Effects of terrain-based altimetry on navigation performance

Kenneth Michael Kratzer

EFFECTS OF TERRAIN-BASED ALTIMETRY ON NAVIGATION PERFORMANCE

by

KENNETH MICHAEL KRATZER

A THESIS

Presented to the Graduate Faculty of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

in

AEROSPACE ENGINEERING

2018

Approved by

Kyle DeMars, Advisor
Henry Pernicka
Serhat Hosder

ABSTRACT

Navigation filters used during an entry, descent, and landing scenario often receive measurements related to the terrain over which the vehicle is traversing. Models of the terrain can be presented in varying degrees of fidelity, providing increasingly realistic and accurate models of the surface. While it is ideal to provide the filter with the most realistic model possible, the performance is limited by the strict nature of the real time scenario. Therefore, it is of interest to determine what level of fidelity is necessary to achieve an accurate filtering solution. Determining how best to incorporate terrain-related data into navigation solutions can help produce the most robust and efficient navigation filter possible.

In the interest of determining a nominal filtering performance, an ellipsoidal model of the surface is used to create a baseline for the navigation filter. Increasingly complex digital elevation models are then utilized to better represent the surface over which the filter is navigating. These different fidelity models are then incorporated into various components of the filter to determine effects on the navigation performance. Simulations are performed that indicate that, as higher fidelity models are used, an increase in tuning noise must also be implemented to counteract the increasingly erratic nature of the terrain. The effects introduced by increasing terrain fidelity are also found to be reduced via the introduction of a lower fidelity estimate to the higher fidelity truth model.

Once simulations using an extended Kalman filter update are analyzed, alternative methods are developed to increase filter performance. Simulations with reduced uncertainties are performed to determine the effects of uncertainty on the terrain-based filtering solution. Following this, nonlinear transformations are implemented in an attempt to better account for the erratic nature of terrain-based models. While at lower fidelities these results provide increased performance, it is found that, as model fidelity is increased, the use of tuning noise or a lower fidelity estimate is once again required for a stable filtering solution.

ACKNOWLEDGMENTS

Firstly, I'd like to thank my advisor Dr. Kyle DeMars. You have provided me the opportunity and support I've needed throughout my graduate school career. Your guidance in both class and research has been vital to my success. You always held us to a higher standard and that has continued to push me to do better. Under your teaching I have gained a greater appreciation for our field of study and the hard work it takes to succeed in it. Without this, I would not have made it here today.

Second I would like to thank my committee members Dr. Hank Pernicka and Dr. Serhat Hosder. Your teachings have provided me with lessons and tools I know I will continue to find useful far into the future. Your passion for your fields of study are an inspiration I can only hope to aspire to. I look forward to taking the lessons you have given me and applying them as I move forward.

I want to give my greatest thanks to my parents and family. You are the ones who inspired me to attempt this journey, and continued to give me encouragement throughout it. You inspired me to work for my dreams and without your love and support I would not have made it to where I am today, so truly, thank you.

Finally, I would like to thank my friends and labmates. In an order that is extremely particular but also entirely nonsensical; James McCabe, Matt Gualdoni, Kari Ward, Christine Schmidt, Cameron Helmuth, Bruce Morrison, Casey Smith, Gunner Fritch, and Sasha Pertrenko. Thank you for the support, advice, and friendship you have provided throughout the past couple years. You all made coming to the lab every day a joyous occasion and I cannot thank you enough for that.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

1. INTRODUCTION

As exploration of the solar system continues to progress, more missions are moving from orbiting and collecting information to landing on the surface to get actual samples. NASA's launch of OSIRIS-REx [19] and the soon-to-be Mars 2020 [1] missions represent two vehicles that aim to land on two very different terrain types, but in both scenarios, a safe and reliable navigation filter is required for mission success. Part of the success that these missions have depends on how the filter can perform while acuiring and processing sensor data relative to the actual terrain of the surface. Sensors such as terrain relative navigation (TRN) [12] or nadir [6] and slant range altimeters [34] all use models of the surface in their measurement processes. Using a model that can accurately represent the real terrain is vital to the stability of the filter in allowing it to converge on an accurate state estimate. Knowing how close to the truth this model needs to be for the filter to perform properly allows a more flexible filter that can better adapt to the mission at hand. As mission planners continue to raise the bar in what they do, where they go, and their scientific utility once there, the need for robust and reliable navigation will need to continue to evolve with more accurate and efficient models to meet those demands.

1.1. BACKGROUND

1.1.1. Sensor Modeling. For a navigation filter performing in the scenario of entry, descent, and landing (EDL), an assortment of onboard sensors are used to find a state solution that reliably estimates the vehicle's position, velocity, and attitude. These sensors can include, but are not limited to, TRN, a star camera [20], slant range altimeter, and nadir-pointing altimeter. The sensors of the vehicle are modeled to provide data on measurements, such as altitude or descent rate, that are used in a navigation filter to produce an estimate of the vehicle state. The state can be updated using these models of the sensor in conjunction with sensor data. These updates depend on the model used in the filter and how precisely they agree with the sensor measurements. Since the sensor finds measurements using real world information, the closer these models can come to mimicking the real world environment, the more accurate a solution the filter will be able to find.

This can be shown when looking at, for instance, the altimeter sensor. This sensor acquires measurements based off of the surface below it, finding the vehicle's altitude above the real terrain. When modeling this measurement the ability to accurately represent that terrain with a high fidelity should provide a solution that more reliably converges to that real world measurement the sensor provides. Therefore, it is desirable to utilize a representation of the surface that can best map the real terrain the sensor uses. The downside to these higher fidelity models is that they often require more computational complexity and memory usage to actively model these sensors in real time. To compensate for this, it is desirable to understand how different model fidelities perform when modeling these sensors and what level of fidelity is required to provide a reliable filtering solution.

When modeling the terrain used for altimeter measurements, there are many fidelities of varying complexity that can be used to process a solution. The most basic of these is a spherical representation of the surface. By taking the radius of the body that is being landed on, a sphere can be constructed that represents the surface at its lowest fidelity. A slightly more complex solution is that of an ellipsoidal model. This uses the eccentricity of the body along with its semi-major axis to create a surface that better represents the body's actual shape. From here, digital elevation models (DEMs) are utilized in an attempt to represent the actual terrain of the body. These DEMs are based off either a reference sphere or ellipsoid and provide elevations of the surface with respect to these reference shapes. The fidelity of the DEM can also vary and, here, includes global models of $^{1}/_{4}{}^{\circ}$, $^{1}/_{16}{}^{\circ}$, $^{1}/_{64}{}^{\circ}$, and $^{1}/_{128}{}^{\circ}$, where these values correspond to the spacing between each point in the DEM. This is where the trade off between accuracy and computational efficiency is once again found. While the $^{1}/_{128}{}^{\circ}$ DEM is significantly more accurate than the $^{1}/_{4}{}^{\circ}$ DEM, it also requires many times the memory and computations in order to locate the proper elevation of the terrain below the spacecraft. Understanding what level of fidelity the filter requires to precisely estimate its state will lead to a filter that is reliable, while maintaining efficiency in how it determines its state estimate.

1.1.2. Filtering Solutions. A navigation filter is used to estimate the state of the vehicle at each time step. In order for the filter to estimate the state between sensor measurement updates, the state is propagated from one time step to the next. The propagation is then used to update the state in conjunction with the new measurements provided from the sensors. This is often done in the minimum mean square error paradigm, which carries the uncertainties of the mean from propagation to update using a matrix of covariances. Traditionally this is performed using a Kalman filter [14] to propagate and update a linear system. Given that the problem of EDL is very much a nonlinear system, this filter must be adapted, and as such, the extended Kalman filter (EKF) [8] is used to linearize the propagation between measurement updates. This problem is complicated even further when the attitude of the spacecraft is taken into consideration. To estimate the attitude, a quaternion representation is employed, which is a non-additive part of the state that the filter must now propagate and update. To overcome this issue, a multiplicative EKF (MEKF) [5] is used and allows a multiplicative update to be performed on the attitude quaternion state. Beyond the problem of propagation and updating the complicated state produced by an EDL scenario, the filter must also account for numerical factors that could cause the filter to diverge or, in worst case scenarios, force the filter to fail and stop working completely. As a preventative measure to these issues, a UDU factorized [3] or cholesky square-root (SR) factorized [32] filter can be implemented. These forms of factorization are then applied to the covariance to create a more robust filter that overcomes the problems often created by numerical issues.

As the filter continues to process measurements, it becomes more confident in the projection of te state into the measurement and reduces the assosiated state-to-measurement uncertainty. If the following measurement then changes drastically from what was expected, the uncertainty of the more confident state-to-measurement mapping may not accurately represent the quality of that solution. As these measurements become more erratic in how they are formed, it can become difficult for the filter to properly approximate a solution that resembles the true sensor solution. This can be seen in the case of filtering altimeter measurements. As the fidelity of the model increases, so too do the number of complexities and variations for which the filter has to account. While the lesser fidelities, such as the spherical or $^1/_4{}^\circ$ DEM models, have a relative smoothness to them that the filter can maintain converging to a proper solution, as these fidelities progress up to the $^1/_{128}{}^\circ$ DEM model, the corresponding elevation changes come so rapidly that the filter finds it difficult to determine an accurate solution. One way of overcoming these possible issues is to use nonlinear transformations that may be able to better account for these rapid variations. Allowing the filter to model its updates in these cases with a nonlinear method, such as an unscented transform (UT) [30] or Monte Carlo sampling [11], the state and its covariance can be better approximated, even with increased surface variations.

1.2. OVERVIEW

For an EDL trajectory, a SR-MEKF is utilized with an inertial measurement unit (IMU), a star camera, and an altimeter as the sensors to estimate the vehicle's state and covariance at each time step. For the altimeter, different fidelities are utilized within the filter to determine how performance is affected as terrain complexity is increased. Testing for this also uses different fidelities within different components of the filter. These include the truth fidelity, which represents the sensor measurement, the estimate of the truth, and the Jacobian of the measurement model. All three of these components can each use a different fidelity model, where the truth has the highest fidelity to best represent the true measurement, allowing the estimate and its Jacobian to be lower fidelities. This variation of fidelities allows the filter to be tested to see how its ability to converge on a solution is affected when lower fidelity, more computationally efficient, estimates are used. This will determine the most reliable and computationally efficient filter for use in this scenario.

As higher fidelity models are used, the increasingly erratic nature of the terrain can cause the corresponding filter solution to be less accurate or even diverge. To maintain stability of the filter solution, tuning noise can be introduced as a way of making the filter less certain prior to each update of its state estimate ad covariance. This erratic terrain can also be counteracted as the components of the filter are changed, such that a lower fidelity estimate is used to compare to that of the truth, creating a smoother surface to estimate upon. To use the more accurate, higher fidelity estimates without tuning noise, a large reduction to the uncertainty can be implemented. Providing the filter with more confidence in its state as soon as the sensor is engaged can create a filter that would more easily maintain this confidence. An easy way to test this is by activating the altimeter sensor earlier in the descent trajectory. The corresponding reduction in the uncertainty can make it easier for the filter to quickly converge on a solution and then, with a more confident solution, better account for the subsequent rapid variations in terrain.

While uncertainty reduction can work with moderate success, it is also of interest to determine how nonlinear transformations of statistics may benefit state estimates. As mentioned before, the more erratic nature of the terrain makes having a nonlinear transformation desirable for handling such changes. To do this a nonlinear transformation is used to approximate the altimeter measurement statistics, such that the filter may be able to better utilize these more complex DEMs. The implementation of a nonlinear transformation can come in many forms, but for the purposes here will be shown with both an unscented transform (UT) and Monte Carlo sampling. The UT provides an example of how these errors can be reduced in a method that can work in real time scenarios. Meanwhile, the Monte Carlo sampling provides a look into how the filter could perform under a “best case” scenario, where computation time is not of consequence.

This study will work to define the filter and models used in Chapter 2 and Chapter 3. With the SR-MEKF and altimeter models described, simulations are detailed in Chapter 4. A discussion of how the filter performs given components of the same fidelities, and then components of different fidelities is given. Having determined how the filter performs under these conditions, alterations to this filtering method are discussed in Chapter 5, which describes how uncertainty can be reduced in simulation, and how the unscented transform and Monte Carlo sampling are implemented. Simulations for these alternative methods are performed and discussed in Chapter 6, followed by a conclusion to this work in Chapter 7.

2. FILTERING

The Kalman filter is known to be used for state estimation of a linear system [14]. In its use here, however, the filter must be adapted to fit the nonlinear scenario of lunar descent navigation. This is the case with many engineering applications of the Kalman filter, which typically leads to the use of linearization. Thus, the extended Kalman filter (EKF) was developed to linearize about the mean using the dynamics of the system and then update using new measurements [8]. This makes the EKF a common choice when filtering of a nonlinear system is required. The construction of an EKF will be discussed, along with the added alterations used in this study, that will eventually constitute the final filter, a square-root multiplicative extended Kalman filter (SR-MEKF).

2.1. EXTENDED KALMAN FILTER

To formulate the EKF, first consider the nonlinear dynamical system, which is taken to be

$$\boldsymbol{x}_k = \boldsymbol{f}(\boldsymbol{x}_{k-1}) + \boldsymbol{M}_{k-1}\boldsymbol{w}_{k-1}\,, \tag{2.1}$$

where $\boldsymbol{x}_k$ is the state at t_k, $\boldsymbol{f}(\cdot)$ represent the nonlinear dynamics of the system, $\boldsymbol{x}_{k-1}$ is the state at t_{k-1}, and $\boldsymbol{M}_{k-1}$ is the mapping matrix for the process noise, which maps the zero-mean white noise given by $\boldsymbol{w}_{k-1}$ into the dynamics.

The nonlinear measurement model is then taken of a similar form

$$\boldsymbol{z}_k = \boldsymbol{h}(\boldsymbol{x}_k) + \boldsymbol{v}_k \tag{2.2}$$

where $\boldsymbol{z}_k$ is the measurement given at time t_k via the measurement model $\boldsymbol{h}(\cdot)$ evaluated at the state $\boldsymbol{x}_k$. The measurement noise is given by $\boldsymbol{v}_k$, which is assumed to be a zero-mean white noise sequence with covariance $\boldsymbol{R}_k$.

2.1.1. Mean and Covariance Propagation. To determine a method for mean propagation the expected value, $E\{\cdot\}$, of Eq. (2.1) is taken as

$$\begin{aligned} E\{\boldsymbol{x}_{\boldsymbol{k}}\} &= E\{\boldsymbol{f}(\boldsymbol{x}_{k-1}) + \boldsymbol{M}_{k-1}\boldsymbol{w}_{k-1}\} \\ &= E\{\boldsymbol{f}(\boldsymbol{x}_{k-1})\} + E\{\boldsymbol{M}_{k-1}\boldsymbol{w}_{k-1}\}\,. \end{aligned} \tag{2.3}$$

When noting that the $\boldsymbol{M}_{k-1}$ mapping matrix is deterministic and that the expected value of $E\{\boldsymbol{x}_{\boldsymbol{k}}\} = \boldsymbol{m}_k$, it can be said that using Eq. (2.3)

$$\boldsymbol{m}_k = E\{\boldsymbol{f}(\boldsymbol{x}_{k-1})\} + \boldsymbol{M}_{k-1}E\{\boldsymbol{w}_{k-1}\}\,. \tag{2.4}$$

To find the expectation of $\boldsymbol{f}(\boldsymbol{x}_{k-1})$ a first-order Taylor Series is expanded about the current mean

$$\boldsymbol{f}(\boldsymbol{x}_{k-1}) \simeq \boldsymbol{f}(\boldsymbol{m}_{k-1}) + \boldsymbol{F}(\boldsymbol{m}_{k-1})(\boldsymbol{x}_{k-1} - \boldsymbol{m}_{k-1}), \tag{2.5}$$

where $\boldsymbol{F}(\boldsymbol{m}_{k-1})$ is the Jacobian of the dynamics given by

$$\boldsymbol{F}(\boldsymbol{m}_{k-1}) = \left[\left. \frac{\partial \boldsymbol{f}(\boldsymbol{x}_{k-1})}{\partial \boldsymbol{x}_{k-1}} \right|_{\boldsymbol{x}_{k-1}=\boldsymbol{m}_{k-1}} \right].$$

Taking Eq. (2.5) and substituting it into Eq. (2.4) this becomes

$$\boldsymbol{m}_k = E\{\boldsymbol{f}(\boldsymbol{m}_{k-1}) + \boldsymbol{F}(\boldsymbol{m}_{k-1})(\boldsymbol{x}_{k-1} - \boldsymbol{m}_{k-1})\} + \boldsymbol{M}_{k-1}E\{\boldsymbol{w}_{k-1}\}$$

and assuming that $\boldsymbol{f}(\cdot)$, $\boldsymbol{F}(\cdot)$, and $\boldsymbol{m}_{k-1}$ are also deterministic this becomes

$$\boldsymbol{m}_k = \boldsymbol{f}(\boldsymbol{m}_{k-1}) + \boldsymbol{F}(\boldsymbol{m}_{k-1})E\{(\boldsymbol{x}_{k-1} - \boldsymbol{m}_{k-1})\} + \boldsymbol{M}_{k-1}E\{\boldsymbol{w}_{k-1}\}\,. \tag{2.6}$$

If the estimation error is defined as

$$\boldsymbol{e}_{k-1} = \boldsymbol{x}_{k-1} - \boldsymbol{m}_{k-1}$$

which can then be substituted into Eq. (2.6) to get

$$\boldsymbol{m}_k = \boldsymbol{f}(\boldsymbol{m}_{k-1}) + \boldsymbol{F}(\boldsymbol{m}_{k-1})E\{\boldsymbol{e}_{k-1}\} + \boldsymbol{M}_{k-1}E\{\boldsymbol{w}_{k-1}\}\,. \tag{2.7}$$

Since $\boldsymbol{w}_{k-1}$ is a zero-mean noise, and $\boldsymbol{e}_{k-1}$ is constructed to be zero-mean for an unbiased *a priori* estimate, the expected value of these processes is zero. This is done such that an unbiased *a posteriori* estimate can be determined, as is intended. When using this in Eq. (2.7) the mean as it evolves over time is finally determined as

$$\boldsymbol{m}_k = \boldsymbol{f}(\boldsymbol{m}_{k-1})\,. \tag{2.8}$$

With the mean propagation now defined the state estimation error covariance needs to be determined. This covariance, $\boldsymbol{P}_k$, is defined as

$$\boldsymbol{P}_k = E\{\boldsymbol{e}_k\boldsymbol{e}_k^T\}, \tag{2.9}$$

where the state estimation error is

$$\boldsymbol{e}_k = \boldsymbol{x}_k - \boldsymbol{m}_k \tag{2.10}$$

and recalling from Eqs. (2.1), (2.8), and (2.5) the estimation error corresponding to the state as it evolves over time can be found by substituting into Eq. (2.10) as

$$\begin{aligned}\boldsymbol{e}_k &= [\boldsymbol{f}(\boldsymbol{x}_{k-1}) + \boldsymbol{M}_{k-1}\boldsymbol{w}_{k-1}] - \boldsymbol{f}(\boldsymbol{m}_{k-1}) \\ &= \boldsymbol{f}(\boldsymbol{m}_{k-1}) + \boldsymbol{F}(\boldsymbol{m}_{k-1})(\boldsymbol{x}_{k-1} - \boldsymbol{m}_{k-1}) + \boldsymbol{M}_{k-1}\boldsymbol{w}_{k-1} - \boldsymbol{f}(\boldsymbol{m}_{k-1}) \\ &= \boldsymbol{F}(\boldsymbol{m}_{k-1})\boldsymbol{e}_{k-1} + \boldsymbol{M}_{k-1}\boldsymbol{w}_{k-1}\,.\end{aligned}$$

This final equation for the state estimation error can now be used in Eq. (2.9) where

$$\begin{aligned}\boldsymbol{e}_k\boldsymbol{e}_k^T &= [\boldsymbol{F}(\boldsymbol{m}_{k-1})\boldsymbol{e}_{k-1} + \boldsymbol{M}_{k-1}\boldsymbol{w}_{k-1}][\boldsymbol{F}(\boldsymbol{m}_{k-1})\boldsymbol{e}_{k-1} + \boldsymbol{M}_{k-1}\boldsymbol{w}_{k-1}]^T \\ &= \boldsymbol{F}(\boldsymbol{m}_{k-1})\boldsymbol{e}_{k-1}\boldsymbol{e}_{k-1}^T\boldsymbol{F}(\boldsymbol{m}_{k-1})^T + \boldsymbol{M}_{k-1}\boldsymbol{w}_{k-1}\boldsymbol{e}_{k-1}^T\boldsymbol{F}(\boldsymbol{m}_{k-1})^T \\ &\quad + \boldsymbol{F}(\boldsymbol{m}_{k-1})\boldsymbol{e}_{k-1}\boldsymbol{w}_{k-1}^T\boldsymbol{M}_{k-1}^T + \boldsymbol{M}_{k-1}\boldsymbol{w}_{k-1}\boldsymbol{w}_{k-1}^T\boldsymbol{M}_{k-1}^T\,,\end{aligned}$$

When taking the expectation of this, $\boldsymbol{F}(\cdot)$, $\boldsymbol{M}(\cdot)$, and $\boldsymbol{m}_{k-1}$ are assumed to be deterministic such that

$$\begin{aligned}E\{\boldsymbol{e}_k\boldsymbol{e}_k^T\} = \boldsymbol{F}(\boldsymbol{m}_{k-1})E\{\boldsymbol{e}_{k-1}\boldsymbol{e}_{k-1}^T\}\boldsymbol{F}(\boldsymbol{m}_{k-1})^T + \boldsymbol{M}_{k-1}E\{\boldsymbol{w}_{k-1}\boldsymbol{e}_{k-1}^T\}\boldsymbol{F}(\boldsymbol{m}_{k-1})^T \\ +\boldsymbol{F}(\boldsymbol{m}_{k-1})E\{\boldsymbol{e}_{k-1}\boldsymbol{w}_{k-1}^T\}\boldsymbol{M}_{k-1}^T + \boldsymbol{M}_{k-1}E\{\boldsymbol{w}_{k-1}\boldsymbol{w}_{k-1}^T\}\boldsymbol{M}_{k-1}^T\end{aligned}$$

sothe remaining expectations presented here include the uncorrelated noise, which are zero-mean and therefore go to zero, the definition of the *a priori* covariance, $\boldsymbol{P}_{k-1}$, and the *a priori* power spectral density, $\boldsymbol{Q}_{k-1}$, where

$$\begin{aligned}E\{\boldsymbol{w}_{k-1}\boldsymbol{e}_{k-1}^T\} &= E\{\boldsymbol{e}_{k-1}\boldsymbol{w}_{k-1}^T\} = 0 \\ E\{\boldsymbol{e}_{k-1}\boldsymbol{e}_{k-1}^T\} &= \boldsymbol{P}_{k-1} \\ E\{\boldsymbol{w}_{k-1}\boldsymbol{w}_{k-1}^T\} &= \boldsymbol{Q}_{k-1}\,,\end{aligned}$$

which finally produces the evolution of the state covariance at each time step

$$\boldsymbol{P}_k = \boldsymbol{F}(\boldsymbol{m}_{k-1})\boldsymbol{P}_{k-1}\boldsymbol{F}(\boldsymbol{m}_{k-1})^T + \boldsymbol{M}_{k-1}\boldsymbol{Q}_{k-1}\boldsymbol{M}_{k-1}^T\,. \tag{2.11}$$

2.1.2. Mean and Covariance Update. Once the state has been propagated along with its respective estimation error covariance the estimated state and covariance need to be updated using new measurement data. To do this it is assumed an *a posteriori* mean is produced via the inclusion of new measurement information. This *a posteriori* mean, $\boldsymbol{m}_k^+$, is defined as

$$\boldsymbol{m}_k^+ = \boldsymbol{a}_k + \boldsymbol{K}_k \boldsymbol{z}_k \tag{2.12}$$

where $\boldsymbol{a}_k$ is a constant vector that will be shown to encapsulate the *a priori* mean, $\boldsymbol{K}_k$ is a linear gain, which will be further developed later on, and $\boldsymbol{z}_k$ is the new measurement defined in Eq. (2.2). Recalling Eq. (2.10) and defining it for both the *a priori* and *a posteriori* errors as

$$\boldsymbol{e}_k^- = \boldsymbol{x}_k - \boldsymbol{m}_k^- \tag{2.13}$$

$$\boldsymbol{e}_k^+ = \boldsymbol{x}_k - \boldsymbol{m}_k^+ , \tag{2.14}$$

which are then solved for the *a priori* and *a posteriori* mean so the *a priori* mean can be solved as

$$\boldsymbol{m}_k^- = \boldsymbol{m}_k^+ + \boldsymbol{e}_k^- - \boldsymbol{e}_k^+ . \tag{2.15}$$

When the errors are once again constructed to be unbiased and zero mean it is found that using Eq. (2.12) the expectation of the *a priori* is

$$
\begin{aligned}
\boldsymbol{m}_k^- + \boldsymbol{e}_k^- - \boldsymbol{e}_k^+ &= \boldsymbol{a}_k + \boldsymbol{K}_k \boldsymbol{z}_k \\
E\{\boldsymbol{m}_k^- + \boldsymbol{e}_k^- - \boldsymbol{e}_k^+\} &= E\{\boldsymbol{a}_k + \boldsymbol{K}_k \boldsymbol{z}_k\} \\
\boldsymbol{m}_k^- &= \boldsymbol{a}_k + \boldsymbol{K}_k \hat{\boldsymbol{z}}_k
\end{aligned}
$$

where $\hat{z}_k$ is the estimated measurement, which is given by $E\{z_k\}$. This is then solved for $\boldsymbol{a}_k$ where

$$\boldsymbol{a}_k = \boldsymbol{m}_k^- - \boldsymbol{K}_k \hat{\boldsymbol{z}}_k ,$$

and substitute it back into Eq. (2.12) to determine the updated mean as

$$\boldsymbol{m}_k^+ = \boldsymbol{m}_k^- + \boldsymbol{K}_k (\boldsymbol{z}_k - \hat{\boldsymbol{z}}_k) . \tag{2.16}$$

Following from this, the *a posteriori* error covariance needs to be defined for the update. This is done with the *a posteriori* covariance expected from the error as

$$\boldsymbol{P}_k^+ = E\{\boldsymbol{e}_k^+ (\boldsymbol{e}_k^+)^T\} \tag{2.17}$$

where the *a posteriori* error is found using Eq. (2.16) to substitute into Eq. (2.15) as being

$$\boldsymbol{m}_k^- = \boldsymbol{m}_k^- + \boldsymbol{K}_k (\boldsymbol{z}_k - \hat{\boldsymbol{z}}_k) + \boldsymbol{e}_k^- - \boldsymbol{e}_k^+$$

where the *a posteriori* error is solved for as

$$\boldsymbol{e}_k^+ = \boldsymbol{e}_k^- - \boldsymbol{K}_k (\boldsymbol{z}_k - \hat{\boldsymbol{z}}_k) . \tag{2.18}$$

Using Eq. (2.13) to substitute in for $\boldsymbol{e}_k^-$ in Eq. (2.18) the *a posteriori* covariance is found as

$$\begin{aligned}\boldsymbol{P}_k^+ &= E\{(\boldsymbol{x}_k - \boldsymbol{m}_k^-)(\boldsymbol{x}_k - \boldsymbol{m}_k^-)^T - (\boldsymbol{x}_k - \boldsymbol{m}_k^-)(z_k - \hat{z}_k)^T \boldsymbol{K}_k^T \\ &\quad - \boldsymbol{K}_k(z_k - \hat{z}_k)(\boldsymbol{x}_k - \boldsymbol{m}_k^-)^T + \boldsymbol{K}_k(z_k - \hat{z}_k)(z_k - \hat{z}_k)^T \boldsymbol{K}_k^T\}\end{aligned}$$

where, when the gain, $\boldsymbol{K}_k$, which is assumed to be deterministic, is pulled out of the equation, it is then found to be

$$\begin{aligned}\boldsymbol{P}_k^+ &= E\{(\boldsymbol{x}_k - \boldsymbol{m}_k^-)(\boldsymbol{x}_k - \boldsymbol{m}_k^-)^T\} - E\{(\boldsymbol{x}_k - \boldsymbol{m}_k^-)(z_k - \hat{z}_k)^T\} \boldsymbol{K}_k^T \\ &\quad - \boldsymbol{K}_k E\{(z_k - \hat{z}_k)(\boldsymbol{x}_k - \boldsymbol{m}_k^-)^T\} + \boldsymbol{K}_k E\{(z_k - \hat{z}_k)(z_k - \hat{z}_k)^T\} \boldsymbol{K}_k^T \,.\end{aligned}$$

To complete the construction of the *a posteriori* covariance, its expectations are shown as the *a priori* covariance, $\boldsymbol{P}_k^-$, that has been previously defined, the cross covariance, $\boldsymbol{C}_k$, and measurement covariance, $\boldsymbol{W}_k$, which correlate to the expect values as

$$\boldsymbol{P}_k^- = E\{(\boldsymbol{x}_k - \boldsymbol{m}_k^-)(\boldsymbol{x}_k - \boldsymbol{m}_k^-)^T\}$$

$$\boldsymbol{C}_k = E\{(\boldsymbol{x}_k - \boldsymbol{m}_k^-)(z_k - \hat{z}_k)^T\} \tag{2.19}$$

$$\boldsymbol{W}_k = E\{(z_k - \hat{z}_k)(z_k - \hat{z}_k)^T\} \tag{2.20}$$

so that the *a posteriori* covariance is found as [35]

$$\boldsymbol{P}_k^+ = \boldsymbol{P}_k^- - \boldsymbol{C}_k \boldsymbol{K}_k^T - \boldsymbol{K}_k \boldsymbol{C}_k^T + \boldsymbol{K}_k \boldsymbol{W}_k \boldsymbol{K}_k^T \,. \tag{2.21}$$

From here, the gain, $\boldsymbol{K}_k$, is found such that the *a posteriori* mean square state estimation error is minimized; the mean square error (MSE) performance index [8] is given by

$$\begin{aligned} J &= E\{(\boldsymbol{e}_k^+)^T \boldsymbol{e}_k^+\} = \text{trace} E\{\boldsymbol{e}_k^+ (\boldsymbol{e}_k^+)^T\} = \text{trace} \boldsymbol{P}_k^+ \\ &= \text{trace}\{\boldsymbol{P}_k^-\} - \text{trace}\{\boldsymbol{C}_k \boldsymbol{K_k^T}\} - \text{trace}\{\boldsymbol{K}_k \boldsymbol{C}_k^T\} + \text{trace}\{\boldsymbol{K}_k \boldsymbol{W}_k \boldsymbol{K}_k^T\} \\ &= \text{trace}\{\boldsymbol{P}_k^-\} - 2\ \text{trace}\{\boldsymbol{K}_k \boldsymbol{C}_k^T\} + \text{trace}\{\boldsymbol{K}_k \boldsymbol{W}_k \boldsymbol{K}_k^T\}\,. \end{aligned} \tag{2.22}$$

Now to find the minimum of this function the derivative needs to be taken [8] where

$$\frac{\partial}{\partial \boldsymbol{K}_k} \text{trace}\{\boldsymbol{K}_k \boldsymbol{C}_k^T\} = \boldsymbol{C}_k$$
$$\frac{\partial}{\partial \boldsymbol{K}_k} \text{trace}\{\boldsymbol{K}_k \boldsymbol{W}_k \boldsymbol{K}_k^T\} = \boldsymbol{K}_k [\boldsymbol{W}_k + \boldsymbol{W}_k^T]$$

so that the derivative of Eq. 2.22 is

$$\frac{\partial J}{\partial \boldsymbol{K}_k} = -2\boldsymbol{C}_k + 2\boldsymbol{K}_k \boldsymbol{W}_k$$

where the measurement covariance, $\boldsymbol{W}_k$, is symmetric, such that it can be simply said that $\boldsymbol{W}_k + \boldsymbol{W}_k^T = 2\boldsymbol{W}_k$. The gain that minimizes the performance index is found by rendering it stationary and solving for the gain such that

$$\frac{\partial J}{\partial \boldsymbol{K}_k} = -2\boldsymbol{C}_k + 2\boldsymbol{K}_k \boldsymbol{W}_k = 0 \tag{2.23}$$
$$\boldsymbol{K}_k = \boldsymbol{C}_k \boldsymbol{W}_k^{-1}\,. \tag{2.24}$$

providing the Kalman gain.

Looking at the measurement, $\boldsymbol{z}_k$, the expected value is denoted by

$$E\{\boldsymbol{z}_k\} = \hat{\boldsymbol{z}}\,. \tag{2.25}$$

It can be solved for the measurement estimate, $\hat{\boldsymbol{z}}$, from the expectation of Eq. 2.2 as

$$\hat{z} = E\{\boldsymbol{h}(\boldsymbol{x}_k) + \boldsymbol{v}_k\}\,. \tag{2.26}$$

Following the same approach used for the dynamics, $\boldsymbol{h}(\boldsymbol{x}_k)$ is expanded in a first-order Taylor Series about the *a priori* state estimate, such that

$$\boldsymbol{h}(\boldsymbol{x}_k) \simeq \boldsymbol{h}(\boldsymbol{m}_k^-) + \boldsymbol{H}(\boldsymbol{m}_k^-)(\boldsymbol{x}_k - \boldsymbol{m}_k^-), \tag{2.27}$$

where $\boldsymbol{H}(\boldsymbol{m}_k^-)$ is the Jacobian of the measurement, which is

$$\boldsymbol{H}(\boldsymbol{m}_k^-) = \left[\left. \frac{\partial \boldsymbol{h}(\boldsymbol{x}_k)}{\partial \boldsymbol{x}_k} \right|_{\boldsymbol{x}_k = \boldsymbol{m}_k^-} \right].$$

Substituting Eq. 2.27 into Eq. 2.26 the measurement estimate can now be expressed as

$$\begin{aligned}
\hat{z} &= E\{\boldsymbol{h}(\boldsymbol{m}_k^-) + \boldsymbol{H}(\boldsymbol{m}_k^-)(\boldsymbol{x}_k - \boldsymbol{m}_k^-) + \boldsymbol{v}_k\} \\
&= E\{\boldsymbol{h}(\boldsymbol{m}_k^-)\} + E\{\boldsymbol{H}(\boldsymbol{m}_k^-)(\boldsymbol{x}_k - \boldsymbol{m}_k^-)\} + E\{\boldsymbol{v}_k\} \\
&= E\{\boldsymbol{h}(\boldsymbol{m}_k^-)\} + E\{\boldsymbol{H}(\boldsymbol{m}_k^-)\boldsymbol{e}_k^-\} + E\{\boldsymbol{v}_k\}
\end{aligned}$$

Assuming that $\boldsymbol{h}(\cdot)$, $\boldsymbol{H}(\cdot)$, and $\boldsymbol{m}_k$ are deterministic and that the *a priori* error, $\boldsymbol{e}_k^-$, and the measurement noise, $\boldsymbol{v}_k$, are constructed as zero-mean, the estimated measurement, which is taken to be the expected measurement, is found to be

$$\hat{z} = \boldsymbol{h}(\boldsymbol{m}_k^-)\,.$$

To complete the update, the cross covariance and residual covariance need to be computed in order to find the Kalman gain and the *a posteriori* mean and covariance. Starting with the cross covariance given in Eq. 2.19, the second term of the expectation is evaluated with Eq. 2.27 being substituted into Eq. 2.2 with the measurement estimate in

Eq. 2.25 subtracted from it such that

$$\begin{aligned}
z_k - \hat{z}_k &= \boldsymbol{h}_k(\boldsymbol{m}_k^-) + \boldsymbol{H}(\boldsymbol{m}_k^-)(\boldsymbol{x}_k - \boldsymbol{m}_k^-) + \boldsymbol{L}_k\boldsymbol{v}_k - \boldsymbol{h}_k(\boldsymbol{m}_k^-) \\
&= \boldsymbol{H}(\boldsymbol{m}_k^-)(\boldsymbol{x}_k - \boldsymbol{m}_k^-) + \boldsymbol{L}_k\boldsymbol{v}_k \qquad (2.28) \\
&= \boldsymbol{H}(\boldsymbol{m}_k^-)(\boldsymbol{e}_k^-) + \boldsymbol{L}_k\boldsymbol{v}_k \, . \qquad (2.29)
\end{aligned}$$

This is then substituted into Eq. 2.19 as

$$\begin{aligned}
\boldsymbol{C}_k &= E\{(\boldsymbol{x}_k - \boldsymbol{m}_k^-)(\boldsymbol{x}_k - \boldsymbol{m}_k^-)^T\}\boldsymbol{H}(\boldsymbol{m}_k^-)^T + E\{(\boldsymbol{x}_k - \boldsymbol{m}_k^-)\boldsymbol{v}_k^T\}\boldsymbol{L}_k^T \\
&= E\{\boldsymbol{e}_k^-(\boldsymbol{e}_k^-)^T\}\boldsymbol{H}(\boldsymbol{m}_k^-)^T + E\{\boldsymbol{e}_k^-\boldsymbol{v}_k^T\}\boldsymbol{L}_k^T
\end{aligned}$$

where $\boldsymbol{H}(\boldsymbol{m}_k^-)$ and $\boldsymbol{L}_k$ are outside of the expectations since they are assumed to be deterministic. With the assumption that the state is uncorrelated to the measurement noise, $\boldsymbol{v}_k$, then becomes zero and leaves the cross covariance as

$$\boldsymbol{C}_k = \boldsymbol{P}_k^-\boldsymbol{H}(\boldsymbol{m}_k^-)^T \, . \qquad (2.30)$$

That leaves the covariance of the measurement as the last to find by multiplying Eq. 2.29 by its transpose as in Eq. 2.20. This gives the covariance as

$$\begin{aligned}
\boldsymbol{W}_k &= (\boldsymbol{H}_k E\{\boldsymbol{e}_k^-\} + \boldsymbol{L}_k E\{\boldsymbol{v}_k\})(\boldsymbol{H}_k E\{\boldsymbol{e}_k^-\} + \boldsymbol{L}_k E\{\boldsymbol{v}_k\})^T \\
&= \boldsymbol{H}_k E\{\boldsymbol{e}_k^-\boldsymbol{e}_k^{-T}\}\boldsymbol{H}_k^T + \boldsymbol{L}_k E\{\boldsymbol{v}_k\boldsymbol{v}_k^T\}\boldsymbol{L}_k^T \\
&= \boldsymbol{H}_k\boldsymbol{P}_k^-\boldsymbol{H}_k^T + \boldsymbol{L}_k\boldsymbol{R}_k\boldsymbol{L}_k^T , \qquad (2.31)
\end{aligned}$$

where, once again, $\boldsymbol{H}(\boldsymbol{m}_k^-)$ and $\boldsymbol{L}_k$ are assumed to be deterministic and fall outside the expected value and where the covariance of the measurement noise, $\boldsymbol{v}_k$, is given as $\boldsymbol{R}_k$. This gives the last definition needed to calculate the *a posteriori* state covariance. Using Eqs.

2.24, 2.30, and 2.31 the covariance in Eq. 2.21 is found, such that the EKF update is now completed.

2.2. MULTIPLICATIVE EKF

The EKF described is designed using a three parameter attitude, but for the filtering solution in use, a quaternion measurement is used. In order to use this within the EKF framework the Multiplicative EKF (MEKF) [5] needs to be implemented. This is done so that the attitude error can be quantified as a three-parameter attitude, while the state uses a quaternion measurement. This approximation is due to the benefit of a small angle assumption in the errors. Since the EKF as it has been defined thus far is updated where the estimated mean is added to the *a priori* mean the quaternion attitude could not be utilized. This is because the quaternion attitude used for the update does not reside in vector space, so it is not additive and can not merely be added to find a valid quaternion as is done in the update for the remaining state parameters. In order to make use of this quaternion attitude update the filter needs to be adapted to the MEKF that will allow the filter to make use of these measurements. Doing this then allows for a filter that can utilize the advantages of a quaternion attitude.

This is first done by defining the quaternion update as

$$\hat{\bar{\boldsymbol{q}}}_k^+ = \delta\bar{\boldsymbol{q}}(\Delta\boldsymbol{\theta}) \otimes \hat{\bar{\boldsymbol{q}}}_k^-$$

so that $\hat{\bar{\boldsymbol{q}}}_k^+$ is the updated quaternion estimate, $\delta\bar{\boldsymbol{q}}(\Delta\theta)$ is the conversion of the 3-parameter attitude to a quaternion, and $\hat{\bar{\boldsymbol{q}}}_k^-$ is the *a priori* attitude estimate. For the MEKF, the update is done in two steps shown as

$$\boldsymbol{m}_{k,1}^{+} = \boldsymbol{m}_{k}^{-} + \boldsymbol{K}_{k}(z - \hat{z}) \tag{2.32}$$

$$\boldsymbol{m}_{k,2}^{+} = \begin{bmatrix} \frac{1}{2}\Delta\hat{\boldsymbol{\theta}}_{k} \\ 1 \end{bmatrix} \otimes \hat{\bar{\boldsymbol{q}}}_{k}^{-} \tag{2.33}$$

where first the additive update, $\boldsymbol{m}_{k,1}^{+}$, is done and then the quaternion update, $\boldsymbol{m}_{k,2}^{+}$ is completed to find the updated state. This quaternion update is then normalized via

$$\hat{\bar{\boldsymbol{q}}}_{k}^{+} = \frac{\hat{\bar{\boldsymbol{q}}}_{k}^{+}}{||\hat{\bar{\boldsymbol{q}}}_{k}^{+}||}$$

where Eq. 2.33 is then brute-force normalized due to the utilized small angles assumption. This update is then show where Eqs. 2.32 and 2.33 are put together as

$$\boldsymbol{m}_{k}^{+} = \begin{bmatrix} \boldsymbol{m}_{k,1}^{+} \\ \boldsymbol{m}_{k,2}^{+} \end{bmatrix}$$

so that both the additive and quaternion sections of the mean get updated to the *a posteriori* mean.

2.3. FILTERING ALTERATIONS

While the filter as described works in filtering the data in an effort to produce an estimate that converges to the true solution, it is not immune to failure. Several anomalies can cause the filter to fail or stop working in some fashion. Problems such as a negative variance value or large covariance growth can cause divergence of the filter or, in the worst cases, stop the filter completely. To preemptively prevent these problems some alterations to the filter can be made. These alterations work to make a more robust and reliable filter that will continue to operate and provide state estimates even in challenging environments.

2.3.1. Square-Root MEKF. The covariance matrix has strict attributes of symmetry and positive definiteness that must be maintained in order for the filter to continue processing measurements. While symmetry can be easily maintained through brute-force symmetrization, maintaining a positive covariance is more difficult. To counteract this issue, an adaptation to the filter to work with square-root factors of covariance matrices is implemented. While this can also be done with UDU for upper triangular factorization [3], here square-root factorization is used within the filter. This uses the Cholesky square-root factor [21, 32] to factorize the covariance of the state to add a layer to numerical reliability. This development is referred to as the square-root MEKF (SR-MEKF) and will be defined for use here, greatly increasing the robustness compared to the MEKF.

To define a method for the filter to work with square-root factors of the covariance matrix it is first noted that given some covariance matrix $\boldsymbol{A}$, the square root factors can be found as

$$\boldsymbol{A} = \boldsymbol{S}\boldsymbol{S}^T ,$$

so that $\boldsymbol{S}$ represents the square root factor of $\boldsymbol{A}$. Applying this to the *a priori* covariance it can be said that

$$\begin{aligned}
\boldsymbol{P}_k^- &= \boldsymbol{S}_k^-(\boldsymbol{S}_k^-)^T \\
\boldsymbol{P}_{k-1}^+ &= \boldsymbol{S}_{k-1}^+(\boldsymbol{S}_{k-1}^+)^T \\
\boldsymbol{N}_{k-1} &= \boldsymbol{T}_{k-1}(\boldsymbol{T}_{k-1})^T ,
\end{aligned}$$

where $\boldsymbol{P}_k^-$ is the *a priori* covariance at time t_k with $\boldsymbol{S}_k^-$ as its square-root factor, $\boldsymbol{P}_{k-1}^+$ is the *a posteriori* covariance at time t_{k-1} with $\boldsymbol{S}_{k-1}^+$ as its square-root factor, and $\boldsymbol{N}_{k-1}$ is the process noise covariance, formally noted in Eq. 2.21 as $\boldsymbol{Q}_{k-1}$, with $\boldsymbol{T}_{k-1}$ as its square-root factor. This gives the *a priori* covariance defined with square-root factors by substituting the covariances in Eq. 2.11 with there square-root factor representations such that

$$\boldsymbol{P}_k^- = \boldsymbol{S}_k^- \boldsymbol{S}_k^{-T} = \boldsymbol{F}(\boldsymbol{m}_{k-1}^+)\boldsymbol{S}_{k-1}^+(\boldsymbol{S}_{k-1}^+)^T(\boldsymbol{F}(\boldsymbol{m}_{k-1}^+))^T + \boldsymbol{M}_{k-1}\boldsymbol{T}_{k-1}\boldsymbol{T}_{k-1}^T\boldsymbol{M}_{k-1}^T ,$$

which can be re-written in factorized terms as

$$\boldsymbol{P}_k^- = \boldsymbol{S}_k^- \boldsymbol{S}_k^- = [\boldsymbol{F}(\boldsymbol{m}_{k-1}^+)\boldsymbol{S}_{k-1}^T | \boldsymbol{M}_{k-1}\boldsymbol{T}_{k-1}][\boldsymbol{F}(\boldsymbol{m}_{k-1}^+)\boldsymbol{S}_{k-1}^T | \boldsymbol{M}_{k-1}\boldsymbol{T}_{k-1}]^T$$

so that

$$\boldsymbol{S}_k^- = [\boldsymbol{F}(\boldsymbol{m}_{k-1}^+)\boldsymbol{S}_{k-1}^+ | \boldsymbol{M}_{k-1}\boldsymbol{T}_{k-1}] .$$

While this does give a valid square-root factor, it is still contained within a non-square matrix. In order to acquire a square matrix of the square-root factor the QR-Decomposition method is utilized. Consider the QR-Decomposition of some matrix $\boldsymbol{A}$, which is given by

$$\boldsymbol{A}^T = \boldsymbol{Q}\boldsymbol{R}^T , \tag{2.34}$$

with $\boldsymbol{Q}$ as a unitary matrix and $\boldsymbol{R}$ as a lower triangular matrix, making $\boldsymbol{R}^T$ an upper triangular matrix. Taking the transpose of Eq. 2.34 then allows us to say

$$\boldsymbol{A} = \boldsymbol{R}\boldsymbol{Q}^T$$

so that when $\boldsymbol{A}$ and $\boldsymbol{A}^T$ are multiplied together it produces

$$\boldsymbol{A}\boldsymbol{A}^T = \boldsymbol{R}\boldsymbol{Q}^T\boldsymbol{Q}\boldsymbol{R}^T .$$

Since $\boldsymbol{Q}$ is a unitary matrix

$$\boldsymbol{A}\boldsymbol{A}^T = \boldsymbol{R}\boldsymbol{R}^T$$

meaning that $\boldsymbol{R}$ is a valid square root factor of $\boldsymbol{A}\boldsymbol{A}^T$. Utilizing these concepts in the square-root factorization of the covariance propagation, the *a priori* covariance is determined by using the QR-decomposition method, qr(·), to get

$$\boldsymbol{S}_k^- = \text{qr}\{[\boldsymbol{F}(\boldsymbol{m}_{k-1}^+)\boldsymbol{S}_{k-1}^T|\boldsymbol{M}_{k-1}\boldsymbol{T}_{k-1}]^T\}^T,$$

where in this case the formulation is designed with transposes included such that it works within the MATLAB framework.

For the *a posteriori* covariance the measurement covariance and the Kalman gain need to be developed in such a way that they can also use the square-root factorized covariances. To start this the measurement covariance is defined as

$$\boldsymbol{W}_k = \boldsymbol{H}(\boldsymbol{m}_{k-1}^+)\boldsymbol{P}_k^-\boldsymbol{H}(\boldsymbol{m}_{k-1}^+)^T + \boldsymbol{L}_k\boldsymbol{R}_k\boldsymbol{L}_k^T \tag{2.35}$$

where covariance components are broken down into square-root factors such that

$$\boldsymbol{W}_k = \boldsymbol{S}_{W_k}\boldsymbol{S}_{W_k}^T \tag{2.36}$$

$$\boldsymbol{P}_k^- = \boldsymbol{S}_k^-(\boldsymbol{S}_k^-)^T \tag{2.37}$$

$$\boldsymbol{R}_k = \boldsymbol{S}_{R_k}\boldsymbol{S}_{R_k}^T\,. \tag{2.38}$$

These equations can then be substituted into Eq. 2.35 where the measurement covariance is represented with square-root factors as

$$\boldsymbol{W}_k = \boldsymbol{S}_{W_k}\boldsymbol{S}_{W_k}^T = \boldsymbol{H}(\boldsymbol{m}_{k-1}^+)\boldsymbol{S}_k^-(\boldsymbol{S}_k^-)^T\boldsymbol{H}(\boldsymbol{m}_{k-1}^+)^T + \boldsymbol{L}_k\boldsymbol{S}_{R_k}\boldsymbol{S}_{R_k}^T\boldsymbol{L}_k^T,$$

where QR-decomposition can then be used to find the solution as

$$\begin{aligned}\boldsymbol{S}_W &= [\boldsymbol{H}(\boldsymbol{m}_{k-1}^{+})\boldsymbol{S}_k^{-}|\boldsymbol{L}_k\boldsymbol{S}_R][\boldsymbol{H}(\boldsymbol{m}_{k-1}^{+})\boldsymbol{S}_k^{-}|\boldsymbol{L}_k\boldsymbol{S}_R]^T \\ &= \mathrm{qr}\{[\boldsymbol{H}(\boldsymbol{m}_{k-1}^{+})\boldsymbol{S}_k^{-}|\boldsymbol{L}_k\boldsymbol{S}_R]^T\}^T \,.\end{aligned}$$

With the measurement covariance computed using square-root factors, the Kalman gain now needs to be changed to also use the factorized covariances. To start this it is reminded that the Kalman gain was defined in Eq. 2.24 and is now evaluated where $\boldsymbol{W}_k$ is substituted out for its square-root factors in Eq. 2.36,from this it is seen that the Kalman gain is represented as

$$\boldsymbol{K}_k = \boldsymbol{C}_k\boldsymbol{S}_W^{-T}\boldsymbol{S}_W^{-1} \tag{2.39}$$

so that the measurement covariance is broken into its square-root factorized form. Following that the cross covariance in Eq. 2.30 needs to be evaluated using the square-root factors of the *a priori* covariance in Eq. 2.37 as

$$\boldsymbol{C}_k = \boldsymbol{S}_k^{-}[\boldsymbol{H}_k\boldsymbol{S}_k^{-}]^T \,. \tag{2.40}$$

With these definitions Eq. 2.39 can be broken into two equations as

$$\begin{aligned}\boldsymbol{U}_k &= \boldsymbol{C}_k\boldsymbol{S}_W^{-T} \\ \boldsymbol{K}_k &= \boldsymbol{U}_k\boldsymbol{S}_W^{-1}\end{aligned} \tag{2.41}$$

giving the Kalman gain using square-root factors.

With these defined a square-root factorized *a posteriori* covariance can now be constructed. This is done by substituting in the square-root factors of Eqs. 2.40 and 2.36 into Eq. 2.21 such that

$$\begin{aligned} \boldsymbol{S}_k^+\boldsymbol{S}_k^{+^T} &= \boldsymbol{S}_k^-(\boldsymbol{S}_k^-)^T - \boldsymbol{K}_k\boldsymbol{S}_{W_k}\boldsymbol{S}_{W_k}^T\boldsymbol{K}_k^T - \boldsymbol{K}_k\boldsymbol{S}_{W_k}\boldsymbol{S}_{W_k}^T\boldsymbol{K}_k^T + \boldsymbol{K}_k\boldsymbol{S}_{W_k}\boldsymbol{S}_{W_k}^T\boldsymbol{K}_k^T \\ &= \boldsymbol{S}_k^-(\boldsymbol{S}_k^-)^T - \boldsymbol{K}_k\boldsymbol{S}_W\boldsymbol{S}_W^T\boldsymbol{K}_k^T . \end{aligned}$$

Then Eq. 2.41 can be taken and manipulated so that

$$\boldsymbol{U}_k = \boldsymbol{K}_k\boldsymbol{S}_W$$

which then allows the *a posteriori* covariance to be defined as

$$\boldsymbol{S}_k^+\boldsymbol{S}_k^+ = \boldsymbol{S}_k^-\boldsymbol{S}_k^{-^T} - \boldsymbol{U}_k\boldsymbol{U}_k^T \tag{2.42}$$

where $\boldsymbol{U}_k$ is a $n \times m$ matrix with state dimension n and measurement dimension m. Matrix $\boldsymbol{U}_k$ can be written as

$$\boldsymbol{U}_k = [\boldsymbol{u}_1 \quad \boldsymbol{u}_2 \quad \cdots \quad \boldsymbol{u}_m]$$

so that the covariance update then becomes

$$\boldsymbol{S}_k^+\boldsymbol{S}_k^+ = \boldsymbol{S}_k^-\boldsymbol{S}_k^{-^T} - \boldsymbol{u}_1\boldsymbol{u}_1^T - \boldsymbol{u}_2\boldsymbol{u}_2^T - \cdots - \boldsymbol{u}_m\boldsymbol{u}_m^T .$$

This then requires a sequence of Cholesky downdates to update the covariance. This is done so that if $m = 2$ the rank-1 downdate is shown as

$$\tilde{\boldsymbol{S}}_k^-(\tilde{\boldsymbol{S}}_k^-)^T = \boldsymbol{S}_k^-(\boldsymbol{S}_k^-)^T - \boldsymbol{u}_1\boldsymbol{u}_1^T$$

which is then used to find the updated square root factors determined by a second rank-1 downdate where

$$\boldsymbol{S}_k^+(\boldsymbol{S}_k^+)^T = \tilde{\boldsymbol{S}}_k^-(\tilde{\boldsymbol{S}}_k^-)^T - \boldsymbol{u}_2\boldsymbol{u}_2^T .$$

This gives the *a posteriori* covariance square-root update, $\boldsymbol{S}_k^+$, which can be done where the Cholesky downdate is represented by choldowndate$(\cdot)$, as

$$\boldsymbol{S}_k^+ = \text{choldowndate}\{\boldsymbol{S}_k^{-T}, \boldsymbol{U}_k\}^T \,.$$

This now allows the covariance of both the propagation and the update to be done using square-root factors, providing the more robust and reliable filter that was intended.

2.3.2. Residual Editing. On occasion, the data provided by the sensors may fall uncharacteristically outside of what would be expected, which can result in a lose of accuracy as the measurement is processed by the filter. This form of unreliable data can occur for numerous reasons, but in order to make the filter robust in such a manner that it can account for this, a residual editing method is implemented. This is a form of checking the measurement residual and its error covariance to determine if the data used to update the filter is reliable, and if not, the data can be ignored and discarded. To do this, a probability gate to compare against is chosen from a χ^2 distribution table [25]. Using the Mahalanobis distance of the residual to compare to this probability gate provides information on if the data update is usable or not [2, 17]. In the case that the data is unreliable, the data is discarded and the *a priori* mean and covariance are used as the *a posteriori* updated mean and covariance. This previously mentioned Mahalanobis distance is found as

$$d = \boldsymbol{r}_k^T \boldsymbol{W}_k^{-1} \boldsymbol{r}_k$$

where $\boldsymbol{W}$ is the measurement covariance and $\boldsymbol{r}$ is the measurement residual. This is compared to the χ^2 probability gate, P_g, given its degrees of freedom, q, where if

$$d > \chi^2(q, P_g)$$

such that the Mahalanobis distance, d, was larger than the χ^2 probability gate, the update would be discarded. This will keep the filter from losing accuracy by diverging unnecessarily from its solution and provide an added increment of robustness to the filter.

2.3.3. Underweighting. In some cases the filter can have a long propagation corresponding to a gap between measurements that forms a large state estimation error covariance that, when updated with an accurate measurement, can cause an over-convergence that can then cause the filter to diverge. To counteract this process a "softening" to the covariance with an underweighting [18, 27] factor can be added to the gain keeping this over-convergence from happening. To do this the gain is altered such that

$$\boldsymbol{K}_k = \boldsymbol{P}_k^- \boldsymbol{H}_k^T \left[\frac{1}{p} \boldsymbol{H}_k \boldsymbol{P}_k^- \boldsymbol{H}_k^T + \boldsymbol{L}_k \boldsymbol{R}_k \boldsymbol{L}_k \right]^{-1},$$

so that the $\frac{1}{p}$ factor can soften the update. This update needs to remain positive where if $p = 1$ it can be seen that the gain is then unchanged from the Kalman gain. The ''standard" value of p is $\frac{5}{6}$, as was used for the Shuttle navigation system [18]. To do this there must be a check that satisfies some cutoff point. This cutoff point can be defined as

$$\left\| \boldsymbol{H}_k \boldsymbol{P}_k^- \boldsymbol{H}_k^T \right\| > \frac{p}{1-p} \left\| \boldsymbol{L}_k \boldsymbol{R}_k \boldsymbol{L}_k^T \right\|,$$

where the softening factor, p, is the acceptable percentage decrease of the uncertainty across a single update. With this "softened" gain implemented a filter can be maintained that, even after long periods of propagation, can converge on a solution quickly without then diverging from that solution unnecessarily.

3. MEASUREMENT MODELS

In order for the spacecraft to determine its state, a suit of sensors must be utilized to relay information to update the filter. Here the use of an inertial measurement unit (IMU), nadir pointing altimeter, and star camera provide these updates to the filter. Many other sensors such as a terrain camera for terrain relative navigation or a non-nadir pointing altimeter for slant range altitude measurements could be used as well to provide more information on the state of the vehicle. While these sensors provide rapid measurements to the filter, the time between those measurements needs to be propagated by the dynamics to estimate the state at the time a new measurement is provided. The use of sensor models is then required to compare this new estimate of the state to that of the state defined by the new measurement from the sensors. These models then need to be defined such that they can work within the framework of the filter, requiring both measurements from the model along with measurement Jacobians required for the EKF. The subsequent sections provide a definition of the sensors in use and an explanation of how they determine their respective measurements.

3.1. INERTIAL MEASUREMENT UNIT

The inertial measurement unit (IMU) modeled for use within the filter is assumed to operate by returning integrated non gravitational acceleration and integrated angular velocity measurements [28]. While IMU measurements can have several error sources, the only ones taken into effect here are a stationary bias and thermo-mechanical noise. Using this information, the IMU model is given by

$$\Delta\boldsymbol{v}_{m,k} = \Delta\boldsymbol{v}_k + \boldsymbol{b}_v + \boldsymbol{w}_{v,k} \tag{3.1a}$$

$$\Delta\boldsymbol{\theta}_{m,k} = \Delta\boldsymbol{\theta}_k + \boldsymbol{b}_\theta + \boldsymbol{w}_{\theta,k}\,, \tag{3.1b}$$

where $\Delta\boldsymbol{v}_k$ is the true integrated non gravitational inertial acceleration expressed in the IMU case frame, $\boldsymbol{b}_v$ is the accelerometer bias, $\boldsymbol{w}_{v,k}$ is the accelerometer noise, $\Delta\boldsymbol{\theta}_k$ is the true integrated angular velocity of the IMU case frame with respect to the inertial frame expressed in the IMU case frame, $\boldsymbol{b}_\theta$ is the gyro bias, and $\boldsymbol{w}_{\theta,k}$ is the gyro noise. The noises and biases here are assumed to be zero mean and covariances of the form

$$E\{\boldsymbol{b}_v\boldsymbol{b}_v^T\} = \boldsymbol{B}_v \qquad E\{\boldsymbol{b}_\theta\boldsymbol{b}_\theta^T\} = \boldsymbol{B}_\theta$$
$$E\{\boldsymbol{w}_{v,k}\boldsymbol{w}_{v,l}^T\} = \boldsymbol{W}_{v,k}\delta_{k,l} \qquad E\{\boldsymbol{w}_{\theta,k}\boldsymbol{w}_{\theta,l}^T\} = \boldsymbol{W}_{\theta,k}\delta_{k,l}\,,$$

where $\delta_{k,l}$ is the Kronecker delta, such that $\delta_{k,l} = 1$ when $k = l$ and is zero otherwise, which makes it so accelerometer and gyro noises are taken as white noise. In order to solve for the true integrated non gravitational acceleration and angular velocity Eq. 3.1 is taken and rearranged via an “inversion” to yield

$$\Delta\boldsymbol{v}_k = \Delta\boldsymbol{v}_{m,k} - \boldsymbol{b}_v - \boldsymbol{w}_{v,k}$$
$$\Delta\boldsymbol{\theta}_k = \Delta\boldsymbol{\theta}_{m,k} - \boldsymbol{b}_\theta - \boldsymbol{w}_{\theta,k}\,,$$

then, the estimated integrated non gravitational acceleration and integrated angular velocity can be found as

$$\Delta\hat{\boldsymbol{v}}_k = \Delta\boldsymbol{v}_{m,k} - \hat{\boldsymbol{b}}_{v,k}$$
$$\Delta\hat{\boldsymbol{\theta}}_k = \Delta\boldsymbol{\theta}_{m,k} - \hat{\boldsymbol{b}}_{\theta,k}\,.$$

3.2. NADIR ALTIMETER

In order to model the surface below the spacecraft, different altimeter sensor models are defined. These models are presented for the nadir altimeter that is continually directed in the direct downward position below the spacecraft. This modeling of the surface can be done with varying degrees of fidelity. While some models, such as spherical, may be simpler for the filter to utilize, it also produces a less precise solution. Meanwhile, models of a higher fidelity topographic model can produce a much more precise altitude solution but have a trade-off in increased complexities that can make producing a stable filtering solution difficult. As such an altimeter needs to be defined that can take advantage of the different model fidelities that are available to the filter.

Before these different model complexities are developed it is of interest to define a general algorithm for the construction of these models. This provides a framework that the subsequent model fidelities can follow to provide solutions to the filter. To start this, the measurement, z, and its estimate, $\hat{z}$, are defined as

$$z = h(\boldsymbol{r}^{f}_{alt}) + b_{alt} + v_{alt} \tag{3.2}$$

$$\hat{z} = h(\hat{\boldsymbol{r}}^{f}_{alt}) + \hat{b}_{alt} \tag{3.3}$$

where it is noted that any variable given a $(\hat{\cdot})$ designation is an estimate of that variable, h represents the height of the vehicle above the surface, which depends on the model in use, b_{alt} represents the bias in the altimeter, and v_{alt} represents the noise.

The deviation of the altimeter between this truth and its estimate, also known as the measurement residual, is represented by δh as

$$\begin{aligned}\delta h &= z - \hat{z} \\ &= \left[h(\boldsymbol{r}^{f}_{alt}) - h(\hat{\boldsymbol{r}}^{f}_{alt})\right] + \delta b_{alt} + v_{alt}\,,\end{aligned}$$

where $\delta b_{alt} = b_{alt} - \hat{b}_{alt}$. The function $h(\boldsymbol{r}^f_{alt})$ can then be expanded in a first-order Taylor Series where

$$h(\boldsymbol{r}^f_{alt}) \simeq h(\hat{\boldsymbol{r}}^f_{alt}) + \frac{\partial h}{\partial \boldsymbol{r}} \delta \boldsymbol{r}^f_{alt}$$

where $\delta \boldsymbol{r}^f_{alt}$ is the deviation of the true altimeter position in the fixed frame from its estimate, which equals $\boldsymbol{r}^f_{alt} - \hat{\boldsymbol{r}}^f_{alt}$. This then yields the measurement deviation as

$$\delta h = \left[\left. \frac{\partial h(\boldsymbol{r}^f_{alt})}{\partial \boldsymbol{r}^f_{alt}} \right|_{\boldsymbol{r}^f_{alt} = \hat{\boldsymbol{r}}^f_{alt}} \right] \delta \boldsymbol{r}^f_{alt} + \delta b_{alt} + v_{alt} \,,$$

where if it is said

$$h_r = \left[\left. \frac{\partial h(\boldsymbol{r}^f_{alt})}{\partial \boldsymbol{r}^f_{alt}} \right|_{\boldsymbol{r}^f_{alt} = \hat{\boldsymbol{r}}^f_{alt}} \right],$$

then

$$\delta h = h_r \delta \boldsymbol{r}^f_{alt} + \delta b_{alt} + v_{alt} \,. \tag{3.4}$$

Now that the altimeter deviation has been found in terms of the fixed frame, it needs to be altered, such that the deviation can be determined via the inertial frame. This will allow the relation of the position to the IMU from where the measurements of the state are taken. To do this the true altimeter position, $\delta \boldsymbol{r}^f_{alt}$, now needs to be defined in terms of the inertial position such that

$$\delta \boldsymbol{r}^f_{alt} = \boldsymbol{r}^f_{alt} - \hat{\boldsymbol{r}}^f_{alt} = \boldsymbol{T}^f_i \delta \boldsymbol{r}^i_{alt} \tag{3.5}$$

where $\boldsymbol{T}^f_i$ is the transformation from the inertial to the fixed frame, and $\delta \boldsymbol{r}^i_{alt}$ is then the deviation of the altimeter position in the inertial frame where

$$\delta \boldsymbol{r}^i_{alt} = \boldsymbol{r}^i_{alt} - \hat{\boldsymbol{r}}^i_{alt} \,. \tag{3.6}$$

The truth and extimate of these positions can then be defined as

$$\boldsymbol{r}^i_{alt} = \boldsymbol{r}^i_{imu} + \boldsymbol{T}^i_b \boldsymbol{r}^b_{alt/imu} \tag{3.7}$$

$$\hat{\boldsymbol{r}}^i_{alt} = \hat{\boldsymbol{r}}^i_{imu} + \hat{\boldsymbol{T}}^i_b \boldsymbol{r}^b_{alt/imu} \,. \tag{3.8}$$

where $\boldsymbol{r}^i_{imu}$ is the position of the IMU in the inertial frame, $\boldsymbol{T}^i_b$ is the transformation from the body frame of the spacecraft to the inertial frame, and $\boldsymbol{r}^b_{alt/imu}$ is the known position of the altimeter with respect to the IMU in the body frame of the spacecraft. Equation 3.6 is then rewritten using Eqs. 3.7 and 3.8 as

$$\begin{aligned} \delta \boldsymbol{r}^i_{alt} &= \left[\boldsymbol{r}^i_{imu} - \hat{\boldsymbol{r}}^i_{imu} \right] + \left[\boldsymbol{T}^i_b - \hat{\boldsymbol{T}}^i_b \right] \boldsymbol{r}^b_{alt/imu} \\ &= \delta \boldsymbol{r}^i_{imu} + \left[\boldsymbol{T}^i_b - \hat{\boldsymbol{T}}^i_b \right] \boldsymbol{r}^b_{alt/imu} \,, \end{aligned} \tag{3.9}$$

where if it is noted via the small angle relationship [5] that

$$\boldsymbol{T}^i_b \approx \hat{\boldsymbol{T}}^i_b + \hat{\boldsymbol{T}}^i_b \left[\delta \boldsymbol{\phi} \times \right] ,$$

Eq. 3.9 can then be evaluated as

$$\begin{aligned} \delta \boldsymbol{r}^i_{alt} &= \delta \boldsymbol{r}^i_{imu} + \hat{\boldsymbol{T}}^i_b \left[\delta \boldsymbol{\phi} \times \right] \boldsymbol{r}^b_{alt/imu} \\ &= \delta \boldsymbol{r}^i_{imu} - \hat{\boldsymbol{T}}^i_b \left[\delta \boldsymbol{r}^b_{alt/imu} \times \right] \delta \boldsymbol{\phi} \,, \end{aligned} \tag{3.10}$$

also noting $\delta\phi$ represents the three-parameter attitude error. Equation 3.10 can now be substituted into Eq. 3.5 to get

$$\delta \boldsymbol{r}^{f}_{alt} = \boldsymbol{T}^{f}_{i} \delta \boldsymbol{r}^{i}_{imu} - \boldsymbol{T}^{f}_{i} \hat{\boldsymbol{T}}^{i}_{b} \left[\boldsymbol{r}^{b}_{alt/imu} \times \right] \delta \phi$$

to represent the deviation in the position of the altimeter in the fixed frame. This is then used in the general altimeter measurement deviation in Eq. 3.4 where it can be shown that a first-order expansion of this becomes

$$\delta h = h_{\boldsymbol{r}} \boldsymbol{T}^{f}_{i} \delta \boldsymbol{r}^{i}_{imu} - h_{\boldsymbol{r}} \boldsymbol{T}^{f}_{i} \hat{\boldsymbol{T}}^{i}_{b} [\boldsymbol{r}^{b}_{alt/imu} \times] \delta \boldsymbol{\phi} + \delta b_{alt} + v_{alt} \,. \tag{3.11}$$

This first-order expansion provides the Jacobian in terms of its elements, which are needed for use in the filtering algorithm. This general formulation will be used going forward for the different fidelity models in use. While simple models, such as spherical, play out very similar to the general algorithm, higher fidelity models require slightly more complex steps to determine the altitude and first-order expansion of the measurement. Therefore, as these models are developed these general equations provide the framework that the subsequent sections will follow.

3.2.1. Spherical Altimeter. The simplest model for use in the altimeter is that of a sphere. This takes the radial distance of the body and uses it to produce a sphere of that radius for the filter to estimate upon. To do this it is first said that

$$h_{sph} = ||\boldsymbol{r}^{i}_{alt}|| - r_{sph} \tag{3.12}$$

so that h_{sph} is the height of the spacecraft above the spherical surface, $||\boldsymbol{r}^{i}_{alt}||$ is the inertial position of the altimeter, and r_{sph} is the radius of the spherical surface. This gives a true and estimate measurement by substituting Eq. 3.12 into Eqs. 3.2 and 3.3 as

$$z = h_{sph}(\boldsymbol{r}^i_{alt}) + b_{alt} + v_{alt}$$
$$\hat{z} = h_{sph}(\hat{\boldsymbol{r}}^i_{alt}) + \hat{b}_{alt} \,.$$

In order to define the $h_{\boldsymbol{r}}$ term of the general altimeter deviation in Eq. 3.11 the partial of Eq. 3.12 is examined with respect to the altimeter's fixed frame position. This is given by

$$\frac{\partial h_{sph}}{\partial \boldsymbol{r}^f_{alt}} = \frac{\partial h_{sph}}{\partial \boldsymbol{r}^i_{alt}} \frac{\partial \boldsymbol{r}^i_{alt}}{\partial \boldsymbol{r}^f_{alt}} = \frac{\partial h_{sph}}{\partial \boldsymbol{r}^i_{alt}} (\boldsymbol{T}^f_i)^{-1} \,. \tag{3.13}$$

so that the partial is now taken with respect to the inertial frame. This partial is then taken by using Eq. 3.12 so that it is then taken as

$$\frac{\partial h_{sph}}{\boldsymbol{r}^i_{alt}} = \frac{\partial \{||\boldsymbol{r}^i_{alt}|| - r_{sph}\}}{\partial \boldsymbol{r}^i_{alt}} = \frac{(\boldsymbol{r}^i_{alt})^T}{||\boldsymbol{r}^i_{alt}||} \,,$$

producing the partial derivative measurement

$$\left[\frac{\partial h(\boldsymbol{r}^f_{alt})}{\partial \boldsymbol{r}^f_{alt}} \bigg|_{\boldsymbol{r}^f_{alt} = \hat{\boldsymbol{r}}^f_{alt}} \right] = \frac{(\hat{\boldsymbol{r}}^i_{alt})^T}{||\hat{\boldsymbol{r}}^i_{alt}||} \boldsymbol{T}^i_f \,.$$

where $\boldsymbol{T}^i_f$ is mearly the inverse of $\boldsymbol{T}^f_i$ from Eq. 3.13. This is then inserted into Eq. 3.11 where

$$\begin{aligned} \delta h &= \frac{(\hat{\boldsymbol{r}}^i_{alt})^T}{||\hat{\boldsymbol{r}}^i_{alt}||} \boldsymbol{T}^i_f \boldsymbol{T}^f_i \delta \boldsymbol{r}^i_{imu} - \frac{(\hat{\boldsymbol{r}}^i_{alt})^T}{||\hat{\boldsymbol{r}}^i_{alt}||} \boldsymbol{T}^i_f \boldsymbol{T}^f_i \hat{\boldsymbol{T}}^i_b [\boldsymbol{r}^b_{alt/imu} \times] \delta \boldsymbol{\phi} + \delta b_{alt} + v_{alt} \\ &= \frac{(\hat{\boldsymbol{r}}^i_{alt})^T}{||\hat{\boldsymbol{r}}^i_{alt}||} \delta \boldsymbol{r}^i_{imu} - \frac{(\hat{\boldsymbol{r}}^i_{alt})^T}{||\hat{\boldsymbol{r}}^i_{alt}||} \hat{\boldsymbol{T}}^i_b [\boldsymbol{r}^b_{alt/imu} \times] \delta \boldsymbol{\phi} + \delta b_{alt} + v_{alt} \end{aligned}$$

providing the altimeter measurement deviation for a spherical altimeter. This provides the simplest model, as it is just the position of the spacecraft subtracted from the radius of the body below it. While it provides a general idea of where the surface will be, because the Moon has a topographic range of 13 km [24], it can still be highly inaccurate.

3.2.2. Ellipsoidal Altimeter. A slightly more complex model is done by implementing an ellipsoidal surface of the body using its eccentricity. This provides a slightly more accurate depiction of the surface than the spherical body, while still maintaining the smooth surface that produces a more stable filter. There are several methods that allow the conversion from Cartesian to Geodetic coordinates, including iterative methods, like in Kaplan and Hegarty [15], or closed form solutions, such as Vermeille's [33] and Sofair's [31] algorithms. While a more comprehensive list of methods can be found [7], for the filter in use Vermeille's algorithm was chosen. This closed-form solution is beneficial in that it allows for the computation of the partial derivatives that are required for use within the filter.

Therefore, when using Vermeille's algorithm, the height of the ellipsoidal surface can be found as

$$h_{ell} = \frac{k + e^2 - 1}{k}\sqrt{d^2 + z^2}$$

where e is the eccentricity, z is a component of the fixed frame position i.e. $r^f_{alt} = [x\ y\ z]^T$, and d and k are given via intermediary steps of Vermeille's algorithm, which are further defined in the Appendix of this work. The truth and estimate of this would then be given as

$$z = h_{ell}(\boldsymbol{r}^f_{alt}) + b_{alt} + v_{alt}$$
$$\hat{z} = h_{ell}(\hat{\boldsymbol{r}}^f_{alt}) + \hat{b}_{alt}$$

where $\boldsymbol{r}^f_{alt}$ is the altimeter position in the planet fixed frame that is used within Vermeille's algorithm, and $\hat{\boldsymbol{r}}^f_{alt}$ is the estimate of that position.

The partial derivative also needs to be defined using the components of this algorithm. The intermediary steps are again shown in the Appendix, but culminate in

$$\frac{\partial h}{\partial \boldsymbol{r}^f_{alt}} = \frac{1}{a^2}(\boldsymbol{r}^f_{alt})^T \mathrm{diag}(H_{x,y},\ H_{x,y},\ H_z)$$

such that

$$\boldsymbol{H}_r = \mathrm{diag}(H_{x,y},\ H_{x,y},\ H_z)\,.$$

The partial derivative can then be written as

$$\left[\frac{\partial h(\boldsymbol{r}^f_{alt})}{\partial \boldsymbol{r}^f_{alt}}\bigg|_{\boldsymbol{r}^f_{alt}=\hat{\boldsymbol{r}}^f_{alt}}\right] = \frac{1}{a^2}(\hat{\boldsymbol{r}}^f_{alt})^T \boldsymbol{H}_r \boldsymbol{T}^f_i\,,$$

Providing what is needed to complete Eq. 3.11 to complete the ellipsoidal altimeter measurement deviation where

$$\delta h = \frac{1}{a^2}(\hat{\boldsymbol{r}}^f_{alt})^T \boldsymbol{H}_r \boldsymbol{T}^f_i \delta \boldsymbol{r}^i_{imu} - \frac{1}{a^2}(\hat{\boldsymbol{r}}^f_{alt})^T \boldsymbol{H}_r \boldsymbol{T}^f_i \hat{\boldsymbol{T}}^i_b [\boldsymbol{r}^b_{alt/imu}\times]\delta\phi + \delta b_{alt} + v_{alt}\,.$$

The slightly more complex ellipsoidal model allows for a better understanding of the surface below the spacecraft, but still gives no representation of the terrain below. To get a true representation of the surface some amount of terrain would need to be included in the model making the ellipsoidal model a relatively simple model by comparison.

3.2.3. Topographical Altimeter. The model that can provide the highest fidelity is the topographical model. This uses digital elevation models (DEMs) of the surface to provide a more realistic representation of the surface below. Doing this provides a sensor model that can better represent the “real world” that it is trying to match. This also allows the estimate of that "real world" sensor to vary in fidelity compared to the true sensor, while still having some representation of the terrain. The following sections will delve into the intricacies of the DEMs in use and the algorithms used to find the topographical altitude.

3.2.3.1. Digital elevation model. In order to find the altitude above a topographical surface a DEM of that surface must be used. DEMs are models that represent the terrain of the surface by providing elevations with respect to a reference shape. For instance, this study uses a DEM of the Moon where the elevations are given over a reference sphere. This means that for an elevation found on the DEM the radius of the Moon must be added to that elevation to provide the actual height of the surface at that point from the center of the Moon. This is then the height that is subtracted from the inertial position of the spacecraft to provide its altitude. Other DEMs may be provided with elevations over a ellipsoidal surface, which will require a different method of topographical modeling than will be described here.

When using the different DEMs, data is usually provided in the structure of a table of information. This table includes vertices of latitude and longitude that are used for locating the position of an elevation. Depending on the data set provided by the DEM different model fidelities can be provided to the filter. DEMs of the Moon are most often provided by the older Clementine Orbiter [22] or the more recent Lunar Reconnaissance Orbiter (LRO) [23], each of which provide a variety of model fidelities. These DEMs are constructed where their fidelity is given by the degree of spacing between measurements. For the Clementine Orbiter those DEMs consist of a 1° model and a more precise $1/4^\circ$ model, while the LRO has global models of $1/4^\circ$, $1/16^\circ$, $1/64^\circ$, and $1/128^\circ$. Depending on the data set in use these DEMs can be applied to different planets, like Earth or Mars, and can have a range of different fidelities.

3.2.3.2. Topographical algorithm. In order to determine the altitude above the nadir direction a algorithm for discerning the elevation of the surface in that direction must be defined. This algorithm is constructed under the assumption that the DEM is based upon a spherical reference sphere. As such the determination of the latitude (ϕ) and longitude (λ) are constructed in a manner that is based off a spherical body, along with the radius of that sphere being added to the elevation provided by the DEM. In order to find the topography above a reference ellipsoid, Vermeille's algorithm in the Appendix would need to be applied to determine these values.

When determining the elevation of the DEM above a reference shape a bicubic interpolation [26] is performed, which interpolates between the points of elevation that the DEM provides. This table of elevations is constructed with latitude (ϕ) and longitude (λ) as the vertices for locating the values required. As such, the first step in finding the proper elevation is to calculate the latitude and longitude of the spacecraft above the planet fixed frame given by

$$\phi = \sin^{-1}\left(\frac{z}{||\boldsymbol{r}_{alt}^{f}||}\right) \quad \text{and} \quad \lambda = \tan^{-1}\left(\frac{y}{x}\right), \tag{3.14}$$

where x, y, and z describe the components of the position in the fixed frame, i.e. $\boldsymbol{r}_{alt}^{f} = [x\ y\ z]^{T}$. This interpolation also requires a section of the DEM corresponding to the latitude and longitude of the interpolation point, that is represented here by $\boldsymbol{D}$. The intricacies of this method are further developed in Chapter 3.2.3.3 but will be represented here in the function $f(\cdot)$ where

$$r_{top} = f(\phi, \lambda, \boldsymbol{D}) + r_{sph}$$

where $\boldsymbol{r}_{sph}$ is the radius of the reference sphere the DEM is based on and results in r_{top}, the height of the topography below the spacecraft in the nadir pointing direction. To then find the height of the altimeter above the surface, the topographic height is taken and subtracted from the distance of the altimeter from the center of the body, i.e.

$$h_{top} = ||\boldsymbol{r}^f_{alt}|| - r_{top}$$

where h_{top} is the height of the altimeter above the topographical surface of the chosen DEM, so that the truth and estimate are found as

$$z = h_{top}(\boldsymbol{r}^f_{alt}) + b_{alt} + v_{alt}$$
$$\hat{z} = h_{top}(\hat{\boldsymbol{r}}^f_{alt}) + \hat{b}_{alt} .$$

A benefit of using a bicubic interpolation is that the partial derivatives of the topography with respect to the latitude and longitude are produced as a result, and represented here by

$$r_{top,\phi} = \frac{\partial r_{top}}{\partial \phi} \quad \text{and} \quad r_{top,\lambda} = \frac{\partial r_{top}}{\partial \lambda} .$$

These deviations are accompanied by the partial derivatives of Eqs. 3.14 given as

$$\frac{\partial \phi}{\partial \boldsymbol{r}^f_{alt}} = \begin{bmatrix} -\frac{xz}{(\sqrt{x^2+y^2}(x^2+y^2+z^2))} & -\frac{yz}{(\sqrt{x^2+y^2}(x^2+y^2+z^2))} & \frac{\sqrt{x^2+y^2}}{(x^2+y^2+z^2)} \end{bmatrix}$$
$$\frac{\partial \lambda}{\partial \boldsymbol{r}^f_{alt}} = \begin{bmatrix} -\frac{y}{(x^2+y^2)} & \frac{x}{(x^2+y^2)} & 0 \end{bmatrix}$$

giving all the partial derivations needed to calculate the Jacobian for Eq. 3.11 where

$$h_r = \frac{(\boldsymbol{r}^i_{alt})^T}{||\boldsymbol{r}^i_{alt}||} - r_{top,\phi}\frac{\partial \phi}{\partial \boldsymbol{r}^f_{alt}}\boldsymbol{T}^f_i - r_{top,\lambda}\frac{\partial \phi}{\partial \boldsymbol{r}^f_{alt}}\boldsymbol{T}^f_i$$

leaving the final topographical altimeter measurement deviation as

$$\delta h = h_r \delta \boldsymbol{r}^i_{imu} - h_r \hat{\boldsymbol{T}}^i_b [\boldsymbol{r}^b_{alt/imu} \times] \delta \boldsymbol{\phi} + \delta b_{alt} + v_{alt} .$$

3.2.3.3. Bicubic algorithm. The bicubic algorithm allows the determination of the elevation of the surface given a set of latitude and longitude points. In the case of the altimeter this is done for the latitude and longitude along the nadir pointing direction. This form of interpolation is chosen partially because it gives the added benefit of providing the Jacobian of the latitude and longitudes of the table being used. To start this interpolation, the sixteen nearest points to the point being interpolated need to be determined. Using the calculated latitude and longitude of the altimeter position from Eq. 3.14, the table is searched to determine which tile of the DEM the sensor falls within. To do this, a bisection search is done to find the lower bounding point of this tile, where the indices, denoted by j and k, are found when

$$\phi_j \leq \phi < \phi_{j+1} \quad \text{and} \quad \lambda_k \leq \lambda < \lambda_{k+1} \, .$$

This is demonstrated in Figure 3.1, where the latitude and longitude found from the altimeter make up the x and y values for the location of the interpolation point (noted by the diamond). The four points surrounding that interpolation point make up the tile (grey area) in which that point falls. The points of the tile and those immediately connected to it make up the extracted grid of sixteen points for the interpolation process, noted as $\boldsymbol{D}$ moving forward. With these

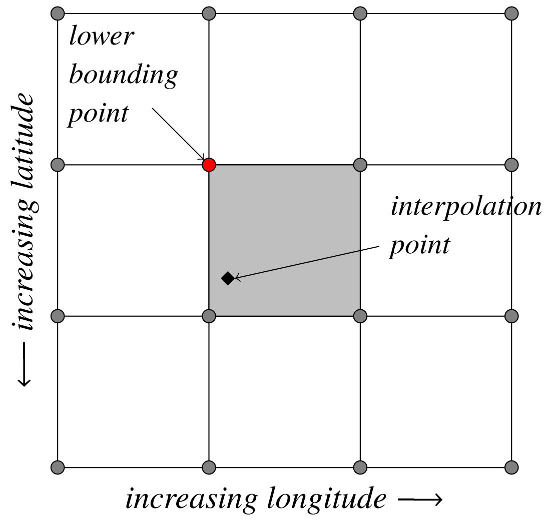


Figure 3.1. Extracted Topography Grid.

points found the interpolation method is started by finding the directional derivatives that are calculated using central differences along the latitude and longitude points as

$$\begin{aligned}\boldsymbol{D}'_{\lambda}(j,k) &= \frac{\boldsymbol{D}(j+1,k)-\boldsymbol{D}(j-1,k)}{\lambda(j+1)-\lambda(j-1)}\\ \boldsymbol{D}'_{\phi}(j,k) &= \frac{\boldsymbol{D}(j,k+1)-\boldsymbol{D}(j,k-1)}{\phi(k+1)-\phi(k+1)}\,.\end{aligned}$$

Here $\boldsymbol{D}'$ is the derivative of the table in the noted direction of latitude (ϕ) or longitude (λ), and k and j are iterated values that loop through the points in $\boldsymbol{D}$ to obtain the derivatives. This is also done for the derivatives across latitude and longitude as

$$\boldsymbol{D}'_{\lambda,\phi}(j,k) = \frac{\boldsymbol{D}(j+1,k+1)-\boldsymbol{D}(j+1,k-1)-\boldsymbol{D}(j-1,k+1)+\boldsymbol{D}(j-1,k-1)}{(\lambda(j+1)-\lambda(j-1))(\phi(k+1)-\phi(k-1))}\,.$$

These sixteen table values and their derivatives are then used to acquire a matrix c that is a known linear transformation with pre-determined coefficients [26]. This matrix is then used to calculate the interpolated point as

$$h = \sum_{n=1}^{4}\sum_{m=1}^{4} c_{i,j}t^{i-1}u^{j-1}\,,$$

where t and u are given as

$$t = \frac{\lambda-\lambda_k}{\lambda_{k+1}-\lambda_k} \qquad \text{and} \qquad u = \frac{\phi-\phi_j}{\phi_{j+1}-\phi_j}\,.$$

The directional derivatives of this interpolation are then found as

$$\begin{aligned}\frac{dh}{d\lambda} &= \sum_{n=1}^{4}\sum_{m=1}^{4}(i-1)c_{i,j}t^{i-2}u^{j-1}\frac{dt}{d\lambda}\\ \frac{dh}{d\phi} &= \sum_{n=1}^{4}\sum_{m=1}^{4}(j-1)c_{i,j}t^{i-1}u^{j-2}\frac{du}{d\phi}\end{aligned}$$

where the derivatives of t and u with respect to λ and ϕ are found as

$$t = \frac{1}{\lambda_{k+1} - \lambda_k} \qquad \text{and} \qquad u = \frac{1}{\phi_{j+1} - \phi_j}\,.$$

3.3. QUATERNION STAR CAMERA

To determine the attitude of the spacecraft, a star camera that produces a quaternion measurement is utilized. These sensor measurements can be determined in a couple of ways. First, they can be processed through the filter [10], or they can use their own pre-processing algorithms, such as QUEST [29] or Davenport's q-Method [16] to produce what will be used as the quaternion measurement. When processing the measurements within the filter the estimated pixel locations and known unit vectors of star locations are processed by the attitude filter. The second method uses estimated and known unit vectors of stars and finds an attitude solution via a pre-processing algorithm that is then processed by the filter. For use within this testing a pre-processing solution is provided to the filter. That quaternion measurement is used to describe the orientation of the star camera frame (sc) with respect to the inertial frame. This true attitude measured by the sensor is then taken as

$$\bar{\boldsymbol{z}} = \bar{\boldsymbol{q}}_{err} \otimes \bar{\boldsymbol{q}}_c^{sc} \otimes \bar{\boldsymbol{q}}_i^c\,,$$

where $\boldsymbol{q}_i^c$ is the true attitude of the case frame of the IMU with respect to the inertial frame, $\bar{\boldsymbol{q}}_c^{sc}$ is the known attitude of the star camera frame with respect to the IMU case frame, and $\bar{\boldsymbol{q}}_{err}$ is rotational error produced from bias and noise, represented here by

$$\bar{\boldsymbol{q}}_{err} = \begin{bmatrix} \sin(\frac{1}{2}||\boldsymbol{\theta}_{err}||)\frac{\boldsymbol{\theta}_{err}}{||\boldsymbol{\theta}_{err}||} \\ \cos(\frac{1}{2}||\boldsymbol{\theta}_{err}||) \end{bmatrix}.$$

where $\boldsymbol{\theta}_{err} = \boldsymbol{b}_{sc} + \boldsymbol{v}_{sc}$, with $\boldsymbol{b}_{sc}$ representing zero-mean, constant, random bias, and $\boldsymbol{v}_{sc}$ as a zero-mean, white-noise sequence. From here, the measurement estimate of the star camera can be determined as

$$\hat{\bar{z}} = \hat{\bar{\boldsymbol{q}}}_{err} \otimes \bar{\boldsymbol{q}}_c^{sc} \otimes \hat{\bar{\boldsymbol{q}}}_i^c \,,$$

where $\hat{\bar{\boldsymbol{q}}}_i^c$ is the estimated attitude quaternion and $\hat{\bar{\boldsymbol{q}}}_{err}$ is given as

$$\hat{\bar{\boldsymbol{q}}}_{err} = \begin{bmatrix} \sin(\frac{1}{2}||\hat{\boldsymbol{\theta}}_{err})\frac{\hat{\boldsymbol{\theta}}_{err}}{||\hat{\boldsymbol{\theta}}_{err}||} \\ \cos(\frac{1}{2}||\hat{\boldsymbol{\theta}}_{err}||) \end{bmatrix},$$

where $\hat{\boldsymbol{\theta}}_{err} = \hat{\boldsymbol{b}}_{sc}$. To determine the deviation $\delta\boldsymbol{q}$ between the true and estimated values a multiplicative form must be defined since the quaternion is not additive; this is done here as

$$\delta\boldsymbol{q} = 2 \cdot \text{vec}(\bar{z} \otimes \hat{\bar{z}}^{-1})\,.$$

This multiplicative quaternion residual is then used in the first-order expansion needed for the filter as

$$\begin{aligned} \delta\boldsymbol{q} &= \boldsymbol{T}_c^{sc} 2 \cdot \text{vec}\{\bar{\boldsymbol{q}}_i^c \otimes (\hat{\bar{\boldsymbol{q}}}_i^c)^{-1}\} + (\boldsymbol{b}_{sc} - \hat{\boldsymbol{b}}_{sc}) + \boldsymbol{v}_{sc} \\ &= \boldsymbol{T}_c^{sc} \delta\boldsymbol{\phi} + \delta\boldsymbol{b}_{sc} + \boldsymbol{v}_{sc} \end{aligned}$$

where $\boldsymbol{T}_c^{sc}$ is the transformation matrix representation of $\bar{\boldsymbol{q}}_c^{sc}$. This provides us with the Jacobian needed for the filtering process.

4. PRELIMINARY RESULTS

As has been described, there are several choices for fidelity within the altimeter sensor models. The interest of this investigation is to examine how the navigation performance is affected by using the different models within the filter. There are several components where different model fidelities can be utilized. These components include the true measurement that represents the sensor output, the estimated measurement used in the filter, and the measurement Jacobian used in the filter. Using different fidelities within these three components is the primary interest here to determine an appropriate configuration to achieve for an accurate and robust filtering solution.

To determine how the filter should perform, a baseline test is carried out along a descent-to-landing trajectory at the Moon. The vehicle is equipped with an IMU, a nadir-pointing altimeter, and a quaternion star camera, which are described in Chapter 3. Sensor specifications for the aforementioned sensors can be found in Table 4.1 [1], where the measurement noise and bias specifications are provided as 1σ standard deviations, along with the operating conditions for the various sensors. It is noted for the star camera that measurements are only produced when the vehicle is not thrusting, due to the resulting vibrations. In Figure 4.1, the descent trajectory is shown as it progresses through the mission elapsed time (MET), along with a visual representation of when the star camera stops generating measurements, due to thrusting for the landing scenario, and when the altimeter starts. Figure 4.2 provides a ground track for the trajectory that follows closely along the equator of the Moon.

[1]These values are from the specifications sheet for the Northrop Grumman LN-200S IMU, which provides specifications in terms of accelerations and body rates. For these values, see: `http://www.northropgrumman.com/Capabilities/LN200FOG/Documents/ln200s.pdf`

Table 4.1. Sensor configuration for the simulated lander.

Sensor	Rate	Noise (1σ)	Bias (1σ)	Operating Condition
Accelerometer	40 Hz	$35\mu g/\sqrt{\text{Hz}}$	$300\mu g$	Always
Gyroscope	40 Hz	$< 0.07°/\sqrt{\text{hr}}$	1°/hr	Always
Altimeter	10 Hz	10 m	0.5 m	alt. < 8 km
Star Camera	1 Hz	30 arc-seconds	10 arc-seconds	When not thrusting

The filter state consists of position, velocity, and attitude states, along with accelerometer, gyro, altimeter, and star camera biases. Since the biases are presented in Table 4.1, the initial uncertainties for the position, velocity and attitude need to be defined and are taken to be

$$\boldsymbol{P}_0 = \text{diag}(100^2, 100^2, 100^2, 0.01^2, 0.01^2, 0.01^2, 0.0572^2, 0.0572^2, 0.0572^2)$$

with units of m^2, (m/s)^2, and degrees^2 for position, velocity, and attitude, respectively. These uncertainty values are comparable to what would be expected during the powered descent phase of an entry, descent, and landing scenario. The filtering solution is comprised of a SR-MEKF that uses underweighting with a factor of $^5/_6$ [18].

To represent a baseline performance, the filter is implemented using a truth model with an ellipsoidal representation of the lunar surface, and an ellipsoidal model is used for both the filter's estimated measurements and its Jacobian. For this baseline, a Monte Carlo simulation is carried out with 200 trials, where the initial filter estimate and measurement noise are re-sampled with each trial. Figures 4.3 – 4.7 show the position, velocity, and attitude, along with the star camera and altimeter residuals for both a single run and a comparison of a single run to the Monte Carlo simulation. The position and velocity in Figures 4.3 and 4.4 show that the altimeter works to enable the navigation filter to converge on a more accurate solution once its measurements are introduced. The results displayed here are in the lunar centered inertial frame and provide errors in $[x\ y\ z]$ components.

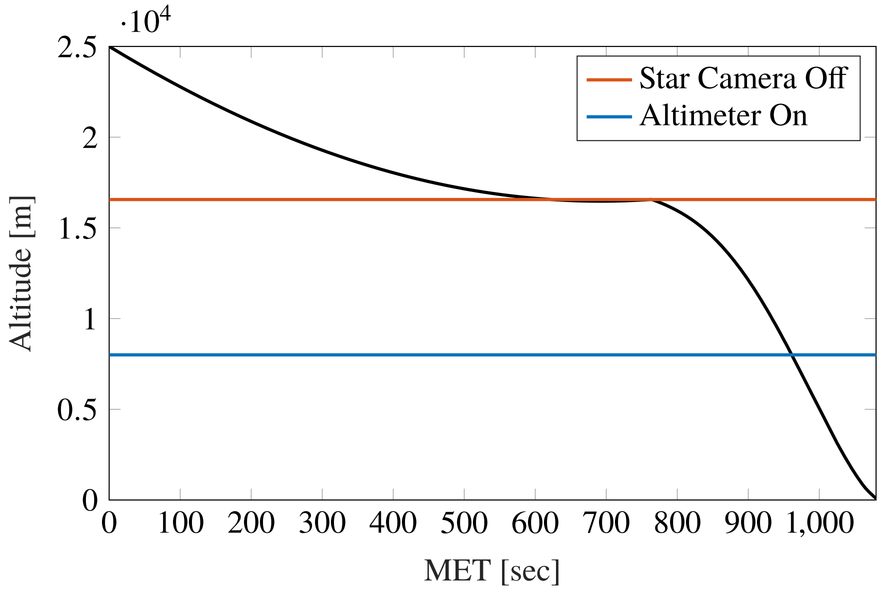


Figure 4.1. Altitude Decent Trajectory.

Considering the results, a convergence is best shown in the x axis direction immediately after the filter acquires altimeter measurements. This larger convergence in the x direction is due to the fact that the trajectory for this descent falls very close to the equator, so most of the measurement data is obtained from the altimeter in that direction. The attitude in Figure 4.5 stays close to zero until around 750 seconds MET where the star camera turns off. The residuals in Figure 4.6 and 4.7 stay within the 3σ covariance interval as is expected for both the star camera and altimeter. For all state errors the Monte Carlo simulation (solid grey) closely follows the single run performance (dashed black), as is expected in this scenario. Given the smooth ellipsoidal surface the navigation filter is using, it is able to easily find a solution as it descends through the trajectory.

4.1. IDENTICAL FIDELITIES

Now that the baseline has been defined, it is of interest to see how increasing model fidelities affect the performance of the navigation filter. These simulations follow the same trajectory and also have 200 trial simulations, similar to the baseline run. All fidelities of the

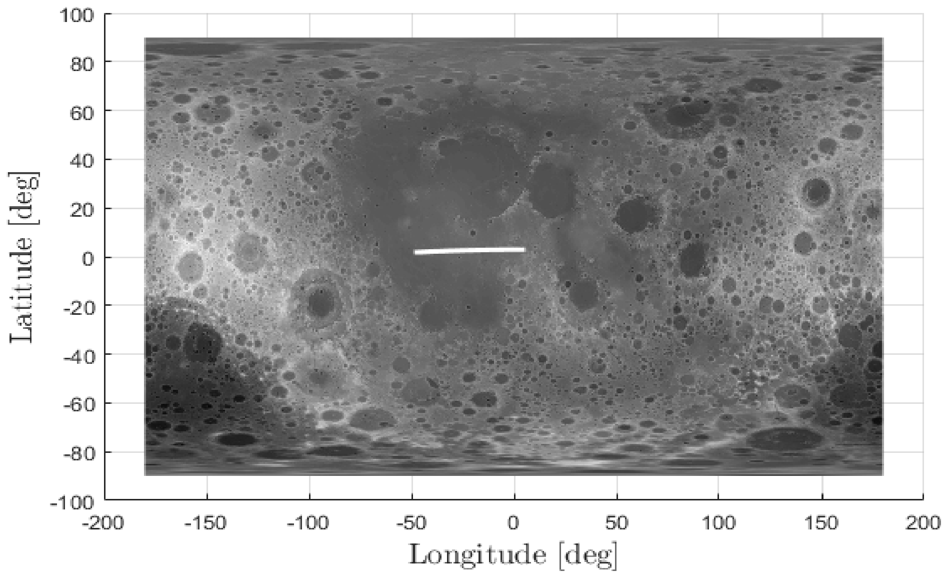
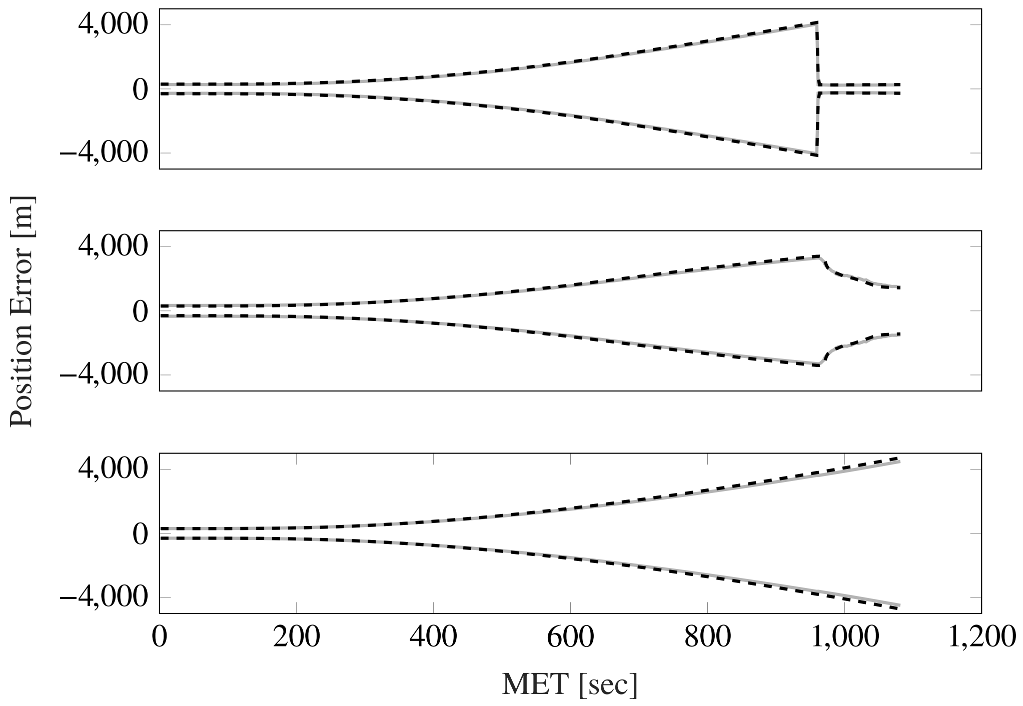


Figure 4.2. Ground track of trajectory starting at left most point and proceeding toward the right.

DEM are implemented, from $1/4°$ to $1/128°$, which are used to model the true measurement, estimated measurement, and measurement Jacobian models. The attitude filtering solutions for these simulations are the same as in the baseline run, so they are excluded here. Also, because the area of interest begins when the altimeter starts taking measurements, the position and velocity errors are only shown during the corresponding time interval.

The smooth and steady convergence seen for the ellipsoidal model changes when the topographic models are used. Considering the $1/4°$ simulation in Figure 4.8, the first aspect of note is how the filter converges onto a solution given the introduction of altimeter measurements. As the filter begins to use these measurements, the uncertainty initially grows, especially seen in the position y axis and velocity x and y axis. This is where the variation of the terrain first start to take effect. As the descent progresses, the filter appears to maintain a more stable solution, portrayed by the steady uncertainty toward the end of the trajectory. When comparing the single run performance (dashed black line) to the Monte Carlo simulation (solid gray line), it is seen that now the single run performance



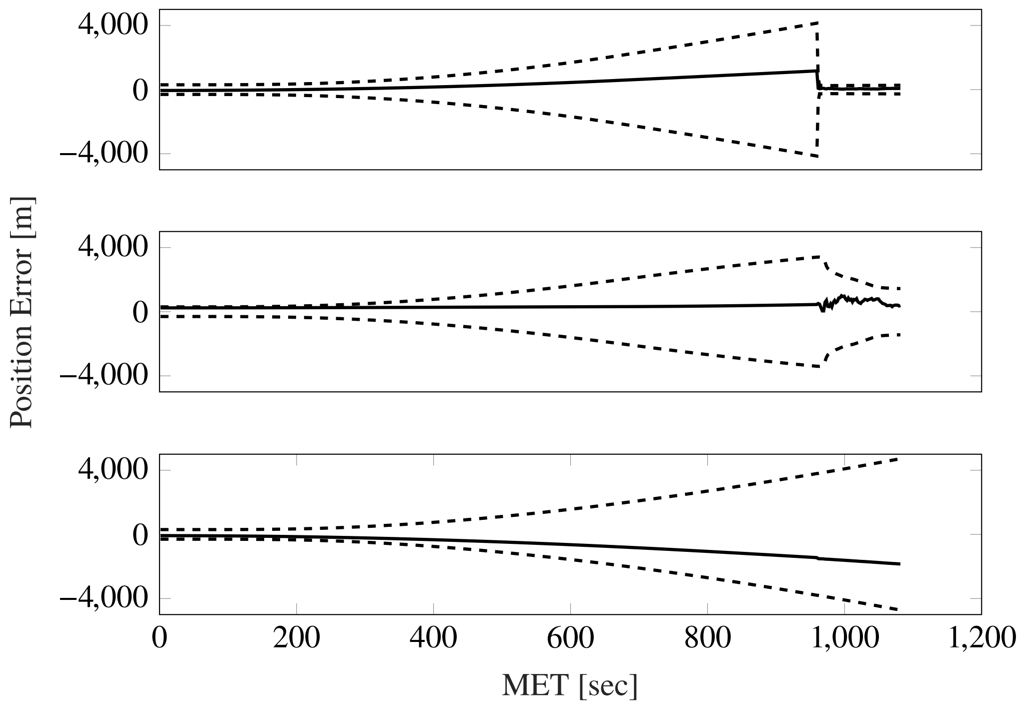
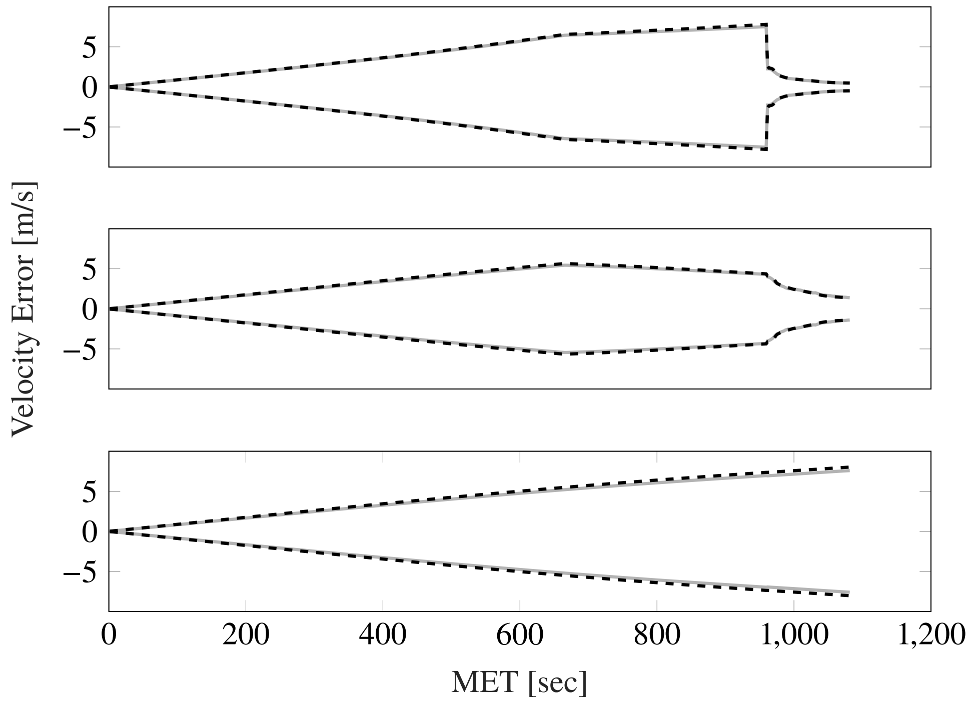


Figure 4.3. Position errors as 3σ intervals (top) for the Monte Carlo simulation (solid gray) and a single filter run (dashed black) and position error (solid black) and 3σ intervals (bottom) for a single run when using the ellipsoidal model.



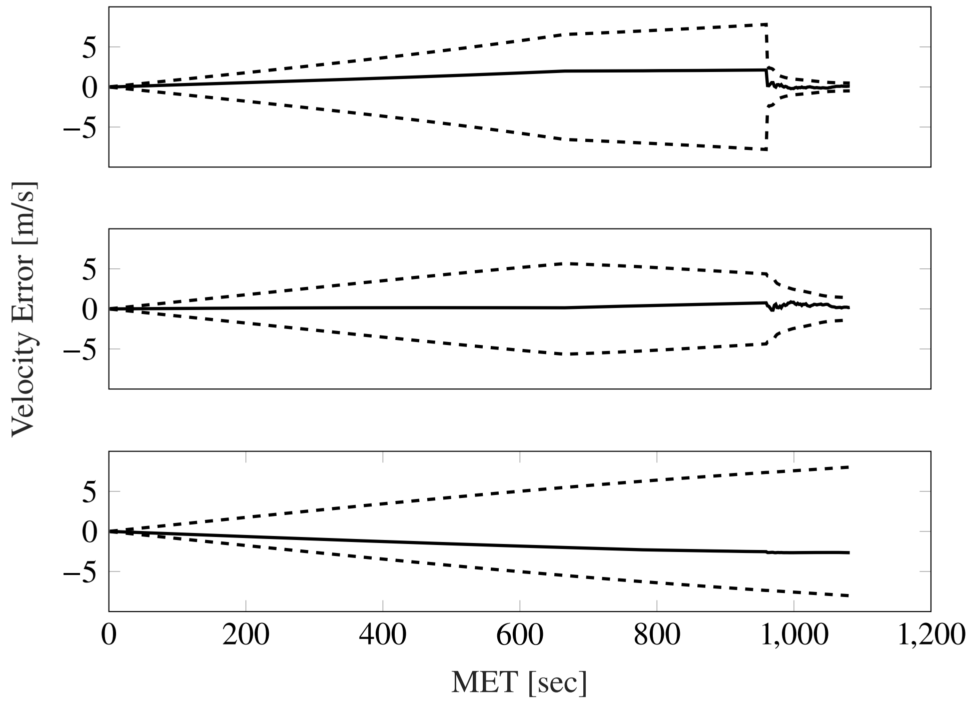


Figure 4.4. Velocity errors as 3σ intervals (top) for the Monte Carlo simulation (solid gray) and a single filter run (dashed black) and position error (solid black) and 3σ intervals (bottom) for a single run when using the ellipsoidal model.

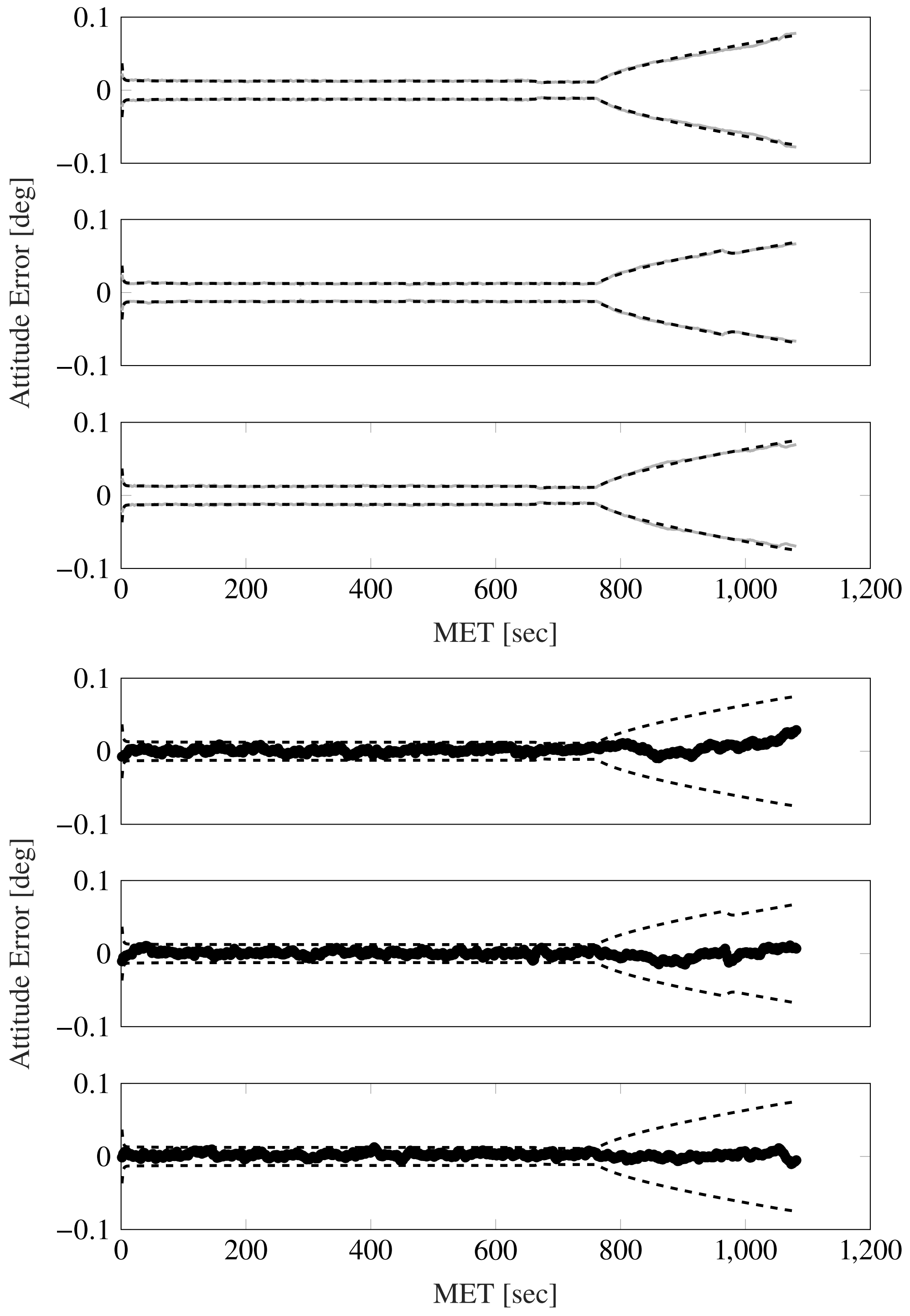
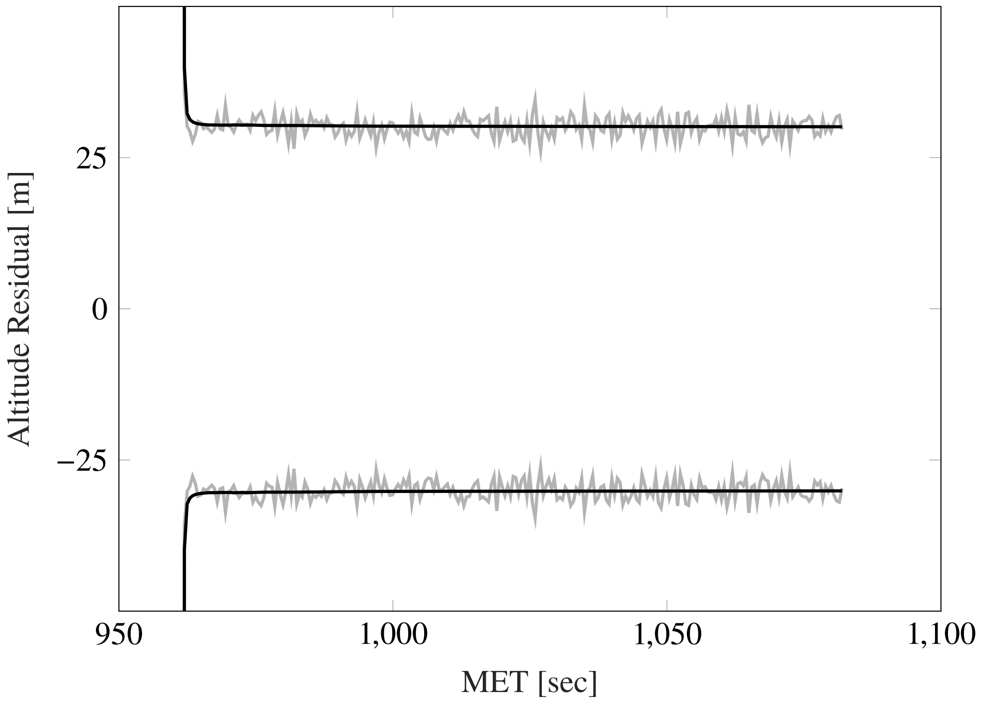


Figure 4.5. Attitude errors as 3σ intervals (top) for the Monte Carlo simulation (solid gray) and a single filter run (dashed black) and position error (solid black) and 3σ intervals (bottom) for a single run when using the ellipsoidal model.



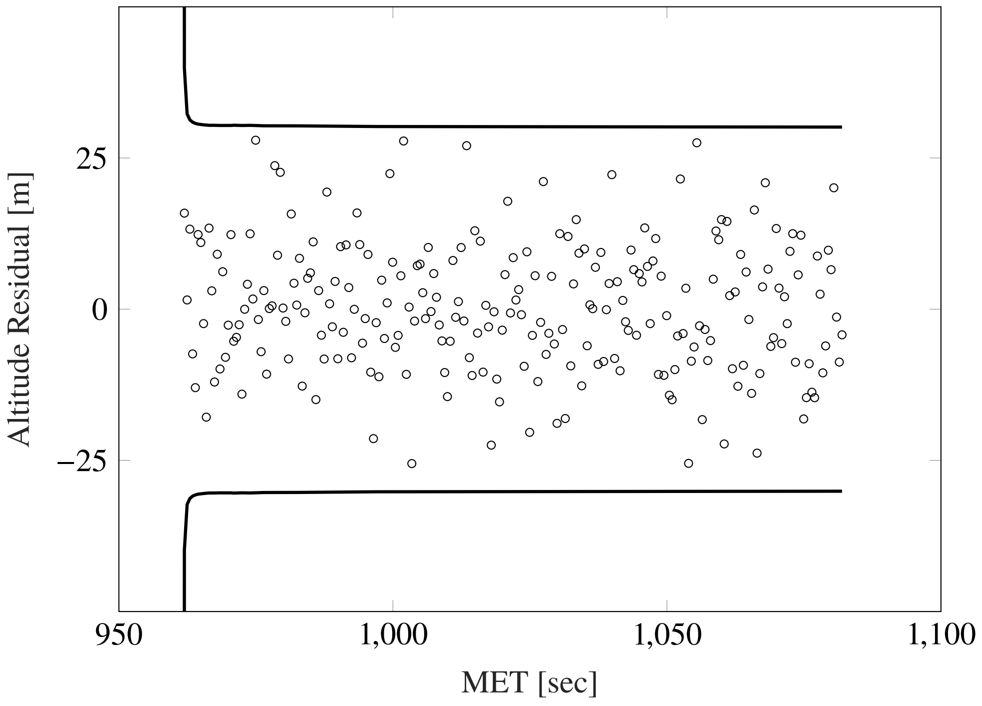
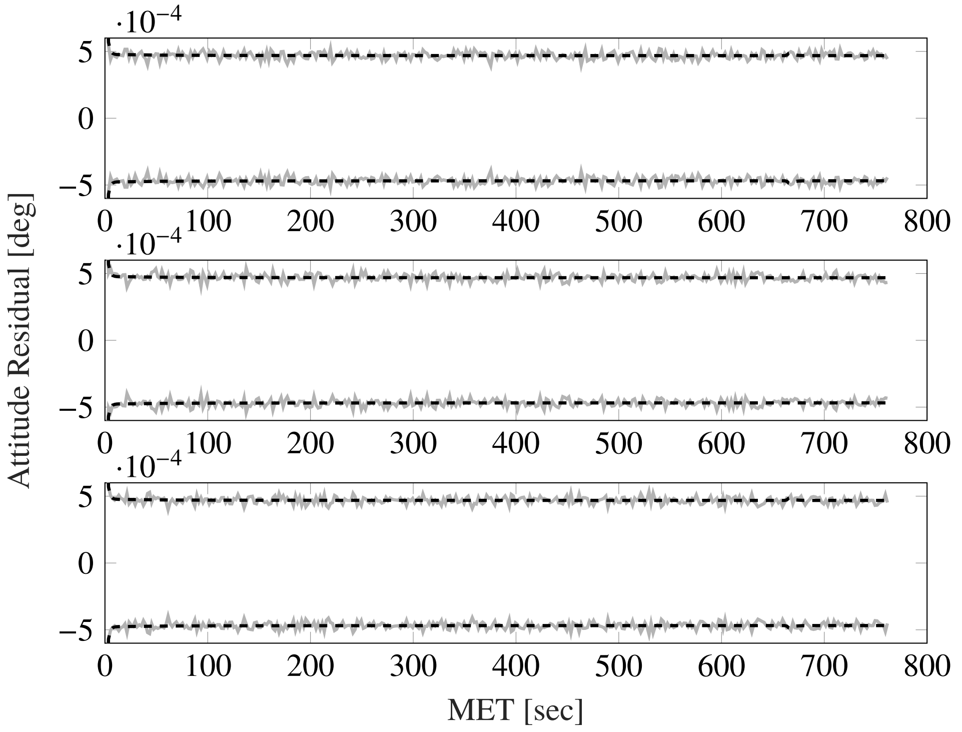


Figure 4.6. Altimeter residual and 3σ interval (top) and altimeter residual 3σ interval (bottom) for the Monte Carlo simulation (solid gray) and a single filter run (solid black) when using the ellipsoidal model.



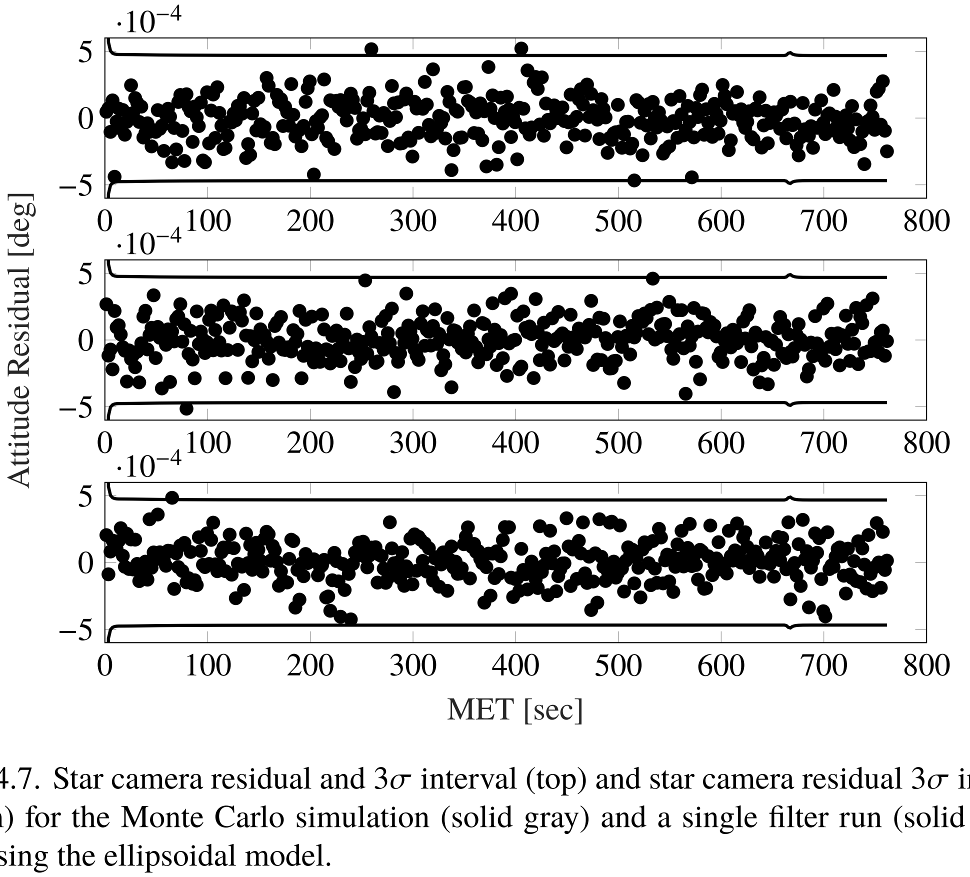


Figure 4.7. Star camera residual and 3σ interval (top) and star camera residual 3σ interval (bottom) for the Monte Carlo simulation (solid gray) and a single filter run (solid black) when using the ellipsoidal model.

is slightly conservative, having a smaller error interval, with respect to the Monte Carlo simulation. While this is not ideal, the simulation is able to converge on a solution by the end of the trajectory that remains steady just outside of the single run performance. The 3σ interval in the altimeter residual matches the single run performance similar to the baseline performance. This provides added confidence to the filtering solution in making the $^1/_4{}^\circ$ DEM a viable option for terrain modeling. The introduction of the $^1/_4{}^\circ$ DEM forces slight instability in the filter until it becomes accustomed to the new measurements. Going forward, it is of interest to note how this changes as fidelities are increased.

The effects of terrain variation on the navigation filter only increase as higher fidelity models are used, making it more difficult for the filter to converge to the true state. Figure 4.9 shows the Monte Carlo simulation results for the $^1/_{16}{}^\circ$ DEM, which quickly shows how the increased variations can negatively affect the filter. The single run performance is vastly more confident compared to the Monte Carlo simulation, showing that the filter has a difficult time converging on the correct solution. This is also reflected in the 3σ altimeter measurement residual intervals. Where the single run performance stays mostly constant, the Monte Carlo result is much more erratic compared to what is observed for the $^1/_4{}^\circ$ DEM.

To counteract these issues, the introduction of tuning noise is used to help the filter determine a solution that matches the single run performance. To do this a tuning parameter of

$$\boldsymbol{Q}_{Tune} = diag([1.5,\ 1.5,\ 1.5,\ 0.005,\ 0.005,\ 0.005])$$

is created where the first three values represent position noise with m^2 units, and the second three represent velocity noise with $(\mathrm{m/s})^2$ units, and the remaining parameters of the state receive no tuning noise. The results of using this tuning noise, seen in Figure 4.10, show that the filter is vastly improved due its introduction. The Monte Carlo simulation is now slightly conservative until the very end of the trajectory compared to the single run performance, maintaining better performance than the $^1/_4{}^\circ$ DEM. The effects of the terrain are still very much present in the "wobble" of the solution, but the filter is still able to converge and maintain a reasonably accurate solution. While this tuning noise introduces a variable that can change depending on the fidelity and navigation filter in use, this shows how useful it can be to improve the performance of the filter.

As the fidelity is increased, introducing more erratic terrain variations, the tuning noise input to the filter must also be increased. For the $^1/_{64}{}^\circ$ DEM this requires the noise to be increased in the position axis to 7.5 meters. In Figure 4.11 the $^1/_{64}{}^\circ$ simulation is shown, where it is seen that the most notable change is that the "wobble" seen in the $^1/_4{}^\circ$ and $^1/_{16}{}^\circ$ DEMs is reduced by comparison. This is due to the increase in tuning noise that works to stabilize the filtering solution. This noise allows the filter to acquire an accurate solution by minimizing the effects of the terrain, and as this noise is increased, the "wobble" previously found is less likely to appear as the terrain effects are reduced. The residual is also of note due to the increase seen to the single run performance compared to the Monte Carlo performance from the larger tuning noise. These same trends of needing a larger tuning noise and the effects of that are also seen when using the $^1/_{128}{}^\circ$ DEM, shown in Figure 4.12. Here an increase in tuning noise is again needed in the position axis, bringing the noise required to 15 meters. The main difference with this higher fidelity is that the altimeter residual single run performance increases even further compared to the Monte Carlo performance due to that tuning noise increase. While these higher fidelities still produce filters that can converge on a solution, the large tuning noises used are not ideal. As fidelities within the components of the filter are varied moving forward it will be of interest to see if these issues can be smoothed out by using a combination of higher and lower fidelity DEMs instead of large tuning noises.

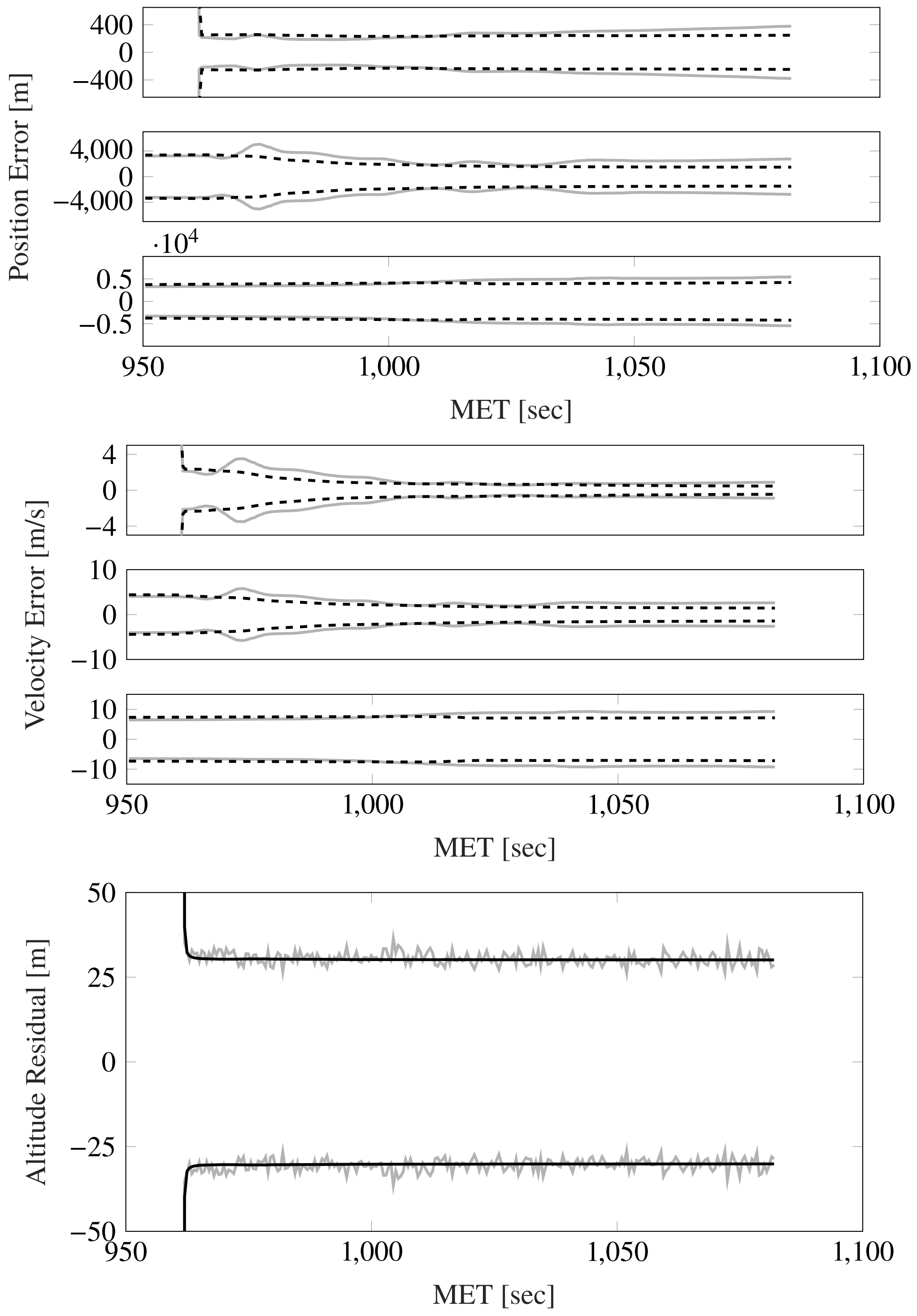


Figure 4.8. Position (top), velocity (middle), and altimeter residual (bottom) 3σ intervals for the Monte Carlo simulation (solid gray) and a single filter run (solid black) when using the $^1/_4{}^\circ$ DEM.

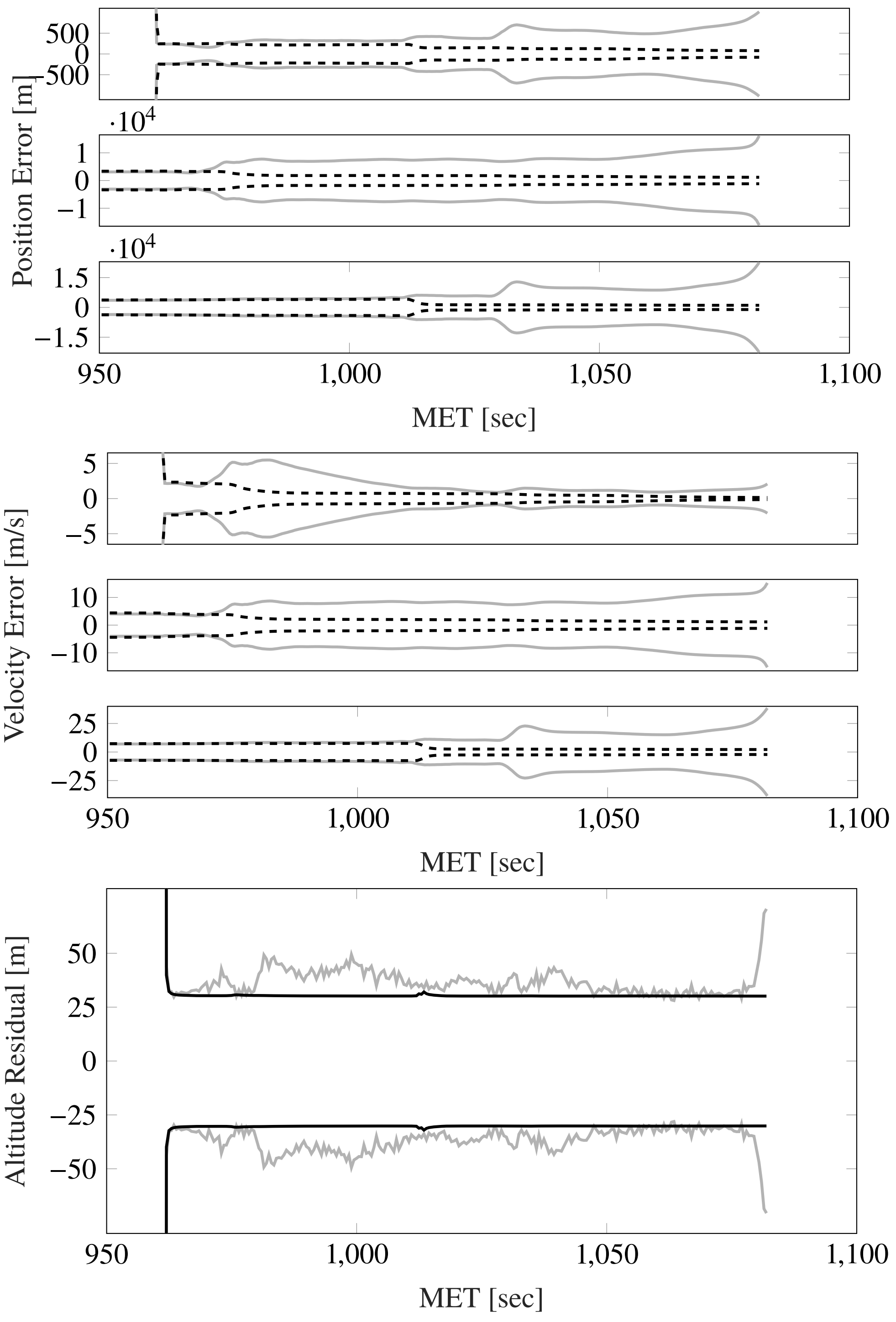


Figure 4.9. Position (top), velocity (middle), and altimeter residual (bottom) 3σ intervals for the Monte Carlo simulation (solid gray) and a single filter run (solid black) when using the $^{1}/_{16}{}^{\circ}$ DEM.

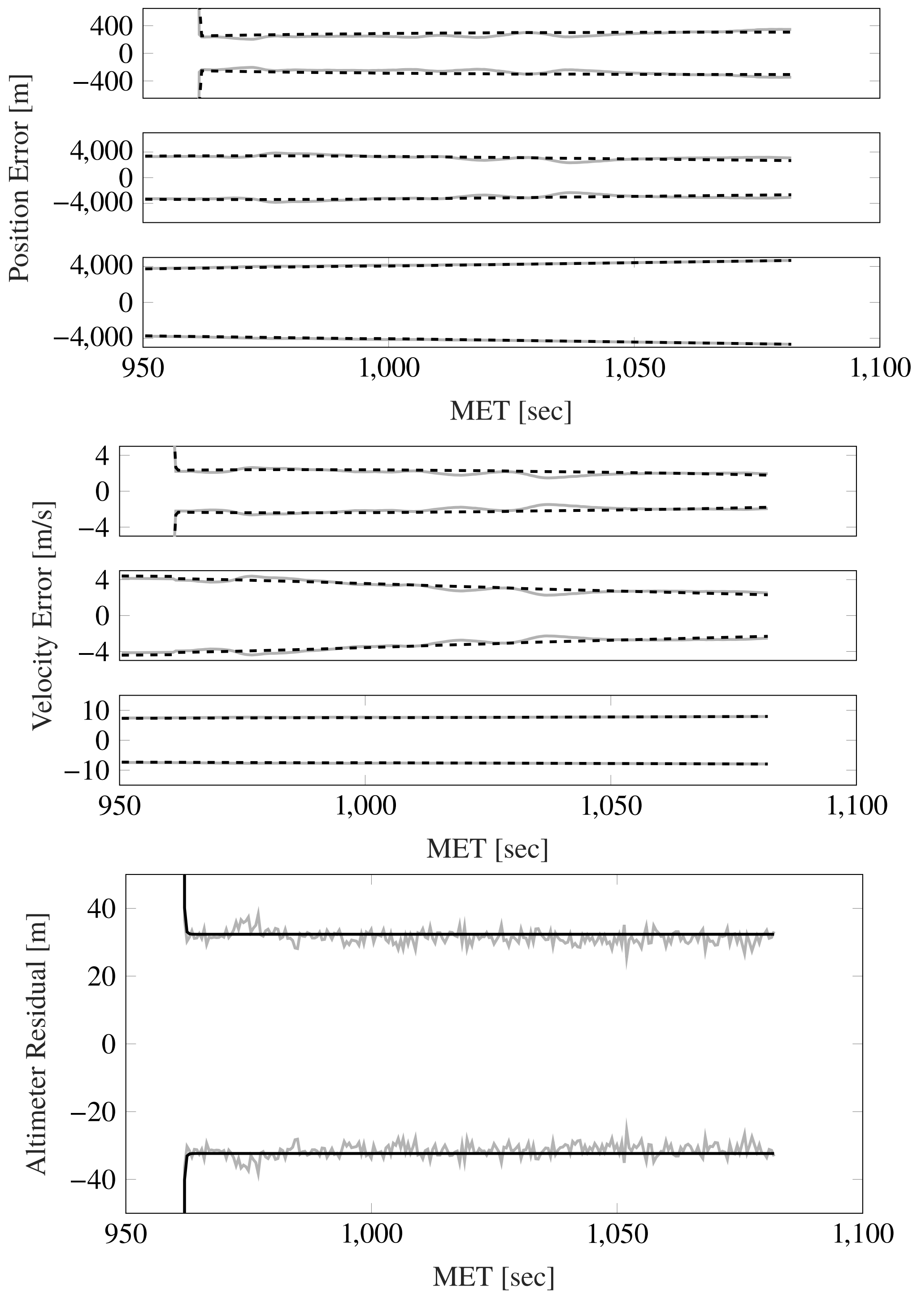
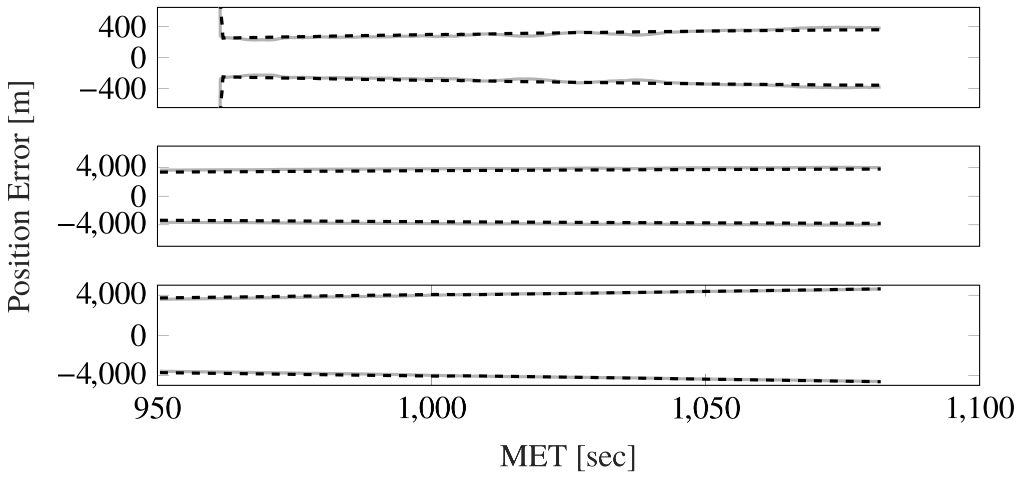
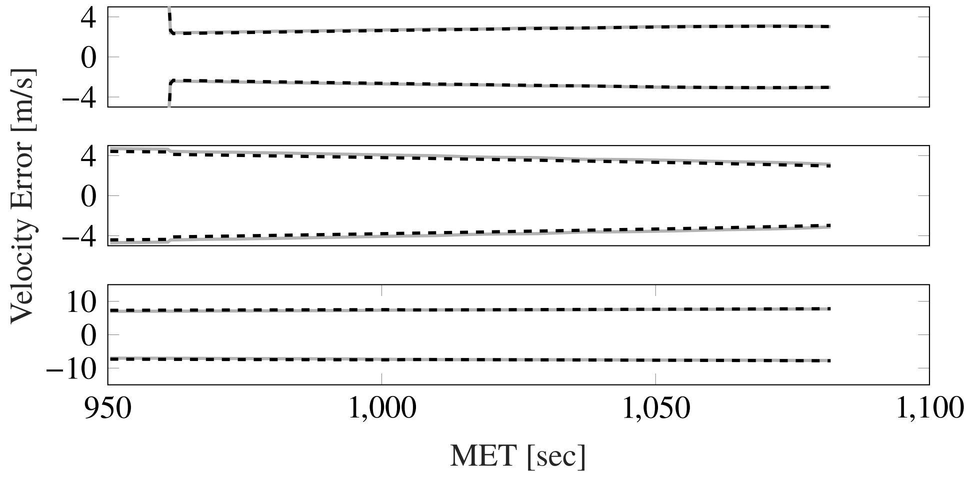


Figure 4.10. Position (top), velocity (middle), and altimeter residual (bottom) 3σ intervals for the Monte Carlo simulation (solid gray) and a single filter run (solid black) when using the $^1/_{16}{}^\circ$ DEM.





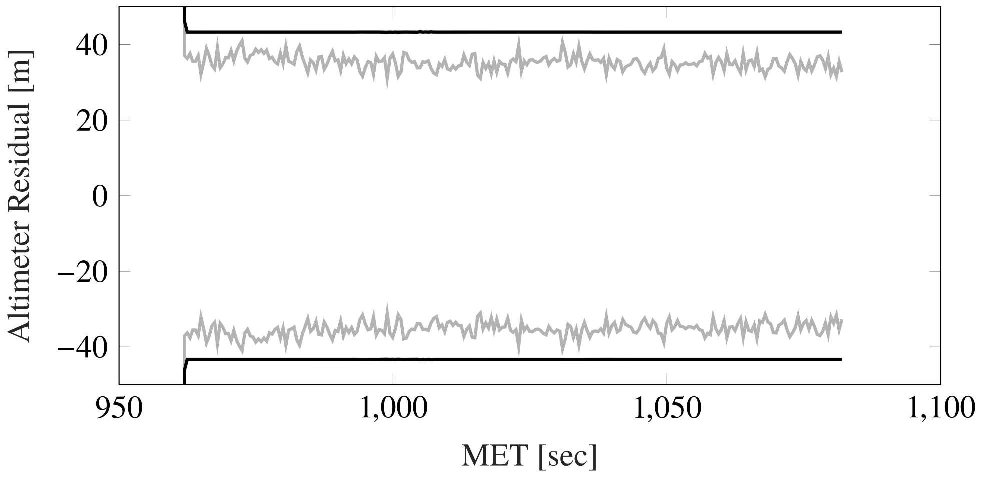


Figure 4.11. Position (top), velocity (middle), and altimeter residual (bottom) 3σ intervals for the Monte Carlo simulation (solid gray) and a single filter run (solid black) when using the $^1/_{64}{}^\circ$ DEM.

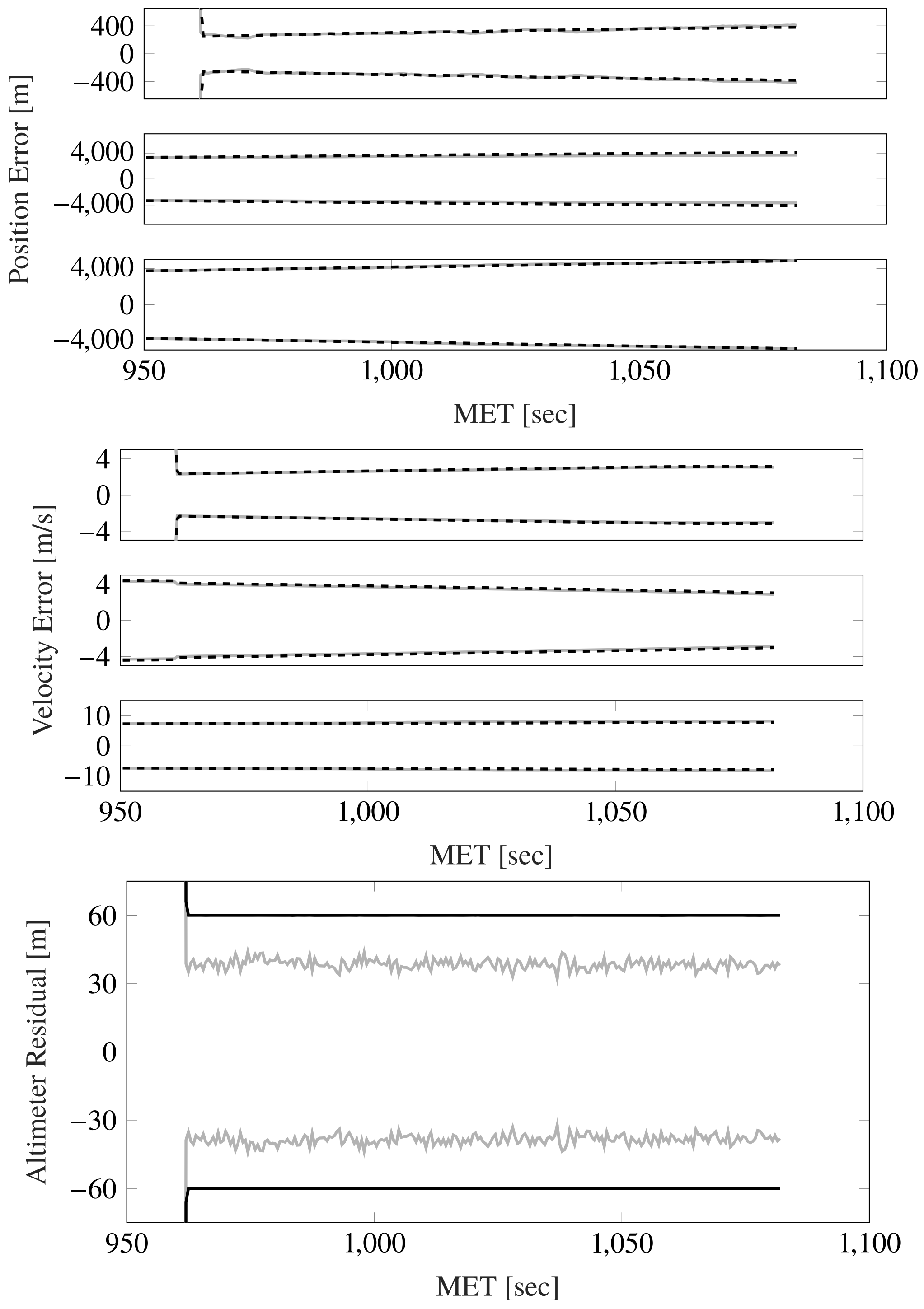


Figure 4.12. Position (top), velocity (middle), and altimeter residual (bottom) 3σ intervals for the Monte Carlo simulation (solid gray) and a single filter run (solid black) when using the $^1/_{128}{}^\circ$ DEM.

4.2. VARYING FIDELITIES

While higher fidelity models are always preferred for determining true measurements, because they better represent the actual surface of the planet, the estimated measurement and measurement Jacobian can use different models to test the resulting effects. A benefit of using these lower fidelity models is that of computational savings due to the simpler model being used. As the problem of entry, descent, and landing takes place in real time, it can often be difficult to get all navigation systems to run on the same computational timing. By providing this flexibility in computational complexity the altimeter sensor models could be more easily adjusted to run within the strict requirements of a real time navigation setting. To investigate the impact of disparate fidelities, different models are implemented within the components (truth, estimate, and Jacobian) to examine their effects. These varying combinations can be seen in Table 4.2 and will be subsequently referenced by their associated configuration number.

Configurations #1 and #2 leverage the same truth and estimate models, but changes the fidelity used within the Jacobian of the filter. While this is unusual, as it is often expected that the estimate and its Jacobian are of the same fidelity, the intent behind this is that the simpler model of the Jacobian may be able to smooth some of the effects introduced by the DEMs. Implementing the Jacobian with a spherical model gives a representation of how the simplest computational model will affect the filter when being used with a higher fidelity estimate. The results of Config. #1 are summarized in Figure 4.13. Comparing this to the $1/4^\circ$ DEM for all components of the filter (shown in Figure 4.8), the differences are minute. The main difference is that the slight growth in errors along the z axis for the Monte Carlo simulation when compared to the single run performance are now diminished. This overall lack of difference between the two makes this configuration a suitable replacement that has the added benefit of better computational efficiency in the filter's Jacobian model. The residual for this configuration is similar to that of identical fidelities with no differences of note from the residual of the filter with a $1/4^\circ$ measurement Jacobian.

Table 4.2. Configuration definitions for testing varying fidelity.

Configuration	Truth Model	Filter Estimate Model	Jacobian Model
#1	$1/4°$ DEM	$1/4°$ DEM	Spherical
#2	$1/16°$ DEM	$1/16°$ DEM	Spherical
#3	$1/16°$ DEM	$1/4°$ DEM	Spherical
#4	$1/128°$ DEM	$1/4°$ DEM	Spherical

The results of Configuration #2 are shown in Figure 4.14, where the effects of the spherical Jacobian are examined with the $1/16°$ DEM truth and estimate with the same tuning noise previously used for identical fidelities with the $1/16°$ DEM. This has similar results to those of Figure 4.9, where again little difference between the two is noticed. While Configuration #1 showed some slight improvement with the spherical Jacobian, when used with the necessary tuning noise of the $1/16°$ truth and estimate the introduction of a spherical Jacobian brings no noticeable change. When comparing the residuals of the altimeter measurements the same indifference is seen between the use of the higher and lower fidelity Jacobians. This again supports the claim that a lower fidelity Jacobian could be used within the navigation filter to provide a solution of similar accuracy that is more computationally efficient.

Moving to Configuration #3, it is of interest to see how the performance of the navigation filter is affected when the truth, estimate, and Jacobian are all different from one another. While this can add further computational savings, it is also of interest to see how a lower fidelity estimate could possibly provide benefits by using a fidelity that has less terrain variations. Since the $1/4°$ DEM is used for the estimate, the tuning noise is reduced from what has been previously implemented with a $1/16°$ truth. The introduction of this lower fidelity estimate requires a tuning noise in the position of only 0.5 meters to be used. Figure 4.15 illustrates the results of Configuration #3, and here it is seen that the effects of the lower fidelity estimate seems to counteract the effects of the higher fidelity truth, making for a smoother convergence. This along with the lower tuning noise needed

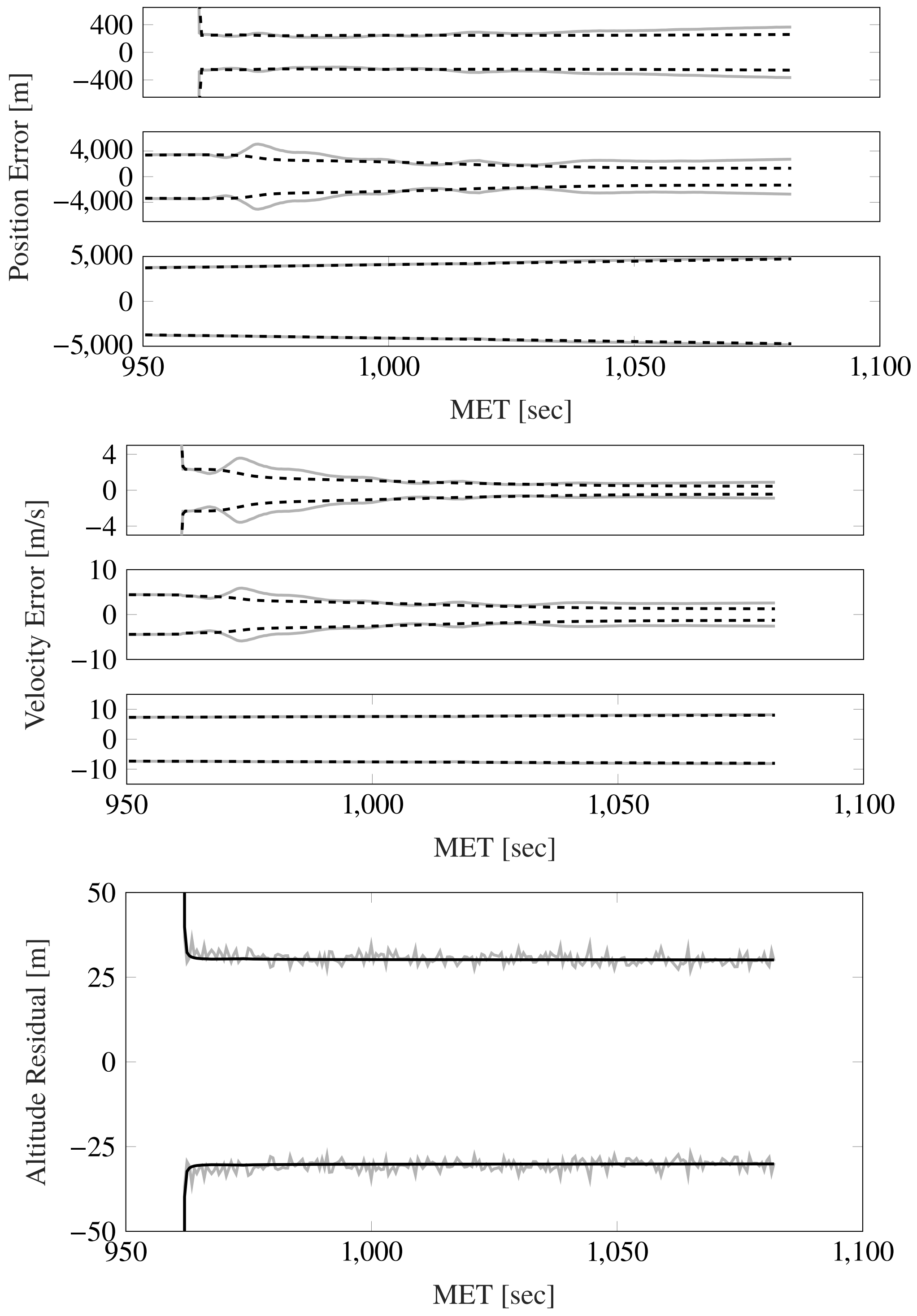


Figure 4.13. Position (top), velocity (middle), and altimeter residual (bottom) 3σ intervals for the Monte Carlo simulation (solid gray) and a single filter run (solid black) when using the $^{1}/_{4}^{\circ}$ DEM for truth and estimate and the spherical Jacobian.

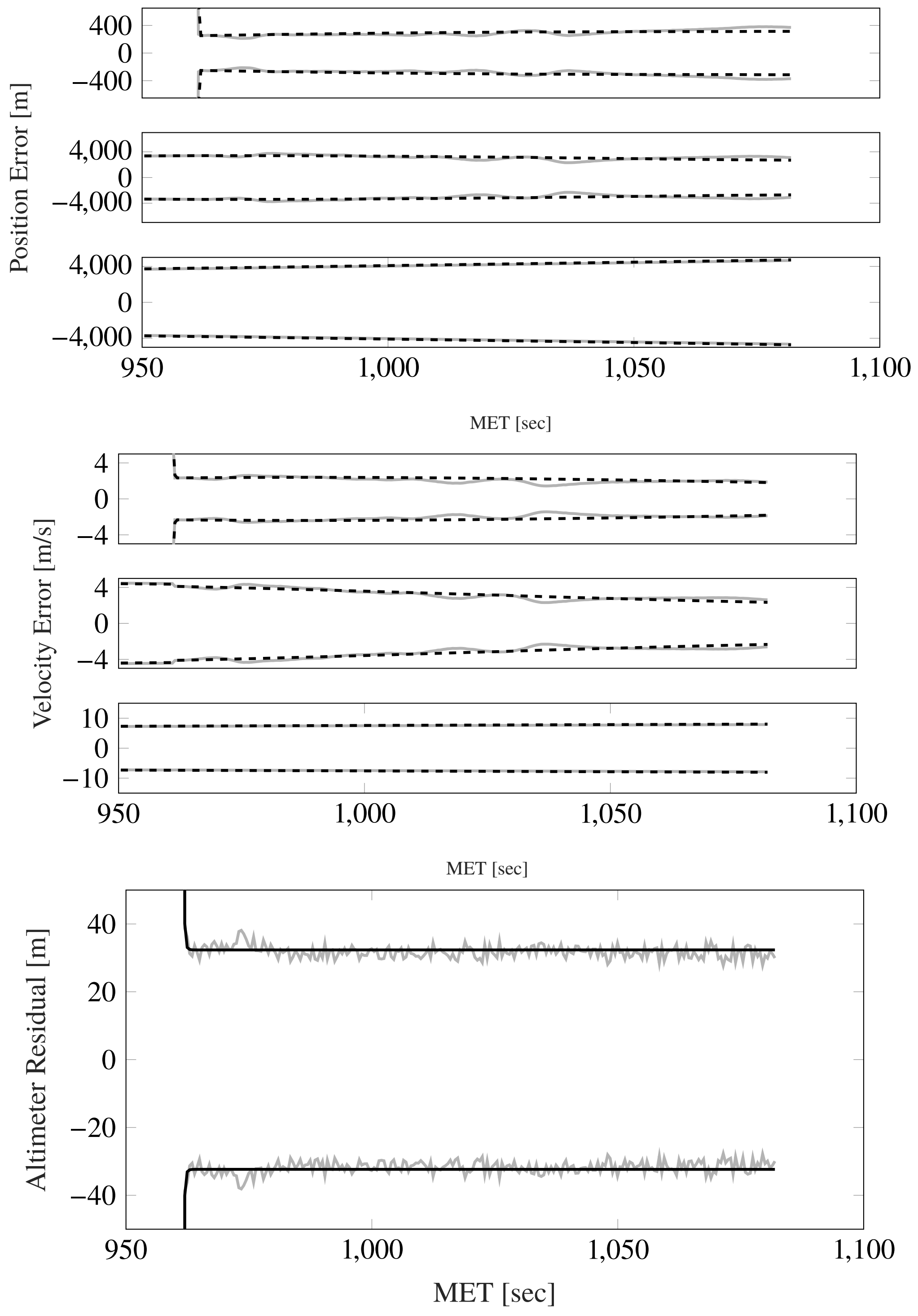


Figure 4.14. Position (top), velocity (middle), and altimeter residual (bottom) 3σ intervals for the Monte Carlo simulation (solid gray) and a single filter run (solid black) when using the $^1/_{16}{}^\circ$ DEM for truth and estimate and the spherical Jacobian.

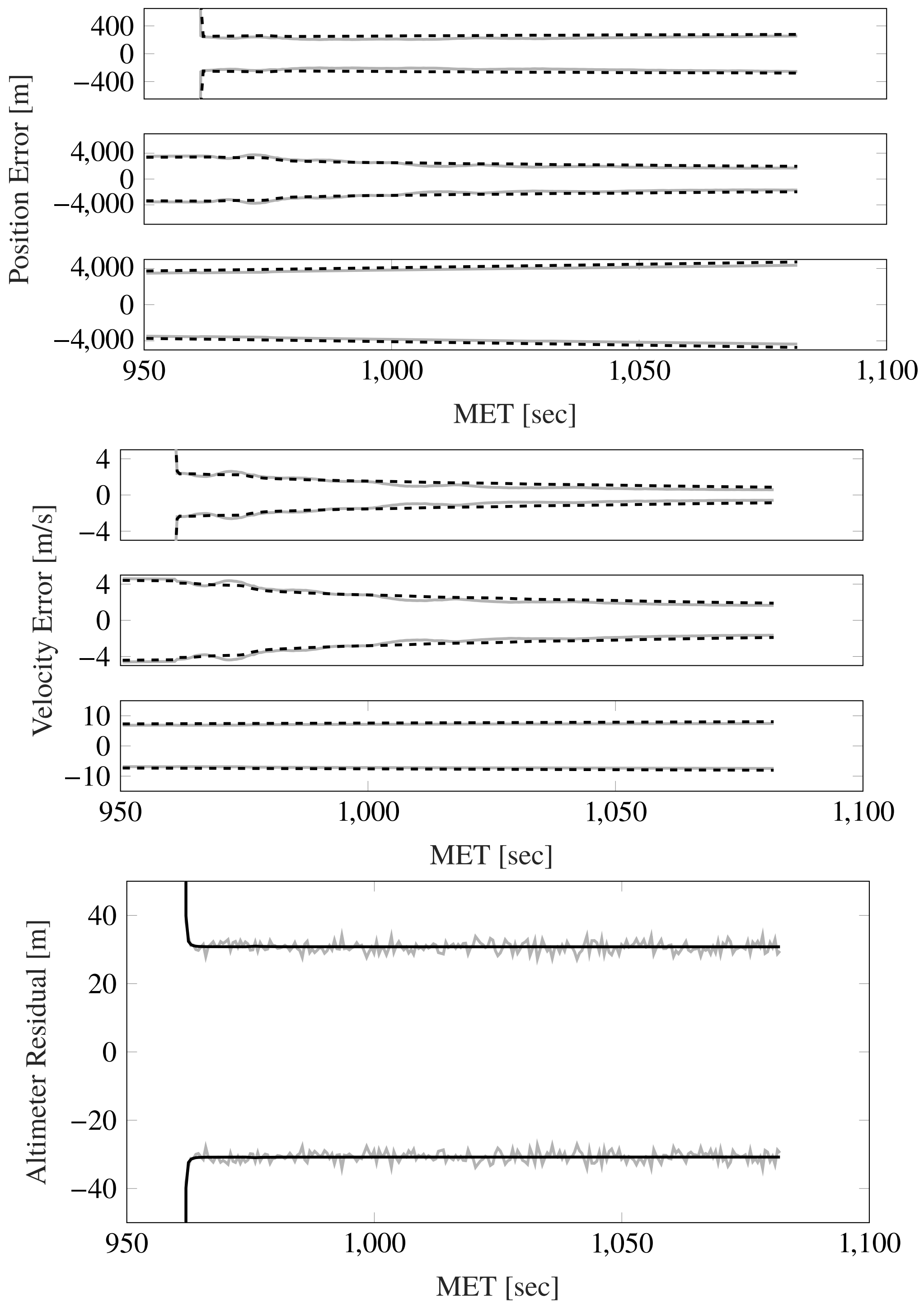


Figure 4.15. Position (top), velocity (middle), and altimeter residual (bottom) 3σ intervals for the Monte Carlo simulation (solid gray) and a single filter run (solid black) when using the $^{1}/_{16}{}^{\circ}$ DEM for truth, $^{1}/_{4}{}^{\circ}$ DEM for the estimate and a spherical Jacobian.

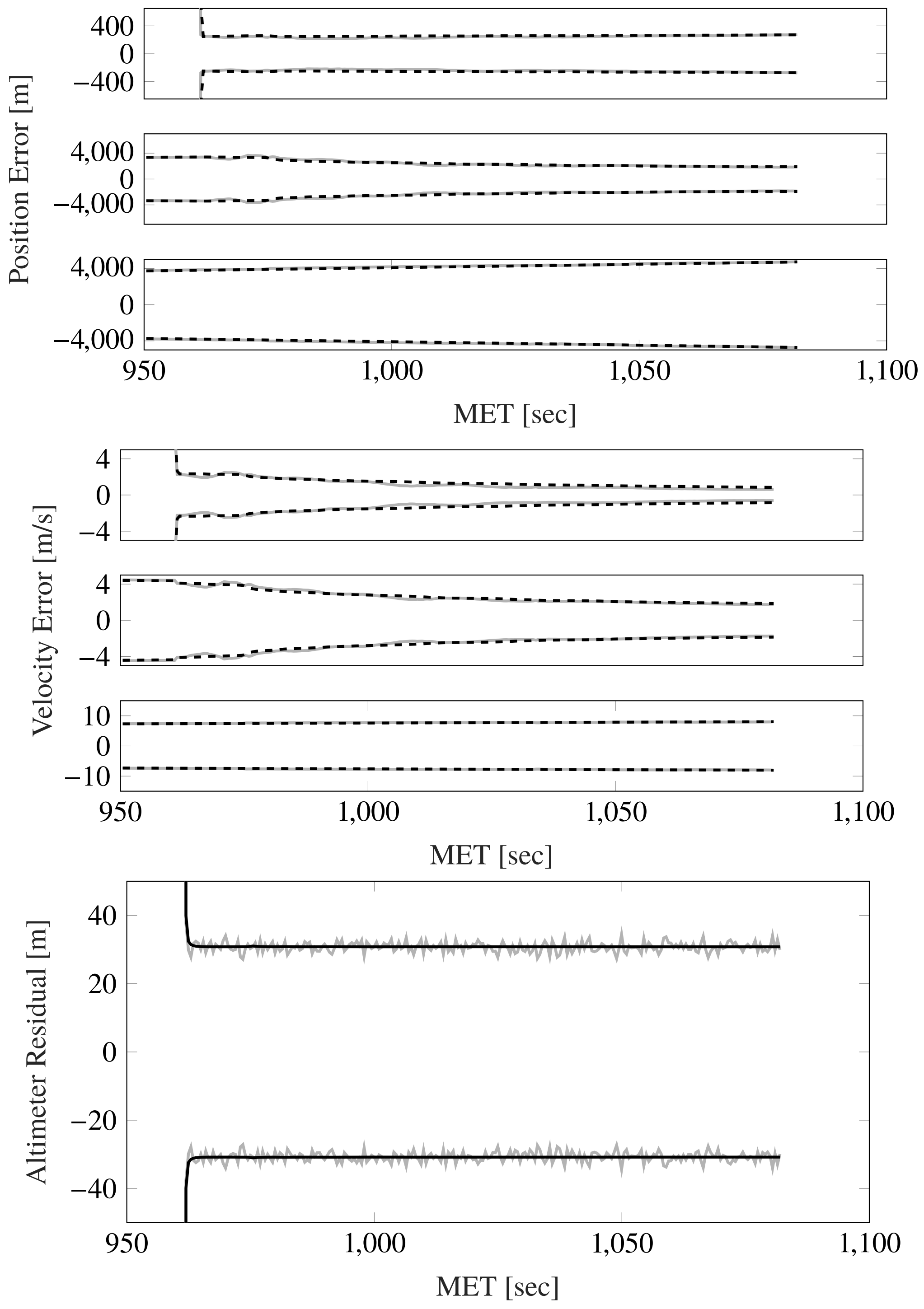


Figure 4.16. Position (top), velocity (middle), and altimeter residual (bottom) 3σ intervals for the Monte Carlo simulation (solid gray) and a single filter run (solid black) when using the $^{1}/_{16}{}^{\circ}$ DEM for truth, $^{1}/_{4}{}^{\circ}$ DEM for the estimate and a spherical Jacobian.

make the filtering solution slightly more accurate. The solution errors in the y position axis and x and y velocity axis continue to converge as the descent continues through the trajectory. While the z axis sees the Monte Carlo simulation fall slightly outside the single run performance, it is not different enough to be of concern and is still very stable as the filter tracks that axis. Both the position and velocity appear to be more stable than previously seen configurations with a $^1/_{16}{}^\circ$ truth, as the lower degree estimate smooths the effects of the large variations. This, along with its slightly more accurate solution, would indicate that using a lower fidelity estimate is not only a viable option, but a beneficial one in working to keep a more stable filter.

Having demonstrated that a lower fidelity estimate is capable of providing a "smoothing effect" for a higher fidelity truth, it is of interest to see how this performs when the highest fidelity, $^1/_{128}{}^\circ$, truth is used with the lowest fidelity DEM, $^1/_4{}^\circ$, estimate. This is the largest difference for the global DEMs and provides a look at how large changes in the variation between the truth and estimate may affect the filter. Configuration #4 is shown in Figure 4.16, where the performance is very similar to that of Configuration #3. The effects of the $^1/_4{}^\circ$ estimate on the much higher fidelity truth work again to smooth out the issues that could be seen with the higher terrain variations. Whereas an increase in truth fidelity also corresponded to an increase in tuning noise necessary to achive good filter performance in previous cases, here the tuning noise used does not change from Configuration #3. Changing the estimate to a lower fidelity allows the filter to smooth the issues of the more accurate DEMs, while also bringing down the necessary tuning noise, making them a preferable option when using a higher fidelity truth within the navigation filter.

The DEMs used within the SR-MEKF framework are able to perform as intended, given some assistance. While the $^1/_4{}^\circ$ fidelity is able to work to some level of success without any tuning noise, for any higher fidelities it is necessary. Given the introduction of this tuning noise, it is seen that the navigation filter is able to take advantage of these higher fidelity models. Even when a higher fidelity truth is used with a much lower fidelity estimate the resulting filter is one that works as is expected when comparing the Monte Carlo simulations to the single run performance.

Going forward, it is still of interest to see what alternative methods could be used to improve performance within the filter. By reducing the uncertainty at the time altimeter measurements begin, the filter may be able to better converge on a solution. It is also true that, while the linearization of the EKF works here, it could be beneficial to use a nonlinear transformation to approximate the measurement. This could give the filter a better idea of the terrain it is trying to converge on and improve the performance. Even though the filter as it is presented here works within the expected requirements, these questions still give reason for further investigation.

5. ALTERNATIVE METHODS

The conclusion resulting from filtering with the SR-MEKF is that as the fidelity of the models better represents the real terrain of the surface, the filter finds it increasingly difficult to properly track along the variations in the terrain. The increase in the rate the terrain elevation changes below the spacecraft makes it more difficult for the filter to converge on a solution without subsequent error growth. It is then of interest to find ways that mitigate these problems in a manner that better allows the filter to converge toward, and maintain, the true solution. This can be done by either reducing uncertainty to the filter, giving it more confidence in its solution, or by allowing the filter to have a better understanding of the area over which it is filtering. The subsequent sections show how to develop such methods that will manipulate the filter to allow for these events to occur.

5.1. UNCERTAINTY REDUCTION

Large uncertainties in the filters state can make it easier for the filter to diverge from the correct solution. If the filter starts with lower uncertainty in its covariance it should be able to better track the spacecraft's position and velocity above the surface. By reducing this uncertainty such that the filter almost knows exactly where it is initially the errors produced should also decrease in size, given the filter's new confidence in its solution. Doing this provides the filter with a smaller area of uncertainty so that it can better understand its current location above the surface. While reducing uncertainties in such a manner in not viable in an actual EDL scenario, it does provide an example of how the filter may perform with a reduction to its uncertainty. Doing this allows confirmation that when the filter is confident of its position and velocity it is able to reliably track them against the variations of higher fidelities as expected.

5.2. INITIATION ALTITUDE

A minor change that can be made to the altimeter sensor is to alter when it is first initiated to acquire measurement data. This is a way to decrease the amount of uncertainty built up by the filter at the time the altimeter starts taking measurements by shortening the amount of time for which uncertainty is propagated without position or velocity related measurements. While the growth of the uncertainty to that point is unchanged, the amount it has grown when the altimeter is initiated will be less. This should allow the filter to more quickly and easily converge to a solution. While this may not represent the specifications of the sensors used in a real world scenario, it can provide insight to how the filter performance may change if future advancements could allow for this kind of alteration.

5.3. UNSCENTED TRANSFORM

Since the EKF and EKF-based filters are constructed by linearizing nonlinear functions, it is of interest to investigate a method that does not rely on linearization, to assess if the aforementioned issues can be alleviated. In order to do that within the SR-MEKF, an unscented transform (UT) [30] is implemented within the filter update when altimeter data is processed. The UT allows the filter to have a better understanding of the nonlinear terrain below it by incorporating the filter's uncertainty through the use of sigma points that capture the first and second moments of the function to create a distribution to approximate a measurement. This estimated measurement is then used to find both the measurement and cross covariances. Doing this allows the filter to perform without the measurement Jacobian that previously described the deviation between the true measurement and its estimate. While this simplifies simulations by taking out one of the three variable components, the real benefit is that this allows the filter to compute its estimate and covariances with a better understanding of the nonlinear terrain over which the vehicle is traversing. To construct this update the formulation of the UT must be defined for use within the filter. Given the nonlinear function

$$z = \boldsymbol{g}(\boldsymbol{x}) + \boldsymbol{v} \tag{5.1a}$$

where $\boldsymbol{x}$ is a random input with mean $\boldsymbol{m}$ and covariance $\boldsymbol{P}$, the UT seeks to approximate the mean, covariance, and cross covariance (with the input $\boldsymbol{x}$) of z. The nonlinear function can be thought of as the altimeter model for the purpose of this discussion, but it is, in general, a nonlinear function that maps the random variable $\boldsymbol{x}$ into the random variable z.

Given the function $g(\boldsymbol{x})$ and the uncertainty of $\boldsymbol{x}$ through the distribution $p(\boldsymbol{x})$, the mean, covariance, and cross covariance of the estimate are

$$\begin{aligned}
\hat{z} &= \int \boldsymbol{g}(\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x} \\
\boldsymbol{W} &= \int (\boldsymbol{g}(\boldsymbol{x}) - \hat{\boldsymbol{x}})(\boldsymbol{g}(\boldsymbol{x}) - \hat{z})^T p(\boldsymbol{x})d\boldsymbol{x} + \boldsymbol{R} \\
\boldsymbol{C} &= \int (\boldsymbol{x} - \boldsymbol{m})(\boldsymbol{g}(\boldsymbol{x}) - \hat{z})^T p(\boldsymbol{x})d\boldsymbol{x} \,.
\end{aligned}$$

To approximate these integrals, sigma points are used to represent uncertainty using $2n + 1$ deterministically selected points, where n is the dimension of $\boldsymbol{x}$. Given the mean and covariance of $\boldsymbol{x}$ as $\boldsymbol{m}$ and $\boldsymbol{P}$, respectively, the sigma points are

$$\begin{aligned}
\boldsymbol{\mathcal{X}}^{(0)} &= \boldsymbol{m} \\
\boldsymbol{\mathcal{X}}^{(i)} &= \boldsymbol{m} + \sqrt{n + \lambda}\boldsymbol{S}_i \\
\boldsymbol{\mathcal{X}}^{(i+n)} &= \boldsymbol{m} - \sqrt{n + \lambda}\boldsymbol{S}_i
\end{aligned}$$

where i iterates through the $2n+1$ points and λ is a scaling parameter given by $\alpha^2(n+\kappa)-n$ with α and κ being parameters designed to determine the spread of the sigma points around the mean. Each of the sigma points developed here has a weight attached [13] where

$$
\begin{aligned}
\boldsymbol{w}_m^{(0)} &= \frac{\lambda}{n+\lambda} \\
\boldsymbol{w}_c^{(0)} &= \frac{\lambda}{n+\lambda} + (1 - \alpha^2 + \beta) \\
\boldsymbol{w}_m^{(i)} &= \frac{1}{2(n+\lambda)} \\
\boldsymbol{w}_c^{(i)} &= \frac{1}{2(n+\lambda)}
\end{aligned}
$$

with $\boldsymbol{w}_m$ being the weighs of the mean, $\boldsymbol{w}_c$ being the weights of the covariances, and β is a parameter that can be used for incorporating prior information of $\boldsymbol{x}$. The sigma points are transformed through the nonlinear function such that

$$
\mathcal{Z}^{(i)} = \boldsymbol{g}(\boldsymbol{\mathcal{X}}^{(i)}) \qquad \forall\, i \in \{0, 1, \ldots, 2n\}
$$

where $\boldsymbol{g}(\cdot)$ can be thought of as the altimeter measurement model. Using the weights, sigma points, and the transformed sigma points the mean, covariance, and cross covariance (with $\boldsymbol{x}$) of z are approximated by the UT as

$$
\begin{aligned}
\hat{z} &= \sum_{i=0}^{2n} w_m^{(i)} \mathcal{Z}^{(i)} \\
\boldsymbol{W} &= \sum_{i=0}^{2n} w_c^{(i)} (\mathcal{Z}^{(i)} - \hat{z})(\mathcal{Z}^{(i)} - \hat{z})^T + \boldsymbol{R} \\
\boldsymbol{C} &= \sum_{i=0}^{2n} w_c^{(i)} (\boldsymbol{\mathcal{X}}^{(i)} - \boldsymbol{m})(\mathcal{Z}^{(i)} - \hat{z})^T \,.
\end{aligned}
$$

This now allows the altimeter measurement estimate to be updated via a nonlinear method that provides the filter with a better understanding of the terrain below it. By taking the $2n + 1$ sigma points and transforming them via the nonlinear function the filter is now able to understand the measurement, not only at its location, but also at various points around its area of uncertainty. This increased understanding the measurement is able to obtain allows for a filter that can better estimate its state given a highly nonlinear environment, such as lunar terrain.

5.4. MONTE CARLO INTEGRATION

As a best case scenario where thousands of sample points can be taken, Monte Carlo integration [4] can be performed within the altimeter update [11]. Doing this in the update works similar to the UT in that several points within the area of uncertainty are used in the construction of the filter update. While the UT uses $2n + 1$ sigma points, a common rule of thumb is that Monte Carlo sampling should use 10^n samples; however, this is unfortunately unobtainable since it would require 10^9 samples for position, velocity, and attitude states alone, which is not practical within the confines of a navigation filter. Instead, several hundred thousand samples are used to estimate the measurement, measurement covariance, and cross covariance as in the UT. While using this method, it has been observed here that the filter develops issues resulting from numerical calculations that force a failure at the Cholesky downdate of the square-root filter. To circumvent this problem, the original development of the square-root update by Potter is derived here, and largely follows the developments in Grewal and Andrews [9].

The Potter square-root covariance update is utilized for scalar measurements, which is valid in the case of the altimeter update implemented here. This is constructed by first defining the covariance update as

$$\boldsymbol{P}_k^+ = \boldsymbol{P}_k^- - \boldsymbol{P}_k^- \boldsymbol{H}_k^T [\boldsymbol{H}_k \boldsymbol{P}_k^- \boldsymbol{H}_k^T + \boldsymbol{R}_k]^{-1} \boldsymbol{H}_k \boldsymbol{P}_k^- ,$$

where the *a priori* covariance is then broken into square root factors as in Eq. 2.37 such that the *a posteriori* covariance can be formulated as

$$\begin{aligned}
\boldsymbol{S}_k^+(\boldsymbol{S}_k^+)^T &= \boldsymbol{S}_k^-(\boldsymbol{S}_k^-)^T - \boldsymbol{S}_k^-(\boldsymbol{S}_k^-)^T \boldsymbol{H}_k^T [\boldsymbol{H}_k \boldsymbol{S}_k^-(\boldsymbol{S}_k^-)^T \boldsymbol{H}_k^T + \boldsymbol{R}_k]^{-1} \boldsymbol{H}_k \boldsymbol{S}_k^-(\boldsymbol{S}_k^-)^T \\
&= \boldsymbol{S}_k^-(\boldsymbol{S}_k^-)^T - \boldsymbol{S}_k^- \boldsymbol{\Lambda}_k [\boldsymbol{\Lambda}_k^T \boldsymbol{\Lambda}_k + \boldsymbol{R}_k]^{-1} \boldsymbol{\Lambda}_k^T (\boldsymbol{S}_k^-)^T \\
&= \boldsymbol{S}_k^- \{\boldsymbol{I} - \boldsymbol{\Lambda}_k [\boldsymbol{\Lambda}_k^T \boldsymbol{\Lambda}_k + \boldsymbol{R}_k]^{-1} \boldsymbol{\Lambda}_k^T\} (\boldsymbol{S}_k^-)^T .
\end{aligned}$$

Here, $\boldsymbol{I}$ is an identity matrix that is the same size as the length of the state vector, and $\boldsymbol{\Lambda}_k$ is defined using Eq. 2.40 as

$$\boldsymbol{\Lambda}_k = (\boldsymbol{S}_k^-)^T \boldsymbol{H}_k^T = (\boldsymbol{S}_k^-)^{-1} \boldsymbol{C}_k \,.$$

When scalar measurements are considered, $\boldsymbol{\Lambda}_k$ becomes a vector, which is expressed as $\boldsymbol{\lambda}_k$, and it can be shown that

$$\boldsymbol{I} - \boldsymbol{\Lambda}_k[\boldsymbol{\Lambda}_k^T \boldsymbol{\Lambda}_k + \boldsymbol{R}_k]^{-1} \boldsymbol{\Lambda}_k^T = \boldsymbol{I} - \frac{\boldsymbol{\lambda}_k \boldsymbol{\lambda}_k^T}{R_k + ||\boldsymbol{\lambda}_k||^2} \,.$$

Defining a term s as

$$s = \frac{1}{R_k + ||\boldsymbol{\lambda}_k||^2} \,,$$

it follows that

$$\boldsymbol{I} - \frac{\boldsymbol{\lambda}_k \boldsymbol{\lambda}_k^T}{R_k + ||\boldsymbol{\lambda}_k||^2} = \boldsymbol{I} - s \boldsymbol{\lambda}_k \boldsymbol{\lambda}_k^T \,,$$

which can then be broken into square-root factors of the form

$$\boldsymbol{I} - s \boldsymbol{\lambda}_k \boldsymbol{\lambda}_k^T = (\boldsymbol{I} - \sigma \boldsymbol{\lambda}_k \boldsymbol{\lambda}_k^T)(\boldsymbol{I} - \sigma \boldsymbol{\lambda}_k \boldsymbol{\lambda}_k^T)^T \quad \text{where}$$
$$\sigma = \frac{1 + \sqrt{1 - s||\boldsymbol{\lambda}_k||^2}}{||\boldsymbol{\lambda}||^2} \,.$$

Now the *a posteriori* covariance can be expressed in terms of square-root factors yielding

$$\boldsymbol{S}_k^+ (\boldsymbol{S}_k^+)^T = \boldsymbol{S}_k^- (\boldsymbol{I} - \sigma \boldsymbol{\lambda}_k \boldsymbol{\lambda}_k^T)(\boldsymbol{I} - \sigma \boldsymbol{\lambda}_k \boldsymbol{\lambda}_k^T)^T (\boldsymbol{S}_k^-)^T \,,$$

from which it is clear that the *a posteriori* square root factor is

$$\boldsymbol{S}_k^+ = \boldsymbol{S}_k^-(\boldsymbol{I} - \sigma \boldsymbol{\lambda}_k \boldsymbol{\lambda}_k^T) \,.$$

This allows the *a posteriori* square-root factor to be computed without using a Cholesky downdate. This development is simpler in that it needs a scalar measurement, but for its use here, it is found to be more reliable and provides solutions similar to the Cholesky downdate procedure without being hindered by numerical issues.

To develop the Monte Carlo integration method, the expectations defined for the measurement, measurement covariance, and cross covariance are now evaluated as integral equations where

$$\begin{aligned}
\hat{z} &= \int \boldsymbol{h}(\boldsymbol{x}) p(\boldsymbol{x}) d\boldsymbol{x} \\
\boldsymbol{W} &= \int (\boldsymbol{h}(\boldsymbol{x}) - \hat{z})(\boldsymbol{h}(\boldsymbol{x}) - \hat{z})^T p(\boldsymbol{x}) d\boldsymbol{x} + \boldsymbol{R} \\
\boldsymbol{C} &= \int (\boldsymbol{x} - \boldsymbol{m}^-)(\boldsymbol{h}(\boldsymbol{x}) - \hat{z})^T p(\boldsymbol{x}) d\boldsymbol{x}
\end{aligned}$$

and are now all of the form

$$I = \int \boldsymbol{g}(\boldsymbol{x}) p(\boldsymbol{x}) d\boldsymbol{x} \,.$$

These integrals can be solved via Monte Carlo integration where

$$I \approx \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{g}(\boldsymbol{x}^{(i)})$$

and is constructed in a similar manner to the unscented transform. The main difference from the UT is that the integration now takes place over, ideally, 10^n different samples which are all equally weighted. To do this, random samples from a normal distribution are drawn according to

$$\mathcal{X}^{(i)} \simeq p_g(\boldsymbol{x}|\boldsymbol{m}, \boldsymbol{P}) .$$

These samples are then transformed though the nonlinear function; i.e.

$$\mathcal{Z}^{(i)} = \boldsymbol{g}(\mathcal{X}^{(i)}) .$$

The nonlinear function, $\boldsymbol{g}(\cdot)$, represents the altimeter measurement model in this case. The measurement estimate, its covariance and the cross covariance are then found as

$$\begin{aligned}
\hat{z} &= \frac{1}{N}\sum_{i=1}^{N} \mathcal{Z}^{(i)} \\
\boldsymbol{W} &= \frac{1}{N}\sum_{i=1}^{N} (\mathcal{Z}^{(i)} - \hat{z})(\mathcal{Z}^{(i)} - \hat{z})^T + \boldsymbol{R} \\
\boldsymbol{C} &= \frac{1}{N}\sum_{i=1}^{N} (\mathcal{X}^{(i)} - \boldsymbol{m})(\mathcal{Z}^{(i)} - \hat{z})^T .
\end{aligned}$$

This now provides a method of Monte Carlo sampling within the altimeter measurement estimate update. In order to gather an understanding of the terrain that is substantially greater than the UT, a large number of samples need to used. While 10^n is not a viable option given the typical dimensions of a navigation filter, using a hundred thousand points will allow the filter to gather significantly more information about the terrain when updating the estimate. Doing this can provide a “best case” scenario of how the filter can perform when it is able to have a good understanding of the terrain it is filtering over.

6. RESULTS OF ALTERNATIVE METHODS

With the alternative methods described, similar tests to the preliminary results are implemented. These tests are performed with similar specifications, including Monte Carlo simulations of 200 trials where the initial filter estimate and measurement noise are re-sampled with each trial. The same sensor specifications and uncertainties are also used, along with similar noise and bias models. The specifications are only altered when that is the governing theory behind the mitigation method, such as with uncertainty reduction or altimeter initiation altitude. A significant aspect of the mitigation methods considered compares the previous SR-MEKF to using a UT in the update of the SR-MEKF. Therefore, moving forward, the runs without the UT update will be referred to as EKF because that is the update method involved, and UT updated runs will be referred to as UT simulations.

6.1. REDUCED UNCERTAINTY

To observe how the terrain affects the filter when uncertainties are significantly reduced at the initial activation of the altimeter, a simulation is conducted where the trajectory starts when the altimeter turns on, such that state errors are at a minimum. To do this, the simulation begins at an altitude of 8 km, which is the same altitude at which the altimeter begins taking measurements. This simulation is also performed with very small uncertainties; as such, the initial covariance is taken to be

$$\boldsymbol{P}_0 = \text{diag}(10^2, 10^2, 10^2, 0.1^2, 0.1^2, 0.1^2, (0.572 \times 10^{-8})^2, (0.572 \times 10^{-8})^2, (0.572 \times 10^{-8})^2)$$

with units of m^2, (m/s)^2, and degrees^2 for position, velocity, and attitude, respectively. Otherwise, all other specifications are the same as in the preliminary results in Chapter 4.

Figure 6.1 shows the results for the reduced uncertainty simulation using the $^1/_{16}{}^\circ$ DEM. Starting the altimeter as soon as the filter initializes allows the filter to perform just as expected, even using the higher fidelity DEM with no tuning noise introduced to the system. The Monte Carlo simulation and the single run performance fall right on top of each other as they should, showing that the filter is performing as expected when all uncertainty is gone. What error growth is seen is due to the natural development of the uncertainty from the IMU and is not due to the altimeter measurements. The altimeter residuals also show a similar performance adding to the confidence in the filter and its solution with low uncertainties. While this is an artificial scenario that would never be seen in actual use, it shows that the filter can perform as expected when there is very little uncertainty in the measurement.

6.2. INITIATION ALTITUDE

As a more realistic example of reducing the uncertainty at the time that the altimeter starts taking measurements is to initiate the altimeter earlier in the descent trajectory. By doing this, the hope is that less, or no, tuning noise will be needed since the filter is more certain of its state from the point altimeter measurements are initially included, similar to the results from Chapter 6.1. To test this, a simulation is carried out with the altimeter starting at 12 km instead of the previous 8 km with $^1/_{16}{}^\circ$ fidelities. To see how this affects the need for process noise a simulation is done where

$$\boldsymbol{Q}_{Tune} = diag([1.5,\ 1.5,\ 1.5,\ 0.005,\ 0.005,\ 0.005]),$$

with a position tuning noise of 1.5 meter, as was used in the preliminary results, and a second with a reduced 1 meter of position tuning noise and the same velocity noise.

These two variations are shown in Figs. 6.2 and 6.3 for the 1 m noise and 1.5 m noise, respectively. Here it is seen that the simulation with 1.5 m of noise looks very similar to the previous simulations, which start at 8 km shown in Figure 4.10. There is

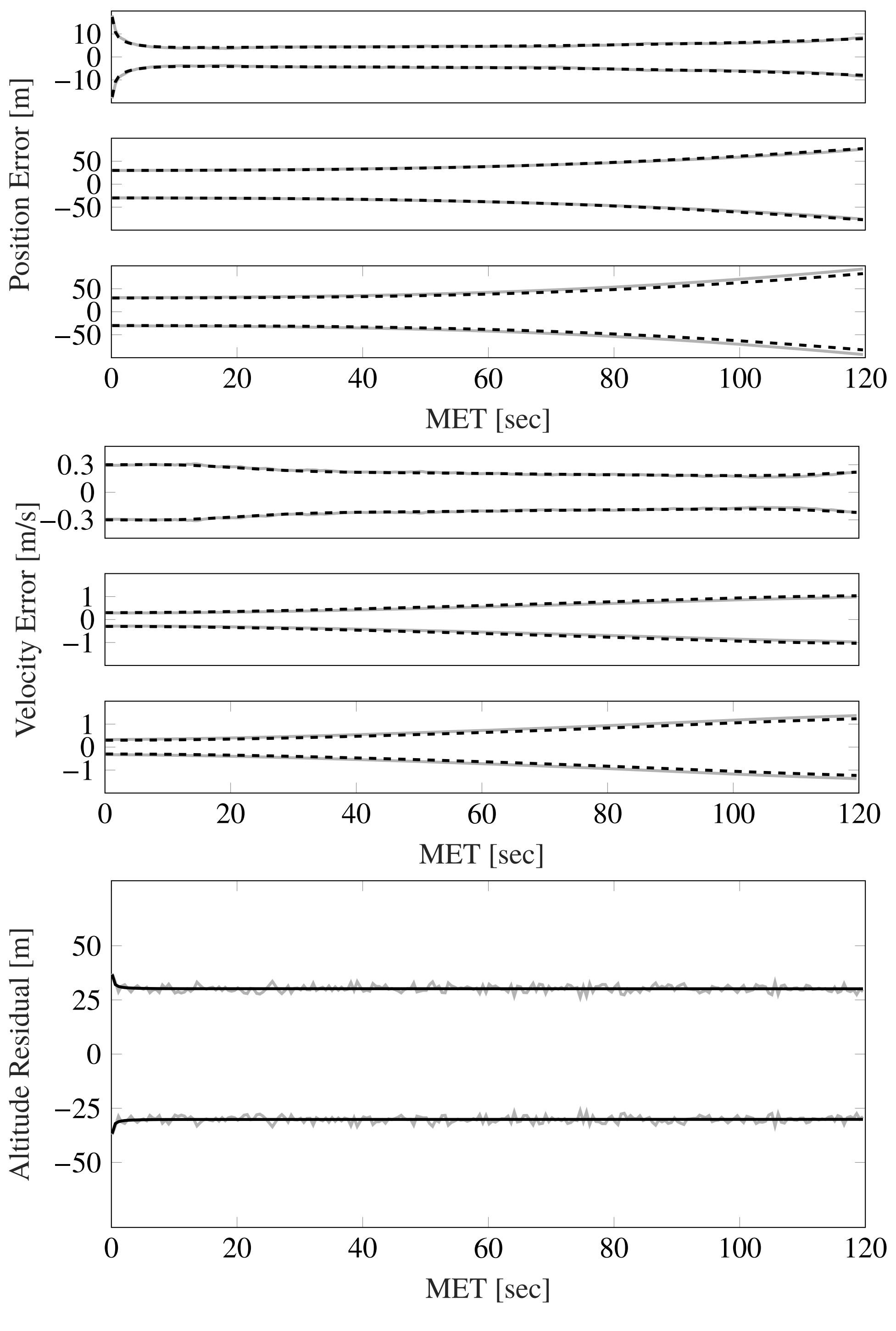


Figure 6.1. Position (top), velocity (middle), and altimeter residual (bottom) 3σ intervals for the Monte Carlo simulation (solid gray) and a single filter run (solid black) for $^1/_{16}{}^\circ$ truth, estimate and Jacobian starting at 8 km with minimal uncertainty.

little difference between them and it appears this reduction in the uncertainty at the time the altimeter initiates has little effect. The results of the simulation with 1 m of noise also seems to support the lack of effects produced by the reduced uncertainty. The position and velocity error intervals for the Monte Carlo both lose some stability and fall further away from the single run performance. They also appear to be more affected by the terrain due to a larger “wobble” in the Monte Carlo results. From this, it can be concluded that the small reduction in uncertainty gained by raising the altimeter initiation altitude by 4 km is not enough to positively influence the filter’s sensitivity to the terrain.

6.3. UNSCENTED TRANSFORM

Using an unscented transform (UT) allows the filter to do a nonlinear transformation that more accurately represent the surface it is tracking over by mapping first and second moments to it. This provides the filter with an update that, through the transformation, has a better understanding of the surface within its area of uncertainty. In theory, this should provide a more stable filter that has a larger understanding of the terrain variations on the surface in the area it is uncertain about. Since the sigma points of the UT stretch across its area of uncertainty the sudden change in terrain elevation should be better foreseen and overcome. The UT has several parameters that determine how the sigma points are selected and weighted. The specifications found to perform the best for the UT used here have an α of 1, a β of 2, and a κ of $3 - n$, providing the largest spread of sigma points.

The UT will be tested by first comparing the $^1/_4{}^\circ$ DEM for all filtering components in Figure 4.8 against the UT with a truth and estimate that have a $^1/_4{}^\circ$ fidelity. The position and velocity uncertainty intervals of the UT are given in Figure 6.4, where upon first comparison of these figures it is seen that the UT converges to a much more accurate solution compared to the EKF. In the instances of the x and z position axis and z velocity axis the UT continues to converge after the point that the EKF no longer does. Along with this, the Monte Carlo simulation is able to follow the expected single run performance much better with the UT in

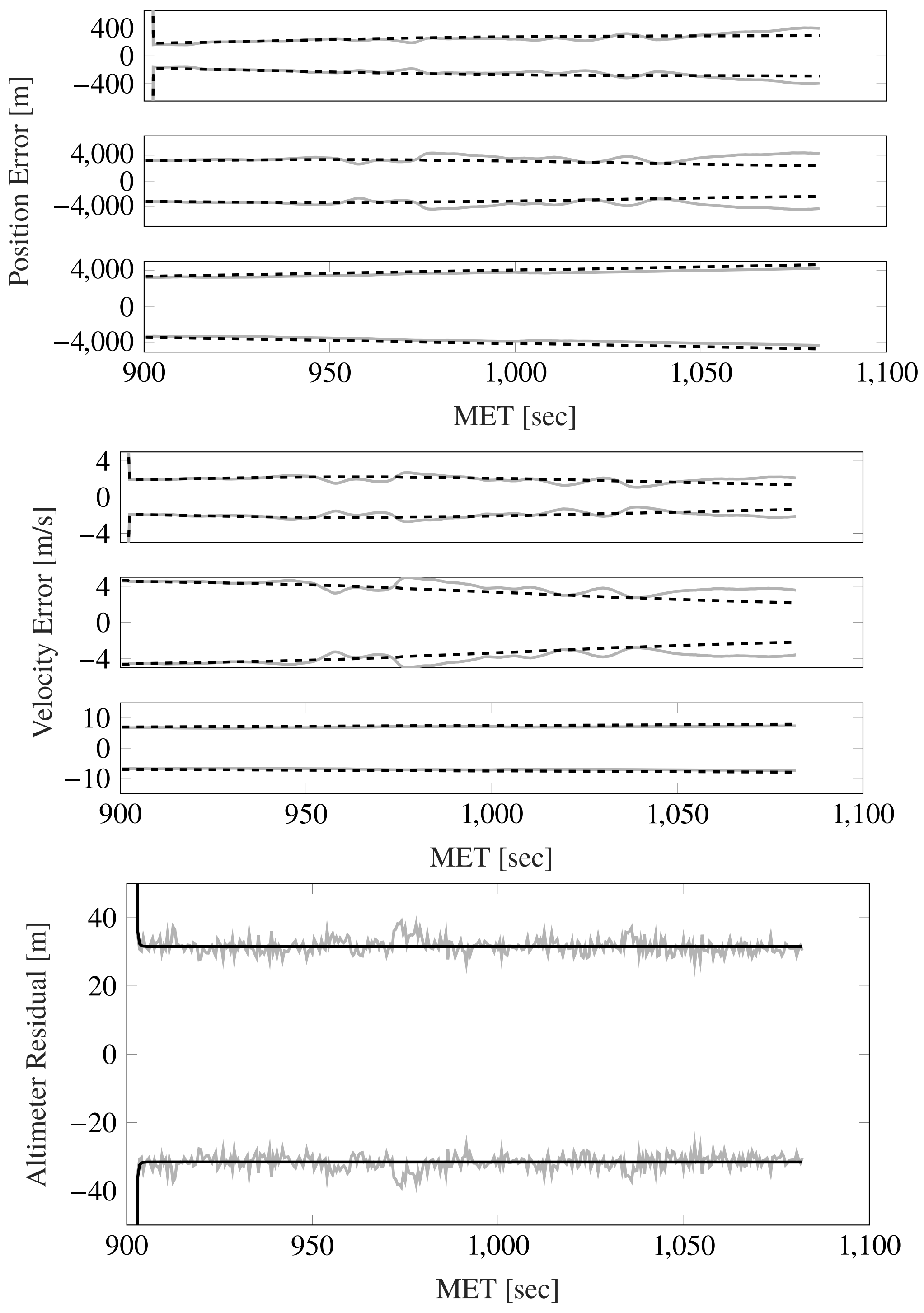


Figure 6.2. Position (top) and velocity (middle) error 3σ interval, and altimeter residual (bottom) 3σ intervals for the Monte Carlo simulation (solid gray) and a single filter run (solid black) when using the $^{1}/_{16}{}^{\circ}$ DEM for truth and estimate with 1 m of position tuning noise from 12 km.

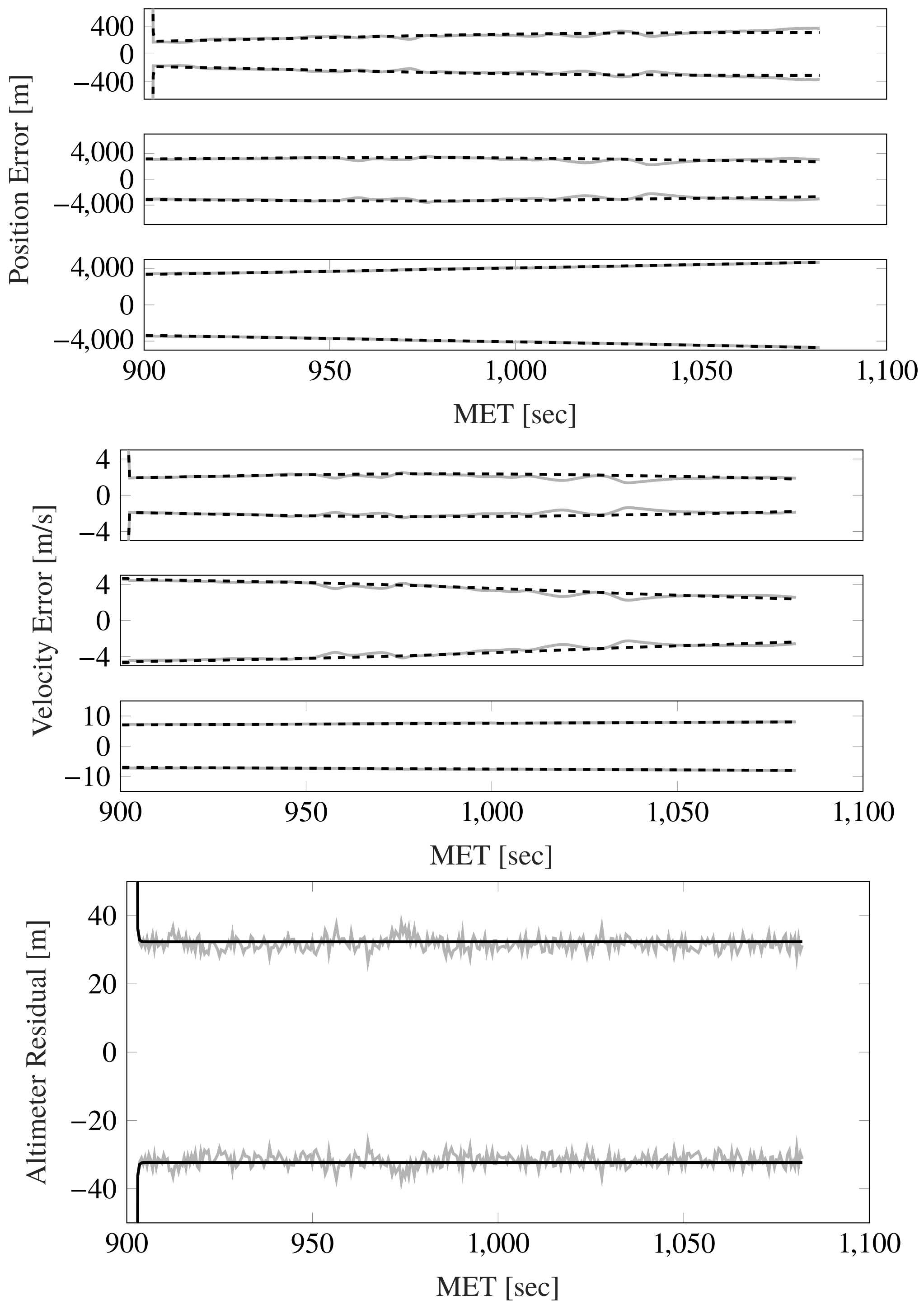


Figure 6.3. Position (top) and velocity (middle) error 3σ interval, and altimeter residual (bottom) 3σ intervals for the Monte Carlo simulation (solid gray) and a single filter run (solid black) when using the $^{1}/_{16}{}^{\circ}$ DEM for truth and estimate with 1.5 m of position tuning noise from 12 km.

place. While it still sits slightly outside the single run performance, it is much more steady and consistent compared to the EKF implementation. Given these results, it appears that the UT can produce a more accurate and stable solution than the EKF is able to given the lower fidelity terrain.

For the simulation of the $^1/_{16}{}^\circ$ DEM in the UT, the introduction of tuning noise is once again needed for the filter to perform as intended. While the $^1/_4{}^\circ$ fidelity was able to show good improvement with its more mild terrain variations, the jump to $^1/_{16}{}^\circ$ proved too unstable for the UT to converge to a proper solution. Attempts were also made to adjust the UT to better perform over the more erratic terrain by manipulating the α, β, and κ, values, but it ultimately proved unsuccessful in alleviating effects from the terrain. In order to allow the filter to perform using the increased fidelity, a tuning noise of 8.5 m in the position and 0.005 m/s in the velocity is used, which is a much larger increase compared to that used in the EKF. The performance for this implementation is shown in Figure 6.5 where the position, velocity, and residual intervals can be seen. While the use of the given tuning noise assists the filter in converging toward a solution, it also seems to reduce the benefits of the UT when compared to how the UT performed in Figure 6.4. The performance here shows a filter that, while the y axis continues to converge slightly as the descent continues, is nowhere near the benefits seen for the $^1/_4{}^\circ$ fidelity UT. Comparing this UT solution to the EKF, the first aspect of note is the altimeter residual interval where the effects of the nonlinear transformation and tuning noise can be best visualized. While the single run performance of the EKF is a stable, straight line once the altimeter is activated, the UT better shows the effects of the nonlinear terrain and the increased tuning noise by having a larger and more erratic single run performance. While the position and velocity error intervals themselves are similarly accurate relative to the EKF, the increased tuning noise and residual instability make it a less compelling option by comparison.

Moving to the $^1/_{64}{}^\circ$ DEM in the UT, the issues brought about by the increased complexities of the DEM only grow even further. The filter has a more difficult time given the amount of variations in the terrain. The tuning noise used needs to be increased significantly in the position to 25 m, while the velocity tuning noise remains 0.005 m/s. The first thing to notice in Figure 6.6 is the altimeter residual intervals, where the single run performance is even more erratic due to the increased terrain variations and further away from the Monte Carlo performance due to the increased tuning noise. Its performance is otherwise similar to the $^1/_{16}{}^\circ$ fidelity except that it falls outside the single run performance in the position and velocity as time progresses. While the Monte Carlo simulation maintains stability due to the large tuning noise that allows the filter to converge to a solution, that solution is not as accurate given the large amount of noise needed. This performance shows how difficult of a time the UT has when the higher fidelities models are used, while the $^1/_4{}^\circ$ model provides good improvement over the EKF, that changes as models with increased variations are introduced. Previously, the introduction of variation between filter components had provided benefits to the filter, so similar simulations will be considered to see their effects on the filtering solution.

When using a fidelity for the truth that is higher than that of the estimate, it is of interest to see if the results using the UT are similar to the corresponding findings of implementing different fidelities in the EKF. Performing a simulation with a truth model of $^1/_{16}{}^\circ$ and an estimate of $^1/_4{}^\circ$ gives the results in Figure 6.7. The first thing to note here is that the introduction of the $^1/_4{}^\circ$ DEM in the estimate once again reduced the tuning noise needed for the filter to operate successfully. While originally the $^1/_{16}{}^\circ$ UT required 8.5 m of tuning noise, here this was reduced to only 3 m. This is still larger than the EKF simulations that used 1.5 m of position tuning noise, which follows the trend that the UT requires more tuning noise to allow a filter that converges on a correct solution. The reduction in tuning noise is also noticeable in the altimeter residual interval. While the single run performance is slightly larger than the Monte Carlo results, they are much closer in magnitude than the

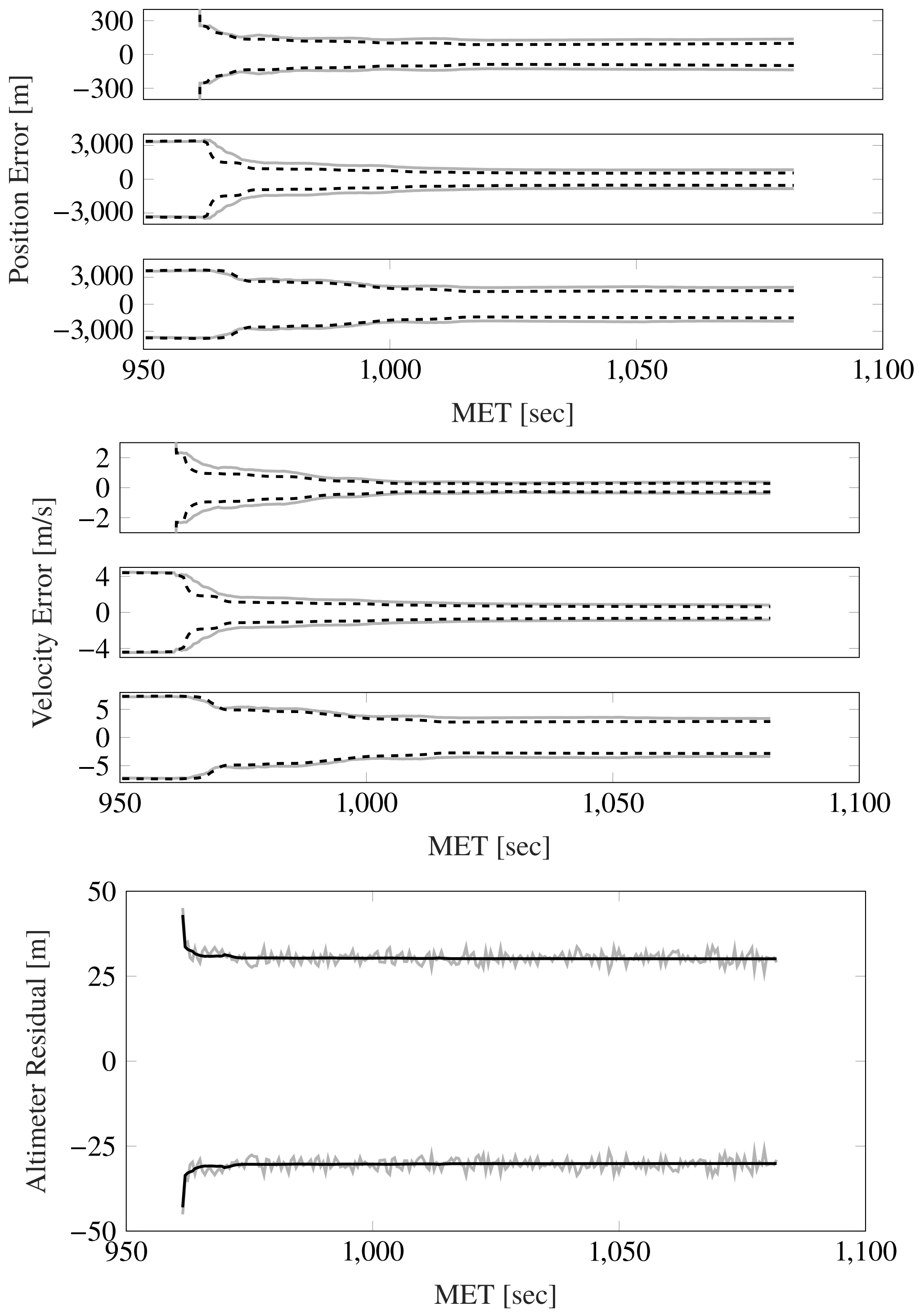


Figure 6.4. Position (top) and velocity (middle) error 3σ interval, and altimeter residual (bottom) 3σ intervals for the Monte Carlo simulation (solid gray) and a single filter run (solid black) when using the $^1/_4{}^\circ$ DEM for truth and estimate with UT.

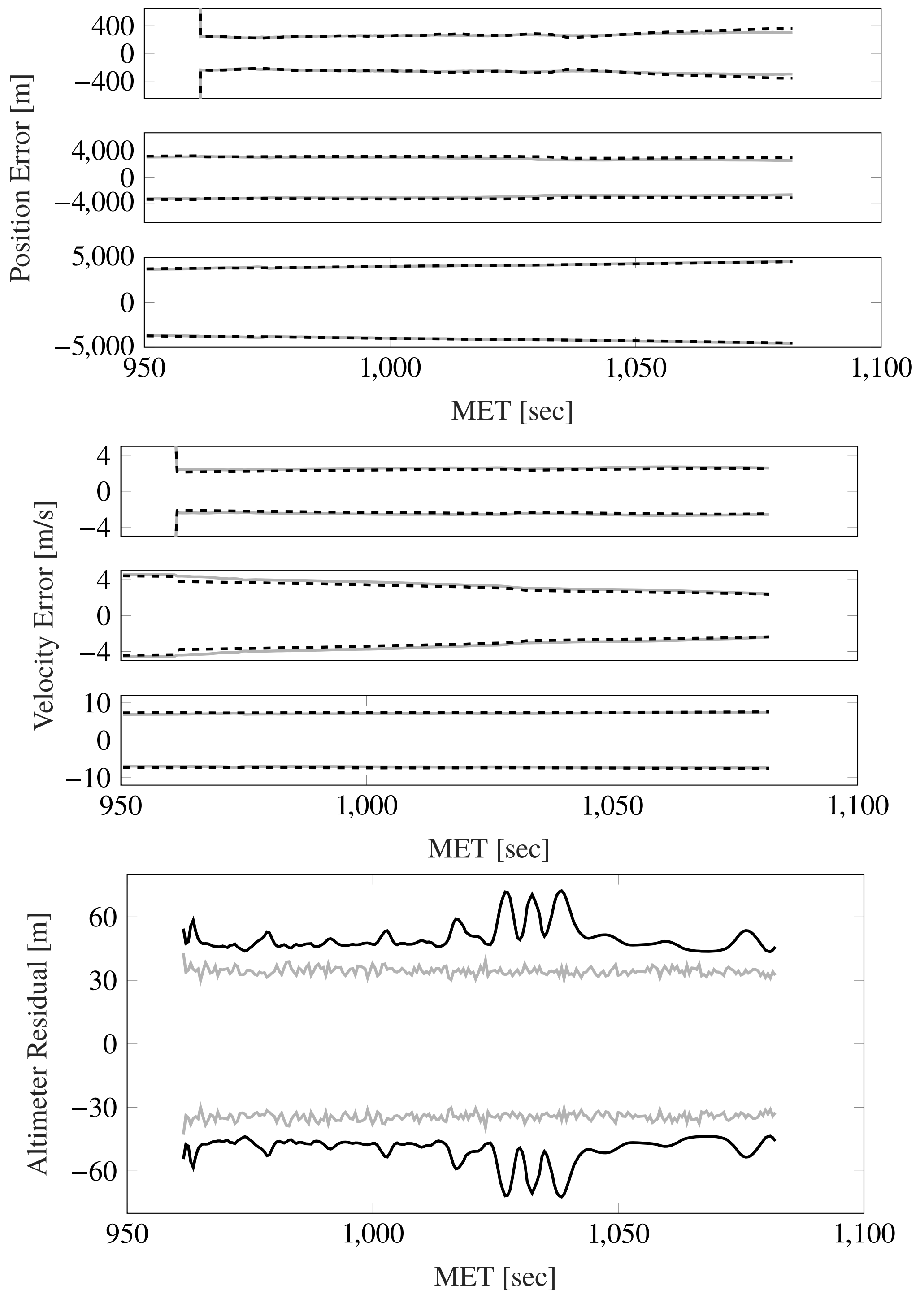


Figure 6.5. Position (top) and velocity (middle) error 3σ interval, and altimeter residual (bottom) 3σ intervals for the Monte Carlo simulation (solid gray) and a single filter run (solid black) when using the $^1/_{16}{}^\circ$ DEM for truth and estimate with UT.

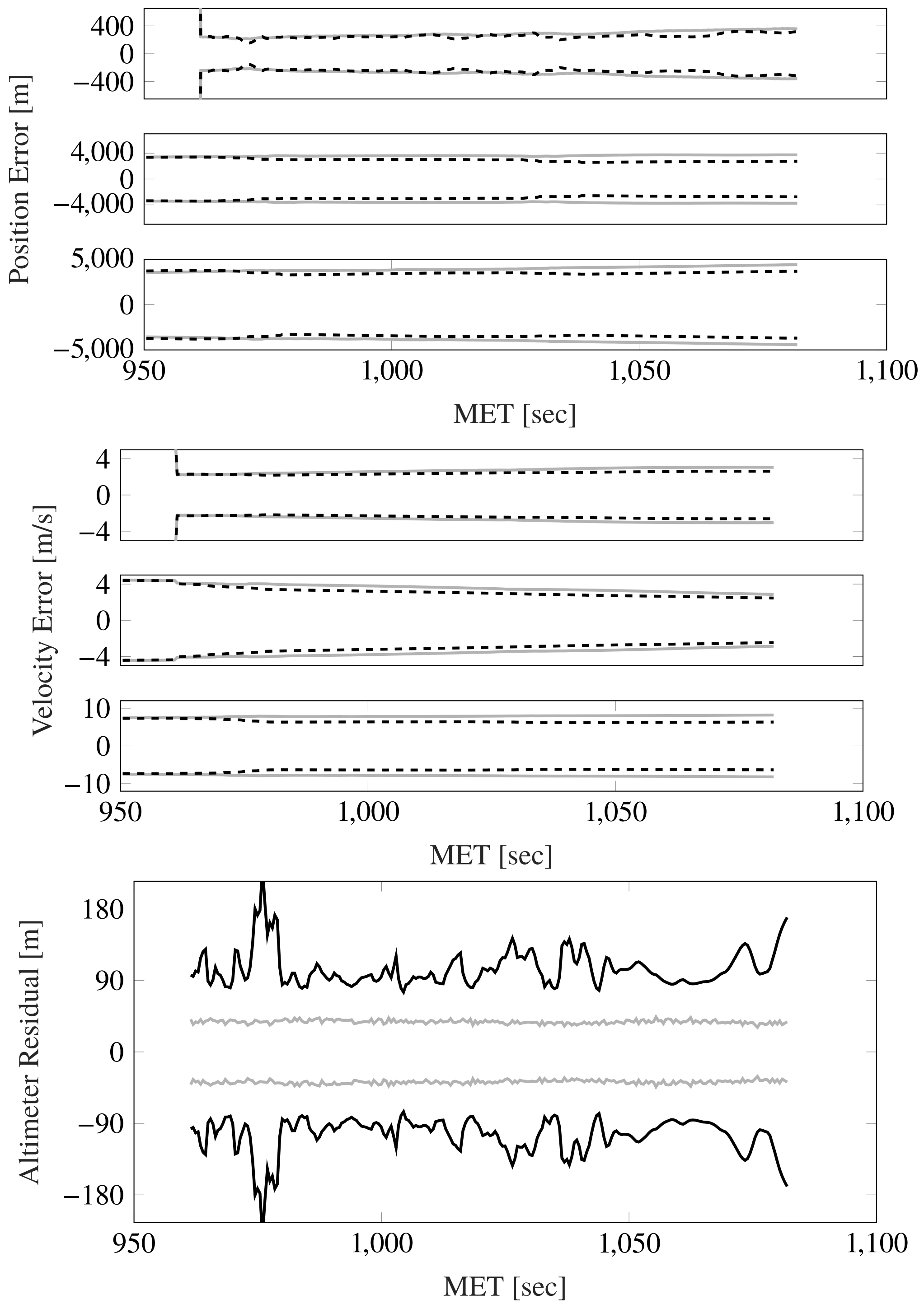


Figure 6.6. Position (top) and velocity (middle) error 3σ interval, and altimeter residual (bottom) 3σ intervals for the Monte Carlo simulation (solid gray) and a single filter run (solid black) when using the $^1/_{64}{}^\circ$ DEM for truth and estimate with UT.

identical $^1/_{16}{}^\circ$ residual due to the lesser noise used. The large variations seen using the $^1/_{16}{}^\circ$ estimate previously are also reduced by the use of the lesser $^1/_4{}^\circ$ estimate. Much like with the EKF, here the introduction of a lesser degree estimate helps to reduce the complexities of the higher fidelity truth to make it easier for the filter to converge and maintain a stable solution.

The trends seen in Figure 6.7 continue when using the $^1/_{128}{}^\circ$ DEM truth with the $^1/_4{}^\circ$ estimate. In Figure 6.8, the lesser degree estimate is able to assist the filter in bringing down the tuning noise to only 2 m in the position, even less than for the $^1/_{16}{}^\circ$ truth. By comparison, with a truth and estimate of the same fidelity, the $^1/_{64}{}^\circ$ DEM required 25 m of tuning noise, and the $^1/_{128}{}^\circ$ DEM would have required much more to function effectively. This further emphasizes the benefits of using a lesser degree DEM estimate that is able to smooth out the effects of the terrain that would otherwise be a much larger issue.

6.4. MONTE CARLO INTEGRATION

To understand how a nonlinear update may perform in a best case scenario, a Monte Carlo sampling is performed. This sampling is done with $100,000$ samples drawn from the uncertainty covariance. These samples are then input through the nonlinear transformation and used to determine the measurement, measurement covariance, and cross covariance for the altimeter. The results of this method are presented in Figure 6.9 where a simulation is performed with a $^1/_{16}{}^\circ$ DEM for the truth and estimate. Comparing these results to the UT with the same fidelities in fig. 6.5 the Monte Carlo simulation seems to assist the filter in a couple areas. In terms of the necessary tuning noise for a stable filter solution the position noise is reduced from the previous 8.5 m for the UT to 3.85 m for the Monte Carlo sampling. Looking at the performance of the filter, benefits are seen in the convergence of the velocity axis along with in position axis y and z. While there are not massive differences, the convergence seen continues further along in the trajectory than with the UT and to a more accurate solution. This sampling also has Monte Carlo simulation results that are slightly

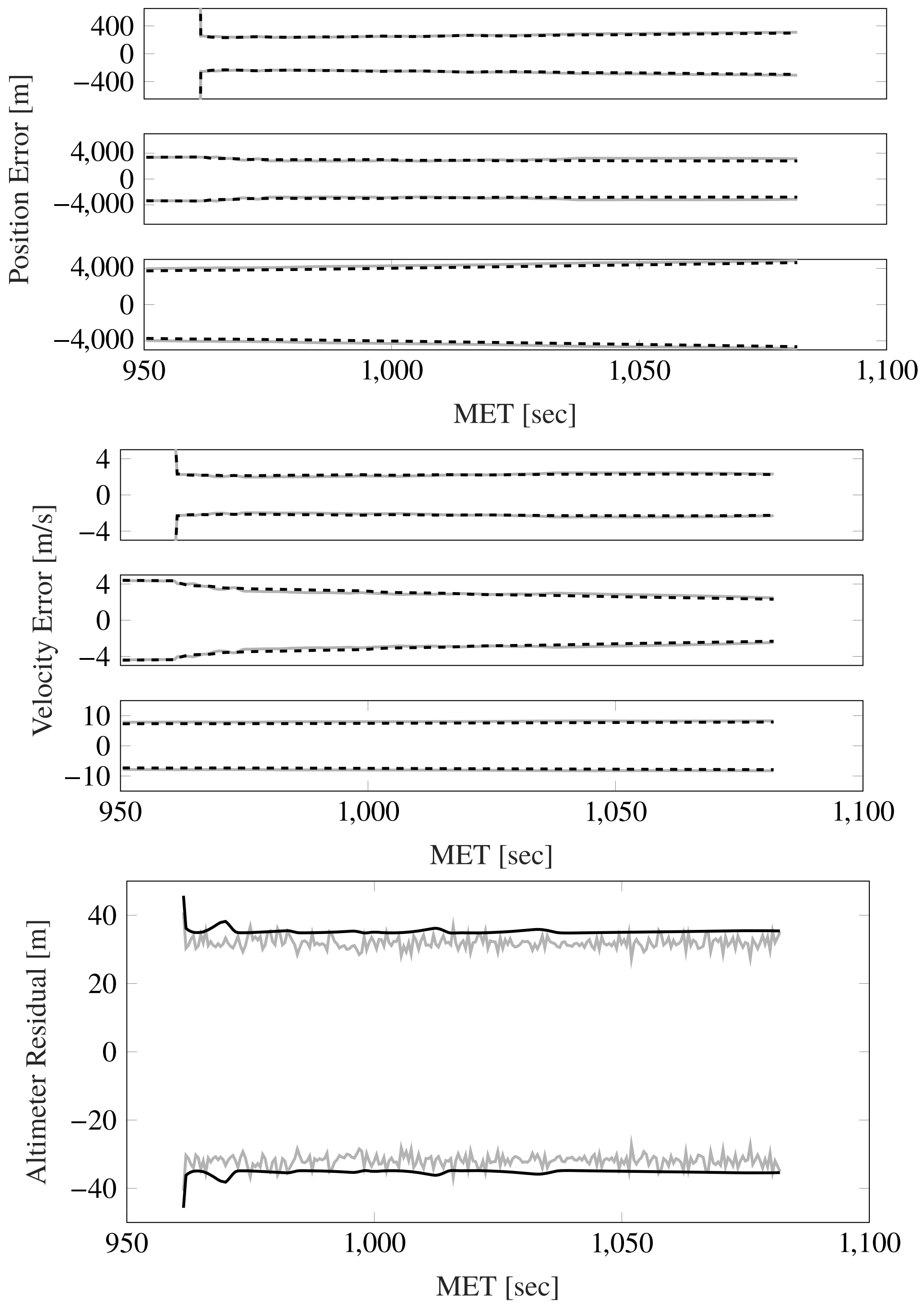


Figure 6.7. Position (top) and velocity (middle) error 3σ interval, and altimeter residual (bottom) 3σ intervals for the Monte Carlo simulation (solid gray) and a single filter run (solid black) when using the $^1/_{16}$° DEM for truth and $^1/_4$° DEM for estimate with UT.

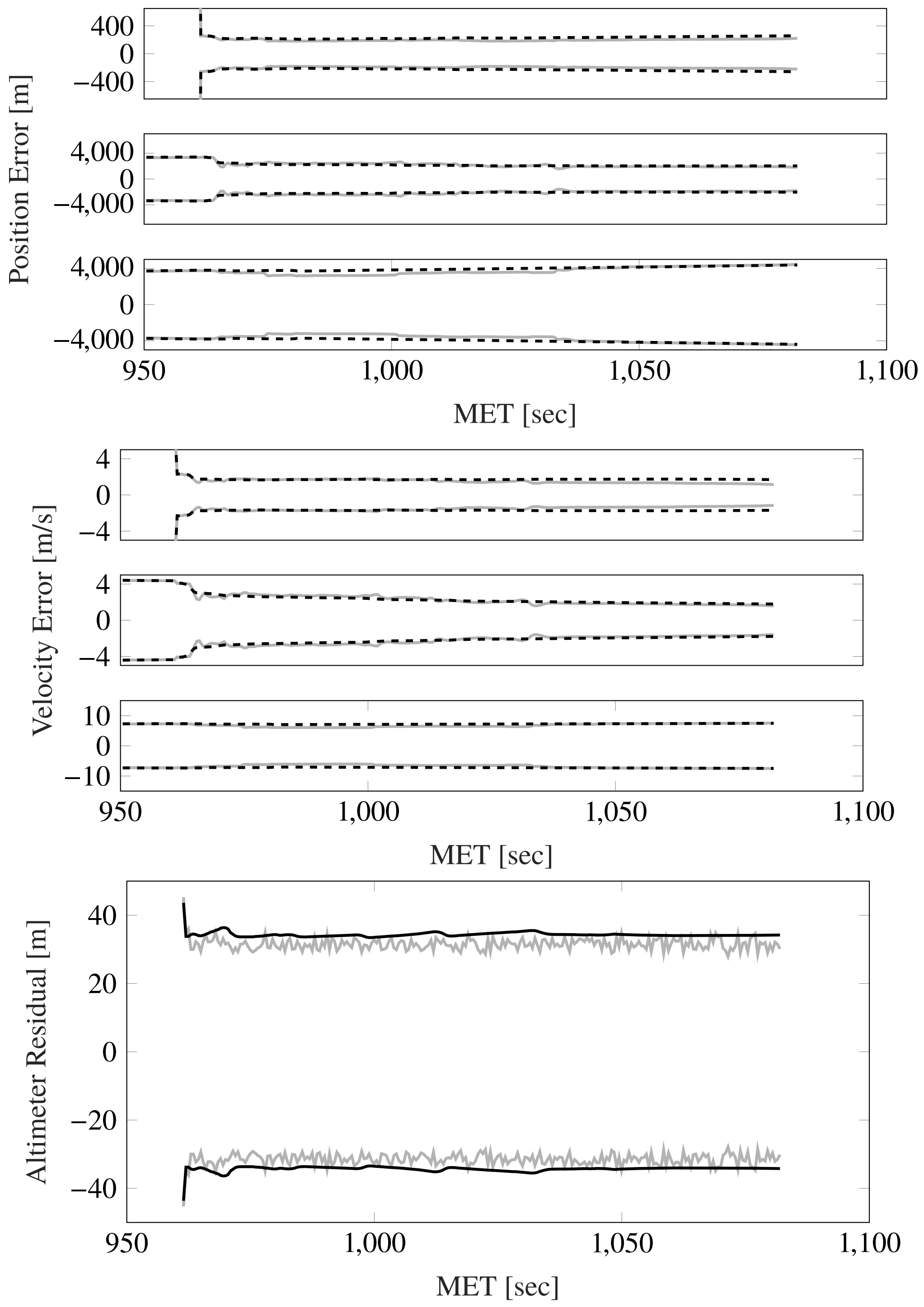
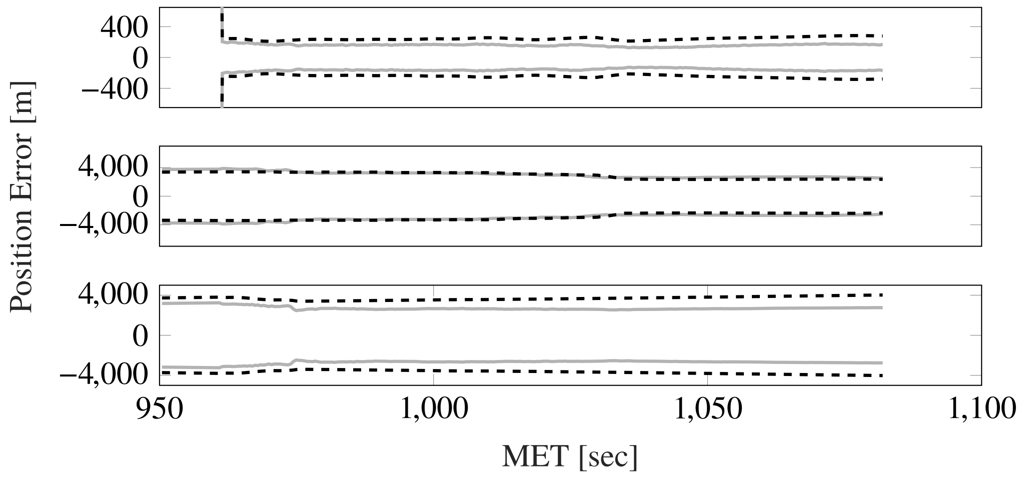
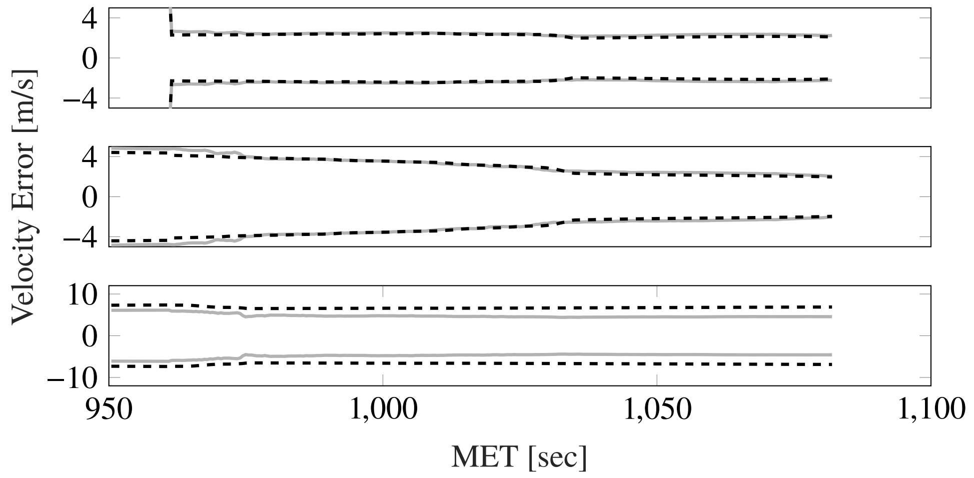


Figure 6.8. Position (top) and velocity (middle) error 3σ interval, and altimeter residual (bottom) 3σ intervals for the Monte Carlo simulation (solid gray) and a single filter run (solid black) when using the $^1/_{16}{}^\circ$ DEM for truth and $^1/_4{}^\circ$ DEM for estimate with UT.

conservative compared to the single run performance. This may be in part to the large number of samples taken here making the filter more sensitive to the tuning noise used. While tuning this method it was more appealing to be slightly conservative compared to the single run performance than having a filter where the Monte Carlo simulation fell outside the expected single run performance. The use of the Monte Carlo sampling provides a slightly better example of nonlinear sampling that allows the tuning noise used to be reduced and finds a slightly more accurate solution compared to the UT.





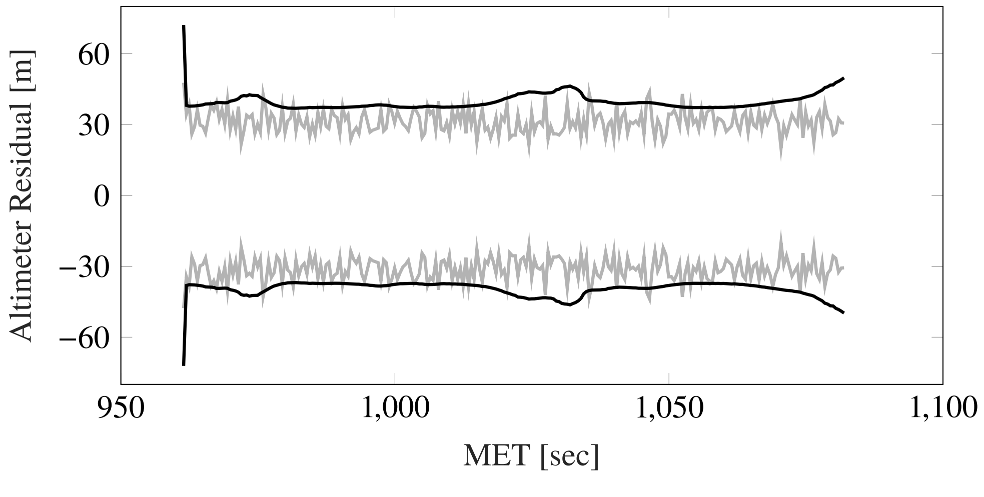


Figure 6.9. Position (top) and velocity (middle) error 3σ interval, and altimeter residual (bottom) 3σ intervals for the Monte Carlo simulation (solid gray) and a single filter run (solid black) when using the $^1/_{16}$° DEM for truth and $^1/_4$° DEM for estimate with UT.

7. CONCLUSION

To provide a filter that can more accurately model the "real world" it is intended to run in, simulations have to be performed that can verify the usefulness of said filter in characteristic scenarios. This was achieved here by simulating an entry, descent, and landing (EDL) scenario on the Moon. Starting at 25 km altitude the filter estimates the state comprised of the position, velocity, and attitude of the vehicle and sensor biases with an IMU, star camera, which was disengaged once thrusting began, and a nadir altimeter, which started at 8 km altitude. To determine filter performance with various fidelities of terrain, different altimeter models were used that ranged from a simple spherical model to complicated DEMs. These fidelities were varied for the altimeter sensor that is tested within the SR-MEKF filtering solution. While all components (truth, estimate, and measurement Jacobian) maintained the same fidelity, the results fell within the expected single run performance for lower fidelities but began to diverge as the fidelity was increased. It was found that this issue could be mitigated by introducing tuning noise to the filter that allowed it to account for the issues brought about by the more realistic terrain. While larger noises were required as fidelities were increased, this could be counteracted by the use of a lower degree estimate. Simulating a higher degree truth with a $1/4^\circ$ estimate and spherical Jacobian provided a filter that could be compared to the more reliable, high fidelity truth while using much less tuning noise than would otherwise be needed.

As fidelity of the altimeter model is increased, it became more difficult for the filter to properly converge to a stable solution, requiring the introduction of tuning noise to the system. This encouraged an inspection into alternative methods that could alleviate stability issues that plagued the higher fidelity models and reduce the tuning noise needed in the process. By reducing the uncertainty the filter has when the altimeter is first initiated, such that both the filter and altimeter start at the same time, a much more stable filter could be configured that performed as expected with higher fidelity models and no tuning noise. A more realistic attempt at uncertainty reduction was created by increasing the altitude at which the altimeter sensor was initiated. The slight reduction in uncertainty this produced was not enough to affect the filter as intended and did not produce any improvement in the results. The need for tuning noise was still required and without a greater reduction in uncertainty, no benefits were gained.

To construct a filter that could better account for the terrain of the surface below it, nonlinear transformations were implemented within the altimeter sensor update. This is done with an unscented transform (UT) that allowed the filter to obtain samples at various points on the surface within the area of uncertainty that allow it to better understand the varied terrain below the filter. The UT update was shown to reduce position and velocity errors compared to the EKF update when using a lower fidelity DEM. As the fidelity was increased, though, the instability of the terrain required the introduction of tuning noise once again. When the fidelity used for the filter estimation was varied from that of the truth, a stable solution was able to be found, once again, with a reduction in tuning noise, due to the lower fidelity estimate. If a transformation method were able to account for the large amount of variations in the higher fidelity DEMs, it could support a much more accurate filter, but as it stands the need for tuning noise reduces these benefits. While using a lower degree estimate helps, it is still not able to provide the more beneficial results seen in the lower fidelity truth and estimate that could be run without the need for tuning noise.

The use of Monte Carlo sampling within the filter allows it to have an even better understanding of the terrain below the vehicle with one-hundred thousand samples drawn from the state uncertainty. This allows the filter to perform better than the UT in reducing the tuning noise needed to provide a stable solution. The large sampling of the Monte Carlo integration method also provided a solution that continued to converge to a more accurate solution than the UT. While it did not reduce the tuning noise lower than what was used with linearization, it was able to provide a solution that kept converging as it continued though descent better than the other methods at the same fidelity. Providing the filter with a better understanding of the nonlinear terrain though this transformation allows for the construction of a more accurate filtering solution through the large Monte Carlo sampling.

As demands on the navigation filter grow with the ever increasing number of missions requiring EDL, the need for more realistic sensor modeling also grows. Simulations here show how topographical terrain can prove difficult for the filter to converge on a solution when the variations of that terrain change rapidly. While higher fidelity models require large amounts of tuning noise, this can be supplemented with lower fidelity estimates to assist in the filtering solution. As a way to supply the filter with more information of its surroundings, nonlinear transformations were utilized. The UT provided a better understanding of the nonlinear terrain and resulted in improvements for a lower fidelity DEM, with more stable and accurate filtering solutions. While these require more time for computation at this fidelity, the results in stability make it a beneficial trade off in performance. As fidelity is increased, the major benefit appears to come from the combination of a higher fidelity truth and lower fidelity estimate. This allows the necessary tuning noise to be reduced to provide a filter that follows its expected single run performance. Monte Carlo sampling provided an idea of how these nonlinear transformations can be assisted by the inclusion of more sample points, both decreasing tuning noise and solution error compared to the UT. Going forward, more research into alternate methods that allow the filter to better understand the terrain below it will in turn allow for more realistic and high fidelity DEMs to be used to provide accurate and reliable filtering solutions.

APPENDIX

VERMEILLE'S ALGORITHM

In order to implement the ellipsoidal altimeter model, a method for determining geodetic coordinates must be defined. To do this Vermeille's Algorithm was chosen to find the height of the vehicle above the point on the ellipsoid below it. Given the ellipsoidal parameters of the semi-major axis, a, and the eccentricity, e, along with the fixed frame position of the vehicle, $\boldsymbol{r}^f_{alt}$, this height can be determined. This is done via a transformation to the selenodetic latitude, longitude, and altitude via a series of intermediary equations provided by Vermeille [33]. Using the necessary parameters these intermediary terms are be calculated as

$$p = \frac{x^2 + y^2}{a^2}, \quad q = \frac{(1 - e^2)z^2}{a^2}, \quad r = \frac{p + q - e^4}{6}, \quad s = \frac{pqe^4}{4r^3},$$
$$t = \sqrt[3]{1 + s + \sqrt{s(2 + s)}}, \quad u = r\left(1 + t + \frac{1}{t}\right), \quad v = \sqrt{u^2 + e^4 q},$$
$$w = e^2 \frac{u + v - q}{2v}, \quad k = \sqrt{u + v + w^2} - w, \quad \text{and} \quad d = \frac{k\sqrt{x^2 + y^2}}{k + e^2}.$$

The altitude for the ellipsoidal altimeter model is then calculated as

$$h = \frac{k + e^2 - 1}{k}\sqrt{d^2 + z^2}$$

providing the selenodetic height, h, above the ellipsoidal surface.

With the altitude above the surface found, the last component needed for the filter is the partial derivative Jacobian of this measurement [6]. This is once again done with a set of intermediary terms defined as

$$S_{x,y} = \frac{qe^4}{4r^4}(2r-p)\,, \quad T_{x,y} = \frac{1}{3t^3}\frac{1}{\sqrt{s(2+s)}}S_{x,y}\,, \quad U_{x,y} = \frac{u}{3r} + r\left(1-\frac{1}{t^2}\right)T_{x,y}\,,$$
$$V_{x,y} = \frac{u}{v}U_{x,y}\,, \quad W_{x,y} = \frac{1}{2v}\left(e^2U_{x,y} + (e^2-2w)V_{x,y}\right)\,,$$
$$K_{x,y} = \frac{1}{2}\frac{1}{k+w}\left(U_{x,y} + V_{x,y} - 2kW_{x,y}\right)\,, \quad D_{x,y} = d\left(\frac{e^2}{k(k+e^2)}K_{x,y} + \frac{1}{p}\right)\,,$$
$$\text{and} \quad H_{x,y} = \frac{1-e^2}{k^2}\sqrt{d^2+z^2}K_{x,y} + \frac{k+e^2-1}{k}\frac{d}{\sqrt{d^2+z^2}}D_{x,y}\,.$$

and

$$S_z = \frac{pe^4(1-e^2)}{4r^4}(2r-q)\,, \quad T_z = \frac{1}{3t^3}\frac{1}{\sqrt{s(2+s)}}S_z\,, \quad U_z = \frac{u(1-e^2)}{3r} + r\left(1-\frac{1}{t^2}\right)T_z\,,$$
$$V_z = \frac{u}{v}U_z + \frac{e^4(1-e^2)}{v}, \quad W_z = \frac{1}{2v}\left(e^2U_z + (e^2-2w)V_z - 2e^2(1-e^2)\right)\,,$$
$$K_z = \frac{1}{2}\frac{1}{k+w}\left(U_z + V_z - 2kW_z\right)\,, \quad D_z = \frac{e^2\sqrt{x^2+y^2}}{(k+e^2)^2}K_z\,,$$
$$\text{and} \quad H_z = \frac{1-e^2}{k^2}\sqrt{d^2+z^2}K_z + \frac{k+e^2-1}{k}\frac{dD_z + a^2}{\sqrt{d^2+z^2}}\,.$$

This then allows the partial derivative of the altitude with respect to the fixed frame position be determined as

$$\frac{\partial h}{\partial \boldsymbol{r}^f_{alt}} = \frac{1}{a^2}(\boldsymbol{r}^f_{alt})^T \text{diag}(H_{x,y},\ H_{x,y},\ H_z)\,,$$

providing the derivative of the selenodetic parameters with respect to the Cartesian coordinates used in determining the altitude solution.

REFERENCES

[1] A. Johnson, Y. Cheng, J. Montgomery, N. Trawny, B. Tweddle, and J. Zheng, 'Real-time terrain relative navigation test results from a relevant environment for mars landing,' 2015 .

[2] Bar-Shalom, Y. and Fortmann, T. E., *Tracking and Data Association*, volume 179, chapter Tracking a Single Target in Clutter, pp. 157–190, Academic Press, 1988.

[3] Bierman, G. J., *Factorization Methods for Discrete Sequential Estimation*, volume 1, Academic Press, 1977.

[4] Caflisch, R. E., 'Monte carlo and quasi-monte carlo methods,' 1998.

[5] Crassidis, J. L. and Junkins, J. L., *Optimal Estimation of Dynamic Systems*, CRC Press, 2012.

[6] DeMars, K. J., 'Precision navigation for lunar descent and landing,' 2007.

[7] Fetherstone, W. and Claessens, S., 'Closed-form transformations between geodetic and ellipsoidal coordinates,' 2008.

[8] Gelb, A., *Applied Optimal Estimation*, M.I.T. Press, 1974.

[9] Grewal, M. S. and Andrews, A. P., *Kalman Filtering: Theory and Practice*, Prentice Hall, 1993.

[10] Jacob E. Darling, C. F., Nathan Houtz and DeMars, K. J., 'Recursive filtering of star tracker data,' 2016 .

[11] Jindrich Dunik, Ondrej Straka, and Miroslav Simandl, 'Stochastic integration filter,' 2013.

[12] Johnson, A. E. and Montgomery, J. F., 'Overview of terrain relative navigation approaches for precise lunar landing,' 2008 .

[13] Julier, S. J. and Uhlmann, J. K., 'A new extension of the kalman filter to nonlinear systems,' 1997 .

[14] Kalman, R. E., 'A new approach to linear filtering and prediction problems,' 1960, **82**, pp. 35–45.

[15] Kaplan, E. D. and Hegarty, C. J., *Understanding GPS Principles and Applications*, Artech House, 2006.

[16] Keat, J. E., 'Analysis of least-squares attitude determination routine DOAOP,' 1977.

[17] Kirubarajan, T. and Bar-Shalom, Y., 'Probabilistic data association techniques for target tracking in clutter,' Proceedings of the IEEE, 2004, **92.3**, pp. 536–557.

[18] Kreigsman, D. A. and Tao, Y.-C., 'Shuttle navigation system for entry and landing mission phases,' 1975, **12**(4), pp. 213–219.

[19] Lorenz, David, Ryan Olds, Alex May, Courtney Mario, and Mark Perry, 'Lessons learned from osiris-rex autonomous navigation using natural feature tracking,' 2017 .

[20] Markley, F. L. and Crassidis, J. L., *Fundamentals of Spacecraft Attitude Determination and Control*, Springer, 2016.

[21] McCabe, J. S. and DeMars, K. J., 'Particle filter methods for space object tracking,' 2014 .

[22] National Aeronautics and Space Administration, 'Planetary data system, clementine data set,' 1996, `http://pds-geosciences.wustl.edu/lunar/clem1-gravity-topo-v1/cl_8xxx/topo/`.

[23] National Aeronautics and Space Administration, 'Planetary data system, lunar reconnaissane orbiter data set,' `http://pds-geosciences.wustl.edu/lro/`, 2017.

[24] National Aeronautics and Space Administration, 'Planetary facts sheet,' 2017, `https://nssdc.gsfc.nasa.gov/planetary/factsheet/moonfact.htm`.

[25] Papoulis, A., *Probability, Random Variables, and Stochastic Processes*, McGraw-Hill, 2nd edition, 1984.

[26] Press, W. H., *Numerical Recipes in Fortran 77*, Cambridge University Press, 2001.

[27] R. Zanetti, K. J. DeMars, and R. H. Bishop, 'Underweighting nonlinear measurements,' 2010, **33**(5), pp. 1670–1675.

[28] Savage, P. G., *Strapdown Analytics*, Strapdown Associates, 2000.

[29] Shuster, M. and Oh, S., 'Spacecraft attitude determination simulation to improve the efficiency of a star tracker,' 1981, **4**, pp. 70–77.

[30] Simon J. Julier, Jeffrey K. Uhlmann, and Hugh F. Durrant-Whyte, 'A new approach for filtering nonlinear systems,' volume 3, 1995 pp. 1628–1632.

[31] Sofair, I., 'Improved method for calculating exact geodetic latitude and altitude,' 1997.

[32] van der Merwe, R., 'Sigma-point kalman filters for probabilistic inference in dynamic state-space models,' 2004.

[33] Vermeille, H., 'Computing geodetic coordinates from geocentric coordinates,' 2004, **78**, pp. 94–95.

[34] Ward, K. C. and DeMars, K. J., 'Including topographical effects in slant-range modeling,' 2018 .

[35] Zanetti, R. and DeMars, K. J., ‘Joseph formulation of unscented and quadrature filters with application to consider states,’ 2013.

VITA

Kenneth Kratzer studied physics at Lewis University starting in the fall of 2012, where he obtained his Bachelor of Science degree in May of 2016. The following fall he started at Missouri University of Science and Technology pursuing a graduate degree in aerospace engineering. He worked as a research assistant under Dr. Kyle DeMars throughout his years there, where he studied the effects of terrain fidelity on navigation performance. In July 2018 he received his Master of Science in Aerospace Engineering from Missouri University of Science and Technology.