignment Problem: Algorithms and Design Issues*. ISSN 20751087, Gyroscopy and Navigation, 2013, Vol. 4, No. 3, pp. 130–146. © Pleiades Publishing, Ltd..

[3] Agnar Sveinsson (2012). *INS/GPS Error Analysis And Integration*. June 2012. M.Sc Research Thesis. School of Science and Engineering Reykjavik University. Iceland

[4] Britting, K. R. (1971).*Inertial Navigation Systems Analysis*.. 111 River Street, Hoboken: John Wiley Sons, Inc.

[5] Christensen, R., & Fogh, N. (2008).*Christensen, R., & Fogh, N. (2008)*. Master's thesis, Aalborg University, Aalborg, Denmark.

[6] Thomas Brunner, Jean-Philippe Lauffenburger, Sébastien Changey and Michel Basset (2015). *Magnetometer-Augmented IMU Simulator: In-Depth Elaboration*. sensors ISSN 1424-8220. Article

[7] Yan G., Wang J., Zhou X. (2015) *High-Precision Simulator for Strapdown Inertial Navigation Systems Based on Real Dynamics from GNSS and IMU Integration* In: Sun J., Liu J., Fan S., Lu X. (eds) China Satellite Navigation Conference (CSNC) 2015 Proceedings: Volume III. Lecture Notes in Electrical Engineering, vol 342. Springer, Berlin, Heidelberg

[8] Richard Giroux, Richard Gourdeau, Rene Jr. Landry (2005). *Extended Kalman filter implementation for low-cost INS/GPS Integration in a Fast Prototyping Environment*. 16^{th} Symposium on Navigation of the Canadian Navigation Society Toronto, Canada, 26-27 April 2005

[9] Mokhtarzadeh, Hamid; Colten, Todd. (2015). *Small UAV Position and Attitude, Raw Sensor, and Aerial Imagery Data Collected over Farm Field with Surveyed Markers*. Retrieved from the Data Repository for the University of Minnesota, http://dx.doi.org/10.13020/D6BC7Z