EXTENDING THE PRACTICAL APPLICABILITY OF THE KALMAN FILTER

A Dissertation

by

JOSE HUMBERTO RAMOS ZUNIGA

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

August 2020

Major Subject: Aerospace Engineering

ABSTRACT

A Schmidt filter is a modification of the Kalman filter that allows to append system parameters as states and considers their uncertainty effect in the filtering process without attempting to estimate such parameters. The states that are only considered but not estimated, are generally known as *consider* or *considered* states. The main contributions of this research are the formulations of a Schmidt-Kalman filter that incorporates the numerical robustness of the well-known square root and factorized filtering forms plus the capacity of actively attempting to update the *considered* states.

The filters formulations proposed in this research are a fundamental extension of the Kalman filter. Therefore, the formulations of this work also apply within the Extended Kalman filter framework. More importantly, they are shown to handle nonlinearities, larger initial uncertainties, and poorly conditioned systems better than a typical Extended or Schmidt Kalman filter. Because the new filters directly based on the Schmidt filter, they offer a novel and straight-forward filtering framework, allowing the use of a more simple filter where a more advanced or elaborated technique could have been needed.

The proposed contributions of this research are organized as follows. First, the Partial-update Kalman filter, a generalized Schmidt filter that allows updating the user-selected consider states partially, is introduced. The indirect or multiplicative error version of this new filter is also derived. An error stability analysis (for linear systems) of the partial-update filter and a discussion on its numerical stability and the potential numerical robustness improvements is presented. Second, a square root formulation to improve the numerical stability of the partial-update Schmidt filter is developed. The derivation of sequential and vector measurement processing schemes for the square root formulation are both presented along with a brief computational complexity and Montecarlo analysis. Third, to gain computational efficiency but still retaining a numerically robust formulation, as an alternative, a U-D factorized version of the partial-update filter is also developed. Fourth, to improve estimation consistency and accuracy of the partial-update filter, baseline methods are proposed to attempt the estimation of the considered states. Finally, formulations proposed in this research are validated through hardware implementations to solve aerospace engineering-related problems.

DEDICATION

To Maria, Eileen and Rihan, my lovely family. To my friends that always have supported and believed in me.

ACKNOWLEDGMENTS

The completion of a doctoral degree is not a task I could have completed by myself. I have been fortunate to have such wonderful persons with me during this journey always to listen, support, and encourage me. I would first like to thank my lovely wife, Maria, for being by my side along the way, helping me in all ways she can, and always believe in me, but above all for being so brave and accept to leave all we had in our country to support me. I would also like to thank my father, Humberto, and my grandmother, Chabelita, my sisters, Mayra and Yajaira, and my brother, Kevin, for always being so supportive even when they are far away. Also, I want to express my gratitude to my wife's family, the Don Garcias, where I have been received as another son and supported in many ways.

I have no words to express my gratitude to my advisor, Dr. Hurtado. I could not be more fortunate to meet and work with him. Thank you for letting me learn from you in both professional and a personal way. I truly appreciate that you always found a way to support me and in one way or another, keeping me on track. Thank you for all your patience, and the support when things were fine and also when they were not that favorable. There is no way to tell how grateful I am for the time you have invested in me. Thank you for believing in me. You will always have all my respect and admiration.

I would like to thank my committee members, Dr. Valasek, Dr. Majji, and Dr. Hur, for their time and invaluable advice for the completion of this work. It is always a pleasure to visit their office to share ideas.

Dr. Kevin Brink, from the Air Force Research Laboratory, was also crucial in the development of this dissertation. I want to thank him for all his advice, support, and the time he invested in me so I could complete this research. Thank you for your patience, guidance, and the opportunity of interning with you. The work at your lab, the Autonomous Vehicles Lab at the Research Education and Engineering Facility (REEF) at the University of Florida, was a course-changer in my academic development. From the REEF, I also want to thank Prashant Ganesh, a fantastic person, engineer, and friend with who I had the pleasure to work during two summer internships.

During my stay at Texas A&M, I met wonderful people that have positively impacted my life and helped me to make the past few years more bearable. I want to thank Vinicius Guimaraes, Niladri Das, Daniel Whitten, Clark Moody, Neil McHenry, Irving Solis, Oscar Barajas, Tim Woodbury, and Davis Adams for their support, encouragement, and friendship. Thanks for all the countless good moments, dinners, and trips that remembered me that there is always time to relax.

Finally, I again thank Dr. Hurtado. This work had not been possible without your full commitment.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a dissertation committee consisting of Professor John Hurtado, Professors John Valasek, and Manoranjan Majji, of the Aerospace Engineering Department, and Professor Pilwon Hur of the Mechanical Engineering Department. Dr. Brink of the Air Force Research Laboratory also served as a mentor for this work.

The data collection for Section 6.2 and 6.3 was performed in part by Prashant Ganesh of the REEF at the University of Florida. Davis Adams of the LASR Laboratory at Texas A&M helped in part to organize simulation and running code for 5.3 and 5.4. All other work conducted for the dissertation was completed by the student independently.

Funding Sources

Graduate study was supported in part by Texas A&M University, Air Force Research Laboratory, and a doctorate fellowship (with ID 411660) from the Mexican National Council for Science and Technology.

TABLE OF CONTENTS

Page

ABSTRACT ii

DEDICATION iv

ACKNOWLEDGMENTS v

CONTRIBUTORS AND FUNDING SOURCES vii

TABLE OF CONTENTS viii

LIST OF FIGURES xii

LIST OF TABLES xix

1. INTRODUCTION 1

1.1 Motivation 1
1.1.1 The Schmidt-Kalman and the partial-update Schmidt filter 1
1.1.2 partial-update Schmidt filter challenges 2
1.2 Literature Review 3
1.3 Numerical issues and factorized formulations 3
1.3.1 Square-root filtering 5
1.3.2 The U-D filter formulations 5
1.4 The Schmidt consider filter 8
1.5 The dissertation objectives and outline 9
1.5.1 Outline 10

2. A GENERALIZATION OF THE SCHMIDT KALMAN FILTER 12

2.1 Discrete Extended Kalman filter framework and notation 12
2.2 The partial-update filter concept 14
2.2.1 The Schmidt-Kalman filter 14
2.2.2 The partial-update Schmidt-Kalman filter 16
2.3 The partial-update for the indirect Kalman filter 18
2.3.1 Indirect filtering 19
2.3.1.0.1 Indirect filter formulation. 19
2.3.2 The conventional MEKF update 20
2.3.3 The partial-update within the Multiplicative EKF 22
2.3.3.1 The PU-MEKF 22
2.3.4 Filter key equations and algorithm 26

2.4 Stability analysis of the partial-update filter ... 27
2.5 Partial-update numerical stability issues ... 31

3. SQUARE ROOT PARTIAL-UPDATE SCHMIDT KALMAN FILTER ... 34

3.1 Introduction & Motivation ... 34
3.1.1 Square Root filtering ... 35
3.2 The square root partial-update Schmidt-Kalman filter ... 37
3.2.1 Measurement update ... 38
3.2.2 Time update ... 42
3.3 Numerical examples ... 43
3.3.1 Body re-entering Earth atmosphere ... 44
3.3.2 Camera to Inertial Measurement Unit (IMU) calibration ... 50
3.4 Monte Carlo runs ... 55
3.4.1 Condition number ... 59
3.5 Processing a vector-valued measurement ... 62
3.5.1 Square root partial-update for vector-valued measurements ... 62
3.5.1.1 Conventional time update with vector-valued measurement processing ... 65
3.5.2 Measurement and process noise covariance decorrelation ... 66
3.5.3 Computational complexity ... 68
3.6 Summary ... 72

4. U-D PARTIAL-UPDATE KALMAN FILTER ... 73

4.1 Introduction ... 73
4.1.1 The UD filter background ... 73
4.2 The conventional UD Kalman filter ... 75
4.2.1 UD temporal update overview ... 75
4.3 The UD partial-update derivation ... 78
4.4 Measurement decorrelation using UD factors ... 80
4.5 Numerical example ... 82
4.5.1 Body re-entering Earth atmosphere single run ... 82
4.6 Monte Carlo runs ... 83
4.7 Numerical complexity ... 87
4.7.1 IMU-camera example ... 89
4.7.2 Summary ... 90

5. DYNAMIC PARTIAL-UPDATE KALMAN FILTER ... 92

5.1 Motivation ... 92
5.2 Dynamic partial-update weights ... 92
5.3 Nonlinearity-aware based method ... 93
5.3.1 Nonlinearity-aware partial-update ... 93
5.3.2 Numerical example ... 98
5.3.3 Pre-tuned partial-update weights as a baseline for DNL method ... 102

5.4 Covariance-aware based method 111
5.4.1 The re-entry falling body 114
5.4.2 Pre-tuned partial-update weights as a baseline for the DC method 120
5.4.3 Comments on DNL and DC partial-update methods 124

6. HARDWARE APPLICATIONS 127

6.1 Online IMU-camera intersensor parameters calibration 128
6.1.1 Introduction 128
6.1.2 Related Work 129
6.1.3 Filter-based IMU-camera calibration algorithm 130
6.1.3.1 Notation 131
6.1.3.2 The propagation step 131
6.1.3.2.1 State vector. 131
6.1.3.2.2 True process model. 132
6.1.3.2.3 Expectation of process model. 133
6.1.3.2.4 Forming the error model. 134
6.1.3.2.5 Covariance propagation. 135
6.1.3.3 PU-MEKF Measurement Update. 137
6.1.3.3.1 Measurement model. 139
6.1.3.4 The partial-update within the Multiplicative EKF 141
6.1.4 Simulations 143
6.1.5 Hardware experiments 148
6.1.5.1 Setup 148
6.1.5.2 Filter Initialization 150
6.1.5.3 Results 151
6.1.6 Summary 152
6.2 A simplified Unmanned Aerial Vehicle state estimation framework 153
6.2.1 Introduction and Motivation 153
6.2.2 High-Level System Overview 157
6.2.2.1 Autopilot 158
6.2.2.2 On-board computer 158
6.2.3 Estimator Design 159
6.2.3.1 Propagation of the Z estimator 160
6.2.3.2 Measurement update for the Z estimator 161
6.2.3.3 Propagation of the XY estimator 161
6.2.3.4 Measurement update for the XY estimator 162
6.2.3.5 Partial-update filter modification 163
6.2.3.6 The XY and Z filters operation frequency 164
6.2.4 Controller Design 164
6.2.5 Hardware implementation 166
6.2.5.1 Flight performance 166
6.2.6 Summary 172
6.3 Angular rate estimation of a non-cooperative space body using RGB-D measurements 172
6.3.1 Introduction and motivation 172

6.3.2 Tracking and estimation system overview 174
6.3.3 Additional notation, modeling and feature treatment 175
6.3.3.1 Additional notation 175
6.3.3.2 Reference frames 175
6.3.3.3 Reprojection model 176
6.3.3.4 Feature extraction 177
6.3.3.5 Feature selection criteria 179
6.3.3.6 Feature position vectors in the arbitrary body frame 179
6.3.4 Extended Kalman Filter 181
6.3.4.1 Process and measurement model 181
6.3.4.2 UD partial-update filter 185
6.3.5 Filter initialization 185
6.3.6 EKF re-initialization 187
6.3.7 Hardware experiments 188
6.3.7.1 Experimental setup 188
6.3.7.2 Experiments 190
6.3.7.3 Angular velocity estimates 191
6.3.8 Summary 193

7. CONCLUSIONS 195

REFERENCES 197

APPENDIX A. HARDWARE PARAMETERS 208

LIST OF FIGURES

FIGURE Page

3.1 Standard EKF and square root partial-update EKF with full updates. The inset on the right shows a zoom-in for the last second of the simulation, displaying significant filter inconsistency with estimates well outside of 3σ bounds. Reprinted with permission from [1]. 47

3.2 Partial-update EKF and square root Partial-Update EKF with $\boldsymbol{\beta} = \begin{bmatrix} 0.9 & 0.9 & 0.75 \end{bmatrix}$ resulting in 90%, 90%, and 75% updates respectively to the position, velocity, and ballistic coefficient states. The inset on the right shows a zoom-in for the last second of the simulation, displaying filter results with estimates within the 3σ bounds. Reprinted with permission from [1]. 48

3.3 Partial-update EKF and square root Partial-Update EKF with $\boldsymbol{\beta} = \begin{bmatrix} 1.0 & 1.0 & 0.1 \end{bmatrix}$ resulting in 100%, 100%, and 10% updates respectively to the position, velocity, and ballistic coefficient states. The inset on the right shows a zoom-in for the last second of the simulation, displaying filter results with estimates within the 3σ bounds. Reprinted with permission from [1].......... 49

3.4 Camera-IMU calibration lever arm estimates when a full update is performed........ 54

3.5 Camera-IMU calibration rotation error estimates when a full update is performed.... 55

3.6 IMU global position when a full update is performed.......... 56

3.7 Camera-IMU calibration lever arm estimates when a partial-update is performed. ... 57

3.8 Camera-IMU calibration rotation estimates when a partial-update is performed. 58

3.9 IMU global position when a full update is performed.......... 59

3.10 Averaged and sampled standard deviation from 500 Monte Carlo runs for the lever arm components. The mean error across the runs is also plotted. Units are in meters. 60

3.11 Averaged and sampled standard deviation from 500 Monte Carlo runs for rotation error (IMU to camera rotation). The mean error across the runs is also plotted. Units are in meters.......... 61

3.12 Monte Carlo standard EKF and square root partial-update EKF with full updates. ... 62

3.13 Monte Carlo runs for the partial-Update EKF and square root partial-update EKF with $\boldsymbol{\beta} = \begin{bmatrix}0.9 & 0.9 & 0.75\end{bmatrix}$ resulting in 90%, 90%, and 75% updates respectively to the position, velocity, and ballistic coefficient states. 63

3.14 Monte Carlo runs for the partial-Update EKF and square root partial-update EKF with $\boldsymbol{\beta} = \begin{bmatrix}0.9 & 0.9 & 0.75\end{bmatrix}$ resulting in 90%, 90%, and 75% updates respectively to the position, velocity, and ballistic coefficient states. 64

3.15 Uncertainty condition number for the partial-update EKF and square root partial-update EKF filter example for $\boldsymbol{\beta} = \begin{bmatrix}0.9 & 0.9 & 0.75\end{bmatrix}$. Reprinted with permission from [1]. 65

3.16 Rough complexity comparison for sequential and batch measurement processing for the Kalman filter. UDU decorrelation is considered in the cost to use sequential filtering. 71

4.1 Standard EKF and UD partial-update EKF with full updates. The inset on the right shows a zoom-in for the last second of the simulation, displaying significant filter inconsistency with estimates well outside of 3σ bounds. 84

4.2 Standard EKF and UD partial-update EKF with partial updates. The inset on the right shows a zoom-in for the last second of the simulation, displaying significant filter inconsistency with estimates well outside of 3σ bounds. 85

4.3 partial-update EKF and UD partial-update EKF with $\boldsymbol{\beta} = \begin{bmatrix}0.9 & 0.9 & 0.75\end{bmatrix}$ resulting in 90%, 90%, and 75% updates respectively to the position, velocity, and ballistic coefficient states. The inset on the right shows a zoom-in for the last second of the simulation, displaying filter results with estimates within the 3σ bounds. 86

4.4 Monte Carlo standard EKF and UD partial-update EKF with full updates. 87

4.5 Monte Carlo runs for the partial-Update EKF and UD partial-update EKF with $\boldsymbol{\beta} = \begin{bmatrix}0.9 & 0.9 & 0.75\end{bmatrix}$ resulting in 90%, 90%, and 75% updates respectively to the position, velocity, and ballistic coefficient states. 88

4.6 Monte Carlo runs for the partial-Update EKF and UD partial-update EKF with $\boldsymbol{\beta} = \begin{bmatrix}0.9 & 0.9 & 0.75\end{bmatrix}$ resulting in 90%, 90%, and 75% updates respectively to the position, velocity, and ballistic coefficient states. 89

5.1 Dynamic nonlinearity-aware method (DNL) without previous tuning, static partial-update with $\boldsymbol{\beta} = \begin{bmatrix}0.9 & 0.9 & 0.75\end{bmatrix}$ and conventional EKF single run for body re-entry problem. Initial error within 1σ. 99

5.2 Dynamic nonlinearity-aware method (DNL) without previous tuning, static partial-update with $\boldsymbol{\beta} = \begin{bmatrix}0.9 & 0.9 & 0.75\end{bmatrix}$ and conventional EKF single run for body re-entry problem. Initial error within 1σ. 100

5.3 Dynamic nonlinearity-aware method (DNL) without previous tuning, static partial-update with $\boldsymbol{\beta} = \begin{bmatrix}0.9 & 0.9 & 0.75\end{bmatrix}$ and conventional EKF single run for body re-entry problem. Initial error within 2σ. 101

5.4 Dynamic nonlinearity-aware method (DNL) without previous tuning, static partial-update with $\boldsymbol{\beta} = \begin{bmatrix}0.9 & 0.9 & 0.75\end{bmatrix}$ and conventional EKF single run for body re-entry problem. Initial error within 2σ. 101

5.5 Monte Carlo runs state histories for the DNL partial-update. 102

5.6 Averaged and sampled standard deviation from 100 Monte Carlo runs for the DNL partial-update. The mean error is also shown. Full update is used as nominal value. . 103

5.7 DNL method, static partial-update with $\boldsymbol{\beta} = [0.9, 0.9, 0.75]$, and conventional EKF single run for body re-entry problem. Initial error within 1σ. 104

5.8 DNL method and static partial-update with $\boldsymbol{\beta} = [0.9, 0.9, 0.75]$ single run for body re-entry problem. Initial error within 1σ. 105

5.9 Dynamic nonlinearity-aware method (DNL) β history for the body re-entry problem. Initial error within 1σ.. 105

5.10 DNL method, static partial-update with $\boldsymbol{\beta} = [0.9, 0.9, 0.75]$ single run for body re-entry problem. 3σ initial errors.. 106

5.11 Dynamic nonlinearity-aware method (DNL) β history for the body re-entry problem. 3σ initial errors. 107

5.12 DNL method and static partial-update with $\boldsymbol{\beta} = [0.9, 0.9, 0.75]$ single run for body re-entry problem. 4σ initial errors.. 108

5.13 Dynamic nonlinearity-aware method (DNL) β history for the body re-entry problem. 4σ initial errors. 108

5.14 Zoomed-in view of dynamic nonlinearity-aware method (DNL), static partial-update with $\boldsymbol{\beta} = [0.9, 0.9, 0.75]$ single run for body re-entry problem. 4σ initial errors....... 109

5.15 Zoomed-in view of dynamic nonlinearity-aware method (DNL) β history for the body re-entry problem. 4σ initial errors.. 109

5.16 Dynamic nonlinearity-aware method (DNL), static partial-update with $\boldsymbol{\beta} = [1, 1, 0.75]$ single run for body re-entry problem subject to higher initial uncertainties and initial errors.. 110

5.17 Dynamic nonlinearity-aware method (DNL) β history for single run for the body re-entry problem subject to higher initial uncertainties and initial errors.............. 111

5.18 Dynamic covariance-aware partial-update method (DC) state errors for a single run of the body re-entry problem subject to 1 σ initial errors. Estimates from static and DNL methods are shown for comparison. 115

5.19 Dynamic covariance-aware partial-update method (DC) absolute errors for a single run of the body re-entry problem subject to 1 σ initial errors. Errors from static and DNL methods are shown for comparison. 116

5.20 Dynamic covariance-aware partial-update method (DC) β history for single run of the body re-entry problem subject to 1 σ initial errors. β produced via DNL method is shown for comparison. 116

5.21 Dynamic covariance-aware partial-update method (DC) absolute errors for a single run of the body re-entry problem subject to 1 σ initial errors. Errors from static and DNL methods are shown for comparison. 117

5.22 Zoomed-in view for dynamic covariance-aware partial-update method (DC) state errors. History of a single run of the body re-entry problem subject to 1 σ initial errors. Estimates from static and DNL methods are shown for comparison. 118

5.23 Dynamic covariance-aware partial-update method (DC) state errors. History of a single run of the body re-entry problem subject to 3 σ initial errors. Estimates from static and DNL methods are shown for comparison. 119

5.24 Dynamic covariance-aware partial-update method (DC) absolute errors. History of a single run of the body re-entry problem subject to 3 σ initial errors. Errors from static and DNL methods are shown for comparison. 119

5.25 Averaged and sampled standard deviation from 100 Monte Carlo runs for dynamic covariance-aware partial-update method (DC) state errors. Estimates from static method are also shown. DNL method is not included as it does not support initial errors higher than 1 σ. Full update is used as the baseline for updates. 120

5.26 Dynamic covariance-aware partial-update method (DC) β history of a single run of the body re-entry problem subject to 3 σ initial errors. The β history produced by the DNL partial-update method is also included for comparison. 121

5.27 Dynamic covariance-aware partial-update method (DC) state errors. History of a single run of the body re-entry problem subject to moderate uncertainties (around $\begin{bmatrix} 2\sigma_{p0} & 2\sigma_{v0} & 3\sigma_{\beta_0} \end{bmatrix}$) and 3 σ initial errors. Estimates from static and DNL methods are shown for comparison. 122

5.28 Dynamic covariance-aware partial-update method (DC) absolute errors. History of a single run of the body re-entry problem subject to moderate uncertainties (around $\begin{bmatrix} 2\sigma_{p0} & 2\sigma_{v0} & 3\sigma_{\beta_0} \end{bmatrix}$) and 3 σ initial errors. Errors from static and DNL methods are shown for comparison. 123

5.29 Dynamic covariance-aware partial-update method (DC) β history of a single run of the body re-entry problem subject to uncertainties (around $\begin{bmatrix}2\sigma_{p0} & 2\sigma_{v0} & 3\sigma_{\beta_0}\end{bmatrix}$) and 3 σ initial errors. The β history produced by the DNL partial-update method is also included for comparison. 123

5.30 Dynamic covariance-aware partial-update method (DC) state errors. History of a single run of the body re-entry problem subject to high uncertainties (around $\begin{bmatrix}2\sigma_{p0} & 2\sigma_{v0} & 5\sigma_{\beta_0}\end{bmatrix}$) and 3 σ initial errors. Estimates from static and DNL methods are shown for comparison. 124

5.31 Dynamic covariance-aware partial-update method (DC) β history of a single run of the body re-entry problem subject to high uncertainties (around $\begin{bmatrix}2\sigma_{p0} & 2\sigma_{v0} & 5\sigma_{\beta_0}\end{bmatrix}$) and 3 σ initial errors. The β history produced by the DNL partial-update method is also included for comparison. 125

5.32 Zoomed-in view for dynamic covariance-aware partial-update method (DC) state errors. History of a single run of the body re-entry problem subject to high uncertainties (around $\begin{bmatrix}2\sigma_{p0} & 2\sigma_{v0} & 5\sigma_{\beta_0}\end{bmatrix}$) and 3 σ initial errors. Estimates from static and DNL methods are shown for comparison. 125

6.1 Relationship between World (W^+), IMU (I^+), camera (C^+) and position of the i^{th} feature found in the arUco target. The arUco reference frame is considered to be the world frame W^+. 139

6.2 Simulated trajectory for a typical IMU-camera calibration run (in blue) and image features (in red). 144

6.3 IMU-camera attitude error for a typical simulation run. Initial condition is a random draw. MEKF and PU-MEKF use the same initial condition. 145

6.4 IMU-camera lever arm errors for a typical simulation run. Initial condition is a random draw. MEKF and PU-MEKF use the same initial condition. 147

6.5 IMU global position error for a typical simulation run. Initial condition is a random draw. MEKF and PU-MEKF use the same initial condition. 147

6.6 Monte Carlo runs for MEKF 149

6.7 Monte Carlo runs for PU-MEKF 150

6.8 Measurement residuals in pixels (px). The residuals indicate filter consistency. These residuals are employed to validate filter results since no true values for the calibration parameters are available. 153

6.9 Lever arm hardware calibration result. The lever arm is considered to be the position vector of the camera with respect to the IMU reference frame. 154

6.10 Position of the IMU with respect to the inertial frame hardware calibration result. ... 155

6.11 Velocity of the IMU with respect to the inertial frame hardware calibration result. ... 156

6.12 Block diagram of the REEF Estimator framework. Reprinted with permission from [2]. ... 159

6.13 The body and body-level frames are co-located and share the same yaw angle. The XY plane in the body-level frame is parallel to the inertial frame XY plane. Reprinted with permission from [2]. ... 160

6.14 Flowchart for the XY and Z PID controllers. Reprinted with permission from [2]. ... 165

6.15 Experimental quadrotor platform. Reprinted with permission from [2]. ... 167

6.16 XY velocity and altitude from the REEF Estimator. The estimates are compared with the ground truth from a motion capture system. Reprinted with permission from [2]. ... 169

6.17 Closed-loop performance of the multirotor. Reprinted with permission from [2]. ... 171

6.18 Coordinate reference frames utilized. Position of the features with respect to the body, attitude and angular velocities are all coordinatized in the camera frame. Reprinted with permission from [55]. ... 177

6.19 The pinhole camera model projects a 3D coordinate $\mathbf{X}$ (resolved in the camera frame) into the image pixel coordinates $(\tilde{u}_i, \tilde{v}_i)$. The camera center coordinates locate the point p on the image plane, which is the intersection of the image plane and the principal axis. The principal axis is along the Z axis of the camera, towards the scene being captured. Reprinted with permission from [55]. ... 178

6.20 Coordinate reference frames utilized. Position of the features with respect to the body, attitude and angular velocities are all coordinatized in the camera frame. Reprinted with permission from [55]. ... 181

6.21 Angular rates obtained through numerical differentiation of differential rotation angles. The differential rotation angles are extracted from the estimated $\delta\mathbf{R}$ from equation 6.90. In this figure a 60 seconds history of the coarse estimates for the angular velocity components is shown. The empirical average is shown in red. The truth value for this specific experiment was $\omega_y = 2\ deg/s$ and $\omega_x = \omega_z = 0\ deg/s$. Reprinted with permission from [55]. ... 187

6.22 The mock asteroid placed on the turn table. The VICON dots can be observed placed at the top on a mounting plate. Reprinted with permission from [55]. ... 189

6.23 Xtion Live Pro camera and the mock asteroid while recording data. The arbitrary reference frame $\mathbf{a}^+$ (on the body), the body plate $\mathbf{b}^+$, the camera optical $\mathbf{o}^+$ and the camera plate $\mathbf{c}^+$ reference frames, have been drawn. Reprinted with permission from [55]. 189

6.24 Image sequence showing the detected features and three of the position vectors that are being tracked. Reprinted with permission from [55]. 192

6.25 Estimated angular velocities for the target body when initial conditions for the angular rates are set to zero. The angular rates are resolved in the optical reference frame $\mathbf{o}^+$. The bottom plot shows the comparison between angular velocity vector magnitudes. Reprinted with permission from [55]. 193

6.26 Estimated angular velocities for the target body when initial conditions for the angular rates are set to zero. The angular rates are resolved in the optical reference frame $\mathbf{o}^+$. The bottom plot shows the comparison between angular velocity vector magnitudes. Reprinted with permission from [55]. 194

LIST OF TABLES

TABLE Page

3.1 Square root partial-update Schmidt-Kalman filter. Reprinted with permission from [1]. 44

3.2 IMU-camera calibration parameters 52

3.3 Re-entering body parameters 58

3.4 Square root partial-update Schmidt-Kalman filter. Vector measurement processing. . 67

3.5 Flops required for batch measurement processing. Flop count considers matrices symmetry [3] 69

3.6 Flops required for sequential measurement processing. Flop count considers matrices symmetry [3] 69

3.7 Flop advantage of sequential over batch processing. 70

3.8 Conventional and partial-update required flops comparison. 71

4.1 U-D partial-update Schmidt-Kalman filter 81

4.2 Re-entering body parameters 85

4.3 Square root and UD partial-update required flops for propagation and update steps combined. 88

4.4 Conventional UD and UD partial-update required flops for one scalar measurement update. 89

6.1 IMU-camera calibration parameters 144

1. INTRODUCTION

1.1 Motivation

In many state estimation applications where the Kalman filter framework is utilized, estimators are often implemented under the assumption that system model parameters are known without error. Although, for some applications, such consideration may not significantly affect the filter, neglecting parameter uncertainty in many other situations can lead to inaccurate state estimates, overconfident filters, and even filter divergence. The consideration of the uncertainty for model parameters, also called nuisance parameters or static biases, may be handled by several approaches. One possibility is to account for their effect via the typical tuning of the measurement covariance matrix $\mathbf{R}$ or the process noise covariance $\mathbf{Q}$ and amount to be added unknown. Although this approach can be straightforward, it does not appropriately account for the colored property of these parameters. A second alternative, and as that of interest for this work, is to attempt the estimation of the nuisance parameters by appending them to the state vector. If this latter approach is implemented, it can be that an Extended Kalman filter is now needed where a linear Kalman filter sufficed before due to nuisance parameters could then appear, along with the core states, in a nonlinear fashion.

The inclusion of the parameters in the state vector increases the filter's complexity, and, as the state incorporates more nuisance parameters, observability questions also start to arise. Furthermore, attempting the estimation of the nuisance parameters via the augmented state and directly treating them as if they were other core states, may cause divergence problems since the nuisance parameters are in many cases weakly or mildly observable.

1.1.1 The Schmidt-Kalman and the partial-update Schmidt filter

In the spirit of alleviating the difficulties mentioned above when augmenting the state vector with nuisance parameters, and still retaining the Kalman filtering framework to solve the estimation problem, the Schmidt-Kalman filter approach was proposed in [4]. A typical Schmidt-Kalman filter or *consider* Kalman filter, is a standard solution for the application of the Kalman filter on systems where it is required to account for uncertainty in the model and measurement parameters for either static or dynamic systems [5]. In contrast with the typical Kalman filter, the Schmidt modification does not attempt to estimate all of the elements of the augmented state vector, instead, it only estimates the core or main states and just *considers* (uses) the nuisance parameters values and their uncertainty for the filter computations. In this way, the uncertainty of the *considered* states is still reflected in the resulting Kalman filter error distribution, generally producing more consistent estimates.

Recent advances have generalized the consider filter a step further, allowing to update the consider states [6] partially. This partial-update technique has been shown to be effective (and statistically consistent) for the same class of systems where the Schmidt filter is useful. However, this new approach also attempts to estimate the considered states so that the estimation consistency can be improved. Mathematically the partial-update filter is grounded in linear system theory as the Extended or Unscented filters. However, in comparison with the Schmidt filter, which either considers (does not update) or updates a state (fully applies the update), the partial-update approach allows the user to apply the chosen percentage of the nominal update.

1.1.2 partial-update Schmidt filter challenges

Although the partial-update Schmidt-Kalman filter addresses the issues of having a nuisance parameter in an augmented state vector and additionally attempts to estimate the nuisance parameters, this novel approach still presents several challenges. First, it inherits the numerical instability problems a typical Kalman filter suffers from, and as such, its actual implementation in hardware could represent difficulties. Second, the selection of the percentages for partially updating the nuisance states is not well defined, and it has been observed to be system dependent. Additionally, the partial-update approach does not leverage occasions where the nuisance parameters observability, and ignores if some other information is available to attempt improving the parameter estimates, and the overall filter performance. Third, the implementation of the partial-update filter has been limited to simulation, and its real-world applicability scope is mostly unknown yet.

1.2 Literature Review

The Kalman filter technique developed by Rudolf Kalman in the 1960s [7], one of the estimation techniques most used today, was not immediately accepted among Rudolf Kalman's peers when it was first presented. Kalman encountered such a reluctance from his peers that he even chose a mechanical engineering journal to publish his work, instead of an electrical engineering journal [3]. Fortunately for him, his perseverance in presenting his formulations brought them significant popularity and acceptance in alternative fields, and his filter rapidly gained the interests of researchers, including NASA engineers. Shortly after a visit to NASA in the fall of 1960 [8], where Kalman presented his formulations at the Ames Research Center, he met Stanley Schmidt, a member of the Dynamics Analysis Branch. Schmidt's team then was working on the midcourse navigation and guidance for the Apollo circumlunar mission, and since they needed a solution to solve the navigation problems given the computational restrictions of the state-of-the-art [8], Stanley Schmidt quickly recognized the Kalman filter technique as the potential solution for the Apollo. Motivated by Kalman's ideas, S. Schmidt soon would realize that even though Rudolf Kalman's development was originally for linear process and measurement models, he could use the Kalman filter on nonlinear systems if he performed a linearization about a nominal or reference trajectory. Shortly, the NASA staff would also infer that such linearization could be improved if performed about the current estimate (produced by the filter) rather than about a reference trajectory. Thanks to conversations with his peers, especially with Richard Battin from the MIT Instrumentation Laboratory, Schmidt played a crucial role in making the Kalman filter an essential component of the Apollo on-board guidance. However, before NASA engineers were able to utilize the Kalman filter for manned missions to the moon, several concerns needed to be addressed first.

1.3 Numerical issues and factorized formulations

With the increase in popularity of the Kalman filter mainly due to the Schmidt research team's success for the circumlunar midcourse navigation, problems with its implementation also started to arise. NASA studies on the effect on midcourse guidance, when fusing radar and on-board sensing, exhibited problems of divergence for the first time along with numerical stability. According to McGee and Schmidt [9], the issues of numerical stability and divergence of the Kalman filter were not noticed sooner due to presumably the low sensitivity of the testing problem to nonlinearities, round-off errors, unmodelled dynamics, and prior statistics. Although the numerical issues that arose first were attributed to the limited 16-bit fixed computational word length of the computers at the time, it was later confirmed that the Kalman filter formulation (structure of operations) itself also affected its numerical stability.

Researchers would soon realize that the numerical instability was mainly due to the Kalman update step involving the subtraction of two positive definite matrices, which implemented on a finite precision computer can considerably degrade the filter estimates and even fracture the theoretical positive definiteness and symmetry of the covariance matrix, leading to a total failure of the estimator. With the impetus of addressing these issues, several techniques were proposed. The fixes were mainly extra ad-hoc operations after propagation and update steps; the key point was maintaining the symmetry and positive definiteness of the covariance matrix. Specifically, according to [9], among the prosed methods to enforce covariance matrix symmetry, researches tried to enforce symmetry by replacing the lower diagonal terms with the upper diagonal terms (or vice-versa), a technique consisting of averaging corresponding off-diagonal terms, computing correlation coefficients and then checking for correlation coefficients greater than one (corrective method), and to add small positive numbers to the diagonal terms. This latter method being essentially inflation of the covariance matrix. Almost two decades later, these ad-hoc techniques would, in fact, be mathematically proved to be appropriate approaches [10].

Artificially inflating the covariance matrix, was one of the solutions proposed to also“control" the Kalman filter divergence problem that engineers started to face at the time [11], [12]. Although the Kalman filter divergence was mainly attributed to modeling errors, round-off errors also were recognized to affect the filter stability [10].

1.3.1 Square-root filtering

The lack of more fundamental methods to improve the numerical robustness of the Kalman filter when limited hardware specifications were available, was the motivation for many researchers to develop alternative recursive forms of the Kalman equations to improve the precision. The contributions of J.E. Potter from the MIT laboratory were crucial in this regard. According to the literature, Potter was the first researcher in publishing a technique able to increase the numerical robustness of the Kalman filter [13]. A filter that, in fact, was flown on the Apollo manned lunar exploration program [3]. Potter realized that using the Cholesky decomposition, one could factorize the covariance matrix and propagate the Cholesky factor rather than the full covariance matrix. By doing this, he rigorously guaranteed the non-negative definiteness of the covariance matrix after each Kalman filter recursion [14]. Although Potter's formulation assimilated the measurement in a sequential way (vector measurements were sequentially processed) and had the limitation of handling filtering systems with no process noise only, it increased the accuracy of the filter.

After Potter's work, many algorithmic advances and developments soon emerged in the 1970s. Articles extending Potter's formulation to include process noise were made available [15]. Sometime later, even surveys were being published on the numerical Kalman filter issues and the set of techniques available at the time on square root filtering to alleviate them, like the one by Kaminski and Schmidt in [16]. In this paper, Kaminski and Schmidt also recall that round-off errors caused numerical problems with the Kalman filter implementation, but issues may also arise if some components of the state vector are far more observable than others.

1.3.2 The U-D filter formulations

Along with the momentum of square root filtering developments, an alternative presented as square-root free formulations became available in the literature for the Kalman filter: the U-D factorized filter. The U-D Kalman filter mechanization uses the U-D factorization to propagate the error covariance matrix. In practice, this filter decomposes or factorizes the covariance matrix into the product three matrices and propagates two of them. The factorization involves a unit upper-triangular (U), a diagonal (D), and a unit lower-triangular (U^{T}) matrix. Gerald Bierman introduced the U-D formulation in [17], where he shows for a simplified case that his formulation guarantees non-negative-definiteness of the covariance matrix and there was no need to perform square-root operations. Later on, Bierman and Catherine L. Thornton would also publish an article that makes a numerical comparison among the conventional Kalman filter, Potter's square-root form, and the U-D formulations. This paper drew two important conclusions: 1) the superior numerical robustness of the factorized formulations (square root and U-D) against the conventional Kalman filter non-factorized form, 2) the Potter formulation can be computationally expensive in comparison with the standard formulation, while the U-D filter offers computational efficiency and increased numerical robustness [18]. Although the U-D filter's execution time may not be faster than the conventional Kalman filter form, it will be competitive compared with it. Despite this, in general, the square root formulation of the Kalman filter is recommended in the literature as a representation that facilitates analytical developments [19].

Even though factorized formulations were, in part, initially developed to alleviate difficulties when dealing with "problematic" or bad conditioned systems, it was later shown that numerical issues could still occur in well-conditioned systems if the filter was implemented in single-precision [18]. Considering this fact and that is not possible to predict if numerical problems will arise, Bierman recommended using a factorized filter in all applications, especially in embedded systems. While some authors imply that factorized filtering was more intended to be used in the early days when computers were more limited [20], and others mentioned that these forms might be obsolete [21], some researches also believe that a factorized filter is essential [22]. In contrast, others believe they are an important tool to cross-validate filter results, leaving out numerical issues [23].

Since the strong wave of first developments related to factorized filtering, researches have seemed to follow the advice of Bierman. However, and for a few exceptions, today, more than using a factorized filter as being cautious for potential single-precision implementations, researchers see the factorized formulations as a guarantee on the non-negative definiteness of the covariance matrix, and a means to reduce computation error effects, as mentioned in [24]. Thus, current literature focuses more on the use of the factorized formulations along with recent filtering techniques with the objective of *robustify* such filters. In [25], for example, the author introduces a square root unscented Kalman filter for visual monocular simultaneous localization and mapping (SLAM). X.Li, in [26] as well applies the square root EKF formulation for underwater SLAM, and in [27], Tai-shan presents an ensemble Kalman filter that incorporates the square root formulation. For very large systems, a U-D filter approach was reported by JPL in [28] for systems with sparse matrices and large-states. In the work of Shovan in [29] a square root cubature-quadrature Kalman filter is developed and shown to gain numerical robustness. Besides improving numerical stability, in work presented in [30] for applications in mobile devices, it is also shown that faster implementations are obtained since the square root formulation allows using single-precision computations (even if double-precision is available). Further developments in unscented smoothing and angles-only orbital navigation that use square root form, are also available in [31] and Jason Schmidt's masters' thesis [32]. I. Arasaratnam even tests in [33] that the square root formulation appropriately handles a singular covariance matrix and can keep the filter running even with perfect measurements. More recent contributions on the topic include an iterative cubature [34], an unscented Schmidt filter [35], and complementary studies on potential methods to compute the square root of the covariance matrix in [36]. Studies on the robustness by testing factorized filters under different word lengths have also been reported in [37]. Factorized filtering, unsurprisingly, has even found applications out of the engineering field in the econometrics literature as a fix for discrepancies in quantities in the state vector [38]. Moreover, other highly specific factorized filtering techniques for stiff systems as the work presented in [39] are still being investigated. In the aerospace engineering side, GPS vehicle navigation that uses square root formulation also are available in the literature [40], [36], and even adaptive approaches can be found to use factorized filters in [41] and [42]. A good background summary on the square root filtering can be found in [23] and more comprehensive treatment in [43].

Specifically speaking on the U-D filter, this formulation remains being the favorite filter for NASA engineers, as mentioned in one of their reports where a relative navigation filter for an

International State Station hosted payload is developed [44]. Due to its high functionality, the U-D filter has barely undergone modifications since its introduction in Thornton's memorandum in [45]. Nevertheless, some work regarding its implementation methods, for example, can be found in [46]. Also, in a paper by C. Souza [47], the information formulation of the U-D has been introduced. Overall, the U-D filter's attributes provide such numerical robustness and performance that is, in fact, the filter of NASA's Orion vehicle for absolute navigation, as presented in [48]. In this development, as in other similar cases in the literature, the U-D formulation attempts to robustify a base filter; specifically, a consider filter; a technique also attributed to Stanley Schmidt.

1.4 The Schmidt consider filter

Among other developments, Schmidt also invented what is known in the literature as Schmidt or consider filter [49]. As described in McGee and Schmidt [9], in a tool created for NASA, Schmidt included a functionality that allowed to *consider* for parameter model errors on filter performance evaluation. The idea was to consider the effect of errors and their uncertainty, and reflect such "knowledge" on the estimated states. The consider or Schmidt filter, somewhat strangely, has adopted different focuses in the literature since its creation. For example, in [50], it is proposed as a means to deal with biases in the dynamic models. However, as presented in [51] is a filter that allows handling errors in measurement parameters, and in [49] is, in part, introduced as a tool to obtain reduced order systems. For other researchers, the Schmidt filter is just a solution to account for the uncertainty of nuisance parameters, that is, parameters required to perform the state estimation but are not the main states of interest [6]. In any case, the principles of the filter are the same: consider states without estimating them.

The Schmidt filter's use from the parameter's uncertainty consideration perspective has been applied widely and is mainly found in aerospace applications. In [52] for example, the filter is used in a Mars entry navigation filter, in [53] for GPS-based on-board real-time orbit determination, in [54] is used for GNSS-based attitude determination, in [51] for target tracking to account for sensor positioning error, and in computer vision for proximity operations as presented in [55]. Again, regardless of the motivation to use a consider filter, the principle is to "delete" states that are not core states and perform the estimation with the remaining states. More general, and in any event, the Schmidt filter intent is to expand the class of problems the typical Kalman filter can handle well. Because of the Schmidt filter capacity to allow stable estimation of systems with low observable states, it is today a companion workhorse for the EKF.

Recently, developments have generalized the consider filter a step further, allowing the *consider* states to be updated. This technique, known in the literature as the partial-update Schmidt filter, has been shown to be useful for a broader class of systems than the Schmidt filter can cover [6]. This new approach attempts to update consider states partially, and by doing so, the estimation consistency and accuracy can be improved. However, there are still challenges on how and when to allow a consider state to be updated. Also, the question of how much the user can partially update a state is open. Moreover, since the partial-update technique is relatively recent, no hardware validation has been made, and no research towards the increase of numerical robustness has been done (as it also suffers from the numerical problems the conventional Kalman filter faces). The significance of closing these gaps for the partial-update filter is that it would extend the practical applicability of the EKF while retaining the EKF structure, which remains a widely used filter in the world.

Although some other methods that attempt to improve the Schmidt filter behavior for non-well-known prior covariance of the consider states have been published in [56], they do not remain straightforward to implement (do not keep the Kalman filter structure) and do not include the capacity to update considered states. In contrast, the partial-update filter retains the conventional Kalman filter structure and is an easy modification.

1.5 The dissertation objectives and outline

The proposed dissertation seeks to extend the applicability of the Schmidt-Kalman filter framework by increasing its numerical robustness and attempting considered state estimation. Towards the goal of increasing robustness against numerical issues, the square root formulation and the U-D factorized form of the partial-update Schmidt Kalman filter are developed. The numerical problems that these formulations attempt to alleviate are mainly due to round-off error, processing of highly accurate measurements, severe discrepancy among states observability, and bad-conditioned problems. Towards estimation of considered states, techniques that leverage occasions when the system nonlinearities are not so severe are proposed.

Since this work builds up from the partial-update filter concept, the resulting filter formulations inherit the increased tolerance to nonlinearities and uncertainty level; at the cost of a minimal additional computational burden that still allows online execution. Moreover, as the proposed formulations are a fundamental restructuring of the Kalman filter equations, they can be directly leveraged in applications where the Kalman and Extended Kalman filter is already used. The filters proposed in this dissertation are also demonstrated in hardware implementations for aerospace-related applications.

1.5.1 Outline

This research is divided into the proposed main contributions: (1) The development of factorized formulations that increase the numerical robustness of a more general Schmidt-Kalman filter, (2) The establishment of baselines to attempt the estimation of considered states and (3) the implementation of the proposed concepts in hardware for aerospace-related applications.

The dissertation chapters are organized as follows. Chapter 1 provides the literature review on the Schmidt filter and starts introducing the partial-update filter concept. Chapter 2 introduces some notation used throughout the dissertation. It includes a more detailed description of the partial-update Schmidt filter and its stability analysis for linear systems. In this chapter, the straightforward extension to a multiplicative extended Kalman filter is presented, along with a brief discussion on the numerical issues that the partial-update filter can face (given that is grounded in the Kalman filter). Chapter 3 develops a square root formulation of the partial-update Schmidt filter to increase its numerical robustness. Simulated cases for the square root partial-update filter are included. Motivated by the high computational burden that the square root formulation can implicate, Chapter 4 develops an alternative factorized formulation: a U-D factorized based filter. This alternative form's objective is to maintain the partial-update and factorized form benefits while having a less computationally expensive filter. Numerical simulations of this filter are presented. Chapter 5 introduces techniques that allow the user to utilize a partial-update filter, without the need for tuning the partial-update weights, and its functionality is shown via numerical simulations. To show the applicability of the partial-update concept in real systems, Chapter 6 includes the results of its hardware implementation. Finally, Chapter 7 presents the conclusions of this research.

2. A GENERALIZATION OF THE SCHMIDT KALMAN FILTER

This chapter mainly introduces underlying mathematical concepts to be referred to in posterior sections of the dissertation. First, a brief description of the Kalman filter framework is included to establish the filtering context and include most of the nomenclature common among the developments presented. Second, the partial-update Schmidt-Kalman filter, the backbone of this work, is introduced. Third, the partial-update formulation, in its indirect form, is derived. Fourth, a Lyapunov stability analysis of the partial-update filter concept for linear systems, is performed, and finally, a discussion on the potential numerical issues of the partial-update filter is included.

2.1 Discrete Extended Kalman filter framework and notation

This section mainly serves to provide nomenclature utilized in this work in the context of Kalman filtering. The Kalman filter was developed to perform state estimation of linear systems originally. In order to support nonlinear systems, one of the more utilized techniques today is the extended Kalman filter (EKF), which operates on the linearized system equations about the current state estimate. This linear approximation of the system is used with the original Kalman filter equations to propagate and update the state and estimated uncertainty. The linearization process, however, can be a source of significant errors in the resulting estimates of a filter. Section 2.2 gives details on how linearization errors may be better handled with the partial-update approach.

For the interests of this research, the propagation and update equations correspond to the discrete Extended Kalman filter, which allows one to implement the algorithm in digital computers that may not have the power to integrate the continuous dynamics every time step. Next, the discrete Extended Kalman filter framework is summarized without derivation [21].

Let a discrete nonlinear dynamic system with state vector $\mathbf{x}_k \in \mathbb{R}^n$, and measurement vector $\tilde{\mathbf{y}}_k \in \mathbb{R}^m$, be represented by

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

$$\tilde{\mathbf{y}}_k = \mathbf{h}_k(\mathbf{x}_k) + \mathbf{v}_k \ , \tag{2.2}$$

$$\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k) \ , \tag{2.3}$$

$$\mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k) \ , \tag{2.4}$$

where $\mathbf{w}_k \in \mathbb{R}^n$ and $\mathbf{v}_k \in \mathbb{R}^n$ are zero-mean Gaussian white-noise processes, with covariances $\mathbf{Q}_k = \mathbf{E}[\mathbf{w}_k^{\mathrm{T}}\mathbf{w}_k]$ and $\mathbf{R}_k = \mathbf{E}[\mathbf{v}_k^{\mathrm{T}}\mathbf{v}_k]$, respectively; with $\mathbf{u}_{\mathrm{k}} \in \mathbb{R}^r$ being the input vector sequence to the system. The function $\mathbf{h}_k(\mathbf{x}_k)$ is the nonlinear measurement model and the sub-indices k denote time instance.

Performing a first-order Taylor series expansion of Equation (2.1) about the current estimate, $\mathbf{x}_{k-1} = \hat{\mathbf{x}}_{k-1}^{+}$, forming the error dynamics, and computing its expectation, the propagation equation for the $n \times n$ error covariance matrix $\mathbf{P}_k$ results in Equation (2.5) with $\mathbf{F}_k$ given by Equation (2.11):

$$\mathbf{P}_k^{-} = \mathbf{F}_{k-1}\mathbf{P}_{k-1}^{+}\mathbf{F}_{k-1}^{\mathrm{T}} + \mathbf{Q}_{k-1} \ . \tag{2.5}$$

The propagation of the state vector $\hat{\mathbf{x}}_k$, is done with the nonlinear dynamics expected value as

$$\hat{\mathbf{x}}_k^{-} = \mathbf{f}_{k-1}(\hat{\mathbf{x}}_{k-1}^{+}, \mathbf{u}_{k-1}) \ . \tag{2.6}$$

Every time an observation is available, the measurement update step for the state and error covariance is performed through the $n \times m$ Kalman gain, $\mathbf{K}_k$, according to the following set of equations. Note that a linearization of the measurement model has also been performed:

$$\mathbf{K}_k = \mathbf{P}_k^{-}\mathbf{H}_k^{\mathrm{T}}(\mathbf{H}_k\mathbf{P}_k^{-}\mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k)^{-1} \ , \tag{2.7}$$

$$\mathbf{P}_k^{+} = (\mathbf{I} - \mathbf{K}_{\mathbf{k}}\mathbf{H}_k)\mathbf{P}_k^{-} \ , \tag{2.8}$$

$$\hat{\mathbf{y}}_k = \mathbf{h}_k(\hat{\mathbf{x}}_k^{-}) \ , \tag{2.9}$$

$$\hat{\mathbf{x}}_k^{+} = \hat{\mathbf{x}}_k^{-} + \mathbf{K}_k(\tilde{\mathbf{y}}_k - \hat{\mathbf{y}}_k) \ , \tag{2.10}$$

where

$$\mathbf{F}_k = \frac{\partial \mathbf{f}_k}{\partial \mathbf{x}}\bigg|_{\hat{\mathbf{x}}_{k-1}^-}, \tag{2.11}$$

and

$$\mathbf{H}_k = \frac{\partial \mathbf{h}_k}{\partial \mathbf{x}}\bigg|_{\hat{\mathbf{x}}_k^-} \quad . \tag{2.12}$$

Here, the hat notation, i.e. $[\hat{\cdot}]$, denotes an expected or estimated value. The notations $[\cdot]^+$ and $[\cdot]^-$ refer to posterior and prior values, respectively.

The set of equations (2.1) to (2.12), constitute the Extended Kalman filter framework. The following chapters adopt the nomenclature here presented, and additional information will be introduced if needed.

2.2 The partial-update filter concept

2.2.1 The Schmidt-Kalman filter

When a Kalman filter state vector involves system parameters or weakly observable states, estimating them in a traditional way can be problematic. In that scenario, the direct use of an EKF can be negatively impacted further, if the system involves measurement and process model nonlinearities, high uncertainties, or some combination, to the extent of leading to estimates degradation or even filter divergence. In such cases, the alternative Schmidt-Kalman filter becomes particularly useful. The Schmidt approach consists of not estimating such the problematic states but maintaining their values and respective covariances fixed; allowing it to behave more linearly and builds more appropriate cross-correlation terms with the core states. In other words, the Schmidt approach enables the filter designer to *consider* the uncertainties of certain states into the Kalman filter solution without attempting to estimate them. By doing so, the class of problems where the conventional Kalman filter framework is useful is broadened. The problematic parameters, or those states that complicate the estimation process if treated as a "traditional" state, are often referred as nuisance states or nuisance parameters. These nuisance parameters, although are often not the main states of interest, their refinement is needed to improve the overall filtering solution.

Mathematically, the conventional formulation of the Schmidt filter starts by partitioning the state vector, $\hat{\mathbf{x}}^{-}$, measurement matrix,$\mathbf{H}$, and Kalman gain $\mathbf{K}$ into states and parameters as

$$\hat{\mathbf{x}}^{-} = \begin{bmatrix} \hat{\mathbf{x}}_x^{-} \\ \hat{\mathbf{x}}_p^{-} \end{bmatrix} , \tag{2.13}$$

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_x & \mathbf{H}_p \end{bmatrix} , \tag{2.14}$$

and

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}_x \\ \mathbf{K}_p \end{bmatrix} . \tag{2.15}$$

Then, the partitioned state vector is substituted into the conventional Kalman filter update equations and the corresponding operations are performed. Finally, and fundamental to the Schmidt filter, the optimal Kalman gain is computed after forcing the parameters' Kalman gain to be zero ($\mathbf{K}_p = \mathbf{0}$).

The resulting update equations after using the optimal gain for the case when $\mathbf{K}_p = \mathbf{0}$, as reported in [5], are

$$\mathbf{P}^{+} = \begin{bmatrix} (I - K_x H_x)\mathbf{P}_{xx}^{-} - K_x H_p \mathbf{P}_{xx}^{-} & (I - K_x H_x)\mathbf{P}_{xp}^{-} - K_x H_p \mathbf{P}_{pp}^{-} \\ [(I - K_x H_x)\mathbf{P}_{xp}^{-} - K_x H_p \mathbf{P}_{pp}^{-}]^{\mathrm{T}} & \mathbf{P}_{pp}^{-} \end{bmatrix} , \tag{2.16}$$

and

$$\begin{bmatrix} \hat{\mathbf{x}}_x^{+} \\ \hat{\mathbf{x}}_p^{+} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{x}}_x^{-} \\ \hat{\mathbf{x}}_p^{-} \end{bmatrix} + \begin{bmatrix} K_x \\ \mathbf{0} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{y}} - H_x \hat{\mathbf{x}}_x^{-} - H_p \hat{\mathbf{x}}_p^{-} \end{bmatrix} . \tag{2.17}$$

From Equations (2.16)-(2.17) it can be clearly seen that on the Schmidt approach, 1) the cross-correlation terms of the covariance matrix become updated and they account for parameter uncertainty, and 2) both parameters not their respective uncertainties are updated.

From Equations (2.16)-(2.17) it can be clearly seen that on the Schmidt approach, 1) the cross-correlation terms of the covariance matrix become updated and they account for parameter uncertainty, and 2) parameters and their respective uncertainties are not updated. The Schmidt filter is popular mainly because it is easy to implement, and it is often sufficient to alleviate issues when dealing with nuisance parameters. Moreover, since the parameters are not estimated, it offers some computational advantage compared to the conventional Kalman filter (KF).

The *considering* approach that allows the Schmidt filter to work on the described scenarios also limits its capabilities. First, while the Schmidt filter can cope with nuisance parameters and system nonlinearities via *considering* them, the Schmidt approach generally cannot react if the nuisance parameters change or slowly vary. Second, if the considered parameters are tightly related to the overall system performance, the Schmidt approach will intrinsically limit the filter performance as no more information is assimilated to improve the parameter belief, and thus core states' belief. Third, the Schmidt filter by design (not attempting to update parameters) ignores situations where nuisance parameters could have been updated, because their observability increased, or nonlinearities are not so severe, and (negative) parameter impact may not be significant.

With these intrinsic, in part limiting, characteristics of the Schmidt approach in mind, a novel concept was created: the partial-update Schmidt-Kalman filter. The idea of this novel filter is to attempt to gain nuisance's states information while trying to keep Schmidt approach benefits. The next subsection gives a brief background on the partial-update filter.

2.2.2 The partial-update Schmidt-Kalman filter

The partial-update Schmidt-Kalman filter (PSKF or partial-update filter for short) is a recent technique that is useful in accommodating measurement updates in nonlinear systems with mildly observable states, as it is an extension of the Schmidt filter. Examples of successful implementations of the partial-update Kalman filter can be found in [55], [57], and [58]. This concept is, in fact, the backbone of this dissertation.

The partial-update Schmidt-Kalman filter [6] is a straightforward modification of the Schmidt Kalman filter that effectively increases the range of uncertainties and associated nonlinearities that the filter can tolerate (compared to the EKF or Schmidt filter) while still producing accurate state estimates with appropriate covariance bounds. The approach does so with almost no extra computational cost and maintains the linear system theory's desirable underpinnings, generating unbiased and consistent results, just as the Kalman or Schmidt filter does. Moreover, the technique is extensible to the UKF [59] and other minimum mean square error approaches.

Effectively, in contrast with the Schmidt filter, the partial-update uses a percentage of the nominal Kalman update to correct the nuisance states. In fact, the formulation of the partial-update filter allows to apply partial updates to any state. A commensurate update percentage is also reflected in the error covariance update, as seen in Equation (2.19). The partial-update is expressed in an element-wise fashion for the states as

$$\hat{\mathbf{x}}_i^{++} = \gamma_i \hat{\mathbf{x}}_i^{-} + (1 - \gamma_i)\hat{\mathbf{x}}_i^{+} \ , \tag{2.18}$$

and for the covariance as

$$\mathbf{P}_{ij}^{++} = \gamma_i \gamma_j \mathbf{P}_{ij}^{-} + (1 - \gamma_i \gamma_j)\mathbf{P}_{ij}^{+} \ . \tag{2.19}$$

Where the update percentages or weights are represented by β_i, and related to γ_i via

$$\gamma_i = 1 - \beta_i \ . \tag{2.20}$$

The notation $[\,\cdot\,]^{++}$ denotes the partial-update value that will overwrite the full state estimates from the conventional Kalman equations to be used at the next propagation step.

In words, the partial-update blends the updated (posterior) vector $\hat{\mathbf{x}}_k^{+}$ computed with Equation (2.10) with the prior state vector obtained with Equation (2.6) via an update or percentage weight β. The partial-update covariance equation (2.19) follows the same idea. The weight $\beta_i \in [0, 1]$ in Equation (2.20), can be thought as the percentage of the updated state being used. Then if β_i equals zero, the Kalman update is totally dropped and the prior is completely kept (Schmidt filter), whereas $\beta_i = 1$ corresponds to a regular EKF (full) update; however, the weight can be set anywhere in between. The notation $\boldsymbol{\beta}$ referring to a vector containing the elements β_i for $i = 1, 2, ..., n$, or alternatively, a diagonal matrix $\boldsymbol{\beta} = \mathrm{diag}[\beta_1, \beta_2, \ldots, \beta_n]$, is used throughout the dissertation as convenient. In any case, n, is the total number of states in the filter (as any state can be partially updated).

While there are more efficient ways to implement the partial-update, specifically by modifying the update step itself, this implementation is still reasonably efficient. Moreover, although Equations (2.18) and (2.54) can be expressed in terms of $\boldsymbol{\beta}$ only, the equations presented are convenient for both proofs and a more natural discussion (i.e. describing weights in terms of the percentage of the update to be applied). The selection of the percentages β is system dependent, but heuristically fast, or core, or more observable states are updated with values of 1, whereas nuisance or weakly observable states, are partially updated with values that can be anywhere inside the permissible range. Methods for online partial-update weight selection are presented in Chapter 5.

The partial-update filter was originally developed to be used within a filter that uses an additive correction, however, and especially in the aerospace community, the use of filters involving attitude multiplicative corrections is common. In the interest of this work, the partial-update formulation is next incorporated into the widely adopted multiplicative extended Kalman filter (MEKF), extending the partial-update application to multiplicative attitude filtering.

2.3 The partial-update for the indirect Kalman filter

The quaternion or Euler parameters, often the chosen attitude parametrization, conveniently provides a singularity-free, although non-unique orientation representation [60]. Although, some care has to be taken since it is an attitude over-parametrization, and a unit norm constraint needs to be preserved to ensure a correct attitude representation, well-known analytical and numerical techniques exist to deal appropriately with quaternion-based attitude developments [61].

Quaternions are very popular in navigation systems and are often utilized within a Kalman filter [62],[63],[64],[65]. Specifically, they are used in an indirect Kalman filter formulation as to preserve the unit-norm constraint and avoid a singular covariance matrix. This indirect Kalman filter formulation is referred to as the Multiplicative Extended Kalman filter (MEKF). This filter operates on the state error, instead of directly using the state dynamics and it updates attitude though a multiplicative correction. Via the MEKF, it is possible to implement a state estimator that benefits from the quaternion representation while satisfying the unit-norm constraint in a built-in fashion.

The partial-update concept can also be applied within the MEKF to fuse both approaches' benefits. Although it has a slightly different interpretation due to the MEKF's indirect formulation, the underlying meaning of partially using the nominal Kalman update still holds. The derivation of the partial-update MEKF, or PU-MEKF for short, is developed next. This formulation is the one implemented on the simulated and hardware implementation cases included in this dissertation.

2.3.1 Indirect filtering

Previous to describing the PU-MEKF update step, indirect filtering, and the conventional MEKF update step are first briefly discussed.

2.3.1.0.1 Indirect filter formulation. Direct incorporation of the attitude quaternion into the filter state comes with two issues. First, the unit quaternion constraint implies state dependence, and thus a singular covariance (theoretically correct; however, it can cause numerical instabilities). Second, the Kalman update operation involves additions; however, the addition operation is not defined for quaternions, and the unit-norm constraint may be violated. In order to overcome these issues, the indirect Kalman filter formulation can be employed [66].

The use of an indirect formulation means that the filter does not directly use/produce the estimate of the state vector; rather, it uses/produces the estimated error of the state vector. And then, with an estimate of the error in hand, the estimate of the actual variable can be recovered. In other words, the PU-MEKF (and MEKF) estimates the departure of the states from the true values. To perform the filtering in this *mode*, the PU-MEKF uses the error dynamics model, instead of the dynamics model.

In the MEKF framework, two definitions of error are used: additive and multiplicative error. In general, additive errors are associated with states other than quaternions, whereas multiplicative errors are associated with the quaternion states. The definition of the error for additive states is that from Equation (2.21), and the definition for multiplicative error is that from Equation (2.22).

$$\delta\mathbf{x} = \mathbf{x} - \hat{\mathbf{x}}\,. \tag{2.21}$$

$$\mathbf{C}(\delta\bar{q}) = \mathbf{C}({}^{I}_{W}\bar{q})\mathbf{C}({}^{I}_{W}\hat{\mathbf{q}})^{\mathrm{T}}\,. \tag{2.22}$$

Intuitively, the additive error, $\delta\mathbf{x}$, is simply computed as the difference between the true value, $\mathbf{x}$, and estimated value, $\hat{\mathbf{x}}$. However, as the addition operation is not defined for rotations, a rotation error is defined in a multiplicative manner as in Equation (2.22). That is, the multiplicative error is defined as the (small) rotation $\mathbf{C}(\delta\bar{q})$ that represents the *rotational difference* between the true and estimated attitude. In other words, the (small) rotation error $\mathbf{C}(\delta\bar{q})$, is the required rotation that when compounded with the current estimated attitude $\mathbf{C}({}^{I}_{W}\hat{\mathbf{q}})$, results in the true attitude $\mathbf{C}({}^{I}_{W}\bar{q})$. This is readily seen if Equation (2.22) is written as,

$$\mathbf{C}({}^{I}_{W}\bar{q}) = \mathbf{C}(\delta\bar{q})\mathbf{C}({}^{I}_{W}\hat{\mathbf{q}})\,. \tag{2.23}$$

Equation (2.21) may alternatively be rearranged in terms of the estimate and error variables as,

$$\mathbf{x} = \delta\mathbf{x} + \hat{\mathbf{x}}\,. \tag{2.24}$$

Equations (2.23) and (2.24) are in fact, the equations that are used to recover the posterior estimates after processing a measurement within the PU-MEKF (or MEKF). Equations (2.23) and (2.24), reveal how the indirect formulation is used: 1) the indirect filter produces the estimates for the errors, $\mathbf{C}(\delta\bar{q})$ and $\delta\mathbf{x}$, and 2) these estimated errors are now combined with the current estimates $\mathbf{C}({}^{I}_{W}\hat{\mathbf{q}})$ and $\hat{\mathbf{x}}$, to produce the improved estimate (posterior).

2.3.2 The conventional MEKF update

Before incorporating the partial-update within the MEKF, the standard MEKF update itself is briefly revisited. The development of the PU-MEKF is deferred for the next section.

After a measurement is received, the measurement update can be performed via standard Kalman filter equations. At time $t = k$, the Kalman gain, and posterior covariance, $\mathbf{P}$, are obtained with [21]

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^{\mathrm{T}} (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k)^{-1} , \tag{2.25}$$

$$\mathbf{P}_k^+ = (\mathbf{I} - \mathbf{K_k} \mathbf{H}_k) \mathbf{P}_k^- , \tag{2.26}$$

and the error state update via

$$\delta \hat{\mathbf{x}}_k^+ = \mathbf{K}_k (\tilde{\mathbf{y}}_k - \hat{\mathbf{y}}_k) = \mathbf{K}_k \mathbf{r} . \tag{2.27}$$

Although the error state estimate, $\delta \hat{\mathbf{x}}_k^+$, is in fact the MEKF output, recall that the interest is to recover the actual state estimate $\hat{\mathbf{x}}$. This is accomplished in two steps. First, the components of the computed Kalman correction, $\delta \hat{\mathbf{x}}_k^+$, are split into additive ($\delta \mathbf{x}_{additive}^+$) and multiplicative ($\delta \hat{\theta}^+$) corrections:

$$\mathbf{K}_k \mathbf{r} = \begin{bmatrix} \delta \hat{\theta}^+ \\ \delta \hat{\mathbf{x}}_{additive}^+ \end{bmatrix}_k . \tag{2.28}$$

Second, the actual states are recovered by applying each type of error definition separately. In this manner, the posterior of the actual additive states is obtained by

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \delta \hat{\mathbf{x}}_{additive_k}^+ , \tag{2.29}$$

while the posterior of actual quaternion states through quaternion multiplication as

$$\bar{q}_k^+ = \delta \bar{q}_k \otimes \bar{q}_k = \begin{bmatrix} \frac{1}{2} \delta \boldsymbol{\theta}_k^+ \\ 1 \end{bmatrix} \otimes \bar{q}_k . \tag{2.30}$$

After the quaternion multiplicative updates take place, a brute force re-normalization is performed.

2.3.3 The partial-update within the Multiplicative EKF

In this section, the partial-update step for the MEKF is derived. The derivation is focused on the multiplicative correction only since the additive correction remains unaltered with respect to the original partial-update form. The derivation shows that when the indirect filter formulation is used, the multiplicative partial-update can be interpreted as a special case of the original partial-update form. Furthermore, it is found that the partial-update for the MEKF requires a slightly different implementation from that of the original partial-update.

2.3.3.1 The PU-MEKF

Direct use of the partial-update formulation in a filter involving state quaternions, would require a partial-update of the form (ignoring time indices for clarity)

$$\bar{q}_i^{++} = \gamma_i \bar{q}_i^- + (1-\gamma_i)\bar{q}_i^+ \ , \tag{2.31}$$

however, this represents two main inconveniences. First, it is not a multiplicative update, making it inconsistent with the multiplicative error definition. Second, if the partial-update is performed in this manner, the quaternion unit norm can be violated. For a multiplicative formulation, however, these issues can be addressed as follows.

Consider the alternative expression of the original partial-update equations in terms of the prior expected state, $\hat{\mathbf{x}}_i^-$ as,

$$\begin{aligned}
\hat{\mathbf{x}}_i^{++} &= \gamma_i \hat{\mathbf{x}}_i^- + (1-\gamma_i)\hat{\mathbf{x}}_i^+ && (2.32)\\
&= \gamma \hat{\mathbf{x}}_i^- + (1-\gamma_i)(\hat{\mathbf{x}}_i^- + \mathbf{K}_i\mathbf{r}) && (2.33)\\
&= \gamma_i \hat{\mathbf{x}}_i^- + \hat{\mathbf{x}}_i^- + \mathbf{K}_i\mathbf{r} - \gamma_i\hat{\mathbf{x}}_i^- - \gamma_i\mathbf{K}\mathbf{r} && (2.34)\\
&= \gamma_i \hat{\mathbf{x}}_i^- - \gamma_i\hat{\mathbf{x}}_i^- + \hat{\mathbf{x}}_i^- + (1-\gamma_i)\mathbf{K}_i\mathbf{r}) && (2.35)\\
&= \hat{\mathbf{x}}_i^- + (1-\gamma_i)\mathbf{K}_i\mathbf{r} && (2.36)\\
&= \hat{\mathbf{x}}_i^- + \bar{\beta}_i\mathbf{K}_i\mathbf{r} \ , && (2.37)
\end{aligned}$$

with $\mathbf{K}_i$ being the i^{ith} row of the Kalman gain $\mathbf{K}$. Equation (2.37) suggests that even before computing the posterior state, the partial update can be applied directly on the correction term, $\mathbf{K}_i\mathbf{r}$, through $\bar{\beta}_i$. This means that the partial-update can take place in the composition of the multiplicative correction for a filter with multiplicative correction (the MEKF in this case). To elaborate on this let the quaternion correction $\delta\bar{q}$ from Equation (2.30) be decomposed into three single small rotations (small angular corrections), through the ϕ, θ and ψ angles. Also, suppose that such angle corrections are partially applied (in a multiplicative fashion) to an estimated quaternion ${}^{I}_{W}\hat{\mathbf{q}}$, and they appear scaled by β factors as,

$$\begin{aligned} \bar{q} &= \delta\bar{q} \otimes {}^{I}_{W}\hat{\mathbf{q}} && (2.38) \\ &= [\delta q_1 \otimes \delta q_2 \otimes \delta q_3] \otimes {}^{I}_{W}\hat{\mathbf{q}} && (2.39) \\ &= \begin{bmatrix} \sin\frac{\beta_1\phi}{2} \\ 0 \\ 0 \\ \cos\frac{\beta_1\phi}{2} \end{bmatrix} \otimes \begin{bmatrix} 0 \\ \sin\frac{\beta_2\theta}{2} \\ 0 \\ \cos\frac{\beta_2\theta}{2} \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 0 \\ \sin\frac{\beta_3\psi}{2} \\ \cos\frac{\beta_3\psi}{2} \end{bmatrix} \otimes {}^{I}_{W}\hat{\mathbf{q}}\,. && (2.40) \end{aligned}$$

Now, since the angle corrections are considered to be very small for the PU-MEKF formulation, then $\delta\bar{q}$ is small (which is originally the case for the MEKF), which means $\sin\frac{\phi}{2} \approx 0$ and $\cos\frac{\phi}{2} \approx 1$. Under this considerations, the quaternion multiplication (as defined in [66]) of the three single small rotations gives

$$\bar{q} = \begin{bmatrix} \frac{\beta_1\phi}{2} \\ 0 \\ 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ \frac{\beta_2\theta}{2} \\ 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 0 \\ \frac{\beta_3\psi}{2} \\ 1 \end{bmatrix} \otimes {}^{I}_{W}\hat{\mathbf{q}} = \begin{bmatrix} \frac{\beta_1\phi}{2} - \frac{\beta_2\theta\psi}{4} \\ \frac{\beta_2\theta}{2} - \frac{\beta_1\theta\phi}{4} \\ \frac{\beta_3\psi}{2} - \frac{\beta_2\phi\theta}{4} \\ \frac{\beta_1\beta_2\beta_3\phi\psi\theta}{8} + 1 \end{bmatrix} \otimes {}^{I}_{W}\hat{\mathbf{q}}\,. \qquad (2.41)$$

By retaining first-order terms only, results in

$$\bar{q} \approx \begin{bmatrix} \frac{\beta_1 \phi}{2} \\ \frac{\beta_2 \theta}{2} \\ \frac{\beta_3 \psi}{2} \\ 1 \end{bmatrix} \otimes {}^{I}_{W}\hat{\mathbf{q}} = \begin{bmatrix} \frac{1}{2}\boldsymbol{\beta}\delta\boldsymbol{\theta} \\ 1 \end{bmatrix} \otimes {}^{I}_{W}\hat{\mathbf{q}} \ , \tag{2.42}$$

where

$$\boldsymbol{\beta} = \mathrm{diag}[\beta_1, \beta_2, \beta_3] \ , \tag{2.43}$$

and

$$\delta\boldsymbol{\theta} = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} \ . \tag{2.44}$$

This indicates that by scaling the correction term $\delta\boldsymbol{\theta}$ the partial-update concept can be applied to the quaternion state, and that this is valid up to a first-order approximation, which holds under the MEKF assumptions.

Following the alternative partial-update formulation from Equation (2.37), the multiplicative partial-update can be written as

$$\delta\hat{\theta}_i^{++} = \delta\hat{\boldsymbol{\theta}}_i^{-} + \bar{\beta}_i \mathbf{K}_i \mathbf{r} = \delta\hat{\boldsymbol{\theta}}_i^{-} + \bar{\beta}_i \delta\hat{\boldsymbol{\theta}}_i^{+} \ , \tag{2.45}$$

which in virtue of the expectation of the state error being zero ($\mathbf{E}[\delta\hat{\theta}^{-}] = 0$), and the scalars β and $\bar{\beta}$ playing the same role, Equation (2.45) simplifies to

$$\delta\hat{\theta}_i^{++} = \beta_i \delta\hat{\boldsymbol{\theta}}_i^{+} \ , \tag{2.46}$$

or in vector form to

$$\delta\hat{\boldsymbol{\theta}}^{++} = \boldsymbol{\beta}\delta\hat{\boldsymbol{\theta}}^{+} \ , \tag{2.47}$$

with

$$\boldsymbol{\beta} = \mathrm{diag}[\beta_1, \beta_2, \beta_3] \ . \tag{2.48}$$

From this development, it can be concluded that to perform a multiplicative partial-update while maintaining the quaternion unit-norm (up to first order), the partial-update needs to happen when constructing the quaternion error correction $\delta\bar{q}$ and not afterwards. Following partial-update notation, a multiplicative partial-update, of a prior attitude, $\bar{q}^-$, is performed via

$$\bar{q}^{++} = \begin{bmatrix} \frac{1}{2}\boldsymbol{\beta}\delta\hat{\boldsymbol{\theta}}^+ \\ 1 \end{bmatrix} \otimes \bar{q}^- \ . \tag{2.49}$$

Since the traditional MEKF produces the error estimate, $\delta\hat{\boldsymbol{\theta}}^+$, this can easily be substituted into Equation (2.49) to produce the partially updated quaternion, and by accordingly *partial-updating* the covariance matrix, one will have a MEKF converted into a PU-MEKF. Similarly, if $\boldsymbol{\beta}$ is chosen to be identity (full update), one recovers the standard MEKF update from the PU-MEKF formulation.

Although recovering multiplicative and additive states from the error estimates require different operations, the partial additive and multiplicative corrections can be computed in the same step. The user just needs to properly identify the partial-update percentages β's to be used on each state, and form the $\boldsymbol{\beta}$ matrix to perform the partial-update, e.g.

$$\boldsymbol{\beta} = \mathrm{diag}\begin{bmatrix} \beta_{\delta\boldsymbol{\theta}_1} & \beta_{\delta\boldsymbol{\theta}_2} & \beta_{\delta\boldsymbol{\theta}_3} & \beta_{w_{\mathbf{P}_I 1}} & \dots & \beta_{\delta\boldsymbol{\alpha}_3} \end{bmatrix} \ , \tag{2.50}$$

and perform the partial-update as

$$\delta\mathbf{x}^{++} = \boldsymbol{\beta}\mathbf{K}(\tilde{\mathbf{y}} - \hat{\mathbf{y}}) = \boldsymbol{\beta}\mathbf{K}\mathbf{r} \ , \tag{2.51}$$

or explicitly

$$\delta\mathbf{x}^{++} = \boldsymbol{\beta}\mathbf{K}\mathbf{r} = \begin{bmatrix} \boldsymbol{\beta_{\delta\theta}}\delta\hat{\theta}^{+} \\ \boldsymbol{\beta_{additive}}\delta\hat{\mathbf{x}}^{+}_{additive} \\ \boldsymbol{\beta_{\delta\alpha}}\delta\hat{\alpha}^{+} \end{bmatrix} . \tag{2.52}$$

Once the partial-update posterior state error, $\delta\mathbf{x}_k^{++}$, is computed, the actual state estimates can be recovered by following additive and multiplicative error definitions.

Finally, since $\mathbf{E}[\delta\mathbf{x}] = 0$, the covariance expression for the error state is

$$\mathbf{P} = \mathbf{E}[(\delta\mathbf{x} - \mathbf{E}[\delta\mathbf{x}])(\delta\mathbf{x} - \mathbf{E}[\delta\mathbf{x}])^{\mathrm{T}}] = \mathbf{E}[\delta\mathbf{x}\delta\mathbf{x}^{\mathrm{T}}] , \tag{2.53}$$

and thus the covariance matrix $\mathbf{P}$ can be partially updated using the original partial-update expression,

$$\mathbf{P}_{ij}^{++} = \gamma_i\gamma_j\mathbf{P}_{ij}^{-} + (1 - \gamma_i\gamma_j)\mathbf{P}_{ij}^{+} , \tag{2.54}$$

or alternatively

$$\mathbf{P}^{++} = \boldsymbol{\Gamma}(\mathbf{P}^{-} - \mathbf{P}^{+})\boldsymbol{\Gamma} + \mathbf{P}^{+} , \tag{2.55}$$

where $\boldsymbol{\Gamma}$ is a diagonal matrix with elements γ_i for $i = 1, 2 \ldots n$ (recall that $\gamma_i = 1 - \beta_i$).

2.3.4 Filter key equations and algorithm

Although the PU-MEKF is a straightforward modification of the MEKF, for the sake of completeness, the following pseudo-code is included to summarize the PU-MEKF generic implementation. Since the computation of the measurement residual matrix is generally application dependent, no specifics about computing this matrix are given in the pseudo-code. However, an application that uses the PU-MEKF and shows its construction is given in Chapter 6.

Algorithm 1 Partial-update Multiplicative Extended Kalman filter (PU-MEKF)

```
Result: Partial-updated posterior estimates $\hat{\mathbf{x}}^{++}$ and $\mathbf{P}^{++}$
Initialize $\hat{\mathbf{x}}^{-}$, $\boldsymbol{\beta}$, $\mathbf{Q}$, $\mathbf{R}$ and $\mathbf{P}^{-}$
for The next time step do
    Obtain propagated state $\hat{\mathbf{x}}_k^{-}$ using system dynamics
    Compute jacobian $\mathbf{F}_k = \frac{\partial \mathbf{f}_k}{\partial \mathbf{x}}\Big|_{\hat{\mathbf{x}}_{k-1}^{-}}$
    Propagate covariance matrix $\mathbf{P}_k^{-}$ using $\mathbf{P}_k^{-} = \mathbf{F}_{k-1}\mathbf{P}_{k-1}^{+}\mathbf{F}_{k-1}^{\mathrm{T}} + \mathbf{Q}_{k-1}$
    if New measurement is available then
        Form the residual measurement matrix $\mathbf{H}_k$
        Compute the Kalman gain $\mathbf{K}_k$ with $\mathbf{K}_k = \mathbf{P}_k^{-}\mathbf{H}_k^{\mathrm{T}}(\mathbf{H}_k\mathbf{P}_k^{-}\mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k)^{-1}$
        Compute the residual $\mathbf{r}_k$ using the incoming measurement, $\tilde{\mathbf{y}}_k$, and expected measure-
        ment, $\hat{\mathbf{y}}_k$, according to $\mathbf{r}_k = \tilde{\mathbf{y}}_k - \hat{\mathbf{y}}_k$
        Compute the correction $\delta\mathbf{x}_k^{++} = \boldsymbol{\beta}\mathbf{K}_k\mathbf{r}_k = \begin{bmatrix} \boldsymbol{\beta_{\delta\theta}}\delta\hat{\theta}_k^{+} \\ \boldsymbol{\beta_{additive}}\delta\hat{\mathbf{x}}_{additive-k}^{+} \end{bmatrix}$
        Use $\boldsymbol{\beta_{\delta\theta}}\delta\hat{\theta}_k^{+}$ to partial-update multiplicative states using $\bar{q}_k^{++} = \begin{bmatrix} \frac{1}{2}\boldsymbol{\beta_{\delta\theta}}\delta\hat{\theta}_k^{+} \\ 1 \end{bmatrix} \otimes \bar{q}^{-}$. This
        recovers attitude estimate.
        Use $\boldsymbol{\beta_{additive-k}}\delta\hat{\mathbf{x}}_{additive-k}^{+}$ to partial-update additive states using $\hat{\mathbf{x}}_k^{+} = \hat{\mathbf{x}}_k^{-} +$
        $\boldsymbol{\beta_{additive-k}}\delta\hat{\mathbf{x}}_{additive-k}^{+}$. This recovers the actual estimates for additive states.
        Via $\mathbf{P}_{ij}^{++} = \gamma_i\gamma_j\mathbf{P}_{ij}^{-} + (1-\gamma_i\gamma_j)\mathbf{P}_{ij}^{+}$, apply partial-update to the covariance matrix $\mathbf{P}$ and
        obtain the current estimate $\mathbf{P}^{++}$
    end
end
```

2.4 Stability analysis of the partial-update filter

The partial-update filter is, in general, an *intermediate* filter lying in between the conventional and the consider filter, and for specific β weight values, it can act as one or another as well. In terms of filter stability, this is relevant, as the partial-update should share conventional and consider filter properties: for linear systems, the estimation error is stable. In this section, the insight of the partial-update being stable is confirmed via the direct Lyapunov method. The stability analysis is based on the developments presented in [21] and [67]. Only the discrete stability analysis is developed, and it is assumed that the true linear system dynamics are described by

$$\mathbf{x}_{k+1} = \mathbf{F}_k \mathbf{x} + \mathbf{B}_k \mathbf{u}_k + \mathbf{G}_k \mathbf{w}_k \; , \tag{2.56}$$

$$\tilde{\mathbf{y}}_{k+1} = \mathbf{H}_{k+1} \mathbf{x}_{k+1} + \mathbf{v}_{k+1} \; . \tag{2.57}$$

Let the following estimation error weighted function be the candidate Lyapunov function

$$\mathbf{V}_k = {\mathbf{e}_k}^{\mathrm{T}} \mathbf{P}_k^{-1} \mathbf{e}_k \; , \tag{2.58}$$

with $\mathbf{e}_k = \hat{\mathbf{x}} - \mathbf{x}$. The necessary condition for the estimation error to be stable in the Lyapunov sense, is that the change in the function $\mathbf{V}_k$ remains at least negative definite after every recursion, in other words,

$$\Delta \mathbf{V}(\mathbf{e}) = {\mathbf{e}_{k+1}}^{\mathrm{T}} \mathbf{P}_{k+1}^{-1} \mathbf{e}_{k+1} - {\mathbf{e}_k}^{\mathrm{T}} \mathbf{P}_k^{-1} \mathbf{e}_k < \mathbf{0} \; . \tag{2.59}$$

To begin, the error $\mathbf{e}_k$ is defined in terms of the transition and measurement matrix. Towards this goal, first recall that the partial update step can be written as

$$\hat{\mathbf{x}}_{k-1}^{+} = \hat{\mathbf{x}}_{k-1}^{-} + (\mathbf{I} - \boldsymbol{\Gamma}) \mathbf{K}_k (\tilde{\mathbf{y}}_{k-1} - \mathbf{H}_{k-1} \hat{\mathbf{x}}_{k-1}^{-}) \; , \tag{2.60}$$

and calling $\tilde{\mathbf{K}}_k = (\mathbf{I} - \boldsymbol{\Gamma}) \mathbf{K}_k$ gives

$$\hat{\mathbf{x}}_{k-1}^{+} = \hat{\mathbf{x}}_{k-1}^{-} + \tilde{\mathbf{K}}_k (\tilde{\mathbf{y}}_{k-1} - \mathbf{H}_{k-1} \hat{\mathbf{x}}_{k-1}^{-}) \; . \tag{2.61}$$

Similarly, the system dynamics can be rearranged as

$$\hat{\mathbf{x}}_k^+ = \mathbf{F}_{k-1}\hat{\mathbf{x}}_{k-1}^+ + \mathbf{B}_{k-1}\mathbf{u} = \mathbf{F}_{k-1}\hat{\mathbf{x}}_{k-1}^+ + \mathbf{F}_{k-1}\tilde{\mathbf{K}}_{k-1}(\tilde{\mathbf{y}}_{k-1} - \mathbf{H}_{k-1}\hat{\mathbf{x}}_{k-1}^-) + \mathbf{B}_{k-1}\mathbf{u}\,. \tag{2.62}$$

Next, the system error can be formed as

$$\mathbf{e}_k = \hat{\mathbf{x}}_k - \mathbf{x}_k = \mathbf{F}_{k-1}\mathbf{e}_{k-1} + \mathbf{F}_{k-1}\tilde{\mathbf{K}}_{k-1}(\tilde{\mathbf{y}}_{k-1} - \mathbf{H}_{k-1}\hat{\mathbf{x}}_{k-1}^-) - \mathbf{G}_{k-1}\mathbf{w}_{k-1}\,, \tag{2.63}$$

and using the measurement equation model in the previous equation, and rearranging terms results in

$$\mathbf{e}_k = \mathbf{F}_{k-1}\mathbf{e}_{k-1} + \mathbf{F}_{k-1}\tilde{\mathbf{K}}_{k-1}\mathbf{H}_{k-1}\mathbf{x}_{k-1} + \mathbf{F}_{k-1}\tilde{\mathbf{K}}_{k-1}\mathbf{v}_{k-1} - \mathbf{F}_{k-1}\tilde{\mathbf{K}}_{k-1}\mathbf{H}_{k-1}\hat{\mathbf{x}}_{k-1}^- - \mathbf{G}_{k-1}\mathbf{w}_{k-1}\,, \tag{2.64}$$

or

$$\mathbf{e}_k = \mathbf{F}_{k-1}(\mathbf{I} - \tilde{\mathbf{K}}_{k-1}\mathbf{H}_{k-1})\mathbf{e}_{k-1} + \mathbf{F}_{k-1}\tilde{\mathbf{K}}_{k-1}\mathbf{v}_{k-1} - \mathbf{G}_{k-1}\mathbf{w}_{k-1}\,. \tag{2.65}$$

Finally, the error at time k can be expressed as

$$\mathbf{e}_k = \mathbf{F}_{k-1}(\mathbf{I} - \tilde{\mathbf{K}}_{k-1}\mathbf{H}_{k-1})\mathbf{e}_{k-1}\,. \tag{2.66}$$

Now, this alternative definition of the estimation error,$\mathbf{e}_k$, is used in the Lyapunov candidate function. This leads to

$$\begin{aligned}
\Delta\mathbf{V}(\mathbf{e}) &= \mathbf{e}_{k+1}\mathbf{P}_{k+1}^{-1}\mathbf{e}_{k+1} - \mathbf{e}_k{}^{\mathrm{T}}\mathbf{P}_k^{-1}\mathbf{e}_k & (2.67)\\
&= \mathbf{e}_k{}^{\mathrm{T}}(\mathbf{I} - \tilde{\mathbf{K}}_k\mathbf{H}_k)^{\mathrm{T}}\mathbf{F}_k^{\mathrm{T}}\mathbf{P}_{k+1}^{-1}\mathbf{F}_k(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{e}_k - \mathbf{e}_k{}^{\mathrm{T}}\mathbf{P}_k^{-1}\mathbf{e}_k & (2.68)\\
&= \mathbf{e}_k{}^{\mathrm{T}}[(\mathbf{I} - \tilde{\mathbf{K}}_k\mathbf{H}_k)^{\mathrm{T}}\mathbf{F}_k^{\mathrm{T}}\mathbf{P}_{k+1}^{-1}\mathbf{F}_k(\mathbf{I} - \tilde{\mathbf{K}}_k\mathbf{H}_k) - \mathbf{P}_k^{-1}]\mathbf{e}_k{}^{\mathrm{T}}\,, & (2.69)
\end{aligned}$$

which translates the problem to show that the bracketed term is negative definite, or at least negative semi-definite for stability,

$$[(\mathbf{I}-\tilde{\mathbf{K}}_k\mathbf{H}_k)^{\mathrm{T}}\mathbf{F}_k^{\mathrm{T}}\mathbf{P}_{k+1}^{-1}\mathbf{F}_k(\mathbf{I}-\tilde{\mathbf{K}}_k\mathbf{H}_k)-\mathbf{P}_k^{-1}]<\mathbf{0}\,. \tag{2.70}$$

Next, the bracketed term is required to be expressed in terms of elements that involve time index k only. To accomplish this a few steps are required.

First, Equation (2.70) is pre-multiplied by $\mathbf{F}_k^{-T}(\mathbf{I}-\tilde{\mathbf{K}}_k\mathbf{H}_k)^{-T}$, and then post-multiplied by $(\mathbf{I}-\tilde{\mathbf{K}}_k\mathbf{H}_k)^{-1}\mathbf{F}_k^{-1}$. This gives rise to,

$$\mathbf{P}_{k+1}^{-1}-\mathbf{F}_k^{-T}(\mathbf{I}-\tilde{\mathbf{K}}_k\mathbf{H}_k)^{-T}\mathbf{P}_k^{-1}(\mathbf{I}-\tilde{\mathbf{K}}_k\mathbf{H}_k)^{-1}\mathbf{F}_k^{-1}<\mathbf{0}\,. \tag{2.71}$$

Pre-multiplying the previous expression by $\mathbf{P}_{k+1}^{-}$ leads to

$$\mathbf{I}-\mathbf{P}_{k+1}^{-}\mathbf{F}_k^{-T}(\mathbf{I}-\tilde{\mathbf{K}}_k\mathbf{H}_k)^{-T}\mathbf{P}_k^{-1}(\mathbf{I}-\tilde{\mathbf{K}}_k\mathbf{H}_k)^{-1}\mathbf{F}_k^{-1}<\mathbf{0}\,. \tag{2.72}$$

Second, the posterior covariance matrix $\mathbf{P}_{k+1}$ is written in *Joseph form* with the objective of using it in the previous equation, and obtain an expression in terms of the same time index. The general covariance update (often called Joseph form), which also applies when using the partial-update concept, is

$$\mathbf{P}_k^{+}=(\mathbf{I}-\tilde{\mathbf{K}}_k\mathbf{H}_k)\mathbf{P}_k^{-}(\mathbf{I}-\tilde{\mathbf{K}}_k\mathbf{H}_k)^{\mathrm{T}}+\tilde{\mathbf{K}}_k\mathbf{R}_k\tilde{\mathbf{K}}_k{}^{\mathrm{T}}\,. \tag{2.73}$$

Third, substituting the Joseph covariance,$\mathbf{P}_k^{+}$, into the traditional covariance propagation equation, a propagated covariance can be obtained as

$$\begin{aligned}\mathbf{P}_{k+1}^{-}&=\mathbf{F}_k\mathbf{P}_k^{+}\mathbf{F}_k^{\mathrm{T}}+\mathbf{G}_k\mathbf{Q}_k\mathbf{G}_k^{\mathrm{T}} \qquad (2.74)\\&=\mathbf{F}_k(\mathbf{I}-\tilde{\mathbf{K}}_k\mathbf{H}_k)\mathbf{P}_k^{-}(\mathbf{I}-\tilde{\mathbf{K}}_k\mathbf{H}_k)^{\mathrm{T}}\mathbf{F}_k^{\mathrm{T}}+\mathbf{F}_k\tilde{\mathbf{K}}_k\mathbf{R}_k\tilde{\mathbf{K}}_k{}^{\mathrm{T}}\mathbf{F}_k^{\mathrm{T}}+\mathbf{G}_k\mathbf{Q}_k\mathbf{G}_k^{\mathrm{T}}\,.\end{aligned}$$

Finally, substituting Equation (2.74) into Equation (2.72), the equation in terms of index k only, is obtained,

$$-\,[\mathbf{F}_k\tilde{\mathbf{K}}_k\mathbf{R}_k\tilde{\mathbf{K}}_k{}^{\mathrm{T}}\mathbf{F}_k^{\mathrm{T}} + \mathbf{G}_k\mathbf{Q}_k\mathbf{G}_k^{\mathrm{T}}][\mathbf{F}_k^{-T}(\mathbf{I}-\tilde{\mathbf{K}}_k\mathbf{H}_k)^{-T}\mathbf{P}_k^{-1}(\mathbf{I}-\tilde{\mathbf{K}}_k\mathbf{H}_k)^{-1}\mathbf{F}_k^{-1}] < \mathbf{0}\,. \quad (2.75)$$

Noticing that the bracketed term on the right is a positive definite matrix, the stability analysis further reduces to check for positive definiteness of the left bracketed term

$$-\,[\mathbf{F}_k\tilde{\mathbf{K}}_k\mathbf{R}_k\tilde{\mathbf{K}}_k{}^{\mathrm{T}}\mathbf{F}_k^{\mathrm{T}} + \mathbf{G}_k\mathbf{Q}_k\mathbf{G}_k^{\mathrm{T}}] < \mathbf{0}\,, \quad (2.76)$$

or equivalently

$$-\,[\mathbf{F}_k\mathbf{K}_k(\mathbf{I}-\boldsymbol{\Gamma})\mathbf{R}_k(\mathbf{I}-\boldsymbol{\Gamma})^{\mathrm{T}}\mathbf{K}_k{}^{\mathrm{T}}\mathbf{F}_k^{\mathrm{T}} + \mathbf{G}_k\mathbf{Q}_k\mathbf{G}_k^{\mathrm{T}}] < \mathbf{0}\,, \quad (2.77)$$

or

$$-\,[\mathbf{F}_k\mathbf{K}_k\boldsymbol{\beta}\mathbf{R}_k\boldsymbol{\beta}^{\mathrm{T}}\mathbf{K}_k{}^{\mathrm{T}}\mathbf{F}_k^{\mathrm{T}} + \mathbf{G}_k\mathbf{Q}_k\mathbf{G}_k^{\mathrm{T}}] < \mathbf{0}\,. \quad (2.78)$$

Since the process noise covariance ,$\mathbf{Q}$, is at least positive semi-definite and the measurement noise covariance, $\mathbf{R}$, is positive definite, the factor that defines the filter stability is the matrix $\boldsymbol{\beta}$. In summary,

- If $\boldsymbol{\beta}$ is singular with one, or up to $(n-1)$ diagonal elements being zero, then the partial-update filter will be stable, but not asymptotically stable.
- If $\boldsymbol{\beta}$ has all diagonal entries different from zero, the system is asymptotically stable.
- If $\boldsymbol{\beta}$ is the zero matrix, no update is performed at all, and if this condition is kept filter divergence may be observed.

2.5 Partial-update numerical stability issues

It is very well known and documented in the literature that early after the Kalman filter development, and as its popularity increased, researchers started observing issues as filter performance degradation, divergence, and even negative covariance values for apparently well-posed problems [3], [9], [10], [16]. The desire to maintain the Kalman filter as a reliable online solution for the estimation problem, drove researchers to invest time to investigate such issues, from which it was found that round-off computation errors and ill-conditioned problems, along with finite computer word-length, were the common triggers for such problems to appear. Several solutions for alleviating these inconveniences, consisting of propagating covariance factors, were proposed, widely developed, and adopted. Square root and UD factorized filters were and are, in fact, the most used today. When these solutions are integrated into the backbone of *conventional* filters, it is possible to have more numerically robust, and stable filters, while keeping the underlying filter properties (at the cost of extra computations inherent to the factorized forms).

Although factorized filters can benefit conventional filter formulations, they may not be too common to see in the literature, but they are in use. The infrequent publication of the alternative formulations may be due to the lack of significant or unperceived of numerical issues for the designer to opt for a change in implementation (a common symptom among practitioners as mentioned in [22]). Furthermore, conventional formulations often result in well-behaved filters when used in simulation, and no extra precision seems to be needed. Nonetheless, recent papers start to increasingly incorporate these square root implementations to ensure the positive semi-definiteness of the covariance matrix, but also to leverage modern computers capacities that allow to invest in a more expensive filter form.

The partial-update filter has been observed to work well on simulation, and when appropriately applied, it can outperform the EKF and Schmidt filter. However, the partial-update formulation is an extension of the Kalman filter, and as such, it inherits the numerical problems bound to the Kalman filter formulations.

The research presented in the following chapters (third and fourth), is devoted to integrating factorized formulations and partial-update filter to provide formulations that incorporate the benefits from both concepts, but that either cannot deliver if implemented separately. The result is a set of filters with augmented ability to support high uncertainties and nonlinearities, and more robust against numerical issues. The proposed formulations, although they are not *bullet-proof*, they certainly extend the class of problems the EKF can handle well.

3. SQUARE ROOT PARTIAL-UPDATE SCHMIDT KALMAN FILTER*

3.1 Introduction & Motivation

Shortly after the original Kalman filter paper [7], however, two primary concerns emerged. The first pertained to the numerical precision of the filter and the second pertained to the filter's robustness for nonlinear systems or measurements. Additionally, it was noted that the subtraction that occurs in the covariance measurement update equation could produce numerical issues when high precision measurements were assimilated on finite precision computers. Moreover, it has been noticed that numerical issues may surface when significant mismatches in state observability are present [16] or when there exists considerable discrepancies in the magnitudes among state elements [68].

These challenges motivated researchers to develop alternative recursive forms of the Kalman equations that would be useful in improving the precision when limited hardware specifications were available. These led to the development of an approach that relied on propagating the square root of the error covariance matrix rather than the error covariance matrix itself. The first occurrence of the square root form is attributed to Potter [13]. Potter noticed that using the square root form of the error covariance, he could rigorously enforce non-negative definiteness of the covariance matrix after each recursion [14]. Researchers later generalized Potter's formulation to process vector measurements and to include process noise. Alternative factorizations that allowed one to perform the filter recursion without the explicit use of the square root of the error covariance matrix [18] also became available. Regardless of the form, the main goal of the square root and other factorized filters is to overcome the previously mentioned numerical problems related to direct propagation of the error covariance matrix.

The square root Kalman filter is mainly a covariance reformulation of the standard Kalman equations, and thus it is still a linear filter. Similar to what is done with a traditional Kalman

*Adapted with permission from "Square root partial-update kalman filter", by J. H. Ramos, K. M. Brink, and J. E. Hurtado, presented at the 22nd International Conference on Information Fusion FUSION 2019, [1], Copyright 2019 by the International Society of Information Fusion.

filer, the square root Kalman filter can be applied in nonlinear systems through a linearized model. That is, the square root formulation does not enhance a filter's ability in addressing nonlinearity, it simply improves numerical conditioning.

The partial-update has been demonstrated to be effective on a wide range of filtering applications and the square root implementation has long been shown to improve the filter's numerical qualities. This chapter presents a development that combines the square root and partial-update formulations within the same filter. The rest of the chapter is organized as follows: Background information is provided in Section 2 including a brief explanation on how the square root formulation helps to improve the numerical robustness in filtering. It also presents the associated equations for the particular square root filter version utilized in this chapter, and a description of the partial-update Schmidt-Kalman filter. Section 3.2 provides the derivation of the square root partial-update Kalman filter. Finally, a nonlinear filtering numerical example that uses the square root partial-update filter is presented in Section 3.3. Monte Carlo runs and computational complexity of the filter are also included. Section 3.6 providing a conclusion statement.

3.1.1 Square Root filtering

In the estimation field, square root filtering refers to utilize a square root factorized representation of the error covariance matrix for purposes of propagation and correction of the estimation error. The goal of reformulating filters using such "square roots" or factorizations, is to increase the precision of the filter itself. By operating on the square root of the error covariance, the filter lowers the condition number of the uncertainty matrix (to be discussed shortly), which is then less prone to numerical issues because fewer significant figures are required during the arithmetic operations. A low condition number is always desirable, mainly for the cases where the computer word length is limited (as in embedded systems), or when the filtering problem is poorly conditioned. Although these type of formulations are numerically more robust, it is at the cost of increasing the computational effort. Nevertheless, the amount of extra computations can still be reasonable which allows a factorized filter to be used in many applications [16].

The definition for the "square root" of a matrix, in contrast from scalar quantities, can vary from one reference to another. For the purposes of this research, the definition for the square root of a matrix is based on the idea of finding a matrix $\mathbf{S}$ that satisfies (3.1). $\mathbf{S}$, is what will be referred to as the square root of the error covariance matrix $\mathbf{P}$.

$$\mathbf{P} = \mathbf{S}\mathbf{S}^{\mathrm{T}} \, . \tag{3.1}$$

Specifically, $\mathbf{S}$ is a lower triangular matrix, and $\mathbf{S}^{\mathrm{T}}$ its transposed. Importantly, it can be noticed that the product $\mathbf{S}\mathbf{S}^{\mathrm{T}}$ is naturally symmetric and positive semidefinite, regardless of the value of the lower triangular matrix $\mathbf{S}$. Thus, numerical difficulties that could cause the covariance matrix $\mathbf{P}$ to become non-symmetric or singular, cannot affect the product $\mathbf{S}\mathbf{S}^{\mathrm{T}}$, thus preserving the theoretical properties of the covariance matrix $\mathbf{P}$ (within the machine precision). Also, as for scalars, the square root $\mathbf{S}$ is not unique. That is, there may be several solutions for $\mathbf{S}$. One very well known method to compute the matrix $\mathbf{S}$ is the Cholesky decomposition [69]. The Cholesky method directly outputs the matrix $\mathbf{S}$ that satisfies Equation (3.1). This method requires that the matrix to be factorized is positive definite and symmetric, which holds for $\mathbf{P}$.

The set of equations that correspond to the square root filter are included here. The assumptions for this approach are taken from Equations (2.1)-(2.4). The state propagation in the EKF is done with Equation (2.6), whereas the square root of the covariance is propagated by solving for $\mathcal{T}$ in Equation (3.2) as an alternative to Equation (2.5). The details of how to find matrix $\mathcal{T}$ are given in Section 3.2.2. The Kalman gain in of Equation (2.7) is replaced by the gain from Equation (3.3) while the covariance update, from Equation (2.8), is now accomplished with Equation (3.4). The state measurement update is computed with the standard form of Equation (2.10).

$$\begin{bmatrix} (\mathbf{S}_k^-)^{\mathrm{T}} \\ \mathbf{0} \end{bmatrix} = \mathcal{T} \begin{bmatrix} (\mathbf{S}^+{}_{k-1})^{\mathrm{T}} \mathbf{F}_{k-1}^{\mathrm{T}} \\ \mathbf{Q}_{k-1}^{T/2} \end{bmatrix} \, , \tag{3.2}$$

$$\mathbf{K}_k = a_i (\mathbf{S}^-{}_k) \phi_i \, , \tag{3.3}$$

$$(\mathbf{S}_k^+) = (\mathbf{S}_k^-)(\mathbf{I} - a_i b_i \phi_i \phi_i^{\mathrm{T}}) \, , \tag{3.4}$$

where

$$a_i = \frac{1}{\phi_i^{\mathrm{T}}\phi_i + \mathbf{R}_i} \,, \tag{3.5}$$

$$\phi_i = (\mathbf{S}_k^-)^{\mathrm{T}}\mathbf{H}_i^{\mathrm{T}} \,, \tag{3.6}$$

$$b_i = \frac{1}{1 \pm \sqrt{a_i \mathbf{R}_i}} \,. \tag{3.7}$$

Importantly, it should be noted that this version of square root filter processes the available measurements in a sequential fashion [16]. Thus, for a particular time k when a measurement vector is available, the update Equations (2.10), (3.3) and (3.4) are executed for $i = 1, 2, \ldots, m$ in order to process each measurement in the measurement vector $\tilde{\mathbf{y}}_k \in \mathbb{R}^m$. In other words, when a vector measurement is available, m updates are performed, one update per element in the measurement vector. Very importantly, for nonlinear systems the Jacobians, $\mathbf{F}$ and $\mathbf{H}$ may need to be recomputed after each measurement assimilation.

Due to the sequential nature of the updates, the measurement covariance matrix $\mathbf{R}$ is assumed to be in diagonal form such that $\mathbf{R}_i$ denotes the i^{th} diagonal element that corresponds to the measurement element $\tilde{\mathbf{y}}_i$ and $\mathbf{H}_i$ represents the i^{th} row of $\mathbf{H}$. Finally, positive sign can be chosen in Equation (3.7) to avoid subtraction. Now with the proper context, in the next section the square root partial-update Schmidt-Kalman filter is derived.

3.2 The square root partial-update Schmidt-Kalman filter

The objective here is to combine the benefits of square root filtering and the partial-update approach into one filter that provides an increase in robustness to uncertainties and numerical issues beyond what is provided by either individual formulation. As with many available techniques, this formulation also has its limits, but its significant contribution in added robustness and its simple implementation, makes it very attractive.

The derivation proposed follows Potter's original form [13] as it assimilates the measurements sequentially. The method proposed here, however, handles process noise. For clarity of exposition, the derivation considers that the i^{th} element of the measurement vector is being processed and the indices are omitted for a and b, since it will be clear that $\mathbf{R}_i$ generates a corresponding a_i and b_i. Similarly the index for ϕ is omitted. Recall that the resulting update equations will need to be executed m times in order to process all measurements in the vector $\tilde{\mathbf{y}}_k \in \mathbb{R}^m$.

3.2.1 Measurement update

It has been previously shown in [6], that the partial-update filter is statistically sound; that is, $\mathbf{E}[\mathbf{e}] = 0, \mathbf{E}[\mathbf{e}^2] = 0$ for linear systems. Thus, if a square root form for the covariance partial-update written in Equation (3.1) can be found, this should maintain the statistical consistency once the error covariance matrix is recovered. Specifically this case, the matrix $\mathbf{S}^{++}$ is sought such that allows to write Equation (2.54) as $\mathbf{P}^{++} = \mathbf{S}^{++}\mathbf{S}^{++\mathrm{T}}$. To begin, the partial-update equations are expressed in matrix form. For clarity sake, the time index k is temporarily dropped. First, the partial-update covariance from Equation (2.54) can be reorganized such that

$$\mathbf{P}_{ij}^{++} = \gamma_i\gamma_j(\mathbf{P}_{ij}^{-} - \mathbf{P}_{ij}^{+}) + \mathbf{P}_{ij}^{+} \,. \tag{3.8}$$

Then, recognizing that the first term on the right-hand side of Equation (3.8) can be written as $\mathbf{\Gamma}(\mathbf{P}^{-} - \mathbf{P}^{+})\mathbf{\Gamma}$, where $\mathbf{\Gamma}$ is a diagonal matrix with elements γ_i for $i = 1, 2 \ldots n$ (recall that $\gamma_i = 1 - \beta_i$), the covariance partial-update can be expressed as

$$\mathbf{P}^{++} = \mathbf{\Gamma}(\mathbf{P}^{-} - \mathbf{P}^{+})\mathbf{\Gamma} + \mathbf{P}^{+} \,. \tag{3.9}$$

Now, from the standard EKF equations (ignoring the time indices for ease of notation), $\mathbf{P}^{+} = (\mathbf{I} - \mathbf{K}\,\mathbf{H})\mathbf{P}^{-}$ is incorporated into Equation (3.9).

$$\mathbf{P}^{++} = \mathbf{P}^{+} + \mathbf{\Gamma}(\mathbf{K}\mathbf{H}\mathbf{P}^{-})\mathbf{\Gamma} \,, \tag{3.10}$$

then replacing $\mathbf{K} = \mathbf{P}^{-}\mathbf{H}^{\mathrm{T}}(\mathbf{H}\mathbf{P}^{-}\mathbf{H}^{\mathrm{T}} + \mathbf{R})^{-1}$ results in

$$\mathbf{P}^{++} = \mathbf{P}^{+} + \mathbf{\Gamma}(\mathbf{P}^{-}\mathbf{H}^{\mathrm{T}}(\mathbf{H}\mathbf{P}^{-}\mathbf{H}^{\mathrm{T}} + \mathbf{R})^{-1}\mathbf{H}\mathbf{P}^{-})\mathbf{\Gamma} \,. \tag{3.11}$$

Then, it is required that $\mathbf{P}^- = (\mathbf{S}^-)(\mathbf{S}^-)^\mathrm{T}$ and $\mathbf{P}^+ = (\mathbf{S}^+)(\mathbf{S}^+)^\mathrm{T}$, and considering that the measurements are processed one at a time. Thus, if $\mathbf{H}_i$ is the i-th row of $\mathbf{H}$, and $\mathbf{R}_i$ the i-th diagonal element of $\mathbf{R}$, $a = (\mathbf{H}_i\mathbf{P}^-\mathbf{H}_i^\mathrm{T} + \mathbf{R}_i)^{-1}$ is an scalar. These actions lead to

$$\mathbf{P}^{++} = (\mathbf{S}^+)(\mathbf{S}^+)^\mathrm{T} + \mathbf{\Gamma}(\mathbf{S}^-)(\mathbf{S}^-)^\mathrm{T}\mathbf{H}_i^\mathrm{T}a\mathbf{H}_i(\mathbf{S}^-)(\mathbf{S}^-)\mathbf{\Gamma}\ . \tag{3.12}$$

The sequential processing assumes that $\mathbf{R}$ is already a diagonal matrix. Further, with the purpose of using Potter's measurement update equations form directly, the scalar a is written as follows:

$$a = \frac{1}{\mathbf{H}_i(\mathbf{S}^-)(\mathbf{S}^-)^\mathrm{T}\mathbf{H}_i^\mathrm{T} + \mathbf{R}_i} = \frac{1}{\phi^\mathrm{T}\phi + \mathbf{R}_i}\ , \tag{3.13}$$

where $\phi = (\mathbf{S}^-)^\mathrm{T}\mathbf{H}_i^\mathrm{T}$. Also, from Potter's formulation, the posterior square root error covariance matrix can be obtained as in Equation (3.14). A summary of Potter's equations can be found in [16].

$$\mathbf{S}^+ = (\mathbf{S}^-)(\mathbf{I} - ab\phi\phi^\mathrm{T})\ , \tag{3.14}$$

where b is defined as

$$b = \frac{1}{1 \pm \sqrt{a\mathbf{R}_i}}\ . \tag{3.15}$$

Now, using the definition of the posterior $\mathbf{S}^+$ from Equation (3.14) in Equation (3.12), the partial-update for the error covariance, which is now in terms of the prior $\mathbf{S}^-$, reads

$$\begin{aligned}\mathbf{P}^{++} = [(\mathbf{S}^-)(\mathbf{I} - ab\phi\phi^\mathrm{T})][(\mathbf{S}^-)(\mathbf{I} - ab\phi\phi^\mathrm{T})]^\mathrm{T} +\\ \mathbf{\Gamma}(\mathbf{S}^-)(\mathbf{S}^-)^\mathrm{T}\mathbf{H}_i^\mathrm{T}\sqrt{a}\sqrt{a}\mathbf{H}_i(\mathbf{S}^-)(\mathbf{S}^-)^\mathrm{T}\mathbf{\Gamma}\ .\end{aligned} \tag{3.16}$$

Because Equation (3.16) is the sum of two matrices and each of these matrices are factorized in a square root manner, Equation (3.17) can be set as a candidate square root of $(\mathbf{P}^{++})$.

$$\begin{bmatrix}(\mathbf{S}^{++})^\mathrm{T}\\ \mathbf{0}\end{bmatrix} = \mathbf{T}\begin{bmatrix}(\mathbf{I} - ab\phi\phi^\mathrm{T})^\mathrm{T}(\mathbf{S}^-)^\mathrm{T}\\ \sqrt{a}\mathbf{H}_i(\mathbf{S}^-)(\mathbf{S}^-)^\mathrm{T}\mathbf{\Gamma}\end{bmatrix}\ . \tag{3.17}$$

Where matrix $\mathbf{T} = \begin{bmatrix} \mathbf{T}_1 & \mathbf{T}_2 \end{bmatrix}$ is a $(n+r) \times (n+r)$ orthogonal matrix, with $\mathbf{T}_1$ being $(n+r) \times n$ and $\mathbf{T}_2$ being $(n + r) \times r$ matrices, meaning that

$$\begin{aligned} \mathbf{T}^{\mathrm{T}}\mathbf{T} &= \begin{bmatrix} \mathbf{T}_1^{\mathrm{T}} \\ \mathbf{T}_2^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \mathbf{T}_1 & \mathbf{T}_2 \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{T}_1^{\mathrm{T}}\mathbf{T}_1 & \mathbf{T}_1^{\mathrm{T}}\mathbf{T}_2 \\ \mathbf{T}_2^{\mathrm{T}}\mathbf{T}_1 & \mathbf{T}_2^{\mathrm{T}}\mathbf{T}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \end{aligned} . \tag{3.18}$$

Thus,

$$\begin{bmatrix} (\mathbf{S}^{++})^{\mathrm{T}} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{T}_1 & \mathbf{T}_2 \end{bmatrix} \begin{bmatrix} (\mathbf{I} - ab\phi\phi^{\mathrm{T}})^{\mathrm{T}}(\mathbf{S}^{-})^{\mathrm{T}} \\ \sqrt{a}\mathbf{H}_i(\mathbf{S}^{-})(\mathbf{S}^{-})^{\mathrm{T}}\mathbf{\Gamma} \end{bmatrix} = \begin{bmatrix} \mathbf{W}_{n\times n} \\ \mathbf{0} \end{bmatrix} . \tag{3.19}$$

Then, if an orthogonal matrix $\mathbf{T}$ can be found such that Equation (3.19) produces an $(n \times n)$ upper triangular $\mathbf{W}$ matrix stacked on top of a lower zero $(r \times n)$ matrix, then it can be recognized that the upper triangular $(n \times n)$ matrix $\mathbf{W}$, is actually equal to $(\mathbf{S}^{++})^{\mathrm{T}}$, which is the desired result. The idea behind the use of the orthogonal matrix $\mathbf{T}$, is to use the factorization shown inside the brackets in Equation (3.17). The transformation $\mathbf{T}$, allows one to find a square root of dimension $(n \times n)$. As it may be noted, $\mathbf{S}^{++} = \begin{bmatrix} (\mathbf{S}^{-})(\mathbf{I} - ab\phi\phi^{\mathrm{T}}) & \mathbf{\Gamma}(\mathbf{S}^{-})(\mathbf{S}^{-})^{\mathrm{T}}\mathbf{H}_i^{\mathrm{T}}\sqrt{a} \end{bmatrix}$ can also act as a square root of $\mathbf{P}^{++}$, however this selection would increase the dimension of the problem to a $(n \times (n + r))$ square root matrix, which is not desirable.

In the following development, the product from Equation (3.20) is performed explicitly to show that the candidate form of Equation (3.17) is a valid square root for $\mathbf{P}^{++}$.

$$\begin{bmatrix} (\mathbf{S}^{++}) & \mathbf{0} \end{bmatrix} \begin{bmatrix} (\mathbf{S}^{++})^{\mathrm{T}} \\ \mathbf{0} \end{bmatrix} =$$

$$[\mathbf{T}_1(\mathbf{I}-ab\phi\phi^{\mathrm{T}})^{\mathrm{T}}(\mathbf{S}^-)^{\mathrm{T}}+\mathbf{T}_2\sqrt{a}\mathbf{H}_i(\mathbf{S}^-)(\mathbf{S}^-)^{\mathrm{T}}\mathbf{\Gamma}]^{\mathrm{T}}[\dots] \tag{3.20}$$

$$\begin{aligned}
&=(\mathbf{S}^-)(\mathbf{I}-ab\phi\phi^{\mathrm{T}})\mathbf{T}_1^{\mathrm{T}}\mathbf{T}_1(\mathbf{I}-ab\phi\phi^{\mathrm{T}})(\mathbf{S}^-)^{\mathrm{T}}+\\
&(\mathbf{S}^-)(\mathbf{I}-ab\phi\phi^{\mathrm{T}})\mathbf{T}_1^{\mathrm{T}}\mathbf{T}_2\sqrt{a}\mathbf{H}_i(\mathbf{S}^-)(\mathbf{S}^-)^{\mathrm{T}}\mathbf{\Gamma}+\\
&\mathbf{\Gamma}\sqrt{a}(\mathbf{S}^-)(\mathbf{S}^-)^{\mathrm{T}}\mathbf{H}_i^{\mathrm{T}}\mathbf{T}_2^{\mathrm{T}}\mathbf{T}_1(\mathbf{I}-ab\phi\phi^{\mathrm{T}})(\mathbf{S}^-)^{\mathrm{T}}+\\
&\mathbf{\Gamma}\sqrt{a}(\mathbf{S}^-)(\mathbf{S}^-)^{\mathrm{T}}\mathbf{H}_i^{\mathrm{T}}\mathbf{T}_2^{\mathrm{T}}\mathbf{T}_2\sqrt{a}\mathbf{H}_i(\mathbf{S}^-)(\mathbf{S}^-)^{\mathrm{T}}\mathbf{\Gamma}
\end{aligned} \tag{3.21}$$

$$\begin{aligned}
&=(\mathbf{S}^-)(\mathbf{I}-ab\phi\phi^{\mathrm{T}})\mathbf{T}_1^{\mathrm{T}}\mathbf{T}_1(\mathbf{I}-ab\phi\phi^{\mathrm{T}})(\mathbf{S}^-)^{\mathrm{T}}+\\
&\mathbf{\Gamma}\sqrt{a}(\mathbf{S}^-)(\mathbf{S}^-)^{\mathrm{T}}\mathbf{H}_i^{\mathrm{T}}\mathbf{T}_2^{\mathrm{T}}\mathbf{T}_2\sqrt{a}\mathbf{H}_i(\mathbf{S}^-)(\mathbf{S}^-)^{\mathrm{T}}\mathbf{\Gamma}
\end{aligned} \tag{3.22}$$

$$\begin{aligned}
&=(\mathbf{S}^-)(\mathbf{I}-ab\phi\phi^{\mathrm{T}})(\mathbf{I}-ab\phi\phi^{\mathrm{T}})(\mathbf{S}^-)^{\mathrm{T}}+\\
&\mathbf{\Gamma}\sqrt{a}(\mathbf{S}^-)(\mathbf{S}^-)^{\mathrm{T}}\mathbf{H}_i^{\mathrm{T}}\sqrt{a}\mathbf{H}_i(\mathbf{S}^-)(\mathbf{S}^-)^{\mathrm{T}}\mathbf{\Gamma}\ .
\end{aligned} \tag{3.23}$$

In Equation (3.23) the expression for the partial-update covariance from Equation (3.16) is recovered, which shows that the factorization proposed in Equation (3.17) is actually a square root for $\mathbf{P}^{++}$. Then, the square root partial-update equations are

$$\begin{bmatrix}(\mathbf{S}^{++})^{\mathrm{T}}\\ \mathbf{0}\end{bmatrix}=\begin{bmatrix}\mathbf{T}_1 & \mathbf{T}_2\end{bmatrix}\begin{bmatrix}(\mathbf{I}-ab\phi\phi^{\mathrm{T}})^{\mathrm{T}}(\mathbf{S}^-)^{\mathrm{T}}\\ \sqrt{a}\mathbf{H}_i(\mathbf{S}^-)(\mathbf{S}^-)^{\mathrm{T}}\mathbf{\Gamma}\end{bmatrix}. \tag{3.24}$$

Or equivalently,

$$\begin{bmatrix}(\mathbf{S}^{++})^{\mathrm{T}}\\ \mathbf{0}\end{bmatrix}=\mathbf{T}\begin{bmatrix}(\mathbf{S}^{+})^{\mathrm{T}}\\ \sqrt{a}\phi^{\mathrm{T}}(\mathbf{S}^-)^{\mathrm{T}}\mathbf{\Gamma}\end{bmatrix}=\begin{bmatrix}\mathbf{W}_{n\times n}\\ \mathbf{0}\end{bmatrix}. \tag{3.25}$$

The problem of finding the orthogonal matrix $\mathbf{T}$, but more importantly (for this application), finding the matrix $\mathbf{W}_{n\times n}$, is well known and several well documented algorithms are available in the literature [16], [69], [70]. This work makes use of the algorithm known as the Modified Gram-Schmidt (MGS), which was outlined in [16] specifically for Kalman filtering.

For the implementation of the square root partial-update Kalman filter, the square root of the covariance is being directly propagated, so $\mathbf{S}$ is available at every time step. For the purposes of this development, $\mathbf{S}$ is propagated directly through the concept shown in the following sub-section (3.2.2). Also recall that once the Modified Gram-Schmidt algorithm has been carried out, the matrix $(\mathbf{S}^{++})^{\mathrm{T}}$ has been generated and the matrix $\mathbf{W}_{\mathbf{n}\times\mathbf{n}}$ is simply retained, since it is the solution $(\mathbf{S}^{++})^{\mathrm{T}}$ for the current measurement partial update step. Finally, notice that one could think about or suggest the direct use of Equation (3.8) to perform the time update by transforming the square root $\mathbf{S}$ into the covariance matrix $\mathbf{P}$, and then going back to the square root via the Cholesky decomposition. However, two main issues would be encountered: 1) it will be computationally expensive to go back and forth for relative large systems and, 2) Equation (3.8) involves a subtraction, which would bring back the original concern over numerical issues due to finite precision.

3.2.2 Time update

The propagation of the square root error covariance matrix $\mathbf{S}$, follows the same factorization idea used to compute $(\mathbf{S}^{++})^{\mathrm{T}}$ in Equation (3.25). Recall that the standard propagation of the error covariance matrix is computed as in Equation (2.5), which it is included here for convenience [68].

$$\mathbf{P}_k^- = \mathbf{F}_{k-1}\mathbf{P}_{k-1}^+\mathbf{F}_{k-1}^{\mathrm{T}} + \mathbf{Q}_{k-1}\,. \tag{3.26}$$

Again, since $\mathbf{P}_{k-1}^+$ is symmetric and positive definite, it can alternatively be written as

$$\mathbf{S}_k^-\mathbf{S}_k^{-\mathrm{T}} = \mathbf{F}_{k-1}\mathbf{S}_{k-1}^+\mathbf{S}_{k-1}^{+\ \mathrm{T}}\mathbf{F}_{k-1}^{\mathrm{T}} + \mathbf{Q}_{k-1}^{1/2}\mathbf{Q}_{k-1}^{T/2}\,. \tag{3.27}$$

From which the following factorization can be proposed [71]

$$\begin{bmatrix} (\mathbf{S}_k^-)^{\mathrm{T}} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathcal{T}_1 & \mathcal{T}_2 \end{bmatrix} \begin{bmatrix} (\mathbf{S}^+{}_{k-1})^{\mathrm{T}}\mathbf{F}_{k-1}^{\mathrm{T}} \\ \mathbf{Q}_{k-1}^{T/2} \end{bmatrix} = \begin{bmatrix} \mathcal{W}_{n\times n} \\ \mathbf{0} \end{bmatrix}, \tag{3.28}$$

such that,

$$\mathbf{P}_k^- = \begin{bmatrix} (\mathbf{S}_k^-) & \mathbf{0} \end{bmatrix} \begin{bmatrix} (\mathbf{S}_k^-)^{\mathrm{T}} \\ \mathbf{0} \end{bmatrix} . \tag{3.29}$$

In a similar manner as before, $\mathcal{T} = \begin{bmatrix} \mathcal{T}_1 & \mathcal{T}_2 \end{bmatrix}$ is a $2n \times 2n$ orthogonal matrix to be found (in this case via Modified Gram-Schmidt), and $\mathcal{W}$ is the solution for $(\mathbf{S}_k^-)^{\mathrm{T}}$. Notice that $\mathcal{T}$ is different from $\mathbf{T}$ in general. Thus, performing MGS for Equation (3.28) provides the direct propagation step for the square root error covariance matrix. Equation (3.26) can be recovered from the product in Equation (3.29), which verifies Equation (3.28) as a valid square root for $\mathbf{P}_k^-$. In Table 3.1 the square root partial-update Schmidt-Kalman filter equations are summarized.

Before moving to examples of the new filter, a few remarks about the development are needed. First, note that the real symmetric matrix $\mathbf{Q}^{T/2}$, is required in the propagation step. This is computed by an eigendecomposition procedure which let express $\mathbf{Q} = VDV^T$, where the columns of V are the eigenvectors of $\mathbf{Q}$, and D is a diagonal matrix which entries are the corresponding eigenvalues [72]. In this way, $\mathbf{Q}^{1/2} = VD^{1/2}$, where $D^{1/2}$ is the square root of D. Second, since the measurements are processed sequentially, the filter updates are performed for $i = 1, 2, .., m$ in order to assimilate the observation vector $\tilde{\mathbf{y}}_k$ completely. It is highly important that the measurement matrix is computed with the most current estimate, and thus the corresponding Jacobians are to be updated. That is, every time a sequential measurement is assimilated, the measurement (and transition) matrix needs to be re-evaluated at the most recent state posterior available. Finally, since a decomposition to obtain $\mathbf{Q}^{T/2}$ is required, the user will have to assess the convenience of this square root filter when dealing with time-varying process noise. A similar assessment may be required in the case of a time-varying $\mathbf{R}$ matrix.

3.3 Numerical examples

In this section simulation results that show the effect of using the square root partial-update filter are presented, and compared with the (conventional) square root EKF. A simulation of standard EKF and square root EKF is included as well. The simulations, also show the agreement between

Table 3.1: Square root partial-update Schmidt-Kalman filter. Reprinted with permission from [1].

Model	$\mathbf{x}_k = \mathbf{f}_{k-1}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) + \mathbf{w}_{k-1}$ $\tilde{\mathbf{y}}_k = \mathbf{h}_k(\mathbf{x}_k) + \mathbf{v}_k$ $\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k)$ $\mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k)$
Initialize	$\hat{\mathbf{x}}_0^+ = \mathbf{x}_0$ $\mathbf{S}_0^+ = chol(\mathbf{P}_0^+)$ [69] $\mathbf{Q}_0^{1/2} = eigendec(\mathbf{Q}_0)$ [72] $\mathbf{\Gamma} = \text{diag}(1-\beta_1, 1-\beta_2, ..., 1-\beta_n)$
Propagation	$\hat{\mathbf{x}}_k^- = \mathbf{f}_{k-1}(\hat{\mathbf{x}}_{k-1}^+, \mathbf{u}_{k-1})$ $Perform\ MGS$ [16] for $\begin{bmatrix} (\mathbf{S}_k^-)^{\mathrm{T}} \\ \mathbf{0} \end{bmatrix} = \mathcal{T} \begin{bmatrix} (\mathbf{S^+}_{k-1})^{\mathrm{T}} \mathbf{F}_{k-1}^{\mathrm{T}} \\ \mathbf{Q}_{k-1}^{T/2} \end{bmatrix}$
Gain	$\mathbf{K}_k = a_i(\mathbf{S^-}_k)\phi_i$ $a_i = \dfrac{1}{\phi_i^{\mathrm{T}}\phi_i + \mathbf{R}_i}; \phi_i = (\mathbf{S}^-)^{\mathrm{T}}\mathbf{H}_i^{\mathrm{T}}$
Update	$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\tilde{\mathbf{y}}_k - \hat{\mathbf{y}}_k)$ $(\mathbf{S}_k^+) = (\mathbf{S}_k^-)(\mathbf{I} - a_i b_i \phi_i \phi_i^{\mathrm{T}})$ $b_i = \dfrac{1}{1 \pm \sqrt{a_i \mathbf{R}_i}}$
Partial-Update	$\hat{\mathbf{x}}_i^{++} = \mathbf{\Gamma}\hat{\mathbf{x}}^- + (\mathbf{I} - \mathbf{\Gamma})\hat{\mathbf{x}}^+$ $Perform\ MGS$ [16] for $\begin{bmatrix} (\mathbf{S}_k^{++})^{\mathrm{T}} \\ \mathbf{0} \end{bmatrix} = \mathbf{T} \begin{bmatrix} (\mathbf{S}_k^+)^{\mathrm{T}} \\ \sqrt{a_i}\phi_i^{\mathrm{T}}(\mathbf{S}_k^-)^{\mathrm{T}}\mathbf{\Gamma} \end{bmatrix}$

the square root partial-update and the standard partial-update estimates.

3.3.1 Body re-entering Earth atmosphere

In this numerical example, which is based on the example presented in [73], the altitude, $\mathbf{x}_1$ (in meters), vertical velocity, $\mathbf{x}_2$ (in meters per second), and the constant ballistic parameter, $\mathbf{x}_3$ (with units of 1/meter), of a body re-entering Earth atmosphere from high altitude and with high velocity, are estimated. It is assumed that the body is constrained to fall vertically, and that a range-measurement system delivers discrete measurements. The measurements are considered to be affected by a zero-mean Gaussian white-noise process.

The discretized nonlinear dynamics of the system are

$$x_{1(k)} = x_{1(k-1)} + x_{2(k-1)}\Delta t \ , \tag{3.30}$$

$$x_{2(k)} = x_{2(k-1)} + (e^{\frac{-x_{1(k-1)}}{k_p}} x_{2(k-1)}^2 x_{3(k-1)} - g)\Delta t \ , \tag{3.31}$$

$$x_{3(k)} = x_{3(k-1)} \ , \tag{3.32}$$

and the range measurement model is

$$y(x_1) = \sqrt{d^2 + (x_1 - h_0)^2)} + \mathbf{v}_k \ , \tag{3.33}$$

where $k_p = 6.1 \times 10^3\,\mathrm{m}$ is a constant that relates the air density with the altitude, and $g = 9.81\,\mathrm{m/s^2}$ is the acceleration due to the gravity.

In the measurement equation, $d = 3 \times 10^4\,\mathrm{m}$ is the horizontal distance from the measuring device to the vertical line traced by the falling body, and $h_0 = 3 \times 10^4\,\mathrm{m}$ is the altitude of the measuring device from ground level. The initial uncertainties, while large, are based on [73], but rounded slightly to accommodate SI units. Their values are $\sigma_{x_1} = 300$, $\sigma_{x_2} = 600$, and $\sigma_{x_3} = 0.33$ and the initial guesses for each state were set with a $\pm 1\sigma$ error, while $\mathbf{R} = 300$. All quantities with appropriate units. Although this is not a true random draw it is sufficient to exercise the filter for an example.

Figure 3.1 shows the results for the standard EKF, along with the square root partial-update filter. Both filters are using full measurement updates ($\boldsymbol{\beta} = \begin{bmatrix} 1.0 & 1.0 & 1.0 \end{bmatrix}$) and they begin with estimates within the 3σ bounds. However, once the body is affected by increased drag forces the filter makes erroneous updates and the estimates start deviating outside the appropriate bounds. This inconsistency is the result of relatively large uncertainty on the initial guesses and the non-linear relationship between position, velocity, and the ballistic parameter. Early updates produce slightly inaccurate results, specifically in the ballistic parameter and these errors compound during future propagation and update steps, leading to velocity and then position state and covariance inconsistencies. While this example is from a single run, the results are representative of a typical run for this scenario. It should also be noted that all three plot insets in Figure 3.1 show both the 3σ and -3σ bounds along with the state error for the last second of the simulation. This is apparent in the position plot, but difficult to see in the other two states. The figure clearly demonstrates the inconsistency between the errors present and associated covariance estimates.

If the initial errors from this example were reduced sufficiently for the ballistic parameter, including tighter initial covariance values, the EKF does provide consistent estimates. This fact suggests that poor linearizations are to blame for the filter's poor performance seen in Figure 3.1. It should also be recalled that the square root implementation provides certain numerical robustness, but not necessarily uncertainty robustness, however the benefit of the square root implementation will be addressed later in this section. When the same two filters are used, EKF and square root EKF, but the partial-update is applied, both filters now show consistent estimates for the same scenario. Results are displayed in Figure 3.2.

For this example the $\boldsymbol{\beta}$ vector associated with the partial-update was selected to be $\boldsymbol{\beta}^{\mathrm{T}} = \begin{bmatrix} 0.9 & 0.9 & 0.75 \end{bmatrix}^{\mathrm{T}}$ (or $\boldsymbol{\Gamma} = \mathrm{diag}(0.1, 0.1, 0.25)$), which means that the position and velocity estimates are updated using 90% of the original update, whereas the ballistic parameter is updated using only a 75%. These update weight values were selected with the intention of limiting the updates mainly for the ballistic parameter, since it is commonly less observable than the other states.

Generally, the values for $\boldsymbol{\beta}$ are selected based on the idea that static or slowly varying states (with minimal process noise) can receive limited updates, whereas more observable or higher process noise states can receive larger update percentages. Additional "tuning" can be utilized to try to achieve the desired filter performance if the necessary data is available. This being a simulation, the data is clearly available for tuning, and the ballistic parameter updates are just sufficiently limited to prevent filter inconsistencies. As shown, the still substantial update of 75% (along with minor limitations of the other two states) was sufficient to avoid the issues seen in the full update case

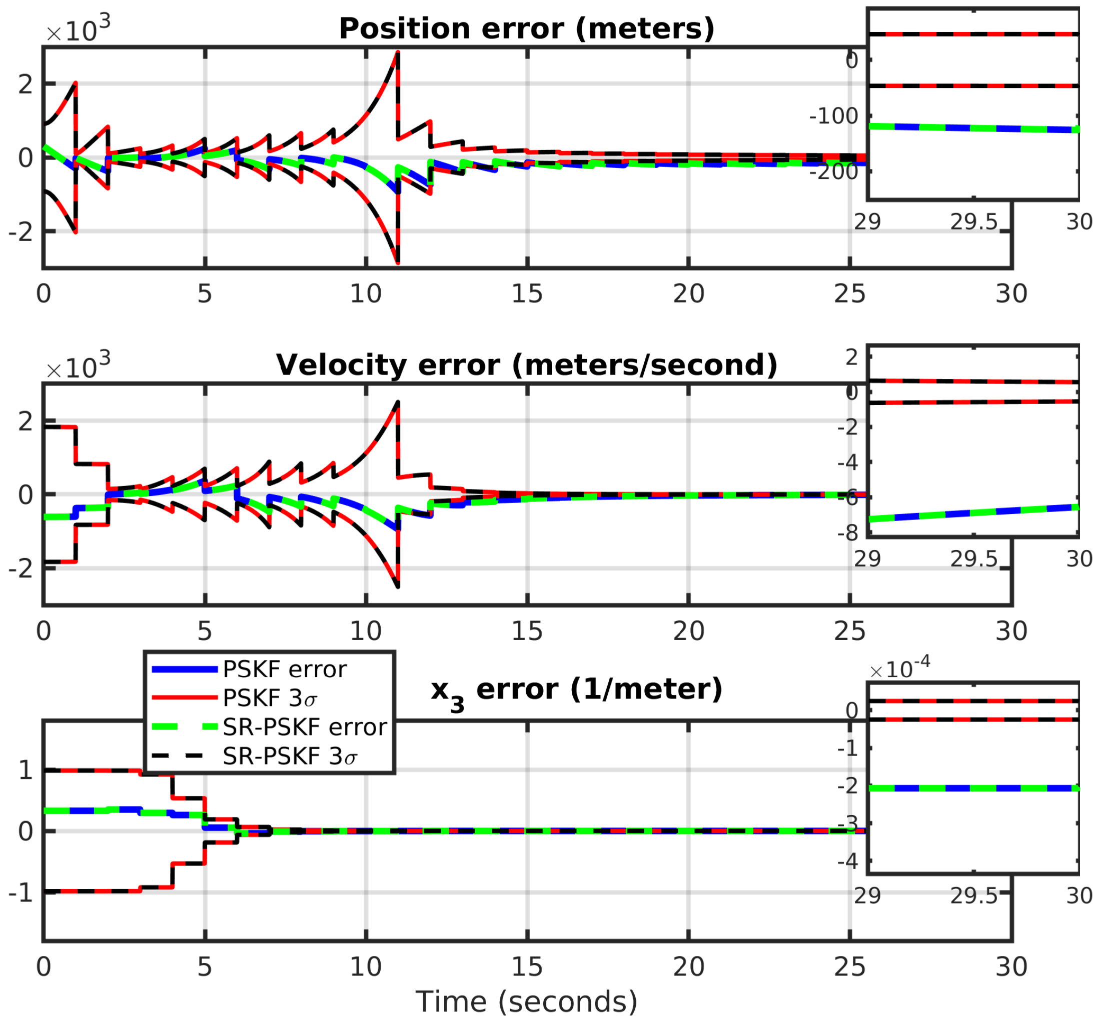


Figure 3.1: Standard EKF and square root partial-update EKF with full updates. The inset on the right shows a zoom-in for the last second of the simulation, displaying significant filter inconsistency with estimates well outside of 3σ bounds. Reprinted with permission from [1].

from Figure 3.1. One can also note the additional few updates steps taken before x_3 covariance values appear to collapse when comparing Figure 3.2 and Figure 3.1.

The explicit choice of $\boldsymbol{\beta}$ values is not particularly finicky, in fact, similar results are obtained with a weight of $\boldsymbol{\beta} = \begin{bmatrix} 1.0 & 1.0 & 0.1 \end{bmatrix}$ which are shown in Figure 3.3. In observing Figure 3.2 and Figure 3.3, the reader may note that the estimated covariance of the position and velocity states are significantly larger through the middle of the run. This is due to the more conservative nature

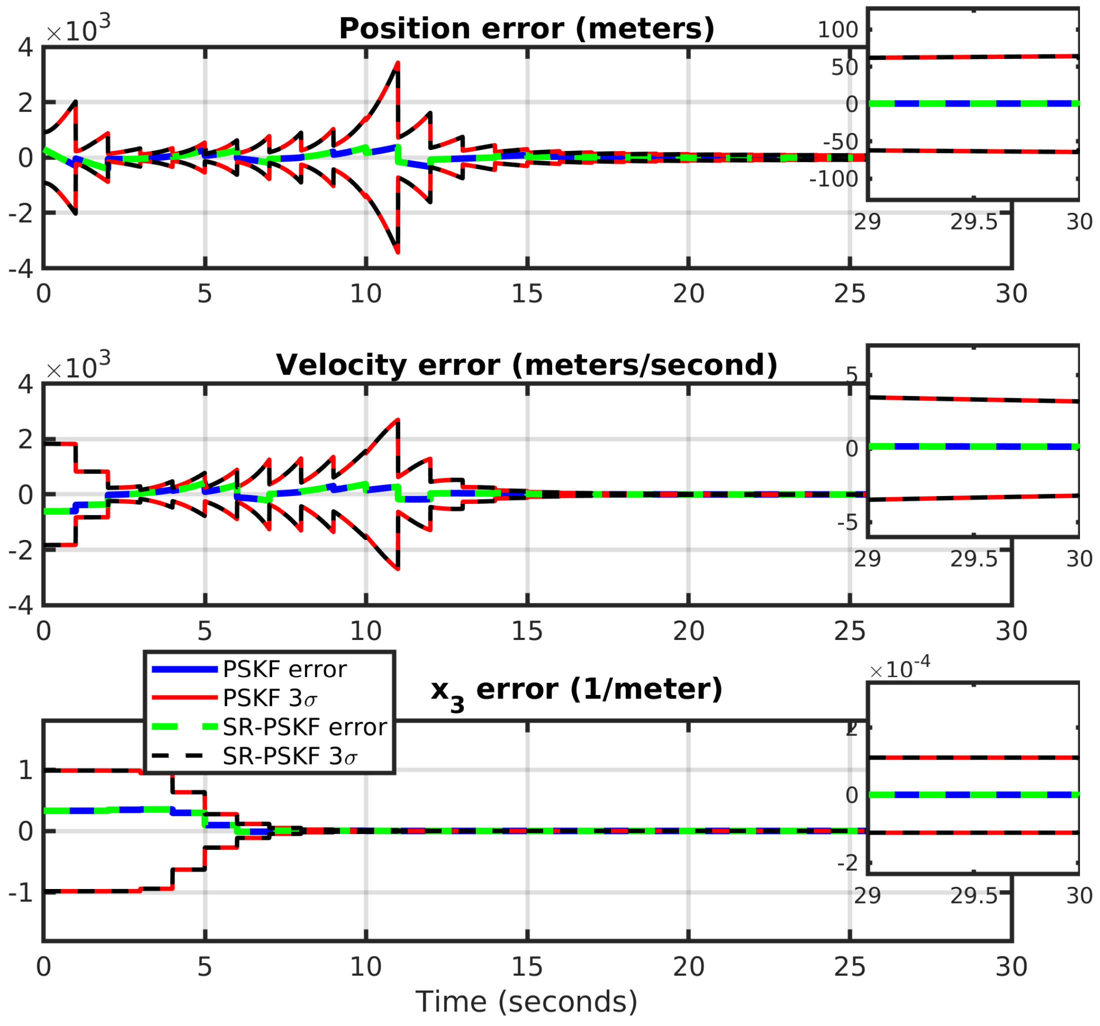


Figure 3.2: Partial-update EKF and square root Partial-Update EKF with $\boldsymbol{\beta} = \begin{bmatrix} 0.9 & 0.9 & 0.75 \end{bmatrix}$ resulting in 90%, 90%, and 75% updates respectively to the position, velocity, and ballistic coefficient states. The inset on the right shows a zoom-in for the last second of the simulation, displaying filter results with estimates within the 3σ bounds. Reprinted with permission from [1].

of the update weighting. By the end of the simulation, however, the filter is still able to converge even though the ballistic coefficient estimates only received 10% updates. This example certainly takes more time to converge, yet, like the first partial update example and unlike the full update example, it maintains appropriate estimates and covariance values throughout.

Also, it may be noted how the velocity uncertainty in both partial-update cases increase faster

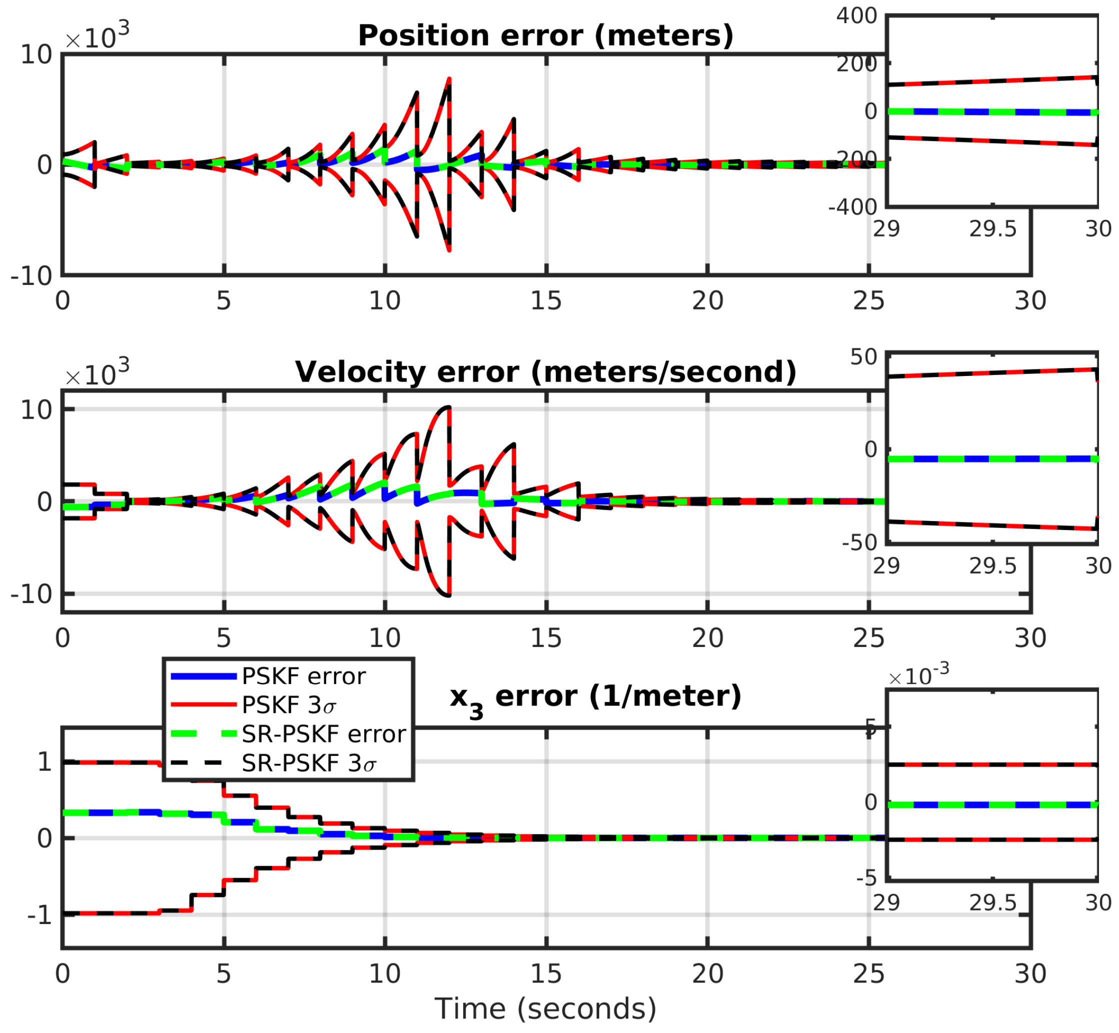


Figure 3.3: Partial-update EKF and square root Partial-Update EKF with $\boldsymbol{\beta} = \begin{bmatrix} 1.0 & 1.0 & 0.1 \end{bmatrix}$ resulting in 100%, 100%, and 10% updates respectively to the position, velocity, and ballistic coefficient states. The inset on the right shows a zoom-in for the last second of the simulation, displaying filter results with estimates within the 3σ bounds. Reprinted with permission from [1].

than those in Figure 3.1, this is especially notable just after the update at time $t = 11s$. This is because the partial updates did not push the ballistic parameter covariance down as tight, thus allowing uncertainty in velocity and position to grow, permitting the ballistic parameter estimate to settle on the correct value over a few more updates (without over or undershooting due to the linearization errors). Effectively the partial-update prevented the filter from “sabotaging" itself early on due to some relatively bad linearizations and avoided the associated repercussions. Finally, as before, the plots in Figure 3.2 and Figure 3.3 show the mathematical equivalence between square root and standard versions of the filter for this example.

3.3.2 Camera to Inertial Measurement Unit (IMU) calibration

This example is a multiplicative extended Kalman filter implementation used to calibrate a camera-Inertial Measurement Unit (IMU) system. This is, the filter is used to estimate the rigid body transformation between a camera optical frame and an IMU frame considering that both sensors are fixed to the same rigid platform. For the purposes of demonstrating the functionality of the square root filter, this section is limited to a minimum description of the multiplicative filter and it does not give further details on the process and measurement models. However, a closer examination of the system is presented in Chapter 6, Section 6.1.3. This specific problem is selected with the purpose of demonstrating the functionality of the square root partial-update filter when the state involves a larger state vector that includes more nuisance parameters (in this case the calibration parameters), and vector measurements are to be assimilated.

The filter operation consist of propagating the process model when an IMU measurement is available, and updating when camera images provide landmark features positions of a pre-known map. Although presented later in this dissertation, the process and measurement models are included in this section to facilitate the present discussion. The continuous process model equations are comprised of rotational and translational kinematics as follows:

$$ {}^{I}_{I_k}\dot{\bar{q}}(t) = \frac{1}{2}\begin{bmatrix} -\lfloor \boldsymbol{\omega}(t) \times \rfloor & \boldsymbol{\omega}(t) \\ -\boldsymbol{\omega}(t)^{\mathrm{T}} & 0 \end{bmatrix} {}^{I}_{I_k}\bar{q}(t) \; , \tag{3.34} $$

$$ {}^{W}\dot{\mathbf{p}}_I = {}^{W}\mathbf{v}_I \; , \tag{3.35} $$

$$ {}^{W}\dot{\mathbf{v}}_I(t) = {}^{W}\mathbf{a}_I(t) = ({}^{I}_{I_k}\mathbf{C} \; {}^{I_k}_{W}\mathbf{C})^{\mathrm{T}}\mathbf{s}(t) + {}^{W}\mathbf{g} \; , \tag{3.36} $$

$$ \dot{\mathbf{b}}_g(t) = \mathbf{n}_{wg}(t) \; , \tag{3.37} $$

$$\dot{\mathbf{b}}_a(t) = \mathbf{n}_{wa}(t) \;, \tag{3.38}$$

$$^{I}\dot{\mathbf{p}}_C = 0 \;, \tag{3.39}$$

$$^{C}_{I}\dot{\mathbf{q}} = 0 \;. \tag{3.40}$$

Such that the state vector is formed as,

$$\mathbf{x} = \begin{bmatrix} ^{I}_{W}\bar{q}^{\mathrm{T}} & ^{W}\mathbf{p}_I^{\mathrm{T}} & ^{W}\mathbf{v}_I^{\mathrm{T}} & \mathbf{b}_g^{\mathrm{T}} & \mathbf{b}_a^{\mathrm{T}} & ^{I}\mathbf{p}_C^{\mathrm{T}} & ^{C}_{I}\bar{q}^{T} \end{bmatrix}^{\mathrm{T}} \;. \tag{3.41}$$

The notation for the filter state variables is as follows: $^{I}_{W}\bar{q} \in \mathbb{R}^4$, $^{W}\mathbf{p}_I \in \mathbb{R}^3$ and $^{W}\mathbf{v}_I \in \mathbb{R}^3$ are the attitude quaternion, position and velocity of the IMU with respect to the world frame,. The IMU gyroscope and accelerometer biases are denoted as $\mathbf{b}_g \in \mathbb{R}^3$ and $\mathbf{b}_a \in \mathbb{R}^3$, respectively, and the IMU-camera calibration is denoted as $^{I}\mathbf{p}_C$. The sought calibration is considered to be the position of the camera with respect to the IMU frame, $^{I}\mathbf{p}_C \in \mathbb{R}^3$, and the attitude of the camera frame with respect to the IMU frame, $^{C}_{I}\bar{q}^{T} \in \mathbb{R}^4$. The measurement model for the IMU-camera system maps the position vectors of detected features into their corresponding pixels via a pinhole camera model. The position vector of the features is constructed according to

$$\begin{bmatrix} h_x & h_y & h_z \end{bmatrix}^{\mathrm{T}} = {}^{C}\mathbf{p}_{F_i} = {}^{C}_{I}\mathbf{C}\left({}^{I}_{W}\mathbf{C}({}^{W}\mathbf{p}_{F_i} - {}^{W}\mathbf{p}_I) - {}^{I}\mathbf{p}_C\right) \;, \tag{3.42}$$

where h_x, h_y and h_z represent the components of the $^{C}\mathbf{p}_{F_i}$ vector (the position vector of the i^{th} feature coordinatized in the camera frame), and $^{C}_{I}\mathbf{C}$ is the passive rotation matrix from the IMU to the camera frame. The pinhole camera model is given as,

$$\tilde{\mathbf{y}}_{F_i} = \begin{bmatrix} \tilde{u}_i \\ \tilde{v}_i \end{bmatrix} = \begin{bmatrix} f_x(h_x/h_z) + c_x \\ f_y(h_y/h_z) + c_y \end{bmatrix}_i + \mathbf{v}_{F_i} \;. \tag{3.43}$$

The term $\mathbf{v}_{F_i}$ in the pinhole camera model of Equation (3.43), represents a white noise zero-mean Gaussian process with covariance matrix $\mathbf{R}_{F_i} = E[\mathbf{v}\mathbf{v}^{\mathbf{T}}]$ that corrupts the pixel measurements.

The simulations shown in this section used the same parameters as those used in Table 6.1. The same data is included in Table 3.2 here with the purpose of making this section more self-contained.

Table 3.2: IMU-camera calibration parameters

State/parameter	Value
IMU-camera attitude uncertainty	2 deg
Lever arm uncertainty	5 cm
IMU attitude uncertainty	2 deg
IMU position uncertainty	5 cm
Camera frame rate	20 Hz
IMU rate	100 Hz
Camera pixel uncertainty	2 px

Figure 3.4 shows the results for the standard EKF, along with the square root partial-update filter. The partial update filter uses the following update percentages or weights:

$$\boldsymbol{\beta}_{{}^I_W\mathbf{q}} = \text{diag}\begin{bmatrix}0.95 & 0.95 & 0.95\end{bmatrix}, \tag{3.44}$$

$$\boldsymbol{\beta}_{{}^W\mathbf{p}_I} = \text{diag}\begin{bmatrix}0.95 & 0.95 & 0.95\end{bmatrix}, \tag{3.45}$$

$$\boldsymbol{\beta}_{{}^W\mathbf{v}_I} = \text{diag}\begin{bmatrix}1 & 1 & 1\end{bmatrix}, \tag{3.46}$$

$$\boldsymbol{\beta}_{\mathbf{b}_g} = \text{diag}\begin{bmatrix}1 & 1 & 1\end{bmatrix}, \tag{3.47}$$

$$\boldsymbol{\beta}_{\mathbf{b}_a} = \text{diag}\begin{bmatrix}1 & 1 & 1\end{bmatrix}, \tag{3.48}$$

$$\boldsymbol{\beta}_{{}^I\mathbf{p}_C} = \text{diag}\begin{bmatrix}0.25 & 0.25 & 0.25\end{bmatrix}, \tag{3.49}$$

$$\boldsymbol{\beta}_{{}^{C}_{I}\mathbf{q}} = \text{diag}\begin{bmatrix} 0.25 & 0.25 & 0.25 \end{bmatrix} . \tag{3.50}$$

Such that,

$$\boldsymbol{\beta} = \text{diag}(\boldsymbol{\beta}_{{}^{I}_{W}\mathbf{q}}, \boldsymbol{\beta}_{{}^{W}\mathbf{p}_I}, \boldsymbol{\beta}_{{}^{W}\mathbf{v}_I}, \boldsymbol{\beta}_{\mathbf{b}_g}, \boldsymbol{\beta}_{\mathbf{b}_a}, \boldsymbol{\beta}_{{}^{I}\mathbf{p}_C}, \boldsymbol{\beta}_{{}^{C}_{I}\mathbf{q}}) . \tag{3.51}$$

Notice that the quaternion states have only three weights due to the indirect filter's nature of the multiplicative formulation. Due to the indirect form, the notation for the state elements should also change, but it is maintained to facilitate discussion; formally, it should refer to error variables as discussed in Chapter 2. The β weights sub-index indicate correspondence with each different state. The block-diagonal matrix $\boldsymbol{\beta}$ contains all of the partial-update weights for the system. This is the matrix that will act on the Kalman gain to perform a partial-update as per Equation 2.51. For the first simulation, both filters (square root and conventional) are using full measurement updates, and they begin with estimates within the 3 σ bounds, but eventually uncertainties and nonlinear effects are not well handled by the filter producing inconsistent estimates. Figures 3.5 and 3.6 also show inconsistencies when full update is used.

On the other hand, as seen in Figures 3.7, 3.8 and 3.9 the partial-update filter performs better for both factorized and conventional formulation. Although the filter takes more time to converge, the produced estimates are definitely more consistent than in the full update case.

Monte Carlo simulations were also run for the partial-update filter to show the consistent behavior given this scenario, and are shown in Figure 3.10 and Figure 3.11. When comparing the sampled standard deviations against those produced by the partial-update filter for the lever arm in Figure 3.10, it can be seen that the partial-update filter is slightly overconfident. However, as the uncertainties and nonlinearities impact decays, the partial-update filter is able to improve its consistency. Similarly, for the rotation calibration error, the filter starts slightly overconfident but it gains consistency quickly. Overall, the Monte Carlo runs show consistency improvement over the full update filter for all of the states. Conversely, a full update filter was seen inconsistent, (Monte Carlo runs are not included) always resembling that behavior observed from a single run

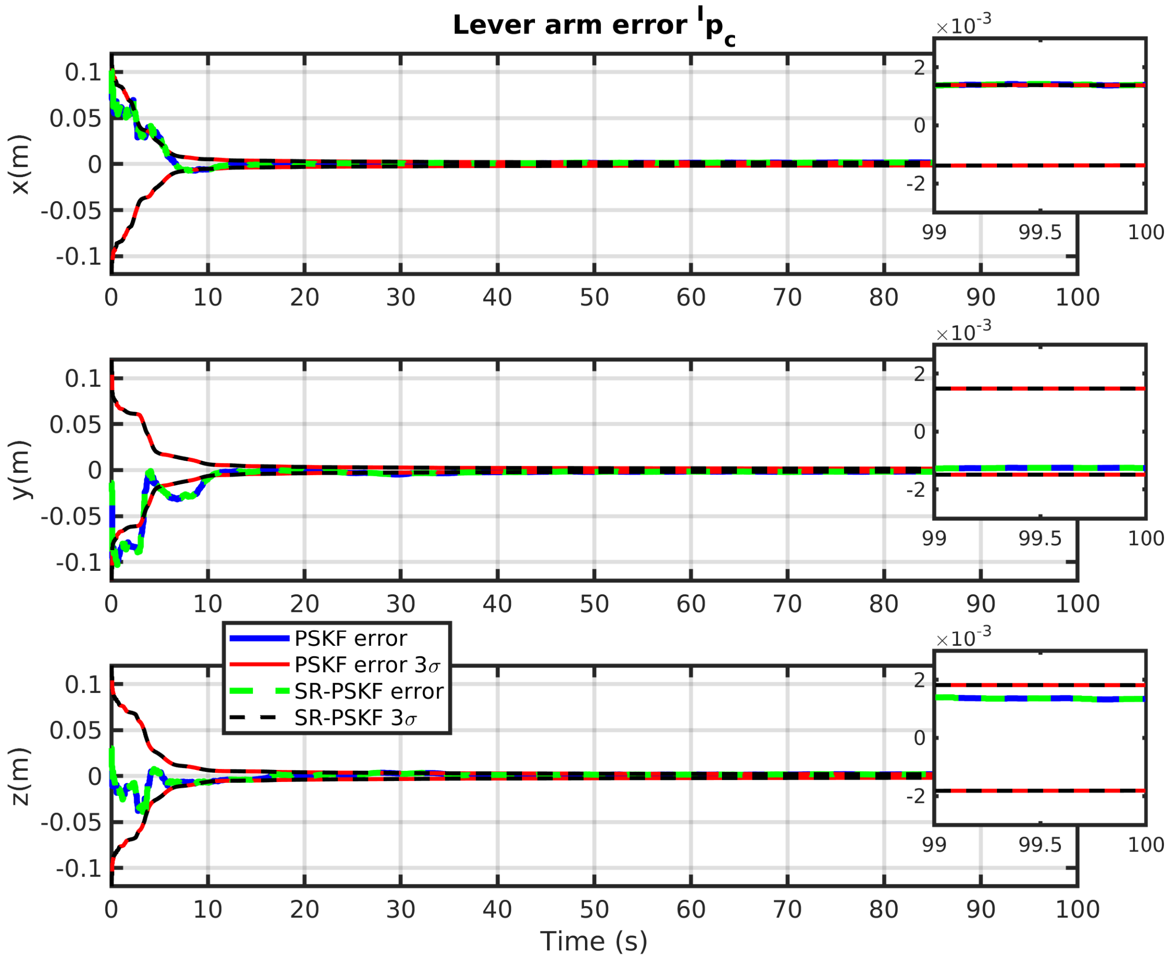


Figure 3.4: Camera-IMU calibration lever arm estimates when a full update is performed.

experiment shown before.

In general, the partial-update filter was able to improve the behavior of the underlying EKF (MEKF to be more specific) via a well-selected percentage update. Similarly to the previous example, the selection of the weights was not finicky for this problem, and a variety of well behaved filter were also obtained with different β values. However, the selected weight values were shown to offer better convergence rate and consistency over other weights. Again, the square root formulation certainly provides extra numerical robustness to the filter, not uncertainty or errors robustness.

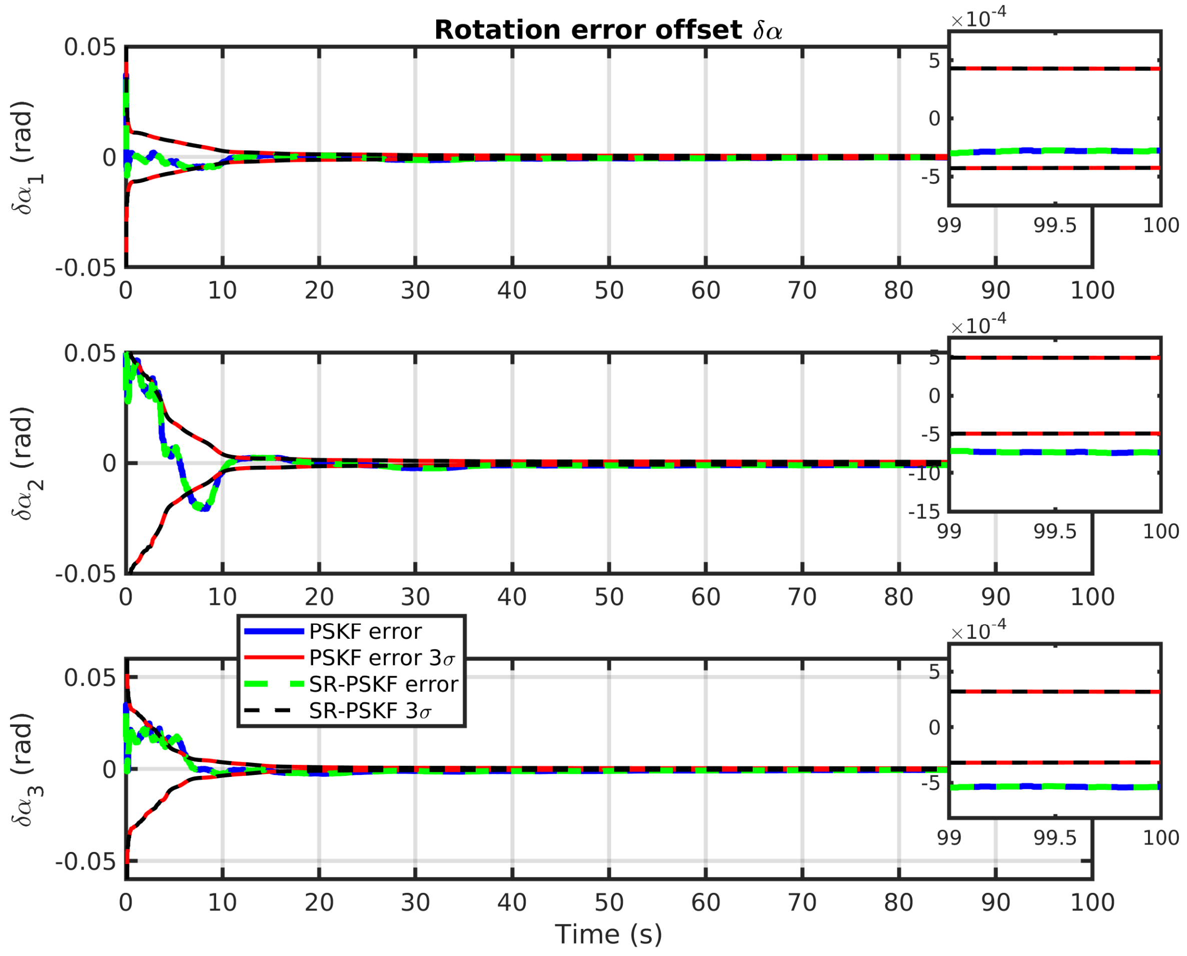


Figure 3.5: Camera-IMU calibration rotation error estimates when a full update is performed.

3.4 Monte Carlo runs

Monte Carlo simulations were ran for both, square root EKF and square root partial-update EKF. A total of 100 runs were executed and the histories of all of the states were recorded. Moreover, the sampled standard deviation and the standard deviation as computed by the filter, were also calculated and are used to check for filter consistency. For both filters the initial conditions the same as for the single run scenario. For convenience, the parameter values used for the simulations are condensed in Table 3.3.

Figure 3.12 shows the 100 EKF runs histories. The EKF shows that in its majority is able to

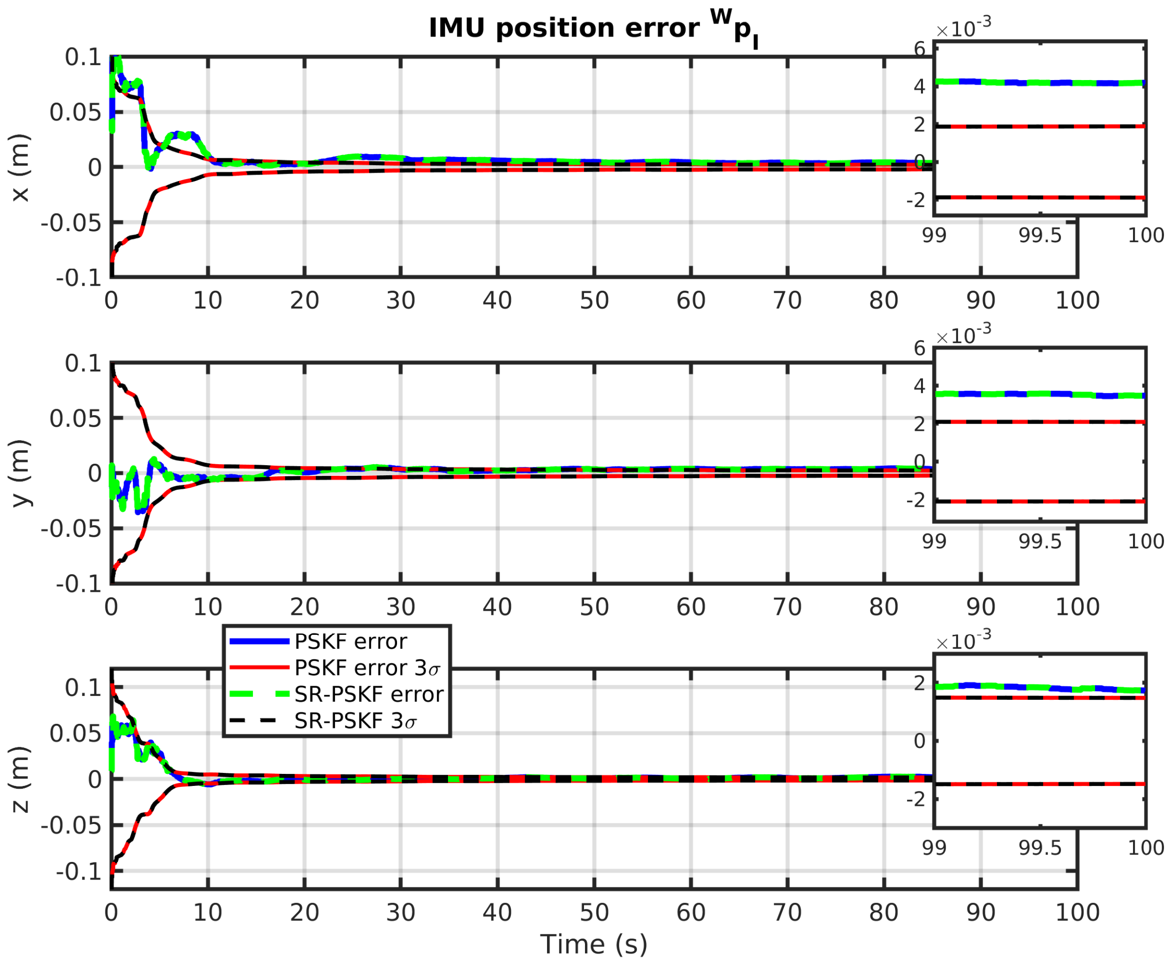


Figure 3.6: IMU global position when a full update is performed.

prevent total failure of the filter, except for two cases (blue curves) that are divergent. However, most of the runs show similar behavior as the one showed for the single run: estimates are not within the proper sigma bounds after around $t = 15$ seconds and the errors do not converge to zero. The square root partial-update technique on the other hand (as depicted in Figure 3.13), shows a dramatic improvement over the EKF. First, the filter presents no runs with divergence. Second, the error histories show a significant magnitude reduction, and third, a superior capacity to handle the initial uncertainties by avoiding the overreaction in the update is achieved, showing that the behavior seen in the single run is in fact, the overall filter behavior.

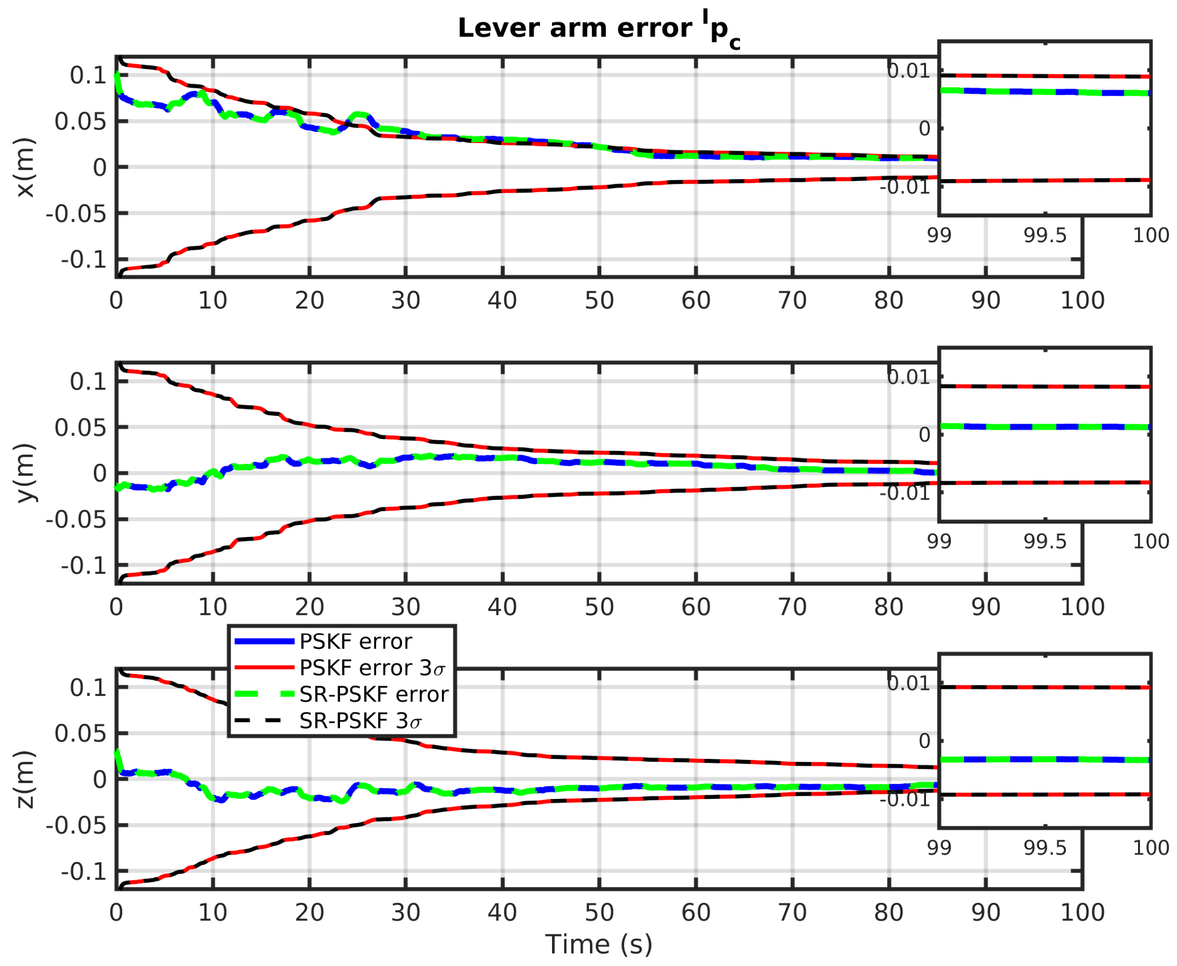


Figure 3.7: Camera-IMU calibration lever arm estimates when a partial-update is performed.

The averaged standard deviation from the Monte Carlo runs and the standard deviation estimated by the filter are depicted in Figure 3.14 and are shown to practically coincide. Since the mean estimation error is around zero, altogether this indicates that the filter is consistent. Since the EKF filter presented divergent cases, only error histories are graphed for it.

Overall, it was observed that if low initial errors were ensured, the EKF can be functional and quickly converge, but is not robust enough to handle errors at the level of the square root partial-update filter. Conversely, the square root partial-update filter was observed to be consistent, and able to handle higher nonlinearities and uncertainties better than the conventional square root EKF

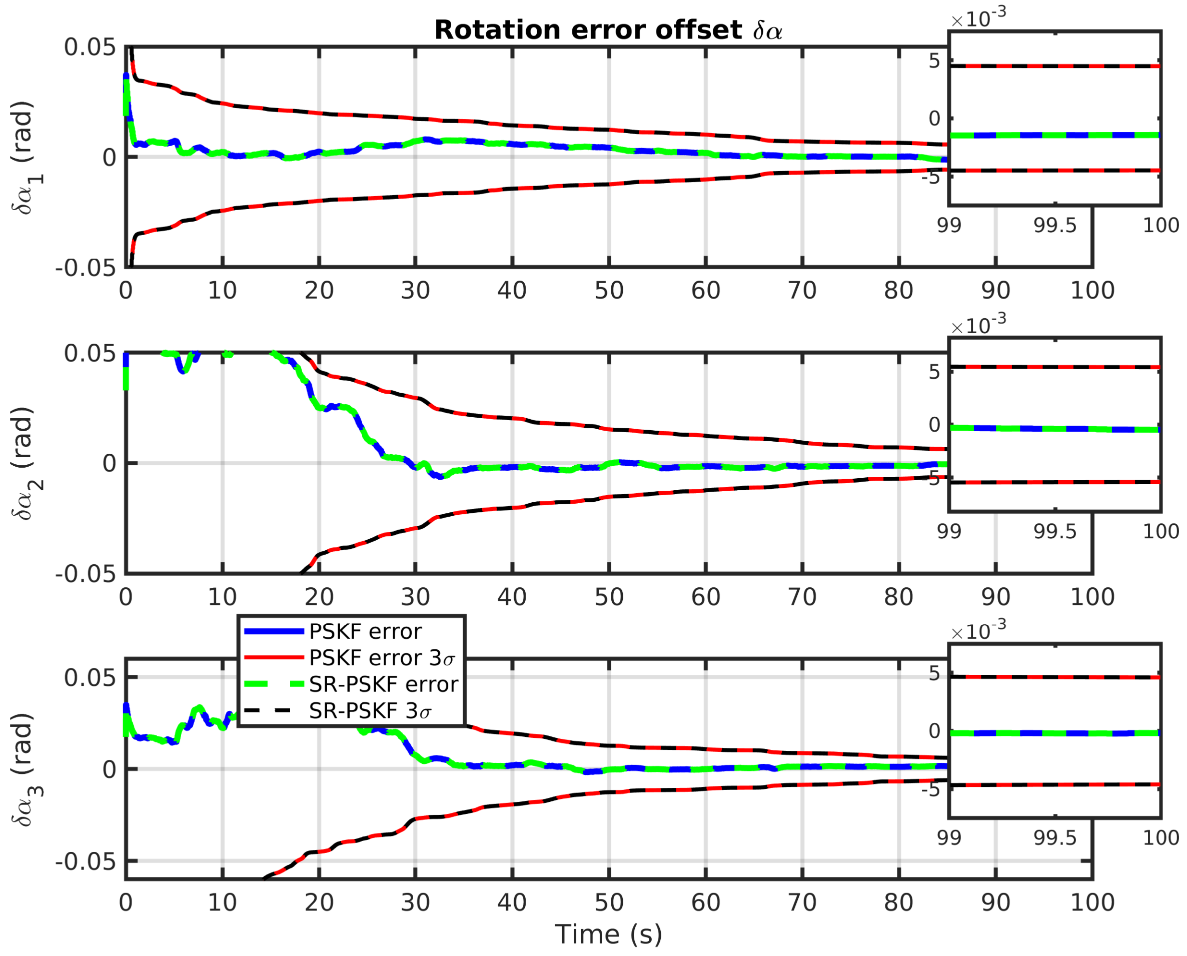


Figure 3.8: Camera-IMU calibration rotation estimates when a partial-update is performed.

Table 3.3: Re-entering body parameters

State/parameter uncertainty	Uncertainty 1σ value
Position	$300\,\mathrm{m}$
Velocity	$600\,\mathrm{m/s}$
Ballistic parameter	$0.33\,\mathrm{m}^{-1}$
Measurement	$300\,\mathrm{m}$

or EKF.

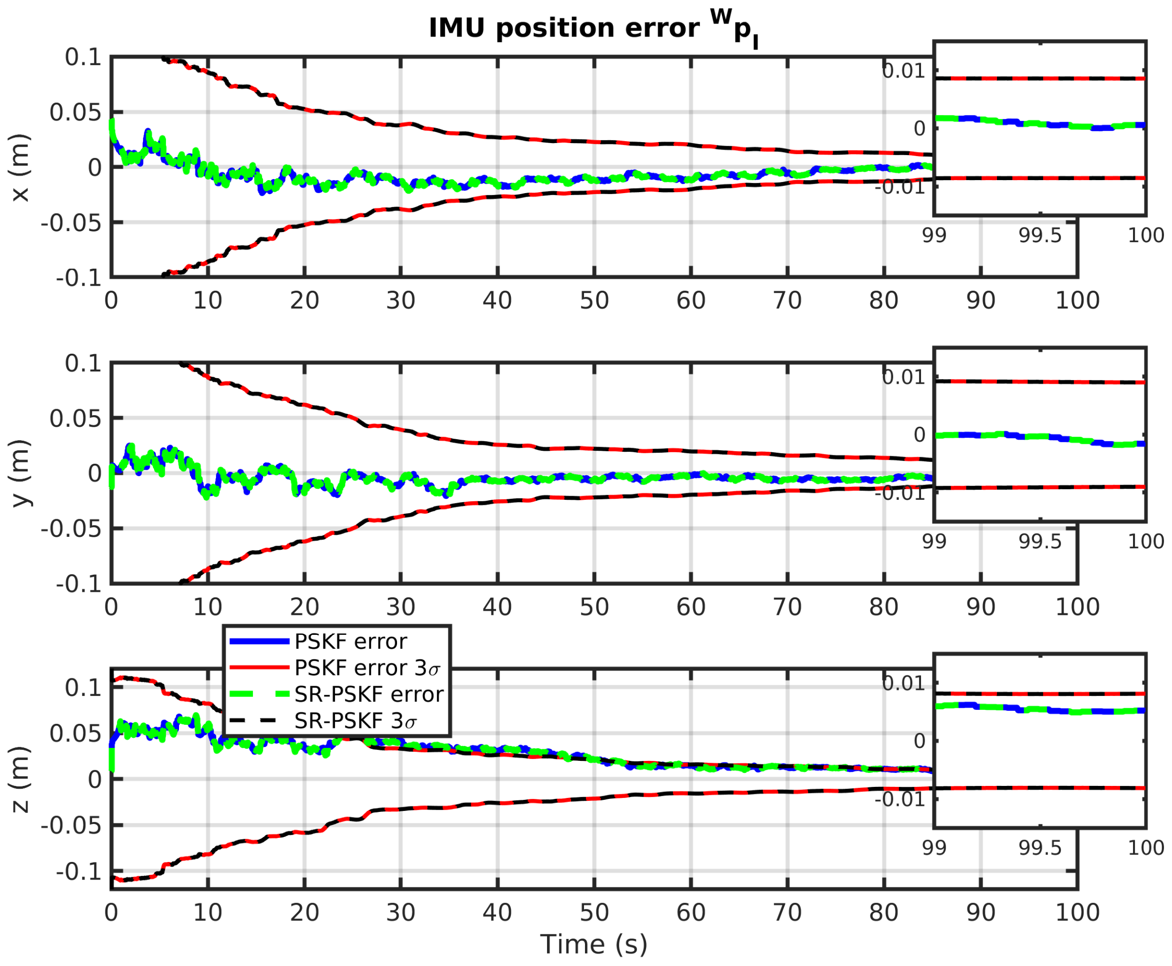


Figure 3.9: IMU global position when a full update is performed.

3.4.1 Condition number

Lastly, the numerical stability afforded by the square root implementation [43] is briefly discussed. The chosen example never threatened the numerical integrity of the filter as it was primarily intended to show the benefit of the partial update, and the benefits of the square root form are well known. Nevertheless, a brief analysis of the condition number is shown here simply to assure the reader that the numerical improvements afforded by the square root form were maintained in this development.

One way to interpret the condition number of a matrix, is to consider it as a measure of how

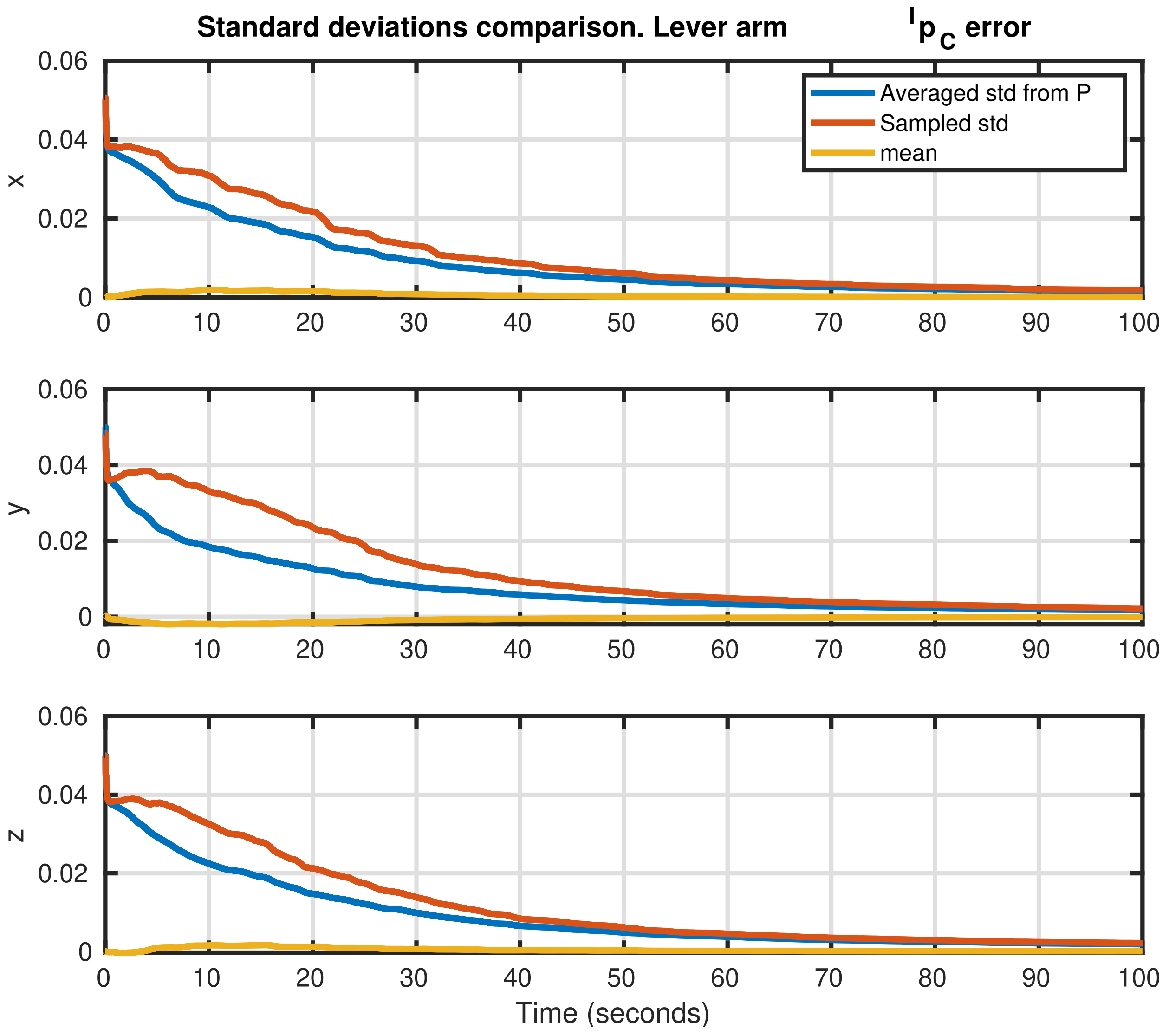


Figure 3.10: Averaged and sampled standard deviation from 500 Monte Carlo runs for the lever arm components. The mean error across the runs is also plotted. Units are in meters.

close to be singular the matrix is, with the convention that the matrix is singular when the condition number is infinite. One way to compute the condition number κ of a matrix $\mathbf{P}$ is through the matrix singular values as:

$$\kappa(\mathbf{P}) = \sigma_{max}(\mathbf{P})/\sigma_{min}(\mathbf{P}) \tag{3.52}$$

While the assessment of the condition number is based on heuristics, a condition number is regarded to be a very large (and generally bad) condition number when $\log_{10}(\kappa) >$ (available ma-

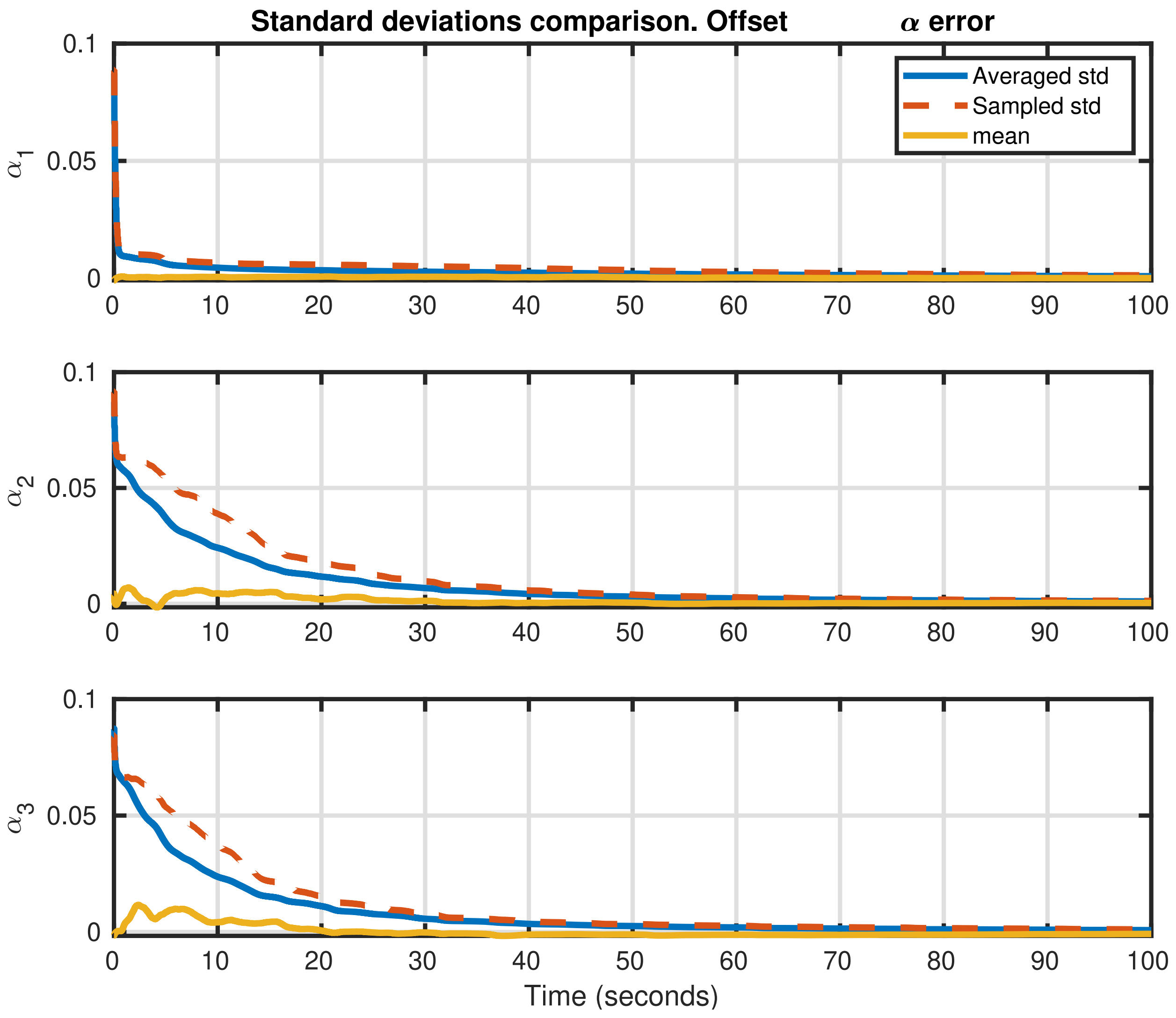


Figure 3.11: Averaged and sampled standard deviation from 500 Monte Carlo runs for rotation error (IMU to camera rotation). The mean error across the runs is also plotted. Units are in meters.

chine precision) [74] and in general it is desirable to keep $\mathbf{P}$ condition number low.

The condition numbers for both partial-update examples (standard and square root) are shown in Figure 3.15. It is clear that even though both partial-update filters (standard and square root version) produce accurate estimators, the square root partial-update does provide a significantly improved (lower) condition number for the uncertainty matrix, as expected. This indicates that the square root partial-update filter developed in this chapter, maintains the square root filter's numerical precision advantage over the traditional filter approach as desired.

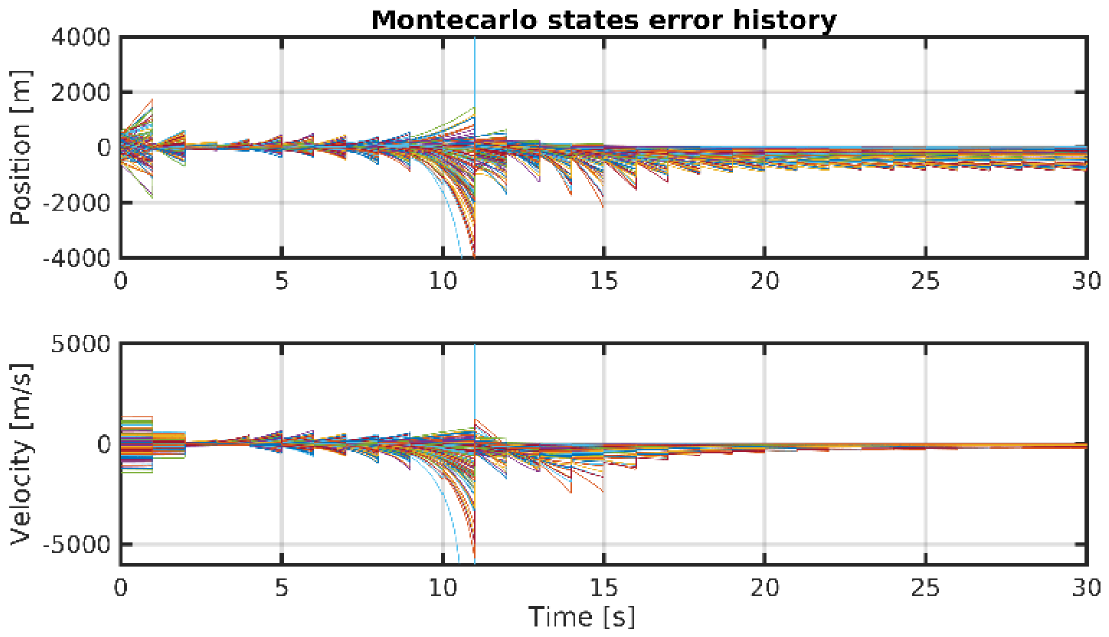


Figure 3.12: Monte Carlo standard EKF and square root partial-update EKF with full updates.

3.5 Processing a vector-valued measurement

The square root filter can also process vector measurements. Formulations such as those presented in [68] and [75] are some alternatives. Similarly, the square-root partial-update can process vector measurements, and its derivation is presented in this section. This formulation of the partial-update filter may not be adequate for embedded system implementation due to its high computational complexity, but it can be a useful filter debugging or validation tool. In any case, it is the user's decision to select the filter form based on the design requirements, and if a sequential square root filter is to be used, the engineer also needs to consider the possible extra cost of diagonalizing the measurement noise covariance matrix.

3.5.1 Square root partial-update for vector-valued measurements

The filter version that can process a vector measurement is a slight modification of the sequential version presented in this chapter. However, the complexity of the algorithm increases dramatically with respect to scalar-valued measurement processing. The extension of the square

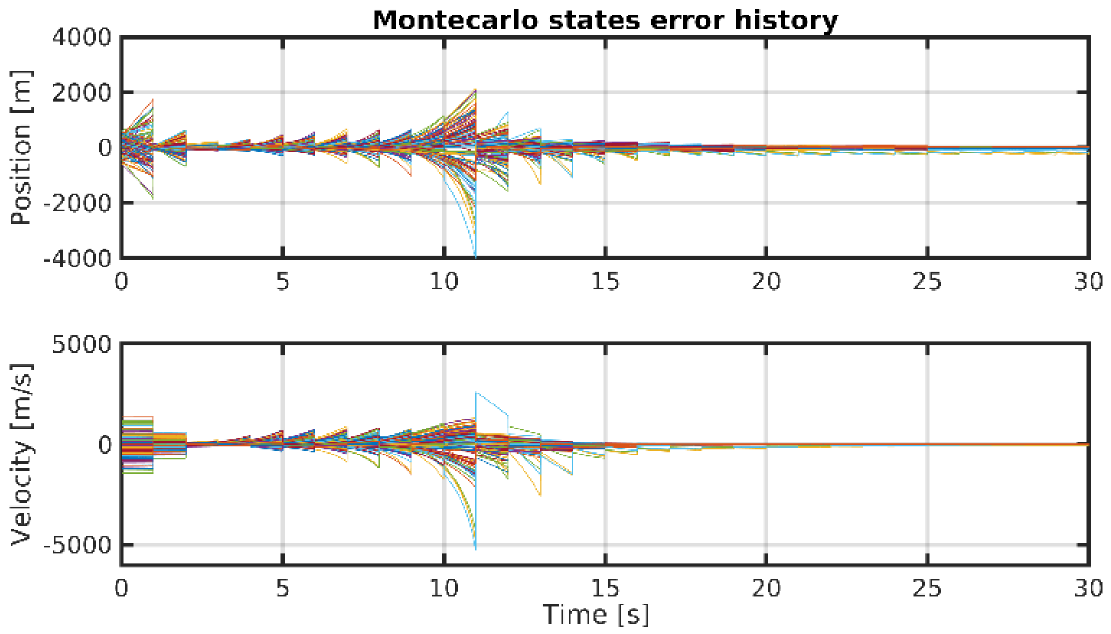


Figure 3.13: Monte Carlo runs for the partial-Update EKF and square root partial-update EKF with $\boldsymbol{\beta} = \begin{bmatrix} 0.9 & 0.9 & 0.75 \end{bmatrix}$ resulting in 90%, 90%, and 75% updates respectively to the position, velocity, and ballistic coefficient states.

root partial-update filter presented here is mostly based on the factorization presented in [68]. This factorization allows one to perform the conventional update via Gram-modified-Schmidt orthogonalization by constructing an augmented matrix. Regarding the partial-update operation, it is still possible if an extra Cholesky decomposition is performed.

Paralleling the development presented in this chapter for the sequential filter, recall

$$\mathbf{P}^{++} = \boldsymbol{\Gamma}(\mathbf{P}^{-} - \mathbf{P}^{+})\boldsymbol{\Gamma} + \mathbf{P}^{+} \, . \tag{3.53}$$

Now, from the standard EKF equations (ignoring the time indices for ease of notation), $\mathbf{P}^{+} = (\mathbf{I} - \mathbf{K}\,\mathbf{H})\mathbf{P}^{-}$ is used into Equation (3.53).

$$\mathbf{P}^{++} = \mathbf{P}^{+} + \boldsymbol{\Gamma}(\mathbf{K}\mathbf{H}\mathbf{P}^{-})\boldsymbol{\Gamma} \, , \tag{3.54}$$

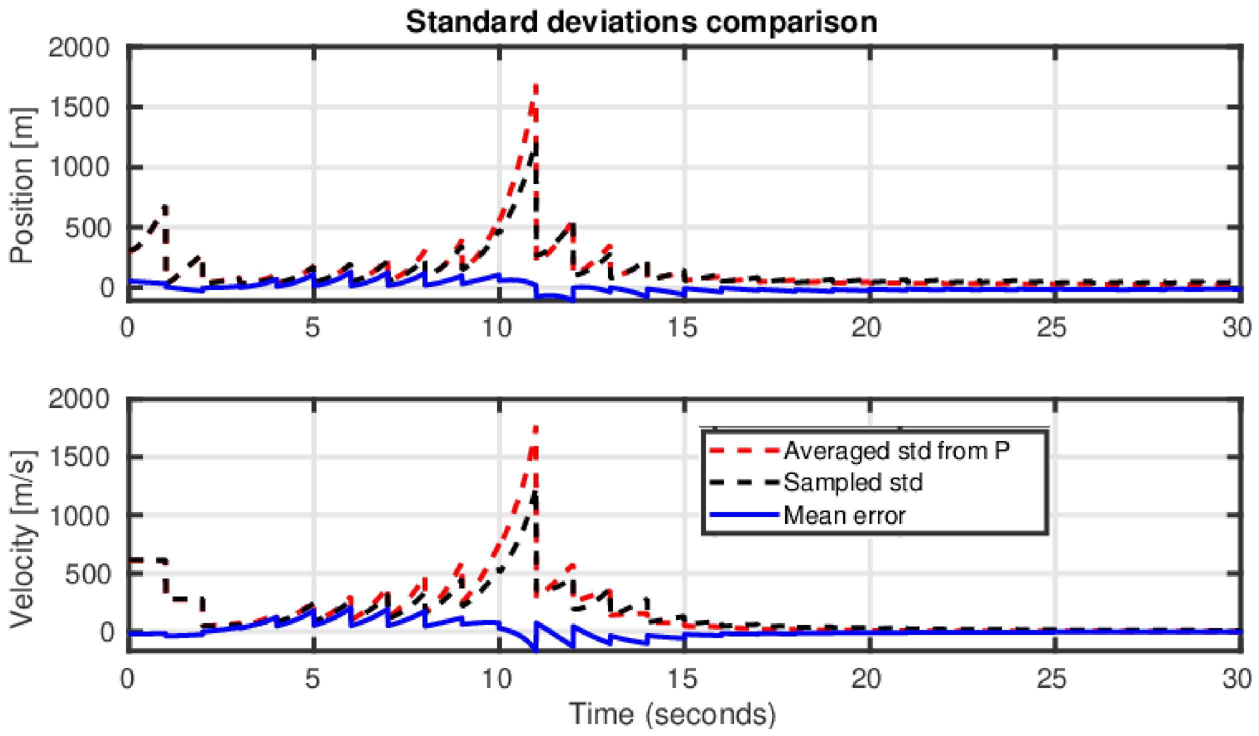


Figure 3.14: Monte Carlo runs for the partial-Update EKF and square root partial-update EKF with $\boldsymbol{\beta} = \begin{bmatrix} 0.9 & 0.9 & 0.75 \end{bmatrix}$ resulting in 90%, 90%, and 75% updates respectively to the position, velocity, and ballistic coefficient states.

then replacing $\mathbf{K} = \mathbf{P}^{-}\mathbf{H}^{\mathrm{T}}(\mathbf{H}\mathbf{P}^{-}\mathbf{H}^{\mathrm{T}} + \mathbf{R})^{-1}$ gives

$$\mathbf{P}^{++} = \mathbf{P}^{+} + \boldsymbol{\Gamma}(\mathbf{P}^{-}\mathbf{H}^{\mathrm{T}}(\mathbf{H}\mathbf{P}^{-}\mathbf{H}^{\mathrm{T}} + \mathbf{R})^{-1}\mathbf{H}\mathbf{P}^{-})\boldsymbol{\Gamma}\ . \tag{3.55}$$

Then, requiring that $\mathbf{P}^{-} = (\mathbf{S}^{-})(\mathbf{S}^{-})^{\mathrm{T}}$ and $\mathbf{P}^{+} = (\mathbf{S}^{+})(\mathbf{S}^{+})^{\mathrm{T}}$. Additionally it is establish that $\tilde{\mathbf{R}} = (\mathbf{H}\mathbf{P}^{-}\mathbf{H}^{\mathrm{T}} + \mathbf{R})^{-1}$. These actions lead to

$$\mathbf{P}^{++} = (\mathbf{S}^{+})(\mathbf{S}^{+})^{\mathrm{T}} + \boldsymbol{\Gamma}(\mathbf{S}^{-})(\mathbf{S}^{-})^{\mathrm{T}}\mathbf{H}_i^{\mathrm{T}}\tilde{\mathbf{R}}\mathbf{H}_i(\mathbf{S}^{-})(\mathbf{S}^{-})\boldsymbol{\Gamma}\ . \tag{3.56}$$

To achieve the required factorization to partial-update, Cholesky decomposition is used on $\tilde{\mathbf{R}}$ as to obtain,

$$\tilde{\mathbf{R}} = \tilde{\mathbf{R}}^{1/2}\tilde{\mathbf{R}}^{T/2}\ . \tag{3.57}$$

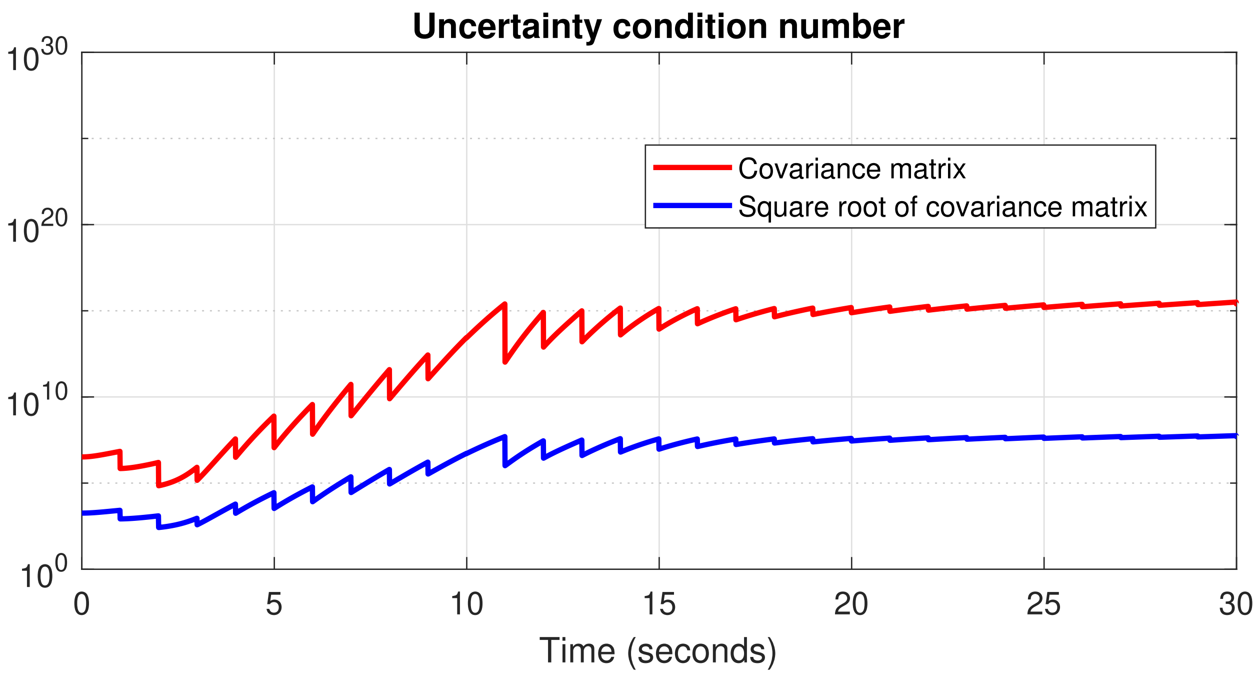


Figure 3.15: Uncertainty condition number for the partial-update EKF and square root partial-update EKF filter example for $\boldsymbol{\beta} = \begin{bmatrix} 0.9 & 0.9 & 0.75 \end{bmatrix}$. Reprinted with permission from [1].

Using Equation (3.57), the expression covariance becomes

$$\mathbf{P}^{++} = (\mathbf{S}^{+})(\mathbf{S}^{+})^{\mathrm{T}} + \boldsymbol{\Gamma}(\mathbf{S}^{-})(\mathbf{S}^{-})^{\mathrm{T}}\mathbf{H}_i^{\mathrm{T}}\tilde{\mathbf{R}}^{1/2}\tilde{\mathbf{R}}^{T/2}\mathbf{H}_i(\mathbf{S}^{-})(\mathbf{S}^{-})\boldsymbol{\Gamma}\,. \tag{3.58}$$

Similarly to the scalar measurement case the candidate square-root (and in fact, a valid square root) is written as,

$$\begin{bmatrix} (\mathbf{S}_k^{++})^{\mathrm{T}} \\ \mathbf{0} \end{bmatrix} = \mathbf{T}\begin{bmatrix} (\mathbf{S}_k^{+})^{\mathrm{T}} \\ \tilde{\mathbf{R}}^{T/2}\mathbf{H}^{\mathrm{T}}(\mathbf{S}_k^{-})^{\mathrm{T}}\boldsymbol{\Gamma} \end{bmatrix}\,. \tag{3.59}$$

3.5.1.1 Conventional time update with vector-valued measurement processing

As the partial-update concept does not affect the propagation step, the method utilized for propagation is identical to the presented for the sequential filter. However, the measurement update (conventional non-partial-update) is modified. In [68] the factorization that allows the conventional update can be used. Paralleling the procedure for the square root filter for scalar measurements, such factorization is set such that an orthogonal matrix needs to be found, but more importantly, that the updated square root of the covariance is computed. Similarly, this process is accomplished via the Modified-Gram-Schmidt. The required factorization for the conventional update is,

$$\begin{bmatrix} \tilde{\mathbf{R}}^{T/2} & \mathbf{K}(\mathbf{R}+\mathbf{H}(\mathbf{S}^-)(\mathbf{S}^-)^{\mathrm{T}}\mathbf{H}^{\mathrm{T}})^{T/2} \\ \mathbf{0} & (\mathbf{S}^+)^{\mathrm{T}} \end{bmatrix} = \tilde{T} \begin{bmatrix} \tilde{\mathbf{R}}^{T/2} & \mathbf{0} \\ (\mathbf{S}^-)^{\mathrm{T}}\mathbf{H}^{\mathrm{T}} & (\mathbf{S}^-)^{\mathrm{T}} \end{bmatrix} . \tag{3.60}$$

Once the MGS is executed with the intent of finding the orthogonal matrix $(n+r)\times(n+r)$, the bottom-right $(n\times n)$ block, $(\mathbf{S}^+)^{\mathrm{T}}$ will be generated. With this information in hand, the partial-update procedure (perform MGS on Equation (3.59)) to process a vector measurement can then be executed.

Table 3.4 summarizes the square root partial-update for vector measurement processing.

As previously mentioned, at a first glance the algorithm seems simple to implement as once the MGS and the Cholesky decomposition are available, there should not be any difficulties implementing this version of the square root filter. Nonetheless, the cost incurred makes it non apt for small computers. The extra computations come mainly from the matrix inversion required to compute the gain (also required in the conventional Kalman filter formulation), a Cholesky decomposition to obtain $\tilde{\mathbf{R}}^{1/2}$ every time a measurement is available, an extra MGS (of a $(n+m)\times(n+m)$ matrix) to perform the conventional update, and finally, a second MGS for a $(n+m)\times n$.

3.5.2 Measurement and process noise covariance decorrelation

If the user desires to process vector measurements in a sequential fashion, measurement covariance matrix must be diagonalized, if needed. Decorrelation can be achieved by different methods. In this section the modified Cholesky decomposition is described, as the factors would be available already (if the square root filter is implemented). Documentation and more details on this diagonalization algorithm is vast and can be found in many linear algebra or filtering books [21], [68], [72]. The use of the decorrelation concept similarly applies to the measurement noise covariance matrix.

Table 3.4: Square root partial-update Schmidt-Kalman filter. Vector measurement processing.

Model	$\mathbf{x}_k = \mathbf{f}_{k-1}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) + \mathbf{w}_{k-1}$ $\tilde{\mathbf{y}}_k = \mathbf{h}_k(\mathbf{x}_k) + \mathbf{v}_k$ $\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k)$ $\mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k)$
Initialize	$\hat{\mathbf{x}}_0^+ = \mathbf{x}_0$ $\mathbf{S}_0^+ = chol(\mathbf{P}_0^+)$ [69] $\mathbf{Q}_0^{1/2} = eigendec(\mathbf{Q}_0)$ [72] $\mathbf{\Gamma} = \mathrm{diag}(1-\beta_1, 1-\beta_2, ..., 1-\beta_n)$
Propagation	$\hat{\mathbf{x}}_k^- = \mathbf{f}_{k-1}(\hat{\mathbf{x}}_{k-1}^+, \mathbf{u}_{k-1})$ $Perform\ MGS$ [16] for $\begin{bmatrix} (\mathbf{S}_k^-)^{\mathrm{T}} \\ \mathbf{0} \end{bmatrix} = \mathcal{T} \begin{bmatrix} (\mathbf{S}^+{}_{k-1})^{\mathrm{T}} \mathbf{F}_{k-1}^{\mathrm{T}} \\ \mathbf{Q}_{k-1}^{T/2} \end{bmatrix}$
Gain	$\mathbf{K}_k = (\mathbf{S}_k^-)(\mathbf{S}_k^-)^{\mathrm{T}} \mathbf{H}_k^{\mathrm{T}} \tilde{\mathbf{R}}$ $\tilde{\mathbf{R}}_k = (\mathbf{H}_k (\mathbf{S}_k^-)(\mathbf{S}_k^-) \mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k)^{-1}$ $\tilde{\mathbf{R}}_k^{1/2} = chol(\tilde{\mathbf{R}}_k)$
Update	$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\tilde{\mathbf{y}}_k - \hat{\mathbf{y}}_k)$ $\begin{bmatrix} \tilde{\mathbf{R}}_k^{T/2} & \mathbf{K}_k(\mathbf{R}_k + \mathbf{H}_k (\mathbf{S}_k^-)(\mathbf{S}_k^-)^{\mathrm{T}} \mathbf{H}_k^{\mathrm{T}})^{T/2} \\ \mathbf{0} & (\mathbf{S}_k^+)^{\mathrm{T}} \end{bmatrix} = \tilde{\mathcal{T}} \begin{bmatrix} \tilde{\mathbf{R}}_k^{T/2} & \mathbf{0} \\ (\mathbf{S}_k^-)^{\mathrm{T}} \mathbf{H}_k^{\mathrm{T}} & (\mathbf{S}_k^-)^{\mathrm{T}} \end{bmatrix}$
Partial-Update	$\hat{\mathbf{x}}_i^{++} = \mathbf{\Gamma} \hat{\mathbf{x}}_k^- + (\mathbf{I} - \mathbf{\Gamma}) \hat{\mathbf{x}}_k^+$ $Perform\ MGS$ [16] for $\begin{bmatrix} (\mathbf{S}_k^{++})^{\mathrm{T}} \\ \mathbf{0} \end{bmatrix} = \mathbf{T} \begin{bmatrix} (\mathbf{S}_k^+)^{\mathrm{T}} \\ \tilde{\mathbf{R}}_k^{T/2} \mathbf{H}_k^{\mathrm{T}} (\mathbf{S}_k^-)^{\mathrm{T}} \mathbf{\Gamma} \end{bmatrix}$

Let the noise measurement covariance be decomposed by Cholesky method into its *square root*

$$\mathbf{R} = \sqrt{\mathbf{R}}\sqrt{\mathbf{R}}^{\mathrm{T}} , \tag{3.61}$$

and consider the measurement equation $\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{v}$, being transformed by $\sqrt{\mathbf{R}}^{-1}$ as

$$\mathbf{z} = \sqrt{\mathbf{R}}^{-1}\mathbf{y} = \sqrt{\mathbf{R}}^{-1}\mathbf{H}\mathbf{x} + \sqrt{\mathbf{R}}^{-1}\mathbf{v} . \tag{3.62}$$

The residual is then

$$\mathbf{e} = \mathbf{z} - \hat{\mathbf{z}} = \sqrt{\mathbf{R}}^{-1}\mathbf{H}(\mathbf{x} - \hat{\mathbf{x}}) + \sqrt{\mathbf{R}}^{-1}\mathbf{v} \; , \tag{3.63}$$

with covariance

$$\mathrm{E}[\mathbf{e}\mathbf{e}^{\mathrm{T}}] = \mathrm{E}[(\sqrt{\mathbf{R}}^{-1}\mathbf{H}(\mathbf{x} - \hat{\mathbf{x}}) + \sqrt{\mathbf{R}}^{-1}\mathbf{v})(\sqrt{\mathbf{R}}^{-1}\mathbf{H}(\mathbf{x} - \hat{\mathbf{x}}) + \sqrt{\mathbf{R}}^{-1}\mathbf{v})^{\mathrm{T}}] \tag{3.64}$$

$$\mathrm{E}[\mathbf{e}\mathbf{e}^{\mathrm{T}}] = \sqrt{\mathbf{R}}^{-1}\mathbf{H}\mathrm{E}[(\mathbf{x} - \hat{\mathbf{x}})(\mathbf{x} - \hat{\mathbf{x}})]^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}\sqrt{\mathbf{R}}^{-T} + \sqrt{\mathbf{R}}^{-1}\mathrm{E}[\mathbf{v}\mathbf{v}]^{\mathrm{T}}\sqrt{\mathbf{R}}^{-T} \tag{3.65}$$

$$\mathrm{E}[\mathbf{e}\mathbf{e}^{\mathrm{T}}] = \sqrt{\mathbf{R}}^{-1}\mathbf{H}\mathbf{P}^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}\sqrt{\mathbf{R}}^{-T} + \sqrt{\mathbf{R}}^{-1}\mathbf{R}\sqrt{\mathbf{R}}^{-T} \; . \tag{3.66}$$

But since

$$\sqrt{\mathbf{R}}^{-1}\mathbf{R}\sqrt{\mathbf{R}}^{-T} = \mathbf{I} \; , \tag{3.67}$$

and naming $\mathbf{H}_z = \sqrt{\mathbf{R}}^{-1}\mathbf{H}$, results in

$$\mathrm{E}[\mathbf{e}\mathbf{e}^{\mathrm{T}}] = \mathbf{H}_z\mathbf{P}\mathbf{H}_z^{\mathrm{T}} + \mathbf{I} \; . \tag{3.68}$$

That is, a measurement noise covariance is now available. Thus, in order to properly use the transformed measurement, in the filter $\sqrt{\mathbf{R}}^{-1}\mathbf{H}$ is used instead of $\mathbf{H}$, and instead of directly computing the residual measurement as $(\mathbf{y} - \mathbf{H}\mathbf{x})$ $(\sqrt{\mathbf{R}}^{-1}\mathbf{y} - \sqrt{\mathbf{R}}^{-1}\mathbf{H}\mathbf{x})$ is implemented.

3.5.3 Computational complexity

As reported in [3] and [16], within the conventional Kalman filter implementation, sequential measurement processing is, in general, more efficient than batch measurement assimilation. Table 3.7, following the computation complexity formulas, as reported in [3], shows a comparison on the number of flops (multiplications and divisions only) needed for batch and sequential measurement assimilation. In the same table, the computational advantage of sequential processing is shown. The graph from Figure 3.16 shows the flops advantage of sequential measurement processing versus batch processing for various values of the number of states and available measurements. From this figure, it can be seen that it is always advantageous to process measurements sequentially, but more importantly, it shows that sequential processing becomes more advantageous quickly for filters with a large number of states. The reason for the sequential filter being more efficient is that many operations are saved mainly because the residual covariance inversion is avoided. Table 3.5 and 3.6 include a breakdown of the number of flops for both sequential and batch processing for the conventional Kalman filter update step.

Table 3.5: Flops required for batch measurement processing. Flop count considers matrices symmetry [3].

Update stage $\mathbf{H}_{m\times n}$ **and** $\mathbf{P}^-_{n\times n}$	Flops (multiply or divide only)
$\mathbf{HP}^-$	mn^2
$\mathbf{H}(\mathbf{HP}^-)^{\mathrm{T}}+\mathbf{R}$	$n(\frac{1}{2}m^2+\frac{1}{2}m)$
$(\mathbf{HP}^-\mathbf{H}^{\mathrm{T}}+\mathbf{R})^{-1}$	$m^3+\frac{1}{2}m^2+\frac{1}{2}m$
$\mathbf{P}^-\mathbf{H}^{\mathrm{T}}(\mathbf{HP}^-\mathbf{H}^{\mathrm{T}}+\mathbf{R})^{-1}$	nm^2
$\mathbf{P}^- - \mathbf{P}^-\mathbf{H}^{\mathrm{T}}(\mathbf{HP}^-\mathbf{H}^{\mathrm{T}}+\mathbf{R})^{-1}\mathbf{HP}^-$	$(\frac{1}{2}(n^2-n)+n)m$
Total	$m^3+\frac{3}{2}nm^2+\frac{1}{2}m^2+nm+\frac{1}{2}m+\frac{3}{2}mn^2$

Table 3.6: Flops required for sequential measurement processing. Flop count considers matrices symmetry [3].

Update stage $\mathbf{H}_{1\times n}$ **and** $\mathbf{P}^-_{n\times n}$	Flops (multiply or divide only)
$\mathbf{HP}^-$	n^2
$\mathbf{H}(\mathbf{HP}^-)^{\mathrm{T}}+\mathbf{R}$	n
$(\mathbf{HP}^-\mathbf{H}^{\mathrm{T}}+\mathbf{R})^{-1}$	1
$\mathbf{P}^-\mathbf{H}^{\mathrm{T}}(\mathbf{HP}^-\mathbf{H}^{\mathrm{T}}+\mathbf{R})^{-1}$	n
$\mathbf{P}^- - \mathbf{P}^-\mathbf{H}^{\mathrm{T}}(\mathbf{HP}^-\mathbf{H}^{\mathrm{T}}+\mathbf{R})^{-1}\mathbf{HP}^-$	$\frac{1}{2}(n^2-n)+n$
Sum $\times m$ measurements	$(\frac{3}{2}n^2+\frac{5}{2}n+1)m$
+ UDU and decorrelation	$\frac{2}{3}m^3+m^2-\frac{5}{3}m+\frac{1}{2}m^2n-\frac{1}{2}mn$
Total	$\frac{2}{3}m^3+m^2-\frac{2}{3}m+\frac{1}{2}m^2n+2mn+\frac{3}{2}mn^2$

In the case of square root filtering, and specifically for the partial-update version, a counting of flops can also be made. The flops required for the conventional square root Kalman filter formula-

Table 3.7: Flop advantage of sequential over batch processing.

Flops advantage	Flops (multiply or divide only)
Batch	$m^3 + \frac{3}{2}nm^2 + \frac{1}{2}m^2 + nm + \frac{1}{2}m + \frac{3}{2}mn^2$
Sequential	$\frac{2}{3}m^3 + m^2 - \frac{2}{3}m + \frac{1}{2}m^2n + 2mn + \frac{3}{2}mn^2$
Sequential advantage over batch	$\frac{1}{2}m^3 - \frac{1}{2}m^2 + m^2n - mn + \frac{7}{6}m$

tion has been extensively well documented and has been analyzed in several scientific papers and books [45],[17],[16]. Here, the computation complexity data reported for square root is used, and the additional operations to perform the partial update operation are included to obtain an estimated overall cost. Table 3.8 shows the approximated cost (multiplications, divides, and square roots) of the square root partial-update filter presented in this chapter, along with the conventional square root filter. Although the use of the partial-update concept requires the extra cost of roughly an extra MGS, now the filter can handle higher uncertainties and nonlinearities at the same time that is more robust numerically. Moreover, it provides the ability to consider on a state-by-state basis at any time and not just a pre-selected set of states, as in the conventional consider filter formulation. Furthermore, the execution of an additional MGS naturally will lead to the triangularization of the square root covariance, which otherwise would be non-triangular with the conventional square root filter. Such a triangularization, in fact, translates into considerable storage savings as the elements above the diagonal are simply zero (for a lower triangular square root matrix), and there is no need to reserve memory space for them.

In any case, the square root filter formulation can add a considerable amount of computations in general. For that reason, alternative ways of factorizing the covariance matrix were developed for Kalman filtering to increase factorized filter efficiency. The UD or modified Cholesky decomposition is an alternative, and in fact, a very efficient one within the Kalman filter framework. It is a more elaborated algorithm, but its efficiency makes the implementation worth it. The next chapter proposes a UD partial-update filter to increase the efficiency of the partial-update filter presented in this chapter.

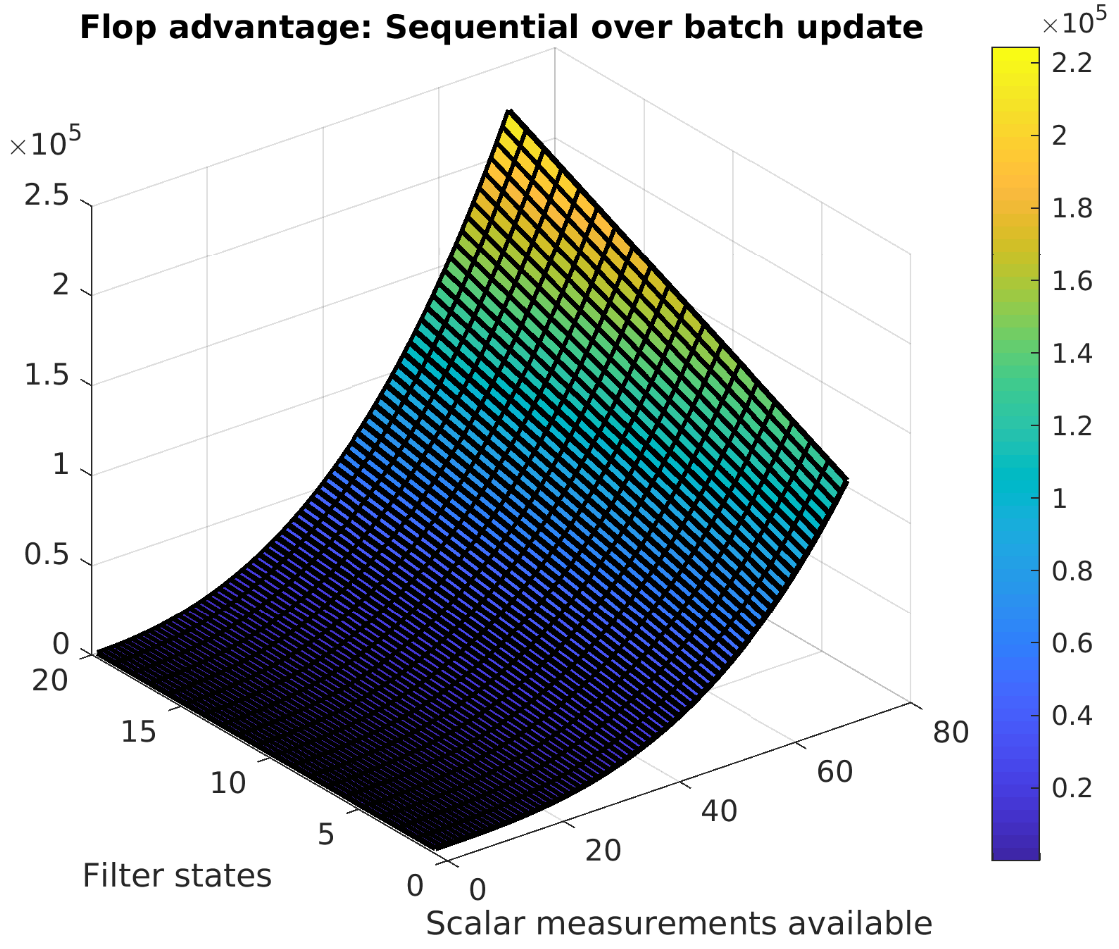


Figure 3.16: Rough complexity comparison for sequential and batch measurement processing for the Kalman filter. UDU decorrelation is considered in the cost to use sequential filtering.

Table 3.8: Conventional and partial-update required flops comparison.

Process	Flops (multiply, divide and square root)
Conventional square root Kalman update	$3n^2 + 4n + \surd + 1$
Incorporation of partial-update extra cost	$n^3 + 3n^2 + 3n + (n+1)\surd \approx 1\ MGS$

3.6 Summary

This chapter presented a square root formulation for the partial-update Schmidt-Kalman filter. This form of the Kalman filter inherits the benefits of the partial-update formulation and combines them with the numerical robustness of the square root form. The result is a filter of higher numerical precision and increased tolerance to nonlinearities and uncertainty level at the cost of almost no additional computational burden. A formulation able to process vector-valued measurements was also presented, and due to its computational burden, it is just seen as a debugging tool and a way to facilitate the implementation of the sequential filter as no re-linearization is needed after measurement assimilation. Lastly, a numerical example, along with Monte Carlo runs, was used to demonstrate the effectiveness of the square root partial-update Schmidt-Kalman filter on a nonlinear system for both numerical stability and robustness to large uncertainties and nonlinearities.

4. U-D PARTIAL-UPDATE KALMAN FILTER

4.1 Introduction

Although the square root formulation was shown to increase the Kalman filter numerical precision and was successfully used in the Apollo missions, the square root Kalman filter presented the issue of considerably increasing the computational cost of the conventional Kalman filter, especially for large systems. Among the alternative proposed techniques to decrease such computational load, the factorized UD Kalman filter update step, originally developed by Bierman and Thornton [76], was a significant advance. This alternative formulation improved the numerical precision of the filter but in a more efficient way than the square root filter. Further, the UD filter has been shown to be more stable than other factorized Kalman filter implementations, being able to handle state vectors with thousands of variables. For these reasons, researchers and even NASA, prefer the UD Kalman filter for hardware implementations [22]. In contrast with the original square root filter, the UD filter uses a factorization that involves an upper triangular matrix $\mathbf{U}$ with 1's on its diagonal, and a diagonal matrix $\mathbf{D}$, such that the covariance matrix is expressed as $\mathbf{P} = \mathbf{U}\mathbf{D}\mathbf{U}^{\mathrm{T}}$. The efficiency of the UD Kalman filter mostly lies in a clever algorithm that significantly exploits the structure of the $\mathbf{U}$ and $\mathbf{D}$ matrices [23].

In this chapter, to reduce the computational complexity of the square root partial-update, the UD factorized version of the partial-update filter is developed.

4.1.1 The UD filter background

Compared with the conventional square root filter formulation, the conventional UD filter does not require square root operations. For this reason, the UD filter is sometimes called square-root free filter. Interestingly, the UD factors can be obtained as a corollary of the Cholesky square-root factors. A closer examination of the Cholesky decomposition of a $n \times n$ matrix, reveals that n square root operations are computed and that those same square-roots appear dividing each column, motivating a pair of alternative Cholesky factors: the UD factors. To illustrate this and further see how the UD factors can be obtained, consider a 3×3 covariance matrix, $\mathbf{P}$, to be analytically factorized via Cholesky decomposition as

$$\mathbf{P} = \mathbf{S}\mathbf{S}^{\mathrm{T}} \,, \tag{4.1}$$

with $\mathbf{S}$ written as

$$\mathbf{S} = \begin{bmatrix} \sqrt{\mathbf{P}_{11} - \frac{\mathbf{P}_{13}^2}{\mathbf{P}_{33}} - \frac{(\mathbf{P}_{12} - \frac{\mathbf{P}_{13}\mathbf{P}_{23}}{\mathbf{P}_{33}})^2}{\mathbf{P}_{22} - \frac{\mathbf{P}_{23}^2}{\mathbf{P}_{33}}}} & \frac{\mathbf{P}_{12} - \frac{\mathbf{P}_{13}\mathbf{P}_{23}}{\mathbf{P}_{33}}}{\sqrt{\mathbf{P}_{22} - \frac{\mathbf{P}_{23}^2}{\mathbf{P}_{33}}}} & \frac{\mathbf{P}_{13}}{\sqrt{\mathbf{P}_{33}}} \\ 0 & \sqrt{\mathbf{P}_{22} - \frac{\mathbf{P}_{23}^2}{\mathbf{P}_{33}}} & \frac{\mathbf{P}_{23}}{\sqrt{\mathbf{P}_{33}}} \\ 0 & 0 & \sqrt{\mathbf{P}_{33}} \end{bmatrix} , \tag{4.2}$$

which can itself be factorized as

$$\mathbf{S} = \begin{bmatrix} 1 & \frac{\mathbf{P}_{12} - \frac{\mathbf{P}_{13}\mathbf{P}_{23}}{\mathbf{P}_{33}}}{\mathbf{P}_{22} - \frac{\mathbf{P}_{23}^2}{\mathbf{P}_{33}}} & \frac{\mathbf{P}_{13}}{\mathbf{P}_{33}} \\ 0 & 1 & \frac{\mathbf{P}_{23}}{\mathbf{P}_{33}} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \sqrt{\mathbf{P}_{11} - \frac{\mathbf{P}_{13}^2}{\mathbf{P}_{33}} - \frac{(\mathbf{P}_{12} - \frac{\mathbf{P}_{13}\mathbf{P}_{23}}{\mathbf{P}_{33}})^2}{\mathbf{P}_{22} - \frac{\mathbf{P}_{23}^2}{\mathbf{P}_{33}}}} & 0 & 0 \\ 0 & \sqrt{\mathbf{P}_{22} - \frac{\mathbf{P}_{23}^2}{\mathbf{P}_{33}}} & 0 \\ 0 & 0 & \sqrt{\mathbf{P}_{33}} \end{bmatrix} . \tag{4.3}$$

By calling the matrix with 1's in the diagonal $\mathbf{U}$ and the matrix containing the square roots as $\sqrt{\mathbf{D}}$, $\mathbf{S}$ is written as

$$\mathbf{S} = \mathbf{U}\sqrt{\mathbf{D}} \,, \tag{4.4}$$

and thus, the covariance matrix reads

$$\mathbf{P} = \mathbf{S}\mathbf{S}^{\mathrm{T}} = \mathbf{U}\sqrt{\mathbf{D}}\sqrt{\mathbf{D}}\mathbf{U}^{\mathrm{T}} = \mathbf{U}\mathbf{D}\mathbf{U}^{\mathrm{T}} \,. \tag{4.5}$$

As can be observed from this development, although the $\mathbf{UD}$ product is equivalent to the Cholesky factor $\mathbf{S}$, the $\mathbf{UD}$ factors do not involve square root operations. Moreover, via this factorization, the check for singularity or positive definiteness is straightforward as it suffices to revise the sign of the diagonal elements of $\mathbf{D}$ (when monitoring for numerical problems that can be affecting the covariance matrix). Also, it is important to note that the determinant of $\mathbf{U}$ is equal to one, and thus has an inverse, and inversion of an upper/lower triangular matrix with 1's in the diagonal, is an upper/lower triangular matrix. This fact will become relevant in the following sections.

Specific algorithms to obtain the modified Cholesky factorization are available in the filtering literature [75], [68], [3], or in linear algebra books or matrix operations books [72] [69]. However, it is recommended to use Kalman filter oriented routines as they are specifically structured to compute the required filter quantities in an efficient way, as those presented in [22]. Similarly, to obtain the $\mathbf{UD}$ factors given the positive semi-definite and symmetric covariance matrix $\mathbf{P}$.

In the following section, and before the development of the UD partial-update filter is presented, a superficial overview of the conventional UD filter is given with the purpose of establishing the appropriate context.

4.2 The conventional UD Kalman filter

Since the partial-update approach modifies the measurement-update step, similar to the previously proposed methods, the conventional UD filter and the UD partial-update form proposed in this chapter share the same temporal update. Although several approaches exist in the literature to execute a time update when using the UD filter, only one form is presented in this work. More specifically, the time-update used for this work is the same presented in [22] and [68]. The selection of the time propagation method, however, is just a user's preference, and it does not affect the proposed partial-update development.

4.2.1 UD temporal update overview

Consider the conventional discrete covariance propagation equation to obtain the prior covariance (with no time indices for clarity),

$$\mathbf{P}^{-} = \mathbf{F}\mathbf{P}^{+}\mathbf{F}^{\mathrm{T}} + \mathbf{G}\mathbf{Q}\mathbf{G}^{\mathrm{T}}\,, \tag{4.6}$$

with $\mathbf{G}$ being the $(n \times q)$ matrix mapping process noise to the state. A direct attempt to obtain a factorization that involves three factors, to start forming $\mathbf{P}^- = \underline{\mathbf{U}}\underline{\mathbf{D}}\underline{\mathbf{U}}^{\mathrm{T}}$ (sub-bar indicates a prior quantity), leads to the candidate form of

$$\mathbf{P}^- = \begin{bmatrix} \mathbf{F}\overset{+}{\mathbf{U}} & \mathbf{G} \end{bmatrix} \begin{bmatrix} \overset{+}{\mathbf{D}} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q} \end{bmatrix} \begin{bmatrix} \overset{+}{\mathbf{U}}^{\mathrm{T}} \mathbf{F}^{\mathrm{T}} \\ \mathbf{G}^{\mathrm{T}} \end{bmatrix} = \mathbf{W}\hat{\mathbf{D}}\mathbf{W}^{\mathrm{T}} \,, \tag{4.7}$$

where the plus sign above variables is to indicate a posterior quantity. The matrix $\hat{\mathbf{D}}$ defined above, is diagonal, but is an $(n+q) \times (n+q)$ matrix . Further, $\mathbf{W} = \begin{bmatrix} \mathbf{F}\overset{+}{\mathbf{U}} & \mathbf{G} \end{bmatrix}$ is a $n \times (n+q)$ matrix and is not upper triangular in general, however, some work can be done to triangularize it. With this in mind, it is sought to satisfy

$$\mathbf{P}^- = \underline{\mathbf{U}}\underline{\mathbf{D}}\underline{\mathbf{U}}^{\mathrm{T}} = \mathbf{W}\hat{\mathbf{D}}\mathbf{W}^{\mathrm{T}} \,, \tag{4.8}$$

in a way that the factors $\underline{\mathbf{U}}$ and $\underline{\mathbf{D}}$ can be computed with proper dimensions given $\mathbf{W}$ and $\hat{\mathbf{D}}$. In other words, although at this point a direct matrix-by-matrix relationship (i.e. $\underline{\mathbf{U}} \neq \mathbf{W}$ nor $\underline{\mathbf{D}} \neq \hat{\mathbf{D}}$) cannot be established, it is desired. To accomplish this, first consider the weighted inner product

$$\mathbf{v}_k \hat{\mathbf{D}} \mathbf{v}_j^T = 0 \quad k \neq j \,, \tag{4.9}$$

where the $(n+q)$ $\mathbf{v}_i$ row vectors can be found via the Weighted Modified Gram-Schmidt (WMGS) orthogonalization procedure (that uses the $(n+q)$ row vectors, $\mathbf{w}_i$, of the matrix $\mathbf{W}$):

$$\mathbf{v}_n = \mathbf{w}_n \,, \tag{4.10}$$

$$\mathbf{v}_k = \mathbf{w}_k - \sum_{j=k+1}^{n} u(k,j)\mathbf{v}_j \quad k = n-1, \ldots, 1 \,, \tag{4.11}$$

$$u(k,j) = \frac{\mathbf{w}_k \hat{\mathbf{D}} \mathbf{v}_j^T}{\mathbf{v}_j \hat{\mathbf{D}} \mathbf{v}_j} \quad j,k = 1, \ldots, n \,. \tag{4.12}$$

Alternatively,

$$\mathbf{w}_k = \mathbf{v}_k + \sum_{j=k+1}^{n} u(k,j)\mathbf{v}_j \quad k = 1, \ldots, n\ , \tag{4.13}$$

or

$$\mathbf{w}_k^T = \mathbf{v}_k^T + \sum_{j=k+1}^{n} u(k,j)\mathbf{v}_j^T \quad k = 1, \ldots, n\ , \tag{4.14}$$

which can be expressed in matrix form as

$$\begin{bmatrix} \mathbf{w}_1 \\ \vdots \\ \mathbf{w}_n \end{bmatrix} = \begin{bmatrix} 1 & u(1,2) & \ldots & u(1,n) \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & u(n-1,n) \\ 0 & \ldots & \ldots & 1 \end{bmatrix} \begin{bmatrix} \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_n \end{bmatrix}\ , \tag{4.15}$$

or

$$\mathbf{W} = \mathbf{U}\mathbf{V}\ . \tag{4.16}$$

With this expression in hand, Equation (4.8) can now be written as

$$\mathbf{P}^- = \underset{\sim}{\mathbf{U}}\underset{\sim}{\mathbf{D}}\underset{\sim}{\mathbf{U}}^{\mathrm{T}} = \mathbf{W}\hat{\mathbf{D}}\mathbf{W}^{\mathrm{T}} = (\mathbf{U}\mathbf{V})\hat{\mathbf{D}}(\mathbf{U}\mathbf{V})^{\mathrm{T}} = \mathbf{U}[\mathbf{V}\hat{\mathbf{D}}\mathbf{V}^{\mathrm{T}}]\mathbf{U}^{\mathrm{T}}\ . \tag{4.17}$$

Since the product $\mathbf{V}\hat{\mathbf{D}}\mathbf{V}^{\mathrm{T}}$ is constructed according to Equation (4.9), the bracketed term in the previous equation is diagonal. Thus, the propagated factors are given by,

$$\underset{\sim}{\mathbf{U}} = \mathbf{U}\ , \tag{4.18}$$

and

$$\underset{\sim}{\mathbf{D}} = \mathbf{V}\hat{\mathbf{D}}\mathbf{V}^{\mathrm{T}}\ . \tag{4.19}$$

In summary, once $\mathbf{W}$ and $\hat{\mathbf{D}}$ are formed, the execution of the WMGS will provide the propagated factors, $\underset{\sim}{\mathbf{U}}$ and $\underset{\sim}{\mathbf{D}}$. Regarding the state propagation, this is executed normally via system dynamics.

4.3 The UD partial-update derivation

To begin with the derivation of the UD partial-update filter, recall the matrix form of partial-update,

$$\mathbf{P}^{++} = \mathbf{\Gamma}(\mathbf{P}^{-} - \mathbf{P}^{+})\mathbf{\Gamma} + \mathbf{P}^{+} \;, \tag{4.20}$$

where again, $\mathbf{\Gamma}$ is the diagonal matrix with elements γ_i for $i = 1, 2 \ldots n$ (recall that $\gamma_i = 1 - \beta_i$). Now, from the standard EKF equations, $\mathbf{P}^{+} = (\mathbf{I} - \mathbf{K}\,\mathbf{H})\mathbf{P}^{-}$ is incorporated into (4.20).

$$\mathbf{P}^{++} = \mathbf{P}^{+} + \mathbf{\Gamma}(\mathbf{K}\mathbf{H}\mathbf{P}^{-})\mathbf{\Gamma} \;, \tag{4.21}$$

then replacing $\mathbf{K} = \mathbf{P}^{-}\mathbf{H}^{\mathrm{T}}(\mathbf{H}\mathbf{P}^{-}\mathbf{H}^{\mathrm{T}} + \mathbf{R})^{-1}$ gives

$$\mathbf{P}^{++} = \mathbf{P}^{+} + \mathbf{\Gamma}(\mathbf{P}^{-}\mathbf{H}^{\mathrm{T}}(\mathbf{H}\mathbf{P}^{-}\mathbf{H}^{\mathrm{T}} + \mathbf{R})^{-1}\mathbf{H}\mathbf{P}^{-})\mathbf{\Gamma} \;. \tag{4.22}$$

Next, the posterior covariance $\mathbf{P}^{+}$ is written in terms of the prior covariance by using

$$\mathbf{P}^{+} = (\mathbf{I} - \mathbf{K}\,\mathbf{H})\mathbf{P}^{-} \tag{4.23}$$

$$= \mathbf{P}^{-} - \mathbf{P}^{-}\mathbf{H}^{\mathrm{T}}(\mathbf{H}\mathbf{P}^{-}\mathbf{H}^{\mathrm{T}} + \mathbf{R})^{-1}\mathbf{H}\mathbf{P}^{-} \;, \tag{4.24}$$

and incorporating it into Equation (4.22) gives

$$\mathbf{P}^{++} = \mathbf{P}^{-} - \mathbf{P}^{-}\mathbf{H}^{\mathrm{T}}(\mathbf{H}\mathbf{P}^{-}\mathbf{H}^{\mathrm{T}} + \mathbf{R})^{-1}\mathbf{H}\mathbf{P}^{-} + \mathbf{\Gamma}\mathbf{P}^{-}\mathbf{H}^{\mathrm{T}}(\mathbf{H}\mathbf{P}^{-}\mathbf{H}^{\mathrm{T}} + \mathbf{R})^{-1}\mathbf{H}\mathbf{P}^{-}\mathbf{\Gamma} \;. \tag{4.25}$$

At this point, a few variables are renamed and expressions rearrangements are done. First, the UD decomposition is indicated for the covariance matrix.

$$\mathbf{P}^{++} = \underline{\mathbf{U}}\underline{\mathbf{D}}\underline{\mathbf{U}}^{\mathrm{T}} - \underline{\mathbf{U}}\underline{\mathbf{D}}\underline{\mathbf{U}}^{\mathrm{T}}\mathbf{H}_i^{\mathrm{T}}(\mathbf{H}_i\underline{\mathbf{U}}\underline{\mathbf{D}}\underline{\mathbf{U}}^{\mathrm{T}}\mathbf{H}_i^{\mathrm{T}} + \mathbf{R}_i)^{-1}\mathbf{H}_i\underline{\mathbf{U}}\underline{\mathbf{D}}\underline{\mathbf{U}}^{\mathrm{T}} + \tag{4.26}$$

$$\mathbf{\Gamma}\underline{\mathbf{U}}\underline{\mathbf{D}}\underline{\mathbf{U}}^{\mathrm{T}}\mathbf{H}_i^{\mathrm{T}}(\mathbf{H}_i\underline{\mathbf{U}}\underline{\mathbf{D}}\underline{\mathbf{U}}^{\mathrm{T}}\mathbf{H}_i^{\mathrm{T}} + \mathbf{R}_i)^{-1}\mathbf{H}_i\underline{\mathbf{U}}\underline{\mathbf{D}}\underline{\mathbf{U}}^{\mathrm{T}}\mathbf{\Gamma} \;.$$

Second, the variables $\underline{\mathbf{w}} = \underline{\mathbf{U}}^{\mathrm{T}}\mathbf{H}_i^{\mathrm{T}}$ and $A_i = (\mathbf{H}_i\underline{\mathbf{U}}\underline{\mathbf{D}}\underline{\mathbf{U}}^{\mathrm{T}}\mathbf{H}_i^{\mathrm{T}} + \mathbf{R}_i)^{-1}$ are introduced, and used in the previous equation. This results in,

$$\mathbf{P}^{++} = \underline{\mathbf{U}}\underline{\mathbf{D}}\underline{\mathbf{U}}^{\mathrm{T}} - \underline{\mathbf{U}}\underline{\mathbf{D}}\underline{\mathbf{w}}A_i\underline{\mathbf{w}}^{\mathrm{T}}\underline{\mathbf{D}}\underline{\mathbf{U}}^{\mathrm{T}} + \mathbf{\Gamma}\underline{\mathbf{U}}\underline{\mathbf{D}}\underline{\mathbf{w}}A_i\underline{\mathbf{w}}^{\mathrm{T}}\underline{\mathbf{D}}\underline{\mathbf{U}}^{\mathrm{T}}\mathbf{\Gamma}\ , \tag{4.27}$$

and by factorizing $\underline{\mathbf{U}}$ and $\underline{\mathbf{U}}^{\mathrm{T}}$, the following expression is obtained

$$\mathbf{P}^{++} = \underline{\mathbf{U}}[\underline{\mathbf{D}} - (\underline{\mathbf{D}}\underline{\mathbf{w}})A_i(\underline{\mathbf{D}}\underline{\mathbf{w}})^{\mathrm{T}}]\underline{\mathbf{U}}^{\mathrm{T}} + \mathbf{\Gamma}\underline{\mathbf{U}}[(\underline{\mathbf{D}}\underline{\mathbf{w}})A_i(\underline{\mathbf{D}}\underline{\mathbf{w}})^{\mathrm{T}}]\underline{\mathbf{U}}^{\mathrm{T}}\mathbf{\Gamma} \tag{4.28}$$

$$= \underline{\mathbf{U}}\{[\underline{\mathbf{D}} - (\underline{\mathbf{D}}\underline{\mathbf{w}})A_i(\underline{\mathbf{D}}\underline{\mathbf{w}})^{\mathrm{T}}] + \underline{\mathbf{U}}^{-1}\mathbf{\Gamma}\underline{\mathbf{U}}[(\underline{\mathbf{D}}\underline{\mathbf{w}})A_i(\underline{\mathbf{D}}\underline{\mathbf{w}})^{\mathrm{T}}]\underline{\mathbf{U}}^{\mathrm{T}}\mathbf{\Gamma}\underline{\mathbf{U}}^{-T}\}\underline{\mathbf{U}}^{\mathrm{T}}\ . \tag{4.29}$$

Now because the curly-bracketed term is positive semi-definite, its UD decomposition can be computed resulting in a decomposition formed by $\mathcal{UDU}^{\mathrm{T}}$. The obtained factors, $\mathcal{UDU}^{\mathrm{T}}$, are then used in Equation (4.28) and the result reads

$$\mathbf{P}^{++} = \underline{\mathbf{U}}\{\mathcal{UDU}^{\mathrm{T}}\}\underline{\mathbf{U}}^{\mathrm{T}}\ . \tag{4.30}$$

The partial-updated (posterior) covariance matrix can also be expressed in terms of UD factors as

$$\mathbf{P}^{++} = \overset{++}{\mathbf{U}}\ \overset{++}{\mathbf{D}}\ \overset{++}{\mathbf{U}}^{\mathrm{T}} = \underline{\mathbf{U}}\{\mathcal{UDU}^{\mathrm{T}}\}\underline{\mathbf{U}}^{\mathrm{T}}, \tag{4.31}$$

and since the product of two upper triangular matrix with 1's in the diagonal is also an upper triangular matrix with 1's in the diagonal, the partial-updated factors can be directly identified as

$$\overset{++}{\mathbf{U}} = \underline{\mathbf{U}}\mathcal{U}\ , \tag{4.32}$$

and

$$\overset{++}{\mathbf{D}} = \mathcal{D}\ , \tag{4.33}$$

which gives the expressions to execute the partial-update.

Before moving to the numerical examples that show the functionality of this formulation, a few remarks are given. First, notice that if $\mathbf{\Gamma}$ is zero (full-update), this formulation becomes the conventional UD filter. Second, although it seems that a considerable amount of operations are needed to compute the second term in Equation (4.28) (the term involving $\mathbf{\Gamma}$) by taking advantage of the matrix structure (upper triangular and diagonals) the extra burden is barely a fraction of a multiplication of two $(n \times n)$ matrices. Third, it is also important to mention that even though this extra (curly bracketed) term needs to be formed and involved in the UD decomposition, the size of the matrix that enters the UD decomposition is of the same size as for the conventional filter; and thus no extra computation is incurred here with respect to the conventional UD filter. Finally, note that in the derivation, there is no assumption on the form of A_i (thus the capital letter notation), allowing that the same formulation can be used to process either vector or scalar measurements. However, the UD factorized filter is generally used in scalar measurement mode as it is more efficient than processing the measurements as a vector. In case of having measurement correlations ($\mathbf{R}$ not a diagonal matrix), similarly to the square root filter, a decorrelation procedure can be executed. The procedure that decorrelates $\mathbf{R}$ via the already computed UD factors is presented in the following subsection.

Table 4.1 summarizes the UD partial-update Schmidt-Kalman filter equations.

4.4 Measurement decorrelation using UD factors

Let the UD decomposition of a non-diagonal measurement noise covariance, $\mathbf{R}_c$, be

$$\mathbf{R}_c = \mathbf{U}_R \mathbf{D}_R \mathbf{U}_R^{\mathrm{T}} \, , \tag{4.34}$$

and consider the measurement equation $\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{v}$, being transformed by $\mathbf{U}_R^{-1}$ as

$$\mathbf{z} = \mathbf{U}_R^{-1}\mathbf{y} = \mathbf{U}_R^{-1}\mathbf{H}\mathbf{x} + \mathbf{U}_R^{-1}\mathbf{v} \, . \tag{4.35}$$

Table 4.1: U-D partial-update Schmidt-Kalman filter

Model	$\mathbf{x}_k = \mathbf{f}_{k-1}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) + \mathbf{w}_{k-1}$ $\tilde{\mathbf{y}}_k = \mathbf{h}_k(\mathbf{x}_k) + \mathbf{v}_k$ $\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k)$ $\mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k)$
Initialize	$\hat{\mathbf{x}}_0^+ = \mathbf{x}_0$ $[\overset{+}{\mathbf{U}}_0, \overset{+}{\mathbf{D}}_0] = udu(\mathbf{P}_0^+)$ [3] $\mathbf{\Gamma} = \mathrm{diag}(1-\beta_1, 1-\beta_2, ..., 1-\beta_n)$
Propagation	$\hat{\mathbf{x}}_k^- = \mathbf{f}_{k-1}(\hat{\mathbf{x}}_{k-1}^+, \mathbf{u}_{k-1})$ $Form$ $\mathbf{W} = \begin{bmatrix} \mathbf{F}\overset{+}{\mathbf{U}} & \mathbf{I} \end{bmatrix}$; $\hat{\mathbf{D}} = \begin{bmatrix} \overset{+}{\mathbf{D}} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q} \end{bmatrix}$ $[\underline{\mathbf{U}}, \underline{\mathbf{D}}] = WMGS(\mathbf{W}, \hat{\mathbf{D}})$ [68]
Gain	$\mathbf{K}_k = \underline{\mathbf{U}}\underline{\mathbf{D}}\underline{\mathbf{w}}A_i$ $A_i = (\underline{\mathbf{w}}^{\mathrm{T}}\underline{\mathbf{D}}\underline{\mathbf{w}} + \mathbf{R}_k)^{-1};\ \underline{\mathbf{w}} = \underline{\mathbf{U}}^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}$
Partial-Update	$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + (\mathbf{I} - \mathbf{\Gamma})\mathbf{K}_k(\tilde{\mathbf{y}}_k - \hat{\mathbf{y}}_k)$ $\mathbf{L} = (\underline{\mathbf{D}}\underline{\mathbf{w}})A_i(\underline{\mathbf{D}}\underline{\mathbf{w}})^{\mathrm{T}}$ $\mathcal{U}\mathcal{D}\mathcal{U}^{\mathrm{T}} = udu(\underline{\mathbf{D}} - \mathbf{L} + \underline{\mathbf{U}}^{-1}\mathbf{\Gamma}\underline{\mathbf{U}}\mathbf{L}\underline{\mathbf{U}}^{\mathrm{T}}\mathbf{\Gamma}\underline{\mathbf{U}}^{-T})$ $\overset{++}{\mathbf{U}} = \underline{\mathbf{U}}\mathcal{U}$ $\overset{++}{\mathbf{D}} = \mathcal{D}$

The residual is then

$$\mathbf{e} = \mathbf{z} - \hat{\mathbf{z}} = \mathbf{U}_R^{-1}\mathbf{H}(\mathbf{x} - \hat{\mathbf{x}}) + \mathbf{U}_R^{-1}\mathbf{v}\ , \tag{4.36}$$

with covariance

$$\mathbf{E}[\mathbf{e}\mathbf{e}^{\mathrm{T}}] = \mathbf{E}[(\mathbf{U}_R^{-1}\mathbf{H}(\mathbf{x} - \hat{\mathbf{x}}) + \mathbf{U}_R^{-1}\mathbf{v})(\mathbf{U}_R^{-1}\mathbf{H}(\mathbf{x} - \hat{\mathbf{x}}) + \mathbf{U}_R^{-1}\mathbf{v})^{\mathrm{T}}]\ , \tag{4.37}$$

$$\mathbf{E}[\mathbf{e}\mathbf{e}^{\mathrm{T}}] = \mathbf{U}_R^{-1}\mathbf{H}\mathbf{E}[(\mathbf{x} - \hat{\mathbf{x}})(\mathbf{x} - \hat{\mathbf{x}})]^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}\mathbf{U}_R^{-T} + \mathbf{U}_R^{-1}\mathbf{E}[\mathbf{v}\mathbf{v}]^{\mathrm{T}}\mathbf{U}_R^{-T}\ , \tag{4.38}$$

$$\mathbf{E}[\mathbf{e}\mathbf{e}^{\mathrm{T}}] = \mathbf{U}_R^{-1}\mathbf{H}\mathbf{P}^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}\mathbf{U}_R^{-T} + \mathbf{U}_R^{-1}\mathbf{R}_c\mathbf{U}_R^{-T}\ . \tag{4.39}$$

But since

$$\mathbf{U}_R^{-1}\mathbf{R}_c\mathbf{U}_R^{-T} = \mathbf{D}_R \; , \tag{4.40}$$

and naming $\mathbf{H}_z = \mathbf{U}_R^{-1}\mathbf{H}$, results in

$$\mathrm{E}[\mathbf{e}\mathbf{e}^{\mathrm{T}}] = \mathbf{H}_z\mathbf{P}\mathbf{H}_z^{\mathrm{T}} + \mathbf{D}_R \; . \tag{4.41}$$

That is, the transformed equation now uses a diagonal measurement noise covariance. In order to properly use the transformed measurement, the filter simply uses $\mathbf{U}_R^{-1}\mathbf{H}$ instead of $\mathbf{H}$, and instead of directly computing the residual measurement with $(\mathbf{y} - \mathbf{H}\mathbf{x})$, $\mathbf{U}_R^{-1}(\mathbf{y} - \mathbf{H}\mathbf{x})$ is used. Rather than using the correlated measurement noise $\mathbf{R}_c$, the filter now uses $\mathbf{D}_R$ and if desired, the filter measurement can be processed sequentially.

4.5 Numerical example

The UD partial-update filter is exercised using the re-entering body scenario that was used for the square-root version of the filter. The process and measurement model, as well as the initial conditions, remain the same for these simulations. The partial-update weights, $\boldsymbol{\beta}$, were also unaltered. Since the UD partial-update and the square-root partial-update filter are mathematically equivalent, the main objective of this numerical simulations is to show that the numerical properties that were gained with the square root factorization, remained for the UD version; with the only difference being the UD filter version more computationally efficient.

4.5.1 Body re-entering Earth atmosphere single run

Similar to the description of the results from the square root partial-update filter, three main simulations are presented. First, in Figure 4.1, the conventional EKF, along with the UD partial-update filter estimates, are depicted for a single run. For this simulation, the partial-update weights are all set to 1 (conventional full update). The estimates for both filters, as expected, show divergence as the EKF cannot handle the simulation's large initial uncertainties. The UD formulation is equivalent to the EKF, and the estimates are not different.

In Figure 4.2, the results of applying a partial-update with $\boldsymbol{\beta} = \text{diag}\,[0.9, 0.96, 0.75]$, are shown. Again, these update weight values were selected to limit the updates mainly for the ballistic parameter, since it is commonly less observable than the other states. In this scenario, the filters are seen to be able to handle the given initial conditions. As in the square root filter, the UD formulation now incorporates the partial-update benefits. In the intent to show that the square root and the UD partial-update formulations are mathematically equivalent, they are graphed in Figure 4.3 and shown to produce the same state and covariance estimates. Although these results are from a single run, they are representative of a typical run for this scenario. For this filter, the condition number plot is omitted as it is equivalent and conveys the same information than for the graph presented in the square root partial-update chapter: the UD factors have lower condition number than the full covariance matrix.

4.6 Monte Carlo runs

Monte Carlo simulations were run for both, UD EKF and UD partial-update EKF. A total of 100 runs were executed, and the histories of all of the states were recorded. The sampled standard deviation and the standard deviation, as computed by the filter were also calculated and used to check for filter consistency. For both filters, the initial conditions are maintained the same from the single run re-entering body scenario from the previous chapter. For convenience, the parameter values used for the simulations are condensed in Table 4.2.

Figure 4.4 shows the 100 EKF runs histories. The EKF shows that, in its majority, it can prevent the filter's total failure, except for two cases (blue curves) that are completely divergent. However, most of the runs show similar behavior as the one shown for the single run: the error estimates are not within the proper sigma bounds after around $t = 15$ seconds, and the errors do not converge to zero. The UD partial-update technique, on the other hand (as depicted in Figure 4.5), and as expected from the results in the previous chapter, shows a dramatic improvement over the EKF. First, for the same sampled initial conditions, the filter presents no runs with divergence. Second, the error histories show a significant magnitude reduction, and third, a superior capacity to handle the initial uncertainties by avoiding the overreaction in the update is achieved.

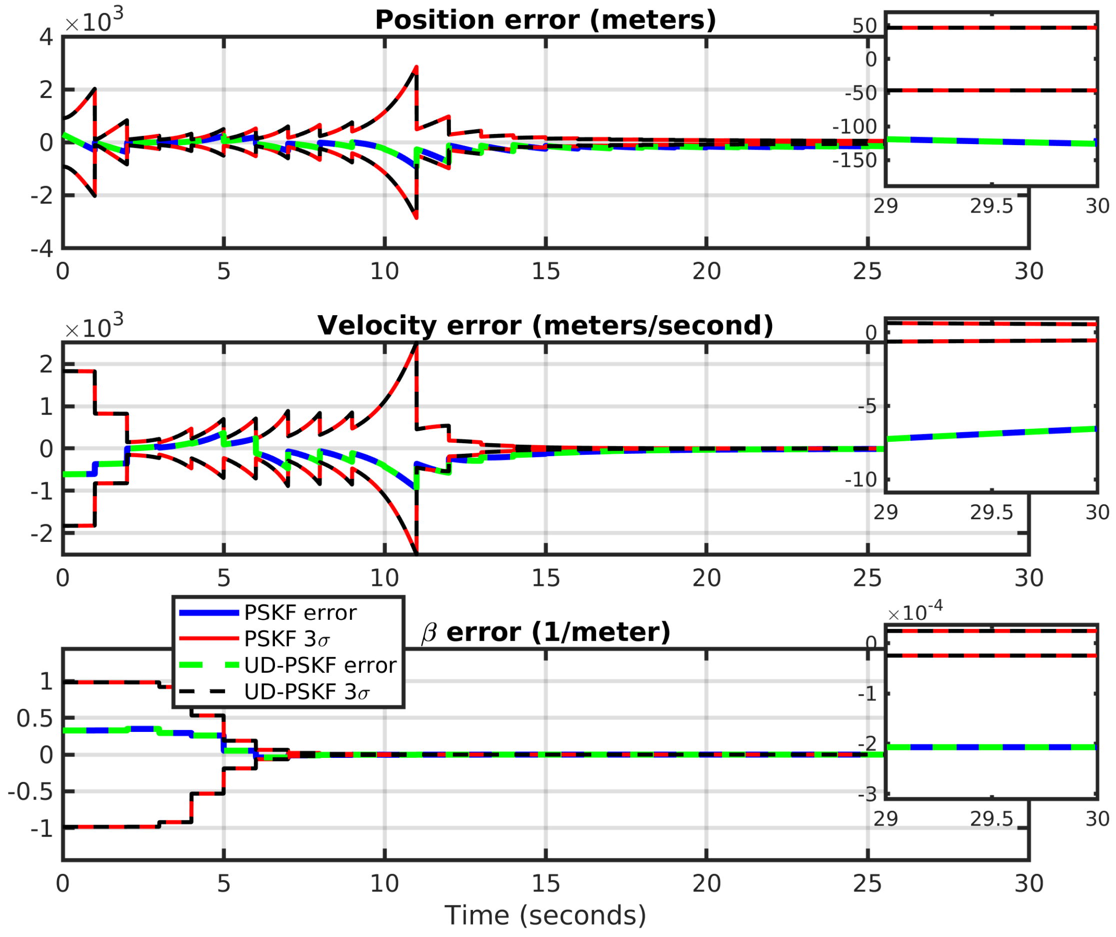


Figure 4.1: Standard EKF and UD partial-update EKF with full updates. The inset on the right shows a zoom-in for the last second of the simulation, displaying significant filter inconsistency with estimates well outside of 3σ bounds.

The averaged standard deviation from the Monte Carlo runs, and the standard deviation estimated by the filter are depicted in Figure 4.6, and are shown to practically coincide, and since the mean estimation error is around zero, this indicates that the filter is consistent. Since the EKF filter presented divergent cases, only standard deviations and mean error were plotted for the partial-update filter (EKF is inconsistent for this scenario).

Overall, it was observed that if low initial errors were ensured, the EKF can be functional and quickly converge to zero error, but is not robust enough to handle errors at the level of the partial-

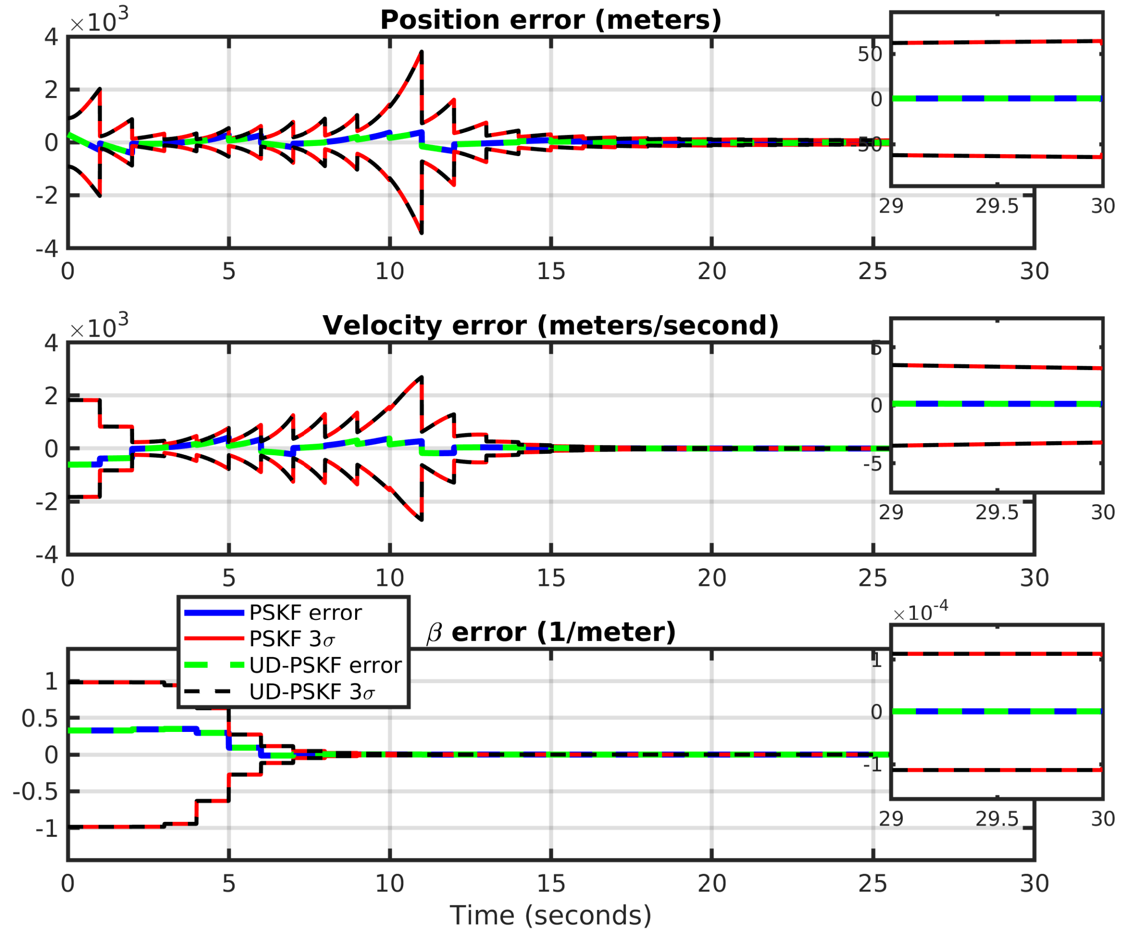


Figure 4.2: Standard EKF and UD partial-update EKF with partial updates. The inset on the right shows a zoom-in for the last second of the simulation, displaying significant filter inconsistency with estimates well outside of 3σ bounds.

Table 4.2: Re-entering body parameters

State/parameter uncertainty	Uncertainty 1σ value
Position	$300\,\mathrm{m}$
Velocity	$600\,\mathrm{m/s}$
Ballistic parameter	$0.33\,\mathrm{m}^{-1}$
Measurement	$300\,\mathrm{m}$

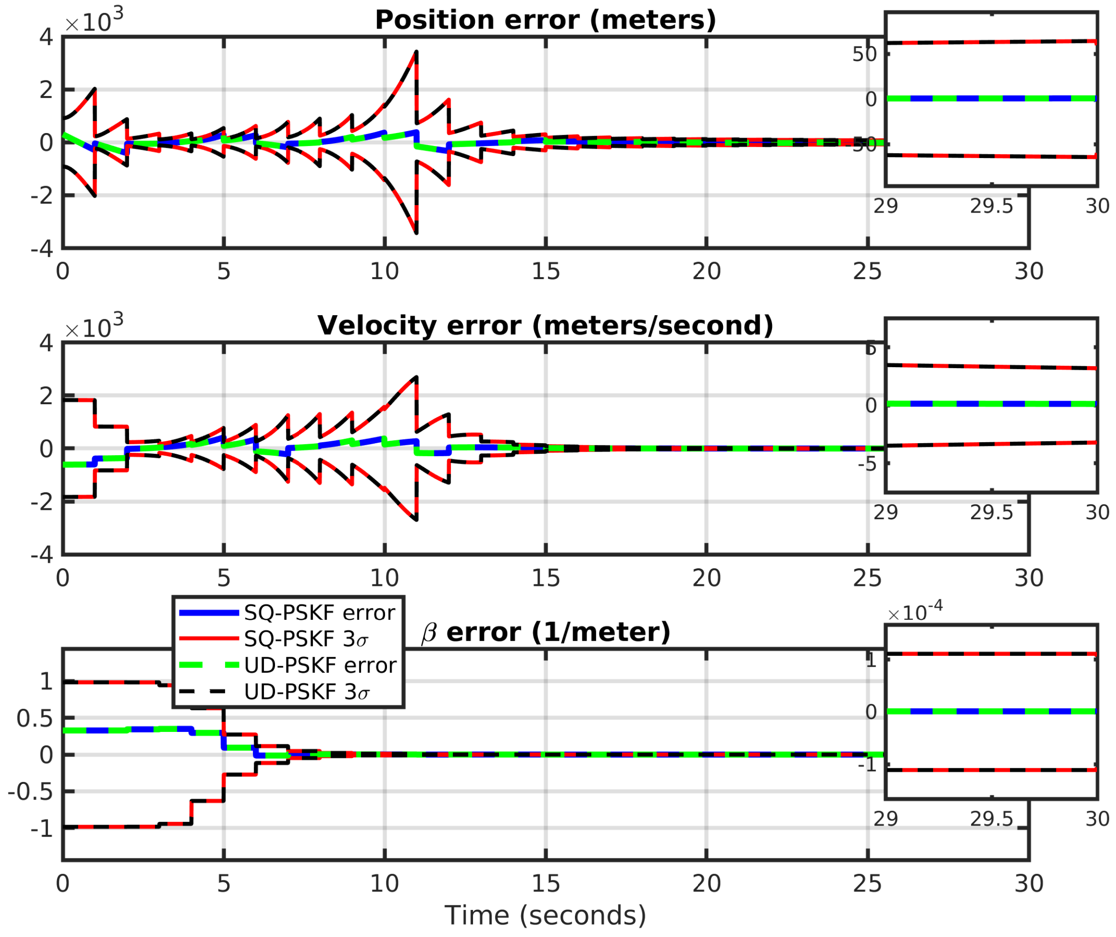


Figure 4.3: partial-update EKF and UD partial-update EKF with $\boldsymbol{\beta} = \begin{bmatrix} 0.9 & 0.9 & 0.75 \end{bmatrix}$ resulting in 90%, 90%, and 75% updates respectively to the position, velocity, and ballistic coefficient states. The inset on the right shows a zoom-in for the last second of the simulation, displaying filter results with estimates within the 3σ bounds.

update UD filter. Conversely, the UD partial-update filter was observed to be consistent, and able to handle higher nonlinearities and uncertainties better than the UD EKF or EKF. Again, the UD partial-update filter results, as expected, match those from the square root partial-update filter, since the change in factorization does not provide any additional robustness against uncertainties, and it is just a mechanism to gain computational efficiency.

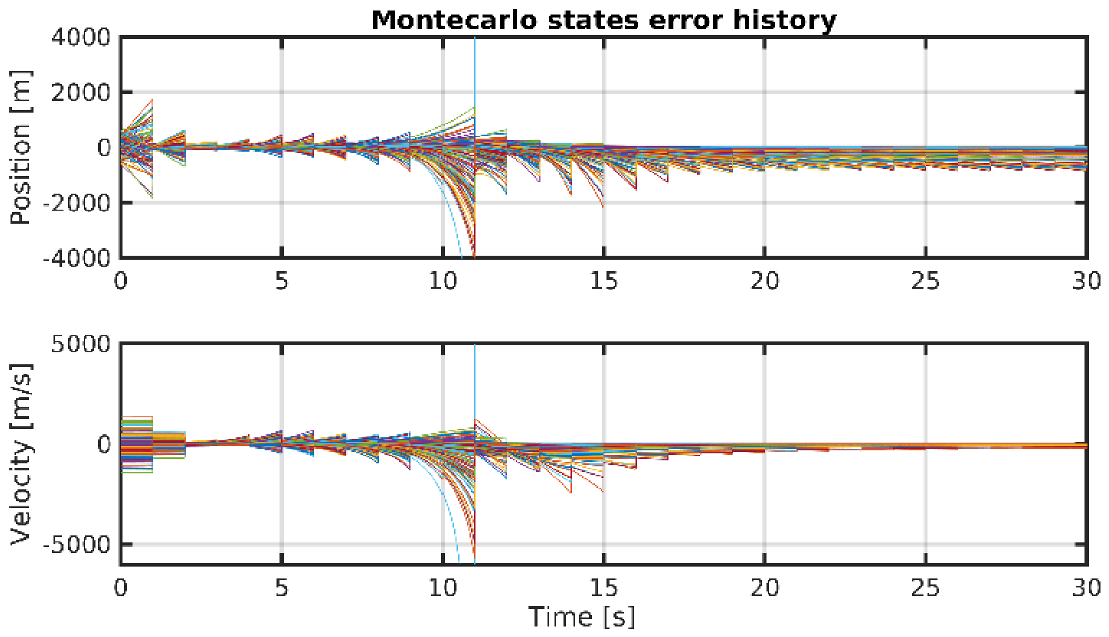


Figure 4.4: Monte Carlo standard EKF and UD partial-update EKF with full updates.

4.7 Numerical complexity

The UD formulation's main objective is to provide an alternative filter in which the computational burden is lower than for the square root formulation, but retains the numerical properties of a factorized filter. The conventional UD filter is more efficient than the conventional square root formulation, and for the case of the partial-update formulations, this is also the case. Table 4.3 includes the approximated computational complexity of the square root and UD partial-update formulations. The costs of executing a conventional UD filter and its partial-update version, are shown in Table 4.4. The required extra effort to perform the partial update is also included in this table.

Table 4.3 shows that the UD partial-update formulation is more efficient than the square root formulation, requiring roughly $0.5n^3$ fewer operations. This computational savings, along with the numerical stability, make the UD formulation generally preferred over factorized formulation alternatives. Regarding the comparison between the conventional UD filter and its partial-update

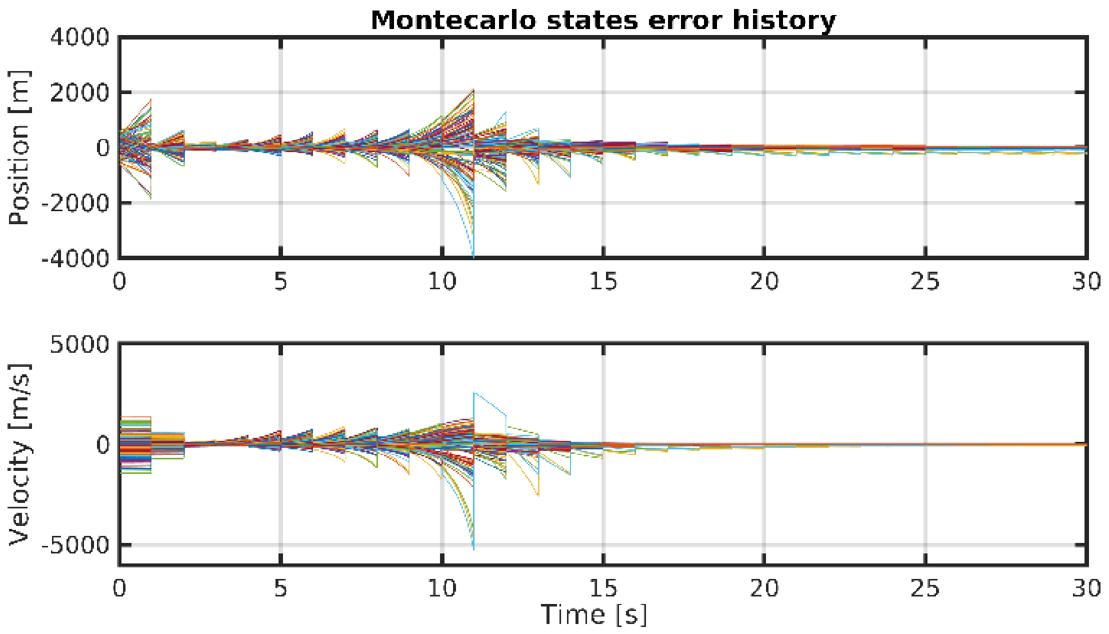


Figure 4.5: Monte Carlo runs for the partial-Update EKF and UD partial-update EKF with $\boldsymbol{\beta} = \begin{bmatrix} 0.9 & 0.9 & 0.75 \end{bmatrix}$ resulting in 90%, 90%, and 75% updates respectively to the position, velocity, and ballistic coefficient states.

Table 4.3: Square root and UD partial-update required flops for propagation and update steps combined.

Filter	Flops (multiply, divide and square root)
Square root partial-update	$2.5n^3 + (q + 7.5)n^2 + (2_{\surd} + 6)n + 2_{\surd} + 1$
UD partial-update	$2n^3 + (q + 4)n^2 + (q + 1)n + 2$

formulation, the flop count (multiplication and division) shows that the partial-update version of the filter requires some extra effort which is roughly equivalent to the product of a full $(n \times n)$ matrix times a $(n \times n)$ triangular matrix. Although it may not be significant for small to medium-size systems, it may be considerable for large systems. However, that is the price of incorporating the partial-update benefits and gaining robustness against system nonlinearity and uncertainty.

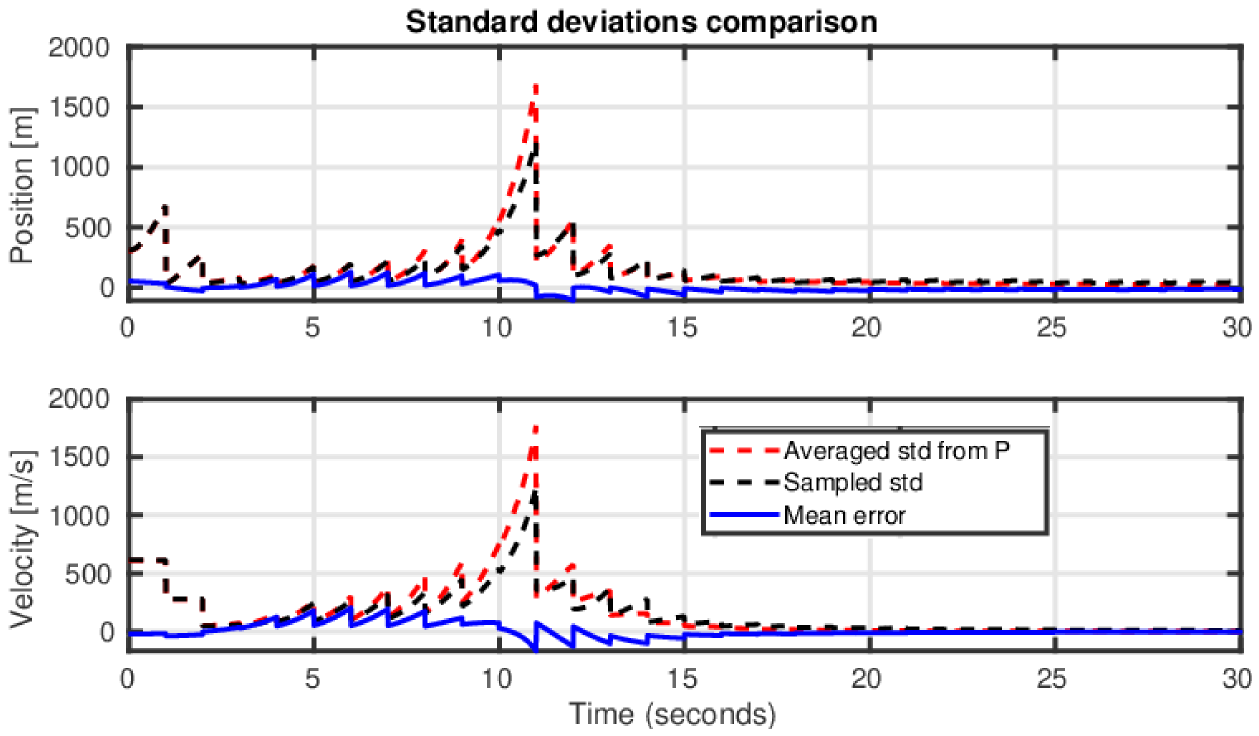


Figure 4.6: Monte Carlo runs for the partial-Update EKF and UD partial-update EKF with $\boldsymbol{\beta} = \begin{bmatrix} 0.9 & 0.9 & 0.75 \end{bmatrix}$ resulting in 90%, 90%, and 75% updates respectively to the position, velocity, and ballistic coefficient states.

Table 4.4: Conventional UD and UD partial-update required flops for one scalar measurement update.

Process	Flops (multiply, divide and square root)
Conventional UD Kalman update	$1.5n^2 + 1.5n$
UD partial-update Kalman update	$0.5n^3 + 3.5n^2 + n + 2$
UD partial-update extra cost	$0.5n^3 + 2n^2 - 0.5n + 2$

4.7.1 IMU-camera example

The IMU-camera calibration problem introduced in the previous chapter was also implemented in UD form. The simulations, as for the re-entry body problem, since they were performed for exactly the same scenario as before, show complete agreement with the square root partial-update filter, as expected. The plots for these runs (UD partial-update filter and conventional EKF) are not shown as they match the behavior previously presented.

4.7.2 Summary

The UD partial-update filter was shown to be more efficient than the square root formulation while retaining the numerical robustness properties (since it does not directly operate on the covariance matrix). Similar to the conventional UD filter, the partial-update version heavily relies on the structure of the involved matrices to lower the number of operations to execute a measurement update. However, alternative algorithms may provide even better efficiency for the UD partial-update filter. In terms of consistency and general behavior, since the UD filter is not different from the square root formulation, no changes were expected; the Monte Carlo and single runs corroborated this.

It is important to note that although the UD and Potter formulations are not the only factorized formulations for the Kalman filter, they are taken as the base of the developments presented in this research because they have been successfully and widely applied. Furthermore, since these formulations are among the most fundamental factorized filters, virtually any available extensions that have been applied to square root or UD filters can be incorporated into this work. Although the Carlson algorithm [77] may be very similar in performance to the UD filter and could have been selected as a means to increase efficiency, the UD filter was preferred as it avoids the computation of square roots and is widely used.

Finally, although the square root filter may not be as attractive as the UD formulation for implementation, it is not all in vain since, in any case, the square root filter can provide a way to corroborate UD filter estimates (alternative factorized filter). Even though the UD formulation can seem complicated at first glance, it only requires the extra coding of the Weighted-Modified Gram-Schmidt routine and the UD decomposition. Once these algorithms are available, the implementation is straightforward, as seen in Table 4.1. If the filtering engineer is dealing with correlated measurements and the desire is to process measurements sequentially, the UD based decorrelation algorithm can be executed (the corresponding cost needs to be considered in the total computational complexity of the filter). If the computation complexity is not a concern for any reason, the UD partial-update formulation presented in this chapter can also directly use vector measurements.

5. DYNAMIC PARTIAL-UPDATE KALMAN FILTER

5.1 Motivation

The partial-update formulation has been shown to improve the robustness and overall performance of the underlying filter. To achieve favorable results when using the partial-update method, however, *appropriate* selection of the update percentages, β, is needed. Even though the partial-update weights can be selected in a number of different manners, the $\beta's$ selection could be put into two possibilities: Static partial-update weight and dynamic partial-update weight selection. The numerical examples from the previous chapters, clearly, are cases with static $\beta's$.

For a static weight selection, whereas it may not be difficult to adjust the β values to obtain a converging filter in general, it certainly requires some experimentation with the system in question. Overall, when selecting the β values, it is sought to balance the negative impact of certain states by limiting update corrections while using most of the available information, and simultaneously, allowing enough state update to compensate for process noise (if present). An informal technique that has been useful for static weight selection has been based on state dynamics speed, assigning small weights to slowly changing states and larger weights to faster states. In conjunction with this idea, the weights may also be based on how "close" to an observation certain states are situated: Directly measured states are almost or fully updated whereas states updated through a long chain of cross-correlation terms are not updated or slightly updated. Nevertheless, even with these ad-hoc rules, the question of how much to slightly or partially update mean, remains. And while static β weights can suffice, filter estimates can certainly be further improved with online selected β weights, as shown with the proposed techniques in this chapter.

5.2 Dynamic partial-update weights

The idea for dynamic weight selection is to provide the partial-update filter (but in general, a Kalman filter) with information that can be used to decide, in a commensurate way, how much of the nominal update should be used. Specifically, the proposed methods use a system's nonlinearity metric to inform the filter on what weights values can be appropriate at a specific update step, such that it can take full advantage of the incoming measurement when possible but be “a careful updater” when required. Following this paradigm, two methods are presented next. Overall, both methods were shown to be of higher capacities than the static partial-update when handling high nonlinearities and uncertainties.

5.3 Nonlinearity-aware based method

For vast low-uncertainty applications, the tolerance of the EKF against slight mismodelling can be enough to prevent the estimator from failing. However, since the conventional EKF retains only the first-order term of the Taylor series expansion of the models in its formulation, if higher-order effects have a significant influence to produce an important mismatch between first order and system models, filter divergence can be originated. After all, the greater the model’s mismatch, the more sub-optimal the filter becomes (propagation and update correction would be carried out for a more distinct system). Based on this fact, the idea of trusting (using) the computed update more when the modeling mismatch is small, and trusting the update less for a significant mismatch, is proposed as a selection method of the partial-update percentages. More specifically, the proposed method uses the ratio of second-order to first-order terms of the Kalman update step equations to inform about how *reliable* the EKF equations can be at a given time step so that the filter can decide if it is "safe" to fully update, or partially update, or just *consider* states. Stated differently, in the case of *insignificant*, high, and very high second-order effects, the filter performs a full, partial, or a consider update, respectively. The next subsection provides the mathematical details on this nonlinearity-based β selection method, refereed from now on, as Dynamic nonlinearity-aware partial-update or DNL for short.

5.3.1 Nonlinearity-aware partial-update

To begin, following the same notation for the Kalman filter framework recall the equations for the discrete second-order Kalman filter (EKF2) presented here without derivation [68]. The dynamics and uncertainty propagation equations are

$$\hat{\mathbf{x}}_k^- = f(\hat{\mathbf{x}}_{k-1}^-, \mathbf{u}_{k-1}, k-1) + \frac{1}{2}\sum_{i=1}^{n} \phi_i \operatorname{Tr}\left[\frac{\partial^2 f_i}{\partial x^2}\Big|_{\hat{\mathbf{x}}_{k-1}^+} \mathbf{P}_{k-1}^+ \right] , \tag{5.1}$$

$$\mathbf{P}_k^- = \mathbf{F}_{k-1}\mathbf{P}_{k-1}^+\mathbf{F}_{k-1}^{\mathrm{T}} + \mathbf{G}_{k-1}\mathbf{Q}_{k-1}\mathbf{G}_{k-1}^{\mathrm{T}} . \tag{5.2}$$

The measurement update equations are

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{K}_k \left[\tilde{\mathbf{y}}_k - h(\hat{\mathbf{x}}_k^- \quad k) \right] - \pi , \tag{5.3}$$

$$\pi = \frac{1}{2}\mathbf{K}_k \sum_{i=1}^{m} \phi_i \operatorname{Tr}\left[\mathbf{D}_{k,i}\mathbf{P}_k^- \right] , \tag{5.4}$$

$$\mathbf{D}_{k,i} \, , = \frac{\partial^2 h_i(\mathbf{x}_k, k)}{\partial x^2}\Big|_{\hat{\mathbf{x}}_k^-} , \tag{5.5}$$

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}_k^{\mathrm{T}}(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k)^{-1} , \tag{5.6}$$

$$\mathbf{P}_k^+ = (\mathbf{I} - \mathbf{K}_{\mathbf{k}}\mathbf{H}_k)\mathbf{P}_k^- . \tag{5.7}$$

Here, the process and measurement model Jacobian are defined as before, and ϕ_i is the single entry vector (with 1 at the i^{th} element) given by,

$$\phi_i^{\mathrm{T}} = \begin{bmatrix} 0 & 0 & \dots & 0 & \dots & 1 & \dots & 0 \end{bmatrix}^{\mathrm{T}} . \tag{5.8}$$

Next, consider the measurement update from Equation (5.3) and the vector $\boldsymbol{\pi}$ expressed together as

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{K}_k \left[\tilde{\mathbf{y}}_k - h(\hat{\mathbf{x}}_k^-, k) \right] - \frac{1}{2}\mathbf{K}_k \sum_{i=1}^{m} \phi_i \operatorname{Tr}\left[\mathbf{D}_{k,i}\mathbf{P}_k^- \right] . \tag{5.9}$$

Further, let the prior state, $\hat{\mathbf{x}}_k^-$, as defined in Equation (5.1) be substituted into the previous equation to form

$$\begin{aligned}\hat{\mathbf{x}}_k^+ = f(\hat{\mathbf{x}}_{k-1}^-, \mathbf{u}_{k-1}, k-1) + \frac{1}{2}\sum_{i=1}^{n}\phi_i \operatorname{Tr}\left[\frac{\partial^2 f_i}{\partial x^2}\bigg|_{\hat{\mathbf{x}}_{k-1}^+}\mathbf{P}_{k-1}^+\right] + \mathbf{K}_k\left[\tilde{\mathbf{y}}_k - h(\hat{\mathbf{x}}_k^-, k)\right] \\ - \frac{1}{2}\mathbf{K}_k\sum_{i=1}^{m}\phi_i \operatorname{Tr}\left[\mathbf{D}_{k,i}\mathbf{P}_k^-\right] .\end{aligned} \tag{5.10}$$

By reorganizing the terms the posterior estimate can be written as,

$$\hat{\mathbf{x}}_k^+ = f(\hat{\mathbf{x}}_{k-1}^-, \mathbf{u}_{k-1}, k-1) + \mathbf{K}_k\left[\tilde{\mathbf{y}}_k - h(\hat{\mathbf{x}}_k^-, k)\right] + \mathbf{Y} , \tag{5.11}$$

where

$$\mathbf{Y} = \frac{1}{2}\left\{\sum_{i=1}^{n}\phi_i \operatorname{Tr}\left[\frac{\partial^2 f_i}{\partial x^2}\bigg|_{\hat{\mathbf{x}}_{k-1}^+}\mathbf{P}_{k-1}^+\right] - \mathbf{K}_k\sum_{i=1}^{m}\phi_i \operatorname{Tr}\left[\mathbf{D}_{k,i}\mathbf{P}_k^-\right]\right\} . \tag{5.12}$$

Recalling the partial-update expression for the states,

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + (\mathbf{I} - \mathbf{\Gamma})\mathbf{K}_k(\tilde{\mathbf{y}}_k - \hat{\mathbf{y}}_k) , \tag{5.13}$$

and expressing it in terms of the function dynamics, measurement function (for the same assumed system in the EKF2) and expanding it, leads to

$$\begin{aligned}\hat{\mathbf{x}}_k^+ &= \hat{\mathbf{x}}_k^- + (\mathbf{I} - \mathbf{\Gamma})\mathbf{K}_k(\tilde{\mathbf{y}}_k - \hat{\mathbf{y}}_k) && (5.14)\\ &= \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\tilde{\mathbf{y}}_k - \hat{\mathbf{y}}_k) - \mathbf{\Gamma}\mathbf{K}_k(\tilde{\mathbf{y}}_k - \hat{\mathbf{y}}_k) && (5.15)\\ &= f(\hat{\mathbf{x}}_{k-1}^-, \mathbf{u}_{k-1}, k-1) + \mathbf{K}_k(\tilde{\mathbf{y}}_k - \hat{\mathbf{y}}_k) - \mathbf{\Gamma}\mathbf{K}_k(\tilde{\mathbf{y}}_k - \hat{\mathbf{y}}_k) && (5.16)\\ &= f(\hat{\mathbf{x}}_{k-1}^-, \mathbf{u}_{k-1}, k-1) + \mathbf{K}_k\left[\tilde{\mathbf{y}}_k - h(\hat{\mathbf{x}}_k^-, k)\right] - \mathbf{\Gamma}\mathbf{K}_k\left[\tilde{\mathbf{y}}_k - h(\hat{\mathbf{x}}_k^-, k)\right] . && (5.17)\end{aligned}$$

Interestingly, a direct term-by-term comparison of the partial-update expression from Equation (5.17) and Equation (5.11), reveals that the term with the partial-update weights,

$$-\mathbf{\Gamma Z} := -\mathbf{\Gamma K}_k \left[\tilde{\mathbf{y}}_k - h(\hat{\mathbf{x}}_k^-, k) \right] , \tag{5.18}$$

can be directly related to second-order terms of the EKF2. That is,

$$-\mathbf{\Gamma K}_k \left[\tilde{\mathbf{y}}_k - h(\hat{\mathbf{x}}_k^-, k) \right] \propto \frac{1}{2} \left\{ \sum_{i=1}^{n} \phi_i \operatorname{Tr} \left[\left. \frac{\partial^2 f_i}{\partial x^2} \right|_{\hat{\mathbf{x}}_{k-1}^+} \mathbf{P}_{k-1}^+ \right] - \mathbf{K}_k \sum_{i=1}^{m} \phi_i \operatorname{Tr} \left[\mathbf{D}_{k,i} \mathbf{P}_k^- \right] \right\} , \tag{5.19}$$

or

$$-\mathbf{\Gamma K}_k \left[\tilde{\mathbf{y}}_k - h(\hat{\mathbf{x}}_k^-, k) \right] \propto \mathbf{Y} . \tag{5.20}$$

Further, Equation (5.20) follows the previously discussed idea of selecting $\gamma_i \in [0, 1]$ values according to the second-order terms influence, since it suggests that:

- $\mathbf{\Gamma}$ should be set with high values if the second-order effects, $\mathbf{Y}$, are large.
- If $\mathbf{Y}$ is small, $\mathbf{\Gamma}$ is to be set with small values.

Noticing that both left and right term of expression (5.19) are $n \times 1$ vectors, individual relationships between second-order terms and the j^{th} partial-update weights can be established as,

$$\mathbf{\Gamma}_{jj} \propto \frac{\frac{1}{2} \left\{ \sum_{i=1}^{n} \phi_i \operatorname{Tr} \left[\left. \frac{\partial^2 f_i}{\partial x^2} \right|_{\hat{\mathbf{x}}_{k-1}^+} \mathbf{P}_{k-1}^+ \right] - \mathbf{K}_k \sum_{i=1}^{m} \phi_i \operatorname{Tr} \left[\mathbf{D}_{k,i} \mathbf{P}_k^- \right] \right\}_j}{-\mathbf{K}_k \left[\tilde{\mathbf{y}}_k - h(\hat{\mathbf{x}}_k^-, k) \right]_j} , \tag{5.21}$$

or alternatively,

$$\mathbf{\Gamma}_{jj} \propto \frac{\mathbf{Y}_j}{\mathbf{Z}_j} . \tag{5.22}$$

To the end of keeping the γ values within the appropriate bounds, the ratio $\frac{\mathbf{Y}_j}{\mathbf{Z}_j}$ is saturated to a maximum value of 1. In theory, $\mathbf{Z}_j$ can be equal to zero when the measurement is equal to the expected measurement value, or when the i^{th} state is known with no uncertainty. But in any case,

$\mathbf{\Gamma}$ will take its maximum value of 1. Since only the magnitude of second-order terms is considered, absolute values are taken and an scale factor f_r is introduced. This gives rise to the equation

$$\mathbf{\Gamma}_{jj} = f_r \frac{|\mathbf{Y}_j|}{|\mathbf{Z}_j|} \, , \tag{5.23}$$

which in terms of $\boldsymbol{\beta}$ reads

$$\boldsymbol{\beta}_{jj} = 1 - \frac{|\mathbf{Y}_j|}{|\mathbf{Z}_j|} \, . \tag{5.24}$$

From the expressions found through the construction of this nonlinearity-aware method, it is important to highlight the following. First, no assumption on the organization of the states within the filter has been made during the derivation of Equation (5.23) such that there is no partition to separate considered states from core states. Thus, Equation (5.23) indicates that the partial-update technique could be interpreted as a technique that, to some degree, compensates for second-order effects. Further, the partial-update technique, when using this nonlinearity-aware method for weights selection, can be seen, at least in part, mimicking second-order effects via the term $\mathbf{\Gamma}\mathbf{K}_k \left[\tilde{\mathbf{y}}_k - h(\hat{\mathbf{x}}_k^-, k)\right]$ to improve filter behavior. Second, since the motivation behind this weight selection approach is to use the degree of local nonlinearity as an indication of the validity of the linearization, the signs of the quantities are ignored, and the metric is based solely on nonlinear effects magnitudes. Finally, the scale factor $\mathbf{f}_{r,j}$ is a design variable and is used to make the nonlinearity metric adjustable to the problem in question. As per experiments done with this β selection method, an adaptive scale factor $f_{r,i}$ (for the i^{th} partially updated state) that involves the measurement residual and the state uncertainty state covariance was found to be convenient. Specifically, the ratio of the measurement residual covariance trace to the measurement noise covariance trace was used. Mathematically,

$$f_{r,i} = \frac{\sigma_{k,i}}{\sigma_{o,i}} \frac{\mathrm{Tr}(\mathbf{H}_k \mathbf{P}_k \mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k)}{\mathrm{Tr}(\mathbf{R}_k)} \, . \tag{5.25}$$

The reason for including the ratio of the traces is to weight the second-order terms-effect commensurate to model mismatch. Moreover, the current standard deviation (which appears normalized by the initial value) is included to increase the impact of residual errors when the system uncertainty is high. Additionally, the ratio of the traces varies from infinity (depending on the uncertainty value) to 1, which provides an adequate scale for the filter steady state. In the case of the residual covariance being large, it will make the filter lower the magnitude of the corresponding update.

Although the computation of second-order terms is required to implement this method, they are only used to approximate a nonlinearity measure to assist with weight selection. In a worst-case scenario, an incorrect second-order term can lead to the temporary use of a conventional EKF or a Schmidt filter, which is not critical if linearization errors are within the bounds of what the filter could support for the next time updates. However, this is system dependent, and safety measures may be needed to prevent filter failure. For systems with constant or slowly varying parameters, for example, although conservative, the use of a Schmidt or partial-update could be safer than a full update, but again, this is system dependent.

In the next sections numerical simulations that exercise the nonlinearity-aware partial-update method (DNL) are presented.

5.3.2 Numerical example

Again, consider the now familiar falling body that is re-entering Earth. As before, the motion is considered to be constrained to a vertical line, and the filter is used to estimate the body altitude, velocity, and the body's ballistic parameter. If the filter parameters are maintained exactly the same as in previous sections, a typical run is as the one from Figure 5.1. The β partial-update weight history is presented in Figure 5.2. From the states plot for the single run, it can be observed that the dynamic partial-update performs at the level of the static partial-update; however, the DNL method covariance is tighter. The reason for the lower covariance is that overall, the DNL method used more of the nominal update than the static partial-update set with $\boldsymbol{\beta} = \begin{bmatrix} 0.9 & 0.9 & 0.75 \end{bmatrix}$. The DNL method, however, does not require experimentation for tuning or selecting initial β. Effectively, the DNL method attempts to make use of most of the information while using state and residual covariances to scale the update percentage commensurate to second-order terms effects.

In Figure 5.2, it is important to note that the filter weight adaptation is capturing two key instants and changing β as to maintain consistent estimates. Firstly, at time $t = 1s$, when the first measurement is assimilated, the filter acts with a considered step since large state uncertainty is present, and the measurement residual is large. Secondly, at time $t = 11s$ after the filter regains information (after losing observability due to range sensor and body alignment), the filter reduces the update since model mismatch, state uncertainty, and measurement residual increase. Nevertheless, the β history allowed estimates to remain within the appropriate bounds and let the filter converge. Although the static partial-update approach achieves comparable results to the dynamic method, a considerable amount of tuning and experimentation was required, whereas the DNL method is run without a trial and error process needed. Also, recall that the conventional EKF is divergent for the run shown in Figure 5.1; the plots are not included.

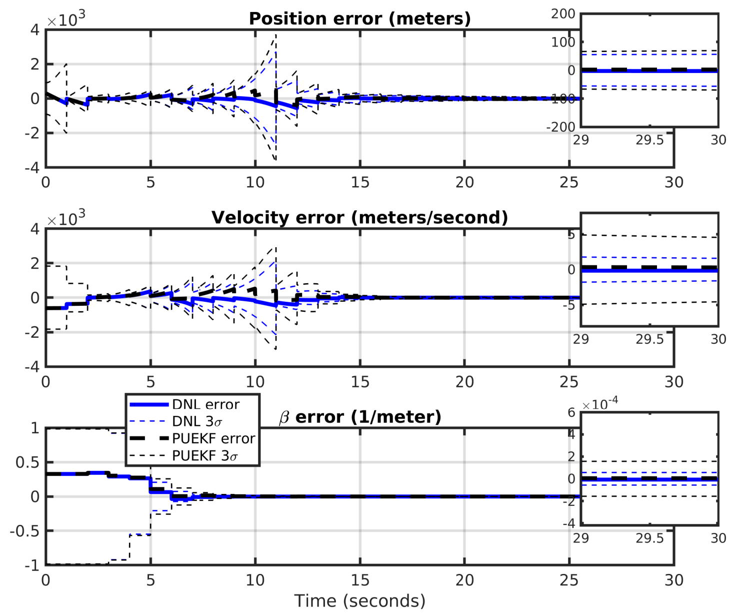


Figure 5.1: Dynamic nonlinearity-aware method (DNL) without previous tuning, static partial-update with $\boldsymbol{\beta} = \begin{bmatrix} 0.9 & 0.9 & 0.75 \end{bmatrix}$ and conventional EKF single run for body re-entry problem. Initial error within 1σ.

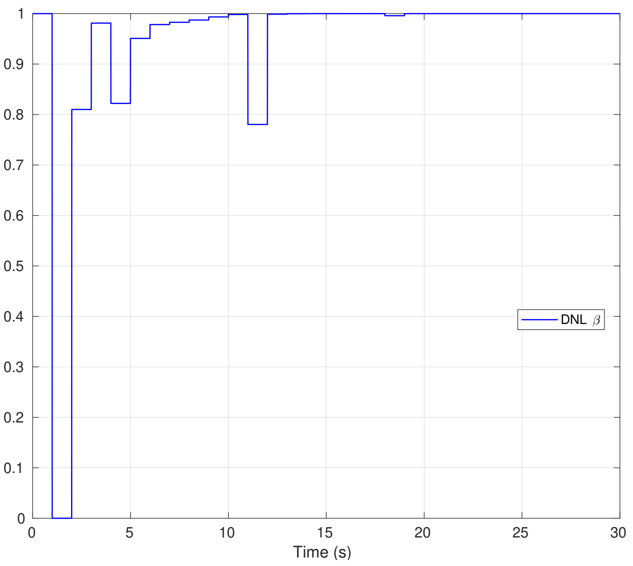


Figure 5.2: Dynamic nonlinearity-aware method (DNL) without previous tuning, static partial-update with $\boldsymbol{\beta} = \begin{bmatrix}0.9 & 0.9 & 0.75\end{bmatrix}$ and conventional EKF single run for body re-entry problem. Initial error within 1σ.

The results for a second run that is initialized with larger state errors are depicted in Figure 5.3. For this scenario, the filters are also observed to provide consistent estimates for both static and dynamic partial-update filters. The corresponding β history is shown in Figure 5.4. This profile, compared to the β history of Figure 5.2 is seen to undergo more aggressive changes due to the higher initial errors. Nonetheless, the filter can handle such a scenario and maintain the estimates within the 3σ bounds.

The DNL method for β selection, however, as any other filter has its limits, and it cannot provide infinite immunity to initial errors if no additional information is provided to the filter (see subsection 5.3.3). For the re-entry body problem, under the simulation parameters selected, the DNL method has been found to support initial errors up to the equivalent of 2 σ. However, the maximum error amount that can be supported is a function of system configuration, including initial uncertainties, initial conditions, measurement frequency, and models nonlinearities. Figure 5.5 shows the results of 100 runs for this problem for errors lower than 2 σ for position and velocity and for up to 3 σ for the ballistic parameter. Figure 5.6 displays the comparison between the estimated and sampled standard deviation. The purpose of this runs is to show that the DNL

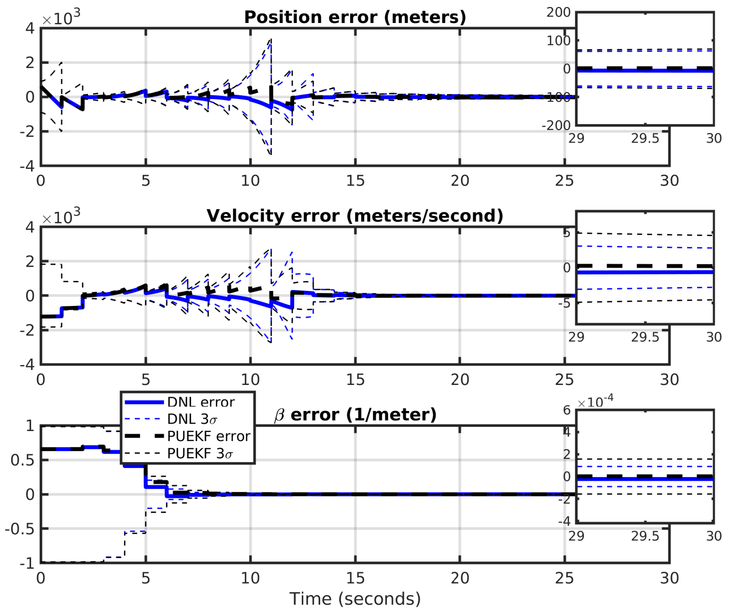


Figure 5.3: Dynamic nonlinearity-aware method (DNL) without previous tuning, static partial-update with $\boldsymbol{\beta} = \begin{bmatrix} 0.9 & 0.9 & 0.75 \end{bmatrix}$ and conventional EKF single run for body re-entry problem. Initial error within 2σ.

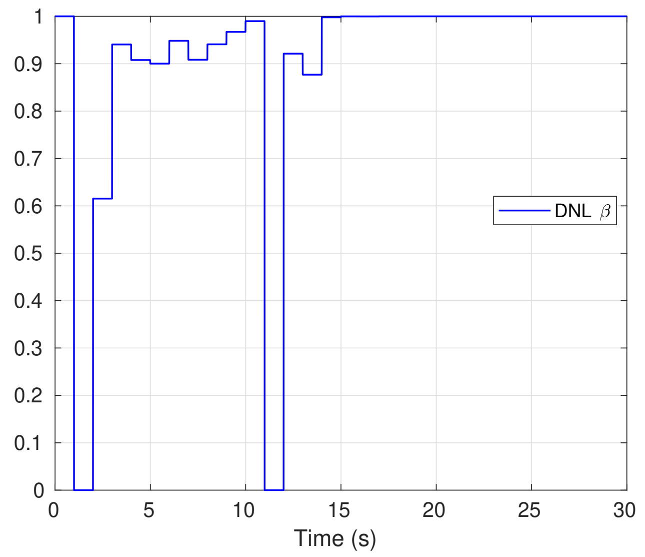


Figure 5.4: Dynamic nonlinearity-aware method (DNL) without previous tuning, static partial-update with $\boldsymbol{\beta} = \begin{bmatrix} 0.9 & 0.9 & 0.75 \end{bmatrix}$ and conventional EKF single run for body re-entry problem. Initial error within 2σ.

method performs well under the given initial uncertainties, not just for a single run. From Figure 5.5, it can be observed that the DNL method is slightly overconfident, which can be due to the filter updates overreaction due to nonlinearities being involved causing the uncertainty to be slightly tighter than it needs to be. Nonetheless, its superior consistency over the EKF is clear since no divergent cases are seen.

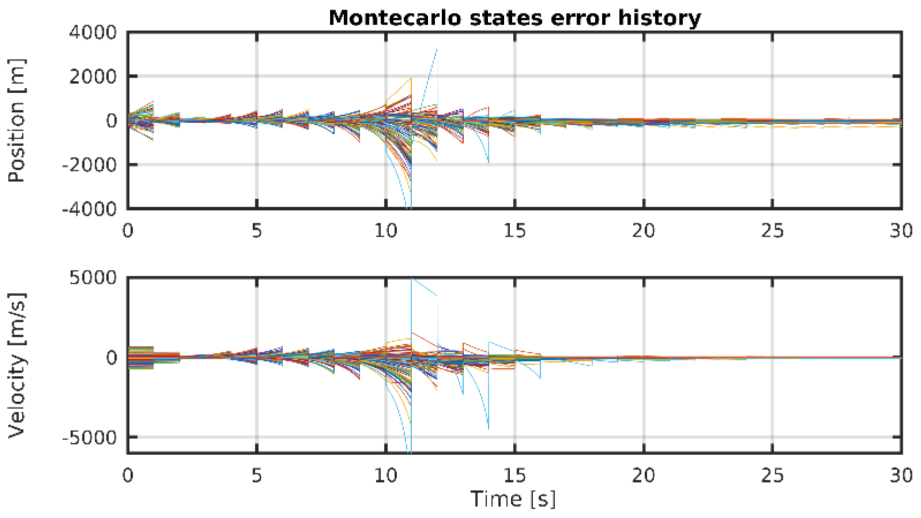


Figure 5.5: Monte Carlo runs state histories for the DNL partial-update.

5.3.3 Pre-tuned partial-update weights as a baseline for DNL method

Although, advantageously, the DNL method can provide an operational filter for cases where no prior information on functional β percentages are available (especially for large systems), the DNL method is also able to accommodate known β values. To incorporate known β values, one shifts the nominal update baseline of the DNL method to be the β values, rather than a full update (or $\beta = 1$). This hybrid method, although effective, it requires tuning and it is over-conservative. In general, dynamic methods alone are recommended and sufficient. The discussion on this hybrid method is presented for the sake of completeness.

To exercise this concept, the same re-entry body problem is employed and the β value from

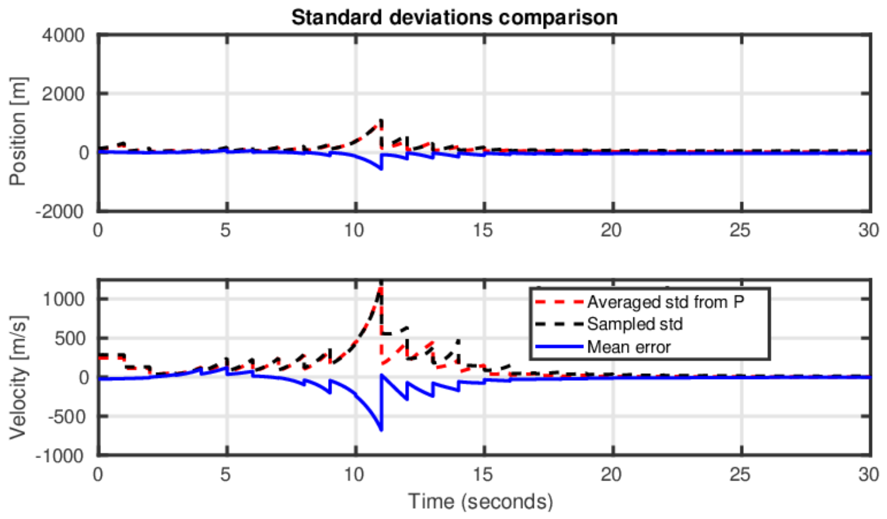


Figure 5.6: Averaged and sampled standard deviation from 100 Monte Carlo runs for the DNL partial-update. The mean error is also shown. Full update is used as nominal value.

the original example (1 σ initial error and same filter parameters) is taken as the weight baseline ($\boldsymbol{\beta} = \begin{bmatrix} 0.9 & 0.9 & 0.75 \end{bmatrix}$). Recall that the dynamic weight selection is only performed for the ballistic parameter. Figure 5.8, illustrates the filter estimates that employs the DNL method with tuned betas and the PU method. From such plots, it is apparent that the DNL has not done much to improve the system behavior, yet it has practically accomplished the same results as the static partial-update but with higher uncertainty. The increase of uncertainty is due to, on average, lower updates as observed from the β percentage profile depicted in Figure 5.9. Nonetheless, this run intends to demonstrate the functionality of the dynamic nonlinearity-aware filter using previously selected (tuned) weights βs. As a reference only, the conventional Kalman filter was run and plotted in Figure 5.7 along with the partial-update methods with the only objective of recalling that it is an inconsistent filter already when the initial condition errors of 1 σ. It is important to note from Figure 5.9, that the β profile has overall changed with respect to the single DNL approach, but more importantly, a decrease in its fluctuation is observed. This is reasonable considering that a partial-update is applied throughout the run. This filter, in contrast with the "pure" DNL filter, leaves most of the work to the pre-tuned weight, mostly being nonreactive at the beginning of the run and after the recovery of measurement information after time $t = 11s$. In other words, whereas the DNL method with baseline one seems to start acting early on to handle the initial drag perturbations to maintain consistent estimates, the policy of the DNL base-lined at $\beta = 0.75$ appears more reactive only to instants when the system nonlinearities can have a larger impact due to higher uncertainty.

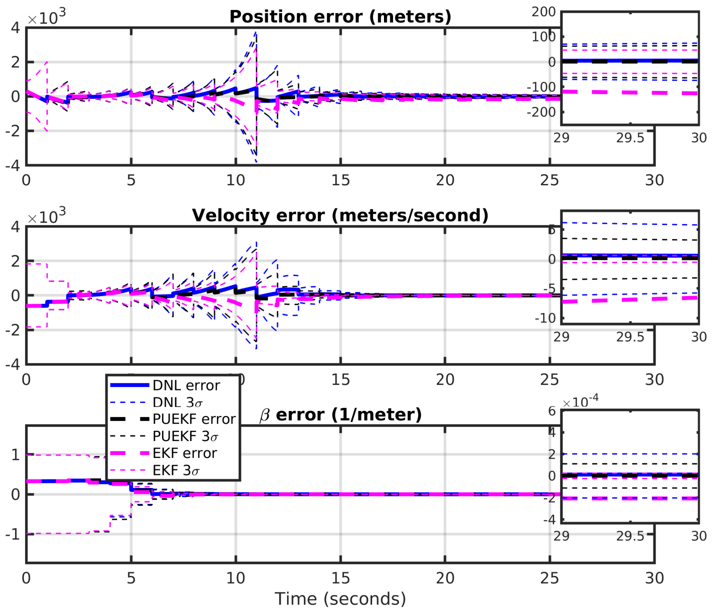


Figure 5.7: DNL method, static partial-update with $\boldsymbol{\beta} = [0.9, 0.9, 0.75]$, and conventional EKF single run for body re-entry problem. Initial error within 1σ.

Now, with the intent of stressing the DNL method further when it incorporates pre-tuned β values, two experiments are performed. First, the initial errors are increased to the equivalent of 3σ for every state. Second, the partial-update filters are required to use a higher percentage to attempt leveraging more measurement information. Recall that a full update causes inconsistent results even for 1 σ errors. Figure 5.10 depicts the results from the single run when a 3σ initial error is used. In addition, Figure 5.11 shows the corresponding history for β. Under this conditions, as per the estimation error plot from Figure 5.10, although slight discrepancy is observed between the dynamic and static filter, the results are overall the same as for 1σ errors from Figure 5.8 with

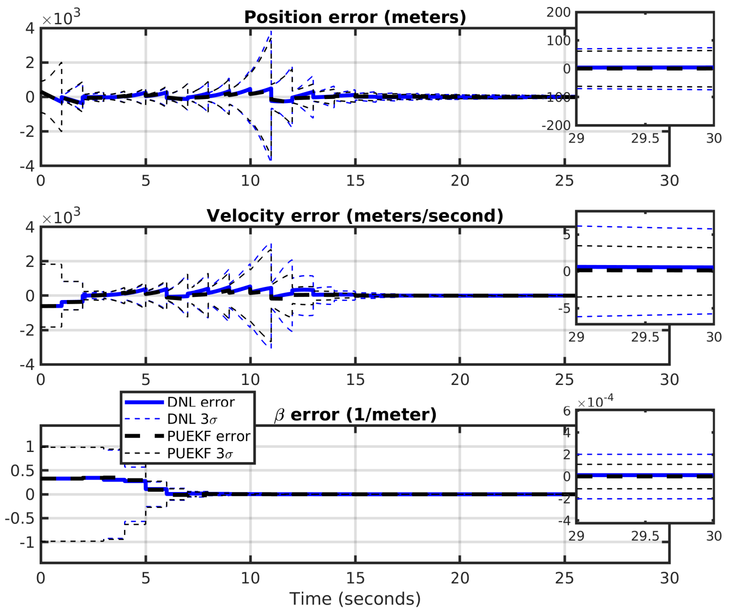


Figure 5.8: DNL method and static partial-update with $\boldsymbol{\beta} = [0.9, 0.9, 0.75]$ single run for body re-entry problem. Initial error within 1σ.

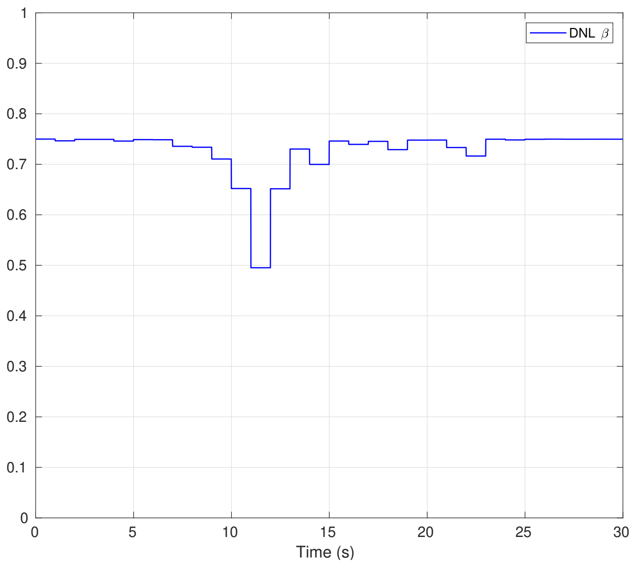


Figure 5.9: Dynamic nonlinearity-aware method (DNL) β history for the body re-entry problem. Initial error within 1σ.

the small difference that the ballistic parameter now appears barely less unbiased in favor to the DNL method. From the β history, it is seen that the dynamic filter required to use larger updates to compensate for the larger errors. The β profile also shows that it is still effectively reacting to high-order effects promptly.

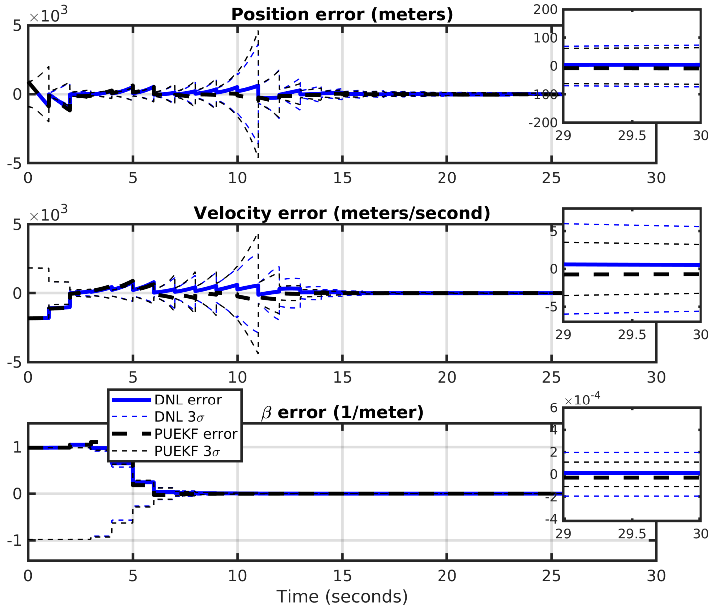


Figure 5.10: DNL method, static partial-update with $\boldsymbol{\beta} = [0.9, 0.9, 0.75]$ single run for body re-entry problem. 3σ initial errors.

An additional experiment that uses initial conditions far in the tail of the assumed Gaussian distribution (4σ) is performed. The results illustrated in Figure 5.12 and 5.13, clearly show inconsistency of the static partial-update for the experiment. This run shows the capability of the partial-update filter to deal with very high uncertainties if it is actively adapting the update percentages while having a pre-tuned β's as the baseline. For this scenario, the β history, although it displays similar behavior to the two previous experiments, for this run the DNL method is observed to have employed, overall, lower updates as it requires to perform a larger second-order effects compensation. It is important to mention that the DLN not only executed a minimum update precisely when

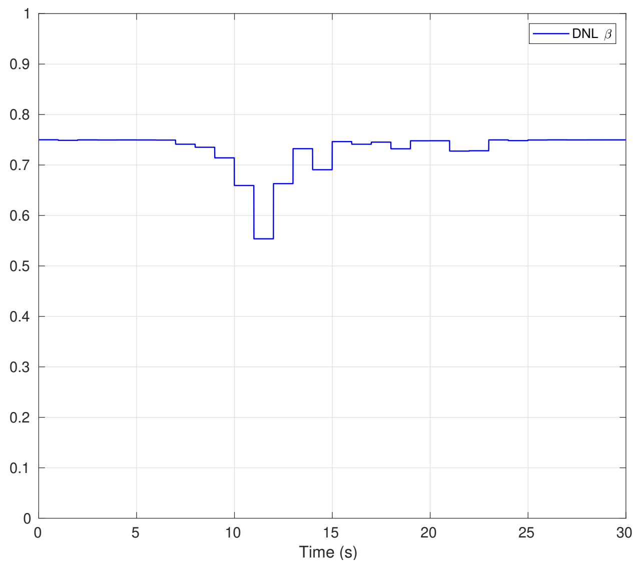


Figure 5.11: Dynamic nonlinearity-aware method (DNL) β history for the body re-entry problem. 3σ initial errors.

required at time $t = 11s$, but it also prevented filter overreactions due to linearization errors early on during the interval $t \in [5, 10]$. By no overreacting, the filter was able to recover and maintain consistent estimates as seen in Figure 5.15 (the zoomed-in version (in the interval $t \in [5, 15]$ s) of Figure 5.12). On the other hand, the second-order terms contributions are unknown to the static partial-update filter, which overshoots, and it eventually fails due to accumulated error mainly in the drag coefficient. By the time the range sensor becomes aligned (horizontally) with the position of the falling body at time $t = 10$, the dynamic partial-update filter is far in a better *position* than the static partial-update, and can handle the reacquisition at time $t = 11s$ and on, while the static partial-update errors are large enough such that the filter is unable to recover and eventually diverges.

The value of the partial-update weights were set to $\boldsymbol{\beta} = \begin{bmatrix} 0.9 & 0.9 & 0.75 \end{bmatrix}$ since by experimentation, they produced appropriate estimates for a wide variety of initial conditions (within the original 3 σ). Although the static partial-update may improve by refining the β values, the DNL method was generally found to be superior in robustness and consistency, especially for scenarios with large uncertainties and initial errors. Figure 5.16, as an example, shows a single run of the

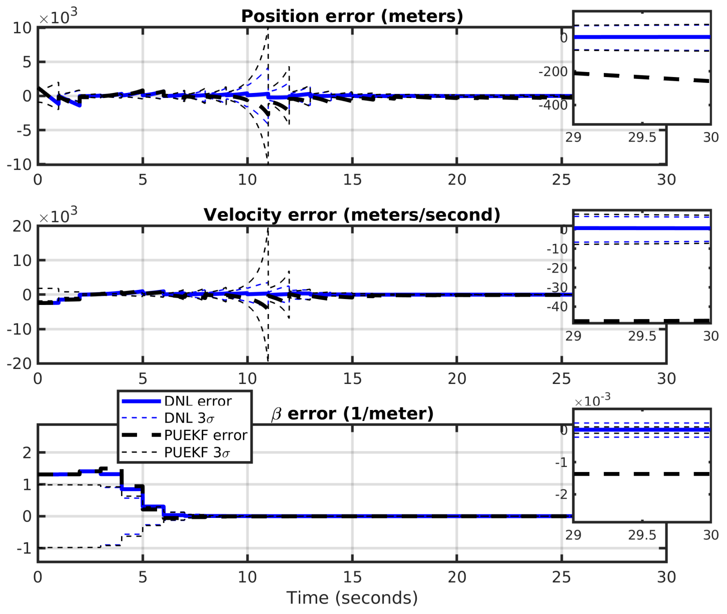


Figure 5.12: DNL method and static partial-update with $\boldsymbol{\beta} = [0.9, 0.9, 0.75]$ single run for body re-entry problem. 4σ initial errors.

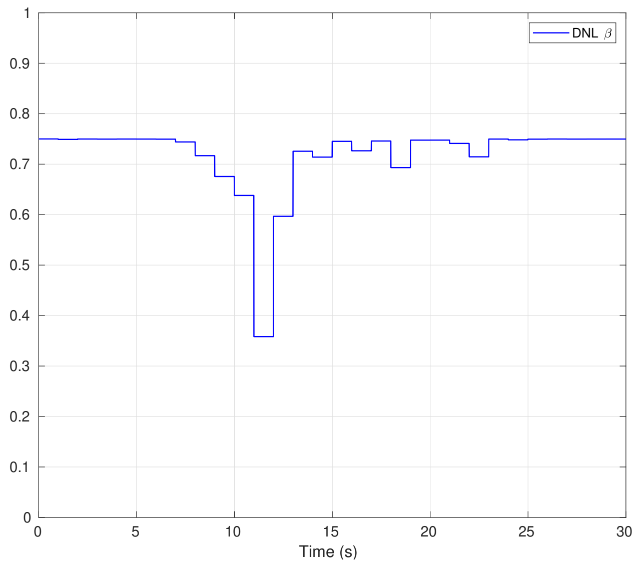


Figure 5.13: Dynamic nonlinearity-aware method (DNL) β history for the body re-entry problem. 4σ initial errors.

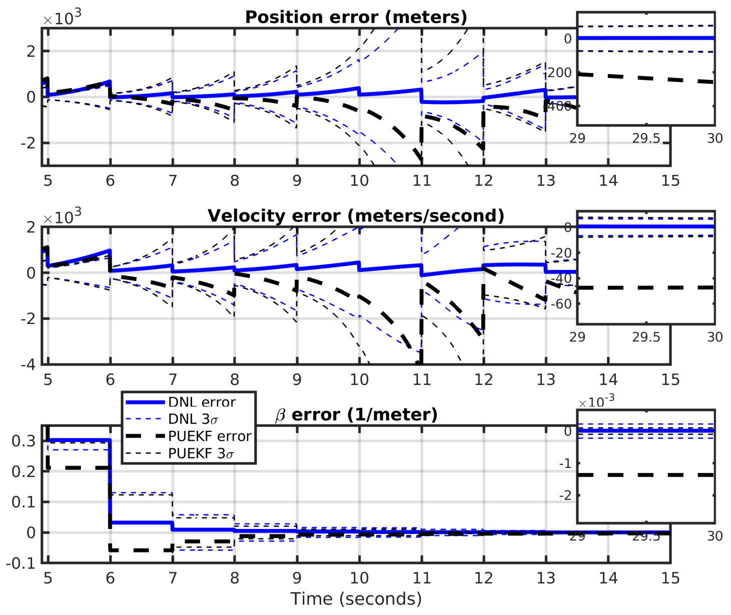


Figure 5.14: Zoomed-in view of dynamic nonlinearity-aware method (DNL), static partial-update with $\boldsymbol{\beta} = [0.9, 0.9, 0.75]$ single run for body re-entry problem. 4σ initial errors.

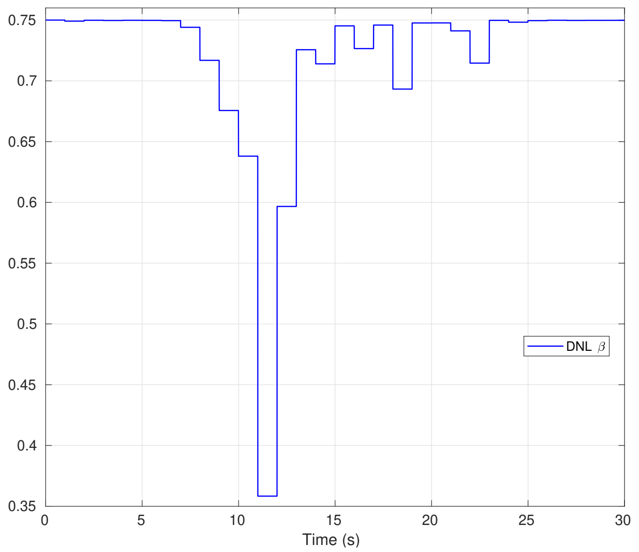


Figure 5.15: Zoomed-in view of dynamic nonlinearity-aware method (DNL) β history for the body re-entry problem. 4σ initial errors.

re-entry body problem where the initial errors are maintained at 3 σ levels but in contrast with previous runs, the initial uncertainty has been doubled for position and velocity, and the uncertainty on the ballistic parameter was augmented five times. For this run, the static partial-update which now employs $\boldsymbol{\beta} = \begin{bmatrix}1 & 1 & 0.75\end{bmatrix}$, is observed to be outside the 3σ bounds for all of the estimation errors, whereas the dynamic β although not fully reaching zero error, the state errors are still within proper bounds for the single run and errors are better than for the static partial-update case. The main observation here is that although the static partial-update filter is enabled to perform larger updates for this experiment to allow it to recover "faster" if no adaptation of β is performed, the filter eventually updates considerably when the model mismatch is important, and this leads to estimates inconsistency. Figure 5.17 illustrates the *value* of the adaptive method, as for the detected second-order terms the filter practically becomes a consider filter from time $t = 8s$ to $t = 17s$ to maintain an operational filter.

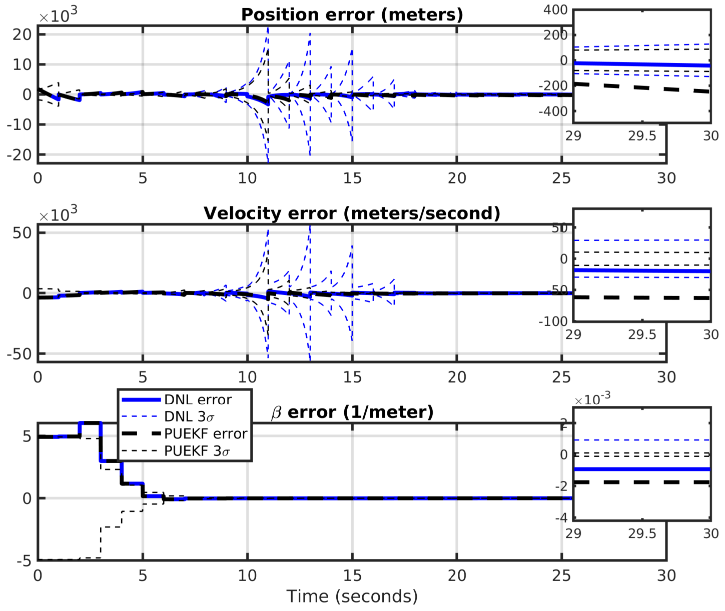


Figure 5.16: Dynamic nonlinearity-aware method (DNL), static partial-update with $\boldsymbol{\beta} = [1, 1, 0.75]$ single run for body re-entry problem subject to higher initial uncertainties and initial errors.

To the end of increasing the capabilities of the nonlinearity-aware method further, a second way

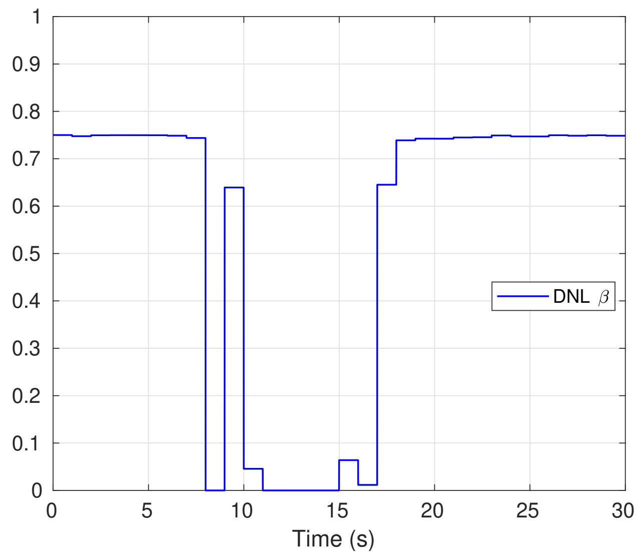


Figure 5.17: Dynamic nonlinearity-aware method (DNL) β history for single run for the body re-entry problem subject to higher initial uncertainties and initial errors.

for the β's selection is formulated. This alternative method, called the covariance-aware method, is based on monitoring second-order terms of the covariance update equation, but it follows the same idea as the nonlinearity-aware method. The covariance-aware method is presented next.

5.4 Covariance-aware based method

This section presents an alternative way of selecting the partial-update weights in an online fashion called Dynamic Covariance-aware partial-update, or DC for short. Paralleling the previous method that monitors second-order effects on the Kalman update terms, the method proposed in this section monitors second-order covariance terms. As before, the idea is to reduce the update magnitude when the ratio of high-order effects to first-order terms is important, and use more of the nominal update if second-order contributions are small. Although the construction of the expressions for selecting the partial-update weights is based on second-order terms appearing in a second-order hybrid Kalman filter, the dynamic covariance method works for the discrete filter as well. To obtain the expressions for the β selection using the covariance-aware method, consider the covariance measurement update expression for the second-order Kalman filter (EKF2),

$$\mathbf{P}_k^+ = \mathbf{P}_k^- - \mathbf{P}_k^-\mathbf{H}_k^{\mathrm{T}}(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k + \mathbf{\Lambda}_k)^{-1}\mathbf{H}_k\mathbf{P}_k^- \ , \tag{5.26}$$

where

$$\Lambda_k(i,j) = \frac{1}{2}\operatorname{Tr}(\mathbf{D}_{k,i}\mathbf{P}_k^-\mathbf{D}_{k,j}\mathbf{P}_k^-) \ , \tag{5.27}$$

and

$$\mathbf{D}_{k,i} = \frac{\partial^2 h_i(\mathbf{x}_k, k)}{\partial x^2}\bigg|_{\hat{\mathbf{x}}_k^-} \ . \tag{5.28}$$

Additionally, consider the covariance partial-update expression

$${\mathbf{P}_k}^{++} = \mathbf{P}_k^+ + \mathbf{\Gamma}\mathbf{P}_k^-\mathbf{H}_k^{\mathrm{T}}(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k)^{-1}\mathbf{H}_k\mathbf{P}_k^-\mathbf{\Gamma} \ . \tag{5.29}$$

This equation can be alternatively written in terms of the prior state covariance as,

$${\mathbf{P}_k}^{++} = \mathbf{P}_k^- - \mathbf{K}_k\mathbf{H}_k\mathbf{P}_k^- + \mathbf{\Gamma}\mathbf{P}_k^-\mathbf{H}_k^{\mathrm{T}}(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k)^{-1}\mathbf{H}_k\mathbf{P}_k^-\mathbf{\Gamma} \ , \tag{5.30}$$

and replacing the Kalman gain by $\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}_k(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^{\mathrm{T}} + \mathbf{R})^{-1}$ leads to

$${\mathbf{P}_k}^{++} = \mathbf{P}_k^- - \mathbf{P}_k^-\mathbf{H}_k(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^{\mathrm{T}} + \mathbf{R})^{-1}\mathbf{H}_k\mathbf{P}_k^- + \mathbf{\Gamma}\mathbf{P}_k^-\mathbf{H}_k^{\mathrm{T}}(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k)^{-1}\mathbf{H}_k\mathbf{P}_k^-\mathbf{\Gamma} \ . \tag{5.31}$$

Next, for the purposes of comparing the partial-update terms against second order uncertainty terms, the $\mathbf{\Lambda}$ term of Equation (5.26) is first extracted from the parenthetical to produce the residual covariance term $(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k)^{-1}$. This is accomplished by applying the matrix inversion lemma. That is,

$$\begin{aligned}
&(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k + \mathbf{\Lambda}_k)^{-1} = \\
&(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k)^{-1} - \\
&(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k)^{-1}\mathbf{\Lambda}_k[(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k)^{-1}\mathbf{\Lambda}_k + \mathbf{I}]^{-1}(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k)^{-1} \ .
\end{aligned} \tag{5.32}$$

Substituting this expression into the EKF2 update covariance of Equation (5.26), results in

$$\begin{aligned}
&\mathbf{P}_k{}^{++} = \\
&\mathbf{P}_k^- - \mathbf{P}_k^-\mathbf{H}_k(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^{\mathrm{T}} + \mathbf{R})^{-1}\mathbf{H}_k\mathbf{P}_k^- + \\
&\mathbf{P}_k^-\mathbf{H}_k^{\mathrm{T}}(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k)^{-1}\mathbf{\Lambda}_k[(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k)^{-1}\mathbf{\Lambda}_k + \mathbf{I}]^{-1}(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k)^{-1}\mathbf{H}_k\mathbf{P}^- \; .
\end{aligned} \tag{5.33}$$

By doing a term-by-term comparison of the partial-update expression of Equation (5.31), and the EKF2 update for the error state covariance of Equation (5.33), the following relationship between second-order covariance effects and partial-update terms can be established,

$$\begin{aligned}
&\mathbf{\Gamma}\mathbf{P}_k^-\mathbf{H}_k^{\mathrm{T}}(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k)^{-1}\mathbf{H}_k\mathbf{P}_k^-\mathbf{\Gamma} \sim \\
&\mathbf{P}_k^-\mathbf{H}_k^{\mathrm{T}}(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k)^{-1}\mathbf{\Lambda}_k[(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k)^{-1}\mathbf{\Lambda}_k + \mathbf{I}]^{-1}(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k)^{-1}\mathbf{H}_k\mathbf{P}^- \; ,
\end{aligned} \tag{5.34}$$

where the symbol $\sim$, is to indicate that terms are related. Equation (5.34) can be compactly written as

$$\mathbf{\Gamma}\delta\mathbf{P}_k^-\mathbf{\Gamma} \sim \mathbf{K}_k\mathbf{\Lambda}_k[(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k)^{-1}\mathbf{\Lambda}_k + \mathbf{I}]^{-1}\mathbf{K}_k{}^{\mathrm{T}} \; , \tag{5.35}$$

where $\delta\mathbf{P}_k^- = \mathbf{P}_k^-\mathbf{H}_k^{\mathrm{T}}(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k)^{-1}\mathbf{H}_k\mathbf{P}_k^-$. To further simplify relation (5.35), the matrix on the right is condensed in just one matrix called $\mathbf{N}$, such that the relationship now reads

$$\mathbf{\Gamma}\delta\mathbf{P}_k^-\mathbf{\Gamma} \sim \mathbf{N}_k \; . \tag{5.36}$$

Based on the previous discussion, the desire is to select $\mathbf{\Gamma}_{jj}$ proportional to second-order effects and with this objective in mind, it is proposed to select the $\mathbf{\Gamma}_{jj}$ values by an straight element-by-element comparison of the diagonal elements of matrix $\mathbf{\Gamma}\delta\mathbf{P}_k^-\mathbf{\Gamma}$ and $\mathbf{N}$. This leads to the proportionality relationship

$$\delta\mathbf{P}_{jj}^-\gamma_j^2 \propto \mathbf{N}_{jj} \; , \tag{5.37}$$

or since $\delta\mathbf{P}^-$ and $\mathbf{N}$ are positive semi-definite,

$$\gamma_j \propto \sqrt{\frac{\mathbf{N}_{jj}}{\delta\mathbf{P}^-_{jj}}} \,. \tag{5.38}$$

Finally, similar to the DNL method, a scale factor f_c is introduced to account for measurement residual covariance effects as,

$$\mathbf{\Gamma}_{jj} = \gamma_j = f_c \sqrt{\frac{\mathbf{N}_{jj}}{\delta\mathbf{P}^-_{jj}}} \,, \tag{5.39}$$

or equivalently

$$\boldsymbol{\beta}_{jj} = \beta_j = 1 - f_c \sqrt{\frac{\mathbf{N}_{jj}}{\delta\mathbf{P}^-_{jj}}} \,. \tag{5.40}$$

The scale factor f_c used for the covariance-aware method is the same as the one used for the nonlinearity-aware method,

$$f_{c,i} = \frac{\sigma_{k,i}}{\sigma_{o,i}} \frac{\mathrm{Tr}(\mathbf{H}_k\mathbf{P}_k\mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k)}{\mathrm{Tr}(\mathbf{R}_k)} \,. \tag{5.41}$$

Similar to the DNL method, the value for γ_i is saturated at a maximum magnitude of one.

In the next section, results from numerical simulations that exercise the dynamic covariance-aware (DC) partial-update filter, are provided.

5.4.1 The re-entry falling body

For convenience, the covariance-aware method was also exercised with the falling body problem. As for the DNL method, two scenarios using the DC method were simulated. First, the filter is used without incorporating any pre-tuned β, and second, the filter is tested with a nominal baseline shifted to the already known and well-tuned β value. For the first case scenario, in general, it was observed that for low initial errors in a typical single run, the covariance-aware, the nonlinearity-aware, and static partial-update methods appear to have similar overall performance, as displayed in Figure 5.18. However, a closer examination of the absolute estimation error, shown in Figure 5.19, reveals that the DC method performs the best among the three filters (in terms of absolute error amount). In fact, the DC performs the best in general. An important observation for this run is that albeit the similarity of the partial-update weights histories (shown in Figure 5.20) for the dynamic methods, the state errors differences between the methods are significant. This difference in performance highlights the importance of an appropriate dynamic β's selection method, and it demonstrates that a varying β can be advantageous, even for relatively small initial errors. From Figure 5.21 is also clear that since the static partial-update filter is unaware of high-order effects, it is less effective if they become significantly large. For this run, this is apparent mostly for the position and velocity states at time $t = 7s$. Finally, it should be noted that due to the DC and DNL method having a nominal update of a hundred percent, the DC method is less conservative than the static partial-update method, but this also helped obtain a more accurate dynamic partial-update filter.

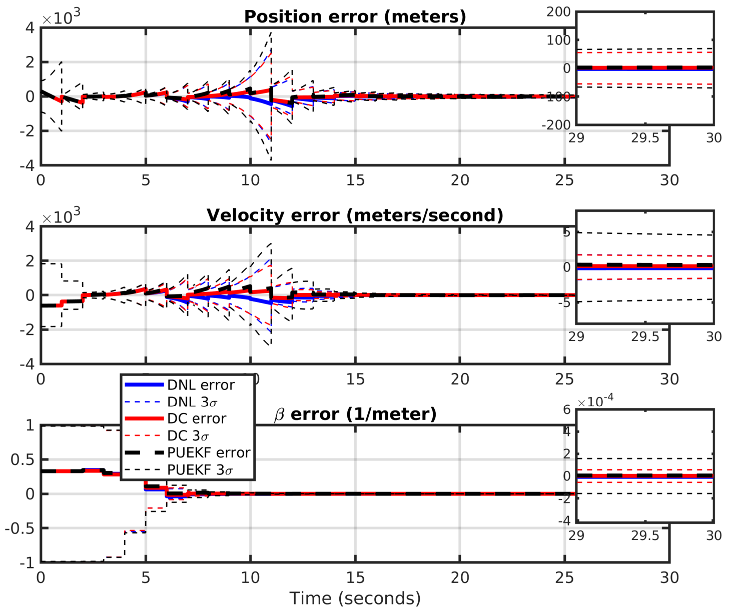


Figure 5.18: Dynamic covariance-aware partial-update method (DC) state errors for a single run of the body re-entry problem subject to 1 σ initial errors. Estimates from static and DNL methods are shown for comparison.

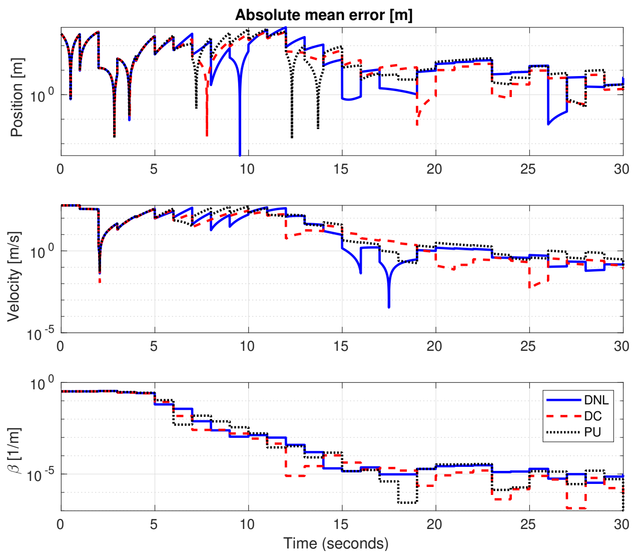


Figure 5.19: Dynamic covariance-aware partial-update method (DC) absolute errors for a single run of the body re-entry problem subject to 1 σ initial errors. Errors from static and DNL methods are shown for comparison.

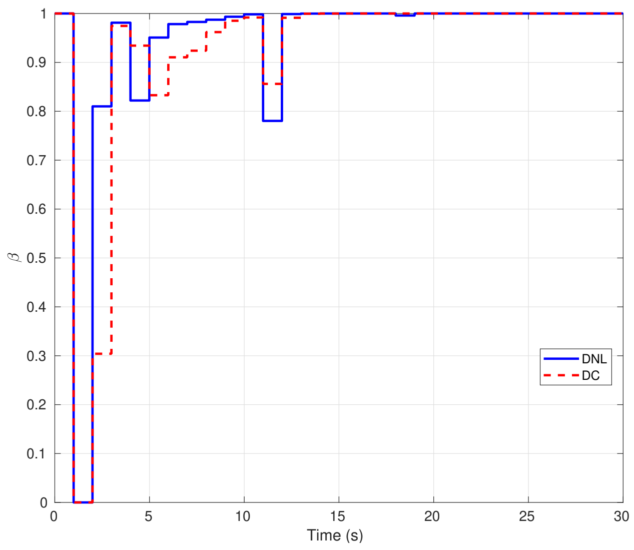


Figure 5.20: Dynamic covariance-aware partial-update method (DC) β history for single run of the body re-entry problem subject to 1 σ initial errors. β produced via DNL method is shown for comparison.

Figure 5.22, a zoomed-in version of Figure 5.18, shows that the main reason for the DC method to have better performance is that although it slightly overshoots on the ballistic parameter early on

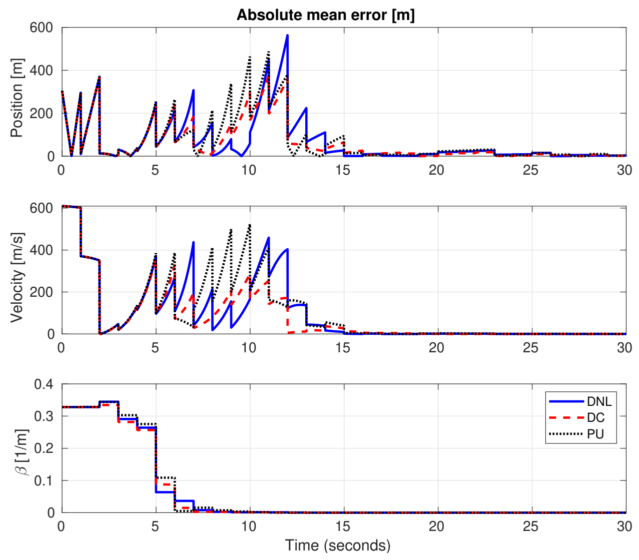


Figure 5.21: Dynamic covariance-aware partial-update method (DC) absolute errors for a single run of the body re-entry problem subject to 1 σ initial errors. Errors from static and DNL methods are shown for comparison.

(at time $t = 6s$) when the drag effects (and thus nonlinearities) influence is larger, it recovers faster than the DNL and static partial-update methods. As a result, even all three filters estimates appear within the 3 σ bounds, DC partial-update filter is superior. The fast recovery of the DC method in this run, suggests that this method uses better the information provided on second-order effects to select the β values.

Although numerous experiments were conducted to evaluate the robustness of the DC method against initial errors and uncertainties, the results always evidenced that 1) the DC method owns better capabilities than the DNL method and 2) the DC method can perform practically as a carefully and well-tuned static partial-update. Figure 5.23 shows a representative run of the filters when they all undergo initial errors that are 3 σ in magnitude. It is clear that the DC partial-update and the static method performed well, and their estimates are within the 3 σ bounds, whereas the DNL method does not. For this scenario, the DNL method was found to support maximum initial errors of up to 1 σ. Although from Figure 5.24, it may seem that the static partial-update method incurred in less error, the computation and comparison of the absolute error histories reveal that the covariance-aware method incurred in the least estimation error among the filters.

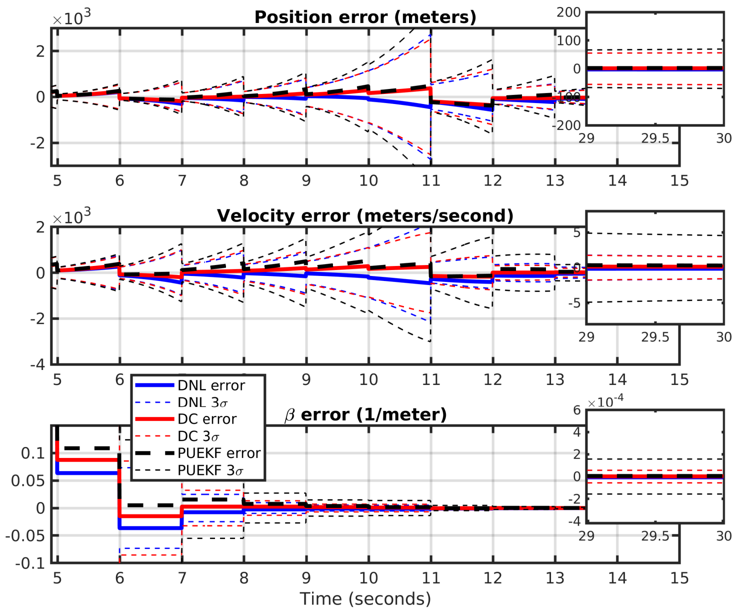


Figure 5.22: Zoomed-in view for dynamic covariance-aware partial-update method (DC) state errors. History of a single run of the body re-entry problem subject to 1 σ initial errors. Estimates from static and DNL methods are shown for comparison.

Additionally, 100 Monte Carlo runs were executed to show that the robustness observed for the DC method is not specific of the random draw used to initialize the filters for that run. From the resulting Monte Carlo runs are displayed in Figure 5.25, there are two main observations. First, the filter appears essentially as consistent as the finely tuned static partial-update. Second, the DC method handles the initial uncertainties and higher nonlinearities better than the pre-tuned static partial-update, and it effectively manages to produce less estimation errors overall. It is also important to mention, that even though the consistency of the filter is not perfect, it should be recalled that the filter is still a linear filter and that the conventional EKF was not operational under the scenarios presented in this and the previous sections. Finally, it must be noted that the static partial-update performs well overall without additional metrics, as in the case for the dynamic partial-update filter. However, recall that the specific β tuning used in this section ($\boldsymbol{\beta} = \begin{bmatrix} 1 & 1 & 0.75 \end{bmatrix}$) required extra effort that involved numerous Monte Carlo experiments in covering and successfully running a variety of scenarios whereas the DC method required no tuning at all.

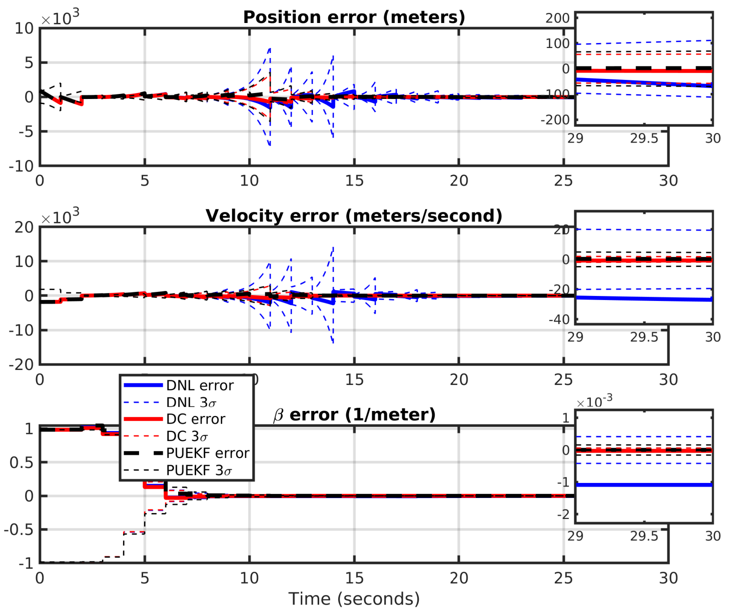


Figure 5.23: Dynamic covariance-aware partial-update method (DC) state errors. History of a single run of the body re-entry problem subject to 3 σ initial errors. Estimates from static and DNL methods are shown for comparison.

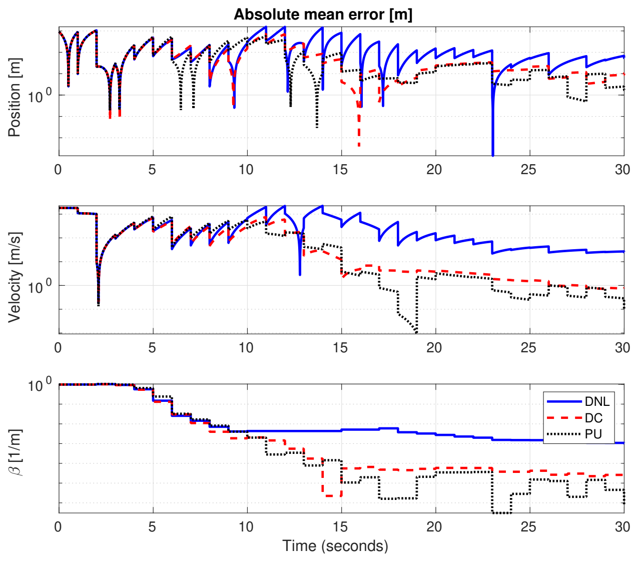


Figure 5.24: Dynamic covariance-aware partial-update method (DC) absolute errors. History of a single run of the body re-entry problem subject to 3 σ initial errors. Errors from static and DNL methods are shown for comparison.

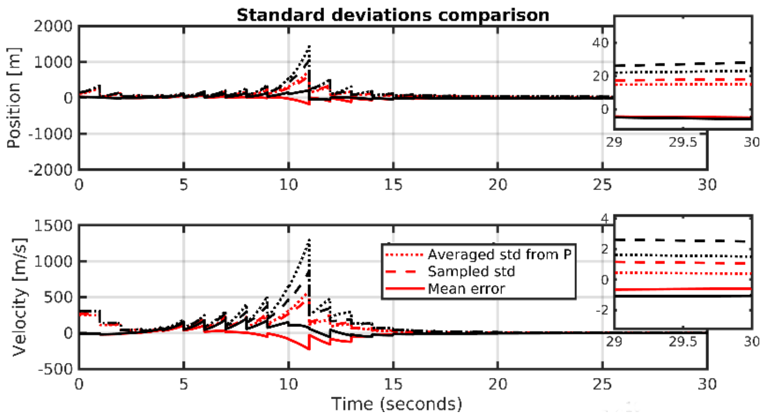


Figure 5.25: Averaged and sampled standard deviation from 100 Monte Carlo runs for dynamic covariance-aware partial-update method (DC) state errors. Estimates from static method are also shown. DNL method is not included as it does not support initial errors higher than 1 σ. Full update is used as the baseline for updates.

An additional comment on the partial-update weight history from Figure 5.26, and in general for the behavior of the β weight when the DNL method is utilized, is that the values, compared to the DC method, tend to fluctuate more aggressively. It appears that this way of quantifying the effect of second-order terms is more *sensitive* than using the second-order covariance terms. Experimentation on scaling the resulting β profile produced by the DNL method was also performed to examine both *metrics*, DNL and DC, when having similar profiles. However, it was found that is not a scale issue, but the experiments suggested that the DNL method does not perform as well as the DC method in general, and the difference is due to the high-order terms quantification. Such sensitivity of the DNL was observed consistently across all experiments involving the dynamic methods. β histories from Figure 5.4 and Figure 5.4, for example, also show this behavior.

5.4.2 Pre-tuned partial-update weights as a baseline for the DC method

The DC method using known β values was observed to be almost identical in behavior and consistency as the DNL (with known weights) method when exercised with relatively low initial errors (1 σ or less). For moderate-high initial errors and uncertainties, however, the covariance-aware method was found to perform the best compared with DNL and static methods. A typical run

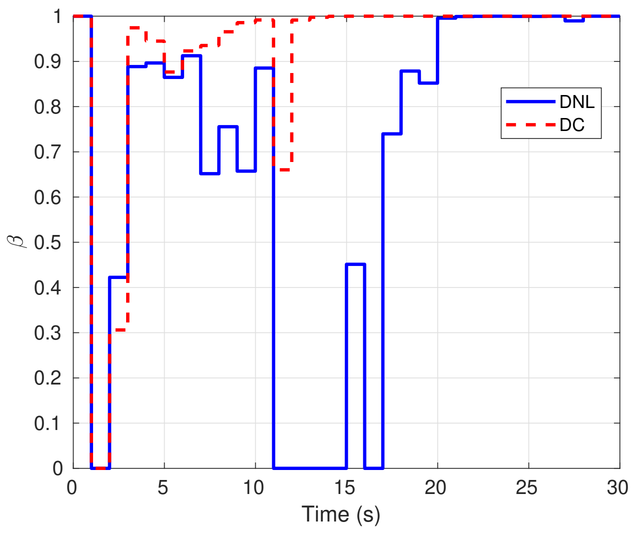


Figure 5.26: Dynamic covariance-aware partial-update method (DC) β history of a single run of the body re-entry problem subject to 3 σ initial errors. The β history produced by the DNL partial-update method is also included for comparison.

of the DC partial-update filter with high initial errors and uncertainties (errors near 3 σ, doubled uncertainty on position and velocity, and tripled uncertainty on ballistic parameter, with respect to base values from Table 4.2) are depicted in Figure 5.27. The corresponding absolute errors are displayed in Figure 5.28. It is worth noting that consistently across experiments, the DNL method generally exhibits larger reactions to second-order terms compared to the DC method, as Figure 5.29 shows.

Since the methods that use known weights were seen to support high initial errors and uncertainties, several experiments were performed to explore their limits. Overall, it was found that the DC method is more robust, and that to cause divergence, for the re-entry body scenario, extreme cases were needed to make it diverge. Although such bad initial conditions may not be realistic, the objective was to find sufficiently adverse conditions for the dynamic filters to diverge to gain insight into the dynamic methods limits.

One of the experiments showing the covariance-aware method diverging, used initial conditions where the uncertainties on position and velocity states were doubled, and the ballistic parameter uncertainty is five times the original σ (with respect to reference parameters given in Table 4.2).

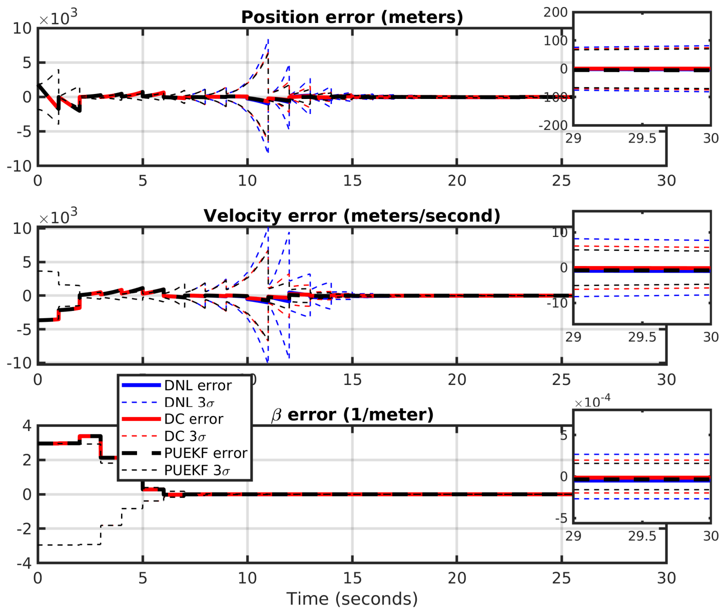


Figure 5.27: Dynamic covariance-aware partial-update method (DC) state errors. History of a single run of the body re-entry problem subject to moderate uncertainties (around $\begin{bmatrix}2\sigma_{p0} & 2\sigma_{v0} & 3\sigma_{\beta_0}\end{bmatrix}$) and 3 σ initial errors. Estimates from static and DNL methods are shown for comparison.

Figure 5.30, shows a filter single run of this scenario with high initial uncertainties. Here, the reason for failure can be attributed to the DC method insufficient reaction to produce large enough weight variations when required as observed from Figure 5.31, making the filter to eventually fail. The zoomed-in version of the estimates errors, shown in Figure 5.32, in fact, confirms that insufficiently lowering the β value when high nonlinearities appeared was the root of divergence. Notice that Figure 5.30 are the results of the same scenario used in the previous section when stressing the DNL method with high uncertainties (see Figure 5.16), but here, is it clearer how the DNL method managed to handle the large nonlinearities. The DNL method reacted to the extent of even temporarily act as a consider filter, mainly due to its larger sensitivity to second-order effects (Figure 5.31), being able to retain a consistent filter, which is even better than the finely tuned static partial-update approach for the presented scenario.

Although these are just the results from one run, the overall behavior showed that it takes a

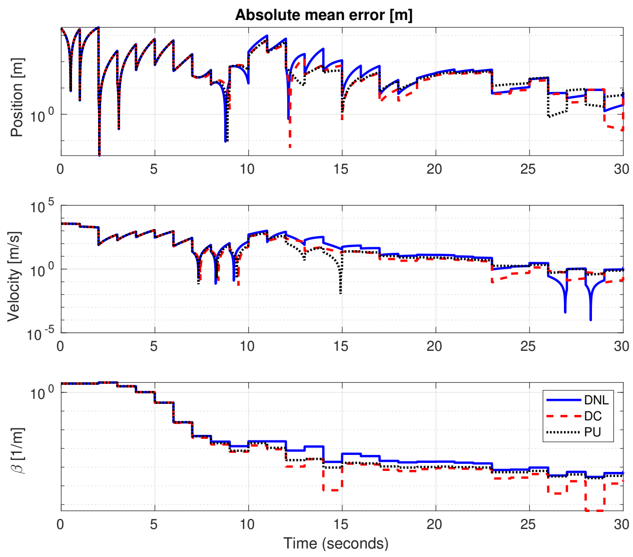


Figure 5.28: Dynamic covariance-aware partial-update method (DC) absolute errors. History of a single run of the body re-entry problem subject to moderate uncertainties (around $\begin{bmatrix} 2\sigma_{p0} & 2\sigma_{v0} & 3\sigma_{\beta_0} \end{bmatrix}$) and 3 σ initial errors. Errors from static and DNL methods are shown for comparison.

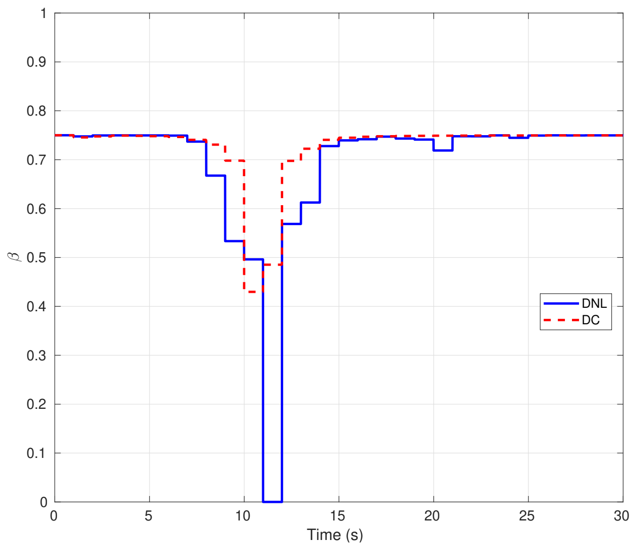


Figure 5.29: Dynamic covariance-aware partial-update method (DC) β history of a single run of the body re-entry problem subject to uncertainties (around $\begin{bmatrix} 2\sigma_{p0} & 2\sigma_{v0} & 3\sigma_{\beta_0} \end{bmatrix}$) and 3 σ initial errors. The β history produced by the DNL partial-update method is also included for comparison.

relatively bad initialized filter to make the dynamic methods fail, especially for the DC method, which was seen to be more robust than the DNL method overall. Scenarios were the DNL method fails and the DC method works, were also seen and more common for high initial errors, the plots are not included here but they look similar to the divergent DNL of Figure 5.23.

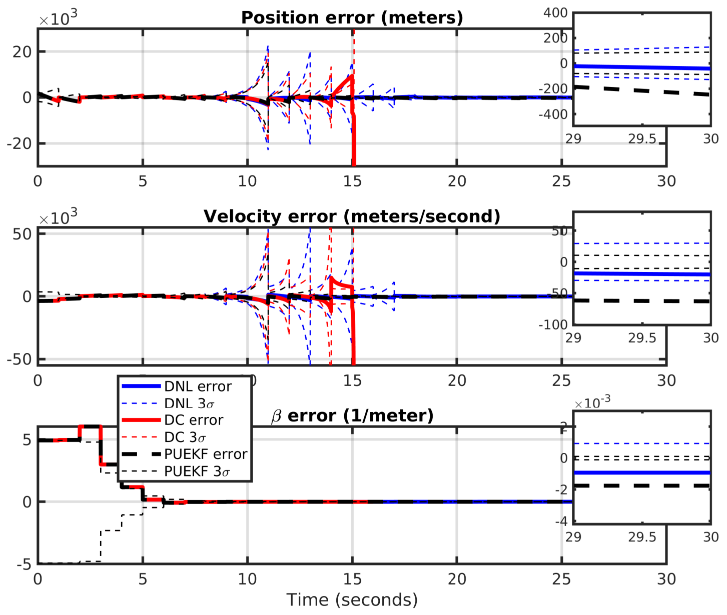


Figure 5.30: Dynamic covariance-aware partial-update method (DC) state errors. History of a single run of the body re-entry problem subject to high uncertainties (around $\begin{bmatrix} 2\sigma_{p0} & 2\sigma_{v0} & 5\sigma_{\beta_0} \end{bmatrix}$) and 3 σ initial errors. Estimates from static and DNL methods are shown for comparison.

5.4.3 Comments on DNL and DC partial-update methods

The dynamic beta selection methods showed similar behavior when the initial errors and uncertainties were relatively small. In those cases, it was observed that a finely tuned static partial-update in practical terms would offer similar benefits as a dynamic β selection method, but at the extra effort of tuning weights. However, as seen from the simulated examples, if nonlinearities are significant, the static partial-update will incur in higher estimation errors. Interestingly, although the dynamic methods appear slightly over or under confident with respect to the static partial-update method, the changes in the 3 σ bounds are relatively small. In other words, the use of a method for dynamically choosing the β's does not sacrifice accuracy importantly; and it makes the filter

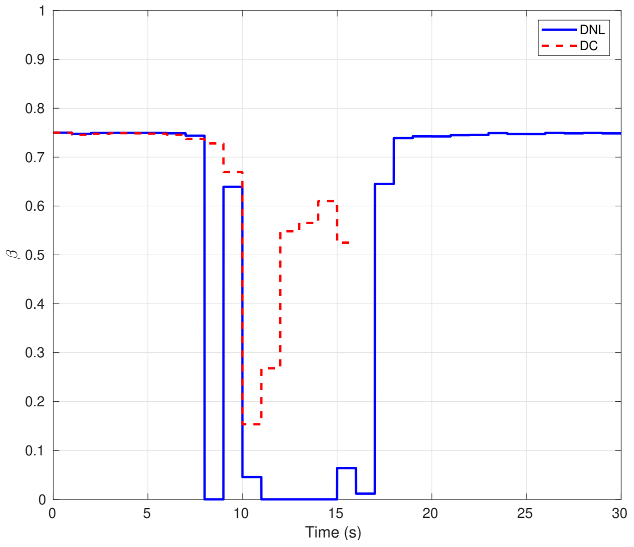


Figure 5.31: Dynamic covariance-aware partial-update method (DC) β history of a single run of the body re-entry problem subject to high uncertainties (around $\begin{bmatrix}2\sigma_{p0} & 2\sigma_{v0} & 5\sigma_{\beta_0}\end{bmatrix}$) and 3 σ initial errors. The β history produced by the DNL partial-update method is also included for comparison.

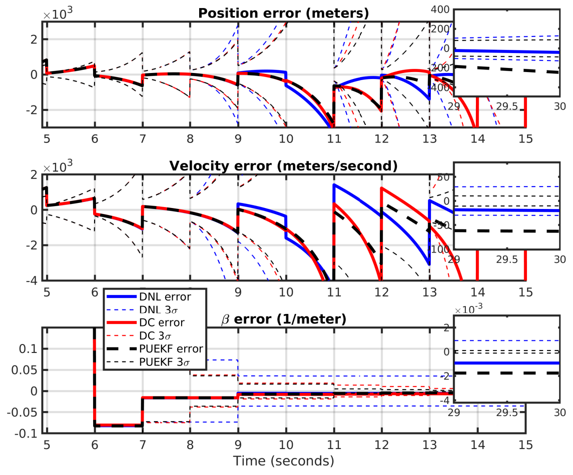


Figure 5.32: Zoomed-in view for dynamic covariance-aware partial-update method (DC) state errors. History of a single run of the body re-entry problem subject to high uncertainties (around $\begin{bmatrix}2\sigma_{p0} & 2\sigma_{v0} & 5\sigma_{\beta_0}\end{bmatrix}$) and 3 σ initial errors. Estimates from static and DNL methods are shown for comparison.

operational that otherwise would be divergent. As seen from the Monte Carlo runs, the agreement of sample and averaged standard deviation, although it is not perfect (due to mainly to high initial errors), is fairly accurate, and certainly far better than the conventional EKF. Moreover, for scenarios with important nonlinearities and uncertainties, the dynamic methods exhibit lower mean estimation error, especially the covariance-aware method. The virtues of the dynamic beta selection methods, however, come at the cost of the computation of second-order terms. On the other hand, the dynamic methods are still attractive since they effectively allow the use of an EKF filter structure where a more advanced filter may be needed.

Even though different metrics and methods can be proposed to perform online β selection, the methods presented here are functional, do not over-specialize the filter,retain the EKF structure, and use information that is already computed within the KF. Finally, based on the presented results, it is always recommended to use the DC method because it provides higher robustness against high initial errors and uncertainties. If computational cost is a priority, the DNL could be considered, but the user needs to be aware of its lower robustness.

6. HARDWARE APPLICATIONS*

This chapter describes three aerospace-related hardware applications of the partial-update filter. The specific applications are selected to demonstrate the partial-update filter's capabilities when the system's nonlinearities are significant, and the nuisance states need to be estimated. Furthermore, the implementations show the partial-update filter's capacity to work well where the nuisance states are constant, varying, or are the states of interest. In all three applications, the use of a Schmidt or a traditional EKF failed, or it did not produce acceptable estimates.

The first application is a filter-based IMU-camera (Inertial Measurement Unit) calibration system. The purpose of this filter is to find the rigid transformation (or calibration) between the camera and IMU to improve the global IMU navigation solution. The calibration parameters of this application are the constant nuisance states. This hardware application uses a UD partial-update filter and is shown to estimate well the calibration parameters with a sub-centimeter and sub-degree accuracy. The second application is a simplified altitude and velocity estimator for Unmanned Aerial Vehicles (UAV). The IMU biases are the nuisance parameters in this case. In contrast to the first application, these nuisance parameters change over time. A UD partial-update filter is also implemented for this application. The results showed a consistent filter for both core and varying nuisance states, providing a highly functional flight platform. The third application is a vision-based partial-update filter to estimate the angular rates of an uncooperative space body. In contrast with the first and second applications, the nuisance states are the angular rates, the primary states of interest. This last application uses a UD partial-update filter to estimate the angular rates.

*Adapted with permission from "Reef estimator: A simplified open source estimator and controller for multirotors", by J. H. Ramos, P. Ganesh, W. Warke, K. Volle and K. Brink, presented at the IEEE National Aerospace and Electronics Conference 2019 [2], Copyright 2019 by the Institute of Electrical and Electronics Engineers; "Vision-based tracking of non-cooperative space bodies to support active attitude control detection", by J. H. Ramos, T. D. Woodbury, and J. E. Hurtado, presented at the AIAA SPACE and Astronautics Forum and Exposition 2018 [55], Copyright 2018 by J. Humberto Ramos,Timothy Woodbury, and John E. Hurtado.

6.1 Online IMU-camera intersensor parameters calibration

6.1.1 Introduction

In recent years vision-aided inertial navigation based on cameras and IMU has captured the attention of many researchers. A camera and IMU working together offer a relatively low-weight and complementary team: while the IMU provides high-frequency system motion propagation, the camera provides slower but fixed and more stable (it does not drift in contrast with IMU biases) feedback measurement signal. When the IMU-camera set is used along with a sequential filtering technique, or similar IMU-camera fusion information algorithm [78], one can obtain an accurate inertial navigation system. Among the factors that contribute to an accurate vision-aided inertial navigation solution, the IMU-camera calibration parameters play a key role.

The knowledge of the relative placement between an inertial measurement unit and a camera, or IMU-camera calibration for short, is crucial [79]. Large inaccuracies in this calibration parameters can cause the system estimates to be biased, degraded, and even to diverge [80]. There exist several methods in the literature for IMU-camera calibration, and are mainly divided into two groups: batch and sequential calibrators. Filter based approaches are commonly more popular than batch approaches because they also provide confidence bounds for the calibration parameters and run online. Filter based approaches essentially estimate ego-states (position, velocity, and attitude) of the IMU, along with the relative pose between the camera and the IMU frame. Systems such as those reported in [81], [64] and [82], for instance, show the need for accounting and estimating the relative pose between the sensors to improve navigation results, which is the reason to perform the calibration in the first place.

The IMU-camera calibration of this hardware implementation is a filter-based approach. Generally, due to the high nonlinearities and uncertainties induced by the unknown IMU-camera transformation, and the nuisance parameters included in the state vector, a traditional EKF or Schmidt filter is often not sufficient to obtain acceptable estimates. For this problem, it is common that users alternatively implement an iterative EKF or even a UKF. This hardware application uses the

PU-MEKF filter (developed in Chapter 2) to accommodate better the calibration parameters, along with system biases (particularly IMU's biases). The PU-MEKF is implemented using the UD factorization presented in Chapter 4. As commonly done in visual-aided inertial navigation systems, the filter uses the IMU measurements to propagate the rotational kinematics model (of the rigid body carrying the IMU-camera system) while the camera provides image features locations measurements. For this application, the features or landmarks locations, are the extracted corners of a set of arUco markers and are considered to be known in the inertial frame, as emulating navigation using an a priori known map.

6.1.2 Related Work

One of the most popular batch techniques for IMU-camera calibration was introduced by Furgale et al. in [83]. The work proposes a batch least-squares solution that performs both a temporal and spatial calibration of an IMU and RGB sensor using fiducial markers. The authors provide an open-source implementation of their code along with relevant documentation. Although this technique provides a systematic solution for finding the rigid body transform, it is not suitable for online implementation. If an IMU-camera reconfiguration occurs, as for systems able to tilt and pan the camera, the calibration procedure needs to be repeated offline. Unfortunately, for this type of batch calibrator, the user does not know if the collected data is enough (qualitative and quantitatively) until the algorithm processes the data. Another least-square solution presented in [84], proposes the use of B-splines, and similarly to [83], it can also be used to identify temporal IMU-camera alignment. This method is not an online approach neither, but it reports accurate results.

In regards to filter-based calibration algorithms, there is a variety of flavors and forms. For example, [85] proposes the use of an unscented Kalman filter (UKF)to perform the calibration, and it even includes gravity vector into the state vector to improve results. In [79], the use of a UKF is also introduced and taken further as it generalizes the IMU-camera calibration problem to a multicamera-IMU system. Although these approaches are shown to be functional, they require the user to be familiar with the unscented approach and its implementation. Also, in the Kalman filters line, [86] proposes an extended Kalman filter. This work includes a nonlinear observability analysis (similar to that presented in [85]), showing that the IMU-camera calibration parameters can be estimated from the camera and IMU measurements alone. More specifically, this method uses an iterative multiplicative Extended Kalman filter (IMEFK) to handle the high nonlinearities that appear in the measurement model. The IMEKF is grounded in an extended Kalman filter (EKF), which makes it practical from the implementation perspective, but it may become computationally expensive as each set of camera measurements are re-processed multiple times at each update step. Other filtering-based approaches include the technique proposed by Li and Mourikis [87]. This technique incorporates the IMU-camera pose into a multi-state constraint Kalman filter, which is shown to achieve acceptable calibration parameters. However, the use of such an algorithm requires more coding effort (with respect to a conventional single-step propagation/update filter) due to the need of book-keeping past system poses. Yang and Shen in [88] propose a method to initialize velocity, scale, and IMU-camera calibration in real-time without requiring artificial markers. To achieve this, the authors propose a probabilistic optimization-based procedure. The method is shown to be functional, but it may require more experienced users for its implementation. Highly precise techniques, like that presented in [89], are also available in the literature, but they come at the cost of requiring significantly accurate modeling. Non-filter-based that use closed-form solutions to perform the calibration as the one presented in [90] are also available. However, they may not provide uncertainty quantification on the calibration parameters and are prone to be sensitive to the calibration target pose.

6.1.3 Filter-based IMU-camera calibration algorithm

In this section, the description of the calibration algorithm is given. As the process is PU-MEKF-based, its description has been divided into propagation and measurement update step. Recall that as conventionally done, the propagation is performed using IMU measurements while the update uses camera measurements. Particularly, Section 6.1.3.2 details the propagation step and Section 6.1.3.3 describes the measurement update step. Before going into the specifics of the proposed calibration filter, the following section establishes additional notation used throughout this hardware implementation description.

6.1.3.1 Notation

The notation ${}^{A}\mathbf{p}_{B}$ is used to denote the vector $\mathbf{p}$ of property B *coordinatized* in reference frame A. For example, ${}^{W}\mathbf{p}_{I}$ represents the position vector of the IMU (I) with respect to the world frame (W). The collection of elements in a column vector is written as ${}^{A}\mathbf{p}$, with no reference frame associated to it. Rotations are parametrized by quaternions. Quaternions are denoted by $\bar{q}$. A rotation matrix such as $\mathbf{C}({}^{B}_{A}\bar{\mathbf{q}})$, for example, is the passive rotation $\mathbf{C}$ constructed from the quaternion parametrization ${}^{B}_{A}\bar{\mathbf{q}}$. Here $\mathbf{C}({}^{B}_{A}\bar{\mathbf{q}})$ is a rotation that takes a vector initially represented in reference frame A, and expresses it in the coordinate frame B. Passive rotations are also written simply as ${}^{B}_{A}\mathbf{C}$, which is considered equivalent to $\mathbf{C}({}^{B}_{A}\bar{\mathbf{q}})$. The operator $\lfloor \mathbf{v}\,\times \rfloor$ constructs a skew-symmetric matrix using the elements of vector $\mathbf{v} = \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix}^{\mathrm{T}}$ according to

$$\lfloor \mathbf{v}\,\times \rfloor = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix}. \tag{6.1}$$

Variables written with a delta prefix, such as $\delta\mathbf{x}$, represent an error quantity. Finally, recall that the error definition is considered as the difference between the true, $\mathbf{x}$, and expected value, $\hat{\mathbf{x}}$, as $\delta\mathbf{x} = \mathbf{x} - \hat{\mathbf{x}}$. Any variations in notation will be clear from the context or will be clarified as needed.

6.1.3.2 The propagation step

This section describes the particulars of the propagation step performed in the IMU-camera calibration implementation.

6.1.3.2.1 State vector. The IMU-camera calibration parameters and the IMU biases are the nuisance states in this filter. Whereas the core or ego-states are the position, velocity, and attitude of the IMU in the inertial frame. The notation for the PU-MEKF states elements as shown in Equation (6.2) is: attitude (${}^{I}_{W}\bar{q} \in \mathbb{R}^4$) and position (${}^{W}\mathbf{p}_{I} \in \mathbb{R}^3$) of the IMU with respect to the world frame, the velocity of the IMU with respect to the world frame (${}^{W}\mathbf{v}_{I} \in \mathbb{R}^3$), IMU gyroscope

($\mathbf{b}_g \in \mathbb{R}^3$) and accelerometer ($\mathbf{b}_a \in \mathbb{R}^3$) biases, and the IMU-camera calibration (${}^I\mathbf{p}_C$,${}^C_I\bar{q}$). The sought calibration is considered to be the position of the camera with respect to the IMU frame, ${}^I\mathbf{p}_C \in \mathbb{R}^3$, and the attitude of the camera frame with respect to the IMU frame, ${}^C_I\bar{q} \in \mathbb{R}^4$. The state vector denoted by $\mathbf{x} \in \mathbb{R}^{23}$, encapsulates the ego-states and calibration parameters as indicated in Equation (6.2).

$$\mathbf{x} = \begin{bmatrix} {}^I_W\bar{q}^{\mathrm{T}} & {}^W\mathbf{p}_I^{\mathrm{T}} & {}^W\mathbf{v}_I^{\mathrm{T}} & \mathbf{b}_g^{\mathrm{T}} & \mathbf{b}_a^{\mathrm{T}} & {}^I\mathbf{p}_C^{\mathrm{T}} & {}^C_I\bar{q}^T \end{bmatrix} . \tag{6.2}$$

6.1.3.2.2 True process model. In this subsection, the process model is discussed. The measurement model is description is deferred for Section 6.1.3.3.1.

The PU-MEKF model uses the IMU's gyroscope and accelerometer measurements to propagate rotational and translational rigid body kinematics. Since the raw IMU measurements inevitably deviate from the true values, a measurement model to consider the corrupted measurements is used [21] before utilizing the IMU measurements. In this work, the 3×1 gyroscope measurement vector (the angular velocity) $\tilde{\boldsymbol{\omega}}$, and the 3×1 accelerometer measurements (specific force) $\tilde{\mathbf{s}}$ are related to their respective true values, $\boldsymbol{\omega}$ and $\mathbf{s}$, via the following models:

$$\boldsymbol{\omega} = \tilde{\boldsymbol{\omega}} - \mathbf{b}_g - \mathbf{n}_g , \tag{6.3}$$

$$\dot{\mathbf{b}}_g = \mathbf{n}_{wg} , \tag{6.4}$$

$$\mathbf{s} = \tilde{\mathbf{s}} - \mathbf{b}_a - \mathbf{n}_a , \tag{6.5}$$

$$\dot{\mathbf{b}}_a = \mathbf{n}_{wa} . \tag{6.6}$$

Here, $\mathbf{n}_g$ is the gyroscope measurement noise, and $\mathbf{n}_{wg}$ is the gyroscope bias drift respectively, whereas $\mathbf{n}_a$ and $\mathbf{n}_{wa}$ are the accelerometer measurement noise and accelerometer bias drift; with $\mathbf{b}_g$ and $\mathbf{b}_a$ being the gyro and accelerometer bias. Both noise and drift (for angular and translational quantities) 3×1 vectors, are modeled as zero-mean white Gaussian noise processes.

This being said, the true process model for the IMU-camera calibration (rotational and translational kinematics) is now summarized in the following equations [66], [81], [86]:

$$ {}^{I}_{I_k}\dot{\bar{q}}(t) = \frac{1}{2}\begin{bmatrix} -\lfloor \boldsymbol{\omega}(t) \times \rfloor & \boldsymbol{\omega}(t) \\ -\boldsymbol{\omega}(t)^{\mathrm{T}} & 0 \end{bmatrix} {}^{I}_{I_k}\bar{q}(t) \ , \tag{6.7} $$

$$ {}^{W}\dot{\mathbf{p}}_I = {}^{W}\mathbf{v}_I \ , \tag{6.8} $$

$$ {}^{W}\dot{\mathbf{v}}_I(t) = {}^{W}\mathbf{a}_I(t) = ({}^{I}_{I_k}\mathbf{C}\ {}^{I_k}_{W}\mathbf{C})^{\mathrm{T}} \ , \mathbf{s}(t) + {}^{W}\mathbf{g} \ , \tag{6.9} $$

$$ \dot{\mathbf{b}}_g(t) = \mathbf{n}_{wg}(t) \ , \tag{6.10} $$

$$ \dot{\mathbf{b}}_a(t) = \mathbf{n}_{wa}(t) \ , \tag{6.11} $$

$$ {}^{I}\dot{\mathbf{p}}_C = 0 \ , \tag{6.12} $$

$$ {}^{C}_{I}\dot{\mathbf{q}} = 0 \ . \tag{6.13} $$

The notation I_k is intended to represent the frame from which the attitude evolution starts at time t_k (when a new IMU measurement arrives), and I, the frame where the evolution ends (after the integration of the equations, for either continuous or closed-discrete form solution). The vector ${}^{W}\mathbf{g}$ is the 3×1 gravity vector in the world frame, and it is assumed to be known without uncertainty. Equation (6.9) describes the evolution of the IMU velocity. Notice that this equation accounts only for IMU motion-induced measurements, as the gravity vector is being subtracted. Finally, equations (6.12) and (6.13) indicate that the calibration parameters are constant.

6.1.3.2.3 Expectation of process model. To form the error dynamics for the PU-MEKF (recall it is an indirect filter), that the expected model is required as well. The expected model is obtained by computing the expected value of equations (6.7)-(6.13) and the IMU measurement models. That is,

$$ \dot{\hat{\mathbf{b}}}_g(t) = \mathbf{0} \rightarrow \hat{\mathbf{b}}_g(t) = \hat{\mathbf{b}}_{g_k} \ , \tag{6.14} $$

$$\hat{\boldsymbol{\omega}}(t) = \tilde{\boldsymbol{\omega}}(t) - \hat{\mathbf{b}}_g(t) = \tilde{\boldsymbol{\omega}}(t) - \hat{\mathbf{b}}_{g_k} \ , \tag{6.15}$$

$${}^{I}_{I_k}\dot{\hat{\bar{q}}}(t) = \frac{1}{2}\begin{bmatrix} -\lfloor \hat{\boldsymbol{\omega}}(t) \times \rfloor & \hat{\boldsymbol{\omega}}(t) \\ -\hat{\boldsymbol{\omega}}^{\mathrm{T}}(t) & 0 \end{bmatrix} {}^{I}_{I_k}\bar{q}(t) \ , \tag{6.16}$$

$$\dot{\hat{\mathbf{b}}}_a(t) = \mathbf{0} \rightarrow \hat{\mathbf{b}}_a(t) = \hat{\mathbf{b}}_{a_k} \ , \tag{6.17}$$

$$\hat{\mathbf{s}}(t) = \tilde{\mathbf{s}}(t) - \hat{\mathbf{b}}_a(t) = \tilde{\mathbf{s}}(t) - \hat{\mathbf{b}}_{a_k} \ , \tag{6.18}$$

$$\dot{\hat{\mathbf{v}}}(t) = {}^{I_k}_{W}\hat{\mathbf{C}}^{\mathrm{T}} \, {}^{I}_{I_k}\hat{\mathbf{C}}^{\mathrm{T}} \hat{\mathbf{s}}(t) + {}^{W}\mathbf{g} \ . \tag{6.19}$$

Here, the biases $\hat{\mathbf{b}}_{g_k}$ and $\hat{\mathbf{b}}_{a_k}$ indicate that the biases are to remain constant during the propagation step (at time $t = k$), however, they are allowed to change within the filter (through process noise) as they are modeled as random walks.

6.1.3.2.4 Forming the error model. Now the error model for the PU-MEKF can be constructed. To form the error model one needs to accordingly substitute the true and expected models (from the two previous subsections) into the additive and multiplicative error definitions from Equations (6.20) and (6.21), and simplify.

$$\delta\mathbf{x} = \mathbf{x} - \hat{\mathbf{x}} \ . \tag{6.20}$$

$$\mathbf{C}(\delta\bar{q}) = \mathbf{C}({}^{I}_{W}\bar{q})\mathbf{C}({}^{I}_{W}\hat{\mathbf{q}})^{\mathrm{T}} \ . \tag{6.21}$$

The expressions reduction results in the following error model equations:

$$\delta\dot{\boldsymbol{\theta}} = \lfloor \tilde{\boldsymbol{\omega}} - \hat{\mathbf{b}}_g \rfloor \delta\boldsymbol{\theta} - \delta\hat{\mathbf{b}}_g - \mathbf{n}_g \rfloor \ , \tag{6.22}$$

$$\delta^W\dot{\mathbf{p}}_I = \delta^W\mathbf{v}_I \ , \tag{6.23}$$

$$\delta^W\dot{\mathbf{v}}_I = -{}^I_W\hat{\mathbf{C}}^T(\delta\mathbf{b}_a + \mathbf{n}_a + \lfloor \tilde{\mathbf{s}} - \hat{\mathbf{b}}_z \rfloor \delta\boldsymbol{\theta}_I) \ , \tag{6.24}$$

$$\delta\dot{\mathbf{b}}_g = \mathbf{n}_{gw} \ , \tag{6.25}$$

$$\delta\dot{\mathbf{b}}_a = \mathbf{n}_{aw} \ , \tag{6.26}$$

$$\delta^I\dot{\mathbf{p}}_C = 0 \ , \tag{6.27}$$

$$\delta\dot{\boldsymbol{\alpha}} = 0 \ . \tag{6.28}$$

Here, δ denotes small departure from true values. Specifically, $\delta\dot{\boldsymbol{\alpha}}$ is the small angle error representation for the attitude offset between the estimated rotation${}^C_I\hat{\bar{q}}$ and the true rotation ${}^C_I\bar{q}$. Similarly, $\delta\dot{\boldsymbol{\theta}}$ represents the IMU attitude error. The error state vector, $\delta\mathbf{x} \in \mathbb{R}^{21}$ becomes,

$$\delta\mathbf{x} = \begin{bmatrix} \delta\boldsymbol{\theta}^{\mathrm{T}} & \delta^W\mathbf{p}_I^{\mathrm{T}} & \delta^W\mathbf{v}_I^{\mathrm{T}} & \delta\mathbf{b}_g^{\mathrm{T}} & \delta\mathbf{b}_a^{\mathrm{T}} & \delta^I\mathbf{p}_C^{\mathrm{T}} & \delta\boldsymbol{\alpha}^{\mathrm{T}} \end{bmatrix} \ . \tag{6.29}$$

The reduction in the dimension of the state vector results from the small attitude error representation (small-angle error, which leads to a minimum attitude parametrization). This, allows 1) to approximate quaternion errors with a three elements parametrization instead of four (for both $\delta\boldsymbol{\theta}$ and $\delta\boldsymbol{\alpha}$), and 2) remove the explicit quaternion unit-norm constraint, which otherwise would induce a singular covariance matrix.

6.1.3.2.5 Covariance propagation. Now, with the error dynamics in hand the Jacobians (with respect to the error variables) for uncertainty propagation are obtained. The PU-MEKF covariance propagation equation is identical to the one used for the standard Kalman filter,

$$\dot{\mathbf{P}} = \mathbf{F}\mathbf{P} + \mathbf{P}\mathbf{F}^{\mathrm{T}} + \mathbf{G}\mathbf{Q}\mathbf{G}^{\mathrm{T}} \ , \tag{6.30}$$

with the Jacobians defined by

$$\mathbf{F} = \frac{\partial \dot{\delta \mathbf{x}}}{\partial \delta \mathbf{x}}\Big|_{\mathbf{E}\delta\mathbf{x}\ ,\ \hat{\mathbf{x}}}, \tag{6.31}$$

and

$$\mathbf{G} = \frac{\partial \dot{\delta \mathbf{x}}}{\partial \mathbf{w}}\Big|_{\mathbf{E}\delta\mathbf{x}\ ,\ \hat{\mathbf{x}}}. \tag{6.32}$$

Calculating the corresponding Jacobians as per Equations (6.31) and (6.32), the continuous-time state transition matrix $\mathbf{F}$, and the input noise matrix $\mathbf{G}$ are obtained as

$$\mathbf{F} = \begin{bmatrix} -\lfloor \hat{\boldsymbol{\omega}} \times \rfloor & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & -\mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ -{}^{I}_{W}\hat{\mathbf{C}}^{\mathrm{T}}\lfloor(\tilde{\mathbf{s}} - \hat{\mathbf{b}}_a) \times \rfloor & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & -{}^{I}_{W}\hat{\mathbf{C}}^{\mathrm{T}} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \end{bmatrix}, \tag{6.33}$$

and

$$\mathbf{G} = \begin{bmatrix} -\mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & -{}^{I}_{W}\hat{\mathbf{C}}^{\mathrm{T}} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \end{bmatrix}, \tag{6.34}$$

where

$$\mathbf{w} = \begin{bmatrix} \mathbf{n}_g^{\mathrm{T}} & \mathbf{n}_a^{\mathrm{T}} & \mathbf{n}_{wg}^{\mathrm{T}} & \mathbf{n}_{wa}^{T} \end{bmatrix}^{T}. \tag{6.35}$$

6.1.3.3 PU-MEKF Measurement Update.

The PU-MEKF uses the linear update from the standard Kalman filter and as such, it assumes a linear measurement model that is corrupted by zero-mean white Gaussian noise $\mathbf{v}_k$ of the form

$$\mathbf{y}_k = \mathbf{H}_k(\hat{\mathbf{x}}_k)\mathbf{x}_k + \mathbf{v}_k \; . \tag{6.36}$$

However, since this is an indirect filter, the linear measurement model operates in a measurement residual space.

$$\mathbf{r}_k = \mathbf{H}_k(\hat{\mathbf{x}}_k)\delta\mathbf{x}_k + \mathbf{v}_k \; . \tag{6.37}$$

This is, the measurement matrix now is a map from state errors to measurement residuals. To obtain the measurement matrix $\mathbf{H}$ required in the measurement update, the measurement residual $\mathbf{r}$ is formed and then linearized.

The measurement residual is defined as the difference between the true and the expected measurement model, this is

$$\mathbf{r}_k = \mathbf{r}_k(\delta\mathbf{x}_k, \mathbf{v}_k, \hat{\mathbf{x}}_k) = \tilde{\mathbf{y}}(\delta\mathbf{x}_k, \hat{\mathbf{x}}_k) + \mathbf{v}_k - \mathbf{E}[\tilde{\mathbf{y}}(\delta\mathbf{x}_k, \hat{\mathbf{x}}_k)] \; . \tag{6.38}$$

Consequently, its first-order Taylor series expansion about the nominal state [$\mathbf{E}[\delta\mathbf{x}_k]$, $\mathbf{E}[\mathbf{v}_k]$, $\hat{\mathbf{x}}_k$], can be carried out as

$$\mathbf{r}_k \approx \mathbf{r}_k(\mathbf{E}[\delta\mathbf{x}_k], \mathbf{E}[\mathbf{v}_k], \hat{\mathbf{x}}_k) + \frac{\partial \mathbf{r}_k}{\partial \delta\mathbf{x}_k}(\delta\mathbf{x}_k - \mathbf{E}[\delta\mathbf{x}_k]) + \frac{\partial \mathbf{r}_k}{\partial \mathbf{v}_k}(\mathbf{v}_k - \mathbf{E}[\mathbf{v}_k]) \; . \tag{6.39}$$

Note that $\hat{\mathbf{x}}$ is not considered a variable when computing the Taylor expansion. Or simply expressed as

$$\mathbf{r}_k \approx \mathbf{H}\delta\mathbf{x}_k + \mathbf{v}_k \; . \tag{6.40}$$

Equation (6.40) is obtained in virtue of $\dfrac{\partial \mathbf{r}_k}{\partial \delta\mathbf{x}_k} = \mathbf{H}_k$, $\dfrac{\partial \mathbf{r}_k}{\partial \mathbf{v}_k} = \mathbf{I}$, $\mathbf{E}[\delta\mathbf{x}_k] = 0$, $\mathbf{E}[\mathbf{v}_k] = 0$ and the expectation of the residual being equal to zero,

$$\mathbf{E}[\mathbf{r}_k] = \mathbf{r}_k(\mathbf{E}[\delta\mathbf{x}_k], \mathbf{E}[\mathbf{v}_k], \hat{\mathbf{x}}_k) = 0 \; . \tag{6.41}$$

Again, note that measurement matrix $\mathbf{H}$ maps from state errors to residuals; thus, the differentiation of the residual $\mathbf{r}$ is with respect to state errors $\delta\mathbf{x}$. In the following section, the specifics to derive the measurement matrix $\mathbf{H}$ for the IMU-camera calibration algorithm are given.

6.1.3.3.1 Measurement model. As mentioned at the beginning of Section 6.1.3, the IMU-camera calibration uses features positions detected in the camera field of view to perform the filter update step. The measurement model thus relates the state to features positions. More specifically, per the linear measurement model from Equation (6.37), the measurement model maps from error state to pixels residuals.

For the IMU-camera system, the measurement model is established by relating the position of a feature (feature position vector) in the camera frame to its position in the image space (pixels) via a camera model. This implementation uses the conventional pinhole camera model from Equation (6.43) to associate the pixel position (denoted $\tilde{u}_i$ and $\tilde{v}_i$) of the i^{th} feature to its position vector in the camera frame, ${}^{C}\mathbf{p}_{F_i}$. From Figure 6.1 it is straightforward to show that the position vector of an F_i feature coordinatized in the camera reference frame is,

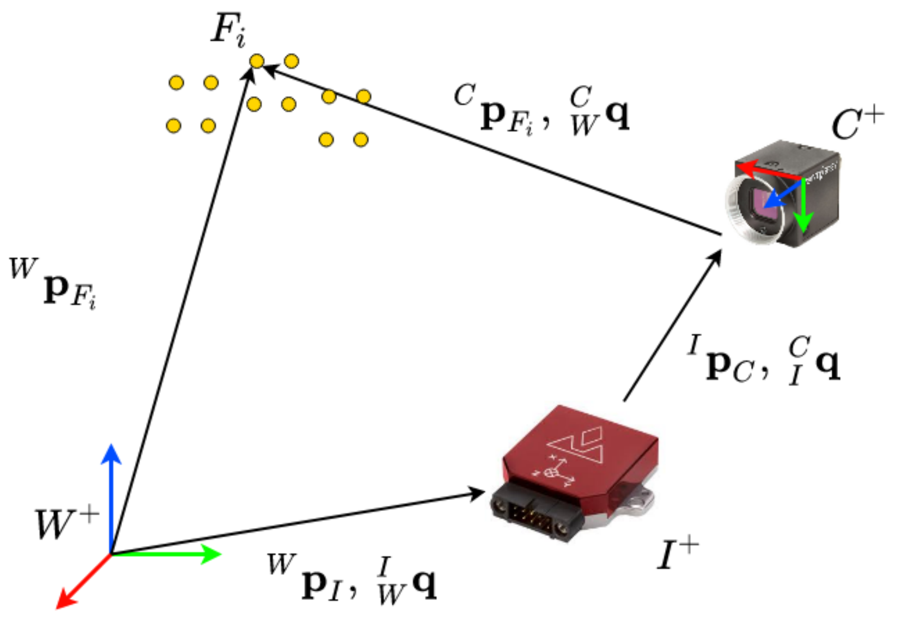

F_i
${}^{C}\mathbf{p}_{F_i}$, ${}^{C}_{W}\mathbf{q}$
C^+
${}^{W}\mathbf{p}_{F_i}$
${}^{I}\mathbf{p}_{C}$, ${}^{C}_{I}\mathbf{q}$
W^+
I^+
${}^{W}\mathbf{p}_{I}$, ${}^{I}_{W}\mathbf{q}$

Figure 6.1: Relationship between World (W^+), IMU (I^+), camera (C^+) and position of the i^{th} feature found in the arUco target. The arUco reference frame is considered to be the world frame W^+.

$$\begin{bmatrix} h_x & h_y & h_z \end{bmatrix}^{\mathrm{T}} = {}^{C}\mathbf{p}_{F_i} = {}^{C}_{I}\mathbf{C}\big({}^{I}_{W}\mathbf{C}({}^{W}\mathbf{p}_{F_i} - {}^{W}\mathbf{p}_I) - {}^{I}\mathbf{p}_C\big) \;, \tag{6.42}$$

where h_x, h_y and h_z represent the components of the ${}^{C}\mathbf{p}_{F_i}$ vector. Once the position vector for a feature is constructed, this can be translated into a projected pixel via the pinhole camera model:

$$\tilde{\mathbf{y}}_{F_i} = \begin{bmatrix} \tilde{u}_i \\ \tilde{v}_i \end{bmatrix} = \begin{bmatrix} f_x(h_x/h_z) + c_x \\ f_y(h_y/h_z) + c_y \end{bmatrix}_i + \mathbf{v}_{F_i} \;. \tag{6.43}$$

The term $\mathbf{v}_{F_i}$ in the pinhole camera model of Equation (6.43), represents a white noise zero-mean Gaussian process with covariance matrix $\mathbf{R}_{F_i} = E[\mathbf{v}\mathbf{v}^{\mathbf{T}}]$ that corrupts the pixel measurements. The expectation of the pinhole camera model is

$$\hat{\mathbf{y}}_{F_i} = \mathbf{E}[\tilde{\mathbf{y}}(\delta\mathbf{x}, \hat{\mathbf{x}})] = \begin{bmatrix} \hat{u}_i \\ \hat{v}_i \end{bmatrix} = \begin{bmatrix} f_x(\hat{h}_x/\hat{h}_z) + c_x \\ f_y(\hat{h}_y/\hat{h}_z) + c_y \end{bmatrix}_i \;. \tag{6.44}$$

To obtain the measurement matrix $\mathbf{H}$, as described at the beginning of Section 6.1.3.3, the measurement residual is formed and then the first-order Taylor series approximation is performed. Since the residual will be a function of $\delta^{C}\mathbf{p}_{F_i}$, and in turn, $\delta^{C}\mathbf{p}_{F_i}$ is a function of $\delta\mathrm{x}$, the linearized model is obtained via chain rule as

$$\mathbf{H}_{F_i} = \frac{\partial \mathbf{r}_{F_i}}{\partial \delta^{C}\mathbf{p}_{F_i}} \left[\frac{\partial \delta^{C}\mathbf{p}_{F_i}}{\partial \delta\mathbf{x}}\right] \;. \tag{6.45}$$

Calculation of the partial derivatives for the F_i feature gives

$$\frac{\partial \mathbf{r}_{F_i}}{\partial \delta^{C}\mathbf{p}_{F_i}} = \frac{1}{\hat{h}_z} \begin{bmatrix} f_x & 0 & -f_x\hat{h}_x/\hat{h}_z \\ 0 & f_y & -f_y\hat{h}_y/\hat{h}_z \end{bmatrix}_i \;, \tag{6.46}$$

and

$$\frac{\partial \delta {}^C\mathbf{p}_{F_i}}{\partial \delta \mathbf{x}} = [{}^C_I\hat{\mathbf{C}}\lfloor\big({}^I_W\hat{\mathbf{C}}({}^W\hat{\mathbf{p}}_{F_i} - {}^W\hat{\mathbf{p}}_I)\big)\times\rfloor, -{}^C_I\hat{\mathbf{C}}\,{}^I_W\hat{\mathbf{C}}, \mathbf{0}_{3\times3}, \mathbf{0}_{3\times3}, \mathbf{0}_{3\times3}, \tag{6.47}$$

$$- {}^C_I\hat{\mathbf{C}}, \lfloor {}^C_I\hat{\mathbf{C}}({}^I_W\hat{\mathbf{C}}({}^W\hat{\mathbf{p}}_{F_i} - {}^W\hat{\mathbf{p}}_I) - {}^I\hat{\mathbf{p}}_C)\times\rfloor] \,. \tag{6.48}$$

As more than one feature may be available at the update step, the measurement matrices for m individual features can be vertically stacked to process all features in a batch manner. However, the measurements also can be processed sequentially (Jacobians need to be re-evaluated after each measurement assimilation). For batch measurement processing, the measurement matrix is constructed as

$$\mathbf{H}_F = \begin{bmatrix} \mathbf{H}_{F_1} \\ \vdots \\ \mathbf{H}_{F_i} \\ \vdots \\ \mathbf{H}_{F_m} \end{bmatrix} , \tag{6.49}$$

and the residual vector as

$$\mathbf{r}_F = \begin{bmatrix} \tilde{\mathbf{y}}_{F_1} - \hat{\mathbf{y}}_{F_1} \\ \mathbf{y}_{F_2} - \hat{\mathbf{y}}_{F_2} \\ \vdots \\ \tilde{\mathbf{y}}_{F_m} - \hat{\mathbf{y}}_{F_m} \end{bmatrix} . \tag{6.50}$$

For the same purposes, the measurement noise covariance can be arranged in a block diagonal matrix as

$$\mathbf{R}_F = \mathrm{diag}[\mathbf{R}_{F_1}, \dots, \mathbf{R}_{F_m}] \,. \tag{6.51}$$

6.1.3.4 The partial-update within the Multiplicative EKF

Once a set of features is received and the measurement matrix, $\mathbf{H}$, is constructed, the partial-update can be executed following the equations from Algorithm 1 presented in Section 2.3.4. To be more specific to the IMU-camera calibration case, the partial-update is done in the following way. First, the $\boldsymbol{\beta}$ matrix is formed

$$\boldsymbol{\beta} = \text{diag}\begin{bmatrix}\beta_{\delta\boldsymbol{\theta}_1} & \beta_{\delta\boldsymbol{\theta}_2} & \beta_{\delta\boldsymbol{\theta}_3} & \beta_{W_{\mathbf{p}_I 1}} & \dots & \beta_{\delta\boldsymbol{\alpha}_3}\end{bmatrix} . \tag{6.52}$$

Then partial-update correction is computed as

$$\delta\mathbf{x}^{++} = \boldsymbol{\beta}\mathbf{K}(\tilde{\mathbf{y}} - \hat{\mathbf{y}}) = \boldsymbol{\beta}\mathbf{K}\mathbf{r} , \tag{6.53}$$

and additive and multiplicative elements are identified as

$$\delta\mathbf{x}^{++} = \boldsymbol{\beta}\mathbf{K}\mathbf{r} = \begin{bmatrix}\boldsymbol{\beta_{\delta\theta}}\delta\hat{\theta}^{+} \\ \boldsymbol{\beta_{additive}}\delta\hat{\mathbf{x}}^{+}_{additive} \\ \boldsymbol{\beta_{\delta\alpha}}\delta\hat{\alpha}^{+}\end{bmatrix} . \tag{6.54}$$

After the partial-update posterior state error, $\delta\mathbf{x}^{++}$, is calculated, the actual state estimates are recovered by following additive and multiplicative error definitions:

$$ {}^{W}\hat{\mathbf{p}}_I{}^{+} = {}^{W}\hat{\mathbf{p}}_I{}^{-} + \boldsymbol{\beta}_{\mathbf{p}_I}\delta{}^{W}\hat{\mathbf{p}}_I{}^{+} , \tag{6.55}$$

$$ {}^{W}\hat{\mathbf{v}}_I{}^{+} = {}^{W}\hat{\mathbf{v}}_I{}^{-} + \boldsymbol{\beta}_{\mathbf{v}_I}\delta{}^{W}\hat{\mathbf{v}}_I{}^{+} , \tag{6.56}$$

$$\hat{\mathbf{b}}_g^{+} = \hat{\mathbf{b}}_g^{-} + \boldsymbol{\beta}_{\mathbf{b}_g}\delta\hat{\mathbf{b}}_g^{+} , \tag{6.57}$$

$$\hat{\mathbf{b}}_a^{+} = \hat{\mathbf{b}}_a^{-} + \boldsymbol{\beta}_{\mathbf{b}_a}\delta\hat{\mathbf{b}}_a^{+} , \tag{6.58}$$

$$ {}^{I}\hat{\mathbf{p}}_C{}^{+} = {}^{I}\hat{\mathbf{p}}_C{}^{-} + \boldsymbol{\beta}_{\mathbf{p}_c}\delta{}^{I}\hat{\mathbf{p}}_C{}^{+} , \tag{6.59}$$

$$ \tag{6.60}$$

The IMU attitude and the IMU-camera attitude offset states become updated according to

$$ {}^{I}_{W}\bar{q}^{+} = \begin{bmatrix}\frac{1}{2}\boldsymbol{\beta}\delta\boldsymbol{\theta}^{+} \\ 1\end{bmatrix} \otimes {}^{I}_{W}\bar{q}^{-} , \tag{6.61}$$

and

$$ {}^{C}_{I}\bar{q}^{+} = \begin{bmatrix} \frac{1}{2}\boldsymbol{\beta}\delta\boldsymbol{\alpha}^{+} \\ 1 \end{bmatrix} \otimes {}^{C}_{I}\bar{q}^{-} \; . \tag{6.62} $$

Finally, the covariance matrix is partially updated via the UD covariance update equations that appear in Table 4.1.

6.1.4 Simulations

In this section, numerical simulations show that the PU-MEKF can perform the IMU-camera calibration achieving sub-centimeter and sub-degree accuracy. The PU-MEKF higher robustness and better consistency are highlighted and compared to those of the conventional MEKF.

The simulation is set such that the IMU-camera system undergoes motion to stimulate all six degrees of freedom. However, the simulated system motion is constrained to ensure that the pinhole camera model predominately has the simulated markers on sight (based on its field of view-FOV). A set of four simulated arUco markers is used to provide a set of 16 available features. The pinhole camera is simulated with a FOV of 58 degrees horizontal and 45 degrees vertically, and a resolution of 640x480 pixels. The camera measurement error standard deviation is considered to be two pixels. The IMU rate is set at 100 Hz, and it is assumed that the image frame rate is 20 Hz. The process noise (IMU noises and bias drift) is taken directly from the VectorNav IMU VN-100 datasheet (this IMU is used in the hardware implementation). The motion history for a single run typically looks like the trajectory shown in Figure 6.2. In this figure, the starting IMU position is indicated with the blue dot.

Again, the simulations intend to highlight the PU-MEKF gain in robustness over the (conventional) MEKF capabilities. In this vein, the filters are stressed by using relatively high initial uncertainty on the global IMU attitude, $\delta\boldsymbol{\theta}$, and IMU-camera attitude error parameters, $\delta\boldsymbol{\alpha}$.

First, the results comparing the MEKF and PU-MEKF outcome for a typical single run under the same motion and conditions, are shown. The initial conditions for the single runs are indicated in Table 6.1, along with relevant IMU-camera calibration parameters. The uncertainties indicated in Table 6.1 represent a 1-σ value. The IMU-camera attitude uncertainty was selected as two times

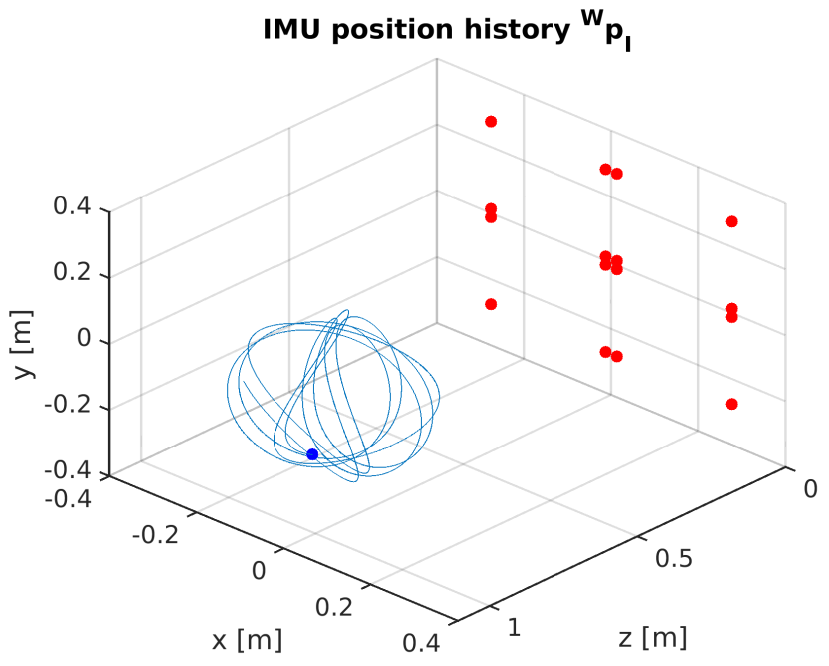
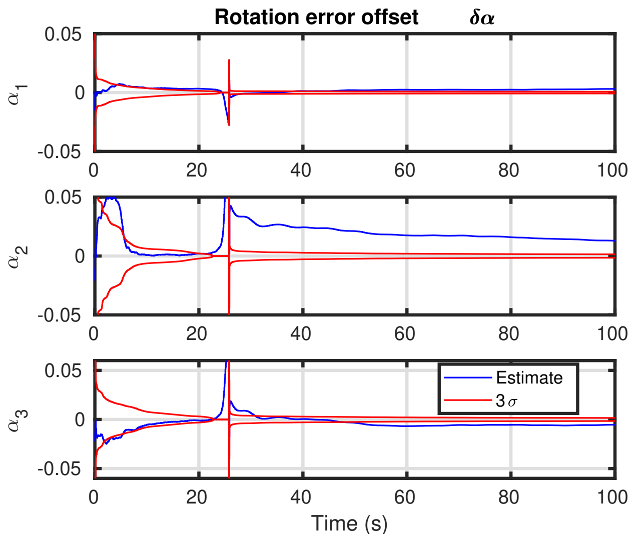


Figure 6.2: Simulated trajectory for a typical IMU-camera calibration run (in blue) and image features (in red).

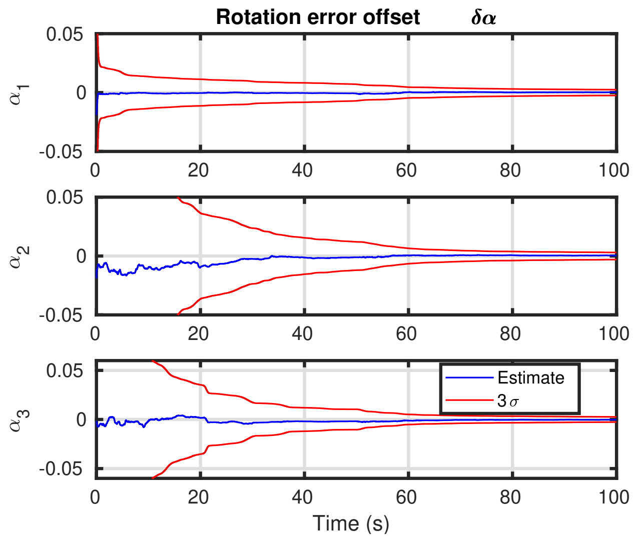
the maximum uncertainty the regular MEKF can handle this scenario without diverging. Although the capabilities of the MEKF for this application are also dependent on simulated motion and simulation parameters, the objective is to show the PU-MEKF robustness gain under the same scenario. It is important to mention that although the initial condition is a random draw from the initial error distribution, both filters use that same random draw as the initial condition.

Table 6.1: IMU-camera calibration parameters

State/parameter	Value
IMU-camera attitude uncertainty	2 deg
Lever arm uncertainty	5 cm
IMU attitude uncertainty	2 deg
IMU position uncertainty	5 cm
Camera frame rate	20 Hz
IMU rate	100 Hz
Camera pixel uncertainty	2 px



(a) MEKF. Units are in radians.



(b) PU-MEKF. Units are in radians.

Figure 6.3: IMU-camera attitude error for a typical simulation run. Initial condition is a random draw. MEKF and PU-MEKF use the same initial condition.

The partial-update weights for this system were set to the following values:

$$\boldsymbol{\beta}_{\delta\theta} = \operatorname{diag}\begin{bmatrix}0.95 & 0.95 & 0.95\end{bmatrix}, \tag{6.63}$$

$$\boldsymbol{\beta}_{\delta {}^W\mathbf{p}_I} = \operatorname{diag}\begin{bmatrix}0.95 & 0.95 & 0.95\end{bmatrix}, \tag{6.64}$$

$$\boldsymbol{\beta}_{\delta {}^W\mathbf{v}_I} = \operatorname{diag}\begin{bmatrix}1 & 1 & 1\end{bmatrix}, \tag{6.65}$$

$$\boldsymbol{\beta}_{\delta\mathbf{b}_g} = \operatorname{diag}\begin{bmatrix}1 & 1 & 1\end{bmatrix}, \tag{6.66}$$

$$\boldsymbol{\beta}_{\delta\mathbf{b}_a} = \operatorname{diag}\begin{bmatrix}1 & 1 & 1\end{bmatrix}, \tag{6.67}$$

$$\boldsymbol{\beta}_{\delta {}^I\mathbf{p}_C} = \operatorname{diag}\begin{bmatrix}0.25 & 0.25 & 0.25\end{bmatrix}, \tag{6.68}$$

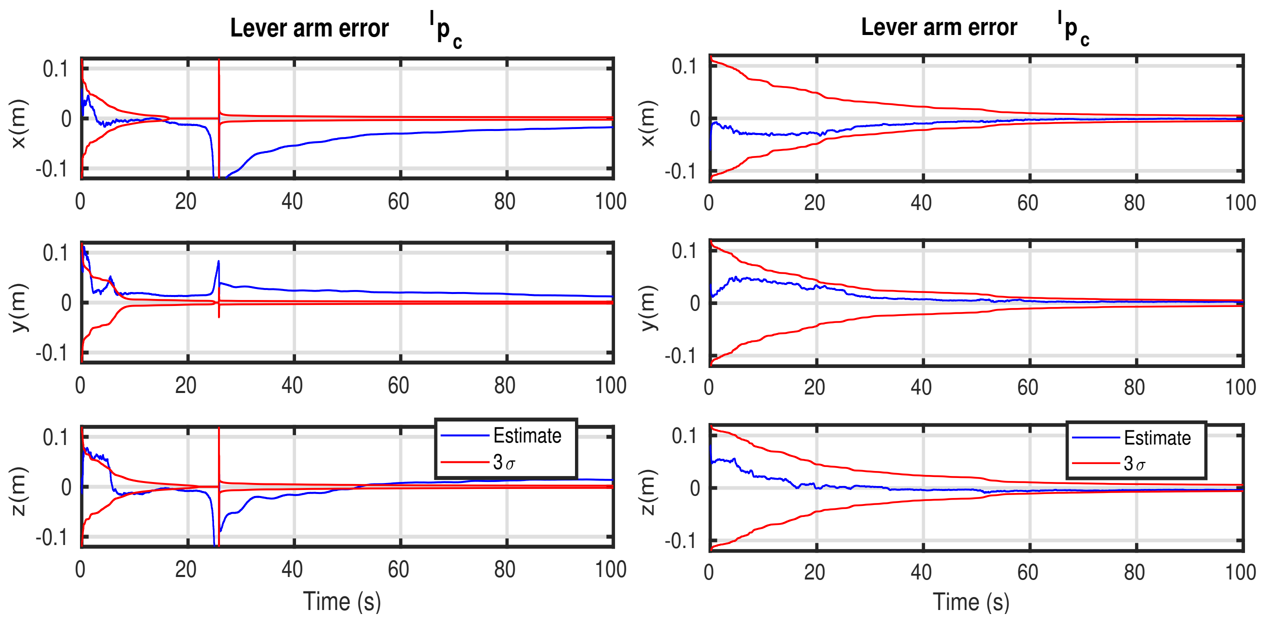
$$\boldsymbol{\beta}_{\delta\alpha} = \text{diag}\begin{bmatrix}0.25 & 0.25 & 0.25\end{bmatrix} . \tag{6.69}$$

Such that,

$$\boldsymbol{\beta} = \text{diag}(\boldsymbol{\beta}_{\delta\theta}, \boldsymbol{\beta}_{\delta^W\mathbf{p}_I}, \boldsymbol{\beta}_{\delta^W\mathbf{v}_I}, \boldsymbol{\beta}_{\delta\mathbf{b}_g}, \boldsymbol{\beta}_{\delta\mathbf{b}_a}, \boldsymbol{\beta}_{\delta^I\mathbf{p}_C}, \boldsymbol{\beta}_{\delta\alpha}) . \tag{6.70}$$

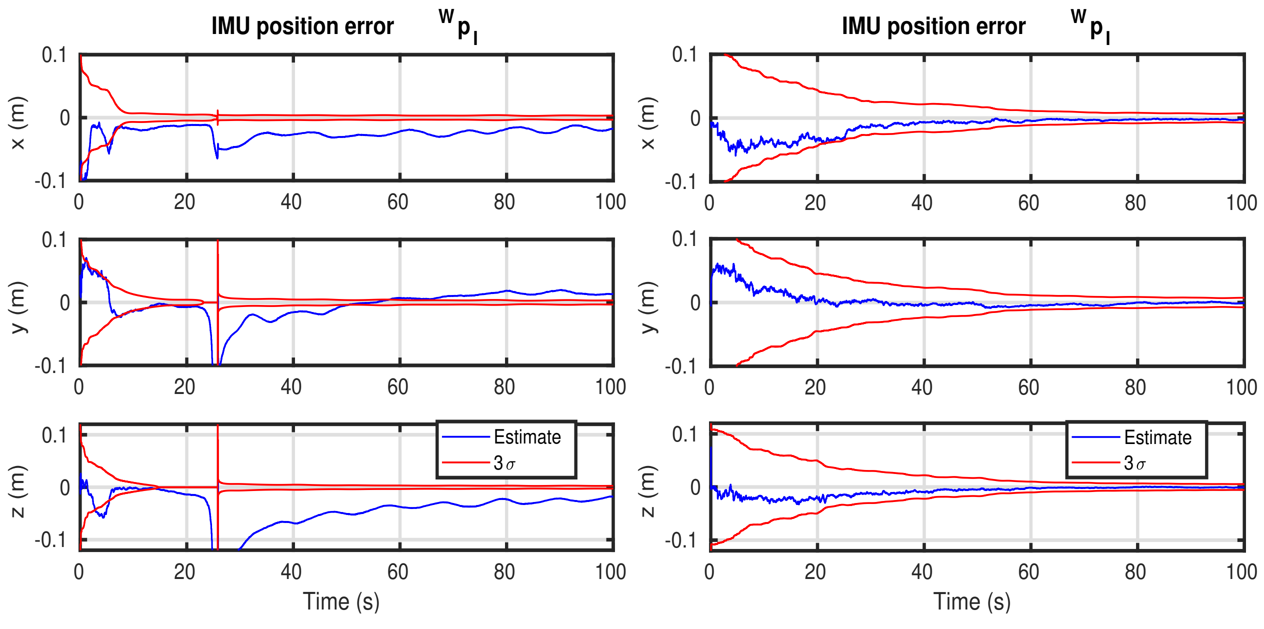
In Figure 6.3, the MEKF estimates for the relative rotation between the camera and IMU are seen to be inconsistent as its estimates are outside the 3σ bounds. This inconsistency stems from the incapacity of the MEKF to handle the relatively high initial uncertainties of this scenario. On the other hand, the PU-MEKF can handle the situation better, and not only do the estimates appear consistent, but the estimation errors are convergent to zero. Effectively, the PU-MEKF reduces initial overreactions of the filter and allows for more proper cross-correlations to be built while information is being gained, and overall helping to prevent divergence. In contrast, the MEKF early on in the simulation attempts large corrections when the uncertainty is high, and the cross-correlations are still being constructed. Although the PU-MEKF converges slowly, it can recover from the initial errors and better handle the nonlinearities than the MEKF. Moreover, once the filter recovers from the initial errors, it produces consistent estimates (estimated error covariance in the filter are well representing the actual errors, and the average error tends to zero as sample size increases), as seen from the 500 Monte Carlo runs in Figure 6.7. Similarly, displayed in Figure 6.4 and 6.5 results for the single run for the lever arm error and the global position estimates, respectively, are seen improved when the PU-MEKF is utilized.

To further investigate the filters, a 500 runs Monte Carlo simulation utilizing the initial uncertainties from Table 6.1 for the random draws, is performed. Figures 6.6 and 6.7 specifically show the position of the IMU with respect to the world frame (${}^I\hat{\mathbf{p}}_W$) (for all 500 runs), which ultimately is what is sought to be improved by finding the IMU-camera calibration. These Monte Carlo results show that the MEKF, from Figure 6.6, it is unable to handle many of the initial conditions, causing navigation errors up to the order of meters or leading to divergence in many others. In contrast, the PU-MEKF shows a more robust behavior and more consistent estimates. In Figures



(a) MEKF (b) PU-MEKF

Figure 6.4: IMU-camera lever arm errors for a typical simulation run. Initial condition is a random draw. MEKF and PU-MEKF use the same initial condition.



(a) MEKF (b) PU-MEKF

Figure 6.5: IMU global position error for a typical simulation run. Initial condition is a random draw. MEKF and PU-MEKF use the same initial condition.

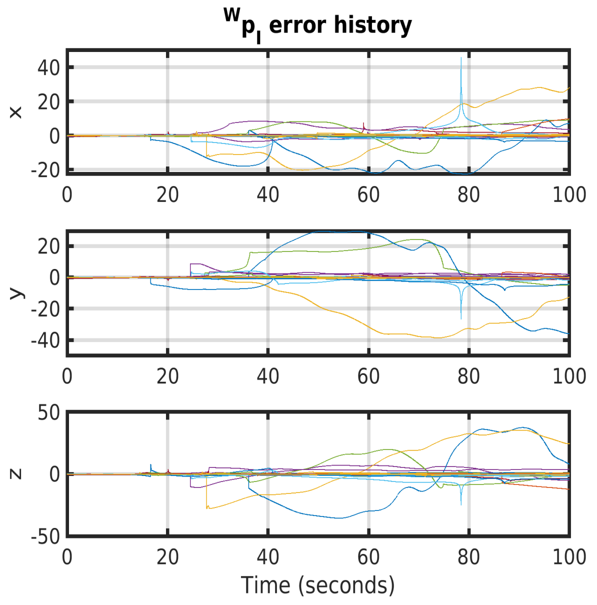
6.6b and 6.7b the averaged standard deviation (computed using the filter covariance estimate) is compared with the sampled standard deviation (computed using the actual estimation errors) from all of the runs. It can be seen that the PU-MEKF estimated covariance, from Figure 6.7b, is such that is very close to standard deviation of the true errors. In fact the PU-MEKF is slightly overconfident, but it is far better than the conventional MEKF results displayed in Figure 6.6b. The PU-MEKF error mean is also plotted in this figure. It also can be seen that early in the simulation, as the filter attempts to estimate the state, the error mean is not precisely around zero (due to the large initial uncertainties, errors, and nonlinearities), however, as the filtering develops the mean of the error tends to zero. The fact that the PU-MEKF standard deviation essentially matches the actual standard deviation of the errors, and that the mean of the estimation errors tends to be zero over time, makes the PU-MEKF a consistent filter for this scenario.

The use of a PU-MEKF for the IMU-camera calibration problem does not remove the need for initializing the filter with proper values, nor suggests that better initialization methods should not be used. Instead, it gives far more room to have bad initial uncertainties while still providing converging results, relaxing the initialization methods and filter requirements. This capabilities of the PU-MEKF, and the fact that it is a sequential filter that runs online, makes it suitable for on-board-online calibration. Finally, these simulations demonstrated that the PU-MEKF could allow the same MEKF structure to broadens its applicability, and specifically for this case, enable its use for IMU-camera calibration. The next section presents the results of its hardware implementation.

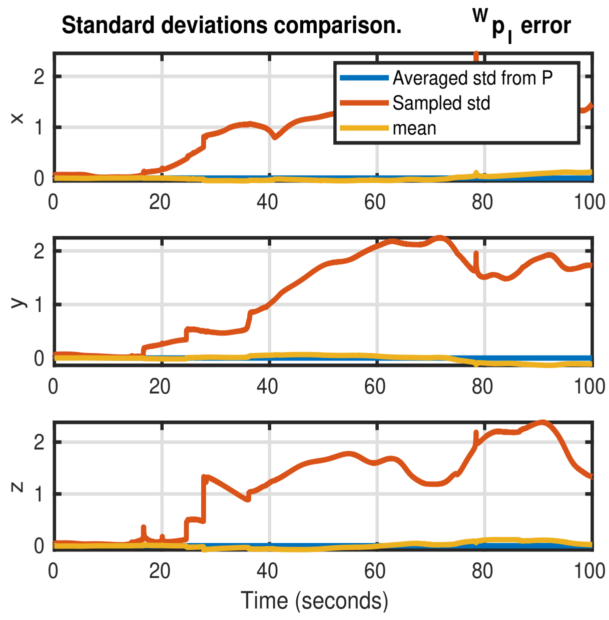
6.1.5 Hardware experiments

6.1.5.1 Setup

The hardware implementation uses the PU-MEKF in UD factorized form. The experiments setup includes an Orbec Astra Pro RGB-D camera, and a VectorNav IMU VN-100 mounted on a rigid support. The arUco target used for calibration includes four markers, providing 16 known features. An Optitrack motion capture system is used to acquire ground reference data for the arUco markers and IMU, but it is not by any means used to validate the approach presented (Optitrack



(a) 500 Monte Carlo runs for IMU global position error (m).

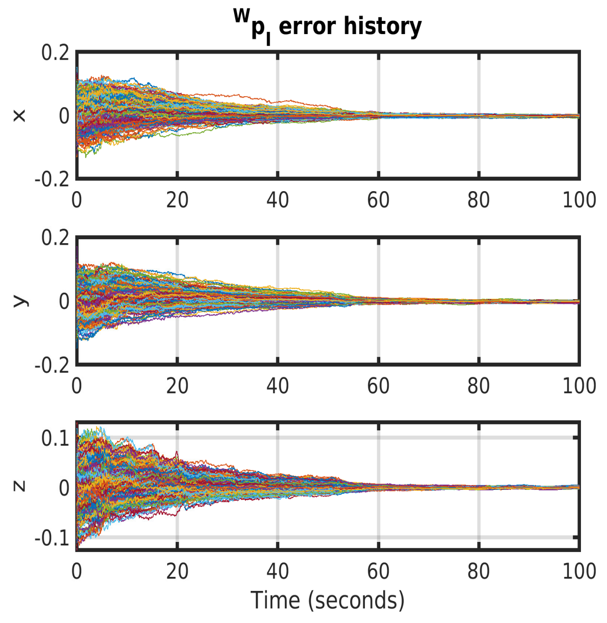


(b) Estimated and actual standard deviation (m^2) for 500 Monte Carlo runs, and filter mean errors (m).

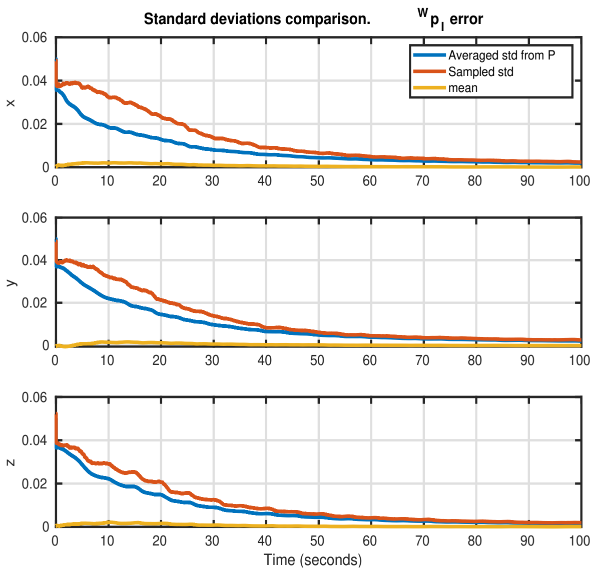
Figure 6.6: Monte Carlo runs for MEKF

beacons are not aligned with the unknown IMU reference orientation). Instead, the re-projected pixel position error consistency, along with calibration parameter results, are used to validate the PU-MEKF approach. The IMU-camera calibration filter uses the partial-update weights of Equations (6.63)-(6.70).

The RGB-D camera images are captured at 20Hz, while the IMU sampling rate is 200 Hz. The calibration filter is run while the IMU-camera system was subject to general motion (as to excite all degrees of freedom) while arUco features are visible 99% of the time. The profile of the position in the hardware experiments tries to mimic those of the simulations (Figure 6.2). Regarding magnitudes of motion velocities, at times, they are as large as the exposure time allows to obtain non-blurred images (maximum 1 m/s). The position amplitudes are as large as possible to allow the motion of the setup (by hand) with ease (approximately motions of up to 60 cm/s of amplitude). Rotational motion covers the range of up to about $\pm$ 45 deg for azimuth and elevation, and up to approximately $\pm$ 90 deg for rotations about the camera axis. Finally, the arUco calibration target is placed vertically and approximately aligned with the gravity vector with the use of the motion



(a) 500 Monte Carlo runs for IMU global position error (m).



(b) Estimated and actual standard deviation (m^2) for 500 Monte Carlo runs, and filter mean errors (m).

Figure 6.7: Monte Carlo runs for PU-MEKF

capture system (calibrated to make one of its axis reference frame align with the gravity vector).

6.1.5.2 Filter Initialization

To initialize the filter, specifically for IMU global position $({}^{I}\mathbf{p}_W)$ and orientation ${}^{I}_{W}\bar{q}$, the following steps are taken. First, the perspective-n-point (PnP) algorithm is executed by using the known features positions and the detected arUco features positions. The PnP solution provides the position of the camera with respect to the world coordinatized in the camera frame, namely $[{}^{W}\mathbf{p}_C]_C$, and the camera attitude with respect to the world frame, ${}^{C}_{W}\mathbf{C}$. Then, as per Figure 6.1 the initial estimate for the IMU position vector is constructed as ${}^{W}\mathbf{p}_I = -{}^{I}_{W}\mathbf{C}^{\mathrm{T}}({}^{I}\mathbf{p}_C + {}^{C}_{I}\mathbf{C}^{\mathrm{T}}[{}^{W}\mathbf{p}_C]_C)$. Similarly, the initial global attitude of the IMU is obtained via ${}^{I}_{W}\mathbf{C} = {}^{C}_{I}\mathbf{C}^{\mathrm{T}}\,{}^{C}_{W}\mathbf{C}$.

Rather than computing the PnP solution once, the initialization procedure averages 20 PnP solutions (Euler angles were used for attitude averaging) while the IMU-camera system remains still. The PnP computation is performed using the `solvePnP` function from OpenCV 3.3.1. Concerning the initial values for the IMU-camera lever arm, they are measured with a ruler for this setup (for the three axis). More complicated setups than the one utilized here may need to rely on 3D-CAD models, or even use additional motion-capture beacons to approximate the IMU-camera lever arm. The IMU-camera relative attitude initialization was simplified by placing the camera and IMU, such that one plane of the IMU reference frame was approximately coincident with a plane of the camera reference frame. In this way, three simple rotations through three angles were eye-balled and rounded to a closed number. Specifically, the IMU to camera attitude initial estimate was constructed by composing three single rotations through Euler angles $[z : 90\ deg, y : 0\ deg, x : 90\ deg]$ (in that order with respect to the local axis). Note that for filter initialization,the quaternion parametrization is extracted from this constructed rotation matrix.

Finally, the initial velocity was set to zero with very low uncertainty, while the biases for gyroscope and accelerometer were initialized with zero values, and their initial covariances are set based on the values included in the VectorNav-100 IMU's datasheet.

6.1.5.3 Results

In this section, the IMU-camera calibration results are presented. As the truth values are not available, the re-projection errors for the features are used as an indicator for calibration accuracy. For all the experiments performed, all IMU degrees of freedom are excited while the arUco features are on sight. The motion trajectories attempt to resemble those from the numerical simulations. The plots for the experiments using a full EKF are not included, but they were either divergent, similar to what is shown in Figure 6.4 or inconsistent as in the results shown in the simulations of chapter 3.

Calibration results for a typical hardware experiment are shown in Figures 6.9-6.11. Figure 6.9 shows the IMU-camera lever-arm having a sub-centimeter accuracy and consistent estimate at the end of the experiment. Although the lever arm calibration is barely sub-centimeter accurate, it is seen that the filter is able to refine the initial estimate and despite the high initial uncertainties (recall that a MEKF is divergent under this scenario). The global IMU position results, which is what ultimately is sought to be improved by refining the IMU-camera parameters, are shown in Figure 6.10. From this figure, it is observed that at the beginning of the experiment, the global

IMU position does not really "track" the Optitrack reference data; instead, it seems to diverge from it. This result is expected because the calibration parameters are initially *unrefined*, in addition, the entire state vector is experiencing transients. By the middle of the run (approximately time $t = 60s$), the filter position estimates and ground reference appear to be more coincident in behavior with the ground reference, indicating that calibration parameters and biases are establishing in more appropriate values. As the experiment progresses towards its end, from Figure 6.9 can be observed that the lever arm evolution reaches a steady-state, while in Figure 6.10 the IMU global position is seen to practically match the motion pattern of the ground reference. Again, although motion pattern matching is expected, this is not considered an indication of appropriate calibration (since motion capture markers and actual IMU frames alignment is unknown). Rather, the measurement residuals are used for consistency check and filter validation. The filter, in fact, shows consistency as from Figure 6.8, where residuals shows to have zero mean, and remain within the 3-σ bounds. Although true values for the calibration parameters are unavailable, the calibration parameters given by the filter were found realistic, and since the residuals were generally small and consistent, the PU-MEKF estimates were considered to be valid. The IMU velocity estimates are also showed agreement with ground truth data as seen in Figure 6.11.

6.1.6 Summary

This hardware application uses a PU-MEKF filter in UD factorization form. Through this application the PU-MEKF was shown to appropiately accommodate constant nuisance parameters in a Schmidt-like form while providing significant robustness against high uncertainty and nonlinearities. All, while retaining a familiar EKF framework.

Numerical results showed that the PU-MEKF could achieve precise IMU-camera calibration while tracking core states. Further, Monte Carlo runs demonstrated that the proposed PU-MEKF is a consistent filter, and although the true parameters are unknown, filter consistency is corroborated by the measurement residuals being relatively small and consistent. Other than requiring exciting all IMU degrees of freedom, the calibration filter needs an arUco target for the feature extraction. Nevertheless, a pattern (either checkerboard or arUco or one of its variants) is often available

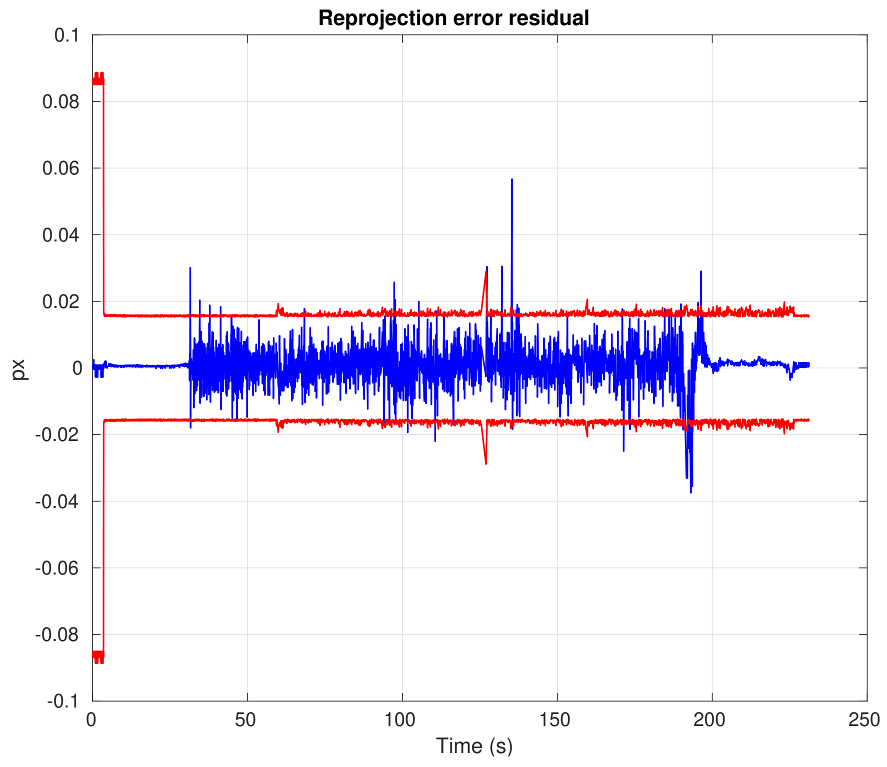


Figure 6.8: Measurement residuals in pixels (px). The residuals indicate filter consistency. These residuals are employed to validate filter results since no true values for the calibration parameters are available.

as they are also used for intrinsic camera calibration. The filter simple initialization method and relatively low computational requirements, make the UD PU-MEKF suitable for on-board implementation, as demonstrated in this application.

6.2 A simplified Unmanned Aerial Vehicle state estimation framework

6.2.1 Introduction and Motivation

For many research projects that use aerial robots, significant effort is dedicated to building and sometimes adapting commercially available components to suit the use case. Furthermore, developing reliable flight software infrastructure diverts considerable amount of time from the main research topic. Attempting to overcome these bottlenecks, a few research groups have opted to use commercial-off-the-shelf (COTS) flight decks such as the PixHawk [91] which is well suited for hobbyist-grade flight and way-point navigation in outdoor environments, but can be challenging to

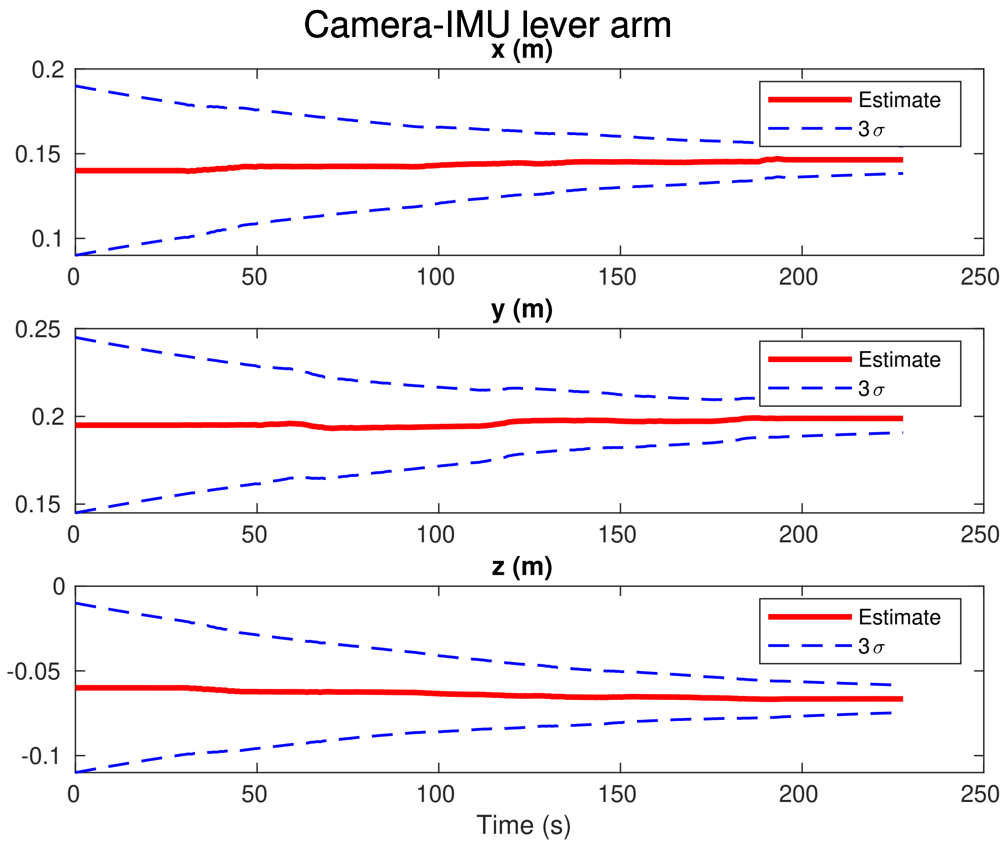


Figure 6.9: Lever arm hardware calibration result. The lever arm is considered to be the position vector of the camera with respect to the IMU reference frame.

augment and customize due to their complex code base. Alternatively, commercial platforms such as the Parrot Bebop [92] and AR Drone [93] are also actively used, but proprietary hardware and software restrict their available functions and behavior. Some research groups have also created their own, very specific navigation solutions [94], [95], [96], but most of them tend to be highly specialized, and modifying them to meet the user needs can be time consuming and there is no guarantee of obtaining favorable results.

In order to mitigate such complications, the Autonomous Vehicle Laboratory (AVL) in the University of Florida's Research and Engineering Education Facility (REEF), and the Land Air and Space Robotics Laboratory (LASR) at Texas A&M University, where researchers have also experienced technical difficulties, in a collaborative effort, have proposed a new starting point to establish flight capabilities: the *REEF Estimator*. The REEF Estimator is an open-source and

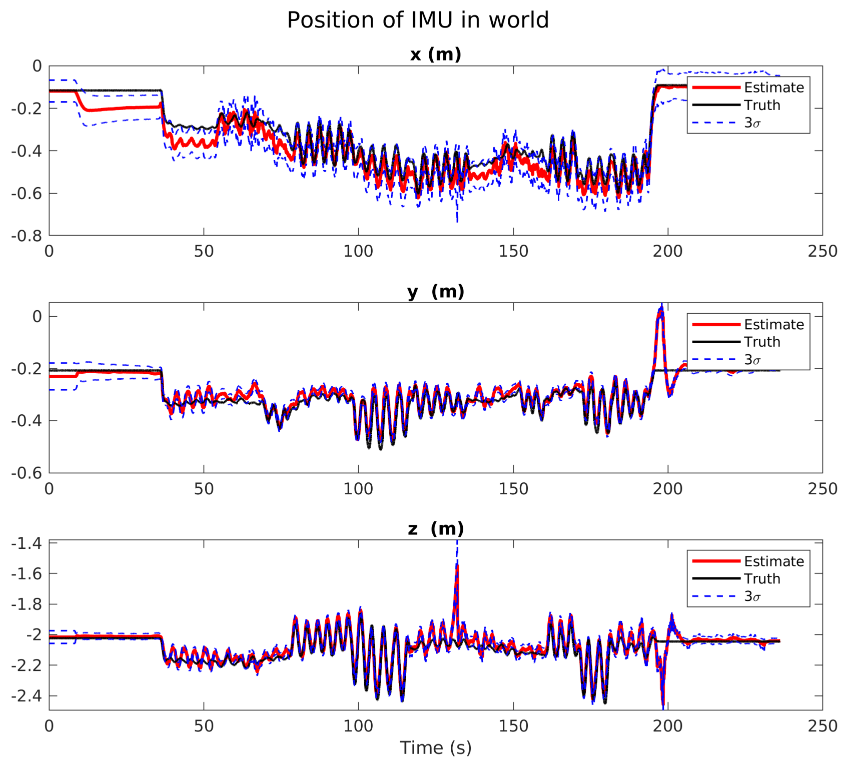


Figure 6.10: Position of the IMU with respect to the inertial frame hardware calibration result.

easy-to-use flight system that allows users to have a vehicle flying in a reliable manner without the need for GPS or motion capture (*for the ease of understanding REEF Estimator refers to the whole system, but this includes the vehicle states estimator and its controller*). The REEF Estimator is not intended to demonstrate state-of-the-art flight capabilities, but rather to be a tractable, functional and easy-to-use implementation that offers new laboratories and students a solid *launching* point for multirotor-based project development. Interested users can find the repository with the code base, flight simulator, hardware list and assembly instructions at `https://github.com/uf-reef-avl/reef_estimator_bundle`.

The attractive features of the REEF estimator are the modularity and simplicity of the estimation approach. The multirotor estimator is broken into more accessible pieces: an attitude filter, a local-level-frame six-state velocity filter, and a three-state altitude filter, along with their associated controllers. Besides, any part can be used independently or replaced if needed. The REEF

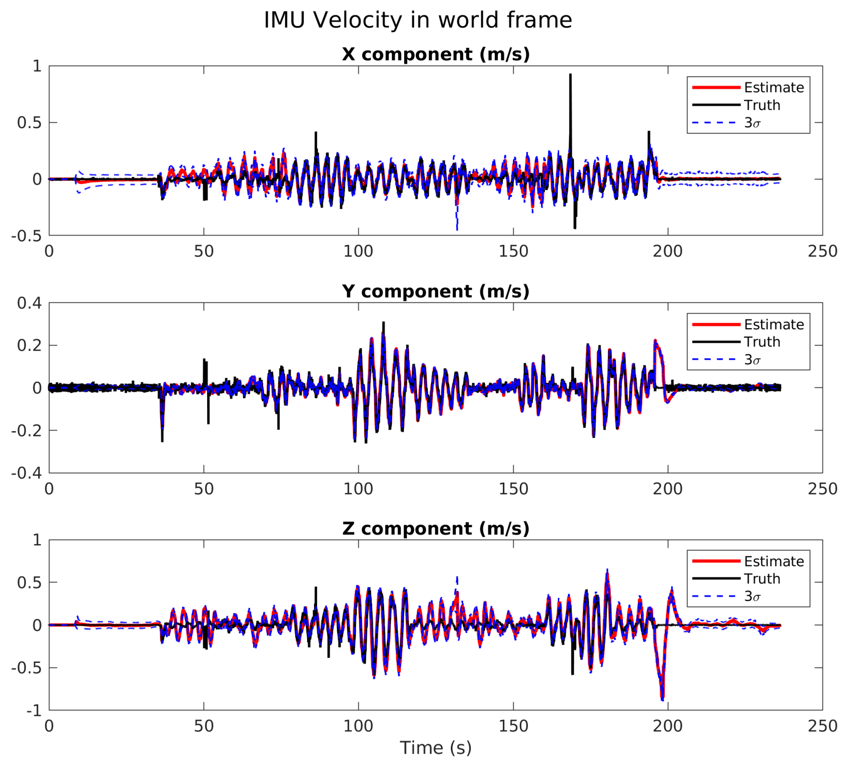


Figure 6.11: Velocity of the IMU with respect to the inertial frame hardware calibration result.

Estimator has been leveraged in several research problems providing a reliable and stable flight for data collection and visual odometry experiments, like the one reported in [97]. Also, the AVL has also used the three-state altitude estimator to introduce students to the EKF, C++ programming, the ROS environment, and even advanced control theory topics, to mention some.

The altitude estimator and controller can be used independently of the lateral velocity estimator to remove throttle control from the hands of less experimented pilots. This setup alone has saved numerous flight vehicles from crashing and makes flying much more straightforward. If desired, the user can use the available REEF Estimator capabilities as-is to, for example, collect data in a more controlled flight setting (this supports several image processing efforts that require flights commensurate with stable autonomous vehicles), or exercising SLAM algorithms that generate maps and desired trajectories below allowable control rates. Laboratories at partnering universities are currently using this setup to support their research experiments. “Out of the box," the REEF

Estimator consists of a modular open-source code base written using the ROS framework [98], COTS components, and extensive documentation that makes it accessible to users of all levels of expertise.

It is important to mention that the REEF estimator's simplicity and functionality rely entirely on the application of the partial-update filter. The REEF estimator uses a UD partial-update filter to be specific. In contrast to the conventional EKF, the partial-update approach was able to maintain a consistent filter. It mainly prevents the mildly observable states IMU biases, the nuisance parameters for this application, from being over updated (in this case mainly due to model simplification) but still to estimate them. Consequently, allowing the filter core states to settle in more appropriate state and uncertainty values that produced a well-behaved estimator. The use of a conventional Schmidt or consider filter was not appropriate since it was unable to cope with the slowly-changing biases, especially for long flights.

The description of this hardware application is organized as follows. Section 6.2.2 gives an overview of the proposed solution. Sections 6.2.3 and 6.2.4 discuss the details on the estimator and controller, respectively. The experiments used to validate the estimator and controller are discussed in Section 6.2.5. Finally, Section 6.2.6 includes a summary of the hardware implementation.

6.2.2 High-Level System Overview

The REEF Estimator is based on an existing attitude estimator available through the ROSFlight [99] autopilot flight deck and a UD partial-update extended Kalman filter with a simplified model for local-level frame velocities and vehicle altitude estimation. Standard PID controllers are used to stabilizing the platform. The simplified state dynamics and controllers are chosen to keep the code readable and straightforward to implement, and while the proposed solution is not optimized in any way, the final product is intuitive to fly and easy to modify if needed. Figure 6.12 shows a block diagram of the framework presented in this section.

6.2.2.1 Autopilot

The REEF estimator uses ROSFlight for attitude estimation. ROSFlight, which inherits the flexibility of ROS, is essentially a plug-and-play autopilot system. It runs on a Flip32 board that also serves as a physical interface between actuators and the on-board computer. Effectively, ROSFlight directly uses the data from the Flip32's on-board IMU to provide attitude estimates. The data coming from other sensors (e.g., sonar) connected to the Flip32 board are also available through the ROS interface (published by ROSFlight). In Figure 6.12, ROSFlight, and the Flip32 board are drawn inside the same box to denote them as the autopilot system. The REEF Estimator that runs on the on-board computer uses the sensor data and the ROSFlight attitude estimate to generate the velocity and altitude estimates of the vehicle.

6.2.2.2 On-board computer

The estimators and controllers are designed and implemented to be executed on an on-board computer like Odroid or Intel NUC. The REEF Estimator (consisting of the estimators and controllers) is drawn in Figure 6.12 as independent white boxes. Although the altitude and velocity estimator are contained in the same module, the user can use or replace them independently as required. As mentioned before, this can be useful if the human pilot desires to take control on the local-frame, leaving the altitude to be controlled by the computer. This mode is the most used at the AVL and LASR laboratories. This hardware implementation uses a commercial RGB-D sensor (to obtain velocity measurements indirectly) and a sonar (for altitude) to demonstrate the REFF estimator's functionality.

Inside the on-board computer box in Figure 6.12, the block called RGB-D Odometry receives RGB-D measurements (color images and depth information) and uses them to obtain velocity measurements. These measurements are then fed into the REEF Estimator. The details on how velocity estimates are generated from RGB-D data are given in Section 6.2.3.4. Finally, a block called velocity reference is also shown in Figure 6.12. This block denotes that the on-board computer can also receive high-level velocity commands from an external source, which can be very useful if a third-party algorithm is generating the guidance information.

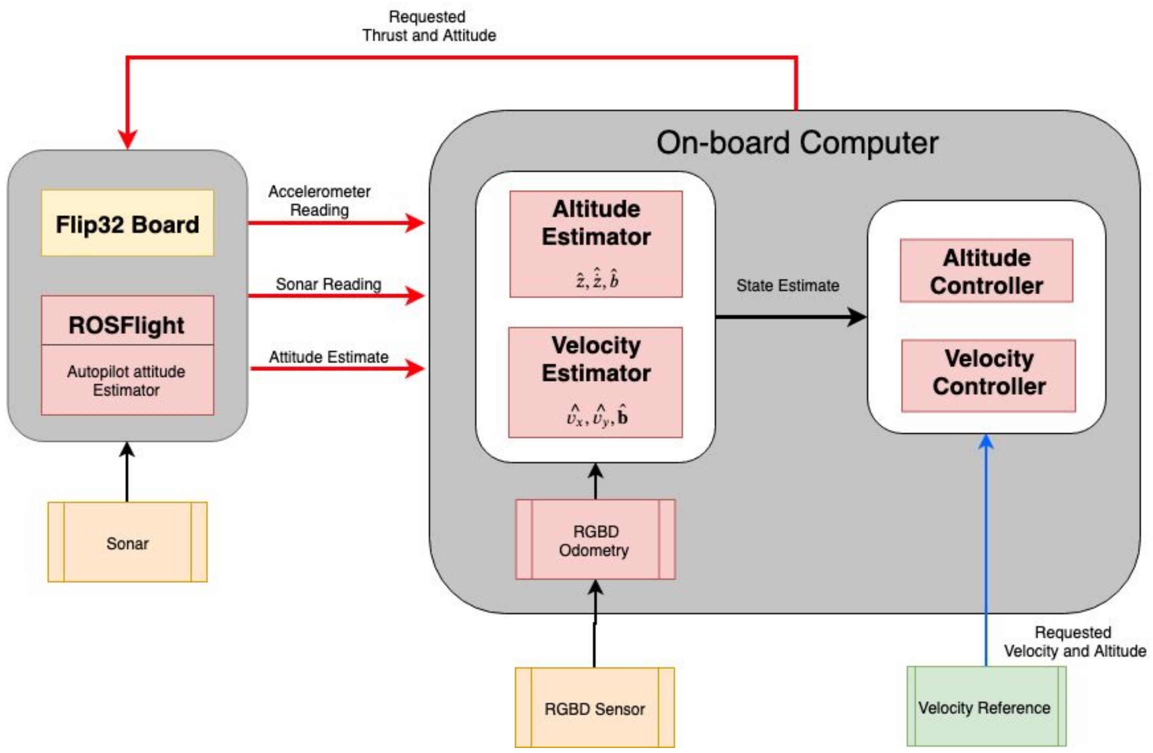


Figure 6.12: Block diagram of the REEF Estimator framework. Reprinted with permission from [2].

6.2.3 Estimator Design

The REEF Estimator framework uses a UD partial-update EKF for XY local-level velocity estimator and a UD linear Kalman filter for the altitude Z. Local-level refers to the fact that the XY velocity estimation is performed in a body-fixed frame in which the XY plane is parallel to the XY plane of the inertial frame as pictured in Figure 6.13. The local-level frame is also referred to as the body-level frame in this implementation. From here on, the x and y velocity estimator is referred as the XY estimator, and the altitude estimator as the Z estimator.

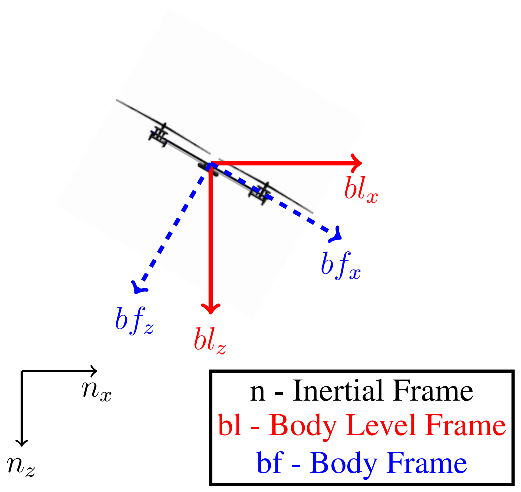


Figure 6.13: The body and body-level frames are co-located and share the same yaw angle. The XY plane in the body-level frame is parallel to the inertial frame XY plane. Reprinted with permission from [2].

6.2.3.1 Propagation of the Z estimator

As illustrated in Figure 6.12, REEF Estimator relies on the attitude estimates coming from the sensors, ROSFlight autopilot, and the IMU readings coming from the Flip32. This data is used for the propagation step. The estimator performs Euler integration to propagate the state vector. The Z estimator state vector is comprised as follows:

$$\hat{\mathbf{x}}_z = \begin{bmatrix} \hat{z} & \hat{\dot{z}} & \hat{b}_{a_z} \end{bmatrix}^T . \tag{6.71}$$

Where $\hat{z}$, $\hat{\dot{z}}$ and $\hat{b}_{a_z}$ is the altitude, vertical velocity and the accelerometer bias (in the direction of gravity) estimates, respectively. The Z estimator outputs are expressed in the gravity aligned inertial frame (positive direction being downwards). The altitude estimator process model is comprised as

$$\dot{\hat{\mathbf{x}}}_z = \begin{bmatrix} \hat{\dot{z}} \\ \tilde{a}_z + \hat{b}_{a_z} - g \\ 0 \end{bmatrix} , \tag{6.72}$$

where ,$\tilde{a}_z$, is the short notation for the third element in the vector $[\tilde{\mathbf{a}}_z]_n$, that represents the accelerometer measurement in the z direction expressed in the inertial frame (n denoting the inertial frame). Specifically, $\tilde{\mathbf{a}}_z$ is obtained by

$$[\tilde{\mathbf{a}}_z]_n = {}^{b}_{n}\mathbf{C}^{\mathrm{T}}[\tilde{\mathbf{a}}_z]_b \, . \tag{6.73}$$

Here, ${}^{b}_{n}\mathbf{C}$ is the attitude matrix given by the ROSFlight autopilot. The ${}^{b}_{n}\mathbf{C}$ is considered to be known without uncertainty, but notice that attitude bias states will be used to address autopilot attitude errors in the XY estimator (discussed in subsection 6.2.3.3). Finally, g is the magnitude of the acceleration due to gravity. Covariance state propagation occurs according to the UD partial-update filter equations from Table 4.1 from Chapter 4.

6.2.3.2 Measurement update for the Z estimator

The Z estimator implementation considers that direct altitude measurements are available, thus the measurement matrix is $\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$. The measurement noise covariance, $\mathbf{R}$, is selected according to the sensor used. The measurement update is performed according to the UD partial-update filter equations from Table 4.1 from Chapter 4.

6.2.3.3 Propagation of the XY estimator

The XY velocity estimator consists of six states with the following state vector:

$$\hat{\mathbf{x}}_{xy} = \begin{bmatrix} \hat{v}_x & \hat{v}_y & \hat{b}_{q_x} & \hat{b}_{q_y} & \hat{b}_{a_x} & \hat{b}_{a_y} \end{bmatrix}^T \, . \tag{6.74}$$

Where $\hat{v}_x$ and $\hat{v}_y$ are the velocities in the x and y directions in the body-level frame; $\hat{b}_{q_x}$ and $\hat{b}_{q_y}$ are the roll and pitch attitude bias; and $\hat{b}_{a_x}$ and $\hat{b}_{a_y}$ are the accelerometer biases in the x and y directions in the body-level frame. The reason for including the attitude bias states, $\hat{b}_{q_x}$ and $\hat{b}_{q_y}$, is to compensate for errors in the ROSFlight attitude estimate. The process model for the XY velocity estimator is the following:

$$\hat{\dot{\mathbf{x}}}_{xy} = \begin{bmatrix} {}^{bl}_{b}\mathbf{C}_{2\times3}(\tilde{\mathbf{a}} + \hat{\mathbf{b}}_a + {}^{b}_{n}\mathbf{C}\mathbf{g}) \\ \mathbf{0}_{3\times1} \\ \mathbf{0}_{3\times1} \end{bmatrix}. \tag{6.75}$$

Here, ${}^{bl}_{b}\mathbf{C}$ is the attitude matrix that takes a vector represented in the body-fixed frame to its representation in the body-level frame. The term ${}^{bl}_{b}\mathbf{C}_{2\times3}$ is the 2×3 top-left block of ${}^{bl}_{b}\mathbf{C}$ matrix, hence ${}^{bl}_{b}\mathbf{C}_{2\times3}(\tilde{\mathbf{a}} + \hat{\mathbf{b}}_a + {}^{b}_{n}\mathbf{C}\mathbf{g})$ is a 2×1 acceleration vector in the body-level frame (acceleration in the x and y directions). The two zero vectors, $\mathbf{0}_{3\times1}$, are the result of considering the biases as random walks with zero-mean Gaussian distribution. Equations (6.72) and (6.75) are directly used for state propagation, whereas the Jacobian matrix $\mathbf{F}_k = \frac{\partial \mathbf{f}_k}{\partial \mathbf{x}}\Big|_{\hat{\mathbf{x}}_{k-1}^{-}}$ is computed at every time-step for propagating the covariance matrix. Notice that the rotation matrices from Equation (6.75) are given by ROSFlight, thus its notation omits the hat tilde.

The simplification of the kinematics (with respect to a dynamical model) is apparent from (6.75). The fact that some modeling details like vehicle physics have not been considered in the process model may produce less accurate estimates than a complete model system. However, as pointed out before, the intention is not to create the perfect filter but rather a simple filter that is easy to understand and modify. The shortcoming is acknowledged through the modularity of the code. If a particular use-case (like rapid motion) cannot be executed due to some of the modeling assumptions, the user can replace the model with a more suitable one, without disrupting the central architecture of the REEF Estimator. In fact, the modularity has been tested by exchanging the UD partial-update EKF based REEF Estimator with the square root form developed in Chapter 3. The whole process involved a swap of files and changes to few lines of code.

6.2.3.4 Measurement update for the XY estimator

The XY filter relies on body-level velocity measurements. For the REEF Estimator code base the algorithm described in [100] is used to obtain delta-pose estimates from RGB and depth images. The delta-pose is then numerically differentiated and transformed from the camera frame to the body-level frame. This transformation is straightforward since attitude estimates are readily available from ROSFlight. Since level-frame velocity measurements are generated, the $\mathbf{H}$ matrix is considered as $\mathbf{H} = [\mathbf{I}_{2\times2}, \mathbf{0}_{2\times6}]$. A fixed measurement error covariance was chosen experimentally.

6.2.3.5 *Partial-update filter modification*

Due to the use of Euler integration on attitude and accelerometer, apparent mismodelling of the system, nonlinearities and nuisance parameters present in the system, running the REEF Estimator with a conventional EKF was no functional. For some flight data the traditional EKF even divergent. Overall, the filter produced completely unrealistic covariance values for the velocity, and thus statistically inconsistent estimates. The first attempt at mitigating this behavior was to increase the process noise covariance (especially for the biases), however this led to poor performance when there were measurement outages, plus the estimates tended to be highly conservative. As an alternative solution, the partial-update Kalman filter was applied. For numerical robustness, the UD partial-update version was used.

Following the notation introduced in previous chapters, partial-update weights for the XY velocity estimator were selected to be close to

$$\begin{aligned}\boldsymbol{\beta_{XY}} &= \operatorname{diag}\begin{bmatrix}\beta_x & \beta_y & \beta_r & \beta_p & \beta_{b_{a_x}} & \beta_{b_{a_y}}\end{bmatrix} \\ &= \operatorname{diag}\begin{bmatrix}1 & 1 & 0.01 & 0.01 & 0.01 & 0.01\end{bmatrix} .\end{aligned} \tag{6.76}$$

Using this partial-update configuration the vehicle to fly in a stable, reliable, and repeatable manner. In other words, a 100% update for body-level frame x and y velocities was applied, whereas a 1% update on the bias attitude (roll β_r and pitch β_p) and accelerometer (b_{a_x} and b_{a_y}) updates was used. Also with an appropriate flying behavior in mind, the altitude estimator Z was set to use percentage values of $\boldsymbol{\beta_Z} = diag\begin{bmatrix}\beta_z & \beta_{\dot{z}} & \beta_{b_z}\end{bmatrix} = \operatorname{diag}\begin{bmatrix}1 & 1 & 0.5\end{bmatrix}$. That is, the updates percentages were 100 % on z position and z velocity, while a 50% update was applied to the bias state.

6.2.3.6 The XY and Z filters operation frequency

Since the accelerometer measurements and attitude estimates are used to propagate the system Equation (6.72) and (6.75), the REEF Estimator runs at the rate at which the IMU delivers measurements. The measurement update on the other hand, is executed at measurement arrival. With the hardware setup presented here, the propagation step is executed at 500Hz while the updates are performed at 40 Hz for sonar and 20 Hz for the RGB-D measurements.

6.2.4 Controller Design

The REEF Estimator uses a cascaded Proportional-Integral-Derivative (PID) control implemented within the ROS environment. The PID controller is well established in the controls community and has been been widely used to control UAVs [101], [102]. Figure 6.14 shows how the cascading in the PID controller occurs (e.g. computations from $z_{request}$ to $\dot{z}_{request}$ to $\ddot{z}_{request}$). The PID controllers in the REEF Estimator receive altitude and body-level velocity requests along with the state estimates from the on-board computer. Then the altitude controller maps the altitude error (difference between the altitude request and current altitude) into a vertical velocity setpoint, which in turn is mapped to a thrust setpoint using the z velocity estimate. The x and y velocity requests are mapped to attitude setpoints (pitch and roll, respectively) based on the error with respect to the velocity estimates. The yaw-rate requests are directly fed to the ROSFlight deck. The error calculations are performed according to the equations that appear in Figure 6.14. The mapping from error (difference between request and state estimate) into attitude and thrust setpoints is performed via the standard PID control law from the following equation:

$$u_k = K_p e_k + K_i \sum_{i=0}^{k} e_i \Delta t_i + K_d \frac{e_k - e_{k-1}}{\Delta t} \ , \tag{6.77}$$

where e denotes the error value, and u is the control signal to be applied (attitude or thrust). K_p, K_i, and K_d denote the proportional, integral, and derivative gains, respectively, and Δt is the time interval between estimates. In Figure 6.14, the controller structure and flow of data through the system is shown. It is worth noting from the bottom box (where the attitude commands θ

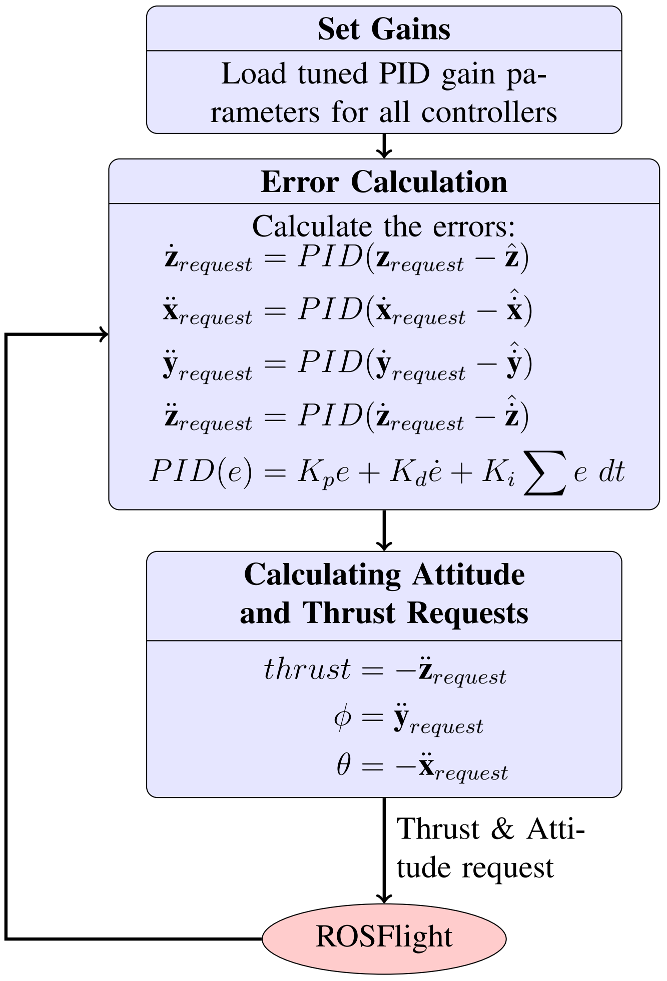

Set Gains
Load tuned PID gain parameters for all controllers
Error Calculation
Calculate the errors:
$\dot{\mathbf{z}}_{request} = PID(\mathbf{z}_{request} - \hat{\mathbf{z}})$
$\ddot{\mathbf{x}}_{request} = PID(\dot{\mathbf{x}}_{request} - \hat{\dot{\mathbf{x}}})$
$\ddot{\mathbf{y}}_{request} = PID(\dot{\mathbf{y}}_{request} - \hat{\dot{\mathbf{y}}})$
$\ddot{\mathbf{z}}_{request} = PID(\dot{\mathbf{z}}_{request} - \hat{\dot{\mathbf{z}}})$
$PID(e) = K_p e + K_d \dot{e} + K_i \sum e\ dt$
Calculating Attitude and Thrust Requests
$thrust = -\ddot{\mathbf{z}}_{request}$
$\phi = \ddot{\mathbf{y}}_{request}$
$\theta = -\ddot{\mathbf{x}}_{request}$
Thrust & Attitude request
ROSFlight

Figure 6.14: Flowchart for the XY and Z PID controllers. Reprinted with permission from [2].

(pitch), ϕ (roll), and thrust request are calculated) that the thrust and pitch values are multiplied by a minus sign. This is just because the design of this platform follows the North-East-Down (NED) reference frame convention.

The controller framework within the REEF estimator was designed to be completely modular, with any software component, input, or output being replaceable or re-configurable with relative ease. For example, the velocity, altitude and yaw rate requests, can come from an RC or a gaming controller, or a high-level path planning algorithm. Additionally, the controller supports multiple modes of operation that intend to facilitate flight tasks. If users desire to use the system for data collection, using the altitude-hold mode would be recommended as they only have to focus on commanding motion on the XY plane. This alone is a big benefit when compared to flying with an RC transmitter commanding thrust and attitude simultaneously.

The current setup of cascaded PID control, while not robust to disturbances or capable of agile flight, is adequate for the intended purpose of expediting the process of establishing flying capabilities. Nevertheless, if the need for a different controller arises, the current code architecture enables its easy integration or substitution.

6.2.5 Hardware implementation

The REEF Estimator was exercised on a hardware setup that uses a commercial S500 quadrotor frame, along with its corresponding brushless DC motors and motor controllers (ESCs). The RGB-D images used to estimate x and y velocity, come from an Orbec Astra Pro camera, while a MaxBotix MB1242 sonar provides altitude measurements. Since the purposes of the vehicle may vary from simple flight data collection, high load SLAM algorithms evaluation, to new control laws or estimators development, an Intel NUC computer with an i5 processor was incorporated. This on-board computer runs the Ubuntu 16.04 operating system and ROS Kinetic. Figure 6.15 shows a picture of the hardware setup utilized to generate the plots presented in this section. Motion capture markers were placed on the multirotor body in order to obtain ground truth data for validation only. However, motion capture data can easily replace velocity or altitude measurements or both if desired.

6.2.5.1 Flight performance

The results presented in this section are from a single flight in altitude-hold mode. The PID controller uses the REEF Estimator states as feedback while data from an OptiTrack motion capture system is recorded for ground truth reference. In this setup the PID controller receives velocity requests from a Logitech F710 gamepad. The appendix A contains the values of all the parameters used by REEF Estimator when generating the plots shown in this section.

Figure 6.16 shows a representative fragment of the XY velocity and Z altitude estimates from a flight. The true velocity (position numerically differentiated) from the motion capture system

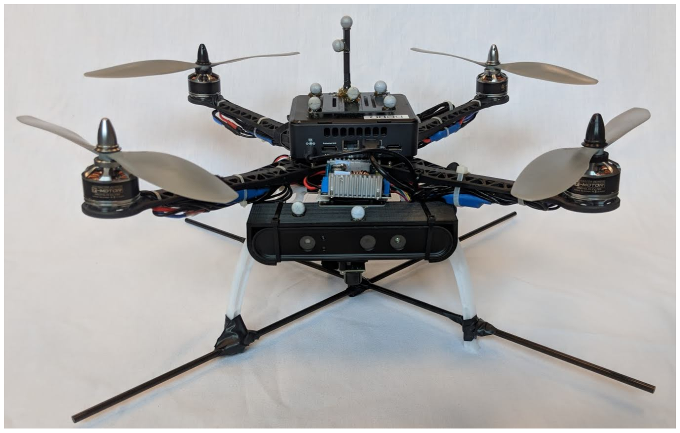
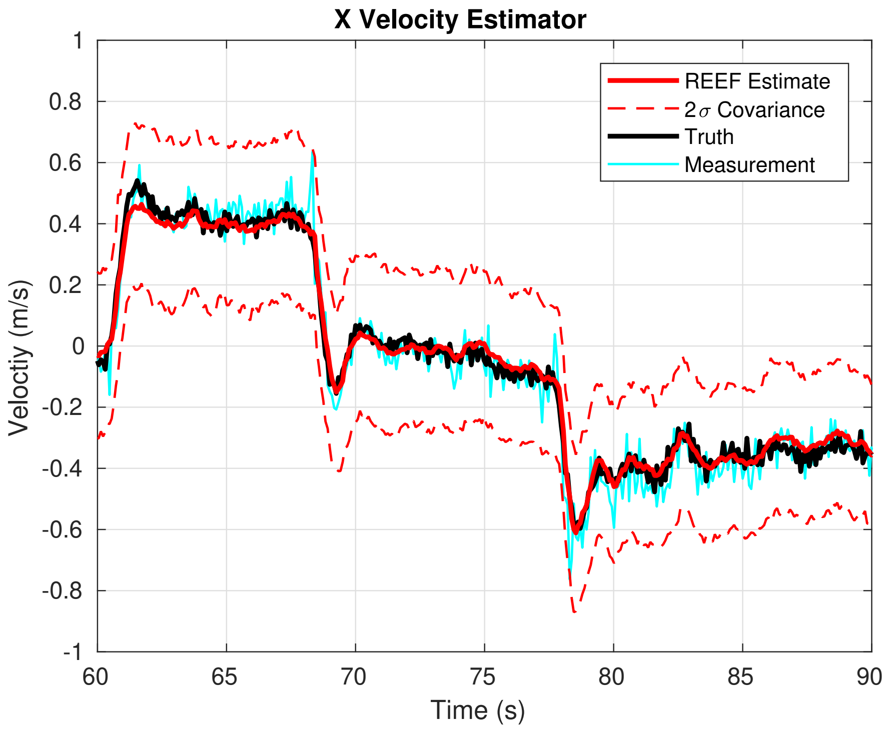


Figure 6.15: Experimental quadrotor platform. Reprinted with permission from [2].

and the RGB-D velocity measurements are also included in the plot. From Figure 6.16a, it can be observed that although the estimates are somewhat conservative, they track the ground truth very closely. The velocity filter in the y direction, as depicted in Figure 6.16b exhibits accurate tracking as well, and the estimated covariance appears lower than the x direction suggesting higher observability in the y (lateral) direction. The fact that the altitude is directly measured is reflected in the altitude filter estimates quality, as they track the ground truth very precisely, as observed in Figure 6.16c. Overall, the UD partial-update filter is able to handle this scenario with varying nuisance parameters well, providing appropriate estimates for the core that allow stable and repeatable flight. Again, although a Schmidt filter was also implemented, it could handle the drifting biases (nuisance parameters) in general, producing divergent estimates in most of the experiments using the described hardware setup.

The tracking performance of the PID controller is shown in Figure 6.17. Although the command tracking is far from being perfect for the x and y velocity controllers (as seen in Figures 6.17a and 6.17b, respectively), the intention is to provide a solid launching point for stable and repeatable flight. In Figure 6.17c, the altitude controller performance is shown to be able to track the reference accurately. The first 20 seconds of Figure 6.17c show the quadrotor to be on the ground waiting for the pilot to command the take-off. The multirotor starts tracking the reference of -1 meter once in the air (recall that NED frame is used). Notice that when the vehicle is on the

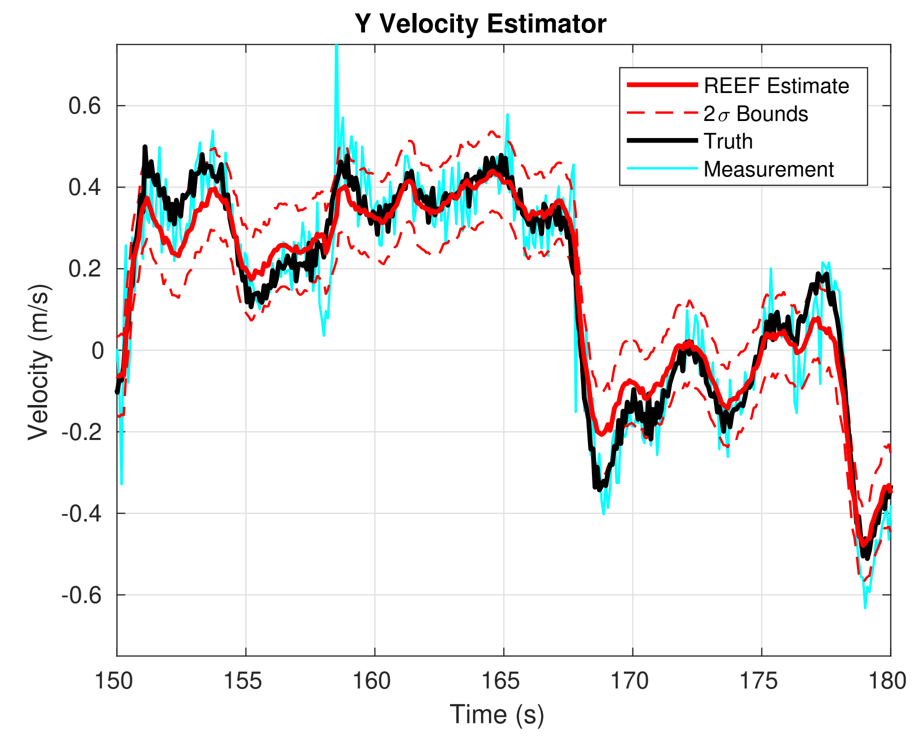


(a)

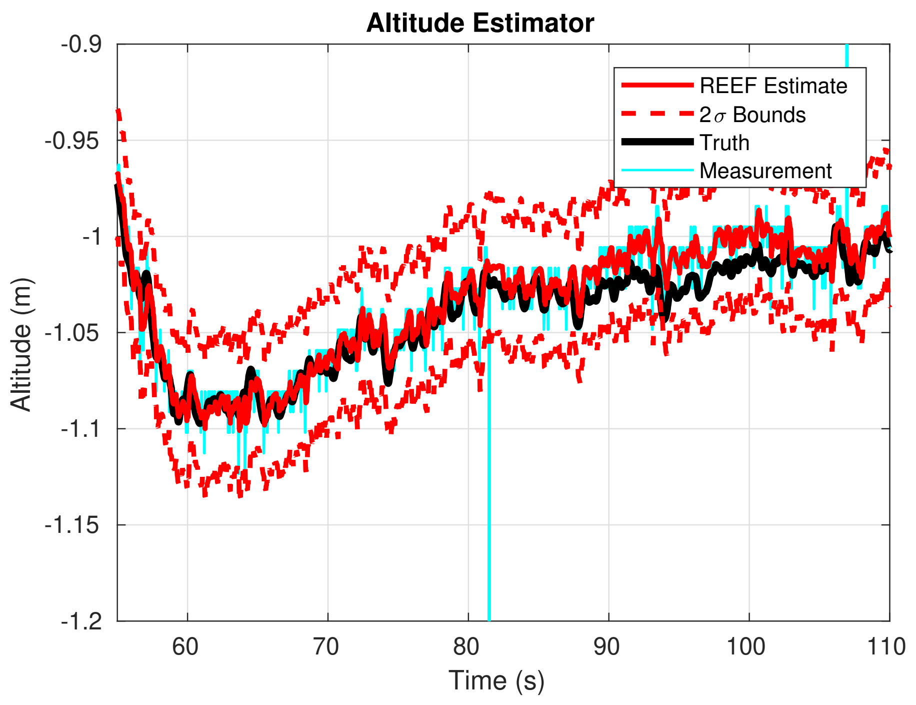
ground, its minimum height is -0.25 meters (this is the mounting height for the altimeter).

The REEF Estimator is equipped with a χ^2 rejection scheme based on the Mahalanobis distance [103]. This scheme allows the filter to ignore erroneous and inconsistent altitude and velocity measurements (that may be due to the occasional generation of outliers). A common outlier for a low-grade ultrasonic sensor (as the one used here) is observed in Figure 6.16c at time $t \approx 81s$. The rejection scheme, however, appropriately ignores the altitude measurement, as can be seen from the unaffected state estimate.

In the case of a velocity request of zero m/s, this controller has been seen to allow the vehicle to drift only about 20 cm over 60 seconds for the x and y axis, although this may be trim quality dependent. The vehicle's global position is not estimated; however, the current controller is written to receive position requests and use a motion capture system as feedback. This setting may be useful for specific experiments indoors as development and testing of guidance laws or SLAM algorithms.

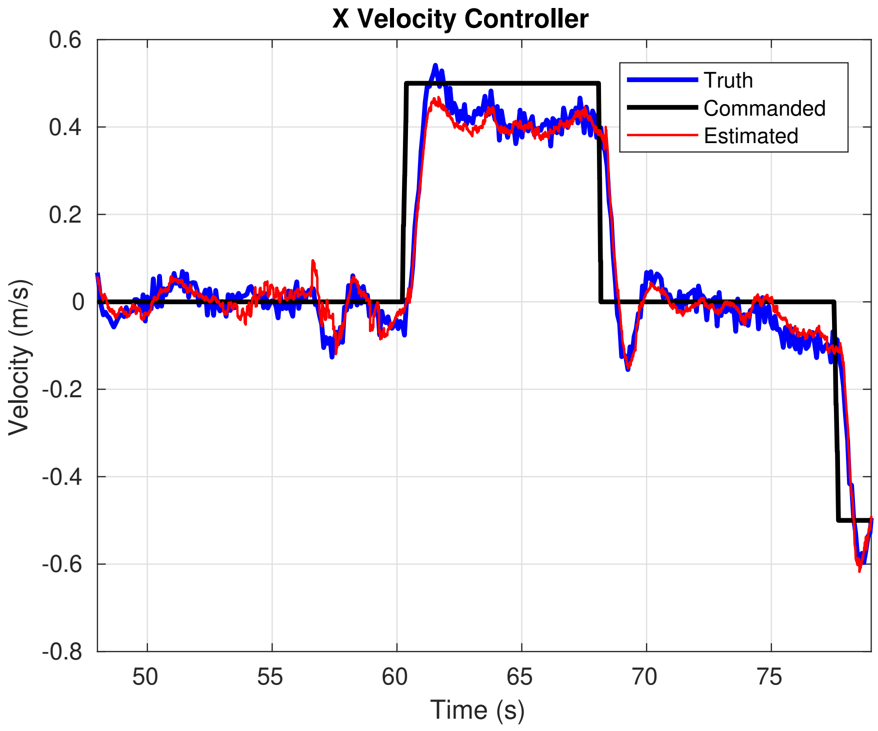


(b)

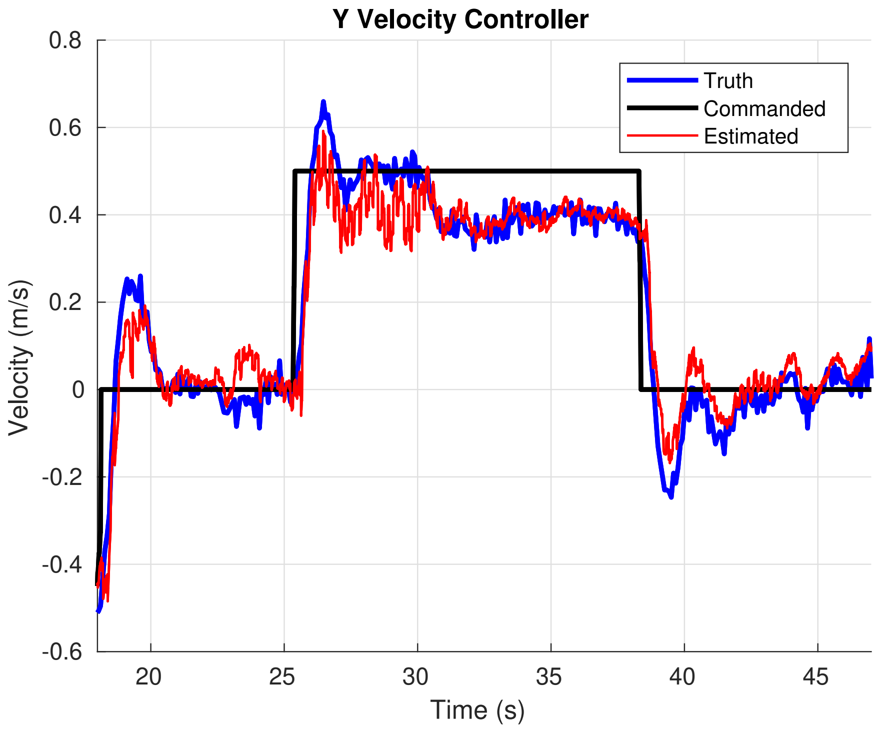


(c)

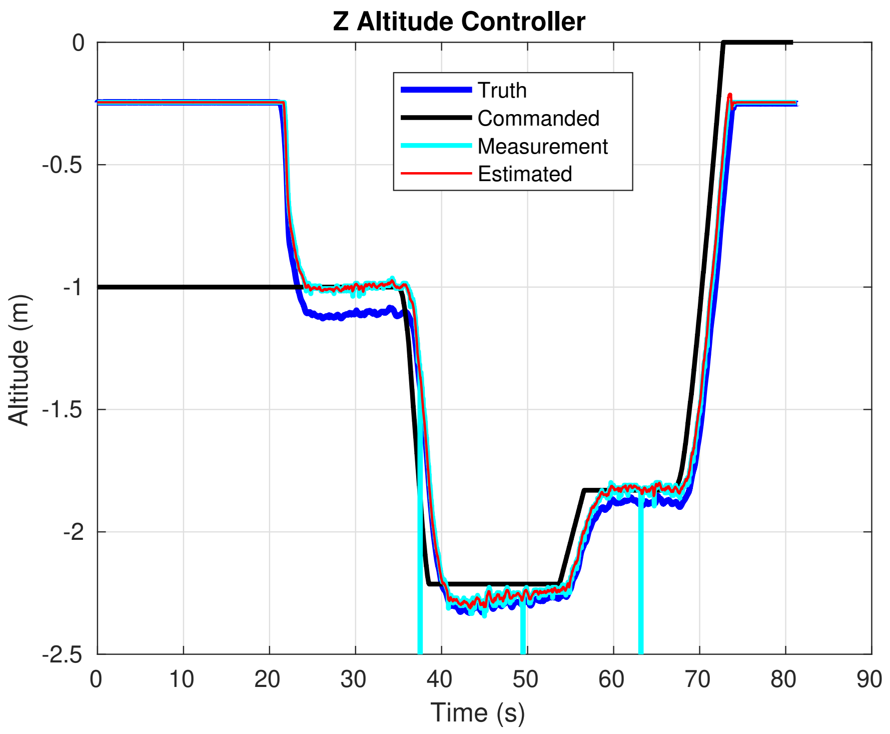
Figure 6.16: XY velocity and altitude from the REEF Estimator. The estimates are compared with the ground truth from a motion capture system. Reprinted with permission from [2].


X Velocity Controller
Truth
Commanded
Estimated
Velocity (m/s)
Time (s)
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
50
55
60
65
70
75

(a)


Y Velocity Controller
Truth
Commanded
Estimated
Velocity (m/s)
Time (s)
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
20
25
30
35
40
45

(b)



(c)

Figure 6.17: Closed-loop performance of the multirotor. Reprinted with permission from [2].

6.2.6 Summary

The open-source REEF Estimator that leverages the partial-update capabilities to provide a simplified estimator for flying vehicles. The results shown used the UD partial-update filter, but the square root was also flown without complications. The main objective of this implementation is to expedite the establishment of flight capabilities that can support specialized research. The results for a typical flight showed the REEF Estimator's functionality and ability to cope with non-linearities and varying nuisance parameters; inherited benefits from the partial-update approach. The REEF estimator package incorporates a PID controller that, in closed-loop, allows stable and repeatable flight. Compared to the IMU-camera calibration hardware from the previous section, a traditional Schmidt filter implementation is neither functional nor safe, especially for relatively long flights with low-grade IMU's, as is in this case. The reason is that varying biases (nuisance parameters) are not estimated, and if their drift is important, the risk of vehicle crash due to filter divergence increases.

The REEF Estimator is well suited for a large variety of end-users, starting for a beginner lacking experience in hardware, estimation, and control theory, to experts who have worked with similar platforms and want a system to support their work in a GPS or motion capture free environment. Since this solution does not depend on vehicle physics, it can be used on a different multirotor frame than the one shown here. In fact, this has been validated on hexacopters and even on ground robots (XY velocity estimator only). The full algorithms, flight simulator, and hardware list along with links to the associated code and further documentation are provided online at `https://github.com/uf-reef-avl/reef_estimator_bundle`.

6.3 Angular rate estimation of a non-cooperative space body using RGB-D measurements

6.3.1 Introduction and motivation

Recent work in [104] has studied the problem of determining if an observed orbital body is exerting internal control of its attitude. This problem is of interest in space situational awareness domain, in which a bad actor might deceptively use a "retired" satellite for covert surveillance.

The task of classifying satellites based on measurements of their angular rate also has implications for orbital debris removal missions, in which mission planners may wish to prioritize debris targets whose motion closely resembles rigid body motion. The work from [104] segmented the active control detection problem into two tasks: 1) estimation of motion and 2) classification of motion based on parameter estimation and statistical testing. The initial work suggested a simple χ^2 test for task #2. Nonlinear optimization was used to demonstrate task #1, but is not suitable for real-time solutions.

This hardware application implements a vision-based technique to estimate the rigid-body angular rate of a target body. This activity directly supports task #1 in the active control detection problem. Here as in [104], any other torques that can potentially be present, such as those caused by gravity gradients or air drag, are assumed negligible over the experiment's time scale. The only significant torques acting on the body should be assumed to come from internal control.

In this application, an RGB-D sensor is used as a stand-in for generic spacecraft measurements, which could come from a high-resolution ground-based radar or a stereo-vision system on a chasing spacecraft. The measurements acquired by the RGB-D sensor are used to build estimates of the relative angular rate of the body being assessed. Since the primary objective of this implementation is to extract rotational kinematics information only, a dynamical model that includes mass properties is not necessary.

In order to get information about the rotational motion, visual features on the body are detected and then filtered by a selection criteria to eliminate low-quality features. The selected features are then tracked between sequential images and used by a UD partial-update EKF to generate estimates of the target's relative angular velocity.

This hardware application involves nonlinear models and nuisance states as in the IMU-camera calibration and the REEF estimator, but in contrast, and very interestingly, this is a case where the nuisance states, namely the angular rates, are the states of interest; a case where the partial-update concept is highly useful. Due to the nature of this scenario, a conventional EKF was seen to be problematic and inconsistent. On the other hand, a Schmidt filter is not even an option because the angular rates would not be estimated at all.

The IMU-camera calibration previously presented may seem similar to the angular rates estimation problem. However, in the IMU-camera calibration application, although the nuisance parameters seem to be the sates of interest, they are not. In that example, the global navigation states are the states of interest, while the calibration parameters are a necessity to improve such navigation states. Put differently, the IMU-camera calibration parameters by themselves are not useful.

6.3.2 Tracking and estimation system overview

Before going into the technical details, this section provides a brief description of the process utilized to obtain the relative angular rate estimates based on RGB-D measurements. Details on the actual implementation are given in the following subsection.

The vision system uses a simple frame-to-frame estimation approach, in which changes between sequential images are processed to estimate the relative angular rate of the object of interest. The basic operation of the vision system can be summarized as follows:

1. Extraction of features: Salient features on the body are detected via a Shi-Tomasi corner detector.

2. Selection of image features: Extracted features that fall outside a pre-selected 3D volume are eliminated. The goal is to avoid using features that are prone to rapidly move out of the image frame, and features that are far beyond the region of interest. Extracted features with no depth information are also eliminated.

3. Tracking: The remaining complying features are tracked frame by frame with a Kanade-Lucas-Tomasi (KLT) tracker.

4. Mapping of image pixels to relative 3D position: The pixel coordinates (u, v) given by the tracker, are mapped to the corresponding 3D positions vectors. The 3D position vectors are given on a algorithm-selected body-fixed frame, but coordinatized in the camera frame.

Figure 6.18 shows the coordinates frames used in this implementation.

5. Filtering: The 3D position vectors of the features are fed to the UD partial-update filter to perform relative angular velocity estimation.

6. Filter re-initialization: Due to the body's rotation features initially detected are lost. The filter re-initializes features when available features are considered insufficient based on a threshold experimentally set.

6.3.3 Additional notation, modeling and feature treatment

6.3.3.1 Additional notation

This hardware application follows the nomenclature used in previous chapters, along with the following additions.

- Rotation matrix that actively transforms a vector are expressed with $\mathbf{R}$.

- When the context requires it, the matrix representation of a vector includes a subscript that indicates the reference frame in which the vector is resolved. In this application, the uppercase X, Y, and Z (not bold) letters are used to represent a 3D coordinate of a feature position. A vector representing the position of a feature, which is resolved in the $\mathbf{c}^+$ frame, for instance, is written as follows:

$$ {}^{C}\mathbf{v}_F = \begin{bmatrix} X_F \\ Y_F \\ Z_F \end{bmatrix}_C . \tag{6.78}$$

- In this hardware application sub-indices i and j are reserved for enumeration and matrices elements, whereas the sub-index k is used to denote time instance.

6.3.3.2 Reference frames

Two main reference frames are defined to perform the estimation of the body's rotational state: the camera optical frame $\mathbf{o}^+$, which has its origin at the optical camera center, and an *arbitrary* body-fixed reference frame $\mathbf{a}^+$. The $\mathbf{a}^+$ frame, defined at the beginning of the experiment at time $t = 0s$, has its origin coincident with the position of the feature that is detected first. At this time, the $\mathbf{a}^+$ frame is set to be aligned with the optical frame. In practice, the "first" feature is the one that occupies the first position of the variable array containing all the detected feature positions. Figure 6.18 illustrates the relationships between the reference frames mentioned above.

In addition to the $\mathbf{o}^+$ and $\mathbf{a}^+$ frames, for purposes of algorithm validation, a few extra reference frames are defined as, shown in Figure 6.18. Reference frame $\mathbf{c}^+$ is the frame attached to the plate where the RGB-D sensor is mounted. Additionally, $\mathbf{b}^+$, is a frame attached to a plate placed on the body. This plate carries motion capture markers for obtaining a reference truth angular rate. Finally, the motion capture system frame, $\mathbf{v}^+$, is also shown in Figure 6.18. The $\mathbf{v}^+$ frame represents the VICON motion capture system inertial frame in which ground reference values are obtained.

6.3.3.3 Reprojection model

The positions of the body features with respect to the RGB-D sensor are obtained via the well-known pinhole camera model by leveraging the depth information. As previously described in the IMU-camera calibration hardware implementation, the pinhole camera model projects a 3D position (X_i, Y_i, Z_i) that is resolved in the camera optical frame (collocated at the camera center), into the corresponding pixel position $(\tilde{u}_i, \tilde{v}_i)$ on the image. Since the RGB-D sensor *generally* provides depth, Z for a detected feature at $(\tilde{u}_i, \tilde{v}_i)$, its 3D position can be directly computed via the calibrated pinhole model

$$\tilde{\mathbf{y}}_{F_i} = \begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix}_O = \begin{bmatrix} Z_i(\tilde{u}_i - c_x)/f_x \\ Z_i(\tilde{v}_i - c_y)/f_y \\ Z_i \end{bmatrix}_O . \tag{6.79}$$

Recall that f_x and f_y are the focal length of the camera in the x and y axis, and that c_x and c_y, are the coordinates of the camera center. Both focal lengths and camera center coordinates are quantities expressed in pixels, and their values for this application are obtained via standard

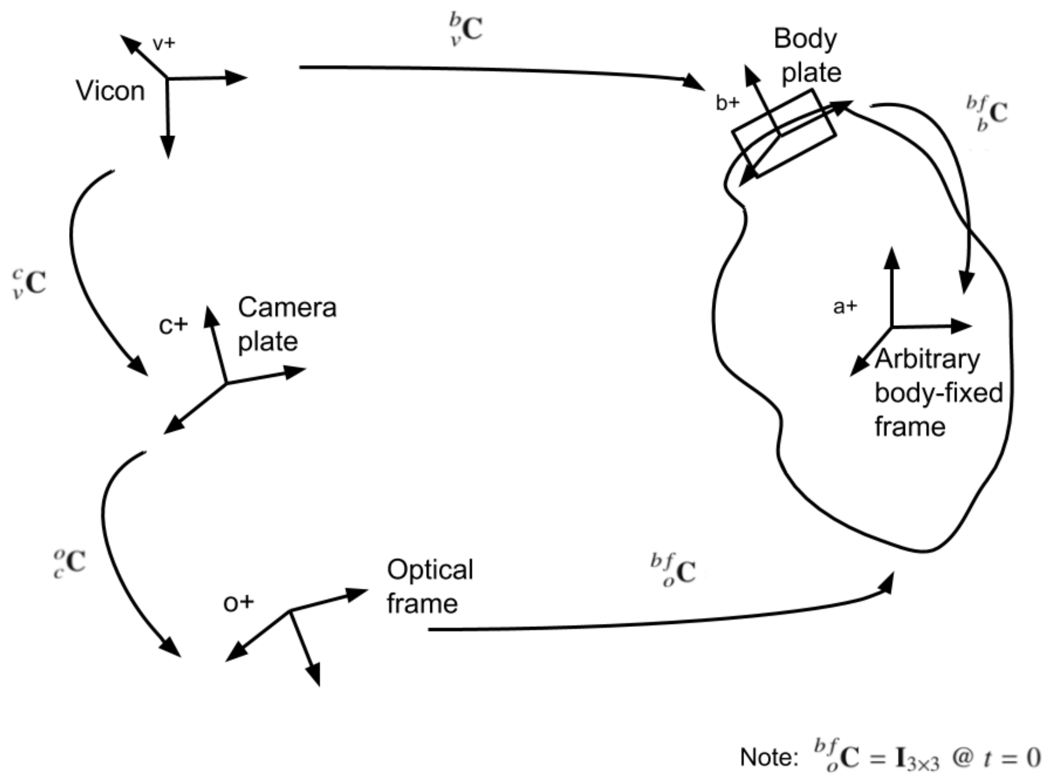


Figure 6.18: Coordinate reference frames utilized. Position of the features with respect to the body, attitude and angular velocities are all coordinatized in the camera frame. Reprinted with permission from [55].

OpenCV functions for camera calibration. Figure 6.19 illustrates how a 3D coordinate for a feature, $\mathbf{X}$, is mapped into a pixel, $\mathbf{x} = [\tilde{u}_i, \tilde{v}_i]^T$, on the image plane.

6.3.3.4 Feature extraction

The presented implementation, leverages the detected change in position of the image points from frame to frame to perform the estimation of the target (body) relative motion. In practice, this is often achieved by matching features in two consecutive frames and computing the rigid transformation (rotation and translation) that aligns the detected features. Whereas it is possible to use essentially all pixels in the frames to perform this computation, it is commonly more tractable to track visually distinctive features and use these only to compute the rigid transformation estimate. The latter approach is used in this hardware implementation as it is more suitable for real-time

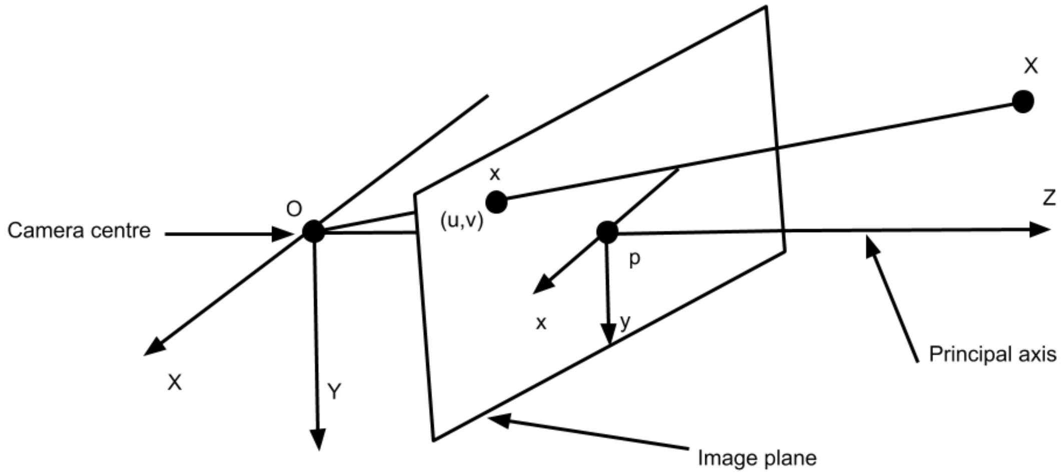


Figure 6.19: The pinhole camera model projects a 3D coordinate **X** (resolved in the camera frame) into the image pixel coordinates $(\tilde{u}_i, \tilde{v}_i)$. The camera center coordinates locate the point p on the image plane, which is the intersection of the image plane and the principal axis. The principal axis is along the Z axis of the camera, towards the scene being captured. Reprinted with permission from [55].

execution.

In this implementation the selection of features is done using a Shi-Tomasi [105] corner detector, followed by a process that retains only *good* features with the objective of increase overall feature traceability. These *good* features are selected following the criteria presented in the following subsection.

Once the initially detected features have been refined, into a better sub-set, feature tracking is performed with a KLT Tracker [106]. The objective of the tracking the features is maintain the same good features from frame to frame to be able to estimate the underlying body's motion. The KLT tracker essentially estimates the displacement of features from frame to frame via an optimization process that minimizes an image intensity metric for the feature's neighborhood. This hardware implementation uses the KLT tracker function from the OpenCV 2.4.9.1 package [107].

6.3.3.5 Feature selection criteria

Feature extraction is restricted to a pre-defined searching volume. A polygon defines this volume on the image coordinates (u, v) and a pre-defined depth. For all the experiments included in this implementation, the maximum sensing depth is set to 3.5m, while the polygon on image coordinates is set to be a rectangle of enough size to cover the body of interest while it rotates. Although the selection of the region of interest can be automated using techniques as blob detection and image motion analysis, this implementation requires manual selection.

Within the polygon that covers the body's motion in analysis, a second region that encloses only features that are not close to body edges is also utilized. The objective of this second region is to help differentiate the long-term-to-short-term available features. The short-term features are those that, due to the body's rotation and proximity to the body's edges, will move out of the image relatively soon after their potential detection occurs. Long-term features are those that will remain visible for a longer period of time because they are relatively far from the body edges. This secondary region is selected manually as well.

Although color information could be used to extract good features from the complete detected set, the combination of the pre-selected volume and the built-in outlier rejection from the Kalman filter (described in Section 6.3.4), showed to be enough for the purposes of this application. Direct use of color information is also less useful in space operations because available lighting tends to be of poor quality.

Features complying with the selection criteria outlined in this section are considered good quality features for body tracking and estimation purposes.

6.3.3.6 Feature position vectors in the arbitrary body frame

Once *good* features are being tracked, the algorithm proceeds to compute the corresponding position vectors. The position vectors of the features, however, do not directly allow the estimation of the body's angular rate; instead, they are used to construct a set of vectors that resemble the body's rotational motion. These vectors defined in the body frame are referred to as the body vectors.

When the algorithm to estimate the angular rate begins, it fixes an arbitrary reference frame on the rotating body when the first measurement arrives. For this implementation, the feature that serves as the arbitrary origin on the body is the highest-quality feature produced by the OpenCV `GoodFeaturesToTrack` function, which implements the Shi-Tomasi corner detector. For this detector, the feature quality is related to the magnitude of the eigenvalues that capture the image motion [105]. At subsequent times (after the first measurement arrives), as the vision system provides the position of more features, they are re-expressed in the rotating body reference frame. This results in a set of body vectors that are tracked from frame to frame and fed as measurements to a partial-update EKF, which uses a motion model to obtain feature position estimates and angular rates of the body. This process is summarized in the next section.

The flow of the information, then, is as follows. The output of the feature detector and tracker is a set of feature positions $(u_i, v_i),\ i = 1, \ldots N$ on the image, and the corresponding depths Z_i. With this information in hand, Equation 6.79 is used to obtain the corresponding 3D coordinates (X_i, Y_i, Z_i) of each feature. Then, the highest quality feature, named (X_o, Y_o, Z_o), is set as the origin (O) of the reference frame that is attached to the body, and with subsequent features available, the body vectors ${}^{O}\mathbf{p}_{F_i}$ are computed by vector subtraction as

$$ {}^{O}\mathbf{p}_{F_i} = \begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix}_O - \begin{bmatrix} X_o \\ Y_o \\ Z_o \end{bmatrix}_O . \tag{6.80} $$

The body vectors, ${}^{O}\mathbf{p}_{F_i}$, are the feature position vectors expressed from the body-fixed frame origin. These body vectors are then used as measurements for rotational motion estimation.

It is important to mention that although the i^{th} body vector, ${}^{O}\mathbf{p}_{F_i}$, for the i^{th} feature, is expressed with respect to the arbitrary reference frame, $\mathbf{a}^{+}$, the ${}^{O}\mathbf{p}_{F_i}$ vector is still resolved in the $\mathbf{o}^{+}$ frame. Thus, the estimated relative angular rates and attitude will be naturally measured with respect to the $\mathbf{o}^{+}$ frame as well. Figure 6.20 shows the relationship between position of features F_i

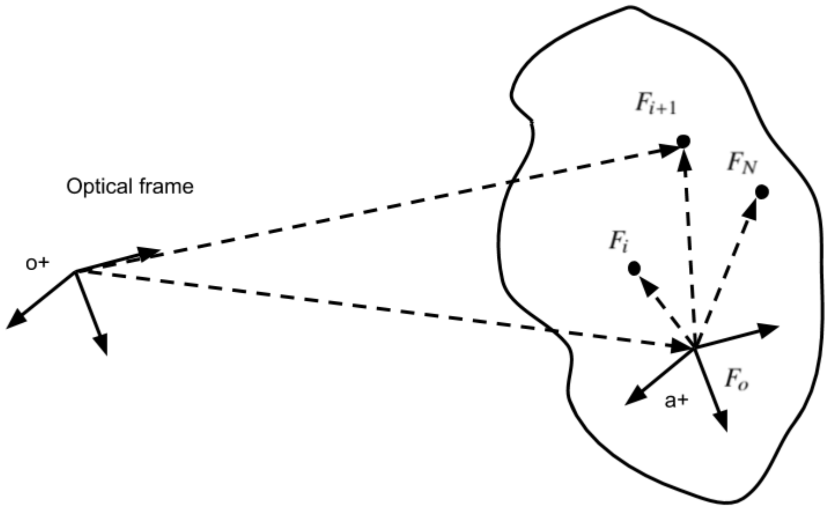


Figure 6.20: Coordinate reference frames utilized. Position of the features with respect to the body, attitude and angular velocities are all coordinatized in the camera frame. Reprinted with permission from [55].

with respect to the optical frame $\mathbf{o}^+$ and the established body-fixed reference frame $\mathbf{a}^+$. The set of body vectors for a generic case are represented in Figure 6.20 as $\mathbf{F}_1, \mathbf{F}_2, \ldots, \mathbf{F}_N$ (omitting the notation $\tilde{\mathbf{p}}$ for clarity). An additional observation is that the origin of the $\mathbf{a}^+$ frame is defined on the surface of the target body. Thus, it will exhibit translational motion with respect to the observer frame, however, for the experiments that were performed, the estimation of the angular rate is not ambiguous, since there is no relative translational velocity between the sensor and the body.

6.3.4 Extended Kalman Filter

An UD partial-update EKF that uses the body vectors that are obtained in Section 6.3.3.6 is implemented to estimate the relative angular rate of the target. This section presents the details.

6.3.4.1 Process and measurement model

The EKF state vector includes the body vectors, ${}^{O}\mathbf{p}_{F_i}$, of each tracked feature (obtained with equation 6.80), and the relative angular velocity of the target body ,${}^{O}\boldsymbol{\omega}_b$ expressed in the RGB-D sensor frame $\mathbf{o}^+$:

$$\mathbf{x} = \begin{bmatrix} {}^{O}\mathbf{p}_F^{\mathrm{T}} & {}^{O}\boldsymbol{\omega}_b^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} . \tag{6.81}$$

Here, ${}^{O}\mathbf{p}_F^{\mathrm{T}}$ encapsulates the N features' position vectors, as

$${}^{O}\mathbf{p}_{F_i}^{\mathrm{T}} = \begin{bmatrix} X_i & Y_i & Z_i \end{bmatrix}^{\mathrm{T}} , \tag{6.82}$$

such that

$${}^{O}\mathbf{p}_F^{\mathrm{T}} = \begin{bmatrix} {}^{O}\mathbf{p}_{F_i}^{\mathrm{T}} & {}^{O}\mathbf{p}_{F_{i+1}}^{\mathrm{T}} & \dots & {}^{O}\mathbf{p}_{F_N}^{\mathrm{T}} \end{bmatrix} . \tag{6.83}$$

More specifically, ${}^{O}\mathbf{p}_{F_i}^{\mathrm{T}}$ is the (body) vector position of the i^{th} feature, measured from the arbitrary origin, $\mathbf{a}^{+}$, on the body frame but coordinatized in the camera frame, $\mathbf{o}^{+}$. ${}^{O}\boldsymbol{\omega}_b^{\mathrm{T}}$, the angular velocity of the body with respect to the camera frame, $\mathbf{o}^{+}$, is composed as follows:

$${}^{O}\boldsymbol{\omega}_b^{\mathrm{T}} = \begin{bmatrix} \boldsymbol{\omega}_x & \boldsymbol{\omega}_y & \boldsymbol{\omega}_z \end{bmatrix} . \tag{6.84}$$

The sub-index for each component corresponds to the axis of rotation.

The discrete system process model is expressed as follows [108]:

$$\mathbf{f}(\mathbf{x}(t),\Delta t)_k = \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \\ X_2 \\ Y_2 \\ Z_2 \\ \vdots \\ X_N \\ Y_N \\ Z_N \\ \boldsymbol{\omega}_x \\ \boldsymbol{\omega}_y \\ \boldsymbol{\omega}_z \end{bmatrix}_k = \begin{bmatrix} X_1 - \omega_z \Delta t Y_1 + \omega_y \Delta t Z_1 \\ \omega_z \Delta t X_1 + Y_1 - \omega_x \Delta t Z_1 \\ -\omega_y \Delta t X_1 + \omega_x \Delta t Y_1 + Z_1 \\ X_2 - \omega_z \Delta t Y_2 + \omega_y \Delta t Z_2 \\ \omega_z \Delta t X_2 + Y_2 - \omega_x \Delta t Z_2 \\ -\omega_y \Delta t X_2 + \omega_x \Delta t Y_2 + Z_2 \\ \vdots \\ X_N - \omega_z \Delta t Y_N + \omega_y \Delta t Z_N \\ \omega_z \Delta t X_N + Y_N - \omega_x \Delta t Z_N \\ -\omega_y \Delta t X_N + \omega_x \Delta t Y_N + Z_N \\ \boldsymbol{\omega}_x \\ \boldsymbol{\omega}_y \\ \boldsymbol{\omega}_z \end{bmatrix}_{k-1} + \ \mathbf{w}_{k-1} \ , \tag{6.85}$$

or

$$\mathbf{f}(\mathbf{x}(t),\Delta t)_k = \begin{bmatrix} (\mathbf{I}_{3\times 3} + \lfloor \boldsymbol{\omega} \times \rfloor \Delta t)^O \mathbf{p}_{F_1} \\ (\mathbf{I}_{3\times 3} + \lfloor \boldsymbol{\omega} \times \rfloor \Delta t)^O \mathbf{p}_{F_2} \\ \vdots \\ (\mathbf{I}_{3\times 3} + \lfloor \boldsymbol{\omega} \times \rfloor \Delta t)^O \mathbf{p}_{F_N} \\ {}^O\boldsymbol{\omega}_b \end{bmatrix}_{k-1} + \ \mathbf{w}_{k-1} \ . \tag{6.86}$$

Where Δt is the propagation interval, $\lfloor \boldsymbol{\omega} \times \rfloor$ is the skew symmetric matrix formed with elements ω_x, ω_y and ω_z, and $\mathbf{I}_{3\times 3}$ is a 3×3 identity matrix. X_i, Y_i and Z_i are body vector components resolved in the $\mathbf{o}^+$ frame.

For this application, it is assumed that the RGB-D sensor provides direct measurements of the position of the features with respect to the current arbitrary origin X_o, Y_o, Z_o. That is, that the camera reads the body vectors directly. Thus, the measurement model can be written simply as

$$\hat{\mathbf{y}}_k = \mathbf{H}\hat{\mathbf{x}}_k + \mathbf{v}_k \ . \tag{6.87}$$

Where $\mathbf{H}$ is defined as

$$\mathbf{H} = \begin{bmatrix} \mathbf{I}_{3N\times 3N} & \mathbf{0}_{3N\times 6} \end{bmatrix} \ , \tag{6.88}$$

here, $\mathbf{v}_k$ is a white noise process with covariance matrix $\mathbf{R}_F = E[\mathbf{v}\mathbf{v}^{\mathbf{T}}]$. For this work, it is assumed that the uncertainty levels for the measurements are known or can be computed from recorded data, and that they are the same for the three dimensions X, Y, Z. However, it must be mentioned that as in [109], a more accurate noise model for a RGB-D type of sensor that considers the correlation between the three coordinates and uncertainty dependency on the depth value could be used.

To use the Extended Kalman Filter propagation equations for the error covariance, the state transition matrix $\mathbf{F}_{k-1}$ and the input noise matrix $\mathbf{G}_{k-1}$ are needed. By linearization of Equation (6.85), the state transition matrix is found to be in the following form:

$$\mathbf{F}_{k-1} = \begin{bmatrix} (\mathbf{I}_{3\times 3} + \lfloor \boldsymbol{\omega} \times \rfloor \Delta t) & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & \dots & -\lfloor {}^{C}\mathbf{p}_{F_i} \times \rfloor \Delta t \\ \mathbf{0}_{3\times 3} & (\mathbf{I}_{3\times 3} + \lfloor \boldsymbol{\omega} \times \rfloor \Delta t) & \mathbf{0}_{3\times 3} & \dots & -\lfloor {}^{C}\mathbf{p}_{F_{i+1}} \times \rfloor \Delta t \\ \vdots & \vdots & \ddots & \dots & \vdots \\ \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & (\mathbf{I}_{3\times 3} + \lfloor \boldsymbol{\omega} \times \rfloor \Delta t) & -\lfloor {}^{C}\mathbf{p}_{F_N} \times \rfloor \Delta t \\ \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & \mathbf{I}_{3\times 3} \end{bmatrix} . \tag{6.89}$$

Where $\mathbf{F}$ is of dimensions $(3N+3)\times(3N+3)$. Whereas $\mathbf{G}$, it is just a matrix with ones along the corresponding entries that map process noise into states. States with process noise are specified in the hardware implementation section. Finally, the EKF covariance is propagated according to the UD partial-update equations from Table 4.1.

6.3.4.2 UD partial-update filter

Due to the nature of this scenario, a conventional EKF was seen to be problematic, producing biased estimates for all state estimates and with overconfident covariance. Moreover, relatively large corrections were observed on the angular rates, even when they were known to be constant during experiments. This was also the case, even for correct initial conditions with low covariance.

As mentioned before, this hardware application involves nonlinear models and nuisance states as the previous applications, but in contrast, and very interestingly, this is a case where the nuisance states, namely the angular rates, are the states of interest; a case where the partial-update concept is highly useful. Specifically, this application demonstrates the successful use of the UD partial-update filter.

6.3.5 Filter initialization

In order to initialize the filter, the information provided by the feature tracker is leveraged to compute a rough estimate for the angular rates and their standard deviations. This subsection outlines the procedure. First, by using feature positions from one frame to the next, a rigid transformation that aligns two subsequent clouds of body vectors is obtained through Singular Value Decomposition (SVD)-based least-squares [110]. Such transformation is computed to approximate the rotational change of the body in motion. Then, angle extraction from the computed rigid transformation is performed. This is, the change in the Euler angles that compose the rigid body transformation (locally), are obtained. Finally, the ratio of the change on the Euler angles to the time-step interval is used as a coarse approximation of the local angular velocity estimate. Based on the notation used in this hardware implementation, ${}^{O}\mathbf{p}_F^{\mathrm{T}} = \begin{bmatrix} {}^{O}\mathbf{p}_{F_1}^{\mathrm{T}} & {}^{O}\mathbf{p}_{F_2}^{\mathrm{T}} & \dots & {}^{O}\mathbf{p}_{F_N}^{\mathrm{T}} \end{bmatrix}$ composes the first point cloud (set of body vectors in the previous frame), and ${}^{O}\mathbf{q}_F^{\mathrm{T}} = \begin{bmatrix} {}^{O}\mathbf{q}_{F_1}^{\mathrm{T}} & {}^{O}\mathbf{q}_{F_2}^{\mathrm{T}} & \dots & {}^{O}\mathbf{q}_{F_N}^{\mathrm{T}} \end{bmatrix}$ the second one (body vectors in the current frame). The solution for the rotation is obtained by minimizing the cost function of Equation (6.91) subject to the constraint that the transformation $\mathbf{R}$ is not a reflection. The implemented SVD-based solution of this problem, incorporates such constraint in the minimization process to search for rotations only [110]. Due to the relatively high frequency of the RGB-D camera measurements and slow angular rate expected for the body, the rotation delivered by the least-squares approach is assumed to be well modeled as a differential rotation (from a frame to the next) $\delta\mathbf{R}$ via :

$$\delta\mathbf{R} = \mathbf{I}_{3\times3} + \begin{bmatrix} 0 & -\omega_z\Delta t & \omega_y\Delta t \\ \omega_z\Delta t & 0 & -\omega_x\Delta t \\ -\omega_y\Delta t & \omega_x\Delta t & 0 \end{bmatrix} , \tag{6.90}$$

and

$$(\delta\mathbf{R}, \mathbf{t}) = \arg \min_{\mathbf{R}\in SO(3), \mathbf{t}\in\mathbf{R}^3} \sum_{i=1}^{n} \|(\mathbf{R}\mathbf{p} + \mathbf{t}) - \mathbf{q}\|^2 . \tag{6.91}$$

As mentioned before, $\delta\mathbf{R}$ is chosen to be parameterized with Euler angles. Specifically, as an Euler 3-2-1 rotation through the angle set $[(\psi, \theta, \phi)]$ (or yaw, pitch, and roll).

It is important to point out that since the pair of 3D point clouds, $\mathbf{p}$ and $\mathbf{q}$, are resolved in the camera reference frame, the small change in attitude, $\delta\mathbf{R}$, is given with respect to the camera frame as well. Again, once the small rotations are approximated, the time interval Δt utilized to estimate the angular rates.

For this application, the coarse estimation of the angular rates is performed continuously during three seconds (as features become available), and then an empirical mean and standard deviation are computed from the collected data. These empirical values mean directly initialize the EKF. In Figure 6.21, a typical history of the coarse angular rates estimates produced by this procedure is shown. Finally, the position of the body vectors is initialized using their computed values directly, and their initial covariance is set experimentally commensurate to the measurement noise values.

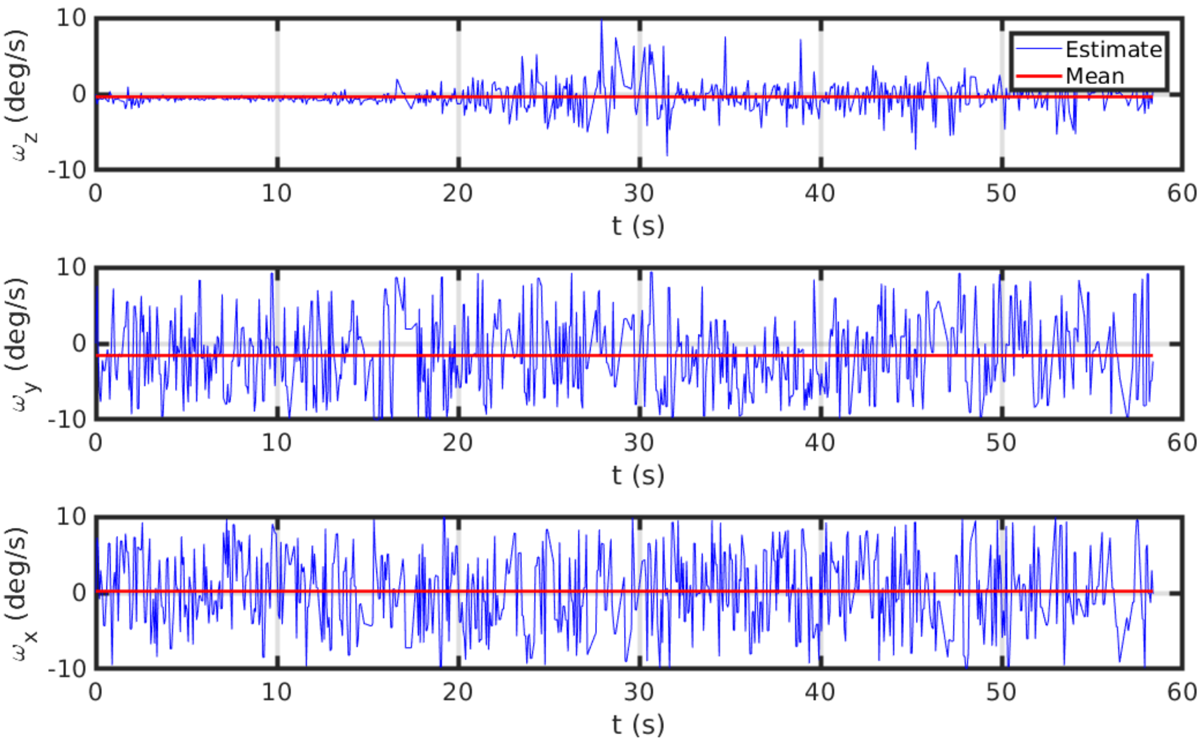


Figure 6.21: Angular rates obtained through numerical differentiation of differential rotation angles. The differential rotation angles are extracted from the estimated $\delta\mathbf{R}$ from equation 6.90. In this figure a 60 seconds history of the coarse estimates for the angular velocity components is shown. The empirical average is shown in red. The truth value for this specific experiment was $\omega_y = 2\ deg/s$ and $\omega_x = \omega_z = 0\ deg/s$. Reprinted with permission from [55].

Since the tracked features eventually move out of the field of view due to the body rotation, the filter needs to be re-initialized when too few features remain in the frame. The filter is re-initialized according to the procedure that is discussed next.

6.3.6 EKF re-initialization

Due to the body rotation or changes in illumination features' visibility can be lost. Thus, if one desires to maintain accurate estimates of the states, a re-initialization procedure is needed. In this implementation, the re-initialization of the partial-update EKF occurs when the number of tracked features falls below a threshold.

The re-initialization of the filter is performed according to the following procedure:

- The current values of the angular velocity are maintained along with their current covariances (cross-correlations are discarded).
- The corner detector is executed, and new features are extracted by following the selection procedure described in Sections 6.3.3.4 and 6.3.3.5. All features previously utilized are discarded.
- Given the position of the new features in image coordinates, predictions of their corresponding body vectors can be obtained via Equation 6.80 and a new $\mathbf{a}^+$ frame is set. Equal initial covariances are assigned to all new feature positions.
- The filter state with the previously estimated angular rate and the new features is then propagated and updated at the next measurement instance.

Since the EKF provides uncertainty information on the tracked features, it can help to the elimination of deceiving-quality features. If a feature is detected and tracked, and it disappears from view, its covariance will start to grow. This implementation, uses an empirical threshold on the covariance of the features such that *high* covariance features are dropped. In this manner, only relatively low-covariance features participate in the filtering. Additionally, drifting features were filtered out by a χ^2 rejection scheme.

6.3.7 Hardware experiments

6.3.7.1 Experimental setup

Details on the hardware experiments and validation of the vision system are now presented. A calibrated XtionPRO Live RGB-D sensor collects images of a non-cooperative body. The sensor is assumed to be on a satellite trying to detect active attitude control in a neighboring body, and it starts by estimating the body's angular rate (this application). An asteroid mockup, shown in Figure 6.22, is used as the target body for experiments. In order to impart rotational motion to the asteroid, a customized rate-configurable turn table is used. The turn table can be seen in

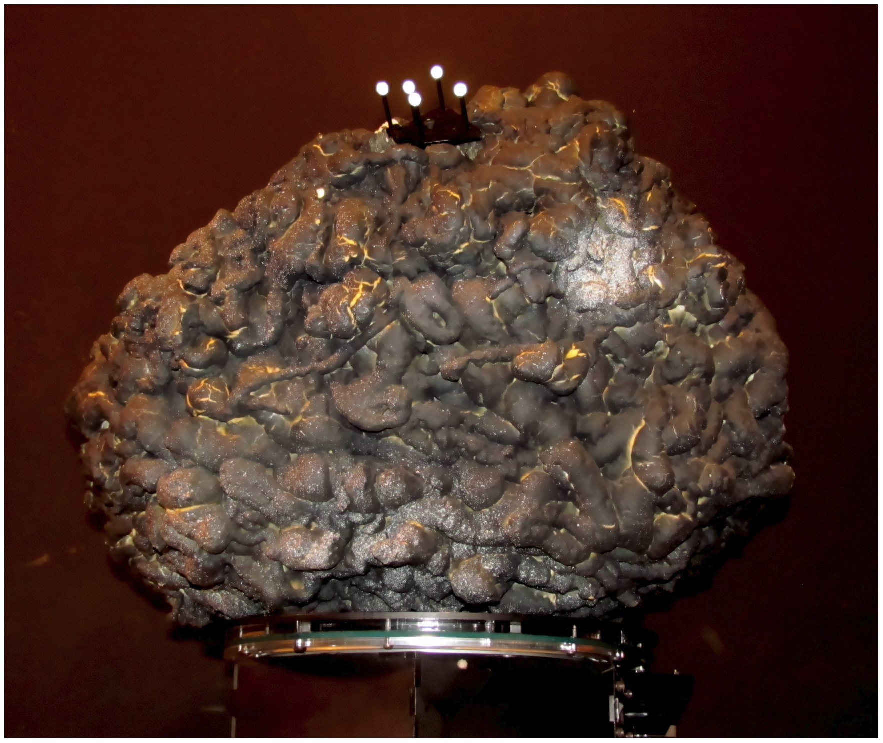


Figure 6.22: The mock asteroid placed on the turn table. The VICON dots can be observed placed at the top on a mounting plate. Reprinted with permission from [55].

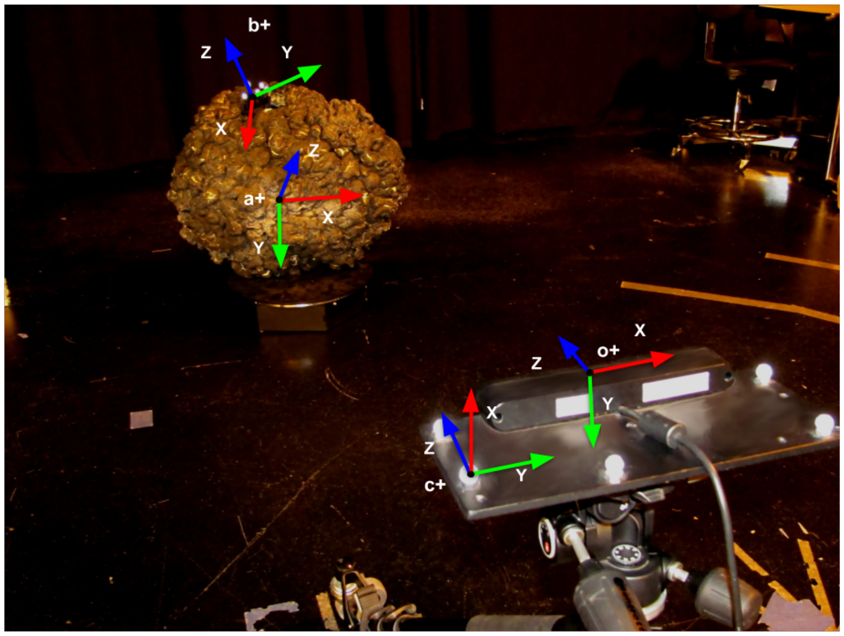


Figure 6.23: Xtion Live Pro camera and the mock asteroid while recording data. The arbitrary reference frame $\mathbf{a}^+$ (on the body), the body plate $\mathbf{b}^+$, the camera optical $\mathbf{o}^+$ and the camera plate $\mathbf{c}^+$ reference frames, have been drawn. Reprinted with permission from [55].

Figure 6.22 under the asteroid. The vision system estimates for the angular rates are validated by comparing its outputs against a laboratory VICON motion capture system. VICON provides attitude measurements for the asteroid naturally resolved in the VICON $\mathbf{v}^+$ frame. In order to evaluate angular velocity estimates, numerically differentiated attitude truth from VICON is used as the reference for truth angular rate. The VICON system tracks the body's motion by means of the plate attached to it carrying retro-reflective beacons. Figure 6.22 displays the plate with the markers placed at the top of the mockup.

The VICON system also tracks the truth pose of the RGB-D sensor. Both asteroid and RBG-D sensor are shown in Figure 6.23. The reference frame on the camera plate is identified as $\mathbf{c}^+$. The calibration (rigid transformation) camera optical frame to the camera plate frame ${}^{C}_{O}\mathbf{C}$, as well as the RGB and IR ("depth") camera registration, are found beforehand. All reference frames used in the experiments are diagrammed in Figure 6.18.

6.3.7.2 Experiments

A series of experiments were performed using different rates for the turntable. For the results shown in this section, the turn table rate was set to 2 deg/s, but all other experiments with different rates (in the range from 1 to 10 deg/s) were found to behave similarly. The resolution of the RGB-D sensor is 640 x 480 pixels, and the frame-rate is 30 Hz. The RGB-D sensor is fixed on a tripod at 2 meters far from the center of the turn table. The video is processed as described in Section 6.3.2, and if needed, reinitialization of the system occurs automatically according to Section 6.3.6. The maximum sensing depth was limited to 3.5 meters, and the area for searching features was manually selected to cover the image area that the asteroid occupies when it rotates. The code was implemented in C++ with OpenCV 2.4.9.1 libraries under the Robot Operating System (ROS kinetic) framework [98].

The process and measurement noise values used to generate the results presented in the next subsections are

$$\mathbf{Q}_F = \mathbf{0}_{3N\times 3N} \ , \tag{6.92}$$

$$\mathbf{Q}_\omega = \mathbf{0}_{3\times3} \ , \tag{6.93}$$

and

$$\mathbf{R}_F = 0.005\mathbf{I}_{3N\times3N} \ . \tag{6.94}$$

The propagation step Δt is simply taken as the inverse of the camera frame-rate, which in this case is 1/30 seconds. The selected β values for the partial-update are

$$\boldsymbol{\beta} = \text{diag}\begin{bmatrix}1.0 & 1.0 & 1.0 & \ldots & 0.05 & 0.05 & 0.05\end{bmatrix} , \tag{6.95}$$

that is, full update for feature positions and 5% of the nominal update for the angular rates.

Figure 6.24 shows a sequence of images (chronologically from a to d) from a single experiment. In each image, the good features to track are shown along with some body vectors in the current arbitrary reference frame, $\mathbf{a}^+$. The detected features are marked as green dots, and the red vectors (arrows) are the estimated body vectors. The origin of the body vectors is at the arbitrary body-fixed reference frame, which corresponds to the best available feature as described in Section 6.3.3.6. For clarity, vectors for the position of the estimated features are only shown for three of the features. The numbers that appear next to each body vector represent the feature quality, being the number one the highest-quality-feature after the origin.

In Figure 6.24 it is possible to observe the vision system re-initialization being executed. Particularly, re-sampling of features occurs from (c) to (d), and a change of origin can be seen in the transition from (a) to (b). This confirms that the re-initialization detailed in Section 6.3.6 behaves as expected in terms of sampling.

6.3.7.3 Angular velocity estimates

As mentioned in Section 6.3.7.1, to evaluate the angular velocity estimates, estimates are compared against numerically differentiated VICON attitude.

Several experiments were conducted using the initial conditions generated by the initialization

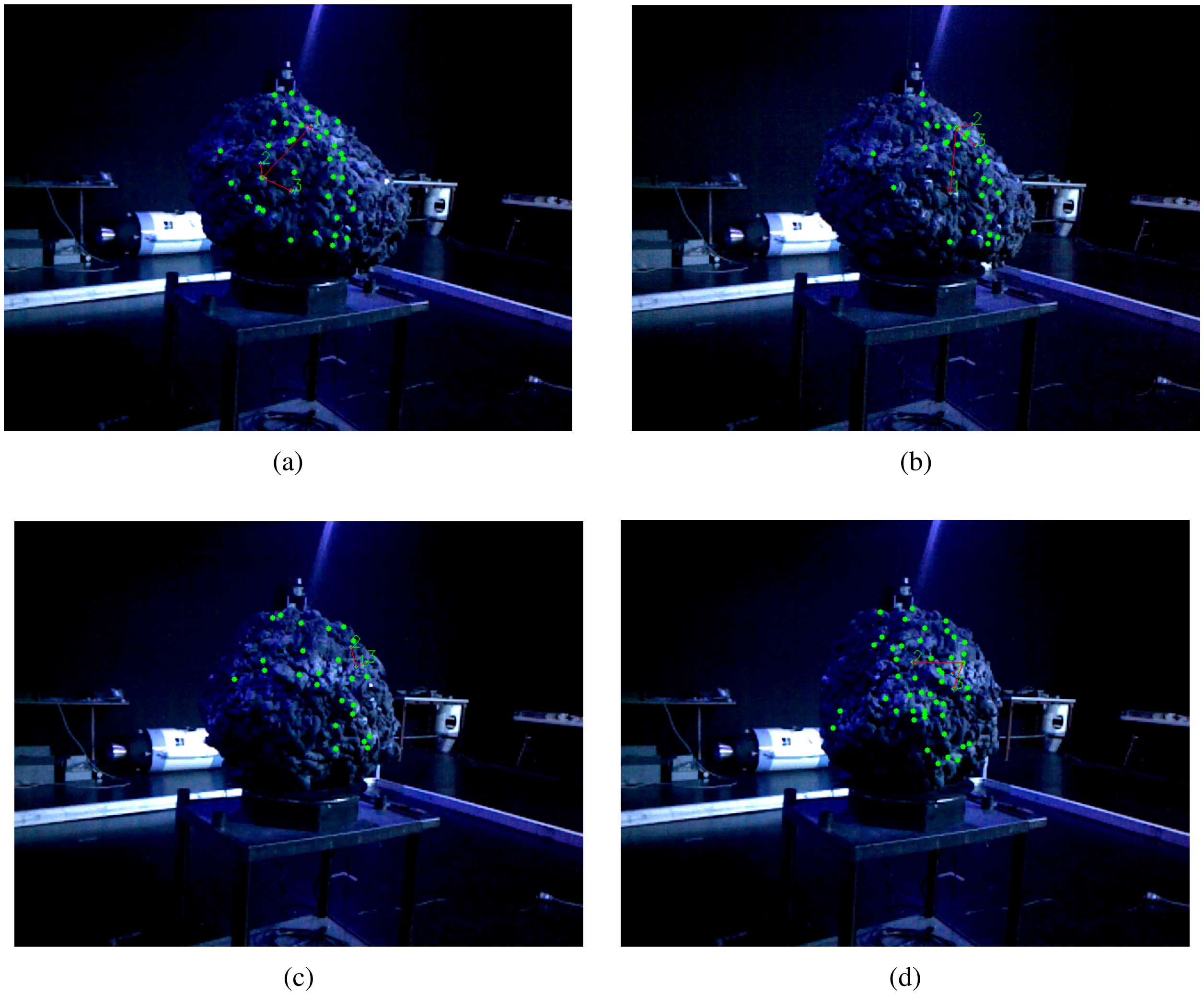


Figure 6.24: Image sequence showing the detected features and three of the position vectors that are being tracked. Reprinted with permission from [55].

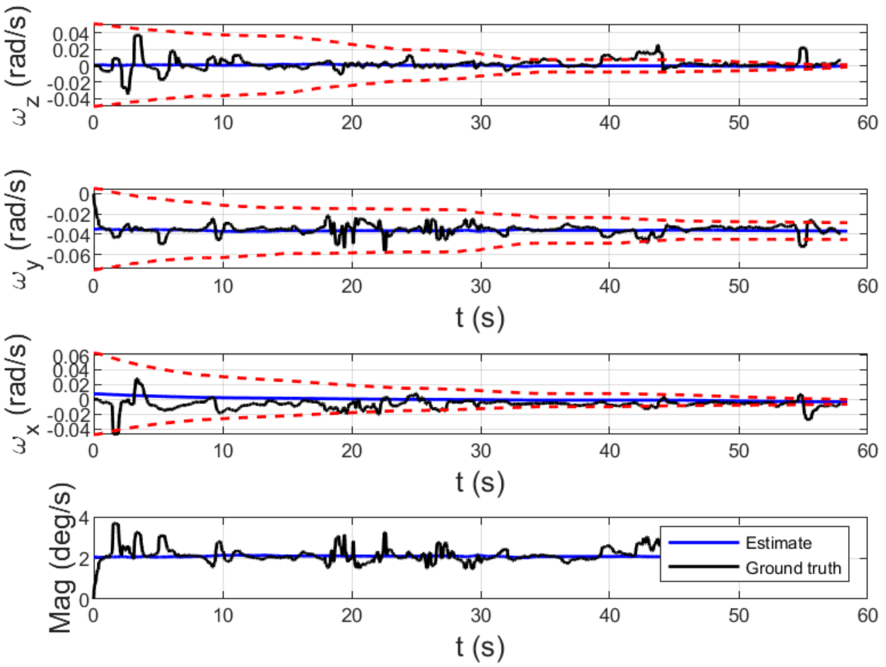


Figure 6.25: Estimated angular velocities for the target body when initial conditions for the angular rates are set to zero. The angular rates are resolved in the optical reference frame $\mathbf{o}^+$. The bottom plot shows the comparison between angular velocity vector magnitudes. Reprinted with permission from [55].

procedure outlined before. Overall, the filter was found to be well-behaved, and in contrast with the experiments that used the conventional EKF, the filter produced consistent and unbiased estimates. Figure 6.25 shows the angular rates estimate time histories for a typical experiment. The estimates for a typical experiment when the angular rates are initialized as zeros are also included, and are depicted in Figure 6.26 to show the UD partial-update EKF functionality.

6.3.8 Summary

The hardware implementation of a vision system utilized for angular rate estimation of uncooperative bodies was described in this section. The UD partial-update approach was found to produce statistically consistent results where an EKF could not. Angular rate estimation results showed that although the UD partial-update EKF takes more time to converge, convergence is

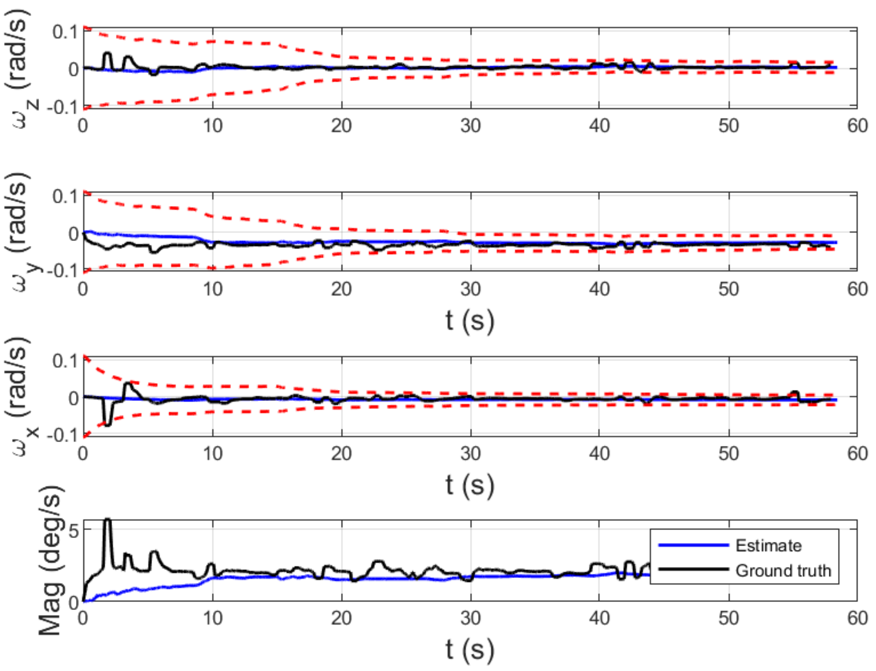


Figure 6.26: Estimated angular velocities for the target body when initial conditions for the angular rates are set to zero. The angular rates are resolved in the optical reference frame $\mathbf{o}^+$. The bottom plot shows the comparison between angular velocity vector magnitudes. Reprinted with permission from [55].

achieved, and consistent estimates are produced. In contrast with the first (with constant nuisance states) and second (varying nuisance states) partial-update filter application, this is a case where the nuisance states, namely the angular rates, are the states of interest, yet, the consistency of the results obtained via the UD partial-update filter demonstrated the power of the approach that still retains the EKF structure.

The results from all three hardware applications confirm that a partial-update EKF is a more flexible filter that accommodates real-world technical challenges in an effective and simple way. Overall, extending the EKF and Schmidt filter application capabilities, regardless of the nuisance states being constant, varying, and even if they are the states of interest.

7. CONCLUSIONS

In this dissertation, approaches to extend the practical applicability of the Extended Kalman filter (EKF) are presented. The proposed methods are based on a recent development called the partial-update filter, that already broadens the capabilities of the Schmidt-Kalman filter. The contributions of this work increase the partial-update filter robustness against numerical issues and high uncertainties, allowing the use of an extended Kalman filter framework where a more advanced filter may be needed, or where consistent estimates would not be obtained otherwise. Since the proposed techniques address fundamental problems within the Kalman filter, they apply widely just as the Schmidt modification, Joseph-update form, and factorized formulations for covariance propagation. Although this study focuses on the EKF, the findings can have a bearing on other filter forms where the measurement update appears linearly.

Based on the results of the partial-update factorized formulations presented in this dissertation, the UD form is recommended because it is more efficient and attractive for actual implementation. Although the factorized formulations require some extra work in terms of computer code, such code is not application-dependent. Thus, once extra code routines are available, the factorized filter implementations are not more complicated than a conventional filter formulation, and practically any user familiar with the Kalman filter can implement a partial-update factorized form. The findings show that the square root and UD partial-update formulations, increase the required computations of the conventional factorized forms due to the calculations related to partial-update terms, and this needs to be considered on the hardware selection process. However, the extra computational effort for relatively small systems is generally not significant to prevent the factorized partial-update formulations from their implementation on a system already using a conventional Kalman filter.

This dissertation establishes two quantitative frameworks for online partial-update weight selection: The nonlinearity-aware and the covariance-aware method. Both methods use a similar policy for update percentage selection, and although they have their virtues and limitations, the results of this research proved them to be functional and to extend the static partial-update approach's capabilities to handle high nonlinearities and uncertainties. However, the implementation of the covariance-aware method for partial-update is recommended over the nonlinearity-aware method, especially for hardware applications where the system has a large number of parameters. Overall, this study strengthens the idea that it is beneficial to vary the amount of update applied to each element in the state vector. Furthermore, the insights gained from this study may be of assistance in developing more sophisticated methods for partial-update weight selection that can consider elements like the quality of cross-covariance terms and measurement sparsity. Since these dynamic methods are entirely based on quantities inherent to the Kalman filter, like the covariance matrix and process and measurement Jacobians, the dynamic partial-update concept can be adapted into any other Kalman filter variants.

The deployment of the partial-update filter in real systems proved the filter functionality and higher consistency compared to the conventional Kalman filter. For most of the hardware applications where the partial-update filter has been used, a static weight was found via a trial and error tedious process. If the user intends to use the partial-update approach in any of its variants, it is recommended to implement a dynamic method to avoid weights tuning. Specifically, the UD covariance-aware dynamic method is recommended. Future work considers to investigate the methods for dynamic weight selection in hardware applications.

Although the partial-update filter has its limits, the findings of this research add to our understanding of the Schmidt filter and show the developments to augment the class of systems that the EKF structure can handle without over-specializing the filter formulation.

REFERENCES

[1] J. H. Ramos, K. M. Brink, and J. E. Hurtado, “Square root partial-update kalman filter,” in *22nd International Conference on Information Fusion FUSION 2019, Ottawa, Canada*, 2019.

[2] J. H. Ramos, P. Ganesh, W. Warke, K. Volle, and K. Brink, “Reef estimator: A simplified open source estimator and controller for multirotors,” in *2019 IEEE National Aerospace and Electronics Conference (NAECON)*, pp. 606–613, IEEE, 2019.

[3] M. S. Grewal and A. P. Andrews, *Kalman Filtering : Theory and Practice Using MATLAB California State University at Fullerton*, vol. 5. 2001.

[4] S. F. Schmidt, “Application of state-space methods to navigation problems,” in *Advances in control systems*, vol. 3, pp. 293–340, Elsevier, 1966.

[5] D. P. Woodbury, M. Majji, and J. L. Junkins, “Considering measurement model parameter errors in static and dynamic systems,” *The Journal of the Astronautical Sciences*, vol. 58, no. 3, pp. 461–478, 2011.

[6] K. M. Brink, “Partial-update schmidt–kalman filter,” *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 9, pp. 2214–2228, 2017.

[7] R. E. Kalman, “A new approach to linear filtering and prediction problems,” *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[8] B. P. Gibbs, *Advanced Kalman Filtering, Least-Squares and Modeling*. 2011.

[9] L. A. McGee and S. F. Schmidt, “Discovery of the Kalman Filter as a Practical Tool for Aerospace and Industry,” *NASA Technical Memorandum*, no. November, p. 21, 1985.

[10] M. Verhaegen and P. Van Dooren, “Numerical aspects of different Kalman filter implementations,” *IEEE Transactions on Automatic Control*, vol. 31, no. 10, pp. 907–917, 1986.

[11] R. J. Fitzgerald, “Divergence of the Kalman Filter,” *IEEE Transactions on Automatic Control*, vol. 16, no. 6, pp. 736–747, 1971.

[12] A. L. C. Quigley, “An approach to the control of divergence in kalman filter algorithms,” *International Journal of Control*, vol. 17, no. 4, pp. 741–746, 1973.

[13] R. H. Battin, *An Introduction to the Mathematics and Methods of Astrodynamics, revised edition.* American Institute of Aeronautics and Astronautics, 1999.

[14] J. Bellantoni and K. Dodge, “A square root formulation of the kalman-schmidt filter.,” *AIAA journal*, vol. 5, no. 7, pp. 1309–1314, 1967.

[15] P. Dyer and S. McReynolds, “Extension of square-root filtering to include process noise,” *Journal of Optimization Theory and Applications*, vol. 3, no. 6, pp. 444–458, 1969.

[16] P. Kaminski, A. Bryson, and S. Schmidt, “Discrete square root filtering: A survey of current techniques,” *IEEE Transactions on automatic control*, vol. 16, no. 6, pp. 727–736, 1971.

[17] G. J. Bierman, “Measurement Updating Using the U-D Factorization.,” *Proceedings of the IEEE Conference on Decision and Control*, vol. 12, pp. 337–346, 1975.

[18] G. J. Bierman and C. L. Thornton, “Numerical comparison of kalman filter algorithms: Orbit determination case study,” *Automatica*, vol. 13, no. 1, pp. 23–35, 1977.

[19] F. Base, “Federated Square Root Filter for,” *Ieee Transactions On Aerospace And Electronic Systems*, vol. 26, no. 3, 1959.

[20] R. G. Brown and P. Y. Hwang, *Introduction to random signals and applied Kalman filtering: with MATLAB exercises.* J Wiley & Sons, 2012.

[21] J. L. Crassidis and J. L. Junkins, *Optimal estimation of dynamic systems.* CRC press, 2011.

[22] J. R. Carpenter and C. N. D’Souza, “Navigation filter best practices,” 2018.

[23] B. P. Gibbs, *Advanced kalman filtering, least-squares and modeling: a practical handbook.* John Wiley & Sons, 2011.

[24] J. Soremen, S. Schmidt, and T. Goka, “Application of Square-Root Filtering for Spacecraft Attitude Control,” *Journal of Guidance and Control*, vol. 2, no. 5, pp. 426–433, 1979.

[25] S. Holmes, G. Klein, and D. W. Murray, “A Square Root Unscented Kalman Filter for visual monoSLAM,” *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 3710–3716, 2008.

[26] X. Li, Y. Feng, R. Huang, X. Zhang, S. Liu, and J. Ai, “The application of square-root cubature Kalman filter in SLAM for underwater robot,” *Proceedings - 2017 Chinese Automation Congress, CAC 2017*, vol. 2017-January, pp. 2183–2187, 2017.

[27] T.-s. Lou, N.-h. Chen, H. Xiong, Y.-x. Li, and L. Wang, “Ensemble consider kalman filtering,” in *2018 IEEE CSAA Guidance, Navigation and Control Conference (CGNCC)*, pp. 1–5, IEEE, 2018.

[28] D. Boggs, M. Ghil, and C. Keppenne, “A stabilized sparse-matrix ud square-root implementation of a large-state extended kalman filter,” 1995.

[29] S. Bhaumik, “Square-root cubature-quadrature Kalman filter,” *Asian Journal of Control*, vol. 16, no. 2, pp. 617–622, 2014.

[30] K. Wu, A. Ahmed, G. A. Georgiou, and S. I. Roumeliotis, “A square root inverse filter for efficient vision-aided inertial navigation on mobile devices.,” in *Robotics: Science and Systems*, vol. 2, 2015.

[31] M. G. Rutten, “Square-root unscented filtering and smoothing,” *Proceedings of the 2013 IEEE 8th International Conference on Intelligent Sensors, Sensor Networks and Information Processing: Sensing the Future, ISSNIP 2013*, vol. 1, pp. 294–299, 2013.

[32] J. Knudsen Schmidt, “Analysis of Square-Root Kalman Filters for Angles-Only Orbital Navigation and the Effects of Sensor Accuracy on State Observability,” p. 166, 2010.

[33] I. Arasaratnam and S. Haykin, “Square-root quadrature Kalman filtering,” *IEEE Transactions on Signal Processing*, vol. 56, no. 6, pp. 2589–2593, 2008.

[34] J. Mu and Y. L. Cai, “Iterated cubature Kalman filter and its application,” *2011 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems, CYBER 2011*, pp. 33–37, 2011.

[35] J. L. Geeraert and J. W. McMahon, “Square-root unscented schmidt-kalman filter,” *Journal of Guidance, Control, and Dynamics*, vol. 41, no. 1, pp. 278–285, 2018.

[36] M. Rhudy, Y. Gu, J. Gross, and M. R. Napolitano, “Evaluation of matrix square root operations for UKF within a UAV GPS/INS sensor fusion application,” *International Journal of Navigation and Observation*, vol. 2011, 2011.

[37] M. Papez and P. Pivonka, *Numerical aspects of inertial navigation*, vol. 12. IFAC, 2013.

[38] Carraro and Sartore, “Square Root Iterative Filter: Theory and Applications to Econometric Models,” *Annales d’Économie et de Statistique*, no. 6/7, p. 435, 2016.

[39] G. Y. Kulikov and M. V. Kulikova, “Square-root Kalman-like filters for estimation of stiff continuous-time stochastic systems with ill-conditioned measurements,” *IET Control Theory & Applications*, vol. 11, no. 9, pp. 1420–1425, 2017.

[40] S. N. Kane, A. Mishra, and A. K. Dutta, “Integrated GPS/DR Vehicle Navigation System Based on Sequential and Square Root Kalman Filters,” *Journal of Physics: Conference Series*, vol. 755, no. 1, 2016.

[41] Y. Zhou, Y. F. Zhang, and J. Z. Zhang, “A new adaptive square-root unscented kalman filter for nonlinear systems,” in *Applied Mechanics and Materials*, vol. 300, pp. 623–626, Trans Tech Publ, 2013.

[42] Y. Zhou, C. Zhang, Y. Zhang, and J. Zhang, “A new adaptive square-root unscented kalman filter for nonlinear systems with additive noise,” *International Journal of Aerospace Engineering*, vol. 2015, 2015.

[43] P. S. Maybeck, *Stochastic models, estimation, and control*, vol. 3. Academic press, 1982.

[44] J. M. Galante, J. Van Eepoel, C. D'Souza, and B. Patrick, "Fast Kalman filtering for relative spacecraft position and attitude estimation for the raven ISS hosted payload," *Advances in the Astronautical Sciences*, vol. 157, pp. 179–196, 2016.

[45] C. L. Thornton, "Triangular covariance factorizations for kalman filtering," *NASA TM 33-798*, 1976.

[46] R. Zanetti and C. D'Souza, "Recursive implementations of the schmidt-kalman 'consider' filter," *The Journal of the Astronautical Sciences*, vol. 60, no. 3-4, pp. 672–685, 2013.

[47] C. DSouza and R. Zanetti, "Information Formulation of the UDU Kalman Filter," *IEEE Transactions on Aerospace and Electronic Systems*, pp. 1–8, 2018.

[48] G. N. Holt and C. D. Souza, "Orion absolute navigation system progress and challenges," *Guidance, Navigation, and Control Conference*, no. August, pp. 1–17, 2012.

[49] D. Woodbury and J. Junkins, "On the consider kalman filter," in *AIAA Guidance, Navigation, and Control Conference*, p. 7752, 2010.

[50] R. Y. Novoselov, S. M. Herman, S. M. Gadaleta, and A. B. Poore, "Mitigating the effects of residual biases with Schmidt-Kalman filtering," *2005 7th International Conference on Information Fusion, FUSION*, vol. 1, pp. 358–365, 2005.

[51] C. Yang, E. Blasch, and P. Douville, "Design of Schmidt-Kalman filter for target tracking with navigation errors," *IEEE Aerospace Conference Proceedings*, no. April, 2010.

[52] T. Lou, H. Fu, Z. Wang, and Y. Zhang, "Schmidt-kalman filter for navigation biases mitigation during mars entry," *Journal of Aerospace Engineering*, vol. 28, no. 4, pp. 1–7, 2015.

[53] Y. Yang, X. Yue, and A. G. Dempster, "GPS-based onboard real-time orbit determination for leo satellites using consider Kalman filter," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 52, no. 2, pp. 769–777, 2016.

[54] Y. Li, H. Wei, M. Wu, H. Zhu, and J. Ye, "Gnss-based attitude determination via schmidt kalman filter," in *China Satellite Navigation Conference*, pp. 621–638, Springer, 2018.

[55] J. H. Ramos, T. D. Woodbury, and J. E. Hurtado, “Vision-based tracking of non-cooperative space bodies to support active attitude control detection,” in *2018 AIAA SPACE and Astronautics Forum and Exposition*, p. 5353, 2018.

[56] T. S. Lou, Z. H. Wang, M. L. Xiao, and H. M. Fu, “Multiple adaptive fading Schmidt-Kalman filter for unknown bias,” *Mathematical Problems in Engineering*, vol. 2014, 2014.

[57] A. Chakraborty, K. Brink, R. Sharma, and L. Sahawneh, “Relative pose estimation using range-only measurements with large initial uncertainty,” in *2018 Annual American Control Conference (ACC)*, pp. 5055–5061, IEEE, 2018.

[58] J. D. Jurado and J. F. Raquet, “Towards an online sensor model validation and estimation framework,” in *2018 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, pp. 1319–1325, IEEE, 2018.

[59] K. M. Brink, “Unscented partial-update schmidt–kalman filter,” *Journal of Guidance, Control, and Dynamics*, vol. 41, no. 4, pp. 929–935, 2017.

[60] J. E. Hurtado, *Kinematic and Kinetic Principles*. Lulu.com, 2012.

[61] N. Trawny, N. Trawny, S. Roumeliotis, and S. Roumeliotis, “Jacobian for conversion from Euler Angles to Quaternions,” *Robotics*, no. 612, 2005.

[62] D. O. Wheeler, D. P. Koch, J. S. Jackson, T. W. McLain, and R. W. Beard, “Relative Navigation: A Keyframe-Based Approach for Observable GPS-Degraded Navigation,” *IEEE Control Systems*, vol. 38, pp. 30–48, August 2018.

[63] R. Zanetti and C. N. D’Souza, “Observability Analysis and Filter Design for the Orion Earth-Moon Attitude Filter,” *Journal of Guidance, Control, and Dynamics*, vol. 39, no. 2, pp. 201–213, 2016.

[64] N. Trawny, A. I. Mourikis, S. I. Roumeliotis, A. E. Johnson, and J. F. Montgomery, “Vision-aided inertial navigation for pin-point landing using observations of mapped landmarks,” *Journal of Field Robotics*, vol. 24, no. 5, pp. 357–378, 2007.

[65] M. Li and A. I. Mourikis, “High-precision, consistent EKF-based visual-inertial odometry,” *The International Journal of Robotics Research*, vol. 32, no. 6, pp. 690–711, 2013.

[66] N. Trawny and S. I. Roumeliotis, “Indirect Kalman filter for 3D attitude estimation,” *University of Minnesota, Dept. of Comp. Sci. & Eng., Tech. Rep*, vol. 2, p. 2005, 2005.

[67] D. P. Woodbury, *Accounting for parameter uncertainty in reduced-order static and dynamic systems*. Texas A&M University, 2011.

[68] D. Simon, *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons, 2006.

[69] G. H. Golub and C. F. Van Loan, *Matrix computations*, vol. 3. JHU Press, 2012.

[70] T. K. Moon and W. C. Stirling, *Mathematical methods and algorithms for signal processing*, vol. 1. Prentice hall Upper Saddle River, NJ, 2000.

[71] B. D. Anderson and J. B. Moore, *Optimal filtering*. Courier Corporation, 2012.

[72] S. Gilbert, *Introduction to linear algebra*, vol. 3. Wellesley-Cambridge Press Wellesley, MA, 1993.

[73] S. Julier, J. Uhlmann, and H. F. Durrant-Whyte, “A new method for the nonlinear transformation of means and covariances in filters and estimators,” *IEEE Transactions on automatic control*, vol. 45, no. 3, pp. 477–482, 2000.

[74] K. R. Rao, D. N. Kim, and J. J. Hwang, *Fast Fourier transform-algorithms and applications*. Springer Science & Business Media, 2011.

[75] G. J. Bierman, *Factorization methods for discrete sequential estimation*. Courier Corporation, 2006.

[76] C. Thornton, “Triangular covariance factorizations for kalman filtering, phd thesis, university of california at los angeles,” 1976.

[77] N. A. Carlson, “Fast triangular formulation of the square root filter.,” *AIAA journal*, vol. 11, no. 9, pp. 1259–1265, 1973.

[78] F. Gustafsson, *Statistical sensor fusion*. Studentlitteratur, 2010.

[79] K. Brink and A. Soloviev, “Filter-based calibration for an imu and multi-camera system,” in *Proceedings of the 2012 IEEE/ION Position, Location and Navigation Symposium*, pp. 730–739, IEEE, 2012.

[80] C. Yang, E. Blasch, and P. Douville, “Design of schmidt-kalman filter for target tracking with navigation errors,” in *2010 IEEE Aerospace Conference*, pp. 1–12, IEEE, 2010.

[81] A. I. Mourikis, N. Trawny, S. I. Roumeliotis, A. E. Johnson, A. Ansar, and L. Matthies, “Vision-aided inertial navigation for spacecraft entry, descent, and landing,” *IEEE Transactions on Robotics*, vol. 25, no. 2, pp. 264–280, 2009.

[82] D. P. Koch, T. W. McLain, and K. M. Brink, “Multi-sensor robust relative estimation framework for {GPS}-denied multirotor aircraft,” in *Unmanned Aircraft Systems (ICUAS), 2016 International Conference on*, pp. 589–597, IEEE, 2016.

[83] P. Furgale, J. Rehder, and R. Siegwart, “Unified temporal and spatial calibration for multi-sensor systems,” in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1280–1286, IEEE, 2013.

[84] E. Mair, M. Fleps, M. Suppa, and D. Burschka, “Spatio-temporal initialization for imu to camera registration,” in *2011 IEEE International Conference on Robotics and Biomimetics*, pp. 557–564, IEEE, 2011.

[85] J. Kelly and G. S. Sukhatme, “Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration,” *The International Journal of Robotics Research*, vol. 30, no. 1, pp. 56–79, 2011.

[86] F. M. Mirzaei and S. I. Roumeliotis, “A kalman filter-based algorithm for imu-camera calibration: Observability analysis and performance evaluation,” *IEEE transactions on robotics*, vol. 24, no. 5, pp. 1143–1156, 2008.

[87] M. Li and A. I. Mourikis, “High-precision, consistent ekf-based visual-inertial odometry,” *The International Journal of Robotics Research*, vol. 32, no. 6, pp. 690–711, 2013.

[88] Z. Yang and S. Shen, “Monocular visual–inertial state estimation with online initialization and camera–imu extrinsic calibration,” *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 1, pp. 39–51, 2016.

[89] J. Rehder and R. Siegwart, “Camera/imu calibration revisited,” *IEEE Sensors Journal*, vol. 17, no. 11, pp. 3257–3268, 2017.

[90] J. Lobo and J. Dias, “Relative pose calibration between visual and inertial sensors,” *The International Journal of Robotics Research*, vol. 26, no. 6, pp. 561–575, 2007.

[91] L. Meier, P. Tanskanen, L. Heng, G. H. Lee, F. Fraundorfer, and M. Pollefeys, “Pixhawk: A micro aerial vehicle design for autonomous flight using onboard computer vision,” *Autonomous Robots*, vol. 33, no. 1-2, pp. 21–39, 2012.

[92] C. Caubel, F. Morra, and B. Vignau-lous, “Flying toy,” Nov. 29 2016. US Patent D772,991.

[93] T. Krajnik, V. Vonasek, D. Fiser, and J. Faigl, “Ar-drone as a platform for robotic research and education,” in *Research and Education in Robotics - EUROBOT 2011* (D. Obdrzalek and A. Gottscheber, eds.), (Berlin, Heidelberg), pp. 172–186, Springer Berlin Heidelberg, 2011.

[94] D. O. Wheeler, D. P. Koch, J. S. Jackson, G. J. Ellingson, P. W. Nyholm, T. W. McLain, and R. W. Beard, “Relative navigation of autonomous gps-degraded micro air vehicles,” *Autonomous Robots*, pp. 1–20, 2020.

[95] P. Gąsior, S. Gardecki, J. Gośliński, and W. Giernacki, “Estimation of altitude and vertical velocity for multirotor aerial vehicle using kalman filter,” in *Recent Advances in Automation, Robotics and Measuring Techniques*, pp. 377–385, Springer, 2014.

[96] S. Wang and Y. Yang, “Quadrotor aircraft attitude estimation and control based on kalman filter,” in *Proceedings of the 31st Chinese Control Conference*, pp. 5634–5639, IEEE, 2012.

[97] M. L. Anderson, K. M. Brink, and A. R. Willis, “Real-time visual odometry covariance estimation for unmanned air vehicle navigation,” *Journal of Guidance, Control, and Dynamics*, vol. 42, no. 6, pp. 1272–1288, 2019.

[98] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, p. 5, Kobe, Japan, 2009.

[99] J. Jackson, G. Ellingson, and T. McLain, "Rosflight: A lightweight, inexpensive mav research and development tool," in *Unmanned Aircraft Systems (ICUAS), 2016 International Conference on*, pp. 758–762, IEEE, 2016.

[100] J. Zhang, M. Kaess, and S. Singh, "Real-time depth enhanced monocular odometry," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4973–4980, IEEE, 2014.

[101] G. Szafranski and R. Czyba, "Different approaches of pid control uav type quadrotor," 2011.

[102] A. L. Salih, M. Moghavvemi, H. A. Mohamed, and K. S. Gaeid, "Flight pid controller design for a uav quadrotor," *Scientific research and essays*, vol. 5, no. 23, pp. 3660–3667, 2010.

[103] R. De Maesschalck, D. Jouan-Rimbaud, and D. L. Massart, "The mahalanobis distance," *Chemometrics and intelligent laboratory systems*, vol. 50, no. 1, pp. 1–18, 2000.

[104] T. D. Woodbury, J. H. Ramos, and J. E. Hurtado, "Attitude-based classification of noncooperative bodies for motion characterization and active control detection," in *2018 AIAA SPACE and Astronautics Forum and Exposition*, p. 5226, 2018.

[105] J. Shi *et al.*, "Good features to track," in *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pp. 593–600, IEEE, 1994.

[106] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," 1981.

[107] *The OpenCV Reference Manual*, 2.4.8.0 ed., February 2018.

[108] N. W. Oumer and G. Panin, “3d point tracking and pose estimation of a space object using stereo images,” in *Pattern Recognition (ICPR), 2012 21st International Conference on*, pp. 796–800, IEEE, 2012.

[109] A. R. Willis, L. R. Sahawneh, and K. M. Brink, “Benchmarking real-time rgbd odometry for light-duty uavs,” in *Three-Dimensional Imaging, Visualization, and Display 2016*, vol. 9867, p. 98670O, International Society for Optics and Photonics, 2016.

[110] O. Sorkine-Hornung and M. Rabinovich, “Least-squares rigid motion using svd,” *Computing*, vol. 1, p. 1, 2017.

APPENDIX A

HARDWARE PARAMETERS

The REEF estimator parameters used to generate the results shown in Section 6.2.5 are included here.

NOTE: The initial value of the Z position in the Z estimator is set at -0.25m since the sonar altimeter is mounted at a height of 0.25m from the ground. The sonar used in this application is only capable of measuring heights greater than 0.25m.

$$\mathbf{P}_{xy_o} = \text{diag}\begin{bmatrix}0.01 & 0.01 & 0.03 & 0.02 & 0.14 & 0.14\end{bmatrix}$$
$$\mathbf{Q}_{xy} = \text{diag}\begin{bmatrix}0 & 0 & 0.03 & 0.03 & 0.1 & 0.1\end{bmatrix}$$
$$\mathbf{x}_{xy_o} = \begin{bmatrix}0 & 0 & 0 & 0 & 0 & 0\end{bmatrix}$$
$$\boldsymbol{\beta}_{xy} = \begin{bmatrix}1 & 1 & 0.01 & 0.01 & 0.01 & 0.01\end{bmatrix}$$
$$\mathbf{P}_{z_o} = \text{diag}\begin{bmatrix}0.025 & 1.0 & 0.09\end{bmatrix}$$
$$\mathbf{Q}_{z} = \text{diag}\begin{bmatrix}0.03 & 0.001\end{bmatrix}$$
$$\mathbf{x}_{z_o} = \begin{bmatrix}-0.25 & 0 & 0\end{bmatrix}$$
$$\boldsymbol{\beta}_{z} = \begin{bmatrix}1.0 & 1.0 & 0.5\end{bmatrix}$$