12-2021

Sensitivity Study for UAV GPS-Denied Navigation in Uncertain Landmark Fields

Samantha D. Burton
Utah State University

SENSITIVITY STUDY FOR UAV GPS-DENIED NAVIGATION IN UNCERTAIN LANDMARK FIELDS

by

Samantha D. Burton

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

of

MASTER OF SCIENCE

in

Mechanical Engineering

Approved:

Randall Christensen, Ph.D. Major Professor	Douglas Hunsaker, Ph.D. Committee Member
Tianyi He, Ph.D. Committee Member	D. Richard Cutler, Ph.D. Interim Vice Provost of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2021

ABSTRACT

Sensitivity Study for UAV GPS-Denied Navigation in Uncertain Landmark Fields

by

Samantha D. Burton, Master of Science

Utah State University, 2021

Major Professor: Randall Christensen, Ph.D.
Department: Mechanical and Aerospace Engineering

This document provides two 2D simulation sensitivity analyses regarding UAV navigation errors within a GPS-denied region. The research focuses on a development and investigation of visual-SLAM in a GPS-denied landmark field, where landmark locations are known a priori but with an associated uncertainty and where absolute position information is limited within the trajectory. Furthermore, there is development and investigation of utilizing multiple-shooting optimal control with visual-SLAM using potential error-reducing cost parameters. Objectives are to quantitatively understand the UAV's sensitivity of position errors to sensor grade and landmark characteristics as well as sensitivity of position errors to guidance and control cost gains.

(199 pages)

PUBLIC ABSTRACT

Sensitivity Study for UAV GPS-Denied Navigation in Uncertain Landmark Fields

Samantha D. Burton

This document provides two 2D simulation sensitivity analyses regarding a drone's flight characteristic (state) errors within a GPS-denied region. The research focuses on a development and investigation of utilizing a camera to simultaneously determine a drone's state while locating landmarks, where there is uncertainty in the landmarks' exact positions prior to the mission (SLAM). This SLAM method is performed in regions with limited access to GPS. Furthermore, there is development and investigation of controlling the drone in conjunction with SLAM using potential error-reducing control parameters. Objectives are to quantitatively understand the UAV's sensitivity of position errors to sensor grade and landmark characteristics as well as sensitivity of position errors to tuned control parameters.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Modern military development is investing research in unmanned aerial vehicle (UAV, drone) missions of reconnaissance, intelligence, surveillance, and payload delivery in hostile regions. In addition to remotely piloting these vehicles, research is pursuing fully autonomous vehicles. To complete missions within hostile territories, these autonomous vehicles must be able to navigate through obstacle- or landmark-filled regions, path plan, and perform collaborative mission planning—all in real time. A large hindrance to executing these capabilities is in GPS-denied (GPSD) territories, where obstructions, such as buildings or mountain valleys, and enemy interference, such as GPS jamming, prevent the UAV from obtaining precise state measurements of itself, allies, and nearby landmarks. Without these measurements, the UAV's movement is hindered, leaving the UAV susceptible to collisions, enemy capture, and mission failure.

To combat this hinderance, research has developed inertial navigation system (INS) sensors and filter statistics algorithms to better define state estimation during a mission; incorporating mapping vision sensors, which increase estimation accuracy as well as geographically locate landmarks, is also common practice [1]. Much of the latest research involves adding new methods to UAV algorithms to enhance specific aspects of the mission. For this research, the main objective is to enhance the framework for mission planners to assess a mission's sensitivity to navigation uncertainties as well as sensitivities for UAV guidance and control (GC) in uncertain environments.

To do so, this research will focus on two objectives. First, this research will focus on the development and investigation of UAV and landmark simultaneous localization and mapping (SLAM) [2]. The research will study the sensitivities using SLAM in a landmark field, where landmark locations are known a priori but with an associated uncertainty and where availability of absolute position information is limited within the trajectory. Second, this research will focus on developing a new GC law cost (see 1.1.2 and 1.1.3) which attempts to approach GC optimization in terms of reducing navigation errors. To explain this approach, consider a linear system. Navigation uncertainties in that linear system grow exactly the same regardless of the trajectory the system takes. Considering a nonlinear system, however, navigation uncertainties are trajectory dependent; and since trajectories are dependent on GC parameters, there is an implication that GC parameters can be exploited to modulate navigation errors. To that end, the research investigates the sensitivity of navigation uncertainties as a function of GC parameters in that new GC law cost. Both the SLAM sensitivity study and the new GC law cost study have potential in identifying guidance, navigation, and control (GNC) characteristics that minimize navigation errors.

The following chapters in this thesis will provide the necessary information to understand and execute this research’s objectives. The remainder of Chapter I will detail a literature review for GNC development. Chapter II will provide design development details within the simulation. Chapter III will display simulation results and the interpretations thereof. Chapter IV will then provide the thesis conclusion.

1.1 Literature Review

This section will detail the development of GNC. First, navigation improvements will be addressed. Next a discussion on generating guidance paths will be presented. Finally, the review will provide a description on types of control law implementation.

1.1.1 Navigation

To know the state of a UAV at any time in a mission is the study of navigation, or state estimation. Accessing real-time state estimates are usually acquired via sensor packages onboard the UAV. With the advances in GPS sensors, the complexities in retaining precise and accurate navigation have reduced. However, when GPS sensor readings are denied, alternative sensors must be utilized. INS such as accelerometers and gyroscopes are the fundamental packages to continue estimating without GPS. With the assistance of a Kalman filter, UAV navigation performance in simulation and physical testing has improved significantly [1]. One can think of this filter as gaussian statistics algorithm—a step-by-step recipe procedure to account for all measurements. Based on prior knowledge of system variances and the current INS readings, the sub-algorithm will output a better approximation of the UAV's current state [3]. An alternative to Kalman filters is the Particle filter. This filter is utilized in highly non-gaussian environments, where statistical resampling and tuning weights are used to estimate aircraft state [4,5]. However, given large uncertainties and drift characteristics [6] associated with INS sensors, the effectiveness of either filter may still be limited as well as either under- or overconservative in estimate uncertainty [4,7].

To improve effectiveness, additional sensors may be incorporated with the INS and filter algorithms. In particular, vision-based (camera) measurements and light detection and ranging (LIDAR) have provided significant success to increasing navigation performance [8–10]. For example, one popular vision-based approach to GPSD navigation is visual odometry, where sequential images of the environment are compared to gauge how much UAV orientation changed from one time instant to the next

[1,2]. Another example is simultaneous localization and mapping (SLAM), where the environmental sensors, be it radar, acoustic, image, etc., creates a real-time terrain mapping; position measurements are then extracted based on changes in new image captures [1,2]. Image techniques are also used with UAV collaborative swarms using pattern-detection. By capturing images during a mission, a UAV swarm can quickly determine one another's position when the images capture specific patterns that were placed on each drone [11]. Alternatively, navigation swarm collaboration can be achieved with indirect GPS sources. Using a high-flying, GPS-allowed drone to send data to low-flying, GPSD drones and/or matching the low-flying drones imagery with previously acquired GPS images (a form of geolocation filtering), successful missions have been accomplish in both simulation and physical implementation [7,12].

Both vision-based measurements and LIDAR sensors are used heavily in the state of the art with preferences based on each sensor's particular advantage. Camera sensors have a wider sensor price range, much like INS packages. Camera payloads typically weigh less and are less bulky than LIDAR [1]. Fundamentally, however, the form of LIDAR's environmental image is of higher value. LIDAR's lasers can still map a terrain without ambient lighting [1]. LIDAR provides terrain depth measurements without the complications of stereo imaging (note, however, RGB-D cameras [8]) and is less dependent on scene "texture" and nonhomogeneous terrain [1,7]. Nevertheless, LIDAR is an active sensor, like radar, where its broadcasted waves are more susceptible to enemy detection than passive vision-based measurements [4]; and, in either case, both camera and LIDAR sensor performance degrade from noise, environmental influences, and processing computation time—especially in regions where landmark positions are only partially known [2,8,11,12].

Particularly for vision-based systems, there is discussion on how UAV state and sensor interaction with the environment influence navigation performance. The discussion is typically in the context of the source of error in the navigation. For collaborative rotor-UAVs executing surveillance, Scaramuzza et al. states that “the closer the robot is to a point in the terrain, the better its sensing ability to monitor this point” [6]. Maddison et al. provides test results showing that “error and uncertainty should be high for points that are far from the camera and near the direction of motion” based on motion stereo camera results [10]. Saske et al. provides a chart with similar results for different camera resolutions and frames per second [11]. The researchers also emphasize the fact that certain camera approaches, like visual odometry and optical flow, significantly accumulate errors and drift over time [6,8]. One study by Babinec and Aeltauer performed extensive research on how the onboard aerial surveillance imagery specifically produces position error. Its findings included the following: Sources of errors are extensive and range from human error to atmospheric turbulence to camera lens deformation. In an image, having a high number of landmarks with known positions “generally…can improve accuracy, but it may not reduce the resulting error to zero.” Average landmark position error increases with landmark distance from the camera’s principal point. Given four landmarks placed in quadrilateral patterns with known positions, error increases radially outward from this quadrilateral. If this quadrilateral is relatively compact, errors are mitigated within the quadrilateral at the expense of higher errors outside the quadrilateral. Higher surveillance errors occur at lower UAV altitudes with camera images farther offset from the center of the quadrilateral [9]. Nuske et al.

also provide insight in terms of geolocation filtering, where “error in vehicle heading is often the dominant source of error” [7]. Also considering geolocation, Schnaufer et al. explains that having landmarks with less accurate coordinates, fewer landmarks, and non-robust image cross-correlation will significantly increase errors [12].

Given the wide range of discussion on error sources in navigation, there appears to be significant qualitative knowledge for mission planners on how to minimize these errors—minimize heading errors (e.g., using high accuracy magnetometers); prior to the mission, accurately locate multiple landmarks along the mission path; minimize UAV-landmark proximity to achieve better sensing ability; etc. However, a quantitative study of navigation uncertainties is lacking.

For example, a particular GPSD payload mission may involve few landmarks, restricted budget, and a fixed-wing UAV. These constraints force mission planners to prioritize what sensor package grades to use. These constraints also force prioritization on how the UAV should travel. Without hovering ability while collecting navigation estimates, like rotorcrafts in state-of-the-art research, mission planners must prioritize how long the UAV can collect landmark information before estimation errors grow too large. In this example, the aforementioned research provides some quantitative understanding. Nevertheless, there still lacks a sensor sensitivity analysis to assist these constrained mission planners in choosing appropriate sensor grades and mission paths.

1.1.2 Guidance

As noted in the 1.1.1 Navigation section, navigation is influenced by how a UAV travels through a GPSD region. This travel, or path planning, is the subject of guidance research. In well-known, clutter-free environments, generated paths may be as simple as drawing a straight line between the UAV’s current position and the final destination. Even as the environments become more cluttered, planners can still build simple paths using a waypoint technique. In this technique, the planner need only specify a “breadcrumb trail” of points that the drone can follow up to the final destination [13]. What becomes problematic for this algorithm is dynamic feasibility. A fixed-wing UAV, for example, may not be capable of turning sharp enough to hit the next waypoint and avoid nearby obstructions. Furthermore, there is little intuition as to whether this somewhat-arbitrarily-chosen waypoint path is necessarily the wisest path for the UAV to follow, especially when the environment becomes less well known. Hence, significant research and development has been dedicated to solving these guidance problems.

Before discussing the state of the art, when considering guidance, development usually splits into two levels: global planning and local planning. Global planning is typically characterized as a pre-mission plan on how the UAV will get to a final destination or, if the final destination must be sought out, the course of travel so the UAV can look for the final destination. Local planning is typically characterized as real-time mission planning, where the UAV will perform specific avoidance maneuvers if previously unknown landmarks or intruders hinder the UAV’s movement along the global path [4]. Interestingly, despite the planning differences, the algorithms that generate global guidance paths can also generate local guidance paths and vice versa. The preferences for using an algorithm globally or locally are simply based the algorithm’s particular advantages (computation time, reliability, adaptability, etc.).

With global and local planning now described, the state of the art will be described. Building off the idea of waypoint following, Anderson et al. provide a real-time path smoothing technique at waypoint turns, where waypoint corners are replaced with dynamically feasible fillets [14]. Anderson’s co-authors, Beard and McLain, simplify this technique further and also lecture on two path planning methods: Voronoi Graphs and Rapidly Exploring Random Trees (RRT) [13]. Voronoi graphs align flight paths such that they “are perpendicular bisectors between” the landmarks [13], which aids in maximizing separation distance for obstruction avoidance. RRT algorithm will randomly consider and build feasible path routes within the environment until one of the feasible routes connects with the final destination. Alternatively, Kendoul provides further insight to potential fields, which develops algorithmic “force fields” generated by the goals and landmarks in the environment. Obstacles generate repulsive forces that repel the vehicle and goals generate attractive forces” [1]. Each of these approaches has their own drawbacks. For example, the more popular RRT planner, in its rudimentary form, is for global planning, where real time implementation, especially for intruder environments [4], become impractical [8]. Nevertheless, the state of the art tends to hybridize algorithms to compensate for such impracticalities. Such is the case with the real-time * (pronounced “star”) algorithms (A*, Theta*, D*) [8,15].

Common to many hybrid algorithms is the incorporation of a heuristic goal based on finding shorter paths to the desired destination. In other words, like potential fields, these algorithms begin to incorporate secondary objectives (goals, costs) while accomplishing the fundamental objective of reaching a destination. Most costs like finding the shortest path are quite common, such as minimizing the time-to-go, solution computation time, and vehicle energy usage as well as safety measures like maximizing obstacle distance or reducing aggressive maneuvers (increasing path smoothness) near landmarks [1,4,8,14,16–18].

Though navigation techniques significantly reduce state estimation errors, guidance techniques may also prove potentially useful in minimizing these errors. Reflecting on section 1.1.1 Navigation's discussion on how UAV interactions with the environment influence navigation performance, one begins to see unique guidance costs that would be developed to minimize navigation errors. Perez-Grau et al. [8] and Scaramuzza et al. [6] offer similar comments: SLAM techniques favor low-speed paths that revisit previously seen environments. Exercising system dynamics in the initial flight path helps generate observability. Perez-Grau further claims that generating low yaw rates and maintaining the camera's view in the flight direction reduces "blind [measurement] movement segments" and that extended travel in a given direction increases position errors in that direction. The camera analysis by Babinec and Aeltauer also implies that errors are mitigated if UAV flight path is such that landmarks are more centered in the camera's field of view (FOV).

Again, these comments mostly provide qualitative insight as to what particular path costs would minimize navigation errors. Yet, there still lacks a quantitative understanding. Many authors do at least provide position error results from their research contributions. Nevertheless, the results show little sensitivity, error-improvement analysis with regard to these navigation costs. They do not provide mission planners with intuition on what navigation costs are more important than others.

1.1.3 Control

With guidance paths developed, the method to physically force the vehicle to execute the guidance trajectory is the study of control. In many cases, the reasoning for implementing a specific type of controller is only for simplicity. Other reasons involve the need to perform aggressive, evasive maneuvers or withstand wind conditions during flight. Furthermore, other reasons are related to execution costs, much like the costs described in the 1.1.2 Guidance section.

In fact, many of the concepts developed in control are related to the concepts built in guidance. Perhaps the simplest concepts of control are carrot chasing (i.e. trajectory following) and waypoint following [13,19]. In carrot chasing, a point is generated to move along a path. The UAV is instructed to chase this point until the mission's end. In waypoint following, straight line and fillet following instructions are given for the drone to hit each waypoint. A more complex control concept involves vector fields, where control instructions are to follow the direction of vector flow that lead to the desired trajectory [19]. However, these basic, path-following control instructions are only a small piece of the underlying control laws that execute these instructions.

Control law development is extensive. Although this literature review will highlight three main control methods—PID, nonlinear, and optimal—there certainly are other novel branches of control, such a machine learning (e.g. neural networks) and fuzzy logic [1]. Of the three main methods, PID control is perhaps the most elementary. Given the error difference between current and desired flight path, PID will send commands based on proportional, integral, and derivative error calculations [20]. These commands are then given intensity (gain) adjustments based on how aggressive a UAV should pursue a trajectory and how much path overshoot is tolerated [13].

Nonlinear control is perhaps the least elementary of the main control methods. Yet, with nonlinearities being a main source of control failure, this control provides satisfactory results where other control laws fail [21]. Nonlinear controls use a variety of techniques such as feedback linearization, sliding mode control, Lyapunov redesign, backstepping, and passivity-based control to achieve sufficient levels of control stability. Kaminer et al. provides an example of UAV nonlinear control with strict performance requirements despite modeling uncertainties and environmental disturbances [22].

In contrast, optimal control places focus on minimizing or maximizing costs like time-to-go and vehicle energy usage discussed in the 1.1.2 Guidance section [23]. While retaining dynamic system limitations, optimal control uses gradient methods to determine control inputs that correspond to these minimized/maximized costs. Most commonly known is LQR control, where the system combines energy (control input), transient state, and terminal state costs into a solvable form using Riccati Equations [24]. Due to a variety of factors such as computational expense, nonlinearities, and system stability, a variety of equation-solving architectures are used to solve optimal control problems. Among these are the popular direct shooting methods, where the system is discretized and solved using optimality conditions [25]; computationally fast collocation methods, which use polynomial curve fits to develop control trajectories [4]; and the quickly-intractable dynamic programming, which finds a global optimal solution by implicitly considering all possible trajectories using Bellman’s principle of optimality [26].

Despite the large amounts of concepts, laws, and solver techniques involved with control, most of the previously mentioned control laws do not attempt to reduce navigation errors. They focus on minimizing flight path error. They assume that the navigation (observer) will provide error compensation and focus on developing a controller that works sufficiently amidst the uncertainty. There appears to be disregard for the possibility that certain control inputs may be inducing significant amounts of uncertainty into the system. Stochastic optimal control [15,27] came closest to addressing this error minimization. Using the Ito equation (linearization to the second derivative) and dynamic programming, Maybeck discusses how to define cost functions such that state expectations are maximized [26]. However, stochastic optimal control is more so tolerating the errors rather than trying to prevent them to begin with. For example, A cost may call for speed minimization. Stochastic optimal control will perform this minimization, tolerating error growth in GPSD, oblivious to the fact that speeding out of GPSD would significantly reduce error growth. In short, the research lacks control studies that focus on navigation error prevention.

CHAPTER 2

DESIGN DEVELOPEMENT

This chapter will detail the simulation's design description. The description will provide an extensive explanation and validation on the indirect, extended Kalman filter design (IEKF, 2.1). The guidance and control law design, using pre-built toolboxes and libraries, will then be outlined (2.2). Simulations in this document are implemented using MATLAB.

2.1 Kalman Filter Development

This section will explain how the Kalman filter is generated. This section will also include the validation steps used to ensure correct implementation of that filter. Among these steps are defining the system states and models (2.1.1), error state mappings (2.1.2) propagation validation (2.1.3), linear error state modeling (2.1.4), linear error measurement modeling (2.1.5), covariance propagation (2.1.6), and discussing the system's measurement estimation capability (2.1.7). Note much of the Kalman filter implementation is based off the derivations of Maybeck [3,28] as well as Christensen and Geller [29].

Before beginning, the reasoning for using an IEKF may be desired. UAV flight is nonlinear; furthermore, particularly with vision sensors, flight measurements are also nonlinear. Such factors detract from using a linear Kalman filter. An extended Kalman filter accommodates for such nonlinearities via linearization. Using an indirect Kalman filter, where error states are tracked over time, as opposed to only tracking states, allow the engineer to incorporate state measurement updates that are not additive (e.g., measurement updates that are in different coordinate frames than the state). The complexities and computations needed to generate an unscented Kalman filter or a particle filter seemed excessive for this foundational study. Furthermore, the simplified nonlinearities associated with the UAV, which are shown in the following 2.1.1 States and Models subsection, did not appear extreme or harsh enough to justify the use of an unscented Kalman filter or particle filter.

2.1.1 States and Models

To simulate the drone forward in time, the system state and state dynamics must be known. Beard and McLain provide an extensive 3D "truth model" for UAVs and have collected coefficient and mass parameters specifically for Aerosonde models [13]. Using Figure 2.1 for reference, this 3D model is simplified to a projection in the 2D plane.

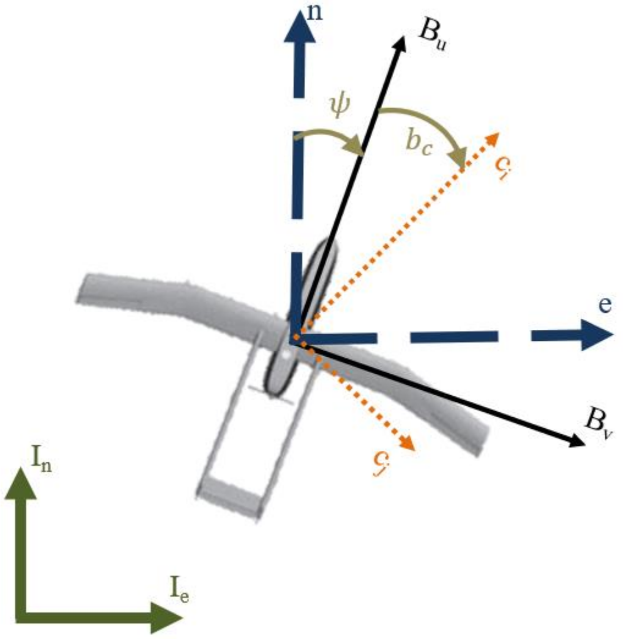


Figure 2.1. 2D UAV depiction with frames of reference. Adapted from [13].

In particular, Figure 2.1 provides the frames of reference associated with the state and dynamics models, that is, an Inertial frame (I_n, I_e), North-East frame (n, e), Body frame

(B_u, B_v), and Camera frame (c_i, c_j). Note that n is always parallel with I_n and e is always parallel with I_e. Additionally, the heading angle (ψ) and a camera bias (i.e. misalignment or offset) angle (b_c) are indicated.

As part of the UAV states associated with Figure 1, the Kalman filter can utilize sensor bias states to predict better estimates of the truth UAV states. This research will incorporate two accelerometer biases, a gyroscope angular rate bias, and a camera angle bias. Furthermore, SLAM utilizes landmark state positions not only to log current landmark estimates, but to also simultaneously assist in estimating UAV states. In this research, zero to two landmarks having north and east coordinates will be appended to the state. As such, the complete truth state model is mathematically defined as

$$\bar{X} = \begin{bmatrix} p_n \\ p_e \\ u \\ v \\ \psi \\ r \\ b_{a_u} \\ b_{a_v} \\ b_\omega \\ b_c \\ \bar{\Lambda}_{ne} \end{bmatrix} \tag{2.1}$$

An overbar will denote a variable's matrix characteristic. The UAV's north position and east position is described by p_n and p_e respectively. u and v define the vehicle's velocity along axis B_u and B_v in the Body frame respectively. Heading angle, ψ, and heading rate, r, are provided with respect to the North-East frame (with north-east-down convention, right-hand-rule is positive). Following the UAV state terms are the bias error terms for the accelerometer measurements along the Body frame's u (b_{a_u}) and v (b_{a_v}) directions as well as the bias errors associated with the UAV's gyroscope (b_ω) and a forward-facing camera (b_c). Scale factor errors are assumed negligible for this study and are not included in the state. Following bias errors is $\overline{\Lambda}_{ne}$, which is a vector of uncertain landmark locations defined as

$$\overline{\Lambda}_{ne} = \begin{bmatrix} \Lambda_{1n} \\ \Lambda_{1e} \\ \Lambda_{2n} \\ \Lambda_{2e} \end{bmatrix} \tag{2.2}$$

where the subscript n or e denote a landmark's north or east coordinates respectively and the numeral subscripts denote whether the coordinate is for the first or second landmark. Note that all equations will be presented assuming two landmarks; for simulations with fewer landmarks, equation rows and columns associated with the unused landmarks are removed.

In conjunction to this truth state is the truth dynamics. Again, using a 2D projection of Beard and McLain's work, these dynamics are defined in state-space form as

$$\dot{\bar{X}} = \bar{f}(\bar{X}, \bar{U}, \bar{w}) = \begin{bmatrix} \dot{p}_n \\ \dot{p}_e \\ \dot{u} \\ \dot{v} \\ \dot{\psi} \\ \dot{r} \\ \dot{b}_{a_u} \\ \dot{b}_{a_v} \\ \dot{b}_{\omega} \\ \dot{b}_c \\ \dot{\bar{\Lambda}}_{ne} \end{bmatrix} = \begin{bmatrix} u\cos(\psi) - v\sin(\psi) \\ u\sin(\psi) + v\cos(\psi) \\ r\,v - \dfrac{C_{D_o}\,\rho\,V_a^2\,S}{2m} + \dfrac{F_T}{m} \\ -r\,u + \dfrac{\Xi_1\,\rho\,V_a^2\,S}{2m} \\ r \\ \dfrac{\Xi_2\,\rho\,V_a^2\,S\,b}{2} \\ -\dfrac{b_{a_u}}{\tau_{a_u}} + w_{a_u} \\ -\dfrac{b_{a_v}}{\tau_{a_v}} + w_{a_v} \\ -\dfrac{b_{\omega}}{\tau_{\omega}} + w_{\omega} \\ -\dfrac{b_c}{\tau_c} + w_c \\ \bar{0} \end{bmatrix} \tag{2.3}$$

where

$$\begin{bmatrix} \Xi_1 \\ \Xi_2 \\ \beta \\ V_a \\ u_r \\ v_r \end{bmatrix} = \begin{bmatrix} C_{Y_0} + C_{Y_\beta}\,\beta + \dfrac{C_{Y_r}\,b\,r}{2\,V_a} + C_{Y_{\widehat{\delta}_r}}\,\widehat{\delta}_r \\ C_{r_0} + C_{r_\beta}\,\beta + \dfrac{C_{r_r}\,b\,r}{2\,V_a} + C_{r_{\widehat{\delta}_r}}\,\widehat{\delta}_r \\ \sin^{-1}\left(\dfrac{v_r}{V_a}\right) \\ \sqrt{u_r^2 + v_r^2} \\ u - u_{wg} \\ v - v_{wg} \end{bmatrix} \tag{2.4}$$

and the system is assumed to have no ambient wind. Moving to the right-hand side of Equation (2.3), $\dot{u}$, $\dot{v}$, and $\dot{r}$ accelerations are functions of air density (ρ), the airframe's magnitude velocity (V_a; relative to surrounding air mass; note that u_r and v_r are relative wind projected onto the body u and v axis respectively and subscript wg denotes wind gust), UAV mass (m), wingspan (b), planform area (S), and side slip angle (β). For simplicity, the control inputs have been simplified into two parameters: ruddervator deflection ($\overset{\frown}{\delta}_r$, referenced as rudder in this document) and thrust input (F_T). Note that $\bar{U} = \begin{bmatrix} F_T & \overset{\frown}{\delta}_r \end{bmatrix}^T$ (subscript T for thrust, superscript T for matrix transpose). The accelerations are also functions of aerodynamic coefficients (uppercase C_*) as defined in [13]. Bias derivatives are modeled as first-order Markov processes (exponentially correlated random variables) [3], which include their respective driving noise terms (w_*) and time constants (τ_*). Note that $\bar{w} = [u_{wg} \quad v_{wg} \quad w_{a_u} \quad w_{a_v} \quad w_{\omega} \quad w_c]^T$ and that Markov processes are used due their versatility in modeling any spectrum between white noise and a constant bias; more information regarding these and future defined noise terms will be provided in the 2.1.6 Covariance Propagation and 2.1.7 Estimation Capability subsections. For this study, the landmarks are assumed to be stationary, hence their dynamics are implanted as the zero vector. The use of Euler angles, rather than quaternions, is maintained for better intuition and simplicity. For these and future equations, the explicit variable dependencies on time are usually dropped for visual clarity purposes.

The equations just described define the system truth state and truth dynamic models for this research—the states and dynamics that are assumed to perfectly, realistically model a physical Aerosonde system. Unfortunately, in reality, the exact, real-time values for many of the equation terms parameters are not directly available; they can, however, be obtained indirectly by means of the UAV sensor package. In the context of this paper, the sensor package includes continuous measurements from the accelerometers and gyroscope as well as discrete measurements (when available) from GPS satellites, magnetometer, and a pinhole camera (implementation with velocimeters and other equipment is left to future work). In mathematical form, these measurements, indicated with a tilde overmark, are modeled as

$$\begin{bmatrix} \tilde{a}_u \\ \tilde{a}_v \\ \tilde{\omega} \end{bmatrix} = \tilde{\bar{y}} = \begin{bmatrix} a_u + b_{a_u} + \eta_{a_u} \\ a_v + b_{a_v} + \eta_{a_v} \\ \omega + b_\omega + \eta_\omega \end{bmatrix} = \begin{bmatrix} \dot{u} - r\,v + b_{a_u} + \eta_{a_u} \\ \dot{v} + r\,u + b_{a_v} + \eta_{a_v} \\ r + b_\omega + \eta_\omega \end{bmatrix} \tag{2.5}$$

and

$$\begin{bmatrix} \tilde{p}_n \\ \tilde{p}_e \\ \tilde{\psi} \\ \tilde{\bar{\Gamma}} \end{bmatrix} = \tilde{\bar{z}} = \bar{h}\big(\bar{X}(t_k), \bar{\eta}(t_k)\big) = \begin{bmatrix} p_n(t_k) + \eta_{p_n}(t_k) \\ p_e(t_k) + \eta_{p_e}(t_k) \\ \psi(t_k) + \eta_\psi(t_k) \\ \bar{\Gamma}(t_k) + \bar{\eta}_{\bar{\Gamma}}(t_k) \end{bmatrix} \tag{2.6}$$

where

$$\bar{\Gamma}(t_k) = \begin{bmatrix} \Lambda_{1j}(t_k)/\Lambda_{1i}(t_k) \\ \Lambda_{2j}(t_k)/\Lambda_{2i}(t_k) \end{bmatrix} \tag{2.7}$$

$$\bar{\eta}_{\bar{\Gamma}}(t_k) = \begin{bmatrix} \bar{\eta}_{\bar{\Gamma}_{[1]}}(t_k) \\ \bar{\eta}_{\bar{\Gamma}_{[2]}}(t_k) \end{bmatrix} \tag{2.8}$$

$$\bar{\Lambda}_{ij}(t_k) = \begin{bmatrix} \Lambda_{1i}(t_k) \\ \Lambda_{1j}(t_k) \\ \Lambda_{2i}(t_k) \\ \Lambda_{2j}(t_k) \end{bmatrix} \tag{2.9}$$

In detail, Equation (2.5) states that the continuous measurements of the 2-axis accelerometer ($\tilde{a}_*$) and the gyroscope ($\tilde{\omega}$) are equal to the true accelerometer acceleration (a_*) or true gyroscope angular rate (ω) plus their respective bias (b_*) plus their respective continuous measurement noise (η_*, modeled as a gaussian random process). Furthermore, Equation (2.5) can be equivalently expressed with the following substitutions: True u-axis accelerometer measurements, $\tilde{a}_u$, equal the vehicle's linear acceleration along axis B_u minus Coriolis acceleration plus bias and noise. True v-axis accelerometer measurements, $\tilde{a}_v$, equal the vehicle's linear acceleration along axis B_v plus Coriolis acceleration, bias, and noise. True gyroscope measurements, $\widetilde{\omega}$, equal the vehicle's true angular velocity, r, plus bias and noise. For visual purposes in future derivations, Equation (2.5) is rearranged to define linear acceleration and angular velocity as follows:

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ r \end{bmatrix} = \begin{bmatrix} \tilde{a}_u + r\,v - b_{a_u} - \eta_{a_u} \\ \tilde{a}_v - r\,u - b_{a_v} - \eta_{a_v} \\ \widetilde{\omega} - b_\omega - \eta_\omega \end{bmatrix} \tag{2.10}$$

Equation (2.6) states that the discrete measurements of the GPS and magnetometer are a function of the truth state at a discrete time, t_k, plus discrete measurement noise $\eta_*(t_k)$. The pinhole-camera measurement model, $\bar{\Gamma}(t_k)$, which is composed of elements from the line-of-sight vector, $\overline{\Lambda}_{ij}(t_k)$, is similarly defined. For Equation (2.8) and future equations, subscript brackets around a number or number set simply refer to the matrix position of that subscript's variable (note $\bar{\eta}(t_k) = [\eta_{p_n}(t_k) \quad \eta_{p_e}(t_k) \quad \eta_\psi(t_k) \quad \bar{\eta}_{\Gamma_{[1]}}(t_k) \quad \bar{\eta}_{\Gamma_{[2]}}(t_k)]$).

Before continuing, there is need to further describe the line-of-sight vector and Equation (2.7)'s camera measurement definition. The description will help provide intuition for later derivations in the Kalman filter. First, a visual is provided.

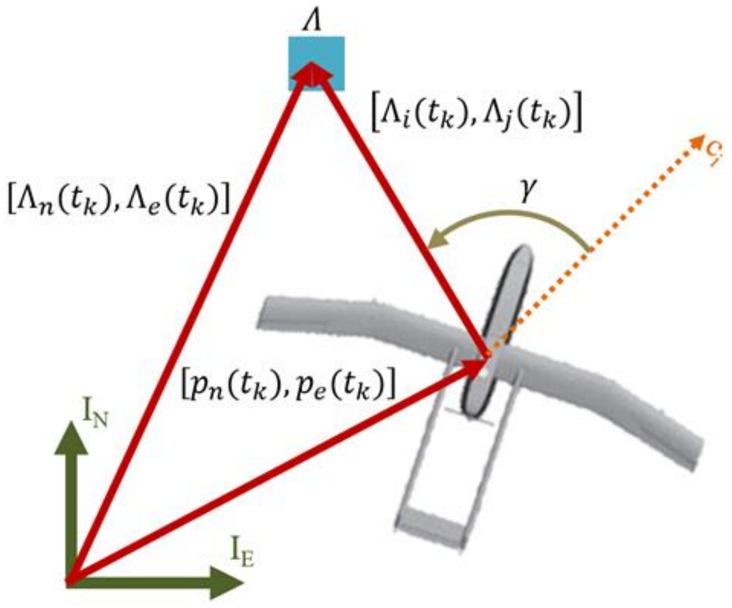

Λ
$[\Lambda_i(t_k), \Lambda_j(t_k)]$
c_i
$[\Lambda_n(t_k), \Lambda_e(t_k)]$
γ
$[p_n(t_k), p_e(t_k)]$
I_N
I_E

Figure 2.2. Visualization of line-of-sight vector derivation and γ. Adapted from [13].

In this figure, a position vector loop is drawn between the Inertial frame origin, drone, and a single landmark, Λ (assuming that the camera is located at the drone's center of mass for simplicity). The truth state, defined by Equation (2.1), yields information regarding the landmark location vector, $[\Lambda_n(t_k), \Lambda_e(t_k)]$, and drone position vector, $[p_n(t_k), p_e(t_k)]$, with respect to the Inertial frame, expressed in the Inertial frame. Obtaining the camera's line-of sight vector, $[\Lambda_i(t_k), \Lambda_j(t_k)]$ (landmark location vector with respect to the drone camera frame, expressed in the camera frame), for a single landmark can then be derived as follows:

$$\begin{aligned}\begin{bmatrix}\Lambda_i(t_k)\\ \Lambda_j(t_k)\end{bmatrix} &= \bar{T}_B^c \bar{T}_I^B \begin{bmatrix}\Lambda_n(t_k)\\ \Lambda_e(t_k)\end{bmatrix} - \bar{T}_B^c \bar{T}_I^B \begin{bmatrix}p_n(t_k)\\ p_e(t_k)\end{bmatrix} \\ &= \bar{T}_B^c \bar{T}_I^B \begin{bmatrix}\Lambda_n(t_k) - p_n(t_k)\\ \Lambda_e(t_k) - p_e(t_k)\end{bmatrix}\end{aligned} \tag{2.11}$$

$$\bar{T}_B^c = \begin{bmatrix}\cos(b_c(t_k)) & \sin(b_c(t_k))\\ -\sin(b_c(t_k)) & \cos(b_c(t_k))\end{bmatrix} \tag{2.12}$$

$$\bar{T}_I^B = \begin{bmatrix} \cos(\psi(t_k)) & \sin(\psi(t_k)) \\ -\sin(\psi(t_k)) & \cos(\psi(t_k)) \end{bmatrix} \tag{2.13}$$

In this derivation, the camera line of sight vector is equivalent to the landmark location vector minus the drone position vector. However, having the need to express all vectors in the same reference frame, the landmark location and drone position vectors must be transformed. This is done via two transformation matrices: One to transform from Inertial frame coordinates to Body frame coordinates, $\bar{T}_I^B$, and one to transform from Body frame to Camera frame coordinates, $\bar{T}_B^C$. A similar derivation is used when augmenting the second landmark line-of-sight vectors to Equation (2.9).

Essentially, the pinhole model is a method for projecting a 2D position into a 1D quantity, as is the nature of cameras. The convention for this projection is to divide the non-bore-axis line-of-sight value by the bore-axis line-of-sight value. Hence, since this study's camera is forward facing, the landmarks' i elements will be in the denominator. Future work in 3D will then have two projected quantities per landmark for one camera, i.e., j/i and $down\ axis/i$ to project from 3D to 2D. As a last note, γ in the figure is the angle between the UAV's heading and the landmark (used in the 3.2 Guidance and Control Law Development section).

The line-of-sight vector and camera model has just now been described. This description was used to help explain part of the sensor packages onboard the UAV. Again, the sensor packages are used because they can indirectly obtain many of the terms in the truth state and truth dynamics models. To utilize the sensor packages, engineers will replace dynamic terms in the truth models with the sensor terms and appropriate, accurate design parameters. This model replacement creates the design model, denoted with the breve overmark, given by Equation (2.14) through Equation (2.23).

$$\breve{\bar{x}} = \begin{bmatrix} \breve{p}_n \\ \breve{p}_e \\ \breve{u} \\ \breve{v} \\ \breve{\psi} \\ \breve{b}_{a_u} \\ \breve{b}_{a_v} \\ \breve{b}_{\omega} \\ \breve{b}_c \\ \breve{\bar{\bar{\Lambda}}}_{ne} \end{bmatrix} \tag{2.14}$$

$$\dot{\breve{\bar{x}}} = \breve{\bar{f}}\left(\breve{\bar{x}}, \breve{\tilde{\bar{y}}}, \breve{\bar{w}}\right) = \begin{bmatrix} \dot{\breve{p}}_n \\ \dot{\breve{p}}_e \\ \dot{\breve{u}} \\ \dot{\breve{v}} \\ \dot{\breve{\psi}} \\ \dot{\breve{b}}_{a_u} \\ \dot{\breve{b}}_{a_v} \\ \dot{\breve{b}}_{\omega} \\ \dot{\breve{b}}_c \\ \dot{\breve{\bar{\bar{\Lambda}}}}_{ne} \end{bmatrix} = \begin{bmatrix} \breve{u}\cos\left(\breve{\psi}\right) - \breve{v}\sin\left(\breve{\psi}\right) \\ \breve{u}\sin\left(\breve{\psi}\right) + \breve{v}\cos(\breve{\psi}) \\ \tilde{a}_u + \tilde{\omega}\,\breve{v} - \breve{b}_{a_u} - \breve{\eta}_{a_u} \\ \tilde{a}_v - \tilde{\omega}\,\breve{u} - \breve{b}_{a_v} - \breve{\eta}_{a_v} \\ \tilde{\omega} - \breve{b}_{\omega} - \breve{\eta}_{\omega} \\ -\dfrac{\breve{b}_{a_u}}{\breve{\tau}_{a_u}} + \breve{w}_{a_u} \\ -\dfrac{\breve{b}_{a_v}}{\breve{\tau}_{a_v}} + \breve{w}_{a_v} \\ -\dfrac{\breve{b}_{\omega}}{\breve{\tau}_{\omega}} + \breve{w}_{\omega} \\ -\dfrac{\breve{b}_c}{\breve{\tau}_c} + \breve{w}_c \\ \bar{0} \end{bmatrix} \tag{2.15}$$

$$\begin{bmatrix} \tilde{a}_u \\ \tilde{a}_v \\ \tilde{\omega} \end{bmatrix} \approx \breve{\tilde{\bar{y}}} = \begin{bmatrix} \breve{a}_u + \breve{b}_{a_u} + \breve{\eta}_{a_u} \\ \breve{a}_v + \breve{b}_{a_v} + \breve{\eta}_{a_v} \\ \breve{\omega} + \breve{b}_{\omega} + \breve{\eta}_{\omega} \end{bmatrix} = \begin{bmatrix} \dot{\breve{u}} - \breve{\omega}\,\breve{v} + \breve{b}_{a_u} + \breve{\eta}_{a_u} \\ \dot{\breve{v}} + \breve{\omega}\,\breve{u} + \breve{b}_{a_v} + \breve{\eta}_{a_v} \\ \breve{\omega} + \breve{b}_{\omega} + \breve{\eta}_{\omega} \end{bmatrix} \tag{2.16}$$

$$\begin{bmatrix} \tilde{p}_n \\ \tilde{p}_e \\ \tilde{\psi} \\ \tilde{\bar{\bar{\Gamma}}} \end{bmatrix} \approx \breve{\bar{\tilde{z}}} = \breve{\bar{h}}\left(\breve{\bar{x}}(t_k), \breve{\bar{\eta}}(t_k)\right) = \begin{bmatrix} \breve{p}_n(t_k) + \breve{\eta}_{p_n}(t_k) \\ \breve{p}_e(t_k) + \breve{\eta}_{p_e}(t_k) \\ \breve{\psi}(t_k) + \breve{\eta}_{\psi}(t_k) \\ \breve{\bar{\bar{\Gamma}}}(t_k) + \breve{\bar{\bar{\eta}}}_{\bar{\Gamma}}(t_k) \end{bmatrix} \tag{2.17}$$

$$\breve{\bar{\bar{\Gamma}}}(t_k) = \begin{bmatrix} \breve{\Lambda}_{1j}(t_k)/\breve{\Lambda}_{1i}(t_k) \\ \breve{\Lambda}_{2j}(t_k)/\breve{\Lambda}_{2i}(t_k) \end{bmatrix} \tag{2.18}$$

$$\breve{\bar{\bar{\eta}}}_{\breve{\bar{\bar{\Gamma}}}}(t_k) = \begin{bmatrix} \breve{\bar{\bar{\eta}}}_{\breve{\bar{\bar{\Gamma}}}_{[1]}}(t_k) \\ \breve{\bar{\bar{\eta}}}_{\breve{\bar{\bar{\Gamma}}}_{[2]}}(t_k) \end{bmatrix} \tag{2.19}$$

$$\breve{\bar{\Lambda}}_{ij}(t_k) = \begin{bmatrix} \breve{\Lambda}_{1i}(t_k) \\ \breve{\Lambda}_{1j}(t_k) \\ \breve{\Lambda}_{2i}(t_k) \\ \breve{\Lambda}_{2j}(t_k) \end{bmatrix} \tag{2.20}$$

$$\begin{aligned} \begin{bmatrix} \breve{\Lambda}_i(t_k) \\ \breve{\Lambda}_j(t_k) \end{bmatrix} &= \breve{\bar{\bar{T}}}_B^c \breve{\bar{\bar{T}}}_I^B \begin{bmatrix} \breve{\Lambda}_n(t_k) \\ \breve{\Lambda}_e(t_k) \end{bmatrix} - \breve{\bar{\bar{T}}}_B^c \breve{\bar{\bar{T}}}_I^B \begin{bmatrix} \breve{p}_n(t_k) \\ \breve{p}_e(t_k) \end{bmatrix} \\ &= \breve{\bar{\bar{T}}}_B^c \breve{\bar{\bar{T}}}_I^B \begin{bmatrix} \breve{\Lambda}_n(t_k) - \breve{p}_n(t_k) \\ \breve{\Lambda}_e(t_k) - \breve{p}_e(t_k) \end{bmatrix} \end{aligned} \tag{2.21}$$

$$\breve{\bar{\bar{T}}}_B^c = \begin{bmatrix} \cos\left(\breve{b}_c(t_k)\right) & \sin\left(\breve{b}_c(t_k)\right) \\ -\sin\left(\breve{b}_c(t_k)\right) & \cos\left(\breve{b}_c(t_k)\right) \end{bmatrix} \tag{2.22}$$

$$\breve{\bar{\bar{T}}}_I^B = \begin{bmatrix} \cos\left(\breve{\psi}(t_k)\right) & \sin\left(\breve{\psi}(t_k)\right) \\ -\sin\left(\breve{\psi}(t_k)\right) & \cos\left(\breve{\psi}(t_k)\right) \end{bmatrix} \tag{2.23}$$

For this design, the velocity states are kept in the Body frame and not transformed to the North-East frame. Initially, making such transformation appeared to be favorable as it would provide consistency with the first and second states and seemed more informative to navigation engineers. This practice is NOT recommended. The author found the practice troublesome for integration and Kalman derivations. Furthermore, obtaining North-East frame values for navigation engineers was more easily calculated after, not during, the simulation.

In Equation (2.15), the values of $\dot{\breve{u}}$ and $\dot{\breve{v}}$ are derived in the same manner as Equation (2.10) by rearranging Equation (2.16). Furthermore, note that the heading rate, r, and heading rate derivative, $\dot{r}$, have been removed since dynamic propagation is not needed due to the availability of continuous, not discrete, gyroscope measurements. A lowercase $\bar{x}$ is used to denote the absence of these states. The design model noise vector is also modified to $\breve{\bar{w}} = [\breve{\eta}_{a_u} \quad \breve{\eta}_{a_v} \quad \breve{\eta}_{\omega} \quad \breve{w}_{a_u} \quad \breve{w}_{a_v} \quad \breve{w}_{\omega} \quad \breve{w}_{c}]^T$, which includes continuous measurement noise with the bias driving noise. All $\breve{\bar{w}}$ is now technically described as process noise, rather than measurement and driving noise.

In the context of this research, this design model will be used for linearization and is a preliminary step to deriving the next model. The caveat to the design model is that, to date, process noise is not measurable. Additionally, the engineer may decide to substitute appropriate design parameters, such as bias, with alternative, simpler, obtainable parameters that sufficiently describe a given parameter. Therefore, making a model that gives the best, realistic state estimate with the alternative parameters, engineers develop a third model—the navigation model. This model will be represented with a hat overmark and is defined as follows:

$$\hat{\bar{x}} = \begin{bmatrix} \hat{p}_n \\ \hat{p}_e \\ \hat{u} \\ \hat{v} \\ \hat{\psi} \\ \hat{b}_{a_u} \\ \hat{b}_{a_v} \\ \hat{b}_{\omega} \\ \hat{b}_{c} \\ \hat{\Lambda}_{ne} \end{bmatrix} \tag{2.24}$$

$$\dot{\hat{\bar{x}}} = \hat{\bar{f}}\left(\hat{\bar{x}}, \hat{\tilde{\bar{y}}}\right) = \begin{bmatrix} \dot{\hat{p}}_n \\ \dot{\hat{p}}_e \\ \dot{\hat{u}} \\ \dot{\hat{v}} \\ \dot{\hat{\psi}} \\ \dot{\hat{b}}_{a_u} \\ \dot{\hat{b}}_{a_v} \\ \dot{\hat{b}}_{\omega} \\ \dot{\hat{b}}_c \\ \dot{\hat{\bar{\Lambda}}}_{ne} \end{bmatrix} = \begin{bmatrix} \hat{u}\cos\left(\hat{\psi}\right) - \hat{v}\sin\left(\hat{\psi}\right) \\ \hat{u}\sin\left(\hat{\psi}\right) + \hat{v}\cos(\hat{\psi}) \\ \tilde{a}_u + \widehat{\omega}\,\hat{v} - \hat{b}_{a_u} \\ \tilde{a}_v - \widehat{\omega}\,\hat{u} - \hat{b}_{a_v} \\ \widetilde{\omega} - \hat{b}_{\omega} \\ -\dfrac{\hat{b}_{a_u}}{\tau_{a_u}} \\ -\dfrac{\hat{b}_{a_v}}{\tau_{a_v}} \\ -\dfrac{\hat{b}_{\omega}}{\tau_{\omega}} \\ -\dfrac{\hat{b}_c}{\tau_c} \\ \bar{0} \end{bmatrix} \tag{2.25}$$

where

$$\begin{bmatrix} \tilde{a}_u \\ \tilde{a}_v \\ \widetilde{\omega} \end{bmatrix} \approx \hat{\tilde{\bar{y}}} = \begin{bmatrix} \hat{a}_u + \hat{b}_{a_u} \\ \hat{a}_v + \hat{b}_{a_v} \\ \widehat{\omega} + \hat{b}_{\omega} \end{bmatrix} = \begin{bmatrix} \dot{\hat{u}} - \widehat{\omega}\,\hat{v} + \hat{b}_{a_u} \\ \dot{\hat{v}} + \widehat{\omega}\,\hat{u} + \hat{b}_{a_v} \\ \widehat{\omega} + \hat{b}_{\omega} \end{bmatrix} \tag{2.26}$$

$$\begin{bmatrix} \tilde{p}_n \\ \tilde{p}_e \\ \tilde{\psi} \\ \tilde{\bar{\Gamma}} \end{bmatrix} \approx \hat{\bar{z}} = \hat{\bar{h}}\left(\hat{\bar{x}}(t_k)\right) = \begin{bmatrix} \hat{p}_n(t_k) \\ \hat{p}_e(t_k) \\ \hat{\psi}(t_k) \\ \hat{\bar{\Gamma}}(t_k) \end{bmatrix} \tag{2.27}$$

$$\hat{\bar{\Gamma}}(t_k) = \begin{bmatrix} \widehat{\Lambda}_{1j}(t_k)/\widehat{\Lambda}_{1i}(t_k) \\ \widehat{\Lambda}_{2j}(t_k)/\widehat{\Lambda}_{2i}(t_k) \end{bmatrix} \tag{2.28}$$

$$\widehat{\bar{\Lambda}}_{ij}(t_k) = \begin{bmatrix} \widehat{\Lambda}_{1i}(t_k) \\ \widehat{\Lambda}_{1j}(t_k) \\ \widehat{\Lambda}_{2i}(t_k) \\ \widehat{\Lambda}_{2j}(t_k) \end{bmatrix} \tag{2.29}$$

$$\begin{aligned} \begin{bmatrix} \widehat{\Lambda}_i(t_k) \\ \widehat{\Lambda}_j(t_k) \end{bmatrix} &= \hat{\bar{T}}_B^c \hat{\bar{T}}_I^B \begin{bmatrix} \widehat{\Lambda}_n(t_k) \\ \widehat{\Lambda}_e(t_k) \end{bmatrix} - \hat{\bar{T}}_B^c \hat{\bar{T}}_I^B \begin{bmatrix} \hat{p}_n(t_k) \\ \hat{p}_e(t_k) \end{bmatrix} \\ &= \hat{\bar{T}}_B^c \hat{\bar{T}}_I^B \begin{bmatrix} \widehat{\Lambda}_n(t_k) - \hat{p}_n(t_k) \\ \widehat{\Lambda}_e(t_k) - \hat{p}_e(t_k) \end{bmatrix} \end{aligned} \tag{2.30}$$

$$\hat{\hat{T}}_B^c = \begin{bmatrix} \cos\left(\hat{b}_c(t_k)\right) & \sin\left(\hat{b}_c(t_k)\right) \\ -\sin\left(\hat{b}_c(t_k)\right) & \cos\left(\hat{b}_c(t_k)\right) \end{bmatrix} \tag{2.31}$$

$$\hat{\hat{T}}_I^B = \begin{bmatrix} \cos\left(\hat{\psi}(t_k)\right) & \sin\left(\hat{\psi}(t_k)\right) \\ -\sin\left(\hat{\psi}(t_k)\right) & \cos\left(\hat{\psi}(t_k)\right) \end{bmatrix} \tag{2.32}$$

$$\hat{\hat{\Lambda}}_{ne}(t_k) = \begin{bmatrix} \hat{\Lambda}_{1n}(t_k) \\ \hat{\Lambda}_{1e}(t_k) \\ \hat{\Lambda}_{2n}(t_k) \\ \hat{\Lambda}_{2e}(t_k) \end{bmatrix} \tag{2.33}$$

Essentially, the navigation model approximates the truth model. The navigation model propagates states forward in time and will account for (subtract out) bias errors to get as near to the true model as possible. The navigation model is what the Kalman filter will use to perform state estimates. Again, like the design model, the navigation model attempts to derive $\dot{\hat{u}}$ and $\dot{\hat{v}}$ by solving for $\dot{\hat{u}}$, $\dot{\hat{v}}$, and $\hat{\omega}$ in Equation (2.26).

2.1.2 Error State Mappings

The prior subsection just derived the state models that are used within the Kalman filter: the design model for linearization in future subsections and the navigation model for state estimate propagation. However, for simulations, the truth model can still be utilized. By propagating both the truth and navigation models forward in time, the results from both models can be compared to confirm that navigation dynamics sufficiently matches and represents the true dynamics throughout the given simulation time. Furthermore, error comparisons of simulation results can be developed with the truth model. To do this dynamics confirmation and error comparison, mappings between the navigation and the truth state are first needed (for clarity, these are mappings between states, not dynamics).

To model the error discrepancies between the true state and the UAV's navigation state, an error state matrix is developed.

$$\delta\bar{x} = \begin{bmatrix} \delta p_n \\ \delta p_e \\ \delta u \\ \delta v \\ \delta\psi \\ \delta b_{a_u} \\ \delta b_{a_v} \\ \delta b_\omega \\ \delta b_c \\ \delta\bar{\Lambda}_{ne} \end{bmatrix} \tag{2.34}$$

A δ is denoted as the perturbation difference between a true state value and that true state value's respective estimation measurement. The relationship between this error matrix, the truth state, and the error state is explicitly written as

$$\bar{x} = \begin{bmatrix} p_n \\ p_e \\ u \\ v \\ \psi \\ b_{a_u} \\ b_{a_v} \\ b_\omega \\ b_c \\ \bar{\Lambda}_{ne} \end{bmatrix} = \begin{bmatrix} \hat{p}_n + \delta p_n \\ \hat{p}_e + \delta p_e \\ \hat{u} + \delta u \\ \hat{v} + \delta v \\ \hat{\psi} + \delta\psi \\ \hat{b}_{a_u} + \delta b_{a_u} \\ \hat{b}_{a_v} + \delta b_{a_v} \\ \hat{b}_\omega + \delta b_\omega \\ \hat{b}_c + \delta b_c \\ \hat{\bar{\Lambda}}_{ne} + \delta\bar{\Lambda}_{ne} \end{bmatrix} \tag{2.35}$$

Again, note that the truth state matrix has been modified, denoted with a lowercase $\bar{x}$, to not incorporate heading rate, which is absent in the navigation state. Alternatively, the mappings needed to transfer between one state to the next can equivalently be defined as

$$\bar{x} = \bar{l}(\hat{\bar{x}}, \delta\bar{x}) = \hat{\bar{x}} + \delta\bar{x} \tag{2.36}$$

$$\hat{\bar{x}} = \bar{m}(\bar{x}, \delta\bar{x}) = \bar{x} - \delta\bar{x} \tag{2.37}$$

$$\delta\bar{x} = \bar{n}(\hat{\bar{x}}, \bar{x}) = \bar{x} - \hat{\bar{x}} \tag{2.38}$$

In conclusion, $\bar{l}(\hat{\bar{x}}, \delta\bar{x})$ is a mapping that corrects the state estimate to a modified true state; $\bar{m}(\bar{x}, \delta\bar{x})$ is the mapping that "injects" errors to create the navigation state, and $\bar{n}(\hat{\bar{x}}, \bar{x})$ is the mapping to calculate the error state.

To confirm consistency between these mappings within the simulation, a validation function is developed. In this validation function, (1) the navigation state is initially set without any pre-occurring errors (equivalent to the truth state). (2) errors are injected into the navigation state using the $\bar{m}$ mapping. (3) The errors are rederived using the $\bar{n}$ mapping. These rederived errors are compared to the initial injected errors to ensure that both are equivalent. (4) Using the $\bar{l}$ mapping, the rederived errors are used to correct the navigation state back into a rederived truth state. This rederived truth state is compared to the original truth state to ensure that both are equivalent.

Table 2.1 yields the results from executing this validation function using arbitrary initial truth state values and arbitrary initial injection errors.

Table 2.1: Mapping error validation.

State	$\delta\bar{x} - \delta\bar{x}_{rederived}$	$\bar{x} - \bar{x}_{rederived}$	Units
p_n	0	0	m
p_e	0	0	m
u	2.8e-15	0	m/s
v	0	0	m/s
ψ	-7.1e-17	0	rad
b_{a_u}	0	-1.7e-18	$\mathrm{m/s^2}$
b_{a_v}	0	-2.6e-18	$\mathrm{m/s^2}$
b_ω	0	-6.8e-21	rad/s
b_c	1.7e-18	-8.7e-19	rad
Λ_{1n}	0	0	m
Λ_{1e}	0	0	m
Λ_{2n}	0	0	m
Λ_{2e}	0	0	m

The table indicates that negligible errors are induced using these three mappings. The fact that errors are generated is not due incorrect mapping implementation. Using another arbitrary set of truth states and injection errors, one can achieve complete, zero error in both validation checks. The error is primarily induced through the binary nature of computer computation, where decimal numbers cannot always be fully represented in a binary format (i.e., round-off error). Most all these numbers are below machine epsilon, approximately at 2.22e-16. The u velocity for $\bar{x} - \bar{x}_{rederived}$ is above machine precision. Errors of the same order of magnitude as u were observed when implementing transformation matrices, where errors are due to lack of machine precision to fully represent sine and cosine values. This implies that, up to machine precision, the mappings used within the simulation are consistent.

2.1.3 Propagation Validation

Having states and dynamics defined, the system can be simulated, propagated, forward in time; and with error state mappings defined, a propagation validation algorithm can be made to ensure that this propagation is realistic in that the navigation model sufficiently matches and represents the true dynamics.

In this propagation validation algorithm, the dynamics are simulated forward in time with Euler's method, manipulated control inputs, measurement synthesis, and removal of noise and measurement restrictions. Discussing each of these four simulation characteristics in order, the Euler method is the selected numerical integration algorithm [25]. Using the Euler method, the navigation and truth states, both having the same initial conditions (Table 2.2) are propagated forward with a time step of 0.01s.

Table 2.2. Initial conditions.

State	$\bar{X}$	$\hat{\bar{x}}$	Units
p_n	400.0e+00	400.0e+00	m
p_e	0.0e+00	0.0e+00	m
u	16.0e+00	16.0e+00	$\mathrm{m/s}$
v	0.0e+00	0.0e+00	$\mathrm{m/s}$
ψ	1.6e+00	1.6e+00	rad
r	0.0e+00	NA	$\mathrm{rad/s}$
b_{a_u}	-3.4e-04	-3.4e-04	$\mathrm{m/s^2}$
b_{a_v}	1.6e-03	1.6e-03	$\mathrm{m/s^2}$
b_ω	-1.9e-07	-1.9e-07	$\mathrm{rad/s}$
b_c	3.2e-03	3.2e-03	rad
Λ_{1n}	560.0e+00	560.0e+00	m
Λ_{1e}	340.0e+00	340.0e+00	m
Λ_{2n}	690.0e+00	690.0e+00	m
Λ_{2e}	440.0e+00	440.0e+00	m

Note that the time step was chosen to achieve the lowest computation time that still provided non-diverging results (especially for the 2.1.6 Covariance Propagation subsection). Implementation of higher-fidelity Runge-Kutta algorithms for non-zero-hold, stochastic differential equations [30], implicit integration, or collocation methods [4] is left for future work.

Regarding the second simulation characteristic, the control inputs to the truth state dynamics in Equation (2.3) are manipulated such that $\dot{u} = -\left(r\ v - \frac{C_{D_O}\ \rho\ V_a^2\ S}{2m}\right) + \frac{\sin\left(\frac{t_\epsilon}{10}\right)}{2}$ and that rudder inputs would execute a smooth sinusoidal behavior between maximum and minimum rudder values, $\overbrace{\delta}_r = \frac{\pi \sin\left(\frac{t_\epsilon}{17}\right)}{2}$. t_ϵ is the current simulation time. t_ϵ is different from t_k in that t_ϵ models "pseudo-continuous" simulation time increments of 0.01 s whereas t_k models time in discrete increments of 1.0 s (using subscript ϵ, rather than the commonly used i, j, or n is to reduce confusion between coordinate and state subscripts). t without any subscripts will represent continuous time in future derivations.

In the third characteristic, measurement synthesis, some re-clarification may be needed first. The measurements that the navigation state receives is the truth measurements (Equation (2.5) and Equation (2.6)); nevertheless, when the navigation state needs to guess these measurements for the Kalman filter, the navigation state will use its measurement equations (Equation (2.26) through Equation (2.32)) to make this guess. With the reclarification addressed, the measurement synthesis characteristic can now be described. To generate a_u, a_v , and ω within $\tilde{\bar{y}}$, the algorithm does not use derivative values from Euler's method nor the angular velocity r. Rather, the algorithm attempts to synthesizes the measurements in a more discrete fashion as

$$\begin{bmatrix} \tilde{a}_u(t_{\epsilon-1}) \\ \tilde{a}_v(t_{\epsilon-1}) \\ \tilde{\omega}(t_{\epsilon-1}) \end{bmatrix} = \begin{bmatrix} \frac{u(t_\epsilon) - u(t_{\epsilon-1})}{t_\epsilon - t_{\epsilon-1}} - v(t_{\epsilon-1})\, r(t_{\epsilon-1}) + b_{a_u}(t_{\epsilon-1}) + \eta_{a_u}(t_{\epsilon-1}) \\ \frac{v(t_\epsilon) - v(t_{\epsilon-1})}{t_\epsilon - t_{\epsilon-1}} + u(t_{\epsilon-1})\, r(t_{\epsilon-1}) + b_{a_v}(t_{\epsilon-1}) + \eta_{a_v}(t_{\epsilon-1}) \\ \frac{\psi(t_\epsilon) - \psi(t_{\epsilon-1})}{t_\epsilon - t_{\epsilon-1}} + b_\omega(t_{\epsilon-1}) + \eta_\omega(t_{\epsilon-1}) \end{bmatrix} \tag{2.39}$$

Such a synthesis provides body frame accelerations similar to how physical accelerometers and gyroscopes discretely collect measurements and also accounts for Coriolis terms as described in [13]. From previous author mistakes, the time variables t_ϵ and $t_{\epsilon-1}$ are explicitly labeled in this equation to help future implementers. The use of variables at $t_{\epsilon-1}$ in the Coriolis, bias, and noise terms, as opposed to variables at t_ϵ, is due to the simulation's organization. First, the truth state is propagated from $t_{\epsilon-1}$ to t_ϵ; second, the continuous measurements at $t_{\epsilon-1}$ are calculated; third, the navigation state is propagated from $t_{\epsilon-1}$ to t_ϵ; Finally, if the 1s time increment has elapsed, discrete measurements are calculated (The author assumes that all discrete measurements become available in the same 1s cycle, given that GPS is not denied and assuming that an image can be fully processed by 1s). Therefore, for the navigation state to accurately propagate to t_ϵ, it needs measurements at $t_{\epsilon-1}$, not measurements at a future time. $\frac{\psi(t_\epsilon)-\psi(t_{\epsilon-1})}{t_\epsilon-t_{\epsilon-1}}$ is not used for $r(t_{\epsilon-1})$ in the Coriolis terms because, although it would make the synthesis more discrete-sensor-realistic, such a substitution would imply that values generated by the accelerometer are influenced by values from the gyroscope. For later derivations, the author assumes that sensor measurements are independent from each other, hence the author uses $r(t_{\epsilon-1})$ in the Coriolis term to prevent inter-sensor dependence.

In the algorithm's fourth characteristic, noise and measurement restriction removal, all $\bar{w}$ and $\bar{\eta}_*(t_k)$ values are set to null in all dynamics and measurement models. Furthermore, GPSD is removed, and line-of-sight measurements are allowed to be processed even after the forward-facing camera passes a landmark. Doing so, allows the navigation state dynamics and measurement models to always receive perfect knowledge of truth state values. Hence, propagating the navigation state using these truth state values allows one to analyze underlying dynamics more clearly and analyze the fidelity of the navigation state dynamics itself without the influence of system noise or measurement restrictions. Note, magnetometer jamming is not analyzed in this document.

Executing the propagation validation simulation with these given characteristics, Figure 2.3 through Figure 2.19 display the outcomes. Error plots are the differences between the truth state and the error state (generated with the $\bar{n}$ mapping) and residual plots are the differences between true discrete measurements (Equation (2.6)) and navigation discrete measurements (Equation (2.27)), i.e., $\tilde{\bar{z}} - \hat{\tilde{\bar{z}}}$.

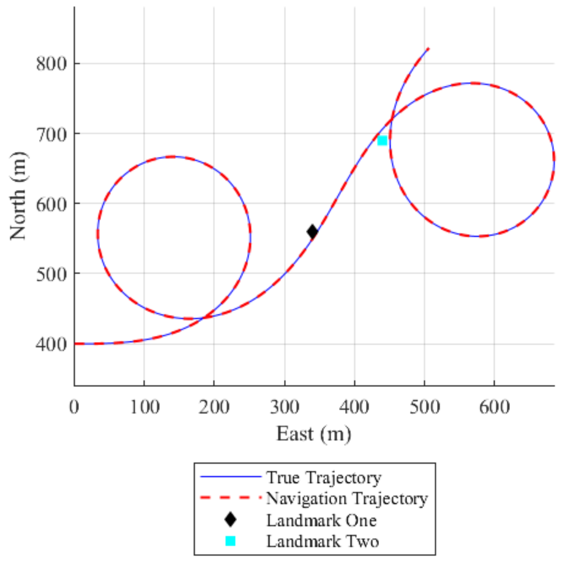


Figure 2.3. Propagation validation of UAV north and east coordinates.

Figure 2.3 shows the UAV’s trajectory with respect to 2D space as it meanders by two landmarks (located at 560 N, 340 E and 690 N, 440 E). With max, 90° rudder input, the UAV can achieve a minimum turn radius of approximately 100 m if traveling at a constant velocity of 16 m/s, which is a similar result found in previously generated 3D simulations. With 90° rudder inputs being unrealistic, an alternative way for achieving this 3D minimum-turn result in a 2D simulation would be to alter the aerodynamic C_* coefficients.

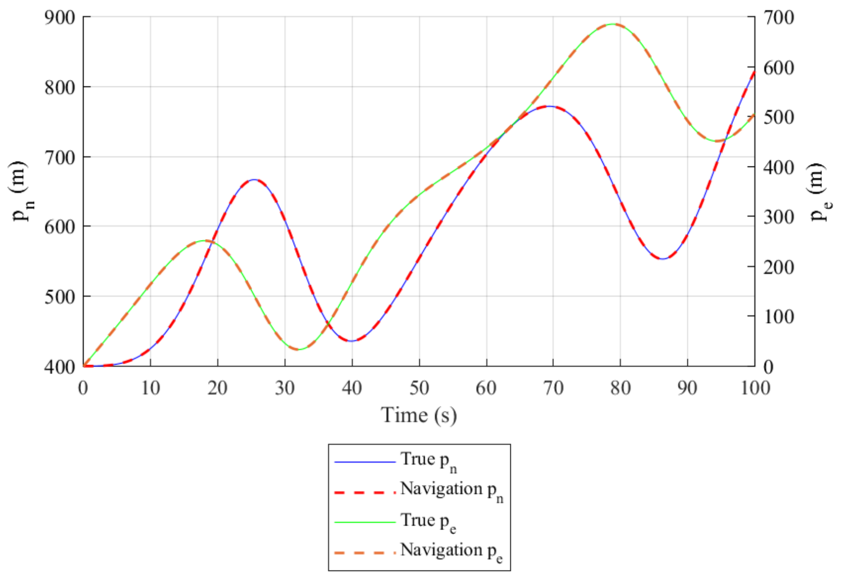


Figure 2.4. Propagation validation of UAV trajectory over time.

The same trajectory with respect to time is presented in within Figure 2.4. Note that double y-axes are used on this and future plots. The two pairs of crests and troughs in the figure correspond to the two loops formed in Figure 2.3. Overall, the UAV navigation state position appears to follow the truth state position well.

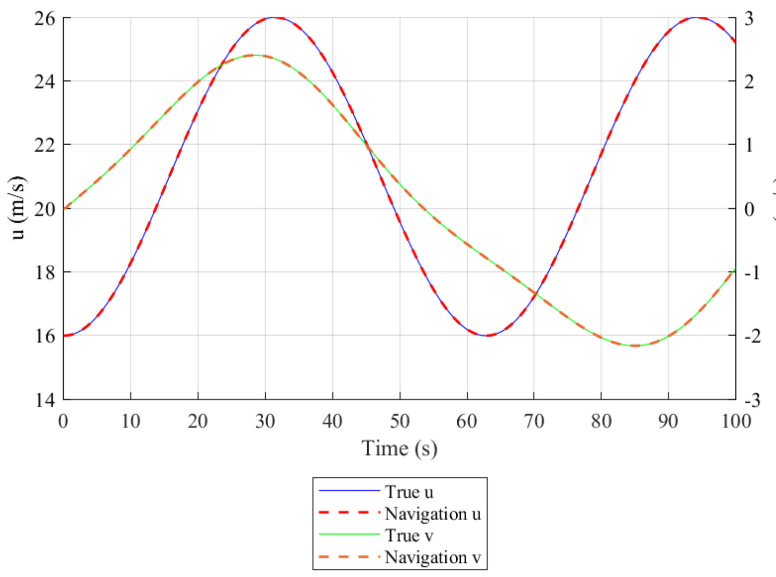


Figure 2.5. Propagation validation of UAV velocity in Body frame over time.

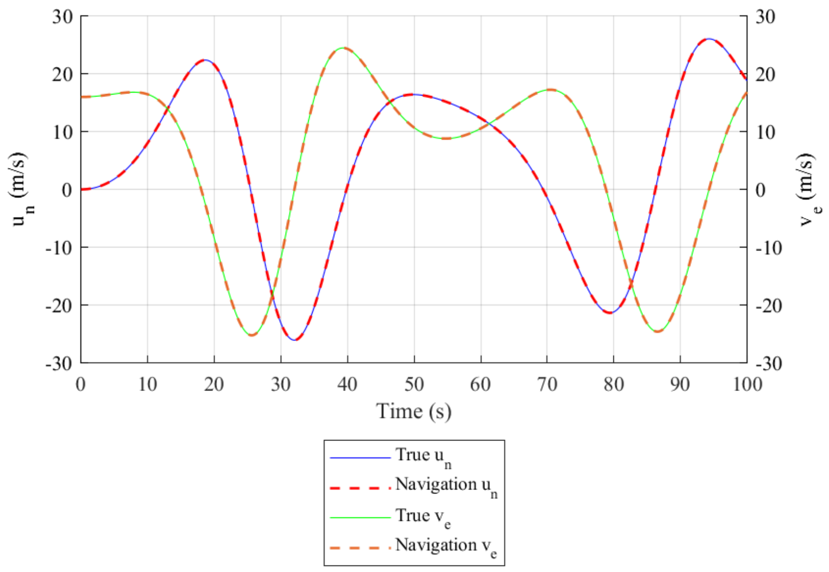


Figure 2.6. Propagation validation in N-E frame over time.

Figure 2.5 and Figure 2.6 provide the UAV's velocity with respect to the Body and N-E frame. The stark difference between the slopes in these two graphs help illustrate the reason why propagating truth and navigation velocities in different coordinate frames using the Euler method on the assumed dynamics can be troublesome. That is, the differing slopes of Body velocities and North-East velocities at a given point, skews the Euler projection. Assuming an approximate maximum velocity of 25 m/s [31], this trajectory's max forward velocity is slightly larger, providing an extreme case to induce larger velocity error into the system.

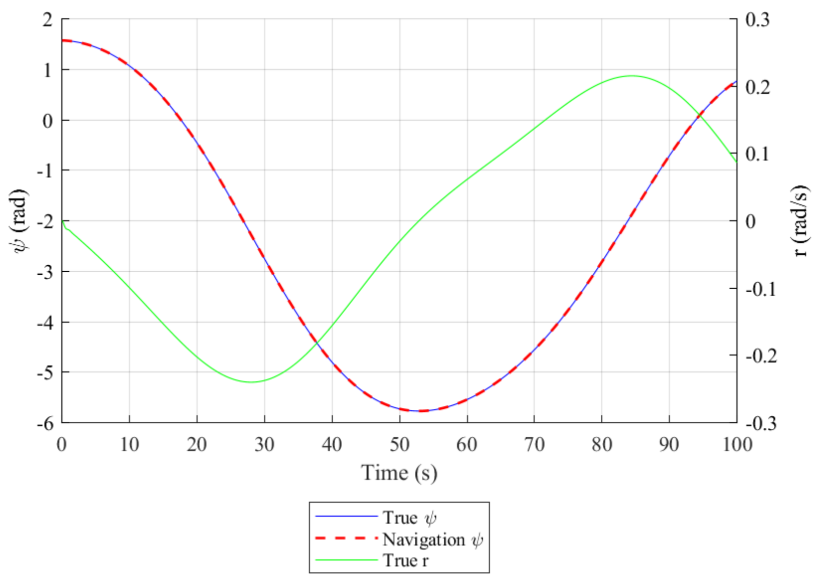


Figure 2.7. Propagation validation of UAV attitude and attitude rate over time.

In terms of attitude, both states have heading; both appear to follow each other well in Figure 2.7. The navigation state has no heading rate to compare with the truth state.

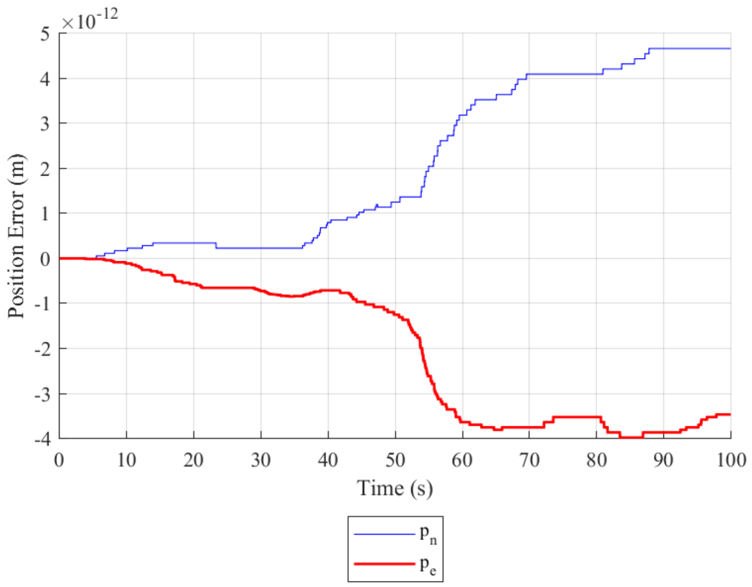


Figure 2.8. Propagation validation of UAV position error over time.

Looking more finely at errors, Figure 2.8 shows that max position discrepancies between the navigation state and truth state are just under 5 pm. Realistically, with GPS errors being on the order of meters, such discrepancies suffice.

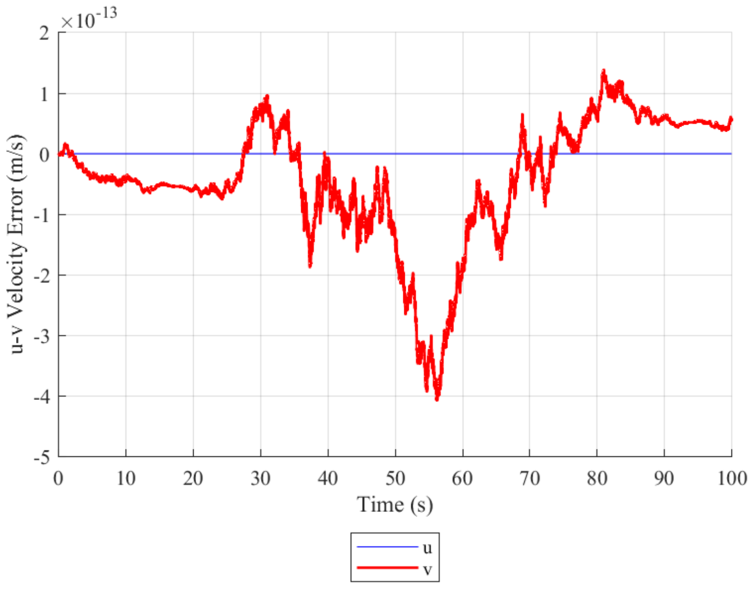


Figure 2.9. Propagation validation of UAV Body frame velocity error over time.

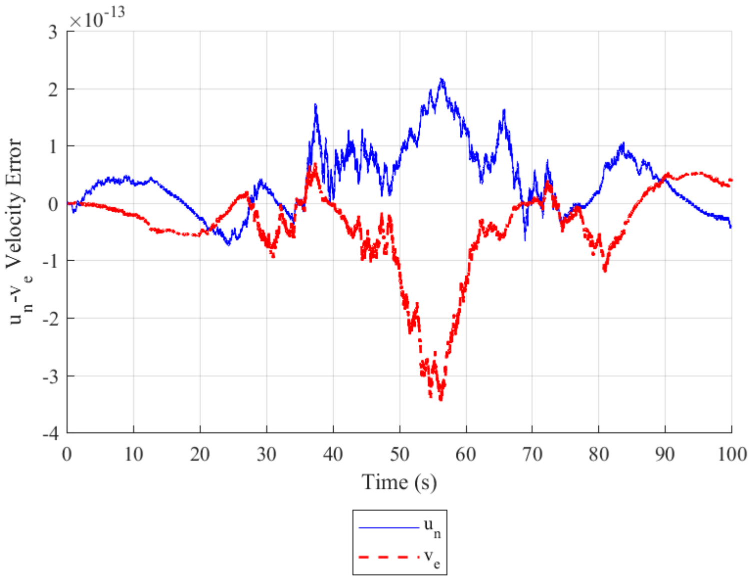


Figure 2.10. Propagation validation of UAV North-East frame velocity error over time.

With position being the integral of velocity, position errors are a recipient of velocity error`s. Figure 2.9 and Figure 2.10 alludes to this, where velocity errors are an order of magnitude smaller than position errors. Figure 2.9 has a unique characteristic where the u velocity error is essentially null. The reasoning for such characteristic is from the measurement synthesis. If periods of large heading rates had occurred while the v component had significant error accumulation, u errors would also have been introduced as the navigation state attempted to account for the Coriolis term. This trajectory did not yield large enough heading rates and lateral velocity simultaneously to introduce such errors.

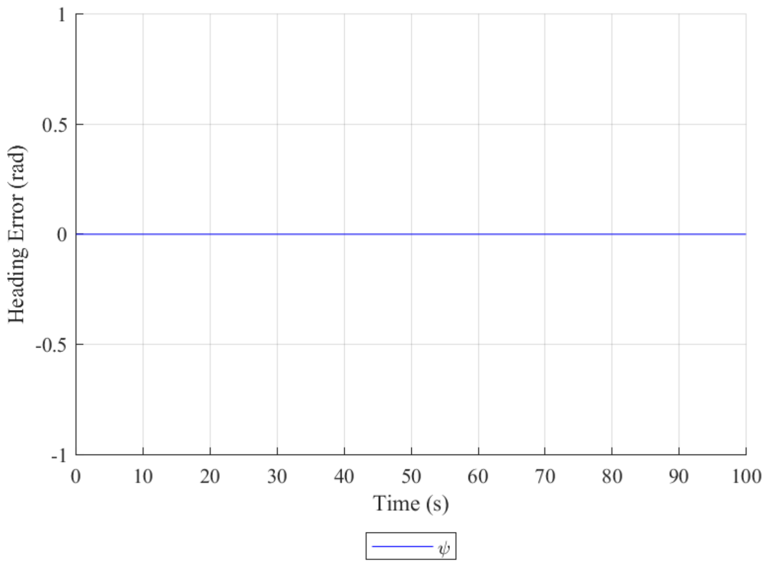


Figure 2.11. Propagation validation of UAV heading error over time.

The errors associated with heading are null in Figure 2.11, indicating that the navigation dynamics can perfectly match truth state heading despite the use of measurement synthesis. Since Figure 2.10 is derived using transformation matrices that use the heading state, zero heading error implies that Figure 2.10 is only erroneous due to velocity errors, not heading errors. However, this implication will not hold in future simulations. In the 2.1.6 Covariance Propagation and later subsections, assuming heading errors do not affect velocity covariance is wrong—errors are due to both.

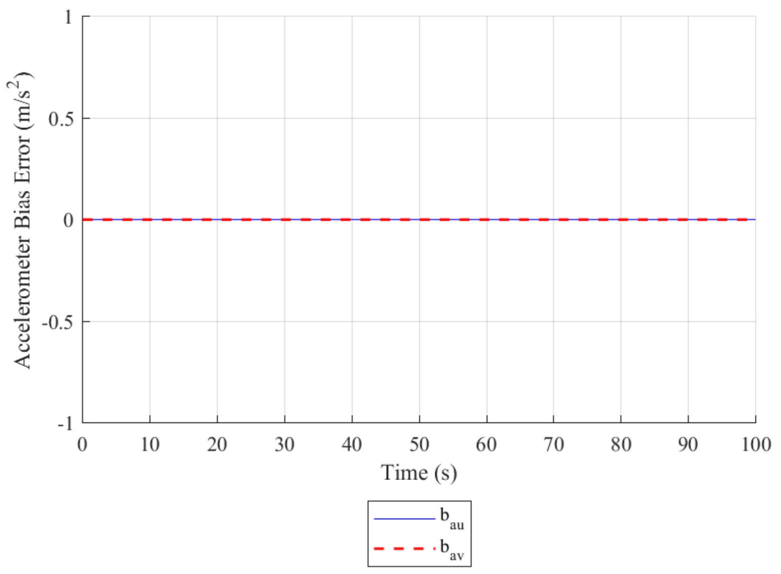


Figure 2.12. Propagation validation of UAV accelerometer bias over time.

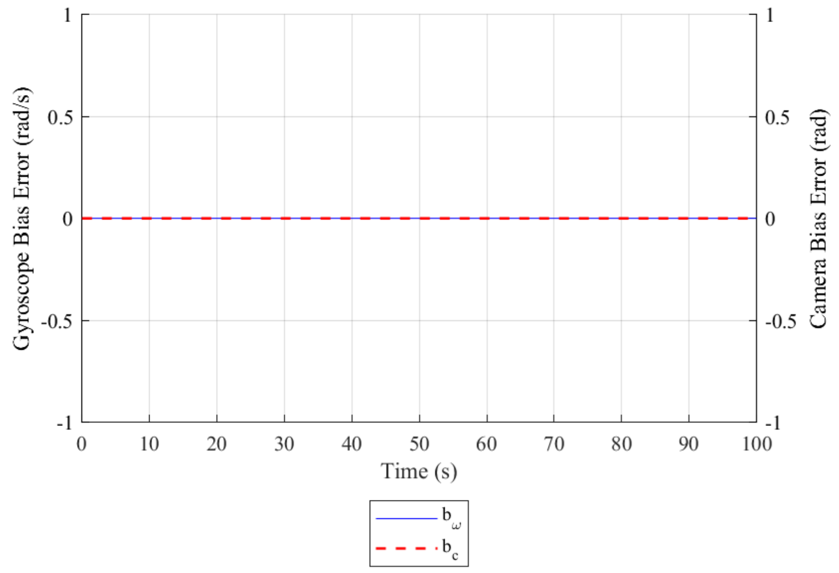


Figure 2.13. Propagation validation of UAV gyroscope bias and camera bias over time.

The bias error results in Figure 2.12 and Figure 2.13 are all null for all time. In other words, Euler integration on these first-order differential equations (when noise is set to null) match exactly in both the truth state and navigation state.

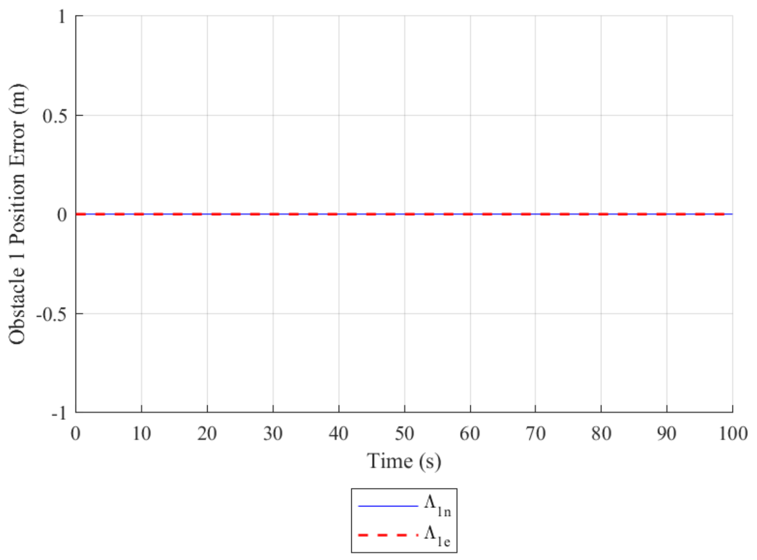


Figure 2.14. Propagation validation of Landmark 1’s position error.

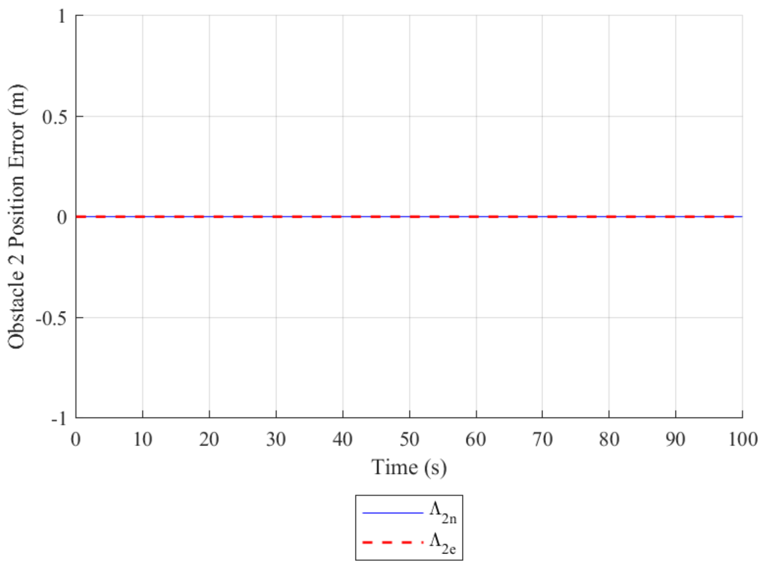


Figure 2.15 Propagation validation of Landmark 2’s position error.

Figure 2.14 and Figure 2.15 describe, because landmark position in this research does not change over time, that no errors incur due to propagation.

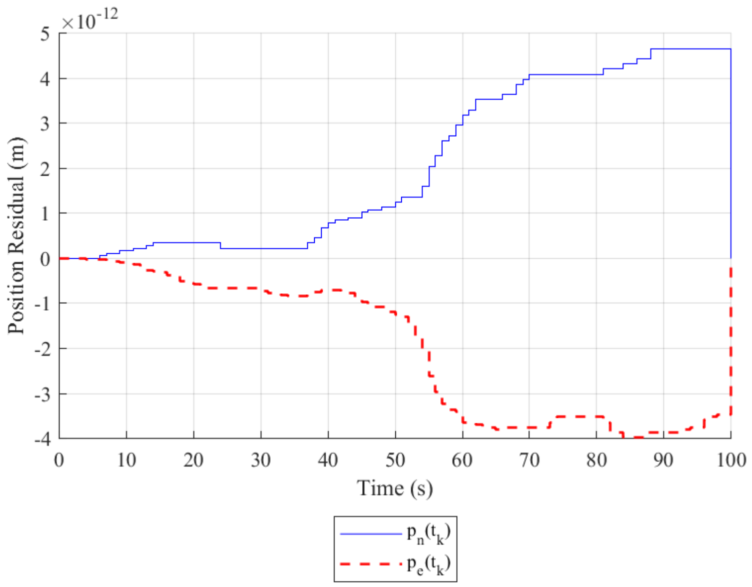


Figure 2.16. Propagation validation of UAV GPS position residuals over time.

The GPS position measurement residuals shown in Figure 2.16 discretely pattern the errors in Figure 2.8, implying that discrepancies between true measurements and navigation measurements are essentially due to the original errors between truth state dynamics and navigation state dynamics.

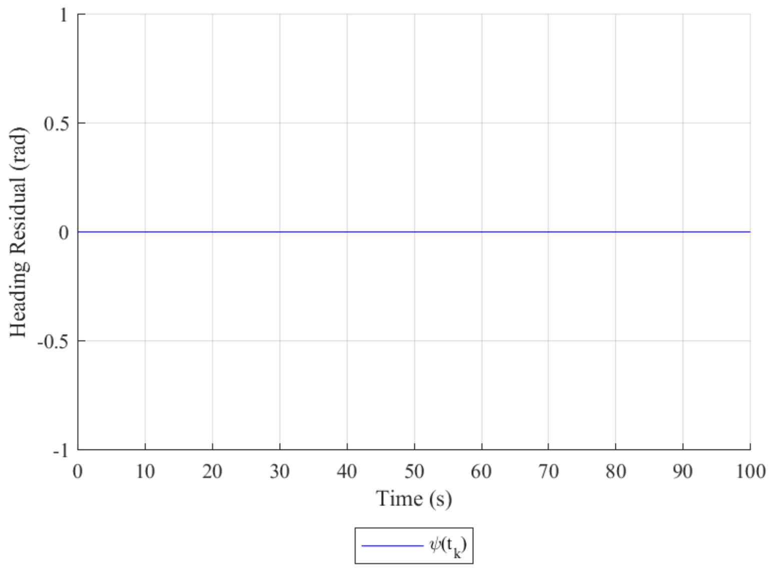


Figure 2.17. Propagation validation of UAV magnetometer residuals over time.

The heading residuals of Figure 2.17, like heading errors, are null, indicating that implementation of heading measurements is correct.

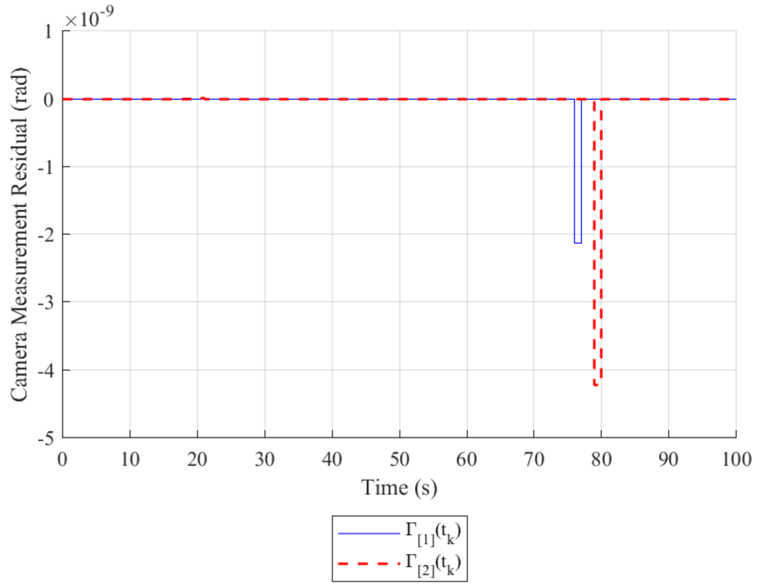


Figure 2.18. Propagation validation of UAV camera measurement residuals over time.

In Figure 2.18, Camera measurement errors have a significantly larger error spike later in the simulation. In fact, error spikes occur throughout the simulation, though negligible in size. The spikes do not necessarily align with the previously seen states or errors. Rather, the spikes align to measurements when the $\bar{\Gamma}(t_k)$ ratios are extreme (e.g when $\Lambda_{1j}(t_k)$ is significantly larger than $\Lambda_{1i}(t_k)$—analogous to image distortion as an object moves further from center in a camera picture [9]). The largest error spike occurs when the vehicle is on the far-right side of Figure 2.3, a position out of the camera's realistic FOV (i.e., the erroneous measurement would not have been received).

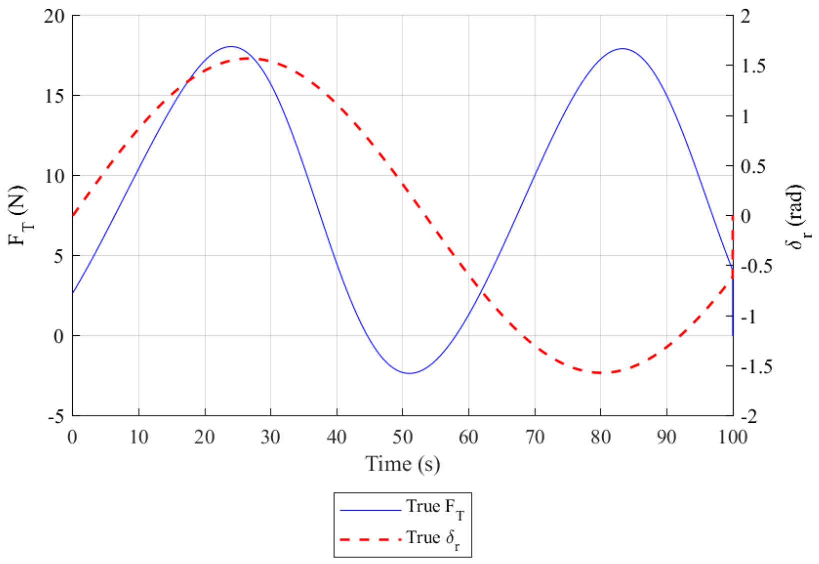


Figure 2.19. Propagation validation of UAV control input.

As aforementioned, the inputs for this propagation validation step were made sinusoidal. Such inputs allowed the vehicle to experience input extremes, even negative thrust. Note that noise-corrupted control inputs will not be considered in this study and is left for future work.

Based on the negligible discrepancies within all these plot results, when noise sources and measurement restrictions are added in future simulations, significant errors that occur will be assumed to not be attributed to incorrect model development.

2.1.4 Linear Error State Modeling

Up to this point, states have been defined, mappings between those states have been developed, and validation that the states are consistent between one another has been confirmed. Moving forward, there are four key equation that are used within the IEKF [3]. The first equation is Equation (2.25)—The navigation state dynamics, $\dot{\hat{\bar{x}}} = \hat{\bar{f}}\left(\hat{\bar{x}}, \hat{\bar{\bar{y}}}\right)$. The remaining three are as follows:

$$\dot{\hat{\bar{P}}} = \breve{\bar{F}}_{\breve{\bar{X}}}(\hat{\bar{x}})\hat{\bar{P}} + \hat{\bar{P}}\breve{\bar{F}}^{T}_{\breve{\bar{x}}}(\hat{\bar{x}}) + \breve{\bar{G}}(\hat{\bar{x}})\breve{\bar{Q}}(\hat{\bar{x}})\ \breve{\bar{G}}^{T}(\hat{\bar{x}}) \tag{2.40}$$

$$\delta\hat{\bar{x}}^{+} = \hat{\hat{K}}\left\{\tilde{\bar{z}} - \hat{\bar{\bar{z}}}^{-}\right\} \tag{2.41}$$

$$\begin{aligned} \hat{\bar{P}}^{+} = {} & \left\{\hat{I} - \hat{\hat{K}}\breve{\bar{H}}_{\breve{\bar{x}}}\left(\hat{\bar{x}}^{-}(t_k)\right)\right\}\hat{\bar{P}}^{-}\left\{\hat{I} - \hat{\hat{K}}\breve{\bar{H}}_{\breve{\bar{x}}}\left(\hat{\bar{x}}^{-}(t_k)\right)\right\}^{T} \\ & + \hat{\hat{K}}\breve{J}(\hat{\bar{x}})\breve{\bar{R}}(\hat{\bar{x}})\breve{J}^{T}(\hat{\bar{x}})\hat{\hat{K}}^{T} \end{aligned} \tag{2.42}$$

In broad explanation, Equation (2.25) generates the “best” (i.e., mean and mode) estimate of the UAV state. Equation (2.40) generates the covariance estimate associated with that state estimate. Equation (2.41) generates an update as to what the current state errors are. Equation (2.42) generates an update as to what the current covariance is. The following paragraph will provide a finer explanation to Equation (2.40). The 2.1.5 Linear Error Measurement Modeling subsection will provide more explanation to the latter two equations.

As aforementioned, an IEKF is a statistics algorithm that attempts to determine a better estimate to a system’s navigation state and provides covariances associated with that navigation state. To do this approximation, the filter relies on linearization of the design model dynamics and design measurement model. The terms generated from these linearizations are many of the terms used in Equation (2.40) through Equation (2.42). Deriving the dynamics linearization ([3,24]) as well providing a validation algorithm for correct implementation will now be discussed.

The truth states, design model states, and navigation state have previously been defined with Equation (2.1), Equation (2.14), and Equation (2.24). Their nonlinear dynamics have also been defined as

$$\dot{\bar{X}} = \bar{f}(\bar{X}, \bar{U}, \bar{w}) \tag{2.43}$$

$$\dot{\breve{\bar{x}}} = \breve{\bar{f}}\left(\breve{\bar{x}}, \breve{\bar{\bar{y}}}, \breve{\bar{w}}\right) = \breve{\bar{f}}\left(\breve{\bar{x}}, \breve{\bar{\bar{y}}}\right) + \breve{\bar{\bar{G}}}(\breve{\bar{x}})\breve{\bar{w}} \tag{2.44}$$

$$\dot{\hat{\bar{x}}} = \hat{\bar{f}}\left(\hat{\bar{x}}, \hat{\bar{\bar{y}}}\right) \tag{2.45}$$

(Equation (2.3), Equation (2.15), and Equation (2.25)). Process noise, $\breve{\bar{w}} = [\breve{\eta}_{a_u} \quad \breve{\eta}_{a_v} \quad \breve{\eta}_{\omega} \quad \breve{w}_{a_u} \quad \breve{w}_{a_v} \quad \breve{w}_{\omega} \quad \breve{w}_c]^T$, is separated from other terms in Equation (2.44) as the accompanying matrix to allow this separation, $\breve{\bar{\bar{G}}}(\breve{\bar{x}})$, will be needed in Equation (2.40). Being able to access $\breve{\bar{\bar{G}}}(\breve{\bar{x}})$ is why derivation evaluation in terms of the design model is used, unlike the navigation model, which does not allow the engineer to retain noise terms.

$$\breve{\bar{\bar{G}}}(\hat{\bar{x}}) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.46}$$

Although only a matrix of constants for this simulation, other simulations may yield more complicated components in $\breve{\bar{G}}(\hat{\bar{x}})$, e.g., when transformation matrices are manipulated on noise-bearing states. When flying in real time, the UAV has no access to ideal $\breve{\bar{x}}$ variable values; hence, $\breve{\bar{G}}$, called the noise coupling matrix, would need to be evaluated using accessible navigation state terms—$\breve{\bar{G}}(\hat{\bar{x}})$:

Returning to the derivation, the nonlinear truth dynamics can be reformatted into their linear equivalent using the method of first-order Taylor series expansion, as the following equation shows:

$$\dot{\bar{x}} \approx \left\{\breve{\bar{f}}\left(\hat{\bar{x}}, \hat{\bar{\bar{y}}}\right) + \breve{\bar{G}}(\hat{\bar{x}})\bar{\breve{w}}\right\} + \breve{\bar{\bar{F}}}_{\breve{\bar{x}}}(\hat{\bar{x}})\delta\bar{x} + H.O.T \tag{2.47}$$

Making this linear equation in terms of the design model, evaluated with navigation terms as the nominal, the truth state consequently will have heading rate removed in this derivation, hence the use of $\dot{\bar{x}}$, not $\dot{\bar{X}}$. Equation (2.47) states that the reduced truth state dynamics is approximately equal to the design model evaluated at the navigation state plus the design model's Jacobian matrix, $\breve{\bar{\bar{F}}}_{\breve{\bar{x}}}(\hat{\bar{x}})$ (i.e. $\left.\frac{\partial \breve{\bar{f}}\left(\breve{\bar{x}}, \breve{\bar{\bar{y}}}\right)}{\partial \breve{\bar{x}}}\right|_{\hat{\bar{x}}}$; breve above $\bar{F}$ to indicate that it is the partial of $\breve{\bar{f}}\left(\breve{\bar{x}}, \breve{\bar{\bar{y}}}\right)$; subscript $\breve{\bar{x}}$ to indicate that the partial is with respect to the design model state; $(\hat{\bar{x}})$ operator to indicate evaluation using navigation model terms, just like $\breve{\bar{G}}(\hat{\bar{x}})$), which is multiplied by the perturbation discrepancy between the truth and navigation model states, $\delta\bar{x}$. A measurement term, $\breve{\bar{\bar{F}}}_{\breve{\bar{y}}}(\hat{\bar{x}})\delta\breve{\bar{\bar{y}}}$, does not appear in the expansion since $\breve{\bar{\bar{y}}}$ in $\breve{\bar{f}}\left(\breve{\bar{x}}, \breve{\bar{\bar{y}}}\right)$ has become a deterministic input (i.e. its noise has been separated out into $\breve{\bar{G}}(\hat{\bar{x}})\bar{\breve{w}}$) and thus there is no error $\delta\breve{\bar{\bar{y}}}$ within $\breve{\bar{f}}\left(\breve{\bar{x}}, \breve{\bar{\bar{y}}}\right)$. Noise term

$\left.\frac{\partial\{\breve{\bar{\bar{G}}}(\breve{\bar{x}})\breve{\bar{w}}\}}{\partial\breve{\bar{x}}}\right|_{\hat{\bar{x}}}\delta\bar{x}$ becomes null since evaluation at the navigation state has $\hat{\bar{w}} = \bar{0}$. Noise term

$\left.\frac{\partial\{\breve{\bar{\bar{G}}}(\breve{\bar{x}})\breve{\bar{w}}\}}{\partial\breve{\bar{w}}}\right|_{\hat{\bar{x}}}\delta\bar{w}$ is grouped with higher order terms from the Taylor expansion, $H.O.T.$

In addition to the Taylor expansion, the truth state dynamics can be represented in yet another format. Similar to error state mappings, the truth and navigation model states are assumed to be related via

$$\bar{x} = \hat{\bar{x}} + \delta\bar{x} \tag{2.48}$$

Which implies that

$$\dot{\bar{x}} = \dot{\hat{\bar{x}}} + \delta\dot{\bar{x}} \tag{2.49}$$

If Equation (2.49) is substituted into the left-hand-side of Equation (2.47), canceling out $\dot{\hat{\bar{x}}}$ with $\breve{\bar{f}}(\hat{\bar{x}}, \hat{\bar{\bar{y}}})$ and assuming that $H.O.T$ are negligible yields that

$$\delta\dot{\bar{x}} \approx \breve{\bar{\bar{F}}}_{\breve{\bar{x}}}(\hat{\bar{x}})\delta\bar{x} + \breve{\bar{\bar{G}}}(\hat{\bar{x}})\breve{\bar{w}} \tag{2.50}$$

Equation (2.50), called the linear state perturbation model or linear error state model, up to a first-order approximation, determines the rate of state errors over time; and its Jacobian term, along with $\breve{\bar{\bar{G}}}(\hat{\bar{x}})$, the current covariance, $\hat{\bar{\bar{P}}}$, and the $\breve{\bar{\bar{Q}}}(\hat{\bar{x}})$ power spectral density matrix ($\breve{\bar{\bar{Q}}}(\hat{\bar{x}})$ will be discussed more in the 2.1.6 Covariance Propagation subsection) are used to calculate covariance rates, $\dot{\hat{\bar{\bar{P}}}}$, in Equation (2.40). In other words, $\dot{\hat{\bar{\bar{P}}}}$ will track the covariances associated with the propagated navigation state throughout the entire simulation. The Jacobian term for this research (again, with design model variables substituted with navigation model values) has been determined to be

$$\breve{\bar{F}}_{\breve{x}}(\hat{\bar{x}}) = \begin{bmatrix} 0 & 0 & \{\cos(\hat{\psi})\} & \{-\sin(\hat{\psi})\} & \hat{\Xi}_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \{\sin(\hat{\psi})\} & \{\cos(\hat{\psi})\} & \hat{\Xi}_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \hat{\omega} & 0 & -1 & 0 & -\hat{v} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\hat{\omega} & 0 & 0 & 0 & -1 & \hat{u} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \left\{\frac{-1}{\hat{\tau}_{a_u}}\right\} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \left\{\frac{-1}{\hat{\tau}_{a_v}}\right\} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \left\{\frac{-1}{\hat{\tau}_{\omega}}\right\} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \left\{\frac{-1}{\hat{\tau}_{c}}\right\} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad (2.51)$$

where braces are used to denote multiplication, separation between a set of terms, and to help prevent confusion between the parentheses used in function operators. Furthermore,

$$\begin{bmatrix} \hat{\Xi}_3 \\ \hat{\Xi}_4 \\ \hat{\omega} \end{bmatrix} = \begin{bmatrix} -\hat{v}\cos(\hat{\psi}) - \hat{u}\sin(\hat{\psi}) \\ \hat{u}\cos(\hat{\psi}) - \hat{v}\sin(\hat{\psi}) \\ \tilde{\omega} - \hat{b}_{\omega} \end{bmatrix} \qquad (2.52)$$

Again, the reason for this derivation, is to acquire $\breve{\bar{G}}(\hat{\bar{x}})$ and $\breve{\bar{F}}_{\breve{x}}(\hat{\bar{x}})$ for the Kalman filter's key equations. Though there is difficulty in validating correct derivation of $\breve{\bar{G}}(\hat{\bar{x}})$ on its own (see 2.1.6 Covariance Propagation subsection), $\breve{\bar{F}}_{\breve{X}}(\hat{\bar{x}})$, however, can be validated with Equation (2.50). To do this, one can compare the linear error state propagation results in the linear error sate model with the true state error $\bar{n}$ mapping and determine whether the linear error state sufficiently matches the true state error. The rest of this subsection will describe the developed validation algorithm that is used to confirm this sufficient matching.

In the linear error state modeling validation algorithm, the four simulation characteristics of the propagation validation algorithm are used. Moreover, all time constants are manipulated to equal 1/10th of the discrete measurement time increments. Doing so ensures that the bias dynamics are exercised (i.e., magnified or stressed) and tests whether the linear error state model can model errors for extreme linear differential equations.

The procedure for the linear error state modeling validation algorithm is as follows: First, with both truth and navigation states starting out at the same initial conditions, a known perturbation error matrix, $\delta\bar{x}$ is injected into navigations state via the $\bar{m}$ mapping. Thus, the truth and navigation states are no longer the same, but their initial errors, as calculated in the $\bar{n}$ mapping, are the same as the initial state of the linear error state model. Second, the truth, the navigation, and the linear error state model are propagated forward in time. Propagation was chosen to stop at the first discrete time measurement (1 s). Stopping at this time is somewhat arbitrary, nevertheless, it provides sufficient vehicle movement history that is worth comparing (the vehicle has already moved ~16 m) and errors are assumed to have not accumulated above that which the first-order approximation can account for. Furthermore, after 1 s increments, the Kalman filter has been assigned to correct those errors, which means, theoretically, the linearization only needs to be valid for one second cycles since the Kalman filter discrete measurement updates will correct errors afterwards. Finally, the propagated errors of the linear error state model are compared to the errors of the $\bar{n}$ mapping at the first discrete time measurement. Table 2.3 provides the results from this validation procedure.

Table 2.3. Linear error state validation.

State	Injected Errors	$\delta\bar{x}_{linear}$	$\delta\bar{x}_{nonlinear}$	Propagation Difference	Units
p_n	2.0e+00	1.7e+00	1.7e+00	7.9e-04	m
p_e	3.0e+00	3.2e+00	3.2e+00	1.4e-03	m
u_n	2.0e-01	1.6e-01	1.6e-01	3.8e-06	$\mathrm{m/s}$
v_e	3.0e-01	2.5e-01	2.5e-01	-2.5e-06	$\mathrm{m/s}$
ψ	4.7e-03	4.7e-03	4.7e-03	-6.5e-16	rad
b_{a_u}	3.7e-01	9.9e-06	9.9e-06	1.7e-21	$\mathrm{m/s^2}$
b_{a_v}	5.6e-01	1.5e-05	1.5e-05	-1.5e-20	$\mathrm{m/s^2}$
b_ω	1.4e-04	3.7e-09	3.7e-09	-4.1e-25	$\mathrm{rad/s}$
b_c	1.1e-02	2.9e-07	2.9e-07	-2.1e-22	rad
Λ_{1n}	10.0e+00	10.0e+00	10.0e+00	0.0e+00	m
Λ_{1e}	10.0e+00	10.0e+00	10.0e+00	0.0e+00	m
Λ_{2n}	15.0e+00	15.0e+00	15.0e+00	0.0e+00	m
Λ_{2e}	15.0e+00	15.0e+00	15.0e+00	0.0e+00	m

Table 2.3's first numeric column provides the selected errors that were injected into the navigation state. These errors are 2 to 3 times that of the 3σ initial (commercial grade) state uncertainty, as will be described in the 2.1.6 Covariance Propagation and 2.1.7 Estimation Capability subsections, except for the landmarks, which used one time the initial 3σ uncertainty or less. The second numeric column shows the errors as determined by the linear error state model at the end of propagation. Third are the errors in terms of the true, nonlinear mapping at the end of propagation. The differences between these two propagation columns are provided in the last numeric column. Both the linear propagations and nonlinear mappings retain the same magnitude of error as the initial injected error, with the exception being the biases, which actually decrease errors by at least four orders of magnitude (i.e., the linear error state model tracks linear dynamics with ease). In terms of propagation difference, UAV position error differences are on the order of mm or less. UAV velocity error differences are also on the order of μm/s. Heading error differences are essentially numerically negligible. More so

negligible are the bias propagation error differences. Landmarks, having zero dynamics, retain zero error differences. Compared to the previous sections, the errors associated with position and velocity are significantly higher; nevertheless, the first-order approximation retain position error differences well below GPS errors and retain velocity error differences well below industry velocimeters errors. Of more concern, perhaps, are the landmark position errors. Though the results initially imply that even 15m of error will be perfectly accounted for, the results implicitly indicate that this error matrix for continuous measurements cannot account for real-time landmark errors—the linear error state model can only assume that landmark errors at the end of the simulation will be the exact same as those at the beginning of the simulation. An alternative matrix, such as the discrete measurement matrix derived in the 2.1.5 Linear Error Measurement Modeling subsection, will need to be used to account for errors in landmark positions. For the purpose of accounting for errors based on the continuous dynamics, however, $\breve{\bar{\bar{F}}}_{\breve{\bar{x}}}(\hat{\bar{x}})$ appears to adequately provide this accounting, even for extremely large errors, and should thus be sufficient for the Kalman filter's key equations.

2.1.5 Linear Error Measurement Modeling

Much of the Kalman filter development up to this point has been with regards to propagating state estimates and state covariance. This propagation process is associated with continuous measurements. Alternatively, the update process of the Kalman filter is associated with the system discrete measurements. In this update process, the Kalman filter decides the legitimacy of discrete measurements. After collecting true, raw discrete measurements, the Kalman algorithm compares these measurements to what the navigation state guessed these discrete measurements (of the a priori state) to be—this is the residual comparison that was performed in the 2.1.3 Propagation Validation subsection, i.e.

$$\tilde{\bar{z}} - \hat{\bar{z}}^{-} = \begin{bmatrix} p_n(t_k) + \eta_{p_n}(t_k) \\ p_e(t_k) + \eta_{p_e}(t_k) \\ \psi(t_k) + \eta_{\psi}(t_k) \\ \bar{\Gamma}(t_k) + \bar{\eta}_{\bar{\Gamma}}(t_k) \end{bmatrix} - \begin{bmatrix} \hat{p}_n(t_k) \\ \hat{p}_e(t_k) \\ \hat{\psi}(t_k) \\ \hat{\bar{\Gamma}}(t_k) \end{bmatrix} \tag{2.53}$$

Using a statistical weighting matrix called the Kalman gain, $\hat{\bar{K}}$, on the residual, the filter will place higher favoring weight on measurements residuals that are more likely (e.g., low Kalman gains are applied to residuals, where the raw measurement is outside its 3σ measurement range). In this manner, the filter generates posteriori statistical guess adjustments, updates, as to what the errors are between the navigation state and the true state of the vehicle—This is third key Kalman equation, Equation (2.41), where "+" superscripts indicate posteriori and "-" superscripts indicate a priori. These errors can then be applied to the navigation state via the $\bar{l}$ mapping in an attempt to make the navigations state equal to the truth state (see 2.1.7 Estimation Capability subsection). For clarity, note that this text has residuals = innovations, as opposed to other texts which classify residuals as $\tilde{\bar{z}} - \hat{\bar{z}}^{+}$.

Just as Equation (2.41) provides updates to the state estimates, Equation (2.42) provides updates to the state covariance. An alternative, simpler form to this equation is

$$\hat{\bar{P}}^{+} = \hat{\bar{P}}^{-} - \hat{\bar{K}}\breve{\bar{H}}_{\breve{x}}\left(\hat{\bar{x}}^{-}(t_k)\right)\hat{\bar{P}}^{-} \tag{2.54}$$

Equation (2.54) shows that adjusted update covariance is equal to the a priori covariance plus a multiple composed of the Kalman gain weight, the measurement geometry matrix, $\breve{\bar{H}}_{\breve{x}}\left(\hat{\bar{x}}^{-}(t_k)\right)$, and the a priori covariance. Equation (2.54) has the same form as Equation (2.41), where the update is basically the current covariance state plus a weighting on the current covariance state (note that $\delta\hat{\bar{x}}^-$ does not show up in Equation (2.41) because estimated state error prior to the measurement update is guessed to be null). The reason for defaulting to Equation (2.42), called the Joseph form, rather than Equation (2.54) for implementation is that the Joseph form has desirable numerical properties in terms of retaining symmetry, retaining positive definiteness, handling errors, and code manipulation (e.g. the capability of zeroing out Kalman gains, which can force the filter to reject measurements).

The terms of the Joseph form covariance update equation need to be identified. Though $\hat{\bar{\bar{K}}}$ and $\breve{\bar{\bar{R}}}(\hat{\bar{x}})$ are left to be discussed in the 2.1.7 Estimation Capability subsection, this subsection will describe $\hat{\bar{\bar{I}}}$, $\breve{\bar{\bar{J}}}(\hat{\bar{x}})$, and $\breve{\bar{\bar{H}}}_{\breve{\bar{x}}}\left(\hat{\bar{x}}^-(t_k)\right)$. $\hat{\bar{\bar{I}}}$ is simply the identity matrix and is the same size as $\hat{\bar{\bar{P}}}$ (same length as $\bar{x}$). $\breve{\bar{\bar{J}}}(\hat{\bar{x}})$ is a separation matrix like $\breve{\bar{\bar{G}}}(\hat{\bar{x}})$. $\breve{\bar{\bar{H}}}_{\breve{\bar{x}}}\left(\hat{\bar{x}}^-(t_k)\right)$ is the measurement geometry matrix obtained in the same manner as $\breve{\bar{\bar{F}}}_{\breve{\bar{x}}}(\hat{\bar{x}})$—linearization. Deriving $\breve{\bar{\bar{J}}}(\hat{\bar{x}})$ and $\breve{\bar{\bar{H}}}_{\breve{\bar{x}}}\left(\hat{\bar{x}}^-(t_k)\right)$ as well as validating $\breve{\bar{\bar{H}}}_{\breve{\bar{x}}}\left(\hat{\bar{x}}^-(t_k)\right)$ will now be explained.

Deriving $\breve{\bar{\bar{H}}}_{\breve{\bar{x}}}\left(\hat{\bar{x}}^-(t_k)\right)$, similar to $\breve{\bar{\bar{F}}}_{\breve{\bar{x}}}(\hat{\bar{x}})$, recall that true discrete measurements, design model discrete measurements, and navigation model discrete measurements have been defined as

$$\tilde{\bar{\bar{z}}} = \bar{h}\left(\bar{X}^-(t_k), \bar{\eta}(t_k)\right) \tag{2.55}$$

$$\breve{\bar{\bar{z}}} = \breve{\bar{h}}\left(\breve{\bar{x}}^-(t_k), \breve{\bar{\eta}}(t_k)\right) = \breve{\bar{h}}\left(\breve{\bar{x}}^-(t_k)\right) + \breve{\bar{\bar{J}}}(\hat{\bar{x}})\breve{\bar{\eta}}(t_k) \tag{2.56}$$

$$\hat{\bar{\bar{z}}} = \hat{\bar{h}}\left(\hat{\bar{x}}^-(t_k)\right) \tag{2.57}$$

(Equation (2.6), Equation (2.17), and Equation (2.27)) The a priori superscript is now included in these definitions for completeness. Classifying discrete measurement noise with higher order terms, $\breve{\bar{\eta}}(t_k)$ is separated out of $\breve{\bar{h}}\left(\breve{\bar{x}}^-(t_k), \breve{\bar{\eta}}(t_k)\right)$ using $\breve{\bar{J}}(\hat{\bar{x}})$

$$\breve{\bar{J}}(\hat{\bar{x}}) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.58}$$

Although only an identity matrix for this simulation, other simulations may yield more complicated components in $\breve{\bar{J}}(\hat{\bar{x}})$, for example, when the noise of one sensor influences the noise of another sensor (The author prevented this influence in the third simulation characteristic of measurement synthesis). $(\hat{\bar{x}})$ is maintained with $\breve{\bar{J}}$ to explicitly indicate, should complicated components arise, navigation state values are used. In the 2.1.7 Estimation Capability subsection, this matrix will actually be broken down for numerical stability and higher fidelity estimation purposes.

Formatting $\tilde{\bar{z}}$ into its first-order linear equivalent via Taylor series expansion yields

$$\tilde{\bar{z}} \approx \left\{\breve{\bar{h}}\left(\hat{\bar{x}}^-(t_k)\right) + \breve{\bar{J}}(\hat{\bar{x}})\breve{\bar{\eta}}(t_k)\right\} + \breve{\bar{\bar{H}}}_{\breve{\bar{x}}}\left(\hat{\bar{x}}^-(t_k)\right)\delta\bar{x} \ + H.O.T \tag{2.59}$$

where $\breve{\bar{\bar{H}}}_{\breve{\bar{x}}}\left(\hat{\bar{x}}^-(t_k)\right) = \left.\frac{\partial\breve{\bar{h}}\left(\breve{\bar{x}}^-(t_k)\right)}{\partial\breve{\bar{x}}}\right|_{\hat{\bar{x}}}$. Note $\left.\frac{\partial\breve{\bar{h}}\left(\breve{\bar{x}}^-(t_k)\right)}{\partial\breve{\bar{\eta}}(t_k)}\right|_{\hat{\bar{x}}} \delta\breve{\bar{\eta}}(t_k)$ and $\left.\frac{\partial\breve{\bar{J}}(\hat{\bar{x}})\breve{\bar{\eta}}(t_k)}{\partial\breve{\bar{x}}}\right|_{\hat{\bar{x}}} \delta\bar{x}$ are null, and $\left.\frac{\partial\breve{\bar{J}}(\hat{\bar{x}})\breve{\bar{\eta}}(t_k)}{\partial\breve{\bar{\eta}}(t_k)}\right|_{\hat{\bar{X}}} \delta\breve{\bar{\eta}}(t_k)$ is grouped with $H.O.T$. Alternatively, another form of the truth measurement can be defined via perturbations

$$\tilde{\bar{z}} = \hat{\tilde{\bar{z}}} + \delta\tilde{\bar{z}} \tag{2.60}$$

If Equation (2.60) is substituted into the left-hand-side of Equation (2.59), canceling out $\hat{\tilde{\bar{z}}}$ with $\breve{\bar{h}}\left(\hat{\bar{x}}^-(t_k)\right)$ and assuming that $H.O.T$ are negligible indicates that

$$\delta\tilde{\bar{z}} \approx \breve{\bar{\bar{H}}}_{\breve{\bar{x}}}\left(\hat{\bar{x}}^-(t_k)\right)\delta\bar{x} + \breve{\bar{J}}(\hat{\bar{x}})\breve{\bar{\eta}}(t_k) \tag{2.61}$$

Although the Jacobian of the first three discrete measurements are explicit functions of $\breve{\bar{x}}$, the camera measurements are implicit functions of $\breve{\bar{x}}$, having values of $\breve{\bar{\Lambda}}_{ij}(t_k)$ and not $\breve{\bar{\Lambda}}_{ne}(t_k)$. To account for these implicit measurements, a sub derivation is required:

Recalling the definition of a single truth line-of-sight vector from the 2.1.1 States and Models subsection,

$$\begin{bmatrix} \Lambda_i(t_k) \\ \Lambda_j(t_k) \end{bmatrix} = \bar{T}_B^c \bar{T}_I^B \begin{bmatrix} \Lambda_n(t_k) - p_n(t_k) \\ \Lambda_e(t_k) - p_e(t_k) \end{bmatrix} \tag{2.62}$$

(Equation (2.11)). Like $\tilde{\bar{z}}$, line-of-sight can be approximated with Taylor series, again, using the design model terms but evaluating with navigation values

$$\begin{bmatrix} \Lambda_i(t_k) \\ \Lambda_j(t_k) \end{bmatrix} \approx \begin{bmatrix} \breve{\Lambda}_i(t_k) \\ \breve{\Lambda}_j(t_k) \end{bmatrix}\Bigg|_{\hat{\bar{x}}} + \begin{bmatrix} \dfrac{\partial \breve{\Lambda}_i(t_k)}{\partial \breve{\bar{x}}} \\ \dfrac{\partial \breve{\Lambda}_j(t_k)}{\partial \breve{\bar{x}}} \end{bmatrix}\Bigg|_{\hat{\bar{x}}} \delta\bar{x} + H.O.T \tag{2.63}$$

Assuming that the truth line-of-sight vector can also be defined by a perturbation equation,

$$\begin{bmatrix} \Lambda_i(t_k) \\ \Lambda_j(t_k) \end{bmatrix} = \begin{bmatrix} \hat{\Lambda}_i(t_k) \\ \hat{\Lambda}_j(t_k) \end{bmatrix} + \begin{bmatrix} \delta\Lambda_i(t_k) \\ \delta\Lambda_j(t_k) \end{bmatrix} \tag{2.64}$$

this perturbation equation is substituted into the left-hand-side of the truth line-of-sight Taylor expansion. Using the same cancellation technique as in prior derivation yields

$$\begin{bmatrix} \delta\Lambda_i(t_k) \\ \delta\Lambda_j(t_k) \end{bmatrix} \approx \begin{bmatrix} \dfrac{\partial \breve{\Lambda}_i(t_k)}{\partial \breve{\bar{x}}} \\ \dfrac{\partial \breve{\Lambda}_j(t_k)}{\partial \breve{\bar{x}}} \end{bmatrix}\Bigg|_{\hat{\bar{x}}} \delta\bar{x} \tag{2.65}$$

Applying this subderivation, the camera measurements of $\breve{\bar{H}}_{\breve{\bar{x}}}\big(\hat{\bar{x}}^-(t_k)\big)\delta\bar{x}$ can be expanded via the chain rule.

$$\delta\bar{\Gamma} \approx \left.\frac{\partial\breve{\bar{\bar{\Gamma}}}\left(\breve{\bar{\bar{x}}}^{-}(t_k)\right)}{\partial\breve{\bar{\bar{\Lambda}}}_{ij}(t_k)}\right|_{\hat{\hat{\Lambda}}_{ij}} \delta\bar{\Lambda}(t_k) = \left.\frac{\partial\breve{\bar{\bar{\Gamma}}}^{-}(t_k)}{\partial\breve{\bar{\bar{\Lambda}}}_{ij}(t_k)}\right|_{\hat{\hat{\Lambda}}_{ij}} \left.\frac{\partial\breve{\bar{\bar{\Lambda}}}_{ij}^{-}(t_k)}{\partial\breve{\bar{\bar{x}}}(t_k)}\right|_{\hat{\hat{x}}} \delta\bar{x} \tag{2.66}$$

The resulting, total $\breve{\bar{\bar{H}}}_{\breve{\bar{\bar{x}}}}\left(\hat{\hat{x}}^{-}(t_k)\right)$ matrix is therefore

$$\breve{\bar{\bar{H}}}_{\breve{\bar{\bar{x}}}}\left(\hat{\hat{x}}^{-}(t_k)\right) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hat{\hat{\Xi}}_5 & \hat{\hat{\Xi}}_7 & 0 & 0 & \hat{\hat{\Xi}}_9 & 0 & 0 & 0 & \hat{\hat{\Xi}}_9 & \{-\hat{\hat{\Xi}}_5\} & \{-\hat{\hat{\Xi}}_7\} & 0 & 0 \\ \hat{\hat{\Xi}}_6 & \hat{\hat{\Xi}}_8 & 0 & 0 & \hat{\hat{\Xi}}_{10} & 0 & 0 & 0 & \hat{\hat{\Xi}}_{10} & 0 & 0 & \{-\hat{\hat{\Xi}}_6\} & \{-\hat{\hat{\Xi}}_8\} \end{bmatrix} \tag{2.67}$$

where

$$\begin{bmatrix} \hat{\hat{\Xi}}_5 \\ \hat{\hat{\Xi}}_6 \\ \hat{\hat{\Xi}}_7 \\ \hat{\hat{\Xi}}_8 \\ \hat{\hat{\Xi}}_9 \\ \hat{\hat{\Xi}}_{10} \end{bmatrix} = \begin{bmatrix} \hat{\hat{\Xi}}_{11}\hat{\hat{\Xi}}_{15} - \hat{\hat{\Xi}}_{12}\hat{\hat{\Xi}}_{16} \\ \hat{\hat{\Xi}}_{13}\hat{\hat{\Xi}}_{15} - \hat{\hat{\Xi}}_{14}\hat{\hat{\Xi}}_{16} \\ \hat{\hat{\Xi}}_{11}\hat{\hat{\Xi}}_{16} + \hat{\hat{\Xi}}_{12}\hat{\hat{\Xi}}_{15} \\ \hat{\hat{\Xi}}_{13}\hat{\hat{\Xi}}_{16} + \hat{\hat{\Xi}}_{14}\hat{\hat{\Xi}}_{15} \\ \hat{\hat{\Xi}}_{11}\hat{\hat{\Xi}}_{17} + \hat{\hat{\Xi}}_{12}\hat{\hat{\Xi}}_{18} \\ \hat{\hat{\Xi}}_{13}\hat{\hat{\Xi}}_{19} + \hat{\hat{\Xi}}_{14}\hat{\hat{\Xi}}_{20} \end{bmatrix} \tag{2.68}$$

$$\begin{aligned} \left.\frac{\partial\breve{\bar{\bar{\Gamma}}}^{-}(t_k)}{\partial\breve{\bar{\bar{\Lambda}}}_{ij}(t_k)}\right|_{\hat{\hat{\Lambda}}_{ij}} &= \begin{bmatrix} \hat{\hat{\Xi}}_{11} & \hat{\hat{\Xi}}_{12} & 0 & 0 \\ 0 & 0 & \hat{\hat{\Xi}}_{13} & \hat{\hat{\Xi}}_{14} \end{bmatrix} \\ &= \begin{bmatrix} \dfrac{-\hat{\Lambda}_{1j}(t_k)}{\hat{\Lambda}_{1i}^2(t_k)} & \dfrac{1}{\hat{\Lambda}_{1i}(t_k)} & 0 & 0 \\ 0 & 0 & \dfrac{-\hat{\Lambda}_{2j}(t_k)}{\hat{\Lambda}_{2i}^2(t_k)} & \dfrac{1}{\hat{\Lambda}_{2i}(t_k)} \end{bmatrix} \end{aligned} \tag{2.69}$$

$$\begin{aligned} &\left.\frac{\partial\breve{\bar{\bar{\Lambda}}}_{ij}^{-}(t_k)}{\partial\breve{\bar{\bar{X}}}(t_k)}\right|_{\hat{\hat{X}}} \\ &= \begin{bmatrix} \hat{\hat{\Xi}}_{15} & \hat{\hat{\Xi}}_{16} & 0 & 0 & \hat{\hat{\Xi}}_{17} & 0 & 0 & 0 & \hat{\hat{\Xi}}_{17} & \{-\hat{\hat{\Xi}}_{15}\} & \{-\hat{\hat{\Xi}}_{16}\} & 0 & 0 \\ \{-\hat{\hat{\Xi}}_{16}\} & \hat{\hat{\Xi}}_{15} & 0 & 0 & \hat{\hat{\Xi}}_{18} & 0 & 0 & 0 & \hat{\hat{\Xi}}_{18} & \hat{\hat{\Xi}}_{16} & \{-\hat{\hat{\Xi}}_{15}\} & 0 & 0 \\ \hat{\hat{\Xi}}_{15} & \hat{\hat{\Xi}}_{16} & 0 & 0 & \hat{\hat{\Xi}}_{19} & 0 & 0 & 0 & \hat{\hat{\Xi}}_{19} & 0 & 0 & \{-\hat{\hat{\Xi}}_{15}\} & \{-\hat{\hat{\Xi}}_{16}\} \\ \{-\hat{\hat{\Xi}}_{16}\} & \hat{\hat{\Xi}}_{15} & 0 & 0 & \hat{\hat{\Xi}}_{20} & 0 & 0 & 0 & \hat{\hat{\Xi}}_{20} & 0 & 0 & \hat{\hat{\Xi}}_{16} & \{-\hat{\hat{\Xi}}_{15}\} \end{bmatrix} \end{aligned} \tag{2.70}$$

$$\begin{gathered}
\begin{bmatrix} \hat{\hat{\Xi}}_{18} \\ \hat{\hat{\Xi}}_{17} \end{bmatrix} = \begin{bmatrix} \hat{\hat{\Xi}}_{15} & \hat{\hat{\Xi}}_{16} \\ \hat{\hat{\Xi}}_{16} & \{-\hat{\hat{\Xi}}_{15}\} \end{bmatrix} \begin{bmatrix} \widehat{\Lambda}_{1n}(t_k) - \hat{p}_n(t_k) \\ \widehat{\Lambda}_{1e}(t_k) - \hat{p}_e(t_k) \end{bmatrix} \\
\begin{bmatrix} \hat{\hat{\Xi}}_{20} \\ \hat{\hat{\Xi}}_{19} \end{bmatrix} = \begin{bmatrix} \hat{\hat{\Xi}}_{15} & \hat{\hat{\Xi}}_{16} \\ \hat{\hat{\Xi}}_{16} & \{-\hat{\hat{\Xi}}_{15}\} \end{bmatrix} \begin{bmatrix} \widehat{\Lambda}_{2n}(t_k) - \hat{p}_n(t_k) \\ \widehat{\Lambda}_{2e}(t_k) - \hat{p}_e(t_k) \end{bmatrix} \\
\begin{bmatrix} \hat{\hat{\Xi}}_{15} \\ \hat{\hat{\Xi}}_{16} \end{bmatrix} = \begin{bmatrix} -\cos\left(\hat{b}_c(t_k) + \hat{\psi}(t_k)\right) \\ -\sin\left(\hat{b}_c(t_k) + \hat{\psi}(t_k)\right) \end{bmatrix}
\end{gathered} \tag{2.71}$$

In this resulting matrix, the complexities arise due to the transformation matrices of Equation (2.62), which are combined using the addition and subtraction trigonometry identities, as seen in Equation set (2.71).

With the derivations now complete, validation is needed to confirm correct implementation for the fourth key Kalman filter equation. Similar to $\breve{\bar{\bar{F}}}_{\breve{\bar{x}}}(\hat{\bar{x}})$'s validation, $\breve{\bar{\bar{H}}}_{\breve{\bar{x}}}\left(\hat{\bar{x}}^{-}(t_k)\right)$ can be validated by means of Equation (2.61). To do this, linear error measurement propagation results in Equation (2.61) can be compared to residual error results, i.e., $\tilde{\bar{z}} - \hat{\tilde{\bar{z}}}$. If there is consistency between both errors, there is implication that $\breve{\bar{\bar{H}}}_{\breve{\bar{x}}}\left(\hat{\bar{x}}^{-}(t_k)\right)$ is consistent with the system measurement model and is appropriate for use within the fourth key equation.

The validation algorithm used to compare these errors is as follows: The simulation is set up using the four simulation characteristics (same τ_* values from Linear Error State Modeling). Starting with the truth and navigation states having the same initial values, the system is simulated forward in time up to the first discrete measurement (1 s). Simulating up to this specific time instant was used for consistency with the Linear Error State Modeling validation. At this point, just before measurements are taken, the same error matrix from the 2.1.4 Linear Error State Model subsection is injected into the navigation state. After measurements are taken, both the residual and the linear error measurement model are computed and the difference between the two is computed. Table 2.4 provides these results.

Table 2.4. Linear error measurement model validation.

Measurement	$\tilde{\tilde{z}} - \hat{\tilde{z}}$	$\delta\tilde{\tilde{z}}_{linear}$	Error Difference	Units
$p_n(t_k)$	2.0e+00	2.0e+00	0.0e+00	m
$p_e(t_k)$	3.0e+00	3.0e+00	0.0e+00	m
$\psi(t_k)$	4.7e-03	4.7e-03	-7.1e-17	rad
$\Gamma_{[1]}(t_k)$	-3.3e-02	-3.3e-02	-2.6e-05	rad
$\Gamma_{[2]}(t_k)$	-3.4e-02	-3.4e-02	-1.2e-04	rad

From the table, the differences between residual and the linearized measurement error for both GPS and heading are null to numerical precision. This is an expected result as these measurements were already linear. Nonlinear camera measurements, on the other hand, have error differences two orders of magnitude smaller than the residual and linearized measurement error. For the camera measurements, Injected errors that are influential in increasing error difference are landmark and heading error. The injected errors for landmarks do push the limits of first-order linearization validity, as the injected landmark errors are on the order of tens of meters, not just meters like GPS. Camera error differences are on the order of tenths of milliradians (hundredths of a degree). To keep the error differences small the author assumes that 15 m will be the 3σ landmark uncertainty threshold for this and future simulations. This assumption implies that the measurement geometry matrix, given that the results are using relatively large error injections, has been correctly implemented and is appropriate for use inside the filter key equations.

With the measurement geometry matrix now validated. This document moves on to discuss how to account for and validate continuous noise within the system in the

Covariance Propagation subsection.

2.1.6 Covariance Propagation

The previous subsections have provided definitions for most of the Kalman filter's key equations; however, these subsections have postponed the definitions of particular noise matrices. These next two subsections will finally provide these definitions. This first subsection will discuss the derivation for the $\breve{\bar{Q}}(\hat{\bar{x}})$ matrix. The next subsection will detail the derivation of the $\breve{\bar{R}}(\hat{\bar{x}})$ and Kalman gain matrix, $\hat{\bar{K}}$.

$\bar{Q}$ is a diagonal matrix composed of the power spectral density (PSD) of the process noise, $\bar{w}$. The detailed meaning of this description begins with how one represents and defines noise. Though Maybeck provides a more explicit explanation [3], this document provides the essential derivation details (using simplified version of the notation from Maybeck). In the truth state, the collection of driving noise terms was given as $\bar{w} = [u_{wg} \quad v_{wg} \quad w_{a_u} \quad w_{a_v} \quad w_{\omega} \quad w_c]^T$. Each of these driving noise terms, except wind gusts as explained below, are assumed to be Gaussian, not only in distribution, but in that they are independent of (Probability of one outcome does not affect the probability of another outcome: $P\big(\bar{x}(t_{\epsilon-1}) \cap \bar{y}(t_{\epsilon-1})\big) = P\big(\bar{x}(t_{\epsilon-1})\big)P\big(\bar{y}(t_{\epsilon-1})\big) \neq P\big(\bar{x}(t_{\epsilon-1})\big)P\big(\bar{y}(t_{\epsilon-1})\big|\bar{x}(t_{\epsilon-1})\big)$) and uncorrelated (covariance kernel: $\bar{\Psi}_{xy}(t_{\epsilon-1}, t_\epsilon) = E\big(\bar{x}(t_{\epsilon-1})\bar{y}^T(t_\epsilon)\big) = E\big(\bar{x}(t_{\epsilon-1})\big)E\big(\bar{y}^T(t_\epsilon)\big) => $ covariance between two probabilities is zero: $\bar{P}_{xy}(t_{\epsilon-1}, t_\epsilon) = \bar{0}$) with each other, the state, and the state dynamics. Note $E(*)$ is the expectation operator, and independence implies uncorrelated but not vice versa. Such a definition is one mathematical way to say that the gaussian driving noise terms across the entire flight are white (i.e., stochastic, random) and within a gaussian range of each noise's individual mean. Because of the noises' gaussian nature, the white noises can be easily, sufficiently represented by way of the statistical first moment (mean) and second central moment (covariance). All noise sources are assumed to have zero mean. To extract the covariance, however, requires more derivation. Note the following covariance equation, which relates covariance kernel $\bar{P}_{xx}(t_{\epsilon-1}, t_\epsilon)$, correlation kernel $\bar{\Psi}_{xx}(t_{\epsilon-1}, t_\epsilon)$, and mean $\bar{m}_x(t_*)$

$$\bar{P}_{xx}(t_{\epsilon-1}, t_\epsilon) = \bar{\Psi}_{xx}(t_{\epsilon-1}, t_\epsilon) - \bar{m}_x(t_{\epsilon-1})\bar{m}_x^T(t_\epsilon) = \bar{\Psi}_{xx}(t_{\epsilon-1}, t_\epsilon) \qquad (2.72)$$

If one also assumes that the driving noise terms are wide-sense-stationary processes, then one can find that the correlation kernel evaluated in the frequency domain—the PSD $\bar{\Psi}_{xx}(\omega)$, can be equated to the Fourier transform of the correlation kernel evaluated in the time domain, $\bar{\Psi}_{xx}(t_{\epsilon-1}, t_\epsilon)$. With inverse rearrangement, the correlation kernel in the time domain, the value being sought, is equal to the integral of the PSD:

$$\begin{aligned}\bar{P}_{xx}(t_{\epsilon-1}, t_\epsilon) = \bar{\Psi}_{xx}(t_{\epsilon-1}, t_\epsilon) &= \int_{-\infty}^{\infty} \bar{\Psi}_{xx}(\omega) e^{-\hat{\tilde{i}}\omega(t_\epsilon - t_{\epsilon-1})}\, d\omega \\ &= \int_{-\infty}^{\infty} \bar{Q}\delta(t_\epsilon - t_{\epsilon-1}) e^{-\hat{\tilde{i}}\omega(t_\epsilon - t_{\epsilon-1})} d\omega\end{aligned} \qquad (2.73)$$

The right-hand-side of Equation (2.73) breaks the PSD into two components: The $\bar{Q}$ matrix, which holds the PSD coefficients, and the Dirac Delta function (units: Hz; $\hat{\tilde{i}} = imaginary\ unit$), which makes the frequency power only active at zero-time difference. Now, at zero-time difference, the time in which Equation (2.73) has the largest value, and using the uncorrelated gaussian assumption, only the diagonal terms of the correlation matrix, the autocorrelation matrix $\bar{\Psi}_{xx}(t_{\epsilon-1}, t_{\epsilon-1}) = E\big(x(t)x^T(t)\big) = \bar{P}_{xx}(t_{\epsilon-1}, t_{\epsilon-1})$, need be assessed (Note that when discussing the navigation state, if transformations between coordinate frames technically do create a correlation, the $\breve{\breve{G}}(\hat{\hat{x}})$ matrix would be used to account for this discrepancy). In a more common form, this derivation summarizes the truth state's driving noise as

$$\begin{gathered} E\left(\bar{w}(t_\epsilon)\right) = \bar{0} \\ E\left(\bar{w}(t_\epsilon)\bar{w}^T(t_{\epsilon-1})\right) = \bar{Q}\,\delta(t_\epsilon - t_{\epsilon-1}) \end{gathered} \tag{2.74}$$

with the top equation denoting that all noise mean is zero and the bottom equation denoting that the covariance kernel is equal to the diagonal PSD matrix times Dirac Delta.

What makes this equation set important is that mean and covariance are significantly easier to calculate, as opposed to computing the entire cumulative distribution function. For one, the mean is always being zero in this derivation; For two, the power spectral densities can be found in a more straight forward procedure. To explain, recall that the bias states were modeled as first-order Markov processes driven by white noise (first-order differential equation, Markov in that the current values is dependent on the prior value). The PSD of a first-order Markov is known to be

$$Q = \frac{2\sigma_{ss}^2}{\tau} \tag{2.75}$$

Where σ_{ss} is the steady-state, one-σ standard deviation of the bias and τ is the time constant of the bias, which are found in specification sheets, sample testing, or theoretical calculations. For wind gust power, null values will be used for these simulations. Nevertheless, for future work, particularly in multi-UAV, tight-formation applications, where wind gusts are significantly influential, a Dryden gust model as described in Beard and McLain [13] can be used. Hence, the truth state $\bar{Q}$ matrix is defined as

$$\bar{Q} = \begin{bmatrix} PSD_{u_{wg}} & 0 & 0 & 0 & 0 & 0 \\ 0 & PSD_{v_{wg}} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{2\sigma_{ss_{a_u}}^2}{\tau_{a_u}} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{2\sigma_{ss_{a_v}}^2}{\tau_{a_v}} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{2\sigma_{ss_{\omega}}^2}{\tau_{\omega}} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{2\sigma_{ss_c}^2}{\tau_c} \end{bmatrix} \tag{2.76}$$

One can think of this matrix as a description of the strength of Brownian motion or the rate at which noise variance grows.

The mean and covariance for the truth state has just been described. The design model means and covariances are derived similarly:

$$\begin{gathered} E\left(\breve{\bar{w}}(t_\epsilon)\right) = \bar{0} \\ E\left(\breve{\bar{w}}(t_\epsilon)\breve{\bar{w}}^T(t_{\epsilon-1})\right) = \breve{\bar{Q}}(\hat{\bar{x}})\,\delta(t_\epsilon - t_{\epsilon-1}) \end{gathered} \tag{2.77}$$

Though not necessarily a function of navigation state, the design model PSD matrix is indicated with a $(\hat{\bar{x}})$ to indicate that the PSD is a function of the noise statistics known to the navigation state. With regards to the measurement noise, their power spectral densities are defined by the square of velocity random walk (VRW) and angular random walk (ARW) values found in accelerometer and gyroscope specification sheets (VRW and ARW can be thought of as integrals of white noise; they are also referred to as noise density). Combining Equation (2.77)'s definition with the concept of Brownian constant diffusion (random walk) process, the design model PSD matrix can be used in the covariance propagation equation (Equation (2.40)). $\breve{\bar{Q}}(\hat{\bar{x}})$ is defined here as

$$\breve{\bar{Q}}(\hat{\bar{x}}) = \begin{bmatrix} \widehat{VRW}_{a_u}^2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \widehat{VRW}_{a_v}^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \widehat{ARW}_{\omega}^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{2\hat{\sigma}_{ss_{a_u}}^2}{\hat{\tau}_{a_u}} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{2\hat{\sigma}_{ss_{a_v}}^2}{\hat{\tau}_{a_v}} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{2\hat{\sigma}_{ss_{\omega}}^2}{\hat{\tau}_{\omega}} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{2\hat{\sigma}_{ss_c}^2}{\hat{\tau}_c} \end{bmatrix} \tag{2.78}$$

Once again, a need to validate correct implementation of this matrix is needed. Although a clear method to validate this matrix individually is not known to the author, this $\breve{\bar{Q}}(\hat{\bar{x}})$ matrix and the $\breve{\bar{G}}(\hat{\bar{x}})$ matrix can be validated together through the covariance propagation equation itself. The idea is that, without discrete measurement updates, there is intuition from prior research that the propagated variance will grow in specific patterns. If these patterns are compared against multiple Monte Carlo simulations, the general trend of errors from these simulations, with the presence of noise, should follow the same variance growth patterns up to first-order errors. These patterns are linear for velocity, quadratic for position, and constant for biases. Landmarks are also expected to be constant, as explained in the 2.1.4 Linear Error State Modeling subsection, where the state dynamics can only assume a constant landmark position.

Recalling the four simulation characteristics, the same Euler's method, manipulated control inputs, measurement synthesis, and measurement restriction procedures are used. The manipulated control input, however, was altered to test other Monte Carlo simulations with differing rudder combinations. When only analyzing a sinusoidal rudder, truth and navigation states have higher likelihood to cross over each other as the UAV loops in the trajectory. The cross over causes error plots to have a sinusoidal nature, masking the extent to which state errors can grow to. Hence, by using a low-rudder input, the truth and navigation states are more likely to diverge from each other and show a worse-case-scenario of the extent that continuous time propagation errors can grow to. This document will present particularly a null-rudder, constant-90°-rudder, and sinusoidal rudder for discussion on extreme cases. Nevertheless, 18 different trajectories for each sensor grade are studied to verify appropriate results. Note that for future testing in the 2.1.7 Estimation Capability subsection, the landmark locations for a null-rudder trajectory are moved to the locations shown in Table 2.5.

Table 2.5. Null-rudder landmark locations.

State	$\bar{X}$	$\hat{\bar{x}}$	Units
Λ_{1n}	400.0e+00	400.0e+00	m
Λ_{1e}	2500.0e+00	2500.0e+00	m
Λ_{2n}	500.0e+00	500.0e+00	m
Λ_{2e}	2500.0e+00	2500.0e+00	m

Noise restrictions are removed from the simulation characteristics, but only driving and process noise, not discrete measurements noise, are activated in the states since discrete measurements are not yet used to update state values and covariances. The active noise measurements for each time step are defined as

$$\bar{w}_{[*]} = \sqrt{\frac{Q_{\bar{w}_{[*]}}}{t_\epsilon - t_{\epsilon-1}}}\zeta \tag{2.79}$$

where ζ is a random, Gaussian-distributed, pseudo-random number centered around zero. Defining noise this way allows one to generate discrete simulation noise that behaves like continuous white noise. Furthermore, the initial truth state is forced to be offset from the navigation state in Table 2.2 and Table 2.5 by adding in an “initial-uncertainty”:

$$\bar{X}_{[*]}(t_\epsilon = 0) = \bar{X}_{[*]}(t_\epsilon = 0) + \zeta\ \sigma_{iu_{\bar{X}_{[*]}}} \tag{2.80}$$

where $\sigma_{iu_{\bar{X}_{[*]}}}$ is the true one standard deviation of the initial uncertainty of $\bar{X}_{[*]}$. Likewise, the initial condition for the covariance matrix is made consistent with the initial state conditions (i.e., the covariance of the state estimates is equivalent to the covariance of the state estimation errors). It is set to only be diagonal such that

$$\hat{\hat{P}}(t_i = 0) = \begin{bmatrix} \hat{\sigma}^2_{iu_{p_n}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \hat{\sigma}^2_{iu_{p_e}} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \Xi^2_{21} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \Xi^2_{22} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \hat{\sigma}^2_{iu_\psi} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \hat{\sigma}^2_{iu_{\Lambda_{2e}}} \end{bmatrix} \tag{2.81}$$

which uses the initial uncertainty of the navigation state, $\sigma_{iu_{\hat{\hat{x}}_{[*]}}}$ (terms Ξ_{21} and Ξ_{22}, which replace $\hat{\sigma}^2_{iu_{u_n}}$ and $\hat{\sigma}^2_{iu_{v_e}}$, are explained in the second following paragraph). In other words, the UAV is positioned and calibrated to have a specific state, however, in truth, this positioning and calibration may be off by approximately $\pm 1\sigma$. To account for the off positioning and calibration, a covariance matrix based on what the engineer thinks is $\pm 1\sigma$ is made. For this simulation, when applicable, $\sigma_{iu_{\bar{X}_{[*]}}} = \hat{\sigma}_{iu_{\hat{\hat{x}}_{[*]}}} = \sigma_{ss_{\bar{X}_{[*]}}} = \hat{\sigma}_{ss_{\hat{\hat{x}}_{[*]}}}$ (subscript ss denotes steady state) is assumed and it is left to future work to generate sensitivity analyses that considers discrepancies between the different types of uncertainties. The sigma and sensor grade values for this validation and future simulations are provided in Table 2.6.

Table 2.6: 1σ values.

Initial State Uncertainty

State	Commercial	Tactical	Navigation	Units
p_n and $\hat{p}_n$	3.3e-01	3.3e-01	3.3e-01	m
p_e and $\hat{p}_e$	3.3e-01	3.3e-01	3.3e-01	m
u and $\hat{u}$	3.3e-02	3.3e-02	3.3e-02	$\mathrm{m/s}$
v and $\hat{v}$	3.3e-02	3.3e-02	3.3e-02	$\mathrm{m/s}$
ψ and $\hat{\psi}$	5.2e-04	5.2e-04	5.2e-04	rad
r	1.5e-05	1.5e-05	2.4e-07	$\mathrm{rad/s}$
b_{a_u} and $\hat{b}_{a_u}$	6.2e-02	2.9e-02	9.8e-04	$\mathrm{m/s^2}$
b_{a_v} and $\hat{b}_{a_v}$	6.2e-02	2.9e-02	9.8e-04	$\mathrm{m/s^2}$
b_ω and $\hat{b}_\omega$	1.5e-05	1.5e-05	2.4e-07	$\mathrm{rad/s}$
b_c and $\hat{b}_c$	1.2e-03	1.2e-03	1.2e-03	rad
Λ_{1n} and $\hat{\Lambda}_{1n}$	10.0e+00	10.0e+00	10.0e+00	m
Λ_{1e} and $\hat{\Lambda}_{1e}$	10.0e+00	10.0e+00	10.0e+00	m
Λ_{2n} and $\hat{\Lambda}_{2n}$	15.0e+00	15.0e+00	15.0e+00	m
Λ_{2e} and $\hat{\Lambda}_{2e}$	15.0e+00	15.0e+00	15.0e+00	m

Steady State Sensor Specification Uncertainty

Sensor	Commercial	Tactical	Navigation	Units
$\tilde{p}_n$	3.3e-01	3.3e-01	3.3e-01	m
$\tilde{p}_e$	3.3e-01	3.3e-01	3.3e-01	m
$\tilde{\psi}$	5.2e-04	5.2e-04	5.2e-04	rad
$\tilde{\omega}$	1.5e-05	1.5e-05	2.4e-07	$\mathrm{rad/s}$
$\tilde{\omega}\ ARW$	6.1e-05	4.4e-05	3.5e-06	$\mathrm{rad/\{s\sqrt{s}\}}$
τ_ω	10.0e+00	1800.0e+00	1800.0e+00	s
$\tilde{a}_u$	6.2e-02	2.9e-02	9.8e-04	$\mathrm{m/s^2}$
$\tilde{a}_u\ VRW$	1.4e-03	1.2e-03	9.8e-04	$\mathrm{m/\{s\sqrt{s}\}}$
τ_{a_u}	10.0e+00	1800.0e+00	1800.0e+00	s
$\tilde{a}_v$	6.2e-02	2.9e-02	9.8e-04	$\mathrm{m/s^2}$
$\tilde{a}_v\ VRW$	1.4e-03	1.2e-03	9.8e-04	$\mathrm{m/\{s\sqrt{s}\}}$
τ_{a_v}	10.0e+00	1800.0e+00	1800.0e+00	s
$\tilde{\bar{\Gamma}}$	1.2e-03	1.2e-03	1.2e-03	rad
τ_c	1800.0e+00	1800.0e+00	1800.0e+00	s

In detail, Table 2.6 provides values for commercial, tactical, and navigation grade UAV systems. Both systems will acquire ± 1.0m, 3σ GPS measurements. The author assumes that the 3σ velocity (both components in the Body frame) is unknown by ± 0.1 m/s. Based off documentation from VectorNav company, heading is unknown by approximately $\pm 0.03°$ 1-σ. Truth state heading rate uncertainty is set equivalent to gyroscope uncertainty. Camera bias for the UAV systems is derived based on using a high definition 1080x1920 camera. Implementation of higher grade or different vision-based measurements for rougher military operations (e.g., wind noise, night flight) is left to future work. Assuming 40° FOV and ± 5 pixels of 3σ uncertainty yields a camera bias of

$$\hat{b}_c = b_c = \frac{40°\pi}{180}\{1920\}\frac{5}{3} = 1.2 \text{ mrad } (1\sigma) \tag{2.82}$$

Future work may consider gimbal-mounting the camera to help reduce camera bias. The author assumes that the initial landmark positions are unknown up to 15 m 3σ, as explained in the 2.1.5 Linear Error Measurement Modeling subsection. For the continuous sensors, values are based on specifications from a (commercial) VN-100, (tactical) LN-200, and a (navigation) Novatel ISA-100C sensor units. The author assumed that the provided data from these sheets were 1σ statistics. One interesting note is that the VRW for the navigation grade is the same number as accelerometer bias—implying that navigation-grade bias is as small as VRW noise. VRW for tactical was unknown, and was assumed as the median between commercial and navigation grades.

Within Equation (2.81), there were two terms, Ξ_{21} and Ξ_{22}, that were not defined. These are terms that account for the uncertainty in velocity due to initial Body frame velocity error and initial heading error. To generate this value, the author toggled the initial nominal Body velocities into North-East velocities. After injecting $\pm 3\sigma$ uncertainties into both heading and North-East velocities, the injected North-East velocities were transformed back into Body velocities with an injected transformation matrix. The 1-σ equivalent of the maximum difference between the original Body velocities and the injected Body velocities were assigned to Ξ_{21} and Ξ_{22} respectively.

Continuing with the validation description, bias time constants are initially set to 1/10th of the 100 s simulation time ($t_{sim} = 100\ s$) to exercise the bias dynamics. Later, all test trajectories are re-evaluated with the true simulation time constants to confirm that all hairlines lie within the given variance bands. In short, the true time constants for navigation grade IMU's are assumed to be 30 min—a value used for tactical grade sensors, but nevertheless a large enough value to force the first-order Markovs to act like biases. The camera misalignment bias was given the same value in both sensor grades for simplicity. For continuous sensor commercial grades, which are more dependent on thermal properties, rough heat transfer calculations were used to determine the time needed for the UAV fuselage to decrease 63% from an assumed max temperature (turned on, stationary, in the sun) to an assumed lowest temperature (turned on, flying $20\frac{m}{s}$, in the shade). This time, which was approximately 10 s, was assumed to be the time constant for both the commercial accelerometer and commercial gyroscope.

Given these settings, 200 Monte Carlo runs are generated with the covariance being accounted for according to Equation (2.40). 200 runs were selected to provide enough intuition on the overall nature of the UAV states. More runs would have required more computational power. Fewer runs would have necessitated the use of confidence interval bands on the determined variance to verify accuracy. Selecting the covariance history from one of the Monte Carlo runs, its variance values are compared against the errors (i.e., hair plots) from the 200 Monte Carlo runs (via mapping n). Note that the covariance history from one Monte Carlo to another Monte Carlo was empirically determined to differ insignificantly. The need to compare the system covariance against so many Monte Carlos is due to the nature of the position and velocity states. Although having zero mean error like process noise by itself, position, velocity, and bias that are corrupted by process noise are not white like process noise by itself. This means that position, velocity, and bias have some correlation between the previous time instant and the next time instant, which implies that their errors from a single Monte Carlo run are not likely to be drastic and span the entire variance ranges. Hence, multiple Monte Carlo simulations are needed to ensure that, position, velocity, and bias do span the covariance statistics. The variance history from the Monte Carlo runs for selected trajectories with a commercial sensor package is shown in the following Figures:

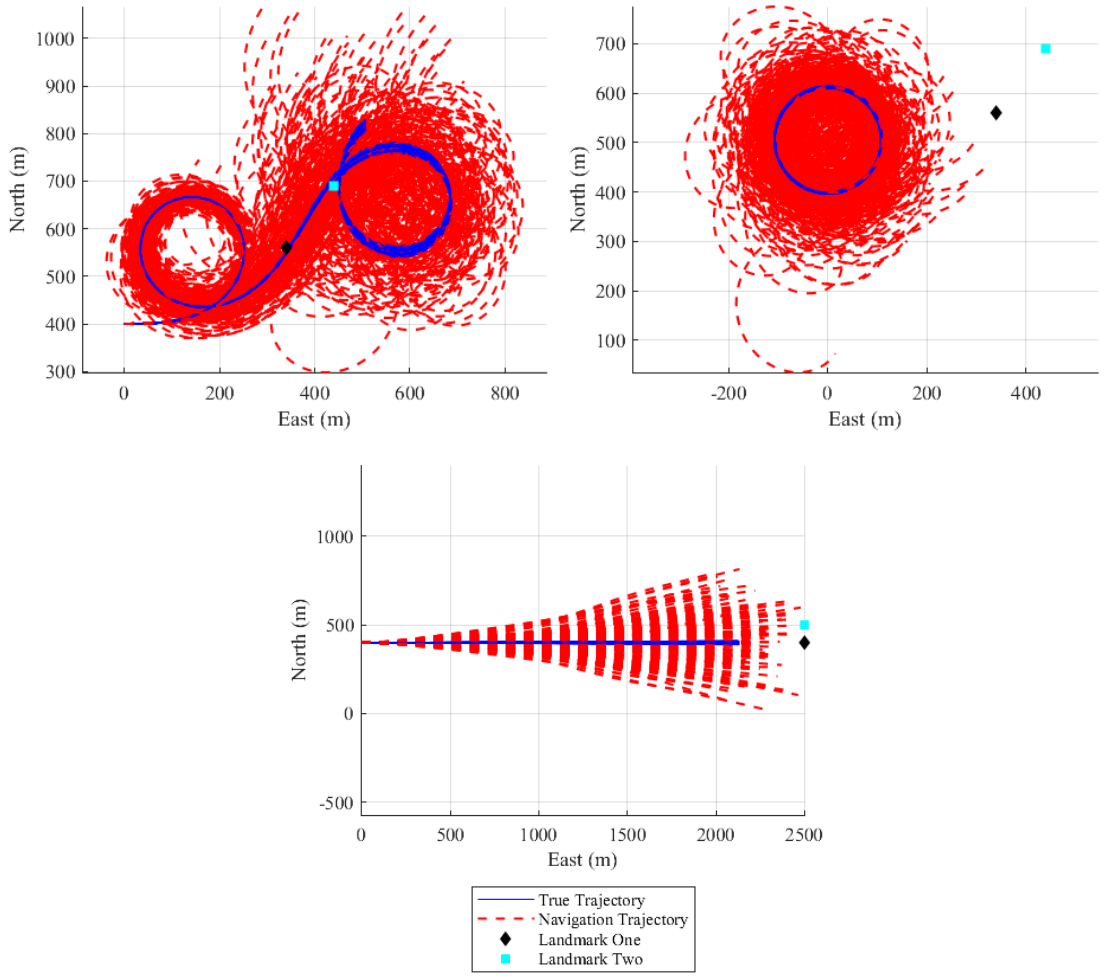


Figure 2.20. Covariance propagation validation trajectory.

To adequately show how trajectories effects covariance results, this validation will provide complete results from three separate simulations. Presented on the top left, top right, and bottom (top, middle, and bottom) respectively, the separate simulations involve trajectories of full dynamics (from previous validations), constant full rudder (90°), and no rudder. All trajectories include sinusoidal thrust (simulations without sinusoidal thrust slightly decrease the magnitude of errors in the hairlines).

Figure 2.20 shows these trajectories for all 200 Monte Carlo runs, in terms of both the truth and navigation states.

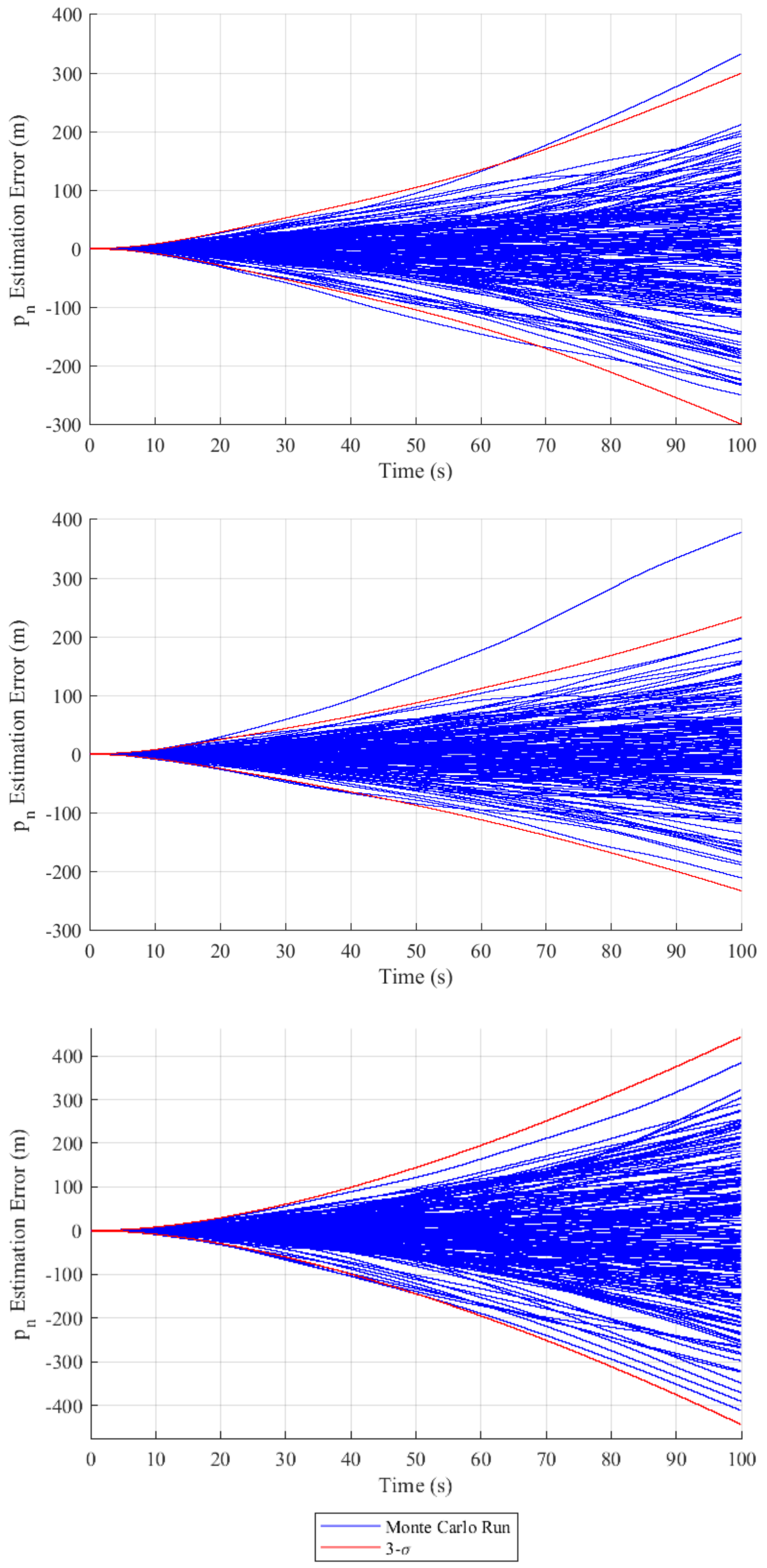


Figure 2.21. Covariance propagation validation for p_n.

Looking at the error plot of p_n in Figure 2.21, most all position error hairlines from the Monte Carlo runs, denoted with dark blue lines, are within the red lines, which denote the estimated 3σ bounds for the given state (derived by converting the estimated variance from the covariance matrix into 3σ values). All solutions exhibit the desired quadratic growth trend. The largest 3σ bounds occur with the straight trajectory, and the smallest 3σ bounds occur with the circular trajectory.

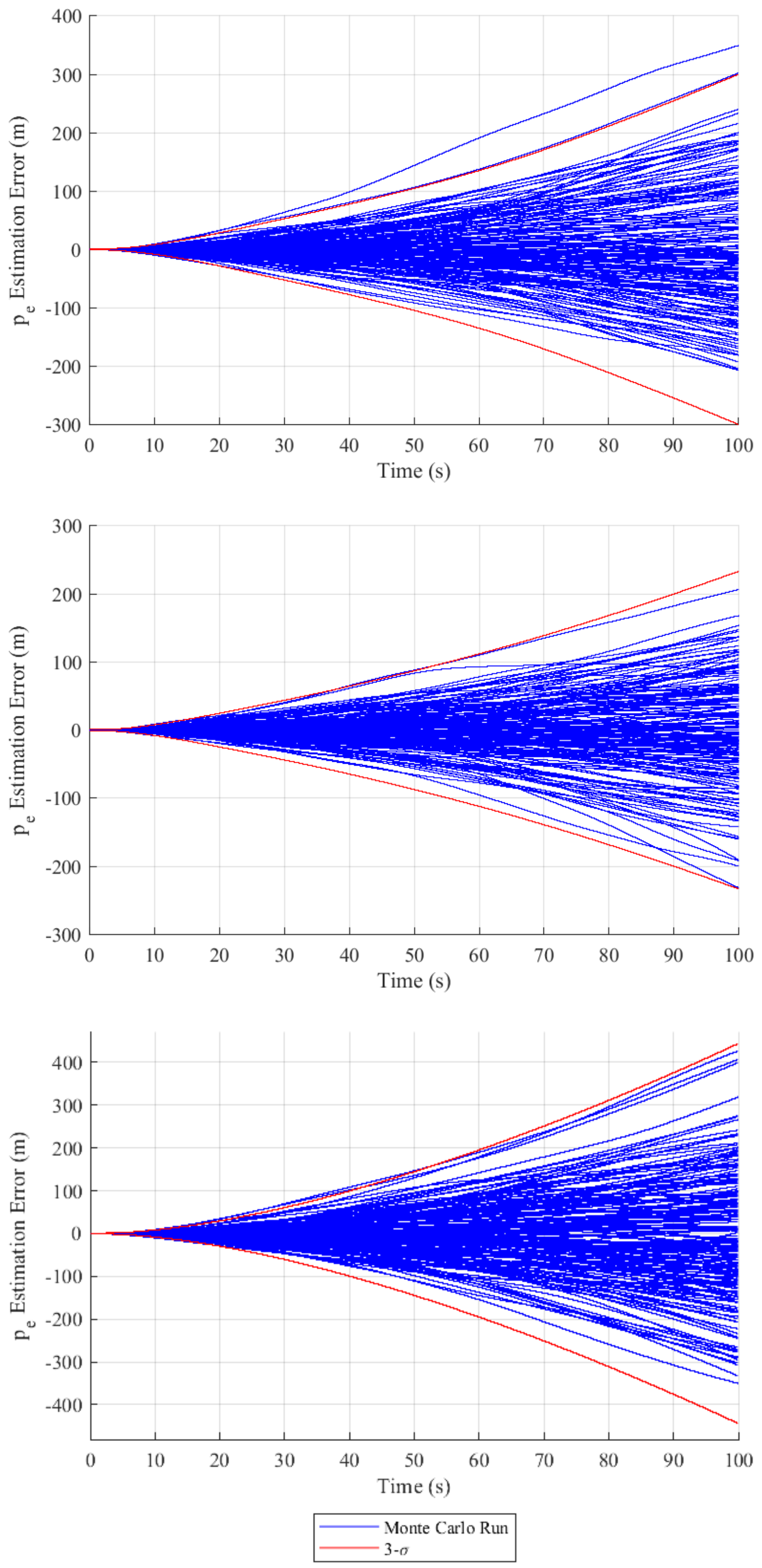


Figure 2.22. Covariance propagation validation for p_e.

The error plots for p_e are shown in Figure 2.22. Again, the desired quadratic growth is manifested. When using true τ values, the range for the minimum and maximum 3σ position values at the end of the simulation are approximately 235 m to 440 m for commercial sensors and approximately 13 m to 20 m for navigation sensors.

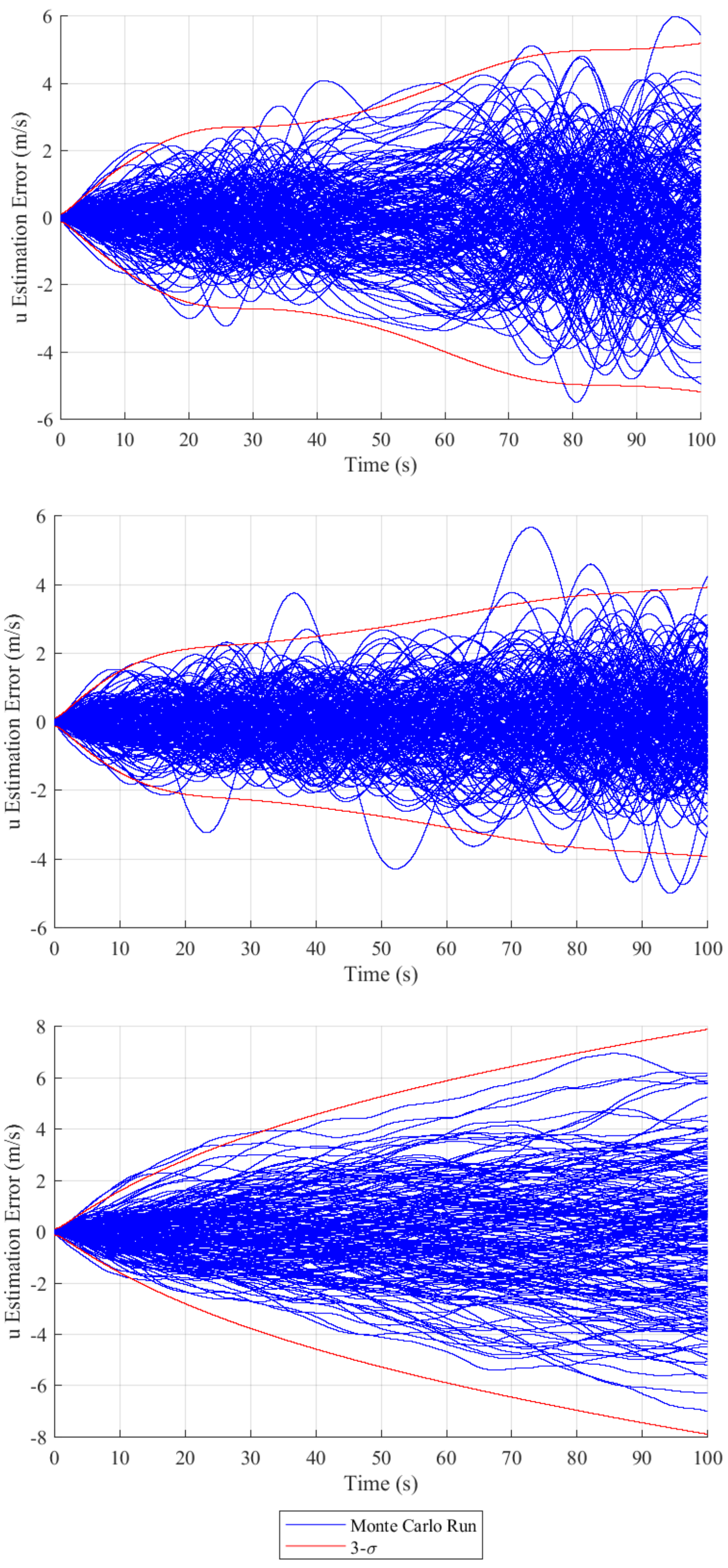


Figure 2.23. Covariance propagation validation for u.

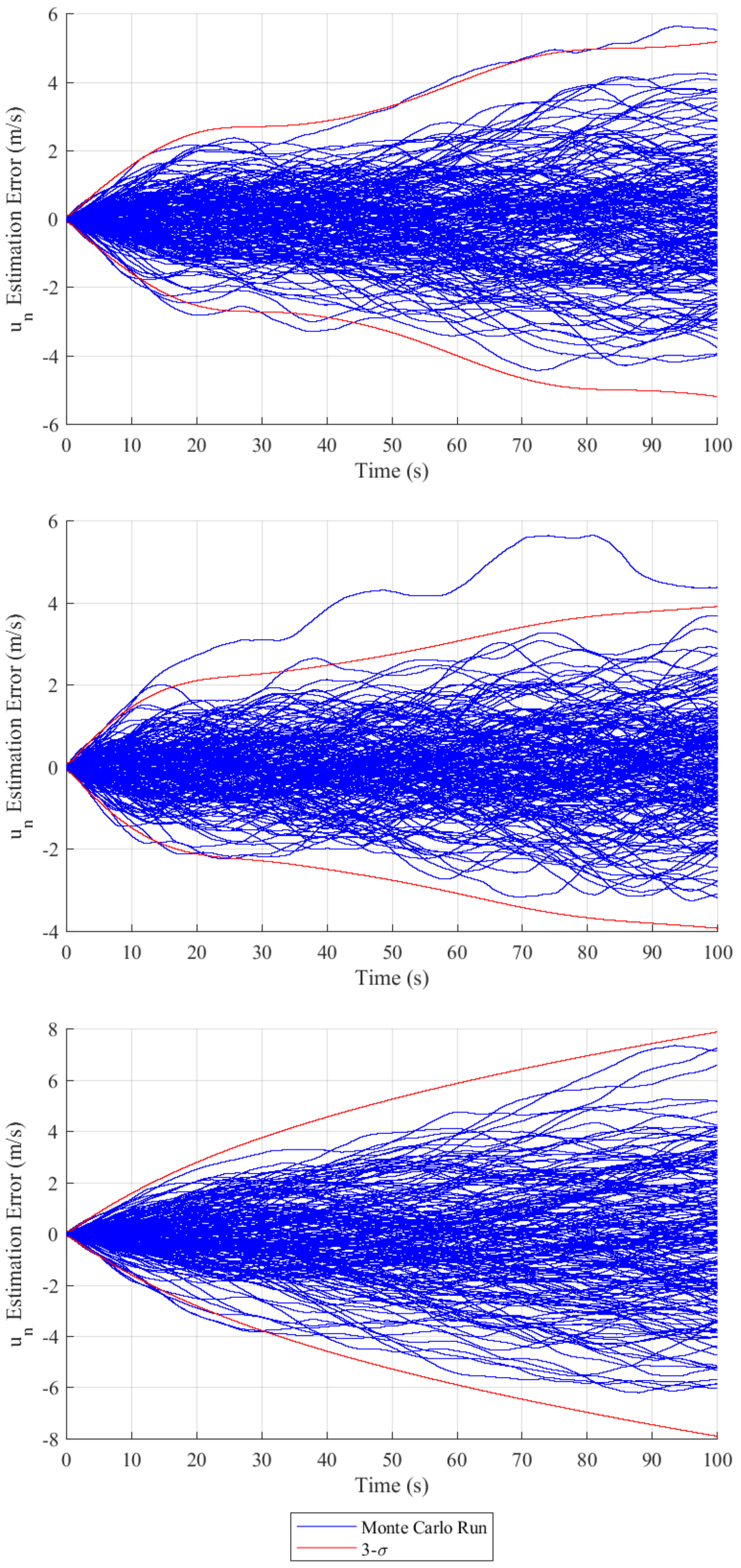


Figure 2.24. Covariance propagation validation for u_n.

Velocity is more susceptible to bumpy, wave-like trends, which are manifested in the top plots of Figure 2.23 and Figure 2.24. The bumpy patterns strongly correlate with thrust and rudder movement and larger heading uncertainty. Particularly, whenever the UAV makes a turn, the truth state and navigation state are more likely to ‘cross paths’ or intersect each other, which consequently cause errors between the truth and navigation state to minimize, hence the wave-like patterns in the hairlines. The covariance propagation equation appears to accurately account for velocity growth despite these wave trends. Furthermore, the constant-rudder and no-rudder solution, although appearing more logarithmic-shaped due to scaling, retain the desired linear growth trend to justify correct implementation. The full-dynamics trajectory is understandably wavier in growth as the covariance propagation equation accounts for the accentuated hairline errors from this extreme trajectory case. Note that to plot velocities in the North-East frame, an affine transformation, analogous to the previously used direction cosine matrices, was used.

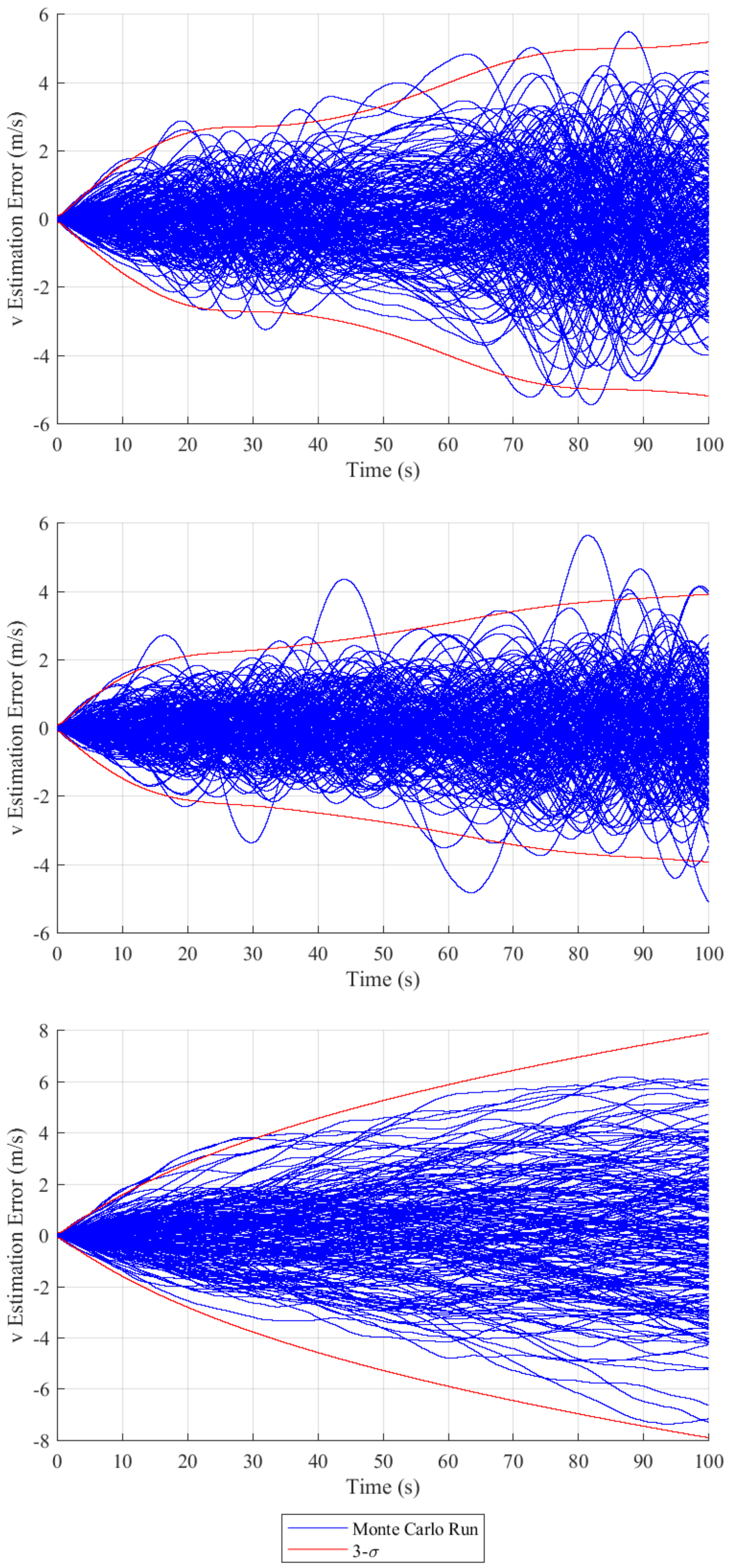


Figure 2.25. Covariance propagation validation for v.

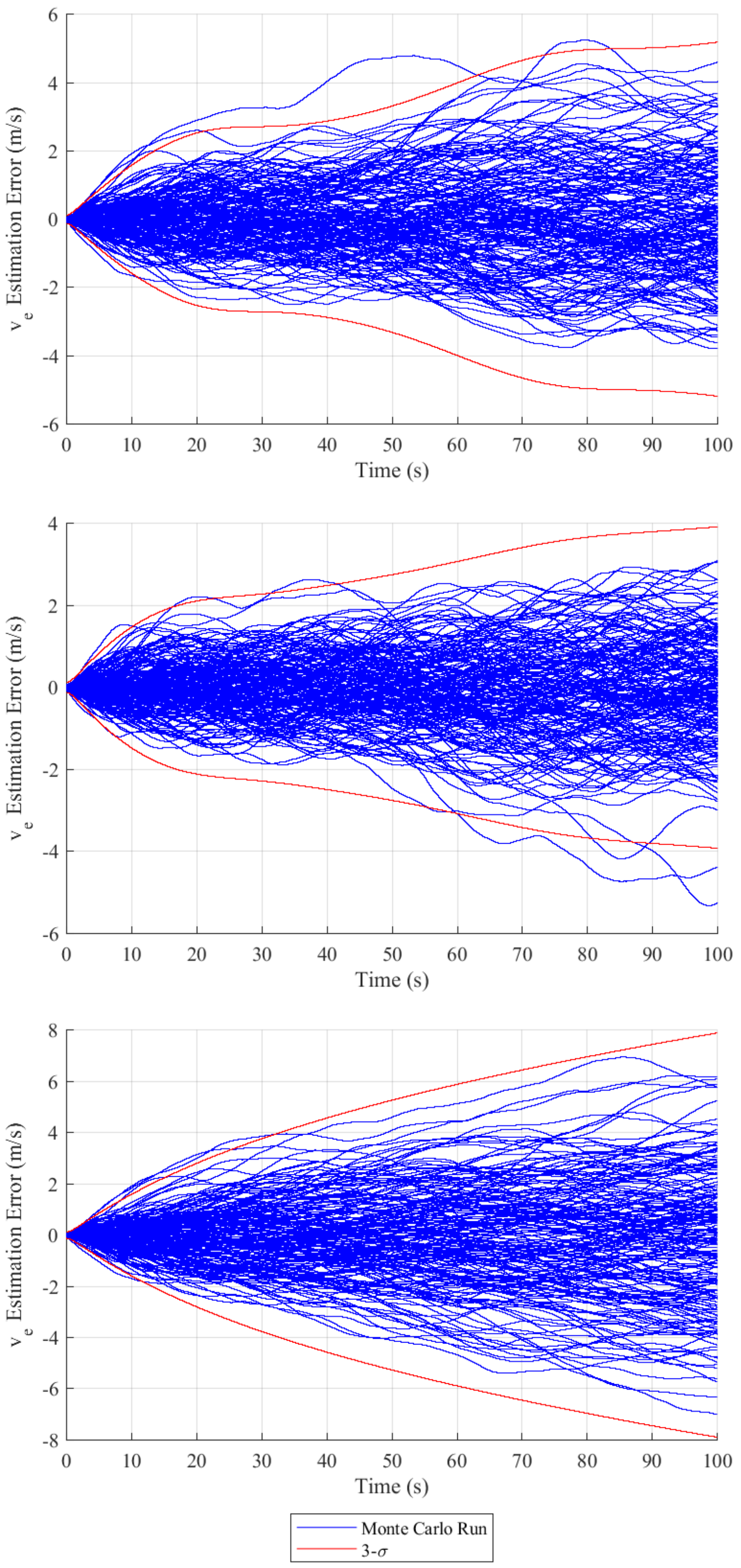


Figure 2.26. Covariance propagation validation for v_e.

Velocity plots in Figure 2.25 and Figure 2.26 show similar results as the prior velocity results, where from testing, the author assumed that a maximum of five periodic hairlines outside 3σ was acceptable results. Using smaller times steps appeared to help reduce unacceptable results. When verifying the constant-rudder trajectories ($\overset{\frown}{\delta}_r \geq 60°$) with true τ values, more than five (six to nine lines) periodically came outside 3σ. This phenomenon mostly occurred after 85 s of simulation time (only the constant-rudder at $\overset{\frown}{\delta}_r = 90°$ yielded hairlines outside 3σ prior to 85 s). To avoid larger computation times using a smaller time step, the author assumed these results acceptable seeing as the trajectories are extreme, discrete updates in future development will resolve such errors before 85 s, and only velocity—not position—plots have these results.

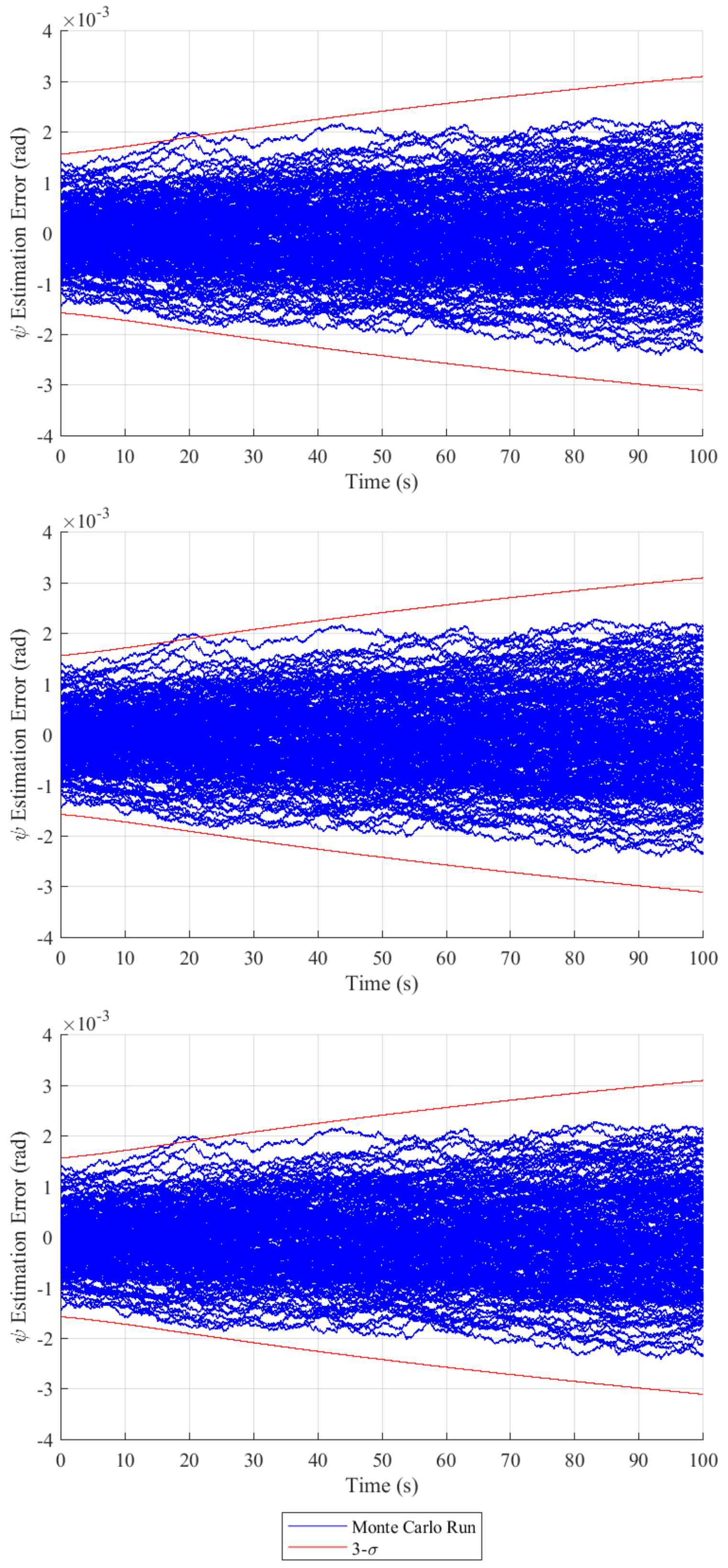


Figure 2.27. Covariance propagation validation for ψ.

The heading results for all three simulations in Figure 2.27 indicate the same conclusion that noise error growth are indifferent between different trajectories. The same is true for navigation sensors, but the resulting plot is different:

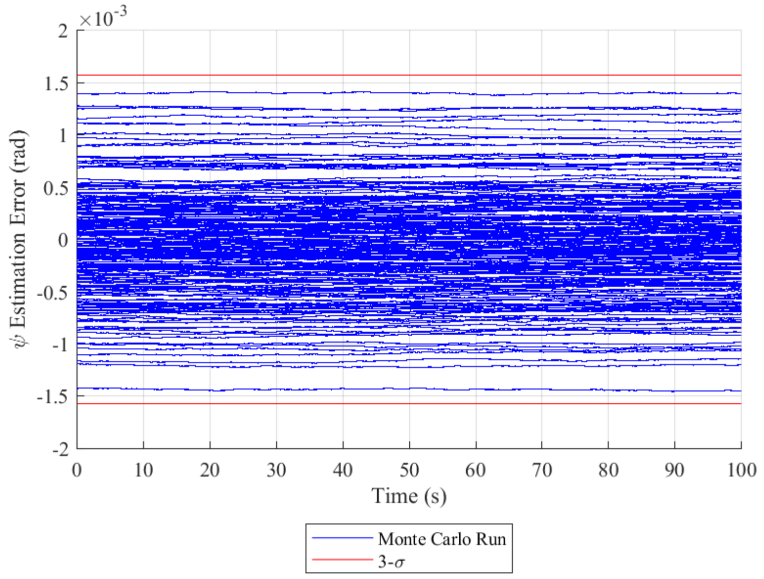


Figure 2.28. Covariance propagation validation for ψ using navigation sensors.

Essentially, Figure 2.28 indicates that noise error becomes negligible compared to the initial error between the navigation state and the truth state when using navigation sensors.

Following the heading states are the biases. Being initialized with the same pseudo-random algorithm and being independent of the system dynamics, bias estimation error hairlines do not differ between simulations. Hence, only one plot is shown per bias state.

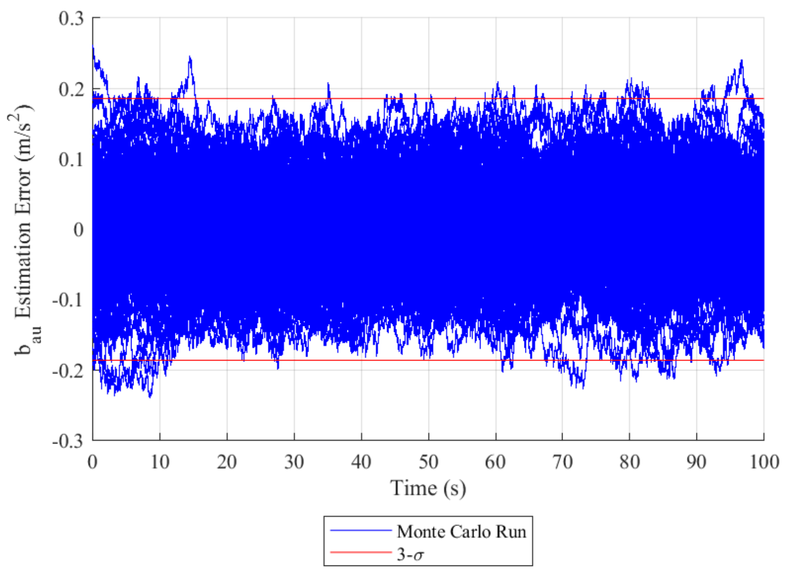


Figure 2.29. Covariance propagation validation for b_{au}.

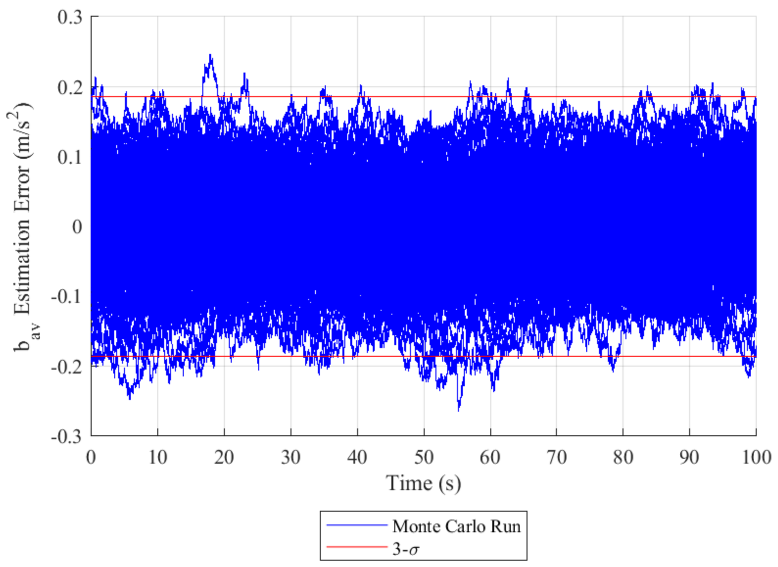


Figure 2.30. Covariance propagation validation for b_{av}.

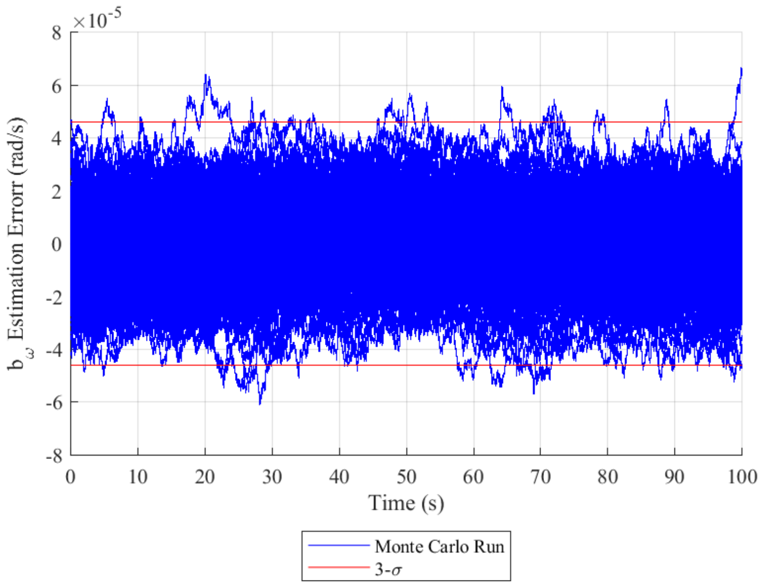


Figure 2.31. Covariance propagation validation for b_ω.

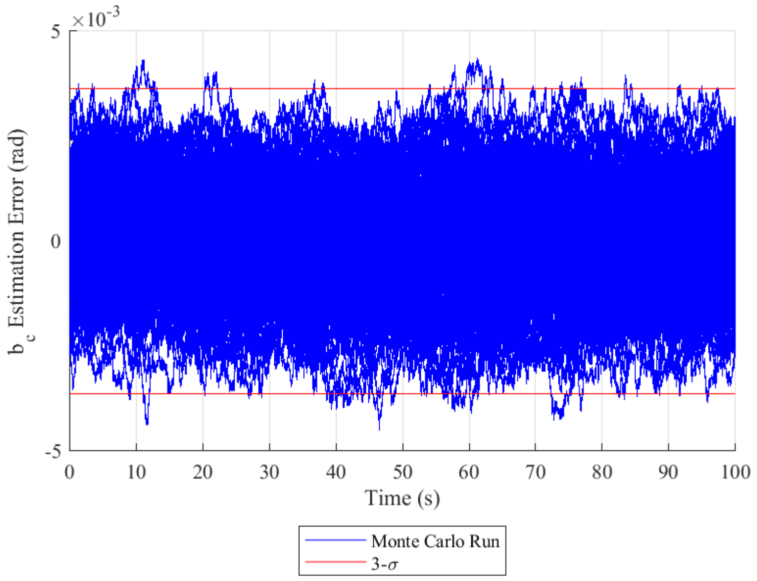


Figure 2.32. Covariance propagation validation for b_c.

All 3σ bias lines remain constant over time, which is consistent with expected growth pattern. Testing the true time constant for each bias, the random walking of error plot lines enters and exits the 3σ boundary in a manner common to results in prior research—

implying correct implementation.

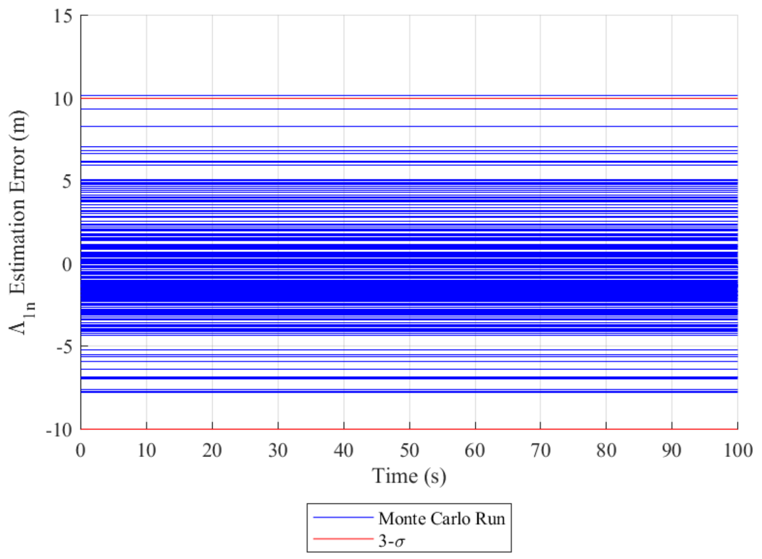


Figure 2.33. Covariance propagation validation for Λ_{1n}.

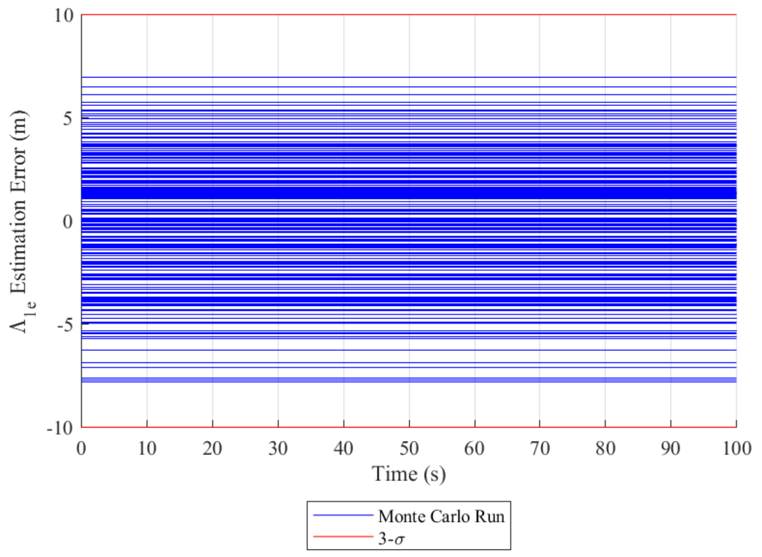


Figure 2.34. Covariance propagation validation for Λ_{1e}.

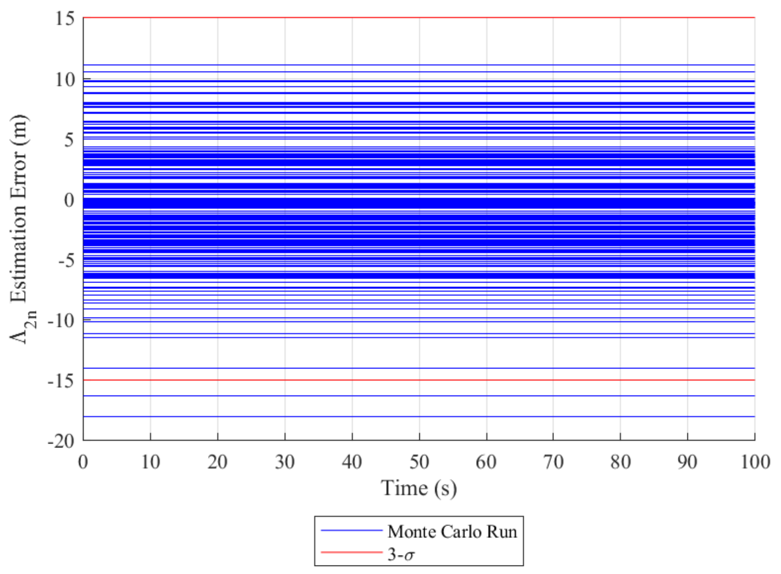


Figure 2.35. Covariance propagation validation for Λ_{2n}.

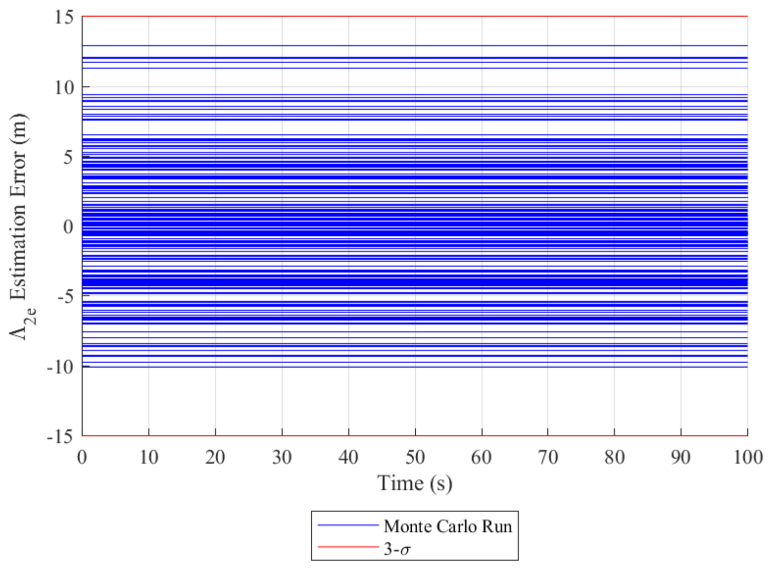


Figure 2.36. Covariance propagation validation for Λ_{2e}.

Plots for the landmarks do not differ neither from sensor grade to sensor grade nor from trajectory to trajectory due to their zero dynamics in the state dynamics, hence only a single plot for each landmark's north and east position are shown. Errors in these landmarks are only due to initial uncertainty.

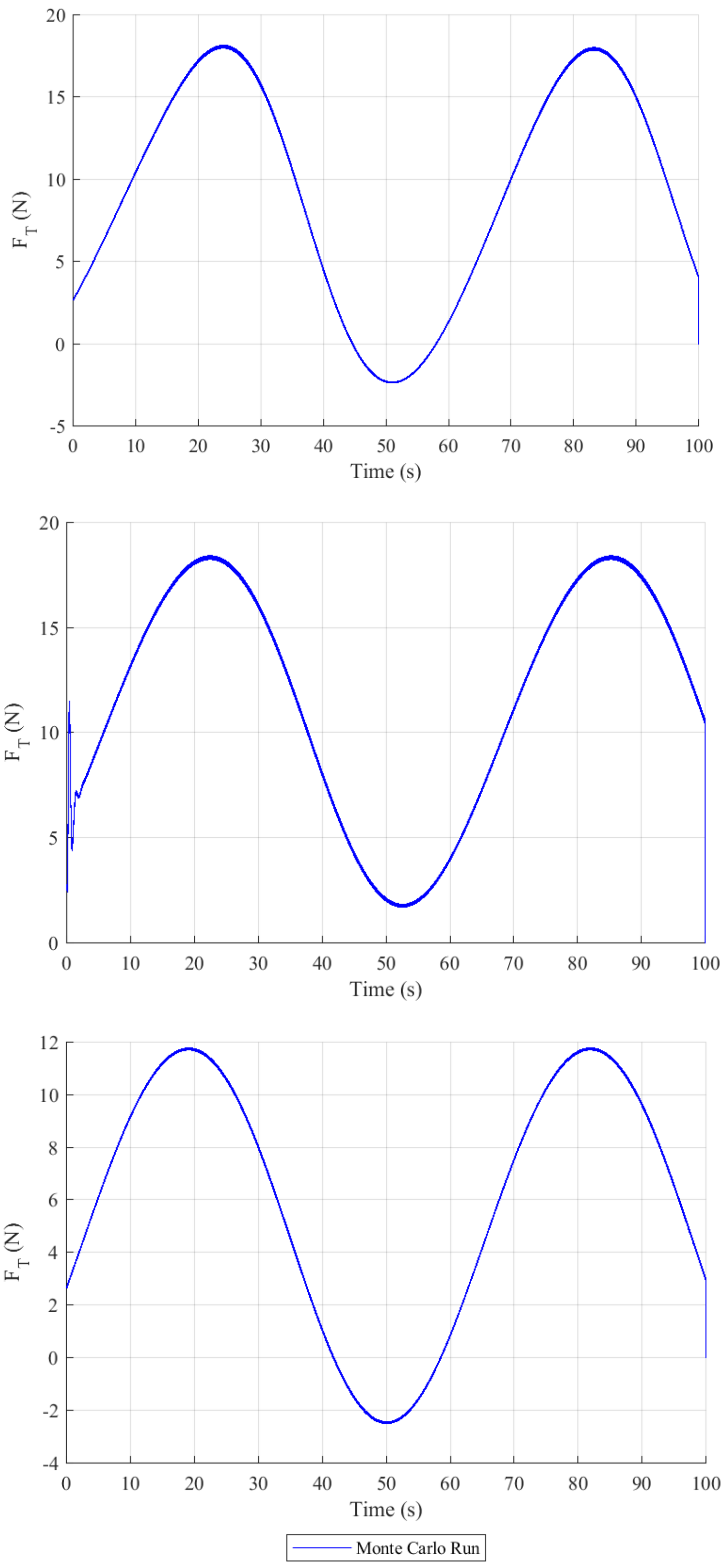


Figure 2.37. Covariance propagation validation for F_T.

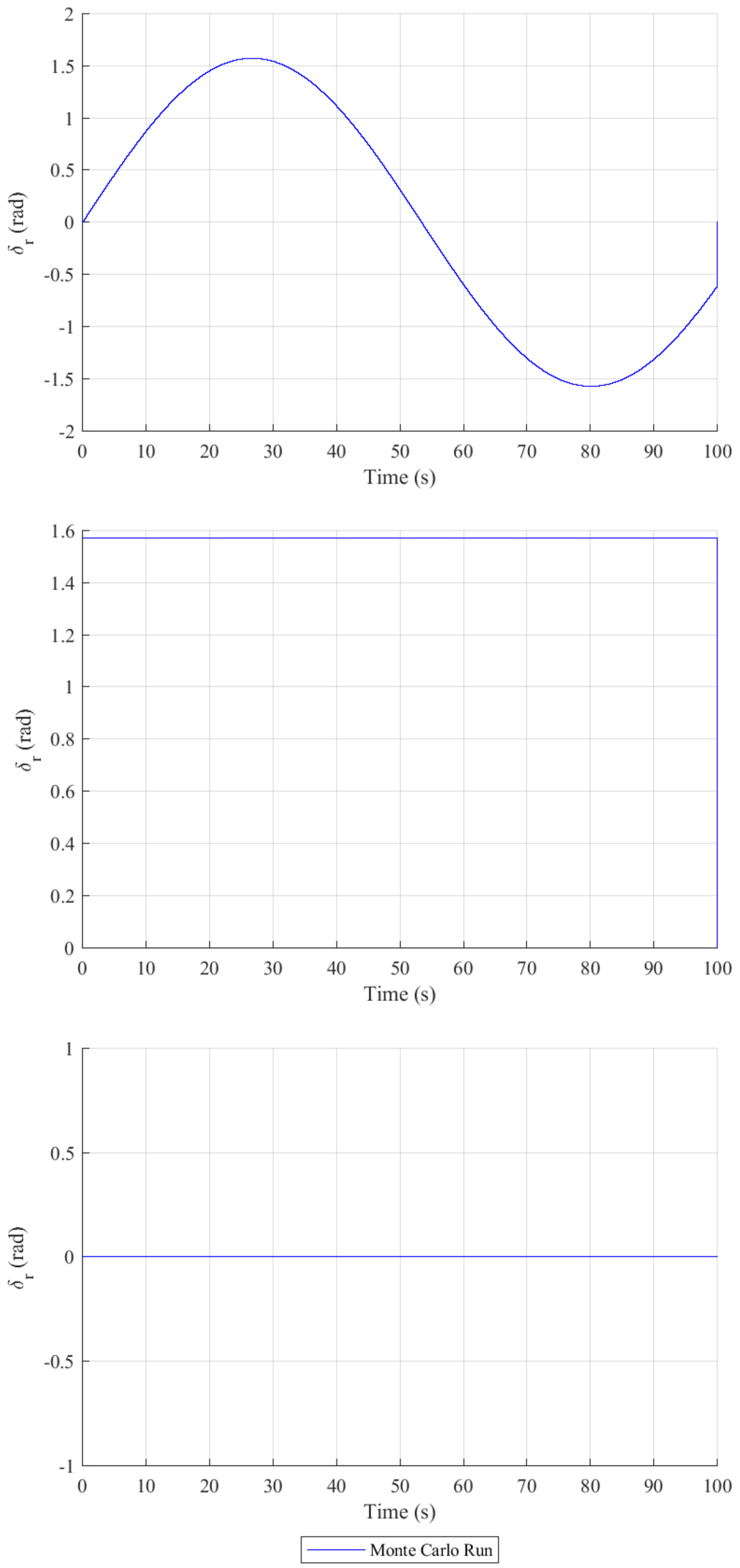


Figure 2.38. Covariance propagation validation for δ_r.

The full-dynamics control input plots are essentially the same as those seen in the initial propagation validation simulation, where only the thrust inputs have differences less than 0.2 N and rudder inputs have no differences between Monte Carlo runs. The constant-full-rudder, and no-rudder control input plots also show little differences between Monte Carlo runs.

From these validation plots, the 3σ estimates encapsulated the error behavior of the state over time. This implies that the matrices $\breve{\bar{G}}(\hat{\bar{x}})$ and $\breve{\bar{Q}}(\hat{\bar{x}})$ are implemented well enough to account for first-order discrepancies. Furthermore, for the biases, this implies that process noise has been correctly implemented. this document now moves on to the propagation update equation, Equation (2.42), to define and explain the discrete noise and gain matrices. A similar validation algorithm will be implemented to validate these matrices as well.

2.1.7 Estimation Capability

The prior section developed equations that track the state given the presence of continuous process noise. This section moves forward to update the state given the presence of discrete noise.

Recalling Equation (2.41) and Equation (2.42), much of the state and covariance updates are dependent on the Kalman gain weighting matrix provided below:

$$\begin{aligned}\hat{\bar{K}} = \hat{\bar{P}}^{-}\breve{\bar{H}}_{\breve{\bar{x}}}^{T}\left(\hat{\bar{x}}^{-}(t_k)\right)\Big\{\breve{\bar{H}}_{\breve{\bar{x}}}\left(\hat{\bar{x}}^{-}(t_k)\right)\hat{\bar{P}}^{-}\breve{\bar{H}}_{\breve{\bar{x}}}^{T}\left(\hat{\bar{x}}^{-}(t_k)\right) \\ + \breve{\bar{J}}(\hat{\bar{x}})\breve{\bar{R}}(\hat{\bar{x}})\breve{\bar{J}}^{T}(\hat{\bar{x}})\Big\}^{-1}\end{aligned} \tag{2.83}$$

This matrix contains many of the terms already defined, including the a priori covariance, the measurement geometry matrix and $\breve{\bar{J}}(\hat{\bar{x}})$. The remaining undefined term is the variance matrix, $\breve{\bar{R}}(\hat{\bar{x}})$, which is analogous to the discrete measurement version of $\breve{\bar{Q}}(\hat{\bar{x}})$. The explanation of $\breve{\bar{R}}(\hat{\bar{x}})$ is as follows:

In the truth state, the collection of discrete measurement noise was defined as $\bar{\eta}(t_k) = \left[\eta_{p_n}(t_k) \quad \eta_{p_e}(t_k) \quad \eta_{\psi}(t_k) \quad \bar{\eta}_{\Gamma_{[1]}}(t_k) \quad \bar{\eta}_{\Gamma_{[2]}}(t_k)\right]$. Each discrete measurement noise term is assumed to be Gaussian, being independent and uncorrelated with each other, the state, and dynamics (white). This assumption allows the discrete measurement noise to be statistically represented via mean and covariance. In a similar manner as the derivation for continuous noise mean and covariance, discrete noise and covariance are defined as

$$\begin{gathered} E\left(\bar{\eta}(t_k)\right) = \bar{0} \\ E\left(\bar{\eta}(t_k)\bar{\eta}^T(t_{k-1})\right) = \bar{R}\,\delta(t_k - t_{k-1}) \end{gathered} \tag{2.84}$$

This discrete form uses Kronecker Delta, $\delta(t_k, t_{k-1})$ (unitless). Furthermore, the $\bar{R}$ matrix is a diagonal of the variances of the measurement noise, $\bar{\eta}(t_k)$, such that

$$\bar{R} = \begin{bmatrix} \sigma^2_{ss_{p_n(t_k)}} & 0 & 0 & 0 & 0 \\ 0 & \sigma^2_{ss_{p_e(t_k)}} & 0 & 0 & 0 \\ 0 & 0 & \sigma^2_{ss_{\psi(t_k)}} & 0 & 0 \\ 0 & 0 & 0 & \sigma^2_{ss_{\Gamma_{[1]}(t_k)}} & 0 \\ 0 & 0 & 0 & 0 & \sigma^2_{ss_{\Gamma_{[2]}(t_k)}} \end{bmatrix} \tag{2.85}$$

Using (t_k) within the σ_{ss} subscripts helps to emphasize that the standard deviations are with respect to the discrete measurements' accuracies, not continuous sensors' or initial conditions' accuracies. Likewise, the design model yields similar results

$$\begin{gathered} E\left(\breve{\bar{\eta}}(t_k)\right) = \bar{0} \\ E\left(\breve{\bar{\eta}}(t_k)\breve{\bar{\eta}}^T(t_{k-1})\right) = \breve{\bar{R}}(\hat{\bar{x}})\delta(t_k - t_{k-1}) \end{gathered} \tag{2.86}$$

$$\bar{\breve{R}}(\hat{\bar{x}}) = \begin{bmatrix} \hat{\sigma}^2_{ss_{p_n(t_k)}} & 0 & 0 & 0 & 0 \\ 0 & \hat{\sigma}^2_{ss_{p_e(t_k)}} & 0 & 0 & 0 \\ 0 & 0 & \hat{\sigma}^2_{ss_{\psi(t_k)}} & 0 & 0 \\ 0 & 0 & 0 & \hat{\sigma}^2_{ss_{\Gamma_{[1]}(t_k)}} & 0 \\ 0 & 0 & 0 & 0 & \hat{\sigma}^2_{ss_{\Gamma_{[1]}(t_k)}} \end{bmatrix} \tag{2.87}$$

Now that the matrices are defined, the rest of this subsection details a validation algorithm to confirm correct implementation. Both $\bar{\breve{J}}(\hat{\bar{x}})$ and $\bar{\breve{R}}(\hat{\bar{x}})$ can be validated together through the covariance update equation itself. The idea is, just like the 2.1.6 Covariance Propagation subsection, that there is intuition that the discrete measurement noises will manifest themselves according to specific patterns within a Monte Carlo simulation. The patterns are zero-mean and saw-tooth with position and velocity magnitudes being significantly smaller than the 2.1.6 Covariance Propagation subsection.

The same simulation set up as the 2.1.6 Covariance Propagation subsection is used with the exceptions that this subsection activates discrete measurement noise, that unmodified time constants are always used, that Kalman filter update cycles are used, and that the same truth state is used for every Monte Carlo. To explain the first two exceptions, the discrete noise measurements for each discrete time step are defined as

$$\bar{\eta}(t_k)_{[*]} = R_{\bar{\eta}(t_k)_{[*]}}\zeta \tag{2.88}$$

For time constants, the author used the same time constants listed in Table 2.6. For the third exception, the update cycles are iterations of calculating the residual ($\tilde{\bar{z}} - \hat{\bar{\hat{z}}}^-$, Equation (2.53)), $\bar{\breve{H}}_{\breve{\bar{x}}}\big(\hat{\bar{x}}^-(t_k)\big)$ (Equation (2.67)), $\hat{\bar{K}}$ (Equation (2.83)), $\hat{\bar{P}}^+$ (Equation (2.42)), $\delta\hat{\bar{x}}^+$ (Equation (2.41)), and the $\bar{l}$ mapping (Equation (2.36), $\bar{x} \approx \hat{\bar{x}}^+ = \hat{\bar{x}}^- + \delta\hat{\bar{x}}^+$) in that respective order to calculate a better estimate of the navigation state and covariance. The reason for cycling is a matter of increased measurement update accuracy and of numerical stability.

To better explain the update cycles, consider, rather than processing all the discrete measurements at once, processing the discrete measurements one at a time. Would there be a difference in update accuracy if, say, camera measurements were processed first rather than GPS measurements? The answer is yes. Recalling the 2.1.1 States and Models subsection, GPS and magnetometer measurements were linear, explicit functions of UAV position and heading respectively, whereas camera measurements were nonlinear, implicit functions of UAV position, landmark position, camera offset bias, and heading. Being a nonlinear function of multiple states makes camera measurements quite sensitive to errors within those multiple states. Therefore, there is higher priority in ensuring that those multiple state are as accurate as possible before processing the camera measurement. Furthermore, consider the Kalman gain weighting matrix (Equation (2.83)). This equation requires a matrix inversion (note that the terms within this inverse are collectively called the residual covariance). Inversions greater than 3x3, where 3x3 matrices have analytical solutions, begin to hinder computation time as well as numerical stability of that inversion. To that end, there is a preference to execute Kalman filter updates in cycles of at most 3x3 sized update matrices beginning with the most reliable, linear measurement followed by progressively more nonlinear measurements.

Creating the update matrices is simply breaking down the discrete measurement matrices $\bar{\breve{J}}(\hat{\bar{x}})$, $\bar{\bar{\breve{H}}}_{\breve{\bar{x}}}\left(\hat{\bar{x}}^{-}(t_k)\right)$, $\bar{\breve{R}}(\hat{x})$, and the residual into smaller matrices that correspond to the measurement of each update cycle. For coding simplicity, especially during measurement denial and debugging, each measurement was processed one at a time, which makes each cycle of $\breve{\bar{J}}(\hat{\bar{x}})$ and $\breve{\bar{R}}(\hat{\bar{x}})$ equal a 1x1 matrix of 1 and $\hat{\sigma}_*^2$ respectively, and each cycle of $\breve{\bar{\bar{H}}}_{\breve{\bar{x}}}\left(\hat{\bar{x}}^-(t_k)\right)$ and residual are single row matrices.

Included in the update cycles are protocols that account for measurement denial. When there is GPSD or a landmark is outside the camera's FOV, the position or camera measurement is not processed as there is no $\tilde{\bar{z}}$ available to calculate the residual. This is done in simulation by nulling out all elements of $\hat{\bar{\bar{K}}}$ during the applicable update cycle. Note that, as is common practice, $\hat{\bar{\bar{K}}}$ elements are set to null if a measurement residual is outside the 3σ equivalent of that measurement's residual covariance (see second following paragraph). This practice is called residual monitoring. The author found that erroneous residuals did not occur frequently enough to, as Maybeck commented, "not to reject large residuals, since they are the only means of correcting the divergence" [3].

In regard to the fourth exception, using the same truth state for each Monte Carlo, there is extreme difficulty in validating this subsection when each Monte Carlo, corrupted by noise and nonlinear measurement updates, is trajectory dependent. Therefore, to reduce trajectory dependence as much as possible, the noise placed in the truth state's initialization was removed and placed in the navigation state's initialization for each Monte Carlo. Control inputs were changed such that $\dot{u} = 8.25 + 8.25 * \sin\left(\frac{t}{10}\right)$ to create the same thrust in each trajectory. Furthermore, the $\breve{\bar{\bar{H}}}_{\breve{\bar{x}}}\left(\hat{\bar{x}}^-(t_k)\right)$ matrix used truth state values instead of navigation state values, $\breve{\bar{\bar{H}}}_{\breve{\bar{x}}}\left(\bar{x}^-(t_k)\right)$.

With this simulation set up, two sets of plots are generated. The first set of plots is of the measurement residuals. According to Maybeck [3], these residuals have zero mean and a covariance matching the residual covariance in Equation (2.83). In these first plots a single Monte Carlo simulation is run to verify consistency between Maybeck and implementation. The use of a Mahalanobis distance to analyze measurement residuals is left to future work, particularly for physical implementation when the additional complexities of image processing, magnetometer dead zones, sensor failure, and satellites moving past the horizon create a need for alternative statistics evaluation. The second set of plots will be error plots of a 200 Monte Carlo simulation with Kalman filtering active. The activated Kalman filtering should induce sawtooth patterns in both the 3σ statistics and in the hairlines themselves, where a sawtooth fall indicates that the discrete measurements helped reduce estimation errors. These two sets of plots are generated below for the three trajectories used in the 2.1.6 Covariance Propagation subsection, with the addition of GPSD regions along the trajectories. First the Monte Carlo trajectories will be shown, followed by the residual plots, and then the state error plots. Note that seven trajectories for both commercial and navigation grade sensors were tested for correct implementation.

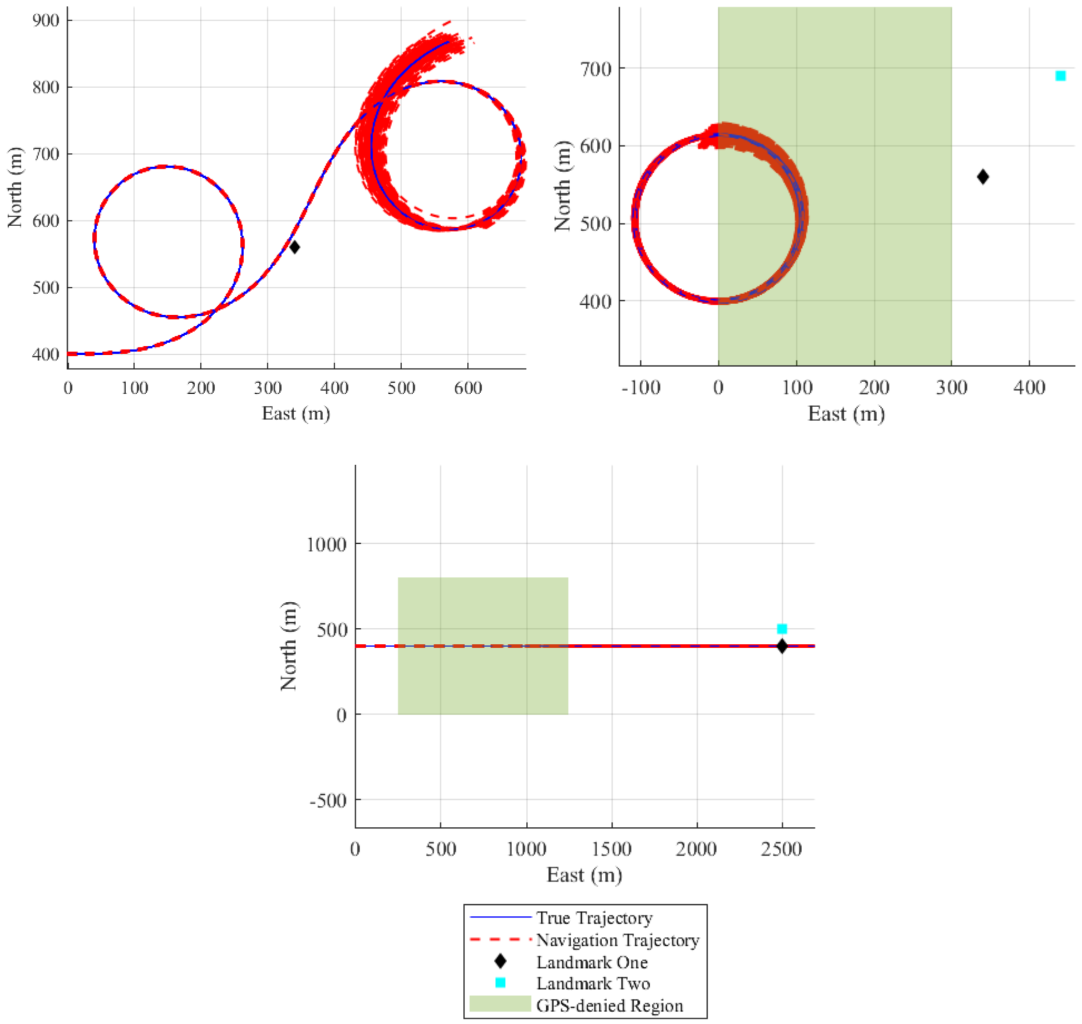


Figure 2.39. Estimation capability trajectories.

Within Figure 2.39, the GPSD regions were set as boxed regions for simplicity. For the first trajectory, GPSD is during the last 30 s. The edges of these regions are defined in Table 2.7.

Table 2.7. GPSD region edges.

Trajectory	North Max Edge	North Min Edge	East Max Edge	East Min Edge	Units
Full Dynamics	NA	NA	NA	NA	m
Constant Rudder	800	200	300	0	m
Null Rudder	800	0	1250	250	m

Only one landmark is used for the full-rudder trajectory to reduce the complexities in understanding discrepancies. The landmarks are positioned in the constant-rudder trajectory such that the UAV is only able to take landmark images while in GPSD. For the null-rudder trajectory, the UAV constantly has view of the landmarks until the last seconds of the simulation.

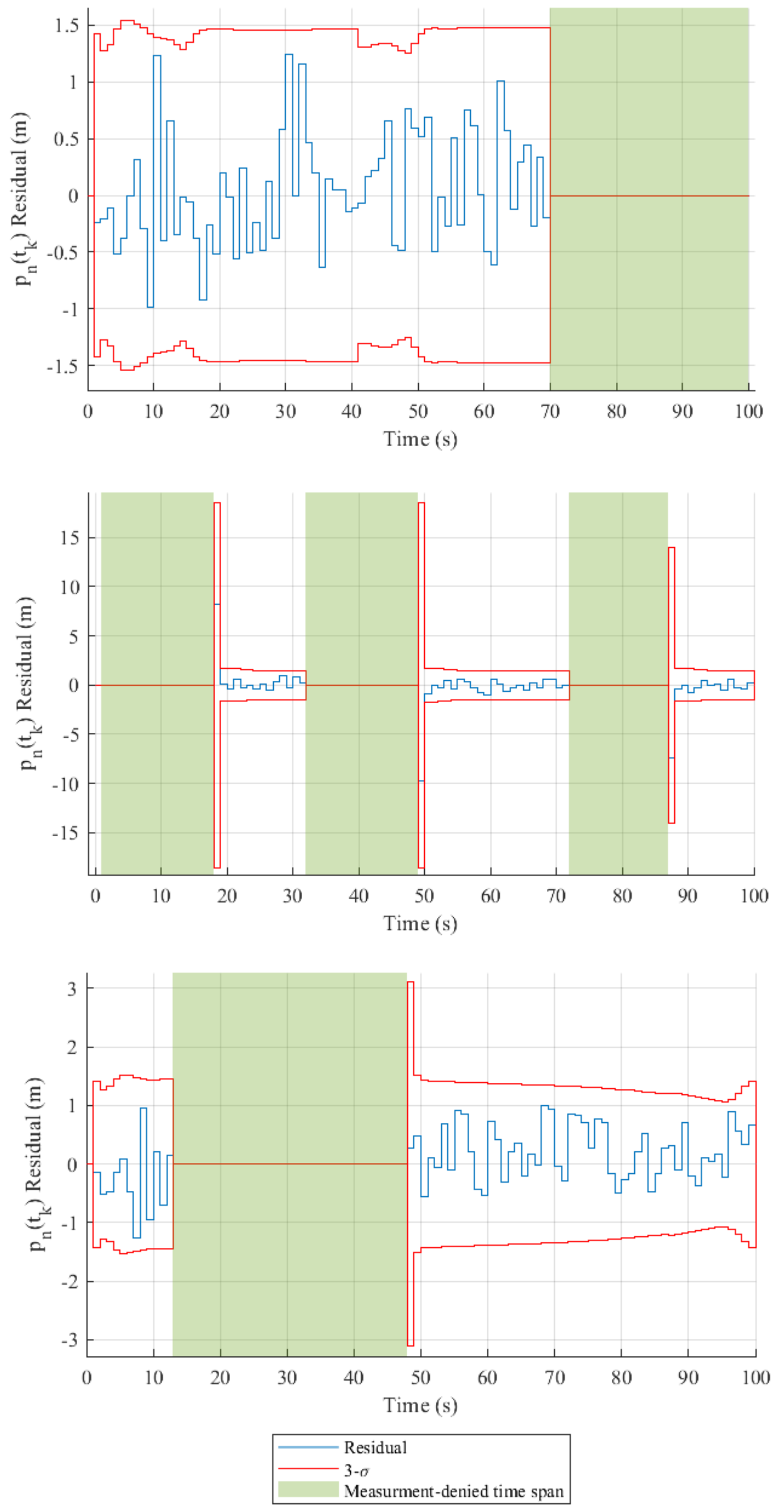


Figure 2.40. Estimation capability $\tilde{p}_n$ residual.

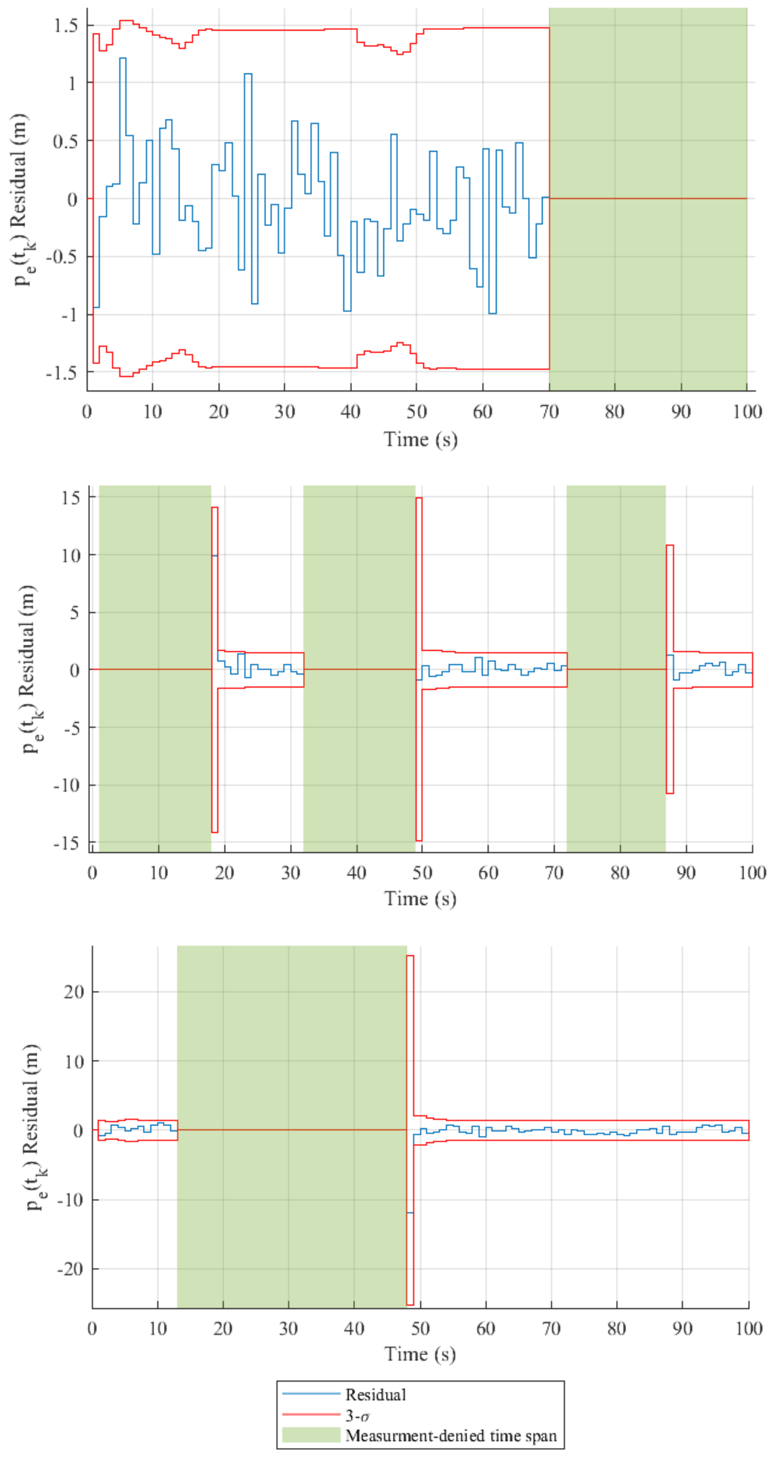


Figure 2.41. Estimation capability $\tilde{p}_e(t_k)$ residual.

Showing the GPS position residuals in Figure 2.40 and Figure 2.41, GPS is denied for one period, three periods, and one period for the full-rudder, constant-rudder, and null-rudder respectively. After periods of denial, the plots have one Kalman cycle of relatively high residual covariance. After this cycle, the residuals are able to return to a reduced, zero-mean, white trend. Notice at the end of the null-rudder trajectory, there is a shallow-sloped decrease of p_n 3σ residuals followed by a dramatic-sloped increase in the 3σ residuals. This is a manifestation of extreme $\bar{\Gamma}(t_k)$ ratios seen in the 2.1.3 Propagation Validation subsection, which will be discussed later in this section.

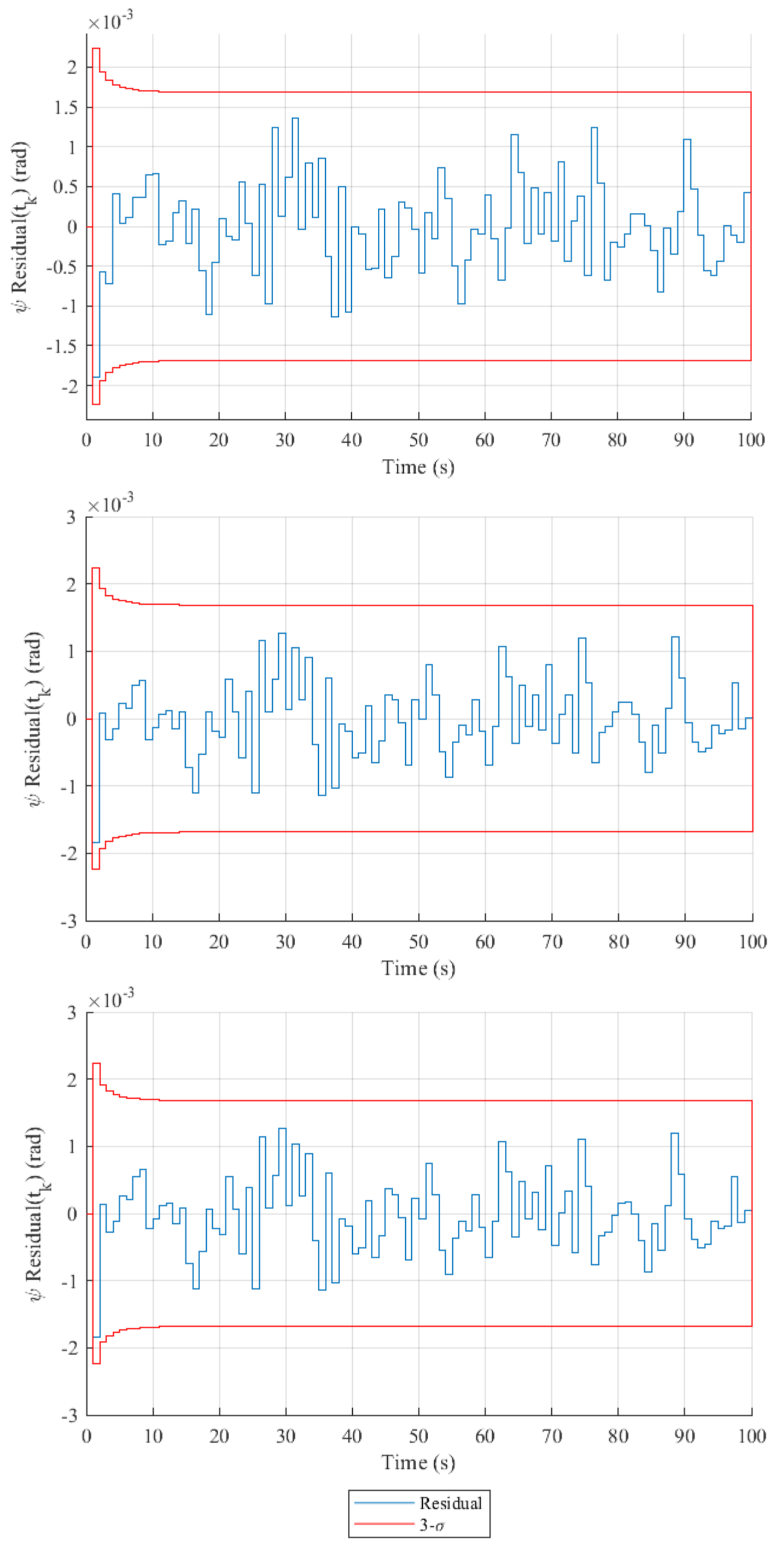


Figure 2.42. Estimation capability $\tilde{\psi}(t_k)$ residual.

Heading for all trajectories produced very similar results. In other words, since magnetometer jamming is not present in this simulation, the linear nature of heading makes it less dependent on trajectory paths. Though the magnetometer residuals for this single Monte Carlo do not span the full width of the residual covariance, the residuals still appear zero-mean, white, and are sufficiently large enough to justify correct implementation.

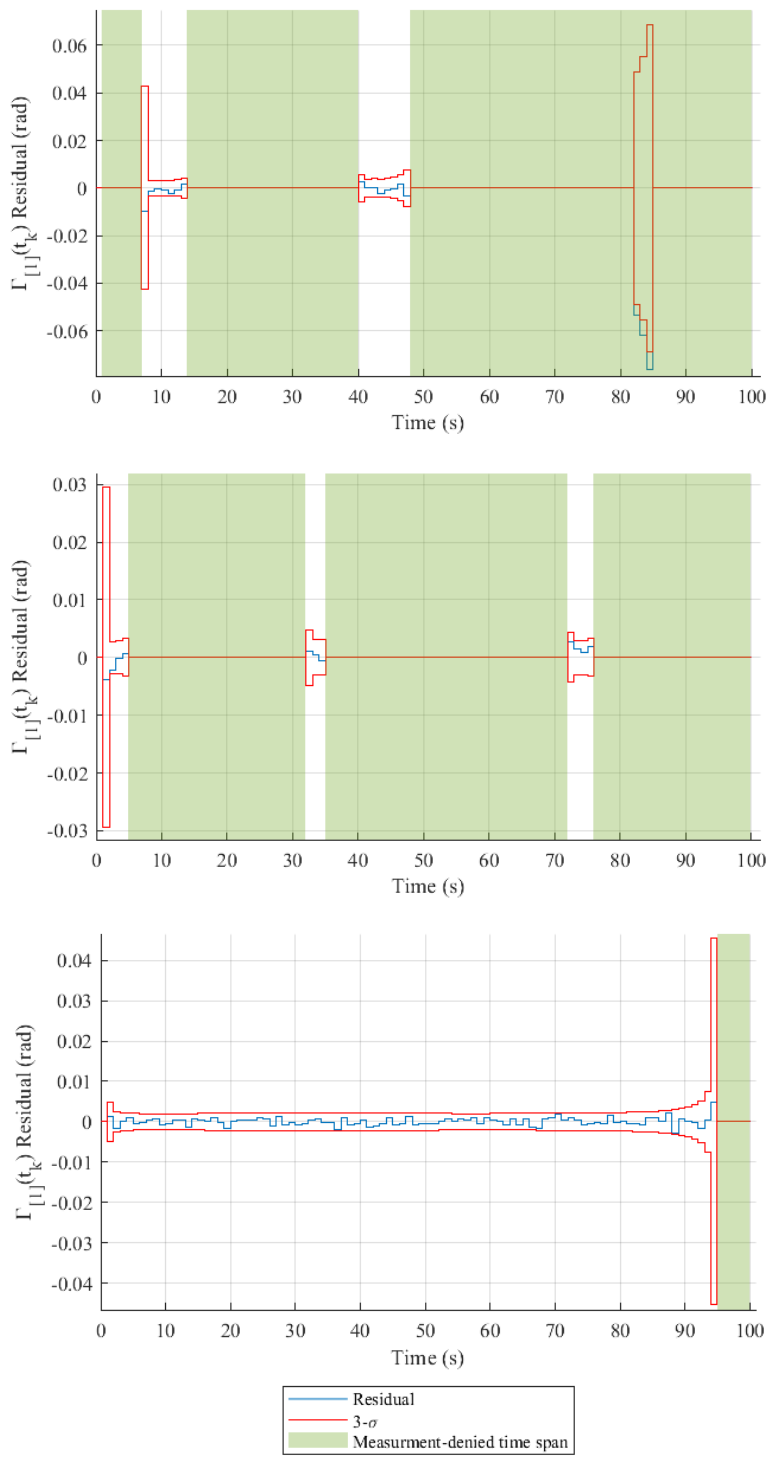


Figure 2.43. Estimation capability $\tilde{\Gamma}_{[1]}(t_k)$ residual.

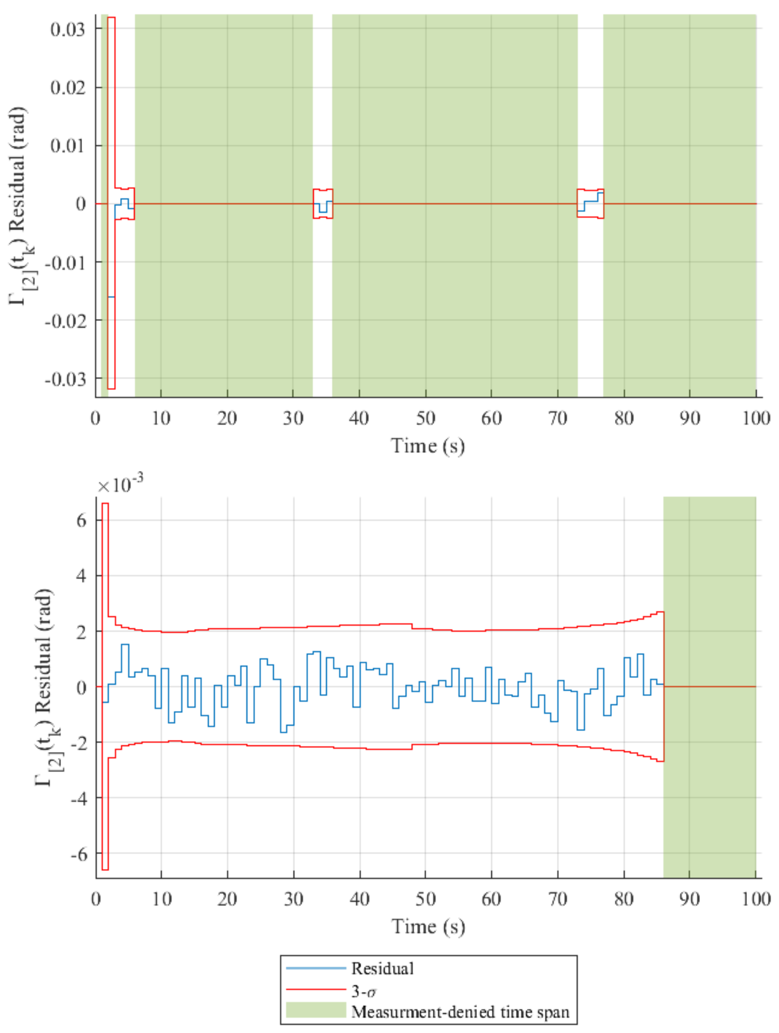


Figure 2.44. Estimation capability $\tilde{\Gamma}_{[2]}(t_k)$ residual.

For the first two trajectories in Figure 2.43 and Figure 2.44 (recall only one landmark is used in the first trajectory), large periods of measurement denial are present, so much so that there is difficulty in confirming that the residuals are zero-mean and white. Nevertheless, processed residuals are within 3σ of the residual covariance. Residuals outside 3σ, $\Gamma_{[1]}$ at 84 s in the first trajectory, are rejected. After measurement denial, these plots do not show large increases of residual covariances like GPS measurements did—they become large when both GPS and camera measurements have been denied for a longer period (With the exception of the first camera update). Looking at the last trajectory, where landmarks were always in the FOV, the zero-mean, white residual trend of the camera measurements are manifest. Note how the null-rudder trajectory still manifests increased error upon the approach of extreme $\bar{\Gamma}(t_k)$ ratios.

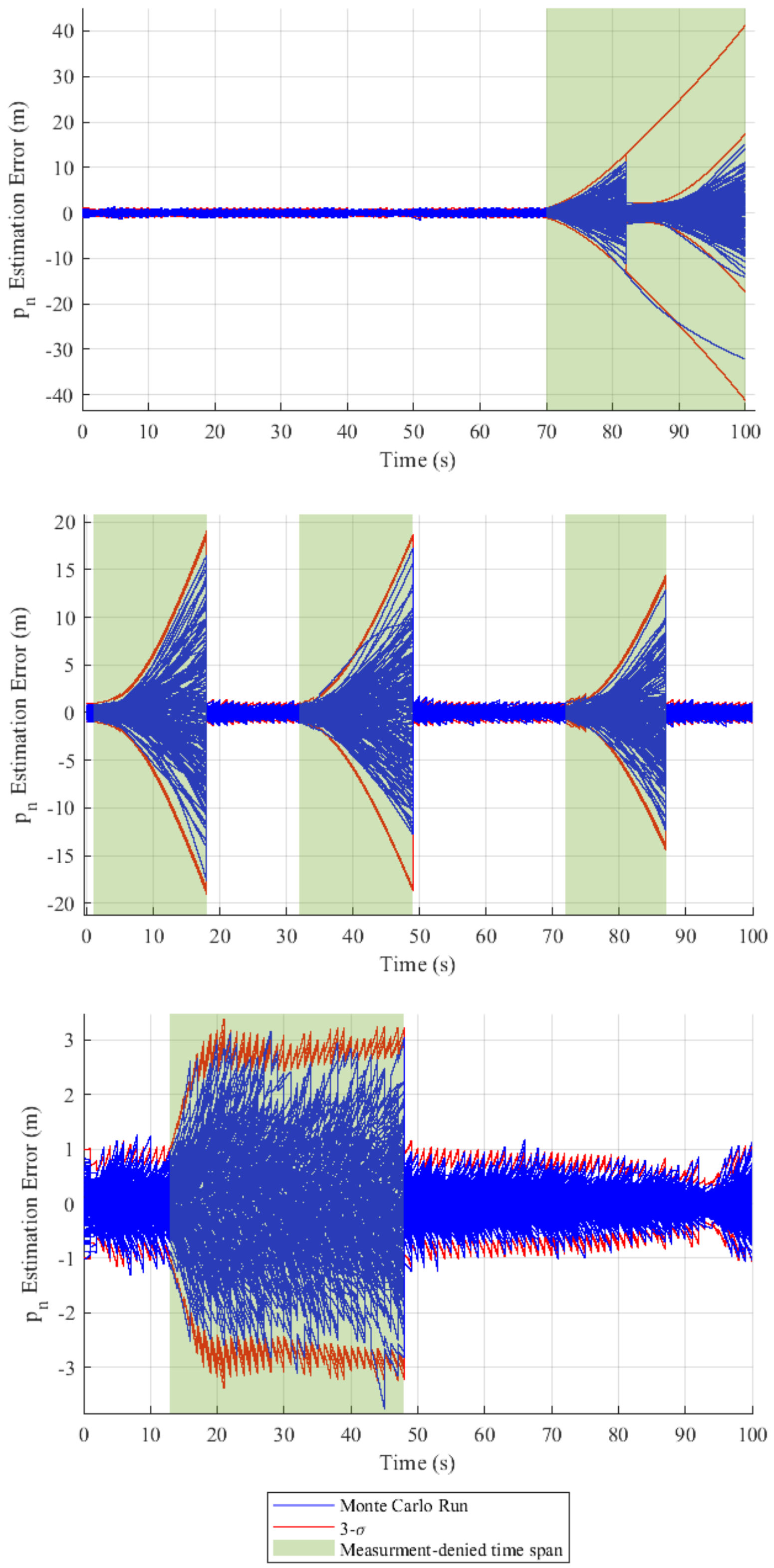


Figure 2.45. Estimation capability p_n errors.

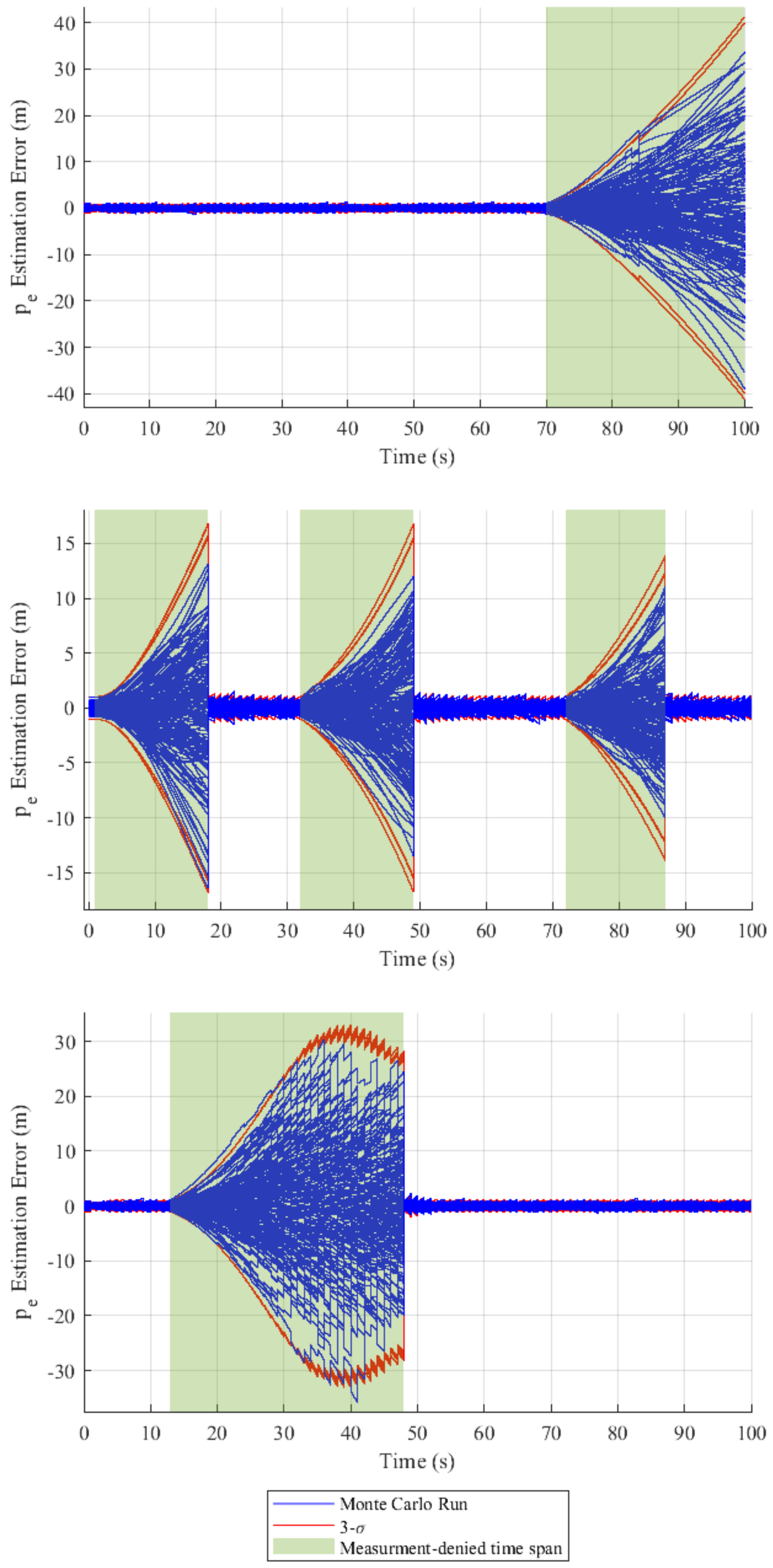


Figure 2.46. Estimation capability p_e errors.

For Figure 2.45and Figure 2.46, since covariance is more trajectory dependent in this subsection, all 200 3σ lines are included. Especially in the first trajectory, the divergence of covariance lines is most readily seen. Less dramatic diversions are from the system noise and discrepancies in the $\breve{\bar{\bar{F}}}_{\breve{\bar{X}}}(\hat{\bar{x}})$ matrix. More dramatic diversions are from rejected camera updates when a particular Monte Carlo had a residual outside 3σ. The trend for most position errors is the same. Upon GPS denial, errors begin to quadratically increase as they did in the prior subsection. Upon GPS availability, the errors drastically reduce and begin displaying the desired sawtooth trend over time. When available, camera measurement updates can play a critical part in correcting errors. Particularly with the null rudder, where the H matrix provides information orthogonal to the direction of sight, P_n errors are kept below 4 m of error! However, as shown in the constant-90° trajectory, minimal camera updates occurring only at the beginning of the GPSD are insignificant in aid. Nevertheless, an appreciation for combining all discrete measurements is manifest in the amount of error reduction there is compared to the 2.1.6 Covariance Propagation subsection. Of the tested trajectories, a maximum of 70 m and 4 m errors were generated for commercial and navigations sensor respectively. Note again, the covariance fluctuation at the end of P_n and during GPSD for P_e in the null rudder trajectory being affected by $\bar{\Gamma}(t_k)$ ratios.

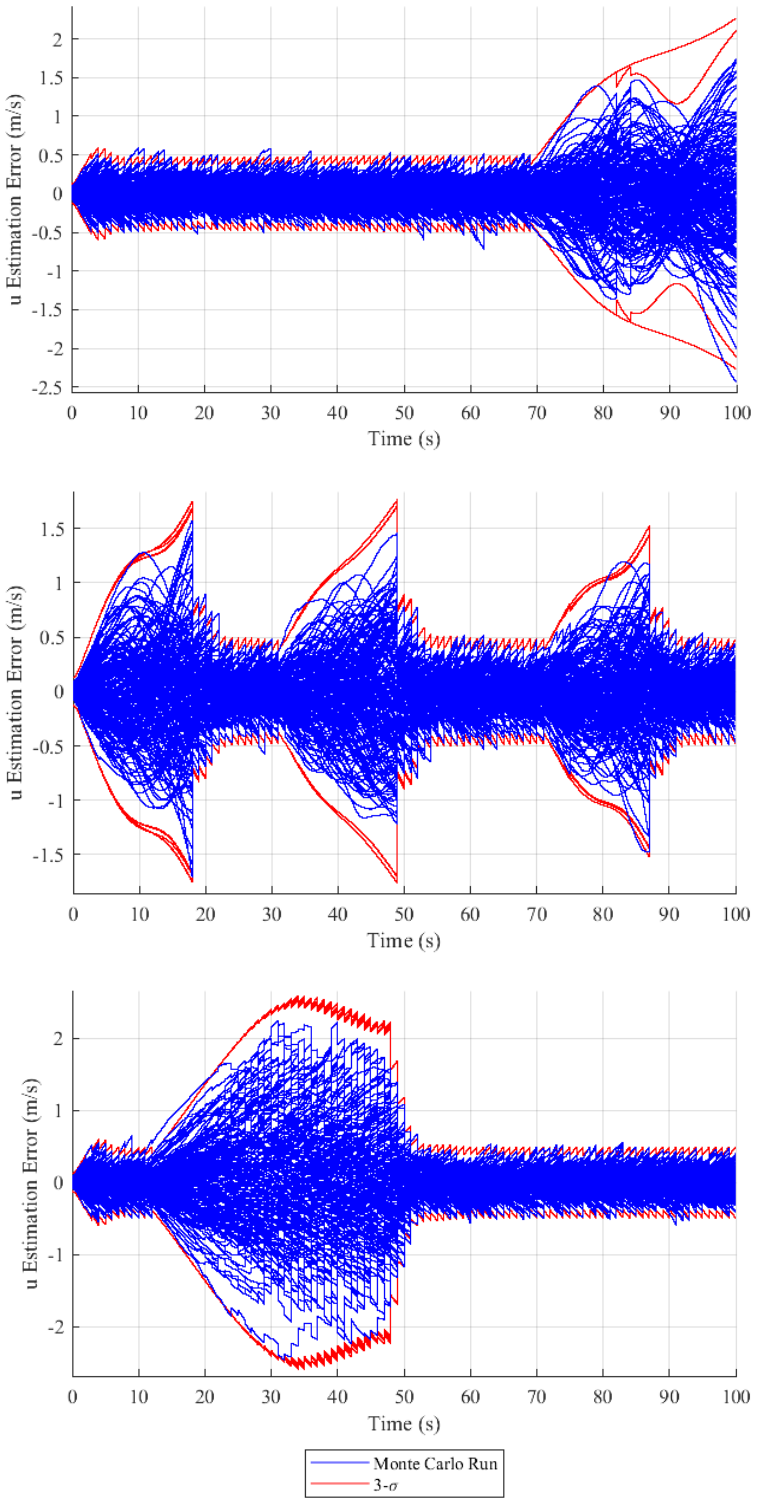


Figure 2.47. Estimation capability u errors.

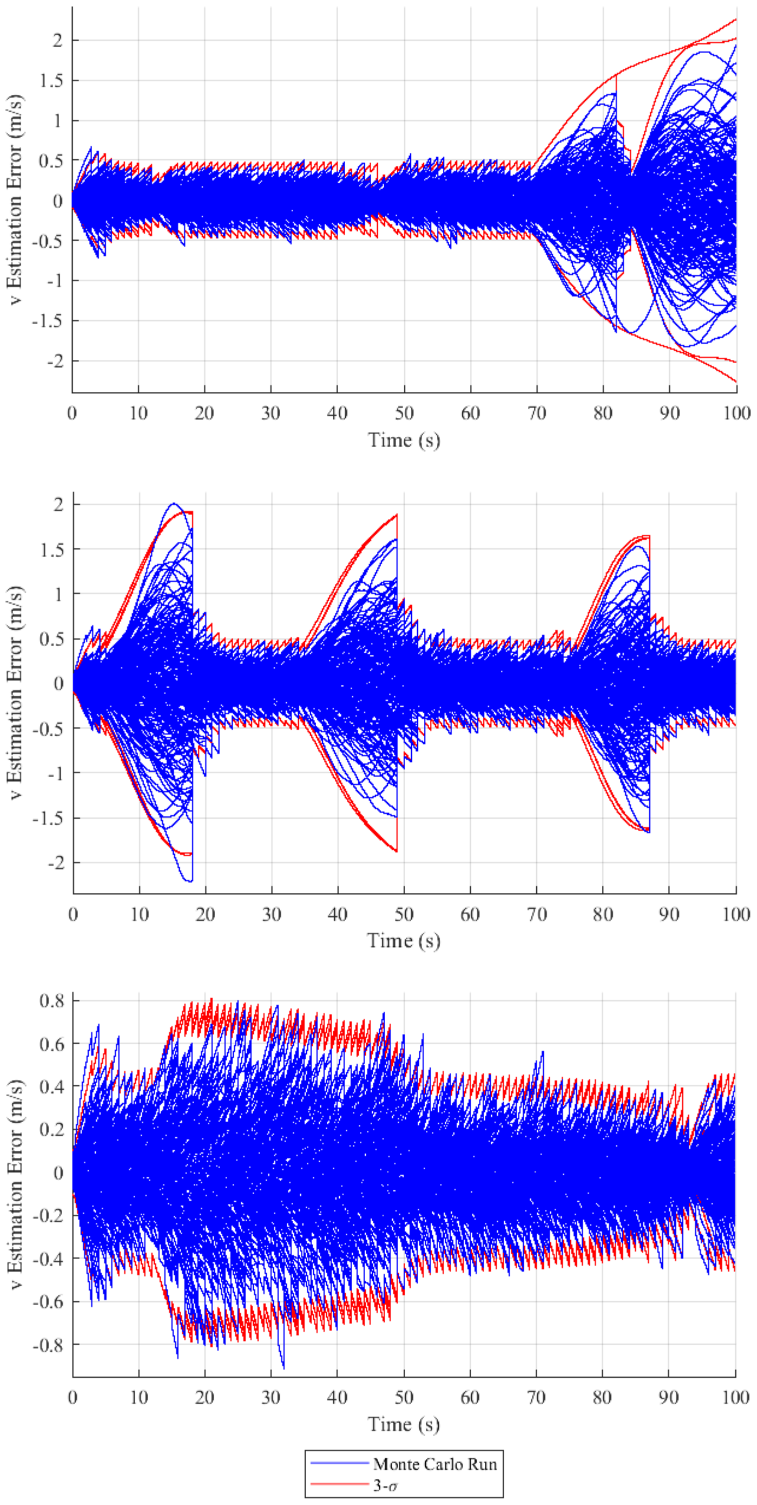


Figure 2.48. Estimation capability v errors.

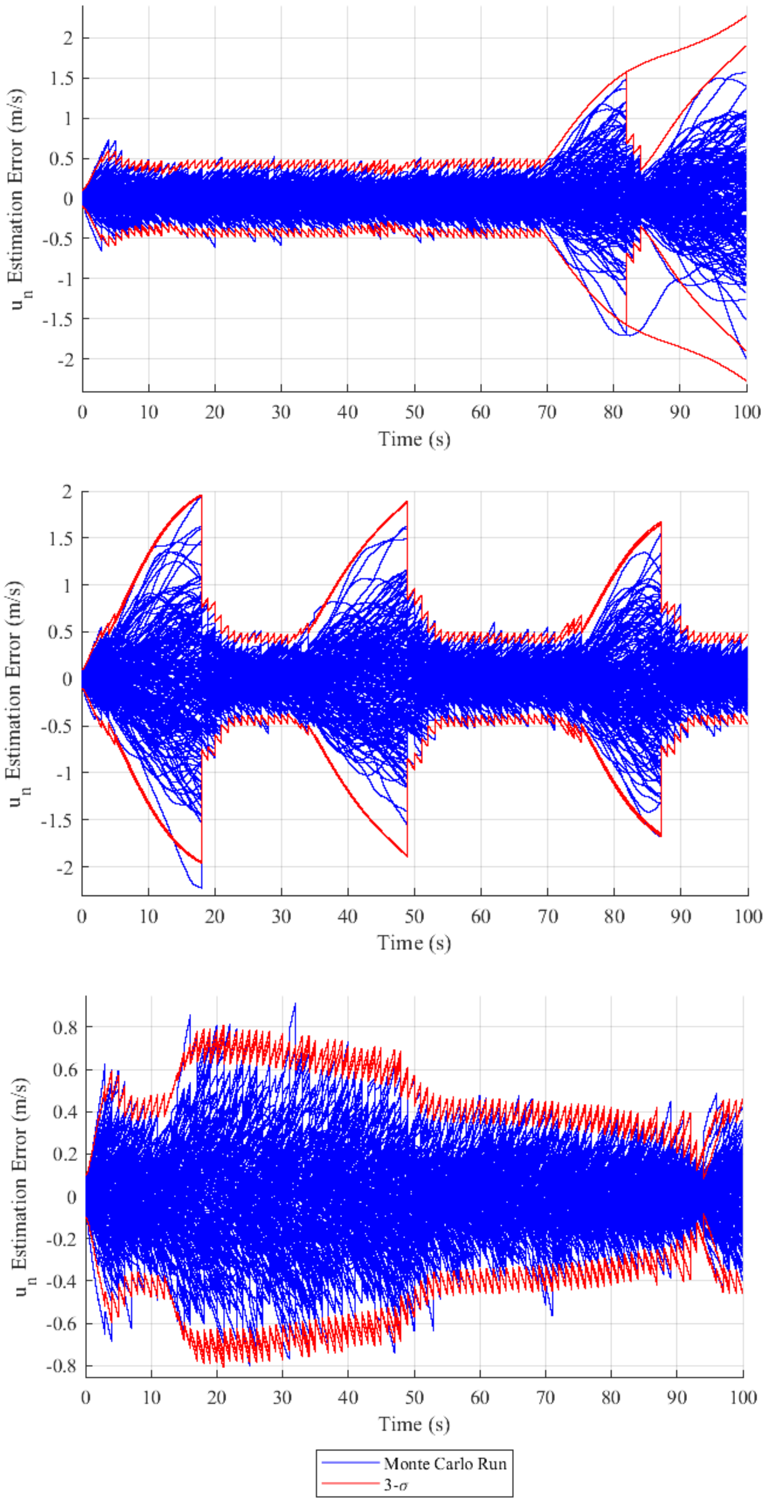


Figure 2.49. Estimation capability u_n errors.

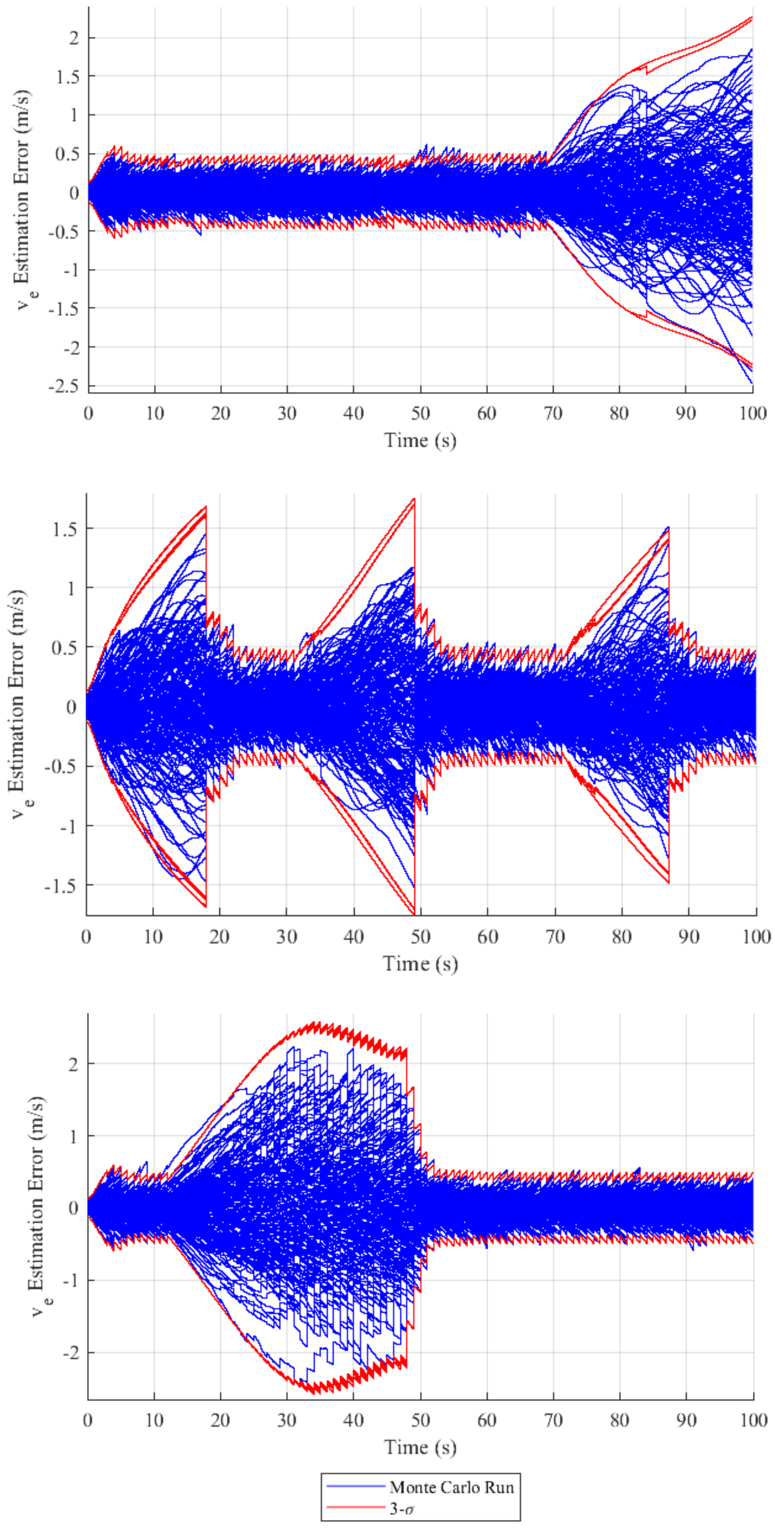


Figure 2.50. Estimation capability v_e errors.

Position errors mimic the velocity errors in Figure 2.47 through Figure 2.50. So do the velocity covariance bands, where, especially during GPSD, the bands can significantly differ from one another. Nevertheless, these bands still encapsulate the 3σ statistics of Monte Carlo errors.

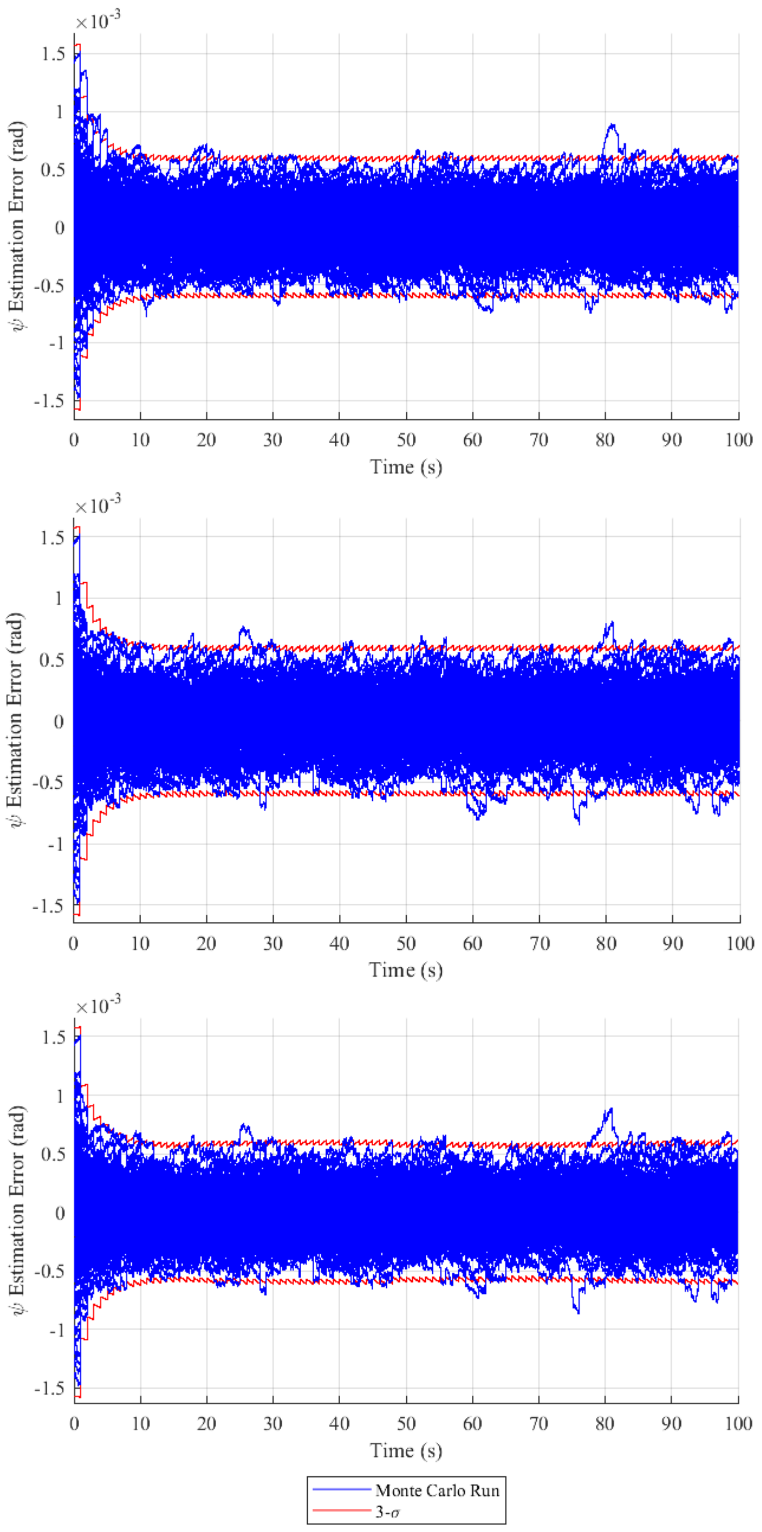


Figure 2.51. Estimation capability ψ errors.

Heading errors have minimal differences from trajectory to trajectory. All three trajectories minimize heading errors within the first 10 s of the simulation.

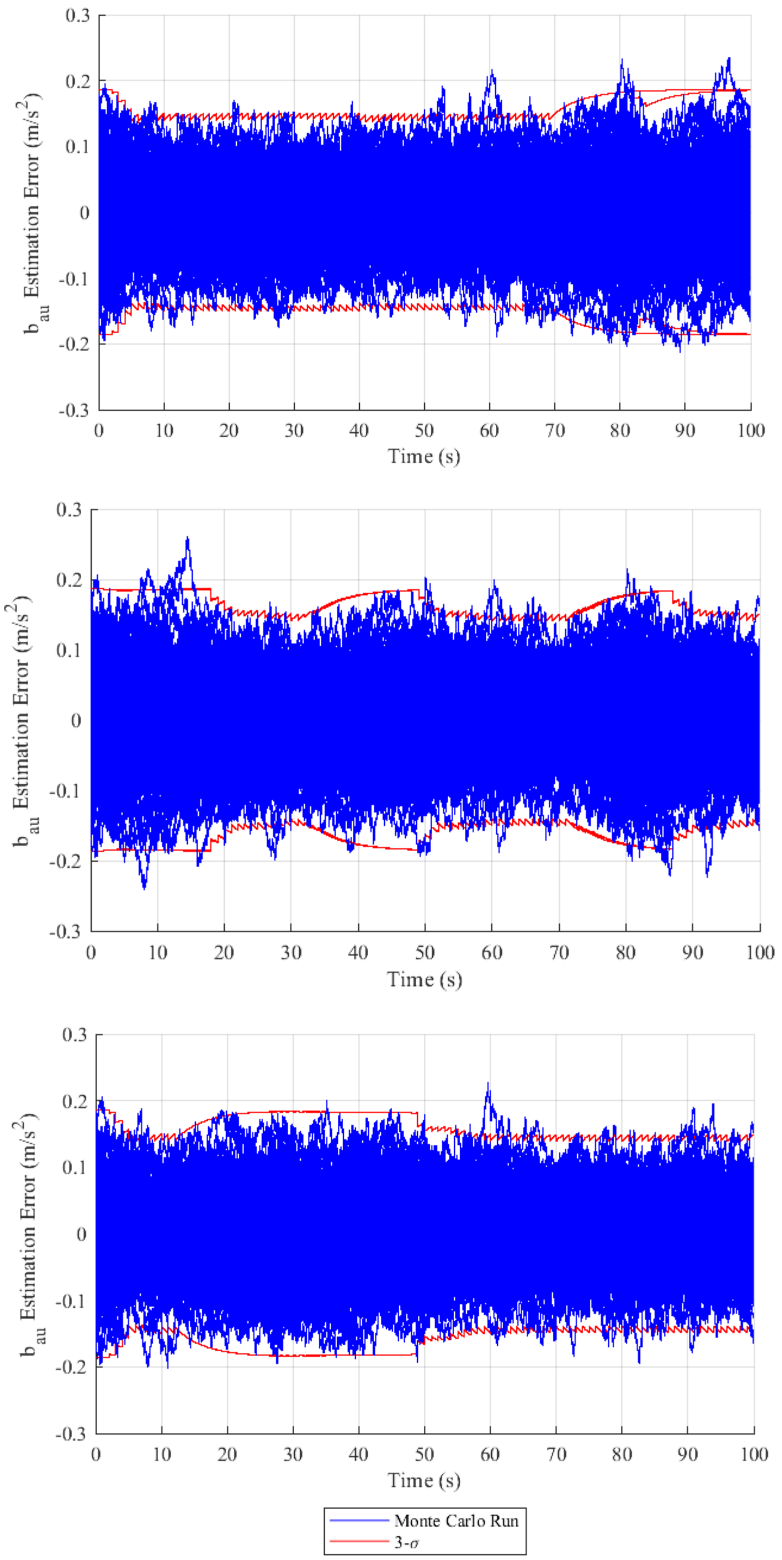


Figure 2.52. Estimation capability b_{a_u} errors.

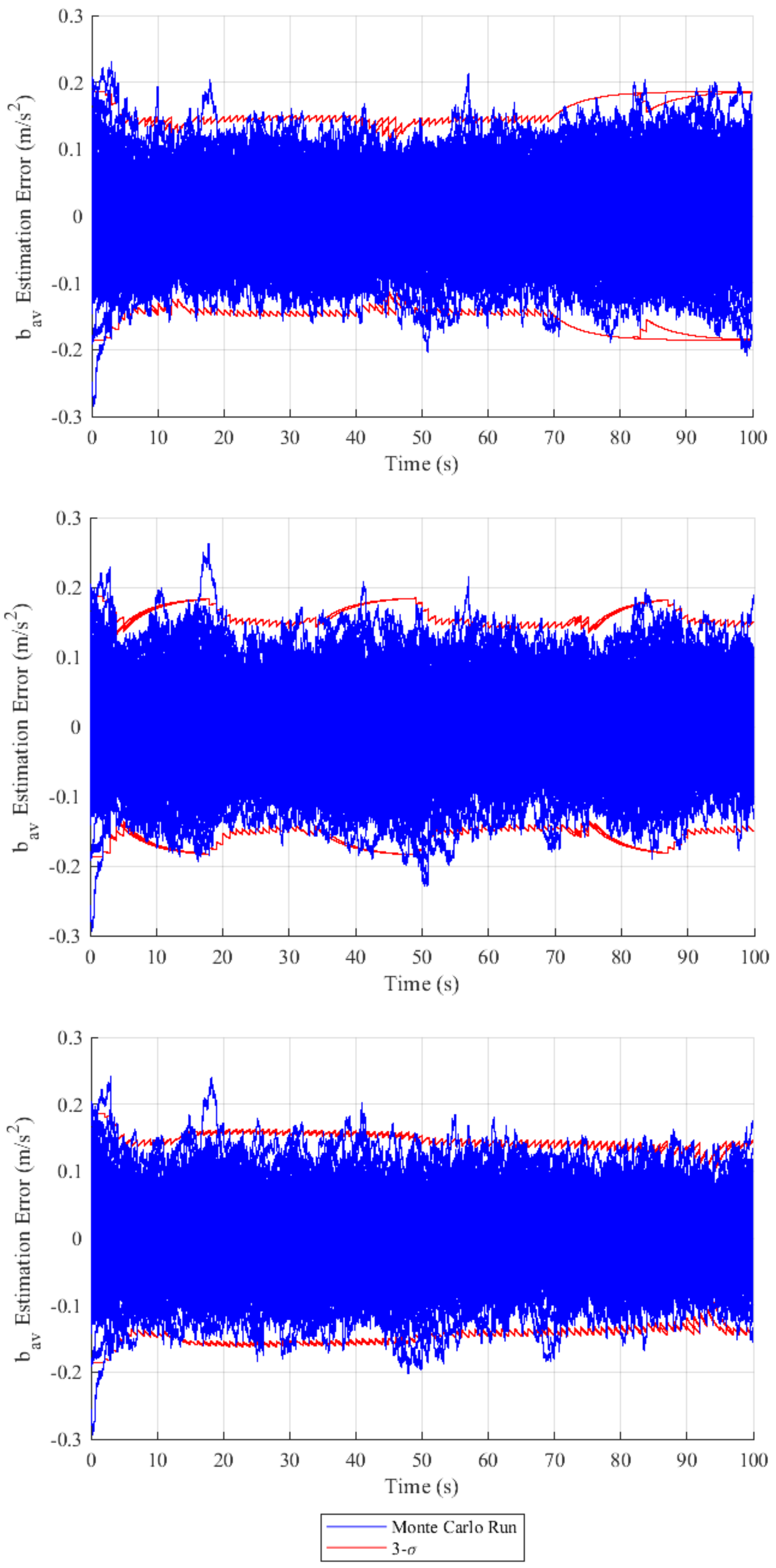


Figure 2.53. Estimation capability b_{a_v} errors.

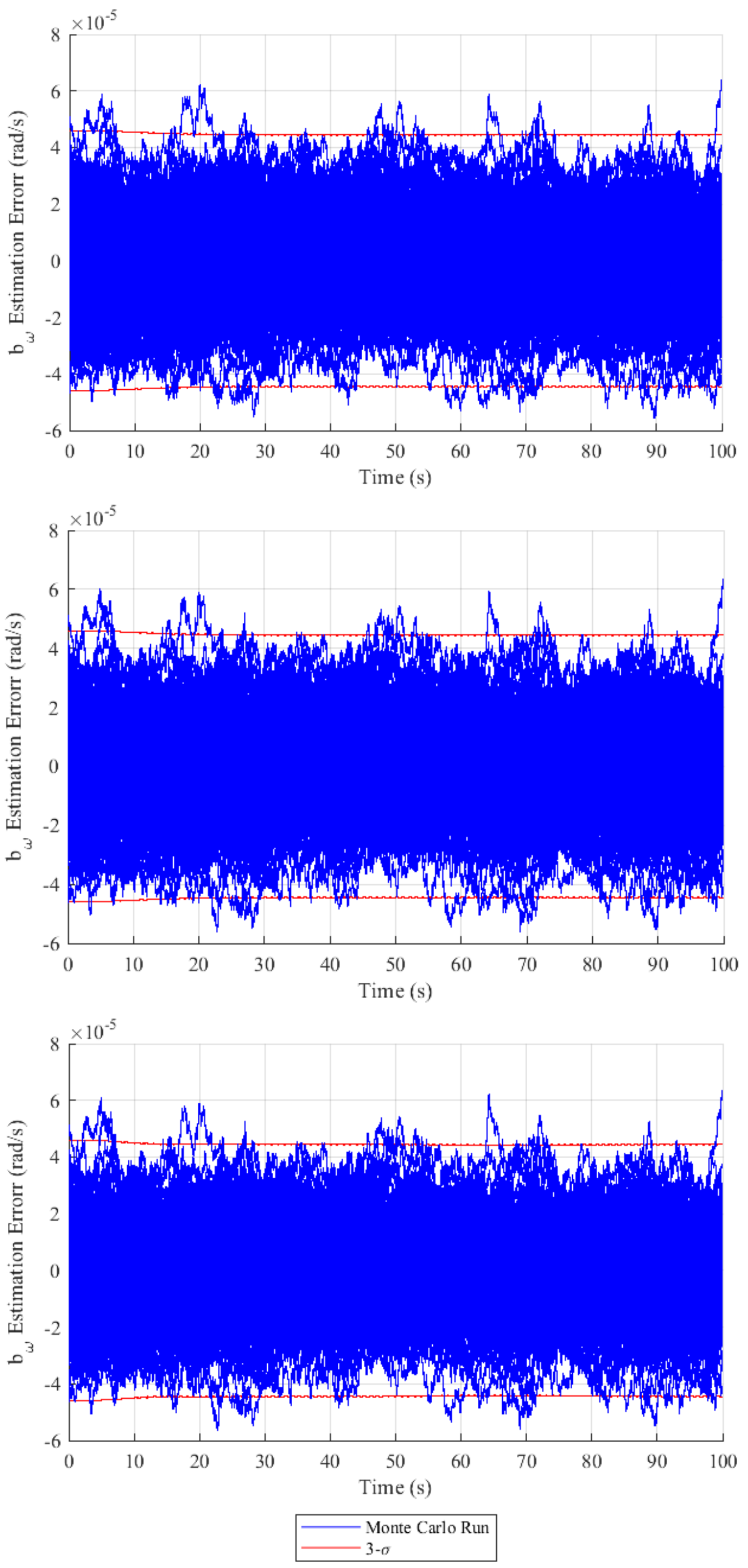


Figure 2.54. Estimation capability b_ω errors.

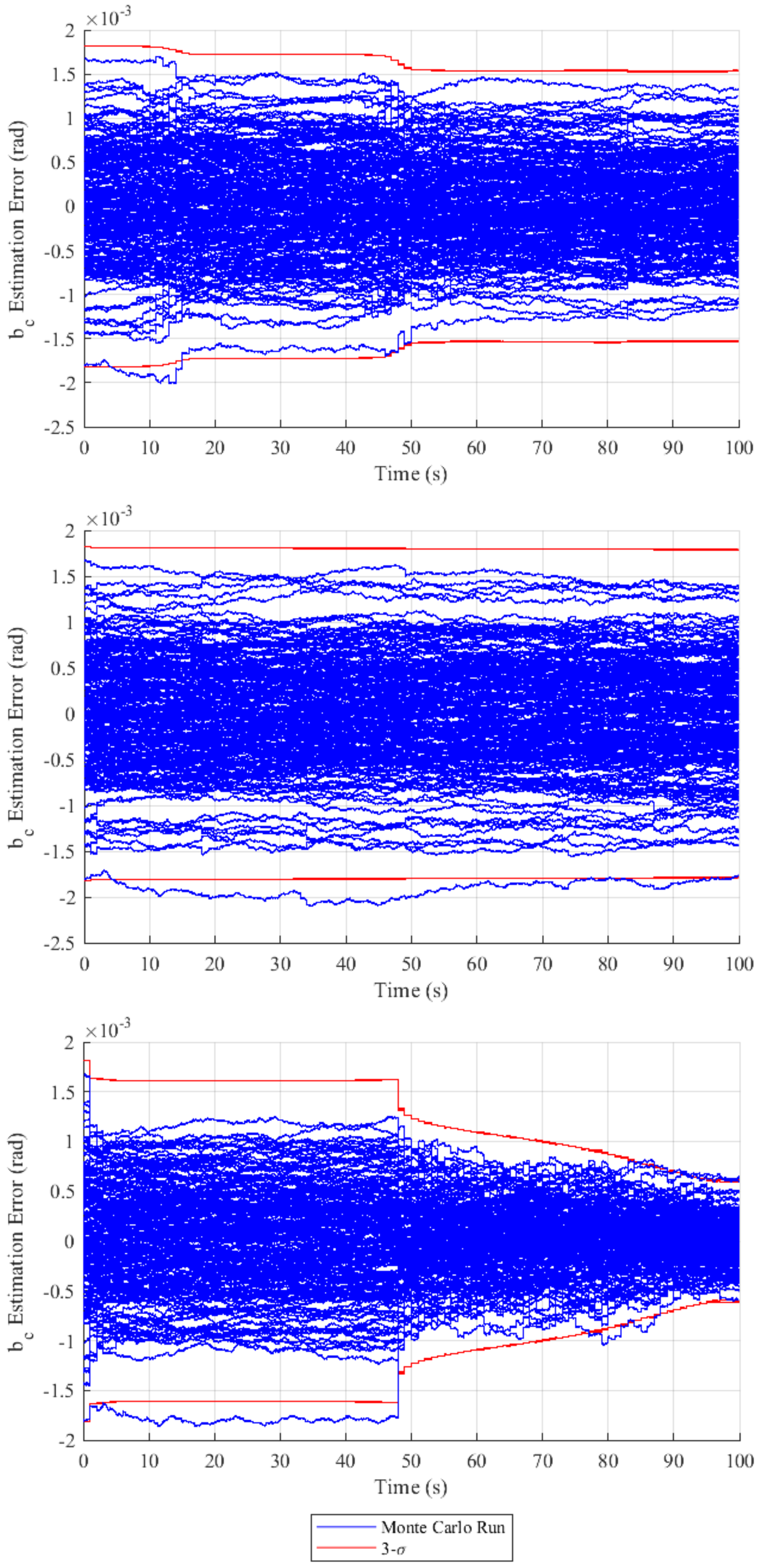


Figure 2.55. Estimation capability b_c errors.

With the very noise-like characteristics of the commercial sensors, the filter has difficulty reducing bias errors significantly over time. The exception to this rule is with the camera bias for null-rudder trajectories, where prolonged periods of camera measurements significantly aid in narrowing the 3σ bandwidth.

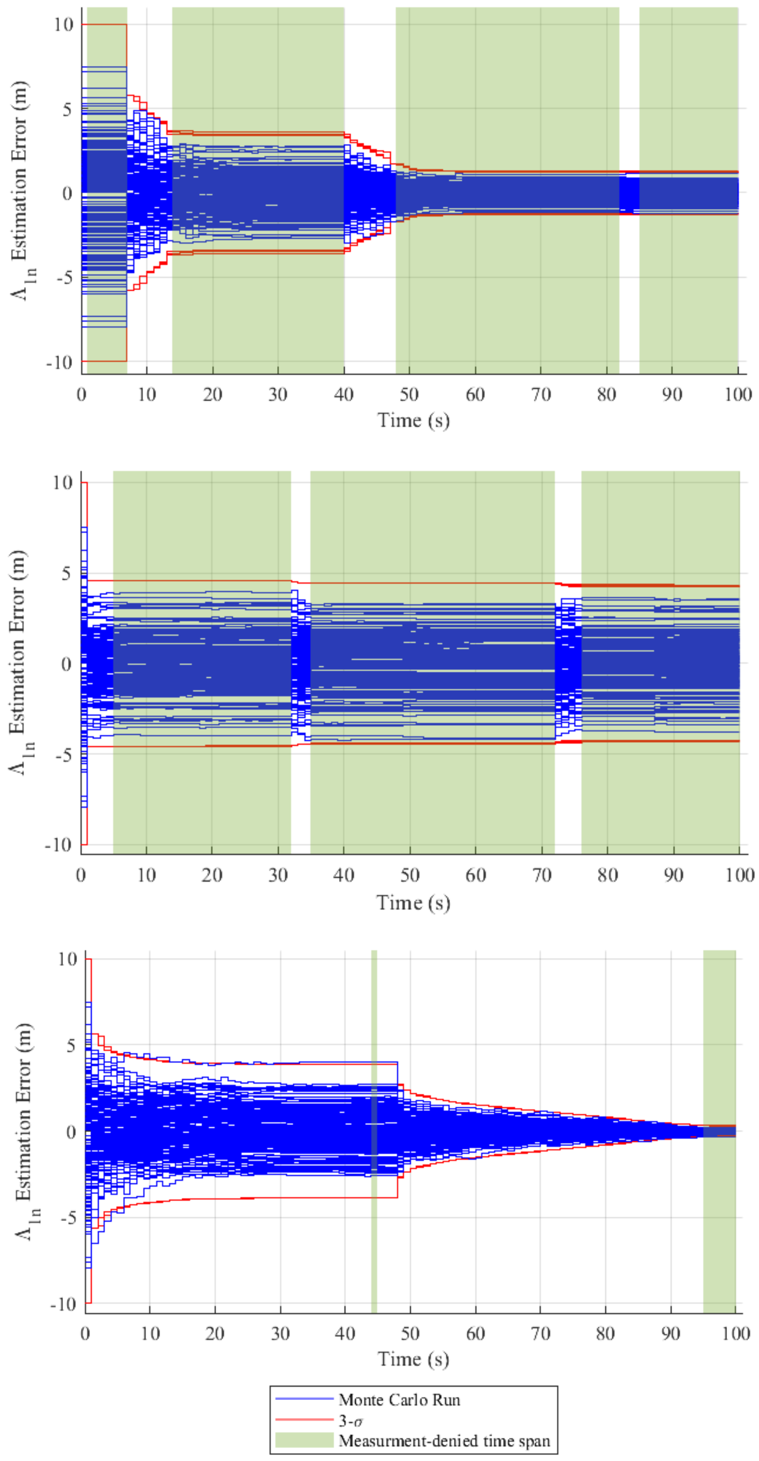


Figure 2.56. Estimation capability Λ_{1n} errors.

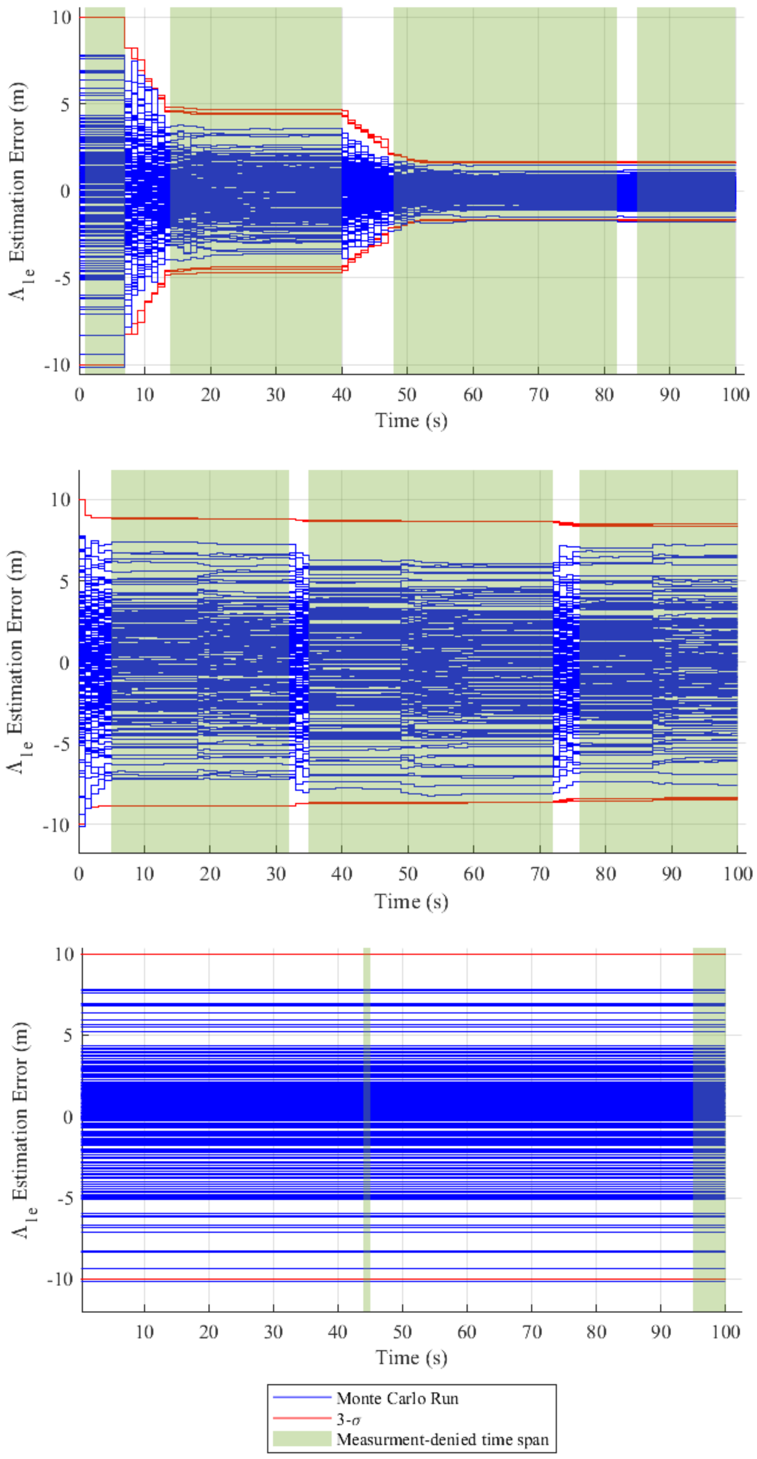


Figure 2.57. Estimation capability Λ_{1e} errors.

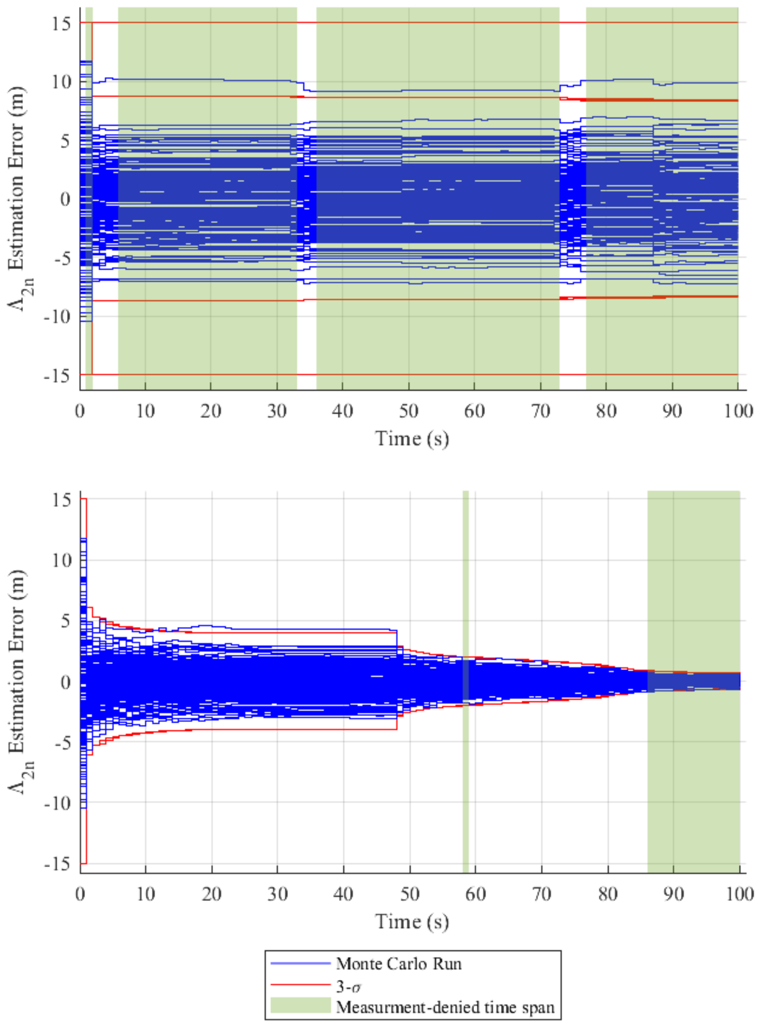


Figure 2.58. Estimation capabilityΛ_{2n} errors.

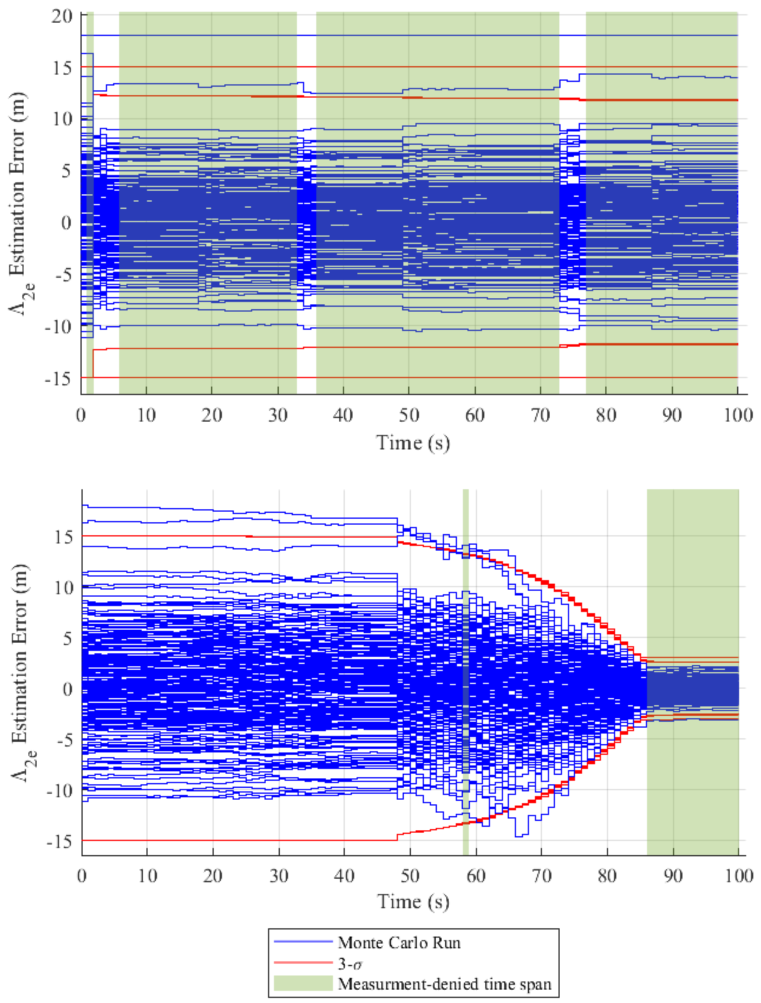


Figure 2.59. Estimation capability Λ_{2e} errors.

For the landmark errors in Figure 2.56 through Figure 2.59, again, shaded regions are camera measurement denials from one Monte Carlo to visualize the general denial regions (and to note the random occurrences of 1 s, outside 3σ, measurement rejections). In the full-dynamics trajectory, the system has the potential of converging landmark error to 1 m 3σ. For the constant-rudder trajectory, errors are only minimally able to reduce errors at the beginning. Brief moments of measurements during GPSD are not substantial enough to yield valuable landmark update information. In terms of landmark north-position for the last trajectory, errors reduce below 5 m 3σ during GPSD and converge to 1 m 3σ in GPS availability. In terms of landmark east-position for the last trajectory, little aid comes from the camera except near the trajectory's end for the second landmark, where the γ angle is offset from the north position enough for the H matrix to provide information for the landmark's east position. These and the prior ensemble plots show proper results, where the variance statistics are representative of the ensemble statistics.

Concerning the $\bar{\Gamma}(t_k)$ ratios, the success for having the 3σ statistics match the hairline plots is conditional on having appropriate $\bar{\Gamma}(t_k)$ ratios during extended periods of GPSD. If position errors grow too large and/or extreme $\bar{\Gamma}(t_k)$ ratios occur, the linearizations of the filter update are no valid, resulting in incorrect updates. This research will not consider trajectories that induce these erroneous camera updates. For future work, a UAV-landmark proximity will likely need to be considered—close enough to minimize navigation errors, but far enough to prevent the large $\bar{\Gamma}(t_k)$ ratios from creating large measurement errors (Note, the proper navigation state $\breve{\bar{\bar{H}}}_{\breve{\bar{x}}}\left(\hat{\bar{x}}^{-}(t_k)\right)$ matrix will need to be used when determining that proximity). Such implementation would also coincide with obstacle avoidance studies.

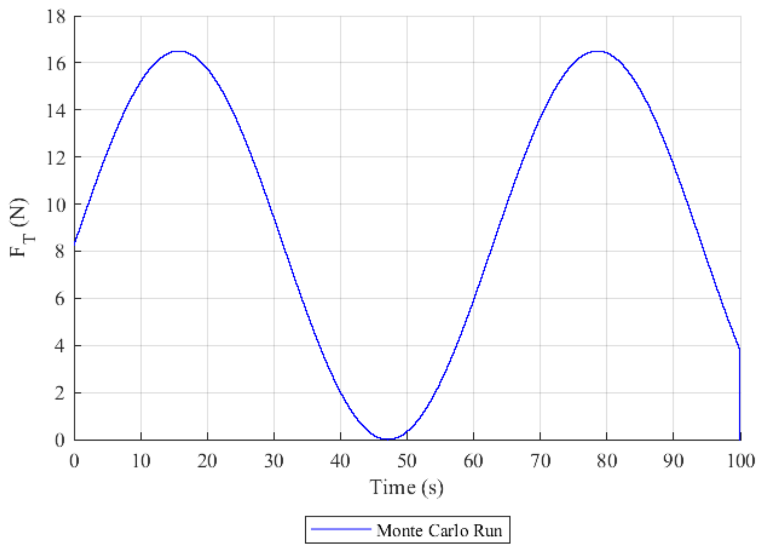


Figure 2.60. Estimation capability F_T.

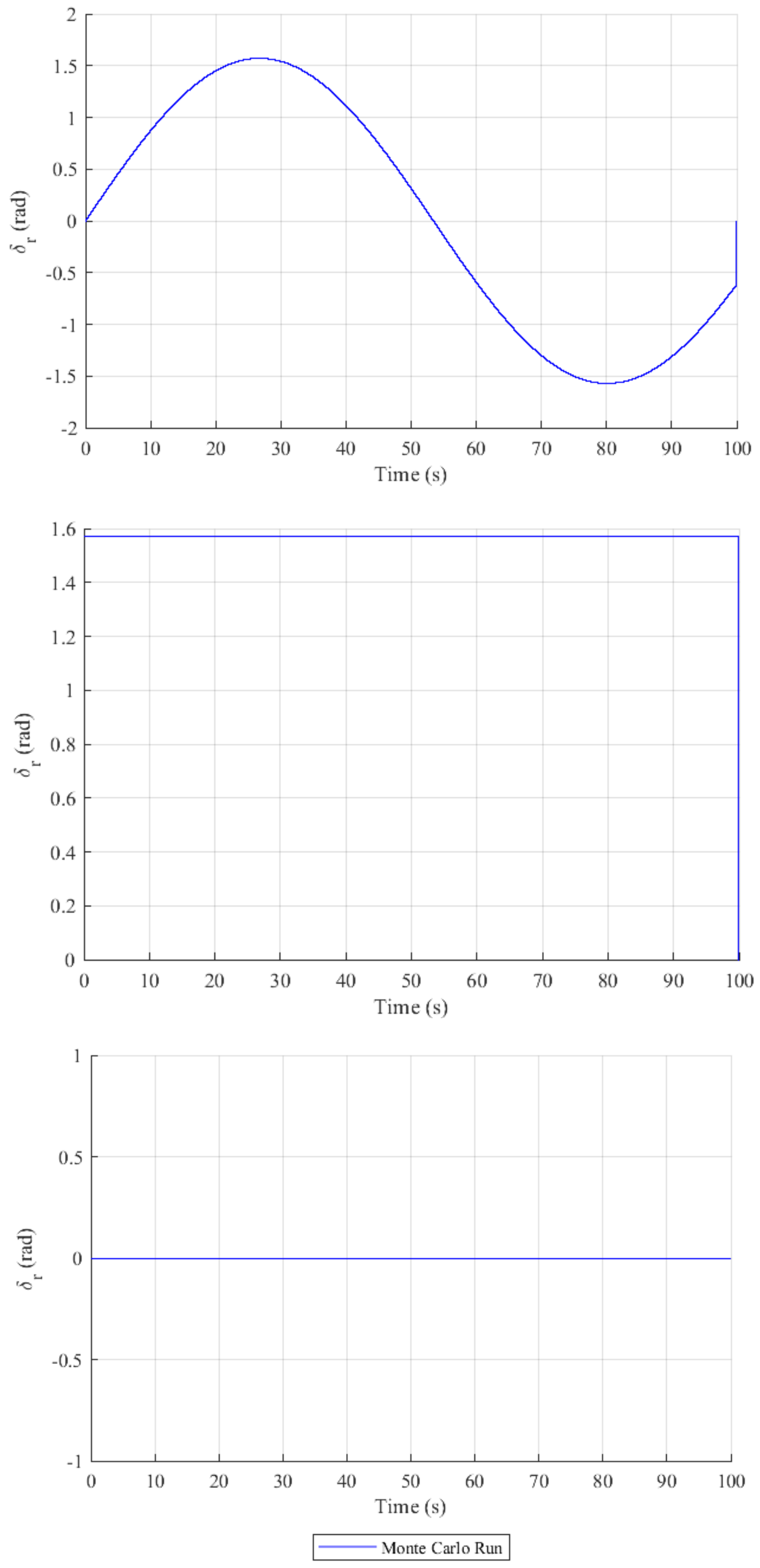


Figure 2.61. Estimation capability δ_r.

For all three trajectories, the exact same thrust was generated. The unique rudder inputs were the same ones used in the 2.1.6 Covariance Propagation subsection.

In this subsection, the discrete update portion of the Kalman filter was validated. Both the $\breve{\bar{\bar{J}}}(\hat{\bar{x}})$ and $\breve{\bar{\bar{R}}}(\hat{\bar{x}})$ matrices and other components of the update cycle generated reasonable results in the presence activated discrete noise, measurement denial, and measurement availability in terms of both residuals and ensemble statistics. To that end, implementation of the UAV's navigation system is assumed to be complete. This document now transitions to the implementation of the UAV guidance and control law.

2.2 Guidance and Control Law Development

Section 2.1 provided an extensive explanation and validation for the UAV's navigation system. The plots generated therein generally used sinusoidal rudder and thrust to exercise harder dynamics. For mission purposes, however, more sophisticated GC is needed. This section, 2.2, will provide an explanation of that GC—optimal control using the direct multiple shooting (simultaneous) method with explicit Euler integration. First, an explanation on the logic of optimal control and the multiple shooting method will be given (2.2.1). Second, an explanation of the selected model and toolbox/library packages used to implement GC will be given (2.2.2). Third, a base-case simulation will be executed to confirm correct implementation of the GC law (2.2.3).

2.2.1 Guidance and Control Logic

Before going into depth on the details of this simulation's specific GC package, a broad discussion and example on the logic of the selected GC law may be desired. The remainder of this paragraph will explain the general reasoning for using this GC law. The following paragraphs will provide a more rudimentary explanation and an example on the logic of how this GC law solves a problem. This section is based off the lectures of Dr. Greg Droge [32]. The main reasoning for using this particular control law is for simplicity and desirable features. Using quadratic-based costs is more straightforward, like LQR control. Using a direct method, where the equation format of the problem is discretized before determining first-order-necessary-conditions (FONC), was available in a convenient library for the author (CasADi). Furthermore, the direct method is typically easier to solve than indirect methods, where the FONC of the problem is found in continuous time using a two-point boundary value problem setup before numerically discretizing the problem. Dynamic programming, where the curse of dimensionality can make solutions intractable, and implementation of collocation methods was not readily known to the author. The type of direct method used for the simulation was the multiple shooting (simultaneous) method. Direct shooting methods are more popular methods; nevertheless, they require knowledge of future state values that are not readily available. In the direct multiple shooting method, future, guessed state (guidance) values are determined by simultaneously minimizing them with the control inputs. This is opposed to the direct single shooting (sequential) method, which uses current control inputs and initial conditions to guess the future state. The multiple shooting method was the preferred method, as it is more well-suited for nonlinear problems, problems with state constraints, as well as problems that are not necessarily stable. Explicit integration (propagation), rather than exact methods (preferences for linear systems) or implicit integration (preferences for higher numerical stability), was used. Euler’s method was used for consistency with the navigation portion of the simulation. Now, with the general reasoning for using this GC law, a more rudimentary explanation and example on how this GC law solves an optimization problem is now provided.

All optimal control problems begin with converting the given problem into an equation format:

$$\min_{F_T}\left(J(\bar{x},\bar{U}) = \left\{\int_{t_0}^{t_{sim}} L\big(\bar{x}(t),\bar{U}(t)\big)\right\} + \phi(\bar{x}(t_{sim}))\right)$$

$$s.t.\ \bar{x}(t) = \bar{f}\big(\bar{x}(t),\bar{U}(t)\big) \qquad (2.89)$$

$$\bar{x}(t_0) \text{ is known}$$

$$\left|\overset{\frown}{\delta}_r\,(t)\right| \le\ 90°$$

For numerical computation in the multiple shooting method, the problem is discretized:

$$\min_{F_T}\left(J(\bar{x},\bar{U}) = \left\{\sum_{t_0}^{t_{sim-1}} L\big(\bar{x}(t_\epsilon),\bar{U}(t_\epsilon)\big)\right\} + \phi(\bar{x}(t_{sim}))\right)$$

$$s.t.\ \bar{x}(t_\epsilon + 1) = \bar{f}\big(\bar{x}(t_\epsilon),\bar{U}(t_\epsilon)\big) \qquad (2.90)$$

$$\bar{x}(t_0) \text{ is known}$$

$$\left|\overset{\frown}{\delta}_r\,(t_\epsilon)\right| \le\ 90°$$

In this equation example, there are three main components. First is the minimization (or maximization) operator, $\min_{F_T}(*)$, where the variables underneath are the variables that the engineer wishes to find the value(s) of (i.e., what the engineer wants to minimize with respect to). In this example, the engineer's primary problem is to minimize fuel usage. A near-equivalent statement is that the engineer's problem is to minimize the control's thrust input. This near-equivalent statement can be expressed with mathematical terms found in the system dynamics; hence, the engineer wishes to find the minimum values of F_T.

Second is the cost function, $J(\hat{\bar{x}}, \bar{U})$. The cost function is composed of equation expressions that represent the engineer's desires both throughout the simulation, $\sum_{t_0}^{t_{sim}-1} L\big(\bar{x}(t_\epsilon), \bar{U}(t_\epsilon)\big)$, such as keeping UAV speed or accelerations low at all times, and expressions to be minimized at certain times or circumstances, $\phi(\bar{x}(t_{sim}))$, such as reducing discrepancy between the desired final destination point and the true final destination point. Note that though costs terms, like the distance discrepancy example, are not necessarily related to minimizing the primary problem's thrust, their presence in the cost function help the engineer include other secondary desires. There is not one standard way to create these cost functions, but for this document's control law, one common practice is used. This practice is a quadratic approach by simply squaring terms that are involved with the minimization. For example, if one thinks that keeping the vehicle in a straightened trajectory will help reduce fuel usage, a near-equivalent statement would be to minimize the rudder input. The near-equivalent cost term can be converted into equation form as

$$L\big(\bar{x}(t_\epsilon), \bar{U}(t_\epsilon)\big) = \overbrace{\delta}_r\,(t_\epsilon)^2 \tag{2.91}$$

or, if the error between a desired rudder input, $\overbrace{\delta}_{rd}$, and the current rudder input wish to be decreased,

$$L\big(\bar{x}(t_\epsilon), \bar{U}(t_\epsilon)\big) = \left\{\overbrace{\delta}_r\,(t_\epsilon) - \overbrace{\delta}_{rd}\,(t_\epsilon)\right\}^2 \tag{2.92}$$

Squaring the rudder terms not only prevents the solution from incorrectly favoring $-90°$ rudder inputs. Squaring is also a useful tool in proving that the system is stable via Lyapunov theory (makes the cost function convex).

Returning to the main components of the optimal control equation, Equation (2.90), the third component is the constraints—components of the problem that cannot or should not be altered, unlike the desirables in the cost function. These are first denoted with $s.t.$ for 'such that.' Typically, the equality portions of the constraints are the system dynamics. Inequality constraints can show up in the form of control constraints such as the range limitations of the rudder input. Equality constraints can solve for the desired variable via a format similar to solving systems of equations. For inequality constraints, acceptable values of the desired variable can be found using Karush-Kuhn-Tucker conditions and algorithms that include techniques such as penalty functions or gradients.

From converting the problem into an equation format, the next step in optimal control involves perturbation theory and FONC using partials of the Hamiltonian, an addendum to the cost function in the form of

$L\big(\bar{x}(t_\epsilon), \bar{U}(t_\epsilon)\big) + \left\{\overbrace{\bar{\lambda}}^{T}(t_\epsilon + 1)\right\} \bar{f}\big(\bar{x}(t_\epsilon), \bar{U}(t_\epsilon)\big)$, where $\overbrace{\bar{\lambda}}$ are costate (a.k.a., eigen or Lagrange multiplier) values. What this step does is find the solution gradient in multidimensional space—the 'direction' or environment of steepest decent/accent in which the UAV can currently move to locally minimize the cost function while still satisfying the constraints. Because such a procedure can only guarantee local minimization, future work will be needed to compare global minimums via dynamic programming.

Even after perturbation and FONC, finding the numerical values of the gradient is difficult. Recall that the Hamiltonian introduced costates at future time periods—$\overbrace{\bar{\lambda}}^{T}(t_\epsilon + 1)$. Consequently, future state values must be known as well. These components of the derivation are a common challenge in optimal control where solutions require unknown values. In terms of the multiple shooting method, the unknown values of the future state are guessed/solved for by including them as variables to find, $\min\limits_{F_T,\bar{X}}(*)$, and appending both desired variables to the constraints. For example, given a linear system with the equality constraints at a single step

$$\begin{gathered}\bar{X}(t_\epsilon + 1) = \bar{A}\bar{X}(t_\epsilon) + \bar{B}\bar{U}(t_\epsilon) \\ \bar{X}(t_0) = \bar{X}(\bar{0})\end{gathered} \tag{2.93}$$

rearranged as

$$\begin{gathered}\bar{A}\bar{X}(t_\epsilon) + \bar{B}\bar{U}(t_\epsilon) - \bar{X}(t_\epsilon + 1) = \bar{0} \\ -\bar{X}(t_0) = -\bar{X}(\bar{0})\end{gathered} \tag{2.94}$$

the appended system of equations, for every step in the simulation, would have the form

$$\begin{gathered}\begin{bmatrix} -\bar{I} & \bar{0} & \cdots & \cdots & \cdots & \bar{0} & \bar{0} & \cdots & \cdots & \cdots & \bar{0} \\ \bar{A} & -\bar{I} & \bar{0} & \cdots & \cdots & \vdots & \bar{B} & \bar{0} & \cdots & \cdots & \vdots \\ \bar{0} & \bar{A} & -\bar{I} & \bar{0} & \cdots & \vdots & \bar{0} & \bar{B} & \bar{0} & \cdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \bar{0} & \vdots & \ddots & \ddots & \ddots & \bar{0} \\ \bar{0} & \cdots & \cdots & \bar{0} & \bar{A} & -\bar{I} & \bar{0} & \cdots & \cdots & \bar{0} & \bar{B} \end{bmatrix} \begin{bmatrix} \bar{X}(t_0) \\ \vdots \\ \bar{X}(\mathrm{t_{sim}}) \\ \bar{U}(t_0) \\ \vdots \\ \bar{U}(\mathrm{t_{sim-1}}) \end{bmatrix} \\ = \begin{bmatrix} -\bar{X}(\bar{0}) \\ \bar{0} \\ \vdots \\ \vdots \\ \vdots \\ \bar{0} \end{bmatrix}\end{gathered} \tag{2.95}$$

where $\bar{I}$ is an identity matrix. From there, the system of equations is solved to find both future state and future control values. Note that these future state values are actually guidance values; i.e., a separate algorithm is not needed to determine guidance. After determining future state values, $\tilde{\bar{\lambda}}^T(t_\epsilon + 1)$ values can be determined and then the numerical value of the system gradient can be determined.

Much like the propagation method used in the navigation portion of the simulation, the appropriate amount of control inputs for the next time step are generally chosen by taking the current control and adding on the gradient times a step size. The cost function is then evaluated with the next step's inputs and states to see if the cost function has reached a local minimum. Note, that, unlike the navigation's time step, the step sized is not constant. Constant step sizes have tendencies to "step over" the minimum multi-dimensional cost solution. As such, approaches such as line search, diminishing step size, Armijo step size, and optimal step size are used to adjust the step size as the algorithm converges onto a local cost minimum. The entire process of (1) finding a gradient, (2) finding an appropriate step size, (3) propagating the control, and (4) determining if a local minimum cost has been reached (indicating no change in control input is needed) is iterated till the end of the simulation.

This subsection has just discussed the logic behind this document's optimal control law. In short, this control law has simple, yet desirable features. After converting a given problem into an equation format composed of the minimization operator, cost, and constraints, a gradient can be found. This gradient can then be used to propagate control inputs forward in time to minimize desired secondary system objectives. With this logic in mind, this document now progresses on to explaining the details of this specific simulation.

2.2.2 Guidance and Control Model and Code Package

This subsection will first detail the components of this documents GC law's equation format, then IPOPT and CasADi package implementation will be explained.

The overall problem for this document is to investigate if certain variables in the mission reduce navigation errors. For the GC law, the problem is more specifically investigating control inputs that might reduce navigation errors. Furthermore, as a multiple shooting method, finding state values is also needed. Hence, the minimization operator is set up as $\min_{\bar{x},\bar{U}}\left(J(\bar{x},\bar{U})\right)$. Skipping over the rest of the cost for now, the constraints consist of initial conditions, state dynamics, and limitations of the control inputs. Although the same initial conditions from the navigation law are used, the GC law is set up with a different assumption of states and dynamics. The set-up mimics the truth model but has a reduction in terms:

$$\breve{\bar{x}} = \begin{bmatrix} \breve{p}_n \\ \breve{p}_e \\ \breve{u} \\ \breve{v} \\ \breve{\psi} \\ \breve{r} \end{bmatrix} \tag{2.96}$$

$$\dot{\breve{\bar{x}}} = \breve{\bar{f}}\left(\breve{\bar{x}}, \hat{\bar{x}}, \breve{\bar{\bar{U}}}\right) = \begin{bmatrix} \dot{\breve{p}}_n \\ \dot{\breve{p}}_e \\ \dot{\breve{u}} \\ \dot{\breve{v}} \\ \dot{\breve{\psi}} \\ \dot{\breve{r}} \end{bmatrix} = \begin{bmatrix} \breve{u}\cos\left(\breve{\psi}\right) - \breve{v}\sin\left(\breve{\psi}\right) \\ \breve{u}\sin\left(\breve{\psi}\right) + \breve{v}\cos\left(\breve{\psi}\right) \\ \breve{r}\,\breve{v} - \dfrac{\hat{C}_{D_o}\,\hat{\rho}\,\breve{V}_a^2\,\hat{S}}{2\hat{m}} + \dfrac{\breve{F}_T}{\hat{m}} \\ -\breve{r}\,\breve{u} + \dfrac{\breve{\Xi}_1\,\hat{\rho}\,\breve{V}_a^2\,\hat{S}}{2\hat{m}} \\ \breve{r} \\ \dfrac{\breve{\Xi}_2\,\hat{\rho}\,\breve{V}_a^2\,\hat{S}\,\hat{b}}{2} \end{bmatrix} \tag{2.97}$$

where

$$\begin{bmatrix} \breve{\Xi}_1 \\ \breve{\Xi}_2 \\ \breve{\beta} \\ \breve{V}_a \\ \breve{u}_r \\ \breve{v}_r \end{bmatrix} = \begin{bmatrix} \hat{C}_{Y_0} + \hat{C}_{Y_\beta}\,\breve{\beta} + \dfrac{\hat{C}_{Y_r}\,\hat{b}\,\breve{r}}{2\,\breve{V}_a} + \hat{C}_{Y_{\tilde{\delta}_r}}\,\breve{\tilde{\delta}}_r \\ \hat{C}_{r_0} + \hat{C}_{r_\beta}\,\breve{\beta} + \dfrac{\hat{C}_{r_r}\,\hat{b}\,\breve{r}}{2\,\breve{V}_a} + \hat{C}_{r_{\tilde{\delta}_r}}\,\breve{\tilde{\delta}}_r \\ \sin^{-1}\left(\dfrac{\breve{v}_r}{\breve{V}_a}\right) \\ \sqrt{\breve{u}_r^2 + \breve{v}_r^2} \\ \breve{u} - \hat{u}_{wg} \\ \breve{v} - \hat{v}_{wg} \end{bmatrix} \tag{2.98}$$

$$\check{\bar{U}} = \begin{bmatrix} \check{F}_T \\ \check{\overbrace{\delta}}_r \end{bmatrix} \tag{2.99}$$

The same position, velocity, attitude, and attitude rate states from the truth model are used in Equation (2.96) and Equation (2.97). Biases are not used as their purpose was in accounting for continuous sensor errors—The control law relies on navigation to take out sensor errors. Landmarks are not used as the method for minimizing errors. In terms of control, error minimization is investigated through the cost function, not through a propagation or update equation. Unlike the navigation design model, Jacobians will be made using only this GC model as there is a need to obtain the Jacobian in terms of control inputs of Equation (2.99). Check overmarks are used to identify GC model variables. Hat overmarks remain on navigation values that the GC law uses consistently. Nevertheless, it is true that the GC law at given times will substitute in navigation equivalent values for defining the current state and many guessed solution values in its numerical initialization. Note again that these equations do not show the explicit dependencies on time in the right-hand-side of these equations for visual clarity purposes. Also, the above equations are continuous time definitions. Discretization is handled by the toolboxes and libraries.

$$\begin{gathered} \left|\check{\overbrace{\delta}}_r\,(t_\epsilon)\right| \leq\; 90° \\ 0\;N \leq \check{F}_T(t_\epsilon) \leq 13.5\;N \end{gathered} \tag{2.100}$$

$$\begin{bmatrix} -2500\,North \\ -2500\,East \\ 16\,\frac{m}{s} \\ -6\,\frac{m}{s} \\ -25\,rad \\ -0.5\,\frac{rad}{s} \end{bmatrix} \leq \begin{bmatrix} \check{p}_n(t_\epsilon) \\ \check{p}_e(t_\epsilon) \\ \check{u}(t_\epsilon) \\ \check{v}(t_\epsilon) \\ \check{\psi}(t_\epsilon) \\ \check{r}(t_\epsilon) \end{bmatrix} \leq \begin{bmatrix} 2500\,North \\ 2500\,East \\ 25\,\frac{m}{s} \\ 6\,\frac{m}{s} \\ 25\,rad \\ 0.5\,\frac{rad}{s} \end{bmatrix}$$

$$\left|\breve{\bar{U}}(t_\epsilon) - \breve{\bar{U}}(t_{\epsilon-1})\right| \leq \begin{bmatrix} 0.8\,N \\ 0.05\,rad \end{bmatrix}$$

In terms of discrete control and state limitations, rudder has the same limitations as the prior example. Thrust is restrained from null to 13.5 N (Assuming the aircraft's cruise speed is approximately 1/10th the aircraft's weight [33]. Future work can examine simulations having maximum thrust conditions, which range at approximately 1/2 to 1/4th the aircraft's weight [34]). Vehicle forward velocity is assumed to remain between 16 m/s and 25 m/s (35 mph and 56 mph) [31]. Although not necessary, limits (based on the results from the 2.1 Kalman Filter Development section) were placed on position, lateral velocity, heading and heading rate to assist the IPOPT/CasADi solver in yielding quicker solutions. Furthermore, restrictions were placed on the control rate of change based on approximations from [35] and [36]. Using rate-of-change constraints (rate limit profiler) was used rather than a PID control approach, where control dynamics are implemented in the UAV's GC dynamics, because the PID approach requires guidance values to solve the generated Laplace transform—guidance values are only available after the appended system of equations are solved; i.e., the PID approach requires values that have yet to be solved for. There is a last potential constraint for future work. This constraint is maintaining UAV-landmark proximities. Implementation might use [37] as a reference, which approaches the problem by keeping the probability of trespassing over a UAV-landmark proximity below a user-defined percent. Preliminary results suggested that alternatively adding this proximity constraint in the cost function with a high gain instead may be more computationally feasible.

Returning to the cost, various factors must be considered. As a start, waypoint following will be used to direct the UAV in and out of a GPSD region. Next, a cost to promote circular trajectories that appeared to reduce error in the 2.1.6 Covariance Propagation subsection is used. A cost to favor trajectories where the landmark is centered inside the camera FOV is used. This helps reduce image distortion and to pursue close UAV-landmark proximities. A cost to race the sensor's quadratic error growth out of GPSD is also used. The total cost function is displayed with Equation (2.101).

$$\min_{\check{\check{x}},\check{\check{\bar{U}}}}\left(\begin{array}{c} J\left(\check{\check{x}},\check{\check{\bar{U}}}\right) = \\ \sum_{t_0}^{t_0+2\Delta t_k}\left\{\begin{array}{c} G_1\sqrt{\{\check{p}_n(t_\epsilon)-\check{p}_{nd}(t_\epsilon)\}^2+\{\check{p}_e(t_\epsilon)-\check{p}_{ed}(t_\epsilon)\}^2} \\ +G_2\sqrt{\{\check{v}_d(t_\epsilon)^2-\check{v}(t_\epsilon)^2\}^2} \\ +G_3\sqrt{\check{\gamma}(t_\epsilon)^2} \\ +G_4\sqrt{\{\check{u}(t_\epsilon)-\check{u}_d(t_\epsilon)\}^2} \end{array}\right\}\end{array}\right) \quad (2.101)$$

In this equation, the minimization operator, $\min\limits_{\check{\check{x}},\check{\check{\bar{U}}}}()$, is tailored to the multiple-shooting method's minimization of both state, $\check{\check{x}}$, and control, $\check{\check{\bar{U}}}$. The cost function, J, is expanded as a summation of the cost terms from the time the control law starts, t_0, to a desired future time, in this case two discrete time periods, $t_0 + 2\Delta t_k$ (incrementally solving the GC law over small time horizons is similar to approaches used in Model Predictive Control (MPC)). Each cost term is associated with a G_* gain/weight multiplier. G_1 is associated with waypoint following, which aims to minimize the distance between the current north and east desired waypoint location, $[\check{p}_{nd}, \check{p}_{ed}]$, and the UAV's current location, $[\check{p}_n, \check{p}_e]$. G_2 uses a near-equivalent statement of minimizing the difference between the desired lateral velocity input, $\breve{v}_d$, and current lateral velocity inputs, $\breve{v}$. G_3 uses a near-equivalent statement of minimizing, $\breve{\gamma}$, which is the angle between the camera's bore axis, c_i, and a vector, $[\Lambda_i, \Lambda_j]$, that points from the UAV to a landmark, Λ, as was shown in Figure 2.2. Finally, G_4 is adjoined with the difference between the UAV's forward speed, $\breve{u}$, and a desired forward speed, $\breve{u}_d$. During the gain tuning process, Equation 3.1 will be adjusted for cost gain tuning.

In more detail, Equation (2.101) did not require a $\phi(\bar{x}(t_{sim}))$ term. The costs include a square root with quadratic squaring. Using a magnitude format provided better result compared with quadratic squaring only. $\breve{v}_d(t_\epsilon)$ is set to a constant max velocity. No $\breve{\gamma}_d$ was required since $\breve{\gamma}_d = 0$. $\breve{\gamma}$ is calculated as

$$\breve{\gamma}(\epsilon) = atan2\left(\frac{\breve{\Lambda}_j(\epsilon)}{\breve{\Lambda}_i(\epsilon)}\right) \tag{2.102}$$

for the currently desired landmark. Note that for the IPOPT/CasADi solver to properly minimize the cost via partial derivatives, variables must be expressed int terms of the states—meaning that the i and j landmark position values must be expanded as they were in Equation (2.11) through Equation (2.13). Forward speed ranged its desired speed from stall speed to max speed. Though waypoint following naturally encourages higher speeds, lower speeds also need to be induced for comparison purposes in this study. Note, lateral velocity gain is used for the circular trajectory cost due to high lateral velocity occurring at high curvature. Furthermore, high heading rate, high rudder input, and curvature are not used instead of high lateral velocity because one can achieve high heading rates and high rudder inputs on a straightened trajectory when the rudder is constantly switching from -90° to 90°; and curvature, denoted as

$$\check{\kappa} = \frac{\check{r}}{\sqrt{\check{u}^2 + \check{v}^2}} \tag{2.103}$$

has conflicting terms when maximizing. Again, in future work, a gain G_5 could be placed with a UAV-landmark proximity cost.

Up to this point, subsection 2.2.2 has described the GC model of state, dynamics, constraints, and cost function. Defining a separate "truth" GC model and "navigation" GC model was deemed unnecessary, especially since control noise was not added to the system and is left for future work. From here, the GC model is implemented into the GC law toolboxes and libraries: IPOPT and CasADi.

IPOPT is a popular, readily available toolbox for solving large optimization problems [38]. The solver uses the interior point line search to find the local minima of the problem. For user-defined properties, the author kept all defaults, including the free MUMPS sparce symmetric indefinite linear solver, except for the number of max iterations, which was set to 40 to reduce computation time. The CasADi library [39] is an interface to IPOPT. CasADi uses a symbolic approach to provide a more user-friendly environment in defining the problem and its Jacobians before sending the problem to IPOPT for solving. The author mimicked the CasADi example pack direct_multiple_shooting.m to implement the direct multiple shooting GC law, including line 117-120's state continuity equality constraint. "relax_bounds" was selected as the "fixed_variable_treatment." With the max iteration and "relax_bounds" settings, GC solutions may not be completely minimized, and constraint limits may not be completely obeyed. Nevertheless, the convergence to minimization and exceeding dynamics by small margins appeared close enough for all intents and purposes.

After implementing the model into the toolbox and library, the toolbox and library are implemented into the simulation. To do this, the author first collects the conditions and parameters for the GC law, calculates two simulation seconds of GC, and then repeats the simulation iterations of state propagation, Kalman update, and GC calculations.

In more detail, the author first collects info on whether the waypoint objective has changed or if a Kalman update has occurred. Both conditions signal that the GC law needs to be reevaluated. When reevaluated, the GC law uses the current navigation values to formulate a GC solution for the next 2 s of the simulation. 2 s was chosen primarily for reduced computation time; however, 2 s also proves advantageous in balancing both local and global path planning. 2 s is local enough to accommodate mission changes, e.g., waypoints, yet global enough to provide more refined control law solutions. To explain "refined," recall that optimal control problems yield "better" solutions when the problem is reevaluated using the first solution as numerical guess inputs for the second evaluation [32]. With that knowledge, given that the control law will update at least once per second in the simulation, the author used the remaining portion of each prior 2 s solution as numerical guess input for the first portion of the next 2 s solution. Furthermore, all numerical guess inputs for the second portion of the next 2 s solution were defaulted to equal the last values of the first 2 s solution. Figure 2.62 provides a visual of this description.

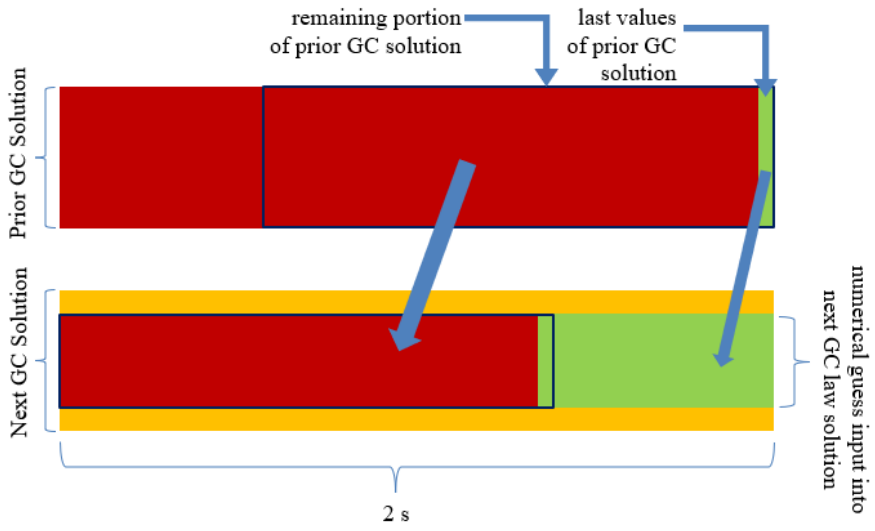

remaining portion of prior GC solution
last values of prior GC solution
Prior GC Solution
Next GC Solution
numerical guess input into next GC law solution
2 s

Figure 2.62. Visual of GC law reevaluation.

Again, having a 2 s GC solution provided an advantage where a global-like solution could be used for refined results in future 2 s solutions.

2.2.3 Guidance and Control Validations

With the minimization function and toolboxes/libraries for the GC law described, this document now moves forward in producing a validation simulation to confirm that portions of the GC law have been implemented correctly. Additionally, this simulation will provide an outline for how the future sensitivity studies will be set up.

For this validation and future sensitivity studies, the simulation is run with the same initial conditions, noise, and commercial sensor package from the 2.1.7 Estimation Capability subsection (though using the same truth-state and using $\breve{\bar{H}}_{\breve{\bar{x}}}\left(\bar{x}^{-}(t_k)\right)$, not $\breve{\bar{H}}_{\breve{\bar{x}}}\left(\hat{\bar{x}}^{-}(t_k)\right)$, for each Monte Carlo is no longer applied). A 200 Monte Carlo simulation is generated for the validation to ensure that low-error dispersions do not break-down the GC law. Using pre-planned waypoints (implementation for real-time waypoint planning is left to future work), the UAV is to first perform two hard rudder maneuvers, to exercise the dynamics, which encourages observability, before entering a GPSD zone. For a given study, landmarks may be scattered on the flight plane to provide navigation aid to the UAV after entering GPSD. Upon reaching the other side of the GPSD zone, the UAV is then instructed to return back through GPS denial to a final destination point. The simulation ends at 100 s or when the waypoint algorithm declares that the UAV sufficiently reached the final destination. Table 2.8 provides the specific GPSD region and waypoint locations.

Table 2.8. Waypoint and GPSD locations.

	North Min Edge	North Max Edge	East Min Edge	East Max Edge	Units
GPSD Region	-400	400	835	235	m
	North	East			Units
Waypoint 1 (Start)	400	0			m
Waypoint 2	239	123			m
Waypoint 3	131	230			m
Waypoint 4	131	918			m
Waypoint 5	-83	918			m
Waypoint 6	-83	230			m
Waypoint 7	-83	-3000			m

With these mission specifications, GPSD and added landmarks will not be used for this subsection’s validation (nevertheless, preliminary results indicated that, given waypoint following in a landmark-void, GPSD field, most commercial-sensor Monte Carlos are able to complete the waypoint mission. Mission failure is due navigation errors growing too large for the control law to accommodate—an issue that is resolved with camera updates). A seventh waypoint is included to aid the selected waypoint algorithm, which is similar to the filleted waypoint approach described in [13] with the added stipulation that the UAV must be withing 100 m of the current waypoint before pursuing the next waypoint. Only the waypoint following cost ($G_1 = 1.7\text{e-}2$, $G_2 = G_3 = G_4 = 0$) will be activated in this validation and the navigation study. Cost evaluation is left to the GC study.

The selected plots below are the GC validation results for a single, commercial sensor Monte Carlo and the trajectory for 200 Monte Carlos:

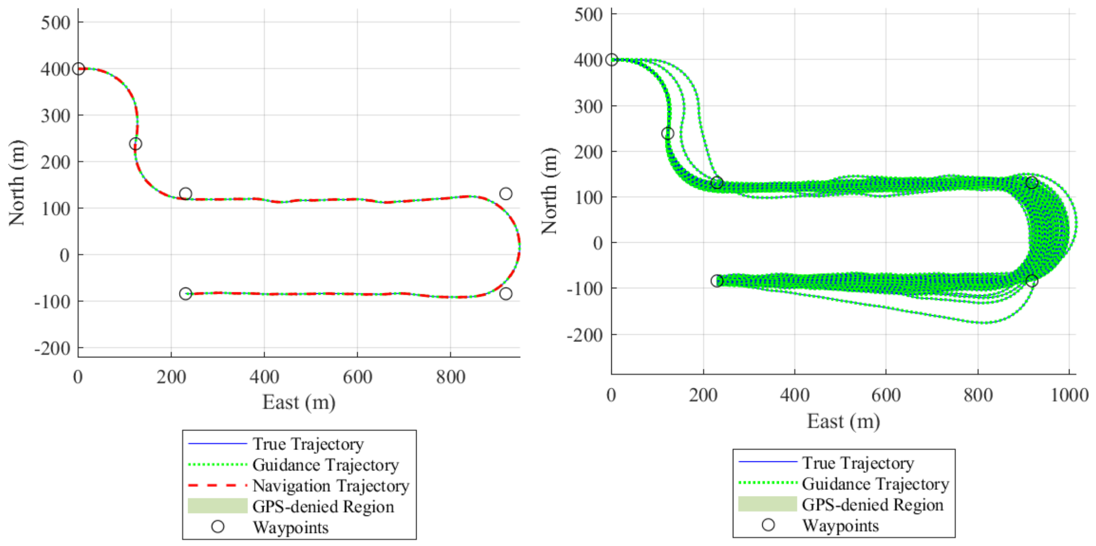


Figure 2.63. GC validation UAV north and east coordinates.

Figure 2.63 Presents the waypoint following trajectory. For the single Monte Carlo, a truth, guidance, and navigation trajectory are plotted. All trajectory types essentially have insignificant divergence from each other. When in the presence of GPSD, truth will significantly diverge from the guidance and navigation trajectory. Guidance does not significantly diverge from navigation since this GC law does not incorporate noise into the control output. For the 200 Monte Carlo plot, navigation plotting is removed for clarity.

Given full access to GPS, waypoint following for this GC law can overshoot and ‘cut corners’ as seen in the 200 Monte Carlo. The IPOPT settings, 2 s GC time horizon window, dispersions, and the filleted waypoint following all contribute to overshooting and cutting corners. Given the few cases of extreme overshooting and corner cutting, trajectory performance of the GC law is deemed acceptable.

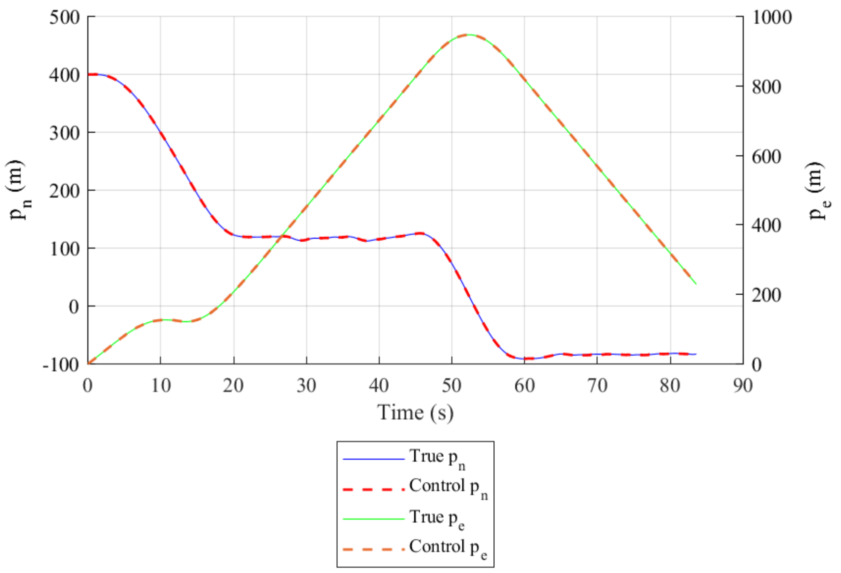
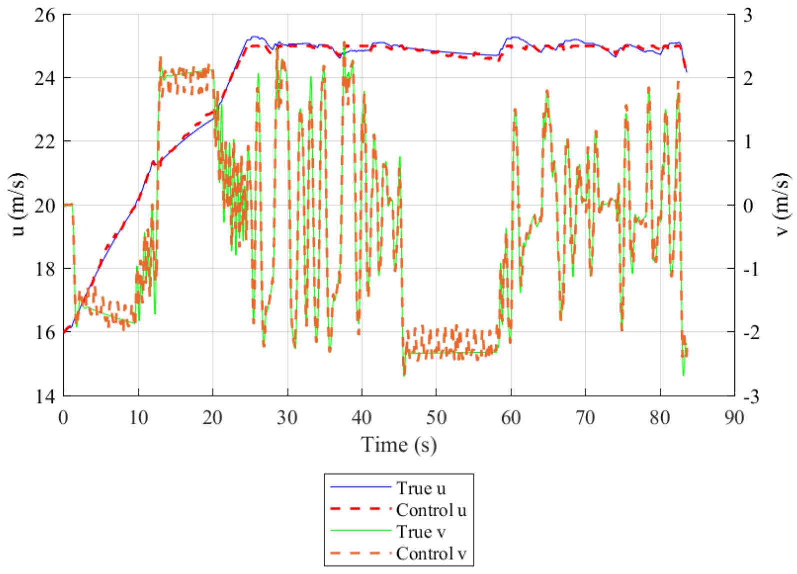


Figure 2.64. GC validation position over time.



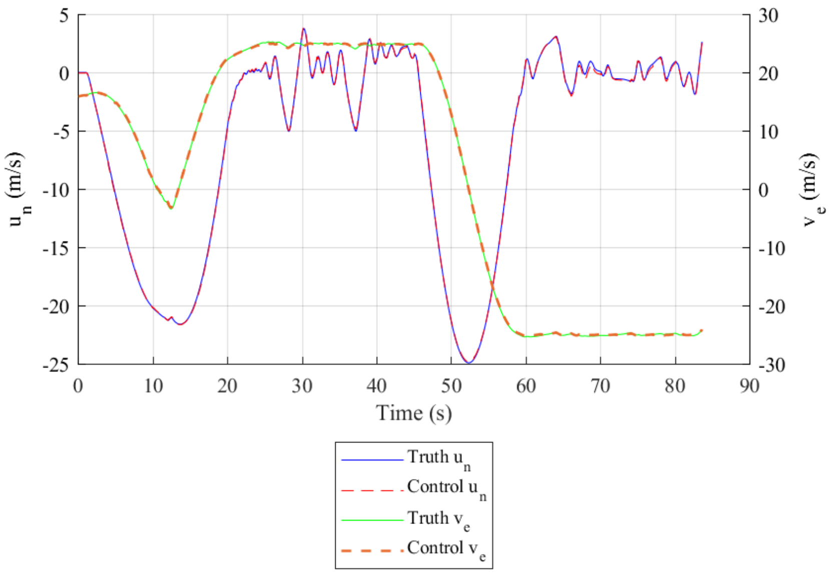


Figure 2.65. GC validation Body and N-E frame velocity over time.

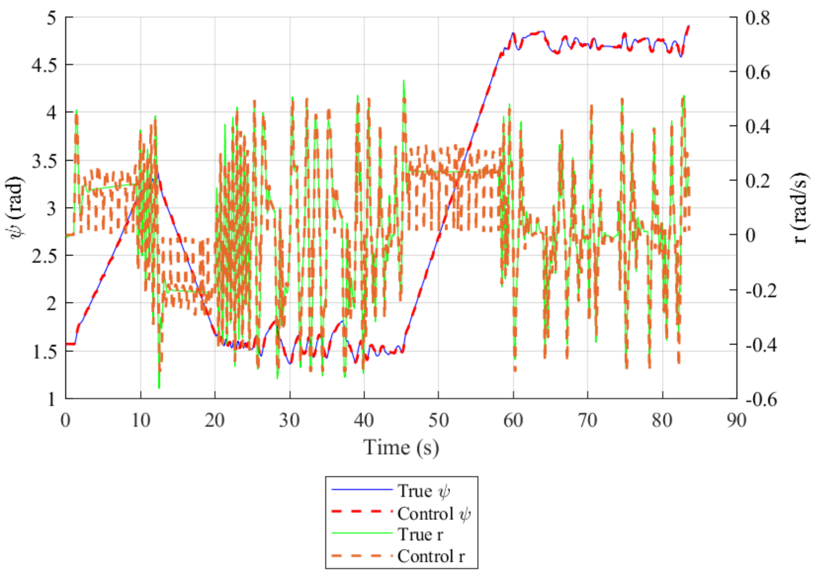


Figure 2.66. GC validation attitude and attitude rate over time.

Figure 2.64 through Figure 2.66 presents the UAV's states as a function of time. Given the scaling, there is a little more visibility in the differences between the control state and the truth state. In particular, control is more erratic in ψ and v. Nevertheless, the differences are not erratic enough to suggest incorrect dynamic implementation. Position, velocity, heading, and heading rate control constraints are obeyed. This does not, however, guarantee that the truth state or control state will always obey these constraints. At constraint boundaries, u for example, the truth states can be found to exceed the boundaries. With relaxed boundary conditions, the control state can also exceed in future simulations; but again, neither exceed significantly enough to be of concern.

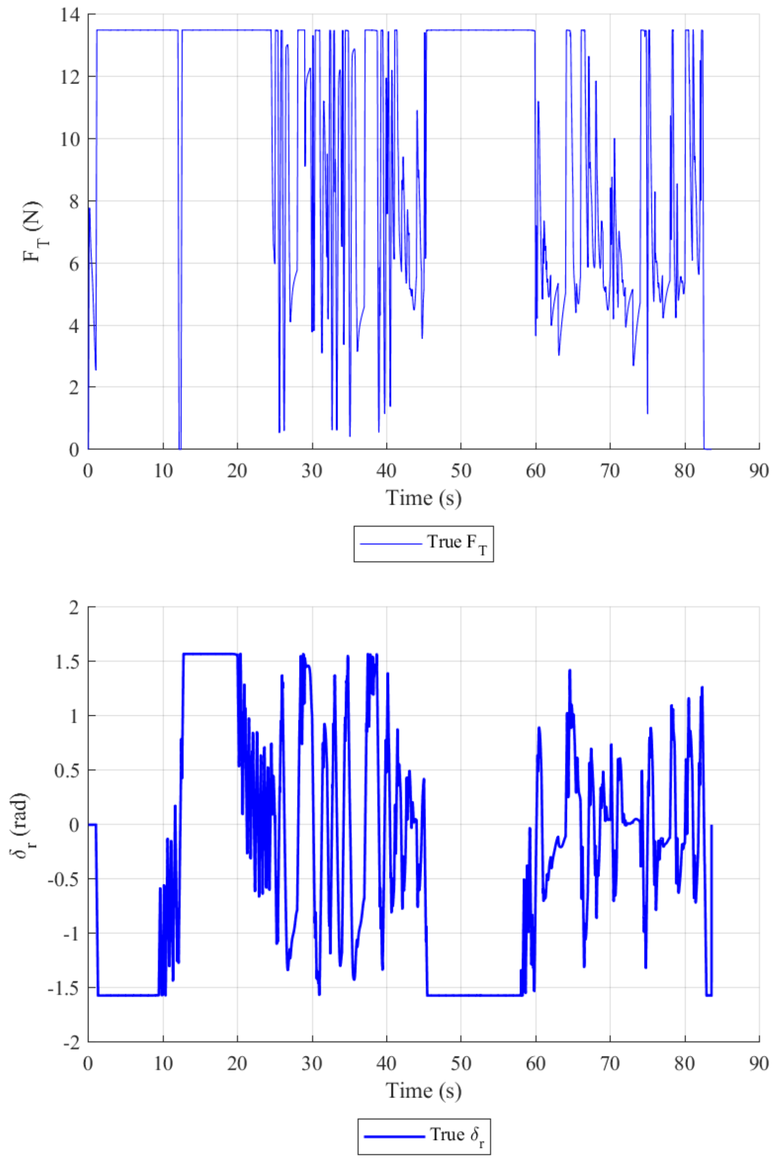


Figure 2.67. GC validation thrust and rudder input.

Figure 2.67 shows the control inputs for a single trajectory. Control bounds were upheld, and the author found that control rates of change were also upheld to the 8th decimal place, which are negligible discrepancies.

The results presented in this subsection helped to validate the implemented GC law. The GC waypoint costs was recognized, and GC model dynamics and constraints were adhered to. Recognition of potential error-reducing costs is left for the second simulation study. From here, this document now transitions to describe and execute the first and second simulation study.

CHAPTER 3

SIMULATION RESULTS

Up to this point, GNC background and development have been given. The purpose of this chapter is to list and describe the types of simulations that will be conducted for the navigation sensitivity analysis and the GC sensitivity analysis. First, specifications specifically used for the navigation sensitivity study and GC study are given (3.1). Following are sections that give specifications unique to the Navigation (3.2) and GC (3.3) study as well as their results and author interpretation.

3.1 Sensitivity Analyses Specifications

This section discusses the studies' specifications. Both studies use the same setup as the control law section with the exception that GPSD, landmarks, and other control gains can be activated. One Monte Carlo is generated for each study test case, instead of 200, with the assumption that the one Monte Carlo's 3σ statistics is sufficient to represent the dispersion outcomes of the other 200 Monte Carlos. To compare test cases with each other, all test cases will retain their root-sum-squared 3σ position statistics (derived from the IEKF's covariance) at each GPSD t_ϵ time step, $\left\|3\sigma\left(p(t_{GPSD})\right)\right\|$, i.e. $3\sqrt{tr\left(P_p(t_{GPSD})\right)}$ using the trace $tr()$ operator. To gain a holistic visual interpretation of all test cases, test cases are grouped into categories of sensor grade, landmark uncertainty, gain values, etc. Using the mean, $\mu\left(3\sqrt{tr\left(P_p(t_{GPSD})\right)}\right)$, for each category, line plots are generated for comparison. For clarity, example trajectories associated with categories are displayed after the applicable plots (Moving left to right on the plot is respectively associated with trajectories top left, top right, and bottom).

Continuing with listing study specifications, position root-sum-squared 3σ vs time plots, are shown for each category's low, medium, and high test cases. Shaded regions within the 3σ vs time plots to represent times during GPSD are not provided to reduce the image clutter of each trajectory's GPSD regions. To limit the study to landmark observations only in GPSD cases, camera observations are not allowed till after the UAV enters GPS denial. To help prevent scenarios where certain landmarks receive "unfair" amounts of camera observations, the author opened up the FOV to 160° to allow camera update availability throughout GPSD (consequently, Equation (2.82) updates to $\hat{b}_c = 2.4$ mrad 1σ). Future work can analyze similar statistics for velocity and attitude states.

3.2 Navigation Sensitivity Study

In this section, the study aims to compare how $\mu_{3\sigma_p}$ is affected with respect to variations in IMU sensor grade, landmark uncertainty, number of landmark observations, measurement geometry, and number of landmarks. For clarity, measurement geometry considers variations in γ, which influences the H matrix's direction of information. Each of the upcoming tables describe the test cases used for each variation. The tables provide columns indicating sensor quality, true landmark position, 3σ landmark uncertainty prior to the mission's start, and the number of seconds allowed before a camera Kalman update. In all these tests, only the waypoint gain is activated ($G_1 = 1.7$e-2, $G_2 = G_3 = G_4 = 0$) to limit the effect GC has on error outcomes. 70 total test cases are conducted as outlined in Table 3.1 through Table 3.5:

Table 3.1. Navigation study IMU sensor grade.

Simulation Number	IMU Sensor Grade	True Λ_{1n} [m]	True Λ_{1e} [m]	Λ_{1*} Initial Uncertainty [m]	True Λ_{2n} [m]	True Λ_{2e} [m]	Λ_{2*} Initial Uncertainty [m]	Camera Δt_k [s]
1	Commercial	331	835	0.5	-284	235	0.5	1
2	Tactical	331	835	0.5	-284	235	0.5	1
3	Navigation	331	835	0.5	-284	235	0.5	1

Table 3.2. Navigation study landmark uncertainty.

Simulation Number	IMU Sensor Grade	True Λ_{1n} [m]	True Λ_{1e} [m]	Λ_{1*} Initial Uncertainty [m]	True Λ_{2n} [m]	True Λ_{2e} [m]	Λ_{2*} Initial Uncertainty [m]	Camera Δt_k [s]
4	Tactical	331	835	0.1	-284	235	0.1	1
5	Tactical	331	835	0.9	-284	235	0.9	1
6	Tactical	331	835	1.7	-284	235	1.7	1
7	Tactical	331	835	2.5	-284	235	2.5	1
8	Tactical	331	835	3.2	-284	235	3.2	1
9	Tactical	331	835	4.0	-284	235	4.0	1
10	Tactical	331	835	4.8	-284	235	4.8	1
11	Tactical	331	835	5.6	-284	235	5.6	1
12	Tactical	331	835	6.4	-284	235	6.4	1
13	Tactical	331	835	7.2	-284	235	7.2	1
14	Tactical	331	835	7.9	-284	235	7.9	1
15	Tactical	331	835	8.7	-284	235	8.7	1
16	Tactical	331	835	9.5	-284	235	9.5	1
17	Tactical	331	835	10.3	-284	235	10.3	1
18	Tactical	331	835	11.1	-284	235	11.1	1
19	Tactical	331	835	11.9	-284	235	11.9	1
20	Tactical	331	835	12.6	-284	235	12.6	1
21	Tactical	331	835	13.4	-284	235	13.4	1
22	Tactical	331	835	14.2	-284	235	14.2	1
23	Tactical	331	835	15.0	-284	235	15.0	1

Table 3.3. Navigation study number of landmark observations.

Simulation Number	IMU Sensor Grade	True Λ_{1n} [m]	True Λ_{1e} [m]	Λ_{1*} Initial Uncertainty [m]	True Λ_{2n} [m]	True Λ_{2e} [m]	Λ_{2*} Initial Uncertainty [m]	Camera Δt_k [s]
24	Tactical	331	835	0.5	-284	235	0.5	0.1
25	Tactical	331	835	0.5	-284	235	0.5	0.25
26	Tactical	331	835	0.5	-284	235	0.5	0.5
27	Tactical	331	835	0.5	-284	235	0.5	1
28	Tactical	331	835	0.5	-284	235	0.5	2
29	Tactical	331	835	0.5	-284	235	0.5	3
30	Tactical	331	835	0.5	-284	235	0.5	4
31	Tactical	331	835	0.5	-284	235	0.5	5
32	Tactical	331	835	0.5	-284	235	0.5	6
33	Tactical	331	835	0.5	-284	235	0.5	7
34	Tactical	331	835	0.5	-284	235	0.5	8
35	Tactical	331	835	0.5	-284	235	0.5	9
36	Tactical	331	835	0.5	-284	235	0.5	10
37	Tactical	331	835	0.5	-284	235	0.5	11
38	Tactical	331	835	0.5	-284	235	0.5	12
39	Tactical	331	835	0.5	-284	235	0.5	13
40	Tactical	331	835	0.5	-284	235	0.5	14
41	Tactical	331	835	0.5	-284	235	0.5	15
42	Tactical	331	835	0.5	-284	235	0.5	16
43	Tactical	331	835	0.5	-284	235	0.5	17
44	Tactical	331	835	0.5	-284	235	0.5	18
45	Tactical	331	835	0.5	-284	235	0.5	19
46	Tactical	331	835	0.5	-284	235	0.5	20

Table 3.4. Navigation study measurement geometry.

Simulation Number	IMU Sensor Grade	True Λ_{1n} [m]	True Λ_{1e} [m]	Λ_{1*} Initial Uncertainty [m]	True Λ_{2n} [m]	True Λ_{2e} [m]	Λ_{2*} Initial Uncertainty [m]	Camera Δt_k [s]
47	Tactical	131	835	0.5	-84	235	0.5	1
48	Tactical	158	835	0.5	-110	235	0.5	1
49	Tactical	184	835	0.5	-136	235	0.5	1
50	Tactical	210	835	0.5	-163	235	0.5	1
51	Tactical	237	835	0.5	-189	235	0.5	1
52	Tactical	263	835	0.5	-215	235	0.5	1
53	Tactical	289	835	0.5	-242	235	0.5	1
54	Tactical	315	835	0.5	-268	235	0.5	1
55	Tactical	342	835	0.5	-294	235	0.5	1
56	Tactical	368	835	0.5	-321	235	0.5	1
57	Tactical	394	835	0.5	-347	235	0.5	1
58	Tactical	421	835	0.5	-373	235	0.5	1
59	Tactical	447	835	0.5	-400	235	0.5	1
60	Tactical	473	835	0.5	-426	235	0.5	1
61	Tactical	500	835	0.5	-452	235	0.5	1
62	Tactical	526	835	0.5	-478	235	0.5	1
63	Tactical	552	835	0.5	-505	235	0.5	1
64	Tactical	579	835	0.5	-531	235	0.5	1
65	Tactical	605	835	0.5	-557	235	0.5	1
66	Tactical	631	835	0.5	-584	235	0.5	1

Table 3.5. Navigation study number of landmarks.

Simulation Number	IMU Sensor Grade	True Λ_{1n} [m]	True Λ_{1e} [m]	Λ_{1*} Initial Uncertainty [m]	Camera Δt_k [s]
67	Tactical	331	835	0.5	1
		-284	235	0.5	
68	Tactical	331	702	0.5	1
		331	835	0.5	
		-284	368	0.5	
		-284	235	0.5	
69	Tactical	331	568	0.5	1
		331	702	0.5	
		331	835	0.5	
		-284	502	0.5	
		-284	368	0.5	
		-284	235	0.5	
70	Tactical	331	435	0.5	1
		331	568	0.5	
		331	702	0.5	
		331	835	0.5	
		-284	635	0.5	
		-284	502	0.5	
		-284	368	0.5	
		-284	235	0.5	
70	Tactical	331	302	0.5	1
		331	435	0.5	
		331	568	0.5	
		331	702	0.5	
		331	835	0.5	
		-284	768	0.5	
		-284	635	0.5	
		-284	502	0.5	
		-284	368	0.5	
		-284	235	0.5	

In more detail, three grades of simulations are used to understand sensor quality. Simulation 2 is used as the nominal case for all other test cases. To understand initial landmark uncertainty, the nominal was altered by varying both landmarks' uncertainties from 0.1 m to 15 m. The smaller 0.5 m landmark uncertainty is used as the nominal to ensure that the camera measurements always retain an active role in GPSD (i.e., the filter will not need to wait for UAV’s errors to grow, for example, past 10 m before a 10 m initial uncertainty landmark’s measurement can be significantly useful). For number of landmark observations, the Δt_k column was varied. What this did was alter the number of seconds before a camera Kalman update. GPS and magnetometer measurement were still allowed to update every 1 s (note the GC law did consequently update after each camera observation, so more computation was needed for the 0.1 to 0.5 tests). To study measurement geometry, landmark north position was altered. In future plots, the position will be described in terms of absolute north distance D from the nearest waypoint (waypoint 4 for Λ_1 and waypoint 6 for Λ_2). To study how the number of landmarks affects position errors there was difficulty since deriving navigation matrices to hold 10 landmarks is cumbersome. To compensate, the author generated a one-landmark simulation. This one landmark would be initialized as the closest landmark to the UAV (with respect to east position). After passing the camera’s FOV, this landmark’s position and covariance statistics would be stored then replaced by the next closest landmark’s statistics. Upon revisiting a landmark, the stored position and variance elements would be placed back into the UAV state and covariance (off diagonals set to zero).

3.2.1 Navigation Sensitivity Results

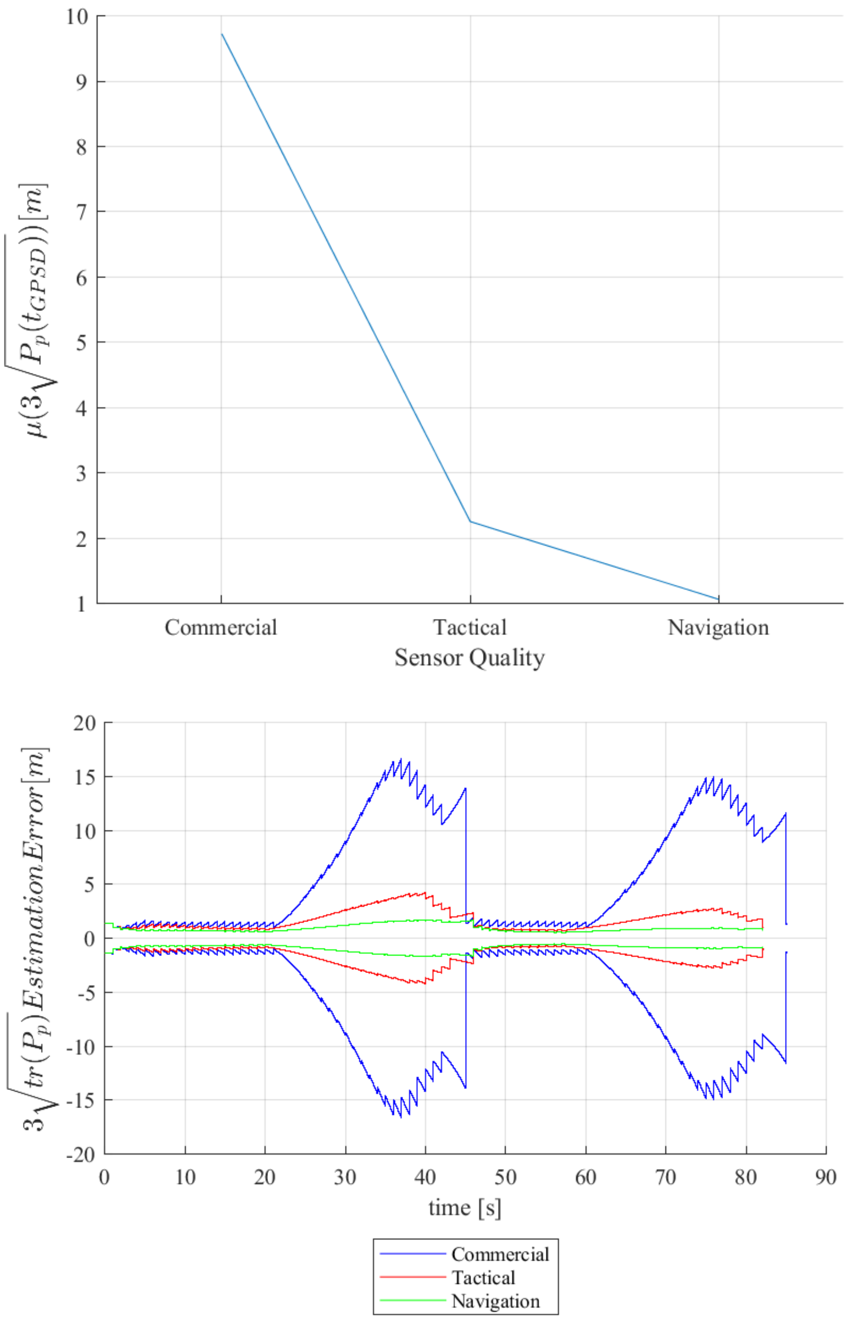


Figure 3.1. Navigation study IMU sensor grade.

Figure 3.1 displays the effects that IMU sensor grade has on the mission. The trend is that mean error decreases with improved sensor grade. With camera observations every second, commercial sensors achieve errors below 10 m. Navigation sensors can achieve errors just above GPS-measurement errors of 1 m. In the estimation error plot, extreme Kalman updates are reduced with improved sensor grade. Furthermore, the commercial grade sensor appears to exit GPS denial slightly faster than the other sensor grades. For future reference, the tactical grade simulation, the nominal simulation, achieved a mean error of 2.25 m.

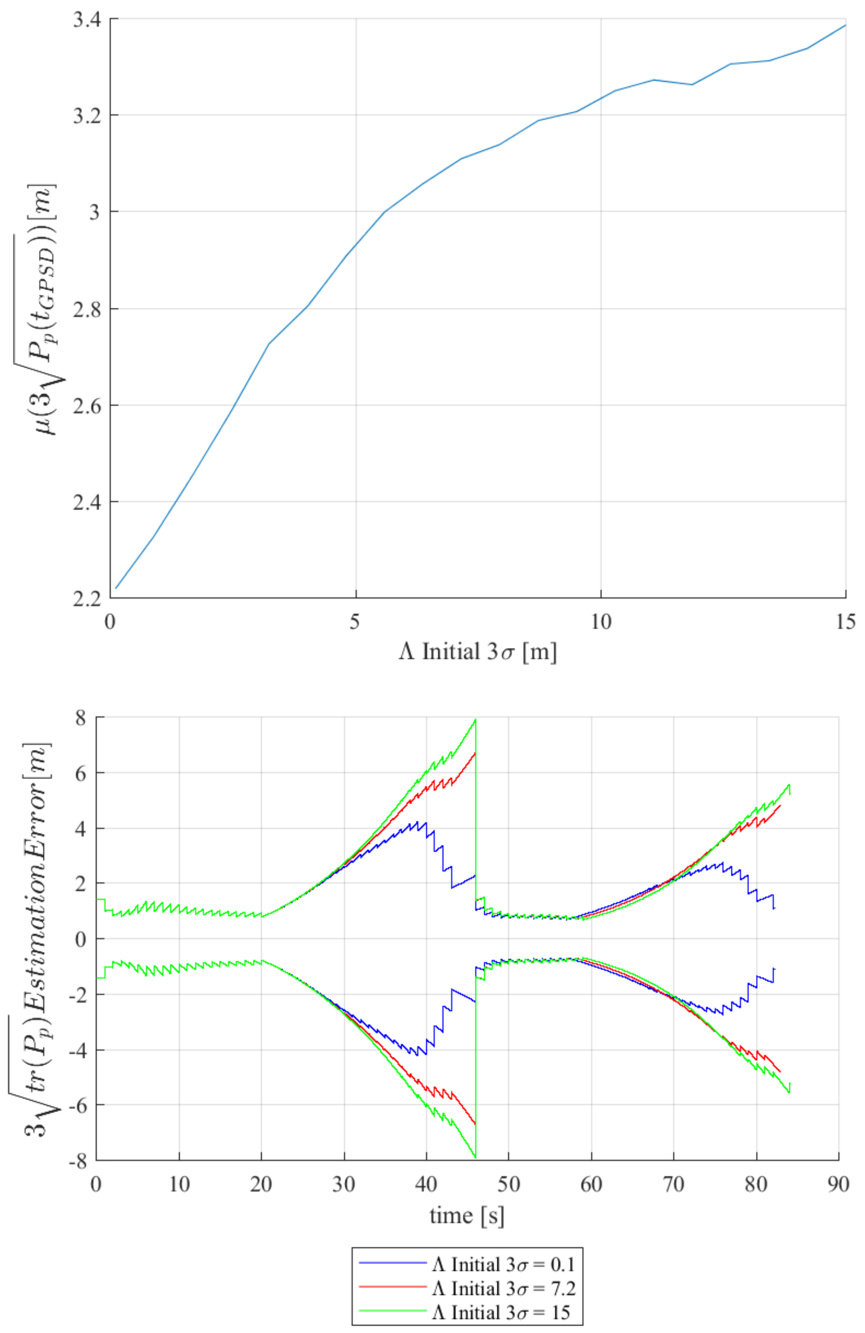


Figure 3.2. Navigation study initial landmark uncertainty.

Figure 3.2 depicts the change of mean error with initial landmark uncertainty. With increasing landmark error comes increasing position error. This positive correlation appears either linear or logarithmic. The nominal in this figure occurred at 0.5 m.

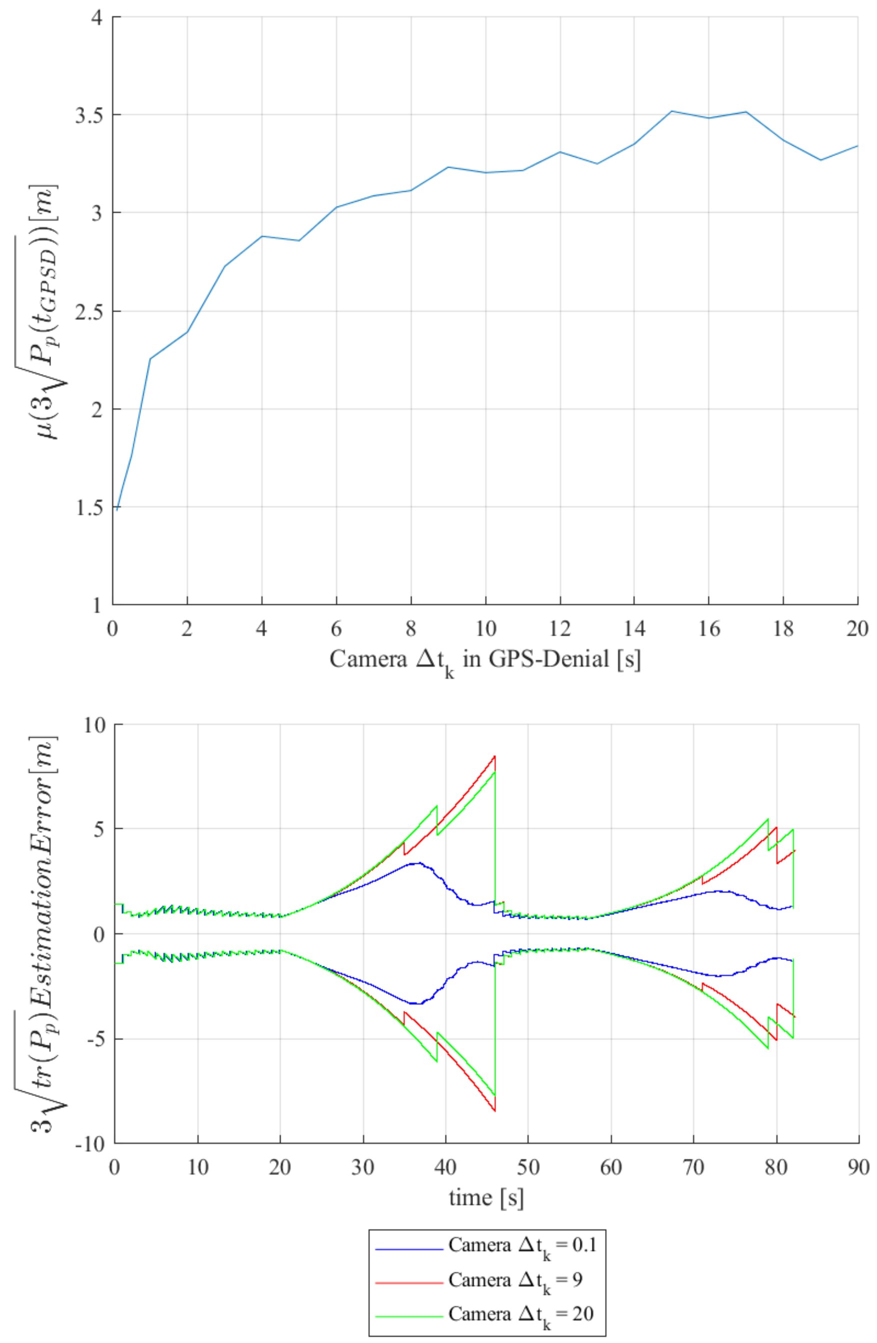


Figure 3.3. Navigation study number of landmark observations.

Figure 3.3 also shows a positive correlation between camera Δt_k and position mean error. The correlation is less smooth but has a more a defined logarithmic shape. A steeper change in position errors occurs with camera updates below the 1 s nominal.

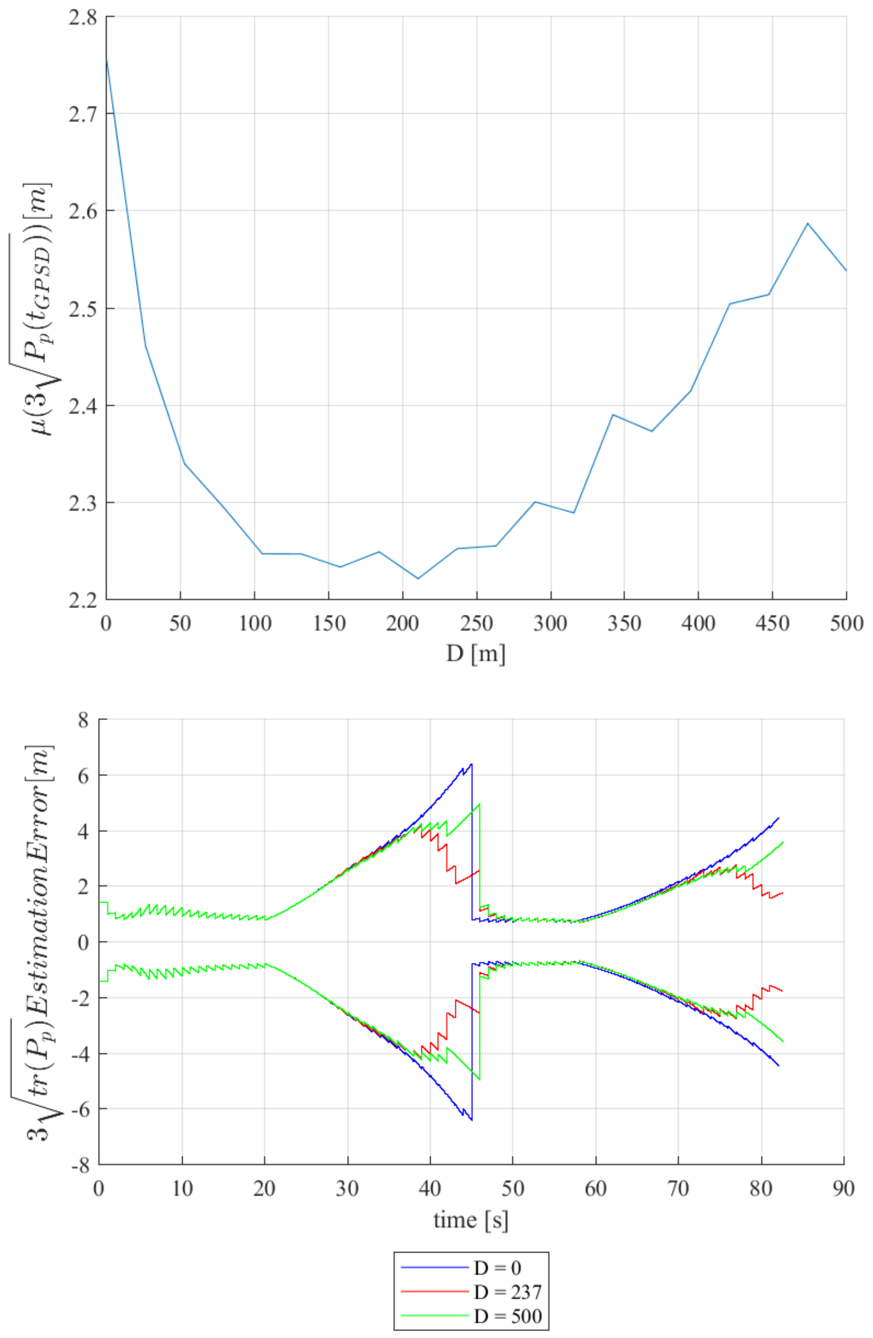


Figure 3.4. Navigation study measurement geometry.

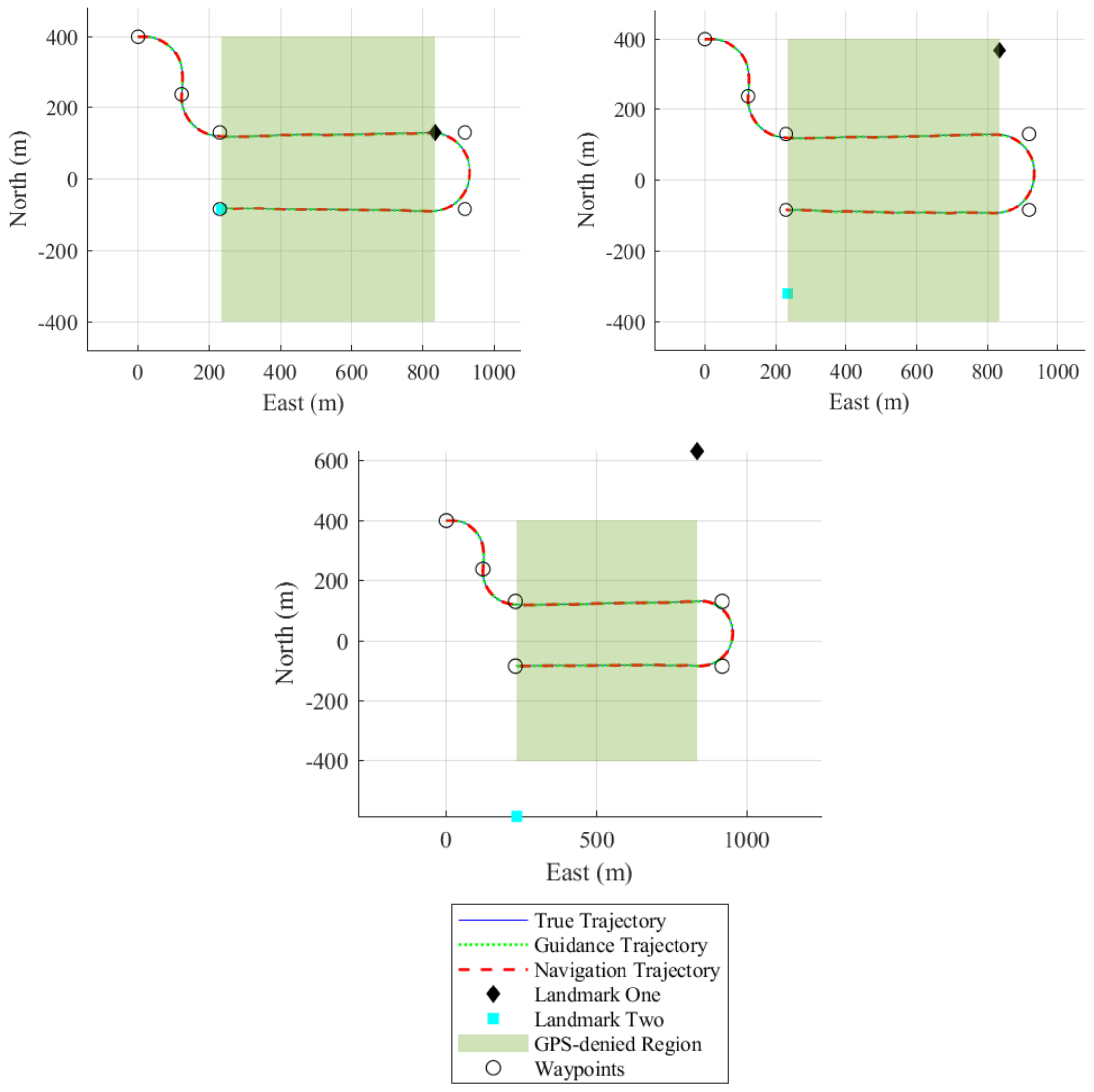


Figure 3.5. Navigation study measurement geometry trajectories.

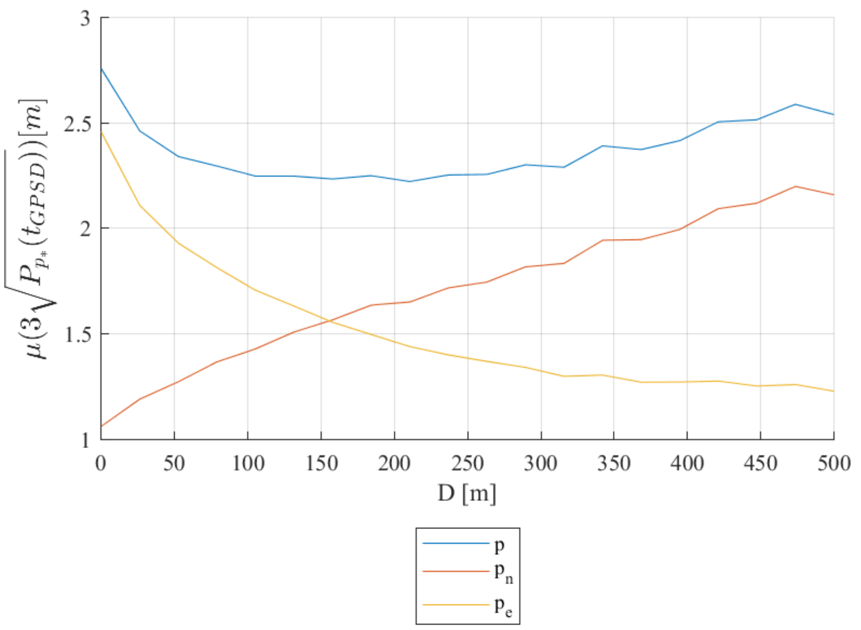


Figure 3.6. Navigation study measurement geometry with p_n and p_e components.

For measurement geometry, although the y-axis range is small, a convex relationship was found between position error and landmark north distance. Distances within $D = 100\ m$ and $D = 250$ (between the first and second trajectories in Figure 3.5) yielded errors slightly less than the nominal. To investigate the relationship, Figure 3.6 was made. This figure replots Figure 3.4 (blue) but also plots the mean 3σ position error for p_n (red) and p_e (orange) individually. p_n shows a positive correlation with D, whereas p_e shows a negative correlation with D.

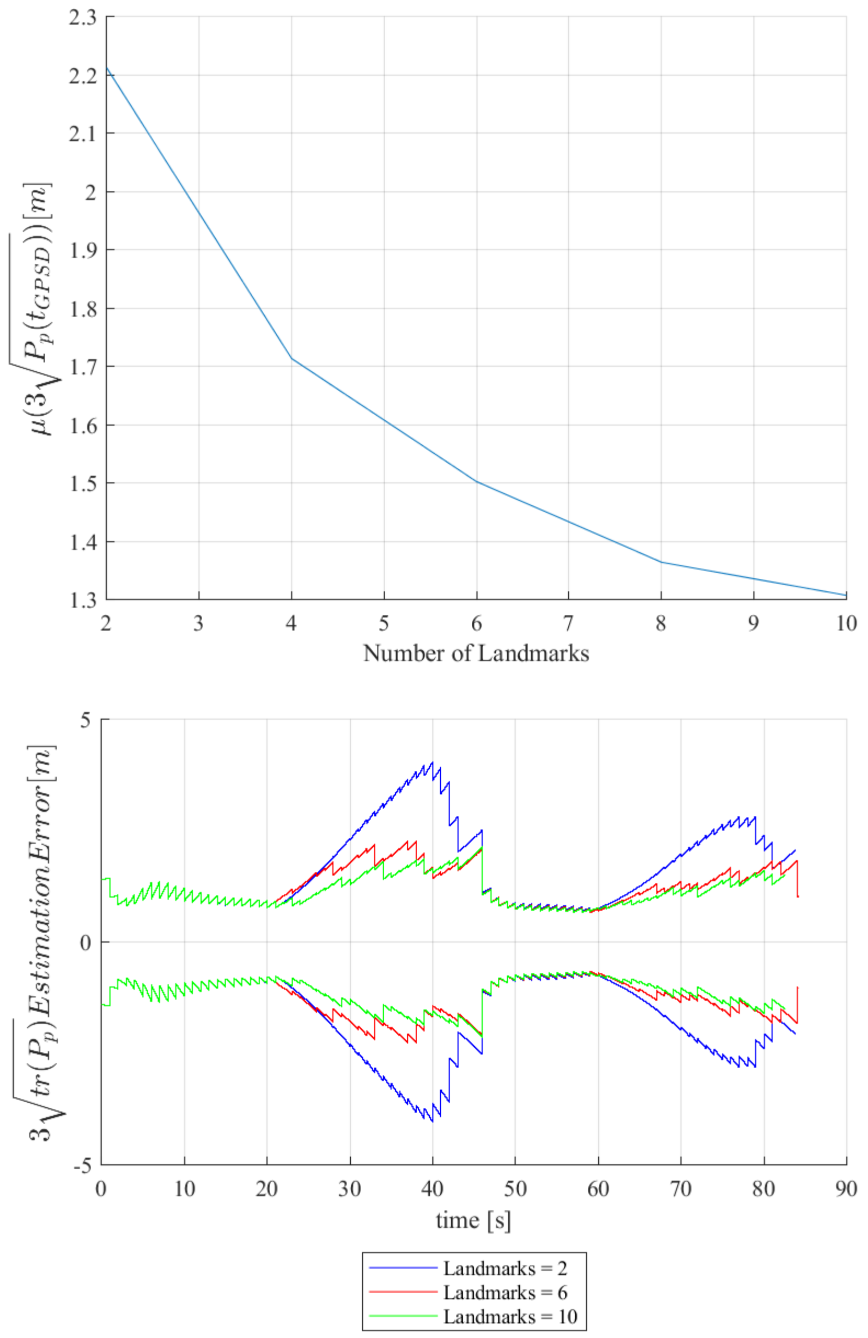


Figure 3.7. Navigation study number of landmarks.

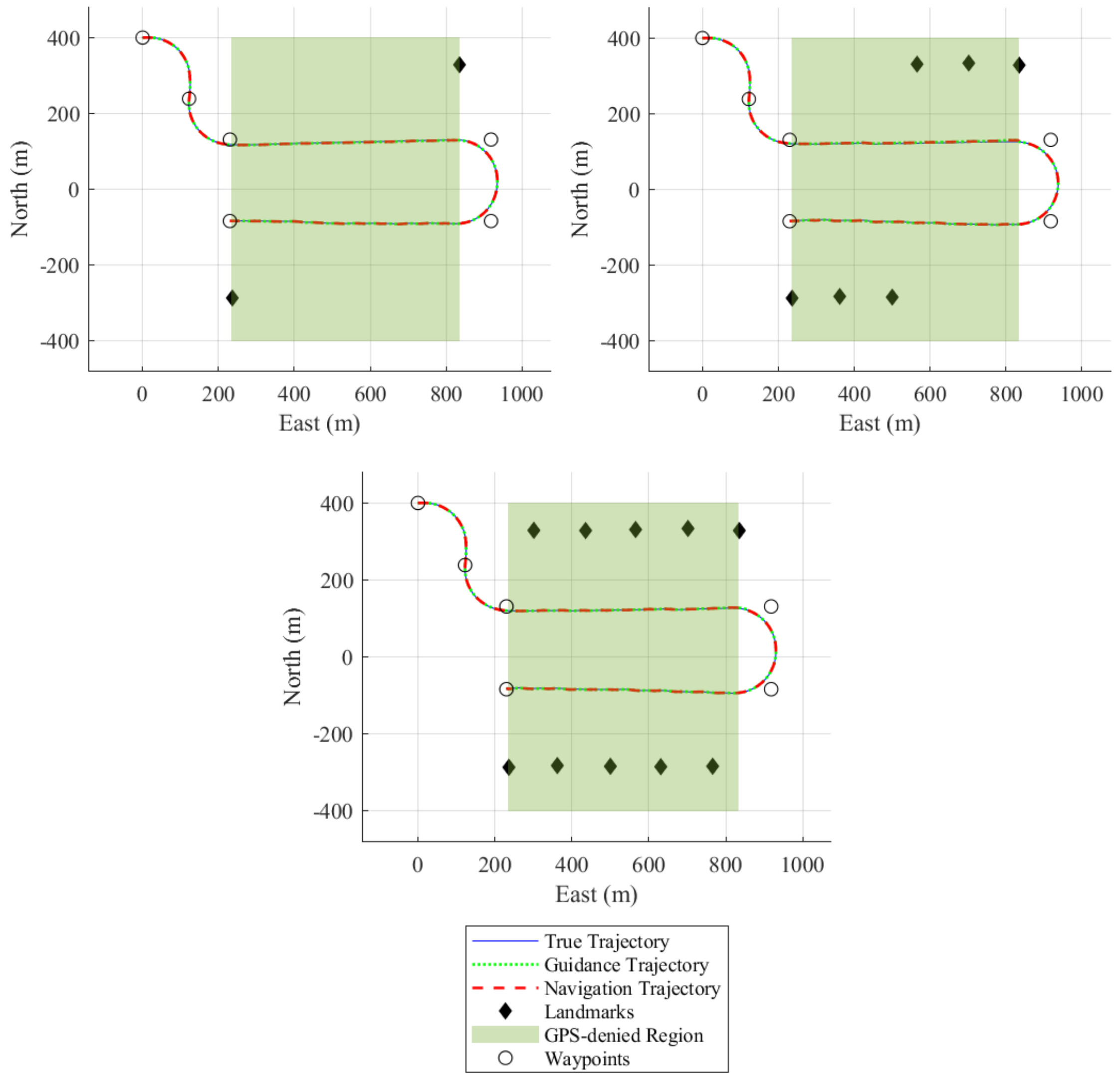


Figure 3.8. Navigation study number of landmarks trajectories.

The additional landmarks in Figure 3.8 varied the nominal's east position. The mean error range between 2 and 10 landmarks from Figure 3.7 is less than 1 m. Nevertheless, more landmarks always produced less position error than the nominal.

3.2.2 Navigation Sensitivity Interpretation

With the given statistics, IMU sensor grade appears to have the most drastic effect on navigation errors. Sensor grade variations produced a range that held both the largest and smallest errors than any other studied variation. Landmark observations have the second highest effect since landmark observations have the second highest error range. However, this larger range is only possible if camera processing speeds are faster than 1 s. Achieving such speeds could become the costliest and most time-consuming objective for real-world implementation. Initial landmark uncertainty seems to have limited effect on position errors. Since the Kalman filter's linearization can only tolerate missions having a maximum of 15 m initial landmark uncertainty, this limits the extent to which landmark uncertainty can affect UAV position errors.

The number of landmarks has a lower effect on position errors than most of the other studied variations. However, perhaps the method of replacing the landmarks in and out of the current state and covariance limited the full potential of multiple landmarks. In this method, the UAV could only observe one landmark per update cycle. Perhaps an alternative method would be to perform replacements and updates with every observable landmark per update cycle.

When considering measurement geometry, the small error range implies that measurement geometry has little effect on position errors. Looking at the mean position error for p_n and p_e individually, there is a tradeoff. For low D distances, p_e error will dominate due to the lack of orthogonal information in the H matrix. Near 270 m, more of p_n's orthogonal information trades off to p_e, and p_n then becomes the dominant source of error. So, since valuable position information is still able to be received at high landmark angles, perhaps a larger influence on position errors is not necessarily by measurement geometry but by FOV. To explain, in these studies, FOV was enlarged to ensure measurement availability throughout GPSD. However, in circumstances where FOV is smaller, simulations with larger D would incur larger position error, since the UAV would no longer see the landmark and would lose access to camera updates within the field.

3.3 Guidance and Control Sensitivity Study

For the GC study, the nominal simulation setup from the navigation study is selected (simulation number 2, $G_1 = $ 1.7e-2, $G_2 = G_3 = G_4 = 0$). The objective is to activate and vary the gains in the GC law to see if any significant reduction in $\mu_{3\sigma_p}$, can be acquired. The following simulations were run:

Table 3.6. GC study gains.

Simulation Number	Activated Gain	Gain Value	Simulation Number	Activated Gain	Gain Value	Desired Speed $\left(\frac{m}{s}\right)$
71		5.1e-04	111		2.8e-02	16.0e+00
72		1.0e-03	112		2.8e-02	16.5e+00
73		1.5e-03	113		2.8e-02	16.9e+00
74		2.1e-03	114		2.8e-02	17.4e+00
75		2.6e-03	115		2.8e-02	17.9e+00
76		3.1e-03	116		2.8e-02	18.4e+00
77		3.6e-03	117		2.8e-02	18.8e+00
78		4.1e-03	118		2.8e-02	19.3e+00
79		4.6e-03	119		2.8e-02	19.8e+00
80	G_2	5.1e-03	120	G_4	2.8e-02	20.3e+00
81		5.7e-03	121		2.8e-02	20.7e+00
82	(v)	6.2e-03	122	(u)	2.8e-02	21.2e+00
83		6.7e-03	123		2.8e-02	21.7e+00
84		7.2e-03	124		2.8e-02	22.2e+00
85		7.7e-03	125		2.8e-02	22.6e+00
86		8.2e-03	126		2.8e-02	23.1e+00
87		8.7e-03	127		2.8e-02	23.6e+00
88		9.3e-03	128		2.8e-02	24.1e+00
89		9.8e-03	129		2.8e-02	24.5e+00
90		1.0e-02	130		2.8e-02	25 e+00
91		1.2e-02				
92		2.4e-02				
93		3.6e-02				
94		4.8e-02				
95		6.0e-02				
96		7.2e-02				
97		8.4e-02				
98		9.6e-02				
99		1.1e-01				
100	G_3	1.2e-01				
101		1.3e-01				
102	(γ)	1.4e-01				
103		1.6e-01				
104		1.7e-01				
105		1.8e-01				
106		1.9e-01				
107		2.0e-01				
108		2.2e-01				
109		2.3e-01				
110		2.4e-01				

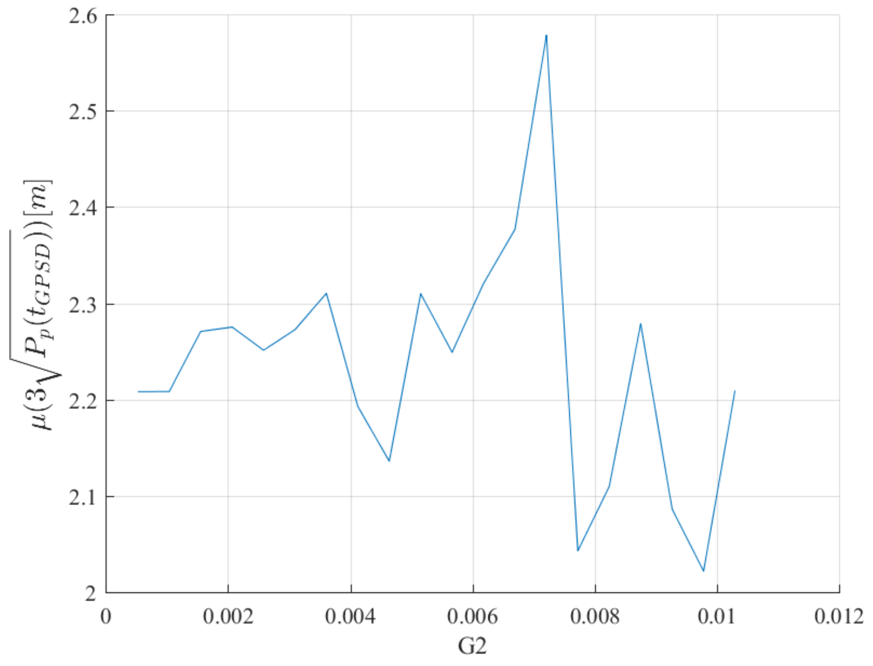
For every GC study, only one potential-error-reducing cost is activated at a time with the constant, nominal waypoint following cost. A relatively low to high gain is applied to the potential-error-reducing cost term. The exception to this is the forward velocity, which uses a constant gain while the desired speed varied. Future work can perform more extensive research of combination gains.

Before continuing, an explanation on gain value selection may be desired. In short, Bryson's method [24] with tuning was used. In detail, linear scaling gains so that, for example, hard rudder inputs are two times more important than waypoint following ($G_1 = 1$, $G_2 = 2$) will not produce the same scaled trajectory. This is due to the relative units of each cost, where much higher gains are needed, for example, for radian costs to "compete" with meter costs. Using Bryson's method alone did not provide the desired competitions between costs. To tune, the author kept Bryson's method for all costs (divide by the largest possible numerator value). For waypoint following, the maximum value was estimated as the largest distance between 2 consecutive waypoints. After applying Bryson's method, the author activated waypoint following only and gradually multiplied more gain until the UAV would make a concise trajectory with limited overshoot at waypoints and straightened paths between waypoints. Keeping the waypoint gain constant, the other costs were individually activated. Low gains had trajectories that favored waypoint following; high gains had trajectories that favored the activated cost. For lateral velocity, the gain that induced the highest curved trajectory while still successfully completing the waypoint mission was chosen as the max gain. For camera-landmark angle, the gain that forced the UAV toward the landmark most, where additional gain was insignificant in reducing γ, was used as the max gain. For forward velocity, the lowest gain that would allow for the lowest velocity to be achieved was used as the constant for all other desired velocities.

Admittedly, the trial-and-error iterations of gain tuning were much like PID tuning, which was not efficient. Bryson's method was not necessarily helpful in converging to the proper gains. Furthermore, the tuning did not appear to be independent of trajectory or simulation time length. Finding a cleaner non-tuning procedure in gain selection, perhaps like feedback linearization, is left to future work.

With the specifications for each study laid out, the document now moves to presenting simulation results and interpretation.

3.3.1 Guidance and Control Sensitivity Results



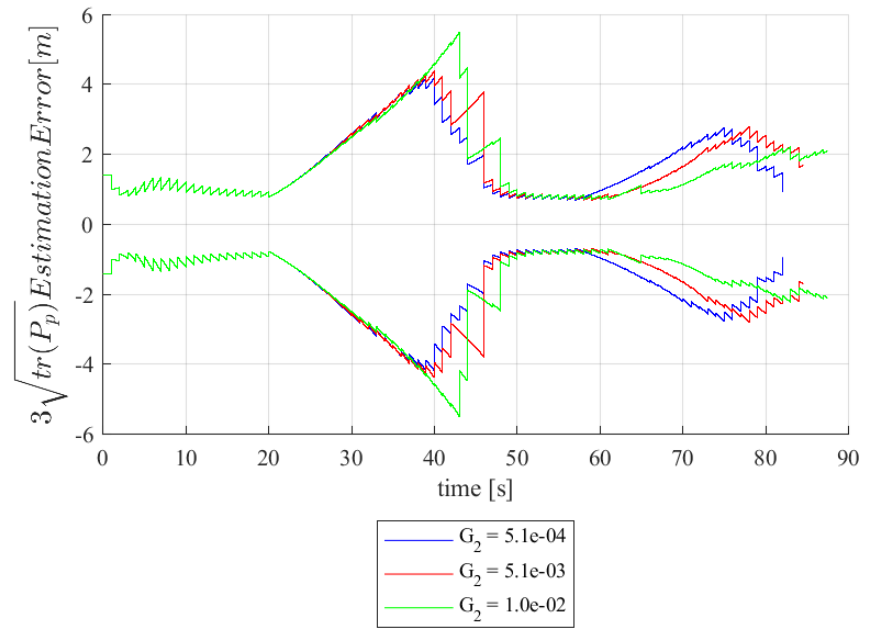


Figure 3.9. GC study lateral velocity (G_1 = 1.7e-2, $G_3 = G_4 = 0$).

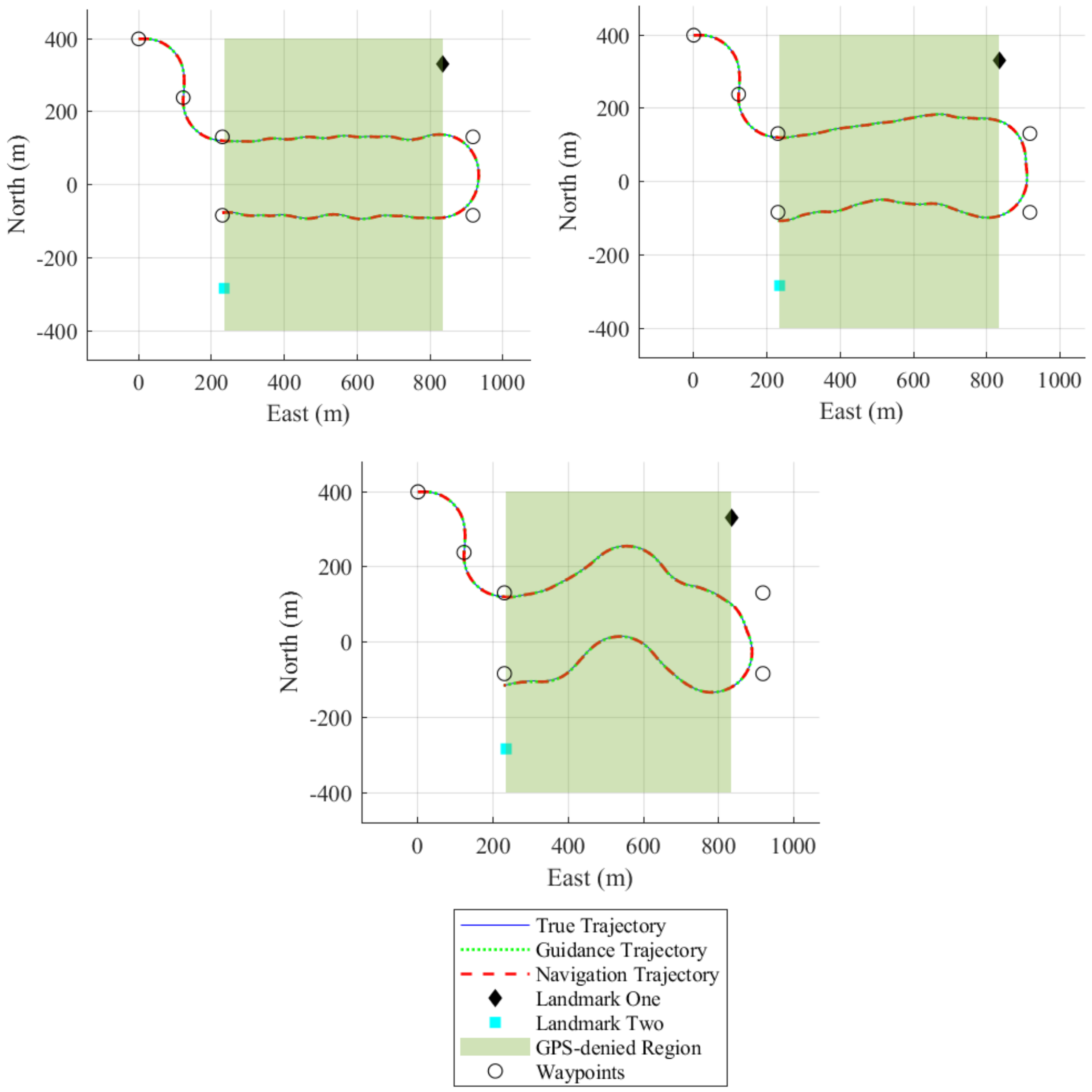


Figure 3.10. GC study lateral velocity trajectories.

Using higher lateral velocity gain does induce higher curvature (observability) in Figure 3.10. Using higher lateral velocity gains also generated the smallest error for all GC simulation test cases in Figure 3.9. However, higher lateral velocity gains had more erratic spikes than other costs. For example, $G_2 = 7.7e - 03$ yielded one of the smallest mean errors, but the prior gain, $G_2 = 7.2e - 03$, yielded lateral velocity's largest mean error (rather than yielding a small mean error).

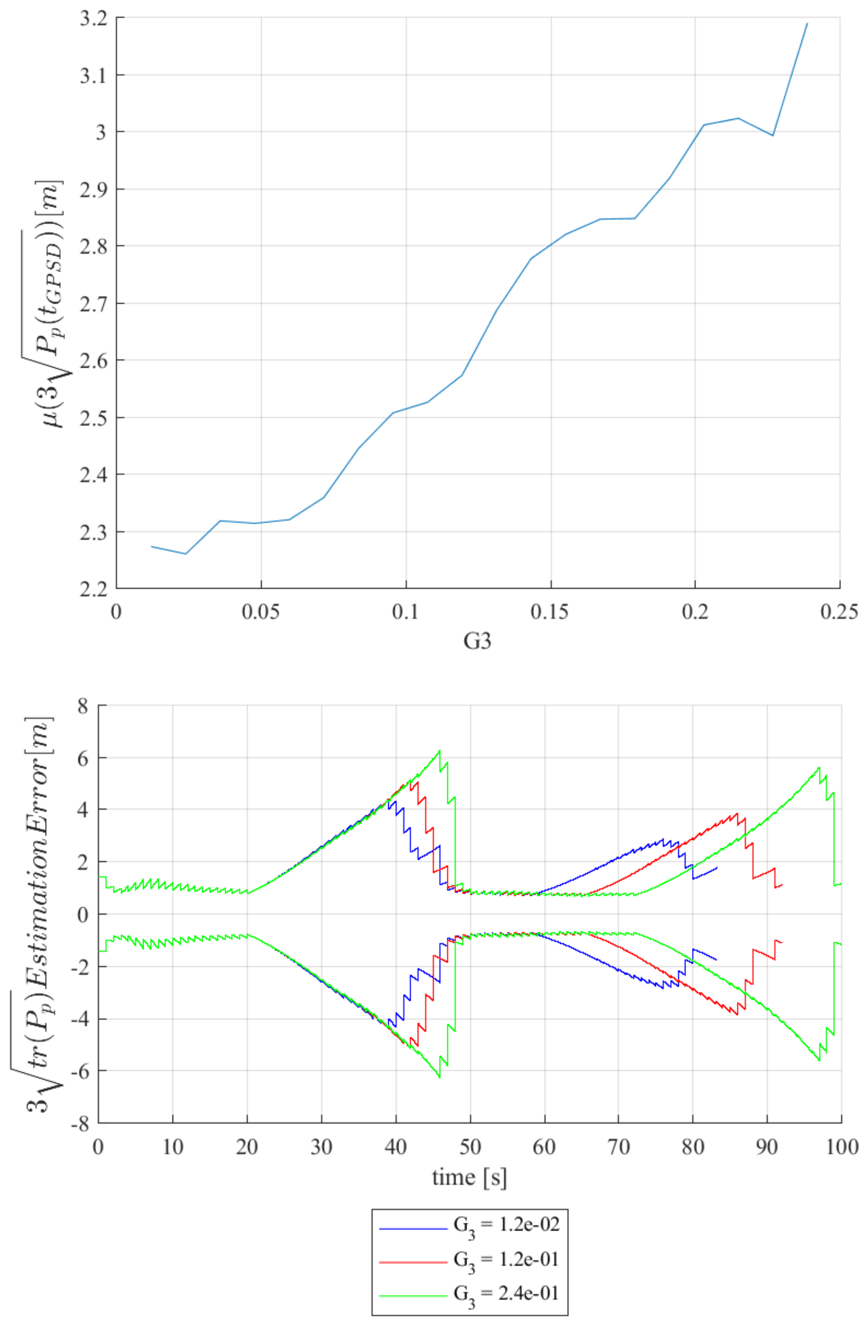


Figure 3.11. GC study landmark centering ($G_1 = 1.7\text{e-}2$, $G_2 = G_4 = 0$).

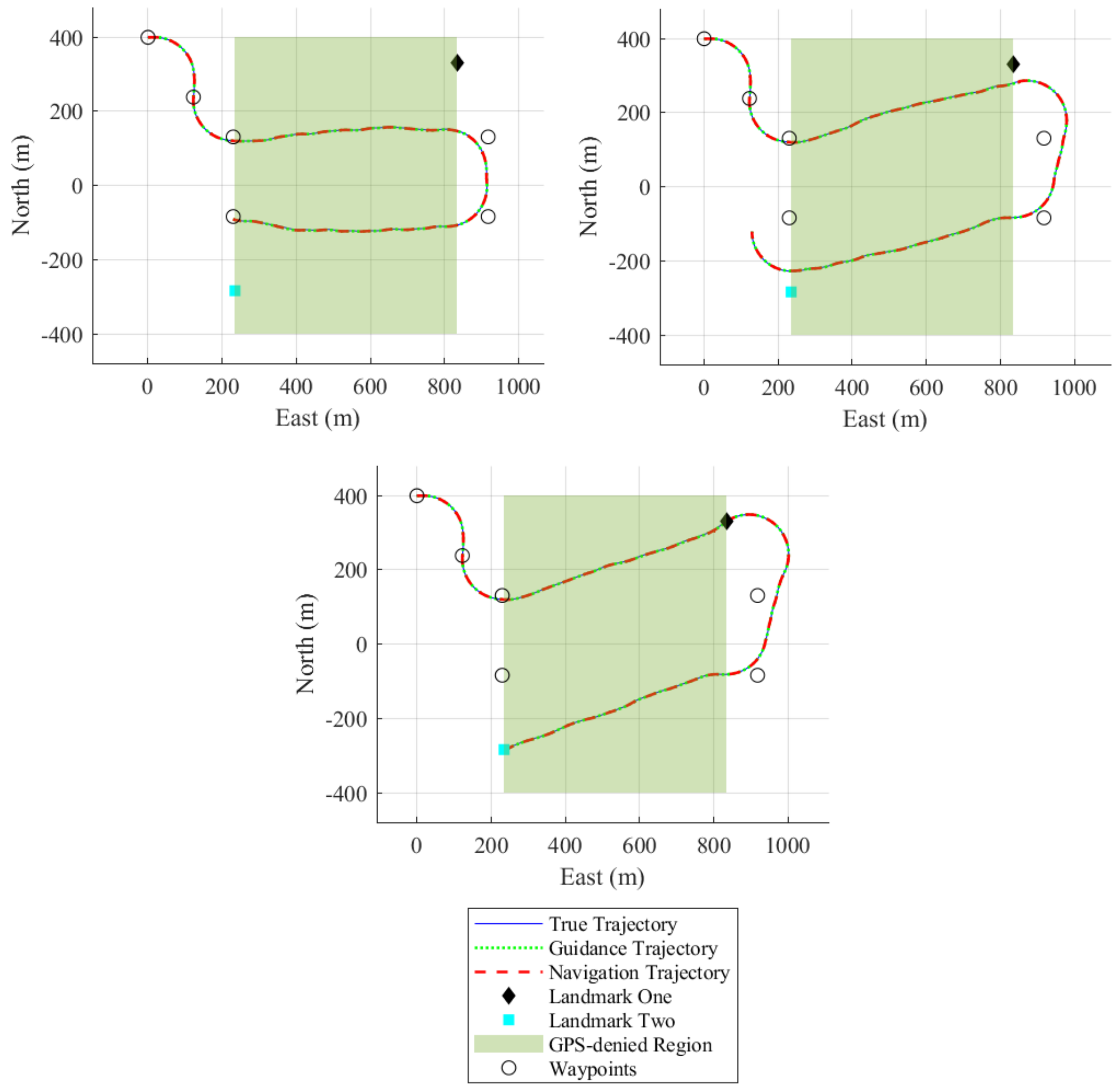
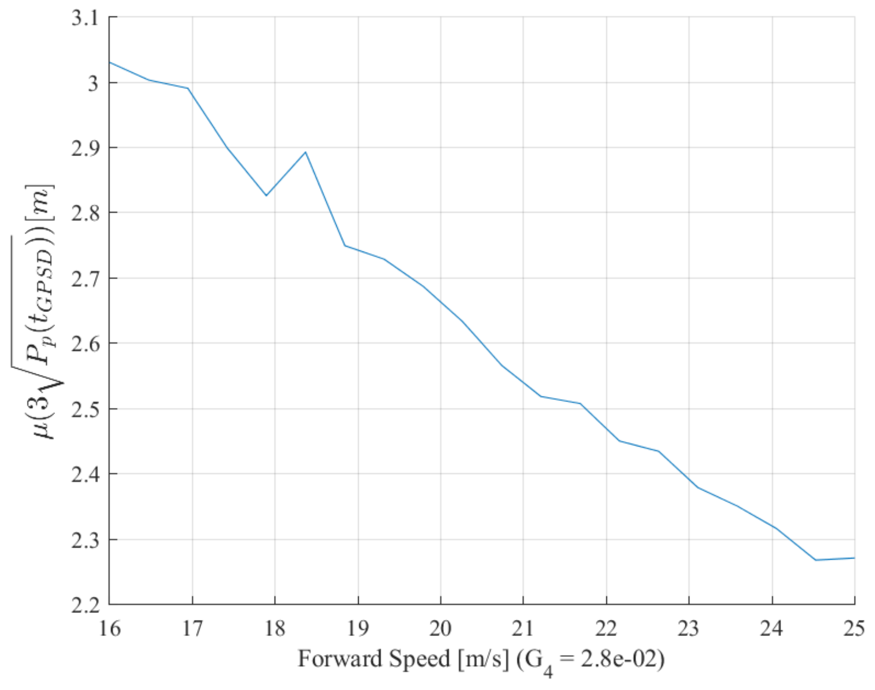


Figure 3.12. GC study landmark centering trajectories.

For landmark centering, Figure 3.12 shows that higher G_3 gains can direct the UAV toward the landmarks and induce higher levels of landmark centering. In Figure 3.11, a positive correlation was found between G_3 and position error. None of the G_3 gains reduced position errors more than the nominal. Reviewing the estimation error plot, shorter simulation times are associated with lower gains.



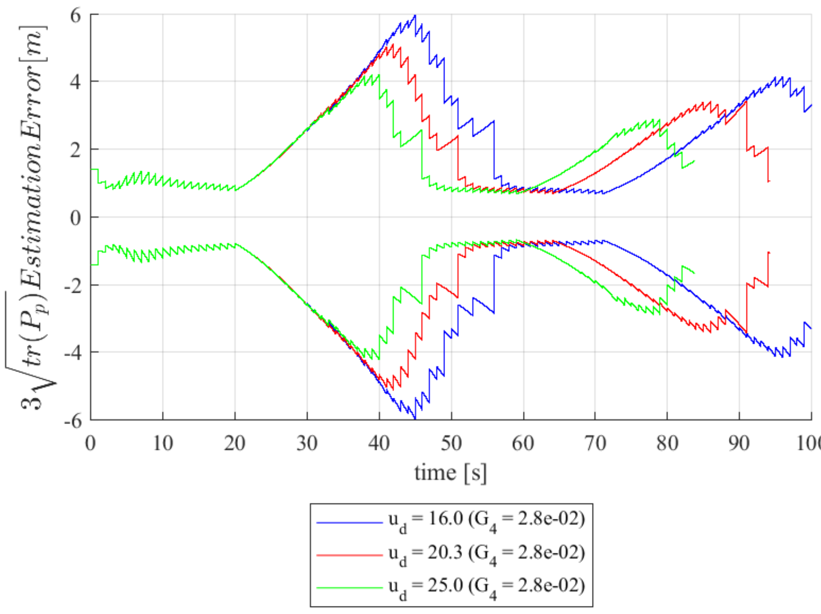


Figure 3.13. GC study forward speed ($G_1 = 1.7\text{e-}2$, $G_2 = G_3 = 0$).

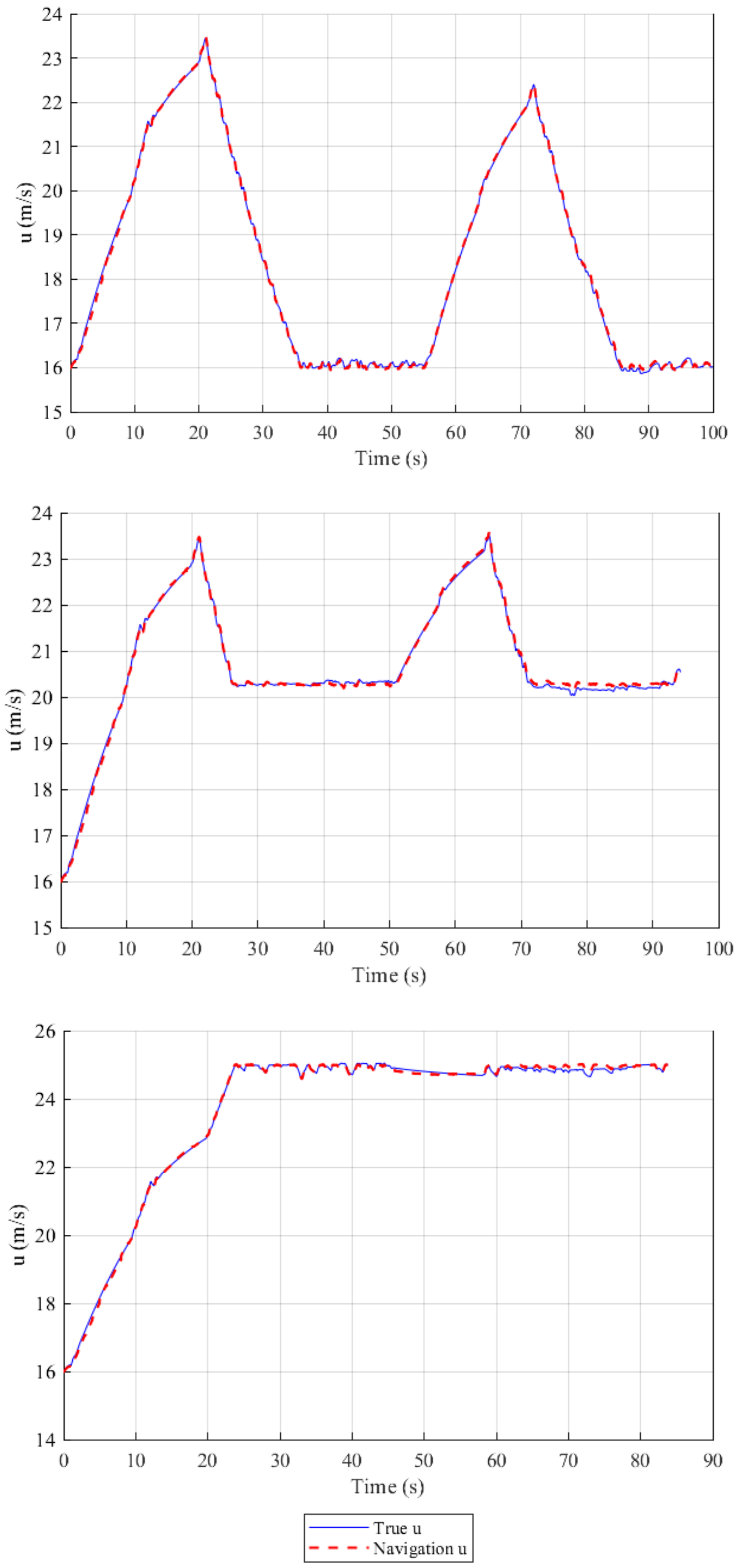


Figure 3.14. GC study forward speed velocities.

To show the influence of the forward velocity cost, forward velocity plots (instead of trajectory plots) for low, medium, and high u_d are shown from top to bottom respectively in Figure 3.14. Spikes in these velocity plots correspond to periods outside GPSD, where only the waypoint following gain is activated (again, waypoint following naturally encourages higher speeds). Inside GPSD, a lower forward velocity cost is able to induce lower speeds A negative correlation is present in Figure 3.13. The presence of the forward speed cost did not improve errors below the nominal, waypoint-only cost (simulation number 2).

3.3.2 Guidance and Control Sensitivity Interpretation

Lateral velocity gain, although minimally, can be used to reduce position errors. However, future work will be needed to avoid the erratic spikes that occurred with one of the higher gains. One advantage with this gain, referring to Figure 3.9’s estimation error plot, is the allowance for more time in GPSD without significantly increasing position errors.

Landmark centering hindered error reduction rather than improved. This appeared to be due to the longer time periods in GPSD that came with higher G_3 values. Perhaps, similar to the navigation study’s measurement geometry results, a larger influence on position errors is not by image distortion, but by FOV. For example, in circumstances where FOV is smaller, landmark centering could have larger influence on reducing position errors, where obtaining more camera updates could become more critical than the time spent in GPSD.

Though none of the forward speeds made lower position error than the nominal, the important item to note is how forward speed affects position uncertainties. The error plot’s linear trend showed how there is no benefit in flying slow in GPSD. For the given mission, the straight trajectories through GPSD naturally encouraged waypoint following to favor higher speeds. However, for missions that include waypoint turning within GPSD, waypoint following would encourage slower speeds to avoid overshooting a desired waypoint. Perhaps in those scenarios, having a high forward speed gain would be more critical to avoid higher position errors that come with larger times in GPSD.

CHAPTER 4

CONCLUSION

Having all GNC background, development, and results completed, this chapter will reflect on the developed key points from all prior chapters.

For modern military development, UAVs must be able to travel successfully through GPSD regions. To help with this objective, this research explored UAV position error sensitivities to sensor grade, landmark characteristics, and GC cost function gains. From prior research, navigation errors reduce significantly with the use of Filters and vision-based measurements; nevertheless, there was lack of quantitative understanding as to the amount in which each mission factor played in reducing navigation errors. Two of these factors are the UAV’s guidance and the UAV’ control. GC can be as simple as waypoint following and carrot chasing or as complicated as the star algorithms and nonlinear control. However, despite all the varieties in guidance and control, little research has been performed to quantitatively assess how these two factors affect navigation errors.

In an attempt to quantitatively assess reducing navigation errors, a GNC simulation was developed. The simulation included an Aerosonde drone with an IEKF, pinhole camera, and a direct multiple shooting method GC law. First, simulation development and validation were described to ensure correct implementation. Second, two simulation studies were conducted. In the validation simulations, truth, design model, and navigation dynamics were defined. Error mappings were determined, and simulation propagation was successfully executed. Following propagation validation was the extensive development of the IEKF’s key equations (Linear Error State Modeling, Linear

Error Measurement Modeling, Covariance Propagation, and Estimation Capability). Explanation, development, and validation of the GC law was also given. The extensiveness of this development provided implementation validity for this document's studies.

The first study investigated how, using a nominal GC law, position errors are affected as a function of IMU sensor grade, landmark uncertainty, number of landmark observations, measurement geometry, and number of landmarks. Sensor grade yielded the most drastic effect on position errors. Landmark uncertainty's effect was limited by the Kalman filter's linearization. Number of landmarks had the second highest effect on position errors. Measurement geometry has a convex relationship with position, where camera measurement information was minimally lost when landmarks were positioned at high or low angles relative to the camera's FOV. Number of landmarks had even less effect on position errors, perhaps due to the method of implementation.

The second study investigated how position errors are affected as a function of GC costs and gains. High gains for a lateral velocity cost can lower position errors, but erratic tendencies involved with the high gains need to be resolved first. Higher gains for a landmark centering cost do not improve position errors due to extended times in GPSD. Forward speed costs influenced position errors, particularly where low speed gains induce larger position errors.

Suggestions for future work were mentioned throughout this document; nevertheless, there is a need for live implementation. Simulation results need to be compared with 3D simulations and physical UAV-flights with a real-time control law. Physical UAV flights will also identify factors whose influence alters the findings in this document, i.e., camera image processing, and wind effects. After live implementation and comparison is achieved, there is a need to expand these studies out to more modern mission factors, for example, multiple cameras, collaborative UAV swarms (inter-vehicle avoidance), and moving landmarks. It is hoped that this document has provided a sufficient baseline for this future work.

BIBLIOGRAPHY

[1] Kendoul, F., 2012, “Survey of Advances in Guidance, Navigation, and Control of Unmanned Rotorcraft Systems,” J. Field Robot., **29**(2), pp. 315–378.

[2] Balamurugan, G., Valarmathi, J., and Naidu, V. P. S., 2016, “Survey on UAV Navigation in GPS Denied Environments,” *2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES)*, pp. 198–204.

[3] Maybeck, P. S., 1979, *Stochastic Models, Estimation, And Control*, Academic Press, New York, New York, United States of America.

[4] Smith, N. E., Cobb, R., Pierce, S., and Raska, V., 2013, “Optimal Collision Avoidance Trajectories for Unmanned/Remotely Piloted Aircraft,” *AIAA Guidance, Navigation, and Control (GNC) Conference*, American Institute of Aeronautics and Astronautics, Boston, MA.

[5] Smith, N. E., Cobb, R., Pierce, S. J., and Raska, V., 2014, “Optimal Collision Avoidance Trajectories via Direct Orthogonal Collocation for Unmanned/Remotely Piloted Aircraft Sense and Avoid Operations,” *AIAA Guidance, Navigation, and Control Conference*, American Institute of Aeronautics and Astronautics.

[6] Scaramuzza, D., Achtelik, M. C., Doitsidis, L., Friedrich, F., Kosmatopoulos, E., Martinelli, A., Achtelik, M. W., Chli, M., Chatzichristofis, S., Kneip, L., Gurdan, D., Heng, L., Lee, G. H., Lynen, S., Pollefeys, M., Renzaglia, A., Siegwart, R., Stumpf, J. C., Tanskanen, P., Troiani, C., Weiss, S., and Meier, L., 2014, “Vision-Controlled Micro Flying Robots: From System Design to Autonomous Navigation and Mapping in GPS-Denied Environments,” IEEE Robot. Autom. Mag., **21**(3), pp.

26–40.

[7] Nuske, S. T., Dille, M., Grocholsky, B., and Singh, S., “Representing Substantial Heading Uncertainty for Accurate Geolocation by Small UAVs,” *AIAA Guidance, Navigation, and Control Conference*, American Institute of Aeronautics and Astronautics, p. 13.

[8] Perez-Grau, F. J., Ragel, R., Caballero, F., Viguria, A., and Ollero, A., 2018, “An Architecture for Robust UAV Navigation in GPS-Denied Areas,” J. Field Robot., **35**(1), pp. 121–145.

[9] Babinec, A., and Apeltauer, J., 2016, “On Accuracy of Position Estimation from Aerial Imagery Captured by Low-Flying UAVs,” Int. J. Transp. Sci. Technol., **5**(3), pp. 152–166.

[10] Madison, R., Andrews, G., DeBitetto, P., Rasmussen, S., and Bottkol, M., “Vision-Aided Navigation for Small UAVs in GPS-Challenged Environments,” *AIAA Infotech@Aerospace 2007 Conference and Exhibit*, American Institute of Aeronautics and Astronautics.

[11] Saska, M., Baca, T., Thomas, J., Chudoba, J., Preucil, L., Krajnik, T., Faigl, J., Loianno, G., and Kumar, V., 2017, “System for Deployment of Groups of Unmanned Micro Aerial Vehicles in GPS-Denied Environments Using Onboard Visual Relative Localization,” Auton. Robots, **41**(4), pp. 919–944.

[12] Schnaufer, B. A., Hwang, P., Nadke, J., McGraw, G. A., and Venable, D., 2012, “Collaborative Image Navigation Simulation and Analysis for UAVs in GPS Challenged Conditions,” *Proceedings of the 2012 IEEE/ION Position, Location and Navigation Symposium*, pp. 719–729.

[13] Beard, R. W., and McLain, T. W., 2012, *Small Unmanned Aircraft: Theory and Practice*, Princeton University Press, Princeton, N.J.

[14] Anderson, E. P., Beard, R. W., and McLain, T. W., 2005, “Real-Time Dynamic Trajectory Smoothing for Unmanned Air Vehicles,” IEEE Trans. Control Syst. Technol., **13**(3), pp. 471–477.

[15] Davis, J. D., and Chakravorty, S., 2007, “Motion Planning Under Uncertainty: Application to an Unmanned Helicopter,” J. Guid. Control Dyn., **30**(5), pp. 1268–1276.

[16] Mettler, B., Kong, Z., Goerzen, C., and Whalley, M., 2014, “Guidance Performance Benchmarking for Autonomous Rotorcraft,” J. Am. Helicopter Soc., **59**(4), pp. 1–16.

[17] Vanegas, F., and Gonzalez, F., 2016, “Uncertainty Based Online Planning for UAV Target Finding in Cluttered and GPS-Denied Environments,” *2016 IEEE Aerospace Conference*, pp. 1–9.

[18] Tsenkov, P., Howlett, J., Whalley, M., Schulein, G., Takahashi, M., Rhinehart, M., and Mettler, B., 2008, “A System for 3D Autonomous Rotorcraft Navigation in Urban Environments,” *AIAA Guidance, Navigation and Control Conference and Exhibit*, American Institute of Aeronautics and Astronautics, Honolulu, Hawaii.

[19] Sujit, P. B., Saripalli, S., and Sousa, J. B., 2014, “Unmanned Aerial Vehicle Path Following: A Survey and Analysis of Algorithms for Fixed-Wing Unmanned Aerial Vehicles,” IEEE Control Syst. Mag., **34**(1), pp. 42–59.

[20] Joseph J. DisStefano III, Allen R. Stubberud, and Ivan J. Williams, 2014, *Schaum’s Outline of Feedback and Control Systems*, McGraw-Hill Education, New York,

Chicago, San Francisco, Athens, London, Madrid, Mexico City, Milan, New Delhi, Singapore, Sydney, Toronto.

[21] Khalil, H. K., 2002, *Nonlinear Systems*, Prentice Hall, Upper Saddle River, New Jersey.

[22] Kaminer, I., Pascoal, A., Xargay, E., Hovakimyan, N., Cao, C., and Dobrokhodov, V., 2010, "Path Following for Small Unmanned Aerial Vehicles Using L1 Adaptive Augmentation of Commercial Autopilots," J. Guid. Control Dyn., **33**(2), pp. 550–564.

[23] Bryson, A. E., and Ho, Y.-C., 1975, *Applied Optimal Control: Optimization, Estimation, and Control*, Hemisphere Pub. Corp. ; distributed by Halsted Press, Washington : New York.

[24] Joao P. Hespanha, 2018, *Linear Systems Theory*, Princeton University Press, Princeton, N.J.

[25] Chapra, S. C., and Canale, R. P., 2015, *Numerical Methods for Engineers*, McGraw-Hill Education, New York, NY.

[26] Maybeck, P. S., 1982, *Stochastic Models, Estimation, and Control*, Academic Press, New York, New York, United States of America.

[27] Kappen, H. J., 2008, "Stochastic Optimal Control Theory," Radboud Univ. Nijmegen Neth., p. 34.

[28] Maybeck, P. S., 1982, *Stochastic Models, Estimation, and Control. Vol. 2: ...*, Acad. Press, New York.

[29] Christensen, R. S., and Geller, D., 2014, "Linear Covariance Techniques for Closed-Loop Guidance Navigation and Control System Design and Analysis," Proc. Inst.

Mech. Eng. Part G J. Aerosp. Eng., **228**(1), pp. 44–65.

[30] Kasdin, N. J., 1995, “Runge-Kutta Algorithm for the Numerical Integration of Stochastic Differential Equations,” J. Guid. Control Dyn., **18**(1), pp. 114–120.

[31] Fathi, M. K., and Malaek, S. M. B., 2019, “Wild Area Aerial Protective Inspector: Proposal for 2018-2019 Undergraduate Individual Aircraft Linear Infrastructure Inspection UAS.”

[32] Droge, G., 2020, “Optimal and Robust Control Lecture Notes.”

[33] Hunsaker, D., 2021, “Re: Max UAV Thrust.”

[34] Hunsaker, D., 2021, “Comments on Thesis.”

[35] Fotouhi, A., Ding, M., and Hassan, M., 2017, “Understanding Autonomous Drone Maneuverability for Internet of Things Applications,” *2017 IEEE 18th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, IEEE, Macau, China, pp. 1–6.

[36] 2015, “Dec 9th, 2015 04:38PM | Controls / Rudder Response | DJI GO Log 2.4.2 | Japan | Airdata UAV (Formerly HealthyDrones.Com).”

[37] Christensen, R., Droge, G., and Leishman, R., 2021, “A Closed-Loop Linear Covariance Framework for Vehicle Path Planning in an Uncertain Obstacle Field,” ArXiv210511998 Cs Eess Stat.

[38] Waechter, A., and Laird, C., “Ipopt Documentation.”

[39] Andersson, J., Gillis, J., Diehl, M., Horn, G., Rawlings, J. B., and Diel, M., “CasADi - A Software Framework for Nonlinear Optimization and Optimal Control: User Documentation for CasADi v3.4.4,” p. 74.