UNIVERSITY OF CALIFORNIA

SANTA CRUZ

A COMPARISON OF ATTITUDE PROPAGATION AND PARAMETERIZATION METHODS FOR LOW-COST UAVs

A thesis submitted in partial satisfaction
of the requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER ENGINEERING

by

Robert Taylor Casey

December 2012

The Thesis of Robert Taylor Casey
is approved:

Professor Gabriel Hugh Elkaim, Chair

Professor Renwick E. Curry

Professor Mark Karpenko
Naval Postgraduate School

Tyrus Miller
Vice Provost and Dean of Graduate Studies

Table of Contents

List of Figures

List of Tables

Abstract

A Comparison of Attitude Propagation and Parameterization Methods for Low-Cost UAVs

by

Robert Taylor Casey

Unmanned aerial vehicles (UAVs) represent an increasingly important and prolific technology in today's world, finding use in myriad applications across multiple domains, including civil, commercial, military, and research environments. Control of these aircraft requires fundamental information on the vehicle's position and orientation in space. Attitude determination algorithms calculate this spatial orientation by propagating the attitude kinematic equations that estimate the current attitude based on previous estimates along with information about the vehicle's angular velocities. Within the domain of low-cost UAVs, numerous options exist for the choice of 1) propagation algorithms, 2) attitude representation, and 3) the assumptions about the behavior of the angular velocity vector between samples within the discrete-time hardware of the embedded system typically running the estimation algorithms. This thesis examines the impact of these three variables upon propagated attitude estimates with respect to accuracy, computational efficiency, and noise response. Noise response is evaluated in terms of the algorithm's ability to track an underlying clean signal in spite of inputs corrupted by additive Gaussian noise. Various propagation methods are evaluated across four attitude representations: the direction cosine matrix, Euler angles, quaternions, and the angle-axis or eigen-axis parameterization. Lastly, the nature of angular velocity (constant, linear, and quadratic) is evaluated in terms of accuracy, computational efficiency, and noise resilience.

The algorithms were tested using simulated angular velocity inputs from analytic functions as well as flight test data from low-cost, fixed wing UAVs. Implementation was done in Matlab as well as Simulink-based test modules to evaluate algorithm performance.

The quaternion parameterization proved most beneficial across all three test metrics, though the DCM representation was only slightly deficient in terms of computational load. Matrix exponential propagation methods offered higher accuracy and better noise response than direct integration using the first-order Euler method, though the latter offered better computational efficiency. For low-cost UAV applications, where MEMS-based sensors have large noise on the gyros, a DCM parameterization using the matrix exponential with linear fit to the data gives the best results for reasonable computation. This is especially true when using either magnetometer or accelerometer feedback into the attitude filter for bias elimination.

To Tajina, my better and wiser half. You mean everything to me, and I rely upon you more than you know! Thank you for your patience, understanding, caring, laughs, and love throughout. Without you, none of this would have been possible or worthwhile. Thank you for hanging in there for this stage of our Hobbit journey together. Let's advance to the next chapter!

ACKNOWLEDGMENTS.

I would like to thank my graduate advisor at UC Santa Cruz, Gabriel Elkaim, for bringing me into the ASL and for encouraging, guiding, and inspiring me over the last few years. Your influence has raised the bar for me in terms of what I have done and for what I hope to do in engineering.

I would also like to thank Ren Curry, member of my thesis committee whose pilot and aircraft expertise, attitude estimation knowledge, and keen insights into the workings of accelerometers and complementary filters propelled my thesis work to new levels.

To Mark Karpenko at the Naval Postgraduate School (NPS), thesis committee advisor and advisor at work, whose insights and feedback have been critical to my current and future work. I look forward to the new and exciting projects which await us just around the corner.

To Michael Ross, my employer at NPS. Thank you for providing an inspiring, liberating environment in which to work and do research. Our research group holds great promise for the future and I hope to shine more light on this already bright star.

To Vladimir Dobrokhodov, fellow researcher at NPS, whose undying patience, professionalism, and clear communication has aided my understanding of Simulink, control systems, unmanned aerial vehicles, and what it means to be a reliable engineer.

To Kevin Jones, fellow researcher at NPS, whose extensive RC flight experience, acumen with respect to solid construction techniques for UAVs, and spooky knowledge of radios I will continue to learn from. Thank you especially for helping getting our plane(s) off the ground!

To Mariano Lizarraga, the designer of the SLUGS autopilot, who has selflessly provided great assistance in person, via email, and through other communication channels to help me work my way through a child's growing pains of getting up to speed with an excellent GNC system.

To Samuel Toepke and Brad Watanabe, fellow graduate students and coworkers with whom I have shared friendship, professional and academic growth, as well as growing pains. Also to Bryant Mairs, whose advanced C, Matlab, Simulink, git, and hardware engineering knowledge and nutty humor got me over many a slump on the road to thesis. Thanks to Pavlo Manovi for numerous engineering and RC airplane discussions, not to mention for acting as safety pilot for our planes. And to Jeremy Gottlieb, fellow mature graduate student and Monterey local, with whom I have shared many enjoyable carpool conversations on robotics, probability and statistics, family, and life.

To Carol Mullane, graduate advising at UCSC, to all the teachers from whom I have taken classes or assisted in the manner of a teaching assistant, and to all staff with whom I have interacted, thank you for welcoming me and treating me with respect, dignity, and professionalism in this excellent learning environment which we know and love as the University of California, Santa Cruz.

And lastly, to my family, current and future. :-) I love you and look forward to spending more time with you soon.

Figure 1: The Mentor RC aircraft in flight. This research platform enables UAV research at the Autonomous Systems Laboratory at the University of California, Santa Cruz.

1 Introduction

1.1 Motivation

1.1.1 UAVs

Unmanned aerial vehicles (UAVs) represent an increasingly important and prolific technology in today's world. These pilotless aircraft allow the accurate and precise delivery of sensing and measurement devices, military, scientific, commercial, and other payloads to locations nearly anywhere on Earth without the need for onboard personnel. For example, NASA has flown the Global Hawk UAV into pre-hurricane storm regions to learn more about tropical storm formation [36]. Other climate scientists have used this technology to study oceanic and atmospheric interactions in Antarctica to further understanding of the Earth's climates [12].

In addition to climate research, UAVs routinely assist in search and rescue operations. In Salina, Kansas, this technology plays a key role in a new post-tornado search and recovery training program [15]. The National Park Service spends over $4 million a year searching for hikers lost in the wilderness; UAVs assist in these

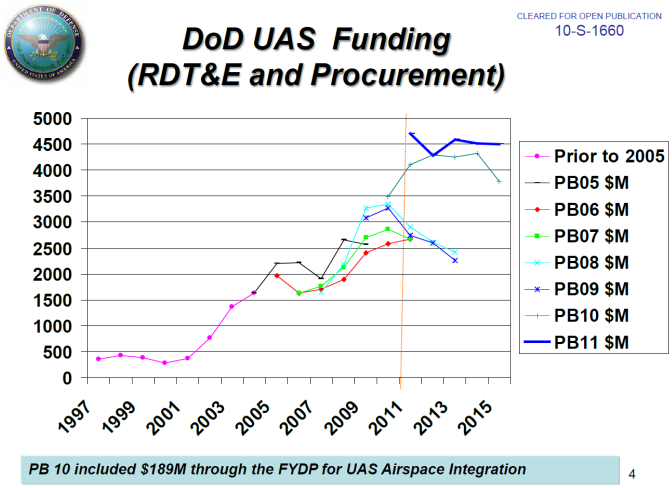


Figure 2: The U.S. DoD plans to invest more in UAS. PB = President's Budget. [72]

efforts by autonomously analyzing the terrain and determining likely routes for lost persons, potentially reducing the time, cost, and effort expended in search and rescue operations [3]. UAVs have myriad commercial uses, such as livestock and oil pipeline monitoring, as well as the more obvious security applications.

The public sector also employs unmanned aerial systems. Recently, the International Association of Chiefs of Police (IACP) adopted guidelines for the use of UAS, specifically on how to use these systems "safely, responsibly, and with respect to an individual's privacy" [60]. This indicates an anticipated increased presence of this technology.

The military relies extensively on UAVs. These vehicles can save lives by not putting pilots in harm's way over hostile territories. The 1960 U2 incident involving the capture of a U.S. spy plane piloted by Gary Powers over Soviet Union airspace was one particularly motivating scenario for the development of unmanned vehicles. This technology has been employed since 2001 by the U.S. military forces for the continued War on Terrorism, and the projected increase in funding indicates significant future interest and activity. Fig. 2 shows a trend of increasing government investment in UAS, especially after the terrorist incident of 9/11.

The term **UAV** typically focuses on the airframe-specific components, while the label **UAS** (Unmanned Aerial System) extends the definition of a UAV to include associated mission-critical systems such as the ground-control station, communication links, and so on. Industry forecasts predict exponential growth of UAV/UAS operations over the next decade. It is estimated that over 20,000 UAS will be produced in the United States alone, while 35,000 are estimated to be produced worldwide from 2010 to 2019 [54], [17]. The wide adoption of this technology stems from the multitude of different applications which have been successfully incorporated into UAVs. Moreover, this trend seems to be accelerating.

The U.S. Department of Defense has developed a comprehensive plan, extending to the year 2030, for missions involving unmanned systems (including UAS). The military applications include target identification and designation, counter-mine warfare, and surveillance and reconnaissance, including CBRNE (Chemical, Biological, Radiological, Nuclear, Explosive) [14].

Sept. 30, 2015 marks the proposed date when the FAA will adopt NextGen (Next Generation Air Transportation System), a program that will usher in a new era of satellite-based (GPS) navigation for commercial aircraft, leaving behind the extant, antiquated, ground-based radio system. A significant part of this Congressional bill is that commercial, private, and military UAVs will be granted greater access to the national airspace (NAS), currently populated only by manned aircraft. Anticipation of NextGen has mobilized scores of professionals from diverse fields to prepare for the associated legal, technical, R & D, economic, environmental, safety, operational, and communications challenges. For example, NASA, the Department of Defense, the Department of Commerce, the Department of Homeland Security, and the FAA are collaborating to accomplish full integration of UAS into the NAS. [4] [70] Substantial changes in the national (and international) airspace are just around the corner, and unmanned air systems will play a key role.

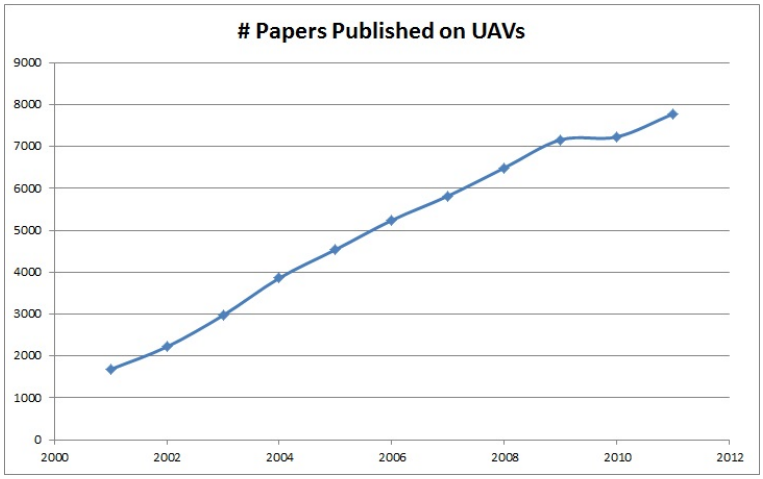


Figure 3: Google Scholar statistics

These changes and challenges call for significant contributions from the R & D community, a continuation of the trend depicted in Fig. 3: an increase in the number of UAV-related publications. The University of California, Santa Cruz's Autonomous Systems Lab (UCSC ASL) [25] as well as the Control & Optimization Laboratories at the Naval Postgraduate School (NPS) [50] are two small, but significant, local contributors to this research effort. Both facilities use the research platform shown in Fig. 1, an inexpensive, COTS RC airplane which allows the rapid development and testing of various controls and flight algorithms.

1.1.2 Attitude Estimation

The nature of the mission to be performed by the UAV, be it surveillance, payload delivery, search and rescue assistance, research, etc. ultimately delineates the operational requirements of the vehicle, such as its size, power constraints, endurance, speed, and, ultimately, cost. The greater the endurance required, the larger the aircraft weight due to increased fuel needs. The larger the airframe, the more airspace needed for takeoffs and landings. The more dynamic maneuvers to be performed, the more sophisticated flight control mechanisms required, and so on.

Since UAVs operate without a human pilot onboard, they require additional electron-

ics to perform the flight control tasks typically performed by humans: an electronic brain known as the autopilot (AP). For the autopilot to control the aircraft and for the UAV to accomplish its mission, it is necessary to determine fundamental information about the location and orientation of the vehicle in space. The focus of this thesis is on attitude determination. The **attitude** of the airframe describes its 3D orientation in space with respect to some reference frame, such as a North-East-Down frame on Earth. A key challenge with estimating the aircraft's attitude is that no sensor currently exists which can directly measure the vehicle's orientation [24] - other sensors must be employed and likely combined. For manned and more expensive unmanned aircraft, sophisticated sensors make this job easier, more reliable, and more accurate. For smaller vehicles and research projects, the size, weight, and/or cost of such sensors makes their use unrealistic. Instead, lower cost sensors with less sophistication are employed. In an attempt to compensate for the limited processing power, memory, and bandwidth of these sensors, system designers create sophisticated algorithms to meet research and mission objectives. This thesis investigates and compares several of these algorithms.

1.2 Contributions

This thesis explores computational attitude estimation algorithms, parameterizations (mathematical representations of attitude), and implementations for low-cost unmanned aerial vehicles. To the best of the author's knowledge, no extant work has examined these topics in such a way. Integration methods are frequently glossed over in discussion, infrequently addressing the specific advantages or disadvantages behind the parameterization or integration method chosen. In this thesis, sixteen different attitude propagation and parameterization methods are examined in depth, expanding upon the benchmark attitude filter currently existing in the SLUGS UAV Autopilot, developed by the UCSC ASL. Parameterization methods include Euler angles, quaternions, the direction cosine matrix, and the angle-axis formulation, while propagation methods focus on forward Euler integration and the matrix ex-

ponential. Furthermore, several angular velocity constraints are tested. Algorithm performance is evaluated through simulation and external flight data, laying the groundwork for future validation and verification through hardware-in-the-loop simulations as well as flight testing.

1.3 Thesis Structure

The remainder of this thesis is arranged in the following sections:

1. Introduction
2. Mathematical Concepts and Tools (Attitude Theory)
3. Attitude Systems
4. Previous Work
5. Comparison of Attitude Parameterizations and Propagation Methods
6. Conclusions and Future Work

The theoretical concepts and system-level aspects of attitude determination are covered in earlier chapters to prepare the reader for the previous work discussion later.

2 Mathematical Concepts and Tools

Nomenclature

wrt - with respect to,

I - the identity matrix, $I \in \mathbb{R}^{3\times3}$,

$\{I\}$ - the inertial frame,

$\{B\}$ - the body frame,

$\{E\}$ - the estimator or filter frame,

R^T - the transpose of matrix R,

R^{-1} - the inverse of square matrix R,

$detR$ - the determinant of matrix R,

${}^B_I R : \{I\} \rightarrow \{B\}$ - a rotation, mapping, or relative orientation of B wrt I,

${}^I_B R : \{B\} \rightarrow \{I\}$ - a rotation, mapping, or relative orientation of I wrt B,

${}^i v$, ${}^b v$ - vectors expressed in the inertial and body frames, respectively,

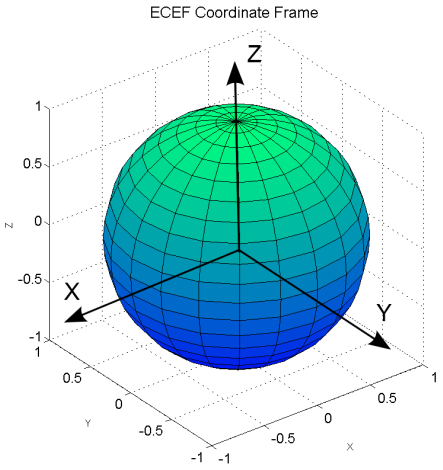
u - a unit vector,

$\|v\|$ - the Euclidean, 2-norm of a vector, i.e. its magnitude.

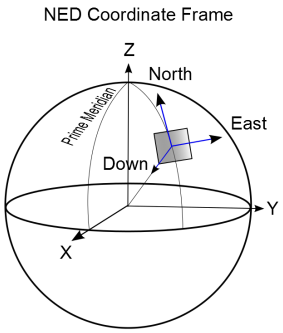
This chapter covers background topics in attitude theory, critical to understanding the attitude complementary filter employed in the SLUGS AP as well as the core research described in this thesis.

2.1 Coordinate Systems and Reference Frames

Since attitude describes the orientation between two frames, the concept of a **reference frame** holds great importance to the discussion of attitude. Reference frames define a set of axes within which the orientation of various objects may be described. Reference frames may be considered to be models of physical references, such as the Earth, the aircraft, the solar system, etc. UAV applications commonly use the inertial or Earth frame, the local frame, and the body frame. The UAV is typically treated as a rigid body within these frames [77], [26].



(a) Earth-Centered, Earth-Fixed (ECEF) frame.



(b) Local Tangent Plane (LTP) or North-East-Down (NED) local frame.

Figure 4: Two commonly used inertial reference frames: ECEF and LTP/NED.

Coordinate systems establish the association of the frame with Euclidean n-dimensional space. As such, coordinate systems help quantify a reference frame through a vector's sense of direction and magnitude, facilitating numeric analysis and calculations [77].

2.1.1 Inertial or Earth frame

Defining the appropriate inertial frame {I} for an application requires careful consideration. The inertial frame for an application is typically that frame in a set not undergoing significant acceleration [1]. A commonly used inertial frame is the Earth frame, such as the Earth-Centered, Earth-Fixed frame (ECEF). Here the center of the coordinate system coincides with the center of the Earth. Fig. 4a shows the basic construction of this frame: the Z-axis points from the center of the Earth out the North Pole, the X axis points out to the Prime Meridian at the Earth's surface, while the Y axis is at right angles to the previous two.

It should be noted that the ECEF frame is not a true inertial frame, since the Earth is both rotating in space and revolving around the sun - it is constantly accelerating. However, most UAVs follow near-Earth trajectories at speeds several

orders of magnitude less than the Earth's rotation; therefore, first, second, and third order approximations of the Earth as an inertial reference for such UAVs are acceptable, as the Earth represents a relatively stationary reference. Further discussion of the ECEF frame is presented in the excellent reference [61].

2.1.2 Local frame

A local frame, such as the NED (North-East-Down) frame or LTP (Local Tangent Plane), is an example of a reference frame often used as the inertial frame for near-Earth aircraft. Fig. 4b graphically describes this orthogonal, right-handed system in which the **North** vector points North and parallel to the earth's surface, the **East** vector points East and parallel to the Earth's surface, and the **Down** vector points towards the center of the Earth, normal to its surface. The local frame is important because the vehicle's **attitude** is often expressed as **the orientation of the vehicle's body frame with respect to the local frame**. Also, the SLUGS AP employs the local NED frame (or **LTP**) for attitude estimation.

2.1.3 Body frame

The body frame, {B}, describes the right-handed system whose origin is at the center of mass of the vehicle and whose axes are:

- X axis - out the nose of the aircraft.
- Z axis - down, perpendicular to the X axis along the axis of longitudinal symmetry, and
- Y axis - the cross product of the previous two vectors, usually out the right wing.

Fig. 5 illustrates the body frame axes on a fighter jet. The body frame is important because most of the UAV's sensors measure vehicle states in the body frame.

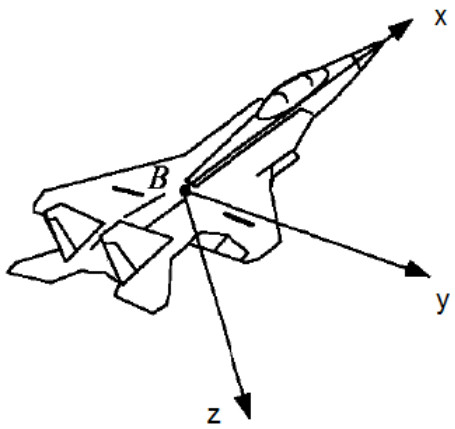


Figure 5: Body frame of the flight vehicle

2.1.4 Estimation frame

The final frame to be discussed is not a "true" frame, per [77], but a coordinate system: the estimation frame. The estimation frame is not considered a reference frame because it models a logical, not a physical, reference. Its use arises in the following context. Solving for the aircraft's attitude involves estimating the body-frame orientation with respect to the inertial frame, i.e. the rotation $I \rightarrow B$ or ${}^{B}_{I}R$. Ideally, the estimation and body frames coincide, but in practice, attitude estimation errors cause some small offset. Thus, the estimate is in a different frame from the body frame. The estimation frame represents the rotation $I \rightarrow E$, i.e. ${}^{E}_{I}R$. More details will be provided in Sec. 2.5.2 and are also available in Ref. [44].

2.2 Attitude Parameterizations

The orientation of a body in space can be expressed in a number of different ways, each with its own benefits and drawbacks. Common to all of these attitude parameterizations is the expression of a rotation in space from one frame to another. The literature on control systems and aviation has heavily covered attitude parameterizations [77], [63], [23], [53], [67]. This section will summarize the various parameterizations used in this thesis.

2.2.1 Direction Cosine Matrix

A common way to express the orientation of a rigid body in 3-dimensional space is by specifying the components of the 3 axes of the body frame with respect to the reference (or inertial) frame. Given that the domain of operation for UAVs is 3-dimensional space, 3 basis vectors are required to uniquely describe a position or orientation within that space. To specify these 3 basis vectors uniquely requires 9 total components: 3 axis vectors with 3 components each. By assembling these 3 vectors into a matrix, the canonical **attitude matrix** is created, which also goes by the more commonly-used name of the **direction cosine matrix** or **DCM**:

$$ {}^{B}_{I}R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{1} $$

The matrix in Eq. 1 is called the direction cosine matrix because each entry represents the cosine of the angle between a body unit vector and a reference axis unit vector. For example, r_{12} is the cosine of the angle between the first axis in the body frame and the second axis in the reference frame. Furthermore, each **column** of this matrix is a basis vector of the inertial frame in the body frame. Correspondingly, each **row** of the matrix is the transpose of a basis vector of the body frame coordinated in the inertial frame [77]. In short, the columns represent *inertial* $\rightarrow$ *body* mappings, while the rows provide *body* $\rightarrow$ *inertial* mappings.

Only 3 of the 9 components of the DCM are independent; the other 6 are in place to enforce two constraints required of any rigid body transformation [49]:

1. preserve vector lengths (unit norms), and
2. preserve dot products (and thus angles).

Item 1 is established by the fact that each column (and row) of the DCM describes a unit vector. Item 2 arises from the fact that any two columns (or rows) of the

DCM are orthogonal to one another. Thus, the DCM R describes an orthonormal basis of vectors [65]. As such, R has several useful properties:

1. $R^{-1} = R^T$. To transform an orientation from the body frame to the inertial frame, using R^T in matrix multiplications rather than calculating a matrix inverse yields significant computational savings.

2. $RR^T = R^T R = I$. The corresponding equivalence $I - R^T R$ can be used as a constraint enforced in computations to arrest the DCM's drift out of its orthonormality. This effect often occurs when performing repeated, floating point operations on rotation matrices due to truncation and other errors.

The key feature to remember about the direction cosine matrix is that it maps vectors from a reference frame (inertial) to another frame (body).

2.2.2 Euler Angles

The next attitude representation method, the Euler angles, is intuitive, pilot-centric, and has a long history of use in aviation. An Euler angle sequence describes an ordered sequence of rotations from the reference frame to arrive at the final orientation of the body frame. The aerospace sequence of 321 or ZYX Euler angles, depicted in Fig. 6, begins with the vehicle body frame coincident with the reference frame of choice; on Earth, a frequently used convention orients the aircraft with wings level, nose pointing north. The vehicle frame is rotated with respect to the reference frame's Z axis to create the **heading** angle, ψ. This angle is also referred to as the **yaw**. The vehicle frame is then rotated with respect to the new Y axis to create the **elevation** or **pitch** angle, θ. Lastly, the frame is rotated once more, but along the newest X axis to generate the **bank** or **roll** angle, ϕ [18] [38]. It is important to note that each rotation is performed about an axis of the moving system rather than the fixed reference. Accordingly, the precise order of rotations identifies a unique permutation from a total of 12 possibilities. This gives rise to a unique mathematical formula for extracting the correct ψ, θ, and ϕ angles from the resulting orientation.

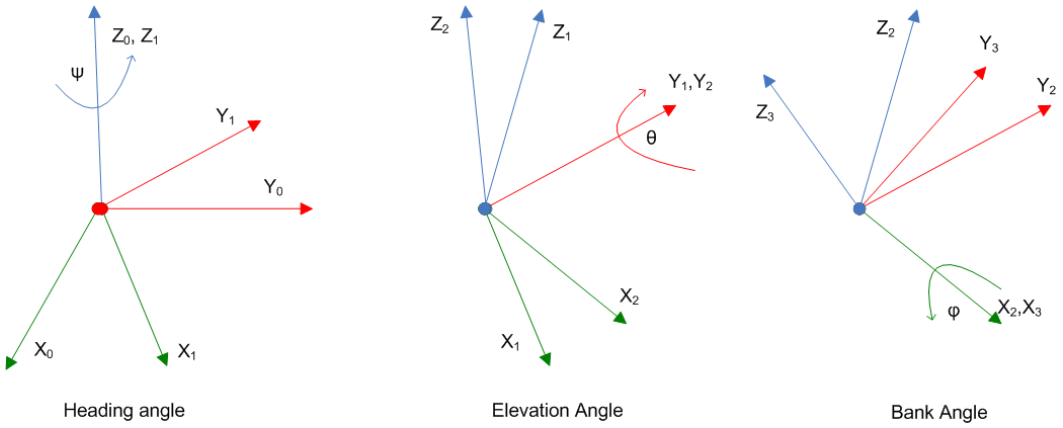


Figure 6: Euler angles: a series of consecutive rotations. Yaw-pitch-roll sequence shown.

The above series of rotations can be encoded as a series of multiplications by the respective rotation matrices [18]:

$$ {}^{B}_{A}R_{ZYX} = \begin{bmatrix} c\psi & -s\psi & 0 \\ s\psi & c\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\theta & 0 & s\theta \\ 0 & 1 & 0 \\ -s\theta & 0 & c\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\phi & -s\phi \\ 0 & s\phi & c\phi \end{bmatrix} \tag{2} $$

where $c\alpha = \cos(\alpha)$ and $s\beta = \sin(\beta)$. After multiplying through, the final orientation is given by the following [18]:

$$ {}^{B}_{A}R_{ZYX} = \begin{bmatrix} c\psi c\theta & c\psi s\theta s\phi - s\psi c\phi & c\psi s\theta c\phi + s\psi s\phi \\ s\psi c\theta & s\psi s\theta s\phi + c\psi c\phi & s\psi s\theta c\phi - c\psi s\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix} \tag{3} $$

To extract the desired Euler angles from the above matrix, the following inverse trigonometric functions can be used:

$$ \phi = atan2(R_{32}, R_{33}) \tag{4} $$

$$ \theta = -asin(R_{31}) \tag{5} $$

$$ \psi = atan2(R_{21}, R_{11}) \tag{6} $$

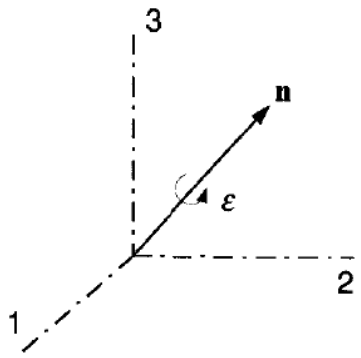


Figure 7: General Rotation described by axis and angle of rotation, [77]

where R_{ab} above refers to the element of the direction cosine matrix ${}^{B}_{A}R_{ZYX}$ at row a, column b. Because of the formulation of Eqs. 4 and 6, mathematical singularities exist at $\theta = \pm 90°$, where the pitch angle is such that the yaw and roll angles are indistinguishable (i.e. parallel). This issue is known as **gimbal lock** and represents one of the key drawbacks to using the Euler angle parameterization.

2.2.3 Angle-Axis

The angle-axis representation of attitude is also known as the **Euler-axis** or **Eigen-axis** parameterization. The name "Eigen-axis" applies due to the observation that the Euler axis $\vec{a}$ is the eigenvector of ${}^{B}_{I}R$ associated with the eigenvalue $+1$ [34]. Also, the naming convention relates to Euler's theorem of rotation, which states that four parameters describe an arbitrary rigid rotation [77]. Three of these parameters describe the axis of rotation, $\vec{n}$ or $\vec{a}$, while the fourth parameter describes the angle of rotation about that axis, ϵ or Φ; Fig. 7 depicts this 3-dimensional construct. For the angle-axis scheme, components are encoded as $\vec{a} = [a_1\ a_2\ a_3]^T$, the axis unit vector, and Φ, known as the **principal Euler angle** [34].

Shuster and Hall offer excellent coverage of this parameterization in [63] and [34].

2.2.4 Quaternions

The quaternion representation of attitude has seen great use in the aerospace industry in the last 30 years [19]. The quaternions are also known as the **Euler**

symmetric parameters [63] or the **Euler-Rodrigues quaternion**[53]. Like the angle-axis scheme, the set of quaternions encodes four rotational parameters through a vector component, $\overrightarrow{v} \in \mathbb{R}^3$, and a scalar component, $s \in \mathbb{R}$, as follows [38], [44], [77]:

$$\mathbb{Q} = \{q = \begin{bmatrix} s \\ \overrightarrow{v} \end{bmatrix} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} cos(\epsilon/2) \\ sin(\epsilon/2) \cdot n_1 \\ sin(\epsilon/2) \cdot n_2 \\ sin(\epsilon/2) \cdot n_3 \end{bmatrix} : \|q\| = 1\}, \text{where} \quad \vec{n} = \begin{bmatrix} n_1 \\ n_2 \\ n_3 \end{bmatrix} \tag{7}$$

Note that this usage follows the Parkinson notation of scalar part first, vector part second [26], [38], [44]. Quaternions can also be expressed in terms of i, j, and k unit vectors:

$$q = q_0 + \boldsymbol{q} = q_0 + \boldsymbol{i}q_1 + \boldsymbol{j}q_2 + \boldsymbol{k}q_3 \tag{8}$$

Quaternions form a mathematical group under the operation of the quaternion product:

$$q_1 \otimes q_2 = \begin{bmatrix} s_1 s_2 - v_1 \cdot v_2 \\ s_1 v_2 + s_2 v_1 + v_1 \times v_2 \end{bmatrix} \tag{9}$$

Furthermore, the sum of the squares of the components equals unity:

$$\sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} = 1. \tag{10}$$

Like the DCM's orthonormality constraint, an issue with quaternions can be drift out of the Eq. 10 constraint, which must be checked through renormalization. The quaternion parameterization is advantageous in that there are fewer numbers involved than with the DCM, rotation is a linear operation [38], and there are no singularity issues as with Euler angles. On the other hand, quaternions do not map easily to human intuition regarding rotations or to control algorithms. As a result, the control algorithms employed on the SLUGS AP use Euler angles, after an in-

termediate conversion step, even while the attitude is parameterized in quaternions. Also, system attitude initialization requires expressing the quaternion components in terms of Euler angles through a series of four trigonometric equations with half angles.

Two key differences between the angle-axis and quaternion parameterizations are that (a) quaternions use half-angles whereas an angle-axis attitude uses the complete angle, and (b) the angle-axis representation suffers from singularities, namely at orientations of $\Phi = 0$ or 2π. This will be more apparent in the kinematic equations described in the subsequent section.

2.2.5 Conversion Formulas

This thesis focuses on attitude propagation methods using the above four representations of attitude. The experimental procedures described later in this paper required conversion formulas between parameterizations to create a common framework for comparisons. For example, the results of all the attitude propagation algorithms were evaluated according to the Euler angle expression of attitude. This section lists a number of these utility formulas.

Euler Angles to DCM [18]

$$R_{ZYX} = \begin{bmatrix} c\psi c\theta & c\psi s\theta s\phi - s\psi c\phi & c\psi s\theta c\phi + s\psi s\phi \\ s\psi c\theta & s\psi s\theta s\phi + c\psi c\phi & s\psi s\theta c\phi - c\psi s\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix} \tag{11}$$

DCM to Euler Angles

$$\phi = atan2(r_{32}, r_{33}) \tag{12}$$

$$\theta = -asin(r_{31}) \tag{13}$$

$$\psi = atan2(r_{21}, r_{11}) \tag{14}$$

Quaternions to DCM [38]

$$R_{ZYX} = \begin{bmatrix} 2q_0^2 - 1 + 2q_1^2 & 2q_1q_2 + 2q_0q_3 & 2q_1q_3 - 2q_0q_2 \\ 2q_1q_2 - 2q_0q_3 & 2q_0^2 - 1 + 2q_2^2 & 2q_2q_3 + 2q_0q_1 \\ 2q_1q_3 + 2q_0q_2 & 2q_2q_3 - 2q_0q_1 & 2q_0^2 - 1 + 2q_3^2 \end{bmatrix} \tag{15}$$

DCM to Quaternions [38]

$$q_0 = \frac{1}{2}\sqrt{trace(R) + 1} \tag{16}$$

$$q_1 = \frac{r_{23} - r_{32}}{4q_0} \tag{17}$$

$$q_2 = \frac{r_{31} - r_{13}}{4q_0} \tag{18}$$

$$q_3 = \frac{r_{12} - r_{21}}{4q_0} \tag{19}$$

where $R = R_{ZYX}$ and $trace(R) = r_{11} + r_{22} + r_{33}$.

Angle-Axis to DCM [38]

$$R_{ZYX} = \begin{bmatrix} a_1^2 + (a_2^2 + a_3^2)\cos\Phi & a_1a_2(1 - \cos\Phi) - a_3\sin\Phi & a_1a_3(1 - \cos\Phi) + a_2\sin\Phi \\ a_1a_2(1 - \cos\Phi) + a_3\sin\Phi & a_2^2 + (a_3^2 + a_1^2)\cos\Phi & a_2a_3(1 - \cos\Phi) - a_1\sin\Phi \\ a_3a_1(1 - \cos\Phi) - a_2\sin\Phi & a_2a_3(1 - \cos\Phi) + a_1\sin\Phi & a_3^2 + (a_1^2 + a_2^2)\cos\Phi \end{bmatrix} \tag{20}$$

DCM to Angle-Axis [63]

$$\Phi = \cos^{-1}\frac{trace(R) - 1}{2} \tag{21}$$

$$\vec{a} = \frac{1}{2\sin\Phi}\begin{bmatrix} r_{23} - r_{32} \\ r_{31} - r_{13} \\ r_{12} - r_{21} \end{bmatrix} \tag{22}$$

where Φ is the principal Euler angle and $\vec{a}$ is the Euler-axis.

Euler Angles to Quaternions [38]

$$q_0 = \cos\frac{\psi}{2}\cos\frac{\theta}{2}\cos\frac{\phi}{2} + \sin\frac{\psi}{2}\sin\frac{\theta}{2}\sin\frac{\phi}{2} \quad (23)$$

$$q_1 = \cos\frac{\psi}{2}\cos\frac{\theta}{2}\sin\frac{\phi}{2} - \sin\frac{\psi}{2}\sin\frac{\theta}{2}\cos\frac{\phi}{2} \quad (24)$$

$$q_2 = \cos\frac{\psi}{2}\sin\frac{\theta}{2}\cos\frac{\phi}{2} + \sin\frac{\psi}{2}\cos\frac{\theta}{2}\sin\frac{\phi}{2} \quad (25)$$

$$q_3 = \sin\frac{\psi}{2}\cos\frac{\theta}{2}\cos\frac{\phi}{2} - \cos\frac{\psi}{2}\sin\frac{\theta}{2}\sin\frac{\phi}{2} \quad (26)$$

Quaternions to Euler Angles [38]

$$\phi = atan2(2q_2q_3 + 2q_0q_1, 2q_0^2 + 2q_3^2 - 1) \quad (27)$$

$$\theta = -asin(2q_1q_3 - 2q_0q_2) \quad (28)$$

$$\psi = atan2(2q_1q_2 + 2q_0q_3, 2q_0^2 + 2q_1^2 - 1) \quad (29)$$

For the correct determination of the bank and heading angles above, quadrant checks should be performed (as with $atan2(y, x)$ or comparable routines).

Angle-Axis to Quaternions [34]

$$q_0 = \cos\frac{\Phi}{2} \quad (30)$$

$$\vec{q} = \vec{a}\sin\frac{\Phi}{2} \quad (31)$$

Quaternions to Angle-Axis

$$\Phi = 2\cos^{-1}(q_0) \quad (32)$$

$$\vec{a} = \frac{2\sin^{-1}(\vec{q})}{\Phi} \quad (33)$$

2.3 Kinematics of Rotation

2.3.1 Overview

Kinematics are central to the topic of attitude determination in that they describe the differential equations which govern rigid body motion, including rotations. The

current attitude of the UAV is determined by propagating or integrating the appropriate nonlinear ordinary differential equations with respect to time, in a manner similar to integrating the linear velocity of a rigid body to obtain its position.

In order to propagate the vehicle's attitude from the previous time step to the current time step, the appropriate kinematic equations must be described. Such equations express the time rate of change of the vehicle's attitude through a relationship between the previous attitude of the vehicle and some expression of the vehicle's angular velocity, $\vec{\omega}$. For example, the direction cosine matrix employs the equation ${}^{B}_{I}\dot{R} = {}^{B}_{I}R\ [{}^{I}_{B}\Omega\times]$. For the DCM as well as quaternion schemes, a skew-symmetric (3×3 or 4×4) $\Omega\times$ matrix of angular velocity components is constructed. For the angle-axis representation, a skew-symmetric matrix of the eigenaxis components is used. For the Euler angle representation, the angular velocities are typically expressed as a column vector of the p, q, and r components associated with the rates of the x, y, and z body-frame axes, that is to say:

$$\vec{\omega} = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{34}$$

2.3.2 Parameterized Formulas

This section presents kinematic formulas for the four attitude parameterizations employed in this thesis.

Euler Angles [75]

$$\dot{\Phi} = \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \frac{1}{\cos\theta} \begin{bmatrix} \cos\theta & \sin\phi\sin\theta & \cos\phi\sin\theta \\ 0 & \cos\phi\cos\theta & -\sin\phi\cos\theta \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{35}$$

Note the singularity related to the condition: $\cos\theta = 0 \Rightarrow \theta = \pm 90°$.

DCM [6]

$$ {}^{B}_{I}\dot{R} = {}^{B}_{I}R \, [{}^{I}_{B}\Omega\times], \tag{36} $$

where

$$ {}^{I}_{B}\Omega\times = \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix}, \tag{37} $$

Here $\Omega\times$ refers to the **skew-symmetric matrix** of angular velocities: p, q, and r.

Quaternions [77]

$$ \dot{q} = \begin{bmatrix} \dot{q_0} \\ \dot{q_1} \\ \dot{q_2} \\ \dot{q_3} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & -p & -q & -r \\ p & 0 & r & -q \\ q & -r & 0 & p \\ r & q & -p & 0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \tag{38} $$

Angle-Axis [63]

$$ \dot{\Phi} = \vec{a} \cdot \vec{\omega} \tag{39} $$

$$ \dot{\vec{a}} = \frac{1}{2}[\vec{a} \times \vec{\omega} - \cot\left(\frac{\Phi}{2}\right) \vec{a} \times (\vec{a} \times \vec{\omega})] \tag{40} $$

where

$$ \vec{a}\times = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}, \tag{41} $$

Note the singularity related to the condition: $\cos\Phi = 0 \Rightarrow \cot\Phi = \infty \Rightarrow \Phi = 0°$ or $\Phi = 360°$.

2.4 Attitude Propagation

The previous section laid out the kinematic equations for the time rate of change of attitude. This section focuses on the theoretical underpinnings of various methods by which to integrate these equations to determine the attitude at the current time step.

In terms of the mathematics involved, attitude propagation can be described by a category of methods used to generate solutions to initial value problems (IVPs). The attitude propagation problem is an initial value problem because all subsequent system states rely on the initial state or initial condition (IC). The initial condition is the starting attitude of the vehicle, often neutral (0°) pitch and roll angles and an application-dependent setting for the yaw angle. On Earth, a common convention is to orient the aircraft wings-level, pointing North.

Numeric approximations rather than analytic solutions are employed due to (a) the discrete vs. continuous sampling architecture of real-time computing systems and (b) the lack of availability of an *a priori* analytic, continuous function expressing the exact angular velocity vector experienced by the aircraft.

Direct Integration If the attitude of the UAV can be represented as a continuous, differentiable function $f(t)$, and the value of this function is known at point t_n, then the value of the function at the point $t_{n+1} = t_n + h$, where h is the fixed step size, can be represented by the Taylor series form:

$$f(t_{n+1}) = f(t_n) + hf'(t_n) + \frac{h^2}{2!}f''(t_n) + \frac{h^3}{3!}f'''(t_n) + \dots \tag{42}$$

Truncated approximations to Eq. 42 are often used for real-time applications. Local and global **discretization** or truncation errors result; **local** errors refer to those associated with the current time step, while **global** errors accumulate over time. A

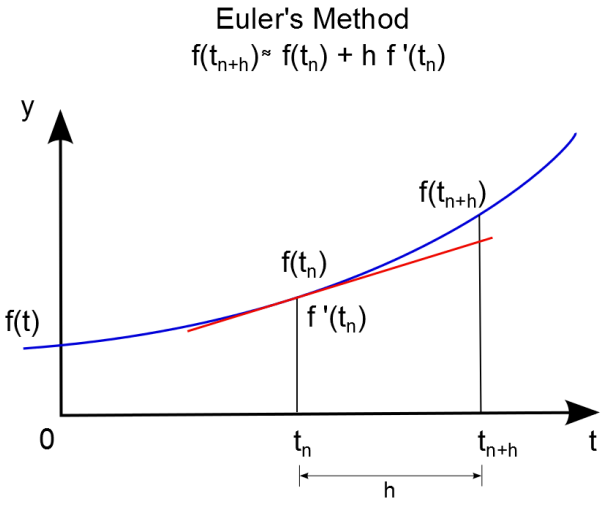


Figure 8: Illustration of how the forward Euler method approximates functional values, via a tangent line f'(t) (red) to the function f(t) (blue) at the previous time step.

frequently used approximation Eq. 42 is the **forward Euler** method, as follows:

$$f(t_{n+1}) = f(t_n) + hf'(t_n) \tag{43}$$

Fig. 8 depicts a graphical interpretation of Euler's method. The (blue) curve f(t) represents the time-varying attitude, the (red) line f'(tn) represents the time rate of change of attitude. Regardless of the parameterization, if the angular velocity vector is constant, then the time rate of change of attitude will be constant, due to the kinematic equations. Because this method essentially follows a line of constant slope, tangent to the function, from the initial point, its use constitutes an assumption that the angular velocity between time steps is constant. The larger the value of h and the more the function changes between time steps, the more this method's approximation diverges from the exact solution. Key advantages to this first-order method are simplicity of implementation and computational efficiency: the function $f'(t)$ (the kinematic equations) need be evaluated only once per time step. A slow speed of convergence and global discretization error proportional to $O(h)$ are the method's key drawbacks [33]. An error of $O(h)$ means that the error approaches 0 at the same rate that h approaches 0. The smaller the value of h, the better

performance this method offers.

Forward Euler methods fall into the family of integration methods known as **one-step methods**, so called because the evaluation of $f(t_{n+1})$ requires knowledge of only the previous function value, $f(t_n)$, and not values before that ($f(t_{n-1}), f(t_{n-2})$, etc.)[73]. As previously stated, the forward Euler method performs a single function evaluation. However, an important class of one-step methods, known as **Runge-Kutta** methods, performs multiple function evaluations of $f'(t)$ at different points in the integration interval for every time step. Note that in order to perform such evaluations, the input must either be analytic or possess function values at every evaluation point. The order or stage of a Runge-Kutta method indicates the number of functional evaluations performed; second-order (Euler-Cauchy) and fourth-order methods are often employed. These methods offer faster convergence than the forward Euler method; for example, the second-order Runge-Kutta method has a local discretization error on the order of $O(h^2)$ [10].

The family of **multi-step methods** requires knowledge of values of $f(t)$ from previous time steps. These methods are also called predictor-corrector methods. First, the method evaluates the attitude of the next time step, $f(t_{n+1})$, explicitly via the predictor phase. Next, an implicit method known as the corrector further refines the value of $f(t_{n+1})$ using the new value of $f'(t_{n+1})$ [73]. This process is then repeated. These methods can have small truncation errors and good stability properties. **Adams-Moulton** or **Adams-Bashforth** methods are commonly used. Sample equations for first-order systems are as follows:

$$py_{n+1} = y_n + \frac{h}{24}(55y'_n - 59y'_{n-1} + 37y'_{n-2} - 9y'_{n-3}) \tag{44}$$

which describes a predictor method, while

$$y_{n+1} = y_n + \frac{h}{24}(9py'_{n+1} + 19y'_n - 5y'_{n-1} + y'_{n-2}) \tag{45}$$

is a corrector for a combined Adams-Bashforth-Moulton method [9].

Matrix Exponential The matrix exponential method offers an approximate closed form solution to the kinematics equations expressed in the previous section. This method arises out of the solution to the differential equation $\dot{X} = Ax$ from linear dynamical systems, namely

$$x(t) = e^{At}x(0), \tag{46}$$

where $x(t) \in \mathbb{R}^n$, $A \in \mathbb{R}^{n\times n}$ (A is square) [7][65], and

$$e^{At} = \sum_{j=0}^{\infty} \frac{A^j}{j!} = I + At + \frac{1}{2}(At)^2 + \frac{1}{6}(At)^3 + \dots \tag{47}$$

where t is the sampling rate. Eq. 46 represents a continuous-time solution. A discrete-time, recursive solution is

$$R(t_{n+h}) = e^{hA}R(t_n), \tag{48}$$

more appropriate to the digital hardware. In Eq. 48, h represents the sampling period and t_n is an arbitrary point in time. For applications to attitude estimation, the skew-symmetric matrix Ω is frequently substituted for the matrix A in Eq. 48. Discrete, real-time implementations of this function typically collect odd or even terms from the infinite series to develop a closed form solution with only two sinusoids, a form often called the Rodrigues' formula. This expression will be discussed later in the implementation section.

If a constant body-rate $\vec{\omega}$ is assumed between time steps, errors in this formulation are on the order of h^2. If a time-averaged angular velocity ($\overline{\Omega}$) is used, then the error is on the order of h^3 [73]. This $\overline{\Omega}$ formulation will also be discussed in the implementation section.

2.5 Attitude Determination using Reference Vector Feedback

2.5.1 Wahba's Problem

In 1965, Grace Wahba made a seminal contribution to attitude determination techniques by proposing, within the domain of satellite navigation, the mathematical problem of identifying the direction cosine matrix R which minimizes in a least squares sense the following cost function:

$$J(R) = \frac{1}{2}\sum_{k=1}^{N} a_k \|\hat{W}_k - R\hat{V}_k\|^2 \tag{49}$$

where $\hat{W}_k$, $k = 1 \dots N$, represents a set of N measured directions or reference vectors in the body frame of the satellite, $\hat{V}_k$, $k = 1 \dots N$, represents the corresponding inertial frame reference vector directions, and a_k, $k = 1 \dots N$, is a set of positive weights [71], [64]. This expression of Wahba's problem describes the fundamental action taken by many attitude determination algorithms: use a set of reference vectors to determine the orientation of the body frame with respect to another reference frame. Since first proposed, this problem has sparked numerous solutions, including Singular Value Decomposition (SVD) approaches [45], the Davenport method, QUEST, ESOQ, FOAM algorithms, and many others [19], [64].

Recall from Sec. 2.1.4 that, due to practical issues, an attitude determination algorithm typically calculates only an estimate, ${}^{E}_{I}\hat{R}$, of the true transformation encoded by the DCM, ${}^{B}_{I}R_{true}$. This estimate lies in the estimator frame, {E}. To bring the estimator frame into alignment with the body frame, i.e. {E} $\rightarrow$ {B}, a rotational correction term is needed. This correction term can be created through another solution to Wahba's problem known as the **algebraic method** or TRIAD approach, presented by Wertz [73]. This technique involves measuring the orientation in the body frame of N = 2 non-parallel, unit reference vectors fixed in the inertial frame. Any two such vectors ${}^{i}\overrightarrow{u}, {}^{i}\overrightarrow{v} \in \mathbb{R}^3$ define an orthogonal, 3-dimensional coordinate system with basis vectors $\overrightarrow{a}$, $\overrightarrow{b}$, and $\overrightarrow{c}$, which can be constructed as follows:

1. Let $\overrightarrow{a} = \overrightarrow{{}^{i}u}$ represent the first orthogonal axis.

2. Construct the second orthogonal axis from the cross product of $\overrightarrow{{}^{i}u}$ and $\overrightarrow{{}^{i}v}$: let $\overrightarrow{b} = \frac{\overrightarrow{u} \times \overrightarrow{v}}{\| \overrightarrow{u} \times \overrightarrow{v} \|}$. By construction, the axis $\overrightarrow{b}$ is orthogonal to $\overrightarrow{a}$.

3. Construct the third orthogonal axis from the cross product of $\overrightarrow{a}$ and $\overrightarrow{b}$: $\overrightarrow{c} = \overrightarrow{a} \times \overrightarrow{b}$. Use the right-handed rule to ascertain the appropriate direction of $\overrightarrow{c}$.

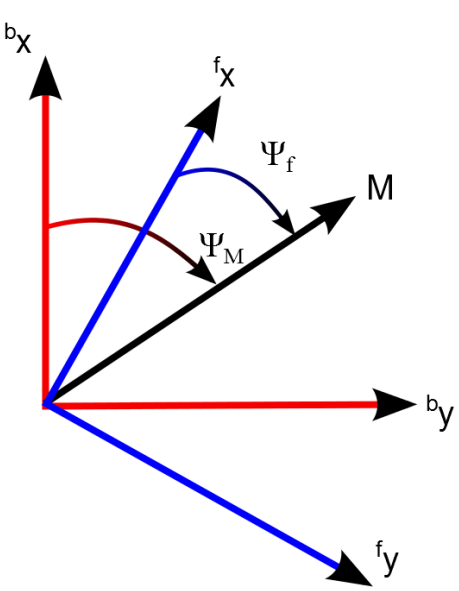
These unit reference vectors can refer to any vector fixed in the inertial frame. Examples include Earth's gravity vector, the geomagnetic field vector, the sun vector, etc. Vector direction, not magnitude, is key for attitude estimation.

The next section describes how reference vectors ${}^{i}u$ and ${}^{i}v$ serve as correction terms for rotating the estimator frame into the body frame, forcing ${}^{E}_{I}\hat{R} \rightarrow {}^{B}_{I}\ R_{true}$.

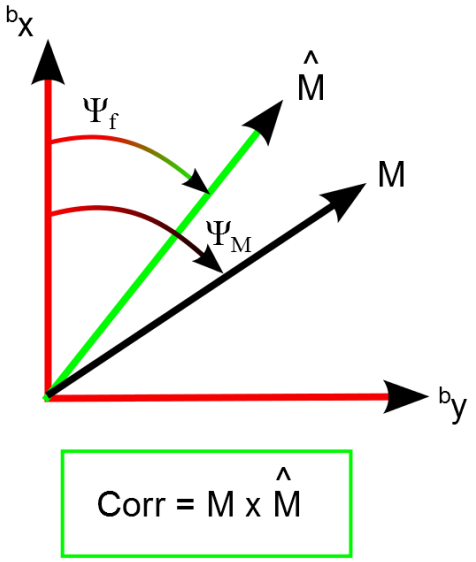
2.5.2 Angular Error Term

Begin with ${}^{b}v_1$, a measurement in body coordinates of a known, inertial reference vector, ${}^{i}v_1$. Calculate a separate estimate of this inertial vector using the DCM estimate $\hat{R}$: ${}^{e}v_1 = \hat{R}{}^{i}v_1$. Note that ${}^{b}v \neq {}^{e}v$ if $\hat{R} \neq R_{true}$. Calculate $R_{err1} = {}^{b}v_1 \times {}^{e}v_1$ if the estimate of the inertial reference vector can move with respect to the body frame, as with gravity or magnetic field reference vectors [21]. Otherwise, calculate $R_{err1} = {}^{e}v_1 \times {}^{b}v_1$ for when the estimate of the inertial reference vector is fixed with respect to the body frame, as with the x-axis and ground track [21]. These steps ensure the correct sign of the cross product and thus the correct direction of the correction.

The following discussion will elaborate on the above procedure for determining the order of the cross products for the correction terms, drawing heavily upon [21]. First, consider the case in which the estimate ($\hat{M}$) of the inertial reference vector **can move** with respect to body frame, such as with gravity or magnetic field reference vectors. Fig. 9 depicts this situation. For initial conditions, assume the ideal condition exists, i.e. that the filter axes (${}^{f}x, {}^{f}y$) are aligned with the body axes



(a) Here the filter frame has drifted out of alignment with the body frame. The filter frame "thinks" that the measurement vector M is only Ψ_f away from the body axes, when it is in fact Ψ_m.



(b) Another body-axis view. The filter frame's offset has caused the estimate $\hat{M}$ to lie closer to the body axes than the measurement, M. Thus the correction term of "measured cross estimate" is required for realignment.

Figure 9: Case in which the estimate frame **can move** with respect to body frame, as with magnetic field vector or gravity vector

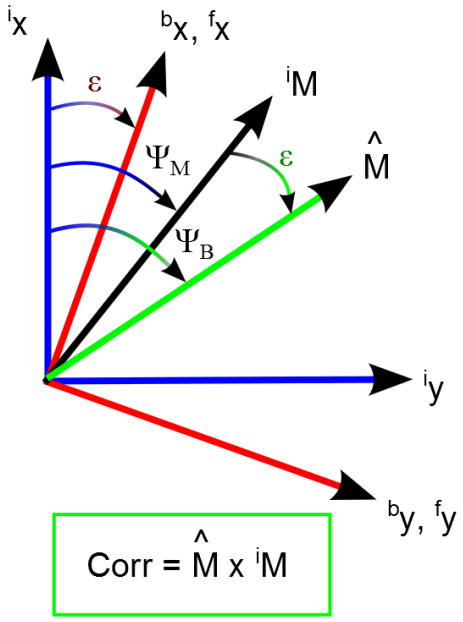


Figure 10: Estimate frame **is fixed** respect to body frame, as with aligning the body x-axis of a UAV with ground track. Here an offset of the body/filter axes towards the measurement by an angle ϵ results in the estimate being farther away from the inertial axes. Thus the correction of "estimate cross measured" is required.

$({}^b x, {}^b y)$. Here the angle to the measured reference and the angle to the estimate is the same, i.e. $\Psi_M = \Psi_F$. Next, assume the filter axes drift away from the body axes toward the measurement direction. Because of this misalignment, the filter reduces the angular distance to the measurement from Ψ_M to Ψ_F. However, the actual angle from the body axes to the measurement Ψ_M has not changed. Thus, to realign the filter axes with the body axes, the correction of "measured cross estimate" ($R_{err1} = {}^i M \times \hat{M}$ or ${}^b v_1 \times {}^e v_1$) will improve the situation.

Fig. 10 shows the situation in which the estimate of the inertial reference vector is **fixed** with respect to the body frame, in the case of the matching of the body x-axis of the UAV and ground track. By definition, the body and filter axes $({}^b x, {}^f x; {}^b y, {}^f y)$ are always aligned. The only misalignment is between the body/filter axes and the inertial reference. For the initial conditions, assume that the body/filter axes are aligned with the inertial axes $({}^i x, {}^i y)$. The angle from the inertial axes to the inertial reference vector ${}^i M$ is Ψ_M. Next, assume that through misalignment errors the body/filter axes drift towards the measurement by the angle ϵ, as shown in Fig. 10. Now the angle from the inertial axes to the estimate $\hat{M}$ has become $\Psi_B = \Psi_M + \epsilon$, but the angle from the inertial axes to the measurement ${}^i M$ remains Ψ_M. Thus, to realign the body/filter measurement estimate $\hat{M}$ with the inertial measurement ${}^i M$, use the correction of "estimate cross measured" ($R_{err1} = \hat{M} \times {}^i M \quad \text{or} \quad {}^e v_1 \times {}^b v_1$).

Repeat the above process with second non-collinear reference vector ${}^i v_2$ to obtain R_{err2}. Generate $R_{err3} = R_{err1} * R_{err2}$, a single, 3D correction term which, when used in feedback, ultimately rotates the estimator frame into alignment with the body frame, forcing ${}^E_I \hat{R} \rightarrow {}^B_I R_{true}$.

The above construction is facilitated by the following:

- The two inertial reference vectors above create a dual representation of attitude according to the algebraic method described above.
- The cross products of the estimates with the measurements of these two inertial

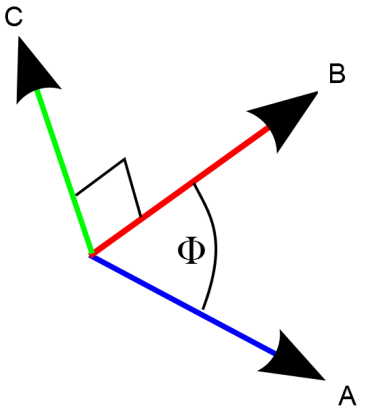


Figure 11: Error Φ between two unit vectors $\vec{A}$ and $\vec{B}$ is expressed by the magnitude of their cross product, $\vec{C}$.

reference vectors create rotational correction terms expressing a 3D rotational offset. Consider that if u_1 is parallel to u_2, then $\|u_1 \times u_2\| = u_1 u_2 \sin(\Phi) = 0$, and that $\|u_1 \times u_2\| \neq 0 \rightarrow \Phi \neq 0$, where Φ represents the angle between u_1 and u_2. Fig. 11 depicts the resultant cross product vector from the combination of two vectors. Though the cross product for two unit vectors yields $\sin(\Phi)$, the $\sin(\Phi)$ can be approximated by the angle Φ itself for small Φ. Thus the cross product can be used to represent the angular offset between the two vectors: an error term.

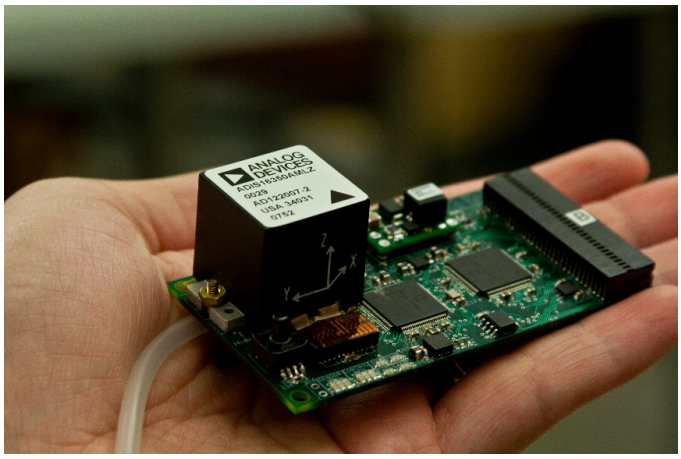


Figure 12: The core hardware of the SLUGS UAV autopilot.

3 Attitude Systems

This section focuses on a high level review of attitude systems, i.e. the electronics and hardware involved. These systems have specific capabilities as well as limitations, an understanding of which can assist the researcher in creating effective attitude estimation algorithms.

3.1 AutoPilot Systems

An autopilot system for UAVs estimates attitude and position from sensor measurements and then commands the actuators (control surfaces and throttle) needed to fly the vehicle. It is the core avionics component in UAS.

3.1.1 SLUGS AutoPilot

The **S**anta Cruz **L**ow-cost **U**AV **G**uidance, Navigation, and Control **S**ystem (SLUGS) is a versatile, flexible, lightweight, and low-power autopilot system developed in the Autonomous Systems Laboratory at the University of California, Santa Cruz. SLUGS was designed primarily for research and experimentation with small unmanned systems, especially fixed-wing UAVs. The AP has already been deployed on several aerial vehicles as well as a ground vehicle (the CruzControl project [11]).

One of the key advantages of the SLUGS AP over other commercially available au-

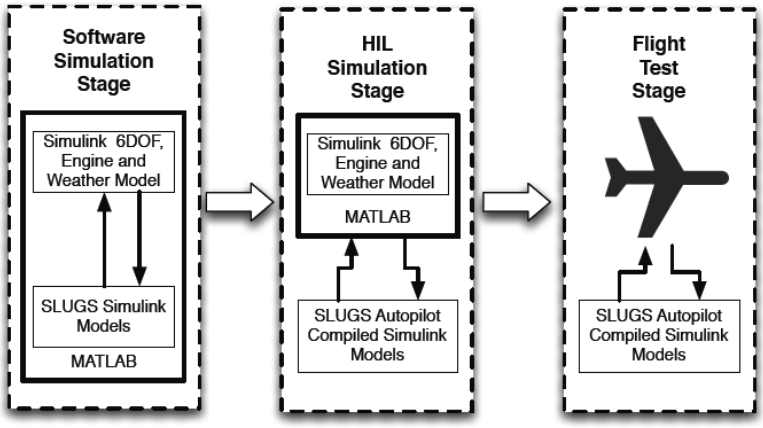


Figure 13: The software workflow of the SLUGS UAV autopilot. Images content of [42].

topilots is the open source nature of its software architecture, allowing the internal GNC routines to be readily modified, tested, and flown [42]. Widely used autopilots such as Piccolo, Kestrel, etc. are capable devices but typically offer only a fixed set of commands for control of the UAV. This needlessly complicates research and dynamic missions, which inherently require flexibility and adaptability to meet objectives, especially when designing and testing new algorithms. The software system of SLUGS is tightly integrated with the Matlab/Simulink environments, using automatic code generation tools to compile sophisticated models into the C code which will run on the target hardware. Because of this design, small to large subsystem modules can be quickly and accurately refactored without the need to directly write source code. This decreases the time spent debugging and increases productivity and research efforts. Furthermore, the transition from pure simulations to hardware-in-the-loop simulations to flight tests is simple and straightforward because the underlying GNC models are the same [41]. Fig. 13 depicts this pipeline, which relies extensively on Matlab/Simulink.

The small size of the stand-alone autopilot hardware, shown in Fig. 12, eases integration into numerous small scale vehicles. Furthermore, the system architecture of SLUGS was designed so that small UAVs would have enough processing

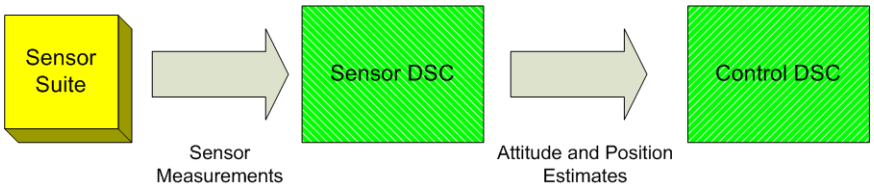


Figure 14: Flow of sensor information in the SLUGS AP

power for fairly complicated control tasks while still allowing the system to be easily and rapidly reprogrammable via Simulink [42]. The embedded system features two dsPIC33F microcontrollers from Microchip:

1. Sensor DSC (Digital Signal Controller) - for sensor measurements and attitude and position estimates, information which is subsequently transmitted to the

2. Control DSC - for guidance and control tasks which create the desired actuator signals to fly the plane.

The actions of the sensor DSC are of most relevance to this thesis. This processing unit samples all the onboard sensors, computes position and attitude estimates at 100 Hz using a complementary filter, and then transmits this data every time step to the control DSC. This information pipeline is depicted in Fig. 14. The attitude filter will be discussed in-depth in the subsequent chapter on previous work, Sec. 4. Further details on the architecture and other system and software aspects of the SLUGS AP may be found in [42],[40], and [41].

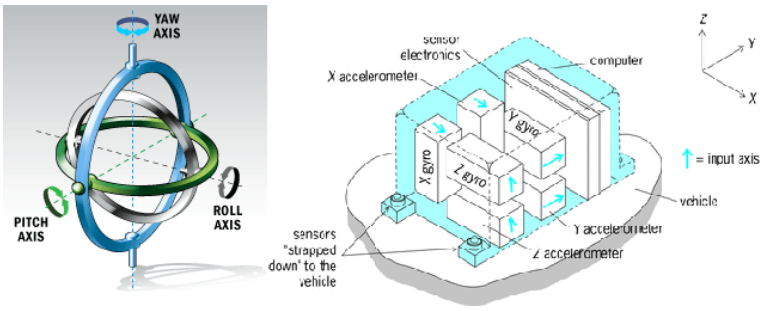


Figure 15: Gimballed sensors (left) can freely rotate independent of the host platform, while strapdown sensors, such as the 3-axis IMU (right), maintain the body-frame orientation of the vehicle. Images content of [66], [39]

.

3.2 Sensing and Measurement Devices

In the first stage of the attitude estimation pipeline depicted in Fig. 14, the autopilot samples the UAV's sensors. What makes attitude estimation more challenging is that no sensing device currently exists which directly measures angular orientation or the attitude of the vehicle [24]. Thus engineers and researchers must express some ingenuity through their selection of sensors, whose data will be combined to give an accurate, timely estimate of the vehicle's attitude. A common approach for sensor suites is to combine angular velocity measurements, provided by multi-axis gyroscopes, with one or more reference vectors in the inertial frame. Frequently employed reference vectors include gravity estimates from accelerometers [44] [58](caveats to be discussed later), heading from magnetometers [57], GPS course-over-ground or inertial velocity [32], visual signatures [69], or thermal indicators [8]. The aforementioned reference vectors typically stem from strapdown, as opposed to gimballed, devices. Strapdown units maintain the same orientation as the body frame of the flight vehicle, while gimballed devices, free to rotate in space, maintain their orientation with respect to the inertial frame, independent of the body movements of the aircraft. Fig. 15 depicts this difference. Gimballed inertial measurement units (**IMUs**) are quite rare in low-cost UAV systems due to their excessive weight. Improvements in strapdown MEMS technologies, computerized sensing devices, and algorithms have made gimballed systems quite rare.

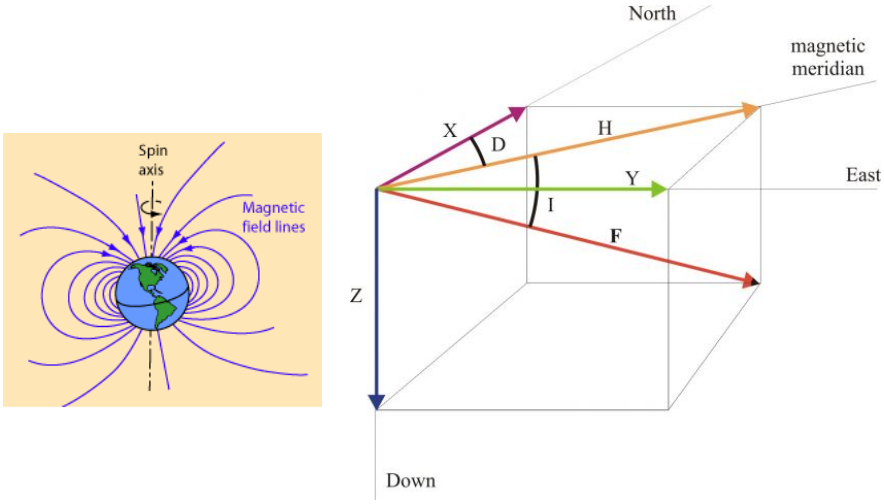
The following subsections give a brief overview of some salient characteristics of various sensors commonly used in attitude solutions. Specifically, GPS and INS (Inertial Navigation System) sensor suites will be discussed, as the SLUGS autopilot polls these types of devices in the first stage of attitude estimation. As this thesis focuses on low-cost UAVs, low-cost sensing devices dominate the discussion.

3.2.1 Rate gyros

A sensor frequently employed in INS or IMU systems is the rate gyro. Rate gyros provide key information for attitude estimation: angular velocities in the body frame of the aircraft. Typically three-axis (xyz) devices, rate gyros or gyroscopes provide high bandwidth information (typically > 100Hz). In general, lower cost gyros output angular rate while higher cost gyros output integrated angular rate [32]. All gyros suffer to a greater or lesser extent from a number of errors, which may be represented by the following model:

$$\Omega^y = \Omega + b + \mu \in \mathbb{R}^3, \tag{50}$$

where the measured angular velocity Ω^y is composed of Ω, the true angular velocity, plus $\mu \in \mathbb{R}^3$, additive measurement noise, and $b = b_0 + b(t, T, c)$, where b_0 represents a DC null-shift bias and $b(t, T, c)$ represents a varying bias which is a function of time (t), temperature (T), and factors relating to quality of construction (c)(and thus cost) [26][44]. Temperature calibration can help compensate for the temperature dependence of the output. A gyro's DC bias can be estimated by averaging the output of the device when it is known to be stationary. Performing such a calibration over a large time span improves the quality of the estimate of low-frequency noise contributed by the bias. The time-varying bias can be estimated and eliminated at runtime through appropriate filtering algorithms, such as PI feedback, Kalman filtering, etc. High frequency noise is another issue with rate gyros and often appears in the form of random walk or zero mean white noise due to thermo-mechanical events [2]. Lastly, gyro saturation can occur on UAVs during high dynamic maneuvers in which the movements of the vehicle occur at a higher rate than the dynamic range



(a) Magnetic field of Earth. Note that the lines traverse from the South to the North poles [31].

(b) Inclination angle, I, and declination angle, D, of the Earth's magnetic field vector, F [22]. The values of these angles vary according to the location on Earth.

Figure 16: Geomagnetic field.

of the sensor. This upper-threshold of the device's dynamic range causes the device to report a constant (max) valued measurement rather than the true angular rate - a nonlinear response. This situation could occur if an an acrobatic aircraft were outfitted with an inappropriate rate gyro.

3.2.2 Magnetometer

Magnetometers, another sensor commonly found in IMUs, measure the direction of the Earth's magnetic field, whose field lines are shown in Fig. 16a. The geomagnetic field protects the Earth's atmosphere and thus life from the harmful effects due to ionized particles emitted by the sun, the solar wind. The magnetic field vector is an inertial reference vector often used in attitude estimation. Some simplistic approaches use only 2 of the 3 available dimensions of this vector to estimate vehicle heading. The magnetic field vector can be graphically represented by a 3D vector pointed into the Earth. The **declination** angle describes the offset of the field vector from geographic or true north, while the **inclination** angle pertains to the vector offset from the horizontal plane parallel to the Earth's surface, as shown in Fig. 16b.

The following magnetic field model bears similarities to Eq. 50's gyroscope model:

$$h^y =_I^B R *^i h + B_h + \mu_h \tag{51}$$

where the measured magnetic field vector h^y is composed of the inertial frame magnetic field vector ${}^i h$, plus body-frame, local magnetic disturbance B_h, plus sensor noise μ_h [44]. Though sensor noise associated with this vector is frequently low, nearby magnetic disturbances can be quite high, especially near the electric motors typically employed on UAVs. Numerous other disturbances associated with nearby metallic sources contribute to the degradation of accuracy in a magnetometer's readings; an in-depth discussion of these effects falls outside the scope of this work, but the reader is referred to [68] for additional information.

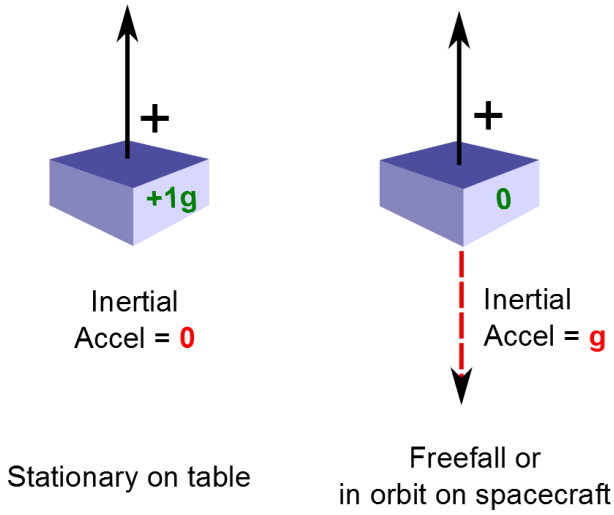


Figure 17: Two key scenarios illustrating accelerometer operation: $a_{meas} = a_i - g$. The measurements reported by a typical device are shown in green on the purple box representing the device. The black arrow is the positive input axis, while the dashed red arrow is inertial acceleration.

3.2.3 Accelerometer

Accelerometers are the third key sensing component of an IMU. The theory behind and notion of what these multi-axis devices actually measure can be confusing. Consider two sample operational scenarios, as depicted in Fig. 17:

1. Accelerometer resting on a table with the positive input axis "up". Reading: $+1g$.

2. Accelerometer in freefall or on a spacecraft in orbit with the positive input axis "up". Reading: $0g$.

The equation $a_{meas} = a_i - g$ describes a view (simplified to ignore noise) of how the above two readings arise. In scenario 1, the inertial acceleration is 0, i.e. $a_i = 0$, and the acceleration due to gravity is $-g$, since it is in the opposite direction as the positive input axis. Thus $a_{meas} = a_i - g = 0 - (-g) = +1g$, as depicted on the left in Fig. 17.

For scenario 2, the inertial acceleration is $-g$, while gravity is, once again, $-g$ according to the direction of the positive input axis; thus $a_{meas} = a_i - g = -g - (-g) = 0g$,

as depicted on the right in Fig. 17.

The above two scenarios illustrate that accelerometers measure the component of "inertial acceleration minus gravity" along each input axis. This is also known as **specific force**, defined as "the non-gravitational force per unit mass" [76].
The way an accelerometer measures specific force is through application of Newton's Second Law, the equation $\vec{F} = m\vec{a}$. Inside the accelerometer is a small proof mass which is held in place by a special force transducer to maintain the position of the mass within the housing. The larger the acceleration undergone by the unit, the larger the stabilizing force required to maintain the position of the proof mass - a force proportional to the acceleration, since the mass is constant. Since both the accelerometer housing and its internal proof mass undergo the same, continuous gravitational acceleration, the device cannot sense gravity [21]. The device then generates a normalized signal associated with the magnitude of the stabilizing force, resulting in the device output of specific force [62].

Accelerometers are high bandwidth devices with high transient response, low drift, and (depending on quality) high noise relative to the rate gyros in flight. This noise primarily stems from the vibrations associated with the electric motors onboard the airframe. Because of this noise, direct integration of the output signal to provide (inertial) velocity estimates leads to rapidly divergent results. Recommended usage of these devices is to attenuate the noise through an appropriate filtering algorithm before using the measurements for estimation or control.

The following mathematical model applies to a typical, low-cost accelerometer [44]:

$$a^y =^i a -^i g + b_a + \mu_a \tag{52}$$

where measured acceleration a^y comes from the sum of the inertial acceleration ${}^i a$, minus the acceleration due to gravity ${}^i g$, plus the bias b_a, and the additive measurement noise μ_a [44]. When coupled with devices producing inertial velocity

measurements, such as a GPS receiver, accelerometers can help provide an estimate of the gravity reference vector.

Figure 18: Space Segment of the Global Positioning System [5].

3.2.4 GPS

A global positioning system (**GPS**) or Global Navigation Satellite System (**GNSS**) receiver is an important, widely used sensor for GNC systems. Without the inertial frame measurements provided by GPS or some other external (i.e. inertial frame) aid, position and attitude estimates of most systems will drift or diverge to the point of uselessness. Due to their complementary characteristics, integrating a GPS receiver with an INS device can yield higher accuracy, higher bandwidth, greater robustness and flexibility in a navigation solution than is typically possible with either device alone [32].

GPS is a multi-lateration, or rho-rho ($\rho - \rho$), positioning system. The space segment of GPS consists of a constellation of 24 or more satellites in orbit around the earth in six orbital planes, as shown in Fig. 18. The user segment is the navigational receiver, a digital signal tracking device which for the purposes of this thesis is a strapdown sensor mounted to the UAV's airframe[52]. Determination of the user's position is performed as follows. The satellites simultaneously broadcast a radio signal encoding the ephemeris data for each satellite, information which allows the

receiver to calculate that particular satellite's position and its timing information (all satellites carry synchronized, atomic clocks). A signal from only one such satellite is insufficient to uniquely determine the user's position on Earth, as the region of reception describes a sphere. However, reception of signals from at least three different satellites allows an intersection point of three spheres to be calculated and a unique position on Earth can then be identified [52]. This position is described by the user's latitude, longitude, and altitude. With movement, the velocity (speed-over-ground or SOG) of movement as well as the direction of movement (course-over-ground or COG) can also be estimated. Device resolution for unaided position estimates is on the order of a few tens of meters in accuracy.

GPS provides passive, low-bias navigational information continuously, worldwide, anywhere within view of the open sky. Mobile and stationary platforms can benefit from the information delivered by a GPS receiver. The drawbacks of the receiver include low bandwidth (position updates $>$ 5-10Hz provide little more than added noise), poor estimates of altitude, and inoperability indoors. The two chief sources of ranging errors associated with the device are radio signal interference within the Earth's ionosphere and multipath error, resulting from the reception of delayed reflections of the GPS signal [52].

GNC systems, automobiles, aviation (especially NextGen), and even wristwatches all use GPS receivers. In the SLUGS autopilot, this sensor provides COG and inertial velocity measurements which improve attitude estimates. GPS has been quite important for unmanned vehicle navigation and will become even more important in 2015 with the FAA's adoption of NextGen, as described in Sec. 1. Refs. [32] and [52] offer extensive coverage of GNSS theory and applications.

4 Previous Work

This section focuses on prior attitude estimation and propagation techniques, primarily in the application domain of low-cost, fixed wing UAVs.

4.1 Attitude Parameterizations

4.1.1 Theory

Relatively few researchers have directly examined the differences in attitude estimation results from different parameterization methods. In Ref. [43], Lovren presents an error analysis of the DCM vs. quaternion methods within the context of classical coning motion. Coning motion, or gyroscopic precession, is akin to the motion of a spinning toy top in which the axis wobbles. To facilitate their analysis, the researchers introduce an orthogonal transformation matrix, $A \in \mathbb{R}^{4\times4}$, whose elements are either direction cosines or quaternion components. Within this construct, an error model was constructed to encode the distortion of the estimated transformation matrix $\hat{A}$ away from the true transformation matrix, A, as $\hat{A} = A + \triangle A = A(I + \Psi)$, where $\Psi = A^T \triangle A$ represents the total error in the transformation estimate. The matrix Ψ can be factored into symmetric and antisymmetric components, Ψ_s and Ψ_u, respectively. The diagonal elements of Ψ_s are the **scale errors**, the off-diagonal elements of Ψ_s are the **skew errors**, and the elements of Ψ_u are the **drift errors**. The researchers then show that quaternions exhibit zero skew errors, while scale and skew errors are "negligibly small" with DCM parameterization, which means they are practically zero.

The work by Phillips [53] parallels that of this thesis, providing excellent coverage of the direction cosine matrix, Euler angles, Euler-Rodrigues quaternions, and the Euler-axis (Eigen-axis or angle-axis) formulations. Though kinematic expressions are presented for all four of the above parameterizations, the paper focuses on quaternions, specifically the propagation of attitude kinematics via high-order, multi-step integration methods, techniques for orthogonality control, and various

computational efficiencies which may be gained via quaternion algebra.

In the early days of aircraft simulation, analog computers ran the first quaternion implementations. Due to the first-order nature of such programs, large orthogonality errors would accumulate and destroy the utility of the simulation after some time. Thus methods of orthogonality control became an important, emerging topic of interest. Phillips' paper examines the Corbett-Wright method in detail using a 4th order Runge-Kutta numeric approximation for propagation. First-order Euler, 2nd-order Runge-Kutta, and 4th-order Adams-Bashforth-Moulton integration methods were also evaluated in terms of total error as well as computational efficiency in the form of cycles per simulated second. As expected, the higher the order of the integration method, the greater the computational load required.

The research presented in [53] differs from this thesis in the following key ways:

- Likely since the domain of application was aircraft modeling and simulation rather than deployment on low-cost UAVs, little to no coverage of algorithm noise response was presented.
- Though multiple attitude representations were presented in the beginning, the bulk of the paper focused on quaternions for integration comparisons. This thesis will present methods for all of the above parameterizations.
- Ref. [53] experimented primarily with higher-order, multi-step integration methods, whereas this work will concentrate on lower-order, single-step methods. This difference in approaches likely arises from the difference in computational resources available for the target application domain: a desktop workstation vs. a low-cost, embedded flight computer (typically a microcontroller or small form-factor PC).
- No experimental justification was given for the statement that numeric integration of DCM kinematics is "excessively time consuming, and a severe computational penalty is paid for its use." In the next chapter, this thesis will

provide such data and illustrate just how severe that penalty is. A note is made that for higher order, multi-step methods such as the 4th-order Runge-Kutta routine, the previous statement regarding the DCM seems quite reasonable. For lower-order, single-step methods, this is not necessarily true.

4.1.2 Applications

Regarding applications of various attitude parameterizations, the MatrixPilot software for the UAV Dev Board system [55] incorporates the direction cosine matrix representation of attitude for the core onboard attitude and transformation calculations. The key motivation for using the DCM was to upgrade the UAV flight control capabilities from governing stable flight to controlling aerobatic flight, as the DCM was "a natural fit to control and navigation." [59] The designers used the elements of the DCM directly in control and navigation. For example, the dot product of the (x) roll axis with the (z) ground axis is one of the direction cosine elements, r_{31}. This element indicates whether or not the roll axis is parallel to the ground and can thus be used in a feedback loop to control pitch [59]. The DCM elements were also used for navigation and within a PI feedback controller for drift and yaw corrections. The drawbacks of the DCM were a) the extra memory to maintain all nine elements onboard and b) numerical errors which required additional effort to reorthonormalize the matrix at each time step. No indication was given as to the magnitude of these errors using the 50 Hz integration steps.

Gebre-Egziabher [30] touches upon the differences between Euler angle and quaternion-based estimators. In this work, it was demonstrated in detail that scheduling the Kalman filter gains for the quaternion estimator was simpler and easier than the corresponding filter in the Euler angle parameterization. However, the process noise matrix used in the Euler angle mechanization exactly matched that for the quaternion implementation. Correspondingly, it was believed that the accuracy of the attitude determination system was ultimately a function of the aiding system's accuracy (eg. complementary filter with reference vectors) and that the parameteriza-

tion choice was inconsequential to system performance. No metrics were presented or comparisons of these two parameterizations were made within the context of the open-loop (or closed loop) integration of the kinematic equations [30].

In Ref. [30], Gebre-Egziabher maintains that the key advantage to the Euler angle parameterization is their intuitive nature, easily mapping to the pilot's controls and heads-up interface. It was suggested that this parameterization be used when generating attitude information solely for pilot-in-the-loop control of the aircraft, implying that unmanned aircraft would benefit less from this representation. Issues with this parameterization include singularities at a 90 degree pitch angle and kinematic equations with nonlinear, transcendental functions, which require extra processing power.

The quaternion parameterization is possibly the most widely used attitude parameterization since the 1980s, according to the attitude survey by Crassidis [19]. The popularity of quaternions is said to be due to the linear kinematic equation and the compact representation compared to the DCM. The SLUGS AP uses the quaternion parameterization in its discrete-integrator for attitude propagation. Similarly, the VTOL MAV described in Ref. [44] uses this 4-vector representation in various complementary filter approaches. Efficiency of code implementation was presented as a chief benefit of quaternions.

The SLUGS AP also employs the DCM for reference vector frame conversions and Euler angles for flight controls, effectively fitting the representation to the task. The chief drawback to this approach is the computational overhead associated with the frequent conversions between representations (quaternions to DCM to Euler angles).

Lastly, the axis-angle representation is not heavily used beyond an intermediate step in attitude estimation [63]. Though spacecraft frequently perform eigen-axis maneuvers, the quaternion parameterization appears to be used more commonly than the eigen-axis scheme. In Ref. [13], the researchers use the angle-axis represen-

tation in a small helicopter. The flight dynamics of hovering might lend advantages to this representation. Otherwise, little analysis of benefits vs. drawbacks of this parameterization was presented.

4.2 Attitude Propagation Methods

Many low-cost UAV attitude systems implement the simple approach of the first order, forward Euler approximation to propagate attitude. The MatrixPilot system [55], the SLUGS AP, as well as Gleason's GNSS work [32] rely on this method. The key assumption behind several of these implementations is that the body-frame angular velocities are constant between samples. If the sampling rate is sufficiently high compared to the observed flight regime, then this assumption may be reasonable. Nevertheless, the slow rate of convergence associated with first order forward Euler could limit its effectiveness for low-bandwidth sampling when more aerobatic maneuvers are performed. To justify the use of this method, researchers frequently state that the convergence, accuracy, and error characteristics of this method are mitigated by incorporating feedback of an inertial reference vector like a GPS update to correct for the resulting drift. The latter ignores the common scenarios of unreliable GPS reception. Other justifications are the low computational complexity and cost of this algorithm. This thesis will later demonstrate that forward Euler integration indeed does have a lower computational load than other, more sophisticated integration schemes.

The next step in complexity beyond forward Euler is the 2nd order Runge-Kutta method, also known as the Euler-Cauchy approach. Though targeted at orbiting satellites rather than near-Earth UAVs, the 1997 patent by McEnnan [47] describes the advantages of attitude propagation methods using this multi-stage method. A paper on RC helicopters by Baldwin [6] mentions, but does not incorporate, the "midpoint" method of the 2nd order Runge-Kutta routine for attitude propagation. One stated, potential advantage is the attenuation of noise associated with the first order Euler method. No experimental validation of these statements were presented.

Higher order Runge-Kutta methods see more frequent use in satellites and other aircraft with greater computational resources than low-cost UAVs. However, as some spacecraft rely on ten-year-old technology, older satellite systems may operate on comparable (or fewer) CPU resources as many of today's lower-end UAS. At any rate, the Gyrostat satellite in [37] used a 4th order Runge-Kutta method for propagating the quaternion attitude estimates. Another work which used Runge-Kutta routines was that of Gavrilets, et al [29]. This MIT research group used an RK4 method for state propagation; the research platform was an RC helicopter with a 5 ft. rotor designed to perform autonomous aggressive maneuvers. The computational payload of this aircraft was significant - 7 lbs of avionics, more than a typical low-cost fixed-wing UAV would likely be able to support. These techniques are very accurate but require greater computational expense due to the multi-step approach.

The closed form solution of the matrix exponential has been infrequently used in the field of UAVs, though it was used as early as 1977 in spacecraft, specifically within NASA's High Energy Astronomy Observatories [28]. Due to the slow sampling period, 100-400ms of the spacecraft's gyros, the closed-form solution was found to be an excellent option for performing the onboard kinematic integration step. Later, in the 1990s, the field of robotic manipulation has employed the matrix exponential in the associated kinematic equations [49], [51]. The formulation appears to be only recently incorporated into more UAV applications. Baldwin, et al. present simulation and experimental test results on complementary filter design in [6], using the matrix exponential for the integration step within the application domain of RC helicopters. Benefits of this approach included some reduction in high frequency noise associated with the first order Euler integration. Lastly, the work by Trawny [67] offers multiple variations of the matrix exponential in the quaternion formulation based on constant and linear assumptions on angular velocity between time steps. A tutorial document for quaternions, Trawny presented no experimental evaluation of the aforementioned algorithms.

4.3 CASE STUDY: Attitude Estimation in the SLUGS Autopilot

An overview of the sensor DSC and the SLUGS AP was presented in the previous chapter on Attitude Systems. This section presents the existing SLUGS attitude and heading reference system as implemented in Matlab/Simulink. Subsequent experiments for this thesis will use the results of this existing algorithm as a benchmark against later improvements to the attitude filter.

4.3.1 Motivation

Curry [20] demonstrate that the complementary filter (CF) used in the SLUGS AP offers comparable performance to an EKF (**Extended Kalman Filter**)-based approach. The CF demands significantly less system resources than an EKF and thus lends itself well to embedded hardware implementations. Indeed, when the position filter implementation on SLUGS was switched from an EKF to a CF, the CPU load on the sensor DSC dropped from nearly 100% to only 33%.

The CF can be modeled as a steady-state Kalman Filter for a certain class of filtering problems. Also, the complementary filter does not consider a statistical description for the noise corrupting the signals, thus there is no unwieldy covariance matrix to propagate between samples. Lastly, the CF works primarily in the frequency domain, whereas the KF is typically a time-domain approach [35].

Fig. 19 shows the Simulink model of this filter, which is the high-level description of the attitude estimation algorithm that gets directly translated into the C code which runs on the autopilot hardware itself. Besides the Simulink model itself, the best references for the attitude estimation algorithm used in the SLUGS AP are [44], [27], and [20]. The SLUGS state estimation routines estimate not only attitude but also position, with the results of each estimator delivered to the other in a feedback

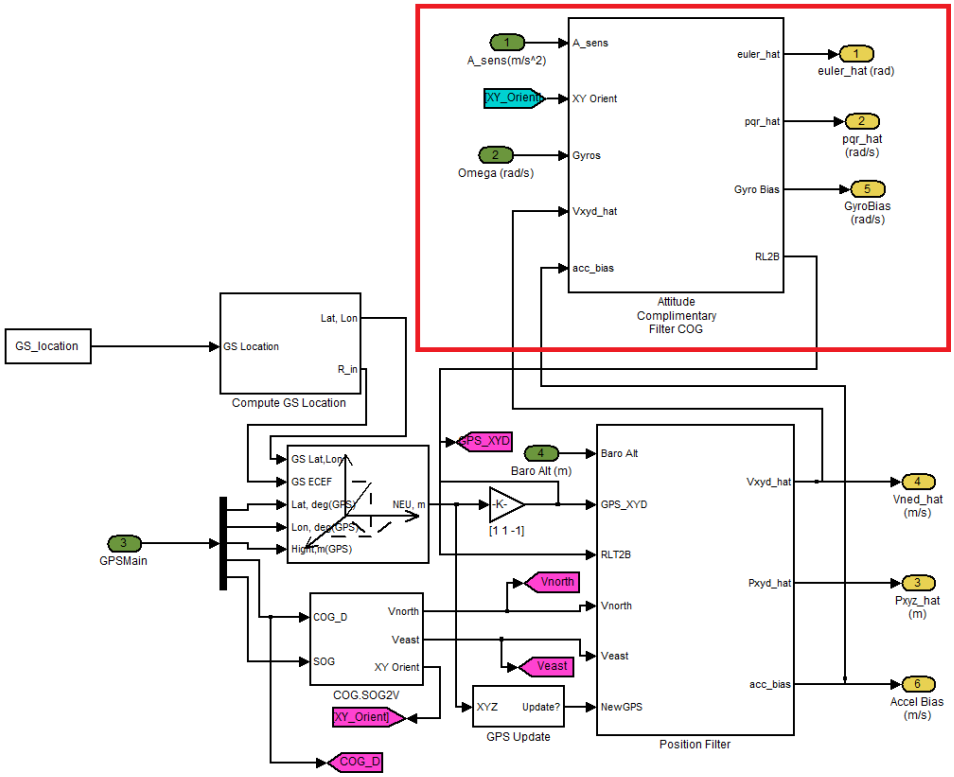


Figure 19: Attitude and Position Filter, Top Level

framework, as shown in Fig. 20. This feedback leads to integrated performance better than either filter's individual performance. An in-depth examination of the position filter lies outside the scope of this work.

Before specifics of the SLUGS complementary filter are discussed, the following complementary filter framework will describe the general features of such an approach. This analysis relies heavily upon the work by Mahony [44]. Consider a signal of interest measured by two different sensors, the difference being the frequency of the measurement noise (μ_1, μ_2) associated with each sensor - one high, the other low. The corresponding measurements y of the signal x may be defined as $y_1 = x + \mu_1$ and $y_2 = x + \mu_2$, where μ_1 is predominately high frequency noise and μ_2 is predominately low frequency noise. Then, one can construct a pair of complementary transfer functions, $F_1(s)$ low-pass and $F_2(s)$ high-pass, such that $F_1(s) + F_2(s) = 1$.

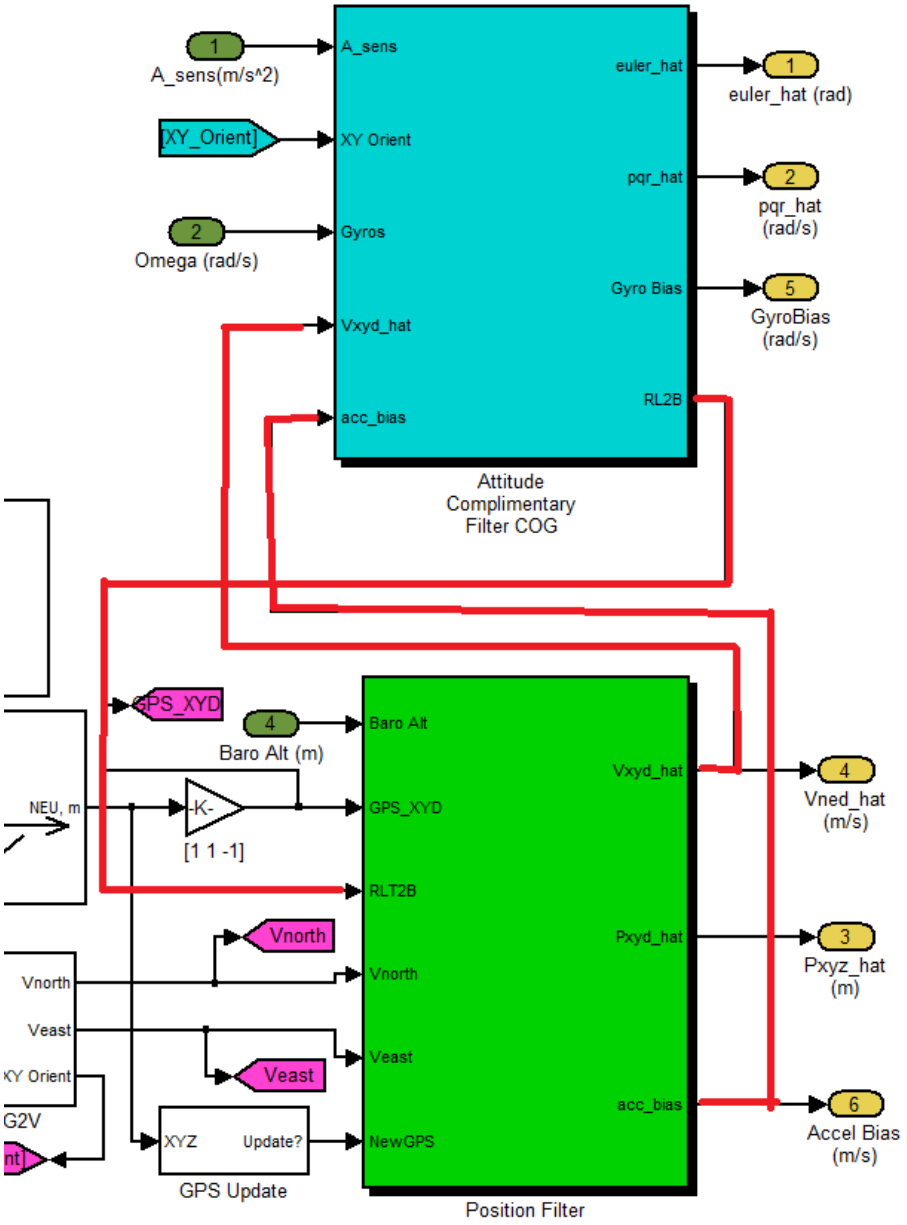


Figure 20: Feedback between the Attitude (top) and Position (bottom) Filters

Using this setup, the resulting filter estimate of our desired signal x is as follows:

$$\hat{X}(s) = F_1(s)Y_1 + F_2(s)Y_2 = X(s) + F_1(s)\mu_1(s) + F_2(s)\mu_2(s) \tag{53}$$

With Eq. 53, the estimate of signal x is all-pass while the noise associated with each sensor is attenuated by the relevant high- or low-pass filter, $F_i(s)$. A common setup with complementary filters is to combine low bandwidth position or orientation information (GPS) with high bandwidth, integrated angular rate information (rate gyros), resulting in the attenuation of noise arising from both information sources.

Inside the SLUGS attitude estimator lies a three-fold complementary filter framework in that there are high frequency data via the gyros, accelerometers, and lower frequency data from the GPS device being fused together in a manner described by Eq. 53.

The next level of detail into the estimation algorithm is the top level of the three-axis attitude filter, Fig. 21. The algorithm may be described as a complementary, discrete-time, feedback filter with forward Euler propagation using the quaternion parameterization. The rate gyros provide angular velocity measurements. To these, the algorithm adds corrections of the associated biases as well as feedback terms from the gravity reference vector ${}^{i}g$ and the track reference vector ${}^{i}\psi_{ref}$ to yield the estimated $\hat{\omega}$. Finally, the algorithm integrates the quaternion kinematic differential equations to produce the current DCM estimate, $RLT2B$, the rotation from local tangent plane to the body frame.

Further analysis of this algorithm will be structured according to the following subsections:

1. **Inputs, Outputs** - parameters and return values
2. **Reference vectors** - the two inertial vectors described in section 2.5
3. **PI controller** - how the correcting action of the reference vectors improves

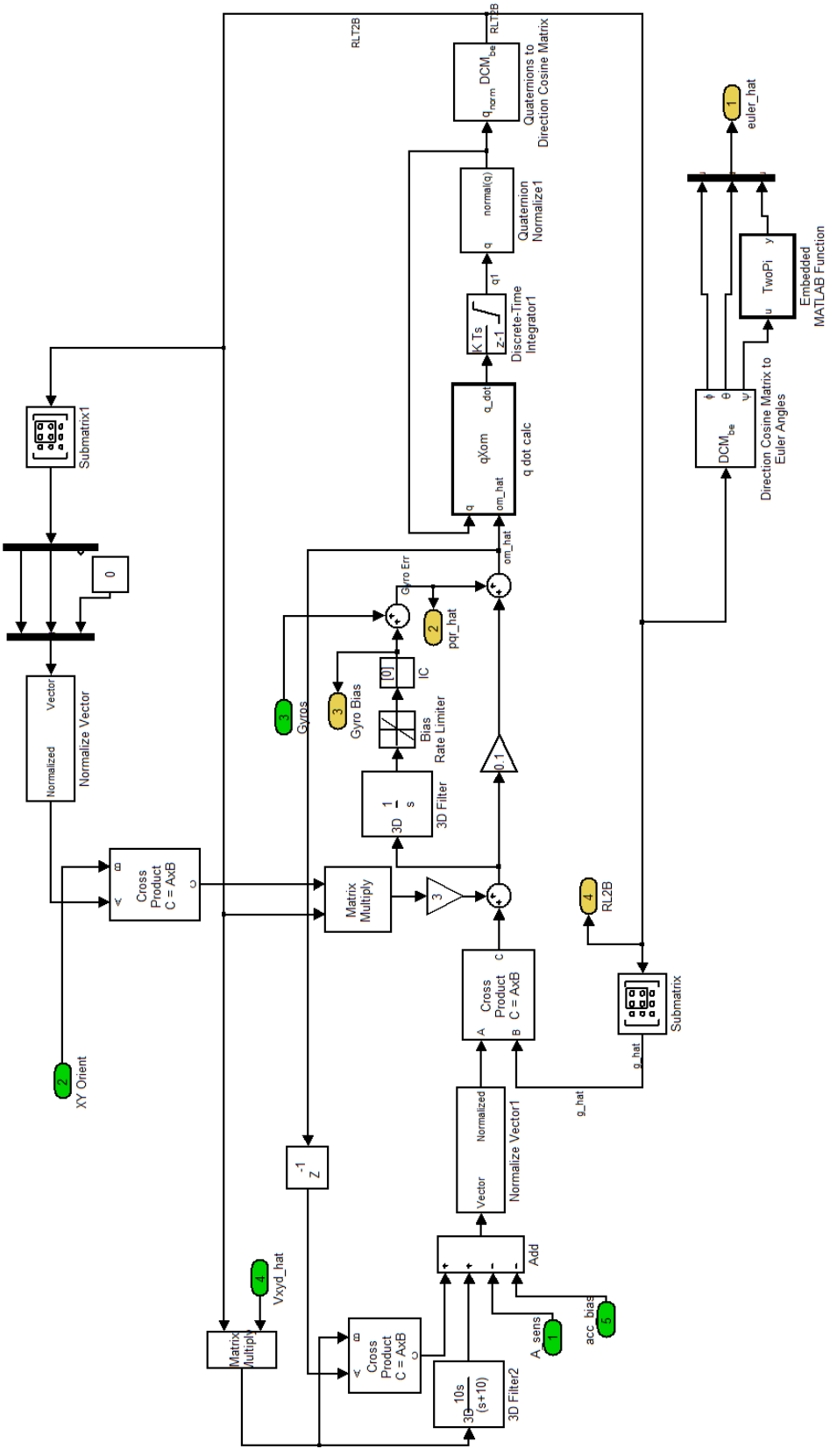


Figure 21: Baseline Attitude Filter

the angular velocity estimate, $\hat{\omega}$.

4. **Propagation** - of attitude kinematics expressed by $\dot{q}$.

5. **Post-Processing** - preparing data for the next stage

4.3.2 Inputs and Outputs

Important landmarks in the Simulink model are as follows:

- **Inputs** (green)
  - $\hat{V}_{xyd}$, estimated inertial velocity,
  - A_{sense}, measured acceleration,
  - acc_{bias} or A_{bias}, accelerometer bias,
  - XY_{orient} or ψ_{ref}, track reference vector,
  - $Gyros$, rate gyro measurements of the angular velocities in the body frame.

- **Outputs** (yellow)
  - $RLT2B$, the DCM from the local NED (Local Tangent Plane) frame to the estimator frame,
  - $GyroBiases$, calculated biases on the rate gyros,
  - $p\hat{q}r$, corrected angular rates expressed in the body frame,
  - $Euler_{hat}$, estimated ZYX Euler angles representing the current attitude of the aircraft.

4.3.3 Reference Vectors from the Inertial Frame

Gravity A commonly used inertial frame reference vector for near-Earth aircraft is that of gravity, the reference for the down direction. This unit vector is represented as the column vector ${}^{i}g = [0; 0; 1]^T$ in the inertial frame (NED) and acts as the leveling reference, which influences the xy plane of the NED frame. The third

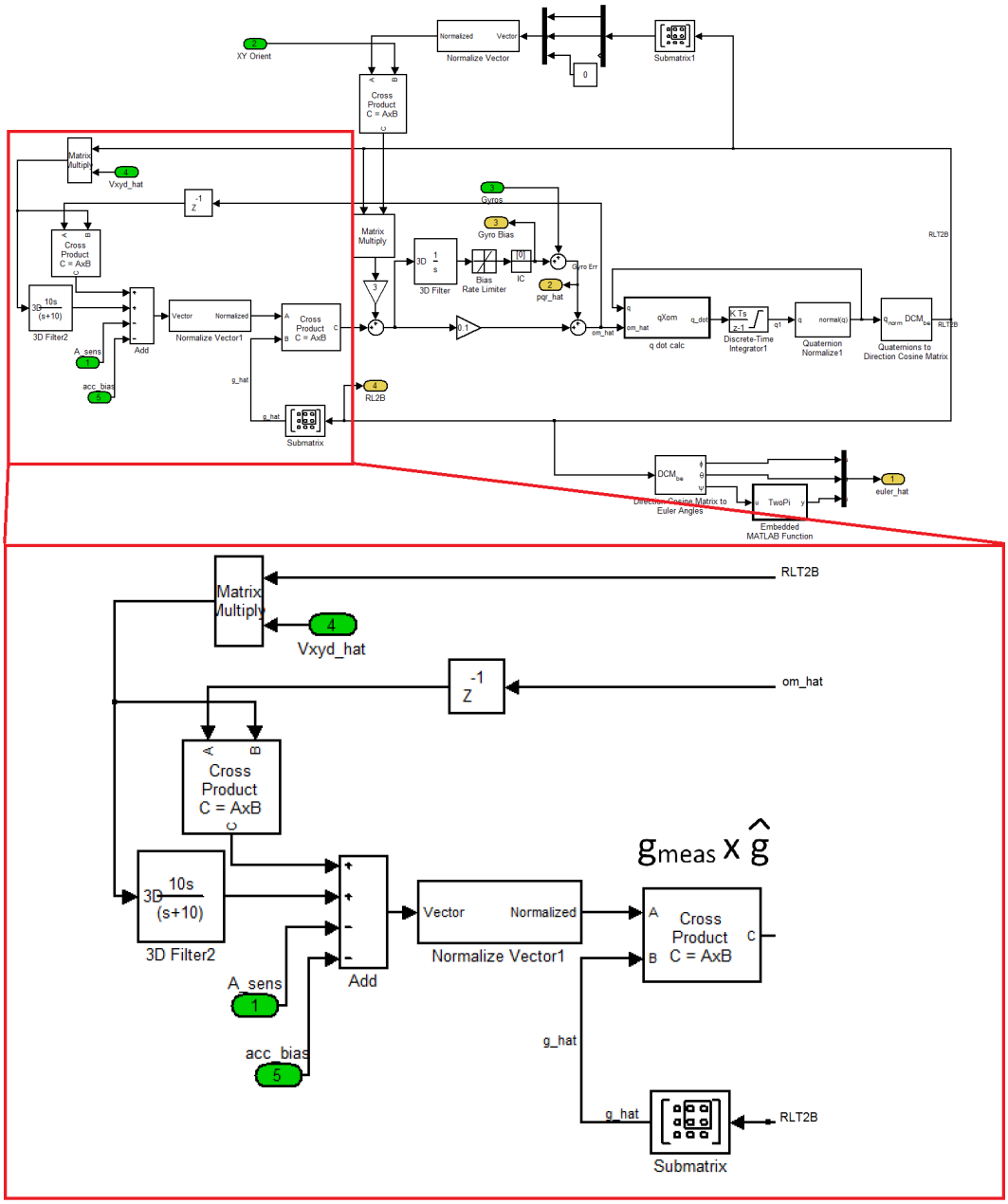


Figure 22: Gravity Correction, context and expanded

column of the DCM represents the inertial frame z-axis expressed in the body frame, ${}^{b}\hat{g}$, an estimate of the gravity vector rotated by the DCM into the frame of the body.

The body frame measurement of the gravity vector is next calculated. This construction is challenging using an accelerometer alone due to the nature of the device, as described in Sec. 3.2.3. By augmenting the accelerometer's body-frame measurements with inertial frame measurements of a different sensor, such as GPS, the algorithm can extract the individual components of acceleration due to gravity and those due to the inertial rate of change of velocity.

Ignoring the noise and bias components of Eq. 52 from Sec. 3.2.3, a simple model of the accelerometer expresses its measurements as the sum of all accelerations experienced by the aircraft except for gravity:

$$ {}^{i}a_m = {}^{i}a - {}^{i}g \tag{54} $$

where ${}^{i}a_m$ represents the accelerometer measurement, ${}^{i}a$ represents inertial acceleration, and ${}^{i}g$ represents the acceleration due to gravity. To calculate an estimate of the gravity vector, rework Eq. 54 by first isolating the gravity component:

$$ {}^{i}g = {}^{i}a - {}^{i}a_m \tag{55} $$

Next, normalize the result to obtain a unit vector, $\hat{u}_g$:

$$ \hat{u}_g = \frac{{}^{i}a - {}^{i}a_m}{\|{}^{i}a - {}^{i}a_m\|} \tag{56} $$

Now, consider the four inputs to the gravity summing block, Figs. 23 and 22.
The first two inputs estimate the inertial acceleration:

$$ {}^{i}a = \frac{dV}{dt} + \hat{\omega} \times V \tag{57} $$

Here we see the vector sum of the linear or translational component, $\frac{dV}{dt}$, and the

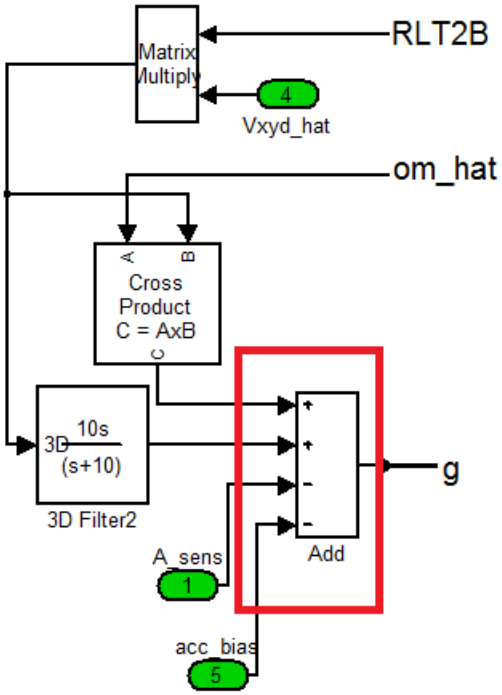


Figure 23: Measured gravity vector summing block

angular or rotational component, $\hat{\omega} \times V$.

1. $\left(\frac{10s}{s+10}\right)\left({}^{B}_{I}R *^{I} \hat{V}\right)$, i.e. $\frac{dV}{dt}$

   Linear velocity component of acceleration. To construct acceleration from velocity, the first order system $\left(\frac{10s}{s+10}\right)$ approximates a numeric derivative. This setup provides for an unchanging derivative at low frequencies, and an amplification of high frequencies. Furthermore, high frequencies are clipped at 20dB, with no derivatives. The step-response of this filter, shown in Fig. 24, is an exponential, natural response $e^{-10t}u(t)$. The filter's settling time of $T_s = \frac{4}{10} = 0.4s$ helps the system reach steady-state before the next inertial velocity update via the GPS receiver, assuming $1Hz$ (typical) sampling. The filter is labeled $3D$ because it processes vectors $v \in \mathbb{R}^3$.

2. $\hat{\omega} \times V$, i.e. $\hat{\omega} \times \left({}^{B}_{I}R * \hat{V}_{xyd}\right)$

   Angular velocity component of centripetal acceleration. Using ${}^{B}_{I}R$, rotate the inertial frame angular velocity into the body frame. The unit delay, z^{-1}, in series with $\hat{\omega}$ (Fig. 22) serves to handle the algebraic loops (or race conditions

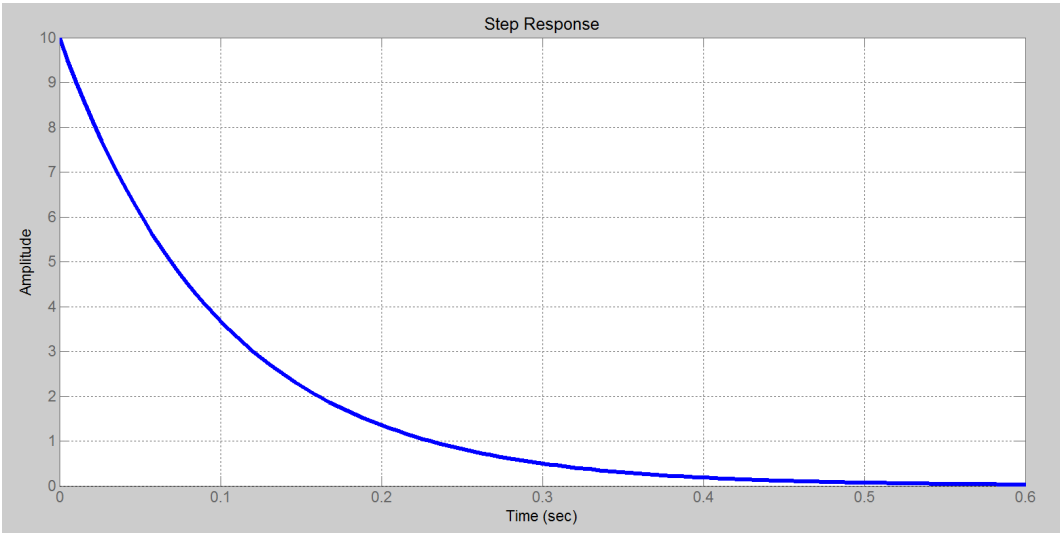


Figure 24: Step response of numeric derivative approximation: $\frac{10s}{s+10}$

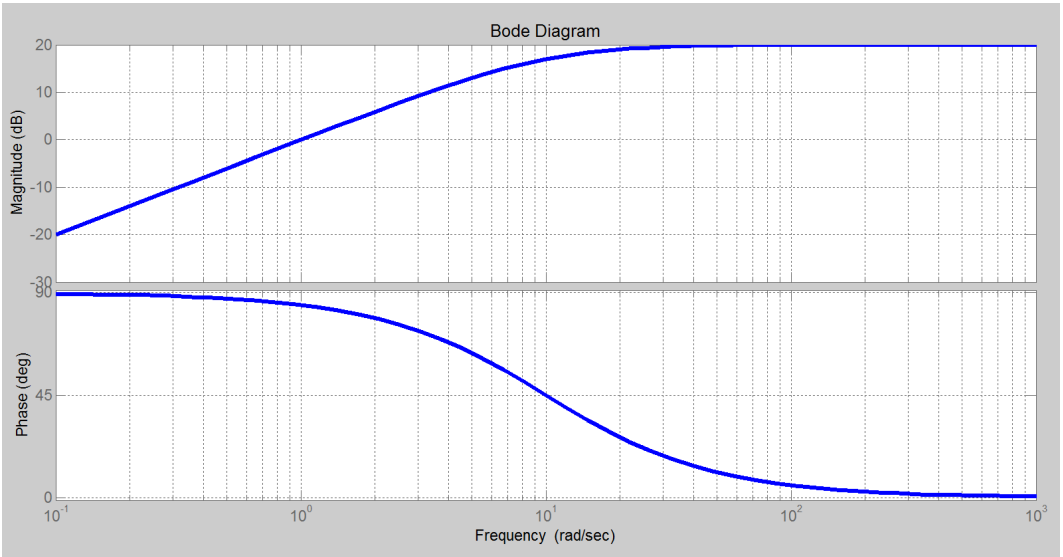


Figure 25: Bode plot of numeric derivative approximation: $\frac{10s}{s+10}$

in electronics) which would otherwise exist within the Simulink environment due to this construction.

3. A_{sense} or a_m

   Measured accelerometer reading. This component is subtracted to estimate the gravity vector per Eq. 56.

4. A_{bias}

   Accelerometer bias. Subtract this additive component for error correction.

The output of this block is the gravity measurement, g_{meas} in the body frame. Taking the cross product of this measurement with the estimated gravity vector taken from the appropriate column of the DCM yields an error term ϵ_g suitable for the feedback loop:

$$\epsilon_g = g_{meas} \times \hat{g} \tag{58}$$

$g_{meas} \times \hat{g}$ is selected instead of $\hat{g} \times g_{meas}$ because the inertial reference vector, gravity, can move with respect to the body axes, per section 2.5.2.

Track (ψ) Estimate A key caveat to the complementary filter approach is that vehicle heading is assumed to coincide with the ground track, a simplifying assumption that breaks down in the presence of significant wind.

As shown in Fig. 26, the second reference vector in the inertial frame is the unit vector of velocity in the xy plane, the ground track, ${}^{i}\psi_{meas}$. The body measurement of inertial heading in the horizontal plane is encoded in the first two elements of the updated DCM, components r_{11} and r_{12} from the following:

$$\overset{B}{I}R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{59}$$

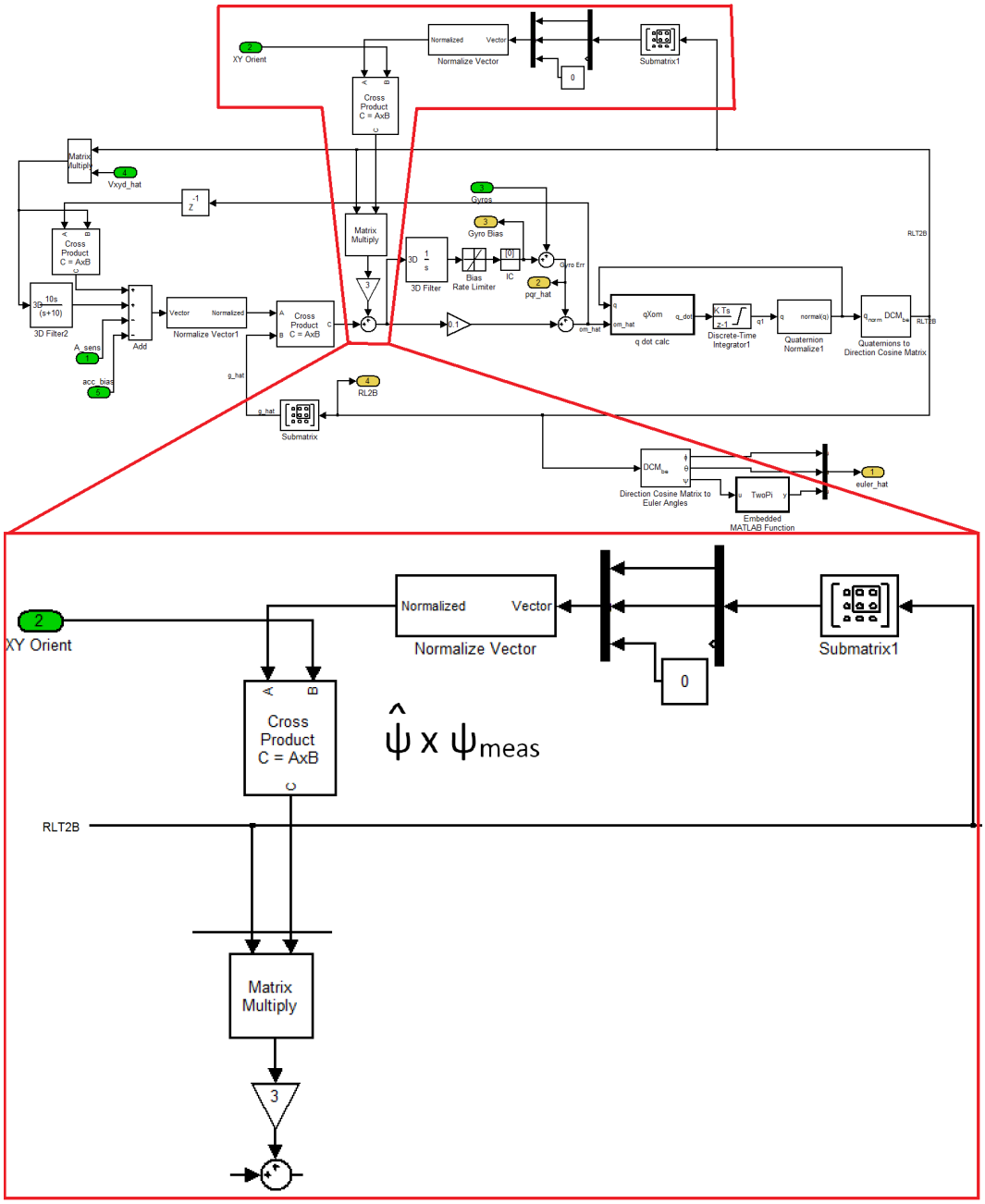


Figure 26: Track Estimate, context and expanded

This corresponds to the direction of the body x-axis as viewed in the inertial frame.

GPS **Course-Over-Ground** (**COG**) was used as the measurement, ${}^{i}\psi_{meas}$. Since the previous two vectors are expressed in the inertial frame, their cross product $({}^{i}\hat{\psi} \times {}^{i}\psi_{meas})$ is computed before rotating the yaw error ϵ_{ψ} into the body frame for feedback. The choice of ${}^{i}\hat{\psi} \times {}^{i}\psi_{meas}$ instead of ${}^{i}\psi_{meas} \times {}^{i}\hat{\psi}$ is due to the fact that the inertial vector of track is fixed with respect to the body x-axis, per section 2.5.2.

4.3.4 PI controller

The Proportional Integral (PI) filter, seen in Figs. 27 and 28, is a key component of the complementary filter. It provides numerous benefits:

1. Corrective feedback to the estimated DCM (driving ${}^{E}_{I}\hat{R} \rightarrow {}^{B}_{I}\ R_{true}$) via the cross product error terms from the gravity and track reference vectors. The integral component arises from the added pole at the origin in the transfer function of the controller. This component accumulates the steady-state error and then feeds this signal back to the controller, driving it to zero. Fig. 28 offers a simplified representation of the PI filter components.

2. Online estimation as well as correction of gyro bias, the system output $GyroBiases$ mentioned in Sec. 4.3.2. This topology will remove a constant bias.

3. Almost global stability of the algorithm (Lyapunov analysis available in [44]).

The corrected angular velocities in the body frame are named $p\hat{q}r$. Fusing $p\hat{q}r$ from the integrator term with the proportional feedback term yields $\hat{\Omega}$, one of two inputs to the quaternion integrator.

Feedback Gains The gain of 3 in series with the GPS COG measurements takes into account the fact that the accelerometers are noisier than the GPS COG measurements, thus this gain was set to trust the GPS measurements more. A proportional gain of $k_P = 0.1s^{-1}$ and an integral gain of $k_I = 0.001s^{-2}$ (or $1s^{-2}$) were used.

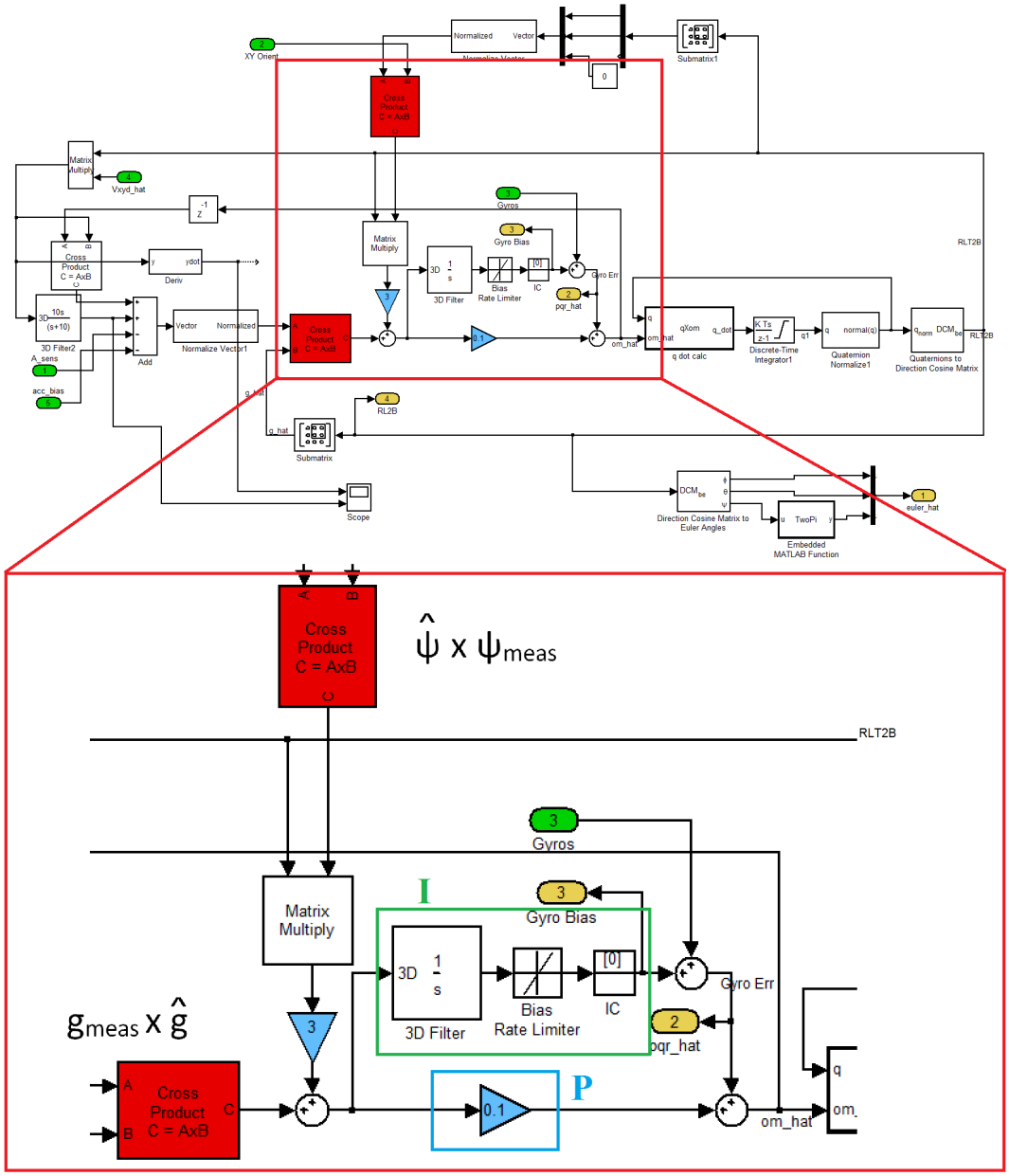


Figure 27: PI filter, context and expanded. Blue rectangle denotes the Proportional component, green rectangle the Integral component.

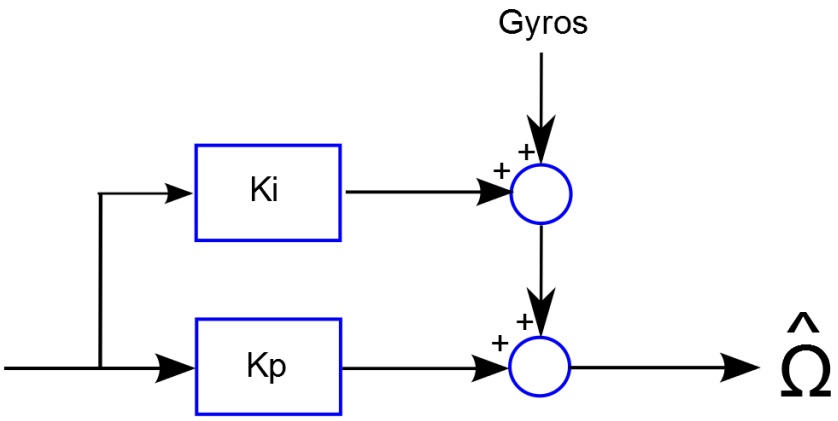


Figure 28: Simplified PI feedback loop

Angular velocity estimate $\hat{\Omega}$ represents the angular rate input to the quaternion integration block. It is the output of the complementary filter with PI feedback loop, which combines all of the gyro data, ψ correction, and leveling g_{est} correction into one, useful piece of information: a corrected estimate of the angular velocities of the aircraft in the body frame. It is calculated from the following:

$$\hat{\Omega} = G_{sense} + k_P(\epsilon_g + 3\epsilon_\psi) + k_I(\epsilon_g + 3\epsilon_\psi), \tag{60}$$

where $G_{bias} = k_I(\epsilon_g + 3\epsilon_\psi)$. ϵ_g is the error from the gravity estimate, and ϵ_ψ is the error from the track estimate through the GPS COG reference vector, both described in Sec. 4.3.3. The rate gyro inputs, G_{sense}, provide high bandwidth measurements of angular velocities in the body frame and thus play a key role in inertial navigation. However, their DC bias, as mentioned in Sec. 3.2.1, must be estimated and removed.

4.3.5 Propagation of the Attitude Kinematics

Fig. 29 depicts the propagation of the attitude kinematics. Note the two inputs, $\vec{q}$ and $\hat{\Omega}$, as well as the two outputs, $\vec{q}$ and $RLT2B$. Let us delineate the output as $\vec{q}_{k+1}$ and the input as $\vec{q}_k$, to specify the recursive, discrete-time nature of the solution.

Given the previous quaternion q and estimated angular velocities $\hat{\Omega}$, the algorithm calculates an estimated DCM ${}^E_I\hat{R}$ or $RLT2B$. The **q dot calc** block evaluates the

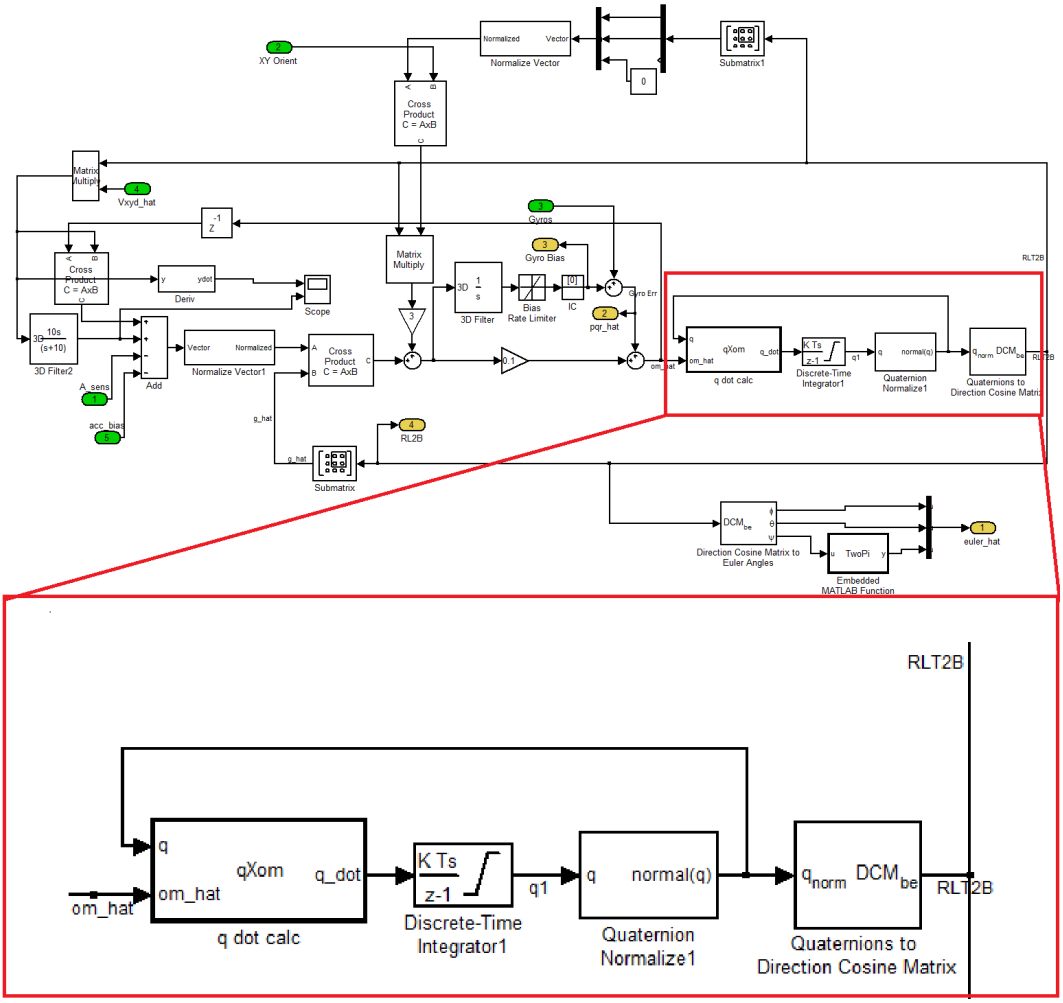


Figure 29: Propagation step, context and expanded

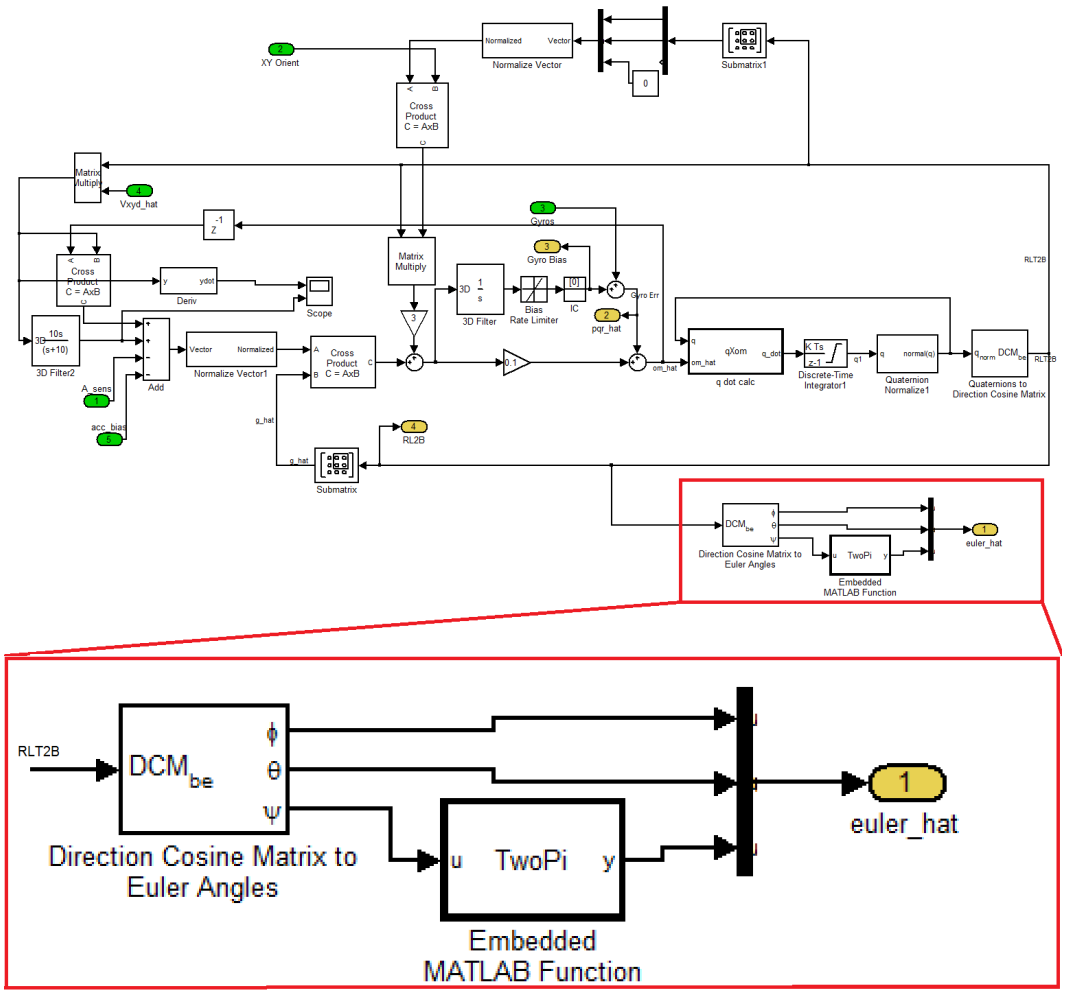


Figure 30: Euler angle conversion, context and expanded

quaternion kinematics of Eq. 38 by setting up the appropriate quaternion product of the previous quaternion and the current *pqr* angular rates. Next, the output of the **q dot calc** block is propagated directly via a discrete-time integrator, specifically the first-order, forward Euler method. Due to the absence of any additional pre-processing of the angular velocity estimate, $\hat{\Omega}$, constant (previous) angular velocity is assumed between time steps.

4.3.6 Post-Processing

After the integration, the output quaternion is fed to the next block and normalized to enforce the constraint $\sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} = 1$. Finally, quaternions are converted to the DCM representation to yield $RLT2B$, the Rotation of the Local

Tangent Plane to Body, which makes its way to the rest of the system.

As seen in Figure 30, the DCM is also converted to Euler angles, *euler_hat*. The **TwoPi** Embedded Matlab function ensures positive yaw outputs. Euler angles are produced only to be transmitted from the sensor DSC to the control DSC, which then uses these estimates in its control laws to fly the aircraft.

4.4 Performance Evaluation

As an algorithm analysis tool, the complementary filter was tested against flight data from Ref. [32] to provide a baseline. The low-cost Micronav INS and a GPS receiver provided inputs to the attitude algorithm, whose output was compared with the benchmark attitude solution estimated by the high-accuracy MIDG II device from Microbotics [48].

Figs. 31 - 33 show the time histories of angular results and errors of the complementary filter outputs versus those from the MIDG II. Table 1 lists the error statistics for these signals. Note that the mean error for all angles is less than 1 degree, which is acceptable given the hardware used.

This concludes the discussion of the baseline algorithm by which the SLUGS autopilot estimates attitude.

Roll (deg)		**Pitch (deg)**		**Yaw/COG (deg)**	
Mean	**Std**	**Mean**	**Std**	**Mean**	**Std**
0.4021	1.4953	-0.2279	1.1961	0.3917	4.4071

Table 1: Angular Errors in degrees

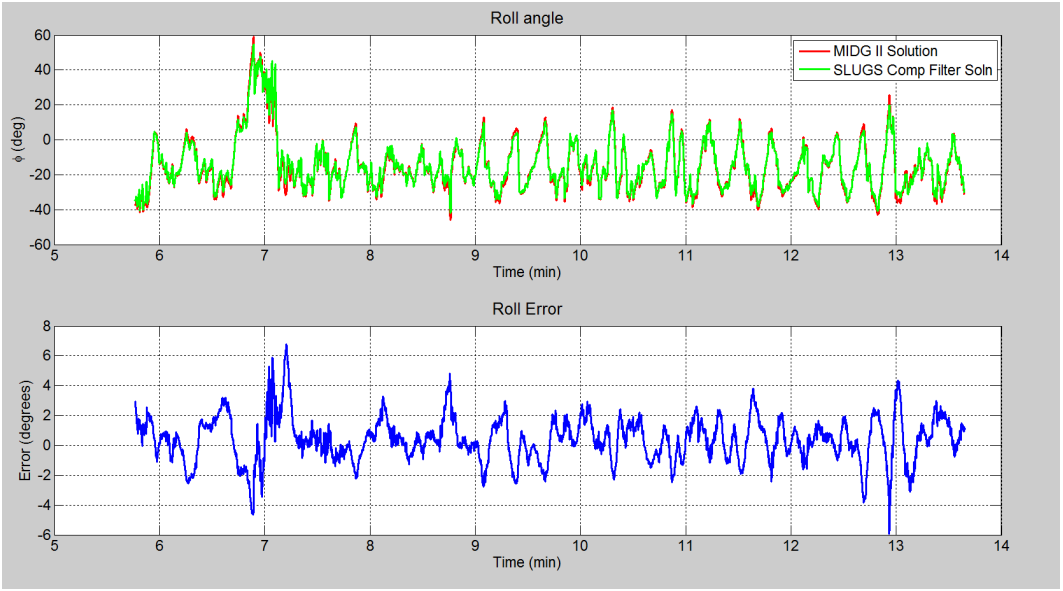


Figure 31: Bank Angles and error.

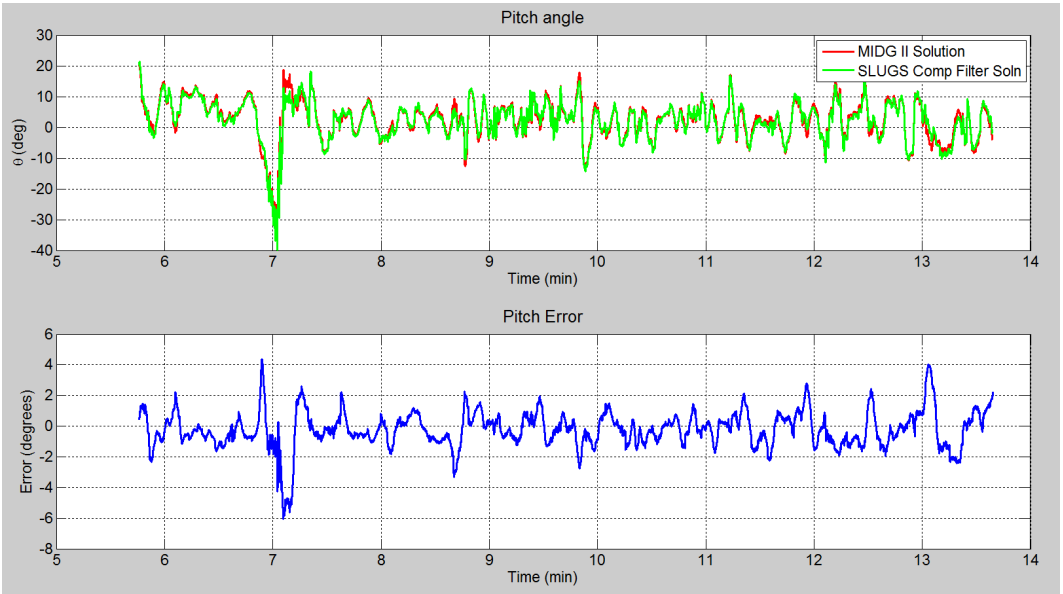


Figure 32: Elevation angles and error.

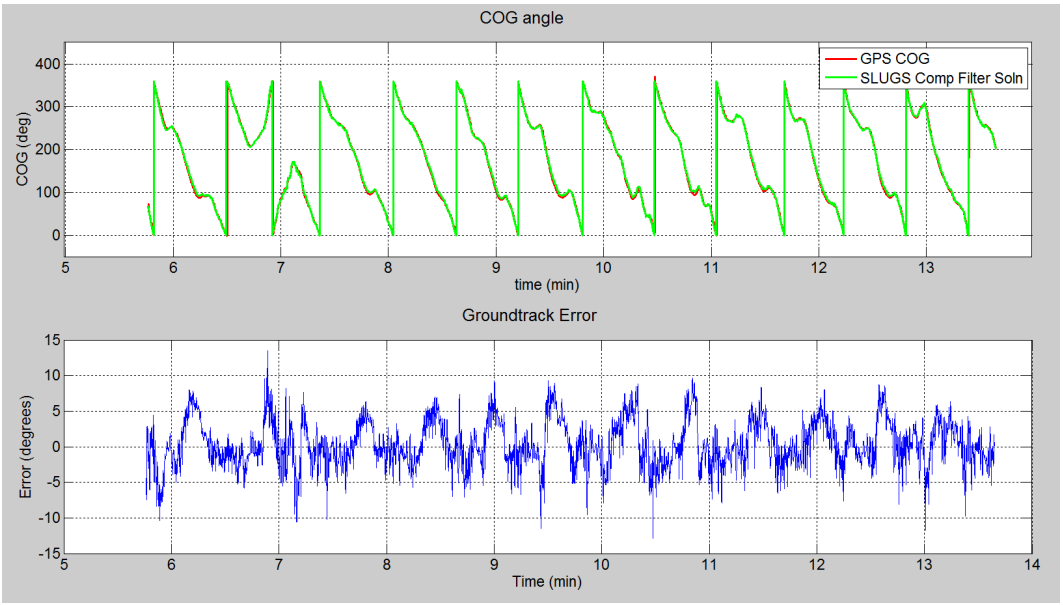


Figure 33: Ground-track angles and error.

5 Comparison of Attitude Parameterizations and Propagation Methods

5.1 Motivation

This section aims to examine in detail various attitude parameterizations and propagation methods which may be employed for the purpose of state estimation for low-cost unmanned (aerial or other) vehicles. This analysis strives to illuminate the specific advantages or disadvantages associated with each particular method. As such, the results will help researchers to evaluate the applicability of various methods, each of which may also be constrained by numerous compromises and mediocre or unknown performance. Having a quantitative comparison of some of the available schemes assists in related design decisions.

For example, a high-precision, targeting mission might require an attitude propagation and parameterization construct with a higher degree of accuracy and less regard for computational cost. Another embedded application might demand lower computational overhead with mid-range accuracy, due to the 8-bit architecture of the target microprocessor. Low cost and low power sensors tend to have greater noise. For such a low cost system, a scheme that attenuates or at least more slowly degrades under noise would be desirable. This section will analyze and discuss the specifics behind which options might better answer such design considerations. The key is to identify the tradeoffs being made and why.

The application domain for this analysis is low-cost, fixed-wing UAVs. Inexpensive, lower-quality, and higher noise sensors provide inertial reference vectors and angular velocities with limited frequency, accuracy, and precision. Though the algorithms and parameterizations examined here are well-suited for fixed-wing UAVs, they can also be used with other mobile platforms, such as rotary-wing UAVs, underwater vessels, or even ground vehicles.

5.2 Test framework

5.2.1 Test Candidates

The attitude propagation methods to be examined are the forward Euler integration method and the closed form solution of the matrix exponential. These propagation methods were chosen because they involve just one step and one functional evaluation of the associated kinematic equations (per Sec. 2.4), and as such are assumed to be quite amenable to real-time embedded system applications. Though more sophisticated multi-step, multi-function evaluation-based methods such as as Runge-Kutta, Adams-Bashforth/Adams-Moulton, etc. are readily available and provide high accuracy results, the scope of their use in this thesis is limited to providing a benchmark for the simpler methods presented.

In addition to evaluating the integration methods, various assumptions regarding the behavior of the angular velocity vector between time-steps are also analyzed. Specifically, three cases will be evaluated: 1) constant, 2) linearly varying, and 3) quadratic varying angular velocity. Many previous works (such as Gleason [32], etc.) follow the constant linear velocity assumption, which is generally reasonable given the high sampling rates typically incorporated in such systems (50+ Hz). Nevertheless, residual error terms could arise from such assumptions, especially when the vehicle performs highly dynamic maneuvers, and could result in increasingly erroneous attitude estimates. This work will explore whether or not there are significant performance differences between the three cases above, within the framework of the two core integration methods. Further details will be discussed in a subsequent section on implementation.

Finally, the experiments to follow will compare and contrast the differences associated with four attitude representations:

1. Euler Angles
2. Direction Cosine Matrix (DCM)

3. Quaternions

4. Angle-axis (or Eigen-axis)

The Euler angle representation is examined because it is pilot-centric, intuitive, and fairly straightforward to use. The DCM parameterization will be evaluated due to its convenience for reference frame conversions, to name just one benefit. Quaternions have gained widespread use, which is in no small part due to their computational efficiency, linearity, and other special properties. The angle-axis or Eigen-axis representation is a throwback to the early days of the first satellite navigation systems and adds further heterogeneity to the battery of test candidates.

More information on these attitude parameterizations may be found in Sec. 2.2.

5.2.2 Evaluation methods

With test candidates identified, the discussion now centers on the nature of the tests to be performed as well as metrics for evaluating test performance.

In general, a thorough test suite will evaluate the algorithm in question under both simulated and actual flight test data. Subjecting the algorithm to simulated inputs will allow precise control of various aspects of the data to discover specific performance properties of the algorithm. Analyzing the algorithm under real-world inputs illuminates any robustness or noise-immunity issues. When used together, the simulated and real inputs can provide a very specific measure of the algorithm's performance, especially if the tests incorporating simulated data are designed carefully.

Unit and integration testing of the algorithms are performed. Unit testing will evaluate the algorithm in isolation, under synthetic data, while integration testing will evaluate algorithmic performance within the framework of the SLUGS complementary filter from the previous chapter, using real-world flight data. This flight data is provided by Gleason [32]. Specifically, *pqr* inputs were provided by the

(now-discontinued) Crossbow MicroNav sensor suite and Euler angle outputs were provided by the high-quality, Microbotics' MIDG II INS/GPS device [48].

For the complementary filter framework of SLUGS, rate gyro data as well as data from GPS and accelerometers are sampled by the system and used to provide bias estimation and sensor fusion. Unit tests with synthetic data employ typical control-systems input waveforms, specifically sinusoids representing per-axis angular velocities.

5.2.3 Performance Metrics

In order to properly compare and contrast the performance of the test suite, metrics are required to evaluate algorithm performance within those tests. These performance metrics are as follows:

1. **Accuracy.** Each algorithm will be evaluated against a benchmark or truth measure. For unit tests with synthetic data, the output of a fourth-order, Runge-Kutta approximation will serve as ground truth, due to its accepted, high accuracy results. For complementary filter-based integration tests with real-world flight data, the Euler angle outputs of the MIDG II sensor will serve as the key benchmark for comparisons. For the unit tests, accuracy will be computed in terms of the RMS error between the algorithm's output and the output of the Runge-Kutta routine. The relative, user-defined classification scale for these errors is depicted in Tables 2 and 3. For the integration tests, the highest end of the scale ("excellent") was set as the level of accuracy provided by the MIDG II sensor (as reported by the datasheet), since this sensor is the benchmark.

2. **Computational Load.** Each algorithm will be evaluated using Matlab's profiler tool, which calculates the algorithm's time to execute the angular velocity processing and propagation routines in seconds. The relative, user-defined scale for these errors is depicted in Table 4.

3. **Noise response.** This metric aims to answer quantitatively how closely an algorithm tracks a desired signal when that signal is corrupted by noise. Statistics are calculated using a difference signal: the algorithm's response to a noise-added signal minus the algorithm's response to a noise-free signal. Root-mean-square-error will again be the key statistic and basis for comparisons.

UNIT TEST METRICS	
RMS Error (deg)	
Magnitude	*Classification*
1.E-03	Excellent
1.E-02	Good
1.E-01	Fair
>1.0	Poor

Table 2: User-defined error scale for Accuracy (degrees)

Note that the above scales were defined by creating bins of all the results. A few points bear mention regarding the real-world flight test data:

1. The angular velocity inputs from the rate gyros likely possess a high degree of noise. Analysis of the individual noise components in this signal was limited due to the fact that neither detailed datasheets nor the (discontinued) sensor itself was available. Still, visual inspection of the signal provides a high-level view of the 3-axis signal characteristics, Fig. 34. Also, Figs. 35a - 35c illustrate that the distribution of each of the p, q, and r components are approximately Gaussian, with a large number of samples at the mean (0, indicating level flight).

2. The flight data is marked by sampling gaps, for reasons unknown. Essentially, there are periods up to 0.18s long where 5-7 data samples are simply missing. Fig. 36 shows the regularity and frequency of occurrence in just the first 13 seconds of flight data. This issue was ameliorated by filling in the gaps through cubic spline interpolation, the results of which were analyzed and found to maintain the statistics of the original signal.

INTEGRATION TEST METRICS	
Mean Error (deg - roll, pitch)	
Range	*Classification*
0.4-1.4	Excellent
1.4-2.4	Good
2.4-3.4	Fair
3.4+	Poor
Mean Error (deg - yaw)	
Range	*Classification*
2-3	Excellent
3-4	Good
4-5	Fair
5-6	Poor
Standard Deviation (deg - all)	
Range	*Classification*
0-2	Excellent
2-4	Good
4-6	Fair
6+	Poor

Table 3: User-defined error scale for Accuracy; high end of the scale based off MIDG II datasheet [48].

COMP. LOAD METRICS	
Range(sec)	*Classification*
0-2.0E-5	Excellent
2.0E-5 - 4.0E-5	Good
4.0E-5 - 6.0E-5	Fair
> 6.0E-5	Poor

Table 4: User-defined error scale for Computational Load

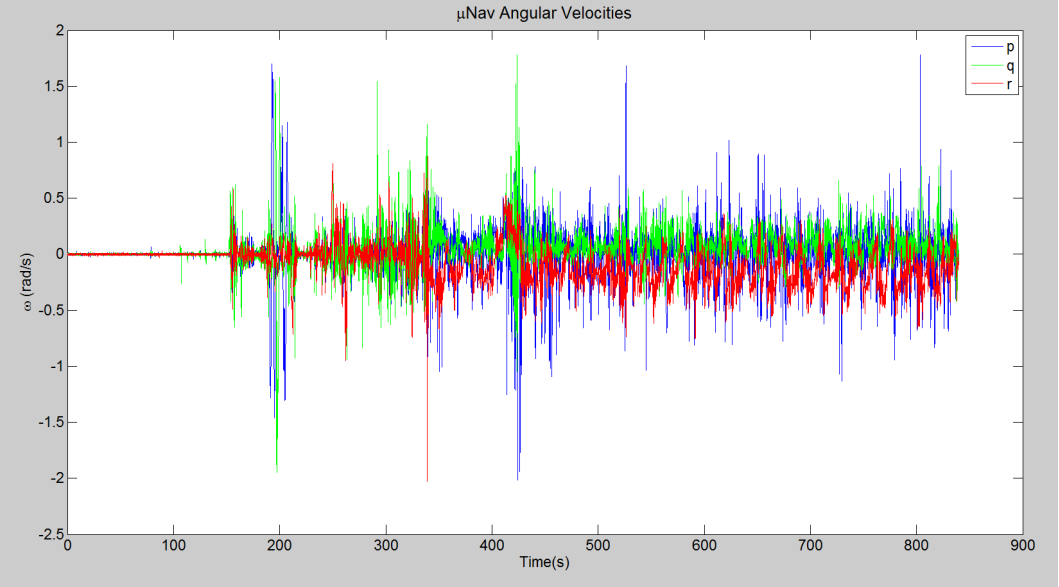


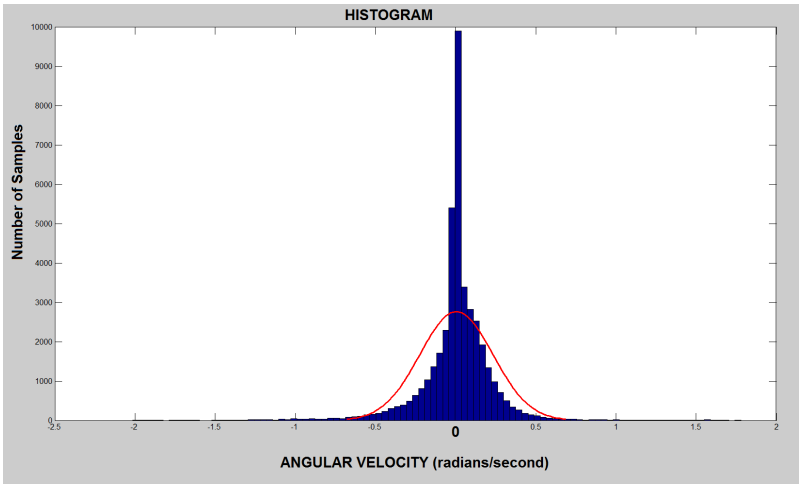
Figure 34: μNav angular velocity time history

3. Sampling rate for the MicroNav system was 50Hz, whereas the sampling time for the SLUGS complementary filter is 100Hz. For complementary filter tests, Simulink accounted for this difference by interpolating between samples.

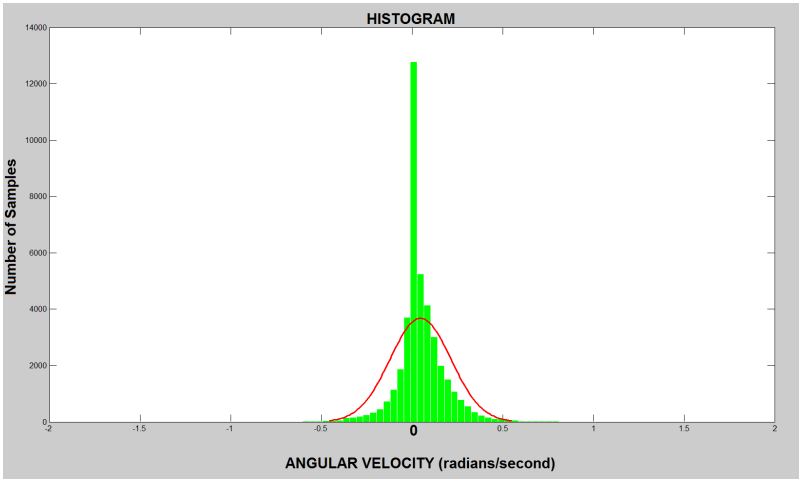
5.2.4 Implementation

Angular Velocity The previous section on test candidates identified the two integration methods to be tested: forward Euler integration and the closed form solution of the matrix exponential. Within these two propagation methods, this thesis also tested three different assumptions regarding the behavior of the angular velocity vector between time samples, 1) constant, 2) linear, and 3) quadratic. The hypothesis to be tested was "does the accuracy of propagation depend (at least in part) upon how well the chosen angular velocity assumption matches the underlying behavior of the exact angular velocity waveform?"

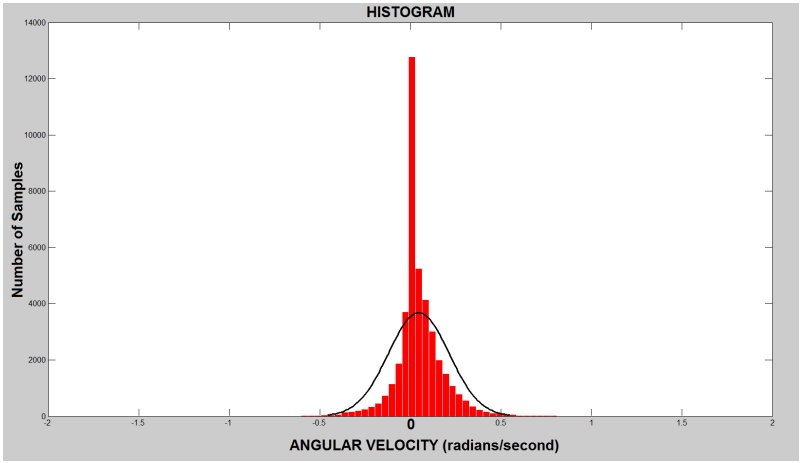
Consider an arbitrary waveform, depicted in Fig. 37, which describes the behavior of one of the three axis body rate vectors with respect to time. Assume that the system samples the angular velocity vector at three different time steps: t_{n-1}, t_n, t_{n+1}, as shown in Fig. 38; note that t_{n+1} represents the current time step, t_n the previous



(a) X-axis



(b) Y-axis



(c) Z-axis

Figure 35: Distributions for Body-frame Angular Velocities reported by the μNav's rate gyros

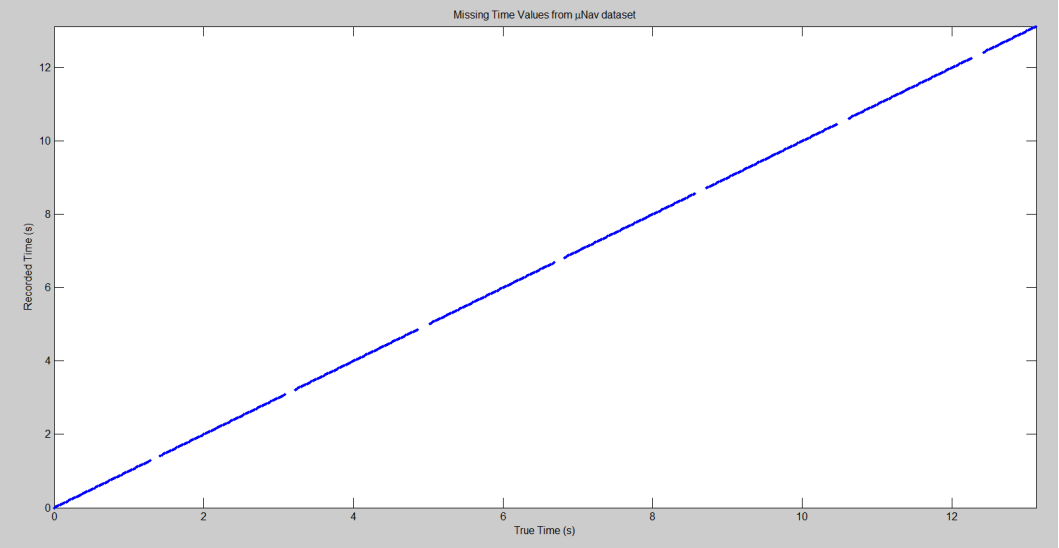


Figure 36: Missing μNav flight data. X axis represents true time, Y axis represents recorded time samples.

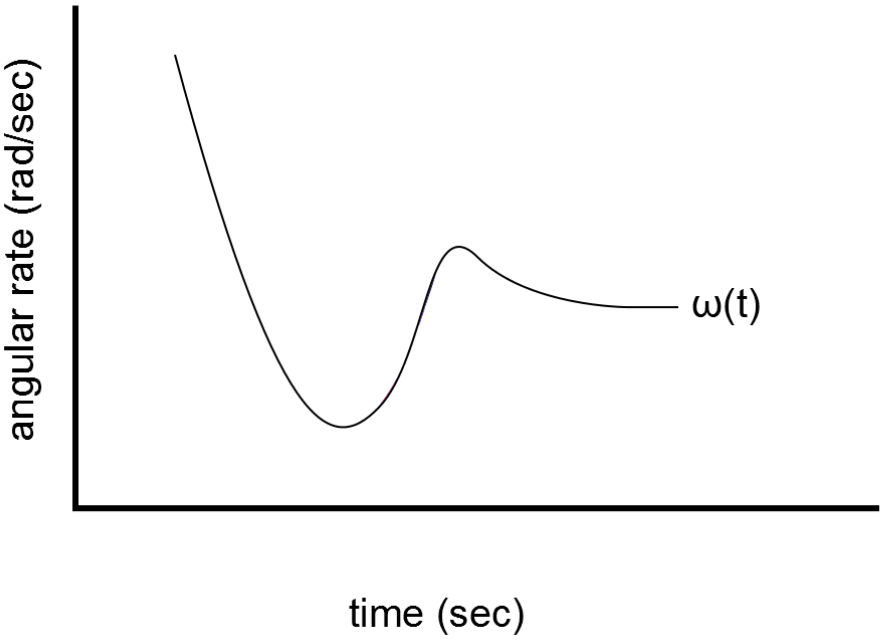


Figure 37: Arbitrary angular velocity function, $\omega(t)$.

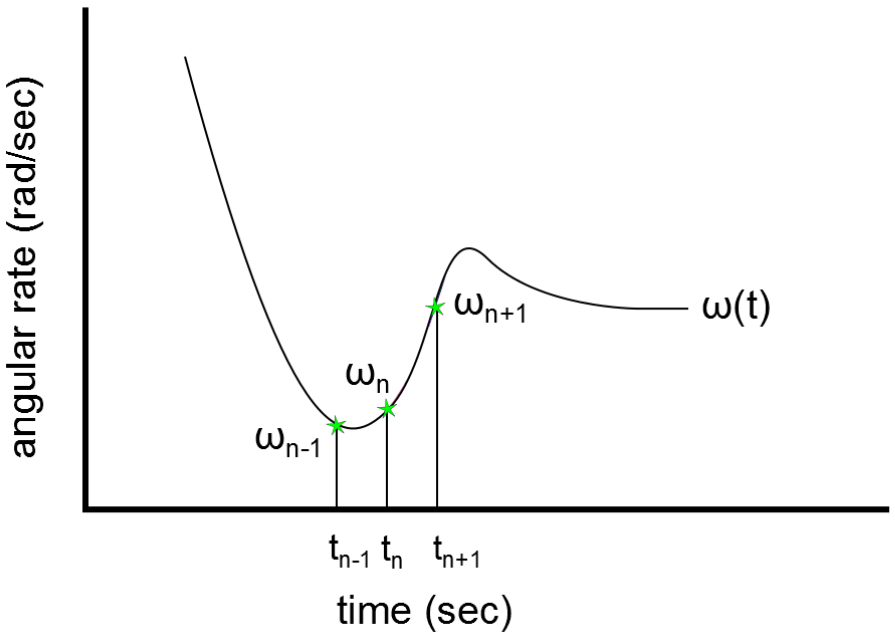


Figure 38: Angular velocity vector sampled at three different time points.

time step, and so on. These samples of the function values are represented by the green stars, labeled $\omega_{n-1}, \omega_n, \omega_{n+1}$. For the sake of simplifying this example, assume that the readings are not corrupted by noise.

The next task is to estimate the attitude at time step t_{n+1} based on some combination of these samples of the angular velocity vector. The simplest approach for this integration is to assume the angular velocity is constant between time steps, in the manner of a zero-order hold. Fig. 39 shows this method resembles the construction of Riemann sums from integral calculus. This method essentially partitions the region between samples, creating a series of equal width rectangles. Note that there are two ways to perform this sampling: 1) use the angular velocity of the previous time step, $\omega(t_n)$, or 2) use the rate from the current time step, $\omega(t_{n+1})$. Fig. 40 shows that this results in either a prediction forward scenario or a backwards estimation approach, respectively. Estimation based on this sampling method requires the least computational load, only one sample is used directly. Note that the shorter the time

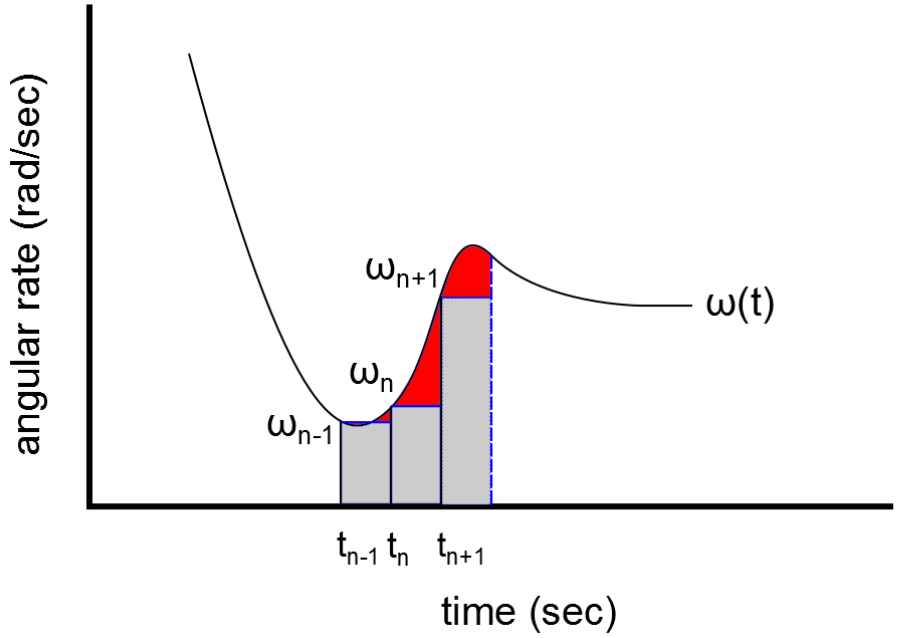


Figure 39: Constant angular velocity between time steps, along the lines of a zero-order hold or Riemann sum. The shaded red region between the approximation function and the function for $\omega(t)$ indicates the discretization error.

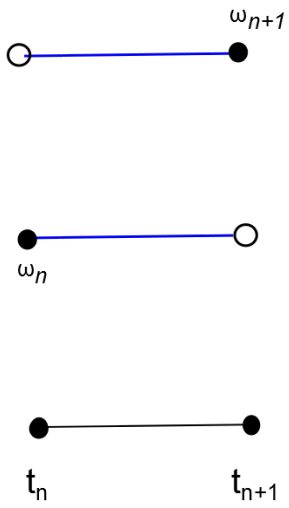


Figure 40: Constant angular velocity for current time step (t_{n+1}) and previous time step (t_n). Note that using the previous angular velocity ($\omega(t_n)$) results in prediction forward, while using the current version ($\omega(t_{n+1})$) results in estimation backward.

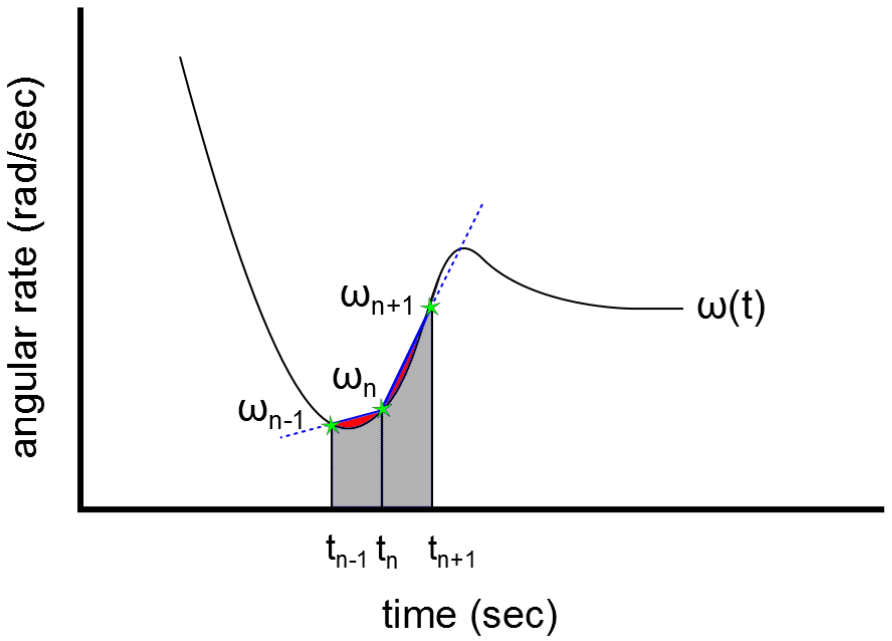


Figure 41: Linearly varying angular velocity assumption, approximated by a method like the trapezoidal rule.

step or width of the approximating rectangles, the more accurate this method can be.

The next method assumes there is some linear variation of the angular velocity vector between time steps. The method works by calculating a simple average of the angular velocities at two time steps, i.e. it uses $\frac{1}{2}(\omega(t_{n+1}) + \omega(t_n))$. This first-order method approximates a numeric derivative and bears resemblance to the **trapezoidal rule** from integral calculus, as shown in Fig. 41. In addition to the additional processing time to compute the average, this method requires two samples of the angular velocity vector rather than just one. The goal of using this approach is to track the angular velocity vector more closely than the constant assumption theoretically allows, and thus lower the error. However, in the presence of high noise and shortened time steps, this method could amplify system noise. Thus, it is up to the design engineer to take into account the noise characteristics of the sampling sensor and post-processing routines when weighting the tradeoffs of better tracking performed by this method versus any associated noise amplification behavior.

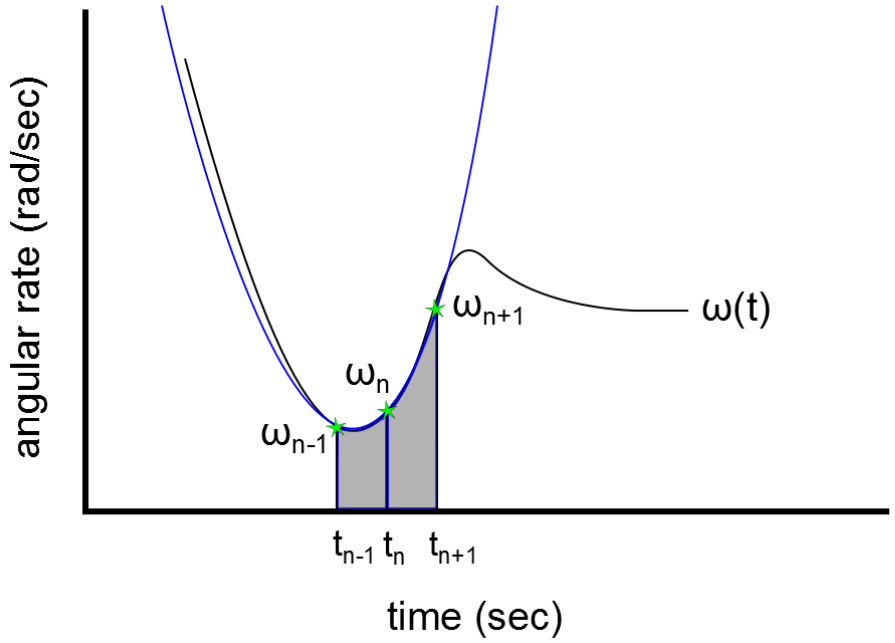


Figure 42: Quadratic-varying angular velocity assumption, approximated by a method like Simpson's rule.

The last method considers the angular velocity vector to be varying according to a quadratic function between time steps, thus it attempts to fit a second-order polynomial to three samples, $\omega(t_{n-1}), \omega(t_n), \omega(t_{n+1})$. This method is the next step beyond the trapezoidal rule, creating a numeric approximation of attitude in a manner similar to **Simpson's rule** from integral calculus:

$$\int_a^b f(x)dx \approx \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right] \tag{61}$$

Fig. 42 depicts this Simpson's rule extension. Before the next section points out the differences between Simpson's rule and the equation this thesis uses to test the quadratic angular velocity, note that both methods will be shown to take the form of **quadrature**, an approximation to a definite integral using a weighted sum of function values.

To test a quadratic-varying angular velocity vector, an expression was derived through an adaptation of an idea presented by Wertz [73], related to quaternion-based attitude propagation using the matrix exponential. Wertz' idea was that the Ω function representing the angular velocity is not constant: it is time-varying. He developed a Taylor series expansion for this function as follows:

$$\overline{\Omega} = \frac{1}{h}\int_{t_n}^{t_{n+1}} \Omega(t)\mathrm{d}t = \Omega_n + \frac{1}{2}\dot{\Omega}_n h + \frac{1}{6}\ddot{\Omega}_n h^2 + \dots \tag{62}$$

where h represents the sample step size. Through the $\overline{\Omega}$ function, Wertz' formulation for the matrix exponential propagation of attitude used **time-averaged** rate information rather than **instantaneous** body rates, which allowed for errors on the order of only h^3. The above $\overline{\Omega}$ function inspired the following discrete, $2nd$ degree approximation:

$$\texttt{OmegaBar} = \Omega_n + \frac{1}{2}\dot{\Omega}_n h + \frac{1}{6}\ddot{\Omega}_n h^2 \approx \overline{\Omega} \tag{63}$$

To implement `OmegaBar` required calculating $\dot{\Omega}_n$ and $\ddot{\Omega}_n$. To match the discrete nature of the sampling environment, a central finite-difference scheme [74] was employed as follows:

$$\dot{\Omega}_n = \frac{\Omega_{n+1} - \Omega_{n-1}}{2h} + 10^{-4} \tag{64}$$

$$\ddot{\Omega}_n = \frac{\Omega_{n+1} - 2\Omega_n + \Omega_{n-1}}{h^2} + 10^{-4} \tag{65}$$

The formulation in Eq. 63 assumes an angular rate that is fully differentiable and continuous. To derive the expression presented in Item 4, substitute Eqs. 64 and 65 into Eq. 63, as follows:

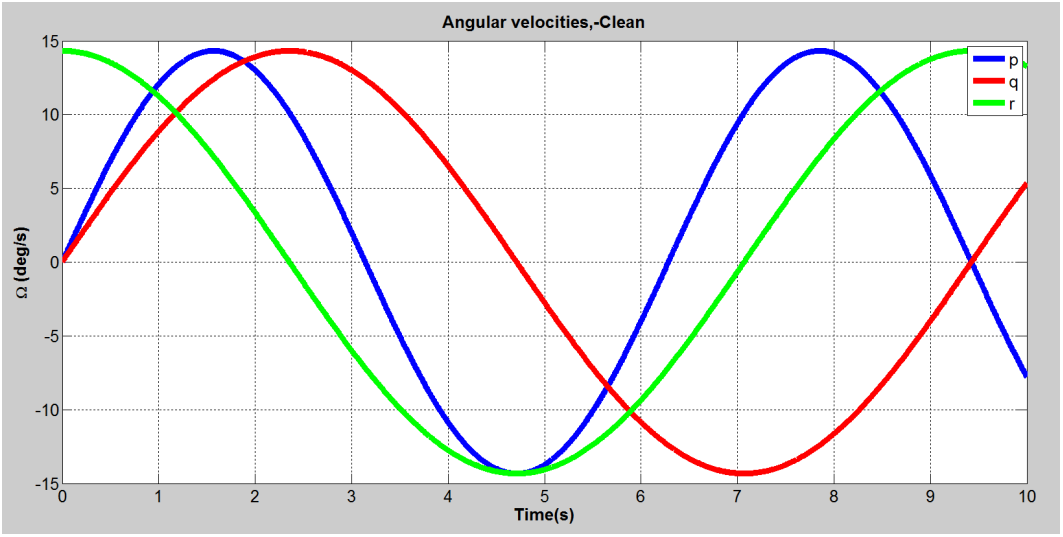


Figure 43: Sinusoidal Input Waveform: p (blue), q (red), r (green) in deg/s

$$\begin{aligned}
\texttt{OmegaBar} &= \Omega_n + \frac{1}{2}\left(\frac{\Omega_{n+1} - \Omega_{n-1}}{2h}\right)h + \frac{1}{6}\left(\frac{\Omega_{n+1} - 2\Omega_n + \Omega_{n-1}}{h^2}\right)h^2 \\
&\text{"} = \Omega_n + \frac{1}{4}\left(\Omega_{n+1} - \Omega_{n-1}\right) + \frac{1}{6}\left(\Omega_{n+1} - 2\Omega_n + \Omega_{n-1}\right) \\
&\text{"} = \left(-\frac{1}{4} + \frac{1}{6}\right)\Omega_{n-1} + \left(1 - \frac{1}{3}\right)\Omega_n + \left(\frac{1}{4} + \frac{1}{6}\right)\Omega_{n+1} \\
&\text{"} = -\frac{1}{12}\Omega_{n-1} + \frac{2}{3}\Omega_n + \frac{5}{12}\Omega_{n+1} \\
\texttt{OmegaBar} &= \frac{1}{12}\left(-\Omega_{n-1} + 8\Omega_n + 5\Omega_{n+1}\right)
\end{aligned}$$

Note that the above differs slightly from the formula for Simpson's rule, Eq. 61, by the weights placed on the different samples. As with the trapezoidal rule, sound use of this method involves considering the tradeoffs of gaining better signal tracking versus amplifying noise. Another consideration is the additional processing time required for calculating `OmegaBar`, discussed further in the subsequent section on computational load results.

Given the above three angular velocity assumptions between time steps, it remains to be discussed the nature of the angular velocity waveforms used for unit testing. As described previously and depicted in Fig. 43, sinusoid input waveforms representing

p, q, and r angular velocities were sent to the test algorithms. The analytical expressions for the sinusoidal waveforms were as follows:

$$\vec{p} = \frac{1}{4}\sin(t) \tag{66}$$

$$\vec{q} = \frac{1}{4}\sin\left(\frac{2t}{3}\right) \tag{67}$$

$$\vec{r} = \frac{1}{4}\cos\left(\frac{2t}{3}\right) \tag{68}$$

where the amplitude of the maximum angular rate was $\frac{1}{4}$ rad/s. The frequencies of the above signals are $f_p \approx 0.16Hz$, $f_q \approx 0.11Hz$, and $f_r \approx 0.11Hz$. The reason the body-frame x-axis inputs were chosen to have higher frequency is because the UAV can typically roll faster than it can pitch or yaw. Given the above, the minimum Nyquist sampling rate to avoid aliasing would be 0.318Hz.

Attitude Propagation Methods Based on the previous description of angular velocity assumptions, the full description of the attitude propagation methods is as follows:

1. Forward Euler with constant angular velocity, from previous time step
2. Forward Euler with linearly varying (averaged) angular velocity
3. Matrix Exponential with constant angular velocity, from previous time step
4. Matrix Exponential with constant angular velocity, from current time step
5. Matrix Exponential with linearly varying (averaged) angular velocity (trapezoidal rule)
6. Matrix Exponential with time-averaged angular velocity (`OmegaBar`)

The forward Euler method used is the exact formulation expressed in Eq. 43:

$$f(t_{n+1}) = f(t_n) + hf'(t_n),$$

where $f(t_{n+1})$ is the attitude estimate for the current time step, $f(t_n)$ is the previous attitude, h is the time step, and $f'(t_n)$ represents the kinematic equation(s) for the parameterization used. The equations for $f'(t_n)$ were the same as those presented in Sec. 2.3.2 for Euler angles (Eq. 35), quaternions(Eq. 38), the direction cosine matrix (Eq. 36), and the angle-axis scheme(Eq. 39).

The four different matrix exponential propagation methods from the list of six methods above were performed with respect to only the quaternion and DCM parameterizations. The quaternion-based scheme employed Wertz' formulation from the spacecraft attitude estimation and control literature [73]:

$$q(t_{n+1}) = \left[\cos\left(\frac{\|\omega\|h}{2}\right)\mathbf{I} + \frac{1}{\|\omega\|}\sin\left(\frac{\|\omega\|h}{2}\right)\Omega_n\right] q(t_n) \tag{69}$$

where $\mathbf{I}$ is the 4×4 identity matrix, ω_n is the norm of the angular velocity vector omega, i.e. $\|\omega\| = \sqrt{\omega_x^2 + \omega_y^2 + \omega_z^2}$, h is the sampling period, and Ω_n is the 4×4 skew-symmetric matrix of pqr angular velocities. Note that these p, q, and r values can be one of 4 varieties, such as $p(t_{n-1})$, $p(t_n)$, $(p(t_n) + p(t_{n+1}))/2$, and $(1/12) * [-p(t_{n-1}) + 8p(t_n) + 5p(t_{n+1})]$.

The DCM-based matrix exponential follows the development by R. Murray from the literature on robotic manipulation [49]:

$$\begin{aligned} R(t_{n+1}) &= R(t_n)e^{\Omega_x}, \text{where} \\ e^{\Omega_x} &= \mathbf{I} + \frac{\Omega_x}{\|\omega\|}\sin(\|\omega\|h) + \frac{\Omega_x^2}{\|\omega\|^2}(1 - \cos(\|\omega\|h)) \end{aligned} \tag{70}$$

where h is the sampling period, $\mathbf{I}$ is the 3×3 identity matrix, Ω_x is the 3×3 skew-symmetric matrix of angular velocities, and $\|\omega\|$ is again the norm of $\vec{\omega}$.

The matrix exponential formulation of Park and Chung [51] was also considered, but not employed:

$$e^{\Omega_x h} = \mathbf{I} + \alpha\Omega_x + \frac{1}{2}\beta(\Omega_x)^2 \tag{71}$$

where $\mathbf{I}$ is the 3×3 identity matrix, and

$$\alpha = sc \quad \text{and} \quad \beta = s^2, \tag{72}$$

and

$$s = \frac{\sin\left(\frac{\|\omega\|h}{2}\right)}{\frac{\|\omega\|}{2}} \quad \text{and} \quad c = \cos\left(\frac{\|\omega\|h}{2}\right). \tag{73}$$

The above formulation allows for efficient computation of the matrix exponential's differential and product inverse. This formulation is mathematically equivalent to Murray's, as follows. Begin by substituting the expressions for α and β into Eq. 71, as follows:

$$e^{\Omega_x h} = \mathbf{I} + \left(\frac{\sin(\|\omega\|h)}{\|\omega\|}\right)\Omega_x + \frac{1}{2}\left(\frac{\sin^2\left(\frac{\|\omega\|h}{2}\right)}{\frac{\|\omega\|^2}{4}}\right)\Omega_x^2 \tag{74}$$

$$\text{"} = \mathbf{I} + \frac{\Omega_x}{\|\omega\|}\sin(\|\omega\|h) + \frac{\Omega_x^2}{\|\omega\|^2}\left(2\sin^2\left(\frac{\|\omega\|h}{2}\right)\right)$$

$$\text{"} = \mathbf{I} + \frac{\Omega_x}{\|\omega\|}\sin(\|\omega\|h) + \frac{\Omega_x^2}{\|\omega\|^2}\left(1 - \cos(\|\omega\|h)\right) \tag{75}$$

which is the same as Eq. 70. Note, however, that Park and Chung's approach is more complicated than Murray's. The additional computational expense incurred by Park and Chung's formula could have resulted in computational savings later, had the domain of operation been in robotic manipulators, where the derivative of the matrix exponential would need to be computed. However, for attitude estimation in this thesis, such a form is unnecessary.

For the unit test truth comparison, a fourth-order Runge-Kutta method was employed. Given the initial condition f_0 for the attitude and the time step h, the desired attitude for the current time step, $f(t_{n+h})$ is obtained through the following:

$$f(t_{n+h}) = f_0 + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4), \text{where} \tag{76}$$

$$k_1 = hf'(t_n, f_0),$$

$$k_2 = hf'(t_{n+h/2}, f_0 + \frac{k_1}{2}),$$

$$k_3 = hf'(t_{n+h/2}, f_0 + \frac{k_2}{2}),$$

$$k_4 = hf'(t_{n+h}, f_0 + k_3).$$

f' is computed by the kinematic formula for $\dot{q}$ expressed in Eq. 38. The exact Matlab implementation used was an adaptation of a program by J. Mathews, featured in an accompaniment to the book [46] in Matlab Central's File Exchange.

The sampling rate used by the unit testing as well as integration testing framework was 100Hz. Future work for these experiments includes testing the above algorithms at multiple sampling rates and evaluating the time dependent performance characteristics.

Noise Tests In order to evaluate algorithm response to noise, a deterministic noise source was added to the sinusoidal angular velocity input waveforms. This noise consisted of a 3-channel (pqr), additive Gaussian signal generated by Matlab's `randn()` function. This additive noise has a mean of 0 and an amplitude 10% of the magnitude of the underlying waveform, resulting in a signal-to-noise ratio of 20dB. The per-axis distributions for this noise are depicted in Figs. 45a - 45c, while the time history of the noise-corrupted inputs is shown in Fig. 44. The time history of the additive noise is visible in Fig. 46.

To ensure that the noise source was deterministic across the different parameteri-

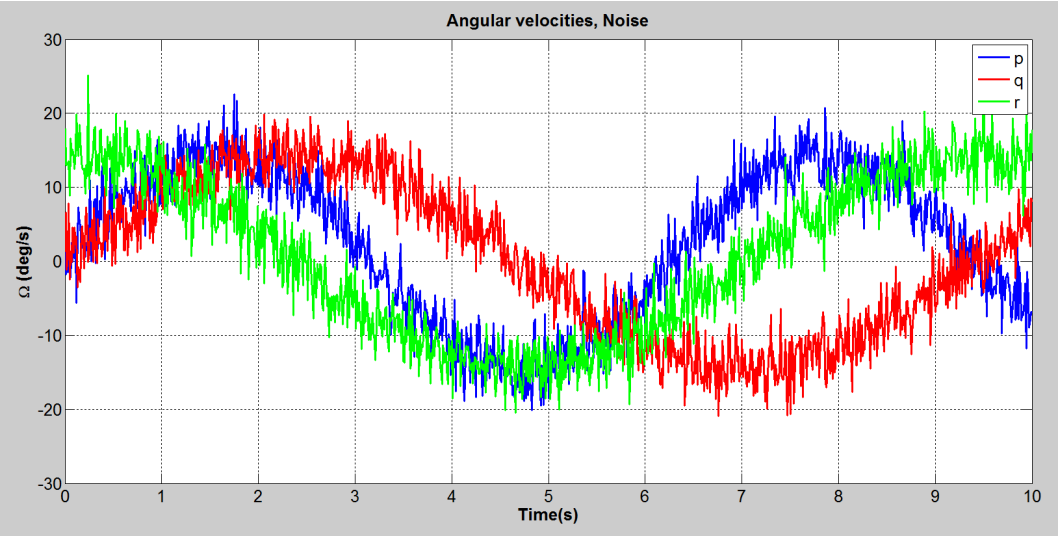
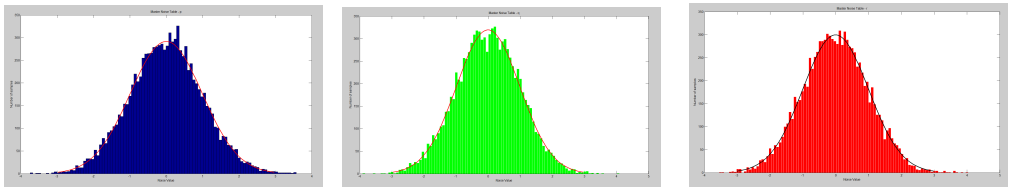


Figure 44: Sinusoid Input Waveforms for p(blue), q(red), and r(green) angular velocities corrupted by additive, Gaussian noise. Units are deg/s.



(a) X-axis (b) Y-axis (c) Z-axis

Figure 45: Distributions for noise added to body-frame angular velocities

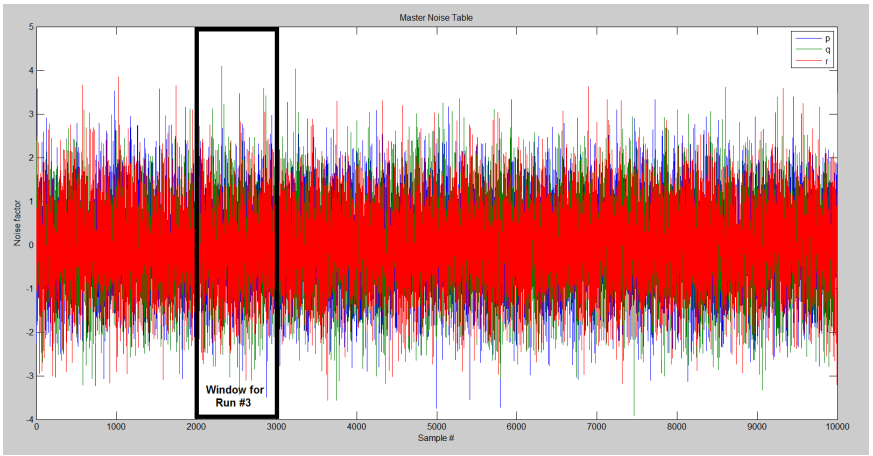


Figure 46: Sliding window approach for selecting deterministic values of noise from a master lookup table. Here the window for the 3rd test iteration is highlighted by the black rectangle.

zation tests, a lookup table of 3-channel noise values was generated before all the test runs. This table could be indexed by time, due to the 0.01s step size between each sample. The size of the table was such that it provided noise values for 10 iterations of each 10s test. For each test iteration, a different section of the table was accessed via an incrementally advancing, sliding window. Fig. 46 succinctly illustrates this concept. This lookup table approach ensured that each algorithm tested was subjected to exactly the same noise values at exactly the same time in the test procedures, ensuring determinism across tests.

In addition to the 10 noise test runs, each algorithm ran once on noise-free data. The residual, difference signal was subsequently created for each test run by subtracting the noise-free output of the algorithm from the noise-added output of the algorithm. These 10 residuals were then concatenated into a single, master difference signal. This concatenated signal was subsequently analyzed according to the squared error and other characteristics. For a benchmark comparison, the fourth-order Runge-Kutta routine also acted upon the aforementioned inputs and the corresponding concatenated difference signal was created.

Integration Tests Integration tests consist of evaluating algorithm performance metrics within the framework of the complementary filter of the SLUGS AP, described in the previous chapter. Each propagation method to be tested replaced the original quaternion integrator and ran for its own separate test cycle. The effect of the complementary filter is to fuse the data from the gyros, accelerometer, GPS, and estimated position and velocity information, improving the angular velocity estimates over what is available from the rate gyros alone. Flight data originated from Crossbow's MicroNav sensor onboard a fixed wing-UAV; these data included rate gyro and accelerometer measurements. The dataset also included higher-quality measurements from an integrated GPS-INS, the MIDG II [48]. This device provided a benchmark for Euler angle outputs.

The aforementioned flight data from Ref. [32] spans 14 minutes; the roll and pitch angle outputs of the complementary filter are computed against this entire time frame. The yaw angle outputs, however, require special handling due to the the attitude filter's architecture. Recall that the SLUGS AP does not compute UAV heading, per se, but ground track, assumed to align with vehicle heading under typical flight conditions of low wind. A key part of the reference vector for ground track is the inertial-frame measurement of GPS COG. The nature of received GPS signals is such that accurate ground track measurements are at best misleading and at worst meaningless without sufficient velocity on the part of the platform using such a device. Fig. 47 plots the groundspeed of the UAV used for the flight data in this experiment. In this figure, one can readily identify that the window of data from about 350s to 830s is more appropriate for providing reliable ground track measurements than the window from 0s to 350s. Therefore, for the purposes of evaluating the yaw/groundtrack outputs in this phase of the experiment, the window of data from about 6 minutes to 13.5 minutes (347s to 820s) will be employed. In Fig. 47, colored circles delineate this window's boundaries. The flight regime of the UAV consisted of flying in concentric ellipses over the ground with some altitude variations. This relatively gentle trajectory validates the slight decoupling of the yaw results from pitch and roll.

5.3 Results

As a truth comparison for unit tests, a fourth-order Runge-Kutta numeric approximation was run on the same *pqr* input data used by the various integration and parameterization algorithms. This multi-stage method propagated the attitude kinematics. A design decision needed to be made in terms of which parameterization method to use for the form of the kinematic equations to be propagated. Because these equations can be nonlinear and differ for each parameterization, it was unknown at the outset of these experiments if a significant difference would result from using a kinematic parameterization different from that of the algorithm being tested. In other words, could there be a significant benefit to the quaternion-based algo-

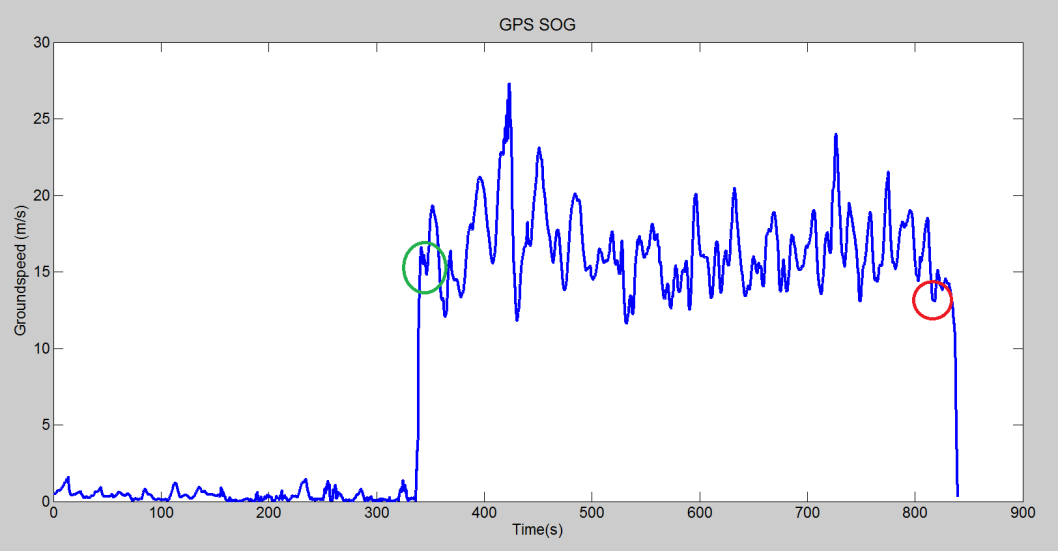


Figure 47: Groundspeed measurements (m/s) taken from the Micronav sensor onboard the UAV used for flight test data collection. Notice the circled starting point (green) and ending point (red) of the data analyzed for yaw (groundtrack) within the integration testing framework

rithms if they used a quaternion kinematics-based RK4 method vs. an RK4 method expressed within a different parameterization? Correspondingly, would there be a significant penalty observed on the part of the Euler-angle-based algorithms which were associated with a comparison against a non-Euler-angle kinematic representation used by the RK4 method? If there were significant differences, then it would make the most sense to compare each algorithm with an RK4 method whose kinematics were expressed in the same parameterization.

To evaluate this problem, the RK4 algorithm was run using the kinematics of quaternions as well as Euler angles. The mean differences between results from the two approaches were found to be on the order of $1e^{-9}$ radians (or $5e^{-8}$ degrees). Similar differences were expected, but not verified, between the propagation of the kinematics of the DCM and the angle-axis methods. Because of these small differences, it was decided that using one parameterization method for the RK4 method to be used as a comparison against all the test algorithms would be sufficient, as long as accuracy differences smaller than $1e^{-8}$ rad ($5e^{-7}$ deg) between any two parameterization methods were considered to be insignificant. In the end, the quaternion method was

chosen to represent the kinematics to be propagated, due to its simplicity, linearity, and computational advantages.

5.3.1 Accuracy

Unit tests - synthetic data This section will evaluate the angular outputs of the different attitude determination algorithms according to the root mean square error as compared to the RK4 method. Time histories of the error plots will also be analyzed; these errors were calculated by subtracting the response of the RK4 method from the response of each propagation algorithm. The full table of RMS errors may be seen in Fig. 48. Table 5 summarizes these and serves as the initial point of discussion.

Roll		**RMSE (deg)**
DCM/Quaternions	Mexp, Trap	1.37E-04
Angle-axis	Fwd. Euler	8.89E-02
Pitch		**RMSE (deg)**
DCM/Quaternions	Mexp, Trap	1.13E-04
Angle-axis	Fwd. Euler	1.18E-01
Yaw		**RMSE (deg)**
DCM	Fwd. Euler	1.24E-02
Angle-axis	Fwd. Euler Trap	2.28E-01
Sum for all Euler angles		**RMSE (deg)**
DCM/Quaternions	Mexp, Trap	5.10E-04
Angle-axis	Fwd. Euler	3.65E-01

Table 5: Summary statistics for best and worst performers from unit testing.
Test metric: Accuracy, Units: degrees.

Perusal of the last column of the RMSE unit test summary table for accuracy (Table 5) highlights the significant, overall differences between the various approaches. The best overall performers, the quaternion- and DCM-based matrix exponential

		RMS Error			
Parameterization	Integration Method	Roll (deg)	Pitch (deg)	Yaw (deg)	Sum (deg)
DCM	Mexp, Trap	1.37E-04	1.13E-04	2.60E-04	**5.0984E-04**
Quaternions	Mexp, Trap	1.37E-04	1.13E-04	2.60E-04	**5.0989E-04**
Quaternions	Mexp, OmegaBar	9.36E-04	3.47E-04	3.73E-04	1.6560E-03
DCM	Mexp, OmegaBar	9.36E-04	3.47E-04	3.73E-04	1.6560E-03
DCM	Fwd. Euler Trap	1.66E-02	7.31E-02	9.59E-03	9.9324E-02
Quaternions	Fwd. Euler Trap	2.06E-02	3.51E-02	5.25E-02	1.0824E-01
Quaternions	Fwd. Euler	7.12E-02	6.88E-02	5.82E-02	1.9817E-01
DCM	Mexp, Const. Prev	7.05E-02	5.15E-02	8.03E-02	2.0235E-01
Quaternions	Mexp, Const. Prev	7.05E-02	5.15E-02	8.03E-02	2.0235E-01
Quaternions	Mexp, Const. Curr	7.05E-02	5.15E-02	8.05E-02	2.0260E-01
DCM	Mexp, Const. Curr	7.05E-02	5.15E-02	8.05E-02	2.0260E-01
Euler angles	Fwd. Euler	7.71E-02	9.41E-02	5.83E-02	2.2952E-01
Euler angles	Fwd. Euler Trap	7.71E-02	9.41E-02	5.83E-02	2.2952E-01
DCM	Fwd. Euler	7.02E-02	9.82E-02	7.66E-02	2.4502E-01
Angle-axis	Fwd. Euler Trap	4.69E-02	1.06E-01	1.85E-01	3.3795E-01
Angle-axis	Fwd. Euler	8.89E-02	1.18E-01	1.58E-01	**3.6502E-01**

Figure 48: RMS Errors (degrees) in Euler angle outputs for all methods; Unit Tests

with averaged angular velocity, performed an order of magnitude better than the next best method, the OmegaBar approach, and **three** orders of magnitude better than the overall worst performer, the angle-axis-based Euler method with constant previous angular velocity. There was a tie for first place here, between the DCM and quaternions, because the size of the difference between these two was $5e^{-8}$ degrees, smaller than the $5e^{-7}$ degree threshold for "significant differences" described in the previous section.

Fig. 49 shows the error between the angular output and the RK4 truth of the DCM-based methods with respect to the roll angle. Notice that the OmegaBar and trapezoidal matrix exponential approaches seem to have nearly zero error at the scale of 0.1 degrees on the y axis. Also notice that the two constant angular velocity, matrix exponential approaches appear to be nearly perfect mirror images of one another about the horizontal line of zero error; in the plots, this line is closely tracked by the green matrix exponential OmegaBar and trapezoidal approaches. When removing the two forward Euler responses for this plot as well as the constant angular velocity assumption, matrix exponential responses, Fig. 50 is the result: a close-up on the OmegaBar and trapezoidal matrix exponential responses. Note the y axis has shrunk to $1e^{-4}$ degrees. In this figure, notice that the trapezoidal version

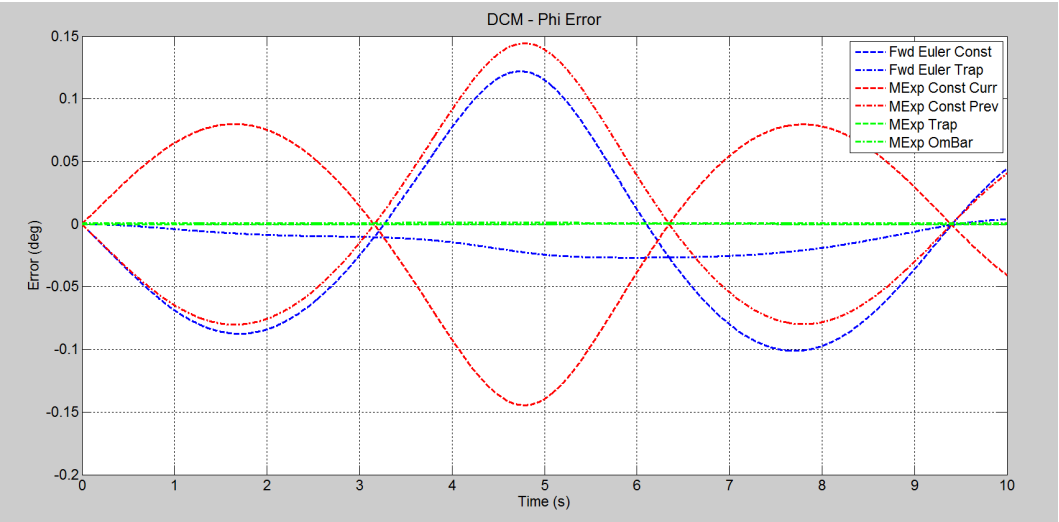


Figure 49: Bank angle error (deg)
DCM-based propagation routines for unit testing

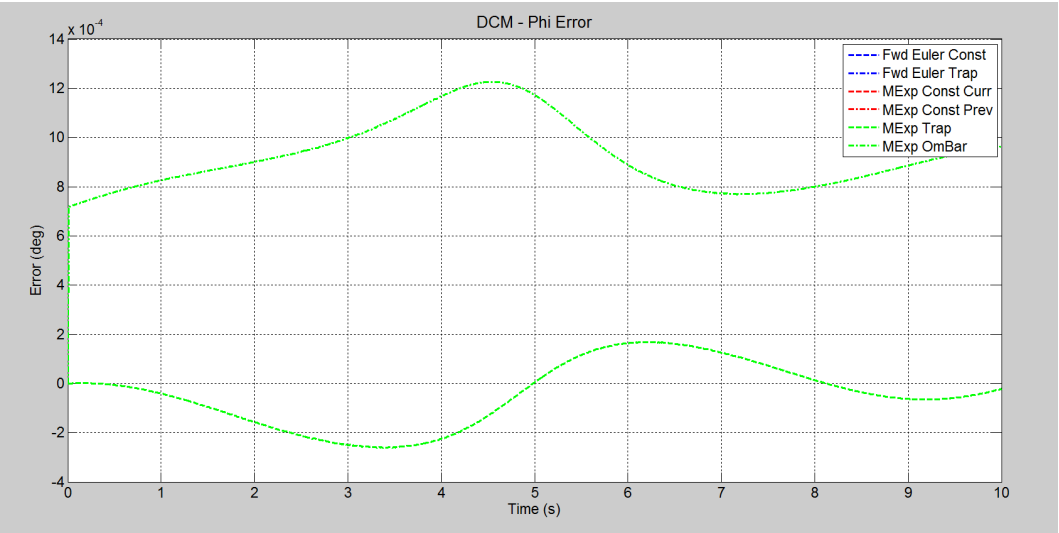


Figure 50: Close up of Bank angle error (deg) for DCM-based matrix exponential routines

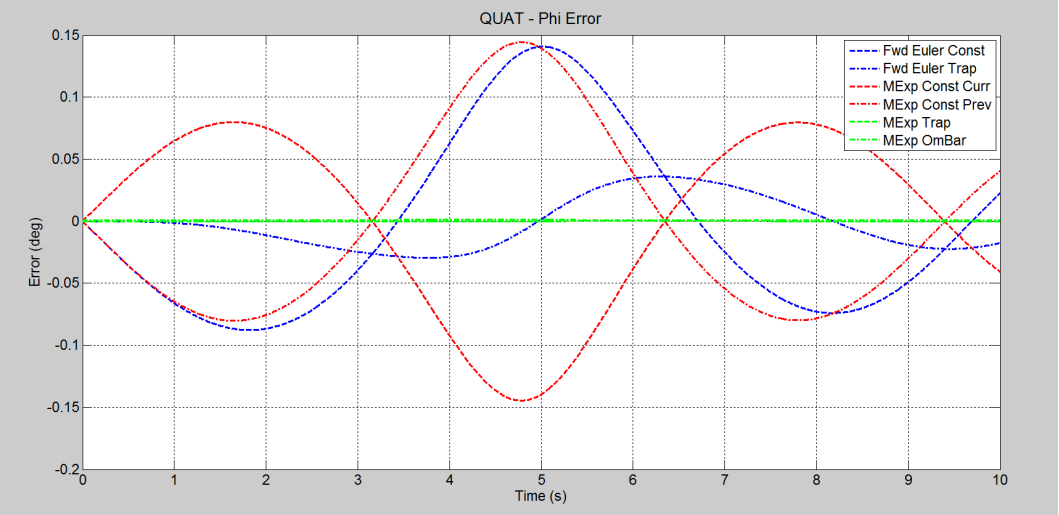


Figure 51: Bank angle error (deg)
Quaternion-based propagation routines for unit testing

(bottom) has the smallest tracking error. In Fig. 51, one can see the similarities between the error signals of the DCM and quaternion methods.

The Euler angle responses, example for roll angle shown in Fig. 52, demonstrate nearly identical responses regardless of the trapezoidal versus the constant previous angular velocity assumption. This is not the case for the angle-axis response, roll plots shown in Fig. 53. Here the statistics reinforce what is readily visible through the time history plot: the trapezoidal method offered smaller RMS error ($2e^{-2}$ deg) than the constant previous angular velocity method ($8e^{-2}$ deg).

Regarding which parameterization performed the forward Euler method with constant angular velocity the best, quaternions took first place, followed by Euler angles. In terms of the using Euler with the trapezoidal method, the DCM slightly edged out the quaternions. With the rest of the propagation methods (matrix exponential) and angular velocity assumptions, quaternions and DCM representations were tied for first place.

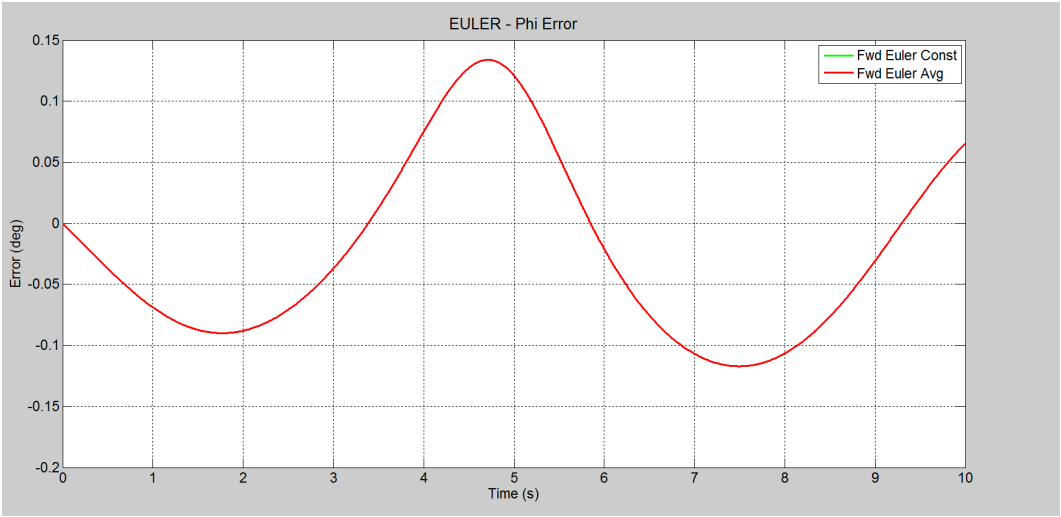


Figure 52: Bank angle error (deg)
Euler angle-based propagation routines for unit testing

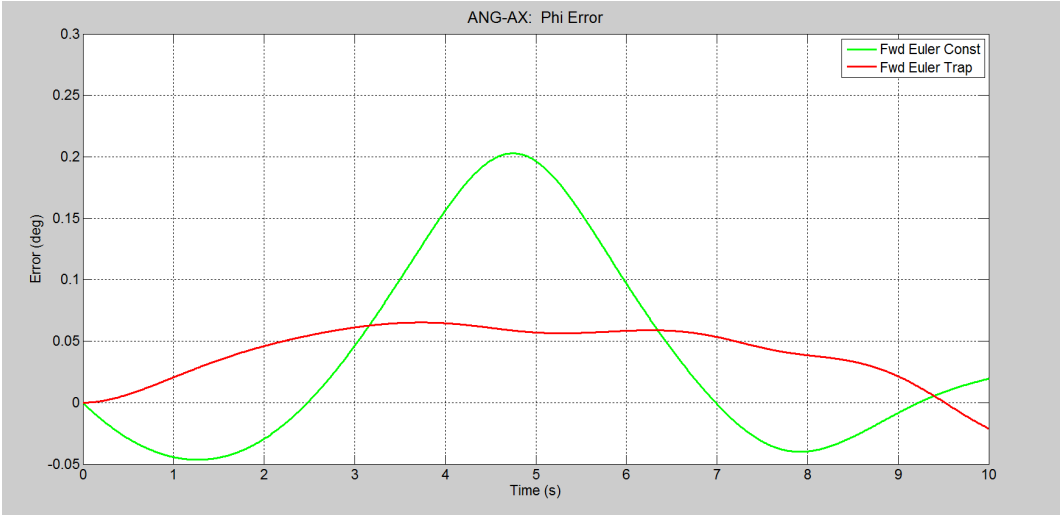


Figure 53: Bank angle error (deg)
Angle-axis-based propagation routines for unit testing

5.3.2 Noise Response

Recall that the aim of the noise response tests is to evaluate quantitatively how closely an algorithm tracks a given input signal, when that signal is corrupted by additive Gaussian noise. Noise response evaluation will be performed by examining the residual errors created by subtracting each algorithm's noise-free output from its noisy output. RMS errors for each method will be calculated and compared. Time history plots will be inspected, and, for comparison, the tracking response of the truth routine will be presented.

Roll		**RMSE (deg)**
DCM/Quaternions (best)	Mexp, Const. Curr	7.68E-01
Angle-axis (worst)	Fwd. Euler Trap	7.87E-01
Pitch		**RMSE (deg)**
DCM/Quaternions (best)	Mexp, OmegaBar	4.58E-01
Angle-axis (worst)	Fwd. Euler	4.64E-01
Yaw		**RMSE (deg)**
DCM/Quaternions (best)	Mexp, Const. Curr	9.52E-01
Angle-axis (worst)	Fwd. Euler	9.70E-01
Sum for all Euler angles		**RMSE (deg)**
DCM/Quaternions (best)	Mexp, Const. Curr	2.18
Angle-axis (worst)	Fwd. Euler	2.22

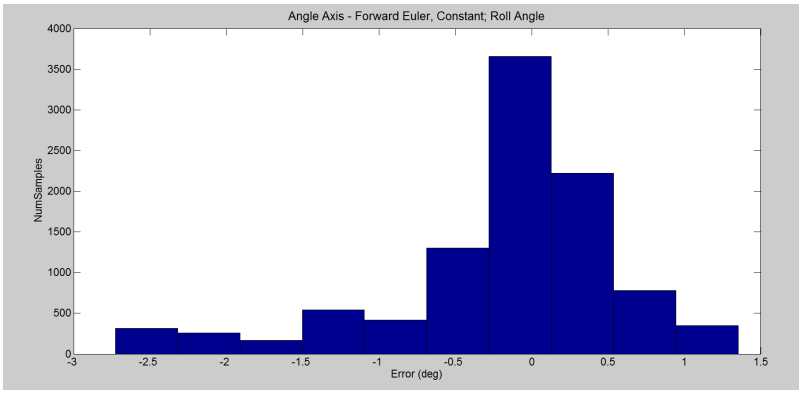
Table 6: Summary statistics for best and worst performers from unit testing.
Test metric: Noise Response, Units: degrees.

In Fig. 54, notice that the RMS errors for all methods were nearly indistinguishable from one another. Even Table 6 shows that the gap in root-mean-squared-errors between the best and worst performers is quite small. Within these small differences, the DCM- and quaternion-based matrix exponential with constant current angular velocity assumption performed with the lowest overall RMSE. Individual roll, pitch, and yaw angle differences were too small to be noteworthy.

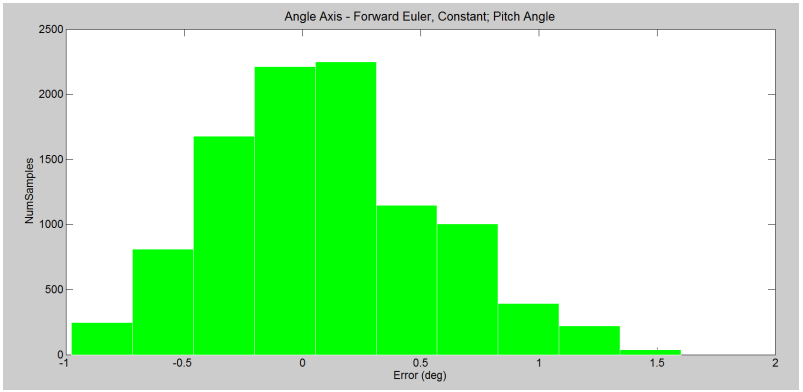
		RMSE			
Parameterization	**Integration Method**	**Roll (deg)**	**Pitch (deg)**	**Yaw (deg)**	**Sum (deg)**
Quaternions	Mexp, Const. Curr	7.68E-01	4.59E-01	9.52E-01	**2.1791**
DCM	Mexp, Const. Curr	7.68E-01	4.59E-01	9.52E-01	**2.1791**
DCM	Mexp, OmegaBar	7.73E-01	4.58E-01	9.58E-01	2.1884
Quaternions	Mexp, OmegaBar	7.73E-01	4.58E-01	9.58E-01	2.1884
Quaternions	Fwd. Euler Trapezoidal	7.75E-01	4.61E-01	9.55E-01	2.1918
Quaternions	Mexp, Trap	7.77E-01	4.60E-01	9.56E-01	2.1927
DCM	Mexp, Trap	7.77E-01	4.60E-01	9.56E-01	2.1927
DCM	Fwd. Euler Trapezoidal	7.76E-01	4.61E-01	9.56E-01	2.1936
Angle-axis	Fwd. Euler Trapezoidal	7.76E-01	4.62E-01	9.62E-01	2.2001
DCM	Mexp, Const. Prev	7.86E-01	4.62E-01	9.60E-01	2.2087
Quaternions	Mexp, Const. Prev	7.86E-01	4.62E-01	9.60E-01	2.2087
Quaternions	Fwd. Euler	7.85E-01	4.63E-01	9.61E-01	2.2088
Euler angles	Fwd. Euler	7.87E-01	4.62E-01	9.60E-01	2.2089
Euler angles	Fwd. Euler Trapezoidal	7.87E-01	4.62E-01	9.60E-01	2.2089
DCM	Fwd. Euler	7.86E-01	4.63E-01	9.60E-01	2.2099
Angle-axis	Fwd. Euler	7.86E-01	4.64E-01	9.70E-01	**2.2196**
	Runge-Kutta 4th order (truth)	4.15E-01	4.07E-01	6.13E-01	1.43

Figure 54: Noise Response - RMS Errors in Euler angle outputs for all methods, Unit Tests (deg). Results are sorted according to the last column.

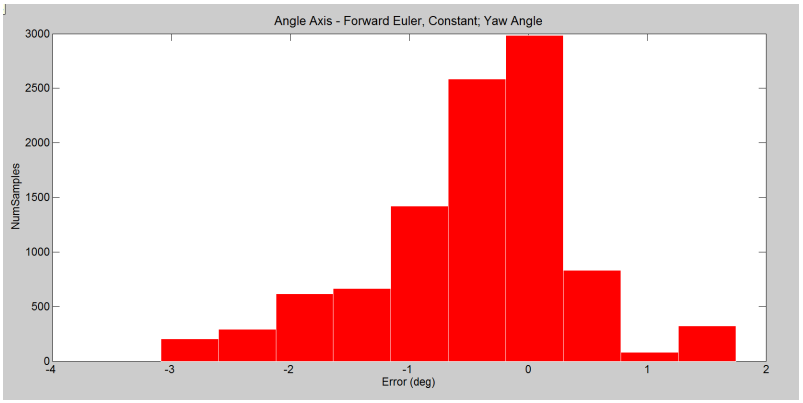
The distributions of the residuals were inspected in hopes of extracting more prominent differences than those seen in the RMSE tables listed prior. Fig. 55 shows distributions for roll, pitch, and yaw angles for the Angle-Axis parameterization with the forward Euler propagation method, constant previous angular velocity. Subsequent inspection of the distributions for all other parameterizations showed few, if any differences. Interestingly, even the distributions for the matrix exponential methods demonstrated remarkable similarity to those seen in Fig. 55. Time history plots of the residual signals were also analyzed, but were found to be in line with the above observations. Given these trends, no data gathered from the various test modules gave any algorithm, parameterization method, or angular velocity assumption a clear advantage over any of the others in terms of noise response.



(a) Roll angle



(b) Pitch angle



(c) Yaw angle

Figure 55: Distributions for residual signal from Angle-Axis parameterization; Noise tests.

5.3.3 Integration Tests

Code port - Matlab to Simulink As seen in the previous chapter, the SLUGS complementary filter for attitude estimation is Simulink-based. Therefore, the Matlab-based test scripts from the unit testing framework needed to be ported to the Simulink environment for integration tests. Due to the inherent differences behind how the two environments work, this code revision inevitably introduced minor differences to the initial program. Nevertheless, by measuring the magnitude of this change, a fair assessment could be made regarding whether or not the Simulink program faithfully reproduced the behavior of the original Matlab program.

After the code port, a standard battery of simulated, sinusoidal inputs with added noise (20dB SNR) was fed to both the Simulink code as well as the Matlab code and differences in the output were calculated. These measurements, as depicted in Table 7, illustrate that typical mean errors (in degrees) between the outputs of the two programs were on the order of $1e^{-3}$, indicating a close fit.

Parameterization	**Mean (deg)**	**Std (deg)**
Direction Cosine Matrix	2.1E-03	4.2E-05
Quaternion	2.1E-03	4.2E-05
Euler angles	-2.7E-03	3.1E-05
Angle-axis	5.4E-03	1.3E-04
All Parameterizations	**1.7E-03**	**6.0E-05**

Table 7: Matlab to Simulink port errors (degrees)

Complementary Filter Results Results for all methods are tabulated in Fig. 56. In the summary table, Table 8, one can see the best and worst overall performers in terms of mean error and standard deviation, when compared to the output of the MIDG II sensor. The quaternion-based matrix exponential method, using the time-averaged OmegaBar angular velocities, exhibited the lowest overall mean errors. Furthermore, this method resulted in a fairly low standard deviation. There were several other methods with even lower standard deviations, but this difference

MEAN ERROR		**(deg)**
Roll		
Quaternions (best)	Fwd. Euler Trap	0.2178
Angle-axis (worst)	Fwd. Euler	0.4695
Pitch		
Quaternions (best)	Mexp, Omg Bar	0.0006
DCM (worst)	Fwd. Euler	1.6427
Yaw (groundtrack)		
DCM (best)	Fwd. Euler	0.0262
Quaternions (worst)	Fwd. Euler Trap	0.3991
Sum for all Euler angles		
Quaternions (best)	Mexp, Omg Bar	0.4234
DCM (worst)	Fwd. Euler	1.9442
STANDARD DEVIATION		**(deg)**
Roll		
Euler angles (best)	Fwd. Euler	1.1631
DCM (worst)	Fwd. Euler	1.6217
Pitch		
Euler angles (best)	Fwd. Euler	1.0213
DCM (worst)	Fwd. Euler	3.2874
Yaw (groundtrack)		
DCM (best)	Fwd. Euler Trap	4.2475
Angle-axis (worst)	Fwd. Euler	4.4445
Sum for all Euler angles		
Euler angles (best)	Fwd. Euler Trap	6.5855
DCM (worst)	Fwd. Euler	9.1634

Table 8: Summary statistics for best and worst performers from integration testing. Test metric: Accuracy, Units: degrees.

		Error (degrees)					
		Mean	Std	Mean	Std	Mean	Std
Parameterization	Integration Method	Roll		Pitch		Yaw (COG)	
Quaternions	Fwd. Euler	2.183E-01	1.2398	-4.020E-02	1.0878	3.917E-01	3.1767
Quaternions	Fwd. Euler Trapezoidal	2.178E-01	1.2429	-3.830E-02	1.0895	3.915E-01	3.1660
Quaternions	Mexp Wertz, Const. Curr Omg	3.581E-01	1.1950	-1.600E-03	1.0388	-9.610E-02	3.1747
Quaternions	Mexp Wertz, Const. Prev Omg	3.572E-01	1.2072	3.100E-03	1.0452	-9.580E-02	3.1540
Quaternions	Mexp Wertz, Trap Omg	3.574E-01	1.1994	7.109E-04	1.0408	-9.580E-02	3.1641
Quaternions	Mexp Wertz, Omg Bar	3.573E-01	1.1995	6.464E-04	1.0409	-9.590E-02	3.1642
DCM	Fwd. Euler	-2.753E-01	1.6217	-1.643E+00	3.2874	-5.670E-02	2.9682
DCM	Fwd. Euler Trapezoidal	-2.726E-01	1.6185	-1.634E+00	3.2859	-5.700E-02	2.9585
DCM	Mexp Murray, Const. Curr Omg	4.299E-01	1.1709	-2.290E-02	1.0234	-1.044E-01	3.1753
DCM	Mexp Murray, Const. Prev Omg	4.302E-01	1.1830	-1.820E-02	1.0299	-1.040E-01	3.1546
DCM	Mexp Murray, Trap Omg	4.299E-01	1.1752	-2.090E-02	1.0256	-1.041E-01	3.1646
DCM	Mexp Murray, Omg Bar	4.299E-01	1.1752	-2.090E-02	1.0257	-1.042E-01	3.1648
Euler angles	Fwd. Euler	4.268E-01	1.1631	-2.720E-02	1.0213	-1.050E-01	3.1751
Euler angles	Fwd. Euler Trapezoidal	4.270E-01	1.1674	-2.510E-02	1.0236	-1.047E-01	3.1644
Angle-axis	Fwd. Euler	4.695E-01	1.2502	-2.000E-03	1.0870	-2.473E-01	3.2327
Angle-axis	Fwd. Euler Trapezoidal	4.693E-01	1.2535	1.300E-03	1.0899	-2.466E-01	3.2224

Figure 56: Mean and standard deviation errors in Euler angle outputs for all methods, Integration Tests (deg)

was no larger than 0.05 degrees.

In terms of standard deviation, the Euler-angle-based, forward Euler propagation method using trapezoidal-averaged angular velocities had the lowest standard deviation with respect to the MIDG IMU, and its mean error was only 0.1 degree higher than the OmegaBar matrix exponential method.

The DCM-based, forward Euler method utilizing the constant previous angular velocity assumption exhibited the highest mean errors and standard deviation across all Euler angle outputs.

The per-angle time history plots for the roll/pitch/COG angular output and error signals of all 3 methods listed above are depicted in Figs. 57 - 59. Notice in the time history of the DCM-based plot for pitch angle (bottom of Fig.58), the significant signal difference with respect to the MIDG II (visible through more red signal near the 2nd half of the test period), whereas the Euler and quaternion approaches (top two figure pairs) followed the MIDG II signal more closely (more green signal overlaying the red).

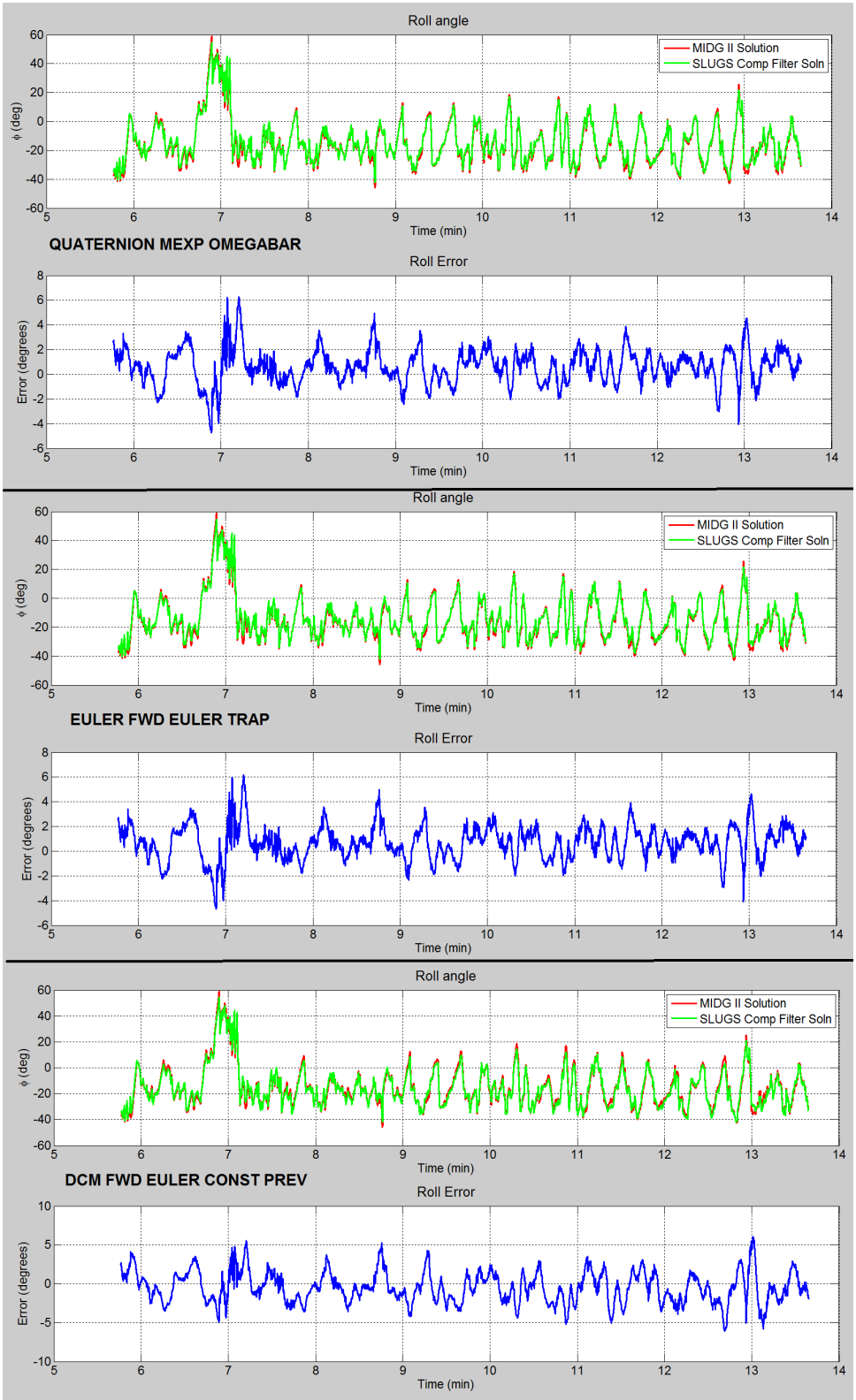


Figure 57: Time History of Roll Angle outputs and Errors with respect to the MIDG II reference.

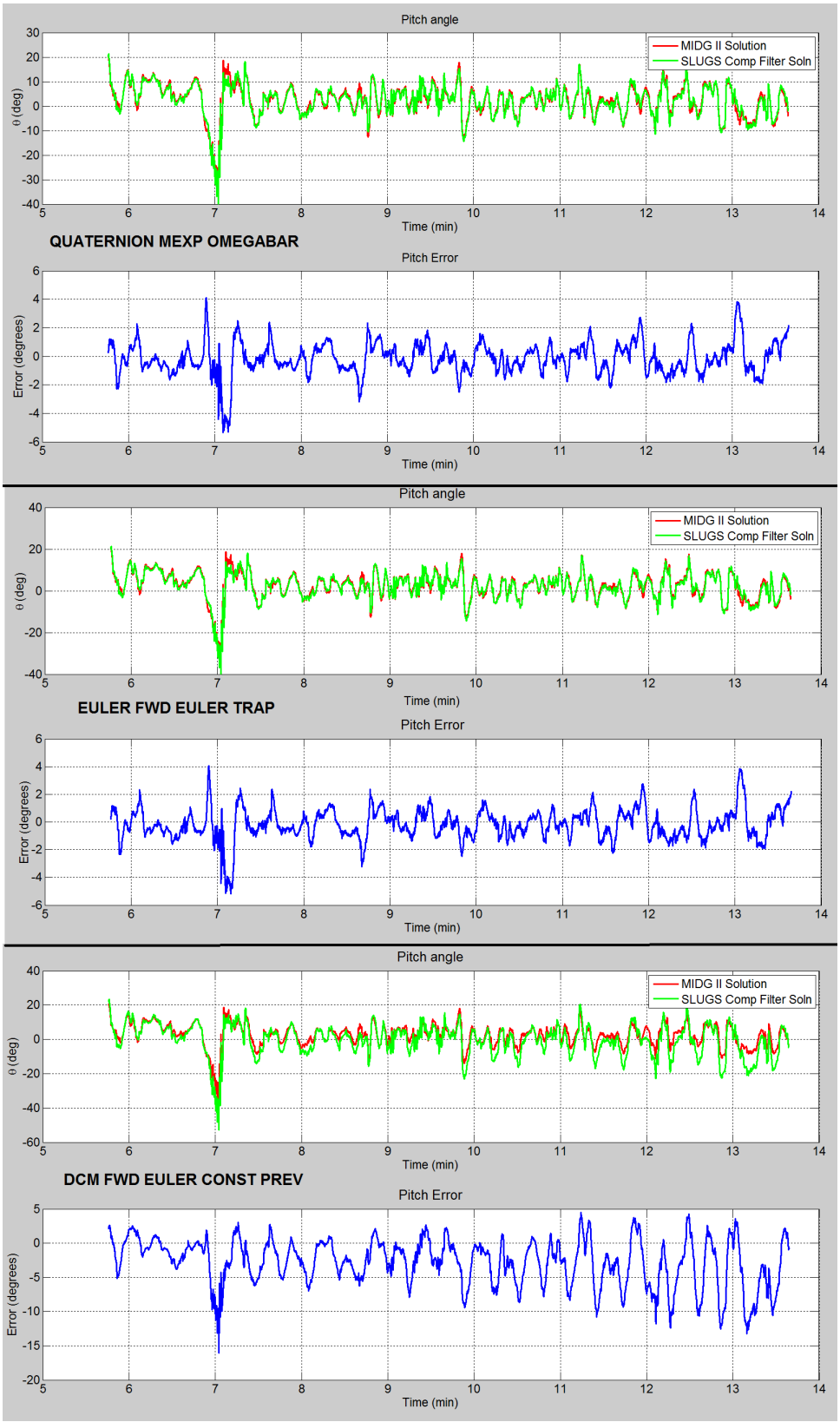


Figure 58: Time History of Pitch Angle outputs and Errors with respect to the MIDG II reference.

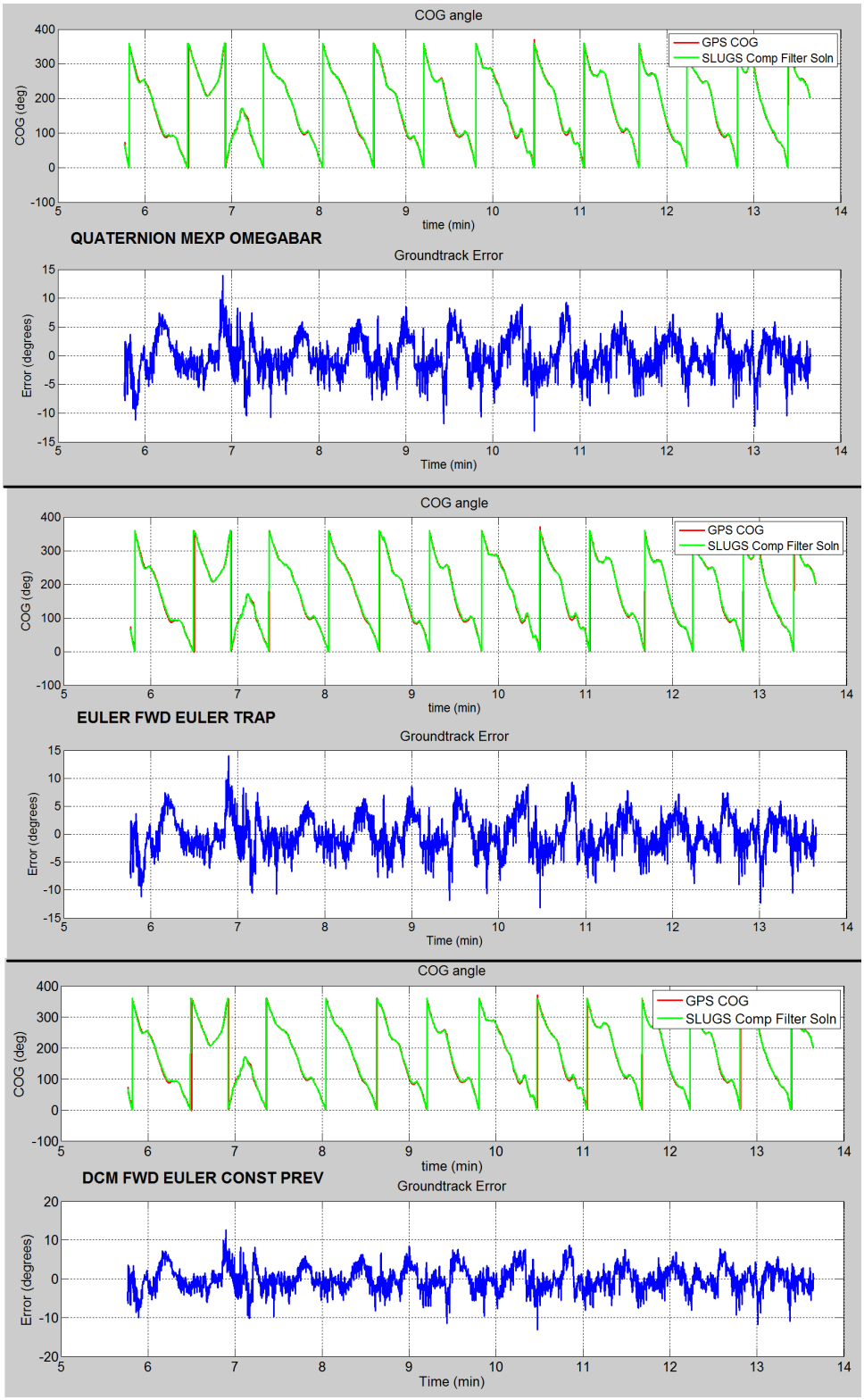


Figure 59: Time History of Yaw (Groundtrack) Angle outputs and Errors with respect to the MIDG II reference.

A few implementation details deserve mention regarding the groundtrack error plots in Fig. 59. Notice in the COG angle plots for this figure (the green and red plots above the blue error plots) the nearly vertical lines indicating rollover from 0 to 360 degrees and vice versa. Due to lack of synchronization between the SLUGS and MIDG systems in terms of when each system performed this rollover, the estimated error demonstrated large spikes during each of these rollover events. The nature of the error was essentially the order of subtraction for the error signal (SLUGS - MIDG) during the rollover. Groundtrack errors larger than 180 degrees indicated an incorrect order of the operands within the subtraction operation. For example, if the SLUGS algorithm estimated a groundtrack of 359 degrees and the MIDG estimated 3 degrees, the correct value of the error should be -4 degrees, not 356 degrees. According to these observations, a post-processing algorithm corrected these errors to produce the plots shown.

5.3.4 Computational Load

The computational load of each parameterization and integration method was evaluated using Matlab's profiler within the unit-testing framework described previously. Each algorithm was executed for five runs of one thousand calls each. The total execution time was measured by recording the sum of the time required by the main, parent function and its children. Profile times did not include the time required for memory allocation.

Fig. 60 provides a view of how all the methods compare with one another. Notice that all methods execute within the same order of magnitude, $1e^{-5}$s, of one another. Still, there is a measurable difference, a factor of 3, between the best performer and the worst performer of the group. Thus, some computational time savings can be realized depending upon on the method chosen.

Regarding the individual propagation methods, notice that the forward Euler methods are frequently faster than matrix exponential methods. This result is expected, since matrix exponential methods typically involve several calls to transcendental, trigonometric functions, whereas forward Euler methods do not. Also, the more calculations which were performed to more thoroughly track the changes in angular velocity, the more time was required. The trapezoidal-averaged ("Avg") versions of the different routines tended to perform a bit more slowly than the constant angular velocity versions of the same method. Also, notice that the $\overline{\Omega}$ (Mexp Bar) approach employed by the DCM and quaternion schemes required more processing time than the trapezoidal averaged or constant angular velocity methods.

In terms of parameterizations, observe that quaternion-based methods tend to dominate the others in terms of speed of execution, possibly due to the absence of transcendental operations required for kinematics propagation. At the other end of the spectrum, the angle-axis parameterization as well as the Euler angles required the most time.

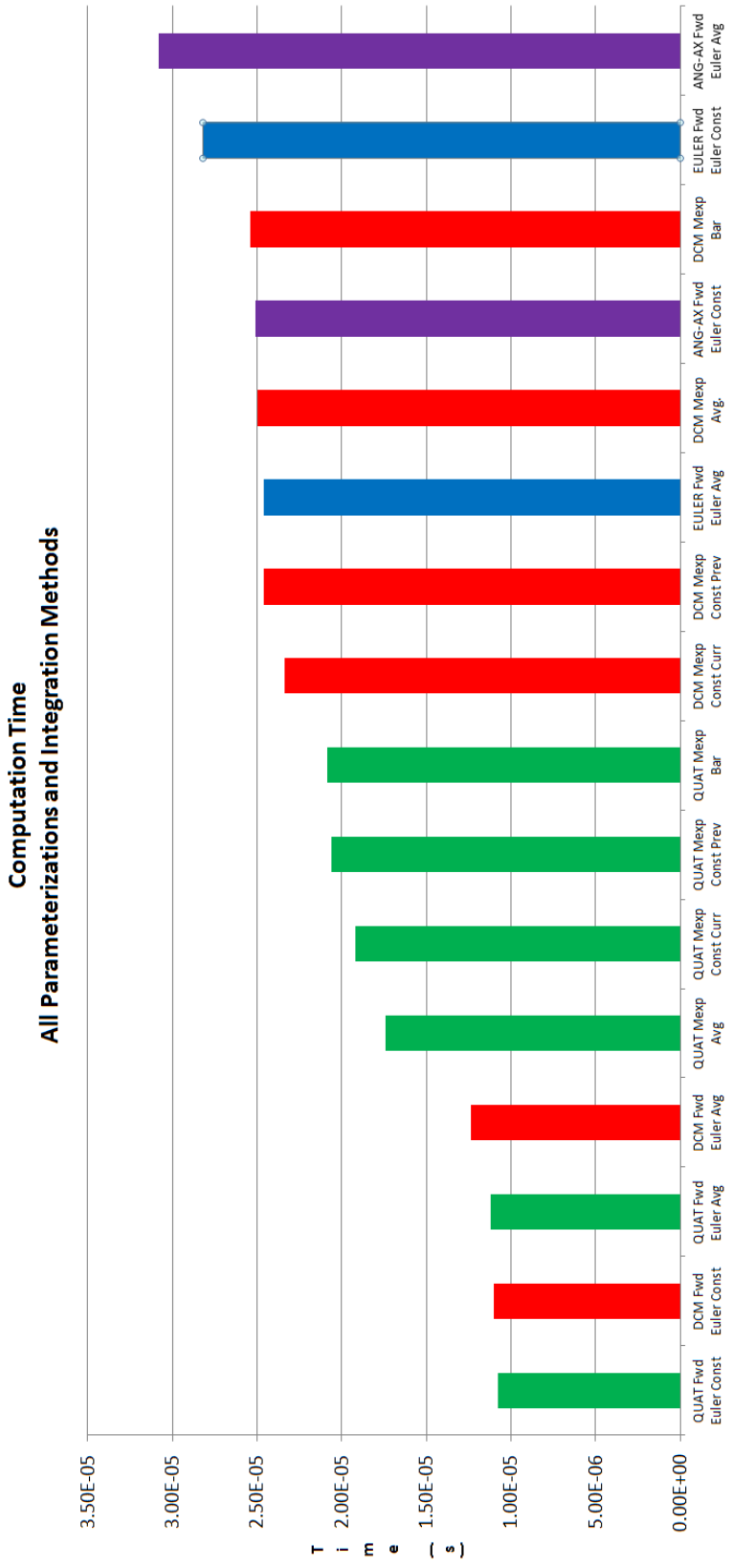


Figure 60: Computational Load Evaluation.
Color Key: Quat = green, DCM = red, Euler = blue, Angle-Axis = purple.

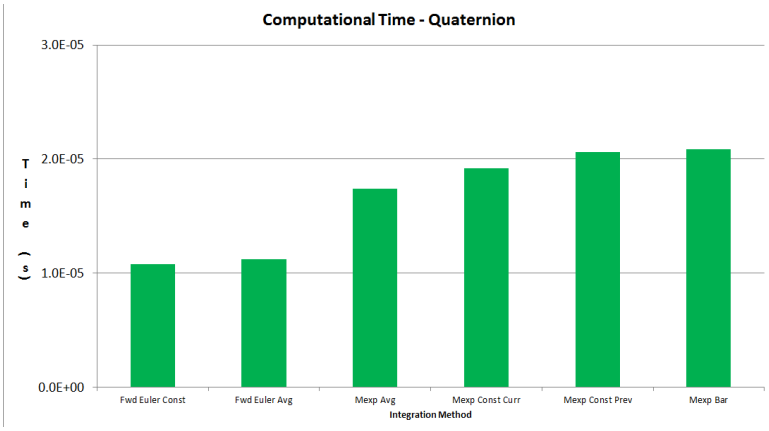


Figure 61: Quaternion-based methods

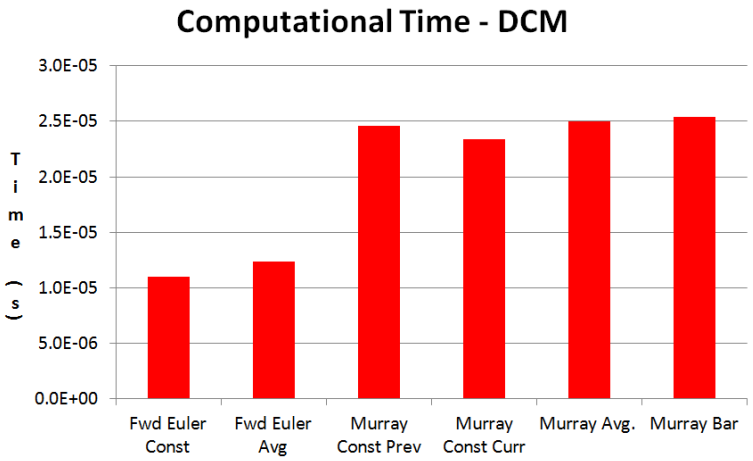


Figure 62: DCM-based methods

The following discussion centers on the per-parameterization integration schemes. Results of Fig. 61, focusing on quaternions, are straightforward. Forward Euler methods take the least time, with the constant omega or averaged omega methods being comparable. A slight performance penalty results from using the matrix exponential. The choice of angular velocity assumption resulted in roughly the same time for execution, with the exception that the trapezoidal averaged approach seemed to nudge out even the constant methods. This difference is likely insignificant; the equality of the approaches could be more visible with more rigorous profiling or other evaluation methods.

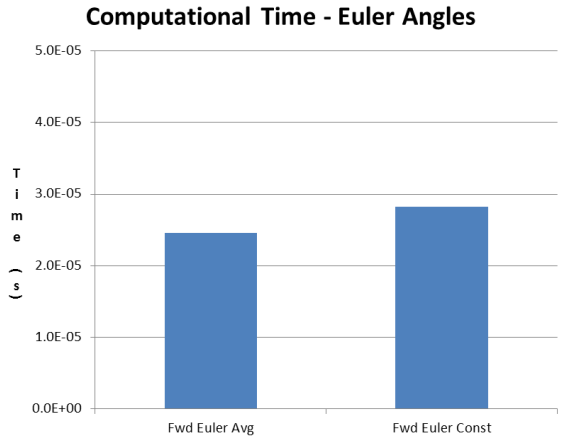


Figure 63: Euler angle-based methods

Fig. 62 shows the best performers within the DCM approach, quite similar to the quaternion setup. The notation **Murray ***, where ***** represents the angular velocity technique used, represents the matrix exponential method presented by R. Murray in [49].

Though the Euler angle and angle-axis parameterizations (Figs. 63 and 64) both employed the speedier forward Euler integration method, their complicated kinematic equations significantly lengthened run-time execution times. Both approaches involved heavy use of trigonometric functions, as seen in the Sec. 2.3. For the angle-axis parameterizations, two separate kinematic formulas had to be propagated for every integration step: 1) the change in the angle and 2) the change in the axis. Furthermore, this representation method required renormalization of the 3-element, axis parameter at each time-step, consuming an additional $1.14e^{-6}$s per execution. Without this renormalization, attitude estimates using this representation scheme would diverge to the point of singularity after only a few minutes (< 7) of run-time using real flight data.

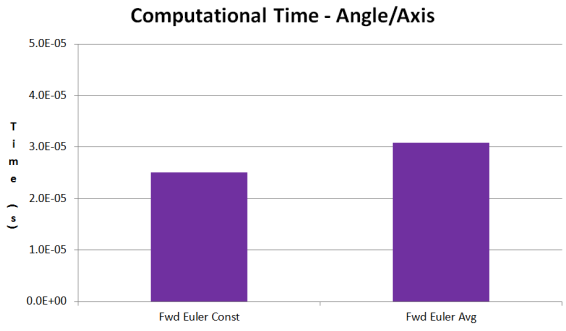


Figure 64: Angle-axis-based methods

Given the above analysis of computational load, the next consideration is how to leverage the advantages of the various methods presented within the framework of the complementary filter in the SLUGS AP. The following analysis will be mostly high-level.

First, consider the computations performed by the existing complementary filter. There are two parameterization conversions: 1) from quaternions to the DCM, and 2) from the DCM to Euler angles. There is a quaternion-based, forward Euler integrator. Immediately after, there is a quaternion normalization routine. There are two cross product multiplications and two more matrix multiplications.

In terms of possible improvements to this algorithm, consider two key approaches: 1) using the quaternion-based matrix exponential (constant current) for integration and then using the quaternion representation for all subsequent reference frame conversions and other multiplications, and 2) using the DCM-based matrix exponential (constant current) for integration and, as with the current filter, using the DCM for reference frame conversions and other multiplications.

For either of the two improvements, there would be some computational savings from the fact that neither the attitude quaternion or DCM would need to be normalized or reorthonormalized. This advantage stems from the use of the matrix exponential method vs. the forward Euler method for attitude propagation. This likely arises from the fact that the forward Euler approximation truncates several terms in the Taylor series expansion in Eq. 42 whereas the matrix exponential includes all terms. The computational savings earned depends on the parameterization. For the quaternions, a simple normalization routine is to simply divide the quaternion by its norm, as follows:

$$\vec{q}_{norm} = \frac{\vec{q}}{\sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2}}. \tag{77}$$

This process requires 4 multiplications, 4 divisions, 3 additions, and 1 square-root

operation. Ignoring the adds, this amounts to 10 floating point operations (counting the square-root as 2 multiplies). For the DCM, a simple approach is presented by W. Premerlani in [59]. Premerlani's routine requires 30 multiplications, ignoring the add and subtract operations, and thus 30 floating point operations.

The SLUGS attitude filter uses matrix multiplication four times: twice in cross products and twice in reference frame conversions. Assuming these operations take the same number of multiplications, then using quaternions involves 15 multiply operations for a total of 60 floating point operations, while a matrix multiplication using the DCM requires 9 multiply operations for a total of 36 floating point operations.

Using the DCM or quaternions throughout would eliminate the quaternion-to-DCM conversion at the end of the current propagation step. This results in a savings of 36 multiplications and thus floating point ops. There would still be the conversion to Euler angles step. For quaternions, this procedure requires 3 calls to inverse trigonometric functions and 20 multiplications - an expensive operation. For the DCM, the conversion costs 3 calls to inverse trigonometric functions - less expensive.

Table 9 summarizes these results in a list of actions vs. associated costs.
The first two sub-tables within this table show that little savings (2 floating point operations) would result from converting the existing SLUGS complementary filter to quaternions-only using the matrix exponential. In fact, the relative cost incurred by the execution of the matrix exponential (vs. the forward Euler, see Fig. 60) outweighs the savings of 2 floating point operations associated with abandoning the forward Euler integration. Thus, converting to quaternion-only representation would result in additional computational expense.

Conversion to the DCM-only makes more sense, based solely on inspection of Table 9. This table shows that using just the DCM halves the number (36 vs. 82) of floating point operations performed in either of the other two approaches using

SLUGS Current (Fwd Euler)

Action	**Cost**
Normalization	10 float ops
Convert to DCM	36 float ops
Matrix mults	36 float ops
Convert to Euler	3 inverse trig fcts
TOTAL	**82 float ops, 3 inverse trig**

Quaternions Only (Matrix Exponential)

Action	**Cost**
Normalization	0 ops
Convert to DCM	0 ops
Matrix mults	60 float ops
Convert to Euler	3 inverse trig fcts, 20 float ops
TOTAL	**80 float ops, 3 inverse trig**

DCM Only (Matrix Exponential)

Action	**Cost**
Normalization	0 ops
Convert to DCM	0 ops
Matrix mults	36 float ops
Convert to Euler	3 inverse trig fcts
TOTAL	**36 float ops, 3 inverse trig**

Table 9: 3 Scenarios - A Computational Load Comparison of Parameterization and Propagation Methods within the SLUGS Complementary Filter

quaternions - a clear savings. However, taking into account Fig. 60 as well, the advantages are not so pronounced, if they exist at all. In this figure, one can see that the quaternion-based forward Euler method is the fastest to execute, requiring only $1e^{-5}$s on average, while the DCM-based matrix exponential takes over twice as long on average, $2.3e^{-5}$s. Thus the comparison boils down to which is larger: the time to execute 46 floating point operations (savings associated with the DCM matrix exponential) or $1.3e^{-5}$s (savings associated with quaternion forward Euler). A small timing script in Matlab demonstrated that 46 floating point operations (multiplying two doubles together) can take roughly $8e^{-7}$s, a significantly shorter period than $1.3e^{-5}$s. Thus there are likely few computational savings associated with converting the existing SLUGS complementary filter to a DCM-only representation using the matrix exponential for propagation.

5.4 Conclusion

This chapter's experimental analysis has evaluated the roles that 3 key variables play with respect to attitude propagation:

- Propagation method
- Parameterization scheme
- Angular velocity assumption

The effects of these variables was analyzed under the influence of simulated, sinusoidal inputs as well as real-world flight data from a typical, low-cost sensor suite used in UAV applications, and according to the metrics of accuracy, computational efficiency, and noise response.

The propagation method of the matrix exponential offered higher accuracy than the forward Euler method, while the forward Euler method was consistently more computationally efficient. In terms of noise response, neither method offered a decided advantage over the other, with results being similar.

In terms of parameterizations, the quaternion approach consistently performed better than the other schemes across all three variables. The DCM approach was a close second in terms of computational load for the tests in this experiment. It is conceivable that through a more comprehensive or theoretical analysis, such as through the tools provided by the $O(n), \Omega(n), \Theta(n)$ notations, slightly different evaluations of computational efficiency would be observed.

In terms of the angular velocity assumption, a scheme's accuracy and computational efficiency appear to be inversely proportional. The more computationally efficient the method, the less accurate it tended to be, and vice versa.

For even the worst case performers under the computational load tests (Fig. 60), the actual impact this load imparts on the sensor DSC is negligible. Consider that the order of magnitude of the worst performing algorithm was $1e^{-5}$s. The SLUGS AP performs attitude estimation at 100Hz, or every 0.01s or $1e^{-2}$s. Sec. 4.3 indicated that the CPU load of the sensor DSC running the attitude filter was 33%. To add $2e^{-5}s$ to a slice from the $1e^{-2}$s needed by this already under-utilized processor would result in less than 2% addition to the total load. Thus, it makes sense to shift the focus in terms of performance metrics away from computational load and closer to accuracy and noise response. Since the unit tests for noise response were inconclusive, accuracy then becomes the sole criteria for incorporating a particular method. For the unit test framework, either the DCM- or quaternion-based matrix exponential with trapezoidal angular velocity offered the best performance. For the complementary filter framework, again this method stood out, with the OmegaBar approach just a tiny amount better in terms of overall error. Given that the SLUGS autopilot already uses the DCM in its various matrix multiplications and cross products for reference frame conversions and the like, then it makes the most sense to move away from the existing quaternion-based, forward Euler approach and move to a DCM-based, matrix exponential approach incorporating a trapezoidal-averaged, angular velocity assumption. This method offers excellent accuracy with minimal

computational impact.

6 Conclusions and Future Work

6.1 Conclusions

This thesis has made significant contributions to the study of various attitude propagation and parameterization methods. Primarily a work in control algorithms, the comparisons within were designed to provide quantitative information for attitude system designers working in the domain of low-cost UAVs, especially fixed wing.

The motivation section of the last chapter proposed that certain algorithms might lend themselves better to certain mission parameters than others. As a result of the experiments and analysis performed, this appears to be the case. For applications demanding higher accuracy with less regard for computational cost, the quaternion-based matrix exponential propagation method utilizing the time-averaged angular velocity assumption offers an excellent fit, given its performance within the SLUGS complementary filter. If computational cost is more of an issue, then the quaternion-based matrix exponential method with the trapezoidal averaged velocity assumption offers the next best fit, given its excellent accuracy characteristics per the unit tests. This method would also serve well an AHRS system to be deployed with a sensor suite suffering from high noise. For a system with severely limited computational resources, requiring minimal computational overhead, yet mid-range accuracy, the quaternion- or DCM-based forward Euler method with constant previous angular velocity assumption would be likely worth implementing on the system in question.

6.2 Future Work

Future work related to this thesis falls into three categories: 1) deepening the analysis already presented, 2) adding more test candidates to the existing set, and 3) improvements to the existing complementary filter of SLUGS.

6.2.1 Further analysis

Further analysis of the existing methods would include actions such as subjecting the algorithms in question to a host of other typical, control-system input signals within the unit test environment, such as doublets, ramps, and parabolas. The doublet-style input would challenge the routines using the constant angular velocity assumption. Also, sampling rate dependencies could be examined by testing the existing algorithms under 150Hz rates, 50Hz, 20Hz, 10Hz, etc. and then analyzing the resulting patterns.

Not all angular velocity assumptions presented herein were tested across all parameterizations: future work would test the OmegaBar approach in Euler angle and angle-axis parameterizations using forward Euler integration.

Hardware-in-the-loop testing, strapdown rotation table testing alongside a high quality IMU (such as the MIDG), and flight tests of the various algorithms are planned tasks after this thesis.

This thesis approached certain aspects of the analysis as if the pitch, roll, and yaw errors were independent of one another, and this likely holds true in the case of benign, mostly level flight. But in the case of more aggressive, acrobatic maneuvers, it might make more sense to evaluate these errors as if they were more tightly coupled rather than decoupled from the different axes. In this case, error metrics tied not to the 3 axes of the Euler angles but perhaps to the axis and angle representation of quaternions or even the eigenaxis representation might provide insightful comparison data not available from the pitch/roll/yaw analysis of output Euler angles. Those analyses would have their own error rates and would be interesting to explore further in future work.

In terms of computational load analysis, more rigorous testing of the algorithms could be employed with different metrics and test environments. Also, a worthy

task could be to evaluate the computational load with the $O(n)$, $\Theta(n)$, and $\Omega(n)$ theoretical tools from algorithm analysis. Even the counting of individual add, subtract, multiply, and divide operations of each algorithm could yield further efficiency insights, as in Phillips [53].

A fourth-order Runge-Kutta routine served as the benchmark for comparison within the unit test environment. The development of an analytic expression for truth or an even more accurate numeric approximation could prove useful. Hypothesized methods might include pre-determining the attitude by employing spherical coordinates from 3-dimensional calculus or other advanced techniques.

6.2.2 New test candidates

It would be interesting to develop an expression for the matrix exponential within the Euler angle as well as angle-axis parameterization and then evaluate the corresponding performance metrics. It is possible that the angle-axis version would bear resemblance to that of quaternions.

Due to advances of computational hardware, multi-stage methods such as 2nd order Runge-Kutta (Euler-Cauchy) or even 4th order deserve evaluation, especially with respect to computational efficiency combined with accuracy. Multi-step methods such as the predictor-correctors also hold promise.

Lastly, the experiments with the modified OmegaBar ($\overline{\Omega}$) provoked further interest in different ways of expressing the angular velocity as a function of time and how this impacts propagation performance. Simpson's rule would be a good comparison against the OmegaBar approach.

6.2.3 Complementary Filter Improvements

Improvements to the SLUGS complementary filter could include the accelerometer feedback modules and adding magnetometer feedback, according to [58] and [57].

Wind estimation would provide another useful piece of information from navigation and control; a good first approach is presented in [56]. Lastly, an excellent addition to the existing algorithms would be a dead-reckoning attitude filter for GPS outages; Conte presents a computer vision-based approach in Ref. [16].

References

[1] About aerospace coordinate systems. http://www.mathworks.com/help/aeroblks/about-aerospace-coordinate-systems.html. Accessed: 09/2012.

[2] Gyroscope. http://www.vectornav.com/support/library?id=85. Accessed: 08/2012.

[3] Uavs perform autonomous search-and-rescue operations. http://www.homelandsecuritynewswire.com/uavs-perform-autonomous-search-and-rescue-operations, July 2010. Accessed: 08/2012.

[4] U.s. joint planning and development office nextgen unmanned aircraft systems r&d roadmap. http://publicintelligence.net/jpdo-nextgen-uas-rd-roadmap/, May 2012. Accessed: 08/2012.

[5] AMSA. Gnss vulnerability workshop. http://www.amsa.gov.au/about_amsa/corporate_information/Recent_Events/2011/March-GNSS.asp, 2011. Accessed: 12/2012.

[6] G. Baldwin, R. Mahony, J. Trumpf, T. Hamel, and T. Cheviron. Complementary filter design on the special euclidean group se(3). *European Control Conference*, 2007.

[7] S. Boyd. Lecture notes for ee263. Introduction to Linear Dynamical Systems, 2008.

[8] P. Brisset, A. Drouin, M. Gorraz, P. S. Huard, and J. Tyler. The paparazzi solution. *Proc. of MAV2006*, 2006.

[9] R. Bronson. *Schaum's Outlines: Differential Equations*. McGraw-Hill, second edition, 1986.

[10] J.C. Butcher. *The Numerical Analysis of Ordinary Differential Equations: Runge-Kutta and General Linear Methods*. John Wiley & Sons, Chichester, Great Britain, 1987.

[11] R. Casey and B. Watanabe. Cruzcontrol. http://byron.soe.ucsc.edu/wiki/cruzcontrol, 2011.

[12] J. Cassano. Antarctic research project: Unmanned aerial vehicles - studying air and sea interactions at terra nova bay. http://icestories.exploratorium.edu/dispatches/antarctic-projects/unmanned-aerial-vehicles/, 2009. Accessed: 08/2012.

[13] T. Cheviron, T. Hamel, R. Mahony, and G. Baldwin. Robust nonlinear fusion of inertial and visual data for position, velocity and attitude estimation of uav. In *ICRA*, pages 2010–2016. IEEE, 2007.

[14] J. Clapper, J. Young, J. Cartwright, and J. Grimes. Unmanned systems roadmap: 2007-2032. http://www.fas.org/irp/program/collect/usroadmap2007.pdf, 2007. Accessed: 08/2012.

[15] M. Clark. Operation practice: Search and rescue, uavs team up. http://www.suasnews.com/2012/03/13949/operation-practice-search-and-rescue-uavs-team-up, March 2012. Accessed: 08/2012.

[16] G. Conte and P. Doherty. An integrated uav navigation system based on aerial image matching. In *Aerospace Conference, 2008 IEEE*, pages 1–10. IEEE, 2008.

[17] Teal Group Corporation. World unmanned aerial vehicle systems study. http://tealgroup.com/index.php?option=com_rokecwid&view=ecwid&Itemid=100020#ecwid:category=2338224&mode=product&product=10081868, 2012. Accessed: 07/2012.

[18] J. Craig. *Introduction to Robotics - Mechanics and Control*. Pearson Prentice Hall, Upper Saddle River, New Jersey, 3rd edition, 2005.

[19] J.L. Crassidis, F.L. Markley, and Y. Cheng. A survey of nonlinear attitude estimation methods. *Journal of guidance control and dynamics*, 30(1):12, 2007.

[20] R. Curry, M. Lizarraga, and G. Elkaim, editors. *The Design of Rapidly Reconfigurable Filters for Attitude and Position Determination.* Proceedings of the AIAA Infotech@Aerospace 2010, 2010.

[21] R.E Curry. Personal communication. Personal Communication, August-December 2012.

[22] Istituto Nazionale di Geofisica e Vulcanologia. Elements of the geomagnetic field. `http://roma2.rm.ingv.it/en/aree_di_ricerca/1/campo_magnetico_terrestre/8/elementi_del_campo_magnetico`. Accessed: 12/2012.

[23] J. Diebel. Representing attitude: Euler angles, unit quaternions, and rotation vectors. *Matrix*, October 2006.

[24] V. Dobrokhodov. Personal communication. Personal Communication, September 2012.

[25] G. Elkaim. Autonomous systems lab. `http://asl.soe.ucsc.edu/`. Accessed: 12/2012.

[26] G. Elkaim. Attitude estimation, screencast tutorial. 2011.

[27] M. Euston, P. Coote, R. Mahony, J. Kim, and T. Hamel. A complementary filter for attitude estimation of a fixed-wing uav. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 340–345. IEEE, 2008.

[28] V. R. Fish and B.G. Chmielewski. *Flight Program Requirements Document for the High Energy Astronomy Observatory-B Attitude Control and Determination Subsystem (FPH-B), Doc. No. DO1137B.* NASA TRW Systems Group, April 1977.

[29] V. Gavrilets, A. Shterenberg, MA Dahleh, and E. Feron. Avionics system for a small unmanned helicopter performing aggressive maneuvers. In *Digital*

Avionics Systems Conference, 2000. Proceedings. DASC. The 19th, volume 1, pages 1E2–1. IEEE, 2000.

[30] D. Gebre-Egziabher. *Design and Performance Analysis of a Low-Cost Aided Dead Reckoning Navigator*. PhD thesis, Stanford University, 2004.

[31] Physics Georgia State University and Astronomy Dept. Magnetic field of the earth. `http://hyperphysics.phy-astr.gsu.edu/hbase/magnetic/magearth.html`. Accessed: 12/2012.

[32] S. Gleason and D. Gebre-Egziabher. *Gnss Applications and Methods*. GNSS technology and applications series. Artech House, 2009.

[33] G. Golub and J. Ortega. *Scientific Computing and Differential Equations: An Introduction to Numerical Methods*. Academic Press, Inc.; Harcourt Brace Jovanovich, San Diego, 1992.

[34] C. Hall. Spacecraft attitude dynamics and control, 2003. Notes from AOE 4140 Course, Spacecraft Dynamics and Control, Virginia Tech.

[35] W. Jr. Higgins. A comparison of complementary and kalman filtering. *IEEE Transactions on Aerospace and Electronic Systems*, 1975.

[36] I. Klotz. Hurricane-hunting drone joins nasa fleet. `http://news.discovery.com/earth/nasa-uav-hurricane-formation.html`, 2010. Accessed: 08/2012.

[37] J. Kuang and S. Tan. Gps-based attitude determination of gyrostat satellite by quaternion estimation algorithms. *Acta Astronautica*, 51(11):743–759, 2002.

[38] J. B. Kuipers. *Quaternions and Rotation Sequences*. Princeton University Press, New Jersey, 1999.

[39] A. Lawrence. *Modern Inertial Technology*. Springer, New York, 2nd edition, 1998.

[40] M. Lizarraga, R. Curry, and G. Elkaim. Reprogrammable uav autopilot system (part 1) - system hardware and software. *Circuit Cellar*, (249), April 2011.

[41] M. Lizarraga, R. Curry, and G. Elkaim. Reprogrammable uav autopilot system (part 2) - testing and results. *Circuit Cellar*, (250), May 2011.

[42] M. Lizarraga, G. Elkaim, and R. Curry. Slugs uav: A flexible and versatile hardware/software platform for guidance navigation and control research. In *Airborne Experimental Test Platforms: From Theory to Flight*, Washington D.C., June 2013. American Control Conference.

[43] N. Lovren and J.K. Pieper. Error analysis of direction cosines and quaternion parameters techniques for aircraft attitude determination. *Aerospace and Electronic Systems, IEEE Transactions on*, 34(3):983 –989, Jul 1998.

[44] R. Mahony, T. Hamel, and J-M Pflimlin. Nonlinear complementary filters on the special orthogonal group. *IEEE Transactions on Automatic Control*, 53(5), June 2008.

[45] F.L. Markley. Attitude determination using vector observations and the singular value decomposition. *Journal of the Astronautical Sciences*, 36(3):245–258, July-September 1988.

[46] J. Mathews. *NUMERICAL METHODS for Mathematics, Science and Engineering*. Prentice Hall, Englewood Cliffs, New Jersey, 2nd edition, 1995.

[47] J. McEnnan. Onboard orbit propagation using quaternions, 2000.

[48] Inc. Microbotics. Midg series ins/gps. `http://www.microboticsinc.com/MIDG_INS-GPS.htm`, 2012. Accessed: 12/2012.

[49] R. Murray, Z. Li, and S. S. Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[50] Dept. of Mechanical and Naval Postgraduate School Aerospace Engineering. Mae research centers. `http://www.nps.edu/Academics/GSEAS/MAE/Research.asp`, 2012.

[51] J. Park and W-K Chung. Geometric integration on euclidean group with application to articulated multibody systems. *IEEE Transactions on Robotics*, 21, 2005.

[52] B. Parkinson, J.J. Spilker, and G. Elkaim. *Global Navigation Satellite System*. Wiley and Sons, 2003.

[53] W.F. Phillips, C.E. Hailey, and G.A. Gebert. Review of attitude representations used for aircraft kinematics. *Journal of aircraft*, 38(4):718–737, 2001.

[54] Joint Planning and Development Office. Nextgen uas research, development and demonstration roadmap. `http://info.publicintelligence.net/JPDO-NextGenUAS.pdf`, March 2012.

[55] W. Premerlani. Gentlenav. `http://code.google.com/p/gentlenav/wiki/MatrixPilot`. Accessed: 07/2012.

[56] W. Premerlani. Notes on wind estimation. `http://code.google.com/p/gentlenav/downloads/list`, Dec 2009.

[57] W. Premerlani. Magnetometer realignment: Theory and implementation. `http://code.google.com/p/gentlenav/downloads/list`, October 2011.

[58] W. Premerlani. Roll-pitch gyro drift compensation. `http://code.google.com/p/gentlenav/downloads/list`, May 2012.

[59] W. Premerlani and P. Bizard. Direction cosine matrix imu: Theory. `http://code.google.com/p/gentlenav/downloads/list`, 2009.

[60] M. Rockwell. Police chiefs adopt privacy guidelines for unmanned aircraft systems. `http://www.gsnmagazine.com/node/27022?c=law_enforcement_first_responders`, August 2012. Accessed: 08/2012.

[61] S. Ronnback. *Developement of a INS/GPS navigation loop for an UAV*. PhD thesis, Lulea Tekniska Universitet, Universitetsomrdet, Porsn; Lule, Sweden, February 2000.

[62] P. Savage. What do accelerometers measure? http://www.strapdownassociates.com/Accels%20Measure.pdf, May 2005. Accessed: 11/2012.

[63] M. Shuster. A survey of attitude representations. *The Journal of the Astronautical Sciences*, 41(4):439–517, 1993.

[64] M.D. Shuster. The generalized wahba problem. *Journal of the Astronautical Sciences*, 54(2):245, 2006.

[65] G. Strang. *Introduction to Linear Algebra, Fourth Edition*. Wellesley-Cambridge Press, 2009.

[66] J. Strickland. What is a gimbal – and what does it have to do with nasa? http://science.howstuffworks.com/gimbal1.htm. Accessed: 11/2012.

[67] N. Trawny and S.I. Roumeliotis. Indirect kalman filter for 3d attitude estimation. *University of Minnesota, Dept. of Comp. Sci. & Eng., Tech. Rep*, 2, 2005.

[68] J. Vasconcelos. *Nonlinear Navigation System Design with Application to Autonomous Vehicles*. PhD thesis, Universidade Técnica de Lisboa, Instituto Superior Técnico, 2010.

[69] M. Veth, F. Anderson, F. Webber, and M. Nielsen. Tightly-coupled ins, gps, and imaging sensors for precision geolocation. In *Proceedings of the 2008 National Technical Meeting of The Institute of Navigation*, pages 487–496, January 2008.

[70] K. Wagstaff. Congress paves way for unmanned drones in u.s. commercial airspace. http://techland.time.com/2012/02/08/congress-paves-way-for-unmanned-drones-in-u-s-commercial-airspace/, February 2012. Accessed: 08/2012.

[71] G. Wahba. Problem 65-1: A least squares estimate of satellite attitude. *SIAM Review*, 7:409, 1965.

[72] D. Weatherington. Unmanned aircraft systems. Technical report, Office of the Under Secretary of Defense for Acquisition, Technology, and Logistics, April 2010. Accessed: 09/2012.

[73] J.R. Wertz. *Spacecraft Attitude Determination and Control*. D. Reidel Publishing Company, 1978.

[74] J. Westerink. Lecture 11 - numerical differentiation, 2004. Notes from CE 341/441, Fall 2004, Notre Dame.

[75] S. Whitmore. Closed-form integrator for the quaternion (euler angle) kinematics equations, 2000.

[76] Wikipedia. Specific force. `http://en.wikipedia.org/wiki/Specific_force`, Oct 2012. Accessed: 12/2012.

[77] P. H. Zipfel. *Modeling and Simulation of Aerospace Vehicle Dynamics*. AIAA, Gainesville, Florida, 2000.