Journal of Signal Processing Systems (2022) 94:1309–1343
https://doi.org/10.1007/s11265-020-01620-4

Efficient Attitude Estimators: A Tutorial and Survey

Hussein Al-Jlailaty[1] · **Mohammad M. Mansour**[1]

Received: 8 July 2020 / Revised: 27 October 2020 / Accepted: 6 December 2020 / Published online: 2 May 2021

Abstract
Inertial sensors based on micro-electro-mechanical systems (MEMS) technology, such as accelerometers and angular rate sensors, are cost-effective solutions used in inertial navigation systems in a broad spectrum of applications that estimate position, velocity and orientation of a system with respect to an inertial reference frame. Although they present several advantages in terms of cost and form factor, they are prone to various disturbances such as noise, biases, and random walk that degrade their orientation estimation. The task of an orientation filter is to compute an optimal solution for the attitude state, consisting of roll, pitch and yaw, through the fusion of angular rate, accelerometer, and magnetometer measurements, regardless of the underlying environmental constraints. The aim of this paper is threefold: first, it serves researchers and practitioners in the signal processing community seeking the most appropriate attitude estimators that fulfills their needs, shedding light on the drawbacks and the advantages of a wide variety of designs. Second, it serves as a survey and tutorial for existing estimator designs in the literature, assessing their design aspects and components, and dissecting their hidden details for the benefit of researchers. Third, a comprehensive list of algorithms is discussed for a fully functional inertial navigation system, starting from the navigation equations and ending with the filter equations, keeping in mind their suitability for power-limited embedded processors. The source code of all algorithms is published, with the aim of it being an out-of-box solution for researchers in the field. The reader will take away the following concepts from this article: understand the key concepts of an inertial navigation system; be able to implement and test a complete stand alone solution; be able to evaluate and understand different algorithms; understand the trade-offs between different filter architectures and techniques; and understand efficient embedded processing techniques, trends and opportunities.

Keywords Coning and sculling compensation · Embedded processors · Kalman filter · Measurement model · Navigation equations · Quaternions · Rotation matrix · System-error dynamics · Tilt errors

1 Introduction

An integrated navigation system exploits the complementary characteristics of different navigation sensors, such as gyroscopes, accelerometers, magnetometers, and global navigation satellite systems GNSS, to increase the precision of the navigation solution. For example, it can compute a high quality pose estimate of a vehicle's position and orientation (up to an accuracy of 0.00l degrees per roll, pitch and yaw axis [1]). Specifically, the underlying attitude (orientation) estimation problem is common to a wide area of applications, ranging from unmanned aerial vehicles (UAVs), virtual reality applications, underwater submersible systems, robots and ground vehicles, to medical instruments and surveying equipment. In addition, recent advances in MEMS have led to a very wide range of low-cost, light-weight and accurate components that increase the reliability of the navigation solution significantly. Nowadays, the navigation technique common to almost all integrated navigation systems is the *strapdown* inertial navigation technique. In a strapdown inertial navigation system (INS), an inertial measurement unit (IMU), mounted to the vehicle, senses accelerations and angular rates for all six degrees of freedom of the vehicle. From this data, a strapdown algorithm

This article belongs to the Topical Collection: *Survey Papers*

✉ Hussein Al-Jlailaty
hma98@mail.aub.edu

Mohammad M. Mansour
mmansour@aub.edu.lb

[1] Department of Electrical and Computer Engineering, Maroun Semaan Faculty of Engineering and Architecture, American University of Beirut, Beirut, Lebanon

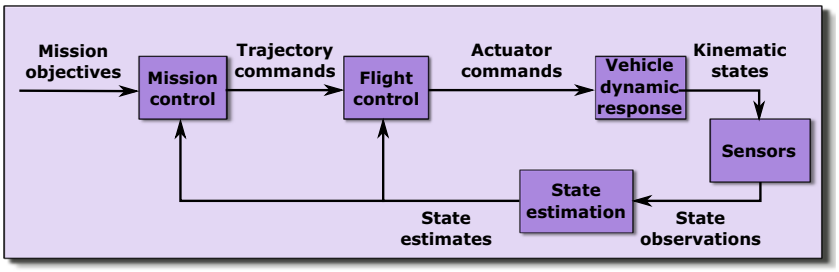

Figure 1 Block diagram illustrating the basic elements in controlling a robot.

(SDA) can compute a navigation solution, assuming that initial position, velocity and attitude are known [2]. INS-SDAs must be reliable and computationally-efficient in order to suit low-end applications with power-limited processing capabilities [3].

The brain of a typical robot vehicle is called a navigation computer or controller [4]. The controller uses on-board sensors to estimate its current position and orientation. Figure 1 shows a simplified block diagram of a generic autonomous vehicle computer, including a mission controller, a state estimator and a command controller. State estimation is implemented by fusing the raw measurements from a set of state-observing sensors, and forming an estimate of the vehicle's state (position and attitude) [5]. The vehicle controller algorithms automatically manipulate the actuators on-board the vehicle to achieve a set of trajectory commands using the system states (position and orientation) as feedback. The trajectory commands are generated by the mission controller, which could be a human operator or a set of algorithms that convert mission objectives into trajectory commands [6] (Fig. 1).

This paper aims at providing an overview of various attitude estimation techniques, building up the knowledge of the reader from basic principles, as well as providing insight in an intuitive manner for concepts hidden between the lines of sophisticated equations governing the system as a whole. The rest of the article is organized as follows:

- Section 2 provides background on the context of why attitude estimation filters are important.
- Section 3 gives a basic overview of attitude estimators currently in use, their components and applications.
- Section 4 describes various coordinate systems used in inertial navigation systems and defines the transformation of coordinates from one frame to another. Of primary concern is their relative orientation (Fig. 2).
- Section 5 discusses and elaborates on applied inertial navigation algorithms, and presents efficient pseudo-codes of various INS algorithms.
- Section 6 develops the basic INS error equations and gives insights on deriving a simplified system model of the same error equations without delving deeply into rigorous mathematical proofs. This aims at giving the reader intuition into one of the most basic components of an inertial navigation system.
- Section 7 discusses the components and the computational aspects of Kalman filters. Also efficient algorithms targeted for embedded processors are presented.
- Section 8 is dedicated to insights of the Kalman filter implementations. It points the readers' attention to some of the practical aspects to be considered when designing an attitude estimation filter.
- Section 9 presents different approaches to solving the attitude estimation problem, which are very efficient in terms of computational load and power consumption.

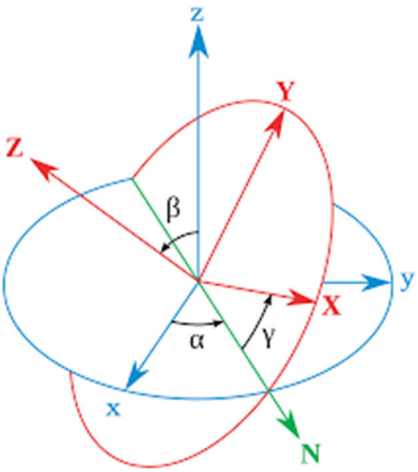

Figure 2 Euler Angles (α, β, and γ).

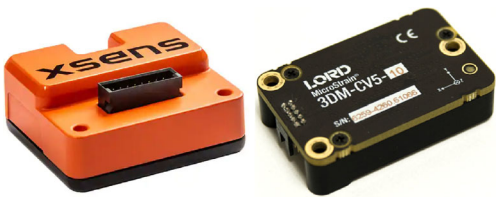

Figure 3 Commercial attitude and heading reference systems with built-in IMUs from XSens Technologies BV and LORD MicroStrain, with three accelerometers, three gyroscopes, three magnetometers and three temperature sensors for sensor calibration. These IMUs include coning and sculling compensation (see Section 5), which enables them to deliver attitude data at low rates without loosing accuracy.

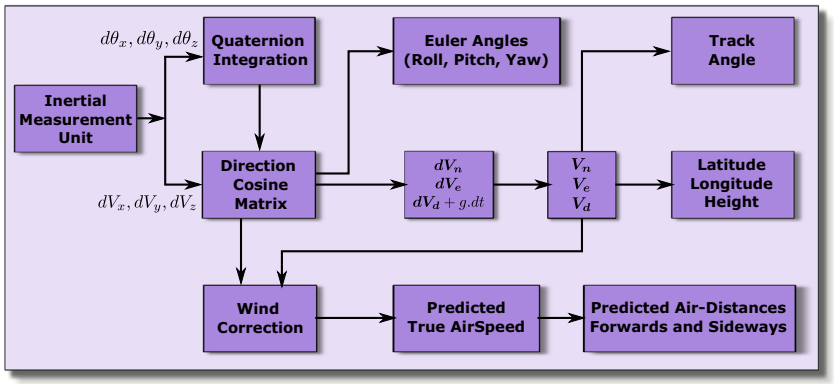

Figure 4 The inertial measurement unit is the basic element in any inertial navigation system. Raw data acquired from the IMU in the form of delta velocities and delta angles are integrated and converted to the navigation frame. When in the navigation frame, delta velocities can be added to estimated wind speeds to predict true airspeed and thus predict air-distances. Air distances are essential in estimating the amount of energy consumption (fuel or battery) for any flying vehicle.

2 Background on Filters

In this section, we provide a background introduction of the position of attitude estimation filters in the context of inertial navigation systems in general. We also demonstrate the development phases, and provide a brief description of the major mile-stones in its history.

2.1 Attitude Estimation Filters and Inertial Systems

The heart of any inertial navigation system is a fully calibrated and embedded inertial measurement unit (IMU) like the 3DM-CV5 from LORD MicroStrain or the series of IMUs from XSens Technologies BV (Fig. 3.). IMUs should deliver accurate temperature-compensated (see Section 8 for sensor calibration and compensation) inertial sensor data from three gyroscopes and three accelerometers to a navigation floating-point digital signal processor (DSP) [7, 8].

Small angular increments ($d\phi$[1]) terms obtained from gyroscope sensors are compensated for fine gyro-bias[2] corrections and then integrated using fast quaternion algorithms (Section 5) to derive a 3×3 (9 element) direction cosine matrix (Fig. 4), which defines the instantaneous orientation of the vehicle relative to the local level earth-centered coordinates (*North, East, Down*) [9, 10]. Sensed velocity increments (dV[3]) obtained from accelerometers are then transformed into delta velocity incremental components (with the aid of the direction cosine matrix) in the local-level earth-referenced coordinate frame.

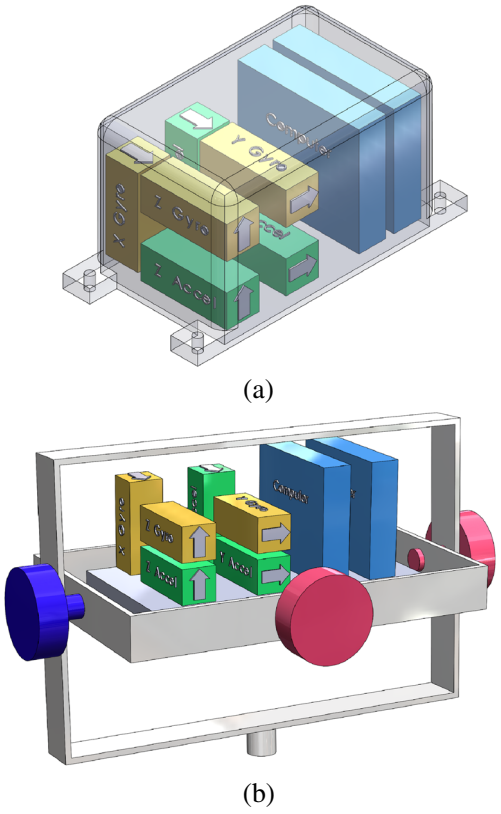

Figure 5 The strapdown system replaces gimbals with a computer that simulates their presence electronically. In the strapdown system, the gyroscopes and accelerometers are rigidly mounted to the vehicle structure so that they move with the vehicle. In a three axis gimballed platform, the gyros alone will try to maintain the platform aligned in inertial space. If the platform is operating in local-level coordinates, the navigation computer must keep the platform horizontal. It does this by sending command signals to the gyros that otherwise would fight the gimbals motion.

[1] Small angular increments, also called Delta theta terms $d\phi$, are small measured angular changes over one IMU cycle.

[2] Gyro bias is the mean angular change per second measured by the gyro when the actual angular rate is zero (stationary case).

[3] Small velocity increments delivered by the accelerometer sensors during a time step dT, also called delta velocity terms.

Since the IMU also senses gravity, the delta velocities contain components of integrated gravity [11]. To compensate for this gravity component in the down direction, we subtract it from the delta-velocity components and then integrate to give velocity components in the local-earth referenced coordinate frame (V_n, V_e and V_d), which are subsequently integrated further to produce updated values of latitude, longitude and altitude. Using the direction cosine matrix, heading, roll and pitch values are computed [12].

Usually a high speed Kalman filter propagated and updated at a predefined computation rate is used to estimate, align, and correct system computed states and residual fine inertial sensor bias values. This is done using measurement aiding from multiple sources including, GPS-receiver's position and velocity, barometric pressure sensors and magnetic heading. Velocity measurements is required to enable the system to maintain a mathematical representation of horizontal [13]. When the vehicle is moving, and especially when velocity is changing, you have no means of separating sensed gravitational acceleration from true acceleration, and a slight mathematical misalignment in your horizontal model will result in erroneous measurements of acceleration (see Section 6) since you will be sensing a component of gravity. If you have a velocity reference (from airspeed sensors, wheel-encoders or GPS, etc.) you can correct the misalignment, maintain a true representation of horizontal, and integrate acceleration and velocity correctly [14]. The way one maintains alignment in modern day navigation systems is to use a Kalman filter [15].

2.2 Development History

The fundamental principles (laws of mechanics and gravitation) on which inertial navigation is based was discovered by Isaac Newton in the seventeenth century [16]. Despite of this, it was about another two centuries before inertial navigation techniques could be demonstrated. A brief chronology of the history of inertial navigation systems is given as follows:

- **1852** - The gyroscopic effect was discovered by Foucault who was the first to use this word.
- **1923** - Schuler invents a device that enables a vertical reference to be defined [17].
- **1920** - Directional gyroscopes and artificial horizon instruments were produced for aircrafts.
- **1930** Boykow introduced the idea of using accelerometers and gyroscopes to build a functional inertial navigation system.
- **1949** - The first publication suggesting the strapdown inertial navigation concepts.
- **1950's** - The accuracy of gyroscopes increases incredibly, reducing their errors from 15°/hour to about 0.01°/hour.
- **1960's** - The start of the ring laser gyroscope and wide spread of the so-called stable platform technology.
- **1961** - NASA awarded MIT laboratory (later to become the Charles Stark Draper Laboratory), a contract for preliminary design study of a guidance and navigation system for Apollo [18, 19].
- **1970's** - Advances in technology converged to make strapdown systems available [20].
- **1980's** - Developments of higher-order gravity models enabled trajectory accuracy improvements of approximately an order of magnitude [21, 22]
- **1990's** -A non-gyroscopic inertial measurement unit was proposed that consisted of a triad of accelerometers mounted on three orthogonal platforms rotating at constant angular velocities [23].

Nearly all IMUs fall into one of the two categories; stable platform systems Fig. 5b or, strapdown systems, Fig. 5a. The difference between the two categories is the frame of

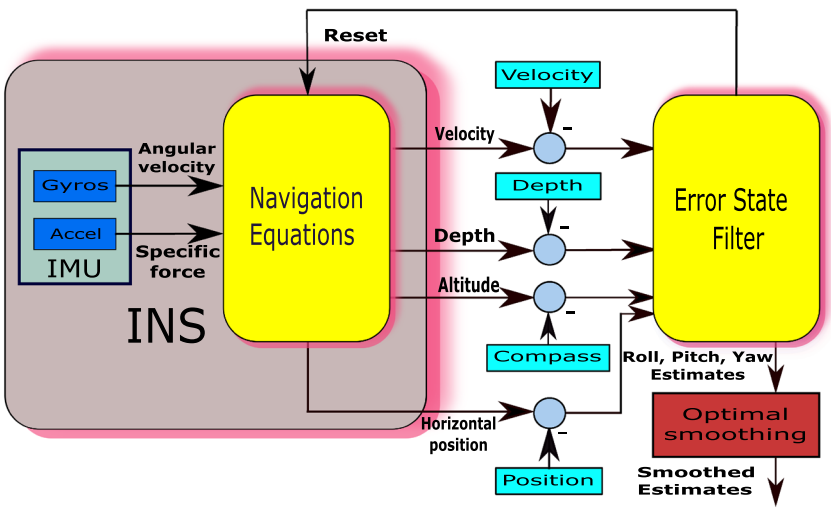

Figure 6 Any GNSS-INS system is composed of **a** INS responsible of predicting velocity, position and heading, **b** State fusion filter that combines the readings from both INS and GNSS to optimally estimate attitude of the system.

Table 1 Types of random error noise sources.

Type	Description	Units	Result of integration
MEMS Gyro Error Characteristics			
Constant Bias	The average output from the gyroscope when it is not undergoing any rotation	°/sec	A steadily growing angular error.
White Noise (Angle Random Walk ARW)	Very high frequency noise that is added to the signal that has an average amount equal to sigma (σ) and with a long term average equal to zero.	°/sec/$\sqrt{\text{Hz}}$	To find error in orientation due to gyro white noise multiply ARW by the square root of the integration time (t).
Bias Stability (Sometimes called Bias Instability)	A bias stability measurement describes how the bias of a device may change over a specified period of time. (Bias Fluctuations) [26]	°/sec	
Rate Random Walk	This is a rate error due to white noise in angular acceleration [27]	$^\circ/\,\text{sec}^{1.5}$	Introduces the opportunity to plan for re-calibration in critical applications that require extended life.
Accelerometer Error Characteristics			
Constant Bias	The average output from the accelerometer when it is not undergoing any movement.	m/sec^2	A steadily growing velocity error.
White Noise (Velocity Random Walk VRW)	Very high frequency noise that is added to the signal that has an average amount equal to sigma (σ) and with a long term average equal to zero.	m/sec/$\sqrt{\text{hr}}$	To find error in velocity due to accelerometer white noise multiply VRW by the square root of the integration time (t).
Bias Stability (Sometimes called Bias Instability)	A bias stability measurement describes how the bias of a device may change over a specified period of time	m/sec	
Acceleration Random Walk	This is an acceleration error due to white noise in jerk (derivative of acceleration) [28].	m/sec$^2/\sqrt{\text{hr}}$	

reference in which the rate-gyroscopes and accelerometers operate.

2.2.1 Stable Platform Systems

In stable platform system types, the inertial sensors are mounted on a platform which is isolated from any external rotational motion. In other words the platform is held in alignment with the global frame. This is achieved by mounting the platform using gimbals (frames) which allow the platform freedom in all three axes. The platform mounted gyroscopes detect any platform rotations. These signals are fed back to torque motors which rotate the gimbals in order to cancel out such rotations, hence keeping the platform aligned with the global frame. To track the orientation of the device the angles between adjacent gimbals can be read using angle pick-offs. To calculate the position of the device the signals from the platform mounted accelerometers are double integrated. Note that it is necessary to subtract acceleration due to gravity from the vertical channel before performing the integration.

2.2.2 Strapdown Systems

In strapdown systems, the inertial sensors are mounted rigidly onto the device, and therefore output quantities are measured in the body frame rather than the global frame. To keep track of orientation the signals from the rate gyroscopes are 'integrated', as described in Section 5. To track position the three accelerometer signals are resolved

into global coordinates using the known orientation, as determined by the integration of the gyro signals. The global acceleration signals are then integrated as in the stable platform algorithm.

Stable platform and strapdown systems are both based on the same underlying principles. Strapdown systems have reduced mechanical complexity and tend to be physically smaller than stable platform systems. These benefits are achieved at the cost of increased computational complexity. As the cost of computation has decreased strapdown systems have become the dominant type of INS.

More recently, there has been significant developments in inertial sensors, especially gyroscopes with large dynamic range giving the strapdown principles opportunity to be realized. This has enabled the complexity and size of inertial navigation systems to be reduced, as well as enabling reliable [24] inertial sensors to be produced at a relatively inexpensive price, which led to significant advancements in a diversity of applications (see Section 3).

3 Aided Inertial Navigation Systems

The inspiration of any system integration concept, is to get superior execution than would be conceivable by any of the stand-alone systems. This section starts with a look at the qualities of GNSS and INS systems that make them so appropriate to combine together [25]. The subtleties for combining the systems together will be examined afterward within this section.

One of the imperative points of interest of inertial (gyroscopes and accelerometers) systems is that they require no interaction with the environment past the client. This is attractive particularly to clients where outside supporting cannot be depended upon or is rare. In another sense, no external interference besides the client is needed for the INS to work properly. On the contrary, GNSS systems, which depend on signals transmitted from satellites, obviously, can't be ensured in all cases, and transitory blackouts going from seconds to minutes might be conceivable, contingent upon the application and working environment (see Fig. 6).

The only restriction concerning the output rate of the inertial system, is the computational power of the INS host computer. Some INS's are able of delivering the navigation state vector at 100 Hz or more. On the other hand, most GNSS receivers have data rates of 1 to 20 Hz, in spite of the fact that a few specialized receivers can give yield up to 100 Hz. Expressed in an another way, the bandwidth of the navigation states delivered by inertial system is regularly much higher than with GNSS, which is vital in guidance and control and for high-dynamic applications.

GNSS and INS are complementary in terms of the information they provide. In particular, despite the fact that GNSS can provide an attitude solution, this is usually dodged in practice because it includes employing multiple receiver antennas and expensive equipment, while attitude is the main output of INS algorithms.

Most critically, GNSS and INS systems are also complementary in terms of their errors (Table 1). While low-cost INS inertial sensors are error unbounded, GNSS provides velocity and position estimates that are limited and bounded in terms of their errors. Also, GNSS systems are dominated by high-frequency errors while INS systems are susceptible to low-frequency errors due to the integration (effectively a low-pass filter) of the mechanization equations (see Section 6).

With respect to what has been mentioned, whenever GNSS and INS systems are fused, the GNSS can deliver high-fidelity position and velocity measurements that can bound the INS system generated errors, which in turn delivers high frequency navigation states (attitude, position and velocity) needed for guidance and control of vehicles. The INS system can also maintain good accuracy in case of outages of GNSS during temporary blockage of receiver antennas. These are the main reasons that motivate the integration of both systems nowadays.

In this article, we will demonstrate to the reader how both systems (INS and GNSS) can be fused together for an optimal navigation solution. It is helpful to always remember that the navigation solution is obtained by integrating the acceleration readings to obtain velocity and by double integrating the sensed accelerometers to obtain position.

3.1 Applications of Inertial Navigation Systems

Inertial navigation systems are used extensively in every day applications, covering aircraft navigation, spacecrafts, robots, unmanned aerospace vehicles and ships. As well,

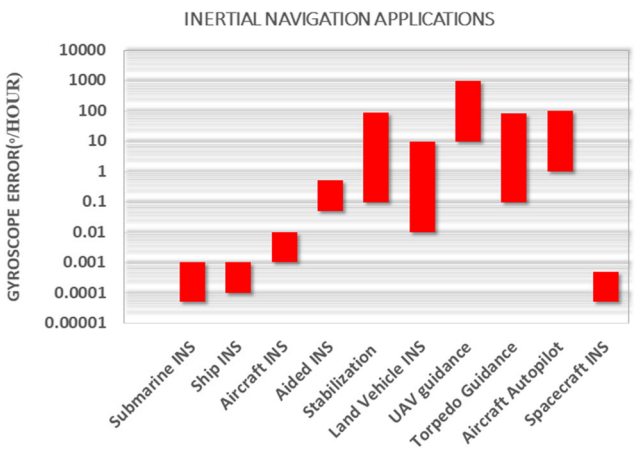

Figure 7 Diverse applications need different accuracies of gyros in strapdown inertial navigation systems.

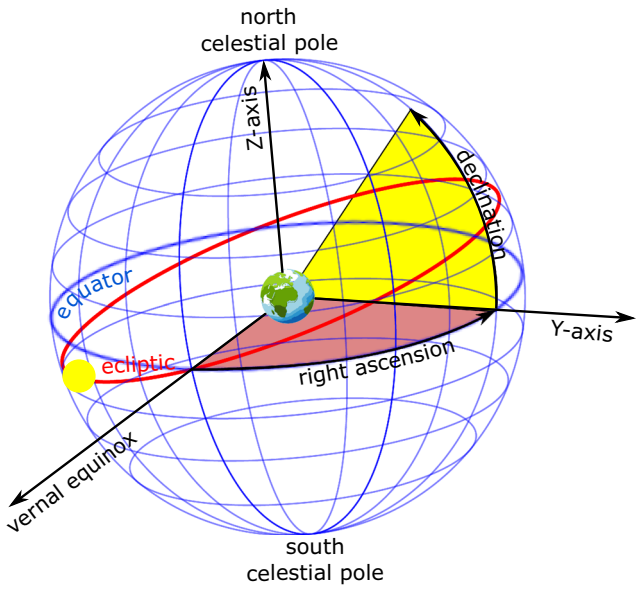

Figure 8 Earth-centered inertial (ECI) frame.

many novel applications include, active suspension of high performance racing cars, Stewart platform simulators and surveying of underground oil pipelines and wells. They can also be applied to many advanced medical equipment, such as MRI devices, surgical robots and intelligent beds. The use of inertial navigation systems is widely spreading in the medical field, for example, in the manufacturing of wheel chairs based on inertial systems. They have been placed on head trackers of disabled people where they can choose where to go and in what direction solely by moving their heads.

Due to such diversity of fields where inertial systems may be applied, a broad range of inertial sensor accuracy is required (especially for gyroscopes, see Fig. 7).

Also, since inertial systems differ in the amount of time they will be required to report accurate data, it is necessary to choose the sensors accordingly. For example, many airborne systems may need to provide accurate position and attitude data for several hundreds of kilometers or several hours. In this instance, it is necessary to rely on inertial sensors having very low residual gyroscope biases, having the order of 0.001 degrees per hour. Other cases involving marine or space applications may be required to provide accurate data for weeks or even months. In these extreme cases, gyroscopes having bias errors on the order of 0.0001 degrees per hour are mandatory. In some cases such as torpedo guidance operating for a few minutes, it is sufficient to rely on sensors with moderate accuracy (0.1 to 100 degrees per hour, see Fig. 7).

4 Coordinate Frames and Transformations

The attitude of a vehicle is defined as its orientation with respect to a reference frame. It is substantial to understand the different coordinate frames used in inertial navigation systems and their transformations to grasp its concepts. In this section we will discuss the basic coordinate frames that have three orthogonal unit vectors and that follow the right-hand rule [29].

4.1 Coordinate Frames

The measurement sensed by an Inertial Measurement Unit (IMU) are three orthogonal components of the body rotation rates and three accelerations in a coordinate frame, which is not directly related to any coordinate frame. These measurements have to be analytically integrated and transformed through several coordinate frames. It is important therefore that all coordinate frames involved in the transformation of the measurements, and results of integration are well defined before any discussion of an inertial navigation system is presented.

4.1.1 Earth-Centered Inertial (ECI) Frame

Newton defines the inertial frame as the frame of reference that does not rotate or accelerate.

Such a frame is not practically realized although, theoretically well defined. It is best approximated as one that is fixed with respect to the distant stars. For all practical

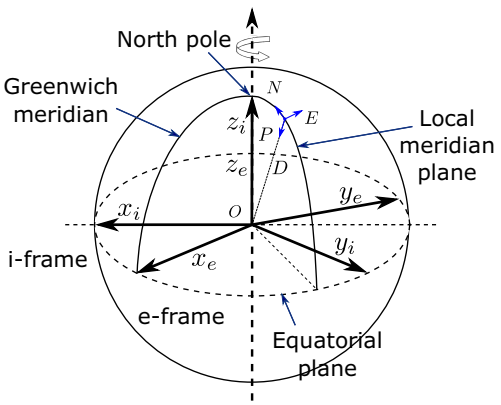

Figure 9 The relative orientation and position of the Earth, inertial frame, and navigation frame.

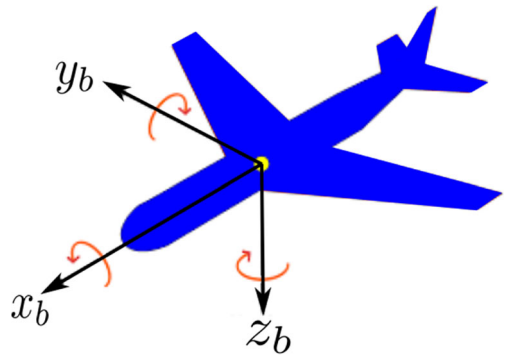

Figure 10 The x-axis of the body frame is aligned with the longitudinal axis of the air-frame. The y-axis is aligned with the right wing, while the z-axis completes the triad.

purposes the inertial frame can be treated as the frame that has the following (see Fig. 8):

- x_i-axis towards the mean vernal equinox.
- y_i-axis completes a right handed system.
- z_i-axis towards the north celestial pole.

4.1.2 Earth-Centered, Earth-Fixed (ECEF) Frame

It is a right-handed coordinate system that rotates with and is attached to the earth, which is why it is called earth fixed. This frame is not inertial since, it revolves around the sun at an average orbital speed of 29.78 km/sec and rotates at a rate of $7.292115.10^{-5}$ rad/sec. The Earth-fixed frame can be defined as follows:

- Its origin at the mass center of the earth.
- x_e-axis pointing towards the Greenwich meridian in the equatorial plane.
- y_e-axis 90 degrees of Greenwich meridian, in the equatorial plane.
- z_e-axis is the axis of rotation of the earth and passes through the north pole.

In fact, it is important to note that the Global Positioning System (GPS) reports the position and velocity of the satellites in the ECEF coordinates system (Fig. 9).

4.1.3 Local-Level or Navigation Frame

It is a non-inertial frame that is commonly used to describe the navigation of a vehicle in a local-level frame. Its axes are aligned along the geodetic directions defined by the earth's surface. Is is defined as follows:

- Its origin is at the mass center of the vehicle under study.
- The x_n-axis points north parallel to the geoid surface.
- The y_e-axis points east parallel to the geoid surface, along a latitude curve.
- The z_d-axis points downward, toward the Earth surface, anti-parallel to the surface outward normal N.

4.1.4 Body Frame

The body frame is a non-inertial reference frame, in which the measurements of a strapdown inertial navigation system are reported. Its axes are aligned with the output axes of the gyroscopes and accelerometers of the Inertial Measurement Unit (IMU). Thus, the raw data composed of the rotation rates and the accelerations experience by the body are coordinatized along the body axes. It is noted that the navigation frame can be rotated to the body frame by three consecutive right-handed rotations about its three axes (see Fig. 10). The definition of the body frame of an inertial navigation system can be summarized a follows:

- Its origin is at the mass center of the inertial navigation system.
- The x_b-axis points towards the front of the INS.
- The y_b-axis points towards the right of the INS.
- The z_b-axis points downwards and perpendicular to the x-y plane.

4.1.5 Platform Frame

The platform frame is a virtual frame created mainly for the derivation of the error equations. It is an image of the navigation frame which is recognized on an on-board computer using the outputs from the sensors. Since these sensors are dominated by noise, the platform frame does not coincide with the navigation frame and has a small deviation error from the navigation frame. The definition of the platform frame is as follows:

- Its origin at the mass center of the vehicle under study.
- x_p-axis slightly misaligned due to attitude errors with the x_n-axis of the navigation frame.

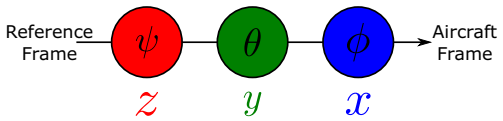

Figure 11 Euler angle sequence corresponding to the three consecutive rotations about the z, y, and x axes, respectively.

- y_p-axis slightly misaligned with the x_e-axis of the navigation frame and perpendicular to the x_p-axis.
- z_p-axis completes an orthogonal right-handed system.

4.1.6 Sensor Frame

Due to installation errors, the body frame does not coincide with the sensitivity axes of the sensors (accelerometers and gyros) used in our inertial navigation system. These errors can be compensated during manufacturing by appropriate calibration. For this reason, we assume here that the body frame and the sensor frame coincide (they are interchangeable).

4.2 Transformations

In this section, the basic mathematical tools that define the transformation between orthogonal coordinate systems are introduced. We will focus mainly on the concepts of *Rotation Matrix*, *Quaternions*, and *Rotation Vectors*.

4.2.1 Rotation Matrix

A common coordinate transformation in this article is the rotation from the *North-East-Down* coordinate frame to the body *x-y-z* coordinate frame via the ordered Euler angles (see BOX A) yaw (ψ), pitch (θ), and roll (ϕ).

A sequence of such distinctive rotations is often called a Euler angle sequence of rotations. The restriction stated above that successive axes of rotations be distinct still permits at least 12 Euler angle sequences. The sequence **xzy** means a rotation about the **x**-axis, followed by a rotation about the new **z**-axis, followed by a rotation about the newer **y**-axis.

Specifically the first rotation is ψ about z which is denoted here as $C_z(\psi)$. The second rotation is $C_y(\theta)$, or θ about y. Finally, the third rotation is $C_x(\phi)$ or ϕ about x (see Fig. 11). These three single axis rotations are written as:

$$C_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix}, \quad (1)$$

$$C_y(\theta) = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix}, \quad (2)$$

$$C_z(\psi) = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (3)$$

Box A: Direction Cosine Matrix

The Euler angles are three angles introduced by Leonhard Euler to describe the orientation of a rigid body with respect to a fixed coordinate system. Leonard Euler (1707-1783) was one of the giants inn mathematics [30]. Euler stated and proved a theorem that states that:
Any two independent orthonormal coordinate frames can be related by a sequence of rotations (not more than three) about coordinate axes, where no two successive rotations may be about the same axis.
A sequence of such rotations is often called a Euler angle sequence of rotations. The restriction stated in the above theorem that successive axes of rotations be distinct still permits at least 12 Euler angle sequences. The sequence xzy means a rotation about the x-axis, followed by a rotation about the new z-axis, followed by a rotation about the newer y-axis.
We are specifically interested in the well-known Euler sequence called the Aerospace sequence. This sequence (**zyx**) is commonly used in aircraft and aerospace applications. For example, a primary flight instrument used in air-crafts, continuously relates the orientation of the aircraft to the local-level *n*-frame mentioned above.

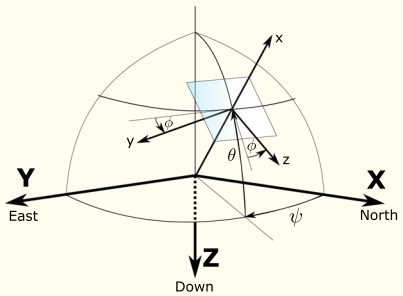

Figure 12 The Euler angles of a vehicle when aligned with the **x**-axis direction.

From the *n*-frame, first a rotation through the angle ψ about the **z**-axis defines the aircraft heading. This is followed by a rotation about the new **y**-axis through an angle θ which defines the aircraft pitch. Finally, the aircraft roll angle ϕ, is a rotation about the newest **x**-axis. These three Euler angle rotations relate the body coordinate frame of the aircraft to the local-level *n*-frame.

Thus, the transformation from body x-y-z coordinate frame coordinate frame to the n-frame (North-East-Down) is written as a cascade of the three single-axis rotations above, which can be solved using standard matrix multiplication:

$$C_n^b = C_z(\psi)C_y(\theta)C_x(\phi) = \begin{bmatrix} \cos\psi\cos\theta & \cos\psi\sin\theta\sin\phi+\sin\theta\sin\phi & \cos\psi\sin\theta\sin\phi+\sin\theta\sin\phi \\ -\sin\psi\cos\theta & -\sin\psi\sin\theta\sin\phi+\cos\theta\cos\phi & \sin\psi\sin\theta\cos\phi+\cos\theta\sin\phi \\ \sin\theta & -\cos\theta\sin\phi & \cos\theta\cos\phi \end{bmatrix}. \tag{4}$$

4.2.2 Quaternions

The rotation matrix describes the rotation of 3 degrees of freedom with 9 quantities, with redundancy. Euler angles and rotation vectors are compact but with singularity as mentioned before.

Normal complex numbers can describe rotations in a plane. Recall that in order to rotate a two degrees-of-freedom vector represented be complex number in the plane by an angle θ, we multiply by $e^{i\theta}$. It can be written in the usual form

$$e^{i\theta} = \cos\theta + i\sin\theta. \tag{5}$$

A quaternion $\mathbf{q}$ has a real part and three imaginary parts. Usually the real part is written first and the three imaginary parts next, as

$$\mathbf{q} = q_0 + q_1 i + q_2 j + q_3 k, \tag{6}$$

where i, j, k are three imaginary parts of the quaternion. These imaginary parts satisfy the following equations:

$$\begin{cases} i^2 = j^2 = k^2 = -1 \\ ij = k, ji = -k \\ jk = i, kj = -i \\ ki = j, ik = -j \end{cases} \tag{7}$$

Alternatively, quaternions are often represented using a scalar and a vector as:

$$\mathbf{q} = [s, \mathbf{v}]^{\mathrm{T}}, \quad s = q_0 \in \mathbb{R}, \quad \mathbf{v} = [q_1, q_2, q_3]^{\mathrm{T}} \in \mathbb{R}^3.$$

Here s is the real part of the quaternion and $\mathbf{v}$ is its imaginary part. If the imaginary part of the quaternion is $\mathbf{0}$ it is called a **real quaternion** and if the real part is 0 it is called **imaginary quaternion**.

4.2.3 Rotation Vector

In fact, a rotation can be described by a **rotation vector and a rotation angle**. Thus we can use a vector whose direction is parallel to the axis of rotation and whose magnitude is equal to the angle of rotation.

Let us introduce a rotation vector $\mathbf{\Phi}$, which is directed along the axis of rotation and has a magnitude equal to the rotation angle in radians. The equation of the rotation vector

Box B: Navigation Frame Relations

We can find the transformation between the n-frame and the e-frame by using Euler angles. First we rotate about the east axis by $(\pi/2 + \phi)$, then rotate about the new z-axis by the angle $-\lambda$ (see Fig. 13:

$$C_n^e = C_z(-\lambda)C_y(\pi/2 + \phi) \tag{10}$$

The result is

$$C_n^e = \begin{bmatrix} -\sin\phi\cos\lambda & \sin\lambda & -\cos\phi\cos\lambda \\ -\sin\phi\sin\lambda & \cos\lambda & -\cos\phi\sin\lambda \\ \cos\phi & 0 & -\sin\phi \end{bmatrix} \tag{11}$$

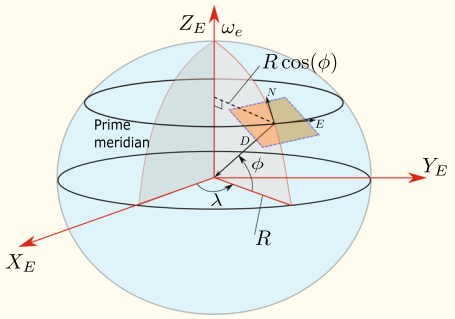

Figure 13 The figure shows the geometrical relationship between the navigation frame and the Earth frame.

Looking closely at Fig. 13 we can find the equation for the angular velocity of the n-frame with respect to the e-frame coordinatized in the n-frame, ω_{en}^n. Clearly, moving along the north direction is accompanied with a mandatory rotation rate, $\dot{\phi}$, of the n-frame around the east axis, to keep it level. Also, any motion in the east direction is accompanied by a rotation rate, $\dot{\lambda}$, of the n-frame around an axis parallel to the Z_E direction, Since the Z_E direction makes an angle ϕ with the north axis, its components along the north and down axes are respectively, $\cos\phi$ and $-\sin\phi$. Therefore:

$$\omega_{en}^n = (\dot{\lambda}\cos\phi, -\dot{\phi}, -\dot{\lambda}\sin\phi) \tag{12}$$

From similar geometric considerations, the angular rate of the n-frame with respect to the i-frame is:

$$\omega_{in}^n = ((\dot{\lambda} + \omega_e)\cos\phi, -\dot{\phi}, -(\dot{\lambda} + \omega_e)\sin\phi) \tag{13}$$

where ω_e is the Earth's rotation rate.

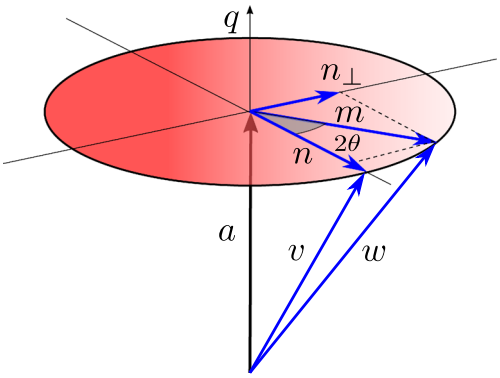

Figure 14 Vector $\boldsymbol{v}$ and its image $\boldsymbol{w}$ are related by the rotation about a vector aligned with the quaternion vector.

can be defined as

$$\mathbf{\Phi} = \|\mathbf{\Phi}\|\mathbf{n} = \begin{bmatrix} \phi_x \\ \phi_y \\ \phi_z \end{bmatrix} = \|\mathbf{\Phi}\| \begin{bmatrix} \cos\alpha \\ \cos\beta \\ \cos\gamma \end{bmatrix}, \tag{8}$$

where $\mathbf{n}$ is the unit vector in the direction of the rotation vector. α, β, γ are the angles between the rotation vector and the coordinate frame axis. The quaternion elements can be represented through the parameters of the rotation vector $\mathbf{\Phi}$ as

$$\begin{aligned} q_0 &= \cos\frac{\phi}{2}, \\ q_1 &= \sin\frac{\|\Phi\|}{2}\frac{\phi_x}{2}, \\ q_2 &= \sin\frac{\|\Phi\|}{2}\frac{\phi_y}{2}, \\ q_3 &= \sin\frac{\|\Phi\|}{2}\frac{\phi_z}{2}. \end{aligned} \tag{9}$$

We can show using Fig. 14 that the image of the vector $\boldsymbol{v}$ under rotation around the vector part of the quaternion $\boldsymbol{q}$, and through an angle "2θ" where $q_0 = \cos\theta$ ($\phi = 2\theta$ in Eq. 9) is the scalar part of the quaternion, $\boldsymbol{q}$, to be the vector $\boldsymbol{w}$.

The vector $\boldsymbol{v}$ can be resolved into a vector $\boldsymbol{a}$ along the quaternion and a vector $\boldsymbol{n}$ perpendicular to $\boldsymbol{a}$, such that, $\boldsymbol{v} = \boldsymbol{a}+\boldsymbol{n}$. Since $\boldsymbol{a}$ is aligned with $\boldsymbol{q}$ it is invariant under rotation. On the other hand, it can be easily proved geometrically that $\boldsymbol{m} = \cos 2\theta\boldsymbol{n} + \sin 2\theta\boldsymbol{n}_\perp$.

5 Applied Inertial Navigation

The main steps in obtaining a solution for a navigation system problem are as follows (Fig. 15):

- Gyro bias corrections
- Quaternion integration
- Direction cosine computation
- Heading, Roll and Pitch computation
- Delta velocity transformations to earth coordinates
- Sensed gravity component removal
- Velocity integration
- Position and altitude integration

5.1 Gyro Bias Corrections

When looking at the output of inertial sensors like gyroscopes and accelerometers, you observe that there is a small offset in the average signal output even if the sensors are not moving. This phenomena is known as sensor bias. This bias is the result of physical properties of the sensors that change over time which often lead to increase in sensor bias. The physical properties of sensors are different and so each sensor needs to be calibrated individually. Usually, gyroscopes are factory calibrated for coarse biases but even though, residual biases still remain in the gyroscope outputs. Remember that bias changes so there is no constant value that can be used to compensate for bias. Advanced

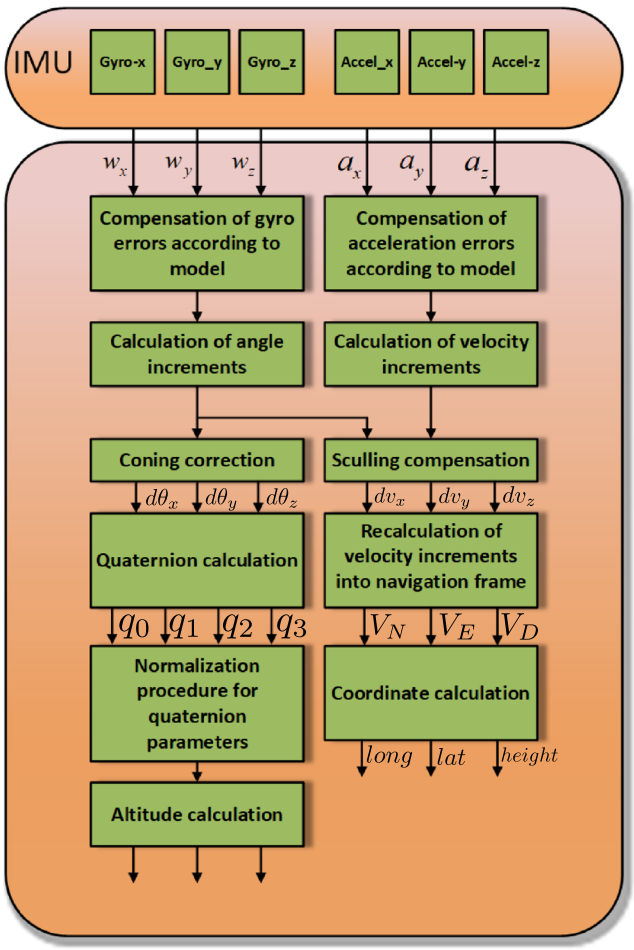

Figure 15 Navigation algorithm flow diagram.

algorithms are run in real-time to estimate and adjust for these biases.

If not corrected for bias, the output orientation of the system will drift with time. Consider a bias of 0.1 deg/s in the gyroscope output. This means we would have a drift of, $0.1 \times 60 = 6$ degrees in the orientation after one minute. Thus, it is crucial to estimate theses biases and compensate for them in the host computer of your inertial navigation system. On-board filters use sensor fusion to predict the biases and correct for them. The $d\phi$ terms extracted from the IMU are compensated for gyroscope biases by a simple subtraction operation as shown in Algorithm 1.[4]

Algorithm 1 Gyro bias corrections.

Input: Estimated biases $\delta \boldsymbol{w}_G = (\delta\omega^s_{Gx}, \delta\omega^s_{Gy}, \delta\omega^s_{Gz})$ and angular rates (w_x, w_y, w_z).

Output: De-biased angular increments $d\phi_x, d\phi_y, d\phi_z$.

1: **while** $IMUread$ **do**
2: $d\phi_x = w_x * dT$
3: $d\phi_y = w_y * dT$
4: $d\phi_z = w_z * dT$
5: $d\phi_x = d\phi_x - \delta\omega^s_{Gx} * dT$
6: $d\phi_y = d\phi_y - \delta\omega^s_{Gy} * dT$
7: $d\phi_z = d\phi_z - \delta\omega^s_{Gz} * dT$
8: **end while**
9: *Gyro Biases estimated maybe either angular rates in* deg / sec*, or angular increments in degrees. In our cases thay are angular rates so they should be multiplied by* dT *to convert to angular increments.*

5.2 Coning Correction

In old navigation systems, the inertial measurement unit (IMU) was mounted on a gimballed platform that was maintained in a horizontal position whenever the vehicle rotated, so that the gyroscopes and the accelerometers did not rotate with the vehicle. Nowadays, in strap-down systems, the gyroscopes and accelerometers are attached and rotate with the vehicle. To obtain velocities and angles you have to time integrate the reported acceleration and angular rates which is a highly non-linear operation. In particular, if you have high speed motion occurring you have to do the integration really fast to prevent errors from creeping in. This implies the IMU should be sampled at a really high rate (e.g., 1000 Hz) to provide raw angular rates and accelerometer data for the integration process. Since this integration is part of the Kalman filter (see Section 7) process, it places a heavy burden on your processor. This forms a major confliction. On the one hand you need to get data a high rate to preserve integration accuracy. On the other hand you can't afford dealing with this high throughput of data. So what the coning algorithm does is to reduce the heavy burden on the navigation processor by performing accurate high speed integration on-board the IMU processor. The output is in the form of delta theta which is the integration of the raw angular rate data. The benefit is that the delta theta quantity has already captured the integration non-linearities using a high speed coning algorithm. The resulting output still retains accuracy even at a slow rate (e.g., 100 Hz). These delta theta quantities are used by the quaternion integration block of the navigation processor to find the attitude of the system.

In order to implement the quaternion integration, the delta theta quantities which form components of the rotation vector $\boldsymbol{\phi} = [d\phi_x, d\phi_y, d\phi_z]$ for one time step should be calculated. The general equation for the dynamics of this vector $\dot{\boldsymbol{\phi}}$ can be expressed by the following equation:

$$\dot{\boldsymbol{\phi}} = \boldsymbol{\omega} + \frac{1}{2}\boldsymbol{\phi} \times \boldsymbol{\omega} + \frac{1}{\phi^2}\left(1 - \frac{\phi \sin\phi}{2(1-\cos\phi)}\right)\boldsymbol{\phi} \times (\boldsymbol{\phi} \times \boldsymbol{\omega}), \tag{14}$$

where $\boldsymbol{\phi}$ is the rotation vector that defines the attitude of the body frame $\boldsymbol{B}$ at general time t relative to frame $\boldsymbol{B}$ at time t_{m-1}, and $\boldsymbol{\omega}$ is the angular rotation rate of frame $\boldsymbol{B}$ relative to inertial space coordinatized in frame $\boldsymbol{B}$.

A more convenient form for practical implementation would require writing the sine and cosine terms as series expansions and ignoring any terms higher than third order. For example, through a series expansion, the scalar multiplier of the $\boldsymbol{\phi} \times (\boldsymbol{\phi} \times \boldsymbol{\omega})$ term in Eq. 14 can be written as:

$$\frac{1}{\phi^2}\left(1 - \frac{\phi \sin\phi}{2(1-\cos\phi)}\right) = \frac{1}{12}\left(1 + \frac{1}{60}\phi^2 + \cdots\right) \approx \frac{1}{12}. \tag{15}$$

Hence, the rate of change of the rotation vector is given by

$$\dot{\boldsymbol{\phi}} \approx \boldsymbol{\omega} + \frac{1}{2}\boldsymbol{\phi} \times \boldsymbol{\omega} + \frac{1}{12}\boldsymbol{\phi} \times (\boldsymbol{\phi} \times \boldsymbol{\omega}). \tag{16}$$

It can be shown through analysis that, to second order accuracy of ϕ,

$$\frac{1}{2}\boldsymbol{\phi} \times \boldsymbol{\omega} + \frac{1}{12}\boldsymbol{\phi} \times (\boldsymbol{\phi} \times \boldsymbol{\omega}) \approx \frac{1}{2}\boldsymbol{\alpha} \times \boldsymbol{\omega}, \tag{17}$$

with

$$\boldsymbol{\alpha} = \int_{t_{m-1}}^{t} \boldsymbol{\omega} d\tau, \tag{18}$$

where $\boldsymbol{\alpha}$ is the integral of $\boldsymbol{\omega}$ from time t_{m-1} to time t. Thus, Eq. 14 becomes to second order accuracy:

$$\dot{\boldsymbol{\phi}} \approx \boldsymbol{\omega} + \frac{1}{2}\boldsymbol{\alpha} \times \boldsymbol{\omega}. \tag{19}$$

Using Eq. 19, it is possible to determine the attitude rotation vector that relates the body B frame attitude at time t_m

[4] The adapted terminology here for estimated gyroscope biases is $\delta \boldsymbol{w}_G = (\delta\omega^s_{Gx}, \delta\omega^s_{Gy}, \delta\omega^s_{Gz})$

relative to time t_{m-1}

$$\boldsymbol{\phi}_m = \int_{t_{m-1}}^{t_m} \left[\boldsymbol{\omega} + \frac{\mathbf{1}}{2}\boldsymbol{\alpha} \times \boldsymbol{\omega}\right] d\tau = \boldsymbol{\alpha}_m + \boldsymbol{\beta}_m, \tag{20}$$

with

$$\boldsymbol{\alpha}_m = \int_{t_{m-1}}^{t_m} \boldsymbol{\omega} d\tau, \tag{21}$$

$$\boldsymbol{\beta}_m = \frac{1}{2}\int_{t_{m-1}}^{t_m} (\boldsymbol{\alpha} \times \boldsymbol{\omega})\, d\tau, \tag{22}$$

where $\boldsymbol{\beta}_m$ is by definition, the coning attitude motion from time t_{m-1} to time t_m. The variable $\boldsymbol{\beta}_m$ has been named coning term since it measures the effects of coning motion present in $\boldsymbol{\omega}$. Coning motion is the condition where the angular velocity vector is itself rotating. As can be easily seen from Eq. 21, $\boldsymbol{\alpha}$ and $\boldsymbol{\omega}$ remain parallel when the angular velocity vector does not rotate. Hence, the $\boldsymbol{\beta}_m$ terms zeroes out since the cross product in its integrand is zero. In this case, Eq. 20 reduces to

$$\boldsymbol{\phi}_m = \int_{t_{m-1}}^{t_m} \boldsymbol{\omega} d\tau. \tag{23}$$

This condition can also be seen directly from Eq. 14 since the second and third terms on the right-hand-side zero out.

5.2.1 Coning algorithm

In this section, we will develop an efficient digital algorithm for calculating the coning term. The integration time in Eq. 22 can be divided into a time up to and after t_{l-1}, where t_{l-1} is between t_{m-1} and t_m. From Eq. 22,

$$\begin{aligned} \boldsymbol{\beta}_l &= \boldsymbol{\beta}_{l-1} + \Delta\boldsymbol{\beta}_l, \quad \boldsymbol{\beta}_m = \boldsymbol{\beta}_l\big|_{t_l=t_m}, \\ \boldsymbol{\beta}_l\big|_{t_l=t_{m-1}} &= 0, \\ \Delta\boldsymbol{\beta}_l &= \tfrac{1}{2}\textstyle\int_{t_{l-1}}^{t_l} (\boldsymbol{\alpha} \times \boldsymbol{\omega})\, d\tau. \end{aligned} \tag{24}$$

A similar process can be utilized to digitize (21) giving the following

$$\begin{aligned} \boldsymbol{\alpha}_l &= \boldsymbol{\alpha}_{l-1} + \Delta\boldsymbol{\alpha}_l, \quad \boldsymbol{\alpha}_m = \boldsymbol{\alpha}_l|_{t_l=t_m}, \\ \alpha_l|_{t_l=t_{m-1}} &= 0, \\ \Delta\boldsymbol{\alpha}_l &= \textstyle\int_{t_{l-1}}^{t_l} \boldsymbol{\omega} d\tau. \end{aligned} \tag{25}$$

Substituting $\boldsymbol{\alpha} = \boldsymbol{\alpha}_{l-1} + \Delta\boldsymbol{\alpha}(t)$ in $\Delta\boldsymbol{\beta}_l$ of Eq. 24 we obtain

$$\begin{aligned} \Delta\boldsymbol{\beta}_l &= \tfrac{1}{2}(\boldsymbol{\alpha}_{l-1} \times \Delta\boldsymbol{\alpha}_l) + \tfrac{1}{2}\textstyle\int_{t_{l-1}}^{t_l} (\Delta\boldsymbol{\alpha}(t) \times \boldsymbol{\omega})\, d\tau, \\ \boldsymbol{\beta}_l &= \boldsymbol{\beta}_{l-1} + \Delta\boldsymbol{\beta}_l, \quad \boldsymbol{\beta}_m = \boldsymbol{\beta}_l\big|_{t_l=t_m}, \\ \boldsymbol{\beta}_l\big|_{t_l=t_{m-1}} &= 0. \end{aligned} \tag{26}$$

Equations 25 and 26 form the basis for a recursive digital algorithm at the high l rate of the on-board IMU processor to calculate the $\boldsymbol{\alpha}_m$ and the coning term $\boldsymbol{\beta}_m$ of the low m rate of Eq. 20. What remains is to determine a digital integration algorithm for the integral term in Eq. 26.

In order to digitize the integral term in Eq. 26, it is wise to consider an linear analytical form for the angular rate vector $\boldsymbol{\omega}$ between any two time steps t_{l-1} and t_l. Approximating $\boldsymbol{\omega}$ profile as a constant $\boldsymbol{a}$ added to a linear build-up in time having rate $\boldsymbol{b}$, we obtain

$$\boldsymbol{\omega} \approx \boldsymbol{a} + \boldsymbol{b}(t - t_{l-1}), \tag{27}$$

where both $\boldsymbol{a}$ and $\boldsymbol{b}$ are constant vectors. Therefore, both constants can be determined from current and previous values of $\Delta\boldsymbol{\alpha}_l$

$$\boldsymbol{a} = \frac{1}{2T_l}(\Delta\boldsymbol{\alpha}_l + \Delta\boldsymbol{\alpha}_{l-1}), \quad \boldsymbol{b} = \frac{1}{T_l^2}(\Delta\boldsymbol{\alpha}_l - \Delta\boldsymbol{\alpha}_{l-1}). \tag{28}$$

Substituting (28) in Eq. 27 and the integral part of Eq. 26 gives

$$\frac{1}{2}\int_{t_{l-1}}^{t_l} (\Delta\boldsymbol{\alpha}(t) \times \boldsymbol{\omega})\, d\tau = \frac{1}{12}(\Delta\boldsymbol{\alpha}_{l-1} \times \Delta\boldsymbol{\alpha}_l). \tag{29}$$

When substituted in Eq. 26, the final result is

$$\Delta\boldsymbol{\beta}_l = \frac{1}{2}\left(\boldsymbol{\alpha}_{l-1} + \frac{1}{6}\Delta\boldsymbol{\alpha}_{l-1}\right) \times \Delta\boldsymbol{\alpha}_l. \tag{30}$$

The overall digital algorithm for $\boldsymbol{\alpha}_m$ and the coning term $\boldsymbol{\beta}_m$ is determined from the above results and abbreviated in Algorithm 2.

Algorithm 2 Coning algorithm.

Input: $(d\phi_x, d\phi_y, d\phi_z)$ at high-speed cycle index l.
Output: $\boldsymbol{\phi}_m$ vector of angular increments at low-speed index m.

$\alpha_l \leftarrow 0 \quad at \quad (t = t_{m-1})$ ▷ *m the low-speed computer cycle index*
$\beta_l \leftarrow 0 \quad at \quad (t = t_{m-1})$
while $t_l < t_m$ **do** ▷ *l is the high-speed computer cycle index*
  $\Delta\boldsymbol{\alpha}_l \leftarrow (d\phi_x, d\phi_y, d\phi_z)$
  $\boldsymbol{\alpha}_l = \boldsymbol{\alpha}_{l-1} + \Delta\boldsymbol{\alpha}_l$
  $\Delta\boldsymbol{\beta}_l = \frac{1}{2}\left(\boldsymbol{\alpha}_{l-1} + \frac{1}{6}\Delta\boldsymbol{\alpha}_{l-1}\right) \times \Delta\boldsymbol{\alpha}_l$
  $\boldsymbol{\beta}_l = \boldsymbol{\beta}_{l-1} + \Delta\boldsymbol{\beta}_l$
end while
$\boldsymbol{\alpha}_m = \boldsymbol{\alpha}_l \quad at \quad (t_l = t_m)$
$\boldsymbol{\beta}_m = \boldsymbol{\beta}_l \quad at \quad (t_l = t_m)$
$\boldsymbol{\phi}_m = \boldsymbol{\alpha}_m + \boldsymbol{\beta}_m$
▷ *$\boldsymbol{\phi}_m$ vector contains the integration of delta theta terms between two m computer cycle indices with very high accuracy.*

5.3 Sculling Compensation

Sculling on the other hand is basically analogous to coning but it has to do with the accelerometers instead of the gyroscopes. Coning relates specifically to an error in your angle measurement and so fundamentally it is coming from gyro data. On the other hand, sculling happens when you have a cyclic linear acceleration in combination with cyclic rotation. We call this sculling because it results in an apparent but erroneous velocity, and the characteristic motion that gives you this erroneous velocity looks like the sculling type of oar, where the oar sweeps back and forth. Without compensation, this would come out in the delta velocity quantity and the output would have that corruption built into it. For example, if you have very fast motion, especially a vibration-like oscillating motion at the same time that you have a slow sampling rate, you will be in trouble without the sculling compensation provided.

Therefore, in order to prevent delta velocity errors from creeping in, it is convenient to account for the body frame rotation $C_{B_{(t)}}^{B_{m-1}}$ during the mth computer cycle index period. To find delta velocities, we integrate the reported accelerometer measurements according to the following

$$\Delta \boldsymbol{v}_m = \int_{t_{m-1}}^{t_m} C_{B_{(t)}}^{B_{m-1}} \boldsymbol{a}_{SF} dt, \tag{31}$$

where $\boldsymbol{a}_{SF}$ is the accelerometer reported values and $C_{B_{(t)}}^{B_{m-1}}$ the general direction cosine matrix defining the attitude of Frame B relative to Frame B_{m-1} for time t greater than t_{m-1}.

The $C_{B_{(t)}}^{B_{m-1}}$ term in Eq. 31 can be expressed as:

$$C_{B_{(t)}}^{B_{m-1}} = I + \frac{\sin\phi(t)}{\phi(t)}[\boldsymbol{\phi}(t)\times] + \frac{1-\cos\phi(t)}{\phi(t)^2}[\boldsymbol{\phi}(t)\times]^2, \tag{32}$$

where $\boldsymbol{\phi}(t) =$ Rotation vector that defines the attitude of the body frame B at general time t relative to frame B_{m-1} at time t_{m-1}, and $\phi(t) =$ Magnitude of $\boldsymbol{\phi}(t)$.

A first order approximation for Eq. 32 neglects $[\boldsymbol{\phi}(t)\times]^2$ and approximates $\sin\phi(t)/\phi(t)$ by unity. Assuming that the m cycle rate is selected fast enough to maintain $\boldsymbol{\phi}(t)$ small, e.g., less that 0.05 radians, we can write $\boldsymbol{\phi}(t) \approx \boldsymbol{\alpha}(t)$. In this case (32) becomes

$$C_{B(t)}^{B_{m-1}} \approx I + [\boldsymbol{\alpha}(t)\times]\,. \tag{33}$$

Substituting (33) in Eq. 31 then yields to first order

$$\begin{aligned}
\Delta \boldsymbol{v}_m &= \textstyle\int_{t_{m-1}}^{t_m} (I + [\boldsymbol{\alpha}(t)\times]\,\boldsymbol{a}_{SF})\,dt \\
&= \textstyle\int_{t_{m-1}}^{t_m} \boldsymbol{a}_{SF} dt + \int_{t_{m-1}}^{t_m} (\boldsymbol{\alpha}(t)\times)\,\boldsymbol{a}_{SF} dt\,, \\
\Delta \boldsymbol{v}_m &= \boldsymbol{v}_m + \textstyle\int_{t_{m-1}}^{t_m} (\boldsymbol{\alpha}(t) \times \boldsymbol{a}_{SF})\,dt\,, \\
\boldsymbol{\alpha}(t) &= \textstyle\int_{t_{m-1}}^{t} \boldsymbol{\omega} d\tau, \quad \boldsymbol{\alpha}_m = \boldsymbol{\alpha}(t_m)\,, \\
\boldsymbol{v}(t) &= \textstyle\int_{t_{m-1}}^{t} \boldsymbol{a}_{SF} d\tau, \quad \boldsymbol{v}_m = \boldsymbol{v}(t_m)\,.
\end{aligned} \tag{34}$$

Equation 34 can be further synthesized if we work on the integral term by first noting that:

$$\begin{aligned}
\tfrac{d}{dt}\,(\boldsymbol{\alpha}(t) \times \boldsymbol{v}(t)) &= \boldsymbol{\alpha}(t) \times \dot{\boldsymbol{v}}(t) + \dot{\boldsymbol{\alpha}}(t) \times \boldsymbol{v}(t) \\
&= \boldsymbol{\alpha}(t) \times \dot{\boldsymbol{v}}(t) - \boldsymbol{v}(t) \times \dot{\boldsymbol{\alpha}}(t)\,.
\end{aligned} \tag{35}$$

Upon re-arranging this equation, we obtain

$$\boldsymbol{\alpha}(t) \times \dot{\boldsymbol{v}}(t) = \frac{d}{dt}\,(\boldsymbol{\alpha}(t) \times \boldsymbol{v}(t)) + \boldsymbol{v}(t) \times \dot{\boldsymbol{\alpha}}(t)\,. \tag{36}$$

Trivially,

$$\boldsymbol{\alpha}(t) \times \dot{\boldsymbol{v}}(t) = \frac{1}{2}\boldsymbol{\alpha}(t) \times \dot{\boldsymbol{v}}(t) + \frac{1}{2}\boldsymbol{\alpha}(t) \times \dot{\boldsymbol{v}}(t)\,. \tag{37}$$

We now substitute for one of the terms on the right to obtain

$$\begin{aligned}
\boldsymbol{\alpha}(t) \times \dot{\boldsymbol{v}}(t) &= \tfrac{1}{2}\tfrac{d}{dt}\,(\boldsymbol{\alpha}(t) \times \boldsymbol{v}(t)) + \\
&\quad \tfrac{1}{2}\,(\boldsymbol{\alpha}(t) \times \dot{\boldsymbol{v}}(t) + \boldsymbol{v}(t) \times \dot{\boldsymbol{\alpha}}(t))\,.
\end{aligned} \tag{38}$$

Knowing that $\dot{\boldsymbol{\alpha}}(t) = \boldsymbol{\omega}$ and $\dot{\boldsymbol{v}}(t) = \boldsymbol{a}_{SF}$, Eq. 38 becomes

$$\begin{aligned}
\boldsymbol{\alpha}(t) \times \boldsymbol{a}_{SF} &= \tfrac{1}{2}\tfrac{d}{dt}\,(\boldsymbol{\alpha}(t) \times \boldsymbol{v}(t)) + \\
&\quad \tfrac{1}{2}\,(\boldsymbol{\alpha}(t) \times \boldsymbol{a}_{SF} + \boldsymbol{v}(t) \times \boldsymbol{\omega})\,.
\end{aligned} \tag{39}$$

Substituting (37) for the integrand in Eq. 34 yields the following

$$\begin{aligned}
\Delta \boldsymbol{v}_m &= \boldsymbol{v}_m + \tfrac{1}{2}\,(\boldsymbol{\alpha}_m \times \boldsymbol{v}_m) + \\
&\quad \textstyle\int_{t_{m-1}}^{t_m} \tfrac{1}{2}\,(\boldsymbol{\alpha}(t) \times \boldsymbol{a}_{SF} + \boldsymbol{v}(t) \times \boldsymbol{\omega})\,dt\,.
\end{aligned} \tag{40}$$

It is easily verified that the integrand in Eq. 40 vanishes for the cases where the angular velocity term $\boldsymbol{\omega}$ and the specific force $\boldsymbol{a}_{SF}$ are non-rotating and having constant magnitudes. We conclude that the integral term in Eq. 40 represents a contribution from rotating high frequency components in $\Delta \boldsymbol{v}_m$.

The integral term in Eq. 40, denoted as "sculling", measures the "constant" contribution to $\Delta \boldsymbol{v}$ under classical sculling motion (mariners propel boats using a single oar with an undulating motion) where the $\boldsymbol{\alpha}(t)$ angular excursion term about one body frame axis is at the same frequency and in phase with the specific force $\boldsymbol{a}_{SF}$ along another axis.

The other terms in Eq. 40, $\boldsymbol{v}_m + \frac{1}{2}\,(\boldsymbol{\alpha}_m \times \boldsymbol{v}_m)$, represent a combination of both low-frequency and high frequency effects. In particular, $\frac{1}{2}\,(\boldsymbol{\alpha}_m \times \boldsymbol{v}_m)$ is denoted as velocity rotation compensation term. With this terminology, Eq. 40

can be re-written as

$$\begin{aligned}
\Delta \boldsymbol{v}_m &= \boldsymbol{v}_m + \Delta \boldsymbol{v}_{Rot_m} + \Delta \boldsymbol{v}_{Scul_m}\,,\\
\Delta \boldsymbol{v}_{Scul_m} &= \textstyle\int_{t_{m-1}}^{t_m} \frac{1}{2}\left(\boldsymbol{\alpha}(t)\times \boldsymbol{a}_{SF} + \boldsymbol{v}(t)\times\boldsymbol{\omega}\right) dt\,,\\
\boldsymbol{\alpha}(t) &= \textstyle\int_{t_{m-1}}^{t} \boldsymbol{\omega} d\tau, \quad \boldsymbol{\alpha}_m = \boldsymbol{\alpha}(t_m)\,,\\
\boldsymbol{v}(t) &= \textstyle\int_{t_{m-1}}^{t} \boldsymbol{a}_{SF} d\tau, \quad \boldsymbol{v}_m = \boldsymbol{v}(t_m)\,,
\end{aligned} \tag{41}$$

and

$$\Delta \boldsymbol{v}_{Rot_m} = \frac{1}{2}\left(\boldsymbol{\alpha}_m \times \boldsymbol{v}_m\right)\,. \tag{42}$$

where $\Delta \boldsymbol{v}_{Rot_m}$ = "Velocity Rotation Compensation" term, and $\Delta \boldsymbol{v}_{Scul_m}$ = "Sculling" term. In order to develop a digital algorithm for calculating the terms in Eq. 41, we follow an identical procedure to that used for the coning algorithm. We consider the integration in Eq. 41 as divided into portions up to and after a general time t_{l-1} within the t_{m-1} to $t_m$interval so that it becomes

$$\begin{aligned}
\Delta \boldsymbol{v}_{Scul}(t) &= \Delta \boldsymbol{v}_{Scul_{l-1}} + \delta \boldsymbol{v}_{Scul}(t)\,,\\
\delta \boldsymbol{v}_{Scul}(t) &= \textstyle\int_{t_{l-1}}^{t} \frac{1}{2}\left(\boldsymbol{\alpha}(\tau)\times \boldsymbol{a}_{SF} + \boldsymbol{v}(\tau)\times\boldsymbol{\omega}\right) d\tau\,,
\end{aligned} \tag{43}$$

Now let us define the next l cycle time within the t_{m-1} to t_m interval so that at t_l we can write

$$\begin{aligned}
\boldsymbol{\alpha}_l &= \boldsymbol{\alpha}_{l-1} + \Delta\boldsymbol{\alpha}_l, \quad \boldsymbol{\alpha}_m = \boldsymbol{\alpha}_l|_{t_l=t_m}\,,\\
\Delta\boldsymbol{\alpha}(\tau) &= \textstyle\int_{t_{l-1}}^{\tau} \boldsymbol{\omega} dt, \quad \Delta\boldsymbol{\alpha}_l = \int_{t_{l-1}}^{t_l} \boldsymbol{\omega} dt\,,\\
\alpha_l|_{t_l=t_{m-1}} &= 0,\\
\boldsymbol{v}_l &= \boldsymbol{v}_{l-1} + \Delta\boldsymbol{v}_l, \quad \boldsymbol{v}_m = \boldsymbol{v}_l|_{t_l=t_m}\,,\\
\Delta\boldsymbol{v}(\tau) &= \textstyle\int_{t_{l-1}}^{\tau} \boldsymbol{a}_{SF} dt \quad \Delta\boldsymbol{v}_l = \int_{t_{l-1}}^{t_l} \boldsymbol{a}_{SF} dt\,,\\
v_l|_{t_l=t_{m-1}} &= 0\,,\\
\Delta\boldsymbol{v}_{Scul_l} &= \Delta\boldsymbol{v}_{Scul_{l-1}} + \delta\boldsymbol{v}_{Scul_l}\,,\\
\delta\boldsymbol{v}_{Scul}(t) &= \textstyle\int_{t_{l-1}}^{t} \frac{1}{2}\left(\boldsymbol{\alpha}(\tau)\times \boldsymbol{a}_{SF} + \boldsymbol{v}(\tau)\times\boldsymbol{\omega}\right) d\tau\,,\\
\Delta\boldsymbol{v}_{Scul_m} &= \Delta\boldsymbol{v}_{Scul_l}\big|_{t_l=t_m}, \Delta\boldsymbol{v}_{Scul_l}\big|_{t_l=t_{m-1}} = 0\,.
\end{aligned} \tag{44}$$

Substituting for the terms $\boldsymbol{\alpha}$ and $\boldsymbol{v}$ using Eq. 34 and incorporating the definition for $\Delta\boldsymbol{\alpha}_l$ and $\Delta\boldsymbol{v}_l$, Eq. 44 becomes

$$\begin{aligned}
\delta\boldsymbol{v}_{Scul_l} = &\tfrac{1}{2}\left(\boldsymbol{\alpha}_{l-1}\times\Delta\boldsymbol{v}_l + \boldsymbol{v}_{l-1}\times\Delta\boldsymbol{\alpha}_l\right) +\\
&\textstyle\int_{t_{l-1}}^{t_l} \frac{1}{2}\left(\Delta\boldsymbol{\alpha}(t)\times\boldsymbol{a}_{SF} + \Delta\boldsymbol{v}(t)\times\boldsymbol{\omega}\right) dt\,.
\end{aligned} \tag{45}$$

As in the coning algorithm design process, we base our development on an assumed form for the angular rate and specific-force vectors during the t_{l-1} to t_l time interval. In this case, we propose a linearly changing angular rate and specific-force vector over the t_{l-1} to t_l time interval, where its coefficients are computed from current and past l cycle sensor samples.Thus we have:

$$\boldsymbol{\omega} \approx \boldsymbol{a} + \boldsymbol{b}(t - t_{l-1}), \quad \boldsymbol{a}_{SF} \approx \boldsymbol{c} + \boldsymbol{d}(t - t_{l-1})\,, \tag{46}$$

where $\boldsymbol{a}.\boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d}$ = Constant vectors. With Eq. 46 and the $\Delta\boldsymbol{\alpha}$ and $\Delta\boldsymbol{v}$ definitions in Eq. 44

$$\begin{aligned}
\Delta\boldsymbol{\alpha}(t) &= \boldsymbol{a}(t - t_{l-1}) + \tfrac{1}{2}\boldsymbol{b}(t - t_{l-1})^2\,,\\
\Delta\boldsymbol{v}(t) &= \boldsymbol{c}(t - t_{l-1}) + \tfrac{1}{2}\boldsymbol{d}(t - t_{l-1})^2\,.
\end{aligned} \tag{47}$$

Substituting (46) and (47) for the integrand in Eq. 45 yields:

$$\begin{aligned}
&\textstyle\int_{t_{l-1}}^{t_l} \frac{1}{2}\left(\Delta\boldsymbol{\alpha}(t)\times\boldsymbol{a}_{SF} + \Delta\boldsymbol{v}(t)\times\boldsymbol{\omega}\right) dt =\\
&\tfrac{1}{12}\left(\boldsymbol{a}\times\boldsymbol{d} + \boldsymbol{c}\times\boldsymbol{b}\right) T_l^3\,.
\end{aligned} \tag{48}$$

where T_l = time interval $t_l - t_{l-1}$, i.e., the l cycle computation period. The constants $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$, and $\boldsymbol{d}$ can be calculated for each t_{l-1} to t_l time interval using successive measurements of integrated angular rate and specific force acceleration increments from the inertial sensors. To determine the constants $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$, and $\boldsymbol{d}$ uniquely, it is required to take sample measurements from two successive intervals. For sensor samples taken at the l cycle rate the results are as follows:

$$\begin{aligned}
\boldsymbol{a} &= \tfrac{1}{2T_l}\left(\Delta\boldsymbol{\alpha}_l + \Delta\boldsymbol{\alpha}_{l-1}\right)\,,\ \boldsymbol{b} = \tfrac{1}{T_l^2}\left(\Delta\boldsymbol{\alpha}_l - \Delta\boldsymbol{\alpha}_{l-1}\right)\\
\boldsymbol{c} &= \tfrac{1}{2T_l}\left(\Delta\boldsymbol{v}_l + \Delta\boldsymbol{v}_{l-1}\right)\,,\ \boldsymbol{d} = \tfrac{1}{T_l^2}\left(\Delta\boldsymbol{v}_l - \Delta\boldsymbol{v}_{l-1}\right)\,.
\end{aligned} \tag{49}$$

Substituting the terms in Eq. 49 in Eq. 48 we obtain the desired equation for $\delta\boldsymbol{v}_{Scul_l}$:

$$\begin{aligned}
\delta\boldsymbol{v}_{Scul_l} = \tfrac{1}{2}\Big[&\left(\boldsymbol{\alpha}_{l-1} + \tfrac{1}{6}\Delta\boldsymbol{\alpha}_{l-1}\right)\times\Delta\boldsymbol{v}_l +\\
&\left(\boldsymbol{v}_{l-1} + \tfrac{1}{6}\Delta\boldsymbol{v}_{l-1}\right)\times\Delta\boldsymbol{\alpha}_l\Big]\,.
\end{aligned} \tag{50}$$

A digital algorithm from the above results and from the coning equations yields the sculling Algorithm 3.

Algorithm 3 Sculling algorithm.

Input: (dv_x, dv_y, dv_z) at high-speed computer cycle-index l.

Output: $\Delta \boldsymbol{v}_m = (dv_x, dv_y, dv_z)$ vector of integrated delta velocity terms at low-speed cycle index m.

$v_l \leftarrow 0 \quad at \quad (t = t_{m-1})$ ▷ *m the low-speed computer cycle index.*

$\Delta \boldsymbol{v}_{Scul_l} \leftarrow 0 \quad at \quad (t = t_{m-1})$

while $t_l < t_m$ **do** ▷ *l is the high-speed computer cycle index*

$\quad \Delta \boldsymbol{v}_l \leftarrow (dv_x, dv_y, dv_z)$

$\quad \boldsymbol{v}_l = \boldsymbol{v}_{l-1} + \Delta \boldsymbol{v}_l$

$$\delta \boldsymbol{v}_{Scul_l} = \frac{1}{2}\left[\left(\boldsymbol{\alpha}_{l-1} + \frac{1}{6}\Delta\boldsymbol{\alpha}_{l-1}\right) \times \Delta \boldsymbol{v}_l + \left(\boldsymbol{v}_{l-1} + \frac{1}{6}\Delta \boldsymbol{v}_{l-1}\right) \times \Delta \boldsymbol{\alpha}_l\right] \tag{51}$$

$\quad \Delta \boldsymbol{v}_{Scul_l} = \Delta \boldsymbol{v}_{Scul_{l-1}} + \delta \boldsymbol{v}_{Scul_l}$

end while

$\boldsymbol{v}_m = \boldsymbol{v}_l \quad at \quad (t_l = t_m)$

$\Delta \boldsymbol{v}_{Scul_m} = \Delta \boldsymbol{v}_{Scul_l} \quad at \quad (t_l = t_m)$

$\Delta \boldsymbol{v}_m = \boldsymbol{v}_m + \Delta \boldsymbol{v}_{Scul_m}$ ▷ *vector that contains the integration of delta velocity terms between two m computer cycle indices with no loss of accuracy.*

5.4 Velocity Increments Transformation

The velocity increments (dv_x, dv_y, dv_z) output from the sculling compensation algorithm are described in the body frame. In order to perform the velocity integration in the navigation coordinate frame, it is essential that we transform their values to the NED frame. This can be easily done with help of the direction-cosine-matrix C_b^n as shown in Algorithm 4.

Algorithm 4 Velocity increment transformation.

Input: (dv_x, dv_y, dv_z) vector of integrated delta velocity terms at low-speed cycle index m.

Output:(dv_n, dv_e, dv_d) vector of integrated delta velocity terms in navigation frame.

$dv_n = c_{11}dv_x + c_{12}dv_y + c_{13}dv_z$

$dv_e = c_{21}dv_x + c_{22}dv_y + c_{23}dv_z$

$dv_d = c_{31}dv_x + c_{32}dv_y + c_{33}dv_z$

$dv_d = dv_d + g.dT$ ▷ *Sensed gravity component removal.*

5.5 Quaternion Integration

Quaternion integration deals with the determination of the quaternion between the body and the navigation frame. A primary advantage of using the quaternion technique lies in the fact that only four unknowns are necessary for calculation of the transformation matrix, while the direction cosine method requires nine. The quaternion can also be expressed as a 4x4 matrix. Thus

$$Q = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \\ -q_1 & q_0 & -q_3 & q_2 \\ -q_2 & q_3 & q_0 & -q_1 \\ -q_3 & -q_2 & q_1 & q_0 \end{bmatrix}, \tag{51}$$

where, as before, q_0, q_1, q_2, q_3 are quaternion components.

It can be shown that the quaternion analog of Puasson equation (see Section 6 (64)) has the form

$$\dot{Q} = \frac{1}{2} Q[w\times], \tag{52}$$

where $[w\times]$ is the skew-symmetric form of the angular velocity vector $\boldsymbol{w}$. The recurrent solution of the above equation can be determined (to first order) as

$$Q_{k+1} = Q_k + \frac{1}{2} Q_k[w\times]dT, \tag{53}$$

or

$$Q_{k+1} = Q_k(I + \frac{1}{2}[w\times]dT) = Q_k d\Lambda, \tag{54}$$

where dT is the sampling period and $dQ = (I + \frac{1}{2}[w\times]dT)$ is usually called the update quaternion. It is the quaternion of a small rotation that can be represented using Eq. 9 as follows

$$\begin{aligned} d\Lambda &= d\lambda_0 + d\lambda_1 i + d\lambda_2 j + d\lambda_3 k\,, \\ d\lambda_0 &= \cos\frac{\|d\Phi\|}{2}\,, \\ d\lambda_1 &= \frac{d\phi_x}{\|d\Phi\|}\sin\frac{\|d\Phi\|}{2}\,, \\ d\lambda_2 &= \frac{d\phi_y}{\|d\Phi\|}\sin\frac{\|d\Phi\|}{2}\,, \\ d\lambda_3 &= \frac{d\phi_z}{\|d\Phi\|}\sin\frac{\|d\Phi\|}{2}\,. \end{aligned} \tag{55}$$

Substituting in Eq. 54 the expression obtained is:

$$Q_{k+1} = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \\ -q_1 & q_0 & -q_3 & q_2 \\ -q_2 & q_3 & q_0 & -q_1 \\ -q_3 & -q_2 & q_1 & q_0 \end{bmatrix} * \begin{bmatrix} d\lambda_0 & d\lambda_1 & d\lambda_2 & d\lambda_3 \\ -d\lambda_1 & d\lambda_0 & -d\lambda_3 & d\lambda_2 \\ -d\lambda_2 & d\lambda_3 & d\lambda_0 & -d\lambda_1 \\ -d\lambda_3 & -d\lambda_2 & d\lambda_1 & d\lambda_0 \end{bmatrix}. \tag{56}$$

But $\sin\frac{\|d\Phi\|}{2}$ and $\cos\frac{\|d\Phi\|}{2}$ can be approximated using a third order expansion of the Taylor series:

$$\sin x \approx x - \frac{x^3}{3!} + \frac{x^5}{5!}, \quad \cos x \approx 1 - \frac{x^2}{2!} + \frac{x^4}{4!}\,. \tag{57}$$

The series expansion of Eq. 55 gives the following formula for the quaternion components:

$$\begin{aligned} d\lambda_0 &= 1 - \frac{\|d\Phi\|^2}{8} + \frac{\|d\Phi\|^4}{384}\,, \\ d\lambda_1 &= r d\phi_x\,, \\ d\lambda_2 &= r d\phi_y\,, \\ d\lambda_3 &= r d\phi_z\,, \end{aligned} \tag{58}$$

where $r = \frac{1}{2} - \frac{\|d\Phi\|^2}{48} + \frac{\|d\Phi\|^4}{3840}$. Substituting (58) into Eq. 51 we obtain Algorithm 5.

Algorithm 5 Efficient quaternion integration.

Input: De-biased angular increments $(d\phi_x, d\phi_y, d\phi_z)$.
Output: Quaternion vector $\boldsymbol{q} = (q_0, q_1, q_2, q_3)$.

$D2 = d\phi_x * d\phi_x + d\phi_x * d\phi_x + d\phi_x * d\phi_x$ ▷ *norm of rotation vector squared*
$D4 = D2 * D2$
$s = 0.5 - \frac{D2}{48} + \frac{D4}{3840}$
$c = -\frac{D2}{8} + \frac{D4}{384}$
$s_x = s * d\phi_x,\ s_y = s * d\phi_y,\ s_z = s * d\phi_z$
$dq_0 = c * q_0 - s_x * q_1 - s_y * q_2 - s_z * q_3$
$dq_1 = c * q_0 + s_x * q_1 + s_z * q_2 - s_y * q_3$
$dq_2 = c * q_0 + s_y * q_1 = s_x * q_2 - s_z * q_3$
$dq_3 = c * q_0 - s_z * q_1 - s_y * q_2 - s_x * q_3$
$q_0 = q_0 + dq_0$
$q_1 = q_1 + dq_1$
$q_2 = q_2 + dq_2$
$q_3 = q_3 + dq_3$

5.6 Normalizing Quaternion Parameters

According to the quaternion properties, its norm should be always equal to one, which means:

$$q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1\,. \tag{59}$$

But unfortunately, the above condition can be violated due to calculation errors or rounding approximations. In order to remove this effect it is necessary to apply a normalization procedure.Since $q_0^2 + q_1^2 + q_2^2 + q_3^2 \approx 1$ then we have:

$$\Delta = 1 - q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1 - \|\boldsymbol{q}\|^2\,, \tag{60}$$

is a very small number. Then normalizing each quaternion parameter by dividing by $\sqrt{1-\Delta}$, and expanding using a Taylor's series formula we obtain:

$$\hat{q}_{norm} = \frac{q}{\sqrt{1-\Delta}} \approx q(1 + \frac{\Delta}{2}) = q * 0.5(3 - \|\boldsymbol{q}\|^2)\,. \tag{61}$$

Algorithm 6 Efficient quaternion normalization.

Input: Quaternion vector $\boldsymbol{q} = (q_0, q_1, q_2, q_3)$.
Output: Normalized vector $\boldsymbol{q} = (q_0, q_1, q_2, q_3)$.

$q_{00} = q_0 * q_0$;
$q_{11} = q_1 * q_1$;
$q_{22} = q_2 * q_2$;
$q_{33} = q_3 * q_3$;
$qq = q_{00} + q_{11} + q_{22} + q_{33}$;
$q_{cor} = 0.5 * (3.0 - qq)$
$q_0 = q_0 * q_{cor}$
$q_1 = q_1 * q_{cor}$
$q_2 = q_2 * q_{cor}$
$q_3 = q_3 * q_{cor}$

5.7 Direction Cosine Matrix Computation

The quaternions compose a four-element unit vector (q_0, q_1, q_2, q_3) obtained from the quaternion integration step. They can be efficiently used to find the elements of the 3-by-3 Direction Cosine Matrix (DCM). The outputted DCM performs the coordinate transformation of a vector in body axes to a vector in local-level navigation frame axes. The following algorithm will be used in the embedded processor to find the 9-elements of the DCM:

$$C_b^n = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix}\,. \tag{62}$$

Algorithm 7 Efficient direction cosine matrix computation.

Input: Normalized vector $\boldsymbol{q} = (q_0, q_1, q_2, q_3)$.
Output: Direction-cosine-Matrix C_b^n

$q_{00} = q_0 * q_0$;
$q_{11} = q_1 * q_1$;
$q_{22} = q_2 * q_2$;
$q_{33} = q_3 * q_3$;
$q_{01} = q_0 * q_1$;
$q_{02} = q_0 * q_2$;
$q_{03} = q_0 * q_3$;
$q_{12} = q_1 * q_2$;
$q_{13} = q_1 * q_3$;
$q_{23} = q_2 * q_3$;
$c_{11} = q_{00} + q_{11} - q_{22} - q_{33}$;
$c_{12} = (q_{12} - q_{03}) * 2$;
$c_{13} = (q_{13} + q_{02}) * 2$;
$c_{21} = (q_{12} + q_{03}) * 2$;
$c_{22} = q_{00} - q_{11} + q_{22} - q_{33}$;
$c_{23} = (q_{23} - q_{01}) * 2$;
$c_{31} = (q_{13} - q_{02}) * 2$;
$c_{32} = (q_{23} + q_{01}) * 2$;
$c_{33} = q_{00} - q_{11} - q_{22} + q_{33}$;

Algorithm 8 Efficient roll, pitch, and heading computation.

Input: Direction-cosine-Matrix C_b^n
Output: Roll ϕ, Pitch θ, Heading ψ.

Compute Roll
$C1 = c_{32}$;
$C2 = c_{33}$;
$ANGLE = atan(C1, C2)$;[5]
if C2 > 0 **then**
  $ROLL = ANGLE$;
end if
if $C2 < 0$ **then**
  if $C1 > 0$ **then**
    $ROLL = ANGLE + \pi$;
  else
    $ROLL = ANGLE - \pi$;
  end if
end if
if $C2 == 0$ **then**
  if $C1 \geq 0$ **then**
    $ROLL = \frac{\pi}{2}$;
  else
    $ROLL = \frac{-\pi}{2}$;
  end if
end if
if $ROLL \geq \pi$ **then**
  $ROLL = ROLL - \frac{\pi}{2}$;
end if
if $ROLL \leq -\pi$ **then**
  $ROLL = ROLL + \frac{\pi}{2}$;
end if
Compute Pitch
$C1 = c_{31}$;
$C11 = C1 * C1$;
if $C11 \geq 1$ **then**
  $C11 = 1$; ▷ *should never be more than 1.*
end if
$C2 = \sqrt{1 - C11}$;
$ANGLE = atan(C1, C2)$;
$PITCH = ANGLE$;
Compute Heading
$C1 = c_{21}$;
$C2 = c_{11}$;
if $C2 == 0$ **then**
  if $C1 \geq 0$ **then**
    $HEADING = \frac{\pi}{2}$;
  else
    $HEADING = \frac{3\pi}{2}$;
  end if
else
  $ANGLE = atan(C1, C2)$;
  if $C2 > 0$ **then**
    $HEADING = ANGLE$;
  else
    if $C1 \geq 0$ **then**
      $HEADING = ANGLE + \pi$;
    else
      $HEADING = ANGLE - \pi$;
    end if
  end if
end if
if $HEADING < 0$ **then**
  $HEADING = HEADING + \frac{\pi}{2}$;
end if

For a fast algorithm for calculating the arc-tangent function $atan(\cdot, \cdot)$, see Appendix.

5.7.1 Roll, Pitch, and Heading Calculation (Euler Angles)

The Euler angles are three angles introduced by Leonhard Euler to describe the orientation of a rigid body with respect to a fixed coordinate system. Leonard Euler (1707-1783) was one of the giants inn mathematics [30]. Euler stated and proved a theorem that states that:

Any two independent orthonormal coordinate frames can be related by a sequence of rotations (not more than three) about coordinate axes, where no two successive rotations may be about the same axis.

When we say that two independent frames are related, we mean that a sequence of rotations about successive coordinate axes will rotate the first frame into the second. The angle of rotation about a coordinate axis is called an Euler angle. A sequence of such rotations is often called a Euler angle sequence of rotations. The restriction stated in the above theorem that successive axes of rotations be distinct still permits at least 12 Euler angle sequences. The sequence **xzy** means a rotation about the **x**-axis, followed by a rotation about the new **z**-axis, followed by a rotation about the newer **y**-axis.

We are specifically interested in the well-known Euler sequence called the Aerospace sequence. This sequence (**zyx**) is commonly used in aircraft and aerospace applications. For example, a primary flight instrument used in

aircrafts, continuously relates the orientation of the aircraft to the local-level n-frame mentioned above.

From the n-frame,first a rotation through the angle ψ about the **z**-axis defines the aircraft heading. This is followed by a rotation about the new **y**-axis through an angle θ which defines the aircraft pitch. Finally, the aircraft roll angle ϕ, is a rotation about the newest **x**-axis. These three Euler angle rotations relate the body coordinate frame of the aircraft to the local-level n-frame.

6 Principles of Inertial Navigation

6.1 Navigation Equations

It is desirable to formulate the navigation equations in the earth-centered, earth-fixed frame (e-frame), since usually the measurements of the GNSS receiver are given in the e-frame. But usually we are more comfortable in dealing with n-frame coordinates since it is more trivial to deal with the *North-East-Down* directions. Recall that the coordinate directions of the n-frame are defined by the local horizon and by the vertical, and centered on the vehicle center of gravity (cog). Strictly speaking, no horizontal motion takes place in this frame since it is attached and fixed to the vehicle. Therefore, the navigation equations are not coordinatized in the n-frame because no horizontal motion takes place in this frame. Nevertheless, we will still refer to the n-frame coordinatization of the navigation equations as an Earth-referenced formulation in which the velocity components are transformed along the n-frame coordinate directions. This concept is rarely discussed in the literature and usually is a source of confusion.

A vector in the e-frame (navigation frame) has coordinates in the i-frame (inertial frame) given by

$$\boldsymbol{x}^i = C_e^i \boldsymbol{x}^e \,, \tag{63}$$

where C_e^i is the transformation matrix from the e-frame to the i-frame. The time derivative of this matrix is given by [31]

$$\dot{C}_e^i = C_e^i \Omega_{ie}^e \,, \tag{64}$$

where Ω_{ie}^e denotes a skew-symmetric matrix with elements from $\boldsymbol{\omega}_{ie}^e = (\omega_1, \omega_2, \omega_3)$[5] is then given by

$$\Omega_{ie}^e = \left[\boldsymbol{\omega}_{ie}^e \times\right] = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} . \tag{65}$$

We also need the second time derivative which from Eq. 64 and the chain rule for differentiation is given by

$$\ddot{C}_e^i = C_e^i \dot{\Omega}_{ie}^e + C_e^i \Omega_{ie}^e \Omega_{ie}^e \,. \tag{66}$$

Now differentiating (63) twice with respect to time yields

$$\begin{aligned} \ddot{\boldsymbol{x}}^i &= \ddot{C}_e^i \boldsymbol{x}^e + 2\dot{C}_e^i \dot{\boldsymbol{x}}^e + C_e^i \ddot{\boldsymbol{x}}^e \\ &= C_e^i \ddot{\boldsymbol{x}}^e + 2C_e^i \Omega_{ie}^e \dot{\boldsymbol{x}}^e + C_e^i (\dot{\Omega}_{ie}^e + \Omega_{ie}^e \Omega_{ie}^e) \boldsymbol{x}^e \,. \end{aligned} \tag{67}$$

Solving for $\ddot{\boldsymbol{x}}^n$ and combining with $\ddot{\boldsymbol{x}}^i = \boldsymbol{g}^i + \boldsymbol{a}^i$ gives the system dynamics for position in the e-frame:

$$\ddot{\boldsymbol{x}}^e = -2\Omega_{ie}^e \dot{\boldsymbol{x}}^e - (\dot{\Omega}_{ie}^e + \Omega_{ie}^e \Omega_{ie}^e) \boldsymbol{x}^e + \boldsymbol{g}^e + \boldsymbol{a}^e \,. \tag{68}$$

Since the earth has a constant angular velocity with respect to the inertial frame, then $\dot{\Omega}_{ie}^e = 0$ and we obtain

$$\ddot{\boldsymbol{x}}^e = -2\Omega_{ie}^e \dot{\boldsymbol{x}}^e - \Omega_{ie}^e \Omega_{ie}^e \boldsymbol{x}^e + \boldsymbol{g}^e + \boldsymbol{a}^e \,. \tag{69}$$

We can transform the navigation equation above into the n-frame merely by substituting $\dot{\boldsymbol{x}}^e = C_n^e \boldsymbol{v}^n$ on the right-hand side of Eq. 69

$$\frac{d}{dt} C_n^e \boldsymbol{v}^n = C_n^e \left(\frac{d}{dt} \boldsymbol{v}^n + \Omega_{en}^n \boldsymbol{v}^n\right) , \tag{70}$$

and on the right hand side of Eq. 69 we use the formula $\Omega_{ie}^n = C_e^n \Omega_{ien}^e C_n^e$, and the result is:

$$\frac{d}{dt} \boldsymbol{v}^n = \boldsymbol{a}^n - (2\Omega_{ie}^n + \Omega_{en}^n) \boldsymbol{v}^n + \boldsymbol{g}^n - C_e^n \Omega_{ie}^e \Omega_{ie}^e \boldsymbol{x}^e \,. \tag{71}$$

The last two terms are, respectively, the gravitational vector and the centrifugal acceleration due to the Earth's rotation, coordinatized in the n-frame. Together they define the gravity vector:

$$\bar{\boldsymbol{g}}^n = \boldsymbol{g}^n - C_e^n \Omega_{ie}^e \Omega_{ie}^e \boldsymbol{x}^e \,. \tag{72}$$

The distinction between the terms gravitation and gravity, refers to the difference between the acceleration due to mass attraction, alone, and the total acceleration, gravitational and centrifugal, that is measured at a fixed point on the rotating earth. Gravity has a direction that coincides with the direction of a plumb line at any given point in space. The direction of a plumb line coincides with the direction of a string to which a freely suspended weight, or plumb bob, is attached.

Finally, by writing $\Omega_{en}^n = \Omega_{in}^n + \Omega_{ei}^n = \Omega_{in}^n - \Omega_{ie}^n$[6] and by manipulating the subscripts, we also obtain

$$2\Omega_{ie}^n + \Omega_{en}^n = \Omega_{in}^n + \Omega_{ie}^n \,. \tag{73}$$

Substituting (73) and (72) into Eq. 71, the desired form of the n-frame navigation equations becomes:

$$\frac{d}{dt} \boldsymbol{v}^n = \boldsymbol{a}^n - (\Omega_{in}^n + \Omega_{ie}^n) \boldsymbol{v}^n + \bar{\boldsymbol{g}}^n \,. \tag{74}$$

The components of the Earth-referenced velocity, $\boldsymbol{v}^n$, the sensed acceleration, $\boldsymbol{a}^n$, and the gravity vector, $\bar{\boldsymbol{g}}^n$, can be

[5] $\boldsymbol{\omega}_{ie}^e$ = the angular velocity of the e-frame with respect to the i-frame, with coordinates in the e-frame. Since the three axis of the e-frame are aligned with the Earth's spin axis, then $\boldsymbol{\omega}_{ie}^e = (0, 0, \omega_e)$, where ω_e is the angular rate of the Earth's rotation.

[6] Relative angular velocities can be added component-wise and they satisfy commutativity.

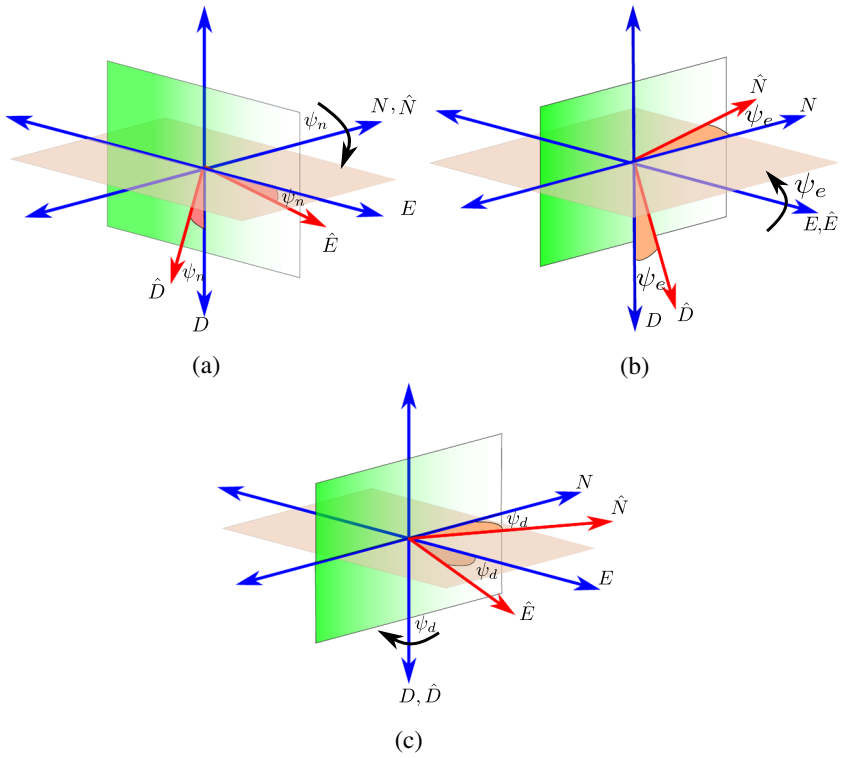

Figure 16 The platform frame $\hat{N}\hat{E}\hat{D}$ is a virtual frame that is slightly misaligned from the true navigation frame. It is mainly created for the derivation of the error equations. It is only recognized by the INS on-board computer and it results from the erroneous gyroscope sensors that are integrated to give this false navigation frame.

described by their north, east, and down components in the n-frame as follows:

$$\boldsymbol{v}^n = \begin{bmatrix} v_n \\ v_e \\ v_d \end{bmatrix}, \quad \boldsymbol{a}^n = \begin{bmatrix} a_n \\ a_e \\ a_d \end{bmatrix}, \quad \bar{\boldsymbol{g}}^n = \begin{bmatrix} \bar{g}_n \\ \bar{g}_e \\ \bar{g}_d \end{bmatrix}. \tag{75}$$

6.2 Error Dynamic Equations in the *n*-Frame

The equations of motion depict the time development of the user's position, speed, and attitude under perfect conditions. The way in which errors proliferate in an INS can be computed by applying a first-order Taylor series development (linearization), or perturbation investigation, to the equations of motion derived in the previous subsection. The coordinatization of the error dynamics in the n-frame represents the traditional and most intuitive scheme of analyzing INS errors.

6.2.1 Orientation Error Dynamics in the *n*-frame

The perturbation $\delta\boldsymbol{a}^n$ is interpreted as the error in the expression in the n-frame of the sensed acceleration. It represents not only accelerometer errors but also orientation errors that are committed when transforming sensed accelerations from the sensor frame (s-frame) to the n-frame. Taking differentials of the relation $\boldsymbol{a}^n = C_s^n \boldsymbol{a}^s$, we obtain:

$$\delta\boldsymbol{a}^n = \delta C_s^n \boldsymbol{a}^s + C_s^n \delta\boldsymbol{a}^s . \tag{76}$$

The differential δC_s^n is caused by errors in the orientation of the s-frame with respect to the n-frame. It is convenient to represent δC_s^n in terms of small error angles, one for each of the n-frame axes: $\boldsymbol{\psi^n} = (\psi_n, \psi_e, \psi_d)^T$. This can be represented in the equivalent form of a skew-symmetric matrix:

$$\boldsymbol{\Psi}^n = \begin{bmatrix} 0 & -\psi_d & \psi_e \\ \psi_d & 0 & -\psi_n \\ -\psi_e & \psi_n & 0 \end{bmatrix}. \tag{77}$$

Since ψ_n, ψ_e and ψ_d represent small angle rotation errors, the transformation matrix from the true n-frame to the erroneously computed n-frame (inside the INS computer), can be written as $I - \boldsymbol{\Psi}^n$. This can be easily achieved by substituting the small angle errors (see Fig. 16) inside the DCM (4) on page 7, and approximating it to first order. Therefore, the computed transformation may be represented as a sequence of two transformations comprising first the true transformation from the body frame (n-frame or s-frame) to the erroneously computed n-frame followed by another transformation from the true n-frame to the erroneously computed n-frame:

$$\hat{C}_s^n = (I - \boldsymbol{\Psi}^n) C_s^n . \tag{78}$$

It is now clear that:

$$\delta C_s^n = \hat{C}_s^n - C_s^n = -\boldsymbol{\Psi}^n C_s^n . \tag{79}$$

Substituting (79) in Eq. 76, we obtain

$$\begin{aligned}\delta \boldsymbol{a}^n &= C_s^n \delta \boldsymbol{a}^s - \boldsymbol{\Psi}^n C_s^n \boldsymbol{a}^s \\ &= C_s^n \delta \boldsymbol{a}^s + \boldsymbol{a}^n \times \boldsymbol{\psi}^n \,.\end{aligned} \tag{80}$$

We now establish the dynamic behavior of the error angles $\boldsymbol{a}^s$ in the form of a differential equation. Taking the differential of $\dot{C}_s^n = C_s^n \Omega_{ns}^s$

$$\begin{aligned}\delta \dot{C}_s^n &= \delta (C_s^n \Omega_{ns}^s) \\ &= \delta C_s^n \Omega_{ns}^s + C_s^n \delta \Omega_{ns}^s \,,\end{aligned} \tag{81}$$

where the perturbation in angular rate, $\delta \Omega_{ns}^s$, is interpreted as the error in the corresponding computed value, denoted by $\hat{\Omega}_{ns}^s$:

$$\delta \Omega_{ns}^s = \hat{\Omega}_{ns}^s - \Omega_{ns}^s \,. \tag{82}$$

Differentiating the second line of Eq. 80 with respect to time and setting the result equal to the right side of Eq. 81, we get

$$-\dot{\boldsymbol{\Psi}}^n C_s^n - \boldsymbol{\Psi}^n C_s^n \Omega_{ns}^s = \delta C_s^n \Omega_{ns}^s + C_s^n \delta \Omega_{ns}^s \,. \tag{83}$$

Substituting for $\boldsymbol{\Psi}^n C_s^n$ and solving for $\dot{\boldsymbol{\Psi}}^n$ yields

$$\dot{\boldsymbol{\Psi}}^n = -C_s^n \delta \Omega_{ns}^s C_n^s \,, \tag{84}$$

in terms of vectors, it is easily verified that this is equivalent to

$$\dot{\boldsymbol{\psi}}^n = -C_s^n \delta \boldsymbol{\omega}_{ns}^s \,, \tag{85}$$

where $\delta \boldsymbol{\omega}_{ns}^s$ is the error in the rotation rate of the s-frame with respect to the n-frame. For small vehicle velocities, the angular velocity of the navigation frame is negligible,[7] consequently, $\boldsymbol{\omega}_{ns}^s \approx \boldsymbol{\omega}_{is}^s$, where $\boldsymbol{\omega}_{is}^s$ is the angular velocity vector delivered by the gyroscope.[8]

Equation 85 can be discovered directly with no need for any rigorous derivation, simply by recognizing that, the errors in the orientation angular rates, $\dot{\psi}_n, \dot{\psi}_e$, and $\dot{\psi}_d$ are nothing but the transformations of the gyroscope angular rate errors, $\delta \omega_{Gx}^s, \delta \omega_{Gy}^s$, and $\delta \omega_{Gz}^s$,[9] (gyroscope biases in practice) from the body frame to the navigation frame, which when expanded can be written in the form (for convenience the body frame and the sensor frame are considered coincident):

$$\begin{bmatrix} \dot{\psi}_n \\ \dot{\psi}_e \\ \dot{\psi}_d \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix} \begin{bmatrix} \delta \omega_{Gx}^s \\ \delta \omega_{Gy}^s \\ \delta \omega_{Gz}^s \end{bmatrix} . \tag{86}$$

In Eq. 86 the 3×3 matrix can be determined by using Algorithm 7, and the $\delta\omega$ terms represent the gyroscope biases along the three axes **x**, **y**, and **z** of the body frame.

Since we are interested in implementing our algorithms on an embedded processor we need to discretize (86). With the help of BOX B on page 9, we obtain the following discretized version:

$$\begin{bmatrix} \psi_n\langle k+1\rangle \\ \psi_e\langle k+1\rangle \\ \psi_d\langle k+1\rangle \end{bmatrix} = \begin{bmatrix} \psi_n\langle k\rangle \\ \psi_e\langle k\rangle \\ \psi_d\langle k\rangle \end{bmatrix} + \begin{bmatrix} c_{11}\langle k\rangle dT & c_{12}\langle k\rangle dT & c_{13}\langle k\rangle dT \\ c_{21}\langle k\rangle dT & c_{22}\langle k\rangle dT & c_{23}\langle k\rangle dT \\ c_{31}\langle k\rangle dT & c_{32}\langle k\rangle dT & c_{33}\langle k\rangle dT \end{bmatrix} \begin{bmatrix} \delta \omega_{Gx}^s\langle k\rangle \\ \delta \omega_{Gy}^s\langle k\rangle \\ \delta \omega_{Gz}^s\langle k\rangle \end{bmatrix} . \tag{87}$$

Box C: State-Space Discretization

A state-space representation of a general continuous time system is written as $\dot{\boldsymbol{x}}(t) = A(t)\boldsymbol{x}(t) + B(t)\omega(t)$. At a certain time epoch $t = t_k$ it gives [32]:

$$\dot{\boldsymbol{x}}(t_k) = A(t_k)\boldsymbol{x}(t_k) + B(t_k)\omega(t_k) \tag{89}$$

For a short sampling period, $dT = t_{k+1} - t_k$, one can write:

$$\dot{\boldsymbol{x}}(t_k) \approx \frac{\boldsymbol{x}\langle k+1\rangle - \boldsymbol{x}\langle k\rangle}{dT} \tag{90}$$

where $\boldsymbol{x}\langle k+1\rangle = \boldsymbol{x}(t_{k+1})$ and $\boldsymbol{x}\langle k\rangle = \boldsymbol{x}(t_k)$. Substituting (90) in (89), one can get:

$$\begin{aligned}\boldsymbol{x}\langle k+1\rangle &= (I + A(t_k)dT)\boldsymbol{x}\langle k+1\rangle + B(t_k)\boldsymbol{w}\langle k\rangle dT \\ &= \Phi(t_{k+1}, t_k)\boldsymbol{x}\langle k\rangle + G(t_{k+1}, t_k)\boldsymbol{w}\langle k\rangle\end{aligned} \tag{91}$$

where $\Phi(t_{k+1}, t_k)$ is a state transition matrix that propagates the system state $\boldsymbol{x}_k$ one time step; $G(t_{k+1}, t_k)$ is an input matrix that plays the role of coloring the input system noise $\boldsymbol{w}_k$, which is usually white with zero mean.

Nb: This discretization is only a first order approximation, but this is compromised by the fact that the uncertainties in the discrete model will be hidden in the coloured white noise vector of the system model equation.

6.2.2 Velocity Error Dynamics in the *n*-Frame

The objective in the n-frame coordinatization is to formulate the error dynamics with respect to the geodetic coordinates

[7] From pure geometric observations of Fig. 13 in Box B on page 9, it is evident that the vehicle velocity in the north and east direction are related to latitude and longitude rate respectively through, $\dot{\phi} = \boldsymbol{v}_n / R$ and $\dot{\lambda} = \boldsymbol{v}_e / R \cos\phi$ where $R \approx 6370$ km is the radius of the Earth.

[8] Gyroscopes deliver vehicle angular velocities with respect to the inertial frame, and since in a strapdown mechanization these inertial sensors are fixed to the vehicle body, their readings are referenced to this frame. (Usually the sensor and body frame are considered aligned to each other with a probable offset between their origins).

[9] The Gyro bias vector adapted in this manuscript is written as $\delta \boldsymbol{\omega}_G = (\delta \omega_{Gx}^s, \delta \omega_{Gy}^s, \delta \omega_{Gz}^s)$ (notation adopted from [28]).

(ϕ, λ, h). We write the perturbation from the compact form of the velocity navigation (74).

$$\begin{aligned}\frac{d}{dt}\delta\boldsymbol{v}^n = &-\delta(\Omega_{in}^n + \Omega_{ie}^n)\boldsymbol{v}^n - (\Omega_{in}^n + \Omega_{ie}^n)\delta\boldsymbol{v}^n \\ &+ \delta\boldsymbol{a}^n + \bar{\boldsymbol{\Gamma}}^n\delta\boldsymbol{p}^n + \delta\bar{\boldsymbol{g}}^n\,.\end{aligned} \tag{88}$$

Of note in the above equation are the errors in the computation of the local gravity vector, $\bar{\boldsymbol{g}}$. This vector is not a constant, and varies as a function of location as identified in the matrix of gravity gradients, $\bar{\boldsymbol{\Gamma}}^n = \partial\bar{\boldsymbol{g}}^n/\partial\boldsymbol{p}^n$. Position errors, $\delta\boldsymbol{p}^n$, lead to errors in $\bar{\boldsymbol{g}}$ which, in turn, lead to velocity errors. Using Eqs. 12 and 13 in Box B on page 9 it can be easily shown that:

$$\delta(\Omega_{in}^n + \Omega_{ie}^n) = \begin{bmatrix} 0 & \delta\dot{\lambda}\sin\phi + (\dot{\lambda}+2\omega_e)\cos\phi\delta\phi & -\delta\dot{\phi} \\ -\delta\dot{\lambda}\sin\phi - (\dot{\lambda}+2\omega_e)\cos\phi\delta\phi & 0 & -\delta\dot{\lambda}\cos\phi + (\dot{\lambda}+2\omega_e)\sin\phi\delta\phi \\ \delta\dot{\phi} & \delta\dot{\lambda}\cos\phi - (\dot{\lambda}+2\omega_e)\sin\phi\delta\phi & 0 \end{bmatrix}. \tag{89}$$

Since our intention is to give a hands-on experience for the reader of this manuscript, it is instructive to provide the approximations usually applied in commercial integrated navigation systems to the error equations. Typically, for general applications where n-frame velocities don't exceed, say, 120 m/s, the term $\delta(\Omega_{in}^n + \Omega_{ie}^n)\boldsymbol{v}^n$ (in view (89) is lower than 10^{-5}m/s^2. This can be neglected in case of low-cost MEMS are being used in the INS, since the the acceleration errors, $\delta\boldsymbol{a}^n$, due to accelerometer biases and gyro drift are much higher. It is important to stress that gyro drifts lead to an erroneous transformation of sensed acceleration from body to navigation frame as shown later in this section. For small navigational velocities, the angular rate of the vehicle Ω_{en}^n, is relatively much smaller than the angular rate of the Earth Ω_{ie}^n, which is already lower than the noise level found in typical MEMS sensors. Thus for an elementary analysis we may consider the second term on the right hand side to be zero in the error dynamics of Eq. 88. Neglecting the gravity related[10] terms in Eq. 88 we obtain:

$$\frac{d}{dt}\delta\boldsymbol{v}^n = \delta\boldsymbol{a}^n\,. \tag{90}$$

Taking the second part of Eq. 80 and using $\boldsymbol{a}^n \approx (a_n, a_e, -\bar{g})$ since in case of low vehicle velocity in the navigation frame the accelerometer in the down direction is overshadowed by the gravity vector, $a_d \approx -\bar{g}$, we obtain:

$$\begin{aligned}\tfrac{d}{dt}\delta\boldsymbol{v}^n &= C_s^n\delta\boldsymbol{a}^s + \boldsymbol{a}^n \times \boldsymbol{\psi}^n \\ &= \begin{bmatrix}\delta a_{An}\\ \delta a_{Ae}\\ \delta a_{Ad}\end{bmatrix} + \begin{bmatrix} a_n\\ a_e\\ -\bar{g}\end{bmatrix} \times \begin{bmatrix}\psi_n\\ \psi_e\\ \psi_d\end{bmatrix} \\ &= \begin{bmatrix}\delta a_{An}\\ \delta a_{Ae}\\ \delta a_{Ad}\end{bmatrix} + \begin{bmatrix} a_e\psi_d + \bar{g}\psi_e\\ -a_n\psi_d - \bar{g}\psi_n\\ -a_n\psi_e - a_e\psi_n\end{bmatrix}.\end{aligned} \tag{91}$$

where we have used the notation $\delta\boldsymbol{a}_A = C_s^n\delta\boldsymbol{a}^s$ (notation adopted from [33]) to identify the accelerometer error vector produced in the body frame, but projected on the navigation frame.

It is instructive to prove the velocity error dynamics, simply by relying on insights into Fig. 17. We shall re-prove the error dynamics for the first component in Eq. 91, δv_n. The interested reader is invited to prove the error dynamic equations of the other terms. From a geometrical viewpoint of Fig. 17a, we have:

$$\hat{a}_n = a_n\cos\psi_e - a_d\sin\psi_e\,. \tag{92}$$

For $\psi_e \approx 0$ and $a_d \approx -\bar{g}$, Eq. 92 becomes:

$$\hat{a}_n = a_n + \bar{g}\psi_e\,, \tag{93}$$

which implies that:

$$\hat{a}_n - a_n = \delta a_n = \bar{g}\psi_e\,. \tag{94}$$

Similarly, looking at Fig. 17b, we have:

$$\hat{a}_n = a_n\cos\psi_d + a_e\sin\psi_d \tag{95}$$

For $\psi_d \approx 0$, Eq. 95 becomes to first order:

$$\hat{a}_n = a_n + a_e\psi_d\,, \tag{96}$$

which implies that:

$$\hat{a}_n - a_n = \delta a_n = a_e\psi_d\,. \tag{97}$$

So the total error in a_n, is the superposition of the terms produced by ψ_d and ψ_e, thus we have:

$$\delta a_n = a_e\psi_d + \bar{g}\psi_e\,. \tag{98}$$

Adding to Eq. 98 the error in acceleration caused by the bias term δa_{An} we obtain:

$$\delta a_n = \frac{d}{dt}\delta\boldsymbol{v}^n = \delta a_{An} + a_e\psi_d + \bar{g}\psi_e\,, \tag{99}$$

which is nothing but the first row in Eq. 91.[11]

[10] It can be showed that for relatively short time of navigation 1-2 hr the gravity gradient terms and the errors in computing the gravity vector are negligible.

[11] Due to continuity of the v_n term, we have $\delta a_n = \delta(\frac{d}{dt}v_n) = \frac{d}{dt}\delta v_n$

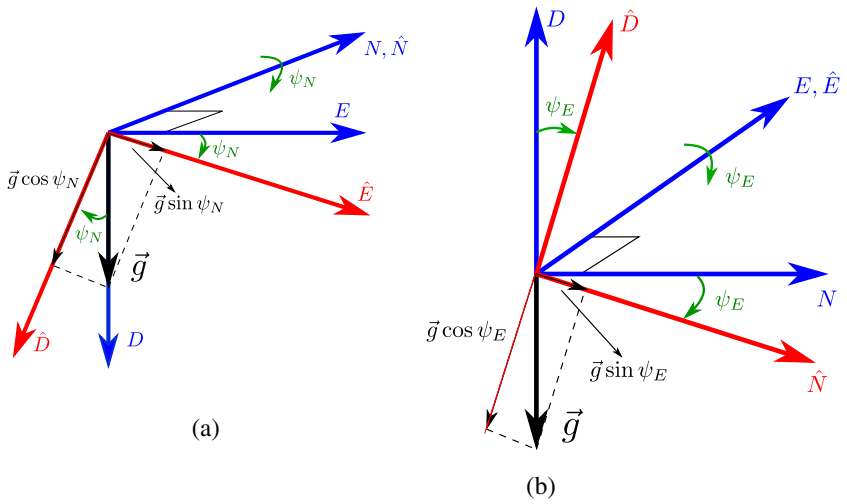

Figure 17 Attitude measurement error produces very predictable errors in velocity estimates . If you can measure velocity and position errors (say, with GPS), then you can figure out your attitude error using a combination of GPS and inertial sensors.

Discretizing (91) with the aid of BOX C, we can easily obtain the following discrete velocity error equations

$$\begin{bmatrix} \delta v_n\langle k+1\rangle \\ \delta v_e\langle k+1\rangle \\ \delta v_d\langle k+1\rangle \end{bmatrix} = \begin{bmatrix} \delta v_n\langle k\rangle \\ \delta v_e\langle k\rangle \\ \delta v_d\langle k\rangle \end{bmatrix} + dT \begin{bmatrix} \delta a_{An}\langle k\rangle \\ \delta a_{Ae}\langle k\rangle \\ \delta a_{Ad}\langle k\rangle \end{bmatrix} + dT \begin{bmatrix} (a_e\langle k\rangle\psi_d\langle k\rangle + \bar{g}\psi_e\langle k\rangle) \\ (-a_n\langle k\rangle\psi_d\langle k\rangle - \bar{g}\psi_n\langle k\rangle) \\ (-a_n\langle k\rangle\psi_e\langle k\rangle - a_e\langle k\rangle\psi_n\langle k\rangle) \end{bmatrix}. \tag{100}$$

6.2.3 Position Error Dynamics in the *n*-Frame

Referring to footnote 7 on page 19 we can write:

$$\begin{aligned} \delta\dot{\phi} &= \tfrac{\delta v_n}{R}, \\ \delta\dot{\lambda} &= \tfrac{\delta v_e}{R\cos\phi} + \tfrac{v_e}{R}\tfrac{\sin\phi}{\cos^2\phi}\delta\phi, \end{aligned} \tag{101}$$

but since $\lambda = v_e/R\cos\phi$ and $\dot{\lambda} \approx 0$ (for small vehicle velocity), then Eq. 101 becomes:

$$\begin{aligned} \delta\dot{\phi} &= \tfrac{\delta v_n}{R}, \\ \delta\dot{\lambda} &= \tfrac{\delta v_e}{R\cos\phi}. \end{aligned} \tag{102}$$

The radius of the Earth, R, is taken to be constant neglecting the ellipsoidal nature of the Earth.

We should also include altitude h in our position error dynamic equations. To do so we can simply write:[12]

$$\delta\dot{h} = -\delta v_d. \tag{103}$$

In order to discretize (102) we approximate the derivatives by a finite difference, resulting in:

$$\begin{aligned} \delta\phi\langle k+1\rangle &= \delta\phi\langle k\rangle + dT\tfrac{\delta v_n\langle k+1\rangle}{R}, \\ \delta\lambda\langle k+1\rangle &= \delta\lambda\langle k\rangle + dT\tfrac{\delta v_e\langle k\rangle}{R\cos\phi\langle k\rangle}, \\ \delta h\langle k+1\rangle &= \delta h\langle k\rangle - dT\delta v_d\langle k\rangle. \end{aligned} \tag{104}$$

[12] The negative sign in error dynamics for the altitude channel is due to the fact that, it is traditional in GPS-INS fusion systems to take altitude h and the down axis in opposite directions.

7 Kalman Filter

The Kalman filter is an estimation strategy, instead of being a filter. The fundamental strategy was designed by R. E. Kalman in 1960 [34], and has been improved further by various researchers since. The filter refreshes the estimates of the state vector which is persistently changing. These estimates are then updated using a set of measurements which are subject to noise [35]. The measurements should be written in terms of the parameters estimated, yet the measurements at a given time need not contain adequate information to uniquely decide the values of the state vector at the time. This is related to the concept of observability of the system. It closely mimics the case of solving a set of equations where the number of unknown variables exceeds the number of equations.

The Kalman filter utilizes information of statistical properties of the system in order to get ideal estimates of the data available. It maintains a set of uncertainties about the estimates that is carried from one iteration to another. It also carries a measure of correlation between the errors in the estimates of the states from iteration to iteration.

The Kalman filter is an efficient algorithm from computing point of view, since it is a recursive algorithm that only processes the latest measurements and forgets the old ones. In contrast, non-recursive algorithms waits until all measurements are available before beginning any estimate which is time and memory consuming.

7.1 Components of the Kalman Filter

The calculation scheme of the Kalman filter algorithm is shown in Fig. 18. The Kalman filter has five major components:

- The state vector and its covariance matrix
- The system model

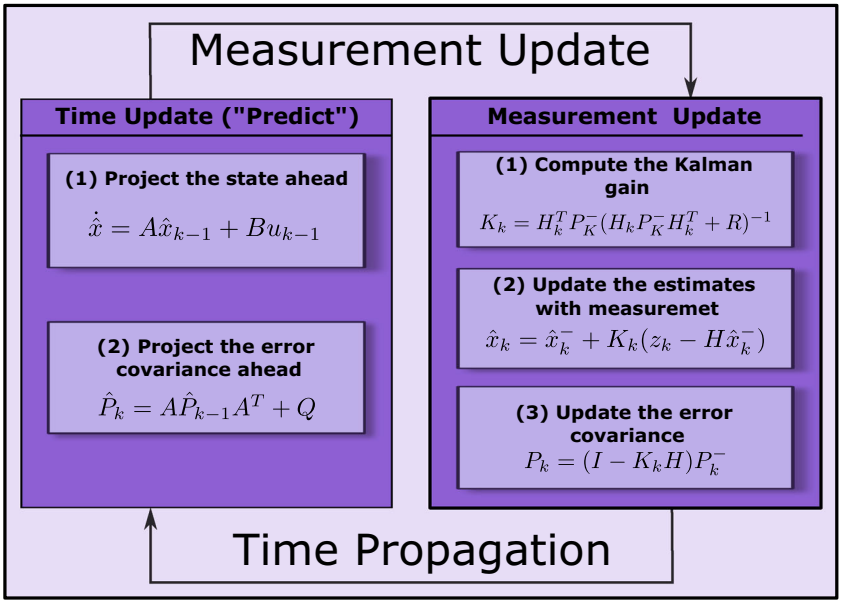

Figure 18 Key update equations of a Kalman filter.

- The measurement vector and its covariance
- The measurement model
- The algorithm

The state vector is a set of parameters that the filter estimates. It is usually composed of the position, velocity and other navigation states or their errors. In our demonstration of the Kalman filter algorithm, we will estimate the errors in the parameters of an INS system, $\delta(\cdot)$, instead of the parameters them self, $(\cdot)$. This implementation is called an error-state implementation. In contrast, when estimating absolute states of the system such as position, velocity, and orientation, the system is known as a total-state implementation. The error-state implementation separates the state into a "large" nominal state $\hat{x}$, and a "small" error state, δx, such that $x = \hat{x} + \delta x$. The error-state implementation can perform better due to the fact that the error dynamic equations we derived are linearized versions of their true equations (due to approximations) and therefore are more accurately evolved in time for small quantities.

The error covariance matrix P, represents the expectation of the square of the deviation of the state vector estimate from the true value of the state vector. The diagonal elements are the variance of the state estimates, while the off-diagonal elements represent the correlation between the errors in the different state estimates. In a Kalman filter, it is required to initialize the state vector and the covariance matrix. Usually, in error-state implementations the state vector is initialized to zero, while the covariance matrix elements are chosen by the designer to reflect the level of confidence of his *a priori* estimates of the initial state vector.

Each complete iteration of a Kalman filter consists of a propagation and an update step. The state vector and covariance matrix after being propagated in time and before updating, are denoted by, $\hat{\boldsymbol{x}}_k^-$, and P_k^-, respectively. Their counterparts following the measurement update are denoted by $\hat{\boldsymbol{x}}_k^+$, and P_k^+.

The vector $\boldsymbol{z}$ consists of a set of measurements related to the state-vector through a deterministic matrix H and with added noise $\boldsymbol{v}$:

$$\boldsymbol{z} = H\boldsymbol{x} + \boldsymbol{v}\,. \tag{105}$$

The measurement innovation, $\delta \boldsymbol{z}^-$, is the difference between the true measurement vector and the one computed from the state vector before a measurement update:

$$\delta \boldsymbol{z}^- = \boldsymbol{z} - H\hat{\boldsymbol{x}}^-\,. \tag{106}$$

The measurement residual, $\delta \boldsymbol{z}^+$, is the difference between the true measurement vector and the one computed from the updated state-vector:

$$\delta \boldsymbol{z}^+ = \boldsymbol{z} - H\hat{\boldsymbol{x}}^+\,. \tag{107}$$

The standard Kalman filter assumes that the measurement errors form a zero-mean Gaussian distribution, uncorrelated in time, and with a noise covariance matrix R. The covariance matrix R is nothing but the expectation of the square of the measurement noise:

$$R = E(\boldsymbol{v}\boldsymbol{v}^T)\,. \tag{108}$$

7.2 Kalman Filter Algorithm

The data flow of the Kalman filter algorithm is shown in Fig. 19

The following steps constitute the Kalman filter algorithm [36]:

1. Calculate the transition matrix, Φ_{k-1};
2. Compute the noise covariance matrix, Q_{k-1};

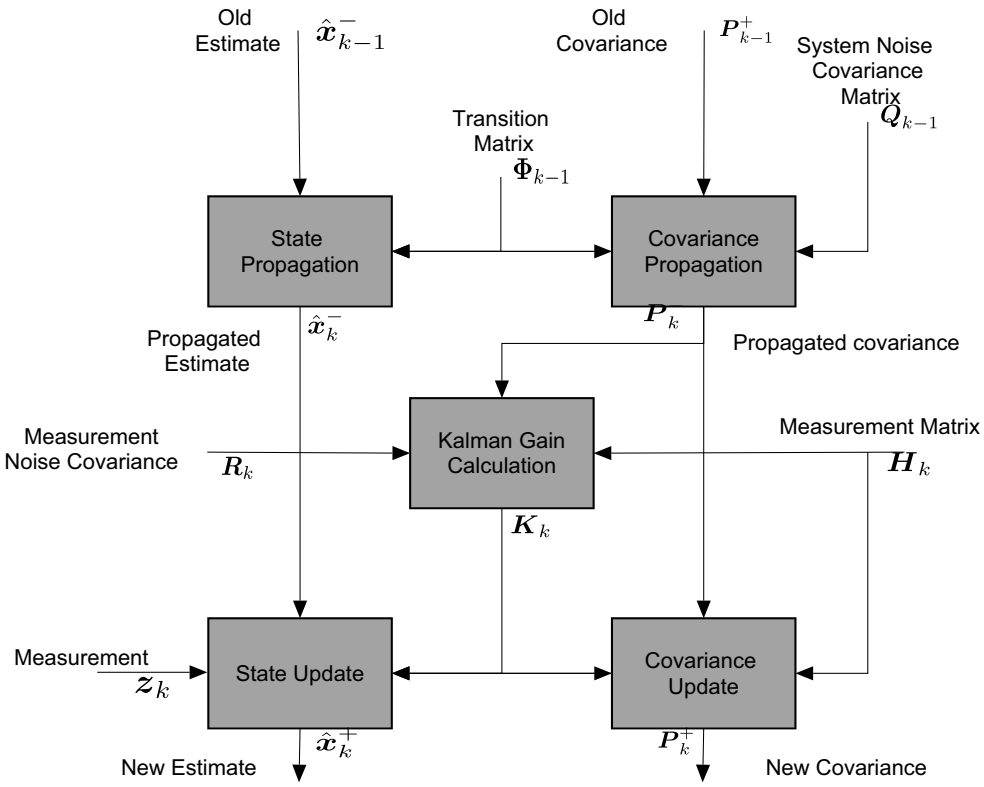

Figure 19 Dataflow graph of a Kalman filter.

3. Propagate the state vector estimate from $\hat{\boldsymbol{x}}_{k-1}^{+}$ to $\hat{\boldsymbol{x}}_{k}^{-}$;
4. Propagate the error covariance matrix from P_{k-1}^{+} to P_{k}^{-};
5. Compute the measurement matrix, H_k;
6. Calculate the measurement noise covariance matrix, R_k;
7. Calculate the Kalman gain matrix K_k;
8. Extract the measurement, z_k;
9. Update the state vector estimate from $\hat{\boldsymbol{x}}_{k}^{-}$ to $\hat{\boldsymbol{x}}_{k}^{+}$;
10. Update the error covariance matrix from P_{k}^{-} to P_{k}^{+};

The first four steps comprise the system propagation phase of the Kalman filter. The last two steps comprise the update phase of the Kalman filter. Later we will see that it is not necessary to execute the update phase of the Kalman filter with every propagation step of the state-vector. On the other hand, the system propagation phase should be executed in every iteration of the Kalman filter.

7.2.1 Transition matrix

The transition matrix describes the dynamics of the system. It defines how the state-vector is propagated with time. It is not a function of any of the state vector parameters. If its elements are a function of time, then it should be updated with every iteration of the Kalman filter. Since we will derive the equations of an error-state Kalman filter, the elements of the state-vector $\boldsymbol{x}$, will take the form of error terms (see Section 6).

$$\begin{aligned}\boldsymbol{x}^T = (&\delta\omega_{Gx}^s, \delta\omega_{Gy}^s, \delta\omega_{Gz}^s, \delta a_{Az}, \psi_n, \psi_e, \psi_d,\\ &\delta v_n, \delta v_e, \delta v_d, \delta\phi, \delta\lambda, \delta h)\,.\end{aligned} \tag{109}$$

The first three terms are gyroscope biases in the x, y, and z directions of the IMU sensor frame. They are considered as random constants, and they are easily modeled as unchanging elements in the state vector. For this reason, we have not derived their error dynamics in the previous section. The fourth term in the state vector is the accelerometer bias in the z direction. It is also modeled as a random constant. We have deleted the accelerometer biases in the x and y directions since they did not improve the accuracy of our estimated state vector. The collected set of transition elements are collected in one transition matrix (110) in a convenient form to directly observe which parameters are correlated, simply by looking at the first row and the first column entries. In contrast, it is very important to estimate the bias in the z direction to prevent the vertical channel, "h", in our Kalman filter from diverging. The transition matrix as seen in BOX C is written as $\Phi_{k-1} = I + A * dT$. It is wise to write the transition matrix without the identity matrix due to the efficiency in calculations achieved in our Kalman filter as will be seen in this section. The elements of the transition matrix will written in Algorithm 9.

$$\Phi_{k-1} = \begin{bmatrix} * \\ \delta\omega^s_{Gx} \\ \delta\omega^s_{Gy} \\ \delta\omega^s_{Gz} \\ \delta a_{Az} \\ \psi_n \\ \psi_e \\ \psi_d \\ \delta v_n \\ \delta v_e \\ \delta v_d \\ \delta\phi \\ \delta\lambda \\ \delta h \end{bmatrix} \begin{bmatrix} * & T_0 & T_1 & T_2 & T_3 & T_4 & T_5 & T_6 & T_7 & T_8 & T_9 & T_A & T_B & T_C \\ T_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ T_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ T_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ T_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ T_4 & T_{40} & T_{41} & T_{42} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ T_5 & T_{50} & T_{51} & T_{52} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ T_6 & T_{60} & T_{61} & T_{62} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ T_7 & 0 & 0 & 0 & 0 & 0 & T_{75} & T_{76} & 0 & 0 & 0 & 0 & 0 & 0 \\ T_8 & 0 & 0 & 0 & 0 & T_{84} & 0 & T_{86} & 0 & 0 & 0 & 0 & 0 & 0 \\ T_9 & 0 & 0 & 0 & T_{93} & T_{94} & T_{95} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ T_A & 0 & 0 & 0 & 0 & 0 & 0 & 0 & T_{A7} & 0 & 0 & 0 & 0 & 0 \\ T_B & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & T_{B8} & 0 & 0 & 0 & 0 \\ T_C & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & T_{C9} & 0 & 0 & 0 \end{bmatrix} . \tag{110}$$

Algorithm 9 Computation of transition matrix elements.

Input: State parameters and time step dT.
Output: Transition matrix, Φ_{k-1}, entries.

$kt[4][0] = c_{11}\langle k\rangle * dT$
$kt[4][1] = c_{12}\langle k\rangle * dT$
$kt[4][2] = c_{13}\langle k\rangle * dT$
$kt[5][0] = c_{21}\langle k\rangle * dT$
$kt[5][1] = c_{22}\langle k\rangle * dT$
$kt[5][2] = c_{23}\langle k\rangle * dT$
$kt[6][0] = c_{31}\langle k\rangle * dT$
$kt[6][1] = c_{32}\langle k\rangle * dT$
$kt[6][2] = c_{33}\langle k\rangle * dT$
$kt[7][5] = \bar{g} * dT$
$kt[7][6] = a_e\langle k\rangle * dT$
$kt[8][4] = -\bar{g} * dT$
$kt[8][6] = -a_n\langle k\rangle * dT$
$kt[9][3] = c_{33}\langle k\rangle * dT$
$kt[9][4] = -a_e\langle k\rangle * dT$
$kt[9][5] = -a_n\langle k\rangle * dT$
$kt[A][7] = (1/R) * dT$ ▷ *A is hexadecimal 10*
$kt[B][8] = 1/(R\cos\phi\langle k\rangle) * dT$ ▷ *B is hexadecimal 11*
$kt[C][9] = -dT$ ▷ *C is hexadecimal 12*

7.2.2 Error Propagation Matrix

In order to propagate the covariance matrix we have to apply the following equation:

$$P_k^- = \Phi_{k-1} P_{k-1}^+ \Phi_{k-1}^T + Q_{k-1} . \tag{111}$$

In order to compute the propagation matrix efficiently, it is beneficial to consider the sparsity of the transition matrix. We shall apply a divide-and-conquer strategy where only the matrix multiplications involving non-zero elements of the transition matrix are executed. We will first compute the matrix, $l = \Phi_{k-1} P_{k-1}^+$, and then multiply the resulting matrix (we call it intermediate matrix) with Φ_{k-1}^T.

To elaborate on the matrix multiplication issue, let us consider that we want to multiply two matrices, T and P, where T is sparse and P is not, such as:

$$T = \begin{bmatrix} 0 & t_{12} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad P = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} . \tag{112}$$

The entry t_{12} contributes only to the first row of the resulting matrix, since the first row of the product uses the terms, $t_{12} * p_{21}$, $t_{12} * p_{22}$, and $t_{12} * p_{23}$, respectively, in its computations.

This strategy decreases the number of accesses to memory where the matrices are stored, since the relevant entries are only loaded once for each non-zero transition matrix entry. This greatly reduces the execution time for the Kalman filter on low-cost embedded processors where resources are limited.

The detailed steps involved in the computation of l, will be shown in Algorithm 10.

To continue the propagation computation of the covariance matrix we shall write the algorithm for the second part of the matrix multiplication and then finally add the system noise covariance matrix, Q_{k-1}. We apply the same strategy as above in Algorithm 11. The final stage in propagating the error covariance matrix is adding system noise as seen in Algorithm 12.

7.2.3 Measurement Matrix and Kalman Gain

The measurement matrix defines how the measurement vector varies with the state vector. This relation in Eq. 105

is repeated here for convenience:

$$z_k = H_k \boldsymbol{x}_k + \boldsymbol{v}_k \,. \tag{113}$$

The measurement noise vector in most applications is considered white with a few exceptions. It has a measurement noise covariance matrix, R_k, that may be assumed constant. In our typical implementation we are directly measuring some state vector elements (GNSS positions and velocities).

The Kalman gain matrix is used to determine the weighting of the measurement information in updating the state estimates. It is a function of the ratio of the uncertainty of the true measurement, z_k to the uncertainty of the measurements predicted from the state estimates, $H\boldsymbol{x}_k^-$.

The Kalman gain matrix is:

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1} \,. \tag{114}$$

7.2.4 State Vector and Error Covariance Update

When ever we obtain a measurement the state vector is updated by the measurement vector using this formula:

$$\hat{\boldsymbol{x}}_k^+ = \hat{\boldsymbol{x}}_k^- + K_k(z_k - H_k \hat{\boldsymbol{x}}_k^-) \,. \tag{115}$$

Similarly, the error covariance matrix is updated with:

$$P_k^+ = (I - K_k H_k) P_k^- \,. \tag{116}$$

As the updated state vector estimate is based on more information, the updated state uncertainties are smaller than before the update.

We will continue this section by providing the detailed algorithms for the second (update) phase of the Kalman filter. This phase is comprised of calculating the matrix gain, K_k, updated state-vector $\hat{\boldsymbol{x}}_k^+$, and updated error covariance matrix, P_k^+. These three steps will be implemented for each new measurement obtained (theoretical details deferred to Section 8).

8 Filter Insights

8.1 Inverse Matrix Calculation

The most complex and time consuming part of the Kalman filter algorithm is finding the inverse of the matrix in the Kalman filter gain, K_k. In order to avoid this tedious calculation, we will prove that it is possible to avoid this inverse matrix calculation by a small trick.

We can write the K_k and the H_k matrices as follows:

$$K_k = \begin{bmatrix} \vdots & \vdots & \vdots \\ K_1 & K_2 & \cdots \\ \vdots & \vdots & \vdots \end{bmatrix}, \tag{117}$$

and

$$H_k = \begin{bmatrix} \cdots & H_1 & \cdots \\ \cdots & H_2 & \cdots \\ \cdots & \vdots & \cdots \end{bmatrix}. \tag{118}$$

Using this notation, we have, $K_k H_k = K_1 H_1 + K_2 H_2 + \cdots$. Thus, it is easily shown that the error covariance matrix can be written as follows:

$$P_k^+ = \underbrace{\underbrace{P_k^- + K_1 H_1 P_k^-}_{P_k^+ \text{ after first measurement}} + K_2 H_2 P_k^- + \cdots}_{P_k^+ \text{ after second measurement}} \tag{119}$$

Note that the sum of the first two terms is nothing but the updated error covariance matrix associated with one measurement. Adding the third term to it, we obtain the error covariance matrix after the second measurement is manipulated. As shown, it is possible to update the error covariance matrix after each reported measurement. This is legitimate as long as we do not propagate the error covariance matrix while manipulating the measurements. It is only required to compute the Kalman gain and correct the state vector after each update of the covariance matrix. The benefit of following this strategy is that the matrix inversion in the Kalman gain computation is transformed to a simple scalar inversion. So by taking any single measurement element only, we can update the covariance matrix and derive and apply system corrections for that single measurement element, much simpler than we can with multiple measurements. In this case, we avoid matrix inversion. It is important that we update everything with a single measurement before using the next measurement, and that we use the updated status before applying the next measurement. All updates must be performed before the next navigation integration cycle. We should always propagate the Kalman filter after every integration of the navigation equations, and only update the Kalman filter (and also make system corrections) whenever we have measurements. The frequency of these updates could be the same or less than the Kalman filter propagation frequency. It depends on the source of our measurements.

8.2 Error Dynamics Approximations

It is also beneficial to address the approximations we made in deriving the error dynamic equations, Eqs. 85 and 90, for those readers who may be concerned. With low-grade IMU we simply forget the Earth's rotation and consider a local flat Earth. It is impossible to detect the very slow rotation of the earth with all the noise and random drift of the gyroscopes. There is no harm in including these factors but they are unlikely to make any improvements.

8.3 Error-State Kalman Filter

It is necessary to remember that we are performing an error-state Kalman filter where the estimated states are INS errors. After each Kalman filter iteration, the estimated states are applied to the corresponding navigation parameters, and thus the state vectors are reset to zero. Consequently, the state vector itself does not need to be propagated forward in time. This type of implementation is called a closed-loop implementation. It is still critical however that the state covariance be propagated using the following equation:

$$P_k^- = \Phi_{k-1} P_{k-1}^+ \Phi_{k-1}^T + Q_{k-1}\,. \tag{120}$$

In this closed-loop technique, since the estimated errors are fed back every iteration, the Kalman filter states are zeroed which keeps the errors of the filter small. This has the effect of minimizing the errors introduced in linearizing the system or process model, since higher order terms in the Taylor series expansion gets smaller and smaller. This is in contrast to the open-loop implementation where there is no feedback, and thus, the states will get larger as time progresses.

We shall also explain a concept which is not very well common between Kalman filter practitioners, related to the idea of how velocity measurements (v_n, v_e, v_d) are essential in estimating tilt errors ψ_n, ψ_e, and ψ_d.

Suppose that the INS is stationary (zero-velocity), and that we have a positive ψ_n error, which means that the system model does not have an exact representation of the rotation of the IMU relative to the local north pointing axis. Instead, it is rotated clockwise by the angle ψ_n about the north pointing axis. Then we will not have the correct transformation value for dv_e after performing the direction cosine transformation of dv_x, dv_y and dv_z (Algorithm 4). The effect is that the east velocity will increment by an error equal to $(\psi_n * g * dT)$. This is the same term relating the error δv_e to ψ_n in the transition matrix (110). Thus, the system computed velocity v_e is no longer zero. But we know it is zero, so we give a measurement to the Kalman filter equal to $-v_e$. This generates a whole set of corrections for the system, including highly weighted corrections to v_e, ψ_n and gyroscope biases. The other terms are only weakly coupled to δv_e in the Kalman filter and have only very small, almost negligible corrections (for these corrections, you need the v_n and v_d measurements). We then apply quaternion correction, which reduces the error in ψ_n (as well as the other terms) so that, next time round, the error in v_e is smaller. This way, after several cycles, we reduce the error in ψ_n until the roll and pitch values and v_e are correct (gyroscope biases take a bit longer).

So as we can see, it is the accelerometers (delta velocities) which determine the alignment and not the gyroscopes. The gyroscopes are only there so that sensed rotation can be immediately applied to the attitude solution instead of waiting until the Kalman filter, aided by velocity measurements, eventually finds the new attitude angles (direction cosine matrix).

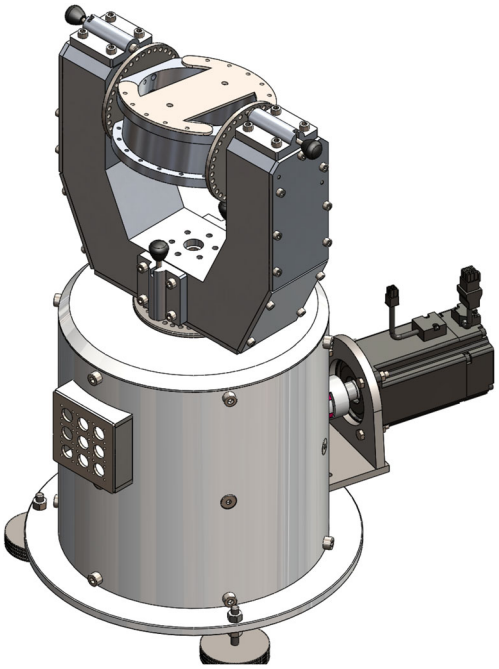

Figure 20 Rate-tables are used for gyroscopes/IMU calibration or hardware-in-loop (HWIL) testing. Payloads are mounted on the table top platen. A pattern of threaded holes accept a variety of test loads. It is often equipped with high speed optical or electrical slip-rings to transmit data to and from the payload under test. It contains a direct drive brushless motor with dedicated amplifiers, controllers and a heavy duty power supply. Some of the main error sources that may be acquired from the calibration process include: sensor-to-axis misalignments, gyroscope and accelerometer scale factor errors, and bias errors. The rate-table is usually mounted inside a temperature chamber in order for the payload unit to be calibrated at equally spaced temperatures ranging from −40°C to +85°C [40].

These are some recommendations for Kalman filter designers:

- Check that you can read and display the IMU data with your software.
- Prove that quaternion integration, body to navigation direction-cosine-matrix computation and roll-pitch-heading routines work without the Kalman filter, by starting level and manually rotating for a short time. To do this you do not need to transform and integrate delta velocities.
- Transform, integrate and display velocities; they should rapidly become very large.
- Develop your Kalman filter by displaying all the variances adjacent to their respective state matrix elements. Note that the typical error of the of a state element "i" (squared) should approximately

Algorithm 10 Efficient propagation of the error covariance matrix (part a).

Input: Transition matrix, Φ_{k-1}, and covariance matrix, P_{k-1}^{+}.

Output: Intermediate propagated covariance matrix, $l = \Phi_{k-1} P_{k-1}^{+}$.

```
Compute Intermediate covariance matrix l = Φ_{k-1} P^+_{k-1}

function KO(i, j)
    for (k= 0;k <13;k++) do
        ko[i][k] = ko[i][k] + kt[i][j] * kp[j][k];
    end for
end function

function KL(void)
    for (i= 0;i <13;i++) do
        for (j= 0;j <13;j++) do
            ko[i][j] = kp[i][j];
        end for
    end for
    i = 4; j = 0;
    KO(i, j);
    i = 4; j = 0;
    KO(i, j);
    i = 4; j = 2;
    KO(i, j);
    i = 5; j = 0;
    KO(i, j);
    i = 5; j = 1;
    KO(i, j);
    i = 5; j = 2;
    KO(i, j);
    i = 6; j = 0;
    KO(i, j);
    i = 6; j = 1;
    KO(i, j);
    i = 6; j = 2;
    KO(i, j);
    i = 7; j = 5;
    KO(i, j);
    i = 7; j = 6;
    KO(i, j);
    i = 8; j = 4;
    KO(i, j);
    i = 8; j = 6;
    KO(i, j);
    i = 9; j = 3;
    KO(i, j);
    i = 9; j = 4;
    KO(i, j);
    i = 9; j = 5;
    KO(i, j);
    i = 10; j = 7;
    KO(i, j);
    i = 11; j = 8;
    KO(i, j);
    i = 12; j = 9;
    KO(i, j);
end function
```

equal to its corresponding i'th diagonal entry in the covariance matrix P_k.

8.4 System Model Error Sources

In the last few years, Microelectromechanical system (MEMS) gyroscopes and accelerometers are beginning to take market away from traditional inertial sensors like fiber optic gyroscopes (FOG) and ring laser gyroscopes (RLG). This take over has been occurring due to improved error characteristics, environmental stability, better bandwidth, enhanced g-sensitivity, and the plethora of of embedded computational power that can run advanced fusion and sensor error modeling algorithms. This transition couldn't have been achieved if not for the advancements in MEMS technology which had remarkable improvements on the error characteristics of the sensors [37–39].

In system modeling in general, we only consider the main error sources. We ignore the terms which have little effect on system performance, but allow for these small effects as additional noise in other terms. This is not perfectly correct, but its is a good compromise, since it simplifies the mathematics significantly, minimises the size of the Kalman filter and works surprisingly well.

For example $d\phi_x$, $d\phi_y$, and $d\phi_z$ are actual measurements and their values would be absolutely correct if there were no errors in scale factors, biases and mis-alignments, and therefore we do not consider errors in these terms (corrected for by rate-table Fig. 20). However, if there were additional errors in $d\phi_x$, $d\phi_y$, and $d\phi_z$, then these errors would come

Algorithm 11 Efficient propagation of the error covariance matrix (part b).

Input: Intermediate propagated covariance matrix, $l = \Phi_{k-1} P_{k-1}^{+}$.

Output: Propagated Covariance matrix, $P_k^{-} = \Phi_{k-1} P_{k-1}^{+} \Phi_{k-1}^{T}$.

Compute Intermediate covariance matrix $l * \Phi_{k-1}^{T}$

function KP(i, j)
  for `(k= 0;k <13;k++)` **do**
    $kp[k][i] = kp[k][i] + kt[i][j] * ko[k][j];$
  end for
end function

function KL($void$)
  for `(i= 0;i <13;i++)` **do**
    for `(j= 0;j <13;j++)` **do**
      $kp[i][j] = ko[i][j];$
    end for
  end for
  $i = 4; j = 0;$
  KP(i, j);
  $i = 4; j = 0;$
  KP(i, j);
  $i = 4; j = 2;$
  KP(i, j);
  $i = 5; j = 0;$
  KP(i, j);
  $i = 5; j = 1;$
  KP(i, j);
  $i = 5; j = 2;$
  KP(i, j);
  $i = 6; j = 0;$
  KP(i, j);
  $i = 6; j = 1;$
  KP(i, j);
  $i = 6; j = 2;$
  KP(i, j);
  $i = 7; j = 5;$
  KP(i, j);
  $i = 7; j = 6;$
  KP(i, j);
  $i = 8; j = 4;$
  KP(i, j);
  $i = 8; j = 6;$
  KP(i, j);
  $i = 9; j = 3;$
  KP(i, j);
  $i = 9; j = 4;$
  KP(i, j);
  $i = 9; j = 5;$
  KP(i, j);
  $i = 10; j = 7;$
  KP(i, j);
  $i = 11; j = 8;$
  KP(i, j);
  $i = 12; j = 9;$
  KP(i, j);
end function

from other sources such as measurement electronics, g-sensitivity and non-linearity, which we have ignored. We have not modelled these errors in the mathematics but instead we do account for these additional errors by having larger values of system noise in the other terms.

A good example where we add "unnecessary" noise in the Kalman filter is in δv_n, δv_e, and δv_d. Where could noise appear in these terms? A very minute amount could come from computing noise (e.g., loss of numerical values which are smaller than the least-significant-bit when integrating), but that's all. So why do we add velocity noise in the Kalman filter? The answer is, it compromises for errors caused by non-considered effects and thus prevents the mathematics in the Kalman filter from exploding due to a reduced mathematical model.

8.5 IMU and GNSS Time Synchronization

We have assumed that GNSS and IMU sytems are time synchronized until now. In practice, they are not. Suppose there is a small time lag between both systems. If the vehicle is moving at constant velocity then the solutions provided by both systems will perfectly match, and thus no error is introduced. In contrast, in case of a small acceleration experienced by the vehicle, the timing lag will manifest itself as a position and velocity difference between the

two system solutions. Therefore, it is mandatory to have dedicated hardware in your system that compensates for this time lag, or take care of it by software.

Algorithm 12 Addition of system noise to error covariance matrix.

Input: Propagated Covariance matrix P_k^- minus noise.
Output: Propagated Covariance matrix P_k^-.

1: **Add system noise covariance matrix to** $\Phi_{k-1} P_{k-1}^+ \Phi_{k-1}^T$
2:
3: **function** KP($void$)
4: **for** `(i= 0;i <13;i++)` **do**
5: $kp[i][i] = kp[i][i] + kq[i] * dt$;
6: **end for**
7: **end function**

9 Other Attitude Filters

A computationally simpler alternative to a Kalman filter for the GPS-INS algorithms in Fig. 18 is to use a feedback controller to correct for gyro drift [41–43]. In the GPS-INS feedback controller shown in Fig. 21, a compensator typically based on proportional–integral control, is used to estimate gyro biases based on a body-frame attitude error vector, $\boldsymbol{e}^b$. $\boldsymbol{e}^b$ is derived by noting that if the output attitude estimate were correct and other error sources were negligible, then the estimated North and Down directions would align with those observed from a yaw reference ($\boldsymbol{\psi}_{ref}$) and a body-frame gravity vector reference ($\boldsymbol{g}_{ref}^b$). The gravity vector reference is derived from the centripetally corrected accelerometer measurements:

$$\boldsymbol{g}_{ref}^b = \boldsymbol{w}_{gyro}^b \times \boldsymbol{V}_{ref}^b - \boldsymbol{f}_{accel}^b \,. \tag{121}$$

The error vector $\boldsymbol{e}^b$ is expressed as the summation of errors generated from the yaw reference ($\boldsymbol{\psi}_{ref}$) and body-frame gravity vector reference ($\boldsymbol{g}_{ref}^b$). So we have:

$$\boldsymbol{e}^b = \boldsymbol{e}_{\psi}^b + \boldsymbol{e}_g^b \,, \tag{122}$$

where

$$\begin{aligned} \boldsymbol{e}_{\psi}^b &= \left(C_x(\hat{\phi}) C_y(\hat{\theta}) C_z(\psi_{ref}) \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \right) \times \left(\hat{C}_{ned}^b \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \right) , \\ \boldsymbol{e}_g^b &= \left(\frac{\boldsymbol{g}_{ref}^b}{\|\boldsymbol{g}_{ref}^b\|} \right) \times \left(\hat{C}_{ned}^b \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \right) , \end{aligned} \tag{123}$$

and $\hat{C}_{ned}^b = \left(C_x(\hat{\phi}) C_y(\hat{\theta}) C_z(\hat{\psi}) \right)$ is the current estimate of the transformation matrix from the ned frame to the body frame.

In Eq. 123, $\boldsymbol{e}_{\psi}^b$ is the rotation vector expressed in body coordinates between the observed North direction defined by $\boldsymbol{\psi}_{ref}$ and the system estimated North-direction.(Note that the cross-product between two vectors, $\times$, between two vectors yields a vector orthogonal to both with a magnitude proportional to the sine of the angle between them). Similarly, $\boldsymbol{e}_g^b$ is the rotation vector in body coordinates between the centripetally corrected gravity direction estimated from the accelerometers and the system-estimated Down direction. As a result, the combined error vector $\boldsymbol{e}^b$, expresses the angular error and the rotation axis between the reference NED coordinate frame and the system-estimated NED frame. This feedback error vector id filtered via a compensator (proportional-integral controller) to generate the estimated gyroscope biases. Subtracting these estimated gyroscope biases from the gyroscope measurements and performing a quaternion integration on the de-biased gyroscope data, we obtain the desired Euler angles.

The drawback of this filer is it its assumption of zero average linear acceleration for Eq. 121 to apply. While this assumption is true for most vehicles spending most of the time cruising with zero acceleration, it may lead to unaccepted results in terms of attitude estimation in case of prolonged or transient linear accelerations. Nevertheless whenever this transient acceleration is gone, it can recover and converge to the true values of attitude in fast manner.

Yet, another computationally efficient non-linear attitude filter is shown in Fig. 22. The algorithm uses a quaternion representation, allowing accelerometer and magnetometer data to be used in an analytically derived and optimised gradient descent algorithm to compute the direction of the gyroscope measurement error as a quaternion derivative. The quaternion derivative describing rate of change of the NED frame relative to the sensor frame can be calculated as [44] shown in Eq. 124:

$$\dot{\boldsymbol{q}} = \frac{1}{2} \hat{\boldsymbol{q}} \otimes \boldsymbol{w} \,, \tag{124}$$

where $\boldsymbol{w} = (0, w_x, w_y, w_z)$ is augmented angular rate measurement vector delivered by the gyroscopes and $\hat{\boldsymbol{q}}$ is the normalized quaternion vector representing the relative orientation between the NED frame and the body coordinate frame.

Provided the initial conditions are known, Eq. 124 can be numerically integrated to solve for $\boldsymbol{q}$. If the time step considered is dT the integration can be done using the following

$$\hat{\boldsymbol{q}}_t = \hat{\boldsymbol{q}}_{t-1} + \dot{\boldsymbol{q}}_t dT \,. \tag{125}$$

Algorithm 13 State-vector and error covariance matrix update.

Input: Propagated state-vector $\boldsymbol{x}_k^-$ and error covariance matrix P_k^-.

Output: Updated state-vector $\boldsymbol{x}_k^+$ and error covariance matrix P_k^+.

Update the Kalman filter for each individual measurement z_k
$i = 7;$
$kz[i] = GpsVeln - v_n;$
UPDATE(i);
CORRECTION($void$);
ZEROING($void$);
$i = 8;$
$kz[i] = GpsVele - v_e;$
UPDATE(i);
CORRECTION($void$);
ZEROING($void$);
$i = 9;$
$kz[i] = GpsVeld - v_d;$
UPDATE(i);
CORRECTION($void$);
ZEROING($void$);
$i = 10;$
$kz[i] = GpsLat - \phi;$
UPDATE(i);
CORRECTION($void$);
ZEROING($void$);
$i = 11;$
$kz[i] = GpsLon - \lambda;$
UPDATE(i);
CORRECTION($void$);
ZEROING($void$);
$i = 12;$
$kz[i] = GpsHeight - h;$
UPDATE(i);
CORRECTION($void$);
ZEROING($void$);
function UPDATE(i)
    for (j= 0;j <13;j++) **do**
        $kw[i][j] = kp[i][j] * tmp;$
    end for
    for (j= 0;j <13;j++) **do**
        for (k= 0;k <13;k++) **do**
            $ko[j][k] = kp[j][i] * kw[i][k];$
        end for
    end for
    for (j= 0;j <13;j++) **do**
        for (k= 0;k <13;k++) **do**
            $kp[j][k] = kp[j][k] - ko[j][k];$
        end for
    end for
    for (j= 0;j <13;j++) **do**
        $kx[j] = kx[j] + ki[i] * kw[i][j]$
    end for
end function
function CORRECTION($void$)
    $\omega_{GBx} = \omega_{GBx} + kx[0];$ ▷ *Gyro Bias Correction*
    $\omega_{GBy} = \omega_{GBy} + kx[1];$ ▷ *Gyro Bias Correction*
    $\omega_{GBz} = \omega_{GBz} + kx[2];$ ▷ *Gyro Bias Correction*
    $a_{Bz} = a_{Bz} + kx[3];$ ▷ *Accel Bias Correction*
    $\psi_n = -kx[4];$
    $\psi_e = -kx[5];$
    $\psi_d = -kx[6];$
    $d\phi_x = \psi_n * c_{11} + \psi_e * c_{21} + \psi_d * c_{31};$
    $d\phi_y = \psi_n * c_{12} + \psi_e * c_{22} + \psi_d * c_{32};$
    $d\phi_z = \psi_n * c_{13} + \psi_e * c_{23} + \psi_d * c_{33};$
    Do Quaternion Integration (Algorithm 5);
    $v_n = v_n + kx[7];$
    $v_e = v_e + kx[8];$
    $v_d = v_d + kx[9];$
    $\phi = \phi + kx[10];$
    $\lambda = \lambda + kx[11];$
    $h = h + kx[12];$
end function
function ZEROING($void$)
    for (i= 0;i <13;i++) **do**
        $kx[i] = 0;$
    end for
end function

In the context of an orientation estimation algorithm, it will initially be assumed that an accelerometer will measure only gravity and a magnetometer will measure only the earth's magnetic field. If the direction of an earth's reference field is known in the earth frame, a measurement of the field's direction within the sensor frame will allow an orientation of the sensor frame relative to the earth frame to be calculated. However, for any given measurement there will not be a unique sensor orientation solution, instead there will be infinite solutions represented by all those orientations achieved by the rotation of the true sensor frame around an axis parallel with the field's direction. A

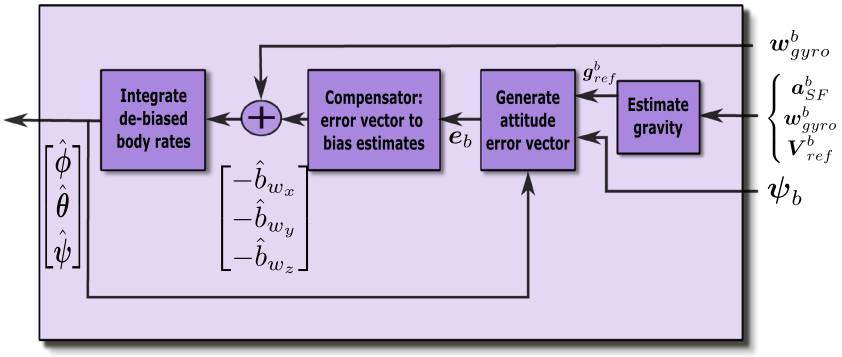

Figure 21 An attitude heading reference system feedback controller estimates the body orientation by fusing high-bandwidth gyro angular rate measurements with low-bandwidth attitude references. The yaw reference comes from a magnetometer or a GPS course-based estimate. Pitch and roll references are acquired from an estimate of the gravity vector via centripetally corrected accelerometer measurements. Essentially, this feedback controller uses a compensator to estimate the gyro biases by regulating the error between the estimated orientation and the orientation expressed by the low-bandwidth attitude references.

quaternion representation requires a single solution to be found. This may be achieved through the formulation of an optimisation problem where an orientation of the sensor, $\hat{\boldsymbol{q}}$, is found as that which aligns a predefined reference direction in the earth frame, $\boldsymbol{d}_{ref}$, with the measured field in the body coordinate frame, $\boldsymbol{s}$; thus solving (126) where $\boldsymbol{f}$ in Eq. 127, defines the objective function [45]:

$$\min_{\hat{\boldsymbol{q}}} \boldsymbol{f}(\hat{\boldsymbol{q}}, \boldsymbol{d}_{ref}, \boldsymbol{s}), \tag{126}$$

where

$$\boldsymbol{f}(\hat{\boldsymbol{q}}, \boldsymbol{d}_{ref}, \boldsymbol{s}) = \hat{\boldsymbol{q}}^* \otimes \boldsymbol{d}_{ref} \otimes \hat{\boldsymbol{q}} - \boldsymbol{s}. \tag{127}$$

Many optimisation algorithms exist but the gradient descent algorithm is one of the simplest to both implement and compute. The equation used to implement the gradient descent algorithm is shown in the block diagram (Fig. 22).

10 Summary

The use of attitude estimation filters have seen an explosive growth in the past few decades, specifically with the growth of the smartphones market. The use of touchscreens for gaming has popularized motion detection and orientation estimation within the portable computing platform. Orientation estimation algorithms for inertial sensors is a is a mature field of research. Modern techniques [12, 46, 47] have focused on simpler algorithms that ameliorate the computational load and parameter tuning burdens associated with conventional Kalman-based approaches. The algorithms presented in this paper employs some cost-effective techniques and is able to offer some key advantages in terms of energy efficiency with out sacrificing accuracy, aiming at deploying this technology in low-cost hardware.

In this article, we have showed the advantages of fusing GNSS and INS systems, in terms of their complementary properties. Various integration architectures are also exploited, that demonstrate how the lower bandwidth of the GNSS system can act as an online calibrator for the INS system. Specifically, how the gyroscope and accelerometer errors are compensated for, using estimated drift and biases.

This article points the reader's attention to various insights necessary for a successful GNSS-INS design. It is a hands-on approach that when followed step-by-step, by applying the presented navigation and fusing filter algorithms, guarantees a completely working and efficient stand-alone system. The authors have made their best to keep this article self-contained. To the best of our

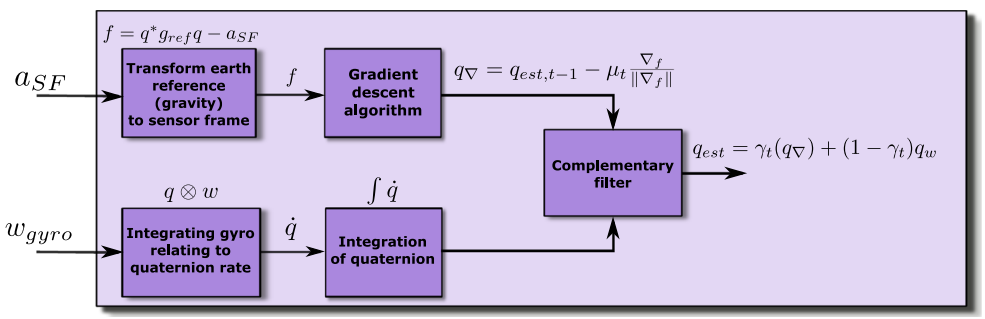

Figure 22 Madgwick Filter based on gradient descent optimization

knowledge, this is the one of the few articles that fills the gaps between the theoretical and applied aspects of inertial navigation systems.

Appendix

Algorithm 14 is an accurate polynomial implementation of the arc-tangent function. It takes two arguments as inputs. It is similar to the four-quadrant inverse tangent function, "atan2(x,y)" in Matlab.

Algorithm 14 Efficient computation of arctan($C1, C2$).

Input: Arguments $C1$ and $C2$.
Output: Arc tangent of the input arguments.

if $|C1| \geq |C2|$ **then**
  $R1 = \frac{C2}{C1}$;
else
  $R1 = \frac{C1}{C2}$;
end if
if $|C1| == -|C2|$ **then**
  $R1 = -1$;
end if
if $|C1| == |C2|$ **then**
  $R1 = 1$;
end if
$R2 = R1 * R1$;
$R3 = R1 * R2$;
$R5 = R3 * R2$;
$R7 = R5 * R2$;
$R9 = R7 * R2$;
$ANG = 0.999896 * R1 - 0.330756 * R3 + 0.181946 * R5 - 0.0876858 * R7 + 0.021997 * R9$;
if $|C1| < |C2|$ **then**
  $ANGLE = ANG$;
else
  if $ANG > 0$ **then**
    $ANGLE = \frac{\pi}{2} - ANG$;
  end if
  if $ANG < 0$ **then**
    $ANGLE = -\frac{\pi}{2} - ANG$;
  end if
end if

References

1. Winkler, S., Wiedermann, G., Gockel, W. (2008). High-accuracy on-board attitude estimation for the GMES Sentinel-2 satellite: concept, design, and first results. In *AIAA Guidance navigation and control conference and exhibit* (p. 7482).
2. Friedland, B. (1978). Analysis strapdown navigation using quaternions. *IEEE Transactions on Aerospace and Electronic Systems*, *5*, 764–768.
3. Huddle, J.R. (1989). Advances in strapdown systems for geodetic applications. In *High precision navigation* (pp. 496–530): Springer.
4. Kealy, A., Retsche, G., Grejner-Brzezińska, D., Gikas, V., Roberts, G. (2011). Evaluating the performance of mems based inertial navigation sensors for land mobile applications. In *Archiwum Fotogrametrii Kartografii i Teledetekcji*, Vol. 22.
5. Barton, J.D. (2012). Fundamentals of small unmanned aircraft flight. *Johns Hopkins APL technical digest*, *31*(2), 132–149.
6. Li, R.-B., LIU, J.-Y., ZENG, Q.-H., HUA, B. (2004). Evolution of mems based micro inertial navigation systems [j]. *Journal of Chinese Inertial Technology*, *6*, 90–96.
7. Nguyen, V.H. et al. (2012). Loosely coupled GPS/INS integration with Kalman filtering for land vehicle applications. In *2012 International Conference on Control, Automation and Information Sciences (ICCAIS)* (pp. 90–95): IEEE.
8. Wang, W., Liu, Z.-Y., Xie, R.-R. (2006). Quadratic extended Kalman filter approach for GPS/INS integration. *Aerospace Science and Technology*, *10*(8), 709–713.
9. Wang, G., Han, Y., Chen, J., Wang, S., Zhang, Z., Du, N., Zheng, Y. (2018). A GNSS/INS integrated navigation algorithm based on Kalman filter. *IFAC-PapersOnLine*, *51*(17), 232–237.
10. Jiang, C., Zhang, S.-B., Zhang, Q.-Z. (2016). A novel robust interval Kalman filter algorithm for GPS/INS integrated navigation. Journal of Sensors, 2016.
11. Wendel, J., Schlaile, C., Trommer, G.F. (2001). Direct Kalman filtering of GPS/INS for aerospace applications. In *International symposium on kinematic systems in geodesy geomatics and navigation (KIS2001)*.
12. Martin, P., & Salaün, E. (2010). Design and implementation of a low-cost observer-based attitude and heading reference system. *Control Engineering Practice*, *18*(7), 712–722.
13. Werries, A., Dolan, J., et al. (2016). Adaptive Kalman filtering methods for low-cost GPS/INS localization for autonomous vehicles. Carnegie-Mellon University, Tech. Rep.
14. Shaghaghian, A., & Karimaghaee, P. (2019). Improving GPS/INS Integration Using FIKF-filtered Innovation Kalman Filter. *Asian Journal of Control*, *21*(4), 1671–1680.
15. Pham, V.T., Nguyen, V.T., Chu, D.T., Tran, D.T. (2015). 15-State extended Kalman filter design for INS/GPS navigation system. *Journal of Automation and Control Engineering*, *2*, 3.
16. Titterton, D.H., & Weston, J.L. (2004). *Strapdown Inertial Navigation Technology*, 2nd edn. London: Institution of Electrical Engineers.
17. Schuler, M. (1923). Die störung von Pendel und Kreiselapparaten durch die Beschleunigung des Fahrzeuges'. *Physikalische Zeitschrift*, *24*(16), 344–350.
18. Battin, R.H. (1982). Space guidance evolution-a personal narrative. *Journal of Guidance, Control, and Dynamics*, *5*(2), 97–110.
19. Draper, C., Wrigley, W., Hoag, D., Battin, R., Miller, J., Koso, D., Hopkins, A., VanderVelde, W. (1965). Guidance and Navigation.
20. Tazartes, D. (2014). An historical perspective on inertial navigation systems. In *2014 International Symposium on Inertial Sensors and Systems (ISISS)* (pp. 1–5): IEEE.
21. Reams, G., Roberts, G., Fulcher, W. (1983). Effects of polar motion on ICBM accuracy. In *Guidance and control conference* (p. 2295).
22. Ford, C. (1983). Gravitational model effects on ICBM accuracy. In *Guidance and control conference* (p. 2296).

23. Merhav, S.J. (1982). A nongyroscopic inertial measurement unit. *Journal of Guidance, Control, and Dynamics*, *5*(3), 227–235.
24. Vagner, M. (2011). MEMS Gyroscope performance comparison using Allan Variance Method. In *Proceedings of the 17th Conference Student EEICT* (pp. 199–203).
25. Gade, K. (2008). Introduction to inertial navigation and Kalman filtering, INS Tutorial.
26. Board, I. (1998). IEEE standard specification format guide and test procedure for single-axis interferometric fiber optic gyros. In *IEEE Std* (pp. 952–1997).
27. Vaccaro, R.J., & Zaki, A.S. (2011). Statistical modeling of rate gyros. *IEEE Transactions on Instrumentation and Measurement*, *61*(3), 673–684.
28. Petkov, P., & Slavov, T. (2010). Stochastic modeling of MEMS inertial sensors. *Cybernetics and information technologies*, *10*(2), 31–40.
29. Leffens, E., Markley, F.L., Shuster, M.D. (1982). Kalman filtering for spacecraft attitude estimation. *Journal of Guidance Control, and Dynamics*, *5*(5), 417–429.
30. Kuipers, J.B. (1999). Quaternions and rotation sequences: a primer with applications to orbits, aerospace, and virtual reality, Princeton University Press, Princeton.
31. Jekeli, C. (2012). *Inertial navigation systems with geodetic applications*. Berlin: Walter de Gruyter.
32. Salychev, O.S. (2012). MEMS-Based inertial navigation: Expectations and reality. Bauman Moscow State Technical University.
33. Salychev, O.S. (2004). *Applied inertial navigation: problems and solutions*. Moscow: BMSTU Press.
34. Kalman, R.E. (1960). A new approach to linear filtering and prediction problems.
35. Groves, P.D. (2015). Principles of GNSS, inertial, and multisensor integrated navigation systems, [Book review]. *IEEE Aerospace and Electronic Systems Magazine*, *30*(2), 26–27.
36. Farrell, J., & Barth, M. (1999). *The global positioning system and inertial navigation* Vol. 61. New York: Mcgraw-hill.
37. Niu, X., Nassar, S., Syed, Z., Goodall, C., El-Sheimy, N. (2006). The development of a mems-based inertial/gps system for land-vehicle navigation applications. In *Proceedings of the 19th international technical meeting of the satellite division of the institute of navigation (ION GNSS 2006), Fort Worth, TX, USA* (pp. 26–29).
38. Goodall, C., Carmichael, S., Scannell, B. (2013). The battle between mems and fogs for precision guidance. Analog Devices Technical Article MS-2432.
39. Schmidt, G.T. (2015). Navigation sensors and systems in gnss degraded and denied environments. *Chinese Journal of Aeronautics*, *28*(1), 1–10.
40. Lawrence, A. (2012). *Modern inertial technology: navigation, guidance, and control*. New York: Springer Science & Business Media.
41. Euston, M., Coote, P., Mahony, R., Kim, J., Hamel, T. (2008). A complementary filter for attitude estimation of a fixed-wing UAV. In *2008 IEEE/RSJ international conference on intelligent robots and systems* (pp. 340–345): IEEE.
42. Beard, R., Kingston, D., Quigley, M., Snyder, D., Christiansen, R., Johnson, W., McLain, T., Goodrich, M. (2005). Autonomous vehicle technologies for small fixed-wing UAVs. *Journal of Aerospace Computing, Information, and Communication*, *2*(1), 92–108.
43. Bryson, M., & Sukkarieh, S. (2004). Vehicle model aided inertial navigation for a UAV using low-cost sensors. In *Proceedings of the Australasian conference on robotics and automation* (pp. 1–9): Citeseer.
44. Cooke, J.M., Zyda, M.J., Pratt, D.R., McGhee, R.B. (1992). NPSNET: Flight Simulation dynamic modeling using quaternions. *Presence: Teleoperators & Virtual Environments*, *1*(4), 404–420.
45. Madgwick, S.O., Harrison, A.J., Vaidyanathan, R. (2011). Estimation of IMU and MARG orientation using a gradient descent algorithm. In *2011 IEEE international conference on rehabilitation robotics* (pp. 1–7): IEEE.
46. Bachmann, E.R., McGhee, R.B., Yun, X., Zyda, M.J. (2001). Inertial and magnetic posture tracking for inserting humans into networked virtual environments. In *Proceedings of the ACM symposium on Virtual reality software and technology* (pp. 9–16).
47. Mahony, R., Hamel, T., Pflimlin, J.-M. (2008). Nonlinear complementary filters on the special orthogonal group. *IEEE Transactions on automatic control*, *53*(5), 1203–1218.

Hussein Al-Jlailaty received the B.E. and M.E. degrees in electromechanical engineering from the Lebanese University (FEA), in 2003 and 2006, respectively, and the M.S. degree in physics and the M.E. degree in mechanical engineering from the Lebanese University and from the American University of Beirut (AUB), Lebanon, in 2005 and 2019, respectively. He worked as a research assistant with the Vision and Robotics Lab (VRL) for two years. He is currently pursuing the Ph.D. degree in electrical engineering with the American University of Beirut (AUB). His research interest lie in the areas of robotics and digital signal processing, inertial navigation systems and reliable designs by algorithms.

Mohammad Mansour received the B.E. and M.E. degrees in computer and communications engineering from the American University of Beirut (AUB), Lebanon, in 1996 and 1998, respectively, and the M.S. degree in mathematics and the Ph.D. degree in electrical engineering from the University of Illinois at Urbana–Champaign (UIUC), Champaign, IL, USA, in 2002 and 2003, respectively. He is currently a tenured Professor and Chairperson of the ECE Department at AUB. His research interests are in energy-efficient, high-performance, and reliable designs by algorithm, architecture, and circuit co-optimizations, with emphasis on emerging applications in 5G wireless communications, signal processing, deep learning networks, computing, and security. He has held visiting and consulting positions with Qualcomm, Broadcom, Intel, and Tensorcom, where he was involved in algorithm and architecture design for baseband receivers and computing systems. He is an active Senior Member of the IEEE.