Adaptive Extended Kalman Filtering Strategies for Autonomous Relative Navigation of Formation Flying Spacecraft

by
Cory Tyler Fraser

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs
in partial fulfillment of the requirements for the degree of

Master of Applied Science
in
Aerospace Engineering

Carleton University
Ottawa, Ontario

This thesis is dedicated to my family;
For your guidance, when the path was dark and lost.
For your support, when the path was difficult and harsh.
For your love, transcending the space this path placed between us.

Non est ad astra mollis e terris via

There is no easy way
from the earth to the stars

Abstract

This research addresses the relative navigation problem for spacecraft formation flying missions in near-Earth orbit. Technological and economic influences have expedited the miniaturization of spacecraft systems, popularizing the notion of satellite teamwork due to the potential reductions in overall costs and complexity. It is now feasible to use a coordinated group of close-proximity satellites, known as a formation, to autonomously perform rendezvous, on-orbit servicing and science-based missions for a fraction of the resources a large-scale satellite would require. Despite the growing interest in formation flying spacecraft however, only a limited number of applications have been flight validated as a result of the difficulties associated with accurately determining the relative motion of the spacecraft within a formation.

To enhance the capabilities of onboard autonomous guidance, navigation and control systems, this thesis presents the development of two adaptive extended Kalman filter navigation algorithms for spacecraft formation flying. The proposed adaptive filters are capable of updating the internal noise characteristics of the Kalman filter in real time, and are viable in all orbit scenarios, including highly elliptical orbits in the presence of perturbations. The first Kalman filter approach uses maximum likelihood estimation techniques to derive analytical adaptations laws for the filter, which are then improved through the novel inclusion of an intrinsic smoothing routine. The second approach uses an embedded fuzzy logic system based on a covariance-matching analysis of the filter residuals, where the fuzzy system has been specifically designed for the spacecraft navigation problem at hand. Numerical simulations of three spacecraft formations are used to demonstrate that the proposed adaptive navigation algorithms are appreciably more robust to filter initialization errors, dynamics modelling deficiencies, and measurement noise than the standard Kalman filter.

Acknowledgements

To Dr. Steve Ulrich, formally my supervisor, but more importantly my mentor, my professor, and my friend; your steadfast support and leadership have made my studies in Ottawa immensely fulfilling. Thank you for providing this opportunity to continually learn from your expertise, and for instilling confidence and direction throughout the span of our collaboration. The dedication you have shown to me, and all of the members of our research group, is remarkable. It has been an honor to work with you, and a whole lot of fun too.

To the professors who have guided me along this journey, including Bruce Burlton, Tarik Kaya, Alex Ellery, Victor Aitken, Howard Schwartz, and many others from Carleton University. The knowledge and wisdom you have shared has been invaluable to my growth as an engineer. And from the days at Dalhousie University, my thanks in particular to Professors Darrel Doman, Jeff Dahn, Timothy Little, and Guy Kember. You managed to imbue a fascination with space elevators, the laws of nature, the rules of mathematics, and the process of learning, into a young undergraduate student trying to find his way. There is no doubt that your teachings laid the foundation upon which this research was built, and I'm truly thankful.

To my friends in Ottawa, Nova Scotia, Montreal, and abroad; you are quite possibly my most influential teachers of all. For teaching me to persevere, to dream, to fail, to succeed, to grow, and to love... there are not enough words to describe everything you have done for me, and my vocabulary lacks the words needed to describe how important you all are to me.

To my family; you are the reason I am here today, and the person I am now is a reflection of everything you have given to me. Thanks to my father for passing on a passion for engineering and problem solving, to my mother for teaching me to love unconditionally and commit wholeheartedly, and to my siblings for all their humour.

Lastly, my appreciation goes out to all those who have invested in my future. This research was financially supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC), through the Alexander Graham Bell Canada Graduate Scholarship. Further financial support was provided by an Ontario Graduate Scholarship, various graduate student entrance scholarships, and a teaching assistant position provided by the Mechanical and Aerospace Engineering Department.

Table of Contents

Abstract **iii**

Acknowledgements **iv**

List of Figures **x**

List of Tables **xiii**

Nomenclature **xx**

1 Introduction **1**

1.1 Motivation 1
1.2 Problem Statement 4
1.3 Previous Work 5
1.4 Thesis Objectives 12
1.5 Contributions 13
1.6 Thesis Organization 14

2 Spacecraft Formation Flying **15**

2.1 Introduction 15
2.2 Frames of Reference 17
2.2.1 The Earth-Centered Inertial (ECI) Frame 17
2.2.2 The Earth-Centered Earth-Fixed (ECEF) Frame 18
2.2.3 The Perifocal Frame 18
2.2.4 The Local-Vertical Local-Horizontal (LVLH) Frame 19
2.3 The Two-Body Problem 20
2.3.1 Kepler's Laws for Celestial Bodies 20
2.3.2 Newton's Law of Universal Gravitation 21
2.4 Classical Orbital Elements 22
2.5 Non-Keplerian Orbital Motion 25

2.5.1 Gravity Perturbations due to Earth's Oblateness 27
2.5.2 Luni-Solar Third-body Accelerations 28
2.5.3 Atmospheric Drag 31
2.5.4 Solar Radiation Pressure 34
2.5.5 Secondary Perturbations 36
2.6 Spacecraft Formation Flying 38
2.6.1 Numerical Orbit Propagation 38
2.6.2 Nonlinear Equations of Relative Motion 41
2.6.3 Regarding the Hill-Clohessy-Wiltshire Equations 43
2.7 Formation Configurations 44
2.7.1 Case 1: PRISMA Formation 44
2.7.2 Case 2: PROBA-3 Formation 46
2.7.3 Case 3: Projected Elliptical Orbit Formation in LEO 48
2.8 Chapter Summary 50

3 Extended Kalman Filtering **51**
3.1 Introduction 51
3.2 The Extended Kalman Filter 54
3.3 Dynamics Modelling 58
3.4 Measurement Modelling 62
3.5 Observability Analysis 65
3.6 Numerical Considerations 67
3.6.1 Integration within the EKF 67
3.6.2 Matrix Inversion Using Equation Solving 68
3.6.3 Matrix Inversion Using Cholesky Decomposition 68
3.7 Overview of the Simulation Environment 69
3.8 Chapter Summary 71

4 Adaptive Kalman Filtering using Maximum Likelihood Estimation **72**
4.1 Introduction 72
4.2 Statistical Preliminaries 73
4.3 The Likelihood Function of the EKF 74
4.4 Solving the MLE Problem 76
4.4.1 Adaptations for the Process Noise Covariance 78
4.4.2 Adaptations for the Measurement Noise Covariance 80
4.4.3 Maintaining Positive Definite Symmetry 81
4.5 The Extended Kalman Smoother 83
4.6 Chapter Summary 84

5 Fuzzy Adaptive Kalman Filtering **85**
5.1 Introduction . . . 85
5.2 First Principles of Fuzzy Set Theory . . . 86
5.3 Overview of FAEKF Architecture . . . 87
5.3.1 Criteria for Innovation-based Covariance Matching . . . 89
5.3.2 Adaptation of the Measurement Noise Covariance . . . 90
5.3.3 Adaptation of the Process Noise Covariance . . . 91
5.4 Fuzzification . . . 91
5.4.1 For the Measurement Noise Covariance . . . 92
5.4.2 For the Process Noise Covariance . . . 93
5.4.3 Membership Functions . . . 93
5.4.4 Discussion on FLS Design and Tuning . . . 96
5.4.5 Logic Rule Bases . . . 98
5.5 The Inference Mechanism . . . 98
5.6 Defuzzification . . . 99
5.7 Chapter Summary . . . 101

6 Numerical Analysis of the Formation Navigation Algorithms **102**
6.1 Introduction . . . 102
6.1.1 Configuring the Simulator . . . 103
6.1.2 Quantifying Performance . . . 104
6.2 Case 1: PRISMA Formation . . . 107
6.2.1 PRISMA Simulation Conditions . . . 107
6.2.2 PRISMA Simulation Results . . . 110
6.2.3 PRISMA Simulation Conclusions . . . 123
6.3 Case 2: PROBA-3 Formation . . . 124
6.3.1 PROBA-3 Simulation Conditions . . . 124
6.3.2 PROBA-3 Simulation Results . . . 127
6.3.3 PROBA-3 Simulation Conclusions . . . 140
6.4 Case 3: PEO in LEO Formation . . . 142
6.4.1 PEO Simulation Conditions . . . 142
6.4.2 PEO Simulation Results . . . 145
6.4.3 PEO Simulation Conclusions . . . 158
6.5 Chapter Summary . . . 159

7 Conclusion **160**
7.1 Thesis Summary . . . 160
7.2 Significance of Work . . . 163

7.3 Recommendations for Future Work 164
7.3.1 Developing a Realistic GPS Measurement Model 164
7.3.2 Closed-loop GNC Testing 164
7.3.3 Configuring the EKF for Alternate Measurements 165
7.3.4 Expanding the Fuzzy Logic System 165
7.3.5 Optimization of the Adaptation Settings 165

References **166**

Appendix A List of Formation Flying Missions **178**

Appendix B Essential Astrodynamics **183**
B.1 Matrix Operations and Reference Frames 183
B.2 Converting Between Reference Frames 186
B.3 Conversions with Classical Orbital Elements 190
B.3.1 Orbital Elements to Inertial States 190
B.3.2 Inertial States to Orbital Elements 191
B.4 Positions of the Sun and Moon 193
B.5 Atmospheric Density Modelling 196
B.6 Deriving the Equations of Relative Motion 199
B.6.1 Partial Derivatives of X-EOM 204
B.6.2 Partial Derivatives of Y-EOM 205
B.6.3 Partial Derivatives of Z-EOM 206
B.6.4 Partial Derivatives of θ-EOM 207
B.6.5 Partial Derivatives of r_t-EOM 208
B.6.6 Hill-Clohessy-Wiltshire Relative Dynamics 209
B.7 Physical Constants and Planetary Parameters 211

Appendix C Kalman Filter Theory **212**
C.1 The Linear Kalman Filter 212
C.2 The Extended Kalman Filter 216
C.3 Summary of the EKF Implementation 219

Appendix D Observability Analysis **220**
D.1 Introduction to Nonlinear Observability 220
D.2 Summary of the System Dynamics 222
D.3 Observability for Relative Spacecraft Motion 223

Appendix E Orbit Propagator MATLAB Code **228**
E.1 Spacecraft Accelerations 229
E.2 Acceleration Due to Drag 231
E.3 Harris-Priester Atmospheric Density Model 232
E.4 Acceleration Due to Solar Radiation Pressure 235
E.5 Solar Illumination Factor for SRP 236
E.6 Acceleration Due to Thirdy Body Gravity 237

Appendix F EKF MATLAB Code **238**
F.1 Simulation Run 239
F.2 EKF Propagation Phase 245
F.3 Nonlinear Dynamics Equations 247
F.4 EKF Linearized State Transition Matrix 248
F.5 EKF Correction Phase 251
F.6 Observability Analysis 254

Appendix G MLE-AEKF MATLAB Code **257**
G.1 MLE Adaptive Extended Kalman Filter 258
G.2 Positive Semi-definite Symmetric Matrix Check 264

Appendix H FAEKF MATLAB Code **266**
H.1 Initialization of the FLS 267
H.2 SISO Fuzzy Logic System 270
H.3 Fuzzy Adaptations 272

List of Figures

1.1 Artistic Renderings of Formation Flying Missions 2
1.2 Distribution Timeline of Formation Flying Missions 3
1.3 Successful Formation Flying Missions 7

2.1 Architecture of a GNC System 16
2.2 The Earth-Centered Inertial Reference Frame 17
2.3 The Earth-Centered Earth-Fixed Reference Frame 18
2.4 The Local Vertical Local Horizontal Reference Frame 19
2.5 The Classical Orbital Elements 23
2.6 Comparative Magnitudes of Orbit Perturbations 26
2.7 Third-body Gravity Geometry 29
2.8 Orbit Propagator Block Diagram 40
2.9 Relative Orbit of the PRISMA Formation 45
2.10 Relative Orbit of the PROBA-3 Formation 47
2.11 Relative Orbit of a Projected Elliptical Orbit Formation in LEO . . . 49

3.1 Kalman Filter Block Diagram 57
3.2 Simulated Measurements Block Diagram 63
3.3 Example of Simulated Position Measurements 64
3.4 Overview of the Formation Flying Relative Navigation Simulator . . . 69

5.1 General Fuzzy Logic Operations 86
5.2 Block Diagram Representation of the FAEKF 87
5.3 Block Diagram Representing the Fuzzy Adaptation Scheme 88
5.4 Input Membership Functions for the Normalized Input Variables . . . 95
5.5 Output Membership Functions for the Normalized Output Variables . 95
5.6 Nonlinear Map of the Fuzzy Logic System 100

6.1 EKF Relative Position Results for the PRISMA Formation 111
6.2 MLE-AEKF Relative Position Results for the PRISMA Formation . . 112

6.3 FAEKF Relative Position Results for the PRISMA Formation 113
6.4 Comparison of the EKF and MLE-AEKF Relative Position Estimation Errors for the PRISMA Formation 114
6.5 Comparison of the EKF and FAEKF Relative Position Estimation Errors for the PRISMA Formation 115
6.6 EKF Relative Velocity Results for the PRISMA Formation 116
6.7 MLE-AEKF Relative Velocity Results for the PRISMA Formation . . 117
6.8 FAEKF Relative Velocity Results for the PRISMA Formation 118
6.9 Comparison of the EKF and MLE-AEKF Relative Velocity Estimation Errors for the PRISMA Formation 119
6.10 Comparison of the EKF and FAEKF Relative Velocity Estimation Errors for the PRISMA Formation 120
6.11 Relative Position Error Distributions for the PRISMA Formation . . 121
6.12 Relative Velocity Error Distributions for the PRISMA Formation . . 122
6.13 EKF Relative Position Results for the PROBA-3 Formation 128
6.14 MLE-AEKF Relative Position Results for the PROBA-3 Formation . 129
6.15 FAEKF Relative Position Results for the PROBA-3 Formation 130
6.16 Comparison of the EKF and MLE-AEKF Relative Position Estimation Errors for the PROBA-3 Formation 131
6.17 Comparison of the EKF and FAEKF Relative Position Estimation Errors for the PROBA-3 Formation 132
6.18 EKF Relative Velocity Results for the PROBA-3 Formation 133
6.19 MLE-AEKF Relative Velocity Results for the PROBA-3 Formation . 134
6.20 FAEKF Relative Velocity Results for the PROBA-3 Formation 135
6.21 Comparison of the EKF and MLE-AEKF Relative Velocity Estimation Errors for the PROBA-3 Formation 136
6.22 Comparison of the EKF and FAEKF Relative Velocity Estimation Errors for the PROBA-3 Formation 137
6.23 Relative Position Error Distributions for the PROBA-3 Formation . . 138
6.24 Relative Velocity Error Distributions for the PROBA-3 Formation . . 139
6.25 EKF Relative Position Results for the PEO Formation 146
6.26 MLE-AEKF Relative Position Results for the PEO Formation 147
6.27 PEO Relative Position Results for the PEO Formation 148
6.28 Comparison of the EKF and MLE-AEKF Relative Position Estimation Errors for the PEO Formation 149
6.29 Comparison of the EKF and FAEKF Relative Position Estimation Errors for the PEO Formation 150

6.30 EKF Relative Velocity Results for the PEO Formation 151
6.31 MLE-AEKF Relative Velocity Results for the PEO Formation 152
6.32 FAEKF Relative Velocity Results for the PEO Formation 153
6.33 Comparison of the EKF and MLE-AEKF Relative Velocity Estimation Errors for the PEO Formation 154
6.34 Comparison of the EKF and FAEKF Relative Velocity Estimation Errors for the PEO Formation 155
6.35 Relative Position Error Distributions for the PEO Formation 156
6.36 Relative Velocity Error Distributions for the PEO Formation 157

List of Tables

2.1 Gravitational Parameters for the Sun and Moon 30
2.2 Initial Orbital Elements of the PRISMA Formation 44
2.3 Spacecraft Properties for the PRISMA Formation 45
2.4 Initial Orbital Elements of the PROBA-3 Formation 46
2.5 Spacecraft Properties for the PROBA-3 Formation 46
2.6 Initial Orbital Elements of the PEO Formation 48
2.7 Spacecraft Properties for the PEO Formation 48

3.1 Summary of the Dynamics Models 70
3.2 Summary of the Measurement Models 70

5.1 Rule Table for the SISO Fuzzy Logic System 98

6.1 Simulator Settings for the PRISMA Formation 107
6.2 Initial EKF Settings for the PRISMA Formation 108
6.3 MLE-AEKF Settings for the PRISMA Formation 109
6.4 FAEKF Settings for the PRISMA Formation 109
6.5 Average RMS Errors and Runtimes from PRISMA Simulation 110
6.6 3D-RMS Errors from PRISMA Simulation 123
6.7 Simulator Settings for the PROBA-3 Formation 124
6.8 Initial EKF Settings for the PROBA-3 Formation 125
6.9 MLE-AEKF Settings for the PROBA-3 Formation 126
6.10 FAEKF Settings for the PROBA-3 Formation 126
6.11 Average RMS Errors and Runtimes from PROBA-3 Simulation 127
6.12 3D-RMS Errors from PROBA-3 Simulation 140
6.13 Simulator Settings for the PEO in LEO Formation 142
6.14 Initial EKF Settings for the PEO in LEO Formation 143
6.15 MLE-AEKF Settings for the PEO in LEO Formation 144
6.16 FAEKF Settings for the PEO in LEO Formation 144
6.17 Average RMS Errors and Runtimes from PEO Simulation 145

6.18 3D-RMS Errors from PEO Simulation 158

A.1 List of Formation Flying Missions 178

B.1 Lunar Position Coefficients 195
B.2 Harris-Priester Density Coefficients 198
B.3 Planetary Parameters from DE430-431 Ephemerides 211
B.4 Earth Parameters and Constants from WGS84 211

Nomenclature

Roman Symbols

A_d	Area exposed to drag
A_i^j	Fuzzy sets
$\tilde{A}_i^j$	Fuzzy input linguistic value
A_r	Area exposed to solar radiation pressure
a	Orbit semi-major axis
$a_{A_i^j}$	Curvature of a sigmoidal fuzzy membership function
$\vec{a}_{drag}$	Perturbing acceleration due to drag
$\vec{a}_{J_2}$	Perturbing acceleration due to J_2
$\vec{a}_{3B}$	Perturbing acceleration due to a third body
$\tilde{B}_q^p$	Fuzzy output linguistic value
b	Orbit semi-minor axis
$b_{A_i^j}$	Center of a sigmoidal fuzzy membership function
$\mathbf{C}_{21}$	Rotation matrix from frame 1 to frame 2
C_d	Aerodynamic drag coefficient
C_r	Solar radiation pressure coefficient
$c_{A_i^j}$	Center of a Gaussian fuzzy membership function
$\mathbf{e}_k$	Estimation error
e	Orbit eccentricity
$e_\oplus$	Ellipsoidal eccentricity of the Earth
$\mathbf{F}_k$	Linearized dynamics matrix
$\mathbf{f}_k$	Dynamics matrix
$\mathbf{G}_k$	Kalman smoothing gain
G	Universal gravitational constant
g	Fuzzy system input scaling gain
$\mathbf{H}_k$	Linearized measurement matrix
$\mathbf{h}_k$	Measurement matrix
$\vec{h}$	Orbit specific angular momentum vector

h	Fuzzy system output scaling gain
$\mathbf{I}$	The identity matrix
$\vec{\mathcal{I}}$	Reference vectrix defining the ECI Frame
i	Orbit inclination
J	Pseudo-likelihood cost function
J_k	Zonal harmonics of the gravitational potential
J_2	Second zonal coefficient of the gravitational potential
$\mathbf{K}_k$	Kalman gain matrix
$\vec{\mathcal{L}}$	Reference vectrix defining the LVLH Frame
M	Mean Anomaly
m	Mass
n_0	Mean orbital motion
n_o	Empirical orbit exponent for the Harris-Priester model
$\mathcal{O}$	Observability matrix
$\mathbf{P}_k$	State error covariance matrix
$P_{\odot}$	Solar radiation pressure at a distance of 1 AU
P_k	Legendre polynomials
p	Orbit semi-latus rectum
p_Z	Probability mass function of a discrete random variable Z
$\mathbf{Q}_k$	Process noise covariance matrix
$\mathbf{R}_k$	Measurement noise covariance matrix
$\vec{r}$	Position vector of the spacecraft
$\mathcal{U}_i$	Universe of discourse for a fuzzy input
$\mathbf{u}_k$	Control input vector
$\hat{\mathbf{u}}_b$	Unit vector defining the apex of the atmospheric density bulge
u_i	Fuzzy system normalized crisp input
$\tilde{u}_i$	Fuzzy system constant linguistic variable
$\mathbf{v}_k$	Measurement noise vector
$\vec{v}$	Velocity vector of the spacecraft
$\vec{v}_r$	Velocity vector of the spacecraft relative to the atmosphere
$\mathbf{w}_k$	Process noise vector
$\mathbf{x}_k$	State vector
$\hat{\mathbf{x}}_k$	Estimated state vector
x	Radial position of the chaser, defined in the LVLH frame
y	Along-track position of the chaser, defined in the LVLH frame
z	Cross-track position of the chaser, defined in the LVLH frame
$\mathbf{z}_k$	Measurement vector

Greek Symbols

α_k	MLE adaptation parameter
α_{RA}	Right ascension
δ_{DEC}	Declination
ε	Obliquity of the ecliptic
ε_k	Fuzzy system covariance adaptation parameter
$\boldsymbol{\eta}_{DOM}$	Degree of mismatch
η_{DOD}	Degree of divergence
$\gamma\Upsilon$	Direction of the vernal equinox
Λ_q	Universe of discourse for a fuzzy output
λ_k	Fuzzy system normalized crisp output
λ_{lag}	Angular lag between the Sun and the diurnal atmospheric density bulge
λ_{lon}	Geodetic longitude
μ	Gravitational parameter
$\mu_{A_i^j}$	Fuzzy system input membership function
$\mu_{B_p^q}$	Fuzzy system output membership function
Ω	Orbit right ascension of the ascending node (RAAN)
ω	Orbit argument of perigee
$\omega_\oplus$	Rotational velocity of the Earth
$\boldsymbol{\Phi}_k$	State transition matrix
Φ	Gravitational potential function of the Earth
ϕ_{lat}	Geodetic latitude
Ψ	Angle to the diurnal atmospheric density bulge apex
$\vec{\rho}$	Position of the chaser relative to the target
ρ_{atm}	Density of the atmosphere
$\sigma_{A_i^j}$	Width of a Gaussian fuzzy membership function
σ	Standard deviation of a random variable
σ^2	Variance of a random variable
$\boldsymbol{\Theta}$	Estimation parameters
θ	True anomaly for the target spacecraft
θ_{GMT}	Greenwich mean time
ν	Innovation (measurement pre-fit residual)
ν_{IF}	Illumination factor
ν^+	Measurement post-fit residual
ξ	Trace of the theoretical residuals covariance matrix
$\hat{\xi}$	Trace of the observed residuals covariance matrix

Superscripts

$+$	An *a posterori* parameter
$-$	An *a priori* parameter
c	Indicator for the chaser spacecraft
q	Indicator for a fuzzy system parameter for $\mathbf{Q}$
r	Indicator for a fuzzy system parameter for $\mathbf{R}$
T	Matrix transpose
t	Indicator for the target spacecraft
$\times$	Cross product matrix operator

Subscripts

c	Indicator for the chaser spacecraft
F	Earth-Centered Earth-Fixed Frame
I	Earth-Centered Inertial Frame
k	A parameter observed at a discrete time t_k
L	Local-Vertical Local-Horizontal Frame
P	Perifocal Frame
t	Indicator for the target spacecraft
x	X-component within the respective reference frame
y	Y-component within the respective reference frame
z	Z-component within the respective reference frame
0	Initial state
$\odot$	Parameter of the Sun
$\oplus$	Parameter of the Earth
$\leftmoon$	Parameter of the Moon

Other Symbols

$\|\mathbf{A}\|$	Euclidean (L_2) norm of matrix $\mathbf{A}$
$\|\mathbf{A}\|$	Determinant of matrix $\mathbf{A}$
$\mathbf{A}^{-1}$	Inverse of matrix $\mathbf{A}$
$\|\vec{a}\|$	Magnitude of vector $\vec{a}$
$\dot{a}$	First time derivative of a
$\ddot{a}$	Second time derivative of a
$\mathcal{N}(\cdot)$	Normal distribution
$\mathcal{U}(\cdot)$	Uniform distribution
$E(\cdot)$	Expectation operator

$\nabla(\cdot)$	Gradient operator
$L(\cdot)$	Likelihood function
$\prod(\cdot)$	Product operator
$\sum(\cdot)$	Summation operator

Acronyms

ADCS	Attitude Determination and Control Subsystem
AEKF	Adaptive Extended Kalman Filter
AU	Astronomical Unit
AVANTI	Autonomous Vision Approach Navigation and Target Identification
BEESAT	Berlin Experimental and Educational Satellite
BIROS	Bi-spectral InfraRed Optical System
CanX	Canadian Advanced Nanospace eXperiment
CNES	French Aerospace Center, *Centre National D'études Spatiales*
COE	Classical Orbital Elements
CSA	Canadian Space Agency
DARPA	Defense Advanced Research Projects Agency
DE	Development Ephemerides
DEM	Digital Elevation Model
DLR	German Aerospace Center, *Deutsches Zentrum für Luft-und Raumfahrt*
DOD	Degree of Divergence
DOF	Degrees of Freedom
DOM	Degree of Mismatch
ECEF	Earth-Centered Earth-Fixed Reference Frame
ECI	Earth-Centered Inertial Reference Frame
EPS	Electrical Power Subsystem
ESA	European Space Agency
EXO-S	Exo-Starshade
FAEKF	Fuzzy Adaptive Extended Kalman Filter
FFIORD	Formation Flying In-Orbit Ranging Demonstration
FLS	Fuzzy Logic System
GEO	Geosynchronous Equatorial Orbit
GMT	Greenwich Mean Time
GNC	Guidance, Navigation, and Control
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GRACE	Gravity Recovery and Climate Experiment

HEO	Highly Elliptical Orbit
IID	Independent Identically Distributed
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
LEO	Low Earth Orbit
LPO	Lagrange Point Orbit
LVLH	Local-Vertical Local-Horizontal Reference Frame
MATLAB	Matrix Laboratory
MISO	Multi-Input Single-Output
MLE-AEKF	MLE Adaptive Extended Kalman Filter
MLE	Maximum Likelihood Estimation
MMAE	Multiple Model Adaptive Estimation
MMS	Magnetospheric Multiscale Mission
NASA	National Aeronautics and Space Administration
NOAA	National Oceanic and Atmospheric Administration
ODE	Ordinary Differential Equation
OD	Orbit Determination
PDS	Positive Definite Symmetric
PMF	Probability Mass Function
PRISMA	Prototype Research Instruments and Space Mission Advancement
PROBA	Project for Onboard Autonomy
PSDS	Positive Semi-Definite Symmetric
RAAN	Right Ascension of the Ascending Node
RCM	Residuals Covariance Matrix
R&D	Research and Development
RMS	Root Mean Square
ROE	Relative Orbital Elements
SAR	Synthetic Aperture Radar
SISO	Single-Input Single-Output
SRCL	Spacecraft Robotics and Control Lab at Carleton University
SRP	Solar Radiation Pressure
TCS	Thermal Control Subsystem
TTC	Telemetry, Tracking and Command
TU	Technical University
UAV	Unmanned Aerial Vehicle
UKF	Unscented Kalman Filter
WGS	World Geodetic System

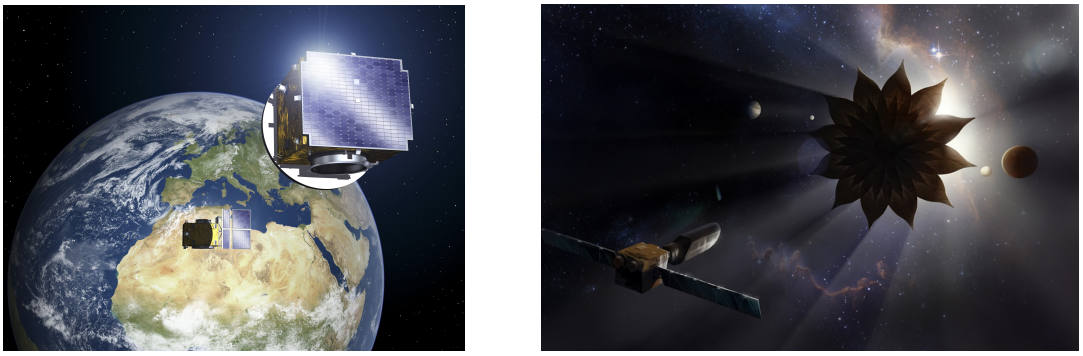
Chapter 1

Introduction

1.1 Motivation

Spacecraft formation flying represents one of the largest transformative technological shifts to influence the space industry in the past two decades. Yet, the implications of this change in space mission design philosophy are only beginning to manifest; a typical visualization of spacecraft orbiting the Earth contains a single school bus-sized, solar panel-covered structure, silently hurtling through an inky void with only our planet for accompaniment. Rarely will one initially consider a scenario where two or more spacecraft appear in that scene, such as the satellite duo shown in Figure 1.1a. Instinctively the risk of collision should preclude the need for two high-velocity objects to operate in such close proximity. However, while concerns of satellite collisions and interference were indeed warranted during the early years of the space age, recent demonstrations of advanced Guidance, Navigation and Control (GNC) systems onboard the PRISMA [1], CanX-4 and CanX-5 [2], and AVANTI [3, 4] spacecraft maintained control accuracy at the sub-meter level, presenting a new host of possibilities for the safe utilization of co-orbiting, cooperative formations.

The benefits of formation flying stem from the distribution of payload systems and operational functionality across multiple, smaller spacecraft. Significant cost savings in the design and launch of micro-satellites [5–7] reinforce the current trends of reducing spacecraft mass and complexity, and coordinating the operations of multiple smaller units for a single objective allows a formation to complete science that would not be possible with a single, monolithic spacecraft alone [8]. Many measurement systems that require precise baseline separations, particularly synthetic aperture radar [9, 10], gravimetry [11], and deep-space observation [12], are significantly enhanced by the flexibility of spacecraft formations and the ease of redundancy they entail.



(a) **(b)**

Figure 1.1: Artistic renderings of formation flying missions. (a) The Observer and Occulter spacecraft of the PROBA-3 mission, which will be used to study the Sun's corona (courtesy of ESA). (b) The Wide-Field Infrared Survey Telescope and Starshade of the EXO-S mission. Scheduled for deployment to the Earth-Sun L2 Lagrange point in 2025, EXO-S will be searching for terrestrial extrasolar planets (courtesy of NASA).

Formation flying spacecraft can further be improved with the addition of autonomous systems, capable of quickly responding and adapting to unexpected changes in the space environment or the mission requirements. By contrast, spacecraft controlled from the ground rely on teams of engineers to analyze telemetry and upload new commands to the onboard GNC software manually, a process which consumes time, accrues personnel costs, and ultimately limits responsiveness and performance of the satellite [13]. In fact, ground-based control architectures are also restricted by radio signal communication delays associated with the large distances between the ground station and the satellite. For these reasons, autonomous formation flying spacecraft can improve the performance of current near-Earth missions and reduce the on-ground operational requirements, while simultaneously opening the door for future deep-space endeavours like the Exo-Starshade (EXO-S) mission planned by the National Aeronautics and Space Administration (NASA) [14], illustrated in Figure 1.1b.

With the growing popularity and advantages of distributed space systems, the wide array of on-going experiments are generally classified by primary mission objective into one of the following categories: Space Science, Earth Science, Earth Observation and Remote Sensing, and Technology Demonstration. The majority of formation flying missions flown to date have been technology demonstrations, and over 70% of the current missions operate in Low Earth Orbit (LEO) [15], which encompasses all orbits with an apogee altitude less than 2 000 km. The remaining missions operate in orbital regimes more commonly employed for monolithic spacecraft, such as Highly Elliptical Orbit (HEO), Geosynchronous Equatorial Orbit (GEO), and deep-space Lagrange Point Orbit (LPO). While both the mission objective and the orbit type will

significantly influence the design of a GNC system, the obtainable control accuracy is fundamentally driven by the navigation system [13], and more importantly for formation flying, the relative navigation system. To be explicit for the context of this research, we'll adopt the formal definition of navigation used by NASA, which states that navigation is:

> *"The measurement and computation necessary to determine the present spacecraft position and velocity."*
>
> — Apollo GNC Progress Report [16]

Given that more than 20 formation flying missions have been launched in the past two decades, and another 10 launches are planned within the next decade (as per the summary given in Figure 1.2), continual advancement of formation flying navigation software is necessary to meet the growing demands of future missions. Virtually all modern navigation solutions are obtained using a combination of measurement systems, mathematical models, and well-established state estimation techniques, but the recent push for autonomy gives merit to the investigation of novel navigation algorithms capable of adapting to the environment without the need for human intervention - these adaptable navigation systems are the motivation for this research.

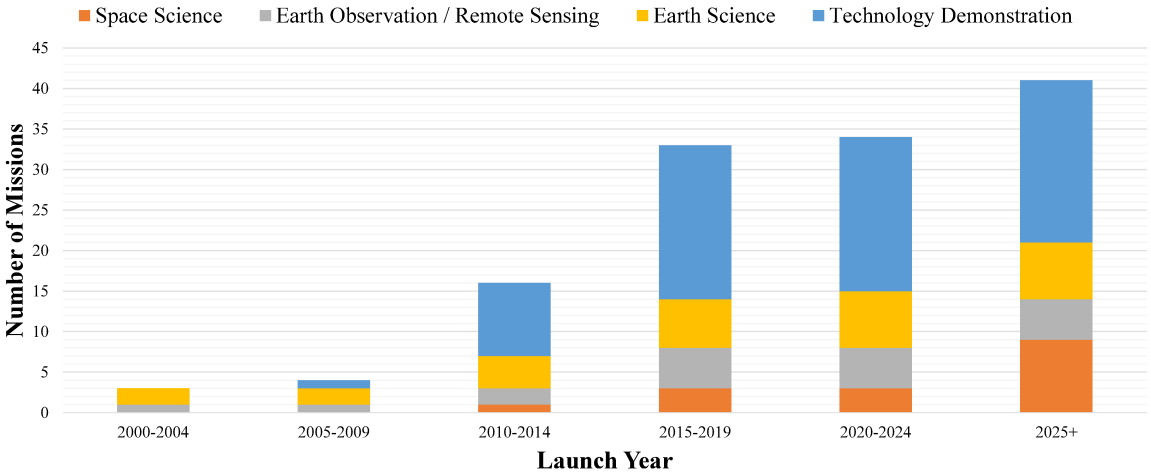


Figure 1.2: A cumulative distribution of past and planned spacecraft formation flying missions, based on the list of compiled missions shown in Appendix A. Primary mission objectives are also indicated, illustrating current research trends and an emphasis on technological demonstration.

1.2 Problem Statement

The increasing performance capabilities and complexity of spacecraft formation flying missions continue to drive stringent requirements for precise guidance, navigation and control systems. As renowned formation flying expert Dr. Simone D'Amico further emphasizes [13], autonomous systems are more than just relevant for future rendezvous, docking, and formation flying applications; they are mandatory for proximity operations, in-orbit servicing and robust collision avoidance procedures, all of which will become critical as LEO becomes more crowded (e.g. see *Kessler Syndrome* as addressed by Hovell and Ulrich [17]). One of the foremost technological challenges to be addressed is therein the development of onboard navigation systems that are accurate and robust for a wide array of formation configurations, operating modes, and orbital environments, while simultaneously adhering to the limited processing capabilities available on spacecraft computers. Knowledge of the relative positions and velocities of the formation has direct implications on both the guidance and control subsystems, so the computationally elegant Kalman filter has historically been selected to provide relative navigation solutions [18].

The Kalman filter [19] is a recursive algorithm that provides an efficient method to fuse information from a measurement system with predictions made by a dynamical model of the expected system behaviour, thereby estimating the system state more accurately than either the measurements or dynamics alone. Not surprisingly, the measurement system, the dynamics model, and the fusion algorithms chosen within the filter each have a significant impact on the final navigation solution. Furthermore, a standard Kalman filter assumes fixed uncertainties in stochastic error parameters within the dynamics and measurement models, and these uncertainties propagate through the filter [20]. Typically the effects of these uncertainties can be mitigated to a reasonable degree through a process known as *tuning* the Kalman filter, but this requires a particular amount of human interaction and experience [21].

The purpose of this thesis is to address the relative state estimation problem for a dual-spacecraft formation, through the design of adaptive Kalman filter strategies. More specifically, this work encompasses the development of a suitable onboard dynamics model describing the motion of one spacecraft with respect to another, the selection of a measurement system capable of providing useful observations of the formation, and the construction of filtering architectures that adapt to uncertainties in the system. As a performance benchmark, the relative position accuracy should be at least two orders of magnitude lower than the separation of the spacecraft, based on current GNC system performances reviewed by Di Mauro *et al.* [15].

1.3 Previous Work

Research into the development and testing of spacecraft GNC systems has been highly active since its inception in the early 1960's, so this section presents the pertinent texts and publications specific to navigation techniques, and filtering algorithms, and formation flying. The state-of-the-art technologies established within this section form the basis upon which novel navigation algorithms can be proposed. Thus, before detailing the contributions of this thesis, it's first important to understand past and current modalities, and their respective limitations.

Comprehensive literature studies relating to spacecraft formation flying have been put forth in recent years, and the latest work by Di Mauro *et al.* [15] highlights 37 formation flying missions planned between 2000 and 2025. Similarly, the review by Bandyopadhyay *et al.* [22] classifies a number of small satellite formation missions, focusing on spacecraft with wet masses less than 10 kg. These summaries provide an excellent introduction into the trends and types of formation missions in existence, but do not address the finer intricacies of the navigation subsystems. Although the assessment made by Sullivan *et al.* [23] is slightly more restricted in scope and details relative motion dynamic models used throughout the past 60 years, a succinct compilation of recent relative navigation filters is not readily available. Nevertheless, the previous surveys were used to create an updated list of formation flying missions that is given in Appendix A, and the remainder of this section addresses particular aspects of the spacecraft navigation problem.

Early Navigation Filtering Techniques

The original methods for estimating the position and velocity of spacecraft considered only a single spacecraft, and were interested in determining the state of the spacecraft with respect to the Earth, or the *absolute* spacecraft state. This process of *orbit determination* (OD) historically uses regressive or statistical techniques to estimate the spacecraft state, by post-processing observations on-ground after a large number of measurements are taken over an extended period of time. This method is referred to commonly as *batch processing* [24]. Algorithms such as Least-Squares or its modified variants are frequently used in batch processing to identify the defining parameters of the spacecraft orbit based on a set of measurements, and the known position of the spacecraft can then be propagated forward in time using a dynamics model until a sufficient number of measurements are collected such that the next batch estimation can be performed. The drawbacks of this technique however relate to the time that is needed to acquire the set of measurements; propagation of the spacecraft

state during the collection period introduces noise due to errors in the dynamics model, so confidence in the real-time state estimates will continually decrease until the next batch of measurements are processed. Additionally, a significant amount of computational power is needed to analyze the accumulated data, hence why this processing is typically performed on-ground. Although the limited number of real-time state estimates from batch techniques are not suitable for the navigation required in formation flying, overall accuracy of the OD routine will be superior, since the resulting orbit parameters are estimated from a significant number of observations. For more information on the general trends of OD techniques over the past 50 years, the article by Vetter [25] provides an excellent retrospective.

Many missions demonstrate the benefits and drawbacks of batch processing, including the GRACE Mission [26], an early demonstration of formation flying capabilities. Neither spacecraft in the GRACE formation had onboard navigation routines, however since the spacecraft maintained a separation distance of roughly 200 km, autonomous navigation was not a requirement. TanDEM-X and TerraSAR-X [15] alternatively, relied on real-time state estimates and therefore used sequential estimation routines within their navigation suite. This provided real-time information to the spacecraft that was used for guidance and control purposes, while more precise batch estimation was later performed on-ground to improve estimates of the baseline separation distance needed for the science payload. Although multiple sequential estimation methods exist, the extended Kalman filter (EKF) is by far the most popular for real-time spacecraft navigation.

Extended Kalman Filtering

The theoretical details of the Kalman filter will be presented in subsequent chapters of this thesis, so a review of several high-profile formation flying missions that used EKF-based navigation systems is given here to demonstrate the flight heritage and on-orbit performance capabilities of the EKF. Firstly, one of the most promising demonstrations of autonomous formation flying technologies to date has been the Prototype Research Instruments and Space Mission Technology Advancement (PRISMA) mission, in operation since 2010 [27]. This program is a joint collaboration involving the Swedish Space Corporation, the German Aerospace Center (DRL), the Technical University of Denmark, and the French Centre National D'études Spatiales (CNES). PRISMA consists of a passive target spacecraft called Tango, and a 6 degrees-of-freedom (DOF) controlled chaser spacecraft called Mango, which are shown in Figure 1.3a. On-orbit experiments with the PRISMA formation demonstrated relative navigation accuracy on the centimeter level [28] for minimum separations of 150 m using an EKF.

(a) **(b)**

Figure 1.3: Successful demonstrations of formation flying technologies. (a) An artist's rendering of the PRISMA formation with Mango in the foreground and Tango in the background (courtesy of DLR). (b) Potential applications for the PRISMA mission include radiation measurements of deep-space objects (courtesy of CNES).

Another experiment entitled the Formation Flying In-Orbit Ranging Demonstration (FFIORD) was designed by CNES and utilized the PRISMA spacecraft to validate a formation flying radio frequency sensor package. An EKF was used during the FFIORD testing to provide estimates of the relative spacecraft positions and velocities; furthermore, several onboard dynamics models were tested within the EKF, including the well-known Clohessy-Wiltshire and Yamanaka-Ankersen equations [29].

The Canadian Advanced Nanosatellite eXperiment (CanX) launched a dual spacecraft formation in June 2014, consisting of the Can-X4 and Can-X5 spacecraft [2]. Absolute state estimation for both spacecraft is computed on the chaser, from which the relative navigation solution is transformed into into a local reference frame for relative motion control. The relative navigation algorithm (also only on board the chaser CanX-5 spacecraft) uses carrier phase differential GPS measurements and an EKF to determine the position of the chaser with respect to the target. Computational efficiency was the driving factor that led to single-difference pseudorange and carrier phase measurements being selected, which similarly improves the convergence performance of the filter [30]. While the context of GPS measurement types are beyond the scope of the proposed filtering algorithms developed for this thesis, use of the EKF onboard the CanX-4/5 mission reinforces its efficacy in autonomous navigation.

Building on the success of early dual spacecraft formations, the Magnetoshperic Multiscale (MMS) mission developed by NASA launched four spacecraft into a tetrahedral formation in March 2015 [31]. Precise formation keeping control was required during science-collection operations, which required knowledge of the spacecraft separation to within 100 m. This accuracy was accomplished by onboard navigation software that used GPS measurements, an EKF, and a high fidelity onboard dynamics

model [32]. Thereafter, the Autonomous Vision Approach Navigation and Target Identification (AVANTI) experiment [3] was performed in June 2016, using the Bispectral InfraRed Optical System (BIROS) and Berlin Experimental and Educational Satellite 4 (BEESAT-4) spacecraft of the DRL Firebird mission [33, 34]. These spacecraft used line-of-sight measurements and an EKF to estimate the relative spacecraft state in terms of Relative Orbital Elements (ROEs), but were only able to achieve relative navigation accuracy at the meter level due to observability issues with using line-of-sight methods [35].

Although the Kalman filter has proven to be quite effective in spacecraft applications, there are still limitations that can compromise the accuracy of the final state estimates provided by the EKF. Deficiencies in the mathematical model used internally by the filter to describe the real-world system can lead to filter divergence over time if the modelling mismatch is too large [36]. For spacecraft, this can be the result from things like neglecting higher order gravitational effects or simply linearizing dynamic terms, for example. Similarly, dynamic biases due to improperly defined system parameters, such as the ballistic coefficient of the spacecraft, the frontal surface area exposed to the Sun, or the atmospheric density, will further degrade the performance of the EKF if left unaddressed [37]. A subset of Kalman filtering research has thus investigated ways to modify the performance of the standard EKF in real-time based on observations of the filter performance, which has since become known as adaptive Kalman filtering.

Adaptive Kalman Filtering

Contrary to the density of in-flight Kalman filtering examples, adaptive Kalman filtering research is still in its infancy. To date, there have been no widely accepted or reported adaptive Kalman filter techniques implemented on spacecraft in orbit, and the development and demonstration of adaptation methods have therefore been restricted to simulation environments, hardware-in-the-loop testing, and theoretical propositions. Undoubtedly the field of adaptive filtering has nevertheless been an active research area, so it is useful to recount several adaptation schemes that have been explored in the past.

The initial suggestion of adapting the Kalman filter to account for unknown *a priori* process and measurement noise statistics was made by Mehra in 1970 [38], and his subsequent work in 1972 [39] identified four stochastic approaches that could possibly be applied. The first approach relies upon Bayesian Estimation, whereby recursive equations for the *a posteriori* probability density of the state and covariance matrices for the Kalman filter are obtained using conditional probabilities and integration.

In practice, the evaluation of these probability integrals over the defined subspace of possible adaptation parameters requires a large number of calculations, making this approach less suitable for real-time implementation. More recently, the Bayesian approach has given rise to a class of algorithms designed for Multiple Model Adaptive Estimation (MMAE) [40], which entails estimating the state of a system through a probabilistic weighting of the average estimates provided by multiple, uniquely-tuned Kalman filters. This is equivalent to designing and running multiple Kalman filters and selecting the best result, which is a somewhat *ad hoc* approach that is not well suited for spacecraft operations. Further discussions on Bayesian estimation and MMAE methods can be found in the text by Maybeck [41].

The second approach proposed by Mehra [39] uses Correlation Methods for analyzing the autocorrelation functions of either the filter outputs or the filter innovations. This method requires the consideration of the past n-lag steps, where n represents the number of states, and applications are typically used in the time-invariant Linear Kalman filter framework [42]. Both of these features make correlation methods less attractive for real-time spacecraft navigation, where the nonlinear dynamics commonly employ a significant number of states that are further dependent on the selected measurement system.

The two remaining approaches proposed by Mehra, referred to respectively as Maximum Likelihood Estimation (MLE) and Covariance-Matching, are both promising for real-time implementation, and the technical details of both of these methods are explored in more detail throughout the later chapters of this thesis.

Using MLE within the Kalman filter is in essence a technique that blends state estimation and system identification; by defining a set of parameters that influence a likelihood function based on the observed measurements and the filter state estimates (*i.e.*, parameters such as the noise covariance matrices, dynamics state transition matrices, or the input mapping matrices), adaptations to these parameters within the Kalman filter can be updated in order to maximize the observed likelihood function. Mehra and Bayard presented an application of a maximum likelihood estimation adaptive extended Kalman filter (MLE-AEKF) for spacecraft attitude estimation in 1995 [43], and subsequent work by Mohamed and Schwarz [20] and Hide and Moore [40] have applied MLE-AEKF schemes to INS/GPS navigation problems. Busse and How further introduced the MLE technique into the spacecraft formation scenario, where it was demonstrated that improvements in relative position knowledge could be obtained compared to the EKF [44].

The final technique commonly considered when adapting the Kalman filter is Covariance Matching, where observations of the filter innovations covariance are

compared with the theoretical innovations covariance predicted by the filter. The goal of Covariance Matching is then to update parameters within the EKF such that the theoretical and observed innovation covariances match, which corresponds to optimal performance of the EKF. In 2011, Jiancheng *et al.* [45] used such an approach to address in-flight alignment errors developing in aircraft GPS measurements, however the proposed implementation actually eliminates the dependency of the Kalman filter on the measurement noise covariance matrix entirely. In the calculation of the Kalman gain, the theoretical innovations covariance matrix is simply replaced with the observed innovations covariance matrix, and so this technique restricts the applicability of the adaptation process to situations where only uncertainty in the measurement noise exists.

Other authors have however applied covariance matching schemes to adapt both the process noise and measurement noise statistics within the Kalman filter, such as the work in [46] which relates to estimating land vehicle position through combining IMU and GPS data. The idea of covariance matching has also given rise to new methods of adapting the Kalman filter in a variety of disciplines, including techniques using optimization [47], forgetting factors [48], neural network inference [49], and fuzzy logic. To conclude this literature review, only the topics associated with fuzzy logic adaptive Kalman filtering will be discussed further.

Fuzzy Adaptive Kalman Filtering

The original concept of Fuzzy Logic was developed by Professor Lotfi Zadeh of the University of California at Berkeley during the 1960's, and was published in a series of articles in 1965 [50], 1968 [51], and 1973 [52]. Classical logic operations rely of the exact definitions of a statement; for example, *1* or *0*, *on* or *off*, and *true* or *false*, are all examples of binary logic sets. Fuzzy logic on the other hand provides a method to include varying degrees of a value within the extremes defined by traditional logic. As an illustrative example, the designer of a temperature controller operating with classical logic would design a system based on the values of 1 or 0, which would correspond to either hot or cold. Using fuzzy logic however, the designer can include other classification of the states, such as cold, chilly, warm, hot, and very hot, thereby improving the sensitivity and versatility of their control laws. The use of fuzzy set theory therefor allows human experience, intuition, and reasoning to be utilized in computer logic decision making processes. Since the inception of fuzzy logic and the associated mathematics, fuzzy systems have appeared in many fields of control theory and state estimation, including within the Kalman filter.

To the best of the author's knowledge, fuzzy adaptive extended Kalman filter (FAEKF) techniques have not yet been applied to the problem of autonomous spacecraft formation flying. Nevertheless, the use of fuzzy adaptive filters in GPS/INS sensor fusion has been readily investigated since the early 2000s, and is shown to improve navigation accuracy. Sasiadek *et al.* [53] originally used a weighted EKF [54–56] to scale the process and measurement covariances exponentially with time. By decreasing the noise covariance as more measurement are taken, the goal of this technique is to place more weight on newer measurements. The base of the scaling-factor was adapted using an innovations-based fuzzy tuning mechanism, where the fuzzy rule set compares the estimated covariance of the EKF residuals with the actual measurement residuals. The adaptation scheme references the mean of the residuals, which from the definition of a Gaussian noise process should be zero. Additional robustness of Sasiadek's FAEKF was proven in [57] when non-white noise was added to the GPS measurements. Work on expanding Sasiadek's technique to incorporate a UKF is also in progress, as per [58].

More recently, da Silva and da Cruz [59] strictly used covariance matching for UAV GPS/INS integration, and applied multiplicative factors to adjust the order of magnitude of the noise covariances. Inputs to the fuzzy logic system were based on bias and oscillation metrics, which built on a method used by Ali [60] that looked at adapting diagonal elements of the measurement noise covariance only, using individual additive components. A useful result from these studies showed that by only adapting the diagonal elements, the fuzzy inference system can be treated as multiple single-input single-output systems, thereby reducing computational complexity.

Additional adaptation mechanisms for adapting the process noise covariances have been explored by Tseng and Lin [61], using scalar metrics of the innovation covariance matrices instead of the full covariance matrices themselves. While this does result in a lumped adaptation of the EKF, this work demonstrated that suitable adjustments to the process noise covariance could be obtained through proper scaling within the FLS, while similarly reducing the FLS routines to simple single-input, single output (SISO) systems. However, their proposed fuzzy adaptation laws were developed for Earth-borne applications and embedded within a cubature Kalman filter, which is a form of Bayesian filter that is computationally more demanding than the EKF. It can therefore be seen that FAEKF methods are indeed capable of improving the estimation accuracy over the traditional EKF, yet the technology has not yet been applied to relative navigation in the context of spacecraft formation flying.

1.4 Thesis Objectives

While many methods for navigation filtering have been investigated in recent years, the concept of autonomous navigation is a nascent technology that requires intelligently designed techniques suitable for on-orbit spacecraft. The objective of this thesis is therefore to design novel adaptive extended Kalman filter strategies for near-Earth orbit spacecraft formations. Two adaptation methods are considered here, one based on the concepts of maximum likelihood estimation, and another that uses a specialized fuzzy logic system. The adaptive EKFs must demonstrate robustness to a variety of measurement and initialization errors, adaptability to multiple formation configurations, and a minimal amount of computational burden.

The first portion of this work develops a numerical simulator for the relative motion of a dual spacecraft formation, and includes the design of a standard EKF based on the nonlinear equations of relative motion. Using the nonlinear equations of relative motion reduces the modelling inaccuracies introduced by common restrictive simplifications of the dynamics, which are typically only applicable to nearly circular orbits [23]. Noise-corrupted observations of the spacecraft formation are generated through a simplified measurement system, and are then processed by the EKF to obtain estimates of the relative states. The EKF serves as the baseline case for comparison with the adaptive EKF techniques.

Adaptive filtering is then explored through the design of an MLE-AEKF, building on the work of Mohamed and Schwarz [20], which highlights several of the advantages and disadvantages of adaptive techniques. The analytic solution of the MLE-AEKF limits the amount of control the user has over the design of the filter, but demonstrates useful filter metrics that can be selected as inputs into other adaptation schemes.

Finally, this thesis presents the design of a fuzzy logic system that can be embedded within an EKF, allowing the filter to autonomously tune the internal noise covariances for optimal performance given unknown *a priori* information about the dynamics and measurement noises. Contrary to previous work with FAEKF methods, the approach taken here includes unique membership function definitions and methods to maintain the positive semi-definite constraints of the covariance matrices. The performance of this FAEKF is then compared with measurement-only, EKF, and MLE-AEKF navigation solutions through numerical simulations. Three realistic formation configurations are considered, and the results of these case studies demonstrate improved relative position and velocity accuracy is obtained by using the adaptive EKF techniques over the standard EKF.

1.5 Contributions

The completion of this thesis has yielded a number of contributions to the fields of spacecraft formation flying, relative navigation in near-Earth orbit, and adaptive Kalman filtering techniques. In particular, the following milestones have been achieved:

- The development of a simulation environment capable of modelling near-Earth orbit spacecraft motion, and the associated measurement observations;
- The design of an EKF for relative navigation, based on the general nonlinear dynamics equations of relative motion;
- The implementation of an MLE-AEKF, with the novel inclusion of an intrinsic smoothing algorithm;
- The formulation and development a novel FAEKF, using fuzzy inference systems based on covariance matching;
- The validation of the adaptive EKF techniques, through a numerical comparison of the relative navigation solutions obtained from each method.

1.6 Thesis Organization

To help the reader develop the context and appropriate insight for each component of the completed research, the remainder of this thesis is divided into a number of chapters. Chapter 2: *Spacecraft Formation Flying* introduces orbital mechanics, along with the associated dynamics the describe how spacecraft move in orbit. Specifically including a description of Keplerian motion, orbital perturbations, and a derivation of the exact nonlinear equations of relative motion for co-orbiting spacecraft, this chapter describes the physical systems that are used in the development of the EKF. Three spacecraft formations will also be presented, and simulations of their nominal relative trajectories are shown.

A overview of Kalman filter theory and its extension to nonlinear systems is provided in Chapter 3: *Extended Kalman Filtering*. Following this review, the design of an EKF for relative formation navigation is presented. A suitable linearized dynamics model is derived for use within the filter, and the simulated measurement system is identified. Making use of Maximum Likelihood Estimation techniques, an innovations-based adaptive EKF is developed in Chapter 4: *Adaptive Kalman Filtering using Maximum Likelihood Estimation*. The MLE-AEKF uses observations of the state error residuals to update the noise covariance matrices within the filter, where the adaptation equations are derived to maximize the likelihood of the measurements given the current state innovations. An introduction to MLE methods is used to formulate the EKF estimation problem, after which adaptations laws for updating the filter process and measurement noise covariance matrices are derived. Lastly, a smoother routine is introduced into the MLE-AEKF to further improve performance.

Building on the insights obtained in the design of the MLE-AEKF, an alternate adaptive EKF using a novel fuzzy adaptation mechanism is developed in Chapter 5: *Fuzzy Adaptive Kalman Filtering*. The FAEKF utilizes a fuzzy inference system based on smoothed observations of the EKF residuals as performance metrics. Methods for adapting the process and measurement noise covariance matrices are devised using a covariance matching methodology. Chapter 6: *Numerical Analysis of the Formation Navigation Algorithms* demonstrates the performance of the designed FAEKF, MLE-AEKF, and standard EKF. Three spacecraft formations in unique orbital configurations are simulated, and a comparison of the resulting estimation accuracies of the filters is analyzed thoroughly.

Chapter 7: *Conclusion* contains final thoughts and a summary of the research completed for this thesis. The contributions to the techniques developed here are framed in the context of spacecraft formation flying technologies, and recommendations for future work are given.

Chapter 2

Spacecraft Formation Flying

Before exploring the filtering techniques developed for this research, the following chapter introduces the spacecraft navigation problem and establishes the underlying framework that dictates how spacecraft move with respect to the Earth, and with respect to each other. The formations considered in this work consist of two spacecraft, referred to as the *target* and the *chaser*, and understanding the relative motion between these spacecraft is a critical step in addressing formation navigation. A review of the relevant mathematical definitions, references frames, and orbital mechanics is therefore presented, after which Keplerian orbital motion is briefly considered and expanded upon to include the four primary orbital perturbations. Equations of relative motion are presented, and to conclude, numerical simulations of three realistic spacecraft formations are used to illustrate the real-world relative motion scenarios that will be considered during the design and testing of the proposed filtering algorithms.

2.1 Introduction

At the highest level, a spacecraft consists of a number of unique subsystems: the communications subsystem, the electrical power subsystem (EPS), the thermal control subsystem (TCS), the propulsion subsystem, the structures subsystem, the telemetry tracking and command (TTC) subsystem, the attitude determination and control subsystem (ADCS), and the guidance, navigation and control (GNC) subsystem [62]. A detailed investigation of each subsystem is well beyond the scope of this thesis, but suffice it to say that the navigation problem addressed by this research is a particular operation within the GNC system.

A typical GNC schematic is shown in Fig. 2.1, detailing the interconnectivity of each component of the system. Indeed other research conducted within Carleton's Spacecraft Robotics and Control Lab (SRCL) focuses on aspects of guidance and

control [63], but along with the previous discussion in Chapter 1, the figure below illustrates how the achievable control accuracy of a spacecraft depends on the accuracy of the guidance system, which in turn relies upon an accurate knowledge of the current spacecraft state. Common sensors for determining the spacecraft states include GPS receivers, optical and infrared cameras, gyroscopes, and Sun sensors, but all of these contain characteristic errors due to measurement noise. The purpose of the navigation system is therefore to process measurements collected by the sensors in order to obtain an estimate of the current spacecraft state with a higher level of accuracy then was provided by the measurements alone.

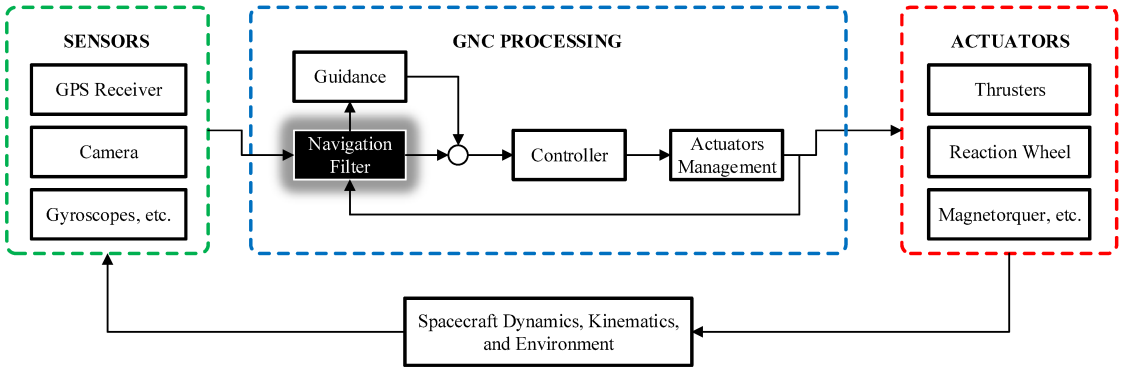


Figure 2.1: General overview of a complete GNC system.

Since access to actual spacecraft hardware and flight data is difficult and expensive to obtain, the majority of preliminary space system design relies on models and simulations of the space environment that realistically mimic how spacecraft would behave in orbit. For the research conducted here, the dynamics of these simulations are commonly referred to as the *real-world models*, or the *truth models*. In reality the spacecraft navigation algorithm cannot have an exact knowledge of the dynamics of the system, so similarly the navigation software developed here will have its own *onboard dynamics model* that is of lower fidelity than the simulated truth model. Thus, it is possible to determine if the navigation solutions are closely matching the true spacecraft states despite the modelling deficiencies within the filter. The following sections will introduce the necessary reference frames and dynamics models necessary to develop the simulated truth orbits of the spacecraft formation.

2.2 Frames of Reference

Several reference frames are used within this thesis to designate the coordinate systems in which the spacecraft dynamics equations and relative motion are defined. This section provides a succinct introduction to the Earth-Centered Inertial (ECI) frame $\mathcal{F}_I$, the Earth-Centered Earth-Fixed (ECEF) frame $\mathcal{F}_F$, the perifocal frame $\mathcal{F}_P$, and the Local-Vertical Local-Horizontal (LVLH) frame $\mathcal{F}_L$. A primer on matrix notation, vectrix notation, and conversions between these frames is given in Appendix B.

2.2.1 The Earth-Centered Inertial (ECI) Frame

When considering near-Earth spacecraft, treating the *geocentric* (Earth-centered) system as an inertial reference frame significantly simplifies the mathematics involved with deriving velocity and acceleration components. The ECI reference frame therefore describes a non-rotating, non-accelerating three-dimensional coordinate system with an origin at the center-of-mass of the Earth. The fundamental plane of this system aligns with the equator, and the direction of the Vernal Equinox (γ_Υ) is used to align the orthonormal unit vector $\vec{\mathcal{I}}_x$. The $\vec{\mathcal{I}}_z$ axis is aligned with the Earth's axis of rotation, and the direction of $\vec{\mathcal{I}}_y$ is found by completing the triad for a right-handed system. The ECI frame is used to describe the *absolute* position and velocity of a spacecraft, and may be referred to in other literature as the *Geocentric Equatorial Inertial* frame or the *Conventional Inertial System.*

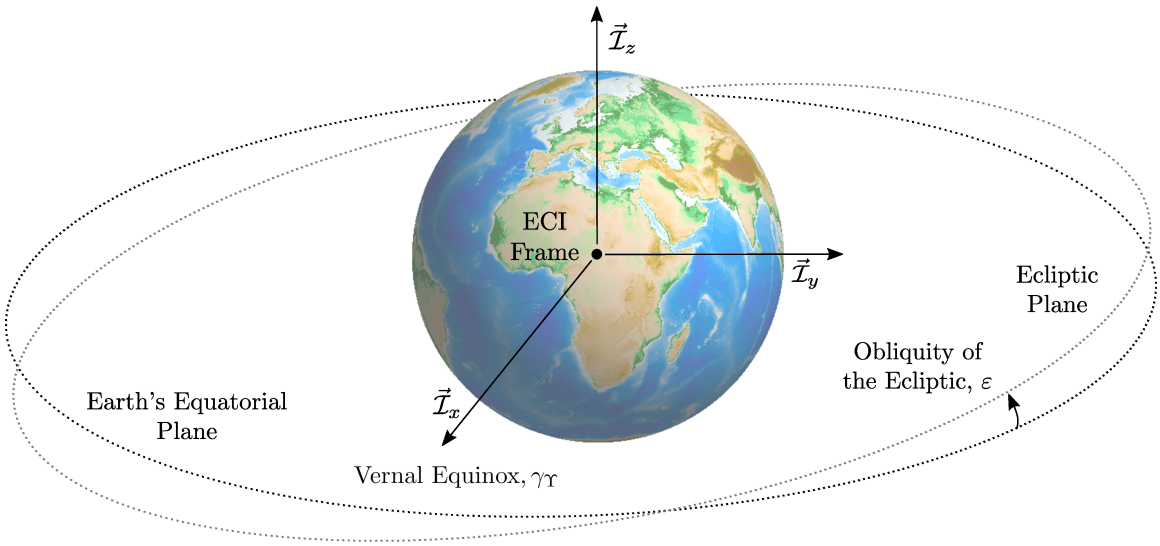


Figure 2.2: The Earth-centered Inertial reference frame, showing the equatorial plane and the obliquity of the ecliptic ε. Relative to the plane of the orbit that the Earth follows around the Sun, the Equatorial plane is inclined by roughly $\varepsilon = 23.4°$.

2.2.2 The Earth-Centered Earth-Fixed (ECEF) Frame

Noting that the ECI frame uses an x-axis aligned constantly with the Vernal Equinox, it is useful to define an alternative system which rotates with the Earth. This *Earth-centered Earth-fixed* system similarly uses geographic north to align the z-axis, however the x-axis always points toward 0° latitude and 0° longitude. The Earth has an angular rotation rate $\omega_\oplus$, and so the ECEF frame is rotated from the ECI frame through an angle θ_{GMT} about the Earth's axis of rotation. The angle θ_{GMT} is called the Greenwich Mean Time (GMT), and an illustration of the ECEF frame is shown in Fig. 2.3.

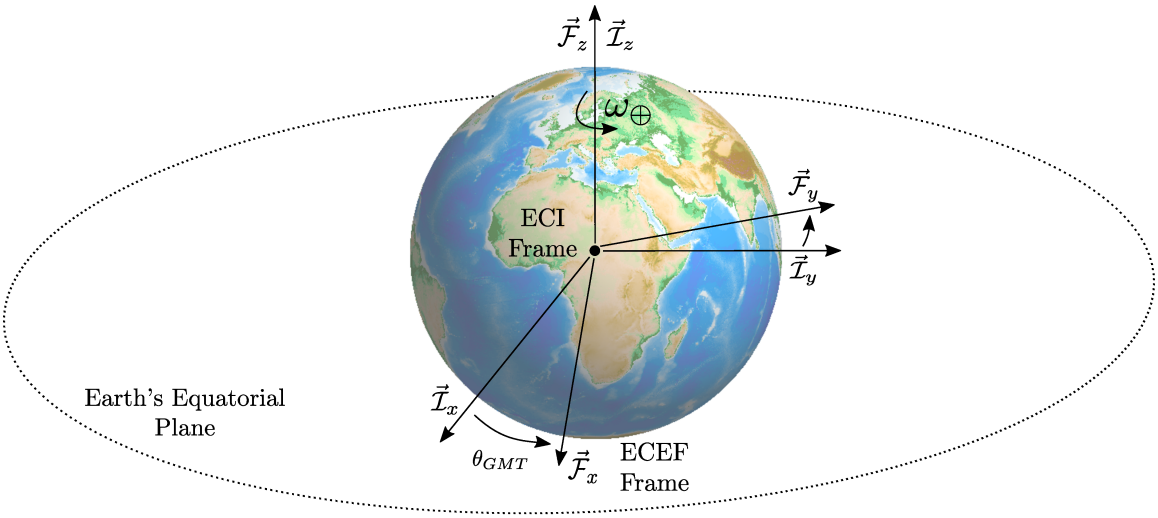


Figure 2.3: The Earth-Centered Earth-Fixed frame, aligned with the geographic north pole and 0° latitude, 0° longitude. The ECEF frame co-rotates along with the Earth, and is ideal for identifying the geodetic locations of a spacecraft in orbit with respect to locations on the Earth.

2.2.3 The Perifocal Frame

A *perifocal* coordinate system is defined by choosing the orbital plane as the fundamental plane for the system, and taking the origin as the center of the primary body being orbited. The $\vec{\mathcal{P}}_x$ axis is directed towards periapsis, and the orbit normal direction $\vec{\mathcal{P}}_z$ is parallel with the orbital momentum vector. As before, $\vec{\mathcal{P}}_y$ completes the dextral system. A figure showing the perifocal reference frame $\mathcal{F}_P$ will be presented later with a discussion of the orbital elements, but for now, it's useful to note that the perifocal system is non-inertial when dealing with non-Keplerian orbits (*i.e.*, when accounting for perturbations).

2.2.4 The Local-Vertical Local-Horizontal (LVLH) Frame

The previous frames are suitable for defining the absolute motion of spacecraft with respect to the Earth, but for the purposes of formation flying it is beneficial to define a new frame originating at the target spacecraft. To this effect, a *Local-Vertical Local-Horizontal* reference frame is useful for developing and visualizing the relative dynamics between two spacecraft, and can be seen in Fig. 2.4. The *radial* direction $\vec{\mathcal{L}}_x$ of the LVLH frame points outwards from the center of the Earth towards the target, and the *cross-track* direction $\vec{\mathcal{L}}_z$ is defined normal to the orbital plane, positive in the direction of the orbit angular momentum vector. Finally, the *along-track* direction $\vec{\mathcal{L}}_y$ follows the right-hand rule to complete the system. Given a target spacecraft with known position vector $\vec{r}_t$ and velocity vector $\vec{v}_t$ in the ECI frame, the orthonormal unit vectors for the LVLH frame can be calculated with

$$\vec{\mathcal{L}}_x = \frac{\vec{r}_t}{r_t} \qquad \vec{\mathcal{L}}_y = \vec{\mathcal{L}}_z \times \vec{\mathcal{L}}_x \qquad \vec{\mathcal{L}}_z = \frac{\vec{r}_t \times \vec{v}_t}{|\vec{r}_t \times \vec{v}_t|} \tag{2.1}$$

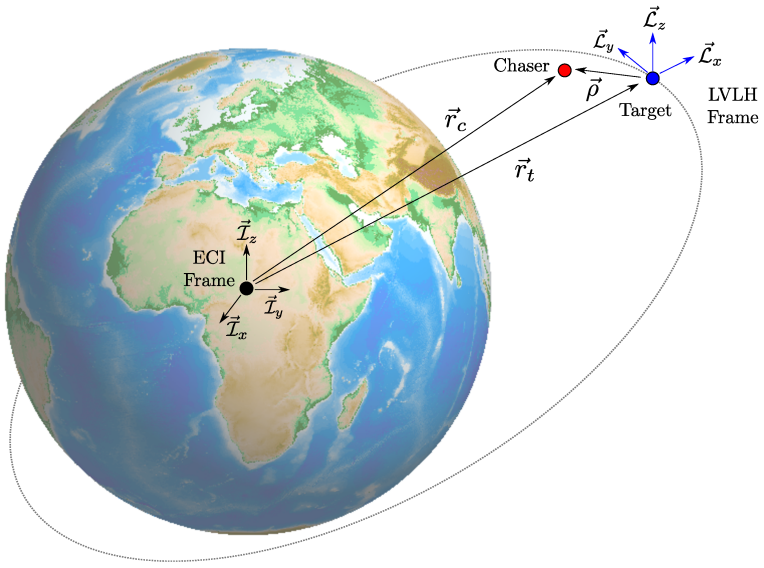


Figure 2.4: The ECI and LVLH reference frames. Absolute spacecraft dynamics are defined within the inertial frame, but the LVLH frame provides an intuitive reference for relative motion.

The LVLH frame appears in literature with several alternate names, including the rectilinear *RIC Frame* [64], the *RTN Frame* [65], and the *Hill's Frame* [66]. As a point of caution, the axes definitions may vary (ie. the x-axis here may correspond to the z-axis in another work), so always verify the definition of the system before considering the equations of motion derived for that system.

2.3 The Two-Body Problem

The reference frames introduced in the previous section are useful for visualizing and qualitatively understanding spacecraft orbits around Earth, so now a mathematical model is required in order to precisely predict how a spacecraft will evolve along an orbit through time. To this effect, the following section introduces the general dynamics of spacecraft motion, from the initial empirical laws established by Kepler to the well-known two-body equation of motion.

2.3.1 Kepler's Laws for Celestial Bodies

As with most physical phenomena, laws pertaining to planetary motion were originally derived through observation. Even before the invention of the telescope in the early 1600's and the groundbreaking work of Galileo, early scholars were able to collect a vast number of observations of celestial objects using the naked eye and early astronomical instruments. In particular, Danish astronomer Tycho Brahe (1546-1601) compiled decades worth of observations of Mars, which were subsequently used by his student Johannes Kepler (1571-1630) to establish the fundamental laws of planetary motion known today. The original publication of Kepler's work came in several installments, including the *Astronomia Nova* in 1609, and the *Epitome Astronomiae Copernicanae*, which was released in three volumes between 1617 and 1621 [67]. Kepler's famous laws describe the motion of planetary bodies, and are paraphrased in common terminology below as adapted from [68].

Kepler's First Law - The Elliptical Orbit Law

A planet orbiting the Sun follows an elliptical trajectory, with the Sun located at one focii of the ellipse.

Kepler's Second Law - The Equal Area Law

A line drawn from the Sun to a planet sweeps through equal areas in the ellipse during equal periods of time.

Kepler's Third Law - The Law of Periods

The square of a planet's orbital period is proportional to the cube of the average distance between the planet and the Sun.

2.3.2 Newton's Law of Universal Gravitation

Kepler's discoveries do indeed describe the geometry of planetary motion, but they do little to explain *why* the planets move the way the do. Nevertheless, the research of Kepler established a framework upon which Sir Isaac Newton (1642-1727) developed a mathematical theory capable of explaining the motion of planets around their stars, which similarly explains the motion of spacecraft around their planets. In 1684, Newton put forth his Universal Law of Gravitation [69], which states that the gravitational force acting on one body due to another body is a function of the mass of the objects and the square of the distance between them. Considering two generic objects of mass m_1 and m_2 respectively, the mathematical representation of Newton's Law of Gravitation shows that m_1 exerts a force on m_2 equal to

$$\vec{F}_{21} = -\frac{Gm_1m_2}{|\vec{r}_{21}|^3}\vec{r}_{21} \tag{2.2}$$

where the position of m_2 with respect to m_1 in an inertial reference frame is contained in the vector $\vec{r}_{21}$, and the Universal Gravitational Constant is $G = 6.6732\times10^{-11}$ Nm^2/kg^2. By making several reasonable assumptions about a spacecraft orbiting a large planetary body, Newton's Law of Gravitation can be used to obtain equations of motion for the spacecraft. The assumptions made here are as follows:

Assumption 1 - The Massive Primary

The mass of the spacecraft m_{sc} orbiting the primary body is negligible compared to the mass of the primary body (ie. the mass of Earth $m_\oplus \gg m_{sc}$).

Assumption 2 - Newtownian Dynamics

The mutually attractive forces of Newtonian gravity are the only forces acting on the two bodies.

Assumption 3 - Spherical Bodies

Both bodies are treated as spherically symmetric objects, such that the can be considered as particle point masses.

The mass of the Earth is roughly $m_\oplus = 5.97 \times 10^{24}$ kg whereas the largest human-constructed object in orbit, the International Space Station, has a mass on the order of 4.20×10^5 kg [70]. It's therefore common to define the two-body gravitational constant of the Earth system $\mu_\oplus$ simply in terms of the mass of the Earth, so

$$G\left(m_\oplus + m_{sc}\right) \approx Gm_\oplus = \mu_\oplus \tag{2.3}$$

For this work, $\mu_\oplus = 398\ 600.435\ 436\ \text{km}^3/\text{s}^2$ as published by NASA in the DE430 planetary parameters [71]. With the preceding assumptions, the acceleration of a spacecraft with respect to the Earth in the geocentric reference frame, denoted as $\ddot{\vec{r}}$, is described by the famous Keplerian two-body orbital equation of motion [66]:

$$\ddot{\vec{r}} = -\mu_\oplus \frac{\vec{r}}{r^3} \tag{2.4}$$

In the absence of external forces, Eq. (2.4) is a second-order vector differential equation that fully defines the spacecraft position and velocity at any instant in time, given an appropriate set of six initial conditions. Position-velocity components are useful for referencing the absolute location of a spacecraft, but an alternate set of descriptors can be defined to give a better representation of the orbit itself; these parameters are commonly referred to as the *Classical Orbital Elements (COEs)*.

2.4 Classical Orbital Elements

Recalling that an orbit about the Earth can be described by an ellipse, the orbital elements describe the size, shape, and orientation of the ellipse, while also indicating the position of a spacecraft along that orbit. Figure 2.5 shows the six orbital elements: the semi-major axis a, the eccentricity e, the inclination i, the Right Ascension of the Ascending Node (RAAN) Ω, the argument of perigee ω, and the true anomaly θ.

Semi-major Axis, a

The semi-major axis represents the size of the orbit, and is the distance from the center of the ellipse to the edge of the long axis, hence *semi-major axis*. In the case of a circular orbit the semi-major axis is simply the orbit radius, but in the general elliptical case, it is the average of the perigee and apogee radii. When dealing with Earth-centered orbits, the semi-major axis is typically measured in kilometers.

Eccentricity, e

The shape and elongation of an ellipse is given by a unitless metric called eccentricity, which quantifies how circular or oval-like the orbit is. An eccentricity of 0 indicates a circular orbit, while eccentricities near 1 are highly elliptical orbits. If $e \geq 1$, then the orbit is no longer an ellipse, and the trajectory will be either parabolic ($e = 1$) or hyperbolic ($e > 1$). The unbounded parabolic and hyperbolic cases are useful for interplanetary transfer missions, but are not addressed further in this work.

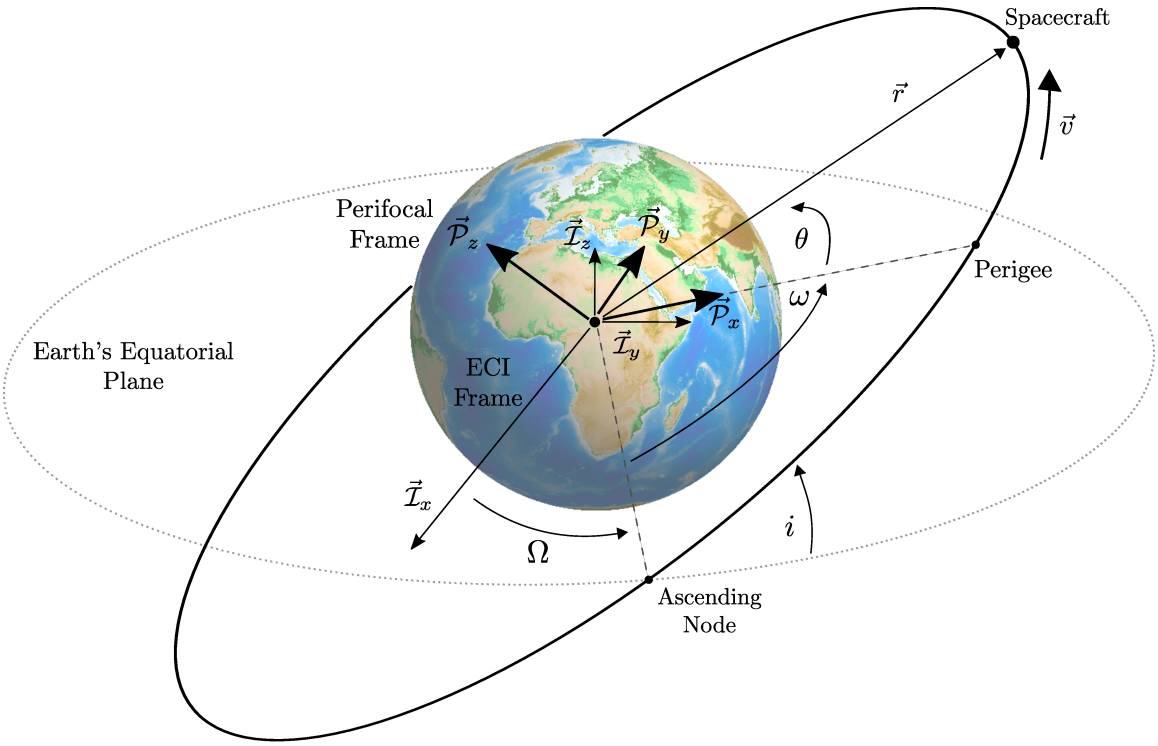


Figure 2.5: The perifocal (orbital) reference frame and the classical orbital elements.

Inclination, i

The inclination is a measure of the angle between the orbital plane and the Earth's equatorial plane. An inclination of 0° means the spacecraft travels over the Equator, while $i = 90°$ signifies a polar orbit. Stated another way, the inclination represents the angle between the positive ECI $\vec{\mathcal{I}}_z$ axis and the direction of the orbit angular momentum vector, where the angular momentum of an orbit is always perpendicular to the plane of the orbit.

Right Ascension of the Ascending Node, Ω

Defining the line of nodes as the line where the orbital plane intersects the equatorial plane, the RAAN is an angle used to define the ascending point on the line of nodes. The angle of Ω is measured from the ECI $\vec{\mathcal{I}}_x$ axis of the reference frame, which corresponds to the line directed from the center of Earth to the First Point of Aries. In other words, the ascending node corresponds to the point where a spacecraft would cross from the Southern Hemisphere into the Northern Hemisphere.

Argument of Perigee, $\boldsymbol{\omega}$

The Argument of Perigee is used to indicate how far the point of perigee is rotated with respect to the ascending node. Therefore, values of $\omega = 0°$ or $\omega = 180°$ would indicate the perigee of the orbit lies on the equatorial plane.

True Anomaly, $\boldsymbol{\theta}$

The final orbital element, the true anomaly, defines the angular position of the spacecraft within the orbital plane with respect to the perigee point. When $\theta = 0°$, the spacecraft is at perigee, and inversely when $\theta = 180°$, the spacecraft is at apogee. In Keplerian motion, all orbital elements are time-invariant except for the true anomaly, which will range from 0° to 360° as a spacecraft completes one revolution of the orbit.

In summary, the set of six classical orbital elements completely defines the position of a spacecraft in orbit; the inclination, RAAN, and argument of perigee capture the orientation of the orbit in three-dimensional space, the semi-major axis describes the size of the orbit, the eccentricity defines the shape of the orbit, and the true anomaly identifies the position of the spacecraft along the orbit. Thus, in addition to the position-velocity state vector resulting from the two-body equations, the dynamics of a spacecraft can be written in terms of the orbital element set

$$\mathbf{oe} = \{\ a,\ e,\ i,\ \Omega,\ \omega,\ \theta\ \}$$

The mathematical relationships needed to convert from position-velocity components to orbital elements (and vice versa) can be found in Appendix B.3. There are also useful software implementation examples of these algorithms given in the textbooks by Curtis [72] and Kluever [73]. For this work, the classical orbital elements are used to define the initial states of the target and chaser spacecraft, and the initial COE sets are converted to position-velocity states for propagation within the orbit simulator. It should be noted however that a substantial amount of research in orbital mechanics has been devoted to characterizing the evolution of the orbital elements as a result of external disturbances that are not accounted for by the two-body equation. While these time-varying orbital element descriptions are beyond the useful scope of this thesis, the intricacies of non-Keplerian motion are critical to consider when dealing with high-fidelity spacecraft simulation, and will be addressed in the following section using the position-velocity state framework.

2.5 Non-Keplerian Orbital Motion

To a first approximation, the two-body equation in (2.4) provides a general description of simplified spacecraft motion. The elegance of this solution however, required the use of a number of assumptions that are simply not true in reality. The Earth for example, while sphere-like, is certainly not a perfect sphere! Most spacecraft will similarly avoid a purely spherical structure due to the lack of surface area for the given volume, and the addition of solar arrays and communications antennas further adds to the complex geometry of real-life spacecraft. Thus the spherical point mass idealizations in Assumption 2 do not hold, and Assumption 3 is commensurately false because the mutually attractive force of gravity is *not* the only force acting on a spacecraft. Forces imposed by the atmosphere, the Sun, the Earth, the Moon, and a myriad of other sources all serve to violate the Keplerian motion model, and give rise to what is known as Perturbed, or non-Keplerian motion.

The theory for analyzing non-Keplerian motion relies of the notion of perturbations, which are small disturbances to the nominal Keplerian trajectory due to the effects of external forces. Two classes of perturbation analyses exist: *general perturbation theory* analytically investigates the effects of disturbances on the orbital elements for a given duration of time, while *special perturbation theory* uses direct numerical integration of the equations of motion to propagate the spacecraft state forward in time. Since the focus of this work is on relative navigation, special perturbation theory will be applied to incorporate perturbing forces into the position-velocity states of the spacecraft.

The propagator developed for this research accounts for perturbations due to the non-sphericity of the Earth, the gravitational influences of the Moon and Sun, the drag forces induced by the atmosphere, and the photonic pressure of Solar radiation. Each following subsection briefly presents the key formulae and parameters for these perturbation models, which follow the summaries by Montenbruck and Gill [74]. The Cowell Approach [75] is used to modify the spacecraft dynamics equations, whereby the sum of the perturbing accelerations acting on the spacecraft is integrated at each time step. Note that the equations of motion developed here can be applied to any spacecraft in the ECI frame, so the case of a general spacecraft is considered to avoid adding designating subscripts to the notation. From this, after expanding the two-body equation to account for perturbing accelerations $\vec{a}_p$, the net acceleration of the spacecraft is

$$\ddot{\vec{r}} = -\mu_\oplus \frac{\vec{r}}{r^3} + \sum \vec{a}_p \tag{2.5}$$

The term $\mu_\oplus$ is used here to explicitly refer to the gravitational parameter of the Earth, as additional planetary bodies will be included shortly.

Figure 2.6 is a reprint of a standard acceleration chart that shows nominal accelerations resulting from various sources as a function of distance from the Earth, and provides validation for the selection of the force models used in this research. The two-body acceleration (marked GM) not surprisingly dominates the system regardless of spacecraft altitude, with the effects of the non-spherical Earth (marked $J_{2,0}$) contributing the largest perturbation at all but extremely low altitudes. Similarly the influence of drag, solar radiation pressure, the Moon, and the Sun, all contribute noticeably to the acceleration of a spacecraft over a wide range of altitudes; since the formations considered in this research span from LEO to HEO conditions, it is appropriate to include these primary perturbations into the dynamics model.

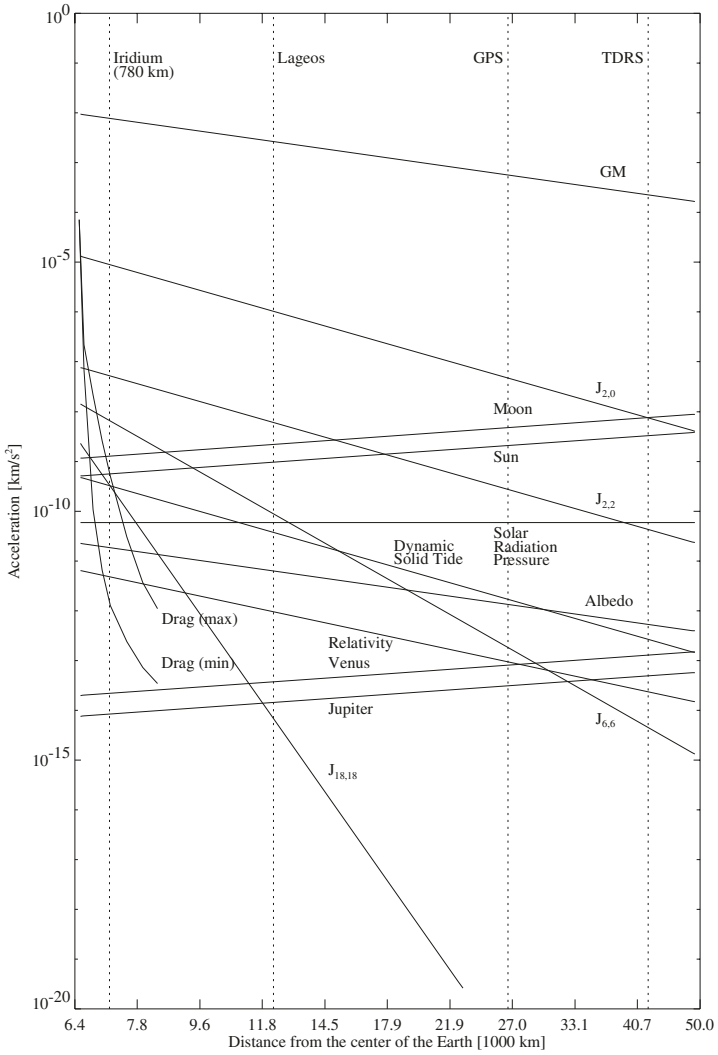


Figure 2.6: The classic comparison chart by Montenbruck and Gill, demonstrating the magnitudes of various orbit accelerations acting on a spacecraft as a function of distance from the Earth [74].

2.5.1 Gravity Perturbations due to Earth's Oblateness

The Keplerian description of orbital motion assumes the Earth is a perfectly spherical point mass, which is quite an oversimplification of the diverse topography of our planet. Since the Earth is indeed not a perfect sphere with uniform density, models using spherical coordinates have been developed to describe the resulting *gravitational potential* of the Earth. Although this potential varies as a function of longitude λ_{lon}, geocentric latitude ϕ_{lat} and distance from the center of mass r, the orbit propagator developed for this research considers only the dominating variations due to latitude.

First, the Earth will be treated as an *oblate spheroid*, where the equatorial radius of the planet is markedly larger than the polar radius. To illustrate the basis for this assumption, it's useful to note that the polar radius of the Earth is roughly 6 356 km, while the equatorial radius is markedly larger at 6 378 km. This *oblateness*, or uneven distribution of mass around the equator, is a result of the Earth being a massive, spinning, deformable body. Thus, for an oblate spheroid symmetric about its axis of rotation, the resulting gravitational potential must have rotational symmetry about the spin axis as well. The rotationally symmetric perturbation potential $\Phi(r, \phi_{lat})$ of the Earth is written using an infinite series [76]:

$$\Phi(r, \phi_{lat}) = \frac{\mu_\oplus}{r} \sum_{k=2}^{\infty} J_k \left(\frac{R_\oplus}{r} \right)^k P_k \left(\cos \phi_{lat} \right) \tag{2.6}$$

where J_k contains the *zonal harmonics* of the planet, and $P_k(\cdot)$ are the *Legendre polynomials*. The J_2 term in the spherical harmonic expansion is over a thousand-times larger than the next largest term, so considering only J_2 and evaluating the gradient of the gravitational potential, the perturbing acceleration due to the oblateness of the Earth can be summarized in vector notation by [66]:

$$\vec{a}_{J_2} = \frac{3\mu_\oplus J_2 R_\oplus^2}{2r^5} \left\{ \left[\frac{5 \left(\vec{r} \cdot \vec{\mathcal{I}}_z \right)^2}{r^2} - 1 \right] \vec{r} - 2 \left(\vec{r} \cdot \vec{\mathcal{I}}_z \right) \vec{\mathcal{I}}_z \right\} \tag{2.7}$$

where the corresponding terms are:

$$\begin{aligned}
\vec{a}_{J_2} \sim&\ \text{Acceleration on the spacecraft due to } J_2 \text{ [m/s}^2\text{]} \\
\vec{r} \sim&\ \text{Spacecraft position vector in the ECI Frame [m]} \\
\mu_\oplus \sim&\ \text{Gravitational parameter of Earth [m}^3\text{/s}^2\text{]} \\
J_2 \sim&\ \text{Second zonal geopotential coefficient [unitless]} \\
\vec{\mathcal{I}}_z \sim&\ \text{Unit vector defining the ECI } z\text{-axis}
\end{aligned}$$

The value of $J_2 = 1082.625 \times 10^{-6}$ is used for this work, as taken from the DE430 planetary parameters defined by NASA [71]. For implementation in MATLAB, the accelerations given by (2.7) are expanded into component form to give the scalar relations below, which align with the form presented in [77] after some rearrangement:

$$a_{J_{2x}} = \frac{\mu_\oplus X}{r^3}\left(\frac{3}{2}J_2\right)\left(\frac{R_\oplus}{r}\right)^2\left(5\frac{Z^2}{r^2}-1\right) \tag{2.8}$$

$$a_{J_{2y}} = \frac{\mu_\oplus Y}{r^3}\left(\frac{3}{2}J_2\right)\left(\frac{R_\oplus}{r}\right)^2\left(5\frac{Z^2}{r^2}-1\right) \tag{2.9}$$

$$a_{J_{2z}} = \frac{\mu_\oplus Z}{r^3}\left(\frac{3}{2}J_2\right)\left(\frac{R_\oplus}{r}\right)^2\left(5\frac{Z^2}{r^2}-3\right) \tag{2.10}$$

where the ECI position of the spacecraft is defined by the coordinates $\{X, Y, Z\}$, and r is the magnitude of the position vector.

2.5.2 Luni-Solar Third-body Accelerations

Throughout this chapter, Keplerian orbital motion has frequently been described as a solution to the two-body problem, since it was assumed that only one large mass was influencing the motion of another smaller mass. In reality, other bodies within the Solar System also impart a gravitational pull on the Earth and nearby spacecraft, and this additional force is referred to as *third-body gravity*. The effects of third-body gravity are therefore added into the real-world spacecraft dynamics model as a perturbation, which is derived by recalling Newton's Law of Gravitation. Firstly, acceleration of a spacecraft due to an arbitrary point mass m_i can be modelled by

$$\ddot{\vec{r}}_i = \mu_i \frac{\vec{r}_{m_i} - \vec{r}}{|\vec{r}_{m_i} - \vec{r}|^3} \tag{2.11}$$

where $\vec{r}$ is the geocentric position of the spacecraft, $\vec{r}_{m_i}$ is the geocentric position of the third body, and μ_i is the gravitational parameter of the third body. Bear in mind that these geocentric positions define motion *with respect to the Earth*, and the Earth itself experiences an acceleration caused by mass m_i, described by

$$\ddot{\vec{r}}_\oplus = \mu_i \frac{\vec{r}_{m_i}}{r_{m_i}^3} \tag{2.12}$$

As a result, the perturbing acceleration on the spacecraft in the ECI Frame must then be written as the acceleration due to mass m_i minus the acceleration of the

moving Earth frame. This gives the perturbing third-body acceleration as

$$\vec{a}_{3B_i} = \mu_i \left(\frac{\vec{r}_{m_i} - \vec{r}}{|\vec{r}_{m_i} - \vec{r}|^3} - \frac{\vec{r}_{m_i}}{r_{m_i}^3} \right) \tag{2.13}$$

with the terms:

$$\begin{aligned} \vec{a}_{3B_i} \sim & \text{ Acceleration on the spacecraft due to the third body } [\mathrm{m/s^2}] \\ \mu_i \sim & \text{ Gravitational parameter of the third body } [\mathrm{m^3/s^2}] \\ \vec{r}_{m_i} \sim & \text{ Third body position vector in the ECI Frame } [\mathrm{m}] \\ \vec{r} \sim & \text{ Spacecraft position vector in the ECI Frame } [\mathrm{m}] \end{aligned}$$

From an Earth-centered perspective, the resulting acceleration caused by a third body acts away from the Earth if the body is collinear with the spacecraft and Earth. When the spacecraft is perpendicular to the line of collinearity, the resulting acceleration in the Earth-centered frame would appear to be acting towards the Earth. Figure 2.7 shows the geometry used in deriving the effects of third-body gravity.

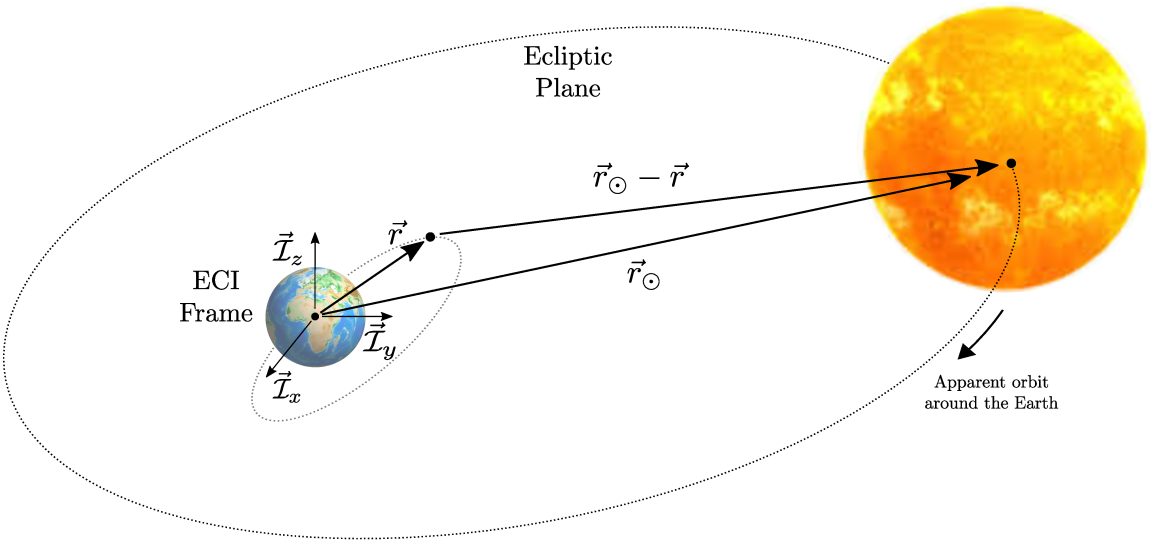


Figure 2.7: The geometry associated with a spacecraft under the influence of third-body gravity, using the Sun as an example. Identical geometrical relationships can be identified for the Moon and other planets within the Solar system.

Equation (2.13) can be used to account for the third-body perturbations due to any number of celestial objects. The two largest contributors to third-body gravity are the Sun and the Moon, and their combined effects are commonly referred to as *luni-solar perturbations*. For Earth-based orbits, the perturbations due to the Moon are roughly twice as large as the perturbations due to the Sun [78], as highlighted in Table 2.1. The corresponding gravitational parameters for the Sun and Moon, $\mu_\odot$ and $\mu_\leftmoon$ respectively, are also given in the table below.

Table 2.1: Gravitational Parameters for the Sun and Moon

Planetary Body	Gravitational Parameter [m^3/s^2]	Perturbation Magnitude [m/s^2]
Sun $\odot$	$132\ 712\ 440\ 041.939\ 400 \times 10^9$	0.5×10^{-6}
Moon $\leftmoon$	$4\ 902.800\ 066 \times 10^9$	1.1×10^{-6}

Variation in the positions of the Sun and Moon are quite small during the short simulation duration used to demonstrate the proposed navigation solutions, so it is assumed that the positions of the Sun and Moon are fixed within the orbit propagator. While high precision orbit propagation over long timescales requires the modelling and updating of the Sun and Moon ephemerides[1] throughout the simulation, the additional processing was not necessary in the context of this research. As such, the orbit propagator for this work allows the geocentric Solar position vector $\vec{r}_\odot$ and Lunar position vector $\vec{r}_\leftmoon$ to be calculated during the initialization of the simulation based on the Julian Date[2], and these positions remain constant during the simulation. The simulation local time is set at midnight November 29, 2018, and the corresponding position of the Sun and Moon are

$$\vec{r}_\odot = \vec{\mathcal{F}}_L^T \begin{bmatrix} -\ 58\ 363\ 949.945 \\ -124\ 361\ 166.068 \\ -\ 53\ 910\ 073.877 \end{bmatrix} \text{ km} \qquad \vec{r}_\leftmoon = \vec{\mathcal{F}}_L^T \begin{bmatrix} -295\ 523.833 \\ 196\ 497.136 \\ 100\ 052.899 \end{bmatrix} \text{ km}$$

These positions are calculated using the celestial ephemerides found in *the Astronomical Almanac*, and follow the low-precision methods suggested by Montenbruck [74]. For additional information on the algorithms used to determine the Sun and Moon position vectors, please consult Appendix B.4.

[1]The planetary ephemerides are a set of tabulated data that can be used to calculate the precise position of a celestial body based on the current date.

[2]The Julian Date is the number of days since noon on January 1, 4713 BC. The Julian time scale is frequently used in astronomy, as it avoids dealing with leap years and months with different numbers of days, thereby providing a uniform and continuous measure of time.

2.5.3 Atmospheric Drag

The perturbations considered until this point have all been gravitational in nature, and the resulting forces acting on a spacecraft due to the Earth's oblateness and the third-body effects have therefore been *conservative* in nature. A conservative force acting on a spacecraft may change the shape or orientation of an orbit, but cannot change the overall energy of that orbit [79]. For spacecraft in LEO however, atmospheric drag forces exist that gradually decrease the energy of an orbit, decrease the semi-major axis a, and can eventually lead to the de-orbiting of a spacecraft if not correctly accounted for in the mission design. As such, effects due to atmospheric drag must be accounted for in orbits that pass below altitudes of 1000 kilometers [80].

For a body moving with some relative velocity $\vec{v}_r$ with respect to the atmosphere, drag forces oppose the motion and act along the opposite direction of the relative velocity vector. The magnitude of the drag force is directly proportional to the frontal area A_d of the body exposed to the fluid flow, which is commonly scaled by a drag coefficient C_d to account for various unmodelled effects such as the molecular composition and temperature of the fluid. The mass of the spacecraft m also affects the contributions of drag, so it is clear that for LEO spacecraft careful consideration must be given to the physical geometry and design of the structure.

Since drag is due to the interaction between the surface of the spacecraft and the atmosphere, the atmospheric density ρ_{atm} plays a significant role in determining the drag forces experienced by the body as well. The resulting acceleration on the spacecraft due to the aforementioned factors is written using the standard drag relation [74]:

$$\vec{a}_{drag} = -\frac{1}{2}\rho_{atm} v_r^2 \frac{C_d A_d}{m} \frac{\vec{v}_r}{|\vec{v}_r|} \tag{2.14}$$

using the following definitions:

$\vec{a}_{drag} \sim$ Acceleration on the spacecraft due to drag [m/s^2]

$\rho_{atm} \sim$ Atmospheric density [kg/m^2]

$C_d \sim$ Aerodynamic drag coefficient [unitless]

$A_d \sim$ Surface area exposed to drag [m^2]

$m \sim$ Mass of the spacecraft [kg]

$\vec{v}_r \sim$ Relative velocity vector of spacecraft with respect to the atmosphere [m/s]

$v_r \sim$ Relative speed of the spacecraft with respect to the atmosphere [m/s]

Modelling Atmospheric Density

The Harris-Priester model, originally presented in 1962 [81], is used to determine the atmospheric density needed to calculate the drag forces acting on the spacecraft. Variation in atmospheric density is inherently complex to model, since it varies as a function of the chemical constituents within the atmosphere, the temperature of the atmosphere, the intensity of solar flux, the time of day, the time of year, and of course, the altitude being considered. Despite the complex nature of this environment, the Harris-Priester model provides a relatively computationally friendly approximation of the atmospheric density for altitudes ranging from 100 km to 1000 km, and has been shown to be within 40% of more complex variants like the Jacchia-Roberts model [82].

A full description of the Harris-Priester model and associated calculations are included in Appendix B.5, but the final atmospheric density is predominantly a function of spacecraft altitude and the location of the diurnal atmospheric density bulge that results from solar heating throughout the day. Thus, calculation of the drag forces rely on the atmospheric density obtained from

$$\rho_{atm}(h) = \rho_m(h) + \left(\rho_M(h) - \rho_m(h)\right) \cos^{n_o}\left(\frac{\Psi}{2}\right) \tag{2.15}$$

where is should be noted that:

$\rho_{atm} \sim$ Atmospheric density [kg/m^2]
$h \sim$ Altitude above the reference Earth ellipsoid [km]
$\rho_m \sim$ Tabulated minimum atmospheric density [kg/m^2]
$\rho_M \sim$ Tabulated maximum atmospheric density [kg/m^2]
$\Psi \sim$ Angle between the spacecraft and the diurnal bulge apex [rad]
$n_o \sim$ Empirical orbit exponent [unitless]

Values of the maximum and minimum densities at a specific altitude are calculated using exponential interpolation between data provided in Appendix Table B.2 for 10 km increments, and the exponent $n_o \in \{2, \ldots, 6\}$ varies depending on the inclination of the orbit. For example, a value of $n_o = 2$ is used for low inclination equatorial orbits, and $n_o = 6$ for high inclination polar orbits.

Modelling Bulk Atmospheric Motion

A first-order approximation of the relative velocity of the spacecraft with respect to the atmosphere can be made by assuming that the atmosphere co-rotates with the Earth at an angular velocity $\omega_\oplus$ [74]. This assumption avoids additional complexity that would be needed to include a high-fidelity wind model of the atmosphere, and means the relative velocity of the spacecraft can simply be calculated as

$$\vec{v}_r = \vec{v} - \vec{\omega}_\oplus \times \vec{r} \tag{2.16}$$

where $\vec{v}$ is the velocity vector of the spacecraft, and Earth's rotation is assumed purely about the z-axis, so

$$\vec{\omega}_\oplus = \begin{bmatrix} 0 & 0 & \omega_\oplus \end{bmatrix}^T \tag{2.17}$$

For the short period of timed covered within the simulations, it is assumed that the angular velocity of the Earth is constant with a magnitude of $\omega_\oplus = 7.2921151467 \times 10^{-5}$ rad/s, following the WGS-84 reference model.

Modelling the Spacecraft

In practice, modelling the exact physical properties of the spacecraft is a difficult task; for one, the mass m of a spacecraft varies throughout the mission duration due to the usage of propellant for station-keeping and orbit correction manoeuvres, and similarly the drag area A_d changes based on the attitude of the spacecraft and the deployment of solar panels and instrumentation. Additionally, the drag coefficient C_d is notoriously hard to estimate a priori, because it is a complex function of the spacecraft surface and the particles in the atmosphere. This can be seen by realizing that the drag coefficient for typical spacecraft can range between $1.5 < C_d < 3.0$, which can have considerable effects on the resulting drag forces [74].

For this work, the drag computations are simplified by assuming fixed and constant values for the mass, frontal area, and drag coefficient of the spacecraft. This is equivalent to averaging the overall drag effects throughout the orbital period of the spacecraft [83], and the worst case drag scenarios are considered by using the largest upper-limit values of the frontal area and drag coefficients. Specific values for the spacecraft parameters are presented later in this chapter, preceding the demonstration of the relative motion simulator.

2.5.4 Solar Radiation Pressure

The final perturbing force included in the real-world orbit model is due to the Sun. The continuous stream of photons projected by the Sun will impact a spacecraft any time the spacecraft is not eclipsed by the Earth, and as light strikes the surface of the spacecraft, the resulting change in momentum over time can noticeably perturb the orbit of the spacecraft. The standard approach for modelling this perturbation is to treat the change in momentum as a pressure that acts upon the exposed area of the spacecraft [74], giving an acceleration that is described by

$$\vec{a}_{SRP} = -\nu_{IF} P_{\odot} \frac{C_r A_s}{m} (AU)^2 \frac{\vec{r}_{\odot} - \vec{r}}{|\vec{r}_{\odot} - \vec{r}|^3} \tag{2.18}$$

Here the following definitions are used:

$\vec{a}_{SRP} \sim$ Acceleration on the spacecraft due to solar radiation pressure [m/s^2]
$\nu_{IF} \sim$ Illumination factor [unitless]
$P_{\odot} \sim$ Nominal solar radiation pressure at 1AU [N/m^2]
$C_r \sim$ Radiation pressure coefficient [unitless]
$A_s \sim$ Surface area exposed to solar radiation [m^2]
$m \sim$ Mass of the spacecraft [kg]
$\vec{r}_{\odot} \sim$ Position vector of the Sun with respect to Earth [m]
$\vec{r} \sim$ Position vector of the spacecraft with respect to Earth [m]

This model is commonly called the Cannonball Model [21], since it assumes that the spacecraft structure is a lumped spherical object that absorbs all photons incident on its surface, independent of orientation. This means the effects of diffuse reflection are treated as negligible, and intuitively the resulting Solar Radiation Pressure (SRP) is a force acting in the opposite direction of the vector pointing from the spacecraft to the Sun. The extent of SRP force is similarly dependent on the mass m and exposed surface area A_s of the spacecraft, much like the force of drag. The coefficient of reflection C_r is dependent on the types of materials used on the surface of the spacecraft, and is typically within the range of $1.2 < C_r < 1.9$ for operational spacecraft [74].

The nominal solar radiation pressure at a distance of 1 AU is treated as constant at $P_{\odot} = 4.563 \times 10^{-6}$ N/m^2, corresponding to a solar flux of 1367 W/m^2 [84]. Above an altitude of roughly 800 km, solar radiation pressure begins to dominate the effects from atmospheric drag [62].

Eclipse Conditions

The illumination factor ν_{IF} is included in Eq. (2.18) to model instances when the spacecraft passes behind the Earth and is shielded from the Sun and corresponding solar radiation pressure. One method of modelling these eclipse conditions is to treat the shadow created by the Earth as an infinite cylinder extending from the Earth, with a radius equal to the average radius of Earth. Then, by identifying if the spacecraft lies within this cylindrical shadow, the appropriate illumination factor is selected.

For this work, either $\nu_{IF} = 0$ in eclipse conditions (*i.e.*, no solar radiation pressure), or $\nu_{IF} = 1$ outside of eclipse (*i.e.*, the spacecraft is exposed to radiation). To determine if the spacecraft lies within the cylindrical shadow of the sun, consider the unit vector $\hat{e}_{\odot}$ pointing from the Earth to the Sun:

$$\hat{e}_{\odot} = \frac{\vec{r}_{\odot}}{r_{\odot}} \tag{2.19}$$

and quickly recall that for two arbitrary vectors $\vec{a}$ and $\vec{b}$, the component of $\vec{b}$ in the direction of $\vec{a}$ is given by

$$\text{comp}_{\vec{a}}\,\vec{b} = \frac{\vec{a}\cdot\vec{b}}{|\vec{a}|} \tag{2.20}$$

Then, the scalar projection of the spacecraft position vector onto the Sun vector can be found as

$$s_{proj} = \text{comp}_{\hat{e}_{\odot}}\,\vec{r}_{sat} = \vec{r}_{sat}\cdot\hat{e}_{\odot} \tag{2.21}$$

Intuitively, if the value of s_{proj} is positive, the spacecraft is on the sun-lit side of Earth and is fully exposed to solar radiation pressure. Another possible scenario for full illumination arises when the spacecraft is on the dark side of the Earth, but lies outside of the cylindrical shadow of the Earth. To check this condition, the projection operator for vector $\vec{b}$ onto vector $\vec{a}$ is used, as follows:

$$\text{proj}_{\vec{a}}\,\vec{b} = \frac{\vec{a}\cdot\vec{b}}{|\vec{a}|}\frac{\vec{a}}{|\vec{a}|} = \frac{\vec{a}\cdot\vec{b}}{|\vec{a}|^2}\vec{a} \tag{2.22}$$

such that the vector projection of the spacecraft position onto the Sun vector can be calculated by

$$\vec{s}_{proj} = \text{proj}_{\hat{e}_{\odot}}\,\vec{r}_{sat} = s_{proj}\hat{e}_{\odot} \tag{2.23}$$

By assuming that the shadow of the Earth has a cylindrical radius $R_{\oplus}$, the magnitude of the vector difference between the spacecraft position and it's projection onto the Sun vector can be used to determine if the spacecraft is within the shadow.

The illumination factor ν_{IF} for this work takes either a value of 1 for full illumination, or 0 for full eclipse in the umbra. Therefore, the two cases are:

$$\nu_{IF} = \begin{cases} 1 & \text{if } s_{proj} > 0 \text{ or } |\vec{r}_{sat} - \vec{s}_{proj}| > R_{\oplus} \\ 0 & \text{otherwise} \end{cases} \tag{2.24}$$

Other methods to account for partial illumination when the spacecraft is in the penumbra are typically applied in high-precision applications [74], however assuming either pure illumination or pure shadow represents a worst-case scenario and is suitable for testing the navigation algorithms developed in this thesis.

2.5.5 Secondary Perturbations

The perturbation sources discussed above, namely the Earth's oblateness, atmospheric drag, solar radiation pressure, and third-body gravity, constitute the standard primary perturbations that affect near-Earth spacecraft. While there are indeed additional perturbations that influence the orbit of spacecraft, the magnitude of these disturbances is minimal for the simulation scenarios considered in this thesis. As such, these secondary perturbations are mentioned here for completeness, but are assumed negligible and are not accounted for in the truth model orbit propagator. The following descriptions have been summarized from the work of Capderou [78], and the mathematical models used to describe many of these perturbations can be found in Chapter 3 of Montenbruck [74].

Planetary Third-body Gravity

Luni-solar perturbations account for the largest deviations due to third-body gravity, but other celestial objects within the Solar System similarly influence the motion of spacecraft about Earth. Venus and Jupiter provide the next largest gravitational perturbations, on the magnitude of 10^{-10} m/s^2 and 10^{-11} m/s^2 depending on the configuration of the planets.

Tidal Forces

The deformable properties of the Earth's surface leads to variations in the gravitational potential of the planet. The crust and mantle of the Earth move over thousands of years, and these *solid Earth tides* redistribute the mass of the planet and must be accounted for in the geopotential model of gravity. The effects of the solid tides are on the order of 10^{-7} m/s^2, and the periodic variations due to the *ocean tides* are an order of magnitude smaller.

Relativistic Effects

Although the speed of near-Earth spacecraft is much less than the speed of light, relativistic acceleration does have an effect of the orbit of a spacecraft over long periods of time. In deriving the motion of a spacecraft in the framework of general relativity, the mass and angular momentum of the rotating Earth cause a curvature in four-dimensional space-time that introduces a small correction factor into the spacecraft acceleration derived in the Newtonian framework. These effects are typically on the order of 10^{-8} m/s^2

Earth Radiation Pressure

Much akin to the effects of Solar radiation pressure, radiation reflected and emitted from the Earth also exerts a force on low-Earth orbit spacecraft. Radiation from the Earth is general considered in two wavelengths: the optical *albedo* radiation that is a result of reflected light from the Sun, and the *infrared* radiation that is emitted from Earth. The combined effect of these Earth radiation pressures are roughly an order of magnitude less than Solar radiation pressure, and decrease rapidly with the square of the spacecraft altitude.

Yarkovsky Forces

The Solar and Earth-borne radiation perturbations mentioned previously only consider the linear momentum transfer that results from photons incident on the surface of the spacecraft, but do not account for the heating that takes place as a result of that radiation. This heating leads to the so-called Yarkovsky thermal effect, which is based on the absorption and anisotropic re-emission of radiation; when the face of an object absorbs radiation, that face increases in temperature and becomes hotter than the reverse face. Since both faces are then at different temperatures, re-emission of the radiation is unevenly distributed and the body is subjected to both a translational force and a rotational torque. The small magnitudes of these disturbances preclude them from use in most near-Earth orbit models, however they are routinely accounted for in deep-space scenarios where the primary Earth-based perturbations are less prevalent [85].

With the orbital perturbations and dynamics models presented in this section, it is now possible to simulate the behaviour of a spacecraft in orbit. The next section explores the details of the numerical simulator used to produce spacecraft trajectories for this thesis, and these trajectories will later be processed by the proposed Kalman filters relative navigation routines.

2.6 Spacecraft Formation Flying

Now that the force models for orbital motion have been established, the following section describes the numerical simulator developed to produce real-world orbit trajectories. The simulator outputs position and velocity states for the target and chaser spacecraft in inertial space, and the relative motion between spacecraft is then evaluated by differencing the absolute states and performing coordinate transformations to map the relative states into the LVLH frame fixed to the target. While this numerical method for evaluating the formation state is suitable for simulating the real-world system, the computational costs of integration are unattractive for onboard navigation algorithms. As such, analytic equations of relative motion are also derived in this section for later use in the proposed Kalman filtering algorithms of the next chapter.

2.6.1 Numerical Orbit Propagation

The force models and dynamical equations developed in the previous sections describe the total acceleration experienced by a spacecraft in orbit, hence the orbital motion of the spacecraft in the ECI frame is simulated numerically through a double integration of Eq. (2.5), with appropriate initial conditions. That is, the velocity vector in the ECI frame is propagated forwards in time from an initial t_0 to the current time t using

$$\vec{v}(t) = \vec{v}_0 + \int_{t_0}^{t} \vec{a}(t)\,\mathrm{d}t \tag{2.25}$$

and similarly the position is obtained using

$$\vec{r}(t) = \vec{r}_0 + \int_{t_0}^{t} \vec{v}(t)\,\mathrm{d}t \tag{2.26}$$

The initial conditions $\vec{r}_0$ and $\vec{v}_0$ are obtained from the initial orbital elements of the spacecraft following the standard algorithm in Appendix B.3. Both the target and chaser spacecraft are simulated in this fashion, and the respective positions $\vec{r}_t$ and $\vec{r}_c$, and velocities $\vec{v}_t$ and $\vec{v}_c$, represent that absolute motion of each spacecraft with respect to the Earth. The relative position and velocity of the formation is then calculated as the vector difference between the target and chaser states, which gives the relative position vector $\delta\vec{r}$ and the relative velocity vector $\delta\vec{v}$ in the ECI frame $\mathcal{F}_I$ as:

$$\delta\vec{r} = \vec{\mathcal{F}}_I^T \delta\mathbf{r}_{\mathcal{I}} = \vec{\mathcal{F}}_I^T \left(\mathbf{r}_c - \mathbf{r}_t\right) \tag{2.27}$$

$$\delta\vec{v} = \vec{\mathcal{F}}_I^T \delta\mathbf{v}_{\mathcal{I}} = \vec{\mathcal{F}}_I^T \left(\mathbf{v}_c - \mathbf{v}_t\right) \tag{2.28}$$

where $\mathbf{r}_c, \mathbf{r}_t, \delta\mathbf{r}_{\mathcal{I}} \in \mathbb{R}^3$ are column matrices containing the respective components of the chaser, target and relative position vectors in the ECI frame, $\mathbf{v}_c, \mathbf{v}_t, \delta\mathbf{v}_{\mathcal{I}} \in \mathbb{R}^3$ are similar matrices for the velocities, and $\vec{\mathcal{F}}_I \in \mathbb{R}^{3\times3}$ is the vectrix that contains the three orthogonal unit vectors of the ECI frame. The relative position and velocity components in the ECI frame are next converted to the LVLH frame through a transformation matrix built from the unit vectors of the LVLH frame, $\vec{\mathcal{L}}_x$, $\vec{\mathcal{L}}_z$, and $\vec{\mathcal{L}}_z$, shown previously in Eq. (2.1). The resulting rotation sequence is encapsulated by the matrix $\mathbf{C}_{\mathcal{LI}} \in \mathbb{R}^{3\times3}$, given by

$$\mathbf{C}_{\mathcal{LI}} = \vec{\mathcal{F}}_L \cdot \vec{\mathcal{F}}_I^T = \begin{bmatrix} \vec{\mathcal{L}}_x & \vec{\mathcal{L}}_y & \vec{\mathcal{L}}_z \end{bmatrix}^T \tag{2.29}$$

and the relative position vector is rotated into the LVLH frame with

$$\delta\mathbf{r}_{\mathcal{L}} = \mathbf{C}_{\mathcal{LI}}\delta\mathbf{r}_{\mathcal{I}} \tag{2.30}$$

This relative position vector in the LVLH frame $\delta\mathbf{r}_{\mathcal{L}} \in \mathbb{R}^3$, obtained by simulating the orbital motion of the individual spacecraft and differencing the states, is used as the *truth model* throughout the remaining chapters. That is, the real-world relative position of the spacecraft formation in the LVLH frame will be assumed to match the relative position $\delta\mathbf{r}_{\mathcal{L}}$, so that the estimation accuracy for the proposed EKF algorithms can be compared. Since the relative navigation algorithms are also responsible for estimating the relative velocity of the chaser with respect to the target, a similar truth model for the relative velocity is obtained by mapping the ECI velocity difference $\delta\mathbf{v}_{\mathcal{I}}$ into the LVLH frame. The non-inertial LVLH frame is rotating with respect to the ECI frame with an angular velocity vector $\boldsymbol{\omega}_{\mathcal{LI}} \in \mathbb{R}^3$, so making use of the *kinematic transport theorem* [66], the relative velocity in the LVLH frame is

$$\delta\mathbf{v}_{\mathcal{L}} = \mathbf{C}_{\mathcal{LI}}\left(\delta\mathbf{v}_{\mathcal{I}} - \boldsymbol{\omega}_{\mathcal{LI}}^{\times}\delta\mathbf{r}_{\mathcal{I}}\right) \tag{2.31}$$

where the superscript $\times$ indicates the skew-symmetric matrix operator, which implies a cross-product matrix multiplication (see Appendix B.1 for a review of matrix operations). Since the LVLH frame is fixed to the target spacecraft and lies in the plane of the orbit, the angular velocity vector of the LVLH frame with respect to the stationary ECI frame can be calculated using the cross product of the target position and velocity vectors as

$$\boldsymbol{\omega}_{\mathcal{LI}} = \frac{\mathbf{r}_t^{\times}\mathbf{v}_t}{\|\mathbf{r}_t\|^2} \tag{2.32}$$

Figure 2.8 provides a block diagram of the real-world orbit propagator created for this thesis. All simulations are completed within the MATLAB-Simulink software environment, and numerical integration for the orbit propagator is performed using Simulink integrator blocks configured with the `ODE4` fixed time-step solver. A step size of 1 second is defined within the simulations, and the absolute and relative error tolerances are set at 10^{-6}, based on empirical results of the orbit propagator. Acceleration components from the various force models are calculated in individual user-defined MATLAB functions, such that the perturbations can easily be activated or deactivated when initializing the simulation routine.

A full list of the astrodynamic constants used for this work are included in Appendix B.7, as based on the Planetary and Lunar Ephemerides DE430/DE431 [71] and the WGS-86 reference ellipsoid [86]. The source code used within the orbit propagator can be found in Appendix E, and diagrams of the orbit propagator integrated within the Kalman filter simulations are presented in subsequent chapters. Demonstrations of the orbit propagator are given at the end of this chapter, where three unique formations and their relative motion trajectories are simulated.

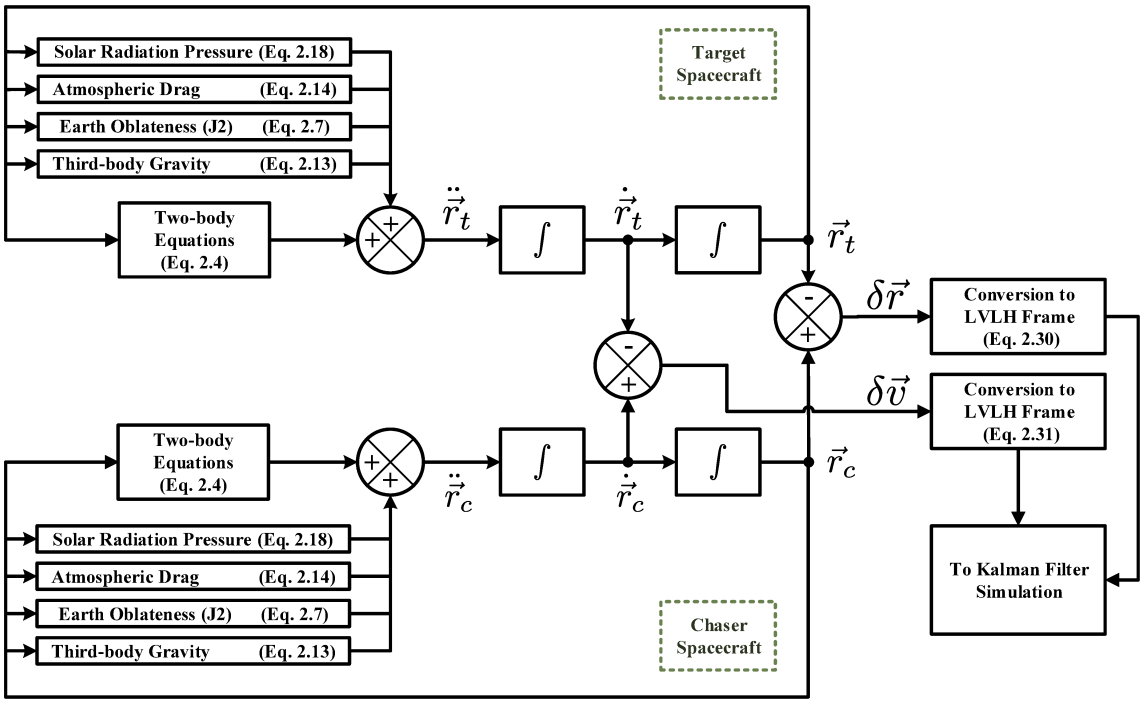


Figure 2.8: A block diagram showing the orbit propagation scheme within the numerical simulator, which creates the truth reference models for the relative position and velocity of the spacecraft formation.

2.6.2 Nonlinear Equations of Relative Motion

In the interest of using an analytic onboard dynamics model within the Kalman filter, the nonlinear equations of relative motion are presented here. The survey conducted by Sullivan *et al.* [23] presents an assessment of the current relative motion dynamics models available in literature, many of which contain simplifications and restrictions regarding the shape and size of the orbits being considered. However, the full nonlinear set of equations is implemented in this thesis in an effort to make the proposed navigation routines robust to a variety of spacecraft formation configurations. While no assumptions about the orbit will be made, it is necessary to make assumptions about the space environment in order to develop analytic equations for relative motion. To begin, it is assumed that:

Assumption 1 Both the target and chaser are in passive orbits around the Earth. This implies that only the unforced dynamics are modelled, however inclusion of relative control inputs to the system can be included without a loss of generality;

Assumption 2 The distance between the target and chaser spacecraft is small, relative to their distance from the center of the Earth, their distance from the moon, and their distance from the Sun;

Assumption 3 As per Assumption 2, the differential effects of atmospheric drag, solar radiation pressure, and third-body gravity are negligible compared to the forces caused by the primary body [13, 87];

Assumption 4 Thus, the two-body equation can be used to describe the motion of the target and chaser spacecraft in the development of relative motion dynamics.

From the assumptions stated above, the equations of motion for the target and chaser spacecraft are written as

$$\ddot{\vec{r}}_t = -\mu \frac{\vec{r}_t}{r_t^3} \qquad\qquad \ddot{\vec{r}}_c = -\mu \frac{\vec{r}_c}{r_c^3} \tag{2.33}$$

where r_t and r_c are the magnitude of the target and chaser position vectors $\vec{r}_t$ and $\vec{r}_c$ with respect to the ECI reference frame $\vec{\mathcal{F}}_I$. Since Earth-based orbits are assumed in this context, μ implies the gravitational parameter of the Earth and the subscript $\oplus$ is unnecessary. Defining the relative position of the chaser with respect to the target by the position vector $\vec{\rho}$, it can be seen from Figure 2.4 shown previously that the position of the chaser is given by

$$\vec{r}_c = \vec{r}_t + \vec{\rho} \tag{2.34}$$

Taking advantage of the definition of the LVLH frame, the relative position vector can be represented by the component set $\{x, y, z\}$ within the LVLH reference frame where the *radial* component x and the *along-track* component y capture the relative motion within the orbital plane of the target spacecraft. The out-of-plane motion is described by the *cross-track* term z, and the resulting relative position vector is

$$\vec{\rho} = \vec{\mathcal{F}}_L^T \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{2.35}$$

A complete derivation of the relative equations of motion is included in Appendix B.6, from which the nonlinear equations of motion that describe the relative position of the chaser with respect to the target in the LVLH frame are:

$$\ddot{x} - \dot{\theta}^2 x - 2\dot{\theta}\left(\dot{y} - y\frac{\dot{r_t}}{r_t}\right) - \frac{\mu}{r_t^2} = -\frac{\mu}{r_c^3}(r_t + x) \tag{2.36}$$

$$\ddot{y} - \dot{\theta}^2 y + 2\dot{\theta}\left(\dot{x} - x\frac{\dot{r_t}}{r_t}\right) = -\frac{\mu}{r_c^3} y \tag{2.37}$$

$$\ddot{z} = -\frac{\mu}{r_c^3} z \tag{2.38}$$

From the geometry of the formation, the distance from the Earth to the chaser spacecraft is given by $r_c = \sqrt{(r_t + x)^2 + y^2 + z^2}$. Additionally, the target spacecraft's true anomaly θ and radius r_t are governed by another set of nonlinear differential equations. Assuming that both the target and chaser spacecraft are in Keplerian orbits as before, the final two relations are

$$\ddot{\theta} = -2\frac{\dot{r_t}}{r_t}\dot{\theta} \tag{2.39}$$

$$\ddot{r_t} = \dot{\theta}^2 r_t - \frac{\mu}{r_t^2} \tag{2.40}$$

Equations (2.36) through (2.40) comprise the set of five exact, nonlinear differential equations of motion that describe the relative dynamics between the target and chaser spacecraft. These equations are the basis for propagating the states of the spacecraft within the LVLH frame, which is discussed later in Chapter 3 during the development of the extended Kalman filter.

2.6.3 Regarding the Hill-Clohessy-Wiltshire Equations

One of the advantages of the EKF algorithms developed for this work, is that they maintain the full nonlinear dynamics equations presented in the preceding section. In past demonstrations of relative navigation for formation flying (such as the successful CanX-4/5 [88] mission), further simplifications are made to the nonlinear dynamics so that closed-form solutions can be obtained. By assuming that the orbit of the target is circular and that the distance between both spacecraft is small, the famous Hill's equations of relative motion can be derived as:

$$\ddot{x} = 3n_0^2 x + 2n_0\dot{y} \tag{2.41}$$

$$\ddot{y} = -2n_0\dot{x} \tag{2.42}$$

$$\ddot{z} = -n_0^2 z \tag{2.43}$$

where n_0 is the mean orbital motion of the spacecraft. These historic equations were proposed in 1878 by George W. Hill (1838–1914) for describing the motion of the Moon about the Earth [89–91], but it wasn't until 1960 that the analytic solutions to these equations were discovered by Clohessy and Wiltshire [92] in their work on spacecraft rendezvous. Derivation of the Hill's equations and the resulting Clohessy-Wiltshire (CW) equations are included in Appendix B.6.6 to be thorough, but these equations are not implemented within the Kalman filtering algorithms due to their limited applicability to non-circular orbits.

Spacecraft are rarely in perfectly circular orbits, and the performance of the CW equations rapidly decreases as the eccentricity of an orbit increases. Since the navigation filters proposed here are designed to be suitable for multiple formation configurations and orbit types, the limitations imposed by the circular target orbit assumption of the Hill's equations are unacceptable. Furthermore, the work of Kuiack and Ulrich [93] attributes larger overall fuel consumption to errors in the known trajectory of a spacecraft due to unmodelled dynamics, as the control manoeuvers to correct the trajectory errors are larger and more frequent. It is indeed desirable to therefore maintain the most accurate relative dynamics model for use in the EKF, so as to avoid additional errors from modeling the true dynamic system.

2.7 Formation Configurations

The analyses of the filtering algorithms developed for this thesis make use of several formation configurations, in an effort to convey the unique range of possible operating conditions where the filters are applicable. The following section introduces three configurations based on operational spacecraft formations, and shows the associated nominal orbit and relative motion trajectories that approximate the true spacecraft motion. These simulations represent the *truth model*, or the true states of the spacecraft at a given time, and will be used throughout the rest of the thesis to verify the accuracy of the navigation solutions. The three formations that are considered here include the on-orbit PRISMA formation launched in 2010, the future PROBA-3 mission planned for launch in 2020, and a generic projected elliptical orbit in low-Earth orbit.

2.7.1 Case 1: PRISMA Formation

The PRISMA mission consists of two small spacecraft, TANGO and MANGO, in a low-Earth, Sun-synchronous orbit. The orbits of both spacecraft have very small eccentricities, and the operational altitude band ranges from 668 km to 749 km. Both spacecraft are therefore subject to the effects of atmospheric drag and solar radiation pressure, which is suitable for demonstrating the perturbation models that are used within the real-world orbit propagator. The initial orbital elements for the PRISMA formation are presented in Table 2.2, based on the report by Montenbruck *et al.* [94].

Table 2.2: Initial Orbital Elements of the PRISMA Formation

Parameter	Target (TANGO)	Chaser (MANGO)
True Anomaly θ	358.903 490 28°	0.00°
Semi-Major Axis a	7 087.297 556 866 34 km	7 087.297 677 331 79 km
Eccentricity e	0.001 454 43	0.001 459 08
Inclination i	98.185 286 13°	98.184 666 76°
RAAN Ω	189.891 384 5°	189.890 860 2°
Argument of Perigee ω	1.097 451 382°	0.00°
Orbital Period	5 938 s 1 h 39 m	5 938 s 1 h 39 m

Physical properties of the TANGO and MANGO spacecraft are given in Table 2.9, based on values used in the testing environment for PRISMA [13]. Figure 2.9 shows the resulting relative orbit for the PRISMA formation, as simulated using the real-word orbit propagator described in Sections 2.3.2 through 2.6.1. Note that the relative motion of the chaser is offset from the target, such that the closest approach of the chaser is maintained above 110 m.

Table 2.3: Spacecraft Properties for the PRISMA Formation

Parameter	Target (TANGO)	Chaser (MANGO)
Mass, m	42.5 kg	154.4 kg
Drag Area, A_d	0.38 m^2	2.75 m^2
Drag Coefficient, C_d	2.25	2.5
SRP Area, A_r	0.55 m^2	2.5 m^2
SRP Coefficient, C_r	1.2 m^2	1.32

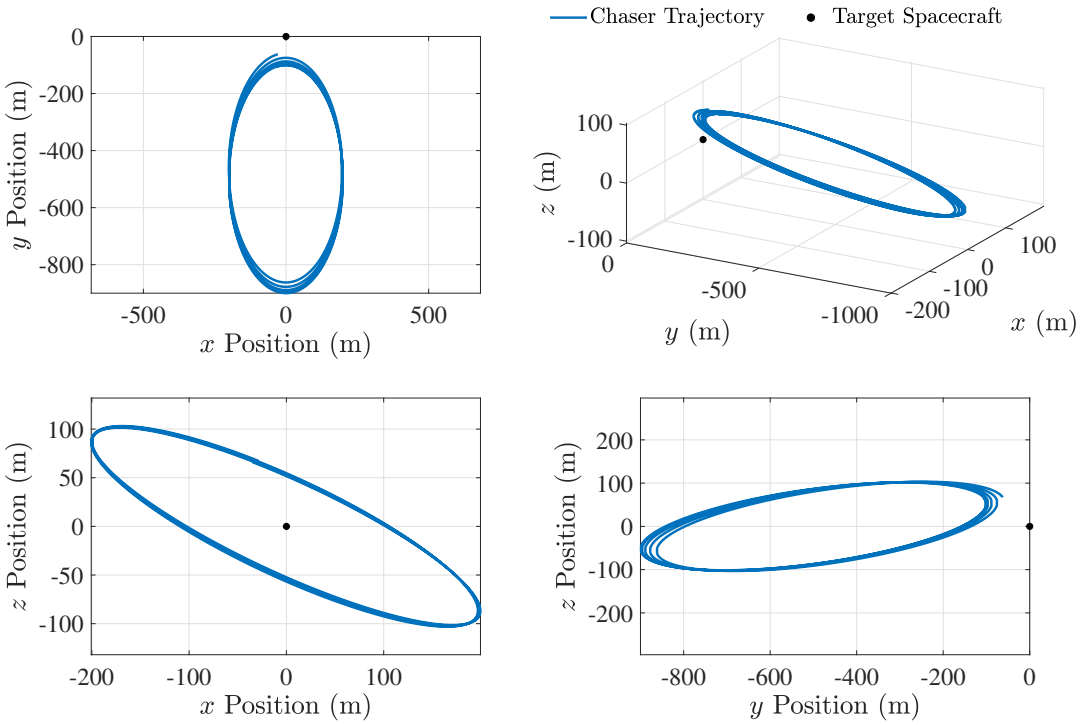


Figure 2.9: Relative orbit of the PRISMA mission over five orbital periods of the target spacecraft, under the influence of perturbations from solar radiation pressure, atmospheric drag, third-body gravity and J_2.

2.7.2 Case 2: PROBA-3 Formation

Contrasting the LEO environment that the PRISMA formation occupies, the PROBA-3 formation uses a highly-elliptical orbit with a considerably large variation in altitude. At perigee, the spacecraft pass below 600 km, and at apogee they pass outside of 60 530 km [95]; this large apogee altitude is selected as a mission requirement for the PROBA-3 observation platform, however simulation of this orbit configuration here provides a means to examine both the real-world propagator models and the navigation algorithms proposed later. The two spacecraft of PROBA-3 are scientific platforms designated as the Occulter and the Coronograph, and the orbital elements and spacecraft parameters listed in Tables 2.4 and 2.5 are used for simulating the PROBA-3 formation, as provided by Peters [96].

Table 2.4: Initial Orbital Elements of the PROBA-3 Formation

Parameter	Target (Occulter)	Chaser (Coronograph)
True Anomaly θ	0°	0°
Semi-Major Axis a	36 943 km	36 943 km
Eccentricity e	0.8111	0.811105
Inclination i	59°	59°
RAAN Ω	84°	84°
Argument of Perigee ω	188°	188°
Orbital Period	70 680 s 19 h 38 m	70 680 s 19 h 38 m

Table 2.5: Spacecraft Properties for the PROBA-3 Formation

Parameter	Target (Occulter)	Chaser (Coronograph)
Mass, m	211 kg	339 kg
Drag Area, A_d	1.77 m^2	3.34 m^2
Drag Coefficient, C_d	2.5	2.5
SRP Area, A_r	1.77 m^2	3.34 m^2
SRP Coefficient, C_r	1.9 m^2	1.29

The relative motion of PROBA-3 is significantly different than that of PRISMA, as shown below in Fig. 2.10. Here it can be seen that the PROBA-3 formation will drift apart without the application of station keeping manoeuvres, but the uncontrolled motion is a suitable truth trajectory for the purposes of this research.

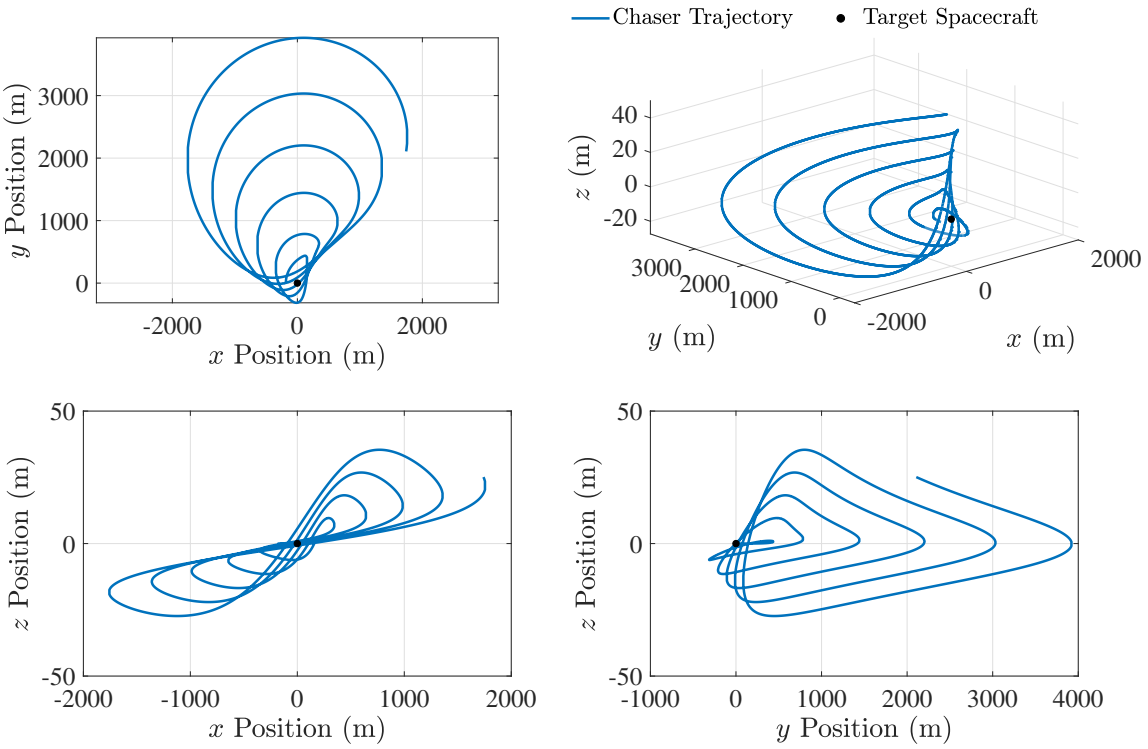


Figure 2.10: Relative orbit of the PROBA-3 formation over five orbital periods of the target spacecraft, showing that the chaser will continue to drift away from the target under the influence of orbital perturbations.

2.7.3 Case 3: Projected Elliptical Orbit Formation in LEO

The last case considered is a Projected-Elliptical Orbit (PEO) operating in the low-Earth orbit environment. This scenario is an arbitrary formation configuration with a medium eccentricity orbit, loosely based on the orbit and spacecraft properties of the PRISMA mission. The selection of this alternate configuration creates a non-circular orbit that has characteristics of a LEO formation, but experiences higher non-linearities in the dynamics due to the eccentricity and altitude variations that occur. Furthermore, the orbit of the chaser envelopes the target, as there is no offset in the centering of the relative orbit. The perigee and apogee altitudes are 372 km and 1 862 km respectively, and the target spacecraft orbital plane is inclined at 98.178°. The remaining COEs describing the PEO formation are given in Table 2.6, and Table 2.7 contains the physical spacecraft properties.

Table 2.6: Initial Orbital Elements of the Projected Elliptical Orbit Formation

Parameter	Target	Chaser
True Anomaly θ	0°	0°
Semi-Major Axis a	7 500 km	7 500 km
Eccentricity e	0.100 00	0.100 05
Inclination i	98.188°	98.178°
RAAN Ω	189.891°	189.891°
Argument of Perigee ω	1.094°	1.094°
Orbital Period	6 464 s 1 h 47.7 m	6 464 s 1 h 47.7 m

Table 2.7: Spacecraft Properties for the Projected Elliptical Orbit Formation

Parameter	Target	Chaser
Mass, m	42.5 kg	154.4 kg
Drag Area, A_d	0.38 m^2	1.3 m^2
Drag Coefficient, C_d	2.25	2.5
SRP Area, A_r	0.55 m^2	2.5 m^2
SRP Coefficient, C_r	1.2 m^2	1.32

Looking at the upper-left plot in Fig. 2.11, note that the relative motion of the PEO formation within the $\vec{\mathcal{L}}_x - \vec{\mathcal{L}}_y$ plane is indeed elliptical. The apparent drifting of the projected ellipse is a result of the orbit perturbations that are included in the orbit simulator, where in pure Keplerian motion the bounded relative orbit would remain stationary. Furthermore, this formation configuration contains larger cross-track motion than either the PRISMA or PROBA-3 cases by design, which will ensure the filtering algorithms are well-suited for full three-dimensional relative motion scenarios.

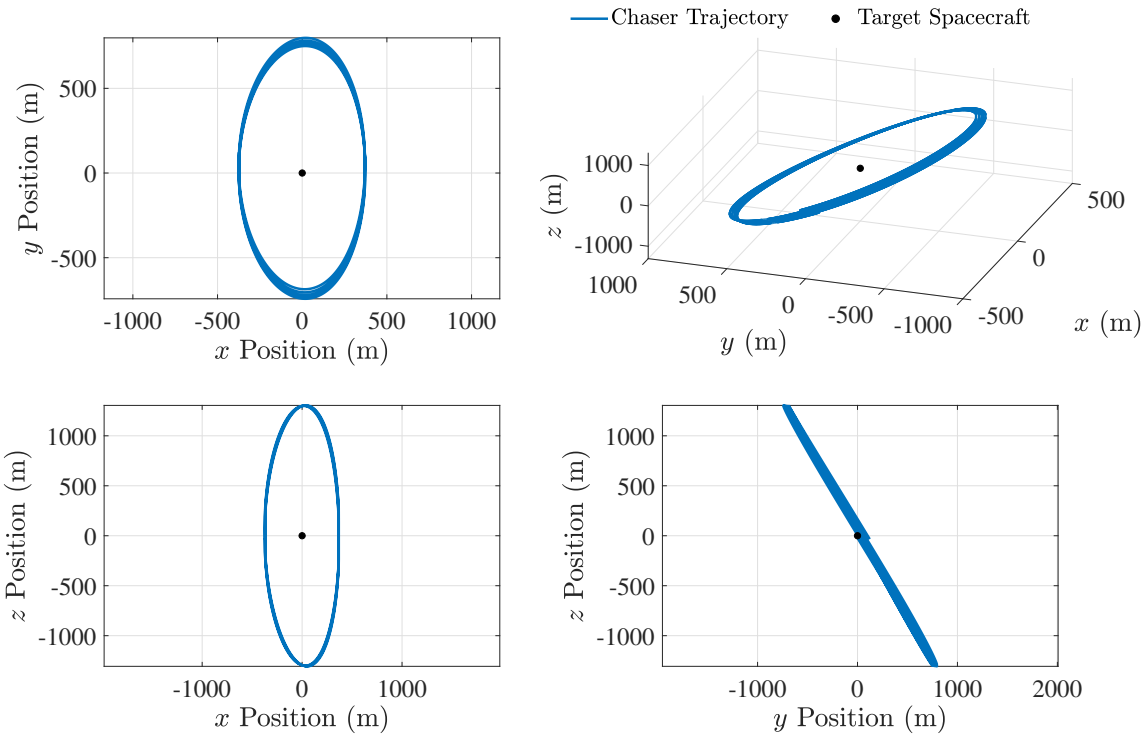


Figure 2.11: A projected elliptical orbit in LEO, simulated for five orbital periods of the target Spacecraft. From its namesake, when this orbit is viewed from above, the projection of the relative motion onto the $x - y$ plane produces an ellipse centered on the target.

2.8 Chapter Summary

This section presented an overview of the physical phenomena that dominate the motion of near-Earth spacecraft, culminating in the development of a numerical simulator that propagates the motion of a spacecraft in inertial space. The conversions necessary to transform the inertial spacecraft states into a local-vertical local-horizontal reference frame were identified, providing an intuitive description of the relative motion from the perspective of the target spacecraft. Additionally, a set of relative dynamics equations were presented that analytically model the motion of spacecraft within the LVLH frame. Using the results presented so far, the next chapter will investigate the EKF technique for state estimation. Real-world navigation systems do not have an exact model for the environment they operate in, so the EKF provides a method of combining information from measurements of the real-world with a dynamics model that approximates the real-world.

Chapter 3

Extended Kalman Filtering

This chapter presents the mathematics of the EKF algorithm, and outlines the design of an EKF capable of estimating the relative position and velocity states of a dual spacecraft formation. The dynamics and measurement models used within the EKF are discussed, and proof of observability for the selected models is given. Lastly, implementation considerations of the EKF are identified as they pertain to simulations used for this research. For an in-depth derivation of the linear and extended variants of the Kalman filter equations, please refer to Appendix C.

3.1 Introduction

When modeling real-world systems, it is rarely possible to have a complete knowledge of every state of the system. A special type of filter, known as an *observer*, can therefore be designed to estimate the states based on dynamic models and available measurements of the system. By fusing information from the dynamics and the measurements, observers are able to provide a better estimate of the states then the individual methods could produce alone. Although many observer-based filtering techniques are available, such as the Luenburger Observer, the Sliding-Mode Observer, the Disturbance Observer and the Particle Filter [97], a method known as Kalman filtering has become the ubiquitous standard for real-time observer design.

Rudolph Kalman (1930-2016) introduced his new approach to minimum mean-square error filtering in 1960 [19], which uses a sequential and recursive algorithm with a predictor-corrector structure. An initial estimate of the state is made based on a model of the system dynamics, and this prediction is then corrected to account for measurements made by real-world sensors [66]. The Kalman filter is a *stochastic* observer, meaning that it uses knowledge of the noise statistics to improve the estimates of the states; throughout the process, the error covariance of the state estimate is

formulated, providing an explicit metric for the uncertainty of the estimate. Several properties of the Kalman filter contribute to it's effectiveness:

Recursive The next estimate of the state is calculated using the preceding estimate and a current measurement. This significantly reduces the computational time compared to batch estimation methods, which analyze an entire data set to produce an estimate of the current system state.

Cumulative Each estimate is influenced by the previous state estimates and measurements, since the error covariance is propagated through the entire routine. Thus, "memory" of the previous information still influences future estimates without adding the computational burden of re-processing the entire data set.

Tunable The Kalman filter incorporates terms that characterize the uncertainty expected within the system, given that noise sources present in dynamic models and measurements will likely be unique. *Tuning* these parameters allows the user to optimize the filter performance based on the specific system and sensors.

As introduced in Chapter 2, the governing orbital mechanics that influence the motion of spacecraft give rise to nonlinear dynamics equations. The relative motion of a spacecraft formation can thus be cast in a general continuous-time framework as

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{w}, t) \tag{3.1}$$

Equations of the nonlinear dynamical model reside within the term $\mathbf{f}(\cdot)$, which is a function of the state vector $\mathbf{x} \in \mathbb{R}^n$ and known control inputs $\mathbf{u} \in \mathbb{R}^q$. Furthermore, the dynamics model includes process noises $\mathbf{w} \in \mathbb{R}^p$ to account for modelling inaccuracies. If a set of measurements $\mathbf{z}_k \in \mathbb{R}^m$ of the system are taken by sensors at a discrete time t_k, a corresponding set of measurement equations $\mathbf{h}(\cdot)$ can be used to model the sensors, thereby mapping the system states into the observation space. The measurement equations include the effects of measurement noises $\mathbf{v}_k \in \mathbb{R}^m$, which gives the following general form for the nonlinear measurement model:

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k, \mathbf{v}_k) \tag{3.2}$$

Note that the measurement model here is defined using discrete time, to align with the sampling behaviour of modern digital sensors. Both noise processes corrupting the dynamics and measurement models are taken to be Gaussian and white, with zero-mean. Further assuming that these noises are uncorrelated with each other, the

noise processes at each discrete time k can be classified by normal distributions with

$$\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k) \qquad \mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k) \qquad E[\mathbf{v}_k \mathbf{w}_k^T] = \mathbf{0}$$

where the notation $E[\cdot]$ is used to designate the expectation operator. With the assumptions listed above, the noise processes can be fully characterized in terms of their positive definite symmetric covariance matrices; using $\mathbf{Q}_k \in \mathbb{R}^{n \times n}$ to define the process noise covariance, and $\mathbf{R}_k \in \mathbb{R}^{m \times m}$ to define the measurement noise covariance, this gives

$$\mathbf{Q}_k = E[\mathbf{w}_k \mathbf{w}_k^T] \qquad \mathbf{R}_k = E[\mathbf{v}_k \mathbf{v}_k^T]$$

Since the purpose of the Kalman filter is to estimate the state of the system, it is useful to consider the error between the true state $\mathbf{x}_k$ and the predicted state $\hat{\mathbf{x}}_k$. The state error $\mathbf{e}_k = \mathbf{x}_k - \hat{\mathbf{x}}_k$ then quantifies the errors in both the prediction and correction steps of the filtering algorithm. The state estimate available before a measurement has been processed is called the *a priori* estimate, denoted by $\hat{\mathbf{x}}_k^-$. This corresponds to the best guess possible with the dynamics described by the physical model. The *a posteriori* state estimate $\hat{\mathbf{x}}_k^+$ is obtained after a measurement is made at the current time t_k. It is standard to assume that the *a priori* and *a posteriori* state estimates are uncorrelated with the process and measurement noise, so

$$E[\mathbf{e}_k^- \mathbf{w}_k^T] = \mathbf{0} \qquad E[\mathbf{e}_k^+ \mathbf{w}_k^T] = \mathbf{0} \qquad E[\mathbf{e}_k^- \mathbf{v}_k^T] = \mathbf{0} \qquad E[\mathbf{e}_k^+ \mathbf{v}_k^T] = \mathbf{0}$$

Moreover, the covariance matrix $\mathbf{P}_k \in \mathbb{R}^{n \times n}$ is defined for the state errors to demonstrate the error distribution. The covariance matrix provides an intuitive feel for the level of uncertainty in the current estimate, as it can be seen that each term in the covariance matrix essentially represents the square of the error between the true and estimated states. Thus, large values within $\mathbf{P}_k^-$ indicate that the *a priori* estimate is inaccurate, while small $\mathbf{P}_k^+$ elements suggest the *a posteriori* estimate is close to the true state. The state error covariance is defined as

$$\mathbf{P}_k = E[\mathbf{e}_k \mathbf{e}_k^T] = E[(\mathbf{x}_k - \hat{\mathbf{x}}_k)(\mathbf{x}_k - \hat{\mathbf{x}}_k)^T] \tag{3.3}$$

In order to proceed with the use of standard Kalman filtering techniques, it is assumed that $\mathbf{f}(\cdot)$ and $\mathbf{h}(\cdot)$ are sufficiently smooth such that they can be approximated using Taylor series expansions. The Taylor expansion is necessary for the development of the EKF, which utilizes only the linear terms of the expansion. Although higher-order Kalman filters have been investigated [98], flight heritage and spacecraft computational restrictions favor the accuracy and efficiency of the standard EKF.

3.2 The Extended Kalman Filter

The objective of the EKF is to obtain an estimate of the system state through sequentially filtering the dynamics and measurements through time. Estimates are herein denoted with the hat operator, such that the state estimate $\hat{\mathbf{x}}_k$ at a given time t_k is defined by

$$\hat{\mathbf{x}}_k = E[\mathbf{x}_k] \tag{3.4}$$

Based on the system model from Eq. (3.1), the state estimate dynamics must adhere to the differential equation:

$$\hat{\mathbf{x}}_{k+1} = E[\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{0})] \equiv \mathbf{f}(\hat{\mathbf{x}}_k, \mathbf{u}_k, \mathbf{0}) \tag{3.5}$$

From the estimated state dynamics in Eq. (3.5), it is seen that the process noise is taken to be zero during the evolution of the state estimates. This is a reasonable postulation because in reality the stochastic noise is not known, but is assumed to be characterized following a zero-mean white Gaussian distribution. Furthermore, the dynamics are assumed to be passive and uncontrolled for this research, so $\mathbf{u}_k = \mathbf{0}\ \forall k$. The terms $\mathbf{u}_k$ and $\mathbf{w}_k$ are therefore omitted in the estimated dynamics arguments throughout the remaining portions of this chapter to simplify the notation.

The Propagation Phase

The first stage of the Kalman filter relies on the estimation of the state before a measurement is collected. Noting that the dynamics are defined in the continuous domain, numerically integrating Eq. (3.5) from the past time step t_{k-1} to the current time step t_k gives an *a priori* state estimate through

$$\hat{\mathbf{x}}_k^- = \hat{\mathbf{x}}_{k-1} + \int_{t_{k-1}}^{t_k} \mathbf{f}(\hat{\mathbf{x}}_{k-1}, t)\ \mathrm{d}t \tag{3.6}$$

Due to the errors in modelling the true dynamics of the system, this propagation will accordingly introduce some error into the state estimate, which must be quantified in the error covariance matrix $\mathbf{P}_k$. Contrasting the nonlinear propagation of the state estimate, the error covariance is propagated using a linearized version of the dynamics. The process of linearization involves evaluating the *Jacobian* of $\mathbf{f}(\cdot)$ at the best state estimate from the previous time step, namely the *a posteriori* state estimate $\hat{\mathbf{x}}_{k-1}^+$, whereby the Jacobian is

$$\mathbf{F}_{k-1} = \frac{\partial \mathbf{f}(\mathbf{x}, t)}{\partial \mathbf{x}}\bigg|_{\hat{\mathbf{x}}_{k-1}^+} \tag{3.7}$$

The state transition matrix, which performs a state mapping from time t_{k-1} to t_k through the time step $\Delta t = t_k - t_{k-1}$, is then obtained from

$$\mathbf{\Phi}_{k-1} = \exp\left[\mathbf{F}_{k-1}\Delta t\right] \triangleq \sum_{n=0}^{\infty} \frac{\left(\mathbf{F}_{k-1}\Delta t\right)^n}{n!} \tag{3.8}$$

In practice, the Taylor series expansion of the matrix exponential shown above is usually used for calculating the matrix exponential efficiently, and using the first $n = 4$ terms is sufficient for simulation within this research. Finally, the *a priori* state error covariance matrix is found by propagating the *a posteriori* error covariance $\mathbf{P}_{k-1}^{+}$ from the previous time step with

$$\mathbf{P}_k^{-} = \mathbf{\Phi}_{k-1}\mathbf{P}_{k-1}^{+}\mathbf{\Phi}_{k-1}^{T} + \mathbf{Q}_k \tag{3.9}$$

The Measurement Phase

Although the first phase of the EKF provides an estimate of the state and the associated error, the accuracy of the propagation is limited by the fidelity of the model and the associated process noise; corrections to the *a priori* estimates are therefore made using measurements of the system. With the measurement model from Eq. (3.2) and the propagated state vector from Eq. (3.6), estimates of the measurements are calculated with

$$\hat{\mathbf{z}}_k = \mathbf{h}(\hat{\mathbf{x}}_k^{-}) \tag{3.10}$$

These estimated measurements will be subject to errors resulting from inaccuracies in modelling the sensor (manifesting in the calculation of $\hat{\mathbf{z}}_k$), and errors within the propagation of the previous state estimate (manifesting in the calculation of $\hat{\mathbf{x}}_k^{-}$). The differences between the actual measurements and the estimated measurements are commonly known as the *innovations*, the *pre-fit residuals*, or simply the *residuals*. The symbol $\boldsymbol{\nu}_k$ is used to denote the residuals here, which are defined by

$$\boldsymbol{\nu}_k = \mathbf{z}_k - \hat{\mathbf{z}}_k \tag{3.11}$$

The physical interpretation of the residuals relates the correctness of the dynamics and measurements models to the observed data. Larger residual values indicate that the measurement estimates calculated from the propagated estimates are not accurately matching the observed measurements, in turn implying that the model for the system does not perfectly describe the system. As proven by Kailath [99], if the filter is performing optimally the innovations should be a zero-mean white Gaussian noise

process. This is discussed further in Chapters 4 and 5, where the innovations serve as a basis for adapting the EKF.

To complete the state correction, the measurement model $\mathbf{h}_k$ is linearized by evaluating its Jacobian at the current best estimate of the state, similar to the procedure that was done for linearizing the dynamics. The linearized measurement matrix is

$$\mathbf{H}_k = \left.\frac{\partial \mathbf{h}(\mathbf{x})}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_k^-} \tag{3.12}$$

and this allows the Kalman gain $\mathbf{K}_k$ to be obtained from the error and measurement noise covariances, where

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T \left(\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k\right)^{-1} \tag{3.13}$$

Note here that the term in parenthesis above corresponds to the theoretical covariance of the residuals $\mathbf{\Sigma}_k$, which is also useful for later adapting the Kalman filter. Stating things formally, the residuals covariance matrix predicted by the filter is defined as

$$\mathbf{\Sigma}_k = \mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k \tag{3.14}$$

The formulation of the Kalman gain is derived through a minimization of the covariance error matrix of the *a posteriori* state vector, which is analogous to maximizing the confidence in the estimated state [66]. From this, the corrected state is found by scaling the residuals with the Kalman gain, so

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{K}_k \left(\mathbf{z}_k - \hat{\mathbf{z}}_k\right) = \hat{\mathbf{x}}_k^- + \mathbf{K}_k \boldsymbol{\nu}_k \tag{3.15}$$

The final calculation within the EKF involves updating the state error covariance matrix. The Joseph Stabilized form [100] of the covariance update is used here for numerical stability:

$$\mathbf{P}_k^+ = \left(\mathbf{I} - \mathbf{K}_k \mathbf{H}_k\right) \mathbf{P}_k^- \left(\mathbf{I} - \mathbf{K}_k \mathbf{H}_k\right)^T + \mathbf{K}_k \mathbf{R}_k \mathbf{K}_k^T \tag{3.16}$$

where $\mathbf{I}$ is the identity matrix with appropriate dimensions. Numerical stability in this case means that $\mathbf{P}_k^+$ will be positive definite symmetric (PDS), so long as $\mathbf{P}_k^-$ is PDS [54]. This symmetric property of the Joseph Stabilized form can be noted from inspection of (3.16). To conclude this introduction to the EKF, a diagram of the general state flow of the observer is shown in Figure 3.1.

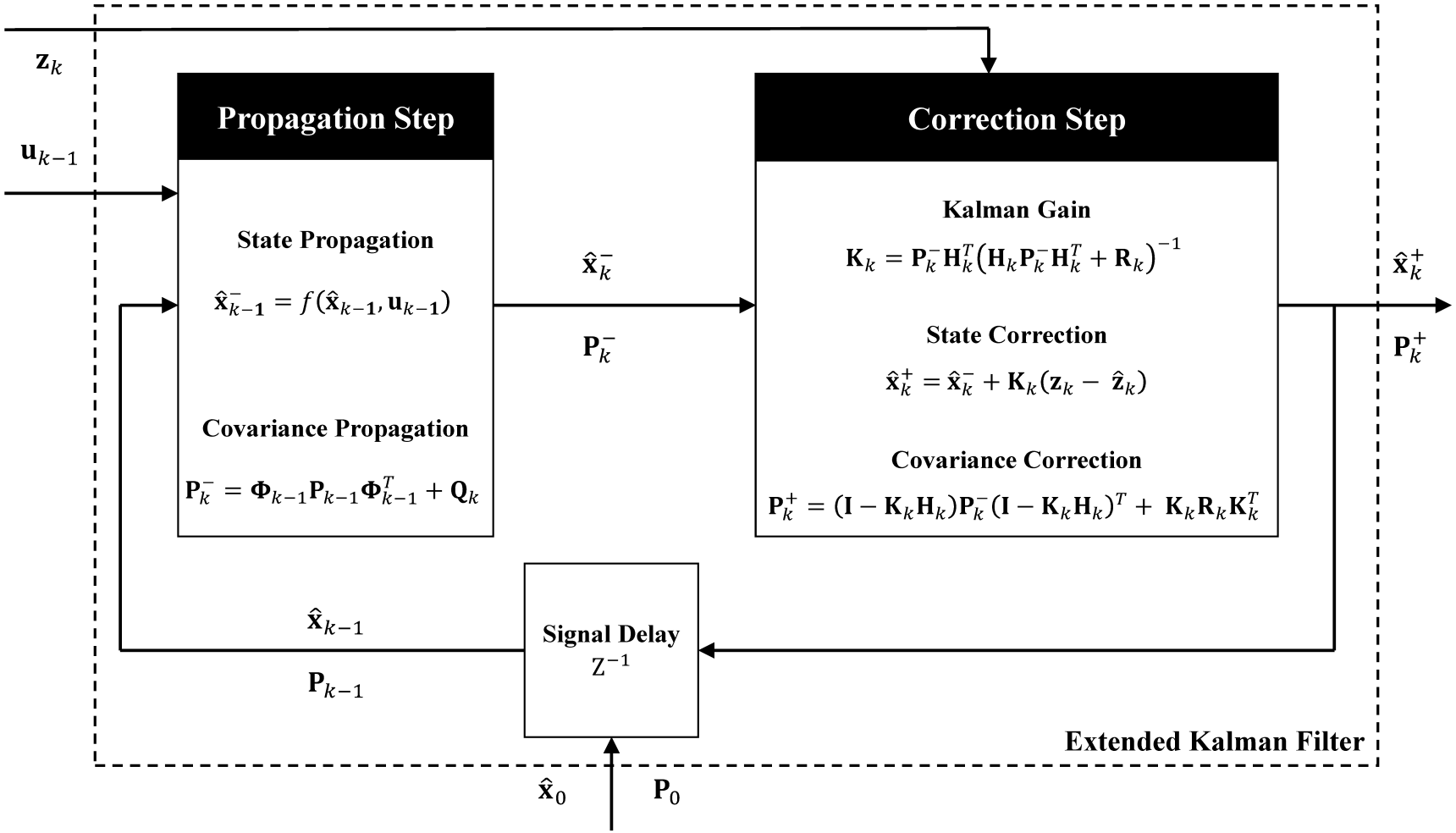


Figure 3.1: Block diagram showing the predictor-corrector structure of the Kalman filter.

3.3 Dynamics Modelling

To cast the relative navigation problem into the Kalman filter framework, this section identifies the state vector and dynamics models used for this research. Recall that the nonlinear equations of relative motion consisted of the five second-order, ordinary differential equations shown in Eqs. (2.36)-(2.40). This implies a total of $n = 10$ system states, for which the state vector $\mathbf{x} \in \mathbb{R}^{10}$ for the formation is defined as

$$\mathbf{x} \triangleq \begin{bmatrix} x & y & z & \theta & r_t & \dot{x} & \dot{y} & \dot{z} & \dot{\theta} & \dot{r}_t \end{bmatrix}^T \tag{3.17}$$

and the corresponding dynamics are described by $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$. For the nonlinear equations of motion, the dynamics vector can be written is state space form using

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\theta} \\ \dot{r}_t \\ \dot{\theta}^2 x + 2\dot{\theta}\left(\dot{y} - y\dfrac{\dot{r}_t}{r_t}\right) + \dfrac{\mu}{r_t^2} - \dfrac{\mu}{r_c^3}(r_t + x) \\ \dot{\theta}^2 y - 2\dot{\theta}\left(\dot{x} - x\dfrac{\dot{r}_t}{r_t}\right) - \dfrac{\mu}{r_c^3} y \\ -\dfrac{\mu}{r_c^3} z \\ -2\dfrac{\dot{r}_t}{r_t}\dot{\theta} \\ \dot{\theta}^2 r_t - \dfrac{\mu}{r_t^2} \end{bmatrix} \tag{3.18}$$

As a reminder, the parameter μ is a constant corresponding to the gravitational parameter, and the radius of the chaser spacecraft is given by $r_c = \sqrt{(r_t + x)^2 + y^2 + z^2}$ and is not a state of specific interest within the estimation algorithm.

Linearization of the dynamics is completed by taking the partial derivatives of the dynamics with respect to the states, which gives the Jacobian matrix $\mathbf{F} \in \mathbb{R}^{n \times n}$:

$$\mathbf{F} = \frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ \ddot{x}_x & \ddot{x}_y & \ddot{x}_z & \ddot{x}_\theta & \ddot{x}_{r_t} & \ddot{x}_{\dot{x}} & \ddot{x}_{\dot{y}} & \ddot{x}_{\dot{z}} & \ddot{x}_{\dot{\theta}} & \ddot{x}_{\dot{r}_t} \\ \ddot{y}_x & \ddot{y}_y & \ddot{y}_z & \ddot{y}_\theta & \ddot{y}_{r_t} & \ddot{y}_{\dot{x}} & \ddot{y}_{\dot{y}} & \ddot{y}_{\dot{z}} & \ddot{y}_{\dot{\theta}} & \ddot{y}_{\dot{r}_t} \\ \ddot{z}_x & \ddot{z}_y & \ddot{z}_z & \ddot{z}_\theta & \ddot{z}_{r_t} & \ddot{z}_{\dot{x}} & \ddot{z}_{\dot{y}} & \ddot{z}_{\dot{z}} & \ddot{z}_{\dot{\theta}} & \ddot{z}_{\dot{r}_t} \\ \ddot{\theta}_x & \ddot{\theta}_y & \ddot{\theta}_z & \ddot{\theta}_\theta & \ddot{\theta}_{r_t} & \ddot{\theta}_{\dot{x}} & \ddot{\theta}_{\dot{y}} & \ddot{\theta}_{\dot{z}} & \ddot{\theta}_{\dot{\theta}} & \ddot{\theta}_{\dot{r}_t} \\ \ddot{r}_{t_x} & \ddot{r}_{t_y} & \ddot{r}_{t_z} & \ddot{r}_{t_\theta} & \ddot{r}_{t_{r_t}} & \ddot{r}_{t_{\dot{x}}} & \ddot{r}_{t_{\dot{y}}} & \ddot{r}_{t_{\dot{z}}} & \ddot{r}_{t_{\dot{\theta}}} & \ddot{r}_{t_{\dot{r}_t}} \end{bmatrix} \tag{3.19}$$

The derivative shorthand of using subscripts is adopted here to simplify the notation, which indicates that the partial derivative for a general function f with respect to an arbitrary variable x is defined as

$$f_x = \frac{\partial f}{\partial x} \tag{3.20}$$

After completing the analytic partial derivatives, a number of elements in the linearized dynamics model go to zero. The simplified Jacobian is written below, and the non-zero partial derivative components are presented on the following pages. The complete set of partial derivatives is presented in Appendix B.6.

$$\mathbf{F} = \frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ \ddot{x}_x & \ddot{x}_y & \ddot{x}_z & 0 & \ddot{x}_{r_t} & 0 & \ddot{x}_{\dot{y}} & 0 & \ddot{x}_{\dot{\theta}} & \ddot{x}_{\dot{r}_t} \\ \ddot{y}_x & \ddot{y}_y & \ddot{y}_z & 0 & \ddot{y}_{r_t} & \ddot{y}_{\dot{x}} & 0 & 0 & \ddot{y}_{\dot{\theta}} & \ddot{y}_{\dot{r}_t} \\ \ddot{z}_x & \ddot{z}_y & \ddot{z}_z & 0 & \ddot{z}_{r_t} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \ddot{\theta}_{r_t} & 0 & 0 & 0 & \ddot{\theta}_{\dot{\theta}} & \ddot{\theta}_{\dot{r}_t} \\ 0 & 0 & 0 & 0 & \ddot{r}_{t_{r_t}} & 0 & 0 & 0 & \ddot{r}_{t_{\dot{\theta}}} & 0 \end{bmatrix} \tag{3.21}$$

Partial derivatives of the radial position dynamics with respect to the states are:

$$\ddot{x}_x = \frac{\partial \ddot{x}}{\partial x} = \frac{\mu}{r_c^3}\left[3\left(\frac{r_t + x}{r_c}\right)^2 - 1\right] + \dot{\theta}^2 \tag{3.22}$$

$$\ddot{x}_{\dot{y}} = \frac{\partial \ddot{x}}{\partial \dot{y}} = 2\dot{\theta} \tag{3.23}$$

$$\ddot{x}_y = \frac{\partial \ddot{x}}{\partial y} = 3\mu\left(\frac{r_t + x}{r_c^5}\right)y - 2\dot{\theta}\frac{\dot{r}_t}{r_t} \tag{3.24}$$

$$\ddot{x}_{\dot{r}_t} = \frac{\partial \ddot{x}}{\partial \dot{r}_t} = -2\frac{\dot{\theta}}{r_t}y \tag{3.25}$$

$$\ddot{x}_z = \frac{\partial \ddot{x}}{\partial z} = 3\mu\left(\frac{r_t + x}{r_c^5}\right)z \tag{3.26}$$

$$\ddot{x}_{\dot{\theta}} = \frac{\partial \ddot{x}}{\partial \dot{\theta}} = 2\left(\dot{\theta}x + \dot{y} - \frac{\dot{r}_t}{r_t}y\right) \tag{3.27}$$

$$\ddot{x}_{r_t} = \frac{\partial \ddot{x}}{\partial r_t} = 2\dot{\theta}\frac{\dot{r}_t}{r_t^2}y - 2\frac{\mu}{r_t^3} + \frac{\mu}{r_c^3}\left[3\left(\frac{r_t + x}{r_c}\right)^2 - 1\right] \tag{3.28}$$

and linearizing the along-track position dynamics yields:

$$\ddot{y}_x = \frac{\partial \ddot{y}}{\partial x} = 2\dot{\theta}\frac{\dot{r}_t}{r_t} + 3\mu\left(\frac{r_t + x}{r_c^5}\right)y \tag{3.29}$$

$$\ddot{y}_{\dot{x}} = \frac{\partial \ddot{y}}{\partial \dot{x}} = -2\dot{\theta} \tag{3.30}$$

$$\ddot{y}_y = \frac{\partial \ddot{y}}{\partial y} = \frac{\mu}{r_c^3}\left[3\left(\frac{y}{r_c}\right)^2 - 1\right] + \dot{\theta}^2 \tag{3.31}$$

$$\ddot{y}_{\dot{r}_t} = \frac{\partial \ddot{y}}{\partial \dot{r}_t} = 2\frac{\dot{\theta}}{r_t}x \tag{3.32}$$

$$\ddot{y}_z = \frac{\partial \ddot{y}}{\partial z} = 3\mu\left(\frac{y}{r_c^5}\right)z \tag{3.33}$$

$$\ddot{y}_{\dot{\theta}} = \frac{\partial \ddot{y}}{\partial \dot{\theta}} = 2\left(\dot{\theta}y - \dot{x} + \frac{\dot{r}_t}{r_t}x\right) \tag{3.34}$$

$$\ddot{y}_{r_t} = \frac{\partial \ddot{y}}{\partial r_t} = 3\mu\left(\frac{r_t + x}{r_c^5}\right)y - 2\dot{\theta}\frac{\dot{r}_t}{r_t^2}x \tag{3.35}$$

The cross-track position derivatives are:

$$\ddot{z}_x = \frac{\partial \ddot{z}}{\partial x} = 3\mu \left(\frac{r_t + x}{r_c^5} \right) z \tag{3.36}$$

$$\ddot{z}_y = \frac{\partial \ddot{z}}{\partial y} = 3\mu \left(\frac{y}{r_c^5} \right) z \tag{3.37}$$

$$\ddot{z}_z = \frac{\partial \ddot{z}}{\partial z} = \frac{\mu}{r_c^3} \left[3 \left(\frac{z}{r_c} \right)^2 - 1 \right] \tag{3.38}$$

$$\ddot{z}_{r_t} = \frac{\partial \ddot{z}}{\partial r_t} = 3\mu \left(\frac{r_t + x}{r_c^5} \right) z \tag{3.39}$$

and the true anomaly dynamics linearization gives:

$$\ddot{\theta}_{r_t} = \frac{\partial \ddot{\theta}}{\partial r_t} = 2\frac{\dot{r}_t}{r_t^2}\dot{\theta} \tag{3.40}$$

$$\ddot{\theta}_{\dot{\theta}} = \frac{\partial \ddot{\theta}}{\partial \dot{\theta}} = -2\frac{\dot{r}_t}{r_t} \tag{3.41}$$

$$\ddot{\theta}_{\dot{r}_t} = \frac{\partial \ddot{\theta}}{\partial \dot{r}_t} = -2\frac{\dot{\theta}}{r_t} \tag{3.42}$$

Lastly, the linearized target radius components are:

$$\ddot{r}_{t_{r_t}} = \frac{\partial \ddot{r}_t}{\partial r_t} = \dot{\theta}^2 + 2\frac{\mu}{r_t^3} \tag{3.43}$$

$$\ddot{r}_{t_{\dot{\theta}}} = \frac{\partial \ddot{r}_t}{\partial \dot{\theta}} = 2\dot{\theta} r_t \tag{3.44}$$

All components of the states and their respective dynamical equations shown above take their usually definitions from the formation flying relative motion descriptions described in Section 2.6. Note that the gravitational parameter μ used here corresponds to the gravitational parameter of the Earth, so could in fact equivalently be replaced with $\mu_\oplus$.

3.4 Measurement Modelling

The measurement model selected for demonstrating the navigation routines relies on direct measurements of the relative position and velocity states of the formation, and a measurement of the absolute position of the target spacecraft through the true anomaly. The measurement vector $\mathbf{h}(\mathbf{x}) \in \mathbb{R}^7$ therefore consists of $m = 7$ measurements, and is defined to be

$$\mathbf{h}(\mathbf{x}) \triangleq \begin{bmatrix} x_m & y_m & z_m & \theta_m & \dot{x}_m & \dot{y}_m & \dot{z}_m \end{bmatrix}^T \tag{3.45}$$

where the subscript m is used here to indicate a measured quantity. The measured relative positions $\{x_m, y_m, z_m\}$, relative velocities $\{\dot{x}_m, \dot{y}_m, \dot{z}_m\}$, and target true anomaly θ_m are calculated from the output of the real-world orbit propagator.

As shown on the next page in Figure 3.2, the inertial target position $\mathbf{r}_t$ and velocity $\mathbf{v}_t$, along with the inertial chaser position $\mathbf{r}_c$ and velocity $\mathbf{v}_c$, are first individually corrupted with zero-mean, white Gaussian noises via

$$\mathbf{r}_t^m = \mathbf{r}_t + \mathcal{N}(\mathbf{0}, \boldsymbol{\sigma}_r) \qquad \mathbf{r}_c^m = \mathbf{r}_c + \mathcal{N}(\mathbf{0}, \boldsymbol{\sigma}_r) \tag{3.46}$$

$$\mathbf{v}_t^m = \mathbf{v}_t + \mathcal{N}(\mathbf{0}, \boldsymbol{\sigma}_v) \qquad \mathbf{v}_c^m = \mathbf{v}_c + \mathcal{N}(\mathbf{0}, \boldsymbol{\sigma}_v) \tag{3.47}$$

Note the m term used to denote the measurements has been moved to a superscript, to avoid the abusive use of subscripts. For all scenarios tested in this research, the standard deviation of the position and velocity noises, $\boldsymbol{\sigma}_r \in \mathbb{R}^3$ and $\boldsymbol{\sigma}_v \in \mathbb{R}^3$ respectively, are treated as equal and spatially-independent noise processes that corrupt each component of the measurement. Writing these explicitly, gives

$$\boldsymbol{\sigma}_r = \begin{bmatrix} \sigma_r \\ \sigma_r \\ \sigma_r \end{bmatrix} \qquad \boldsymbol{\sigma}_v = \begin{bmatrix} \sigma_v \\ \sigma_v \\ \sigma_v \end{bmatrix} \tag{3.48}$$

Next, these noisy states are used to calculate the ECI-to-LVLH rotation matrix using the direction cosines previously given in Eq. (2.1), and the noise-corrupted relative position and velocity components are transformed into the LVLH frame using Eqs. (2.29) through (2.32) as was done in the truth model. The resulting LVLH components, with the embedded random noise, are then passed to the EKF through the measurement vector given above in Eq. (3.45). Performing a linearization on the measurement model gives the appropriate Jacobian $\mathbf{H} \in \mathbb{R}^{m \times n}$ as

$$\mathbf{H} = \frac{\partial \mathbf{h}}{\partial \mathbf{x}} = \begin{bmatrix} \mathbf{I}_{4\times4} & \mathbf{0}_{4\times1} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times2} \\ \mathbf{0}_{3\times4} & \mathbf{0}_{3\times1} & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times2} \end{bmatrix} \tag{3.49}$$

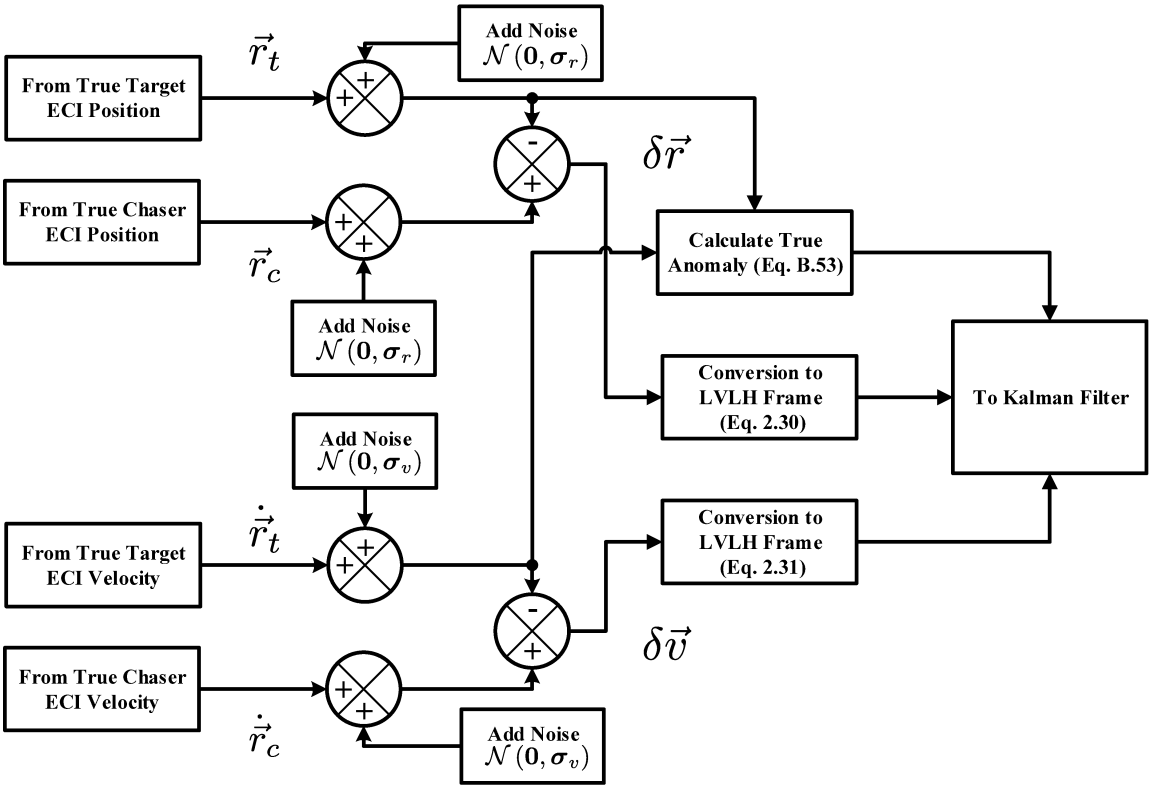


Figure 3.2: A block diagram overview of the measurement simulation process. After infusing the inertial states of the target and chaser spacecraft with noise, the relative ECI states are converted to the LVLH frame and passed to the EKF as measurements.

An example of the resulting measurements that are passed to the EKF are shown in Figure 3.3. Due to the frame transformation and the nonlinear relations between the ECI inertial states and the LVLH states, the imposed noises in the LVLH frame are no longer purely white Gaussian noise; thus, the resulting trajectory no longer matches the truth model.

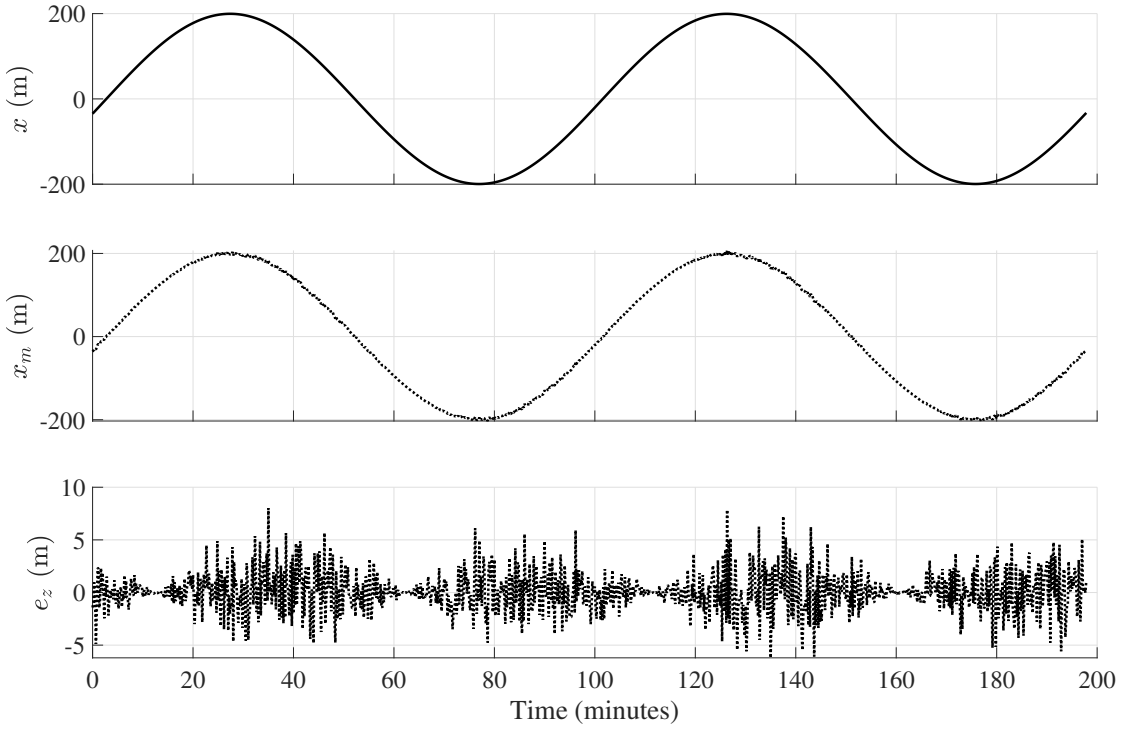


Figure 3.3: An example of LVLH radial relative position components for the nominal PRISMA formation, corrupted with noise. The true state is shown by x, the noisy measurements by x_m, and the error by e_x.

3.5 Observability Analysis

Before proceeding further, the concept of observability will briefly be discussed as it relates to the Kalman filter. For the general time-varying nonlinear system described by a state vector $\mathbf{x} \in \mathbb{R}^n$ and an output vector $\mathbf{y}(t) \in \mathbb{R}^p$, the system is said to be locally observable over the interval $t \in [0, T]$ if the mapping from the initial state $\mathbf{x}_0$ to the output $\mathbf{y}(t)$ is one-to-one [101]. Stated differently, local observability implies the initial state x_0 can be reconstructed from knowledge contained in the outputs over a given time interval. This notion of local observability was established by Bartosiewicz in 1995 [102]. Like the observability conditions for linear systems, a check for nonlinear observability involves determining if the observability matrix $\mathcal{O}(\mathbf{x})$ is full rank, such that all its columns are independent, and equivalently

$$\text{rank } \mathcal{O}(\mathbf{x}) = n \tag{3.50}$$

Now, whereas the linear observability test matrix is simple to construct, the nonlinear equivalent relies on manifold calculus and the use of Lie derivatives. The details of these topics in differential geometry are beyond the scope of this thesis, but suffice it to say that the first-order Lie derivative of the output $\mathbf{h}(\mathbf{x})$ along a vector field $\mathbf{f}(\mathbf{x})$ is denoted by $\mathcal{L}_f\mathbf{h}(\mathbf{x}) \in \mathbb{R}^p$ and is expressed as

$$\mathcal{L}_f\mathbf{h}(\mathbf{x}) \triangleq \frac{\partial \mathbf{h}(\mathbf{x})}{\partial \mathbf{x}}\mathbf{f}(\mathbf{x}) = \nabla\mathbf{h}(\mathbf{x})\mathbf{f}(\mathbf{x}) \tag{3.51}$$

where $\nabla(\cdot)$ represents the vector gradient. A recursive scheme for evaluating the next i^{th}-order Lie derivatives is then found through induction to be given by

$$\mathcal{L}_f^i\mathbf{h}(\mathbf{x}) \triangleq \frac{\partial \mathcal{L}_f^{i-1}\mathbf{h}(\mathbf{x})}{\partial \mathbf{x}}\mathbf{f}(\mathbf{x}) = \nabla\mathcal{L}_f^{i-1}\mathbf{h}(\mathbf{x})\mathbf{f}(\mathbf{x}); \quad \forall i > 0 \tag{3.52}$$

and the zeroth-order Lie derivative is simply the measurement matrix $\mathbf{h}(\mathbf{x})$, so

$$\mathcal{L}_f^0\mathbf{h}(\mathbf{x}) = \mathbf{h}(\mathbf{x}) \tag{3.53}$$

The observability matrix $\mathcal{O}(\mathbf{x}) \in \mathbb{R}^{pn \times n}$ for the nonlinear system is then defined as

$$\mathcal{O}(\mathbf{x}) \triangleq \frac{\partial}{\partial \mathbf{x}} \begin{bmatrix} \mathcal{L}_f^0\mathbf{h}(\mathbf{x}) \\ \mathcal{L}_f^1\mathbf{h}(\mathbf{x}) \\ \vdots \\ \mathcal{L}_f^{n-1}\mathbf{h}(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} \nabla\mathcal{L}_f^0\mathbf{h}(\mathbf{x}) \\ \nabla\mathcal{L}_f^1\mathbf{h}(\mathbf{x}) \\ \vdots \\ \nabla\mathcal{L}_f^{n-1}\mathbf{h}(\mathbf{x}) \end{bmatrix} \tag{3.54}$$

The size of this nonlinear observability matrix clearly depends on the number of states and measurements in the system, so the computation of the higher-order derivatives can become quite intensive. As such, Butcher and Wang [101] highlight that a sufficient (but not necessary) condition for observability can be obtained by only constructing the first N rows of $\mathcal{O}(\mathbf{x})$, or by constructing only a subsequent number of rows necessary to show that the truncated version of $\mathcal{O}(\mathbf{x})$ is full rank. Further numerical metrics for quantifying observability, including the gramian, the observability index, and the estimation condition number, are also summarized by Butcher and Wang's paper, but for the work conducted for this thesis, the construction of the observability matrix was necessary to identify which states require measuring.

For brevity, only the first $N = 3$ Lie derivatives are presented in the observability matrix $\mathcal{O}_N(\mathbf{x})$ below, demonstrating the full rank condition *so long as θ is measured*. A thorough development of the observability matrix and the corresponding components is given in Appendix D, along with the definitions of the observability matrix terms $\alpha_{k,j}$ for $k \in 1, \ldots, 7$ and $j \in 1, \ldots, 10$.

$$\mathcal{O}_N(\mathbf{x}) = \left[\begin{array}{cccccccccc}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
\hdashline
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
\ddot{x}_x & \ddot{x}_y & \ddot{x}_z & 0 & \ddot{x}_{r_t} & 0 & \ddot{x}_{\dot{y}} & 0 & \ddot{x}_{\dot{\theta}} & \ddot{x}_{\dot{r}_t} \\
\ddot{y}_x & \ddot{y}_y & \ddot{y}_z & 0 & \ddot{y}_{r_t} & \ddot{y}_{\dot{x}} & 0 & 0 & \ddot{y}_{\dot{\theta}} & \ddot{y}_{\dot{r}_t} \\
\dddot{z}_x & \dddot{z}_y & \dddot{z}_z & 0 & \dddot{z}_{r_t} & 0 & 0 & 0 & 0 & 0 \\
\hdashline
\ddot{x}_x & \ddot{x}_y & \ddot{x}_z & 0 & \ddot{x}_{r_t} & 0 & \ddot{x}_{\dot{y}} & 0 & \ddot{x}_{\dot{\theta}} & \ddot{x}_{\dot{r}_t} \\
\ddot{y}_x & \ddot{y}_y & \ddot{y}_z & 0 & \ddot{y}_{r_t} & \ddot{y}_{\dot{x}} & 0 & 0 & \ddot{y}_{\dot{\theta}} & \ddot{y}_{\dot{r}_t} \\
\dddot{z}_x & \dddot{z}_y & \dddot{z}_z & 0 & \dddot{z}_{r_t} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \ddot{\theta}_{r_t} & 0 & 0 & 0 & \ddot{\theta}_{\dot{\theta}} & \ddot{\theta}_{\dot{r}_t} \\
\alpha_{51} & \alpha_{52} & \alpha_{53} & 0 & \alpha_{55} & \alpha_{56} & \alpha_{57} & \ddot{x}_z & \alpha_{59} & \alpha_{5,10} \\
\alpha_{61} & \alpha_{62} & \alpha_{63} & 0 & \alpha_{65} & \alpha_{66} & \alpha_{67} & \ddot{y}_z & \alpha_{69} & \alpha_{6,10} \\
\alpha_{71} & \alpha_{72} & \alpha_{73} & 0 & \alpha_{75} & \dddot{z}_x & \dddot{z}_y & \dddot{z}_z & 0 & \dddot{z}_{r_t}
\end{array}\right] \tag{3.55}$$

3.6 Numerical Considerations

The EKF equations presented in the previous section are implemented in MATLAB functions within the Simulink environment. Several routines are needed to complete numerical operations within the EKF, specifically for the filter propagation scheme, and alternate matrix inversion formulations are available within the software to mitigate issues from round-off errors and matrix conditioning.

3.6.1 Integration within the EKF

From the definition of the EKF, we have the updated *a priori* state estimates that are obtained via Eq. (3.6), which is rewritten here for convenience:

$$\hat{\mathbf{x}}_k^- = \hat{\mathbf{x}}_{k-1} + \int_{t_{k-1}}^{t_k} \mathbf{f}(\hat{\mathbf{x}}_{k-1}, t)\ \mathrm{d}t$$

and the relative motion dynamics within Eqs. (2.36)-(2.40) give the state equation for the system as $\mathbf{f}(\mathbf{x}, t)$. Applying the discrete-time Runge-Kutta-Merson (RKM) method [103] to the EKF case, the differential equations described by $\mathbf{f}(\mathbf{x}, t)$ are used to propagate the state through a time-step $\Delta t = t_k - t_{k-1}$ as follows:

$$\hat{\mathbf{x}}_k^- = \hat{\mathbf{x}}_{k-1} + \frac{\Delta t}{6}\left(\mathbf{k}_1 + 4\mathbf{k}_4 + \mathbf{k}_5\right) \tag{3.56}$$

where the coupling coefficients are calculated by

$$\begin{aligned}
\mathbf{k}_1 &= \mathbf{f}(\hat{\mathbf{x}}_{k-1}) \\
\mathbf{k}_2 &= \mathbf{f}(\hat{\mathbf{x}}_{k-1} + \frac{1}{3}\mathbf{k}_1\Delta t) \\
\mathbf{k}_3 &= \mathbf{f}(\hat{\mathbf{x}}_{k-1} + \frac{1}{6}\mathbf{k}_1\Delta t + \frac{1}{6}\mathbf{k}_2\Delta t) \\
\mathbf{k}_4 &= \mathbf{f}(\hat{\mathbf{x}}_{k-1} + \frac{1}{8}\mathbf{k}_1\Delta t + \frac{3}{8}\mathbf{k}_3\Delta t) \\
\mathbf{k}_5 &= \mathbf{f}(\hat{\mathbf{x}}_{k-1} + \frac{1}{2}\mathbf{k}_1\Delta t - \frac{3}{2}\mathbf{k}_3\Delta t + 2\mathbf{k}_4\Delta t)
\end{aligned}$$

This 5-stage, 4th-order Runge-Kutta formulation was compared with several other forms, and was found to provide superior accuracy without unnecessary processing overhead. A detailed analyses and description of other integration methods onboard spacecraft is given in Montenbruck's state interpolation review [104], and support for the RKM technique on PROBA is mentioned in de Lafontaine *et al.* [105].

3.6.2 Matrix Inversion Using Equation Solving

In attempting to deal with several numerical issues that arose during the design and testing of the proposed EKFs, several alternate inversion algorithms were coded and can be used within the correction phase of the filter. These methods for handling equations with matrix inversions are discussed here as a potentially interesting side-note, but are not of particular importance in the analysis of the navigation routines.

Regarding Eq. (3.13), recall that the Kalman gain calculation requires the inversion of the residuals covariance matrix $\mathbf{\Sigma}_k$. The classical MATLAB inverse using `inv()` is notoriously slow and can potentially be unstable, so standard inversions of $\mathbf{\Sigma}_k$ within the EKF are implemented as

$$\mathbf{\Sigma}_k^{-1} = \mathbf{\Sigma}_k \setminus \mathbf{I} \tag{3.57}$$

where $\mathbf{I}$ is the identity matrix of the same dimension as $\mathbf{\Sigma}_k$, and '\' is the left matrix division operator, which can also be called using the function `mldivide()`. The matrix `inv()` function uses an LDL-decomposition method to explicitly solve the inverse of the matrix in question, which tends to be more computationally intensive and is further restrictive to square matrices. The `mldivide()` function on the other hand, performs a QR-decomposition to simply solve the system of equations proposed in the function call. As such, using left matrix division was found to be the most computationally efficient method for evaluating the covariance matrix inverse.

3.6.3 Matrix Inversion Using Cholesky Decomposition

The second option for inverting the residuals covariance takes advantage of the Positive Semi-Definite Symmetric (PSDS) nature of the matrix itself. Referencing the method proposed by Krishnamoorthy *et al.* [106], first a Cholesky factorization is used to obtain the upper triangular matrix $\mathbf{M}_{up}$, followed by a left matrix division discussed above to obtain the inverted triangular matrix $\mathbf{M}_{up}^{-}$. The calculations for this technique can be written as

$$\mathbf{M}_{up} = \texttt{chol}(\mathbf{\Sigma}_k) \tag{3.58}$$

$$\mathbf{M}_{up}^{-} = \mathbf{M}_{up} \setminus \mathbf{I} \tag{3.59}$$

$$\mathbf{\Sigma}_k^{-1} = \mathbf{M}_{up}^{-}\mathbf{M}_{up}^{-} \tag{3.60}$$

This method was found to improve the stability of the matrix inversions required within the EKF (over the `inv()` function), but was slightly more computationally expensive than the MATLAB matrix left division presented above. As such, this matrix inversion is left as an optional setting within the EKF program.

3.7 Overview of the Simulation Environment

Numerical simulations of the EKF are completed using the MATLAB-Simulink software suite, and Figure 3.4 provides an overview of the complete spacecraft navigation simulator that was developed for this research. The simulator consists of the following: the real-world orbit propagator used to simulate the true motion of the formation, the measurement simulator used to add noise to the true states, and the extended Kalman filter used to estimate the states of the formation. By comparing the state estimates from the EKF with the true formation states from the real-world orbit propagator, the performance of the EKF can be quantified through the *estimation error*.

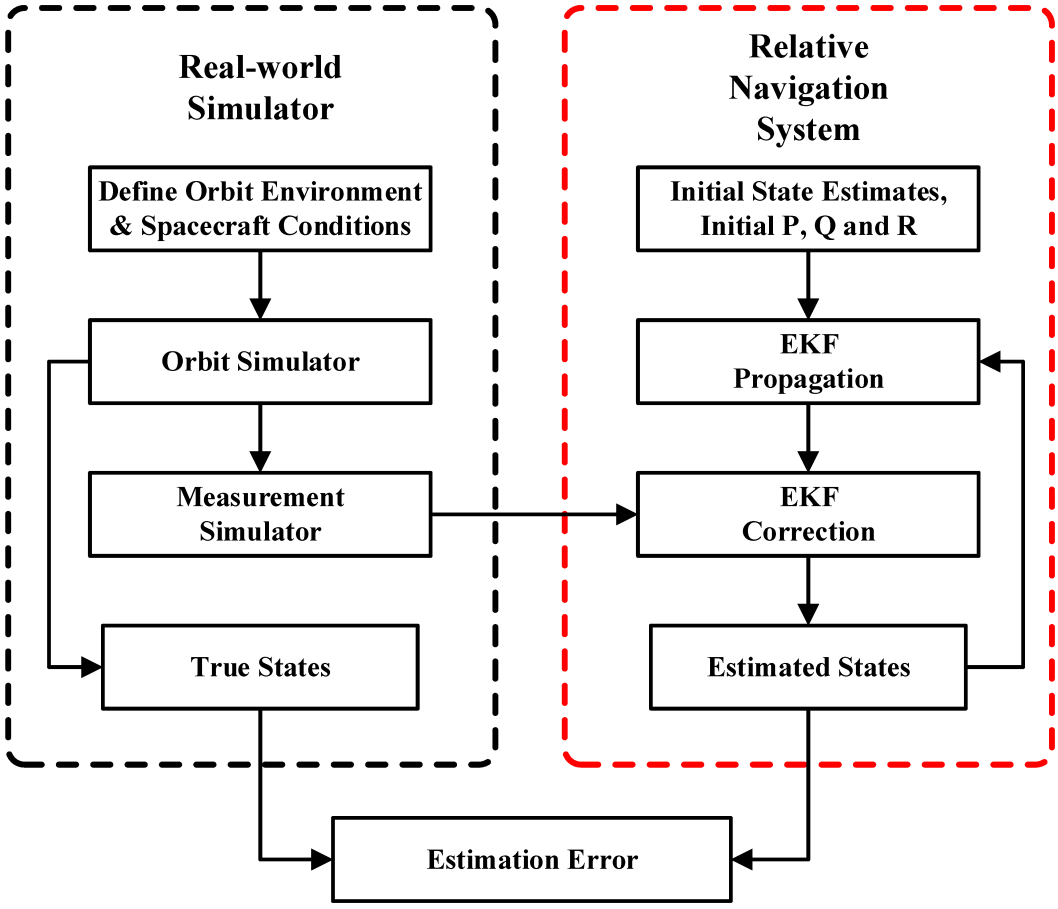


Figure 3.4: An overview of the formation flying relative navigation simulator developed for this thesis. Since the true states of the system are known exactly, the estimation error provides an explicit metric to determine if the EKF is providing accurate state estimates.

To illustrate that the simulated real-world models and the onboard EKF models are indeed different, Table 3.1 summarizes the dynamics models used for this research. The discrepancies between the filter and the real-world dynamics models introduce errors into the EKF estimates analogous to the errors that would be experienced due to modelling deficiencies within an on-orbit spacecraft navigation system. Thus, despite not having access to flight hardware, the simulated navigation system will be susceptible to realistic imperfections in the filter design.

Table 3.1: Summary of the Dynamics Models

Dynamic Force Source	**Truth Model**	**Filter Model**
Earth Gravity	Two-body + J_2	Two-body Only
Third-body Gravity	Analytical Model	-
Atmospheric Drag	Harris-Priester Model	-
Solar Radiation Pressure	Cannonball Model	-

Similarly for the measurements and measurement models, Table 3.2 outlines the differences between the sensor measurements and the measurement models used within the EKF algorithms. The addition of noise into the measurements ensures that the estimated measurements within the filter do not equal the true states of the formation. Further modifications to the measurement models could be made to account for specific sensors and measurement types, however the focus of this work considers only a generic, noise-corrupted measurement to demonstrate the EKF techniques.

Table 3.2: Summary of the Measurement Models

Measurement Components	**Truth Model**	**Filter Model**
Relative Position	ECI Position + Noise, Transformed to LVLH	Direct Measurement
Relative Velocity	ECI Velocity + Noise, Transformed to LVLH	Direct Measurement
Target True Anomaly	Calculated using Noisy ECI States	Direct Measurement

3.8 Chapter Summary

This chapter presented the theory behind the extended Kalman filter, a recursive state estimation technique suitable for nonlinear systems. The dynamics of relative spacecraft motion, a set of five second-order differential equations, were formulated into the EKF framework along with the required linearized model of the dynamics. The measurements of the spacecraft formation provided to the EKF were identified, and the onboard measurement model was similarly derived by linearizing the measurement model. A proof of observability was given using Lie Derivatives and the nonlinear observability matrix, and several remarks were given regarding numerical implementation issues within the filter. Having established an EKF capable of estimating the relative states of a spacecraft formation, the next stage of this research addresses the design of an adaptive EKF using Maximum Likelihood Estimation.

Chapter 4

Adaptive Kalman Filtering using Maximum Likelihood Estimation

The previous chapter approached the navigation problem using a standard extended Kalman filter, assuming that knowledge of the process and measurement noise sources were sufficiently well-known, and a manual tuning process had yielded reasonably accurate definitions of the process and measurement noise covariance matrices within the filter. This chapter expands upon the EKF, to incorporate an adaptation mechanism capable of updating the noise covariance matrices within the filter, using maximum likelihood estimation. An overview of the statistical theory of MLE is presented here, after which analytic adaptation laws are derived for the EKF. Inclusion of an intrinsic smoothing scheme within the MLE algorithm is also investigated, as an improvement to MLE methods found previously in literature.

4.1 Introduction

While a linear Kalman filter with perfect knowledge of a linear system will indeed be the optimal state estimator in terms of minimum error variance, the majority of real-world systems are nonlinear and violate many of the assumptions made within the Kalman filter derivation. Nevertheless, the EKF is renowned for its ability to perform reasonably well even when many of the founding assumptions are no longer valid, however there is certainly merit in determining if the filter performance can be improved upon under non-ideal conditions. Internal characteristics of the Kalman filter, including the initial conditions, dynamical constants, process noise covariances, and measurement noise covariances, all represent *parameters* that influence the performance of the EKF, which are chosen at the discretion of the filter designer [21]. A well-known

method for *parameter estimation* in system identification problems is the method of maximum likelihood estimation, and using MLE within the EKF was proposed in the 1970's by Mehra [38, 39].

At the highest level, the objective of the MLE method is to determine the set of parameters $\boldsymbol{\Theta}$ that maximize a *likelihood function* $L\left(\boldsymbol{\Theta}|\mathbf{Z}_N\right)$, based on a set of observed measurements $\mathbf{Z}_N$ that are influenced by the parameters. When the likelihood function is continuously differentiable with respect to the parameters, the solution of the MLE problem can be obtained by solving the likelihood equation via:

$$\frac{\partial L\left(\boldsymbol{\Theta}|\mathbf{Z}_N\right)}{\partial \boldsymbol{\Theta}} = \mathbf{0}^T \tag{4.1}$$

While this simple form likely gives the impression that the MLE solutions should be relatively straightforward, only a limited number of likelihood functions $L\left(\boldsymbol{\Theta}|\mathbf{Z}_N\right)$ have actually been defined for practical use; a discussion on the various candidate likelihood functions is given by Maybeck [41]. Throughout the remaining sections of this chapter, an appropriate likelihood function will be identified and used to find analytic solutions for Eq. (4.1) in the context of the EKF.

4.2 Statistical Preliminaries

Before proceeding to find the MLE-based adaptation laws developed for this work, a brief review of pertinent statistical theory is provided here based on the text by Leon-Garcia [107]. As introduced previously, the MLE technique is a parameter estimation routine that seeks to maximize the likelihood function $L\left(\boldsymbol{\Theta}|\mathbf{Z}_N\right)$ for a general set of system parameters $\boldsymbol{\Theta} \in \mathbb{R}^n$, where n indicates the number of parameters to be estimated. To be slightly more proper in the terminology, the observed data $\mathbf{Z}_N \in \mathbb{R}^{m\times N}$ is a set of N random samples of a discrete random variable Z, which are defined by

$$\mathbf{Z}_N = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_N\} \tag{4.2}$$

and it is assumed that the measurement vectors $\mathbf{z}_i \in \mathbb{R}^m \; \forall i = 1, \ldots, N$ are independent. In the case of the Kalman filter scenario, these measurement vectors simply correspond to sets of measurements collected at sequential instances in time. Furthermore, the observations are identically distributed with a *Probability Mass Function* (PMF) denoted by $p_Z(\mathbf{z}_k)$, defined for the discrete random variable Z as

$$p_Z(\mathbf{z}_k) \triangleq P\left(Z = \mathbf{z}_k\right) \tag{4.3}$$

The PMF is an indicator of the probability that the discrete random variable is equal to a particular value exactly. Many types of PMFs exist and correspond with the various types of random variables, such as the Bernoulli, the binomial, the geometric, the Poisson, and the Gaussian. A general parameter estimator is then defined as a function of the observed data, in the form of

$$\hat{\mathbf{\Theta}}\left(\mathbf{Z}_N\right) = g\left(\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_N\right) \tag{4.4}$$

where $\hat{\mathbf{\Theta}} \in \mathbb{R}^n$ contains the estimates of the true parameters $\boldsymbol{\theta}$, which are in turn random variables. The likelihood function of the observed data is a function of the parameters, given by

$$L\left(\mathbf{\Theta}|\mathbf{Z}_N\right) = L\left(\mathbf{\Theta}|\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_N\right) \tag{4.5}$$

$$= p_Z\left(\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_N|\mathbf{\Theta}\right) \tag{4.6}$$

where $p_Z\left(\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_N|\mathbf{\Theta}\right)$ is the joint PMF evaluated at the observations, conditioned on the parameter value $\mathbf{\Theta}$. Intuitively, the likelihood function describes the probability that the available data set $\mathbf{Z}_N$ could be obtained using the given parameters $\mathbf{\Theta}$. If the sampled data sets are Independent and Identically Distributed (IID), the joint PMF can be simplified dramatically by applying Bayes Rule [108]:

$$p_Z\left(\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_N|\mathbf{\Theta}\right) = \Big[p_Z(\mathbf{z}_1|\mathbf{\Theta})\Big]\Big[p_Z(\mathbf{z}_2|\mathbf{\Theta})\Big]\ldots\Big[p_Z(\mathbf{z}_N|\mathbf{\Theta})\Big] \tag{4.7}$$

$$= \prod_{i=1}^{N} p_Z(\mathbf{z}_i|\mathbf{\Theta}) \tag{4.8}$$

Thus, once a suitable PMF for the EKF is inserted into Eq. (4.8), all that remains is to evaluate the partial derivative of the resulting likelihood function, and determine the parameter set that maximizes the likelihood function.

4.3 The Likelihood Function of the EKF

The development of the MLE algorithm for this research follows closely with various works [20, 21, 41], with the goal of adapting the process and measurement noise covariances $\mathbf{Q}_k$ and $\mathbf{R}_k$, respectively. Therefore, the set of parameters to be estimated is defined as

$$\mathbf{\Theta} \triangleq \{\mathbf{Q}_k, \mathbf{R}_k\} \tag{4.9}$$

As introduced previously, the MLE technique can be treated as an optimization routine that maximizes the likelihood function $L(\boldsymbol{\Theta}|\mathbf{Z}_N)$ for a general set of system parameters. Taking the log-likelihood form of the problem is a useful way to simplify the mathematics, thanks to the monotonicity of the logarithm function, allowing the MLE problem to be written equivalently with

$$\max_{\boldsymbol{\Theta}} \left\{ \ln L(\boldsymbol{\Theta}|\mathbf{Z}_N) = \ln p(\mathbf{Z}_N|\boldsymbol{\Theta}) \right\} \tag{4.10}$$

The term $p(\mathbf{Z}_N|\boldsymbol{\Theta})$ represents the joint PMF for observations of the EKF, conditioned on the parameters. The observations considered here are in fact the residuals of the EKF, because the innovations and the measurements contain the same statistical information. Calling to mind the innovation property of an optimal filter, as presented by Kailath [39, 99], the residuals $\boldsymbol{\nu}_k$ of the EKF are assumed to be distributed as zero-mean, white Gaussian noise with covariance $\boldsymbol{\Sigma}_k$, such that

$$p_{\mathcal{V}}(\boldsymbol{\nu}_k|\boldsymbol{\Theta}) = \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_k) \tag{4.11}$$

Since the innovation property pertains to the linear Kalman filter, a relaxation of the strict equality of the PMFs for the measurement and innovations is implicit. By inserting the PMF of the innovations into Eq. (4.8), the probability of the measurements is given in terms of the product of PMFs of the innovations [109]:

$$p_Z(\mathbf{Z}_N|\boldsymbol{\Theta}) \approx \prod_{i=i_0}^{N} p_{\mathcal{V}}(\boldsymbol{\nu}_N|\boldsymbol{\Theta}) \tag{4.12}$$

where the symbol $\prod(\cdot)$ represents the product operator. Another reason this PMF only *approximates* the full measurement PMF relates to the lower limit change of the product operator from 1 to i_0. Using the entire set of measurements accumulated by the filter (*i.e.*, $i = 1, 2, ..., k$) would be unrealistic for online estimation within the EKF, so instead only a fixed-set of past measurements is processed. The onboard likelihood function is therefore evaluated using a window of size N, which contains data from point $i_0 = k - N + 1$ to the current point k.

The necessary information required to construct the likelihood function is now available. Since the innovations process follows a Gaussian distribution, the likelihood function is written using the standard matrix probability function for Gaussian

distributions [110], which is

$$L\left(\boldsymbol{\Theta}|\mathbf{Z}_N\right) = \prod_{i=i_0}^{N} \frac{1}{\sqrt{(2\pi)^m |\boldsymbol{\Sigma}_i|}} \exp\left[-\frac{\boldsymbol{\nu}_i^T \boldsymbol{\Sigma}_i^{-1} \boldsymbol{\nu}_i}{2}\right] \tag{4.13}$$

The term $|\boldsymbol{\Sigma}_i|$ refers to the determinant of matrix $\boldsymbol{\Sigma}_i$, and, as a reminder, $m = 7$ is the number of measurements in each measurement vector. Taking the natural logarithm of the likelihood function eliminates the multiplicity, simplifying the result to

$$\ln L\left(\boldsymbol{\Theta}|\mathbf{Z}_N\right) = -\frac{1}{2} \sum_{i=i_0}^{N} \left[\ln|\boldsymbol{\Sigma}_i| + \boldsymbol{\nu}_i^T \boldsymbol{\Sigma}_i^{-1} \boldsymbol{\nu}_i + c_i\right] \tag{4.14}$$

where the constant term $c_i = m \ln(2\pi)$ is independent of the parameters. The goal of maximizing the log-likelihood function is equivalent to minimizing the negative log-likelihood function, so a pseudo-likelihood cost function is defined by negating Eq. (4.14) and neglecting the constant term and coefficient:

$$J\left(\boldsymbol{\Theta}|\mathbf{Z}_N\right) \triangleq \sum_{i=i_0}^{N} \left[\ln|\boldsymbol{\Sigma}_i| + \boldsymbol{\nu}_i^T \boldsymbol{\Sigma}_i^{-1} \boldsymbol{\nu}_i\right] \tag{4.15}$$

Through this process, the MLE solution $\hat{\boldsymbol{\Theta}}_{MLE}$ for the EKF parameters reduces to minimizing the pseudo-likelihood cost function $J\left(\boldsymbol{\Theta}|\mathbf{Z}_N\right)$, as summarized in the optimization problem below. Imposing positive definite constraints upon the $\mathbf{Q}_k$ and $\mathbf{R}_k$ matrices as per their definition, the MLE solution is:

$$\begin{aligned} \hat{\boldsymbol{\Theta}}_{MLE} = \underset{\mathbf{Q}_k, \mathbf{R}_k}{\arg\min} \quad & \left[J\left(\boldsymbol{\Theta}|\mathbf{Z}_N\right)\right] \\ \text{subject to} \quad & \mathbf{Q}_k > 0 \\ & \mathbf{R}_k > 0 \end{aligned} \tag{4.16}$$

4.4 Solving the MLE Problem

Several definitions and assumptions are required before advancing with the general derivation of the MLE adaptations. First, recall that the residuals are purely a function of the measurements and the measurement model, while the filter residual covariance matrix included contributions from both the process and measurement noise covariances. This indicates that the covariance of the residuals is key to the

MLE adaptation mechanism within the filter, and as a reminder can be written

$$\begin{aligned}\boldsymbol{\Sigma}_k &= \mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k && (4.17)\\ &= \mathbf{H}_k \left(\boldsymbol{\Phi}_{k-1} \mathbf{P}_{k-1}^+ \boldsymbol{\Phi}_{k-1}^T + \mathbf{Q}_k \right) \mathbf{H}_k^T + \mathbf{R}_k && (4.18)\end{aligned}$$

Secondly, general adaptation parameters at the current time step t_k are denoted by α_k; this variable is essentially a placeholder throughout the derivation, and can be swapped for elements of $\mathbf{Q}_k$ or $\mathbf{R}_k$ as appropriate. Following the approach used by Mohamed [20], the MLE solution to the minimization problem is obtained analytically by taking the derivative of Eq. (4.15) with respect to the adaptive parameter α_k, and setting the result to zero:

$$\frac{\partial}{\partial \alpha_k} \Big[J\left(\boldsymbol{\Theta} | \mathbf{Z}_N\right) \Big] = 0 \tag{4.19}$$

Several assumptions are required here in order to obtain a closed-form solution, and are stated explicitly below:

Assumption 1 - Adaptations Through the Covariance of the Residuals

Adaptations of the EKF are implemented through the covariance matrix of the residuals, meaning that the following holds for all time steps t_k:

$$\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}_k(\alpha_k) \qquad \frac{\partial \boldsymbol{\Sigma}_k(\alpha_k)}{\partial \alpha_k} \neq 0 \tag{4.20}$$

Assumption 2 - Independent State Estimates

Adaptations of the parameter α_k have no bearing on the current filter states, so throughout the adaptations:

$$\frac{\partial \mathbf{x}_k}{\partial \alpha_k} = 0 \tag{4.21}$$

Assumption 3 - Independent Filter Models

Both the state transition matrix and the measurement models at the current time step are independent of α_k, and both are also time invariant. This implies:

$$\frac{\partial \boldsymbol{\Phi}_k}{\partial \alpha_k} = 0 \qquad \frac{\partial \mathbf{H}_k}{\partial \alpha_k} = 0 \tag{4.22}$$

With these assumptions, the adaptation equation which maximizes the likelihood function for the adaptation parameter is

$$\sum_{i=i_0}^{N} \mathrm{Tr}\left\{\left[\boldsymbol{\Sigma}_k^{-1} - \boldsymbol{\Sigma}_k^{-1}\boldsymbol{\nu}_k\boldsymbol{\nu}_k^T\boldsymbol{\Sigma}_k^{-1}\right]\left[\frac{\partial \mathbf{R}_k}{\partial \alpha_k} + \mathbf{H}_k\frac{\partial \mathbf{Q}_k}{\partial \alpha_k}\mathbf{H}_k\right]\right\} = 0 \tag{4.23}$$

A full derivation of the partials in Eq. (4.19) can be found in Reference [41]. From Eq. (4.23) it is clear that adaptations of both $\mathbf{Q}_k$ and $\mathbf{R}_k$ are possible. This result stems from the fact that both the process and measurement noise covariances factor into the residual sequence, with $\mathbf{Q}_k$ through the *a priori* state error covariance and $\mathbf{R}_k$ through the residual covariance. Next, unique solutions that provide update laws for both noise covariance matrices will be investigated.

4.4.1 Adaptations for the Process Noise Covariance

The adaptation equation for the process noise covariance is found by solving Eq. (4.23) under the conditions that $\mathbf{R}_k$ is assumed known and independent of the adaptation parameter. This implies

$$\frac{\partial \mathbf{R}_k}{\partial \alpha_k} = 0 \tag{4.24}$$

for this portion of the derivation. Thus, each diagonal entry of the process noise covariance is treated as an adaptable term, so at each time step t_k the adaptation parameters $\alpha_n = \mathbf{Q}_{nn}$ correspond to the n^{th} row or column index of the process noise matrix. Equation (4.23) then simplifies to

$$\sum_{i=i_0}^{N} \mathrm{Tr}\left\{\mathbf{H}_i^T\left[\boldsymbol{\Sigma}_i^{-1} - \boldsymbol{\Sigma}_i^{-1}\boldsymbol{\nu}_i\boldsymbol{\nu}_i^T\boldsymbol{\Sigma}_i^{-1}\right]\mathbf{H}_i\right\} = 0 \tag{4.25}$$

Inserting Eq. (4.18) into the Kalman gain formulation from Eq. (3.13), leads to another expression for the Kalman gain in terms of the residual covariances:

$$\mathbf{K}_i = \mathbf{P}_i^-\mathbf{H}_i^T\boldsymbol{\Sigma}_i^{-1} \tag{4.26}$$

which after substituting into Eq. (4.25) and rearranging gives

$$\sum_{i=i_0}^{N} \mathrm{Tr}\left\{\left[\mathbf{P}_i^-\right]^{-1}\left[\mathbf{K}_i\mathbf{H}_i\mathbf{P}_i^- - \mathbf{K}_i\boldsymbol{\nu}_i\boldsymbol{\nu}_i^T\mathbf{K}_i^T\right]\left[\mathbf{P}_i^-\right]^{-1}\right\} = 0 \tag{4.27}$$

Identifying that any covariance terms will be positive definite symmetric by definition, $\mathbf{P}_i^-$ can be eliminated from Eq. (4.27), yielding

$$\sum_{i=i_0}^{N} \mathrm{Tr}\left\{\mathbf{K}_i\mathbf{H}_i\mathbf{P}_i^- - \mathbf{K}_i\boldsymbol{\nu}_i\boldsymbol{\nu}_i^T\mathbf{K}_i^T\right\} = 0 \tag{4.28}$$

To further reduce the working equation, the *a posteriori* error covariance can be simplified assuming that the Kalman gain is optimal [98], giving the form

$$\mathbf{P}_i^+ = \mathbf{P}_i^- - \mathbf{K}_i\mathbf{H}_i\mathbf{P}_i^- \tag{4.29}$$

and after rearranging and inserting Eq. (4.29) into Eq. (4.28) gives:

$$\sum_{i=i_0}^{N} \mathrm{Tr}\left\{\mathbf{P}_i^- - \mathbf{P}_i^+ - \mathbf{K}_i\boldsymbol{\nu}_i\boldsymbol{\nu}_i^T\mathbf{K}_i^T\right\} = 0 \tag{4.30}$$

The process noise covariance $\mathbf{Q}_k$ is introduced into the solution through the *a priori* state error covariance $\mathbf{P}_k^-$ from Eq. (3.9), which after substituting leads to:

$$\sum_{i=i_0}^{N} \mathrm{Tr}\left\{\mathbf{Q}_i + \boldsymbol{\Phi}_{i-1}\mathbf{P}_{i-1}^+\boldsymbol{\Phi}_{i-1}^T - \mathbf{P}_i^+ - \mathbf{K}_i\boldsymbol{\nu}_i\boldsymbol{\nu}_i^T\mathbf{K}_i^T\right\} = 0 \tag{4.31}$$

Finally, an estimate of the process noise covariance can then be obtained, denoted here by $\hat{\mathbf{Q}}_k$ and calculated with

$$\hat{\mathbf{Q}}_k = \frac{1}{N}\sum_{i=i_0}^{N}\left[\mathbf{K}_i\boldsymbol{\nu}_i\boldsymbol{\nu}_i^T\mathbf{K}_i^T + \left(\mathbf{P}_i^+ - \boldsymbol{\Phi}_{i-1}\mathbf{P}_{i-1}^+\boldsymbol{\Phi}_{i-1}^T\right)\right] \tag{4.32}$$

The parenthesized term in the above equation contains the change in the error covariances between the current and previous time steps. If the filter has reached steady-state operation, this term is negligible and the Kalman gain becomes constant. This is an attractive simplification in a computational sense, as less information from the previous filter steps need to be maintained in memory. The estimate of the process covariance then reduces to the form used by Mohamed and Schwarz [20]:

$$\hat{\mathbf{Q}}_k = \mathbf{K}_k\left[\frac{1}{N}\sum_{i=i_0}^{N}\boldsymbol{\nu}_i\boldsymbol{\nu}_i^T\right]\mathbf{K}_k^T \tag{4.33}$$

Notice the terms in brackets here contains the innovations of the filter averaged over the N-steps of the fixed window. This term is referred to as the *observed* innovations

covariance matrix, denoted $\hat{\mathbf{\Sigma}}_{k|N}$, which is defined by

$$\hat{\mathbf{\Sigma}}_{k|N} \triangleq \frac{1}{N} \sum_{i=i_0}^{N} \boldsymbol{\nu}_i \boldsymbol{\nu}_i^T \tag{4.34}$$

Using this notion of the observed innovations covariance matrix, the Q-adaptation equation can be reduced to

$$\hat{\mathbf{Q}}_k = \mathbf{K}_k \hat{\mathbf{\Sigma}}_{k|N} \mathbf{K}_k^T \tag{4.35}$$

From this result, the Q-adaptation law for the EKF requires only knowledge of the current Kalman gain $\mathbf{K}_k$, and the past N vectors of observed residuals from the filter. Another simplification to this equation could reduce the computational complexity further by eliminating the summation and considering only the latest set of residual $\boldsymbol{\nu}_k$, however the statistical averaging provided by storing the past N data points was found to provide better results. The value of N is purely a design choice, but can significantly change the performance of the MLE adaptations and the computational time of the simulations (*i.e.*, as N increases, so too do the computing costs of storing the additional data, and the relationship is not linear). The value of N will therefore always be identified prior to demonstrating the results of a simulation in this thesis.

4.4.2 Adaptations for the Measurement Noise Covariance

Solving for the measurement noise adaptations uses the same procedure as above, albeit with $\mathbf{Q}_k$ designated as the known and adaptation-independent parameter. Setting

$$\frac{\partial \mathbf{Q}_k}{\partial \alpha_k} = 0 \tag{4.36}$$

and choosing the adaptation terms as $\alpha_n = \mathbf{R}_{nn}$, the derivatives in Eq. (4.23) yield

$$\sum_{i=i_0}^{N} \mathrm{Tr} \left\{ \mathbf{\Sigma}_i^{-1} \left[\mathbf{\Sigma}_i - \boldsymbol{\nu}_i \boldsymbol{\nu}_i^T \right] \mathbf{\Sigma}_i^{-1} \right\} = 0 \tag{4.37}$$

which can also be expanded using the expression for the residual covariance from Eq. (4.18), giving

$$\sum_{i=i_0}^{N} \mathrm{Tr} \left\{ \mathbf{\Sigma}_i^{-1} \left[\mathbf{H}_i \mathbf{P}_i^{-} \mathbf{H}_i^T + \mathbf{R}_i - \boldsymbol{\nu}_i \boldsymbol{\nu}_i^T \right] \mathbf{\Sigma}_i^{-1} \right\} = 0 \tag{4.38}$$

Invariance of the estimation process is assumed for the given memory window, implying that $\mathbf{\Sigma}_i^{-1}$ is approximately constant throughout the filtered data in memory [111, 112].

This allows an explicit estimate of the measurement noise covariance to be calculated, which is similar but not identical to the results obtained by [20, 41]:

$$\hat{\mathbf{R}}_k = \frac{1}{N}\sum_{i=i_0}^{N}\left[\boldsymbol{\nu}_i\boldsymbol{\nu}_i^T - \mathbf{H}_i\mathbf{P}_i^-\mathbf{H}_i^T\right] \tag{4.39}$$

Although mathematically sound, implementation of Eq. (4.39) has been shown to give poor estimates of $\mathbf{R}_k$ that break the positive definite constraint. Instead, an alternate equation for the R-adaptation scheme can be derived by using several Kalman filter relations, and the necessary substitutions are shown in [20]. The desired formulation aligns with that of Maybeck [41], and has been re-derived using expectation maximization techniques [109, 113]. As such, the estimated measurement noise covariance is

$$\hat{\mathbf{R}}_k = \frac{1}{N}\sum_{i=i_0}^{N}\left[(\boldsymbol{\nu}_i^+)(\boldsymbol{\nu}_i^+)^T + \mathbf{H}_i\mathbf{P}_i^+\mathbf{H}_i^T\right] \tag{4.40}$$

Notice that in the equation above, the innovations have been replaced with the *post-fit residuals* defined as $\boldsymbol{\nu}_i^+$, which are given by

$$\boldsymbol{\nu}_i^+ \triangleq \mathbf{z}_i - \hat{\mathbf{z}}_i^+ \tag{4.41}$$

Post-fit residuals are calculated from corrected measurement estimates $\hat{\mathbf{z}}_i^+$, using Eq. (3.10) evaluated at the corrected state estimate $\hat{\mathbf{x}}_i^+$.

4.4.3 Maintaining Positive Definite Symmetry

As was mentioned in the introduction to Kalman filter terminology, the process and measurement noise covariances $\mathbf{Q}_k$ and $\mathbf{R}_k$ for any real system should be PDS matrices by definition. This means that in the case of MLE adaptations, it is crucial to ensure the estimated covariances $\hat{\mathbf{Q}}_k$ and $\hat{\mathbf{R}}_k$ do not break the PDS constraints. Formally, a symmetric $n \times n$ matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ is *positive definite* if for all non-zero vectors $\mathbf{x} \in \mathbb{R}^n$, the following holds:

$$V(x) = \mathbf{x}^T\mathbf{A}\mathbf{x} > 0 \tag{4.42}$$

where $V(x)$ is a scalar-valued function. If $\mathbf{A}$ is PDS, all of the eigenvalues are positive real numbers, and the inverse $\mathbf{A}^{-1}$ will also be positive definite. Furthermore, a matrix can only be positive definite is there exists a Cholesky Decomposition. The work of Higham [114] uses the Cholesky decomposition to address the problem of determining

Positive Semi-Definite Symmetric (PSDS) matrices, which simply expands the criteria of the PDS matrix to include $V(x) \geq 0$, and this method can be implemented within the MLE-AEKF to prevent the estimated noise covariances from becoming non-positive definite.

For a given square matrix $\mathbf{A}$ defined above, the nearest PSDS matrix can be determined using Higham's algorithm by first symmetricizing $\mathbf{A}$ and determining the polar factors $\mathbf{H}$ using singular-value decomposition (SVD):

$$\mathbf{B} = \frac{\mathbf{A} + \mathbf{A}^T}{2} \tag{4.43}$$

$$[\mathbf{U}, \mathbf{\Sigma}, \boldsymbol{V}] = \texttt{svd(B)} \tag{4.44}$$

$$\mathbf{H} = \mathbf{V}\mathbf{\Sigma}\mathbf{V}^T \tag{4.45}$$

Next, the positive approximant $\hat{\mathbf{A}}$ of matrix $\mathbf{A}$ is computed and ensured symmetric:

$$\hat{\mathbf{A}} = \frac{1}{2}\left[\left(\frac{\mathbf{B}+\mathbf{H}}{2}\right) + \left(\frac{\mathbf{B}+\mathbf{H}}{2}\right)^T\right] \tag{4.46}$$

The Cholesky decomposition of $\hat{\mathbf{A}}$ is then used to test for PSD, and an update to $\hat{\mathbf{A}}$ is made if the test fails. The iterative update procedure is based on the minimum eigenvalue of $\hat{\mathbf{A}}$, and each k^{th} iteration of the update routine proceeds as follows:

$$\lambda_{min} = \min \lambda(\hat{\mathbf{A}}) \tag{4.47}$$

$$\hat{\mathbf{A}} = \hat{\mathbf{A}} + \left(-\lambda_{min} k^2 + \mathbf{I}\epsilon(\lambda_{min})\right) \tag{4.48}$$

The number of iterations is restricted to $k \leq k_{max}$, where $k_{max} = 5$ is used in this work to limit the computational overhead of the procedure. Empirical testing of the algorithm demonstrated that convergence to the nearest PSDS matrix is consistently achieved within 3-4 iterations. The choice of using this algorithm within the filter rests with the user, but is not a strict requirement.

Remark 4.1 *In theory, the process noise must only be Positive Semi-Definite Symmetric, meaning that there could be eigenvalues of* $\mathbf{Q}_k$ *that are zero, but this assumes the dynamic model perfectly matches the real-world scenario and is not useful in practice. In all cases,* $\mathbf{R}_k$ *must be positive definite symmetric to ensure the inverse of the residuals covariance is always defined as the state error covariance goes to zero.*

4.5 The Extended Kalman Smoother

Typically, the EKF is applied as a recursive scheme dependent only on estimates from the previous filter step and measurements at the current time. However, incorporating a memory characteristic into the filter can improve estimates of the state and error covariances by harnessing information contained in the entire available data set. Since the adaptation laws presented above already require a fixed-length memory window, estimates from the EKF used in the MLE adaptations laws are processed using the extended Kalman smoother [109], a nonlinear variation of the smoother developed by Rauch *et al.* [115]. The smoother runs backwards through a window of size N, which contains the set of data points from $i_0 = k - N + 1$ to the latest point k. Initial smoothed estimates are taken from the filter at the current time step:

$$\hat{\mathbf{x}}_{N|N} = \hat{\mathbf{x}}_k^+ \qquad\qquad \mathbf{P}_{N|N} = \mathbf{P}_k^+$$

where the term $\hat{\mathbf{x}}_{k|N}$ reads as "the smoothed state estimate at time k, given N data points". Using these initial conditions, the smoothed state and error covariances for the data within the memory window are calculated using

$$\mathbf{G}_{k-1} = \mathbf{P}_{k-1}^+ \mathbf{\Phi}_{k-1}^T \left(\mathbf{P}_k^-\right)^{-1} \tag{4.49}$$

$$\hat{\mathbf{x}}_{k-1|N} = \hat{\mathbf{x}}_{k-1}^+ + \mathbf{G}_{k-1}\left(\hat{\mathbf{x}}_{k|N} - \hat{\mathbf{x}}_k^-\right) \tag{4.50}$$

$$\mathbf{P}_{k-1|N} = \mathbf{P}_{k-1}^+ + \mathbf{G}_{k-1}\left(\mathbf{P}_{k|N} - \mathbf{P}_k^-\right)\mathbf{G}_{k-1}^T \tag{4.51}$$

$$\forall k = N-1, N-2, \ldots, i_0$$

where $\mathbf{G}_k \in \mathbb{R}^{n\times m}$ is a smoothing gain calculated as the MLE estimator for the current state estimates given the entire set of measurements available. For additional information on the smoothing algorithm, the reader is directed to reference [115].

The purpose of the EKF Smoother in the context of this work, is to develop smoothed data within the MLE adaptation scheme that will improve the estimates of the $\mathbf{Q}_k$ or $\mathbf{R}_k$ matrices. The methodology for including the smoother within the MLE adaptation scheme came about after investigating Expectation Maximization techniques in Kalman filtering [111], where significantly more processing power is required.

For the Q-adaptation equation, the estimate of $\mathbf{Q}_k$ can be improved by using the smoothed state estimates in the calculation of the residuals. Constructing a smoothed estimate of the measurements via

$$\hat{\mathbf{z}}_{i|N} = \mathbf{H}_{k|N}\hat{\mathbf{x}}_{k|N} \tag{4.52}$$

where the smoothed linearized measurement matrix $\mathbf{H}_{k|N}$ is found similar to that in Eq. (3.12), the final process noise covariance adaptation can be updated from Eq. (4.33) as

$$\hat{\mathbf{Q}}_k = \mathbf{K}_k \left[\frac{1}{N} \sum_{i=i_0}^{N} \left(\mathbf{z}_i - \hat{\mathbf{z}}_{i|N} \right) \left(\mathbf{z}_i - \hat{\mathbf{z}}_{i|N} \right)^T \right] \mathbf{K}_k^T \tag{4.53}$$

Again to capitalize on the smoothing process used earlier, the smoothed measurements and covariances are inserted into the R-adaptation scheme as well. Thus, the final adaptation for $\mathbf{R}_k$ is obtained by modifying Eq. (4.40) accordingly:

$$\hat{\mathbf{R}}_k = \frac{1}{N} \sum_{i=i_0}^{N} \left[\left(\mathbf{z}_i - \hat{\mathbf{z}}_{i|N} \right) \left(\mathbf{z}_i - \hat{\mathbf{z}}_{i|N} \right)^T + \mathbf{H}_{i|N} \mathbf{P}_{i|N}^{+} \mathbf{H}_{i|N}^T \right] \tag{4.54}$$

4.6 Chapter Summary

A review of the maximum likelihood estimation method of parameter identification was given in this chapter, and a suitable likelihood candidate function based on the innovations process of the EKF was selected. A set of adaptation laws for both the process noise covariance and the measurement noise covariance matrices were derived, by assuming that adaptations of the EKF can be implemented through the theoretical covariance of the innovations that are calculated within the filter. An internal smoothing algorithm was added to the MLE routine to expand upon these adaptation laws, thereby taking advantage of the additional information stored in the limited-memory window of the filter. The resulting MLE-AEKF designed in this chapter demonstrates that output from the EKF can be used to adapt the filter, and further serves as a baseline theoretical adaptive EKF that will be compared with both the standard EKF from Chapter 3, and the novel Fuzzy Adaptive EKF proposed in the next chapter.

Chapter 5

Fuzzy Adaptive Kalman Filtering

The following chapter presents the design of a novel fuzzy adaptive extended Kalman filter with specific applications to the relative navigation of formation flying spacecraft. A review of fuzzy logic theory is given first, followed by an overview of the proposed fuzzy system and its integration into the EKF. Performance metrics for the EKF are formulated in the context of a covariance matching objective, from which fuzzy adaptation laws of the process and noise covariance matrices within the EKF are developed. The fuzzy logic systems used to provide output parameters to the adaptation laws based on the input performance metrics are described in detail. To conclude, this chapter discusses the design and tuning considerations that were encountered during the development of the proposed FAEKF, and the final nonlinear mapping provided by the FLS is illustrated.

5.1 Introduction

The motivation for this chapter is to expand upon the notion of adaptive Kalman filtering that was investigated through the design of the MLE-AEKF in the previous chapter. While the MLE-AEKF yielded analytical equations for adapting the Kalman filter, many assumptions and simplifications were made that could potentially hinder the estimation accuracy of the filter. Through the use of Fuzzy Logic however, it is possible to construct adaptation laws based on human intuition and an understanding of the behaviour of the EKF itself. Thus, the experience gained from developing the standard EKF and the MLE-AEKF can be formulated into a numerical adaptation scheme specific to the objectives of spacecraft formation navigation. Before presenting the proposed fuzzy adaptation laws for the EKF however, a brief review of fuzzy logic theory is given in the following section.

5.2 First Principles of Fuzzy Set Theory

The seminal work of fuzzy mathematics was presented by Lotfi Zadeh (1921-2017) of the University of California, who in 1965 introduced the definition of a *fuzzy set* [50]. Contrasting the strictly discrete sets used in classical logic states, such as *1* and *0* or *on* and *off*, fuzzy sets are used to describe states with a *degree of membership*. The degree of membership of a state is taken from a continuous range between 0 and 1, and quantifies how closely the state corresponds to some linguistic classification. For example, consider classifying two distance measurements into the category of "large separations over 100 m": the first measurement is 200 m, and the second is 1000 m. Using *ordinary set theory* (classical logic), both of these measurements would be classified identically because they are both distances over 100 m. Using fuzzy set theory however, these measurements can be classified with different degrees of membership, and different control strategies can be applied commensurately.

The process of mapping a numerical value, or a *crisp input*, into a fuzzy set, is called *fuzzification*. The resulting *fuzzy inputs* can then be interpreted through a *rule base* that is defined to achieve the control objectives of the FLS. Fuzzy logic can be implemented similarly to ordinary logic, with corresponding fuzzy operators like AND, OR, and NOT. A first application of fuzzy set theory to controller design was made by Mamdani *et al.* [116], which led to the widely accepted Mamdani class of fuzzy systems. A Mamdani fuzzy system gives outputs that are unique fuzzy sets, thereby being intuitive and well-suited for encoding human intuition. The other popular method of fuzzy inference proposed by Takagi and Sugeno [117], reduces the FLS outputs to constant or linear expressions of the inputs, but is beyond the scope of this research.

After the fuzzy outputs have been inferred from the fuzzy inputs and the rule bases, the fuzzy outputs are then *defuzzified* to give numerical *crisp output* values that correspond to the control or adaptation parameters desired for the system. Figure 5.1 shows the methodology behind a typical fuzzy logic system, and the FLS developed for this thesis is discussed in more detail shortly.

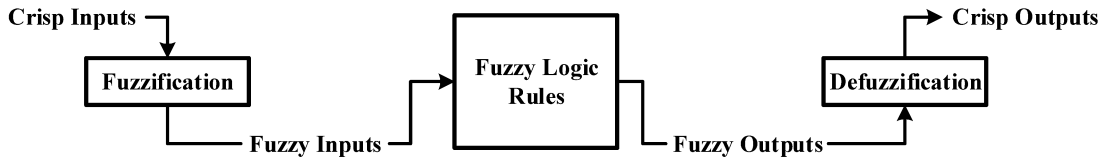


Figure 5.1: The general procedure and terminology of a fuzzy logic system, through which crisp inputs to the system are converted into fuzzy sets, processed by linguistic-based fuzzy logic operators, and output as crisp numerical values.

5.3 Overview of FAEKF Architecture

The fuzzy adaptive extended Kalman filter consists of a standard EKF with a feedback adaptation loop that adjusts the process and measurements noise covariances in real-time. The adaptations are calculated by the FLS after the correction phase of the EKF, and the block diagram in Fig. 5.2 illustrates the proposed FAEKF developed for this research. As is the EKF and MLE-AEKF scenarios, the spacecraft formation simulator presented in Chapter 2 provides noisy measurements to the EKF, and the embedded FLS within the EKF updates the process and measurement noise covariance matrices at each time step of the simulation. The overall performance of the FAEKF is similarly based upon the difference between the true states from the formation simulator, and the estimated states from the filter.

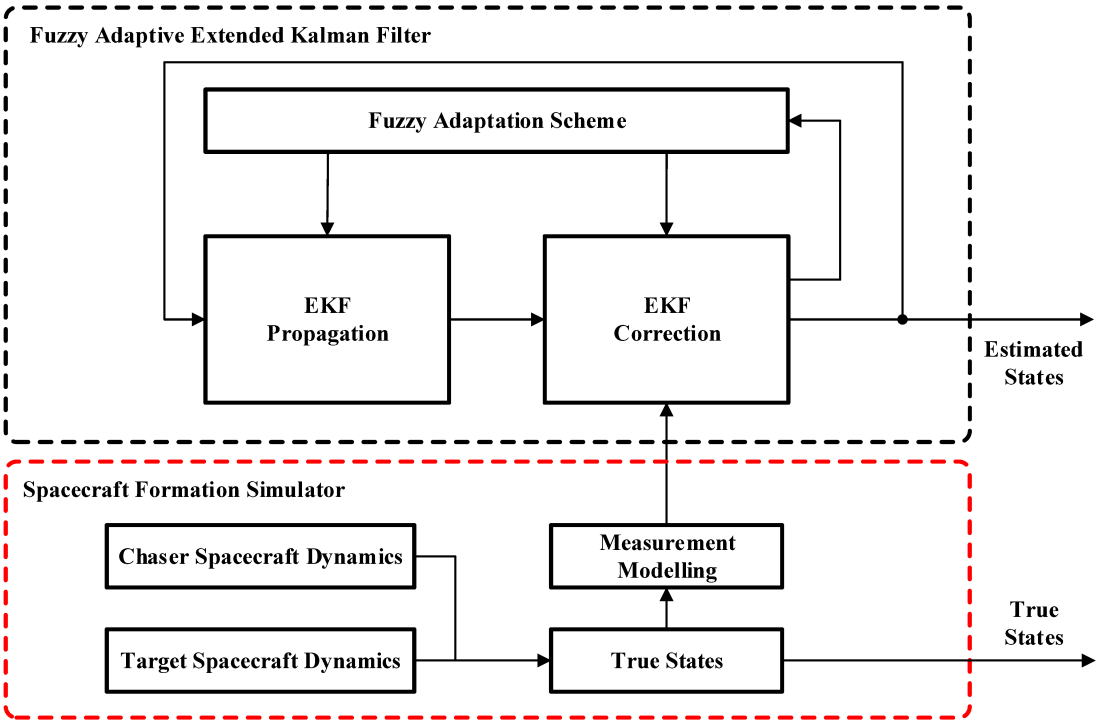


Figure 5.2: A block diagram representation of the FAEKF simulation.

The FAEKF block diagram shown previously in Fig. 5.2 calculates updates to the noise covariance matrices through the fuzzy adaptation scheme shown in Fig. 5.3; (note that this figure shows adaptations for the measurement noise covariance, but an identical structure is used for the process noise updates). Each following section will investigate the components of the designed FLS system, particularly the choice of inputs, fuzzification, rule bases, inference techniques, and defuzzification methods.

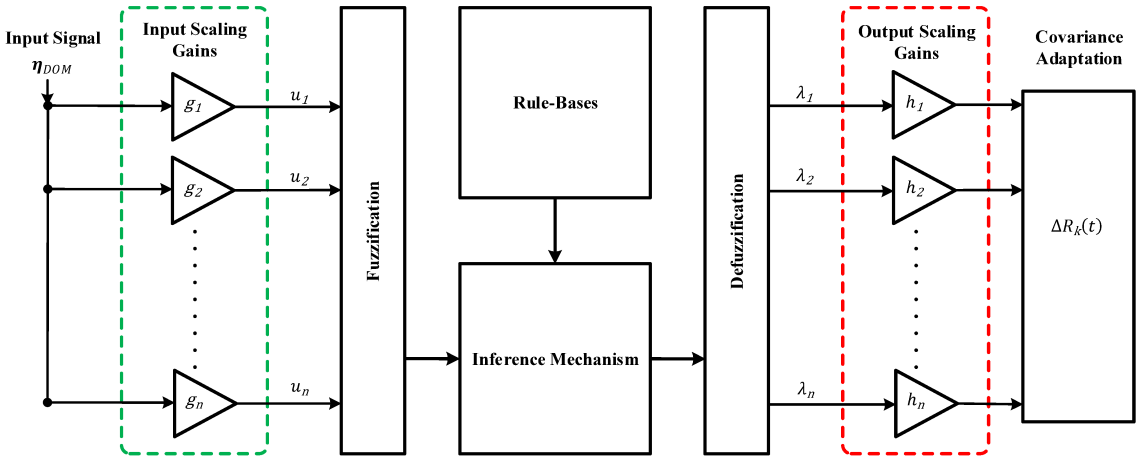


Figure 5.3: Block diagram representing the fuzzy adaptation scheme.

Within the context of an adaptive filter, the only modifications to the original EKF equations relate to the process and measurement noise covariance matrices. In Chapter 3, the original definition of the noise covariances stated that $\mathbf{Q}_k$ relates to the confidence in the dynamics model, and $\mathbf{R}_k$ relates to the confidence in the available measurements, where both $\mathbf{Q}_k$ and $\mathbf{R}_k$ were taken as constants. Throughout Chapter 4 it was shown that time-varying estimates of the covariance matrices could be developed based on observations of the filter residuals, using a gradient-type minimization of the negative log-likelihood function of the residuals. This chapter similarly applies the notion of time-varying covariance matrix estimates, as functions of the previous covariance and the outputs from a Fuzzy Logic System (FLS). That is to say, for a given FLS input u_k and the corresponding FLS output λ_k, the estimated process and measurement covariance matrices are described by nonlinear mapping functions:

$$\hat{\mathbf{Q}}_{k+1} = f\Big(\mathbf{Q}_k, \lambda_k(u_k)\Big) \tag{5.1}$$

$$\hat{\mathbf{R}}_{k+1} = f\Big(\mathbf{R}_k, \lambda_k(u_k)\Big) \tag{5.2}$$

5.3.1 Criteria for Innovation-based Covariance Matching

Whereas the previous MLE-AEKF sought to obtain closed-form solutions that maximized the likelihood function of the EKF based on the covariance of the residuals, the Fuzzy Adaptive EKF developed here uses a covariance matching method to adapt the process and measurement noise covariances, such that the observed residuals covariance matrix (RCM) matches the theoretical RCM calculated by the filter. For this chapter, the theoretical RCM will again be defined with

$$\boldsymbol{\Sigma}_k = \mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k \tag{5.3}$$

and the observed RCM uses the definition from the last chapter, where a fixed-window averaging of the past N residuals gives

$$\hat{\boldsymbol{\Sigma}}_{k|N} \triangleq \frac{1}{N} \sum_{i=i_0}^{N} \boldsymbol{\nu}_i \boldsymbol{\nu}_i^T \tag{5.4}$$

Using the innovation-based covariance matching ideas discussed in Chapter 1, the proposed fuzzy adaptive update laws are based on the discrepancies between the actual observed filter covariance and the theoretical covariance bounds suggested by the filter. The first performance metric selected for adapting the EKF has been demonstrated by several other authors [59, 60, 118], and is coined the *Degree of Mismatch* (DOM). The DOM matrix $\boldsymbol{\eta}_{DOM} \in \mathbb{R}^{m \times m}$ is simply the element-by-element difference between the theoretical covariance of the residuals $\boldsymbol{\Sigma}_k \in \mathbb{R}^{m \times m}$ calculated within the EKF, and the experimental covariance of the residuals sampled from the filter output $\hat{\boldsymbol{\Sigma}}_{k|N} \in \mathbb{R}^{m \times m}$, which have previously been presented in Eqs. (5.3) and (5.4), respectively. So, components of the DOM can be thought of as error signals, with the full matrix written as

$$\boldsymbol{\eta}_{DOM} \triangleq \boldsymbol{\Sigma}_k - \hat{\boldsymbol{\Sigma}}_{k|N} \tag{5.5}$$

Another parameter constructed to detect sub-optimal performance of the filter is the *Degree of Divergence* (DOD) [118, 119], which will be denoted by $\eta_{DOD} \in \mathbb{R}$. Evaluating the trace of the residual covariance matrix gives a scalar value $\xi \in \mathbb{R}$ that can be used to indicate overall divergence of the filter, or outliers within the data. These traces can be identified for both the theoretical RCM and the observed RCM,

as denoted by ξ and $\hat{\xi}$, respectively:

$$\xi \triangleq \text{Tr}[\mathbf{\Sigma}_k] \tag{5.6}$$

$$\hat{\xi} \triangleq \boldsymbol{\nu}_k^T \boldsymbol{\nu}_k = \text{Tr}[\boldsymbol{\nu}_k \boldsymbol{\nu}_k^T] = \text{Tr}[\hat{\mathbf{\Sigma}}_k] \approx \text{Tr}[\hat{\mathbf{\Sigma}}_{k|N}] \tag{5.7}$$

Following the idea of the Degree of Mismatch metric introduced previously, the degree of divergence η_{DOD} yields a scalar parameter that captures differences between the true filter performance and the theoretical performance. In this case, larger values of η_{DOD} signify the filter is providing poorer estimates of the state.

$$\begin{aligned} \eta_{DOD} &\triangleq \boldsymbol{\nu}_k^T \boldsymbol{\nu}_k - \text{Tr}[\mathbf{\Sigma}_k] = \text{Tr}[\boldsymbol{\nu}_k \boldsymbol{\nu}_k^T] - \text{Tr}[\mathbf{\Sigma}_k] \\ &= \text{Tr}[\hat{\mathbf{\Sigma}}_k] - \text{Tr}[\mathbf{\Sigma}_k] = \text{Tr}[\hat{\mathbf{\Sigma}}_k - \mathbf{\Sigma}_k] \\ &= \hat{\xi} - \xi \\ &= \text{Tr}[\boldsymbol{\eta}_{DOM}] \end{aligned} \tag{5.8}$$

The fuzzy logic system used for adapting the measurement noise covariance $\mathbf{R}_k$ involves significantly more terms than the adaptation of the process noise covariance $\mathbf{Q}_k$, due to the selected adaptation mechanism. As such, the following sections present the FLS designs for R-adaptation first, and then follow with the simpler designs for the Q-adaptations.

5.3.2 Adaptation of the Measurement Noise Covariance

Adaptations of the measurement noise covariance $\mathbf{R}_k$ are made using the degree of mismatch $\boldsymbol{\eta}_{DOM}$ as the input to the fuzzy logic system. The residual covariance and the measurement noise covariance matrices have the same dimension, such that each element in the DOM matrix directly corresponds to an element in the noise covariance matrix. Taking advantage of this relationship, and assuming that the covariances are diagonal, each element of $\mathbf{R}_k$ is adapted individually using a scale factor $\varepsilon_k^r \in \mathbb{R}$. The proposed update law for the measurement noise covariance matrix is defined

$$\hat{\mathbf{R}}_{k+1}(i,i) \triangleq \varepsilon_k^r(i) \mathbf{R}_k(i,i) \tag{5.9}$$

where the scale factor $\varepsilon_k^r(i)$ corresponds to the i^{th} diagonal element of the noise covariance matrix. Each scale factor is dependant on an output gain $h^r(i)$ and the output of the FLS $\lambda_k^r(i) \in \mathbb{R}\ \forall i = 1, ..., 7$, through the relation

$$\varepsilon_k^r(i) \triangleq 1 + h^r(i) \lambda_k^r(i) \tag{5.10}$$

It should further be realized that the design of the FLS must ensure $h^r \lambda_k^r > -1 \; \forall k$, such that $\hat{\mathbf{R}}_k$ is always positive definite. Further discussion on the FLS gains is provided in a later discussion regarding the selected membership functions.

5.3.3 Adaptation of the Process Noise Covariance

For situations where noise in the system dynamics are variable or poorly defined, adaptations of the process noise covariance matrix $\mathbf{Q}_k$ are achieved using the degree of divergence η_{DOD} as the input to a single fuzzy logic system. Having presented both the state and measurement models available in the spacecraft formation flying problem, it can be seen that only partial observation of the relative states is made; more colloquially, only 7 measurements of the system are available, but there are 10 states to be estimated. Knowing that the dimension of the process noise covariance is larger than the dimension of the residuals covariance, a direct correlation between the two cannot be made; therefore, the scalar DOD is used to develop a rudimentary solution. Applying a single scaling factor $\varepsilon_k^q \in \mathbb{R}$ to the process noise covariance, all elements of $\mathbf{Q}_k$ are adjusted using the update law

$$\hat{\mathbf{Q}}_{k+1} \triangleq \varepsilon_k^q \mathbf{Q}_k \tag{5.11}$$

The covariance scale factor ε_k^q is a function of the FLS output λ_k^q multiplied by an output gain h^q, that is

$$\varepsilon_k^q \triangleq 1 + h^q \lambda_k^q \tag{5.12}$$

From these relations, it can be seen that if the output of the FLS is zero, the covariance scale factor is unity and the system reduces to the standard Kalman filter. As before, the design of the FLS should ensure $h^q \lambda_k^q > -1 \; \forall k$, to maintain a positive definite symmetric $\hat{\mathbf{Q}}_k$.

5.4 Fuzzification

The fuzzification process is used to map the crisp inputs from either the DOD or DOM metrics into corresponding fuzzy sets. The following section specifies the gains used to scale the inputs and outputs of the FLS, the linguistic variables used to describe the inputs and outputs, and the membership functions used to assign the degree of membership to the crisp values. The fuzzy logic rules are also introduced, and the methodology behind their role in the covariance matching technique is given.

5.4.1 For the Measurement Noise Covariance

Adaptations for the measurement noise covariance $\mathbf{R}_k$ are found using seven single-input single-output fuzzy systems, so the FLS input and output indices are $i = 1, 2, \ldots, n = 7$ and $q = 1, 2, \ldots, m = 7$, respectively. The input to the i^{th} FLS is the corresponding diagonal element of the degree of mismatch matrix $\boldsymbol{\eta}_{DOM}$, and the controllable universe of discourse for each of the DOM components are defined to be ± 20 m^2 for the position mismatch, ± 1 m^2/s^2 for the velocity mismatch, and ± 0.1 rad^2 for the true anomaly mismatch. Values outside of these ranges result in a saturated input signal, which means that the output of the FLS is identical for any values outside the controllable universe of discourse. Following common practice, the fuzzy system is normalized to allow for efficient visual comparison of the different FLS schemes used in conducting this research.

Nominal input scaling gains $g_i^r \in \mathbb{R}\ \forall i = 1, \ldots, 7$ are chosen to be $g_1^r = g_2^r = g_3^r = 0.05$, $g_4^r = 10$, and $g_5^r = g_6 = g_7^r = 1$, which yield normalized crisp inputs $u_i \in \mathcal{U}_i\ \forall i = 1, ..., 7$ within the controllable universe of discourse $\mathcal{U}_i = \begin{bmatrix} -1 & 1 \end{bmatrix}$. Constant *linguistic variables* $\tilde{u}_i$ are used to describe the time-varying crisp inputs u_i, and the linguistic variables for the inputs are defined as follows:

$$\begin{aligned} \tilde{u}_1, \tilde{u}_2, \tilde{u}_3 &\sim \text{ Degree of mismatch for the } x,y,z\text{-position components} \\ \tilde{u}_4 &\sim \text{ Degree of mismatch for the true anomaly } \theta \text{ component} \\ \tilde{u}_5, \tilde{u}_6, \tilde{u}_7 &\sim \text{ Degree of mismatch for the } x,y,z\text{-velocity components} \end{aligned}$$

The set of *linguistic values* that are assigned to the input linguistic variables are defined as

$$\tilde{A}_i = \left\{ \tilde{A}_i^j \ : \ i = 1, \ldots, n = 7 \ : \ j = 1, \ldots, N_i = 5 \right\} \tag{5.13}$$

where $\tilde{A}_i^j$ is the j^{th} linguistic value of linguistic variable $\tilde{u}_i$, and $N_i = 5$ is the number of input linguistic values. An equivalent definition can be made for the outputs, where $\tilde{B}_q^p$ is the p^{th} linguistic value of output linguistic variable $\tilde{\lambda}_q$. For a given number of output linguistic values $M_q = 7$, the set of linguistic values is

$$\tilde{B}_q = \left\{ \tilde{B}_q^p \ : \ q = 1, \ldots, m = 7 \ : \ p = 1, \ldots, M_q = 7 \right\} \tag{5.14}$$

The input linguistic variables used in this work are "negative high (NH), negative low (NL), zero (ZERO), positive low (PL)", and "positive high (PH)". Similarly, the output linguistic variables are "negative maximal (NMAX), negative minimal (NMIN), zero (ZERO), positive minimal (PMIN)" and "positive maximal (PMAX)".

5.4.2 For the Process Noise Covariance

The process noise covariance $\mathbf{Q}_k$ is adapted using one SISO fuzzy system. Indices for the FLS input and output are then simply $i = n = 1$ and $q = m = 1$, respectively. The input to the FLS is the degree of divergence η_{DOD}, and the universe of discourse for the DOD is defined as ± 200 m^2. The input is normalized by an input scaling gain selected as $g^q = -5 \times 10^{-3}$, and this gives the normalized crisp input $u \in \mathcal{U}$ within the controllable universe of discourse $\mathcal{U} = \begin{bmatrix} -1 & 1 \end{bmatrix}$. The time-varying crisp input u is described by a corresponding constant linguistic variable $\tilde{u}$, which for the Q-adaptations is:

$$\tilde{u} \sim \text{ Degree of divergence of the residuals covariance matrix} \tag{5.15}$$

The set of linguistic values that can be assigned to the input linguistic variables are defined as

$$\tilde{A}_i = \left\{ \tilde{A}_i^j \ : \ i = n = 1 \ : \ j = 1, \ldots, N_i = 5 \right\} \tag{5.16}$$

where $\tilde{A}_i^j$ is the j^{th} linguistic value of linguistic variable $\tilde{u}_i$, and the number of input linguistic values is $N_i = 5$. For the outputs, $\tilde{B}_q^p$ is the p^{th} linguistic value of output linguistic variable $\tilde{\lambda}_q$. Since the Q-adaptations only use a single FLS, the number of output linguistic values is $M_q = 1$, and the set of linguistic values is

$$\tilde{B}_q = \left\{ \tilde{B}_q^p \ : \ q = m = 1 \ : \ p = M_q = 1 \right\} \tag{5.17}$$

The input linguistic variables used for the Q-adaptations are the same as those used for the R-adaptations, namely: "negative high (NH), negative low (NL), zero (ZERO), positive low (PL)", and "positive high (PH)", and the output linguistic variables are again taken as "negative maximal (NMAX), negative minimal (NMIN), zero (ZERO), positive minimal (PMIN)" and "positive maximal (PMAX)".

5.4.3 Membership Functions

The nonlinear mapping from the crisp inputs u_i to the crisp outputs λ_q requires that the inputs first be mapped into their respective fuzzy sets, given as A_i^j. The fuzzy sets characterize how much the particular value of an input meets the criteria defined by each of the relevant linguistic values. This characterization is defined in terms of the membership function $\mu_{A_i^j}(u_i)$, which maps the input universe of discourse $\mathcal{U}_i \mapsto [0, 1]$. Intuitively, the higher the degree of membership for a given linguistic value, the more certainty we have that the linguistic value accurately describes the observed input.

Mathematically, the fuzzy sets for the input are defined as

$$A_i^j = \left\{ \ \left(u_i, \mu_{A_i^j}(u_i)\right) \ : \ u_i \in \mathcal{U}_i \ \right\} \qquad i = 1, \ldots, n = 7 \quad j = 1, \ldots, N_i = 5 \quad (5.18)$$

Membership functions for the inputs are modelled using a combination of Sigmoidal and Gaussian curves, which ensures all fuzzified inputs have non-zero membership over the breadth of the universe of discourse. The Sigmoid functions are used to saturate the extrema of the universe of discourse, preventing inputs outside the effective control range from yielding net-zero memberships. Thus, the two outermost membership functions of the inputs are written as

$$\mu_{A_i^j}(u_i;\ a_{A_i^j}, b_{A_i^j}) = \frac{1}{1 + \exp\left[-a_{A_i^j}(u_i - b_{A_i^j})\right]} \qquad i = 1, \ldots, 7 \quad j = 1, 5 \qquad (5.19)$$

where $a_{A_i^j} \in \mathbb{R}$ defines the curvature of the function and $b_{A_i^j} \in \mathbb{R}$ designates the center of the function (*i.e.*, where the function crosses 0.5). The remaining inputs are classified using Gaussian membership functions centered at $c_{A_i^j}$ with widths of $\sigma_{A_i^j}$, giving

$$\mu_{A_i^j}(u_i;\ c_{A_i^j}, \sigma_{A_i^j}) = \exp\left[\frac{-(u_i - c_{A_i^j})^2}{2\sigma_{A_i^j}^2}\right] \qquad i = 1, \ldots, 7 \quad j = 2, 3, 4 \qquad (5.20)$$

Equivalent mathematical definitions are made for the output membership functions $\mu_{B_i^j}$, which are all modelled using the Gaussian curves given by Eq. (5.20). To be thourough, these can be expressed by

$$\mu_{B_q^p}(\lambda_q;\ c_{B_q^p}, \sigma_{B_q^p}) = \exp\left[\frac{-(\lambda_q - c_{B_q^p})^2}{2\sigma_{B_q^p}^2}\right] \qquad q = 1, \ldots, 7 \quad p = 1, \ldots, 5 \qquad (5.21)$$

Figures 5.4 and 5.5 show the normalized input and output membership functions resulting from Eqs. (5.19) and (5.20), where each curve is defined with 100 points spread equidistant along the respective universe of discourses. The parameters for the Sigmoidal membership functions are $a_{A_i^j} = \{-0.75, 0.75\}$ and $b_{A_i^j} = 25$, and the Gaussian membership functions are centered at $c_{A_i^j} = \{-0.5, 0, 0.5\}$ and $c_{B_i^j} = \{-1, -0.25, 0, 0.25, 1\}$, with widths of $\sigma_{A_i^j} = \sigma_{B_i^j} = 1/12$. These membership functions are used for both the Q-adaptations and the R-adaptation schemes.

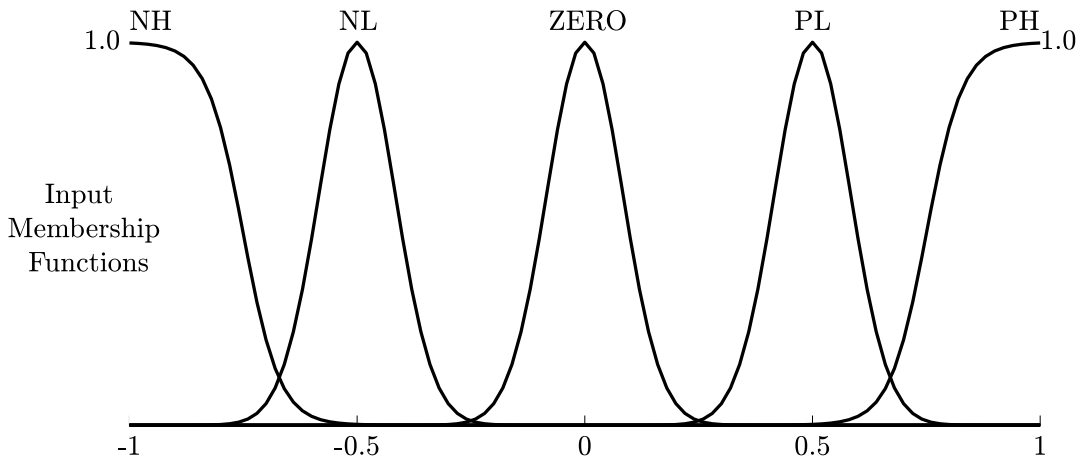


Figure 5.4: Input membership functions for the normalized input variables.

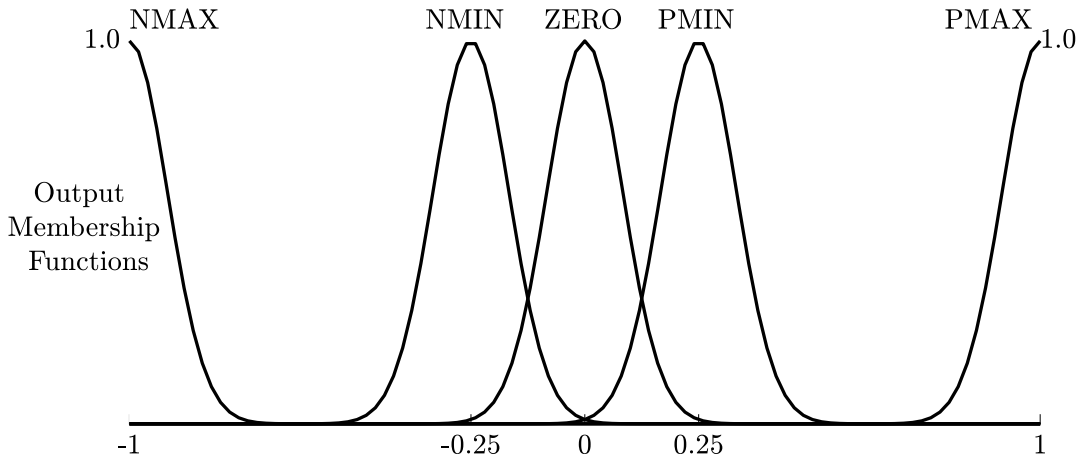


Figure 5.5: Output membership functions for the normalized output variables.

5.4.4 Discussion on FLS Design and Tuning

To provide insight into the design of the fuzzification scheme used here, briefly consider the effects of the input gains and the membership functions within the FLS. The input scaling gains control the magnitude of the respective input relative to the entire universe of discourse, thereby allowing the designer to identify specific inputs as "more critical" or "more sensitive" than others if desired. This can be thought of alternatively as modifying the membership functions of the inputs: if the input gains are set equal to one, the membership functions directly correspond to the linguistic values they refer to, but if the scaling gains are set larger than one, the FLS behaves as if the membership functions are contracted and characterize smaller numerical values. For this work, the input gains are chosen to be less than one, such that inputs are normalized and the membership functions represent larger values than their actual defined universe of discourse between 0 and 1. In terms of adaptation performance, reducing the input scaling gains leads to smaller and slower adaptations of the system.

Difficulties were encountered with the R-adaptations when the covariances were decreased too much, such that the filter was placing more weight on the measurements than was ideal. This was seen when behaviour of the noise distribution added to the measurements (*i.e.*, white Gaussian with a 1 Hz frequency) would be visible within the filter estimates. To mitigate this, the input scaling gains were further reduced so that the sensitivity of the FLS would be reduced for comparable values of the DOM. This effectively means that the range of crisp input values defining desirable filter performance is expanded.

Tuning the Output Membership Functions

Another unique property of the FLS presented here is the distribution of the output membership functions; relocating the position of the center of the output membership functions allows the fuzzy system to provide different control responses [120]. Consider a set of i-membership functions defined by their centers c^i, where $i \in \{-2, -1,\ 0,\ 1,\ 2\}$. Including a typical input gain g and output gain h, the common linearly-distributed centers of these membership functions can be expressed by

$$c^i = \left(\frac{h}{g}\right) i \tag{5.22}$$

The uniform spacing of the output membership functions described by Eq. (5.22) means that regardless of the magnitude of the inputs, the magnitude of the output will be proportional. However, it can be useful to have large control responses in

situations where large inputs are encountered, and smaller, finer control responses as the input approaches the desired value. One way to achieve this behaviour is by positioning the centers of the output membership functions following

$$c^i = \frac{1}{2}\left(\frac{h}{g}\right)\text{sign}\,(i)\,i^2 \tag{5.23}$$

A result of this distribution is that as the membership functions move farther away from zero, they are also spaced farther apart. For smaller inputs, the behaviour of this fuzzy system will be similar, but now for larger inputs the resulting response will be larger. Thus, by spacing the output membership functions instead of increasing the input gain, larger adaptation responses are obtained without amplifying noises in the inputs to the fuzzy system.

The above center spacing method applies for a general universe of discourse with arbitrary scaling gains. For this research however, the membership functions are defined on a normalized range, where the equally-spaced centers of the membership functions are $c^i,\ i \in \{-1, -0.5,\ 0,\ 0.5,\ 1\}$. In this normalized case, the spreading function equivalent to Eq. (5.23) can readily be simplified to

$$c^i = \text{sign}\,(i)\,i^2 \tag{5.24}$$

This yields the centers of the output membership functions that were presented earlier, which are selected with centers at $c^i = \{-1, -0.25, 0, 0.25, 1\}$.

A Note on Adaptation Schemes with Poor Performance

Prior to implementing the scaling adaptations described above, another method using additive components was tested. While the additive adaptations could be tuned to work for specific formation configurations and noise levels, the limited magnitude of the additive parameter restricts EKF performance in dissimilar formation scenarios. That is to say, an additive system tuned for circular low Earth orbits would saturate when used in large elliptical orbits, resulting in significantly longer convergence times for the appropriate covariance matrices. A similar delay in convergence is also observed if large initial covariance estimates are used. To illustrate this further, consider the R-adaptation equations below using the additive update $\Delta\mathbf{R}_k$, and remember that $h^r(i)$ is constant and $\lambda_k^r(i)$ is restricted to its universe of discourse.

$$\hat{\mathbf{R}}_{k+1}(i,i) \triangleq \mathbf{R}_k(i,i) + \Delta\mathbf{R}_k(i,i) \tag{5.25}$$

$$\Delta\mathbf{R}_k(i,i) = h^r(i)\lambda_k^r(i) \tag{5.26}$$

5.4.5 Logic Rule Bases

The fuzzy logic framework can be used to provide highly specialized control and adaptation laws specific to an application, since the designer is effectively able to code human knowledge and reasoning into *Rule Bases* for the FLS. Using the linguistic variables introduced previously, governing IF-THEN rules for the FLS are written in terms of a *premise* and a *consequent*. For this work, with the inputs corresponding to the premises and the outputs corresponding to the consequents, the rules are written as:

IF $\tilde{u}_1$ is PL **THEN** λ_1 is PMIN
IF $\tilde{u}_4$ is ZERO **THEN** λ_4 is ZERO
IF $\tilde{u}_7$ is NMAX **THEN** λ_7 is NH

and so forth. These rules are a large design consideration when customizing the fuzzy system, and have been chosen to align with the innovation-based covariance matching scheme presented earlier. As an example, if the degree of mismatch for a particular covariance element is large, this implies the theoretical RCM is larger than the observed RCM, and the theoretical noise covariance should be decreased. With this in mind, the linguistic rules are defined such that the output of the FLS will adapt the noise covariance matrix accordingly, and Table 5.1 summarizes the rule bases used for the fuzzy system. For the table shown below, there are a total of $n_r = 5$ rules, so the rule base index can be defined as $i_r = 1, \ldots, 5$.

Table 5.1: Rule Table for the SISO Fuzzy Logic System

Inputs $\{ \tilde{u}_i \}$	**NH**	**NL**	**ZERO**	**PL**	**PH**
Outputs $\{ \tilde{\lambda}_q \}$	NMAX	NMIN	ZERO	PMIN	PMAX

5.5 The Inference Mechanism

After successfully mapping the inputs into their respective fuzzy sets, and defining rule bases that contain the logical relations to be maintained by the FLS, the process of *inference* is used to determine how likely the input corresponds to a particular output. Since each of the rules applied here involves a single premise only, and assuming singleton fuzzification, the degree of certainty to which the i_r^{th} rule applies for the i^{th} input is simply given by the membership function of the input evaluated for that rule.

Hence:

$$\mu_{i_r}(u_i) = \mu_{A_i^j}(u_i) \qquad i = 1, \ldots, 7, \quad i_r = 1, \ldots, 5 \tag{5.27}$$

The inference stage is then completed using the MIN operator, whereby the ouput membership functions $\mu_{B_q^i}(\lambda_q)$ of the implied fuzzy sets B_q^i are evaluated as

$$\begin{aligned} \mu_{B_q^{i_r}}(\lambda_q) &= \min\left[\mu_{i_r}(u_i),\ \mu_{B_q^p}(\lambda_q)\right] \\ i &= q = 1, \ldots, 7 \\ p &= 1, \ldots, 5 \\ i_r &= 1, \ldots, 5 \end{aligned} \tag{5.28}$$

Much like the input membership functions, the implied fuzzy set B_q^i quantifies the level of membership the output has for a specific crisp output, albeit only taking into account the i_r^{th} rule. The MIN operator identifies the intersection of the two fuzzy membership functions within the argument, and therefore ensures that uncertainties in the premises (through $\mu_{A_i^j}$) limit the confidence of the conclusions from the inference process. Implied fuzzy sets have been favoured for this work, as the use of aggregated implied fuzzy sets was found to be computationally burdensome, as predicted by Passino [120].

5.6 Defuzzification

The defuzzification process calculates crisp values for the outputs $\lambda_q \in \Lambda_q$ from within their appropriate universe of discourse Λ_q, using the center of gravity (COG) method for implied fuzzy sets [121]. Defuzzification with COG gives the crisp output as

$$\lambda_q = \frac{\sum_{i_r=1}^{n_r} b_{i_r}^q \int_{\Lambda_q} \mu_{B_q^{i_r}}(\lambda_q)\, \mathrm{d}\lambda_q}{\sum_{i_r=1}^{n_r} \int_{\Lambda_q} \mu_{B_q^{i_r}}(\lambda_q)\, \mathrm{d}\lambda_q} \qquad q = 1, \ldots, 7 \tag{5.29}$$

Here $n_r = 5$ is the number of linguistic rules for the adaptations, and $b_{i_r}^q \in \mathbb{R}$ is the center of area of the membership function of the output fuzzy set B_q^p, which corresponds to the implied fuzzy set $B_q^{i_r}$ for the i_r^{th} rule. Several things to note regarding Eq. (5.29): first, the integral term in the numerator equates to the area under the membership function of the implied fuzzy set, and a larger area corresponds to a higher certainty with which that particular rule applies. Since this integral exists in the denominator as well however, the FLS must be designed to ensure the system

is always defined, so it must hold that

$$\sum_{i_r=1}^{n_r} \int_{\Lambda_q} \mu_{B_q^{i_r}}(\lambda_q)\, \mathrm{d}\lambda_q \neq 0 \tag{5.30}$$

The crisp normalized outputs are lastly re-dimensionalized through multiplication with the appropriate scaling gains. The universe of discourse for the normalized outputs λ_q spans $\Lambda = \begin{bmatrix} -1 & 1 \end{bmatrix}$, and the nominal output scaling gains are $h_1^r = h_2^r = h_3^r = h_r^r = 10^{-4}$, $h_4^r = h_\theta^r = 10^{-4}$, and $h_5^r = h_6^r = h_7^r = h_v^r = 10^{-3}$ for the R-adaptation scheme. For Q-adaptations, the single output gain is $h^q = 0.01$. Once the crisp fuzzy output has been un-normalized, the update to either $\mathbf{Q}_k$ or $\mathbf{R}_k$ is calculated through Eqs. (5.10) or (5.12), respectively.

Similar to the influence of the input scaling gains discussed earlier, the output gains have a significant impact on the adaptation performance of the FLS. Output gains less than one are equivalent to contracting the output membership functions such that they correspond to smaller crisp output values, while gains larger than one have the effect of making the associated linguistics quantify larger numbers. Qualitatively, increasing the output gains increases the covariance adaptation rate.

Nonlinear Adaptation Curve

With the selected input gains, membership functions, rule bases, and inference methods presented in this section, the resulting nonlinear mapping of the crisp inputs to the crisp outputs is defined by the adaptation curve shown in Fig. 5.6. Here it can be seen that for larger inputs, corresponding to larger discrepancies in the theoretical and observed residual covariances, the adaptation response will be larger, while the response will be smaller and more precise as the inputs approach zero (*i.e.*, when the covariances are matching).

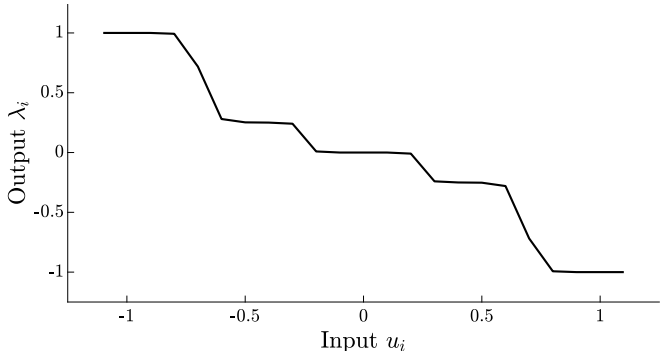


Figure 5.6: Nonlinear map of the fuzzy logic system.

5.7 Chapter Summary

The fuzzy logic system presented in this chapter is designed to update the process and measurement noise covariances within the extended Kalman filter developed in Chapter 3. Using a FLS allows the designer to incorporate human intuition and expert knowledge of the EKF behaviour into the adaptation laws, thereby providing a flexible adaptation strategy compared to the analytic MLE-AEKF adaptations developed in Chapter 4. Instead of trying to minimize a likelihood function, the FLS uses performance metrics based on the theoretical and observed residual covariance matrices of the EKF. The resulting fuzzy rule base is defined such that updates to the noise covariance matrices will minimize the difference between the theoretical and observed covariances of the residuals.

Chapter 6

Numerical Analysis of the Formation Navigation Algorithms

The penultimate chapter of this thesis presents the numerical simulation results of the three EKF algorithms developed for the purposes of relative spacecraft navigation. The objective of this chapter is to illustrate that the designed filters are able to estimate the relative position and velocity states of a spacecraft formation with better accuracy than the measurements provided to the filters. The performances of the standard EKF, the MLE-AEKF, and the FAEKF are demonstrated through three case studies using realistic spacecraft formations in unique configurations, to verify that the filters are suitable in a variety of orbital environments. A comparison of the accuracy and computational complexity of the filters is given, and the strengths and weaknesses of each algorithm are identified.

6.1 Introduction

As introduced throughout the previous chapters, simulations of the formation flying navigation filters for this thesis are completed in the MATLAB/Simulink environment. Specifically, MATLAB R2018a and Simulink 9.1 as provided by The Mathworks Inc. are utilized, operating on an AMD Phenom II 2.90 GHz processor. The orbit propagator and the EKF algorithms populate a single Simulink model, and are therefore run in parallel. Only one EKF variant is used during each simulation; a number of user inputs therefore need to be specified prior to running a test case, and the following subsections identify these settings. Once the operation of the simulator has been clarified, a brief discussion on the error metrics of the EKF will be given.

6.1.1 Configuring the Simulator

Each of the spacecraft formations studied in this chapter are simulated using the same Simulink model, albeit with different Kalman filter settings, tuning parameters, and initial conditions. The options listed below must be defined before beginning a simulation, and have a significant influence on the overall data that will be collected.

Simulation Duration

The orbital scenarios considered here use a simulation duration equal to twice the orbital period of the target spacecraft. Since the altitudes of the formation configurations differ, this approach is more appropriate than simply defining a fixed simulation duration. Two orbital periods is sufficient to observe the steady state behaviour of the Kalman filters, and demonstrates the full range of nonlinear dynamics that exist between the perigee and apogee of the orbits.

Orbital Perturbations

The real-world orbit propagator can be modified to include all, some, or none of the orbital perturbations outlined in Section 2.5. Corresponding flags within the initialization script of the simulation determine which perturbations are calculated and included in the truth model for the spacecraft formation. The four primary perturbations due to J_2, atmospheric drag, Luni-Solar third-body gravity, and Solar radiation pressure are included in the analyses presented here. A parameter file containing the necessary astronomical constants for the orbit propagator is automatically loaded during the initialization of the simulator.

Selecting the Formation

The three spacecraft configurations considered in this chapter include the PRISMA formation, the PROBA-3 formation, and the Projected Elliptical Orbit in Low Earth Orbit formation (which is abbreviated to PEO from here onwards). Appropriate parameter input files are defined for each formation, based on values available in literature. The user specifies the name of the desired formation, and the initialization script loads the appropriate input file.

Choosing the Kalman Filtering Algorithm

The three Extended Kalman filters are included in a single Simulink diagram through the use of a *variant subsystem* block. This block allows all three filters to be connected to the same input/output data channels, while only activating and computing the filter model selected by the user.

Initializing the Kalman Filtering

The Kalman filter requires four pieces of information during the first step of implementation: an initial state estimate $\mathbf{x}_0$, and initial state error covariance $\mathbf{P}_0$, and initial process noise covariance $\mathbf{Q}_0$, and an initial measurement noise covariance $\mathbf{R}_0$. Specific values for each of these matrices are given in the appropriate analysis section throughout this chapter, but they are written in general as:

$$\mathbf{x}_0 = \begin{bmatrix} x_0 & y_0 & z_0 & \theta_0 & r_{t_0} & \dot{x}_0 & \dot{y}_0 & \dot{z}_0 & \dot{\theta}_0 & \dot{r}_{t_0} \end{bmatrix}^T \tag{6.1}$$

$$\mathbf{P}_0 = \text{diag}\begin{bmatrix} \sigma_x^2 & \sigma_y^2 & \sigma_z^2 & \sigma_\theta^2 & \sigma_{r_t}^2 & \sigma_{\dot{x}}^2 & \sigma_{\dot{y}}^2 & \sigma_{\dot{z}}^2 & \sigma_{\dot{\theta}}^2 & \sigma_{\dot{r}}^2 \end{bmatrix} \tag{6.2}$$

$$\mathbf{Q}_0 = \text{diag}\begin{bmatrix} q_x & q_y & q_z & q_\theta & q_{r_t} & q_{\dot{x}} & q_{\dot{y}} & q_{\dot{z}} & q_{\dot{\theta}} & q_{\dot{r}} \end{bmatrix} \tag{6.3}$$

$$\mathbf{R}_0 = \text{diag}\begin{bmatrix} r_x & r_y & r_z & r_\theta & r_{\dot{x}} & r_{\dot{y}} & r_{\dot{z}} \end{bmatrix} \tag{6.4}$$

The subscript 0 indicates that these are initial conditions. Note that the covariance matrices here are initialized as diagonal matrices, where the terms $\{\sigma^2, r, q\}$ are the corresponding covariance components for the respective states or measurements. In all scenarios, the initial state estimates are offset from the true states to ensure that the EKF can indeed converge to the desired state estimates. A summary table of the initialized parameters used in simulation is presented at the beginning of each case study.

6.1.2 Quantifying Performance

Comparisons between the various filter algorithms are based on the resulting *estimation error* from the simulations, which defines the difference between the true state of the formation and the state estimated by the filter. Drawing on the original definition of error from within the Kalman filter derivation, the state estimation error at the given time t_k is denoted by $\mathbf{e}_k$, and is calculated with

$$\mathbf{e}_k = \mathbf{x}_k - \hat{\mathbf{x}}_k \tag{6.5}$$

where $\mathbf{x}_k$ contains the true states as obtained from the spacecraft formation simulator, and $\hat{\mathbf{x}}_k$ is the estimated states from the Kalman filter. Many of the plots in this chapter will present the estimation errors for the relative position components x, y, z and the

relative velocity components $\dot{x}, \dot{y}, \dot{z}$, which will be compared with the corresponding state errors of the measurements.

Error covariance bounds of the Kalman filter are also displayed throughout this chapter, to demonstrate that the filter estimation errors stay within the covariance bounds predicted by the filter covariance $\mathbf{P}_k$. The standard deviation σ of the i^{th} state is calculated from the corresponding i^{th} diagonal element of the state error covariance matrix of the filter. Thus, at a given time t_k, the theoretical standard deviation of the state error is given by

$$\sigma_k(i) = \sqrt{\mathbf{P}_k(i,i)} \tag{6.6}$$

Both the negative and positive 3σ-bounds of the filter are plotted when considering the estimation errors, thereby illustrating the range of errors that the filter estimates should fall between with 99% certainty.

While the estimation error and covariance bounds provide reasonable metrics for comparing the navigation solutions at a given instant in time, it is useful to develop a scalar term that quantifies the overall filter performance over an extended period of time. The Root-Mean-Square (RMS) error is one such temporal error term, which considers the past N data points from the filter output. Calculating the RMS error for the i^{th} state of the estimated state vector $\hat{\mathbf{x}}$, denoted here as $\hat{\mathbf{x}}(i)$, the corresponding error is given by

$$e_{i|N}^{RMS} = \sqrt{\frac{1}{N}\sum_{j=j_0}^{n}\left[\mathbf{x}_j(i) - \hat{\mathbf{x}}_j(i)\right]^2} \tag{6.7}$$

The number of data points used in the calculation of the RMS error is $N = n - j_0 + 1$, where n represents the total number of data points collected in the simulation, and j_0 represents the starting point for the RMS calculation. Since the initial portion of the EKF estimation requires a period of convergence to the true states, the starting point of the RMS calculation for the analyses here consider only the data from the second orbit of the two-orbit simulations; this implies that the EKF has converged to the true state trajectory after one orbital period.

For these analyses, the RMS errors are calculated for the relative position and relative velocity terms separately, and the mean of the RMS errors are obtained to demonstrate the estimation error expected along a single axis. The averaged RMS

errors are calculated using

$$\bar{e}_r^{RMS} = \text{Average Position RMS} = \frac{1}{3}\left(e_{x|N}^{RMS} + e_{y|N}^{RMS} + e_{z|N}^{RMS}\right) \tag{6.8}$$

$$\bar{e}_v^{RMS} = \text{Average Velocity RMS} = \frac{1}{3}\left(e_{\dot{x}|N}^{RMS} + e_{\dot{y}|N}^{RMS} + e_{\dot{z}|N}^{RMS}\right) \tag{6.9}$$

The last metric used to compare the performances of the proposed EKF algorithms is the three-dimensional Root-Mean-Square (3D-RMS) error. As an alternate to the averaged RMS errors, the 3D-RMS errors quantify a spatial distribution of the errors. Since the 3D-RMS metric combines the error components in quadrature, 3D-RMS values are inherently larger than the single-dimensional RMS values. The position and velocity 3D-RMS errors are calculated as shown below, where it is important to note that x, y and z used here indicate the specific LVLH states:

$$\text{Position 3D-RMS} = \sqrt{\frac{1}{N}\sum_{j=j_0}^{n}\left[(x_j - \hat{x}_j)^2 + (y_j - \hat{y}_j)^2 + (z_j - \hat{z}_j)^2\right]} \tag{6.10}$$

$$\text{Velocity 3D-RMS} = \sqrt{\frac{1}{N}\sum_{j=j_0}^{n}\left[\left(\dot{x}_j - \hat{\dot{x}}_j\right)^2 + \left(\dot{y}_j - \hat{\dot{y}}_j\right)^2 + \left(\dot{z}_j - \hat{\dot{z}}_j\right)^2\right]} \tag{6.11}$$

In summary, the estimation errors are the key indicator of the accuracy of the EKF. Throughout the case studies presented in this chapter, time histories of the estimated states are plotted, along with the corresponding estimation errors and covariance bounds. Histograms of the estimation errors are also given, to illustrate the distributions of the errors from the various filters. Lastly, tables containing the RMS and 3D-RMS errors are also included, to summarize and reinforce the overall performance of each EKF.

The following sections contain the analyses of the three spacecraft formation scenarios. First, the PRISMA formation will be considered, followed by the PROBA-3 formation, and lastly the PEO formation. Each section presents the relevant orbital parameters, spacecraft characteristics, and Kalman filter settings, along with the resulting data analysis. Although conclusions on the filter performances will be drawn with respect to each formation scenario, final conclusions regarding the combined observations of the three scenarios are given at the end of this chapter.

6.2 Case 1: PRISMA Formation

The PRISMA formation consists of two spacecraft, the target (named Tango) and the chaser (named Mango). Details of the orbital characteristics and spacecraft properties can be found in Tables 2.2 and 2.3 from Chapter 2.7. The PRISMA formation operates in LEO, which implies high absolute spacecraft velocities and a relatively short orbital period of 99 minutes. As shown previously in Fig. 2.9, the chaser spacecraft follows a bounded relative orbit about the target, offset in the radial direction. The separation between spacecraft is on the order of 120 m, which means the acceptable level of error in the position estimates should be on the order of one meter.

6.2.1 PRISMA Simulation Conditions

The top-level simulator settings for the PRISMA case are summarized in Table 6.1, and the corresponding initial conditions for the EKF are listed on the following page in Table 6.2. The simulation duration is twice the orbital period of the target spacecraft, and a sampling frequency of 1 Hz is used for processing the measurements. Random zero-mean Gaussian noise is added to the measurements based on RMS accuracy specifications of the NovAtel FlexPak6 GNSS Receiver[1]. The standard deviations for the noise processes added to each component of the position and velocity measurements are $\sigma_r = 1.2$ m and $\sigma_v = 0.03$ m/s, respectively. It is critical to reiterate these noises are added to measurements in the inertial frame, not the LVLH frame; consequently the noise characteristics of the filter can not be expected to directly match the added noise, due to the transformation between the ECI and LVLH frames.

Table 6.1: Simulator Settings for the PRISMA Formation

Options	Settings	
Simulation Duration	3 hrs, 18 mins	(11 876 s)
Orbital Perturbations	J_2, Luni-Solar Third Body, Drag, SRP	
Spacecraft Formation	PRISMA (Mango and Tango)	
Measurement Noises	$\sigma_r = 1.2$ m	$\sigma_v = 0.03$ m/s
Measurement Sampling	$T = 1.0$ s	($f = 1$ Hz)

[1]https://www.novatel.com/assets/Documents/Papers/FlexPak6.pdf

Table 6.2: Initial EKF Settings for the PRISMA Formation

Initial States $\mathbf{x}_0$	$x_0 = -54.72$ m	$\dot{x}_0 = 210.73$ mm/s
	$y_0 = -86.82$ m	$\dot{y}_0 = 71.70$ mm/s
	$z_0 = 45.99$ m	$\dot{z}_0 = 83.19$ mm/s
	$\theta_0 = 358.9$ deg	$\dot{\theta}_0 = 0.0608$ deg/s
	$r_{t_0} = 7\ 077.04$ km	$\dot{r}_{t_0} = -0.21$ m/s
Initial State Error Covariance $\mathbf{P}_0$	$\sigma_x^2 = 100$ m^2	$\sigma_{\dot{x}}^2 = 1$ m^2/s^2
	$\sigma_y^2 = 100$ m^2	$\sigma_{\dot{y}}^2 = 1$ m^2/s^2
	$\sigma_z^2 = 100$ m^2	$\sigma_{\dot{z}}^2 = 1$ m^2/s^2
	$\sigma_\theta^2 = 1$ deg^2	$\sigma_{\dot{\theta}}^2 = 0.01$ deg^2/s^2
	$\sigma_{r_t}^2 = 10\ 000$ m^2	$\sigma_{\dot{r}_t}^2 = 100$ m^2/s^2
Initial Process Noise Covariance $\mathbf{Q}_0$	$q_x = 0.2$ m^2	$q_{\dot{x}} = 5 \times 10^{-3}$ m^2/s^2
	$q_y = 0.2$ m^2	$q_{\dot{y}} = 5 \times 10^{-3}$ m^2/s^2
	$q_z = 0.2$ m^2	$q_{\dot{z}} = 5 \times 10^{-3}$ m^2/s^2
	$q_\theta = 1 \times 10^{-3}$ deg^2	$q_{\dot{\theta}} = 1 \times 10^{-6}$ deg^2/s^2
	$q_{r_t} = 5 \times 10^{-3}$ m^2	$q_{\dot{r}_t} = 5 \times 10^{-5}$ m^2/s^2
Initial Measurement Noise Covariance $\mathbf{R}_0$	$r_x = 20$ m^2	$r_{\dot{x}} = 0.5$ m^2/s^2
	$r_y = 20$ m^2	$r_{\dot{y}} = 0.5$ m^2/s^2
	$r_z = 20$ m^2	$r_{\dot{z}} = 0.5$ m^2/s^2
	$r_\theta = 0.01$ deg^2	

Within the filter, the relative position states of the formation are initialized to values that constitute initial state errors of approximately 3% of the amplitude of the respective state. For the MLE-AEKF, the moving-window size was selected as $N = 30$ to provide reasonable smoothing performance without excessively straining the computational requirements. Similarly in the FAEKF, the averaging of residuals uses the past $N = 30$ observations. This value of N was selected empirically, based on knowledge that N must be large enough to smooth the data but small enough capture the dynamical environment throughout the spacecraft orbit (*i.e.*, if a large value of N encompasses an entire orbit, the periodic nature of the relative dynamics would be averaged to roughly zero, while a very small N provides negligible smoothing). Selecting the same value of N for both the MLE-AEKF and FAEKF also provides reasonable grounds for comparing the computational costs of each method.

Rather than implementing the PSDS check within the covariance updates, simply restricting the adaptations of the covariances to the diagonal elements was found to provide superior numerical performance. The settings for the MLE-EKF and the FAEKF are presented in Table 6.3 and Table 6.4, respectively.

Table 6.3: MLE-AEKF Settings for the PRISMA Formation

Options	Settings	
MLE Smoothing	$N = 30$	
Nearest PSDS Check	$Q = \text{OFF}$	$R = \text{OFF}$
Diagonalized Adaptations	$Q = \text{ON}$	$R = \text{ON}$

Table 6.4: FAEKF Settings for the PRISMA Formation

Options	Settings	
Residual Averaging	$N = 30$	
Input Fuzzy Gains	$g^q = -5 \times 10^{-3}$	$h^q = 1 \times 10^{-3}$
Output Fuzzy Gains	$g_1^r = 5 \times 10^{-2}$	$h_1^r = 1 \times 10^{-4}$
	$g_2^r = 5 \times 10^{-2}$	$h_2^r = 1 \times 10^{-4}$
	$g_3^r = 5 \times 10^{-2}$	$h_3^r = 1 \times 10^{-4}$
	$g_4^r = 1 \times 10^{1}$	$h_4^r = 1 \times 10^{-4}$
	$g_5^r = 1 \times 10^{0}$	$h_5^r = 1 \times 10^{-3}$
	$g_6^r = 1 \times 10^{0}$	$h_6^r = 1 \times 10^{-3}$
	$g_7^r = 1 \times 10^{0}$	$h_7^r = 1 \times 10^{-3}$

6.2.2 PRISMA Simulation Results

The resulting average RMS errors for the PRISMA simulations are presented in Table 6.5, along with the relative runtime for each variant of the EKF. The relative runtimes are calculated with respect to the non-adaptive EKF, and the costs of the adaptation schemes agree with the expected trends: the MLE-AEKF is more complex than the FAEKF and the standard EKF by over a factor of 2, due to the storing and processing required by the smoothing routine and update laws within the MLE algorithm. The R-MLE adaptations in Eq. (4.54) have a larger number of calculations than the Q-MLE adaptations in Eq. (4.53), which is reflected in the larger runtime for the R-MLE scenario. Not surprisingly, attempting to estimate both noise covariances at the same time leads to the QR-MLE-AEKF displaying the highest level of computational complexity. The FAEKF is less computationally demanding than the MLE-AEKF, with the R-FAEKF being more costly than the Q-FAEKF. Since the R-FAEKF involves processing seven inputs through the FLS and the Q-FAEKF requires only one, the R-FAEKF understandably requires a longer runtime.

More interestingly however, are the estimation errors of the filtering algorithms. First note that the standard EKF is entirely capable of providing better position and velocity estimates than the measurements alone. Furthermore, adaptations of the process noise matrix by both the MLE-AEKF and the FAEKF improve the estimates of the EKF, yielding position estimation errors less than 30% those of the EKF, and velocity estimation errors less than 1% those of the EKF. The following figures present the state estimates and estimation errors of the proposed filtering algorithms. For brevity, only plots of the EKF, the Q-MLE-AEKF, and the Q-FAEKF are shown, since Q-adaptations provided the best overall performance in this case study, and the EKF provides a baseline for comparison. Histograms of the error distributions are also given, and relevant comments on each plot are included in the corresponding captions.

Table 6.5: Average RMS Errors and Runtimes from PRISMA Simulation

Method	Position (cm)	Velocity (cm/s)	Relative Runtime
Measurements	168.36	4.18	N/A
Standard EKF	48.52	2.12	1.00
Q-MLE-AEKF	13.68	0.05	2.29
R-MLE-AEKF	60.40	3.92	2.47
QR-MLE-AEKF	26.83	0.32	2.96
Q-FAEKF	14.37	0.05	1.25
R-FAEKF	48.95	1.76	1.29
QR-FAEKF	34.69	0.99	1.30

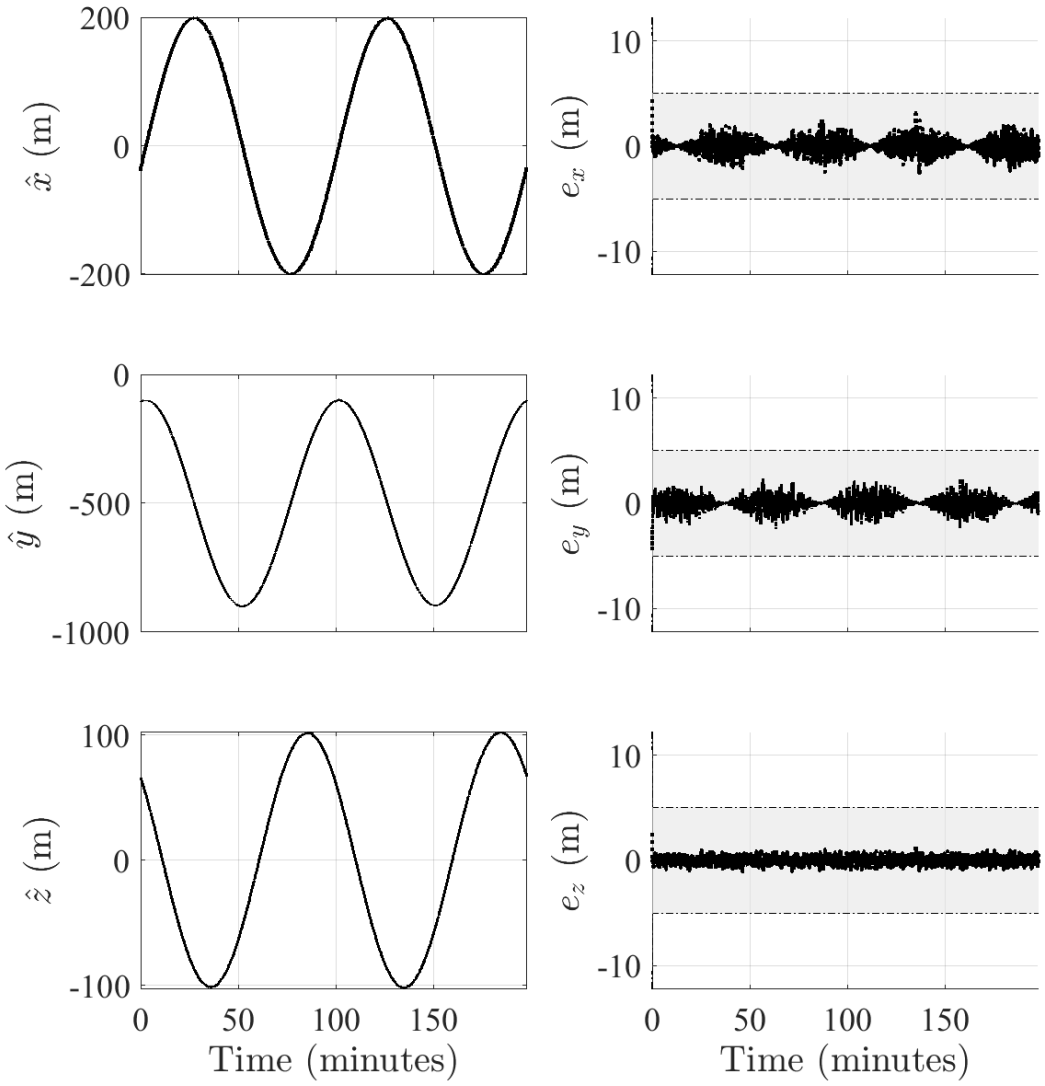


Figure 6.1: EKF relative position estimation results for the PRISMA formation. (Left) The estimated states. (Right) The estimation errors and 3σ covariance bounds. Here a reasonable level of accuracy is demonstrated by the EKF, as the estimates have converged to the true trajectory and the majority of the errors are within the predicted covariance bounds. This represents the baseline performance of the EKF.

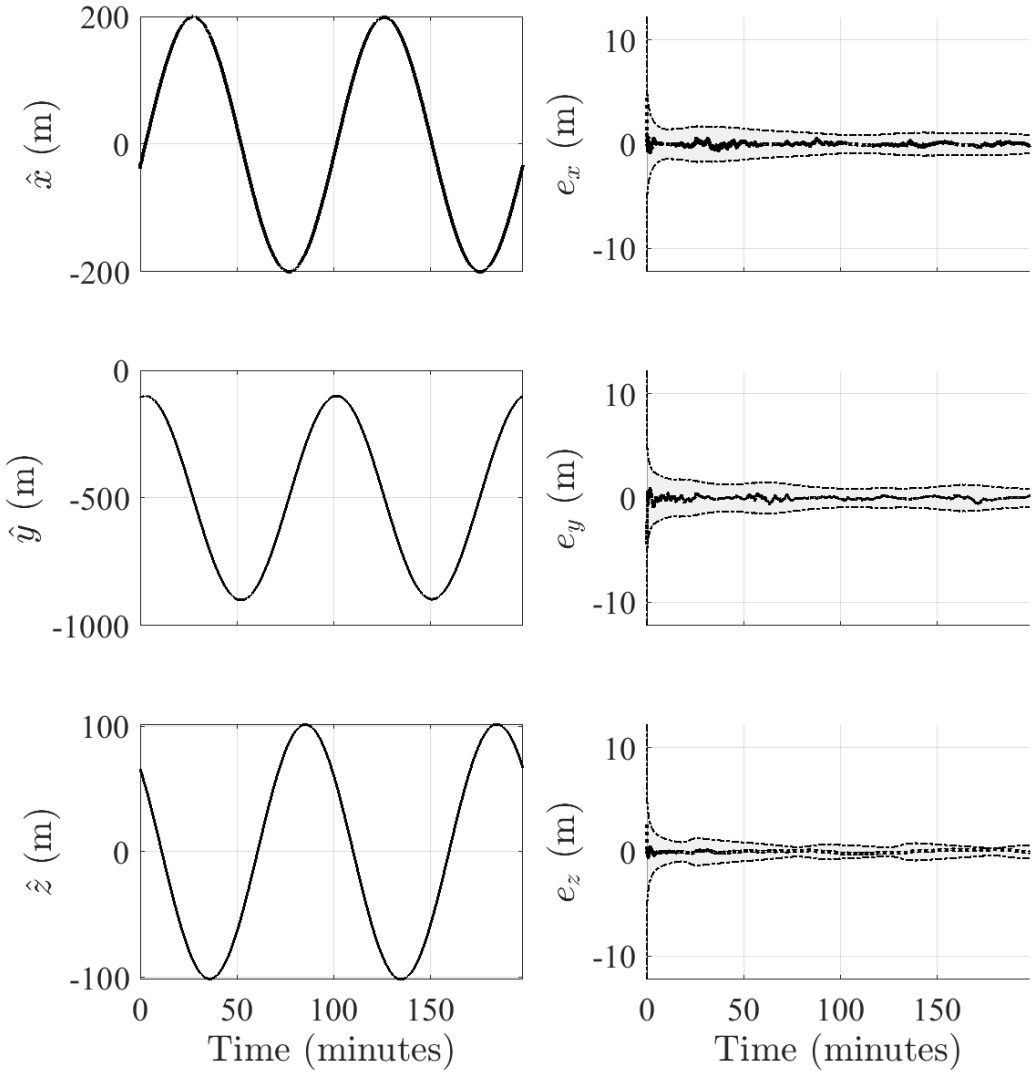


Figure 6.2: MLE-AEKF relative position estimation results for the PRISMA formation using Q-adaptations. (Left) The estimated states. (Right) The estimation errors and 3σ covariance bounds. The Q-MLE-AEKF provides the best overall estimation performance for the PRISMA scenario, in the context of average RMS errors. The estimation errors and covariance bounds are clearly lower for the Q-MLE-AEKF than those of the EKF.

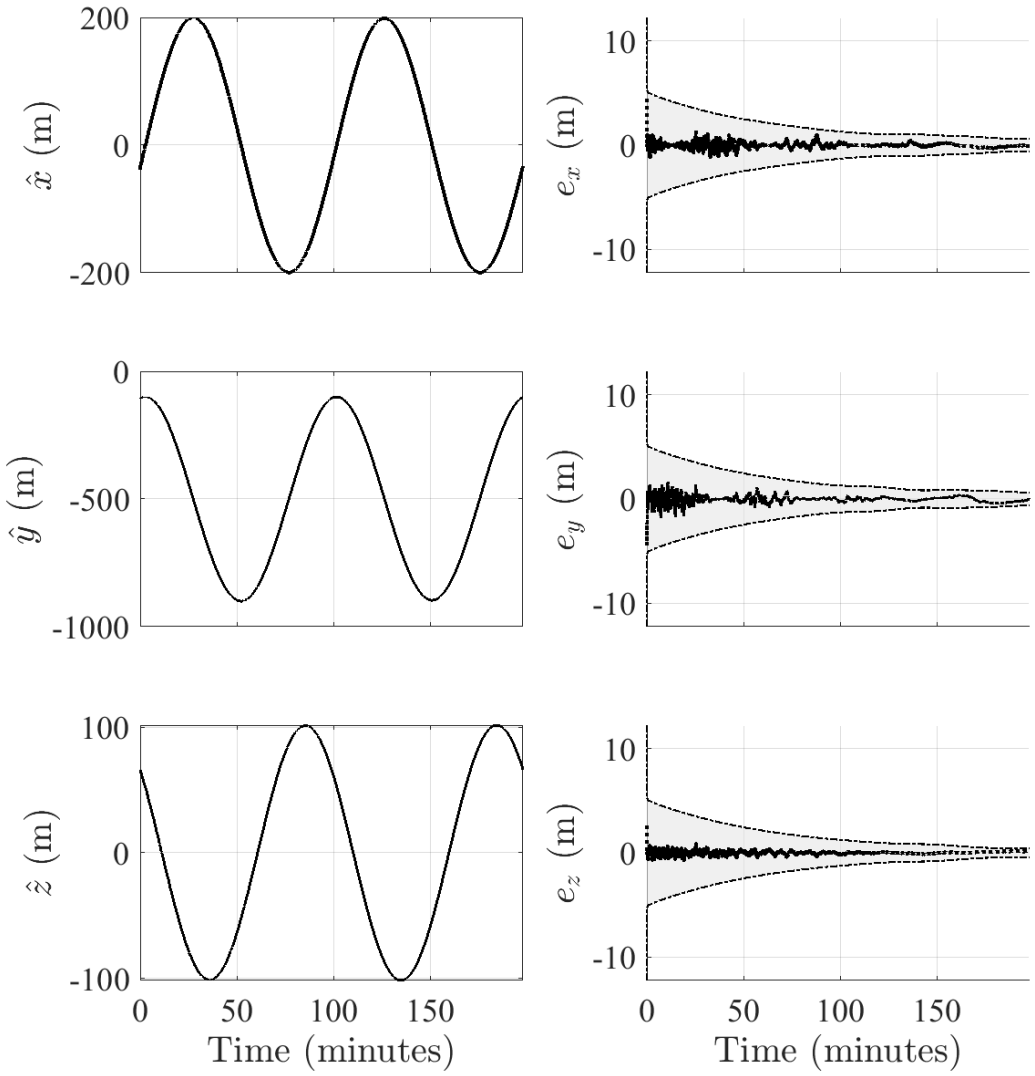


Figure 6.3: FAEKF relative position estimation results for the PRISMA formation using Q-adaptations. (Left) The estimated states. (Right) The estimation errors and 3σ covariance bounds. The Q-FAEKF similarly yields smaller estimation errors than those of the EKF, however the covariance bounds here show that the adaptations of the fuzzy system take longer to adjust the process noise covariance matrix. Observing the estimation errors within the covariance bounds, it can also be seen that the magnitudes of the errors reduce throughout the simulation duration, due to the adaptations.

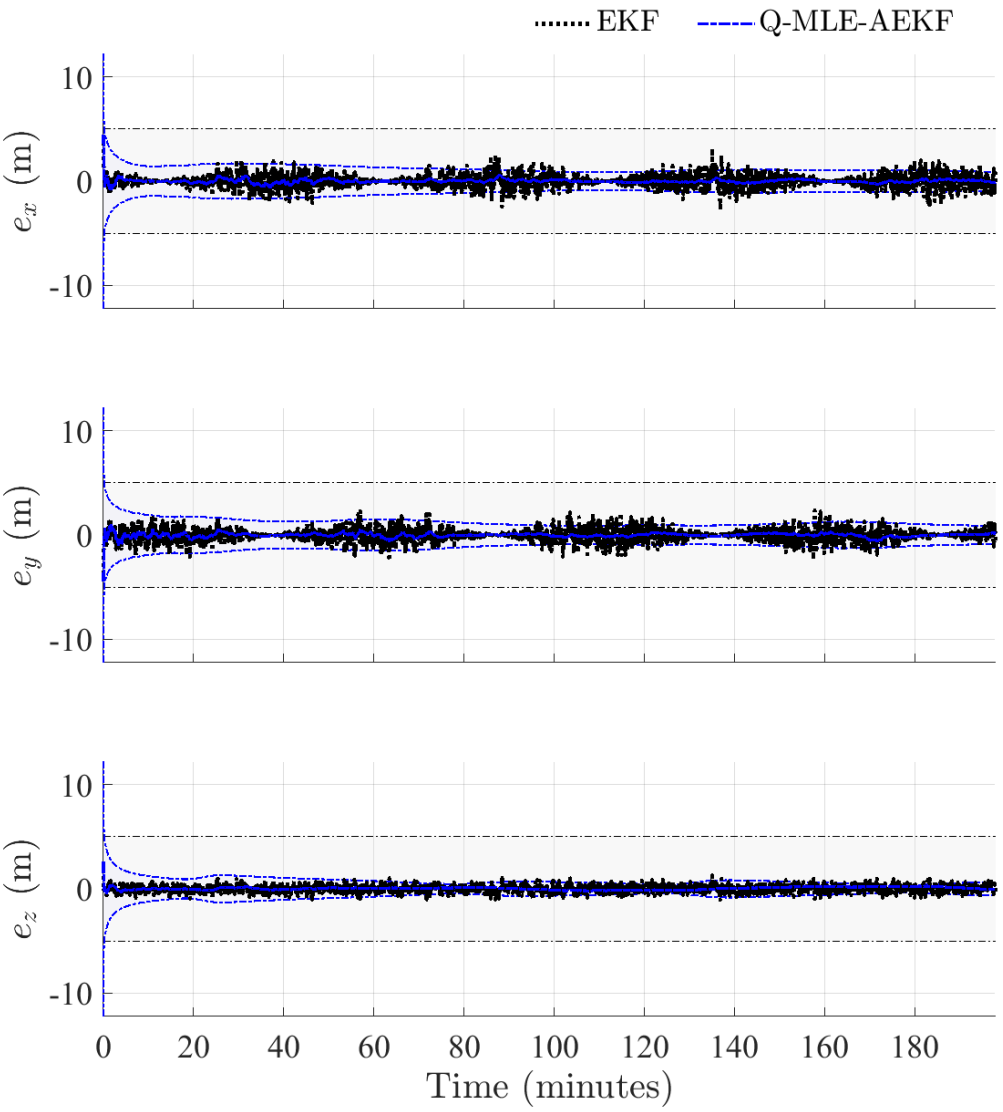


Figure 6.4: Comparison of relative position estimates for the PRISMA Formation, between the EKF and the MLE-AEKF with Q-adaptations. The estimation errors and 3σ covariance bounds of the Q-MLE-AEKF are smaller than those of the EKF, as a result of the adaptations made by the MLE algorithm. A portion of the measurement noise is still corrupting the estimates provide by the EKF, while the Q-MLE-AEKF is placing more weight on the dyanmics model within the filter, thereby smoothing the state estimates and reducing the estimation error.

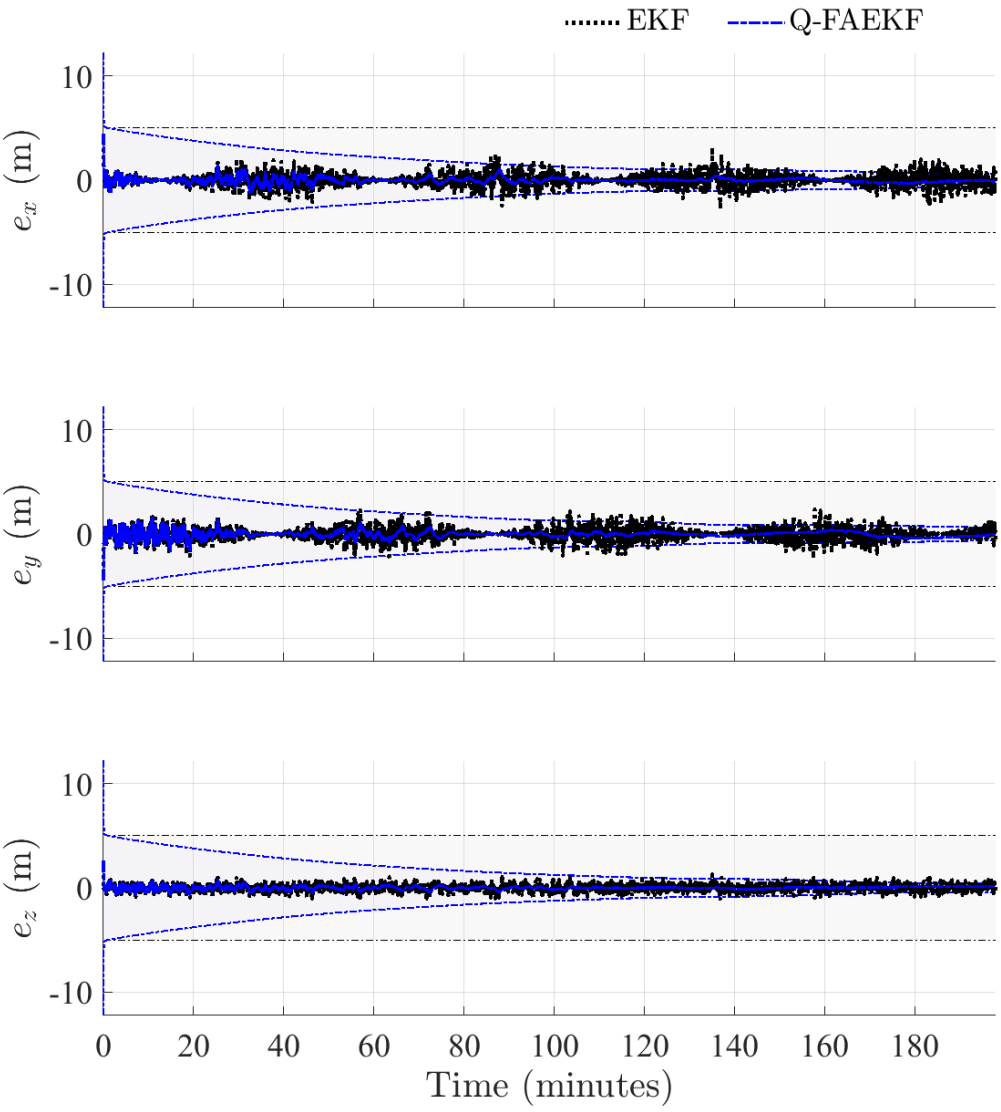


Figure 6.5: Comparison of relative position estimates for the PRISMA formation, between the EKF and the FAEKF with Q-adaptations. Again the gradual improvement of the Q-FAEKF estimate are shown by the decreasing estimation errors and 3σ covariance bounds shown here. Much like the Q-MLE-AEKF, the FAEKF is able to improve the accuracy of the position estimates by modifying the process noise covariance matrix to account for the dynamical errors in the filter model.

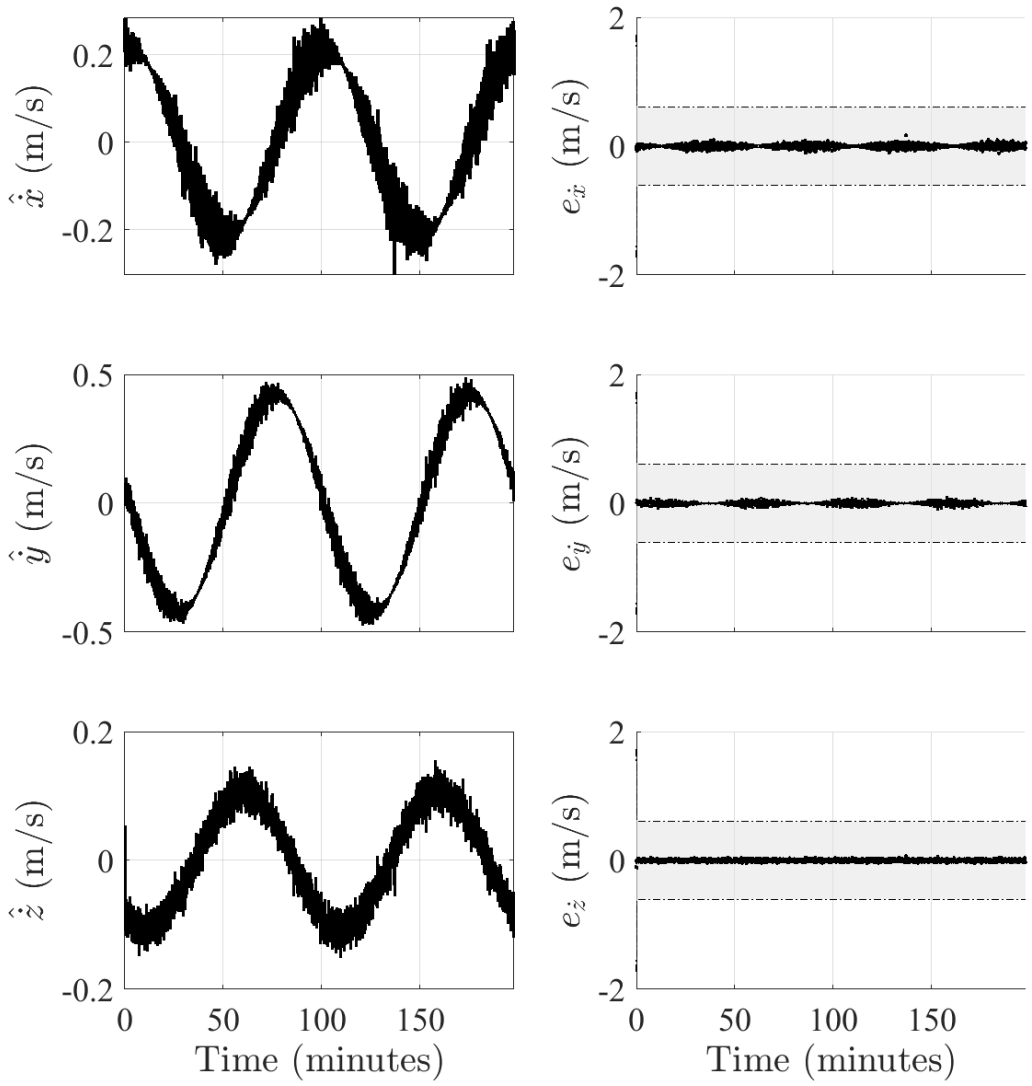


Figure 6.6: EKF relative velocity estimation results for the PRISMA formation. (Left) The estimated states. (Right) The estimation errors and 3σ covariance bounds. The velocity estimates from the EKF show a considerable amount of noise present, which permeates the filter through the measurements. Convergence to the true states has been achieved however, and the state errors are well within the covariance bounds.

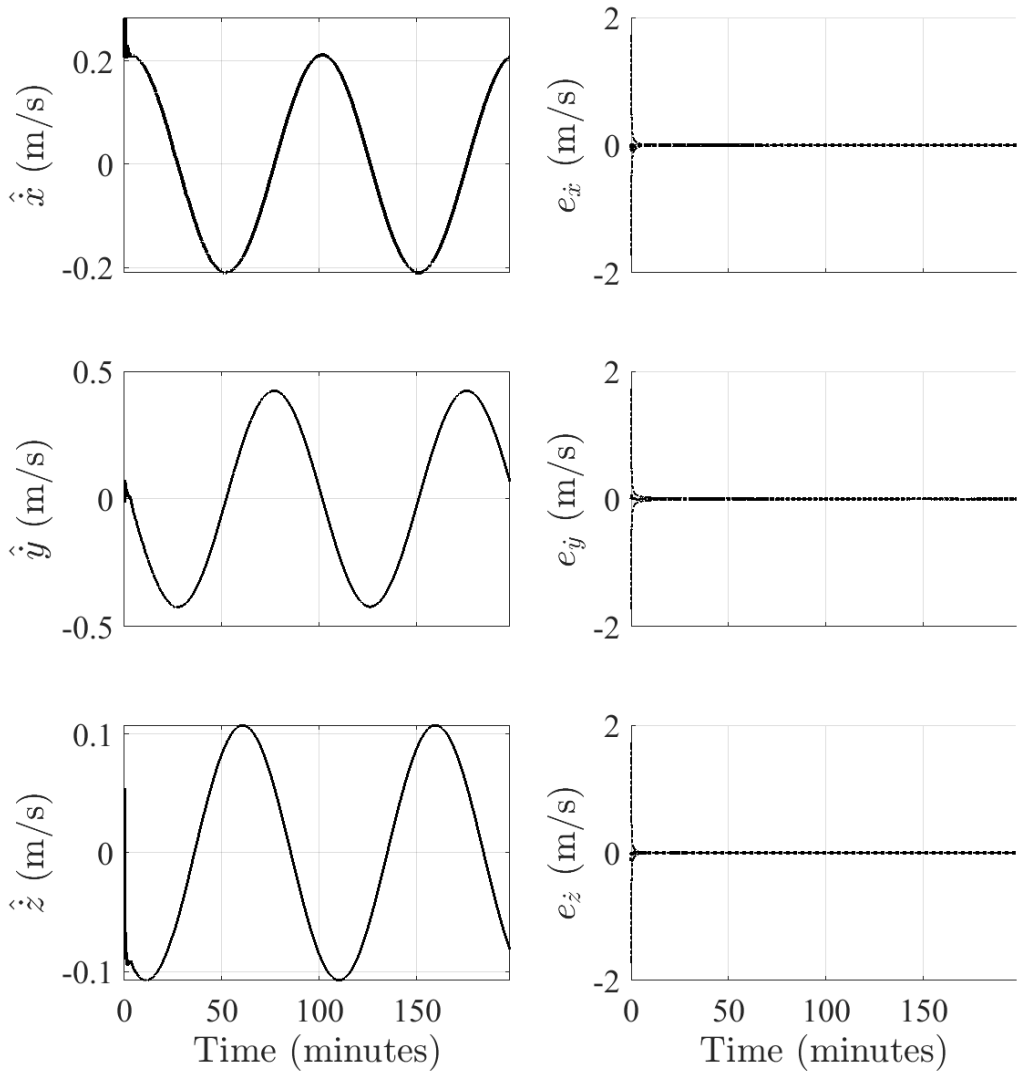


Figure 6.7: MLE-AEKF relative velocity estimation results for the PRISMA formation using Q-adaptations. (Left) The estimated states. (Right) The estimation errors and 3σ covariance bounds. The Q-adaptations have significantly improved the estimation results of the relative velocities, as indicated by the near-vanishing covariance bounds and near-zero estimation errors. (Note that the scale of the estimation error plots on all figures is identical, allowing for a visual comparison.) The initial estimation errors are also visible, where the states begin at an offset value and rapidly converge to the true state trajectories.

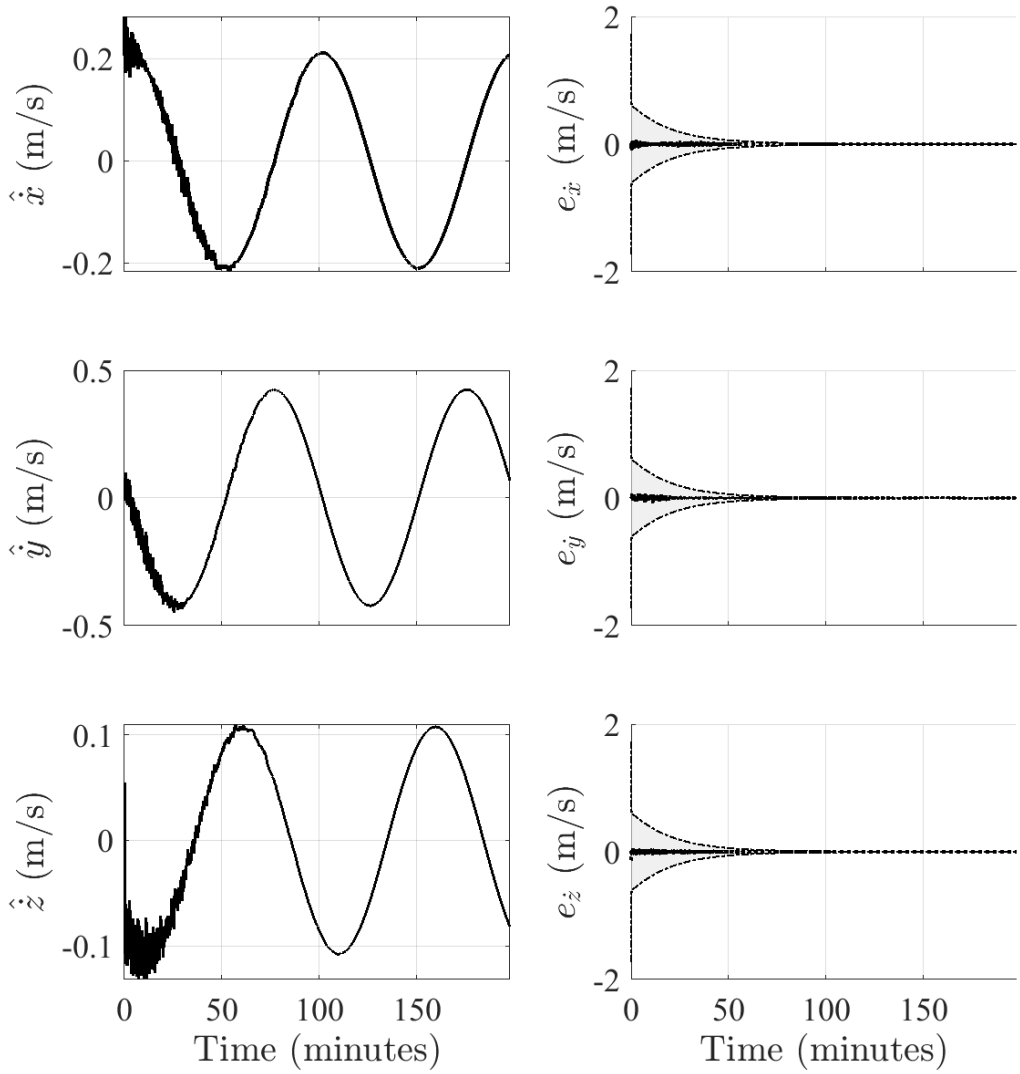


Figure 6.8: FAEKF relative velocity estimation results for the PRISMA formation using Q-adaptations. (Left) The estimated states. (Right) The estimation errors and 3σ covariance bounds. Implementing Q-adaptations via the FAEKF provides better velocity state estimates than using the EKF alone, as the estimation errors shown above are decrease as more data is processed.

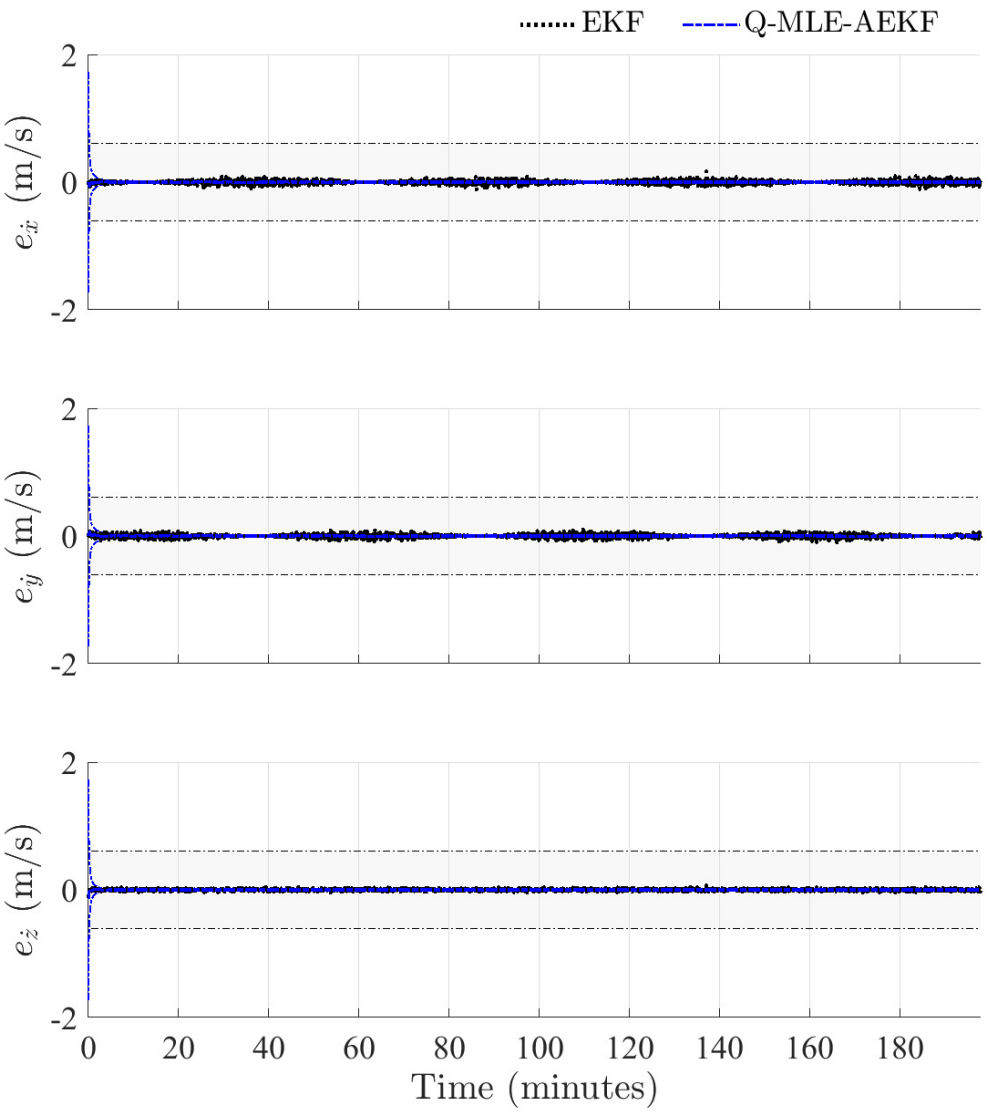


Figure 6.9: Comparison of relative velocity estimates for the PRISMA formation, between the EKF and the MLE-AEKF with Q-adaptations. Directly comparing the estimation errors and 3σ covariance bounds shows how quickly the MLE adaptation laws are able to improve the performance of the EKF. Much of the measurement noise that can be seen in the EKF radial and along-track estimation errors are reduced in the case of the Q-MLE-AEKF, and the error covariances are accordingly much lower.

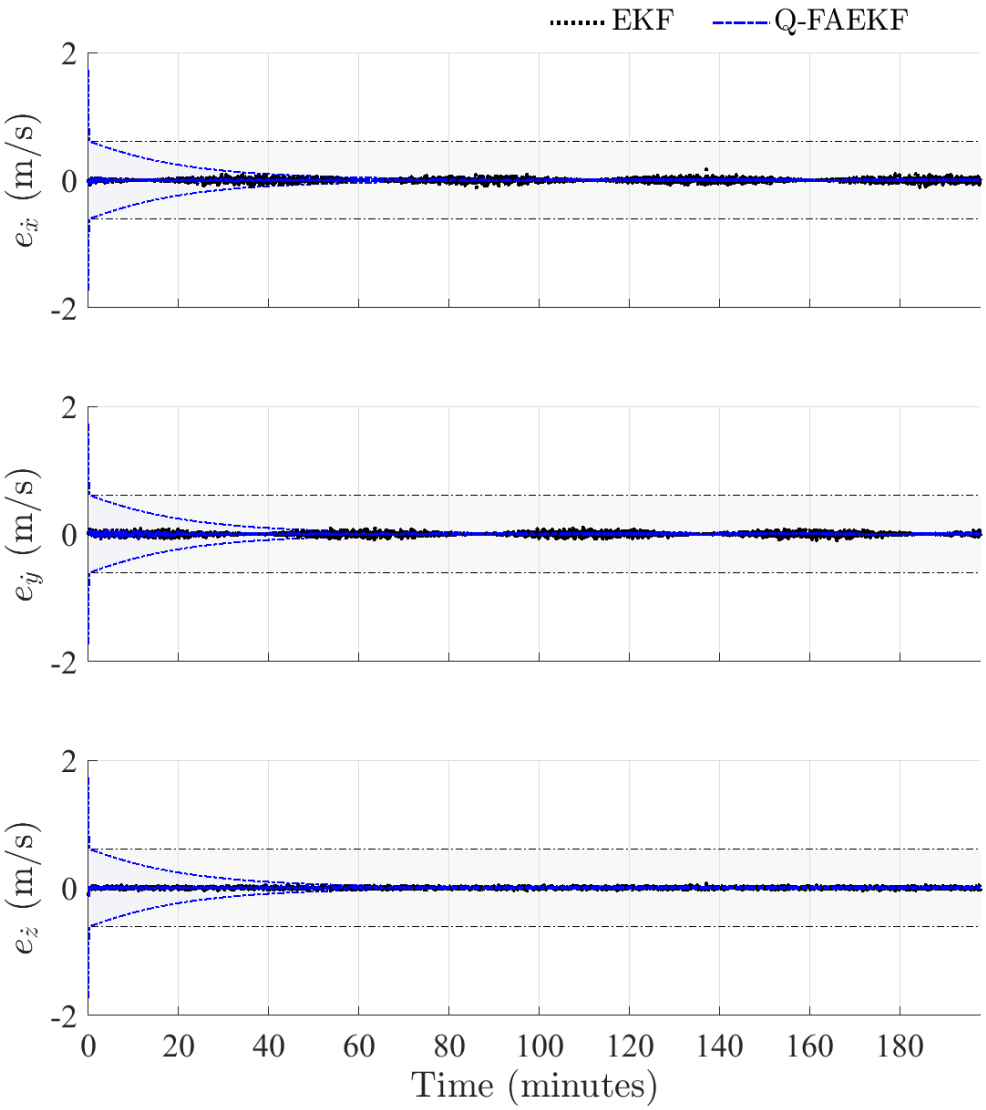


Figure 6.10: Comparison of relative velocity estimates for the PRISMA formation, between the EKF and the FAEKF with Q-adaptations. The estimation errors and 3σ covariance bounds shown here support the previous deductions regarding the Q-FAEKF; although the adaptation process is slower than the Q-MLE-AEKF, the Q-FAEKF also provides velocity estimates that are more accurate than the EKF.

Figure 6.11 provides a set of histograms comparing the position estimation error distributions for the PRISMA scenario. The first row compares the EKF and the GPS measurements, and shows that the estimation errors of the EKF are more frequently distributed at smaller values than those of the measurements (*i.e.*, the EKF improves the position estimation accuracy relative to the GPS measurements).

The second row compares the Q-MLE-AEKF and the EKF, where the adaptive EKF markedly outperforms the standard EKF. The third row compares the Q-FAEKF and the Q-MLE-AEKF, where it is more difficult to determine which method provides the best estimation accuracy. It was established from the average RMS errors that the Q-MLE-AEKF does indeed provide estimation errors that are 6.9 mm less than those of the Q-FAEKF, so the third row here is effectually demonstrating that both methods provide a comparable level of estimation accuracy.

As a final note, see that the range of the error distributions (the horizontal axes) decreases from along the first, second, and third rows. This indicates that the overall estimation accuracy increases when reading down the rows.

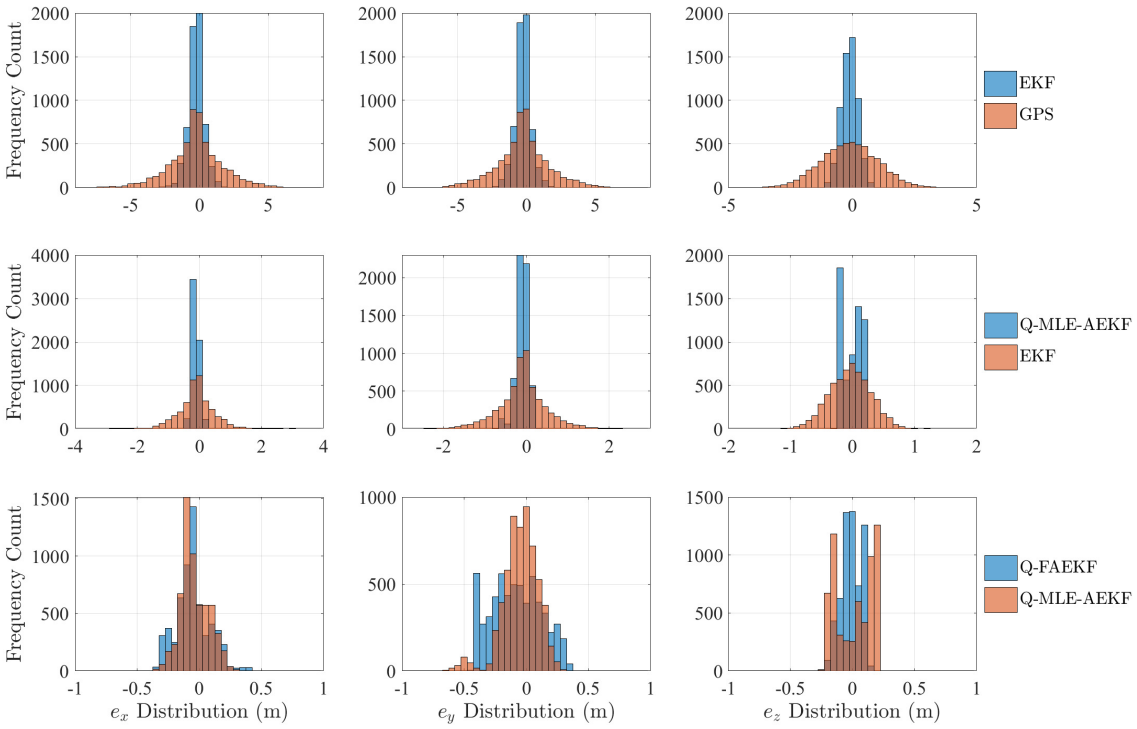


Figure 6.11: Relative position error distributions for the PRISMA formation.

Velocity estimation error distributions are compiled in Fig. 6.12, and these histograms visually establish the reduced error magnitudes from the adaptive EKF simulations. The standard EKF and the measurement relative velocity errors are given in the first row, where the tight clustering of the EKF error distribution demonstrates the improvements provided by the filter.

The Q-MLE-AEKF and the EKF estimation errors are given in the second row, with the Q-MLE-AEKF clearly reducing the magnitude of the velocity estimation errors. The Q-FAEKF and Q-MLE-AEKF estimation errors are grouped in the last row. As with the position errors, directly comparing the velocity error distributions for the two successful adaptation schemes primarily illustrates that their estimation accuracies are nearly equivalent.

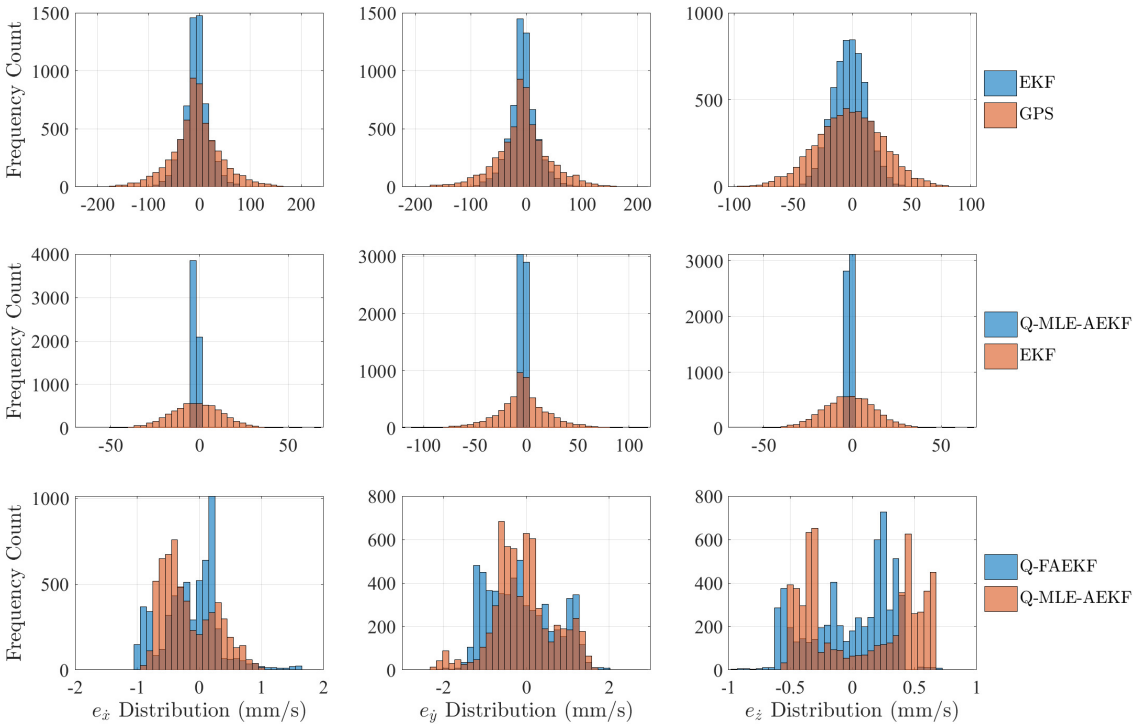


Figure 6.12: Relative velocity error distributions for the PRISMA formation.

6.2.3 PRISMA Simulation Conclusions

The array of figures shown for the simulations of the PRISMA formation have demonstrated that both the Q-MLE-AEKF and the Q-FAEKF provide the best estimation performance out of the proposed algorithms, for this particular formation configuration. The average RMS values supported these results, as do the 3D-RMS values shown below in Table 6.6. Both methods outperform the standard EKF, and although both provide comparable accuracy in the relative velocity estimates, the position estimates using the MLE adaptations are marginally more accurate than the estimates from the FAEKF adaptations. This additional accuracy comes at the cost of additional computational time however, with the Q-MLE-AEKF requiring 1.8 times more processing time than the Q-FAEKF. Lastly, the position estimation errors obtained in this scenario are well within the predefined one-meter accuracy objective.

To clarify why the Q-adaptation routines provide the best results in this scenario, consider the dynamical and measurement environments of the system tested here: in Low Earth Orbit, the nonlinear effects of perturbations are significant and dominate the errors due to the measurements. The altitude range spanned by the formation means that the spacecraft are subjected to varying degrees of non-conservative forces from atmospheric drag, and similarly the effects of J_2 will be prevalent at these altitudes. These effects are unmodelled by the EKF's dynamics model, so by adapting the process noise covariance matrix of the EKF, a more appropriate weighting between the dynamics and the measurements of the filter can be obtained using Q-adaptations.

Table 6.6: 3D-RMS Errors from PRISMA Simulation

Method	Position (cm) [% Error]*	Velocity (cm/s) [% Error]*	Relative Runtime
Measurements	299.10 [2.67%]	7.42 [32.43%]	N/A
Standard EKF	86.20 [0.77%]	3.76 [16.44%]	1.00
Q-MLE-AEKF	23.89 [0.21%]	0.10 [0.44%]	2.29
R-MLE-AEKF	105.11 [0.94%]	6.91 [30.21%]	2.47
QR-MLE-AEKF	48.60 [0.43%]	0.63 [2.75%]	2.96
Q-FAEKF	26.46 [0.24%]	0.10 [0.44%]	1.25
R-FAEKF	86.85 [0.77%]	3.12 [13.63%]	1.29
QR-FAEKF	61.58 [0.55%]	1.76 [7.70%]	1.30

*The position and velocity percent errors are calculated relative to the minimum distance and speed between the spacecraft, respectively.
Note: Qualitatively the 3D-RMS errors present the same overall performance conclusions observable from the average RMS errors seen at the beginning of the analysis, however they quantify the spatial resolution of the estimation as opposed to the resolution in a single axis.

6.3 Case 2: PROBA-3 Formation

The scientific platform of the PROBA-3 formation consists of a coronograph spacecraft and an occulter spacecraft, constituting the target and chaser, respectively. In practice, the occulter actively controls its motion to ensure the coronograph is shielded from the Sun during observation periods, but passive motion of both spacecraft is assumed for this simulation. The specific parameters of the formation orbit and spacecraft characteristic were presented in Chapter 2.7, and can be referenced from Tables 2.4 and 2.5. PROBA-3 is in a highly elliptic orbit, which an orbital period on the order of 20 hours. Since the relative motion is unbounded for this formation, the smallest separation between spacecraft occurs at the beginning of the simulation and is roughly 200 m. A desirable level of position estimation accuracy for the PROBA-3 scenario is therefore on the order of two meters.

6.3.1 PROBA-3 Simulation Conditions

Table 6.7 contains the settings for the PROBA-3 simulations, and Table 6.8 gives the state of the initialized EKF. The simulation duration is again twice the orbital period of the target spacecraft, and the sampling frequency and measurement noises are identical to the previous case study. Given the highly eccentric orbit and the resulting variation in the perturbations and dynamics that will be experienced by the spacecraft, the initial tuning of the process noise covariance matrix has been increased.

Table 6.7: Simulator Settings for the PROBA-3 Formation

Options	Settings
Simulation Duration	39 hrs, 15 mins $\quad (141\ 331$ s)
Orbital Perturbations	J_2, Luni-Solar Third Body, Drag, SRP
Spacecraft Formation	PROBA-3 (Coronograph and Occulter)
Measurement Noises	$\sigma_r = 1.2$ m $\qquad \sigma_v = 0.03$ m/s
Measurement Sampling	$T = 1.0$ s $\qquad (f = 1$ Hz)

Table 6.8: Initial EKF Settings for the PROBA-3 Formation

Initial States $\mathbf{x}_0$	$x_0 = -204.72$ m	$\dot{x}_0 = 10.00$ mm/s
	$y_0 = 20.00$ m	$\dot{y}_0 = 407.86$ mm/s
	$z_0 = -20.00$ m	$\dot{z}_0 = 10.0$ mm/s
	$\theta_0 = 0$ deg	$\dot{\theta}_0 = 0.0835$ deg/s
	$r_{t_0} = 6\ 978.58$ km	$\dot{r}_{t_0} = 0.00$ m/s
Initial State Error Covariance $\mathbf{P}_0$	$\sigma_x^2 = 100$ m^2	$\sigma_{\dot{x}}^2 = 1$ m^2/s^2
	$\sigma_y^2 = 100$ m^2	$\sigma_{\dot{y}}^2 = 1$ m^2/s^2
	$\sigma_z^2 = 100$ m^2	$\sigma_{\dot{z}}^2 = 1$ m^2/s^2
	$\sigma_\theta^2 = 1$ deg^2	$\sigma_{\dot{\theta}}^2 = 0.01$ deg^2/s^2
	$\sigma_{r_t}^2 = 10\ 000$ m^2	$\sigma_{\dot{r}_t}^2 = 100$ m^2/s^2
Initial Process Noise Covariance $\mathbf{Q}_0$	$q_x = 2.0$ m^2	$q_{\dot{x}} = 5 \times 10^{-2}$ m^2/s^2
	$q_y = 2.0$ m^2	$q_{\dot{y}} = 5 \times 10^{-2}$ m^2/s^2
	$q_z = 2.0$ m^2	$q_{\dot{z}} = 5 \times 10^{-2}$ m^2/s^2
	$q_\theta = 1 \times 10^{-2}$ deg^2	$q_{\dot{\theta}} = 1 \times 10^{-5}$ deg^2/s^2
	$q_{r_t} = 5 \times 10^{-2}$ m^2	$q_{\dot{r}_t} = 5 \times 10^{-4}$ m^2/s^2
Initial Measurement Noise Covariance $\mathbf{R}_0$	$r_x = 20$ m^2	$r_{\dot{x}} = 0.5$ m^2/s^2
	$r_y = 20$ m^2	$r_{\dot{y}} = 0.5$ m^2/s^2
	$r_z = 20$ m^2	$r_{\dot{z}} = 0.5$ m^2/s^2
	$r_\theta = 0.01$ deg^2	

Initial position estimates within the Kalman filter are offset from the true values by 20 m in each direction, and the velocity estimates are similarly offset by 20 mm/s in each direction. The moving-window sizes for both the MLE-AEKF and the FAEKF algorithms are again selected as $N = 30$. Table 6.15 contains the settings used for the MLE-AEKF, and Table 6.16 contains the settings for the FAEKF, as they were implemented in the PROBA-3 formation simulations. Note that the scaling gains in the FAEKF were re-tuned from the nominal settings used in the PRISMA scenario.

Table 6.9: MLE-AEKF Settings for the PROBA-3 Formation

Options	Settings	
MLE Smoothing	$N = 30$	
Nearest PSDS Check	$Q =$ OFF	$R =$ OFF
Diagonalized Adaptations	$Q =$ ON	$R =$ ON

Table 6.10: FAEKF Settings for the PROBA-3 Formation

Options	Settings	
Residual Averaging	$N = 30$	
Input Fuzzy Gains	$g^q = -3 \times 10^{-3}$	$h^q = 1 \times 10^{-2}$
Output Fuzzy Gains	$g_1^r = 1 \times 10^{-2}$	$h_1^r = 5 \times 10^{-5}$
	$g_2^r = 1 \times 10^{-2}$	$h_2^r = 5 \times 10^{-5}$
	$g_3^r = 1 \times 10^{-2}$	$h_3^r = 5 \times 10^{-5}$
	$g_4^r = 5 \times 10^{2}$	$h_4^r = 5 \times 10^{-5}$
	$g_5^r = 5 \times 10^{-1}$	$h_5^r = 5 \times 10^{-4}$
	$g_6^r = 5 \times 10^{-1}$	$h_6^r = 5 \times 10^{-4}$
	$g_7^r = 5 \times 10^{-1}$	$h_7^r = 5 \times 10^{-4}$

6.3.2 PROBA-3 Simulation Results

Table 6.11 summaries the average RMS errors for each Kalman filter variant from the PROBA-3 simulations. Even for this case of a highly eccentric orbit, the standard EKF yields relative position estimation errors at the sub-meter level, and relative velocity errors on the scale of 20 mm/s. The MLE-AEKF and FAEKF are able to improve the performance of the EKF, however there is an interesting disparity between which adaptation scheme is ideal; In the case of the MLE-AEKF, adapting both the process and measurement noise covariances provides the most accurate position estimates with an average RMS of 20.32 cm, while adapting only the process noise covariance is the most accurate FAEKF method, with an average RMS of 40.27 cm. Another interesting result is that the Q-adaptation methods for both the MLE-AEKF and FAEKF provide better relative position estimates than the QR-adaptations. This unexpected result is likely due to the initial tunings of the covariance matrices, and the differences in the relative weightings between the position and velocity covariance elements. Nevertheless, variants of both the MLE-AEKF and the FAEKF demonstrate better estimation of the relative formation states than the classical EKF.

Relative runtimes for the various Kalman filter scenarios follow the trends identified in the PRISMA test case, namely that the QR-adaptations are the most computationally expensive, followed by the R-adaptations and then the Q-adaptations. The MLE-AEKF simulations are also more demanding than the FAEKF trials, which agrees with observations from Case 1. The following pages present state estimation time history plots and error histograms for the EKF, the QR-MLE-AEKF, and the Q-FAEKF, as these filters showcase the best position estimation accuracies in terms of average RMS errors.

Table 6.11: Average RMS Errors and Runtimes from PROBA-3 Simulation

Method	Position (cm)	Velocity (cm/s)	Relative Runtime
Measurements	162.98	4.05	N/A
Standard EKF	67.52	2.08	1.00
Q-MLE-AEKF	92.31	0.03	2.59
R-MLE-AEKF	111.64	4.05	2.74
QR-MLE-AEKF	20.32	0.13	2.78
Q-FAEKF	40.27	1.48	1.23
R-FAEKF	68.89	2.04	1.56
QR-FAEKF	42.46	1.64	1.63

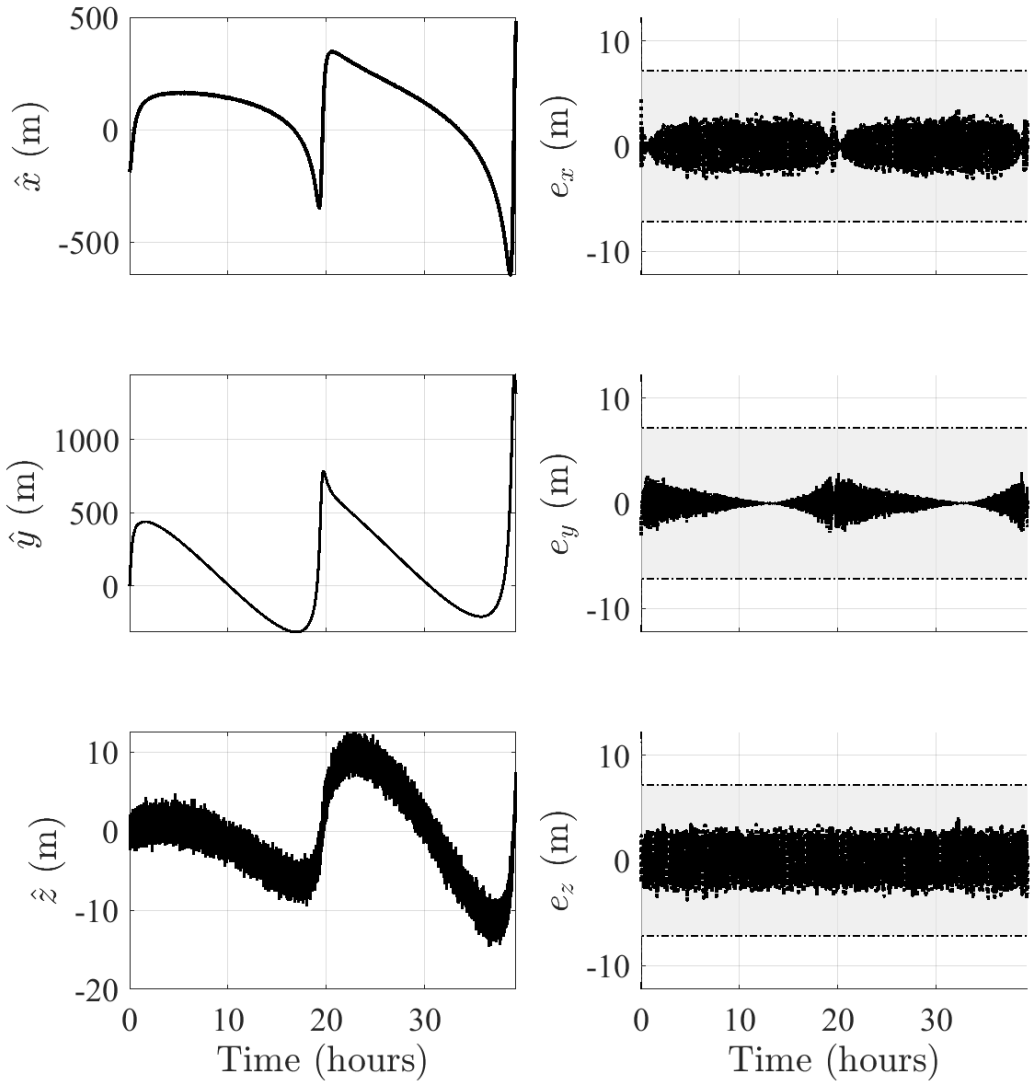


Figure 6.13: EKF relative position estimation results for the PROBA-3 formation. (Left) The estimated states. (Right) The estimation errors and 3σ covariance bounds. While the state estimates converge to the true trajectories and the estimation errors do remain bounded within the filter covariances, characteristics of the measurement noise still remain in the estimated states.

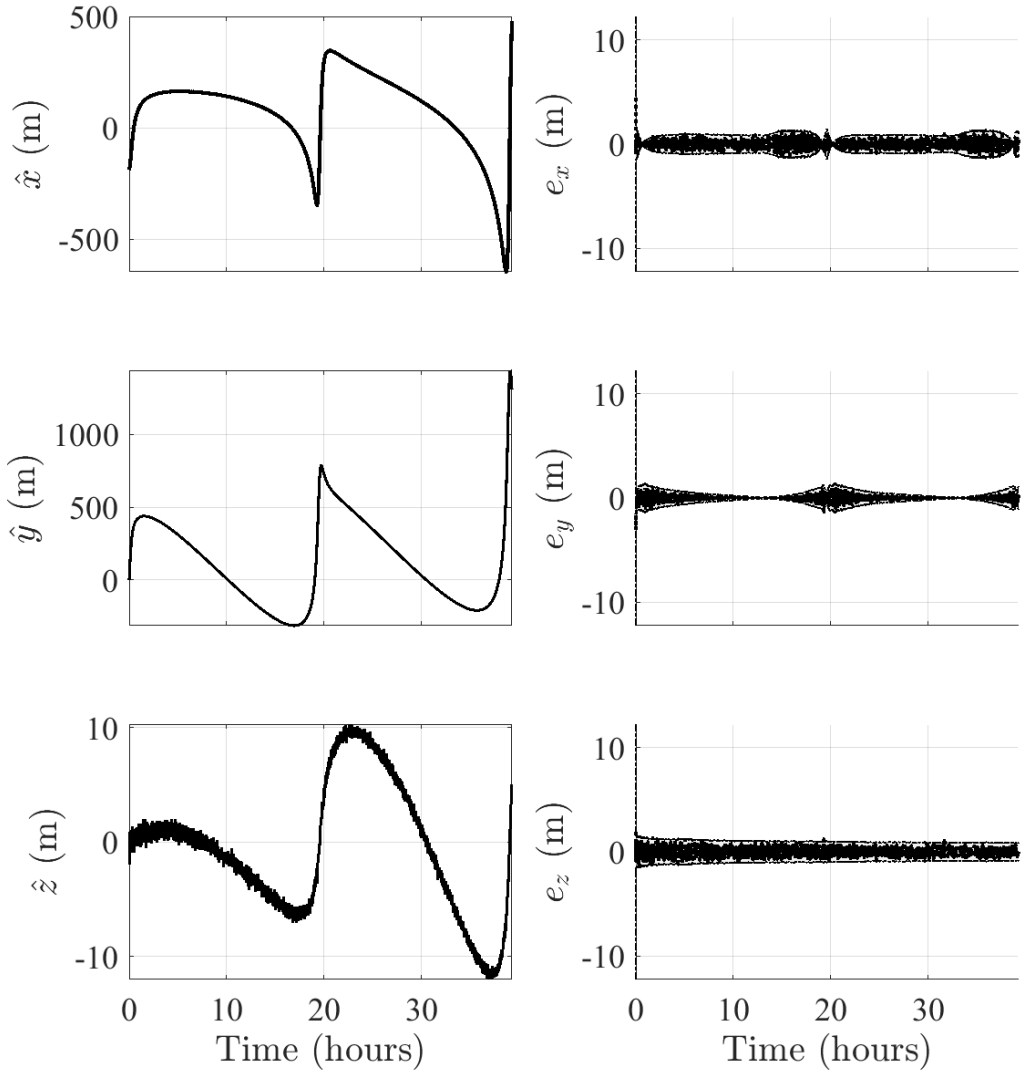


Figure 6.14: MLE-AEKF relative position estimation results for the PROBA-3 formation using QR-adaptations. (Left) The estimated states. (Right) The estimation errors and 3σ covariance bounds. Adapting both the process and measurement noise covariance matrices within the QR-MLE-AEKF leads to lower errors in the position estimates than those given by the EKF.

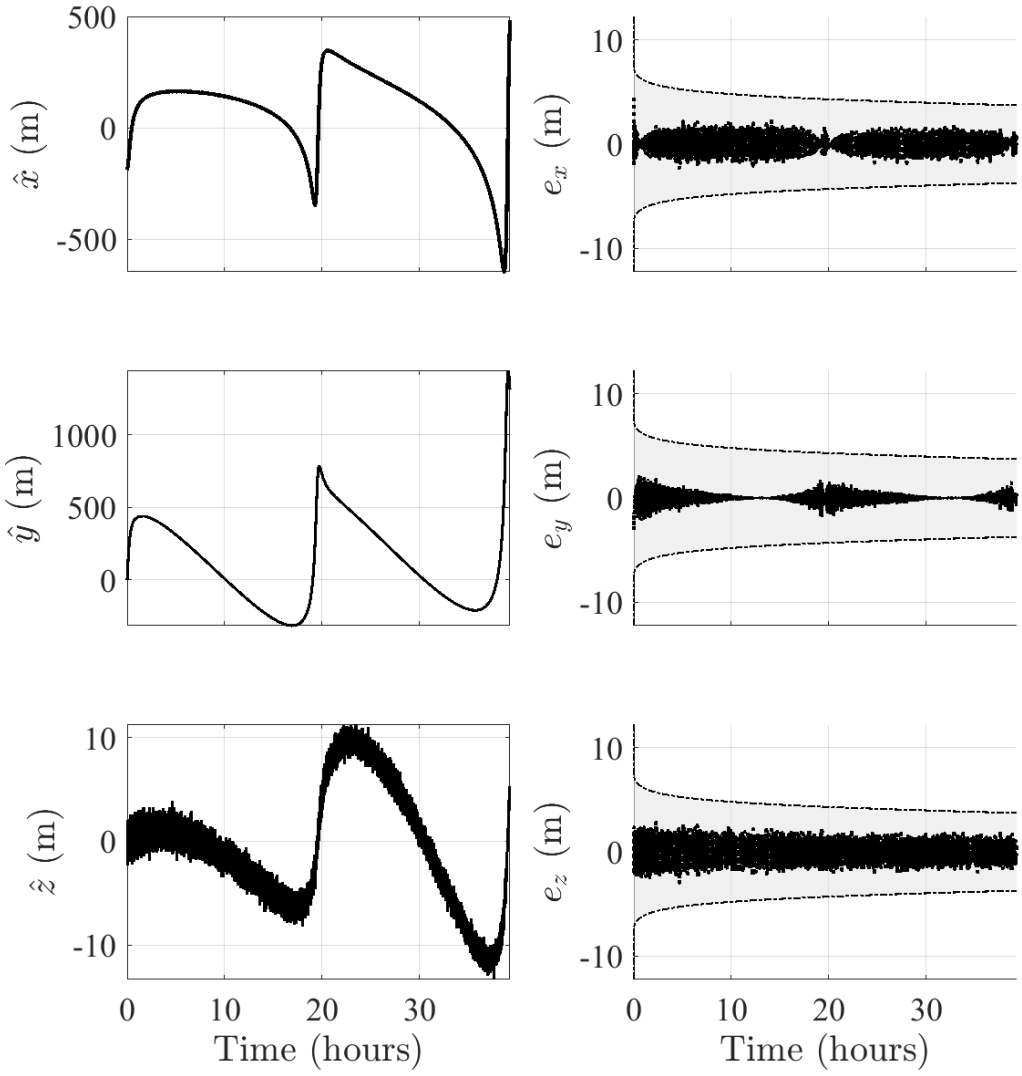


Figure 6.15: FAEKF relative position estimation results for the PROBA-3 formation using Q-adaptations. (Left) The estimated states. (Right) The estimation errors and 3σ covariance bounds. Estimates from the FAEKF quickly converge to the true states, and the estimation errors through the simulation are smaller than those of the EKF. The adaptations occur at a slower rate than those of the MLE-AEKF, as a reflection of the selected tuning of the FLS.

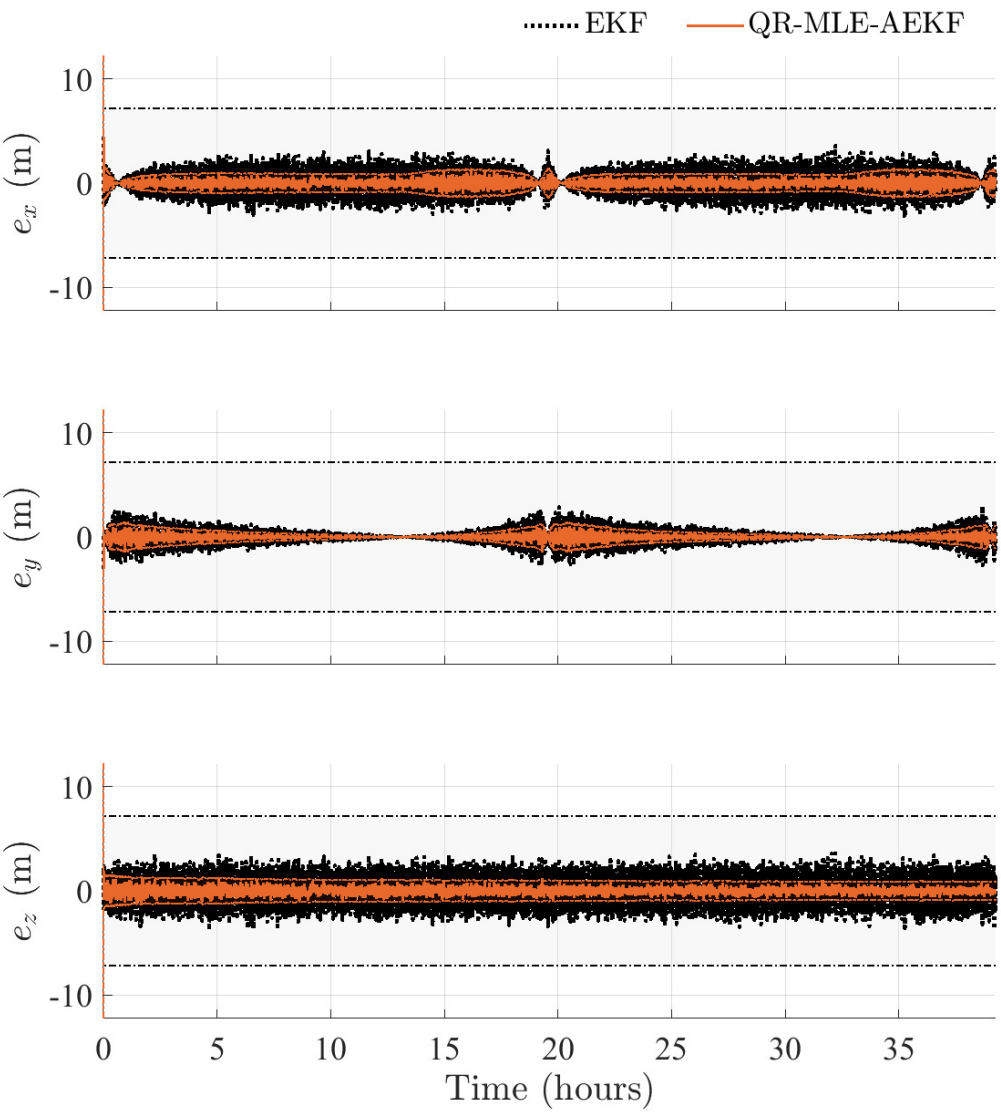


Figure 6.16: Comparison of relative position estimates for the PROBA-3 formation, between the EKF and the MLE-AEKF with QR-adaptations. This figure nicely showcases the improved performance of the QR-MLE-AEKF relative to the EKF, as the magnitudes of the estimation errors from the adaptive filter are well within the errors of the non-adaptive filter. Adapting both the process and measurement noise covariances via the MLE equations proved to be the most accurate estimation routine in the PROBA-3 case study, as the extreme changes in altitude and highly nonlinear dynamics and measurements contribute to strongly time-varying errors in the dynamics and measurement models of the EKF.

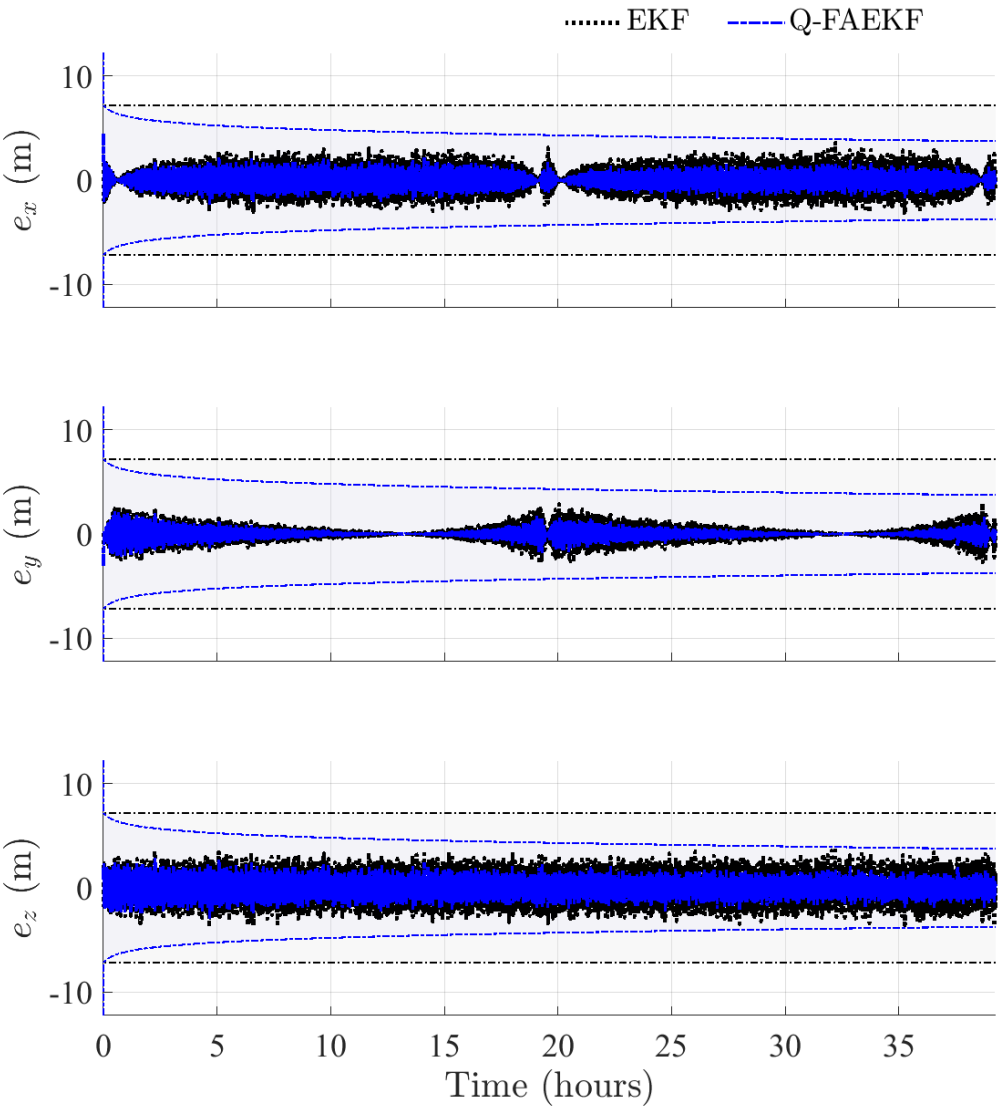


Figure 6.17: Comparison of relative position estimates for the PROBA-3 formation, between the EKF and the FAEKF with Q-adaptations. As with the MLE-AEKF, the estimation errors and 3σ covariance bounds shown here support the fact that the FAEKF is able to suitably adjust the process noise covariance of the filter, such that the position estimates are more accurate than the EKF alone.

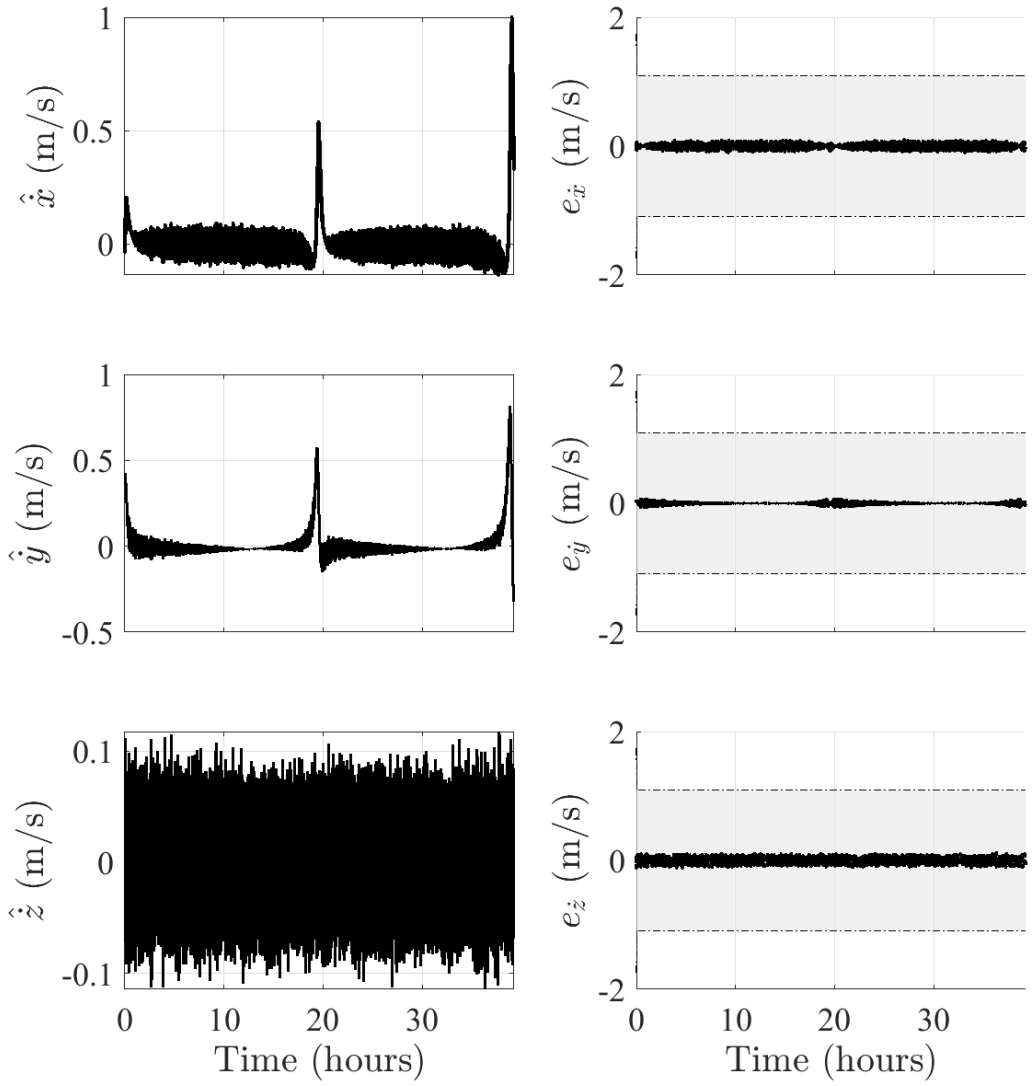


Figure 6.18: EKF relative velocity estimation results for the PROBA-3 formation. (Left) The estimated states. (Right) The estimation errors and 3σ covariance bounds. Similar conclusions can be drawn from these velocity estimates as those drawn from the position estimates; the EKF provides a convergent solution and is able to track the true states of the system, but the final estimated states still contain a high level of measurement noise.

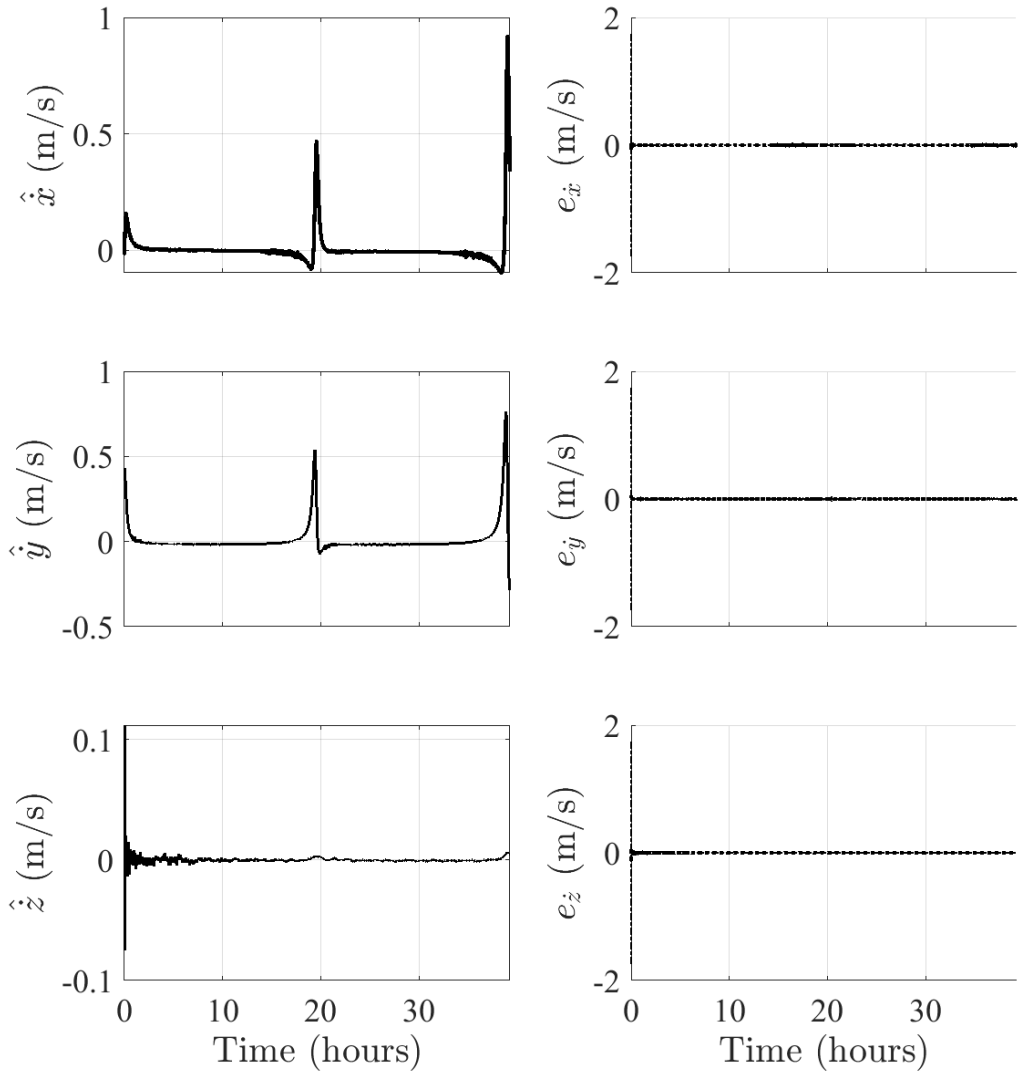


Figure 6.19: MLE-AEKF relative velocity estimation results for the PROBA-3 formation using QR-adaptations. (Left) The estimated states. (Right) The estimation errors and 3σ covariance bounds. Compared to the non-adaptive EKF, the relative velocity estimates from the MLE-AEKF are consistently closer to the true values. Noticeably in the cross-track direction (z), adapting both the process and measurement noise covariance matrices significantly reduces the amount of noise in the velocity estimate.

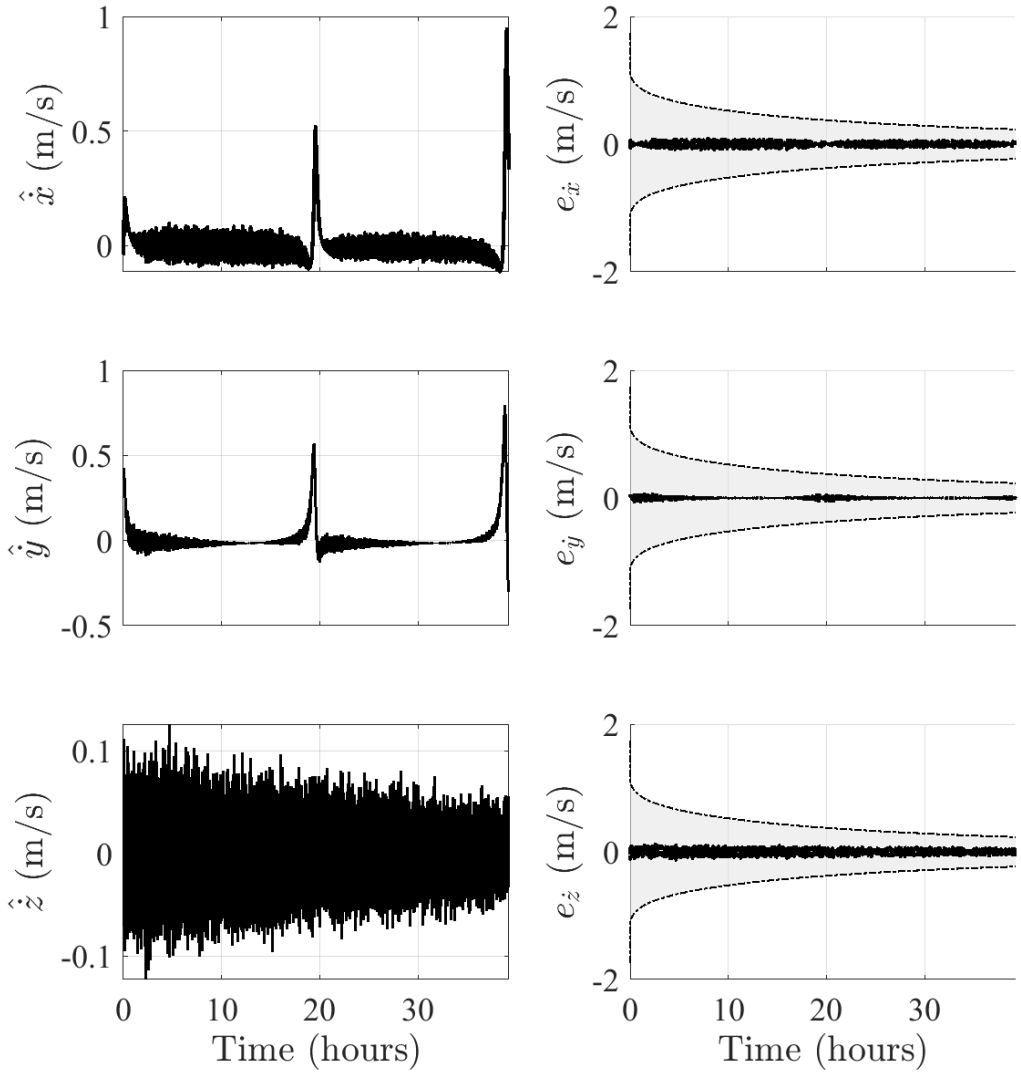


Figure 6.20: FAEKF relative velocity estimation results for the PROBA-3 formation using Q-adaptations. (Left) The estimated states. (Right) The estimation errors and 3σ covariance bounds. The gradual improvement of the velocity estimation errors is apparent in this case, mainly through observing the estimation of $\dot{z}$. However, this process is gradual, which suggests that adjusting the gains of the fuzzy system within the FAEKF could potential improve the performance further.

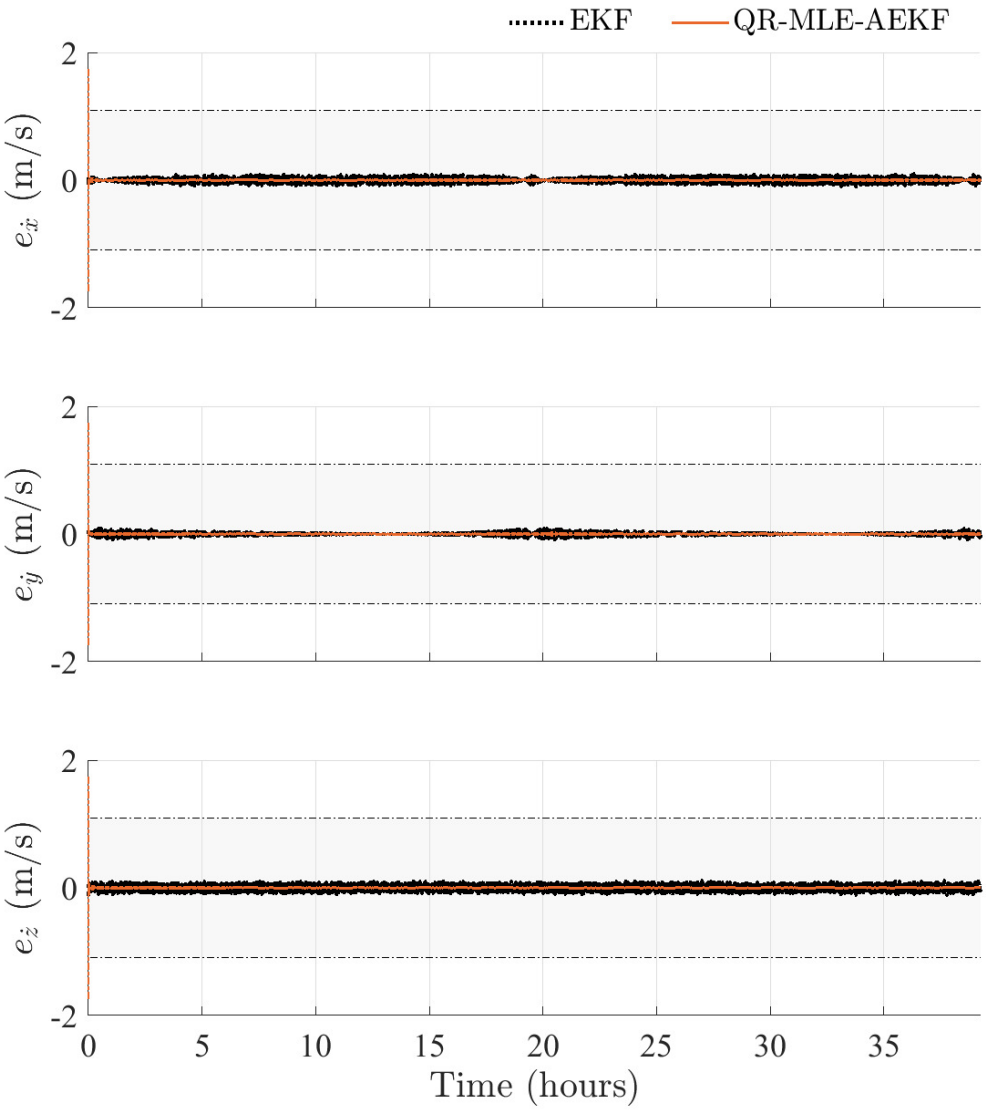


Figure 6.21: Comparison of relative velocity estimates for the PROBA-3 formation, between the EKF and the MLE-AEKF with QR-adaptations. The estimation errors and 3σ covariance bounds are shown as well. After the initial convergence of the filters, the MLE-AEKF consistently yields smaller errors in the estimates of the relative velocity states of the formation.

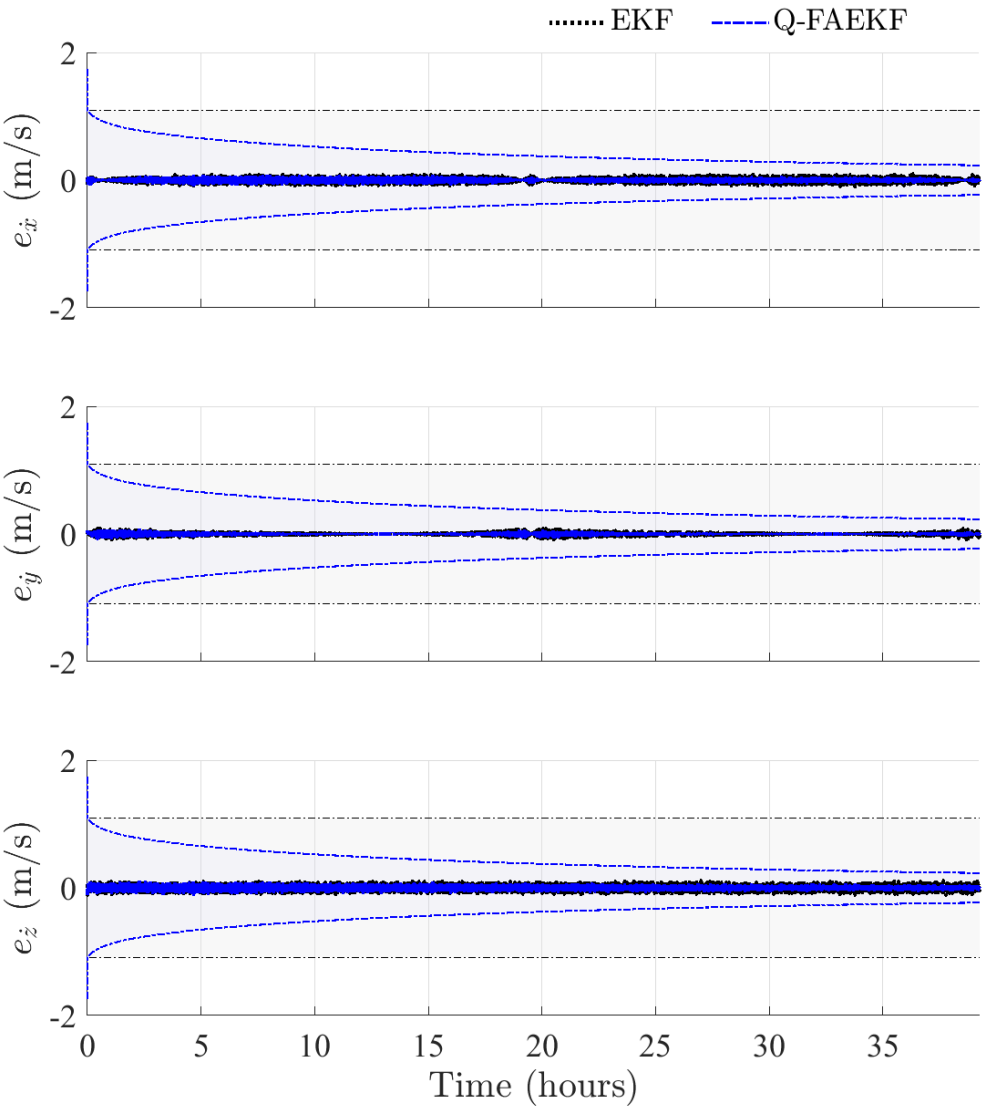


Figure 6.22: Comparison of relative velocity estimates for the PROBA-3 formation, between the EKF and the FAEKF with Q-adaptations. The estimation errors and 3σ covariance bounds show that the Q-FAEKF provides relative velocity estimates that are more accurate than the EKF, however the FAEKF results are not superior to the results from the MLE-AEKF in this scenario.

Histograms of the position estimation error distributions for the PROBA-3 case study are shown below in Fig. 6.23. The EKF and the GPS measurements are compared in the first row, where as expected, the EKF provides better position estimates than the noisy measurements it is provided.

The QR-MLE-AEKF and the EKF are compared in the second row, effectively demonstrating the reduction in error distribution of the adaptive filter. Lastly, the Q-FAEKF and the QR-MLE-AEKF are compared in the third row, where it can be realized that the MLE-based adaptation scheme consistently provides smaller errors than the Fuzzy Logic-based scheme. While the position estimation results from the MLE-AEKF and FAEKF algorithms were comparable in the PRISMA case study, for the PROBA-3 scenario here, there MLE-AEKF achieves unmistakably better estimation accuracy.

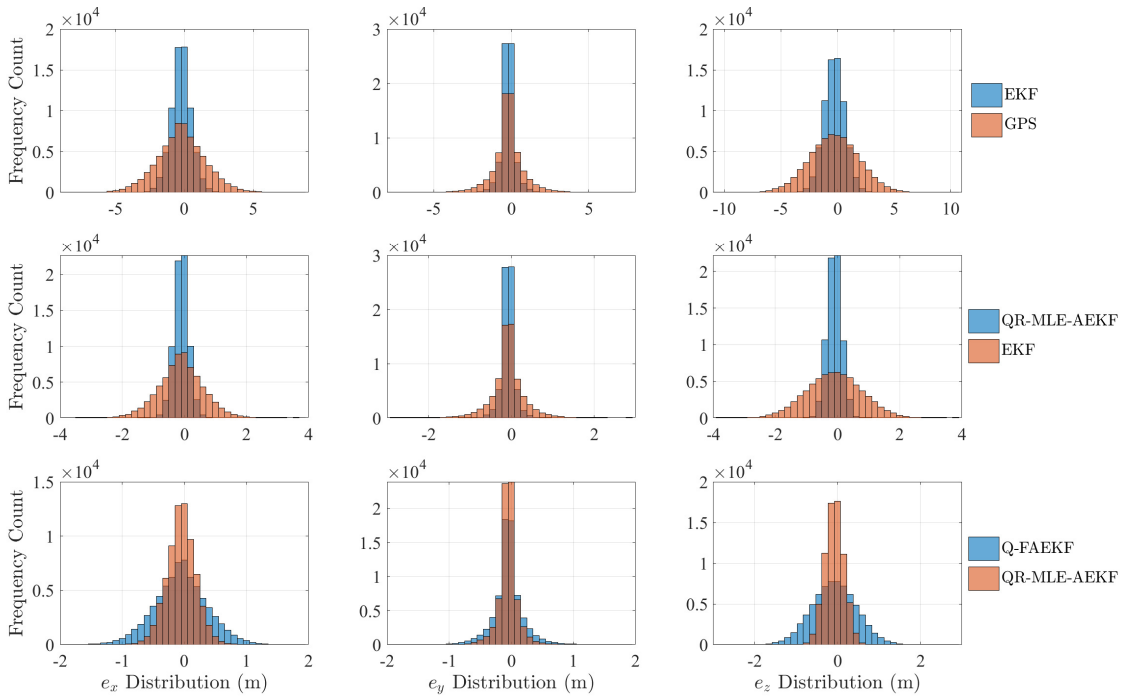


Figure 6.23: Relative position error distributions for the PROBA-3 formation.

Many of the performance conclusions obtained from the position error distributions are reinforced when examining the velocity error distributions given in Fig. 6.24. The first two rows show that the EKF velocity estimates have consistently smaller errors than the measurements alone, and the QR-MLE-AEKF estimates are significantly more accurate than the EKF estimates. The third row, which compares the Q-FAEKF and the QR-MLE-AEKF, verifies that the MLE-AEKF provides the more accurate relative velocity estimates than both the Q-FAEKF and the classical EKF.

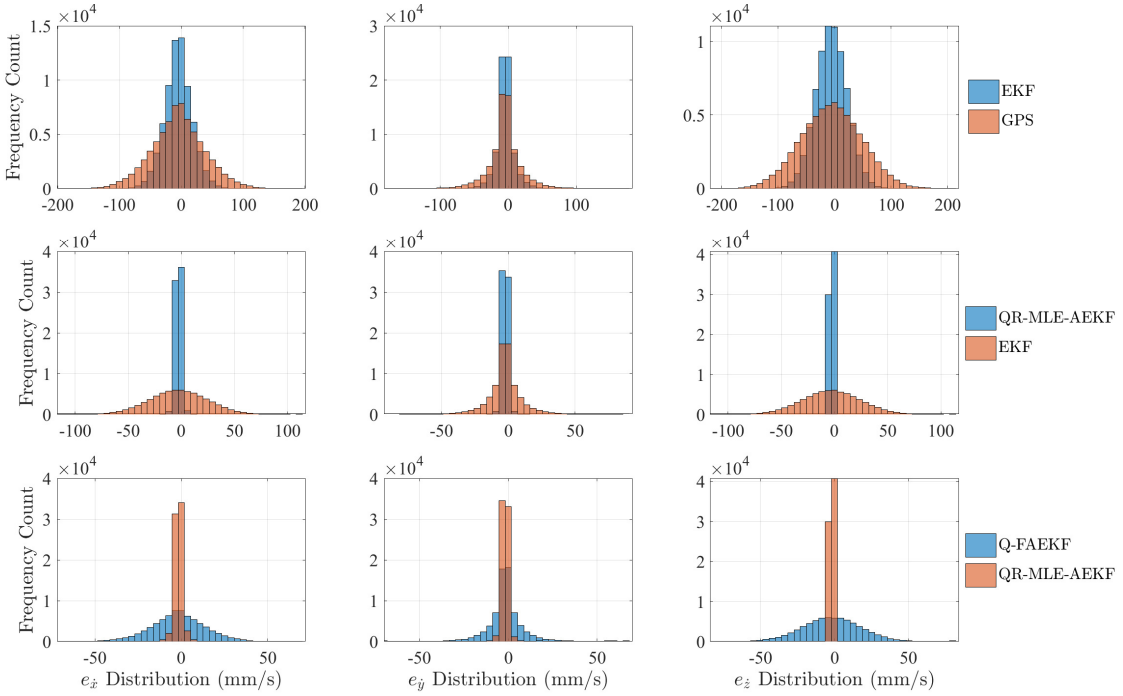


Figure 6.24: Relative velocity error distributions for the PROBA-3 formation.

6.3.3 PROBA-3 Simulation Conclusions

In the context of the PROBA-3 formation, the MLE-AEKF with adaptations of both the process and measurement noise covariances provided the best estimation performance of the tested EKF variants. Although the 3D-RMS errors in Fig. 6.12 indicate that relative velocity estimation of the Q-MLE-AEKF outperforms the estimation of the QR-MLE-AEKF, the former has position errors larger than even the EKF. With this in mind, the QR-MLE-AEKF is logically the most desirable filter in this case study, as both the position and velocity estimates are better than the other EKF and FAEKF variants. The processing time for the QR-MLE-AEKF is roughly 2.3 times larger than the time for the Q-FAEKF, but the additional overhead appears warranted as the estimation errors from the QR-MLE-AEKF are less than 50% of the errors from the Q-FAEKF. Even though the spacecraft separation grows throughout this scenario, the position accuracies observed are well within the two-meter objective.

The explanation as to why the QR-adaptations provided the best results in this scenario can be traced back to the orbital environment that was simulated: a highly elliptical orbit, which varied in altitude between 600 km and 60 500 km. This altitude variation results in different orbital perturbations dominating the spacecraft motion during different portions of the orbit, which adds further complexity to the highly nonlinear motion already imposed by the elliptical orbit. Thus, adapting the process noise covariance is necessary to account for these unmodelled effects within the EKF. Furthermore, the same level of measurement noise is used in this simulation, so no additional degradation of the estimates should occur purely based on the measurement model. The reason then that QR-adaptations provide the best overall results, is that the initial tuning of the noise covariance matrix is not optimal for the scenario, and allowing the filter to modify both the process and measurement noise covariances resulted in a tuning that was better than estimating the process covariance alone.

Table 6.12: 3D-RMS Errors from PROBA-3 Simulation

Method	Position (cm) [% Error]	Velocity (cm/s) [% Error]	Relative Runtime
Measurements	294.81 [3.15%]	7.33 [857.11%]	N/A
Standard EKF	122.15 [1.30%]	3.76 [439.66%]	1.00
Q-MLE-AEKF	146.74 [1.57%]	0.05 [5.85%]	2.59
R-MLE-AEKF	195.72 [2.09%]	7.33 [857.11%]	2.74
QR-MLE-AEKF	35.78 [0.38%]	0.25 [29.23%]	2.78
Q-FAEKF	72.77 [0.78%]	2.67 [312.21%]	1.23
R-FAEKF	124.17 [1.33%]	3.69 [431.48%]	1.56
QR-FAEKF	76.65 [0.82%]	2.95 [344.05%]	1.63

A Practical Note on the PROBA-3 Scenario

The simulations of the PROBA-3 formation made use of an assumption that measurements of the formation were available throughout the entirety of the orbit, while the errors passed to the EKF algorithms were selected to be representative of common GPS navigation inaccuracies. However, the 24 primary satellites of the GPS constellation sit in semi-synchronous, nearly-circular orbits with an average altitude of 20 200 km, and at least four instantaneous GPS signals must be acquired in order for a spacecraft to construct a measurement. Since the highly elliptical orbit of PROBA-3 results in altitudes that exceed 60 000 km, loss of GPS signal reception can indeed become a problem, and this characteristic of HEO spacecraft has been explored in the literature by authors including Rupp *et al.* [122] and Vigneron [83]. Promising theoretical analyses by Marmet *et al.* [123] have nevertheless established the feasibility of using GPS for navigation in geostationary equatorial orbits, geostationary transfer orbits, and highly elliptical orbits, and on-orbit results presented by Davis *et al.* [124] claimed GPS signal lock could be achieved up to an altitude of 58 000 km. With these concerns in mind, adequate handling of realistic GPS signal performance would be necessary to improve the fidelity of the PROBA-3 analysis conducted here.

Although modelling the GPS constellation, antenna patterns, and receiver properties are beyond the scope of this thesis, handling measurement drop-out is well within the theoretical capabilities of the EKF algorithm, and can be approached in several ways. The first, and typically least accurate, is to simply skip the correction phase of the EKF whenever measurements are unavailable. For the given duration of the measurement drop-out, this equates to only using the dynamics propagation step of the EKF to calculate the estimated states. Additional modelling of the dynamics errors can be performed mitigate the compounding errors that occur using this approach, but this requires further assumptions regarding the stochastic nature of the errors.

Another approach to deal with the loss of GPS lock, albeit involving more complexity, is to switch to a different measurement system. Spacecraft commonly employ multiple sensors for navigation, attitude determination, and scientific objectives, so measurements other than the primary GPS solutions could be used to provide relative navigation estimates. For example, a close-proximity formation that experiences GPS signal loss could make use of optical or infrared cameras as proposed in [65], thus maintaining knowledge of the relative motion between the spacecraft during the outage. This estimation would require a different Kalman filter and the resulting state estimates would likely be more erroneous than those obtained using GPS, but such line-of-sight techniques could nevertheless provide more confidence in the relative position of the spacecraft until the formation is back in range of the GPS constellation.

6.4 Case 3: PEO in LEO Formation

The last formation configuration considered in this thesis consists of a projected elliptical orbit between the chaser and the target. The spacecraft characteristics used in this study are similar to those of the PRISMA formation, but the defining orbital elements and resulting relative motion are different. In this case, both the target and the chaser orbits have an eccentricity around 0.1, two orders of magnitude larger than the original PRISMA mission. The relative orbit of the chaser is also centered on the target, whereas the true PRISMA formation features an offset relative orbit. Orbit parameters and spacecraft properties for the PEO formation were presented earlier in Table 2.6 and Table 2.7, and the trajectory of the spacecraft was shown in Fig. 2.11. Due to the non-zero eccentricity in this LEO scenario, the orbital environment varies in altitude from between 350 km to 1 900 km, and the spacecraft will therefore be subjected to varying degrees of orbital perturbations throughout each orbit. The separation between spacecraft oscillates between 370 m and 1 500 m, so ideally the position estimation accuracy will be within four meters.

6.4.1 PEO Simulation Conditions

For completeness, the simulation settings for the PEO case are shown in Table 6.13. To demonstrate the performance of the filtering algorithms in the presence of larger measurement noises, the standard deviations of the noise processes have been increased by an order of magnitude from the previous two cases. The initial measurement noise covariance matrix has been increased in light of the additional noise, and the initial conditions of the EKF are also modified to account for the different configuration of the formation. Table 6.14 lists the initial settings for the EKF. This case study therefore provides insight into the performance of the filter adaptation schemes in a situation where the measurements noise are improperly defined or unknown within the filter.

Table 6.13: Simulator Settings for the PEO in LEO Formation

Options	Settings
Simulation Duration	3 hrs, 35 mins (12 928 s)
Orbital Perturbations	J_2, Luni-Solar Third Body, Drag, SRP
Spacecraft Formation	PRISMA (Mango and Tango)
Measurement Noises	$\sigma_r = 12$ m $\quad \sigma_v = 0.3$ m/s
Measurement Sampling	$T = 1.0$ s $\quad (f = 1$ Hz)

Table 6.14: Initial EKF Settings for the PEO in LEO Formation

Initial States $\mathbf{x}_0$	$x_0 = -395.00$ m	$\dot{x}_0 = 2.00$ mm/s
	$y_0 = 20.00$ m	$\dot{y}_0 = 852.70$ mm/s
	$z_0 = -42.49$ m	$\dot{z}_0 = -1\ 404.48$ mm/s
	$\theta_0 = 0$ deg	$\dot{\theta}_0 = 0.0684$ deg/s
	$r_{t_0} = 6\ 750.05$ km	$\dot{r}_{t_0} = 0.00$ m/s
Initial State Error Covariance $\mathbf{P}_0$	$\sigma_x^2 = 100\ \text{m}^2$	$\sigma_{\dot{x}}^2 = 1\ \text{m}^2/\text{s}^2$
	$\sigma_y^2 = 100\ \text{m}^2$	$\sigma_{\dot{y}}^2 = 1\ \text{m}^2/\text{s}^2$
	$\sigma_z^2 = 100\ \text{m}^2$	$\sigma_{\dot{z}}^2 = 1\ \text{m}^2/\text{s}^2$
	$\sigma_\theta^2 = 1\ \text{deg}^2$	$\sigma_{\dot{\theta}}^2 = 0.01\ \text{deg}^2/\text{s}^2$
	$\sigma_{r_t}^2 = 10\ 000\ \text{m}^2$	$\sigma_{\dot{r}_t}^2 = 100\ \text{m}^2/\text{s}^2$
Initial Process Noise Covariance $\mathbf{Q}_0$	$q_x = 0.2\ \text{m}^2$	$q_{\dot{x}} = 5 \times 10^{-3}\ \text{m}^2/\text{s}^2$
	$q_y = 0.2\ \text{m}^2$	$q_{\dot{y}} = 5 \times 10^{-3}\ \text{m}^2/\text{s}^2$
	$q_z = 0.2\ \text{m}^2$	$q_{\dot{z}} = 5 \times 10^{-3}\ \text{m}^2/\text{s}^2$
	$q_\theta = 1 \times 10^{-3}\ \text{deg}^2$	$q_{\dot{\theta}} = 1 \times 10^{-6}\ \text{deg}^2/\text{s}^2$
	$q_{r_t} = 5 \times 10^{-3}\ \text{m}^2$	$q_{\dot{r}_t} = 5 \times 10^{-5}\ \text{m}^2/\text{s}^2$
Initial Measurement Noise Covariance $\mathbf{R}_0$	$r_x = 100\ \text{m}^2$	$r_{\dot{x}} = 2.5\ \text{m}^2/\text{s}^2$
	$r_y = 100\ \text{m}^2$	$r_{\dot{y}} = 2.5\ \text{m}^2/\text{s}^2$
	$r_z = 100\ \text{m}^2$	$r_{\dot{z}} = 2.5\ \text{m}^2/\text{s}^2$
	$r_\theta = 0.05\ \text{deg}^2$	

Following the initial EKF conditions established in the previous two case studies, the relative position and velocity states of the formation are offset from the true values by 20 m and 20 mm/s, respectively. The same moving window size of $N = 30$ is used for both the MLE-AEKF and the FAEKF, and the adaptations were restricted to the diagonal elements of the appropriate covariance matrices. These settings for the MLE-AEKF are summarized in Table 6.15, and the parameters used in the FAEKF are given in Table 6.16.

Table 6.15: MLE-AEKF Settings for the PEO in LEO Formation

Options	Settings	
MLE Smoothing	$N = 30$	
Nearest PSDS Check	$Q =$ OFF	$R =$ OFF
Diagonalized Adaptations	$Q =$ ON	$R =$ ON

Table 6.16: FAEKF Settings for the PEO in LEO Formation

Options	Settings	
Residual Averaging	$N = 30$	
Input Fuzzy Gains	$g^q = -5 \times 10^{-3}$	$h^q = 1 \times 10^{-2}$
Output Fuzzy Gains	$g_1^r = 5 \times 10^{-2}$	$h_1^r = 1 \times 10^{-4}$
	$g_2^r = 5 \times 10^{-2}$	$h_2^r = 1 \times 10^{-4}$
	$g_3^r = 5 \times 10^{-2}$	$h_3^r = 1 \times 10^{-4}$
	$g_4^r = 1 \times 10^{1}$	$h_4^r = 1 \times 10^{-4}$
	$g_5^r = 1 \times 10^{0}$	$h_5^r = 1 \times 10^{-3}$
	$g_6^r = 1 \times 10^{0}$	$h_6^r = 1 \times 10^{-3}$
	$g_7^r = 1 \times 10^{0}$	$h_7^r = 1 \times 10^{-3}$

6.4.2 PEO Simulation Results

Despite the increased measurement noises used in this case study, all filters are successfully able to converge to the true state trajectory and provide relative position and velocity estimates that are more accurate than the measurements themselves. Table 6.17 is a compilation of the average RMS errors and the relative runtimes observed in the PEO simulations. The MLE-AEKF schemes require more processing time than the FAEKF schemes as discussed previously, and similarly the R-adaptations are more costly than the Q-adaptations. These computational considerations match the results from the previous two case studies.

With the additional measurement noises applied in this study, adapting the measurement noise covariance matrix is presumably the ideal way to improve filter performance, and the results of this case study verify that prediction. In both the MLE-AEKF and the FAEKF tests, the R-adaptation schemes provide the best position estimation results, with average RMS errors of 250 cm and 275 cm, respectively. In contrast, both the Q-MLE-AEKF and Q-FAEKF provide position estimates that are more erroneous than the non-adaptive EKF. This recapitulates the fact that identifying and selecting the proper adaptation method for a particular scenario is indeed a design consideration in and of itself.

As in the previous two cases, the adaptive EKF algorithms are able to obtain position estimation accuracies within 1% of the minimum spacecraft separation distance, which here corresponds to an average error of less than four meters. The support for the numerical results of the PEO in LEO simulations are provided in the form of state and error time history plots throughout the following pages, along with an analysis of the error distributions for the various estimation routines. For this scenario with increased measurement noises, figures relating to the EKF, the R-MLE-AEKF, and the R-FAEKF are shown.

Table 6.17: Average RMS Errors and Runtimes from PEO Simulation

Method	Position (cm)	Velocity (cm/s)	Relative Runtime
Measurements	1 674.18	41.71	N/A
Standard EKF	404.81	29.68	1.00
Q-MLE-AEKF	883.16	0.21	2.32
R-MLE-AEKF	250.88	11.74	2.56
QR-MLE-AEKF	251.84	3.00	2.58
Q-FAEKF	1 337.28	33.36	1.17
R-FAEKF	275.06	10.74	1.28
QR-FAEKF	1 371.68	41.71	1.30

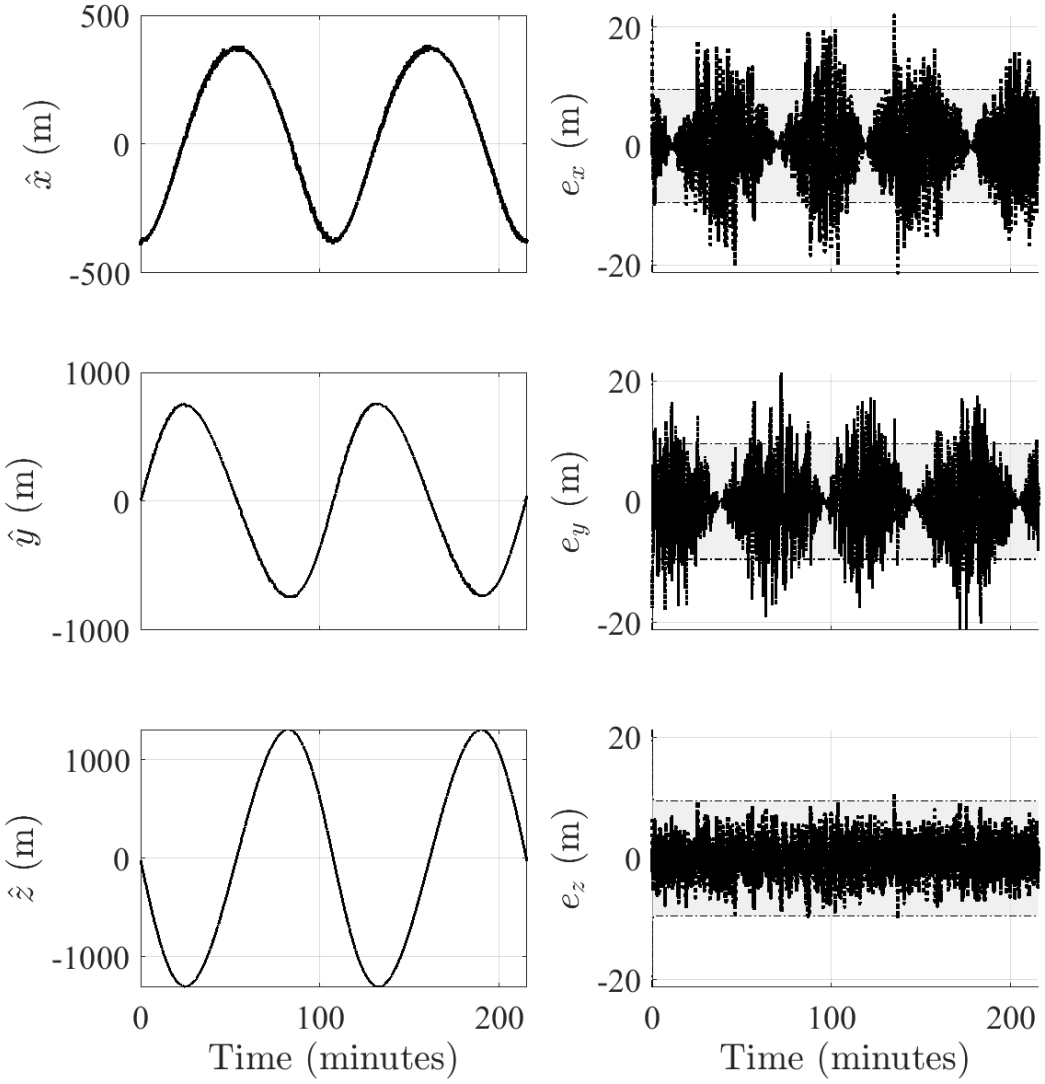


Figure 6.25: EKF relative position estimation results for the PEO formation. (Left) The estimated states. (Right) The estimation errors and 3σ covariance bounds. Although the filter converges to the true states, the additional noise in the measurements has compromised the performance of the EKF. Many of the state errors fall outside of the filter covariance bounds, and this highlights that the nominal filter tuning in this scenario is not ideal.

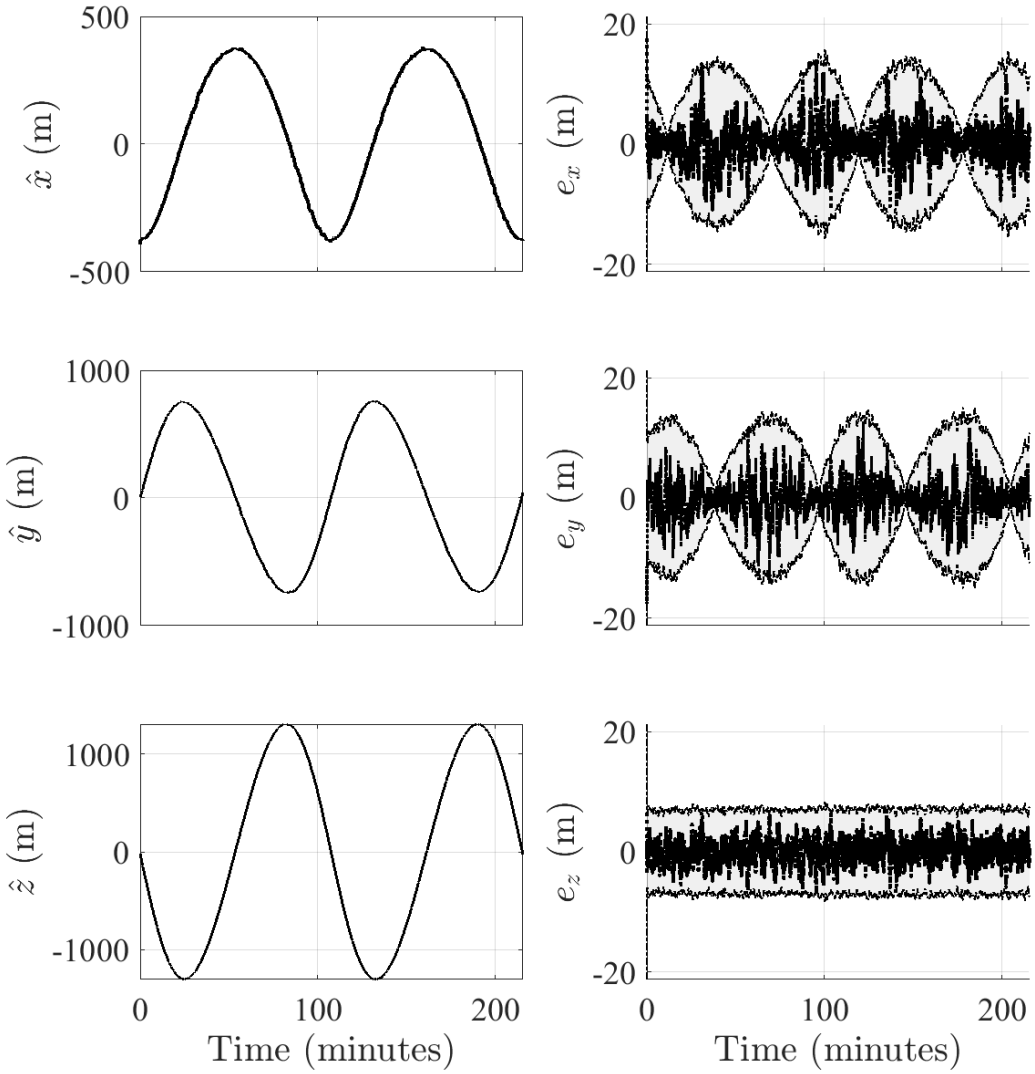


Figure 6.26: MLE-AEKF relative position estimation results for the PEO formation using R-adaptations. (Left) The estimated states. (Right) The estimation errors and 3σ covariance bounds. By updating the measurement noise covariance matrix in real-time, the R-MLE-AEKF yields estimation errors that are smaller in magnitude than the EKF. The errors are now within the covariance bounds of the filter as well.

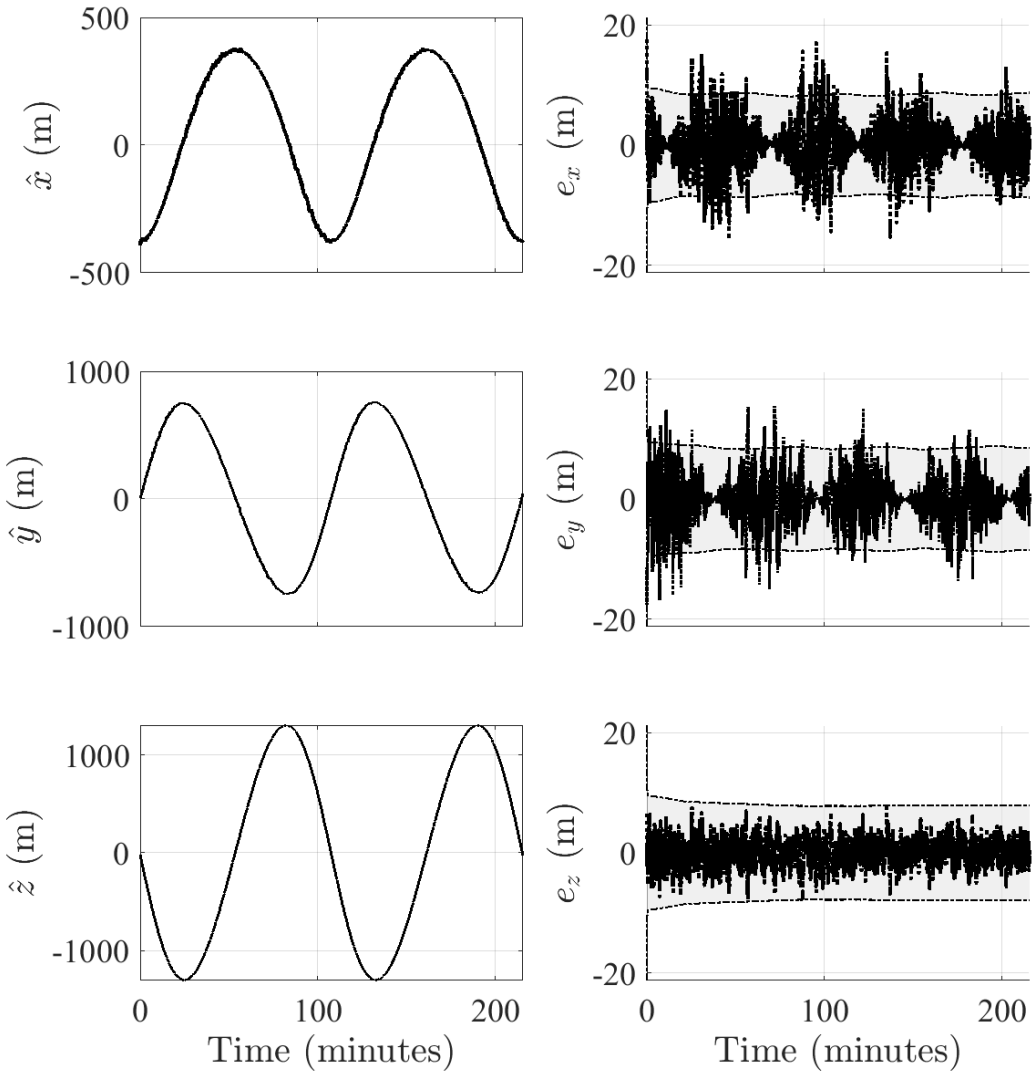


Figure 6.27: FAEKF relative position estimation results for the PEO formation using R-adaptations. (Left) The estimated states. (Right) The estimation errors and 3σ covariance bounds. The in-plane position errors occasionally exceed the filter covariance bounds at the beginning of the simulation, but the frequency of these occurrences decreases throughout the duration of the test. This gradual improvement in estimation performance is characteristic of the current tuning of the FAEKF, but these results nevertheless indicate that the estimation errors of the R-FAEKF are lower than those of the EKF.

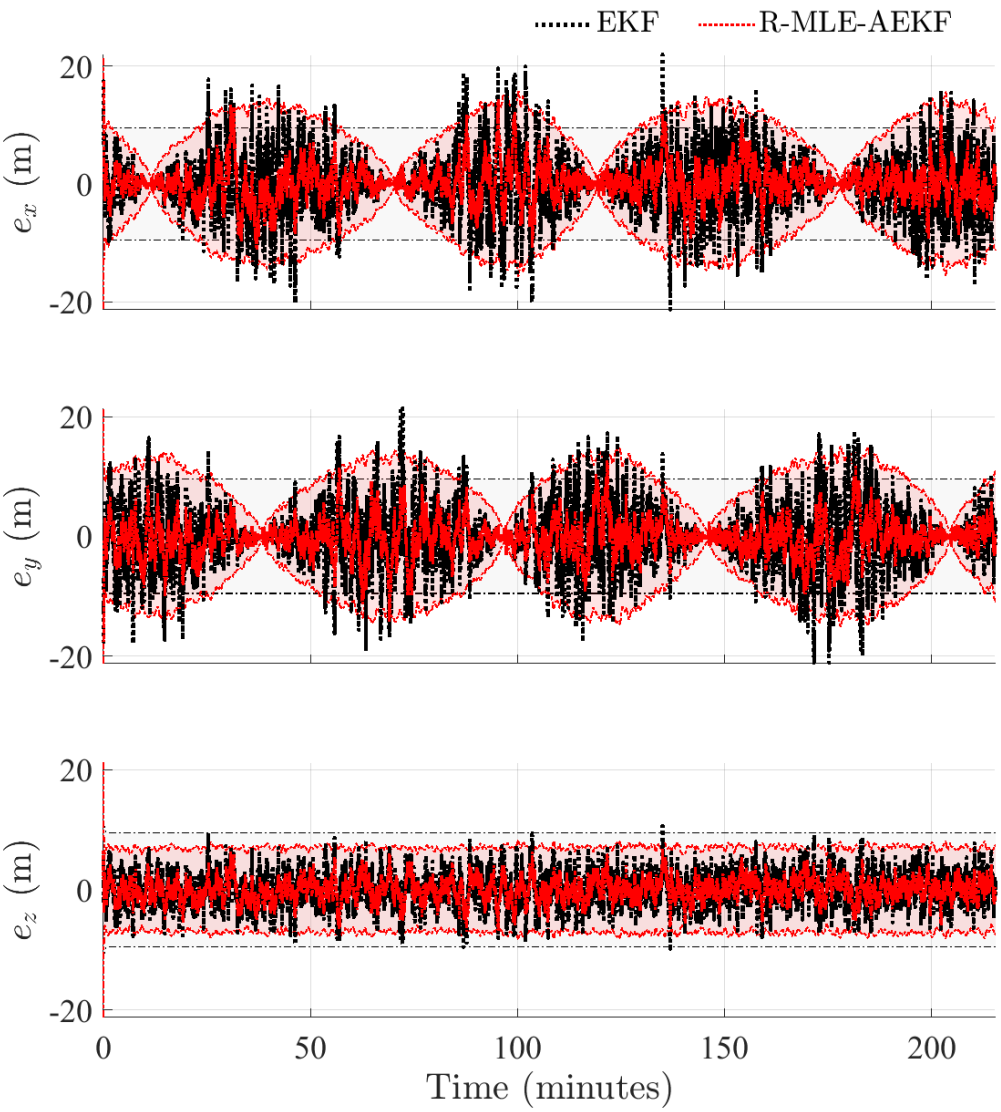


Figure 6.28: Comparison of relative position estimates for the PEO formation, between the EKF and the MLE-AEKF with R-adaptations. The estimation errors and 3σ covariance bounds are shown. The adaptive filter reduces the errors, are their associated covariance bounds can be seen to oscillating in phase with the errors.

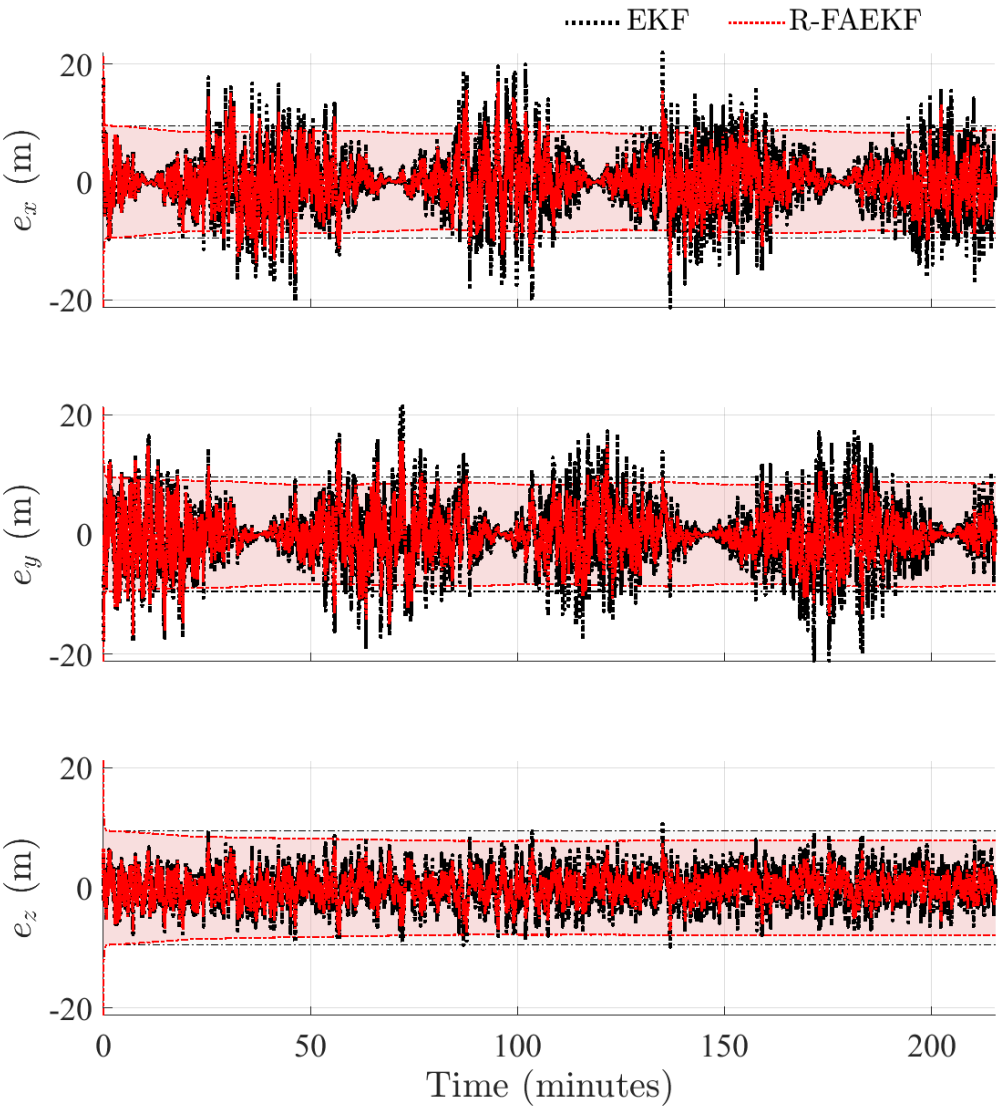


Figure 6.29: Comparison of relative position estimates for the PEO formation, between the EKF and the FAEKF with R-adaptations. The estimation errors and 3σ covariance bounds are shown, confirming that the adaptations from the FLS within the adaptive EKF successfully reduce the errors compared to the EKF. The responsiveness of the FAEKF is notably slower than then MLE-AEKF, as evidenced by the rate of covariance convergence.

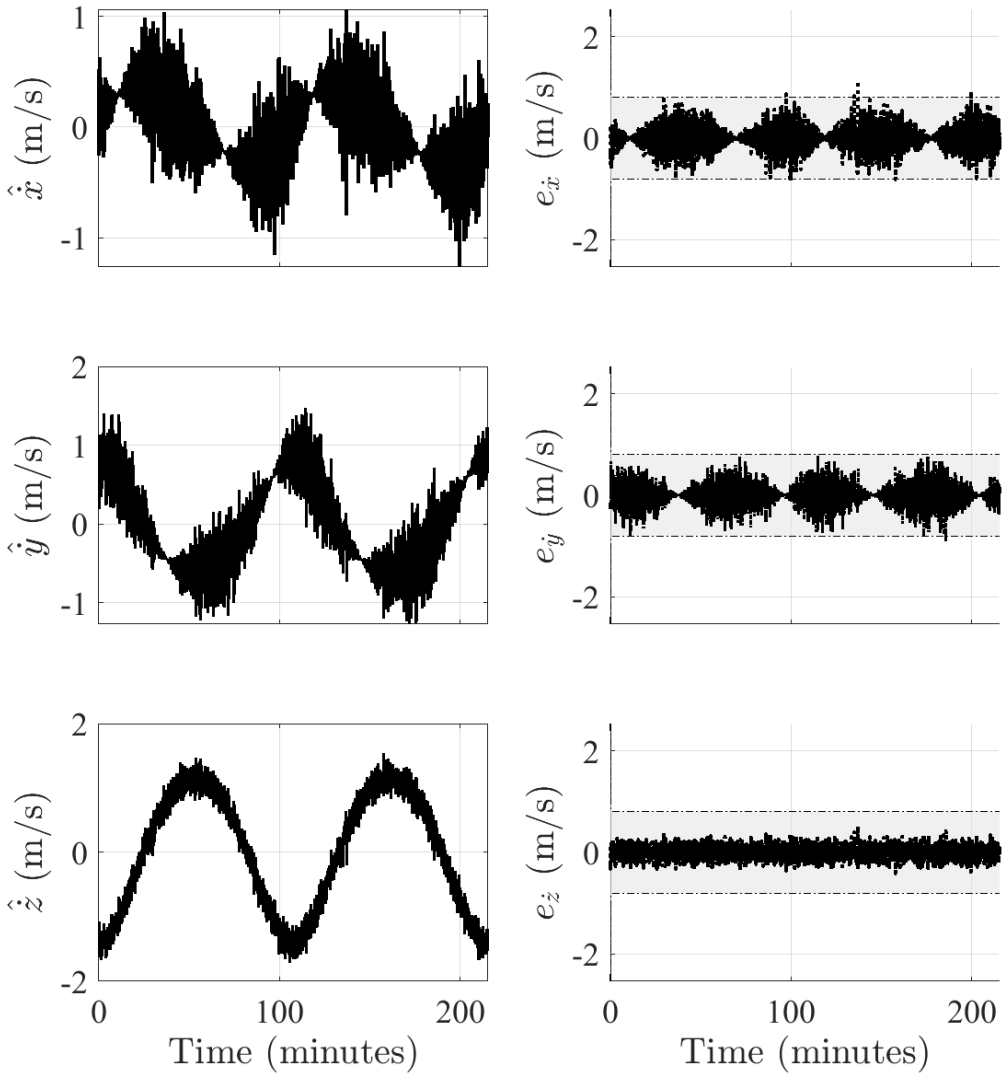


Figure 6.30: EKF relative velocity estimation results for the PEO formation. (Left) The estimated states. (Right) The estimation errors and 3σ covariance bounds. Although the velocity estimation errors stay within the filter covariance bounds, the resulting state estimation trajectories are still noisy. Without properly adapting the EKF, the increased noise infused in the measurements corrupts the final state estimates of the filter.

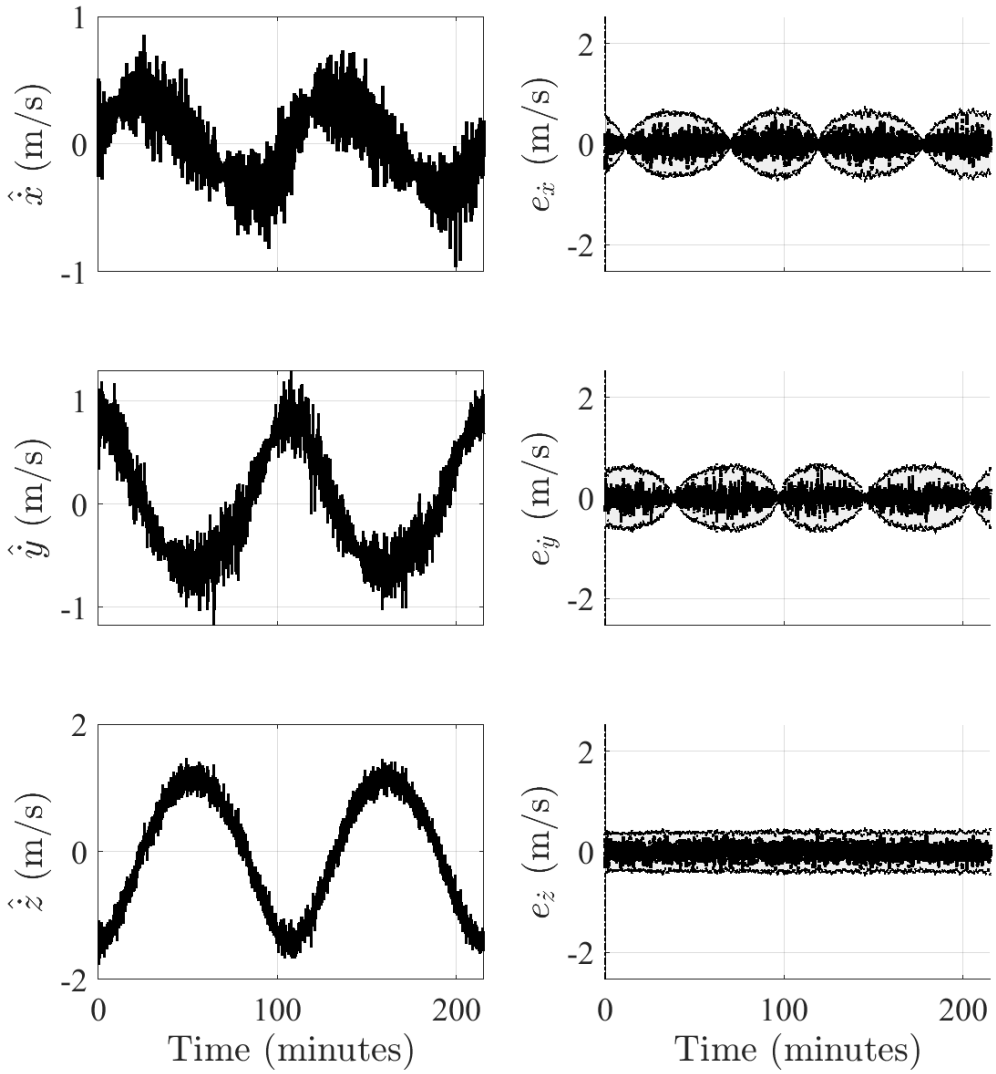


Figure 6.31: MLE-AEKF relative velocity estimation results for the PEO formation using R-adaptations. (Left) The estimated states. (Right) The estimation errors and 3σ covariance bounds. Adapting the measurement noise covariance matrix has reduce the magnitude of the estimation errors, and similarly the filter covariance bounds now match the periodic nature of errors.

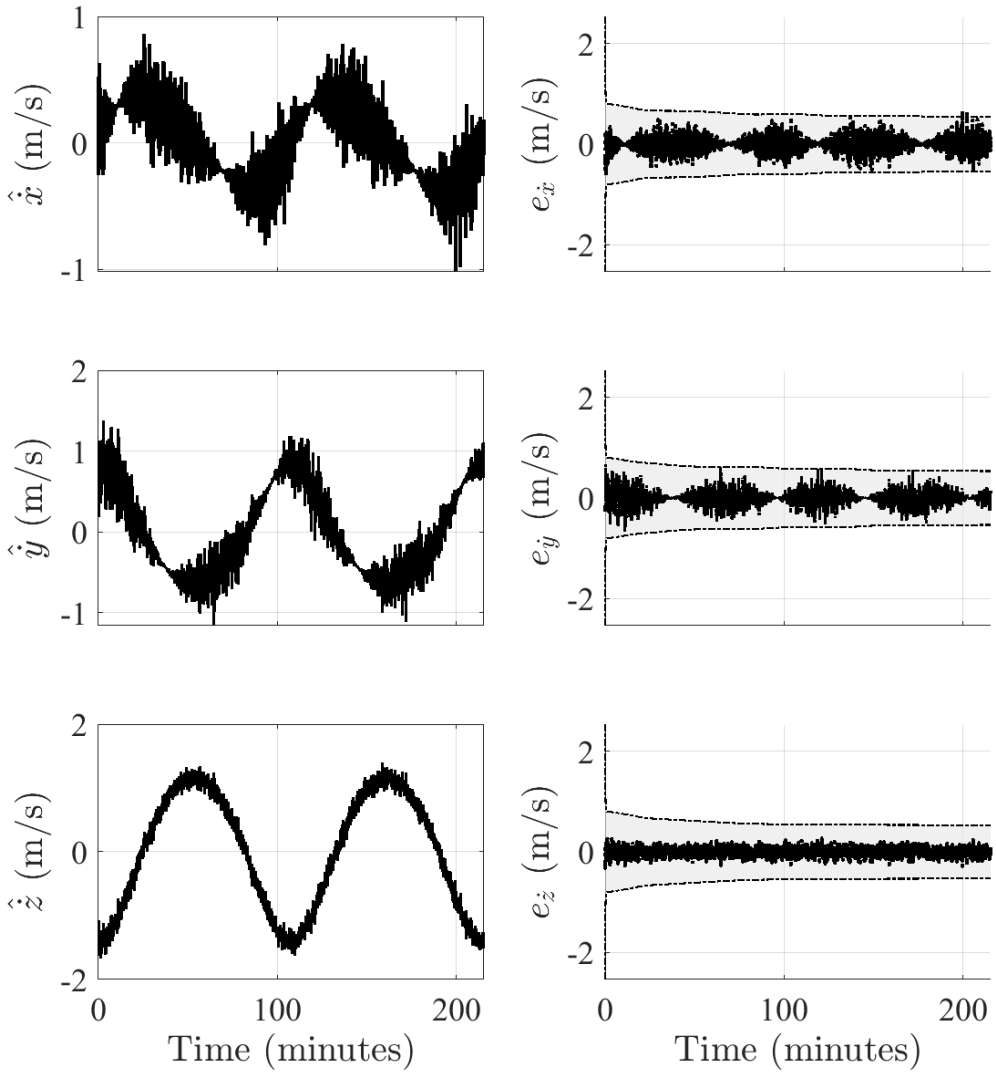


Figure 6.32: FAEKF relative velocity estimation results for the PEO formation using R-adaptations. (Left) The estimated states. (Right) The estimation errors and 3σ covariance bounds. Contrary to the MLE-AEKF covariance bounds, the FAEKF bounds shown here gradually decrease over time.

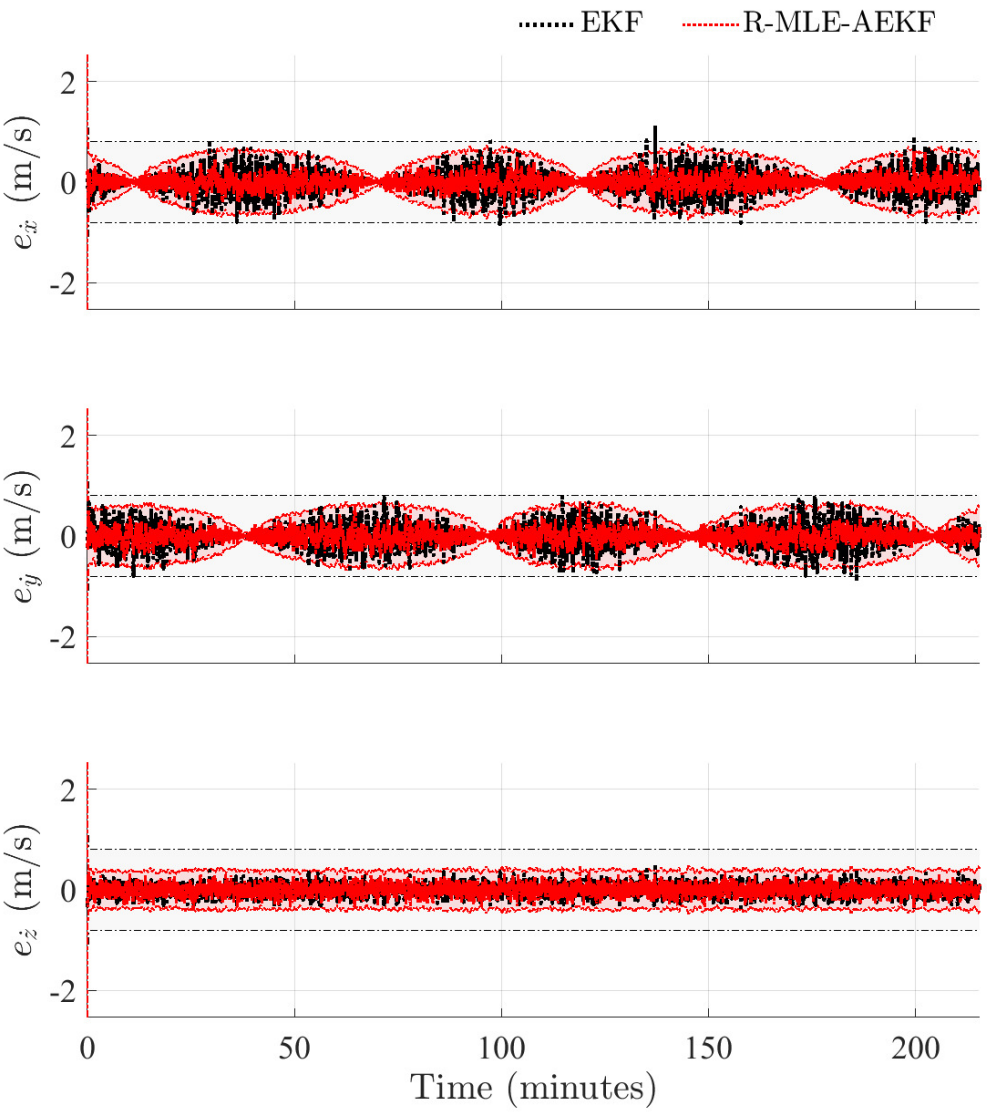


Figure 6.33: Comparison of relative velocity estimates for the PEO formation, between the EKF and the MLE-AEKF with R-adaptations. The estimation errors and 3σ covariance bounds of the R-MLE-AEKF are both lower than the corresponding error and covariance bounds of the EKF.

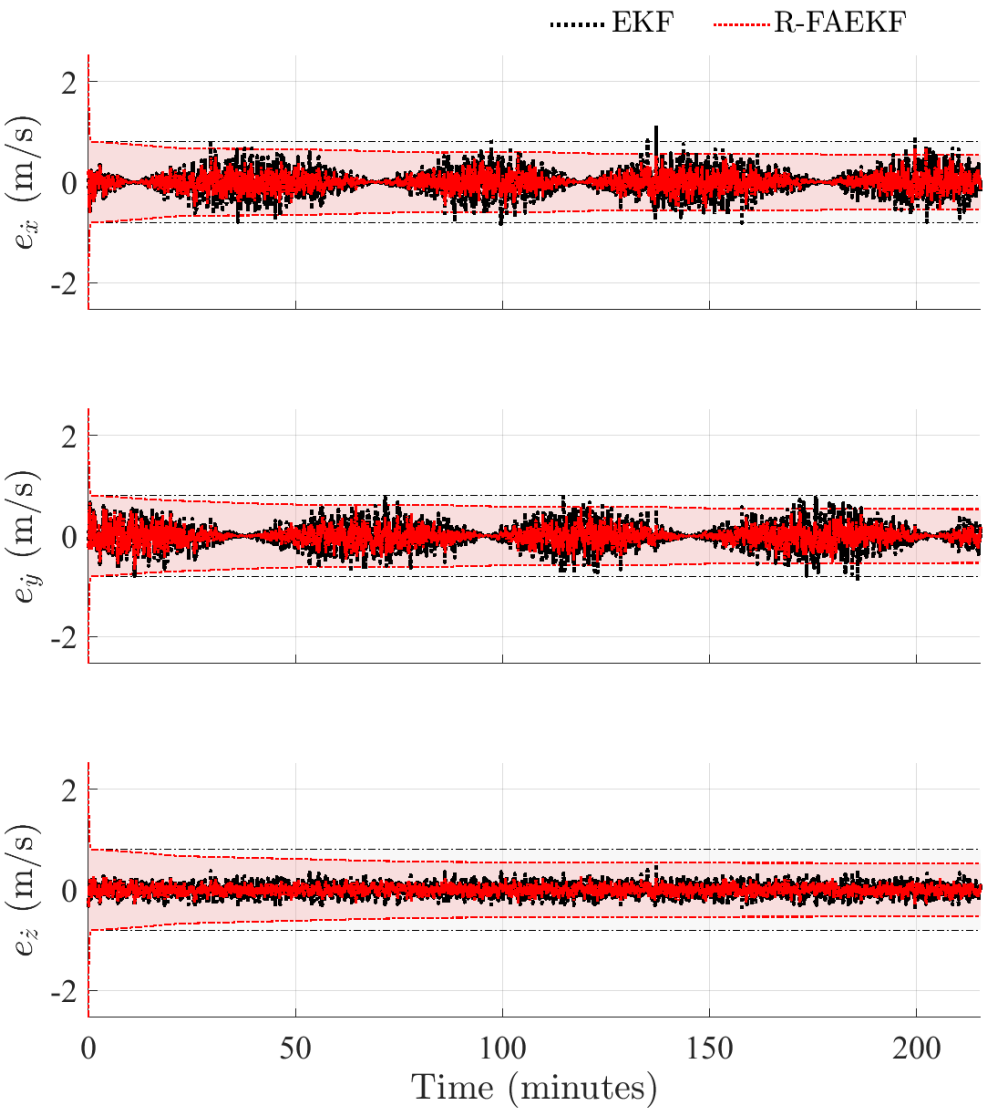


Figure 6.34: Comparison of relative velocity estimates for the PEO formation, between the EKF and the FAEKF with R-adaptations. As expected, the estimation errors of the EKF are consistently larger than the errors from the R-FAEKF.

The position estimation errors distributions for the PEO case study are contained in Fig. 6.35, and use the same format as the previous histograms: the first row compares the EKF and the GPS measurements, the second row compares the R-MLE-AEKF and the EKF, and the third row compares the R-FAEKF and the R-MLE-AEKF. The magnitude of the measurement noise in this scenario leads to a widely distributed error plot for the measurements, spanning ±50 m. However, the baseline EKF is able to greatly reduce this distribution to within ±25 m, and from the average RMS values shown previously, the position errors from the EKF are 25% as large as the measurement errors.

Considering the second row, the errors from the R-MLE-AEKF are more tightly-grouped around zero than the EKF values. This signifies that the errors from the adaptive filter are more frequently at a lower magnitude than the errors of the non-adaptive filter, and additionally substantiates that adapting the measurement noise covariance is a suitable tactic for dealing with the larger noises in the measurements. To conclude, the position estimation errors of the R-MLE-AEKF and R-FAEKF are similar, but the MLE-AEKF yields errors that are more frequently smaller than the errors from the FAEKF.

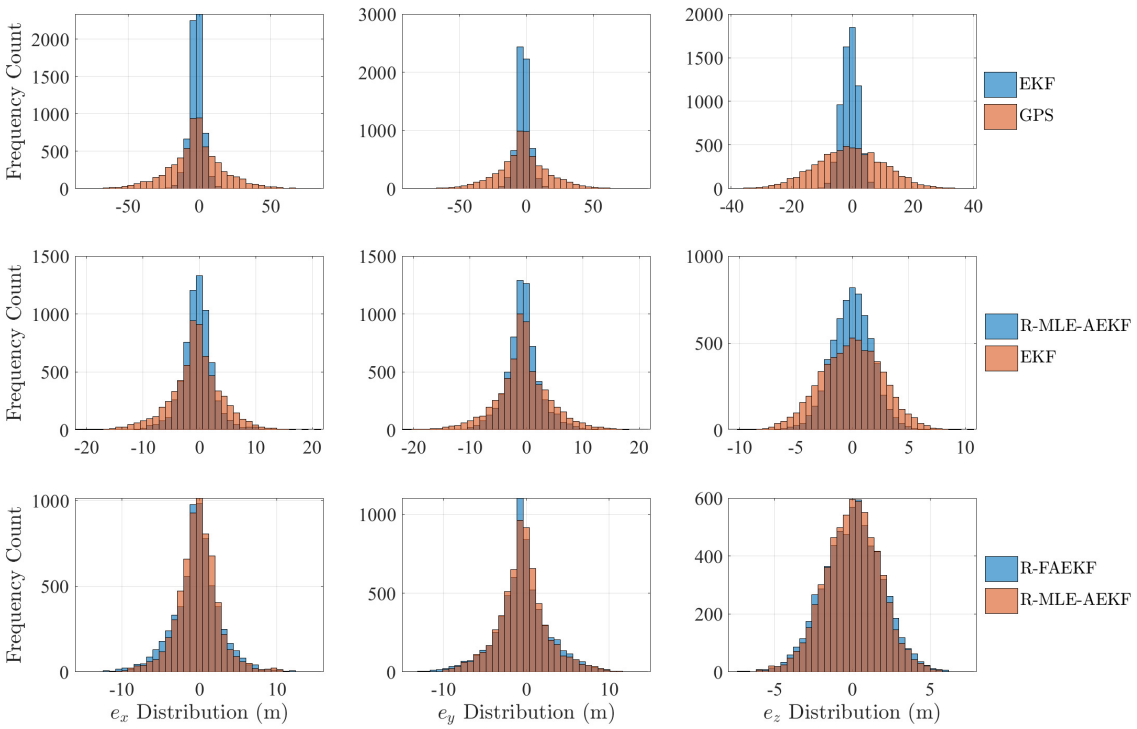


Figure 6.35: Relative position error distributions for the PEO formation.

The last figure in these numerical analyses presents the velocity estimation error distributions for the PEO simulations. Figure 6.36 demonstrates clearly in the first row that the velocity estimates from the EKF are more accurate than the measurements provided to the EKF, a result that is not unexpected. Interestingly, the second row exposes that the span of the velocity estimation errors from the R-MLE-AEKF is nearly as large as the EKF. Although the width of both these symmetric distributions is similar, the sharpness of the R-MLE-AEKF is greater than that of the EKF. Thus, more estimation errors from the MLE-AEKF are smaller than the errors from the EKF, which corroborates with the smaller average RMS achieved by the MLE-AEKF compared to the EKF.

The third row confirms that the velocity estimates obtained by the R-FAEKF are consistently more accurate than the estimates by the R-MLE-AEKF. The average RMS error for the FAEKF is only 10 mm/s lower than that of the MLE-AEKF, but the histograms shown below provide an astute representation of the filter performances.

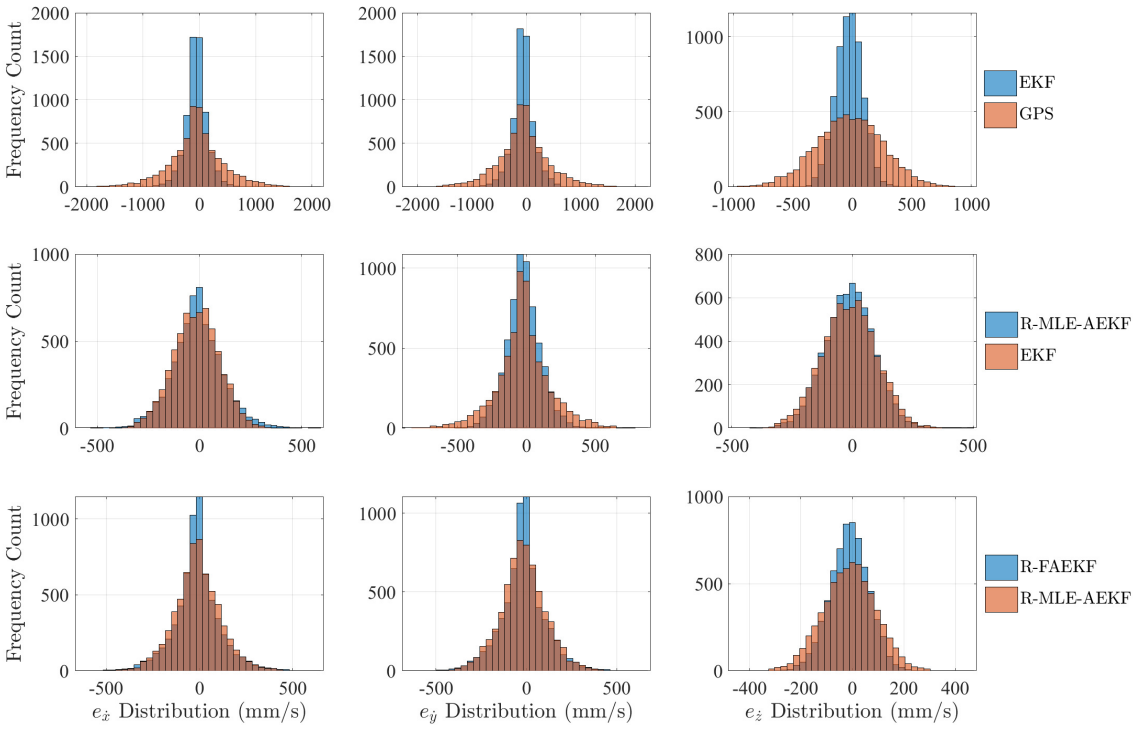


Figure 6.36: Relative velocity error distributions for the PEO formation.

6.4.3 PEO Simulation Conclusions

The EKF, MLE-AEKF, and FAEKF demonstrated improved relative position and velocity estimation throughout this study of a spacecraft formation in a projected elliptical orbit. Referring to Table 6.18 below, the effects of increasing the measurement noise led to poor performance of the Q-adaptation routines. This is a foreseeable result, as the formulations of both the MLE-AEKF and the FAEKF do not distinguish between noises due to measurement errors and noises due to modelling mismatch; they simply adapt to observations based on the residuals. As such, selecting an appropriate adaptation scheme for a particular scenario is a consideration that must be made before implementing these adaptation filters in a practical setting.

No discrepancies in the processing requirements for the Kalman filter variants were detected in the PEO simulations. Combined QR-adaptation was the most expensive, followed by R-adaptation and then Q-adaptation, with the FAEKF taking 30% longer than the EKF, and the MLE-AEKF taking twice as long as the FAEKF.

Recalling the nominal PRISMA simulations from Case 1, note that the 3D-RMS position errors of the standard EKF were reduced by 72% through the use of the MLE-AEKF for resolving errors in the dynamical models. Here, use of the MLE-AEKF in dealing with measurement noise gives a 3D-RMS position error reduction upwards of 38%. While further analysis would be needed to determine if the absolute improvements obtainable by the separate Q- and R-adaptation schemes are indeed restricted by the characteristics of the system, the proof-of-concepts shown here confirm that the MLE-AEKF and FAEKF are able to improve the performance of the EKF in a variety of spacecraft formation configurations and orbital environments.

Table 6.18: 3D-RMS Errors from PEO Simulation

Method	Position (cm) [% Error]	Velocity (cm/s) [% Error]	Relative Runtime
Measurements	2 974.18 [7.98%]	74.11 [204.28%]	N/A
Standard EKF	719.32 [1.93%]	29.68 [81.89%]	1.00
Q-MLE-AEKF	1 697.27 [4.56%]	0.42 [1.16%]	2.32
R-MLE-AEKF	443.61 [1.19%]	20.41 [56.31%]	2.56
QR-MLE-AEKF	457.24 [1.23%]	5.88 [16.22%]	2.58
Q-FAEKF	2 375.70 [6.38%]	59.27 [163.53%]	1.17
R-FAEKF	487.72 [1.31%]	19.01 [52.45%]	1.28
QR-FAEKF	2 415.14 [6.48%]	74.10 [204.45%]	1.30

6.5 Chapter Summary

This chapter has provided numerical validation of the proposed extended Kalman filtering algorithms, through the simulation of three unique spacecraft formation scenarios. The first scenario considered the PRISMA formation in a low-eccentricity, low-Earth orbit. In these conditions, unmodelled orbital perturbations are expected to dominate the state estimation errors of the EKF, and it was shown that the proposed Q-adaptation methods were able to drastically improve the accuracy of the state estimates. The second scenario simulated the highly-elliptical orbit of the PROBA-3 formation, where unmodelled dynamics were suspected to be less influential. Here, the QR-adaptations using the MLE-AEKF provided the best improvement in filter performance, indicating that modifications to both the process noise and measurement noise covariance matrices were necessary. The last case study used a representative spacecraft formation in a LEO-based projected-elliptical orbit with mild eccentricity, and featured an increase in the measurement noises. The performance of the R-adaptive Kalman filter schemes accordingly provided better state estimates in this scenario than the non-adaptive EKF.

In all three case studies, the performance of the Maximum Likelihood Estimation EKF was superior to that of the Fuzzy Adaptive EKF. The analytic MLE adaptation laws displayed rapid responsiveness, while the current tuning of the FAEKF provided slower adaptations based on the current tuning of the FLS. The improved performance of the MLE-AEKF comes at the cost of the additional processing power needed to complete the smoothing and adaptation calculations within the MLE routine, as the MLE-AEKF required twice the computational time of the FAEKF. While the MLE-AEKF estimation errors were lower than those of the FAEKF in these case studies, they were rarely better by a factor of two. This means that the accuracy-to-complexity is not directly proportional between the two methods, and careful consideration of the processing budget and desired navigation accuracy would need to be considered before implementing one of these methods for onboard operation. Above all, by intelligently selecting to adapt either the process noise covariance or the measurement noise covariance (or both) within the EKF, the adaptation schemes presented in this chapter successfully demonstrate two methods for improving the estimation accuracy of the standard EKF in the presence of modelling errors, measurement noise, and poorly-defined initial noise statistics within the filter itself.

Chapter 7

Conclusion

The final chapter of this thesis provides a summary of the conducted research, and expresses the significance of this work. A list of published and submitted articles regarding this research is presented, and a number of different avenues for future work are identified.

7.1 Thesis Summary

Spacecraft formations boast many distinct advantages over the classical monolithic spacecraft predominantly used over the past 60 years. The costs of designing, building, launching, and operating a formation can be significantly lower than those associated with a single spacecraft, as a result of the reduced physical size and mass of the structure, the redundancy of the systems and sensors, and the operational flexibility that are characteristic of formation flying spacecraft. Having separate, independent platforms facilitates a variety of unique scientific missions as well, including synthetic aperture radar interferometry, gravimetry, and deep-space observation. Not only do the benefits of coordinated spacecraft manifest in diverse mission profiles, but the use of onboard autonomous systems can further improve the on-orbit performance of the spacecraft themselves, thereby reducing the workload placed on the respective ground teams for the mission. The rapidly developing technologies associated with these cooperative teams of spacecraft rely on the accuracy and robustness of the guidance, navigation and control systems, in order to ensure safe and efficient operation.

This thesis addressed the navigation problem for spacecraft formation flying. More specifically, these thesis investigated the relative navigation problem of estimating the position and velocity states of one spacecraft in the formation with respect to another. The concept of on-orbit state estimation has been well explored in the past, and the extended Kalman filter is the single most widely-accepted method used by

operational spacecraft formations to-date. However, the performance of the EKF depends upon a number of factors, including the selected onboard dynamics model and the measurement sensors available on the spacecraft. Thus, although successful past missions have implemented the EKF and indeed demonstrated the robustness of the filter to modelling uncertainties, the achievable accuracy of the filter is limited by the *a priori* knowledge of the dynamical process and measurement noise characteristics.

In light of these deficiencies, this thesis proposed two methodologies for adaptive Extended Kalman filtering, both of which are suitable for spacecraft formations in perturbed (non-Keplerian) orbits. The first AEKF algorithm used Maximum Likelihood Estimation to develop adaptation equations for the process and measurement noise covariance matrices within the filter. The objective of the MLE routine was to maximize the likelihood that the observed measurements corresponded to the current tuning of the EKF, and the adaptation laws were derived through the residuals covariance matrix of the filter. An novel addition to the MLE algorithm was made, through the inclusion of an intrinsic fixed-window smoother capable of harnessing information from past state estimates to improve the convergence of the MLE cost function. The MLE derivations resulted in analytic adaptation equations for updating the noise covariance matrices within the EKF.

The second AEKF algorithm used covariance matching via an embedded fuzzy logic system, similarly allowing the process and measurement noise covariances of the EKF to be updated online based on standard observations of the EKF output. Two unique Mamdani-type single-input, single output fuzzy logic systems were developed to adapt the noise covariance matrices based on the difference between the theoretical covariance of the residuals predicted by the filter, and the observed covariance of the residuals calculated from the filter output. The logic rule base for the FLS was selected to achieve the covariance matching objective, and the fuzzy membership functions within the FLS were designed to intelligently adapt the noise covariance based on the extent of the residual covariance mismatch.

Prior to testing the proposed relative navigation filters, an orbit propagator was developed specifically for this research. The formation flying simulator generated absolute motion trajectories for both the target and chaser spacecraft, individually in the Earth-Centered Inertial reference frame. Beyond the standard two-body gravitational potential, appropriate dynamics models were included to account for the effects of the Earth's oblateness, Solar radiation pressure, atmospheric drag, and third-body perturbations due to the Moon and Sun. After differencing the absolute positions and velocities of the spacecraft, the relative states were transformed to a Local-Vertical Local-Horizontal frame for processing by the navigation system.

The non-adaptive EKF, the MLE-AEKF, and the FAEKF, were then validated using numerical simulations of three spacecraft formations. Firstly, the studies verified that all three proposed EKF methods achieve position accuracies more than two orders-of-magnitude less than the separation distance between the spacecraft, which represents the baseline level of accuracy for many state-of-the-art formation missions. Secondly, both adaptive EKFs were shown to yield better position and velocity estimates than those of the non-adaptive EKF. This was demonstrated in a variety of different formation operating conditions, including a low-eccentricity near-Earth orbit formation, a highly-elliptical formation with large separations, and a low-Earth, moderately elliptical formation subjected to large measurement noises.

In terms of navigation accuracy, the MLE-AEKF consistently resulted in the smallest estimation errors throughout the simulated cases. Adaptations to the noise covariance matrices were rapid and capable of adapting accordingly throughout the complete orbit of the formation. When the appropriate adaptation scheme for the given scenario was selected (*e.g.*, adapting the process noise in the presence of uncertainties in the dynamics model), the MLE-AEKF was more accurate than both the EKF and the FAEKF. This accuracy does come with additional computational complexity, and the MLE-AEKF required the longest processing times of the proposed EKF strategies.

The FAEKF was more computationally efficient than the MLE-AEKF, generally requiring only half of the processing time needed by the MLE technique. The resulting accuracy of the FAEKF was however lower than that of the MLE-AEKF, and the adaptations to the noise covariance matrices were notably slower. This is a result of the tuning of the FLS within the FAEKF, as the adaptation laws are directly influenced by the output scaling gains of the fuzzy system. Nevertheless, the FAEKF outperformed the EKF, and provided estimation errors on the same order of magnitude as those seen using the MLE-AEKF.

In conclusion, the adaptive EKF methods proposed in this thesis were successfully developed and shown to produce relative position and velocity estimates that were more accurate than estimates provided by the traditional EKF. The MLE-AEKF demonstrated lower estimation errors than the FAEKF in general, at the cost of additional computational complexity and processing time. The FAEKF however achieved comparable performance to the MLE-AEKF in several scenarios, and required only 50% of the processing time needed by the MLE-AEKF. The design of the FAEKF is also more flexible than the analytic equations of the MLE-AEKF, as the user is free to tune the FAEKF fuzzy system based on the desired adaptation performance. Thus, the adaptive filtering algorithms developed here present two unique methods to improve the real-time performance of the well-known Extended Kalman Filter.

7.2 Significance of Work

In addition to the personal growth and knowledge gained through the completion of this thesis, the significance of the conducted research is demonstrated by the publications and conference presentations that have been authored, which include:

Conference Proceedings

Fraser, C. and Ulrich S., "An Adaptive Kalman Filter for Spacecraft Formation Navigation using Maximum Likelihood Estimation with Intrinsic Smoothing", *American Control Conference*, Milwaukee, Wisconsin, 25-27 June 2018, pp. 5843-5848.

Fraser, C. and Ulrich S., "A Fuzzy Adaptive Kalman Filter for Spacecraft Formation Navigation", *American Control Conference*, Philadelphia, Pennsylvania, 10-12 July 2019, under review.

7.3 Recommendations for Future Work

Future work for this research could expand upon all three aspects of the formation flying navigation architecture that have been discussed throughout this thesis. At a high level, improving the fidelity of the numerical simulation environment, modelling different real-world sensors for the measurement system, and modifying the adaptation schemes within the proposed EKFs are all of practical importance for developing a more realistic navigation system. The following section will touch briefly on a few specific recommendations for future work.

7.3.1 Developing a Realistic GPS Measurement Model

While the formation measurements used in this work relied on infusing the absolute position and velocity states of the spacecraft with zero-mean, white Gaussian noises of comparable magnitude to those seen in typical GPS receivers, another option would be to use raw GPS data directly. As demonstrated in such works as D'Amico [13], Kroes *et al.* [26] and Montenbruck *et al.* [94], using GPS code or carrier phase pseudo-ranges can improve navigation solutions to millimeter-level accuracy. Furthermore, taking advantage of multiple GPS measurements, whether through differencing or a combination of data types, can help to eliminate measurement errors that result from ionospheric interference, GPS satellite clock biases, and receiver clock errors. Incorporating GPS measurements into the navigation algorithms proposed in this thesis would require modification of the measurement model within the EKF, and the development or use of a suitable GPS signal simulator.

7.3.2 Closed-loop GNC Testing

A natural extension of the proposed navigation routines would be their incorporation into a complete guidance, navigation and control system. This would reveal the benefits that stem from improving the relative state estimation accuracy of the formation, both in terms of achievable control accuracy of the formation and the amount of fuel required to maintain the formation. Within the context of the Spacecraft Robotics and Control Lab here at Carleton, the integration of this work and the guidance and control algorithms proposed by Kuiack [63] would yield a closed-loop GNC system for formation flying, and would be an interesting direction for future study.

7.3.3 Configuring the EKF for Alternate Measurements

The analyses conducted in this thesis relied on measurements representative of GPS solutions, however the performance of GPS-based navigation systems is limited to spacecraft formations operating in or near LEO-type geometries. This means that HEO and deep-space spacecraft are therefore outside the practical scope of GPS-based navigation solutions, and future work could address this issue. For example, the work of Rupp *et al.* [122] has shown that GPS navigation errors increase to 30 cm for spacecraft in HEO, and similar error magnitudes have been described by Vigneron [83]. There is clearly merit in investigating navigation methods that do not require the use of GPS measurements to address future formation mission designs, both for near-Earth and interplanetary applications. Methods using angles-only measurements have been proposed by Woffinder *et al.* [125], Lovell and Lee [126], and Sullivan *et al.* [127]. These methods rely on line-of-sight measurements only and can be obtained from optical cameras, thereby making angles-only navigation solutions suitable for obits that fall outside the range of the GPS constellation.

7.3.4 Expanding the Fuzzy Logic System

The FLS developed for this thesis was able to improve upon the performance of the EKF solely through the use of single-input single-output systems, but possible improvements to the FAEKF could be investigated from the standpoint of the fuzzy inference system. Using multiple-input single-output fuzzy systems, as seen in robotics applications [128] for example, could allow additional performance metrics to be factored into the adaptation rules. The work of da Silva and da Cruz [59] is a representative case where the bias and oscillation of the state estimates were used to characterize the filter performance, and other metrics undoubtedly exist that would be practical for the FAEKF. What these metrics might be remains to be seen, but the concept would nevertheless be an interesting conduit through which to incorporate additional human understanding into the fuzzy filter framework.

7.3.5 Optimization of the Adaptation Settings

Within both the MLE-AEKF and the FAEKF, the number of N past data points used for smoothing the residuals must be defined by the user. Further optimization of the value of N could be investigated to determine the influence of the parameter on the overall estimation accuracy. Since the value of N must be small enough to capture the dynamics of the system but large enough to provide smoothing, a method to optimize N given the orbital configuration and period would be beneficial.

References

[1] S. D'Amico, J. Ardaens, and R. Larsson, "Spaceborne Autonomous Formation-Flying Experiment on the PRISMA Mission," *Journal of Guidance, Navigation and Control*, vol. 35, no. 3, pp. 834–850, 2012.

[2] G. Bonin, N. Roth, S. Armitage, J. Newman, B. Risi, and R. E. Zee, "CanX–4 and CanX–5 Precision Formation Flight: Mission Accomplished!" in *Proceedings of the 29th Annual AIAA/USU Conference on Small Satellites*, ser. Technical Sessions I: All Systems Go! Utah State University, 2015.

[3] G. Gaias and J.-S. Ardaens, "Flight Demonstration of Autonomous Noncooperative Rendezvous in Low Earth Orbit," *Journal of Guidance, Control, and Dynamics*, vol. 41, no. 6, pp. 1337–1354, 2018.

[4] G. Gaias, J. Ardaens, and C. Schultz, "The AVANTI Experiment: Flight Results," in *10th International ESA Conference on Guidance, Navigaiton and Control System*. ESA, 2017.

[5] A. Poghosyan and A. Golkar, "CubeSat Evolution: Analyzing CubeSat Capabilities for Conducting Science Missions," *Progress in Aerospace Sciences*, vol. 88, pp. 59–83, 2017.

[6] D. Selva and D. Krejci, "A Survey and Assessment of the Capabilities of Cubesats for Earth Observation," *Acta Astronautica*, vol. 74, pp. 50–68, 2012.

[7] J. Bouwmeester and J. Guo, "Survey of Worldwide Pico-and Nanosatellite Missions, Distributions and Subsystem Technology," *Acta Astronautica*, vol. 67, no. 7-8, pp. 854–862, 2010.

[8] S. Nag, C. K. Gatebe, D. W. Miller, and O. L. de Weck, "Effect of Satellite Formations and Imaging Modes on Global Albedo Estimation," *Acta Astronautica*, vol. 126, pp. 77–97, 2016.

[9] A. Moccia and A. Renga, *Distributed Space Missions for Earth System Monitoring*. Springer Science & Business Media, 2013, ch. Bistatic Synthetic Aperture Radar, pp. 3–59.

[10] G. Krieger, I. Hajnsek, K. P. Papathanassiou, M. Younis, and A. Moreira, "Interferometric Synthetic Aperture Radar (SAR) Missions Employing Formation Flying," *Proceedings of the IEEE*, vol. 98, no. 5, pp. 816–843, May 2010.

[11] B. Tapley, J. Ries, S. Bettadpur, D. Chambers, M. Cheng, F. Condi, B. Gunter, Z. Kang, P. Nagel, R. Pastor, T. Pekker, S. Poole, and F. Wang, "GGM02 -

An Improved Earth Gravity Field Model from GRACE," *Journal of Geodesy*, vol. 79, no. 8, pp. 1–11, 2005.

[12] C. S. Cockell, T. Herbst, A. Léger, O. Absil, C. Beichman, W. Benz, A. Brack, B. Chazelas, A. Chelli, and H. Cottin, "Darwin - An Experimental Astronomy Mission to Search for Extrasolar Planets," *Experimental Astronomy*, vol. 23, no. 1, pp. 435–461, 2009.

[13] S. D'Amico, "Autonomous Formation Flying in Low Earth Orbit," Ph.D. dissertation, Technical University of Delft, 2010.

[14] S. Seager, W. Cash, S. Domagal-Goldman, N. J. Kasdin, M. Kuchner, A. Roberge, S. Shaklan, W. Sparks, M. Thomson, M. Turnbull *et al.*, "Exo-S: Starshade Probe-Class Exoplanet Direct Imaging Mission Concept Final Report," NASA Report 15-1155, Tech. Rep., 2015.

[15] G. Di Mauro, M. Lawn, and R. Bevilacqua, "Survey on Guidance Navigation and Control Requirements for Spacecraft Formation-Flying Missions," *Journal of Guidance, Control, and Dynamics*, pp. 1–22, 2017.

[16] D. G. Hoag, "Apollo Navigation, Guidance, and Control Systems: A Progress Report," Cambridge, MA: MIT Instrumentation Library, Tech. Rep., 1969.

[17] K. Hovell and S. Ulrich, "Postcapture Dynamics and Experimental Validation of Subtethered Space Debris," *Journal of Guidance, Control, and Dynamics*, vol. 41, no. 2, pp. 519–525, 2018.

[18] O. Montenbruck and S. D'Amico, *Distributed Space Missions for Earth System Monitoring*. Springer Science & Business Media, 2013, ch. GPS Based Relative Navigation, pp. 185–223.

[19] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[20] A. H. Mohamed and K. P. Schwarz, "Adaptive Kalman Filtering for INS/GPS," *Journal of Geodesy*, vol. 73, no. 4, pp. 193–203, 1999.

[21] B. P. Gibbs, *Advanced Kalman Filtering, Least-Squares and Modeling: A Practical Handbook*. Wiley, 2011.

[22] S. Bandyopadhyay, G. P. Subramanian, R. Foust, D. Morgan, S.-J. Chung, and F. Hadaegh, "A Review of Impending Small Satellite Formation Flying Missions," in *53rd AIAA Aerospace Sciences Meeting*, 2015, p. 1623.

[23] J. Sullivan, S. Grimberg, and S. D'Amico, "Comprehensive Survey and Assessment of Spacecraft Relative Motion Dynamics Models," *Journal of Guidance, Control, and Dynamics*, pp. 1–23, 2017.

[24] R. H. Merson, "A Brief Survey of Satellite Orbit Determination," *Mathematical and Physical Sciences*, vol. 262, no. 1124, pp. 71–78, 1967.

[25] J. R. Vetter, "Fifty Years of Orbit Determination: Development of Modern Astrodynamics Methods," in *Johns Hopkins APL Technical Digest*, vol. 27, no. 3, 2007, pp. 239–252.

[26] R. Kroes, O. Montenbruck, W. Bertiger, and P. Visser, "Precise GRACE Baseline Determination using GPS," *GPS Solutions*, vol. 9, no. 1, pp. 21–31, 2005.

[27] E. Gill, S. D'Amico, and O. Montenbruck, "Autonomous Formation Flying for the PRISMA Mission," *Journal of Spacecraft and Rockets*, vol. 44, no. 3, pp. 671–681, 2007.

[28] S. D'Amico, J. Ardaens, and S. D. Florio, "Autonomous Formation Flying Based on GPS - PRISMA Flight Results," *Acta Astronautica*, vol. 82, no. 1, pp. 69–79, 2013.

[29] M. Delpech, P.-Y. Guidotti, T. Grelier, and J. Harr, "RF Based Navigation for PRISMA and Other Formation Flying Missions in Earth Orbits," *Advances in the Astronautical Sciences*, vol. 135, no. 2, pp. 1533–1551, 2009.

[30] N. H. Roth, "Navigation and Control Design for the CanX-4/-5 Satellite Formation Flying Mission," Master's thesis, University of Toronto, 2011.

[31] C. R. Tooley, R. K. Black, B. P. Robertson, J. M. Stone, S. E. Pope, and G. T. Davis, "The Magnetospheric Multiscale Constellation," *Space Science Review*, no. 199, pp. 23–76, 2016.

[32] L. B. Winternitz, W. A. Bamford, S. R. Price, J. R. Carpenter, A. C. Long, and M. Farahmand, "Global Positioning System Navigation Above 76,000 KM for NASA's Magnetospheric Multiscale Mission," *Navigation: Journal of The Institute of Navigation*, vol. 64, no. 2, pp. 289–300, 2017.

[33] E. Lorenz, S. Mitchell, T. Sauberlich, C. Paproth, W. Halle, and O. Frauenberger, "Remote Sensing of High Temperature Events by the Firebird Mission," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XL, no. 7, pp. 461–467, 2015.

[34] A. B. Crew, H. E. Spence, J. B. Blake, D. M. Klumpar, B. A. Larsen, T. P. O'Brien, S. Driscoll, M. Handley, J. Legere, S. Longworth, K. Mashburn, E. Mosleh, N. Ryhajlo, S. Smith, L. Springer, and M. Widholm, "First multipoint In Situ Observations of Electron Microbursts: Initial Results from the NSF FIREBIRD II Mission," *Journal of Geophysical Research: Space Physics*, vol. 121, pp. 5272–5283, 2016.

[35] G. Gaias and J.-S. Ardaens, "In-orbit Experience and Lessons Learned from the AVANTI Experiment," *Acta Astronautica*, pp. 1–11, 2017.

[36] R. Garhwal, A. Halder, and M. Sinha, "An Adaptive Fuzzy State Noise Driven Extended Kalman filter for Real Time Orbit Determination," in *58th International Astronautical Congress*, 2007.

[37] F. H. Schlee, C. J. Standish, and N. F. Toda, "Divergence in the Kalman Filter," *AIAA Journal*, vol. 5, no. 6, pp. 1114–1120, 1967.

[38] R. Mehra, "On the Identification of Variances and Adaptive Kalman Filtering," *IEEE Transactions on Automatic Control*, vol. 15, no. 2, pp. 175–184, 1970.

[39] R. K. Mehra, "Approaches to Adaptive Filtering," *IEEE Transactions on Automatic Control*, vol. 17, no. 5, pp. 693–698, 1972.

[40] C. Hide, T. Moore, and M. Smith, "Adaptive Kalman filtering for Low-Cost INS/GPS," *The Journal of Navigation*, vol. 56, no. 1, pp. 143–152, 2003.

[41] P. S. Maybeck, *Stochastic Models, Estimation and Control*. Academic Press, 1982, vol. 2.

[42] L. Zanni, J.-Y. Le Boudec, R. Cherkaoui, and M. Paolone, "A Prediction-Error Covariance Estimator for Adaptive Kalman Filtering in Step-Varying Processes: Application to Power-System State Estimation," *IEEE Transactions on Control Systems Technology*, vol. 25, pp. 1683–1697, 2017.

[43] R. Mehra, S. Seereeram, D. Bayard, and F. Hadaegh, "Adaptive Kalman Filtering, Failure Detection and Identification for Spacecraft Attitude Estimation," in *Proceedings of the 4th IEEE Conference on Control Applications*. IEEE, 1995, pp. 176–181.

[44] F. D. Busse, J. P. How, and J. Simpson, "Demonstration of Adaptive Extended Kalman Filter for Low-Earth-Orbit Formation Estimation using CDGPS," *Navigation: Journal of the Institute of Navigation*, vol. 50, no. 2, pp. 79–93, 2003.

[45] F. Jiancheng and Y. Sheng, "Study on Innovation Adaptive EKF for In-Flight Alignment of Airborne POS," *IEEE Transactions on Instrumentation and Measurement*, vol. 60, no. 4, pp. 1378–1388, 2011.

[46] F. A. Ghaleb, A. Zainal, M. A. Rassam, and A. Abraham, "Improved Vehicle Positioning Algorithm using Enhanced Innovation-based Adaptive Kalman Filter," *Pervasive and Mobile Computing*, vol. 40, pp. 139–155, 2017.

[47] W. Li, D. Gong, M. Liu, J. Chen, and D. Duan, "Adaptive Robust Kalman Filter for Relative Navigation using Global Positioning System," *IET Radar, Sonar and Navigation*, vol. 7, no. 5, pp. 471–479, 2013.

[48] C. Hu, W. Chen, Y. Chen, and D. Liu, "Adaptive Kalman Filtering for Vehicle Navigation," *Journal of Global Positioning Systems*, vol. 2, no. 1, pp. 42–47, 2003.

[49] A. Chatterjee and F. Matsuno, "A Neuro-Fuzzy Assisted Extended Kalman Filter-based Approach for Simultaneous Localization and Mapping (SLAM) Problems," *IEEE Transactions on Fuzzy Systems*, vol. 15, no. 5, pp. 984–997, 2007.

[50] L. A. Zadeh, "Fuzzy Sets," *Information and Control*, vol. 8, no. 3, pp. 338–353, 1965.

[51] ——, "Fuzzy Algorithms," *Information and Control*, vol. 12, no. 2, pp. 94–102, 1968.

[52] ——, "Outline of a New Approach to the Analysis of Complex Systems and Decision Processes," *IEEE Transactions on Systems, Man, and Cybernetics*, no. 1, pp. 28–44, 1973.

[53] J. Z. Sasiadek, Q. Wang, and M. B. Zeremba, "Fuzzy Adaptive Kalman Filtering for INS/GPS Data Fusion," in *Proceedings of the 15th IEEE International Symposium on Intelligent Control*, July 2000, pp. 181–186.

[54] F. L. Lewis, L. Xie, and D. Popa, *Optimal and Robust Estimation*. CRC Press, 2008.

[55] B. Anderson, D. O. Moore, and B. John, *Optimal Filtering*. Dover Publications, 1979.

[56] G. C. Goodwin and S. S. Kwan, *Adaptive Filtering Prediction and Control*. Adaptive Filtering Prediction and Control, 1984.

[57] J. Z. Sasiadek and Q. Wang, "Low Cost Automation using INS/GPS Data Fusion for Accurate Positioning," *Robotica*, vol. 21, pp. 255–260, 2003.

[58] S. Yazdkhasti, J. Z. Sasiadek, and S. Ulrich, "Performance Enhancement for GPS/INS Fusion by Using a Fuzzy Adaptive Unscented Kalman Filter," in *21st International Conference on Methods and Models in Automation and Robotics (MMAR)*, Aug 2016, pp. 1194–1199.

[59] A. L. da Silva and J. J. da Cruz, "Fuzzy Adaptive Extended Kalman Filter for UAV INS/GPS Data Fusion," *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, vol. 38, no. 6, pp. 1671–1688, 2016.

[60] J. Ali, "Strapdown Inertial Navigation System/Astronavigation System Data Synthesis Using Innovation-based Fuzzy Adaptive Kalman Filtering," *IET Science, Measurement and Technology*, vol. 4, no. 5, pp. 246–255, 2010.

[61] C.-H. Tseng, S.-F. Lin, and D.-J. Jwo, "Fuzzy Adaptive Cubature Kalman Filter for Integrated Navigation Systems," *Sensors*, vol. 16, no. 8, pp. 1167–1189, 2016.

[62] J. R. Wertz, *Space Mission Analysis and Design*, 3rd ed. Microcosm Press, 1999.

[63] B. Kuiack, "Spacecraft Formation Guidance and Control on J2-Perturbed Eccentric Orbits," Master's thesis, Carleton University, 2018.

[64] C. M. Lane, "Formation Design and Relative Navigation in High Earth Orbits," Ph.D. dissertation, University of Colorado, 2007.

[65] J. Sullivan and S. D'Amico, "Adaptive Filtering for Maneuver-Free Angles-Only Navigation in Eccentric Orbits," in *27th Spaceflight Mechanics Meeting*, ser. AAS 17-402, 2017.

[66] A. de Ruiter, C. Damaren, and J. R. Forbes, *Spacecraft Dynamics and Control*. Chichester: John Wiley & Sons, 2013.

[67] J. L. Russell, "Kepler's Laws of Planetary Motion: 1609-1666," *The British Journal for the History of Science*, vol. 2, no. 1, pp. 1–24, 1964.

[68] C. A. Kleuver, *Space Flight Dynamics*, 1st ed. Wiley, 2018, ch. 1, pp. 1–6.

[69] I. Newton, *The Principia: The Authoritative Translation: Mathematical Principles of Natural Philosophy*. University of California Press, 2016, B. Cohen (Translator), A. Whitman (Translator), J. Budenz (Translator).

[70] Subcommittee of Space and Aeronautics, "NASA's International Space Station Program: Status and Issues," in *Committee on Science and Technology - One Hundred Tenth Congress*, 2008, p. 4.

[71] W. M. Folkner, J. G. Williams, D. H. Boggs, R. S. Park, and P. Kuchynka, "The Planetary and Lunar Ephemerides DE430 and DE431," NASA Jet Propulsion Laboratory, Tech. Rep. 42-196, 2014.

[72] H. D. Curtis, *Orbital Mechanics for Engineering Students*, 2nd ed. Butterworth-Heinemann, 2009, ch. 4, pp. 203–229.

[73] C. A. Kleuver, *Space Flight Dynamics*, 1st ed. Wiley, 2018, ch. 3, pp. 60–75.

[74] O. Montenbruck and E. Gill, *Satellite Orbits: Models, Methods, and Applications*. Springer, 2001.

[75] M. MacDonald and V. Badescu, *The International Handbook of Space Technology*. Springer Berlin Heidelberg, 2014, ch. 4, pp. 75–88.

[76] H. D. Curtis, *Orbital Mechanics for Engineering Students*. Butterworth-Heinemann, 2013.

[77] T. Alfriend, S. Vadali, P. Gurfil, J. How, and L. Breger, *Spacecraft Formation Flying: Dynamics, Control and Navigation*. Elsevier, 2010.

[78] M. Capderou, *Handbook of Satellite Orbits: From Kepler to GPS*. Springer, Jan. 2014, ch. Chapter 6: Satellite in Real (Perturbed) Orbit, pp. 163–244.

[79] V. A. Chobotov, *Orbital Mechanics*, 3rd ed. American Institute of Aeronautics and Astronautics, 2002, ch. 9, pp. 193–213.

[80] ——, *Orbital Mechanics*, 3rd ed. American Institute of Aeronautics and Astronautics, 2002, ch. 8, pp. 185–190.

[81] I. Harris and W. Priester, "Time-dependent Structure of the Upper Atmosphere," *Journal of the Atmospheric Sciences*, vol. 19, no. 4, pp. 286–301, 1962.

[82] C. E. Roberts, "An Analytic Model for Upper Atmosphere Densities Based Upon Jacchia's 1970 Models," *Celestial Mechanics*, vol. 4, no. 3-4, pp. 368–377, 1971.

[83] A. C. Vigneron, "Nonlinear Filtering for Autonomous Navigation of Spacecraft in Highly Elliptical Orbit," Master's thesis, Carleton University, 2014.

[84] H. D. Curtis, *Orbital Mechanics for Engineering Students*, 3rd ed. Butterworth-Heinemann, 2013, ch. 12: Introduction to Orbital Perturbations, pp. 695–715.

[85] M. Brož, "Yarkovsky Effect and the Dynamics of the Solar System," Ph.D. dissertation, Charles University in Prague, 2006.

[86] N. G.-I. Agency, "World Geodetic System 1984: Standardization Document," Department of Defence, Tech. Rep., 2014.

[87] S. Spiridonova, "Formation Dynamics in Geostationary Ring," *Celestial Mechanics and Dynamical Astronomy*, vol. 125, no. 4, pp. 485–500, 2016.

[88] J. K. Eyer, "A Dynamics and Control Algorithm for Low Earth Orbit Precision Formation Flying Satellites," Ph.D. dissertation, University of Toronto, 2009.

[89] G. W. Hill, "Researches in Lunar Theory, Chapter 1," *American Journal of Mathematics*, vol. 1, no. 1, pp. 5–26, 1878.

[90] ——, "Researches in Lunar Theory, Chapter 2," *American Journal of Mathematics*, vol. 1, no. 2, pp. 129–147, 1878.

[91] ——, "Researches in Lunar Theory, Chapter 3," *American Journal of Mathematics*, vol. 1, no. 3, pp. 245–260, 1878.

[92] W. H. Clohessy, "Terminal Guidance System for Satellite Rendezvous," *Journal of the Aerospace Sciences*, vol. 27, no. 9, pp. 653–658, 1960.

[93] B. Kuiack and S. Ulrich, "Nonlinear Analytical Equations of Relative Motion on J2-Perturbed Eccentric Orbits," *AIAA Journal of Guidance, Control, and Dynamics*, In Print.

[94] O. Montenbruck, M. Delpech, J. Ardaens, N. Delong, and S. D'Amico, "Cross-Validation of GPS- and FFRF-based Relative Navigation for the PRISMA Mission," in *4th ESA Workshop on Satellite Navigation User Equipment Technologies NAVITEC2008*, 2008.

[95] J. Peyrard, D. E. Olmos, A. Agenjo, A. Kron, and A. Cropp, "Design and Prototyping of PROBA-3 Formation Flying System," *International Journal of Space Science and Engineering*, vol. 2, no. 1, pp. 16–34, 2014.

[96] T. V. Peters, J. Brancob, D. Escorial, L. T. Castellani, and A. Cropp, "Mission Analysis for PROBA-3 Nominal Operations," *Acta Astronautica*, vol. 102, pp. 296–310, 2014.

[97] J. M. Ali, N. H. Hoang, M. A. Hussain, and D. Dochain, "Review and Classification of Recent Observers Applied in Chemical Process Systems," *Computers and Chemical Engineering*, vol. 76, pp. 27 – 41, 2015.

[98] A. Gelb, Ed., *Applied Optimal State Estimation*. MIT Press, 1974.

[99] T. Kailath, "An Innovations Approach to Least-Squares Estimation–Part I: Linear Filtering in Additive White Noise," *IEEE Transactions on Automatic Control*, vol. 13, no. 6, pp. 646–655, Dec. 1968.

[100] 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.

[101] E. A. Butcher and J. Wang, "On Kalman Filtering and Observability in Nonlinear Sequential Relative Orbit Estimation," *Journal of Guidance, Control and Dynamics*, vol. 40, no. 9, pp. 2167–2182, 2017.

[102] Z. Bartosiewicz, "Local Observability of Nonlinear Systems," *Systems & Control Letters*, vol. 25, no. 4, pp. 295–298, 1995.

[103] J. C. Butcher, *Numerical Methods for Ordinary Differential Equations*, 2nd ed. John Wiley & Sons, 2008, ch. 33: Runge-Kutta Methods with Error Estimates, pp. 198–202.

[104] O. Montenbruck and E. Gill, "State Interpolation for On-board Navigation Systems," *Aerospace Science and Technology*, vol. 5, no. 3, pp. 209–220, 2001.

[105] J. De Lafontaine, J. Buijs, P. Vuilleumier, P. Van den Braembussche, and K. Mellab, "Development of the PROBA Attitude Control and Navigation Software," in *Spacecraft Guidance, Navigation and Control Systems*, vol. 425, 2000, pp. 427–441.

[106] A. Krishnamoorthy and D. Menon, "Matrix Inversion using Cholesky Decomposition," in *Proc. and Applications (SPA) 2013 Signal Processing: Algorithms, Architectures, Arrangements*, Sep. 2013, pp. 70–72.

[107] A. Leon-Garcia, *Probability, Statistics, and Random Processes for Electrical Engineering*, 3rd ed. Pearson/Prentice Hall, 2008.

[108] ——, *Probability, Statistics, and Random Processes for Electrical Engineering*, 3rd ed. Pearson/Prentice Hall, 2008, ch. Chapter 8: Statistics, pp. 411–469.

[109] V. A. Bavdekar and S. C. Patwardhan, "Identification of Noise Covariances for State Estimation of a Continuous Fermenter using Extended EM Algorithm," in *Proceedings of the 9th International Symposium on Dynamics and Control of Process Systems*, M. Kothare, M. Tade, A. V. Wouwer, and I. Smets, Eds., 2010, pp. 689–694.

[110] A. Leon-Garcia, *Probability, Statistics, and Random Processes for Electrical Engineering*, 3rd ed. Pearson/Prentice Hall, 2008, ch. Chapter 6: Vector Random Variables, pp. 303–346.

[111] V. A. Bavdekar, A. P. Deshpande, and S. C. Patwardhan, "Identification of Process and Measurement Noise Covariance for State and Parameter Estimation using Extended Kalman Filter," *Journal of Process Control*, vol. 21, no. 4, pp. 585–601, 2011.

[112] K. A. Myers and B. D. Tapley, "Adaptive Sequential Estimation with Unknown Noise Statistics," *IEEE Transactions on Automatic Control*, vol. 21, no. 4, pp. 520–523, Aug. 1976.

[113] M. R. Ananthasayanam, M. S. Mohan, N. Naik, and R. M. O. Gemson, "A Heuristic Reference Recursive Recipe for Adaptively Tuning the Kalman Filter Statistics, Part 1: Formulation and Simulation Studies," *Sadhana*, vol. 41, no. 12, pp. 1473–1490, 2016.

[114] N. J. Higham, "Computing a Nearest Symmetric Positive Semidefinite Matrix," *Linear Algebra and its Applications*, vol. 103, pp. 103–118, 1988.

[115] H. E. Rauch, F. Tung, and C. T. Striebel, "Maximum Likelihood Estimates of Linear Dynamic Systems," *AIAA Journal*, vol. 3, no. 8, pp. 1445–1450, 1965.

[116] E. H. Mamdani and S. Assilian, "An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller," *International Journal of Man-Machine Studies*, vol. 7, no. 1, pp. 1–13, 1975.

[117] T. Takagi and M. Sugeno, "Fuzzy Identification of Systems and Its Applications to Modeling and Control," *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-15, no. 1, pp. 116–132, Jan. 1985.

[118] D. J. Jwo, F. C. Chung, and T. P. Weng, *Adaptive Kalman Filter for Navigation Sensor Fusion, Sensor Fusion and its Applications*. Intech, 2010, ch. 4, pp. 65–90.

[119] D.-J. Jwo and T.-S. Cho, "A Practical Note on Evaluating Kalman Filter Performance Optimality and Degradation," *Applied Mathematics and Computation*, vol. 193, no. 2, pp. 482–505, 2007.

[120] K. Passino and S. Yurkovich, *Fuzzy Control*. Addison-Wesley, 1998.

[121] S. Ulrich and J. Z. Sasiadek, "Direct Fuzzy Adaptive Control of a Manipulator with Elastic Joints," *Journal of Guidance, Control, and Dynamics*, vol. 36, no. 1, pp. 311–319, 2012.

[122] T. Rupp, S. D'Amico, O. Montenbruck, and E. Gill, "Autonomous Formation Flying at DLR's German Space Operations Center (GSOC)," in *58th International Astronautical Congress*, 2007, pp. 1–11.

[123] F.-X. Marmet, J. Maureau, M. Calaprice, and J. P. Aguttes, "GPS/Galileo navigation in GTO/GEO orbit," *Acta Astronautica*, vol. 117, pp. 263–276, 2015.

[124] G. Davis, M. Moreau, R. Carpenter, and F. Bauer, "GPS-based Navigation and Orbit Determination for the AMSAT AO-40 Satellite," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference, Monterey, CA*, 2002.

[125] D. C. Woffinder and D. K. Geller, "Relative Angles-Only Navigation and Pose Estimation for Autonomous Orbital Rendezvous," *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 5, pp. 1455–1469, 2007.

[126] T. A. Lovell and T. Lee, "Nonlinear Observability for Relative Satellite Orbits with Angles-Only Measurements," in *24th International Symposium on Space Flight Dynamics*, 2014.

[127] J. Sullivan and S. D'Amico, "Nonlinear Kalman Filtering for Improved Angles-Only Navigation Using Relative Orbital Elements," *Journal of Guidance, Control, and Dynamics*, pp. 1–18, 2017.

[128] S. Ulrich, "Direct Adaptive Control Methodologies for Flexible-Joint Space Manipulators with Uncertainties and Modeling Errors," Ph.D. dissertation, Carleton University, 2012.

[129] D. Folta and A. Hawkins, "Results of NASA's First Autonomous Formation Flying Experiment: Earth Observing-1 (EO-1)," in *AIAA Guidance, Navigation and Control Conference*, 2002.

[130] F. D. Busse, "Precise Formation-State Estimation in Low Earth Orbit Using Carrier Differential GPS," Ph.D. dissertation, Stanford University, 2003.

[131] R. Kroes, "Precise Relative Positioning of Formation Flying Spacecraft using GPS," Ph.D. dissertation, Technical University of Delft, 2006.

[132] R. B. Friend, "Orbital Express Program Summary and Mission Overview," in *Sensors and Systems for Space Applications II*, vol. 6958. International Society for Optics and Photonics, 2008, pp. 695 803 – 6 958 011.

[133] O. Montenbruck, R. Kahle, S. D'Amico, and J.-S. Ardaens, "Navigation and Control of the TanDEM-X Formation," *The Journal of the Astronautical Sciences*, vol. 56, no. 3, p. 341, Sep. 2008.

[134] J. Ardaens and S. D'Amico, "Spacebourne Autonomous Relative Control for Dual Satellite Formations," *Journal of Guidance, Control and Dynamics*, vol. 32, no. 6, pp. 1859–1870, 2009.

[135] G. Grieger, M. Zink, and M. Bachmann, "TanDEM-X: A Radar Interferometer with Two Formation-Flying Satellites," *Acta Astronautica*, vol. 89, pp. 89–98, 2013.

[136] R. Kahle, H. Runge, J.-S. Ardaens, S. Suchandt, and R. Romeiser, "Formation Flying for Along-Track Interferometric Oceanography - First In-Flight Demonstration with TanDEM-X," *Acta Astronautica*, vol. 99, pp. 130–142, 2014.

[137] S. D. Florio and S. D'Amico, "The Precise Autonomous Orbit Keeping Experiment on the PRISMA Mission," *The Journal of Astronautical Sciences*, vol. 56, no. 4, pp. 477–494, 2008.

[138] S. Persson, S. Veldman, and P. Bodin, "PRISMA - A Formation Flying Project in Implementation Phase," *Acta Astronautica*, vol. 65, no. 9, pp. 1360–1374, 2009.

[139] M. T. Zuber, D. E. Smith, M. M. Watkins, S. W. Asmar, A. S. Konopliv, F. G. Lemoine, H. J. Melosh, G. A. Neumann, R. J. Phillips, and S. C. Solomon, "Gravity Field of the Moon from the Gravity Recovery and Interior Laboratory (GRAIL) Mission," *Science*, vol. 339, no. 6120, pp. 668–671, 2013.

[140] W. M. Klipstein, B. W. Arnold, D. G. Enzer, A. A. Ruiz, J. Y. Tien, R. T. Wang, and C. E. Dunn, "The Lunar Gravity Ranging System for the Gravity Recovery and Interior Laboratory (GRAIL) Mission," *Space Science Reviews*, vol. 178, no. 1, pp. 57–76, 2013.

[141] O. Montenbruck, G. Allende-Alba, J. Rosello, M. Tossaint, and F. Zangerl, "Precise Orbit and Baseline Determination for the SAOCOM-CS Bistatic Radar Mission," *Navigation: Journal of The Institute of Navigation*, vol. 65, no. 1, pp. 15–24, 2018.

[142] G. Allende-Alba, O. Montenbruck, S. Hackel, and M. Tossaint, "Relative Positioning of Formation-flying Spacecraft using Single-receiver GPS Carrier Phase Ambiguity Fixing," *GPS Solutions*, vol. 22, no. 3, pp. 1–11, May 2018.

[143] E. Blomberg, L. Ferro-Famil, M. J. Soja, L. M. H. Ulander, and S. Tebaldini, "Forest Biomass Retrieval From L-Band SAR Using Tomographic Ground Backscatter Removal," *IEEE Geoscience and Remote Sensing Letters*, vol. 15, no. 7, pp. 1030–1034, Jul. 2018.

[144] J. Ardaens, S. D'Amico, and A. Cropp, "GPS-based Relative Navigation for the PROBA-3 Formation Flying Mission," *Acta Astronautica*, vol. 91, pp. 341–355, 2013.

[145] M. Landgraf and A. Mestreau-Garreau, "Formation Flying and Mission Design for PROBA-3," *Acta Astronautica*, no. 82, pp. 137–145, 2013.

[146] J. S. Llorente, A. Agenjo, C. Carrascosa, C. de Negueruela, A. Mestreau-Garreau, A. Cropp, and A. Santovincenzo, "PROBA-3: Precise Formation Flying Demonstration Mission," *Acta Astronautica*, vol. 82, no. 1, pp. 38–46, 2013.

[147] A. Moreira, G. Krieger, I. Hajnsek, K. Papathanassiou, M. Younis, P. Lopez-Dekker, S. Huber, M. Villano, M. Pardini, M. Eineder, F. D. Zan, and A. Parizzi, "Tandem-L: A Highly Innovative Bistatic SAR Mission for Global Observation of Dynamic Processes on the Earth's Surface," *IEEE Geoscience and Remote Sensing Magazine*, vol. 3, no. 2, pp. 8–23, Jun. 2015.

[148] M. Eineder, I. Hajnsek, G. Krieger, A. Moreira, and K. Papathanassiou, "Tandem-L: Satellite Mission Proposal for Monitoring Dynamic Processes on the Earth's Surface," German Aerospace Center (DLR), Tech. Rep., 2016.

[149] J. J. C. M. Bik, P. N. A. M. Visser, and O. Jennrich, "LISA Satellite Formation Control," *Advances in Space Research*, vol. 40, no. 1, pp. 25–34, 2007.

[150] H. D. Curtis, *Orbital Mechanics for Engineering Students*, 3rd ed. Butterworth-Heinemann, 2013, ch. 4: Orbits in Three Dimensions, pp. 187–227.

[151] N. Hatten and R. P. Russell, "A Smooth and Robust Harris-Priester Atmospheric Density Model for Low Earth Orbit Applications," *Advances in Space Research*, vol. 59, no. 2, pp. 571–586, 2017.

[152] A. C. Long, J. O. Cappellari, C. E. Velez, and A. J. Fuchs, "Goddard Trajectory Determination System Mathematical Theory," NASA Goddard Space Flight Center, Tech. Rep., 1989.

[153] N. Imagery and M. Agency, "World Geodetic System 1984 - Amendment 1," Department of Defense, Tech. Rep., 2000.

[154] R. E. Kalman and R. S. Bucy, "New Results in Linear Filtering and Prediction Theory," *Transactions of the ASME–Journal of Basic Engineering*, vol. 83, no. 1, pp. 95–108, 1961.

[155] V. C. Aitken, "Nonlinear Discrete-time Systems: Application in Image Sequence Analysis," Ph.D. dissertation, Carleton University, 1995.

[156] J. Sasiadek and S. Ulrich, "Extended Kalman Filtering for Flexible Joint Space Robot Control," in *American Control Conference*, 2011, pp. 1021–1026.

[157] R. Hermann and A. J. Krener, "Nonlinear Controllability and Observability," *IEEE Transactions on Automatic Control*, vol. 22, no. 5, pp. 728–740, 1977.

Appendix A

List of Formation Flying Missions

The following table contains a detailed list of past, current, and planned spacecraft missions that utilize formation flying. The review by Di Mauro *et al.* [15] served as the foundation for this table, and additional entries were added throughout the preparation of this thesis. This list attempts to encapsulate the majority of the high-profile missions in development, however there were undoubtedly missions that have not been accounted for here; as such, this list is likely not completely exhaustive. As a note, missions identified in **bold font** have been discussed in more detail within the thesis in Chapter 1.

Table A.1: List of Formation Flying Missions

Launch	Mission	Organization	Objective	References
2000	CLUSTER	NASA, ESA	To study the interaction between the solar wind and Earth's magnetosphere.	[15]
2001	EO-1/ Landsat-7	NASA	To test a set of advanced land imaging instruments and to experiment with onboard autonomy.	[129]
2002	**GRACE**	DLR, NASA	To derive global high-resolution models of the Earth's gravity field.	[11, 26, 130, 131]
2006	Japan Canada Joint Collaboration Satellites	CSA	To demonstrate the viability of a space technology qualifying system based on micro-satellite (Autonomous Formation Flight experiment with aerodynamic drag control and GPS-based relative navigation).	[15]

Continued on next page

Table A.1 – *Continued from previous page*

Launch	Mission	Organization	Objective	References
2007	Orbital Express	DARPA, NASA	To demonstrate satellite servicing operations including rendezvous, proximity operations, capture, docking, and fluid transfer.	[132]
2010	**TanDEM-X**	DLR	To generate a global, consistent, timely and high-precision DEM of the Earth.	[15, 133–136]
2010	Formation Autonomy Spacecraft	University of Texas	To investigate enabling technologies crucial for satellite formations, including on-orbit microthrust capability, relative navigation, attitude determination, and satellite crosslink communications.	[15]
2010	**PRISMA**	Swedish Space Corporation	To perform GNC and sensor technology experiments for future formation-flying missions.	[1, 13, 15, 27, 29, 137, 138]
2011	**GRAIL**	NASA	To measure the moon's gravity field in unprecedented detail.	[15, 139, 140]
2011	FAST-D	TU Delft	To demonstrate autonomous formation flying using various communication architectures with distributed propulsion systems and micro electromechanical systems technology.	[15]
2011	DICE	Utah University	To map the geomagnetic SEDb plasma bulge and plume formations in Earth's ionosphere.	[15]
2012	AeroCube-4	The Aerospace Corporation (USA)	To demonstrate formation rephasing using atmospheric drag and deployable wings.	[15]
2012	HummerSat-1	DFH Satellite Company, Ltd (China)	To demonstrate the capability of close formation flying technologies such as relative navigation, guidance and control, intersatellite crosslink, and command.	[15]

Continued on next page

Table A.1 – *Continued from previous page*

Launch	Mission	Organization	Objective	References
2012	Shi Jang-9	China National Space Administration	To demonstrate the functionality of a range of newly developed formation-flying techniques and components.	[15]
2014	ANGELS	US Air Force Research Lab	To evaluate techniques for detection, tracking, and characterizing of space objects, as well as attribution of actions in space.	[15]
2014	**CanX-4/ CanX-5**	University of Toronto	To demonstrate formation-flying technology such as differential GPS techniques for high-accuracy relative position determination, fuel-efficient algorithms, and autonomy in maintenance of dual-formation high-accuracy control systems.	[2, 15, 88]
2015	FIREBIRD-II	Los Alamos Labs	To measure characteristics of magnetospheric microbursts in the Van Allen radiation belts.	[15, 34]
2015	MMS	NASA	To investigate the physics of magnetic reconnection.	[15]
2015	Tianwang-1	Nanjing University (China)	To demonstrate formation flying and intersatellite communication between three satellites.	[15]
2016	AAReST	NASA, ESA	To demonstrate the hardware and techniques needed to autonomously assemble a reconfigurable space telescope in orbit.	[15]
2016	AeroCube-7	The Aerospace Corp. (USA)	To demonstrate communications and proximity operations capabilities for CubeSats and other spacecraft.	[15]
2016	SENTINEL-1A/1B	ESA	To provide Copernicus program (previously known as Global Monitoring for Environment and Security) and national services with SAR data for land and ocean monitoring.	[15]

Continued on next page

Table A.1 – *Continued from previous page*

Launch	Mission	Organization	Objective	References
2016	Jason-3/2	NOAA	To provide ocean surface topography measurements.	[15]
2016	**BIROS/ BEESAT-4 (AVANTI)**	DLR, TU Berlin	To demonstrate the feasibility of autonomous, fuel-efficient, and safe proximity operations.	[3, 4, 15, 35]
2016	CARNYVAL-X	NASA, Korea	To validate technologies that allow two spacecraft to fly in formation along an inertial line of sight (*i.e.*, align two spacecraft to an inertial source).	[15]
2016	DelFFI	TU Delft	To characterize low thermosphere with enhanced scientific return by using distributed observation on various geometric baselines and demonstrate autonomous formation flying using various GNC architectures.	[15]
2016	CubeSat Prox-ops Demo	Tyvak Inc.	To demonstrate various rendezvous, proximity operations, and docking scenarios in order to validate and characterize several miniature low power avionics technologies for application to future NASA missions.	[15]
2016	SAMSON	Israel Institute of Technology	To demonstrate long-term autonomous cluster flight of multiple satellites and determine the position of a cooperative terrestrial emitter based on time difference of arrival and/or frequency difference of arrival.	[15]
2017	Prox-1	Georgia Institute of Technology	To demonstrate proximity operations for space situational awareness through the use of a low-thrust propulsion system for orbital maneuvering, as well as visible and infrared imaging for reconnaissance.	[15]
2018	XEUS (Cancelled?)	NASA, ESA	To provide a large-aperture x-ray telescope combined with high spectral and time resolution instruments, capable of investigating matter under extreme conditions and the evolution of the early universe.	[15, 130]

Continued on next page

Table A.1 – *Continued from previous page*

Launch	Mission	Organization	Objective	References
2018	Rascal	St.Louis University	To demonstrate key technologies for proximity operations and space situational awareness such as infrared imaging, 6DOF propulsion, RF proximity sensing, and automated operations.	[15]
2018	SAOCOM-CS	CONAE (Argentia)	To perform tomography mapping of forests and ice formations using L-band bistatic SAR.	[141–143]
2019	**PROBA-3**	ESA	To demonstrate technologies and techniques for highly precise satellite formation flying.	[15, 95, 96, 105, 144–146]
2022	TanDEM-L	DLR	To map biomass, ice structures, terrestrial deformations, and other dynamic process on the Earth's surface using L-band bistatic SAR.	[10, 147, 148]
2025	**EXO-S**	NASA	To discover and analyze terrestrial extrasolar planets using a space telescope.	[14, 15]
2025	Stellar Imager	NASA	To study solar and stellar magnetic activities and their impact on space weather, planetary climates, and life.	[15]
TBD	MASSIM	NASA	To study black holes using an X-ray telescope.	[15]
TBD	mDOT	Stanford University, NASA	To study exozodiacal dust and exoplanets by direct imaging methods while demonstrating a miniaturized occulter/telescope spacecraft pair.	[15]
TBD	SULFRO	Chinese Space Academy	To study the history of the dark age using an ultra-low-frequency observatory.	[15]
TBD	SWIFT	NASA JPL	To investigate the formation-flying technologies when an enormous number (1000 or more) of femtosatellite spacecraft are exploited.	[15]
2034	LISA	NASA, ESA	To study the formation of compact binary stars, and explore the nature of gravity and black holes.	[149]

Appendix B

Essential Astrodynamics

A primer on some of the key topics in orbital mechanics is presented in the following chapter. To provide an introduction into the physics governing satellite navigation, a summary of reference frame descriptions and transformations is covered, along with the governing laws of orbital mechanics resulting from the Keplerian two-body problem. The Classical Orbital Elements (COEs) are briefly discussed, after which a full derivation of the nonlinear equations of relative motion is shown. Lastly, the linearization process reducing these nonlinear equations to the Hill-Clohessy-Wiltshire form is completed.

B.1 Matrix Operations and Reference Frames

Before establishing specific reference frames, let's define a way to identify these frames. Mathematical analysis of spacecraft involves dealing with three-dimensional space, so vector notation is a useful way to describe the position, velocity and acceleration of the systems being considered. Taking a standard unit vector set $\{\hat{x}_1, \hat{y}_1, \hat{z}_1\}$ that forms a basis for a frame $\mathcal{F}_1 \in \mathbb{R}^3$, a generic vector $\vec{r}$ can be written as a linear combination:

$$\vec{r} = r_x\hat{x}_1 + r_y\hat{y}_1 + r_z\hat{z}_1 \qquad \text{or} \qquad \vec{r} = \begin{bmatrix}\hat{x}_1 & \hat{y}_1 & \hat{z}_1\end{bmatrix}\begin{bmatrix}r_x\\ r_y\\ r_z\end{bmatrix} \tag{B.1}$$

From the matrix form on the right, it is clear that the vector $\vec{r}$ can be represented by two components: a column matrix $\mathbf{r}_1$ containing the scalar components $\{r_x, r_y, r_z\}$, and a matrix of unit vectors describing the orientation of the frame. To simplify the notation, this matrix of vectors is defined as a *vectrix* and arranged as a column vector:

$$\vec{\mathcal{F}}_1 = \begin{bmatrix} \hat{x}_1 \\ \hat{y}_1 \\ \hat{z}_1 \end{bmatrix} \tag{B.2}$$

Now, the description of $\vec{r}$ can be shown in the compact form of Eq. (B.3). As will soon be shown, this vectix form is intuitive for describing frames of reference and can easily be used to approach coordinate transformations and rotations. For more information of vectrix notation and the properties of the mathematics, de Ruiter provides an excellent introduction in [66].

$$\vec{r} = \vec{\mathcal{F}}_1^T \mathbf{r}_1 \tag{B.3}$$

The physical length of a vector is independent of the reference frame that is used to described the vector, and can be determined from the Euclidean norm of the components of the vector:

$$r = |\vec{r}| = \|\mathbf{r}\| = \sqrt{\mathbf{r}^T \mathbf{r}} = \sqrt{r_x^2 + r_y^2 + r_z^2} \tag{B.4}$$

For two vectors $\vec{r}_1$ and $\vec{r}_2$ given in a particular reference frame $\mathcal{F}_1$, the vector cross product is given in terms of the vector components by

$$\vec{r}_1 \times \vec{r}_2 = \vec{\mathcal{F}}_1^T \mathbf{r}_1^\times \mathbf{r}_2 \tag{B.5}$$

where the superscript $\times$ indicates the skew-symmetric cross-product operator matrix, which has the property $[\mathbf{r}^\times]^T = -[\mathbf{r}^\times]$, and is defined as

$$\mathbf{r}^\times \triangleq \begin{bmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{bmatrix} \tag{B.6}$$

A principal rotation from reference frame $\mathcal{F}_1$ to $\mathcal{F}_2$ can be represented using a rotation matrix $\mathbf{C}_{21}$. Sometimes a rotation matrix is called a *direction cosine matrix* (DCM), but either way, it can be constructed with vectrix notation as

$$\mathbf{C}_{21} = \vec{\mathcal{F}}_2 \cdot \vec{\mathcal{F}}_1^T \tag{B.7}$$

Rotation matrices are (by design) orthonormal matrices, such that the transpose of the matrix is equivalent to the inverse of the matrix. This is a useful property, as it means that we can transform back-and-forth between frames simply by transposing the DCM.

Additionally, this avoids the computational burden of calculating the inverse of the matrix. Formally, these properties of the rotation operator can be stated

$$\mathbf{C}_{21}\mathbf{C}_{21}^{T} = \mathbf{I}_{3\times 3} \tag{B.8}$$

$$\mathbf{C}_{21}^{-1} = \mathbf{C}_{21}^{T} = \mathbf{C}_{12} \tag{B.9}$$

For the purpose of this research, rotations between the ECI frame $\mathcal{F}_I$, the ECEF frame $\mathcal{F}_F$, the Perifocal frame $\mathcal{F}_P$, and the LVLH frame $\mathcal{F}_L$ are required, and the frames and the respective rotations are presented next. In general, the coordinate transformation of position components $\mathbf{r}_1$ in a frame $\mathcal{F}_1$ to the position components in a different frame $\mathcal{F}_2$, is completed by

$$\mathbf{r}_2 = \mathbf{C}_{21}\mathbf{r}_1 \tag{B.10}$$

Principal Rotations

While working in physical three-dimensional space defined by x, y, and z coordinate axes, it is common to make use of elementary rotation matrices, which denote a rotation about one of the coordinate axes. Using *Euler angles* to demonstrate the principal rotation matrices, a rotation about the x-axis through a *roll* angle ϕ is defined by

$$\mathbf{C}_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \tag{B.11}$$

Similarly a *pitch* rotation through angle θ about the y-axis corresponds to

$$\mathbf{C}_y(\theta) = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \tag{B.12}$$

and lastly the *yaw* rotation through ψ about the z-axes is summarized with

$$\mathbf{C}_z(\psi) = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{B.13}$$

Composite transformations can be used to describe one frame of reference with respect to another, whereby three successive principal rotations matrices can map the orientation of one frame to another. For example, the 3-2-1 rotation sequence that

relates $\mathcal{F}_1$ to $\mathcal{F}_2$ involves a rotation about the principal z-axis of $\mathcal{F}_1$, a rotation about the y-axis of the first intermediate frame, and finally a rotation about the x-axis of the second intermediate frame. The total rotation matrix from Frame $\mathcal{F}_1$ to Frame $\mathcal{F}_2$ can the be found as

$$\begin{aligned}\mathbf{C}_{21}(\phi,\theta,\psi) &= \mathbf{C}_x(\phi)\mathbf{C}_y(\theta)\mathbf{C}_z(\psi) \\ &= \begin{bmatrix} c_\theta c_\psi & c_\theta s_\psi & -s_\theta \\ s_\phi s_\theta c_\psi - c_\phi s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & s_\phi c_\theta \\ c_\phi s_\theta c_\psi + s_\phi s_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi & c_\phi c_\theta \end{bmatrix}\end{aligned} \tag{B.14}$$

where shorthand notations have been used to represent the trigonometric identities, such that $s_\theta = \sin\theta$ and $c_\theta = \cos\theta$. Having established the mathematical formalities necessary to describe orbital mechanics, we are ready to describe the frames of reference specific to the problem of spacecraft navigation.

B.2 Converting Between Reference Frames

The relevant frames of reference used within this research are presented in Section 2.2, and the rotation matrices used to convert between frames are presented here. Although the matrices here are presented specifically for the case of transforming into the ECI or LVLH frames, the reciprocal rotation matrices can be derived by transposing the appropriate matrices thanks to the orthonormal properties of the rotation matrix.

Perifocal to ECI Rotation using the COEs

A position vector in the perifocal (orbital) frame can be transformed into the geocentric equatorial (ECI) frame by three rotations: a first through angle ω about the $\vec{\mathcal{P}}_z$-axis of the perifocal frame; a second through angle i about the node line; and a third through angle Ω about the $\vec{\mathcal{I}}_z$-axis of the ECI frame. Eq. (B.15) contains the necessary 3-1-3 rotation matrix needed to convert from the perifocal frame to the ECI frame.

$$\mathbf{C}_{\mathcal{IP}} = \begin{bmatrix} c_\Omega c_\omega - s_\Omega c_i s_\omega & -c_\Omega s_\omega - s_\Omega c_i c_\omega & s_\Omega s_i \\ s_\Omega c_\omega + c_\Omega c_i s_\omega & -s_\Omega s_\omega + c_\Omega c_i c_\omega & -c_\Omega s_i \\ s_i s_\omega & s_i c_\omega & c_i \end{bmatrix} \tag{B.15}$$

ECEF to ECI Rotation using a Reference Epoch

The transformation from the ECEF frame to the ECI frame is a single rotation about the $\vec{\mathcal{I}}_z$-axis, through an angle defined by the Greenwich Mean Time $\theta_{GMT} = \omega_{\oplus}(t - t_0)$, determined by the Earth's rotation rate $\omega_{\oplus}$ and the time since the reference epoch t_0. Note that this is a negative rotation about the $\vec{\mathcal{I}}_z$-axis, as the corresponding transform from the ECI to ECEF frame would be a positive rotation about the $\vec{\mathcal{I}}_z$-axis. Since the transformations used in this research went from ECEF to ECI, the appropriate matrix is presented here, as:

$$\mathbf{C}_{\mathcal{I}\mathcal{F}} = \begin{bmatrix} \cos(\theta_{GMT}) & -\sin(\theta_{GMT}) & 0 \\ \sin(\theta_{GMT}) & \cos(\theta_{GMT}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{B.16}$$

ECI to LVLH Rotation using ECI States

Components of the position $\mathbf{r}_{\mathcal{I}}$ and velocity $\mathbf{v}_{\mathcal{I}}$ states in the ECI frame can be converted to the LVLH frame through a transformation matrix built from the unit vectors of the LVLH frame, $\vec{\mathcal{L}}_x$, $\vec{\mathcal{L}}_z$, and $\vec{\mathcal{L}}_z$. The rotation sequence is given by

$$\mathbf{C}_{\mathcal{L}\mathcal{I}} = \vec{\mathcal{F}}_L \cdot \vec{\mathcal{F}}_I^T = \begin{bmatrix} \vec{\mathcal{L}}_x & \vec{\mathcal{L}}_y & \vec{\mathcal{L}}_z \end{bmatrix}^T \tag{B.17}$$

and the relative position vector is rotated into the LVLH frame with

$$\mathbf{r}_{\mathcal{L}} = \mathbf{C}_{\mathcal{L}\mathcal{I}}\mathbf{r}_{\mathcal{I}} \tag{B.18}$$

Conversion of the ECI velocity into the LVLH frame makes use of the dynamic *Transport Theorem* for dealing with the rotating reference frame. The relative velocity in the LVLH frame is

$$\mathbf{v}_{\mathcal{L}} = \mathbf{C}_{\mathcal{L}\mathcal{I}}\left(\mathbf{v}_{\mathcal{I}} - \boldsymbol{\omega}_{\mathcal{L}\mathcal{I}}^{\times}\mathbf{r}_{\mathcal{I}}\right) \tag{B.19}$$

The angular velocity vector of the LVLH frame with respect to the stationary ECI frame can be calculated using the cross product of the target position and velocity as

$$\boldsymbol{\omega}_{\mathcal{L}\mathcal{I}} = \frac{\mathbf{r}_{\mathcal{I}}^{\times}\mathbf{v}_{\mathcal{I}}}{\|\mathbf{r}_{\mathcal{I}}\|^2} \tag{B.20}$$

ECEF Coordinates to Geodetic Latitude, Longitude, and Altitude

Given a set of Cartesian coordinates $\{X, Y, Z\}$ in the ECEF frame, the geodetic longitude λ_{long}, latitude ϕ_{lat} and altitude h are obtained using the following algorithm proposed by Montenbruck [74]. Firstly, initialize the term

$$\Delta Z = Z e_{\oplus}^2 \tag{B.21}$$

where the ellipsoidal eccentricity of Earth $e_{\oplus}$ is a function of the flattening factor $f_{\oplus}$:

$$e_{\oplus} = \sqrt{1 - (1 - f_{\oplus})^2} \tag{B.22}$$

Next, iterate through the following three equations until convergence is achieved, or a pre-defined number of iterations have been completed:

$$\sin \phi_{lat} = \frac{Z + \Delta Z}{\sqrt{X^2 + Y^2 + (Z + \Delta Z)^2}} \tag{B.23}$$

$$N = \frac{R_{\oplus}}{\sqrt{1 - e_{\oplus}^2 \sin^2 \phi_{lat}}} \tag{B.24}$$

$$\Delta Z = N e_{\oplus}^2 \sin \phi_{lat} \tag{B.25}$$

where $R_{\oplus}$ is the mean radius of the Earth. After convergence is achieved, the geodetic coordinates can be calculated as

$$\lambda_{lon} = \arctan\left(\frac{Y}{X}\right) \tag{B.26}$$

$$\phi_{lat} = \arctan\left(\frac{Z + \Delta Z}{\sqrt{X^2 + Y^2}}\right) \tag{B.27}$$

$$h = \sqrt{X^2 + Y^2 + (Z + \Delta Z)^2} - N \tag{B.28}$$

To deal with potential numerical issues for spacecraft at the poles, the following steps are implemented to ensure the proper altitude is calculated. In brief, if the X and Y components of the spacecraft are within a metre of the pole, the altitude will be taken as the current Z component minus the semi-minor axis of the Earth $b_{\oplus}$, such that

$$b_{\oplus} = \sqrt{R_{\oplus}^2 \left(1 - e_{\oplus}^2\right)} \tag{B.29}$$

$$h = \begin{cases} |Z| - b_{\oplus} & \text{if } |X| < 1 \text{ m} \quad \text{and} \quad |Y| < 1 \text{ m} \\ h & \text{otherwise} \end{cases} \tag{B.30}$$

ECI Position Vector to Polar Angles

With spacecraft or planetary body position coordinates $\mathbf{r} = [X\ Y\ Z]^T$ in the ECI frame, the angles of right ascension α_{RA} and declination δ_{DEC} to the object are calculated following Algorithm 4.1 in the textbook by Curtis [150]. The magnitude of the position vector is evaluated first:

$$r = \|\mathbf{r}\| = \mathbf{r}^T\mathbf{r} = \sqrt{X^2 + Y^2 + Z^2} \tag{B.31}$$

after which the direction cosines of the position vector are

$$l = \frac{X}{r} \qquad m = \frac{Y}{r} \qquad n = \frac{Z}{r} \tag{B.32}$$

The angle of declination is the calculated from

$$\delta_{DEC} = \arcsin(n) \tag{B.33}$$

and the angle of right ascension is found with

$$\alpha_{RA} = \begin{cases} 0 & \text{if } X = Y = 0 \\ \arccos\left(\frac{l}{\cos(\delta_{DEC})}\right) & \text{if } X, Y \neq 0 \text{ and } m > 0 \\ 2\pi - \arccos\left(\frac{l}{\cos(\delta_{DEC})}\right) & \text{if } X, Y \neq 0 \text{ and } m \leq 0 \end{cases} \tag{B.34}$$

B.3 Conversions with Classical Orbital Elements

Long-term tracking and monitoring of spacecraft orbits commonly deals with the classical orbit elements, while numerical simulations and control lows are frequently defined in terms of position and velocity states. It's therefore useful to be able to quickly convert between these two systems, and the following section provides the necessary equations to do so. Most introductory orbital mechanics textbooks will contain further details on these conversions, but De Ruiter and Curtis [76] are good places for additional information.

B.3.1 Orbital Elements to Inertial States

The COEs are used in this research to define the intial conditions for all of the spacecraft formations scenarios that are analyzed, and these COEs must be converted to the corresponding inertial position $\vec{r}$ and velocity $\vec{v}$ states for use in the numerical orbit propagator. From a given orbital element set

$$\mathbf{oe} = \{\ a,\ e,\ i,\ \Omega,\ \omega,\ \theta\ \}$$

the magnitude of the position vector can be calculated with

$$r = |\vec{r}| = \frac{a(1-e^2)}{1+e\cos\theta} = \frac{p}{1+e\cos\theta} \tag{B.35}$$

where p is known as the *semi-latus rectum*. From the definition of the perifocal reference frame $\mathcal{F}_p$, the position and velocity vectors in the orbital frame can then be found from

$$\vec{r} = \vec{\mathcal{F}}_P^T \mathbf{r}_\mathcal{P} = \vec{\mathcal{F}}_P^T \begin{bmatrix} r\cos(\theta) \\ r\sin(\theta) \\ 0 \end{bmatrix} \tag{B.36}$$

$$\vec{v} = \vec{\mathcal{F}}_P^T \mathbf{v}_\mathcal{P} = \vec{\mathcal{F}}_P^T \begin{bmatrix} -\sqrt{\frac{\mu}{p}}\sin(\theta) \\ \sqrt{\frac{\mu}{p}}\Big(e+\cos(\theta)\Big) \end{bmatrix} \tag{B.37}$$

To rotate the perifocal components $\mathbf{r}_\mathcal{P}$ and $\mathbf{v}_\mathcal{P}$ into the inertial frame of reference, recall the perifocal-to-ECI rotation matrix from Eq. (B.15), and apply it as

$$\mathbf{r}_\mathcal{I} = \mathbf{C}_{\mathcal{IP}}\mathbf{r}_\mathcal{P} \tag{B.38}$$

$$\mathbf{v}_\mathcal{I} = \mathbf{C}_{\mathcal{IP}}\mathbf{v}_\mathcal{P} \tag{B.39}$$

B.3.2 Inertial States to Orbital Elements

Given the inertial position vector $\vec{r}$ and velocity vector $\vec{v}$ of a spacecraft, the semi-major axis can be determined by first calculating the specific energy of the orbit with

$$\varepsilon = \frac{v^2}{2} - \frac{\mu}{r} \tag{B.40}$$

where $r = |\vec{r}|$ is the radial distance of the spacecraft and $v = |\vec{v}|$ is the speed. The semi-major axis a of the orbit is then found from

$$a = -\frac{\mu}{2\varepsilon} \tag{B.41}$$

The specific angular momentum vector $\vec{h}$ of the orbit is

$$\vec{h} = \vec{r} \times \vec{v} \tag{B.42}$$

which is related to the eccentricity vector $\vec{e}$ through

$$\vec{e} = \frac{\vec{v} \times \vec{h}}{\mu} - \frac{\vec{r}}{r} \tag{B.43}$$

To solve for the eccentricity of the orbit, one simply takes the magnitude of the eccentricity vector as $e = |\vec{e}|$. The orientation of the orbit with respect to the ecliptic can then be defined, where the inclination angle i is the angle between the orbit angular momentum vector and the z-axis of the inertial frame. Thus, the inclination is given by

$$i = \arccos\left(\frac{\vec{h} \cdot \vec{\mathcal{I}}_z}{h}\right) = \arccos\left(\frac{h_z}{h}\right) \tag{B.44}$$

The right ascension of the ascending node (RAAN) is defined as the angle between the ascending node and the ECI x-axis, and therefore depends on the nodal vector $\vec{n}$. To find the node vector, use

$$\vec{N} = \vec{\mathcal{I}}_z \times \vec{h} \tag{B.45}$$

and the RAAN is then calculated with

$$\Omega = \arccos\left(\frac{\vec{N} \cdot \vec{\mathcal{I}}_x}{N}\right) = \arccos\left(\frac{N_x}{N}\right) \tag{B.46}$$

where the proper quadrant for the RAAN must be selected based upon

$$\Omega = \begin{cases} 0 \leq \Omega \leq 180°, & \text{if } \vec{N} \cdot \vec{\mathcal{I}}_y \geq 0 \\ 180 < \Omega < 360°, & \text{if } \vec{N} \cdot \vec{\mathcal{I}}_y < 0 \end{cases} \tag{B.47}$$

The direction of perigee is defined from the eccentricity vector, such that the argument of perigee ω can be calculated using

$$\omega = \arccos\left(\frac{\vec{N} \cdot \vec{e}}{Ne}\right) \tag{B.48}$$

Similarly the proper quadrant must be identified for the RAAN, following

$$\omega = \begin{cases} 0 \leq \omega \leq 180°, & \text{if } \vec{e} \cdot \vec{\mathcal{I}}_z \geq 0 \\ 180 < \omega < 360°, & \text{if } \vec{e} \cdot \vec{\mathcal{I}}_z < 0 \end{cases} \tag{B.49}$$

Lastly, the true anomaly θ must be found. Making use of the argument of latitude u, which is the angle between the node vector and the position vector, start by calculating

$$u = \arccos\left(\frac{\vec{N} \cdot \vec{r}}{Nr}\right) \tag{B.50}$$

then determine the correct quadrant with

$$u = \begin{cases} 0 \leq u \leq 180°, & \text{if } \vec{r} \cdot \vec{\mathcal{I}}_z \geq 0 \\ 180 < u < 360°, & \text{if } \vec{r} \cdot \vec{\mathcal{I}}_z < 0 \end{cases} \tag{B.51}$$

and conclude by finding the true anomaly as

$$\theta = u - \omega \tag{B.52}$$

B.4 Positions of the Sun and Moon

The position of the Sun and the Moon with respect to the Earth are calculated using ephemerides published by *The Astronomical Almanac*, as evaluated by the algorithms shown in [84]. Higher fidelity models of Luni-Solar motion can be obtained using Chebyshev polynomials as demonstrated in Montenbruck [74], but the low-precision ephemerides are suitable for the time-scales considered in this research.

Solar Position Vector

To begin, it is assumed that the number of Julian days since J2000, n, is known. The mean anomaly $M_\odot$ and mean longitude of the Sun $L_\odot$ are then calculated by

$$M_\odot = 357.529^\circ + 0.98560023^\circ n \qquad (0^\circ \leq M_\odot \leq 360^\circ) \tag{B.53}$$

$$L_\odot = 280.459^\circ + 0.98564736^\circ n \qquad (0^\circ \leq L_\odot \leq 360^\circ) \tag{B.54}$$

The apparent solar ecliptic longitude $\lambda_\odot$ can then be found with the formula

$$\lambda_\odot = L_\odot + 1.915^\circ \sin(M_\odot) + 0.02^\circ \sin(2M_\odot) \qquad (0^\circ \leq \lambda_\odot \leq 360^\circ) \tag{B.55}$$

and the obliquity angle ε defines the orientation of the ecliptic plane by

$$\varepsilon = 23.439^\circ - \left(3.56 \times 10^{-7}\right) n \tag{B.56}$$

A unit vector $\hat{\mathbf{u}}_\odot$ pointing from the Earth to the Sun is constructed from the longitude and obliquity angles, where

$$\hat{\mathbf{u}}_\odot = \begin{bmatrix} \cos(\lambda_\odot) \\ \sin(\lambda_\odot)\cos(\varepsilon) \\ \sin(\lambda_\odot)\sin(\varepsilon) \end{bmatrix} \tag{B.57}$$

and the distance from the Earth to the Sun $r_\odot$ is given in units of AU by

$$r_\odot = \left(1.00014 - 0.01671\cos(M_\odot) - 0.000140\cos(2M_\odot)\right) \text{AU} \tag{B.58}$$

The geocentric position vector of the Sun $\mathbf{r}_\odot$ is then

$$\mathbf{r}_\odot = r_\odot \hat{\mathbf{u}}_\odot = \begin{bmatrix} r_\odot \cos(\lambda_\odot) \\ r_\odot \sin(\lambda_\odot)\cos(\varepsilon) \\ r_\odot \sin(\lambda_\odot)\sin(\varepsilon) \end{bmatrix} \tag{B.59}$$

Lunar Position Vector

Similarly for the Moon, the obliquity of the ecliptic is required do determine the position of the Moon with respect to the Earth, as is the Julian Day number. However, the Almanac equations for the Moon make use of the number of Julian centuries since J2000, which is denoted T_0. Using the century notation, the obliquity angle of the ecliptic plane is

$$\varepsilon = 23.439° - 0.0130042T_0 \tag{B.60}$$

Calculation of the lunar longitude $\lambda_{☾}$, latitude $\delta_{☾}$, and horizontal parallax $\varphi_{☾}$ is completed with the following set of equations:

$$\lambda_{☾} = b_0 + c_0T_0 + \sum_{i=1}^{6} a_i \sin(b_i + c_iT_0) \qquad (0° \leq \lambda_{☾} \leq 180°) \tag{B.61}$$

$$\delta_{☾} = \sum_{i=1}^{4} d_i \sin(e_i + f_iT_0) \qquad (0° \leq \delta_{☾} \leq 360°) \tag{B.62}$$

$$\varphi_{☾} = g_0 + \sum_{i=1}^{4} g_i \cos(h_i + k_iT_0) \qquad (0° \leq \varphi_{☾} \leq 180°) \tag{B.63}$$

where the lunar coefficients $a_i, b_i, c_i, d_i, e_i, f_i, g_i, h_i, k_i \; \forall i = 1, \ldots, 6$ are provided in Table B.1 on the following page. The unit vector $\hat{\mathbf{u}}_{☾}$ from the Earth to the Moon is calculated from

$$\hat{\mathbf{u}}_{☾} = \begin{bmatrix} \cos(\delta_{☾})\cos(\lambda_{☾}) \\ \cos(\delta_{☾})\sin(\lambda_{☾})\cos(\varepsilon) - \sin(\delta_{☾})\sin(\varepsilon) \\ \cos(\delta_{☾})\sin(\lambda_{☾})\sin(\varepsilon) + \sin(\delta_{☾})\cos(\varepsilon) \end{bmatrix} \tag{B.64}$$

and the Earth-Moon distance $r_{☾}$ is

$$r_{☾} = \frac{R_{\oplus}}{\sin(\varphi_{☾})} \tag{B.65}$$

where $R_{\oplus}$ is the equatorial radius of the Earth. To conclude, the geocentric position vector of the Moon $\mathbf{r}_{☾}$ is calculated with

$$\mathbf{r}_{☾} = r_{☾}\,\hat{\mathbf{u}}_{☾} = \frac{R_{\oplus}}{\sin(\varphi_{☾})} \begin{bmatrix} \cos(\delta_{☾})\cos(\lambda_{☾}) \\ \cos(\delta_{☾})\sin(\lambda_{☾})\cos(\varepsilon) - \sin(\delta_{☾})\sin(\varepsilon) \\ \cos(\delta_{☾})\sin(\lambda_{☾})\sin(\varepsilon) + \sin(\delta_{☾})\cos(\varepsilon) \end{bmatrix} \tag{B.66}$$

Table B.1: Coefficients for Calculating the Position of the Moon [76]

Longitude $\lambda_{☾}$			
i	a_i	b_i	c_i
0	–	218.32	481,267.881
1	6.29	135.0	477,198.87
2	−1.27	259.3	−413,335.36
3	0.66	235.7	890,534.22
4	0.21	269.9	954,397.74
5	−0.19	357.5	35,999.05
6	−0.11	106.5	966,404.03

Latitude $\delta_{☾}$			
i	d_i	e_i	f_i
0	–	–	–
1	5.13	93.3	483,202.03
2	0.28	220.2	960,400.89
3	−0.28	318.3	6,003.15
4	−0.17	217.6	−407,332.21
5	–	–	–
6	–	–	–

Horizontal Parallax $\varphi_{☾}$			
i	g_i	h_i	k_i
0	0.9508	–	–
1	0.0518	135.0	477,198.87
2	0.0095	259.3	−413,335.38
3	0.0078	253.7	890,534.22
4	0.0028	269.9	654,397.70
5	–	–	–
6	–	–	–

B.5 Atmospheric Density Modelling

Of the various models that exist to characterize the properties of the upper atmosphere, the classic Modified Harris-Priester model presents a relatively computationally simple method to estimate the density of the atmosphere at a given altitude. This technique is more accurate then the standard altitude-only exponential models of the atmosphere, but does not account for the wide range of parameters considered by high-fidelity atmospheric models. The paper by Hatten [151] provides a useful overview of the Harris-Priester model, which is summarized here for completeness.

Within the truth orbit propagator for this research, calculation of the atmospheric density begins with known a spacecraft position vector $\vec{r}$, defined in the ECEF reference frame (although it is common to neglect the polar motion of Earth and simply use the spacecraft position defined in the ECI frame [152]. The unit vector describing the direction of the spacecraft is $\hat{\mathbf{r}}$, and the spacecraft geocentric altitude h is evaluated using Eq. (B.30). Similarly, the position of the Sun in the ECI reference frame is defined by $\vec{r}_{\odot}$. Since the location of the diurnal bulge is dependent on the location of the Sun, the unit vector $\hat{\mathbf{u}}_b$ is constructed to indicate the direction of the apex bulge with

$$\hat{\mathbf{u}}_b = \begin{bmatrix} \cos(\delta_{\odot})\cos(\alpha_{\odot} + \lambda_{lag}) \\ \cos(\delta_{\odot})\sin(\alpha_{\odot} + \lambda_{lag}) \\ \sin(\delta_{\odot}) \end{bmatrix} \tag{B.67}$$

where $\alpha_{\odot}$ is the right ascension of the Sun and $\delta_{\odot}$ is the declination of the Sun, as calculated using Eqs. (B.33) and (B.34). The angle λ_{lag} indicates the amount of angular lag between the Sun vector and the apex of the bulge, which is taken as $\lambda_{lag} = 30°$. The deviations in density as a result of the diurnal effects are modelled using a trigonometric power formula as

$$\cos^{n_o}\left(\frac{\Psi}{2}\right) = \left(\frac{1}{2} + \frac{1}{2}\hat{\mathbf{r}} \cdot \hat{\mathbf{u}}_b\right)^{\frac{n_o}{2}} \tag{B.68}$$

Here Ψ represents the angle between the peak of the diurnal density variation and the spacecraft position vector, but calculation of the density variations can be determined without solving this value explicitly. The orbit exponent $2 < n_o \leq 6$ is selected based on the inclination of the orbit to account for variations in density due to latitude, with $n_o = 2$ for equatorial orbits and $n_o = 6$ for polar orbits.

Next, the tabulated density values shown in Table B.2 are used to identify the minimum and maximum atmospheric densities at the given spacecraft altitude h. Minimum and maximum scale heights, H_{m_i} and H_{M_i}, respectively, are calculated

through exponential interpolation by

$$H_{m_i} = \frac{h_i - h_{i+1}}{\ln\left(\frac{\rho_m\left(h_{i+1}\right)}{\rho_m\left(h_i\right)}\right)} \qquad h_i \leq h < h_{i+1} \tag{B.69}$$

$$H_{M_i} = \frac{h_i - h_{i+1}}{\ln\left(\frac{\rho_M\left(h_{i+1}\right)}{\rho_M\left(h_i\right)}\right)} \qquad h_i \leq h < h_{i+1} \tag{B.70}$$

Here, index i corresponds to the i^{th} entry in the density table, and identifies the appropriate upper and lower boundaries that altitude layer. Similar minimum and maximum densities are calculated from the table as well, where

$$\rho_{m_i}(h) = \rho_{m_i}(h) \exp\left(\frac{h_i - h}{H_{m_i}}\right) \qquad h_i \leq h < h_{i+1} \tag{B.71}$$

$$\rho_{M_i}(h) = \rho_{M_i}(h) \exp\left(\frac{h_i - h}{H_{M_i}}\right) \qquad h_i \leq h < h_{i+1} \tag{B.72}$$

Lastly, the density of the atmosphere at the current altitude of the spacecraft is calculated using Eq. (2.15), which is restated here for convenience:

$$\rho_{atm}(h) = \rho_m(h) + \left(\rho_M(h) - \rho_m(h)\right) \cos^{n_o}\left(\frac{\Psi}{2}\right) \tag{B.73}$$

Careful attention should be paid to the units of the tabulated values of the scale heights and density values for the Harris-Priester coefficients. The following page contains the these coefficients for these values, in Table B.2.

Table B.2: Harris-Priester density coefficients for altitudes ranging from 100 km to 1000 km, for nominal solar activity [74].

h [km]	ρ_m [g/km^3]	ρ_M [g/km^3]	h [km]	ρ_m [g/km^3]	ρ_M [g/km^3]
100	497400.0	497400.0	420	1.558	5.684
120	24900.0	24900.0	440	1.091	4.355
130	8377.0	8710.0	460	0.7701	3.362
140	3899.0	4059.0	480	0.5474	2.612
150	2122.0	2215.0	500	0.3916	2.042
160	1263.0	1344.0	520	0.2819	1.605
170	800.8	875.8	540	0.2042	1.267
180	528.3	601.0	560	0.1488	1.005
190	361.7	429.7	580	0.1092	0.7997
200	255.7	316.2	600	0.08070	0.6390
210	183.9	239.6	620	0.06012	0.5123
220	134.1	185.3	640	0.04519	0.4121
230	99.49	145.5	660	0.03430	0.3325
240	74.88	115.7	680	0.02632	0.2691
250	57.09	93.08	700	0.02043	0.2185
260	44.03	75.55	720	0.01607	0.1779
270	34.30	61.82	740	0.01281	0.1452
280	26.97	50.95	760	0.01036	0.1190
290	21.39	42.26	780	0.008496	0.09776
300	17.08	35.26	800	0.007069	0.08059
320	10.99	25.11	840	0.004680	0.05741
340	7.214	18.19	880	0.003200	0.04210
360	4.824	13.37	920	0.002210	0.03130
380	3.274	9.955	960	0.001560	0.02360
400	2.249	7.492	1000	0.001150	0.01810

B.6 Deriving the Equations of Relative Motion

The satellite formation considered within this research consists of two spacecraft. A *target* spacecraft is typically assumed to be passively travelling along its orbit and a manoeuvrable *chaser* controls the relative state between the two. Although in practice both of these spacecraft could be equipped with thrusters capable of producing control forces, this initial analysis assumes passive flight such that the two-body equations of motion for the target and chaser spacecraft are given respectfully by

$$\ddot{\vec{r}}_t = -\mu \frac{\vec{r}_t}{r_t^3} \qquad\qquad \ddot{\vec{r}}_c = -\mu \frac{\vec{r}_c}{r_c^3} \tag{B.74}$$

where r_t is the magnitude of the target's position vector $\vec{r}_t$ with respect to the ECI reference frame $\vec{\mathcal{F}}_I$. Note that r_t can be calculated from Eq. (B.35) above. Defining the relative position of the chaser with respect to the target by the position vector $\vec{\rho}$, it can be seen from Figure 2.4 that differencing the target and chaser position vectors leads to

$$\vec{\rho} = \vec{r}_c - \vec{r}_t \tag{B.75}$$

To determine the governing differential equations of motion $\ddot{\vec{\rho}}$, we will form descriptions of the motion based on the kinematics of the rotating frame, and set these results equal to the two-body dynamics. Taking advantage of our definition of the LVLH frame, the relative position vector can be represented by the component set $\{x, y, z\}$ within the LVLH reference frame where the *radial* component x and the *along-track* component y capture the relative motion within the orbital plane of the target spacecraft. The out-of-plane motion is described by the *cross-track* term z.

$$\vec{\rho} = \vec{\mathcal{F}}_L^T \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{B.76}$$

Since we have the inertial chaser position vector $\vec{r}_c$, the inertial chaser velocity $\dot{\vec{r}}_c$ and acceleration $\ddot{\vec{r}}_c$ can be obtained through the kinematics of the relative motion. Determining the velocity and acceleration involves taking two time derivatives of the position vector, but the LVLH frame is rotating with respect to the inertial frame and so additional terms $\mathring{\vec{r}}_c$ and $\mathring{\mathring{\vec{r}}}_c$ are used to denote the time derivatives as seen in the

rotating reference frame.

$$\dot{\vec{r}}_c = \mathring{\vec{r}}_c + \vec{\omega}_{LI} \times \vec{r}_c \tag{B.77}$$

$$\ddot{\vec{r}}_c = \mathring{\ddot{\vec{r}}}_c + \dot{\vec{\omega}}_{LI} \times \vec{r}_c + 2\vec{\omega}_{LI} \times \mathring{\vec{r}}_c + \vec{\omega}_{LI} \times (\vec{\omega}_{LI} \times \vec{r}_c) \tag{B.78}$$

Using the true anomaly θ and the magnitude of the radius vector r_t of the target spacecraft, the components of the translational and angular velocity vectors can be written in the LVLH reference frame:

$$\vec{r}_c = \vec{\mathcal{F}}_L^T \begin{bmatrix} r_t + x \\ y \\ z \end{bmatrix} \qquad \mathring{\vec{r}}_c = \vec{\mathcal{F}}_L^T \begin{bmatrix} \dot{r}_t + \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} \qquad \mathring{\ddot{\vec{r}}}_c = \vec{\mathcal{F}}_L^T \begin{bmatrix} \ddot{r}_t + \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix}$$

$$\vec{\omega}_{LI} = \vec{\mathcal{F}}_L^T \begin{bmatrix} 0 \\ 0 \\ \dot{\theta} \end{bmatrix} \qquad \dot{\vec{\omega}}_{LI} = \vec{\mathcal{F}}_L^T \begin{bmatrix} 0 \\ 0 \\ \ddot{\theta} \end{bmatrix}$$

Inserting these relations into Eq. (B.78) for $\ddot{\vec{r}}_c$ and completing the matrix multiplication leads to

$$\ddot{\vec{r}}_c = \vec{\mathcal{F}}_L^T \begin{bmatrix} \ddot{r}_t + \ddot{x} - \ddot{\theta} y - 2\dot{\theta}\dot{y} - \dot{\theta}^2 (r_t + x) \\ \ddot{y} + \ddot{\theta} (r_t + x) + 2\dot{\theta} (\dot{r}_t + \dot{x}) - \dot{\theta}^2 y \\ \ddot{z} \end{bmatrix} \tag{B.79}$$

Now we can perform a similar derivation on the target position vector $\vec{r}_t$. Orientation of the LVLH frame is established based on the target position vector, and this constrains the motion to the $\vec{\mathcal{L}}_x$ direction. Thus, the relevant kinematic terms are

$$\dot{\vec{r}}_t = \mathring{\vec{r}}_t + \vec{\omega}_{LI} \times \vec{r}_t \tag{B.80}$$

$$\ddot{\vec{r}}_t = \mathring{\ddot{\vec{r}}}_t + \dot{\vec{\omega}}_{LI} \times \vec{r}_t + 2\vec{\omega}_{LI} \times \mathring{\vec{r}}_t + \vec{\omega}_{LI} \times (\vec{\omega}_{LI} \times \vec{r}_t) \tag{B.81}$$

$$\vec{r}_t = \vec{\mathcal{F}}_L^T \begin{bmatrix} r_t \\ 0 \\ 0 \end{bmatrix} \qquad \mathring{\vec{r}}_t = \vec{\mathcal{F}}_L^T \begin{bmatrix} \dot{r}_t \\ 0 \\ 0 \end{bmatrix} \qquad \mathring{\ddot{\vec{r}}}_t = \vec{\mathcal{F}}_L^T \begin{bmatrix} \ddot{r}_t \\ 0 \\ 0 \end{bmatrix}$$

Inserting these terms into Eq. (B.81) and reducing leads to the following expression for the acceleration of the target spacecraft:

$$\ddot{\vec{r}}_t = \vec{\mathcal{F}}_L^T \begin{bmatrix} \ddot{r}_t - \dot{\theta}^2 r_t \\ 0 \\ 0 \end{bmatrix} \tag{B.82}$$

Thus, using the kinematic-derivation equations for $\ddot{\vec{r}}_t$ and $\ddot{\vec{r}}_c$, the first expression for the relative position vector becomes

$$\ddot{\vec{\rho}}_1 = \ddot{\vec{r}}_c - \ddot{\vec{r}}_t = \vec{\mathcal{F}}_L^T \begin{bmatrix} \ddot{x} - \ddot{\theta} y - 2\dot{\theta}\dot{y} - \dot{\theta}^2 (r_t + x) + \dot{\theta}^2 r_t \\ \ddot{y} + \ddot{\theta} (r_t + x) + 2\dot{\theta} (\dot{r}_t + \dot{x}) - \dot{\theta}^2 y \\ \ddot{z} \end{bmatrix} \tag{B.83}$$

From the two-body equations of motion we know that the chaser and target acceleration vectors can be defined by Eq. (B.74) so let's put these into the LVLH frame with

$$\ddot{\vec{r}}_c = -\vec{\mathcal{F}}_L^T \frac{\mu}{r_c^3} \begin{bmatrix} r_t + x \\ y \\ z \end{bmatrix} \qquad \ddot{\vec{r}}_t = -\vec{\mathcal{F}}_L^T \frac{\mu}{r_t^3} \begin{bmatrix} r_t \\ 0 \\ 0 \end{bmatrix} = -\vec{\mathcal{F}}_L^T \frac{\mu}{r_t^2} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \tag{B.84}$$

Constructing the second expression for the relative motion gives

$$\ddot{\vec{\rho}}_2 = \ddot{\vec{r}}_c - \ddot{\vec{r}}_t = -\vec{\mathcal{F}}_L^T \begin{bmatrix} \frac{\mu}{r_c^3} (r_t + x) + \frac{\mu}{r_t^2} \\ \frac{\mu}{r_c^3} y \\ \frac{\mu}{r_c^3} z \end{bmatrix} \tag{B.85}$$

Now setting the two representations (B.83) and (B.85) equal such that $\ddot{\vec{\rho}}_1 = \ddot{\vec{\rho}}_2$,

$$\vec{\mathcal{F}}_L^T \begin{bmatrix} \ddot{x} - \ddot{\theta} y - 2\dot{\theta}\dot{y} - \dot{\theta}^2 (r_t + x) + \dot{\theta}^2 r_t \\ \ddot{y} + \ddot{\theta} (r_t + x) + 2\dot{\theta} (\dot{r}_t + \dot{x}) - \dot{\theta}^2 y \\ \ddot{z} \end{bmatrix} = -\vec{\mathcal{F}}_L^T \begin{bmatrix} \frac{\mu}{r_c^3} (r_t + x) + \frac{\mu}{r_t^2} \\ \frac{\mu}{r_c^3} y \\ \frac{\mu}{r_c^3} z \end{bmatrix} \tag{B.86}$$

Isolating for our variables of interest and gathering terms, we can further decompose the matrices into component form since both representations are in $\vec{\mathcal{F}}_L$. Identifying equations for $\ddot{x}$, $\ddot{y}$ and $\ddot{z}$ we obtain

$$\ddot{x} = \ddot{\theta}y + 2\dot{\theta}\dot{y} + \dot{\theta}^2 x - \frac{\mu}{r_c^3}(r_t + x) - \frac{\mu}{r_t^2} \tag{B.87}$$

$$\ddot{y} = -\ddot{\theta}(r_t + x) - 2\dot{\theta}(\dot{r}_t + \dot{x}) + \dot{\theta}^2 y - \frac{\mu}{r_c^3} y \tag{B.88}$$

$$\ddot{z} = -\frac{\mu}{r_c^3} z \tag{B.89}$$

While the three equations above describe the radial, along-track and cross-track motion in the LVLH frame, they provide no information about the position of the frame itself. Inspecting the equations reveals that there are still differential terms that must be accounted for, specifically the true anomaly θ and the target radius r_t. If we can define equations for both of these terms, then the entire system will be defined.

First we can remember the relation between orbital angular momentum and the orbital position, which for the target would be $h = r_t^2\dot{\theta}$. Taking the time derivative of the angular momentum gives $\dot{h} = 2r_t\dot{r}_t\dot{\theta} + r_t^2\ddot{\theta}$, and since this derivation assumes the target orbit is Keplerian and angular momentum is conserved, setting this derivative to zero leads to the equation for the true anomaly acceleration

$$\ddot{\theta} = -2\frac{\dot{r}_t}{r_t}\dot{\theta} \tag{B.90}$$

An equation for the target radius can be found by looking at the previous $\ddot{\vec{r}}_t$ equations that were derived. Setting Eq. (B.82) equal to Eq. (B.84), we can conclude that the differential equation of motion for the target spacecraft position is

$$\ddot{r}_t = \dot{\theta}^2 r_t - \frac{\mu}{r_t^2} \tag{B.91}$$

With the two new differential equations defining the motion of the target spacecraft, we can make the appropriate substitutions into the previous equations for $\ddot{x}$, $\ddot{y}$ and $\ddot{z}$. It should also be noted that although r_c exists in the final equations, it is not a necessary state variable and therefore has no governing differential equation. Geometrically we see that $r_c = \sqrt{(r_t + x)^2 + y^2 + z^2}$, so r_c is simply a function of the other states.

This completes the derivation of the equations summarized on the following page, which make up the 10-dimensional nonlinear system for spacecraft formation flying shown in (B.92).

$$\begin{aligned}
\ddot{x} &= \dot{\theta}^2 x + 2\dot{\theta}\left(\dot{y} - y\frac{\dot{r}_t}{r_t}\right) + \frac{\mu}{r_t^2} - \frac{\mu}{r_c^3}(r_t + x) \\
\ddot{y} &= \dot{\theta}^2 y - 2\dot{\theta}\left(\dot{x} - x\frac{\dot{r}_t}{r_t}\right) - \frac{\mu}{r_c^3} y \\
\ddot{z} &= -\frac{\mu}{r_c^3} z \\
\ddot{\theta} &= -2\frac{\dot{r}_t}{r_t}\dot{\theta} \\
\ddot{r}_t &= \dot{\theta}^2 r_t - \frac{\mu}{r_t^2}
\end{aligned} \tag{B.92}$$

The corresponding linearized Jacobian matrix $\mathbf{F}_k$ of the relative dynamics is shown below, and the corresponding partial derivatives are included on the following pages.

$$\mathbf{F}_k = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\ddot{x}_x & \ddot{x}_y & \ddot{x}_z & 0 & \ddot{x}_{r_t} & 0 & \ddot{x}_{\dot{y}} & 0 & \ddot{x}_{\dot{\theta}} & \ddot{x}_{\dot{r}_t} \\
\ddot{y}_x & \ddot{y}_y & \ddot{y}_z & 0 & \ddot{y}_{r_t} & \ddot{y}_{\dot{x}} & 0 & 0 & \ddot{y}_{\dot{\theta}} & \ddot{y}_{\dot{r}_t} \\
\ddot{z}_x & \ddot{z}_y & \ddot{z}_z & 0 & \ddot{z}_{r_t} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \ddot{\theta}_{r_t} & 0 & 0 & 0 & \ddot{\theta}_{\dot{\theta}} & \ddot{\theta}_{\dot{r}_t} \\
0 & 0 & 0 & 0 & \ddot{r}_{t_{r_t}} & 0 & 0 & 0 & \ddot{r}_{t_{\dot{\theta}}} & 0
\end{bmatrix} \tag{B.93}$$

B.6.1 Partial Derivatives of X-EOM

$$
\begin{aligned}
\ddot{x}_x &= \frac{\partial \ddot{x}}{\partial x} = \frac{\mu}{r_c^3}\left[3\left(\frac{r_t + x}{r_c}\right)^2 - 1\right] + \dot{\theta}^2 \\
\ddot{x}_y &= \frac{\partial \ddot{x}}{\partial y} = 3\mu\left(\frac{r_t + x}{r_c^5}\right) y - 2\dot{\theta}\frac{\dot{r}_t}{r_t} \\
\ddot{x}_z &= \frac{\partial \ddot{x}}{\partial z} = 3\mu\left(\frac{r_t + x}{r_c^5}\right) z \\
\ddot{x}_\theta &= \frac{\partial \ddot{x}}{\partial \theta} = 0 \\
\ddot{x}_{r_t} &= \frac{\partial \ddot{x}}{\partial r_t} = 2\dot{\theta}\frac{\dot{r}_t}{r_t^2} y - 2\frac{\mu}{r_t^3} + \frac{\mu}{r_c^3}\left[3\left(\frac{r_t + x}{r_c}\right)^2 - 1\right] \\
\ddot{x}_{\dot{x}} &= \frac{\partial \ddot{x}}{\partial \dot{x}} = 0 \\
\ddot{x}_{\dot{y}} &= \frac{\partial \ddot{x}}{\partial \dot{y}} = 2\dot{\theta} \\
\ddot{x}_{\dot{z}} &= \frac{\partial \ddot{x}}{\partial \dot{z}} = 0 \\
\ddot{x}_{\dot{\theta}} &= \frac{\partial \ddot{x}}{\partial \dot{\theta}} = 2\left(\dot{\theta} x + \dot{y} - \frac{\dot{r}_t}{r_t} y\right) \\
\ddot{x}_{\dot{r}_t} &= \frac{\partial \ddot{x}}{\partial \dot{r}_t} = -2\frac{\dot{\theta}}{r_t} y
\end{aligned}
\tag{B.94}
$$

B.6.2 Partial Derivatives of Y-EOM

$$
\begin{aligned}
\ddot{y}_x &= \frac{\partial \ddot{y}}{\partial x} = 2\dot{\theta}\frac{\dot{r_t}}{r_t} + 3\mu\left(\frac{r_t + x}{r_c^5}\right) y \\
\ddot{y}_y &= \frac{\partial \ddot{y}}{\partial y} = \frac{\mu}{r_c^3}\left[3\left(\frac{y}{r_c}\right)^2 - 1\right] + \dot{\theta}^2 \\
\ddot{y}_z &= \frac{\partial \ddot{y}}{\partial z} = 3\mu\left(\frac{y}{r_c^5}\right) z \\
\ddot{y}_\theta &= \frac{\partial \ddot{y}}{\partial \theta} = 0 \\
\ddot{y}_{r_t} &= \frac{\partial \ddot{y}}{\partial r_t} = 3\mu\left(\frac{r_t + x}{r_c^5}\right) y - 2\dot{\theta}\frac{\dot{r_t}}{r_t^2} x \\
\ddot{y}_{\dot{x}} &= \frac{\partial \ddot{y}}{\partial \dot{x}} = -2\dot{\theta} \\
\ddot{y}_{\dot{y}} &= \frac{\partial \ddot{y}}{\partial \dot{y}} = 0 \\
\ddot{y}_{\dot{z}} &= \frac{\partial \ddot{y}}{\partial \dot{z}} = 0 \\
\ddot{y}_{\dot{\theta}} &= \frac{\partial \ddot{y}}{\partial \dot{\theta}} = 2\left(\dot{\theta} y - \dot{x} + \frac{\dot{r_t}}{r_t} x\right) \\
\ddot{y}_{\dot{r_t}} &= \frac{\partial \ddot{y}}{\partial \dot{r_t}} = 2\frac{\dot{\theta}}{r_t} x
\end{aligned}
\tag{B.95}
$$

B.6.3 Partial Derivatives of Z-EOM

$$
\begin{aligned}
\ddot{z}_x &= \frac{\partial \ddot{z}}{\partial x} = 3\mu \left(\frac{r_t + x}{r_c^5} \right) z \\
\ddot{z}_y &= \frac{\partial \ddot{z}}{\partial y} = 3\mu \left(\frac{y}{r_c^5} \right) z \\
\ddot{z}_z &= \frac{\partial \ddot{z}}{\partial z} = \frac{\mu}{r_c^3} \left[3 \left(\frac{z}{r_c} \right)^2 - 1 \right] \\
\ddot{z}_\theta &= \frac{\partial \ddot{z}}{\partial \theta} = 0 \\
\ddot{z}_{r_t} &= \frac{\partial \ddot{z}}{\partial r_t} = 3\mu \left(\frac{r_t + x}{r_c^5} \right) z \\
\ddot{z}_{\dot{x}} &= \frac{\partial \ddot{z}}{\partial \dot{x}} = 0 \\
\ddot{z}_{\dot{y}} &= \frac{\partial \ddot{z}}{\partial \dot{y}} = 0 \\
\ddot{z}_{\dot{z}} &= \frac{\partial \ddot{z}}{\partial \dot{z}} = 0 \\
\ddot{z}_{\dot{\theta}} &= \frac{\partial \ddot{z}}{\partial \dot{\theta}} = 0 \\
\ddot{z}_{\dot{r}_t} &= \frac{\partial \ddot{z}}{\partial \dot{r}_t} = 0
\end{aligned}
\tag{B.96}
$$

B.6.4 Partial Derivatives of θ-EOM

$$
\begin{aligned}
\ddot{\theta}_x &= \frac{\partial \ddot{\theta}}{\partial x} = 0 \\
\ddot{\theta}_y &= \frac{\partial \ddot{\theta}}{\partial y} = 0 \\
\ddot{\theta}_z &= \frac{\partial \ddot{\theta}}{\partial z} = 0 \\
\ddot{\theta}_\theta &= \frac{\partial \ddot{\theta}}{\partial \theta} = 0 \\
\ddot{\theta}_{r_t} &= \frac{\partial \ddot{\theta}}{\partial r_t} = 2\frac{\dot{r}_t}{r_t^2}\dot{\theta} \\
\ddot{\theta}_{\dot{x}} &= \frac{\partial \ddot{\theta}}{\partial \dot{x}} = 0 \\
\ddot{\theta}_{\dot{y}} &= \frac{\partial \ddot{\theta}}{\partial \dot{y}} = 0 \\
\ddot{\theta}_{\dot{z}} &= \frac{\partial \ddot{\theta}}{\partial \dot{z}} = 0 \\
\ddot{\theta}_{\dot{\theta}} &= \frac{\partial \ddot{\theta}}{\partial \dot{\theta}} = -2\frac{\dot{r}_t}{r_t} \\
\ddot{\theta}_{\dot{r}_t} &= \frac{\partial \ddot{\theta}}{\partial \dot{r}_t} = -2\frac{\dot{\theta}}{r_t}
\end{aligned}
\tag{B.97}
$$

B.6.5 Partial Derivatives of r_t-EOM

$$
\begin{aligned}
\ddot{r}_{t_x} &= \frac{\partial \ddot{r}_t}{\partial x} = 0 \\
\ddot{r}_{t_y} &= \frac{\partial \ddot{r}_t}{\partial y} = 0 \\
\ddot{r}_{t_z} &= \frac{\partial \ddot{r}_t}{\partial z} = 0 \\
\ddot{r}_{t_\theta} &= \frac{\partial \ddot{r}_t}{\partial \theta} = 0 \\
\ddot{r}_{t_{r_t}} &= \frac{\partial \ddot{r}_t}{\partial r_t} = \dot{\theta}^2 + 2\frac{\mu}{r_t^3} \\
\ddot{r}_{t_{\dot{x}}} &= \frac{\partial \ddot{r}_t}{\partial \dot{x}} = 0 \\
\ddot{r}_{t_{\dot{y}}} &= \frac{\partial \ddot{r}_t}{\partial \dot{y}} = 0 \\
\ddot{r}_{t_{\dot{z}}} &= \frac{\partial \ddot{r}_t}{\partial \dot{z}} = 0 \\
\ddot{r}_{t_{\dot{\theta}}} &= \frac{\partial \ddot{r}_t}{\partial \dot{\theta}} = 2\dot{\theta} r_t \\
\ddot{r}_{t_{\dot{r}_t}} &= \frac{\partial \ddot{r}_t}{\partial \dot{r}_t} = 0
\end{aligned}
\tag{B.98}
$$

B.6.6 Hill-Clohessy-Wiltshire Relative Dynamics

The set of nonlinear equations derived previously are exact for general orbits of the target spacecraft. However, some satellite orbits are nearly circular in practice, and the relative positions may be small compared to the orbital radius of the target spacecraft. It is therefore useful to make several simplifying assumptions that lead to a set of analytical equations of motion with closed-form solutions.

Firstly if we assume the target spacecraft orbit is circular, the radius r_t and true anomaly rate $\dot{\theta}$ must be constant. This means $\ddot{r}_t = \dot{r}_t = 0$, $\ddot{\theta} = 0$ and $\dot{\theta} = n$, where n is the mean orbital motion of the target. This reduces the 5 exact nonlinear equations of relative motion to the 3 equations below

$$\begin{aligned}
\ddot{x} &= n^2 x + 2n\dot{y} + \frac{\mu}{r_t^2} - \frac{\mu}{r_c^3}(r_t + x) \\
\ddot{y} &= n^2 y - 2n\dot{x} - \frac{\mu}{r_c^3} y \\
\ddot{z} &= -\frac{\mu}{r_c^3} z
\end{aligned}$$

The system can further be simplified if we assume $r_t \gg \{x, y, z\}$, such that we can approximate r_c with a Taylor Series expansion. Taking only the first-order terms for the expansion of r_c, neglecting higher-order terms and knowing $n = \sqrt{\mu / r_t^3}$, we get

$$\begin{aligned}
-\frac{\mu}{r_c^3}(r_t + x) \approx n^2 \left(2x - r_t\right) \approx 2n^2 x - \frac{\mu}{r_t^2} \\
-\frac{\mu}{r_c^3} y \approx -n^2 y \\
-\frac{\mu}{r_c^3} z \approx -n^2 z
\end{aligned}$$

Inserting these approximations into the differential equations leads to the final form of the linearized equations of motion. Expressed as a second-order system of homogeneous differential equations, the Hill's equations of relative motion are [90]:

$$\ddot{x} - 3n^2 x - 2n\dot{y} = 0 \tag{B.99}$$

$$\ddot{y} + 2n\dot{x} = 0 \tag{B.100}$$

$$\ddot{z} + n^2 z = 0 \tag{B.101}$$

These equations define decoupled out-of-plane and in-plane modes, where we see that the cross-track motion is now pure oscillation. Secular drift can be seen in the along-track components, which means that in general, the in-plane motion will be

unstable unless a corrective thrust manoeuvre is applied. The analytic solutions to the Hill's equations were developed by Clohessy & Wiltshire [92], and the resulting Clohessy-Wiltshire (CW) equations can be written as follows:

$$x(t) = 4x_0 + \frac{2\dot{y}_0}{n} + \frac{\dot{x}_0}{n}\sin(nt) - \left(3x_0 + \frac{2\dot{y}_0}{n}\right)\cos(nt) \tag{B.102}$$

$$y(t) = y_0 - \frac{2\dot{x}_0}{n} - \Big(6nx_0 + 3\dot{y}_0\Big)t + \left(6x_0 + \frac{4\dot{y}_0}{n}\right)\sin(nt) + \frac{2\dot{x}_0}{n}\cos(nt) \tag{B.103}$$

$$z(t) = z_0\cos(nt) + \frac{\dot{z}_0}{n}\sin(nt) \tag{B.104}$$

The CW equations are commonly put into compact matrix notation, where the relative position $\delta\mathbf{r}(t)$ and relative velocity $\delta\mathbf{v}(t)$ components in the LVLH frame can be written as a function of time with

$$\delta\mathbf{r}(t) = \mathbf{\Phi}_{rr}(t)\delta\mathbf{r}_0 + \mathbf{\Phi}_{rv}(t)\delta\mathbf{v}_0 \tag{B.105}$$

$$\delta\mathbf{v}(t) = \mathbf{\Phi}_{vr}(t)\delta\mathbf{r}_0 + \mathbf{\Phi}_{vv}(t)\delta\mathbf{v}_0 \tag{B.106}$$

where the appropriate submatrices are:

$$\mathbf{\Phi}_{rr}(t) = \left[\begin{array}{cc:c} 4 - 3\cos(nt) & 0 & 0 \\ 6\Big(\sin(nt) - nt\Big) & 1 & 0 \\ \hdashline 0 & 0 & \cos(nt) \end{array}\right] \tag{B.107}$$

$$\mathbf{\Phi}_{rv}(t) = \left[\begin{array}{cc:c} \frac{1}{n}\sin(nt) & \frac{2}{n}\Big(1 - \cos(nt)\Big) & 0 \\ \frac{2}{n}\Big(\cos(nt) - 1\Big) & \frac{1}{n}\Big(4\sin(nt) - 3nt\Big) & 0 \\ \hdashline 0 & 0 & \frac{1}{n}\sin(nt) \end{array}\right] \tag{B.108}$$

$$\mathbf{\Phi}_{vr}(t) = \left[\begin{array}{cc:c} 3n\sin(nt) & 0 & 0 \\ 6n\Big(\cos(nt) - 1\Big) & 0 & 0 \\ \hdashline 0 & 0 & -n\sin(nt) \end{array}\right] \tag{B.109}$$

$$\mathbf{\Phi}_{vv}(t) = \left[\begin{array}{cc:c} \cos(nt) & 2\sin(nt) & 0 \\ -2\sin(nt) & 4\cos(nt) - 3 & 0 \\ \hdashline 0 & 0 & \cos(nt) \end{array}\right] \tag{B.110}$$

Here the partition lines indicate the decoupled nature of the in-plane and out-of-plane motion. Lastly, it can be noted that

$$\mathbf{\Phi}_{vr}(t) = \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{\Phi}_{rr}(t) \qquad \mathbf{\Phi}_{vv}(t) = \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{\Phi}_{rv}(t) \tag{B.111}$$

B.7 Physical Constants and Planetary Parameters

Table B.3 contains parameters for celestial objects as taken from the Planetary and Lunar Ephemerides DE430 and DE431, released by NASA in 2014 [71]. Below, Table B.4 provides the characteristic parameters of the Earth ellipsoid, and other general constants as defined within the WGS-84 model [86, 153].

Table B.3: Planetary parameters from DE430-431 Ephemerides

Parameter	Symbol	Value	Units
Earth gravitational parameter	$\mu,\ \mu_\oplus$	398 600.435 436	km^3/s^2
Solar gravitational parameter	$\mu_\odot$	132 712 440 041.939 400	km^3/s^2
Lunar gravitational parameter	$\mu_\leftmoon$	4 902.800 066	km^3/s^2
Earth radius	$R_\oplus$	6 378.136	km
Solar radius	$R_\odot$	696 000.000	km
Lunar radius	$R_\leftmoon$	1 738.000	km

Table B.4: Earth Parameters and Constants from WGS84

Parameter	Symbol	Value	Units
Mass	$M_\oplus$	$5.972\ 186 \times 10^{24}$	kg
Semi-major axis	$a_\oplus$	6 378.173	km
Semi-minor axis	$b_\oplus$	6 356.752	km
Flattening Factor	$1/f_\oplus$	298.257 223 563	[unitless]
Mean Angular Velocity	$\omega_\oplus$	$7.292\ 115\ 147 \times 10^{-5}$	rad/s
Speed of Light	c	299 792 458	m/s
Universal Gravity Constant	G	$6.674\ 280 \times 10^{-11}$	$m^3/kg\ s^2$

Appendix C

Kalman Filter Theory

The following chapter provides a derivation of the original discrete-time Kalman Filter for linear systems, to introduce the basic concepts and operating principles of the technique. The algorithm is then expanded to include nonlinear dynamic and measurement models. Since the dynamics defining spacecraft formation flying can be modelled as continuous-time systems, a combined continuous-discrete formulation of the nonlinear Kalman Filter. These derivations have been adapted from the original works presented by Kalman [19], Bucy [154], while the summaries developed by De Ruiter [66], Alfriend [77] and Aitken [155] are also useful references.

C.1 The Linear Kalman Filter

A general discrete-time system can be represented by

$$\dot{\mathbf{x}}_{k+1} = \mathbf{\Phi}_k \mathbf{x}_k + \mathbf{B}_k \mathbf{u}_k + \mathbf{w}_k \tag{C.1}$$

The continuous-time signal is sampled every T seconds, such that with k as an integer sampling index, the discrete-time is given by $t = kT$. The other components of this model are

$$\begin{aligned}
\mathbf{x}_k \in \mathbb{R}^n &\sim \text{ State Vector} \\
\mathbf{u}_k \in \mathbb{R}^q &\sim \text{ Control Input Vector} \\
\mathbf{w}_k \in \mathbb{R}^p &\sim \text{ Process Noise Vector} \\
\mathbf{\Phi}_k \in \mathbb{R}^{n \times n} &\sim \text{ State Transition Matrix} \\
\mathbf{B}_k \in \mathbb{R}^{n \times q} &\sim \text{ Control Mapping Matrix for } \mathbf{u}_k \mapsto \mathbf{x}_k
\end{aligned}$$

Observations of the system can also be represented by a linear discrete-time model, where the measurement equation is written as:

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k \tag{C.2}$$

using the notations as

$$\begin{aligned} \mathbf{z}_k \in \mathbb{R}^m &\sim \text{ Measurement Vector} \\ \mathbf{v}_k \in \mathbb{R}^m &\sim \text{ Measurement Noise Vector} \\ \mathbf{H}_k \in \mathbb{R}^{m \times n} &\sim \text{ Measurement Mapping Matrix for } \mathbf{x}_k \mapsto \mathbf{z}_k \end{aligned}$$

The process noise and measurement noise are typically assumed to be Gaussian and white, such that they have zero mean, they are uncorrelated, and they are normally distributed. Thus for all discrete points k,

$$\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k) \qquad \mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k) \qquad E[\mathbf{v}_k \mathbf{w}_k^T] = \mathbf{0}$$

The process noise covariance $\mathbf{Q}_k$ and the measurement noise covariance $\mathbf{R}_k$ are symmetric matrices that capture the additive effects of the noise in the system throughout the time step, and are represented by

$$\mathbf{Q}_k = E[\mathbf{w}_k \mathbf{w}_k^T] \qquad \mathbf{R}_k = E[\mathbf{v}_k \mathbf{v}_k^T]$$

and the corresponding covariances are

$$\begin{aligned} \mathbf{Q}_k \in \mathbb{R}^{n \times n} &\sim \text{ Process Noise Covariance} \\ \mathbf{R}_k \in \mathbb{R}^{m \times m} &\sim \text{ Measurement Noise Covariance} \end{aligned}$$

Since the purpose of the Kalman filter is to estimate the state of the system, it is useful to consider the error between the true state $\mathbf{x}_k$ and the predicted state $\hat{\mathbf{x}}_k$. Expressing the state error as $\mathbf{e}_k = \mathbf{x}_k - \hat{\mathbf{x}}_k$, we can then quantify the errors in both the prediction and correction steps of the filtering algorithm. The state estimate available before a measurement has been processed is called the *a priori* estimate, denoted by $\hat{\mathbf{x}}_{k+1}^-$. This corresponds to the best guess possible with the dynamics described by the physical model. The *a posteriori* state estimate $\hat{\mathbf{x}}_{k+1}^+$ is then obtained after a measurement is made at the current time t_k. It is standard to assume that the *a priori* and *a posteriori* state estimates are uncorrelated with the process and measurement

noise, so

$$E[\mathbf{e}_k^- \mathbf{w}_k^T] = \mathbf{0} \qquad E[\mathbf{e}_k^+ \mathbf{w}_k^T] = \mathbf{0} \qquad E[\mathbf{e}_k^- \mathbf{v}_k^T] = \mathbf{0} \qquad E[\mathbf{e}_k^+ \mathbf{v}_k^T] = \mathbf{0}$$

Finally, a covariance matrix $\mathbf{P}_k \in \mathbb{R}^{n \times n}$ is defined for the state errors. The covariance matrix provides an intuitive feel for the level of uncertainty in the current estimate, as it can be seen that each term in the covariance matrix will essentially represent the square of the error between the true and estimated states. Thus, large values within $\mathbf{P}_k^-$ would indicate that the *a priori* estimate may be inaccurate, while small $\mathbf{P}_k^+$ elements suggest the *a posteriori* estimate is close to the true state.

$$\mathbf{P}_k = E[\mathbf{e}_k \mathbf{e}_k^T] = E[(\mathbf{x}_k - \hat{\mathbf{x}}_k)(\mathbf{x}_k - \hat{\mathbf{x}}_k)^T] \tag{C.3}$$

Now that the dynamics and measurement models have been established and the system noise processes have been characterized along with the errors in the state estimates, it is time to explore the prediction and correction phases that calculate the estimated states.

The Dynamics Propagation Phase

Before implementing the Kalman Filter, several terms need to be initialized. An initial state of the system $\hat{\mathbf{x}}_0$ must be defined, along with an estimate of the error covariance $\mathbf{P}_0$. Similar estimates of the process noise covariance $\mathbf{Q}_k$ and the measurement noise covariance $\mathbf{R}_k$ must be made, and these two matrices represent the tunable parameters within the filter.

The dynamics of the system have already been modelled through Equation (C.1), so now both the *a priori* state estimate and the error covariance can be obtained by propagating the previous best estimates through a single time step. This is done using the state transition matrix $\mathbf{\Phi}_{k-1}$, via

$$\hat{\mathbf{x}}_k^- = \mathbf{\Phi}_{k-1} \hat{\mathbf{x}}_{k-1} + \mathbf{B}_{k-1} \mathbf{u}_{k-1} \tag{C.4}$$

$$\mathbf{P}_k^- = \mathbf{\Phi}_{k-1} \mathbf{P}_{k-1} \mathbf{\Phi}_{k-1}^T + \mathbf{Q}_k \tag{C.5}$$

In this phase, $\mathbf{u}$ is a known input assuming the system is controlled. Also note that the process noise $\mathbf{w}$ has been omitted, as it is impossible to know the noise that will affect the system. Therefore, the physical dynamics of the system are propagated in time using the noise-free model. For linear-time invariant systems of the standard form $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$, the discrete state transition matrix is constant, and evaluated

with a time step $\Delta t = t_k - t_{k-1}$ as shown below. Note that a Taylor Series Expansion is usually used for calculating the matrix exponential efficiently, and using the first 4 or 5 terms is usually sufficient for most well-behaved dynamics systems.

$$\mathbf{\Phi}_{k-1} = e^{\mathbf{A}(t_k - t_{k-1})} \triangleq \sum_{n=0}^{\infty} \frac{(\mathbf{A}\Delta t)^n}{n!} \tag{C.6}$$

The Measurement Correction Phase

Now that an *a priori* state has been propagated to the current time step t_k from the dynamics, we want to correct that estimate using the measurements $\mathbf{y}_k$, which are taken at discrete time t_k. From the measurement model in Equation (C.2) we can generate an estimated measurement based on the *a priori* state, with $\hat{\mathbf{z}}_k = \mathbf{H}_k \hat{\mathbf{x}}_k^-$. The corrected *a posteriori* state is then estimated using a simple linear equation based on the error between the true and estimated measurements, and the Kalman Gain $\mathbf{K}_k \in \mathbb{R}^{n \times m}$. The update equation is written as

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{K}_k \left(\mathbf{z}_k - \hat{\mathbf{z}}_k\right) \tag{C.7}$$

After some algebraic manipulation, the *a posteriori* error covariance can be expressed as a function called the Joseph Formula, which is

$$\mathbf{P}_k^+ = \left(\mathbf{I} - \mathbf{K}_k \mathbf{H}_k\right) \mathbf{P}_k^- \left(\mathbf{I} - \mathbf{K}_k \mathbf{H}_k\right)^T + \mathbf{K}_k \mathbf{R}_k \mathbf{K}_k^T \tag{C.8}$$

With this covariance update we have almost everything we need to implement the Kalman Filter - all that remains is to determine the scaling gain $\mathbf{K}_k$. To do this we will establish an optimization problem since the goal is to obtain an optimal estimate for the system states. With the estimate errors characterized by $\mathbf{P}_k^+$, this covariance matrix is a suitable candidate for a cost function $J_k(\mathbf{K}_k)$. The Kalman Gain should provide the best possible state estimates, which means minimizing the error covariance. An appropriate optimization process is then to minimize the sum of the diagonal elements of the covariance matrix, which is equivalent to minimizing the sum of squared-errors:

$$\begin{aligned} \text{Minimize} \quad J_k(\mathbf{K}_k) &= \text{trace}\left[\mathbf{P}_k\right] \\ &= \text{trace}\left[\left(\mathbf{I} - \mathbf{K}_k \mathbf{H}_k\right) \mathbf{P}_k^- \left(\mathbf{I} - \mathbf{K}_k \mathbf{H}_k\right)^T + \mathbf{K}_k \mathbf{R}_k \mathbf{K}_k^T\right] \end{aligned} \tag{C.9}$$

Taking the derivative of $J_k(\mathbf{K}_k)$ with respect to $\mathbf{K}_k$, and setting the result equal to zero minimizes the objective function. Noticing the quadratic matrix form that exists inside the trace function above, the identity $\frac{\partial}{\partial L}\left[\text{trace}(\mathbf{LML}^T)\right] = 2\mathbf{LM}$ is used to simplify the derivative to

$$\frac{\partial J_k(\mathbf{K}_k)}{\partial(\mathbf{K}_k)} = 2\left(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k\right)\mathbf{P}_k^- \frac{\partial}{\partial \mathbf{K}_k}\left(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k\right) + 2\mathbf{K}_k\mathbf{R}_k \tag{C.10}$$

Luckily, $\frac{\partial}{\partial \mathbf{K}_k}\left(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k\right) = -\mathbf{H}_k^T$, so setting (C.10) equal to zero reduces to:

$$-\left(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k\right)\mathbf{P}_k^-\mathbf{H}_k^T + \mathbf{K}_k\mathbf{R}_k = \mathbf{0} \tag{C.11}$$

Shifting several terms around and isolating $\mathbf{K}_k$ leads to the final equation for the optimal Kalman gain, shown in Equation (C.12). The new equation for the Kalman Gain can also be used to simplify the *a posteriori* covariance update, into a form that is slightly more compact than the original Joseph Formula.

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}_k^T\left(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^T + \mathbf{R}_k\right)^{-1} \tag{C.12}$$

$$\mathbf{P}_k^+ = \left(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k\right)\mathbf{P}_k^- \tag{C.13}$$

C.2 The Extended Kalman Filter

While the general Kalman Filter is optimal for linear models, the technique can be applied to nonlinear system by incorporating a linearization of the nonlinear functions within the dynamics and measurement models. This technique is known as the Extended Kalman Filter (EKF), which provides a method of estimating state variables for nonlinear dynamics and measurement models [156]. There are formulations of the EKF specifically for discrete-time systems [155], however since the dynamic models for spacecraft navigation are derived in continuous-time form, a hybrid EKF algorithm is more applicable. Thus, the Continuous-Discrete Extended Kalman Filter is based on the following nonlinear dynamics and measurement equations:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}, \mathbf{u}, t) + \mathbf{w}(t) \tag{C.14}$$

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k) + \mathbf{v}_k \tag{C.15}$$

Here the measurement model is treated as a discrete system due to the inherent sampling of digital sensors, while the dynamics are expressed in continuous time. In the linear case of the Kalman Filter, the dynamics and measurement equations were propagated using $\mathbf{H}$ and $\mathbf{\Phi}$, which were constant-valued matrices; however, this cannot be done in the nonlinear case. To solve this issue, it is assumed that the nonlinear functions $\mathbf{f}$ and $\mathbf{h}$ are sufficiently smooth such that they can be approximated using a Taylor Series. Using only the linear terms of the expansion, we linearize the models about a nominal trajectory, and can then apply the original Kalman Filter techniques.

The Dynamics Propagation Phase - Prediction

For the system dynamics, the state can be propagated discretely by linearizing the dynamics and evaluating a state transition matrix for each time step. The nonlinear model is linearized by taking the first term of the Taylor's Series Expansion evaluated at the current best state estimate, so this becomes the Jacobian of $\mathbf{f}$ evaluated at the *a posteriori* estimate from the previous time step:

$$\mathbf{F}_{k-1} = \left.\frac{\partial \mathbf{f}(\mathbf{x}, t)}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}_{k-1}^{+}} \tag{C.16}$$

From this linearization, we see that $\mathbf{F}$ is purely a function of time, since the partial derivatives are evaluated at particular values of $\hat{\mathbf{x}}(t)$. This means each successive time t_k will have a different state transition matrix - this is different than what the linear Kalman Filter required! From each linearized state transition matrix $\mathbf{F}_{k-1}$, the discrete-time state transition matrix through a time step Δt to the current time is given by:

$$\mathbf{\Phi}_{k-1} = \exp\left[\mathbf{F}_{k-1}\Delta t\right] \tag{C.17}$$

With the state transition matrix defined, the propagation of the state vector and error covariance matrix follow from the Linear Kalman Filter equations shown in the previous section. Note that because the nonlinear model incorporates the control input, there is no explicit $\mathbf{u}$ term in the dynamics propagation and we have

$$\hat{\mathbf{x}}_k^- = \mathbf{\Phi}_{k-1}\hat{\mathbf{x}}_{k-1} \tag{C.18}$$

$$\mathbf{P}_k^- = \mathbf{\Phi}_{k-1}\mathbf{P}_{k-1}\mathbf{\Phi}_{k-1}^T + \mathbf{Q}_k \tag{C.19}$$

Alternatively, the *a priori* state estimate can be computed by numerically integrating the nonlinear equations from discrete time t_{k-1} to t_k. A similar integration

can be used to propagate the covariance matrix as well:

$$\hat{\mathbf{x}}_k^- = \hat{\mathbf{x}}_{k-1}^+ + \int_{t_{k-1}}^{t_k} \mathbf{f}\left[\hat{\mathbf{x}}(t)\right] \mathrm{d}t \Big|_{\hat{\mathbf{x}}_{k-1}} \tag{C.20}$$

$$\mathbf{P}_k^- = \mathbf{P}_{k-1}^+ + \int_{t_{k-1}}^{t_k} \dot{\mathbf{P}}\left[\hat{\mathbf{x}}(t)\right] \mathrm{d}t \Big|_{\hat{\mathbf{x}}_{k-1}} \tag{C.21}$$

where $\dot{\mathbf{P}}$ corresponds to the Riccati Covariance Differential Equation:

$$\dot{\mathbf{P}}_k = \mathbf{F}_{k-1}\mathbf{P}_{k-1} + \mathbf{P}_{k-1}\mathbf{F}_{k-1}^T + \mathbf{Q}_k \tag{C.22}$$

In practice, numerically integrating the noise-free nonlinear dynamics equations in (C.20) is usually the preferred approach for propagating the state estimate; this maintains the full nonlinear behaviour of the physical system. Due to the computational burden of solving the Riccati Covariance differentials, propagation of the covariance typically uses the linearized state transition matrix, as shown in (C.19). Although there are many ways to perform the required integrations numerically, the 5-step, 4th-order Runge-Kutta-Merson technique [103, 105] has been shown to provide a desirable blend of precision and computational efficiency.

The Measurement Correction Phase

The measurement update of the EKF is similar to the LKF, except now the nonlinear measurement model is linearized to obtain $\mathbf{H}_k$, similar to the derivation of $\mathbf{F}_k$ in the dynamics linearization. After taking the Jacobian of the measurement equations $\mathbf{h}(\mathbf{x})$, the terms are evaluated at the new *a priori* estimated state $\hat{\mathbf{x}}_k^-$:

$$\mathbf{H}_k = \frac{\partial \mathbf{h}(\mathbf{x})}{\partial \mathbf{x}}\Big|_{\hat{\mathbf{x}}_k^-} \tag{C.23}$$

This step differs from the linearization of the dynamics equations, which were evaluated along a trajectory based on the previous *a posteriori* states $\hat{\mathbf{x}}_{k-1}^+$. Here, we are linearizing about the new state trajectory that was obtained by propagating the dynamics forward one time step - since the prediction phase provided new state estimates, it makes sense to use these estimates in the next correction phase. After linearizing the measurement model, the original Kalman Filter equations can be used to calculate the Kalman Gain, the corrected state estimates, and the corrected error covariances. The following page provides an flowchart of the calculation in a typical EKF algorithm.

C.3 Summary of the EKF Implementation

1. Initialize the filter parameters $(\mathbf{x}_0, \mathbf{P}_0, \mathbf{Q}_0, \mathbf{z}_0, \mathbf{R}_0)$

2. Propagate the state estimate and error covariance

$$\hat{\mathbf{x}}_k^- = \hat{\mathbf{x}}_{k-1} + \int_{t_{k-1}}^{t_k} \mathbf{f}\left[\hat{\mathbf{x}}(t)\right] \, \mathrm{d}t \Big|_{\hat{\mathbf{x}}_{k-1}}$$

$$\mathbf{F}_{k-1} = \frac{\partial \mathbf{f}(\mathbf{x}, t)}{\partial \mathbf{x}}\Big|_{\mathbf{x}=\hat{\mathbf{x}}_{k-1}}$$

$$\mathbf{\Phi}_{k-1} = e^{\mathbf{F}_{k-1}\Delta t}$$

$$\mathbf{P}_k^- = \mathbf{\Phi}_{k-1}\mathbf{P}_{k-1}\mathbf{\Phi}_{k-1}^T + \mathbf{Q}_k$$

3. Calculate the Kalman Gain

$$\mathbf{H}_k = \frac{\partial \mathbf{h}(\mathbf{x})}{\partial \mathbf{x}}\Big|_{\hat{\mathbf{x}}_k^-}$$

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}_k^T\left(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^T + \mathbf{R}_k\right)^{-1}$$

4. Correct the state estimate and error covariance

$$\hat{\mathbf{z}}_k = \mathbf{H}_k\hat{\mathbf{x}}_k^-$$

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{K}_k\left(\mathbf{z}_k - \hat{\mathbf{z}}_k\right)$$

$$\mathbf{P}_k^+ = \left(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k\right)\mathbf{P}_k^-\left(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k\right)^T + \mathbf{K}_k\mathbf{R}_k\mathbf{K}_k^T$$

5. Repeat steps 2 through 5 for every time step of the data being processed.

Appendix D

Observability Analysis

This appendix is designed to provide a brief standalone overview of nonlinear observability, and its applicability to the spacecraft formation flying system considered in this thesis. The concept of observability is introduced along with several useful references on the subject, after which the dynamics and measurement models used within the Extended Kalman Filter (EKF) are presented for convenience. A derivation of the observability matrix is given, and used to confirm the local observability of all states within the EKF algorithms proposed in the thesis.

D.1 Introduction to Nonlinear Observability

The concept of observability is well defined and relatively straightforward for linear systems, and is therefore a standard topic in most introductory courses on control theory. Dating back to the 1960's, observability describes how well the states of a system can be reconstructed from outputs of that system. However, the extension of observability criteria to nonlinear systems is a more recent development, proposed in 1977 by Hermann & Krener [157]. Given the nonlinear framework of the spacecraft formation flying dynamics used in the research proposed here, this appendix will accordingly focus on the observability criteria for nonlinear systems.

For the time-varying nonlinear system described by a state vector $\mathbf{x} \in \mathbb{R}^n$ and an output vector $\mathbf{y}(t) \in \mathbb{R}^p$, the system is said to be locally observable over the interval $t \in [0, T]$ if the mapping from the intial state $\mathbf{x}_0$ to the output $\mathbf{y}(t)$ is one-to-one [101]. Stated differently, local observability implies that the initial state x_0 can be reconstructed from the knowledge contained in the outputs over the given time interval. This weaker notion of local observability was established by Bartosiewicz in 1995 [102]. Similar to the observability conditions for linear systems, a check for local observability involves determining if an observability matrix $\mathcal{O}(\mathbf{x})$ is full rank, such

that

$$\text{rank } \mathcal{O}(\mathbf{x}) = n \tag{D.1}$$

While the linear observability test matrix is simple to construct from the time-invariant state transition and measurement matrices, the nonlinear equivalent relies on manifold calculus and the use of Lie derivatives. The intricacies of differential geometry are beyond the scope of this work, but taking the key results needed for observability analyses shows that the first-order Lie derivative of the output $\mathbf{h}(\mathbf{x})$ along a vector field $\mathbf{f}(\mathbf{x})$ is denoted by $\mathcal{L}_f\mathbf{h}(\mathbf{x}) \in \mathbb{R}^p$, and can be expressed by

$$\mathcal{L}_f\mathbf{h}(\mathbf{x}) \triangleq \frac{\partial \mathbf{h}(\mathbf{x})}{\partial \mathbf{x}}\mathbf{f}(\mathbf{x}) = \nabla\mathbf{h}(\mathbf{x})\mathbf{f}(\mathbf{x}) \tag{D.2}$$

where the $\nabla(\cdot)$ operator represents the vector gradient with respect to the state. The next i^{th}-order Lie derivatives can then be formulated recursively as

$$\mathcal{L}_f^i\mathbf{h}(\mathbf{x}) \triangleq \frac{\partial \mathcal{L}_f^{i-1}\mathbf{h}(\mathbf{x})}{\partial \mathbf{x}}\mathbf{f}(\mathbf{x}) = \nabla\mathcal{L}_f^{i-1}\mathbf{h}(\mathbf{x})\mathbf{f}(\mathbf{x}); \quad \forall i > 0 \tag{D.3}$$

The zeroth-order Lie derivative is simply the measurement matrix $\mathbf{h}(\mathbf{x})$, so it can be stated that

$$\mathcal{L}_f^0\mathbf{h}(\mathbf{x}) = \mathbf{h}(\mathbf{x}) \tag{D.4}$$

Using this Lie derivative notation, the observability matrix $\mathcal{O}(\mathbf{x}) \in \mathbb{R}^{pn\times n}$ for the nonlinear system is defined as

$$\mathcal{O}(\mathbf{x}) \triangleq \frac{\partial}{\partial \mathbf{x}}\begin{bmatrix} \mathcal{L}_f^0\mathbf{h}(\mathbf{x}) \\ \mathcal{L}_f^1\mathbf{h}(\mathbf{x}) \\ \vdots \\ \mathcal{L}_f^{n-1}\mathbf{h}(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} \nabla\mathcal{L}_f^0\mathbf{h}(\mathbf{x}) \\ \nabla\mathcal{L}_f^1\mathbf{h}(\mathbf{x}) \\ \vdots \\ \nabla\mathcal{L}_f^{n-1}\mathbf{h}(\mathbf{x}) \end{bmatrix} \tag{D.5}$$

The size of this nonlinear observability matrix clearly depends on the number of states and measurements in the system, so the computation of the higher-order derivatives can become quite intensive. As such, Butcher & Wang [101] highlight that a sufficient (but not necessary) condition for observability can be obtained by only constructing the first n rows of $\mathcal{O}(\mathbf{x})$, or by constructing only a subsequent number of rows necessary to show that the truncated version of $\mathcal{O}(\mathbf{x})$ is full rank.

D.2 Summary of the System Dynamics

The dynamics of relative motion between the target and chaser spacecraft are developed in Chapter 2 of this thesis, and are summarized as

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) = \begin{bmatrix} \dot{x} & \dot{y} & \dot{z} & \dot{\theta} & \dot{r}_t & \ddot{x} & \ddot{y} & \ddot{z} & \ddot{\theta} & \ddot{r}_t \end{bmatrix}^T \tag{D.6}$$

where the governing differential equations are given in Eqs (2.36)-(2.40). The linearized dynamics matrix is obtained by taking the derivative of the dynamics equations with respect to the state vector, which for the case presented here gives

$$\frac{\partial \dot{\mathbf{x}}}{\partial \mathbf{x}} = \frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ \ddot{x}_x & \ddot{x}_y & \ddot{x}_z & 0 & \ddot{x}_{r_t} & 0 & \ddot{x}_{\dot{y}} & 0 & \ddot{x}_{\dot{\theta}} & \ddot{x}_{\dot{r}_t} \\ \ddot{y}_x & \ddot{y}_y & \ddot{y}_z & 0 & \ddot{y}_{r_t} & \ddot{y}_{\dot{x}} & 0 & 0 & \ddot{y}_{\dot{\theta}} & \ddot{y}_{\dot{r}_t} \\ \ddot{z}_x & \ddot{z}_y & \ddot{z}_z & 0 & \ddot{z}_{r_t} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \ddot{\theta}_{r_t} & 0 & 0 & 0 & \ddot{\theta}_{\dot{\theta}} & \ddot{\theta}_{\dot{r}_t} \\ 0 & 0 & 0 & 0 & \ddot{r}_{t_{r_t}} & 0 & 0 & 0 & \ddot{r}_{t_{\dot{\theta}}} & 0 \end{bmatrix} \tag{D.7}$$

The components within the Jacobian matrix above are the partial derivatives of the dynamics equations, as presented in Section 3.3. Measurements of the relative x, y and z positions and velocities in the Local-Vertical, Local-Horizontal Frame are provided to the EKF, along with virtual measurements of the target true anomaly θ, calculated from the target's ECI position and velocity states. The corresponding measurement dynamics and partial derivatives are therefore written with

$$\mathbf{h}(\mathbf{x}) = \begin{bmatrix} x_m & y_m & z_m & \theta_m & \dot{x}_m & \dot{y}_m & \dot{z}_m \end{bmatrix}^T \tag{D.8}$$

$$\frac{\partial \mathbf{h}}{\partial \mathbf{x}} = \mathbf{H} = \begin{bmatrix} \mathbf{I}_{4\times4} & \mathbf{0}_{4\times1} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times2} \\ \mathbf{0}_{3\times4} & \mathbf{0}_{3\times1} & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times2} \end{bmatrix} \tag{D.9}$$

D.3 Observability for Relative Spacecraft Motion

Recalling that the zeroeth-order Lie derivative is simply the measurement model of the system, it is known that

$$\mathcal{L}_f^0 \mathbf{h}(\mathbf{x}) = \mathbf{h}(\mathbf{x}) \tag{D.10}$$

where $\mathbf{h}(\mathbf{x})$ is given by (D.8). The gradient of the zeroeth-order Lie Derivative of $\mathbf{f}(\mathbf{x})$ along $\mathbf{h}(\mathbf{x})$ can then be established, and corresponds to the linearized measurement model used within the EKF. As such, this yields

$$\nabla \mathcal{L}_f^0 \mathbf{h}(\mathbf{x}) = \frac{\partial \mathbf{h}}{\partial \mathbf{x}} = \mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{D.11}$$

The first-order Lie Derivative requires more work to derive, and is seen to be

$$\mathcal{L}_f^1 \mathbf{h}(\mathbf{x}) = \mathbf{H}\mathbf{f}(\mathbf{x}) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\theta} \\ \dot{r}_t \\ \ddot{x} \\ \ddot{y} \\ \ddot{z} \\ \ddot{\theta} \\ \ddot{r}_t \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\theta} \\ \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} \tag{D.12}$$

Taking the gradient of the first-order Lie Derivative leads to the following result:

$$\nabla \mathcal{L}_f^1 \mathbf{h}(\mathbf{x}) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ \ddot{x}_x & \ddot{x}_y & \ddot{x}_z & 0 & \ddot{x}_{r_t} & 0 & \ddot{x}_{\dot{y}} & 0 & \ddot{x}_{\dot{\theta}} & \ddot{x}_{\dot{r}_t} \\ \ddot{y}_x & \ddot{y}_y & \ddot{y}_z & 0 & \ddot{y}_{r_t} & \ddot{y}_{\dot{x}} & 0 & 0 & \ddot{y}_{\dot{\theta}} & \ddot{y}_{\dot{r}_t} \\ \ddot{z}_x & \ddot{z}_y & \ddot{z}_z & 0 & \ddot{z}_{r_t} & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{D.13}$$

As a reminder, the symbol $\ddot{x}_x$ is shorthand notation used to indicate the partial derivative of $\ddot{x}$ with respect to x, and so on for the other dynamical states. The second-order Lie Derivative in the spacecraft formation scenario is

$$\mathcal{L}_f^2\mathbf{h}(\mathbf{x}) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ \ddot{x}_x & \ddot{x}_y & \ddot{x}_z & 0 & \ddot{x}_{r_t} & 0 & \ddot{x}_{\dot{y}} & 0 & \ddot{x}_{\dot{\theta}} & \ddot{x}_{\dot{r}_t} \\ \ddot{y}_x & \ddot{y}_y & \ddot{y}_z & 0 & \ddot{y}_{r_t} & \ddot{y}_{\dot{x}} & 0 & 0 & \ddot{y}_{\dot{\theta}} & \ddot{y}_{\dot{r}_t} \\ \ddot{z}_x & \ddot{z}_y & \ddot{z}_z & 0 & \ddot{z}_{r_t} & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\theta} \\ \dot{r}_t \\ \ddot{x} \\ \ddot{y} \\ \ddot{z} \\ \ddot{\theta} \\ \ddot{r}_t \end{bmatrix} = \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \\ \ddot{\theta} \\ L_1 \\ L_2 \\ L_3 \end{bmatrix} \tag{D.14}$$

where L_1, L_2 and L_3 represent the following functions

$$L_1 = \dot{x}\ddot{x}_x + \dot{y}\ddot{x}_y + \dot{z}\ddot{x}_z + \dot{r}_t\ddot{x}_{r_t} + \ddot{y}\ddot{x}_{\dot{y}} + \ddot{\theta}\ddot{x}_{\dot{\theta}} + \ddot{r}_t\ddot{x}_{\dot{r}_t} \tag{D.15}$$

$$L_2 = \dot{x}\ddot{y}_x + \dot{y}\ddot{y}_y + \dot{z}\ddot{y}_z + \dot{r}_t\ddot{y}_{r_t} + \ddot{x}\ddot{y}_{\dot{x}} + \ddot{\theta}\ddot{y}_{\dot{\theta}} + \ddot{r}_t\ddot{y}_{\dot{r}_t} \tag{D.16}$$

$$L_3 = \dot{x}\ddot{z}_x + \dot{y}\ddot{z}_y + \dot{z}\ddot{z}_z + \dot{r}_t\ddot{z}_{r_t} \tag{D.17}$$

The gradient of the second-order Lie Derivative can be written as

$$\nabla\mathcal{L}_f^2\mathbf{h}(\mathbf{x}) = \begin{bmatrix} \ddot{x}_x & \ddot{x}_y & \ddot{x}_z & 0 & \ddot{x}_{r_t} & 0 & \ddot{x}_{\dot{y}} & 0 & \ddot{x}_{\dot{\theta}} & \ddot{x}_{\dot{r}_t} \\ \ddot{y}_x & \ddot{y}_y & \ddot{y}_z & 0 & \ddot{y}_{r_t} & \ddot{y}_{\dot{x}} & 0 & 0 & \ddot{y}_{\dot{\theta}} & \ddot{y}_{\dot{r}_t} \\ \ddot{z}_x & \ddot{z}_y & \ddot{z}_z & 0 & \ddot{z}_{r_t} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \ddot{\theta}_{r_t} & 0 & 0 & 0 & \ddot{\theta}_{\dot{\theta}} & \ddot{\theta}_{\dot{r}_t} \\ \alpha_{51} & \alpha_{52} & \alpha_{53} & 0 & \alpha_{55} & \alpha_{56} & \alpha_{57} & \ddot{x}_z & \alpha_{59} & \alpha_{5,10} \\ \alpha_{61} & \alpha_{62} & \alpha_{63} & 0 & \alpha_{65} & \alpha_{66} & \alpha_{67} & \ddot{y}_z & \alpha_{69} & \alpha_{6,10} \\ \alpha_{71} & \alpha_{72} & \alpha_{73} & 0 & \alpha_{75} & \ddot{z}_x & \ddot{z}_y & \ddot{z}_z & 0 & \ddot{z}_{r_t} \end{bmatrix} \tag{D.18}$$

where the various $\alpha_{k,j}$ matrix components are shown on the following page, for $k \in 1, \ldots, 7$ and $j \in 1, \ldots, 10$. The second-order form of the partial derivative shorthand notation is needed within the $\alpha_{k,j}$ terms, where for example $\ddot{x}_{zx}$ represents

$$\ddot{x}_{zx} = \frac{\partial \ddot{x}_z}{\partial x} = \frac{\partial}{\partial x}\frac{\partial \ddot{x}}{\partial z} = \frac{\partial^2 \ddot{x}}{\partial x \partial z} \tag{D.19}$$

$$\alpha_{51} = \frac{\partial L_1}{\partial x} = \dot{x}\ddot{x}_{xx} + \dot{y}\ddot{x}_{yx} + \dot{z}\ddot{x}_{zx} + \dot{r}\ddot{x}_{rx} + \ddot{y}_x\ddot{x}_{\dot{y}} + \ddot{\theta}\ddot{x}_{\dot{\theta}x} \tag{D.20}$$

$$\alpha_{52} = \frac{\partial L_1}{\partial y} = \dot{x}\ddot{x}_{xy} + \dot{y}\ddot{x}_{yy} + \dot{z}\ddot{x}_{zy} + \dot{r}\ddot{x}_{ry} + \ddot{y}_y\ddot{x}_{\dot{y}} + \ddot{\theta}\ddot{x}_{\dot{\theta}y} + \dot{r}\ddot{x}_{\dot{r}y} \tag{D.21}$$

$$\alpha_{53} = \frac{\partial L_1}{\partial z} = \dot{x}\ddot{x}_{xz} + \dot{y}\ddot{x}_{yz} + \dot{z}\ddot{x}_{zz} + \dot{r}\ddot{x}_{rz} + \ddot{y}_z\ddot{x}_{\dot{y}} \tag{D.22}$$

$$\alpha_{55} = \frac{\partial L_1}{\partial r_t} = \dot{x}\ddot{x}_{xr} + \dot{y}\ddot{x}_{yr} + \dot{z}\ddot{x}_{zr} + \dot{r}\ddot{x}_{rr} + \ddot{y}_r\ddot{x}_{\dot{y}} + \ddot{\theta}_r\ddot{x}_{\dot{\theta}r} + \ddot{r}_r\ddot{x}_{\dot{r}} + \ddot{r}\ddot{x}_{\dot{r}r} \tag{D.23}$$

$$\alpha_{56} = \frac{\partial L_1}{\partial \dot{x}} = \ddot{x}_x + \ddot{y}_{\dot{x}}\ddot{x}_{\dot{y}} \tag{D.24}$$

$$\alpha_{57} = \frac{\partial L_1}{\partial \dot{y}} = \ddot{x}_y + \ddot{\theta}\ddot{x}_{\dot{\theta}\dot{y}} \tag{D.25}$$

$$\alpha_{58} = \frac{\partial L_1}{\partial \dot{z}} = \ddot{x}_z \tag{D.26}$$

$$\alpha_{59} = \frac{\partial L_1}{\partial \dot{\theta}} = \dot{x}\ddot{x}_{x\dot{\theta}} + \dot{y}\ddot{x}_{y\dot{\theta}} + \dot{r}\ddot{x}_{r\dot{\theta}} + \ddot{y}_{\dot{\theta}}\ddot{x}_{\dot{y}} + \ddot{y}\ddot{x}_{\dot{y}\dot{\theta}} + \ddot{\theta}_{\dot{\theta}}\ddot{x}_{\dot{\theta}} + \ddot{\theta}\ddot{x}_{\dot{\theta}\dot{\theta}} + \ddot{r}_{\dot{\theta}}\ddot{x}_{\dot{r}} + \ddot{r}\ddot{x}_{\dot{r}\dot{\theta}} \tag{D.27}$$

$$\alpha_{5,10} = \frac{\partial L_1}{\partial \dot{r}_t} = \dot{y}\ddot{x}_{y\dot{r}} + \ddot{x}_r + \ddot{y}_{\dot{r}}\ddot{x}_{\dot{y}} + \ddot{\theta}_{\dot{r}}\ddot{x}_{\dot{\theta}} + \ddot{\theta}\ddot{x}_{\dot{\theta}\dot{r}} \tag{D.28}$$

The second-order partial derivative terms in these matrices are not presented here, as they are quite lengthy and cumbersome. They are easily derivable using software that features an analytic solver, such as Maple or the MATLAB Symbolic Toolbox. Thus, for the purposes of confirming observability for the proposed extended Kalman filter algorithms, it is sufficient to demonstrate that these terms are non-zero.

$$\alpha_{61} = \frac{\partial L_2}{\partial x} = \dot{x}\ddot{y}_{xx} + \dot{y}\ddot{y}_{yx} + \dot{z}\ddot{y}_{zx} + \dot{r}\ddot{y}_{rx} + \ddot{x}_x\ddot{y}_{\dot{x}} + \ddot{\theta}\ddot{y}_{\dot{\theta}x} + \ddot{r}\ddot{y}_{\dot{r}x} \tag{D.29}$$

$$\alpha_{62} = \frac{\partial L_2}{\partial y} = \dot{x}\ddot{y}_{xy} + \dot{y}\ddot{y}_{yy} + \dot{z}\ddot{y}_{zy} + \dot{r}\ddot{y}_{ry} + \ddot{x}_y\ddot{y}_{\dot{x}} + \ddot{\theta}\ddot{y}_{\dot{\theta}y} \tag{D.30}$$

$$\alpha_{63} = \frac{\partial L_2}{\partial z} = \dot{x}\ddot{y}_{xz} + \dot{y}\ddot{y}_{yz} + \dot{z}\ddot{y}_{zz} + \dot{r}\ddot{y}_{rz} + \ddot{x}_z\ddot{y}_{\dot{x}} \tag{D.31}$$

$$\alpha_{65} = \frac{\partial L_2}{\partial r_t} = \dot{x}\ddot{y}_{xr} + \dot{y}\ddot{y}_{yr} + \dot{z}\ddot{y}_{zr} + \dot{r}\ddot{y}_{rr} + \ddot{x}_r\ddot{y}_{\dot{x}} + \ddot{\theta}_r\ddot{y}_{\dot{\theta}} + \ddot{\theta}\ddot{y}_{\dot{\theta}r} + \ddot{r}_r\ddot{y}_{\dot{r}} + \ddot{r}\ddot{y}_{\dot{r}r} \tag{D.32}$$

$$\alpha_{66} = \frac{\partial L_2}{\partial \dot{x}} = \ddot{y}_x + \ddot{\theta}\ddot{y}_{\dot{\theta}\dot{x}} \tag{D.33}$$

$$\alpha_{67} = \frac{\partial L_2}{\partial \dot{y}} = \ddot{y}_y + \ddot{x}_{\dot{y}}\ddot{y}_{\dot{x}} \tag{D.34}$$

$$\alpha_{68} = \frac{\partial L_2}{\partial \dot{z}} = \ddot{y}_z \tag{D.35}$$

$$\alpha_{69} = \frac{\partial L_2}{\partial \dot{\theta}} = \dot{x}\ddot{y}_{x\dot{\theta}} + \dot{y}\ddot{y}_{y\dot{\theta}} + \dot{r}\ddot{y}_{r\dot{\theta}} + \ddot{x}_{\dot{\theta}}\ddot{y}_{\dot{x}} + \ddot{x}\ddot{y}_{\dot{x}\dot{\theta}} + \ddot{\theta}_{\dot{\theta}}\ddot{y}_{\dot{\theta}} + \ddot{\theta}\ddot{y}_{\dot{\theta}\dot{\theta}} + \ddot{r}_{\dot{\theta}}\ddot{y}_{\dot{r}} + \ddot{r}\ddot{y}_{\dot{r}\dot{\theta}} \tag{D.36}$$

$$\alpha_{6,10} = \frac{\partial L_2}{\partial \dot{r}_t} = \dot{x}\ddot{y}_{x\dot{r}} + \ddot{y}_r + \dot{r}\ddot{y}_{r\dot{r}} + \ddot{x}_{\dot{r}}\ddot{y}_{\dot{x}} + \ddot{\theta}_{\dot{r}}\ddot{y}_{\dot{\theta}} + \ddot{\theta}\ddot{y}_{\dot{\theta}\dot{r}} \tag{D.37}$$

$$\alpha_{71} = \frac{\partial L_3}{\partial x} = \dot{x}\ddot{z}_{xx} + \dot{y}\ddot{z}_{yx} + \dot{z}\ddot{z}_{zx} + \dot{r}\ddot{z}_{rx} \tag{D.38}$$

$$\alpha_{72} = \frac{\partial L_3}{\partial y} = \dot{x}\ddot{z}_{xy} + \dot{y}\ddot{z}_{yy} + \dot{z}\ddot{z}_{zy} + \dot{r}\ddot{z}_{ry} \tag{D.39}$$

$$\alpha_{73} = \frac{\partial L_3}{\partial z} = \dot{x}\ddot{z}_{xz} + \dot{y}\ddot{z}_{yz} + \dot{z}\ddot{z}_{zz} + \dot{r}\ddot{z}_{rz} \tag{D.40}$$

$$\alpha_{75} = \frac{\partial L_3}{\partial r_t} = \dot{x}\ddot{z}_{xr} + \dot{y}\ddot{z}_{yr} + \dot{z}\ddot{z}_{zr} + \dot{r}\ddot{z}_{rr} \tag{D.41}$$

For the spacecraft formation dynamics and measurement models used in this research, the first three Lie derivatives are sufficient to show that observability is ensured when the relative position, relative velocity, and target true anomaly are available as measurements. The first $N = 3$ Lie derivatives are presented in the observability matrix below, demonstrating the full rank condition, and the partition lines indicate the zeroeth, first, and second order Lie derivative portions of the matrix. The resulting truncated observability matrix $\mathcal{O}_N(\mathbf{x})$ can be summarized as:

$$\mathcal{O}_N(\mathbf{x}) = \left[\begin{array}{cccccccccc}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
\hdashline
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
\ddot{x}_x & \ddot{x}_y & \ddot{x}_z & 0 & \ddot{x}_{r_t} & 0 & \ddot{x}_{\dot{y}} & 0 & \ddot{x}_{\dot{\theta}} & \ddot{x}_{\dot{r}_t} \\
\ddot{y}_x & \ddot{y}_y & \ddot{y}_z & 0 & \ddot{y}_{r_t} & \ddot{y}_{\dot{x}} & 0 & 0 & \ddot{y}_{\dot{\theta}} & \ddot{y}_{\dot{r}_t} \\
\ddot{z}_x & \ddot{z}_y & \ddot{z}_z & 0 & \ddot{z}_{r_t} & 0 & 0 & 0 & 0 & 0 \\
\hdashline
\ddot{x}_x & \ddot{x}_y & \ddot{x}_z & 0 & \ddot{x}_{r_t} & 0 & \ddot{x}_{\dot{y}} & 0 & \ddot{x}_{\dot{\theta}} & \ddot{x}_{\dot{r}_t} \\
\ddot{y}_x & \ddot{y}_y & \ddot{y}_z & 0 & \ddot{y}_{r_t} & \ddot{y}_{\dot{x}} & 0 & 0 & \ddot{y}_{\dot{\theta}} & \ddot{y}_{\dot{r}_t} \\
\ddot{z}_x & \ddot{z}_y & \ddot{z}_z & 0 & \ddot{z}_{r_t} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \ddot{\theta}_{r_t} & 0 & 0 & 0 & \ddot{\theta}_{\dot{\theta}} & \ddot{\theta}_{\dot{r}_t} \\
\alpha_{51} & \alpha_{52} & \alpha_{53} & 0 & \alpha_{55} & \alpha_{56} & \alpha_{57} & \ddot{x}_z & \alpha_{59} & \alpha_{5,10} \\
\alpha_{61} & \alpha_{62} & \alpha_{63} & 0 & \alpha_{65} & \alpha_{66} & \alpha_{67} & \ddot{y}_z & \alpha_{69} & \alpha_{6,10} \\
\alpha_{71} & \alpha_{72} & \alpha_{73} & 0 & \alpha_{75} & \ddot{z}_x & \ddot{z}_y & \ddot{z}_z & 0 & \ddot{z}_{r_t} \\
\alpha_{51} & \alpha_{52} & \alpha_{53} & 0 & \alpha_{55} & \alpha_{56} & \alpha_{57} & \ddot{x}_z & \alpha_{59} & \alpha_{5,10}
\end{array}\right] \tag{D.42}$$

Appendix E

Orbit Propagator MATLAB Code

This appendix contains samples of the MATLAB code used to simulate the orbital motion of a spacecraft, according to the theory presented in Chapter 2. These MATLAB functions are called within a user-defined function block within the formation flying navigation Simulink model. Orbital perturbations are activated based on the simulator settings defined for the test scenario, and the corresponding perturbing accelerations are only calculated when necessary. The net acceleration acting on the spacecraft is passed back to the Simulink model for integration into the required velocity and position components of the spacecraft. The following functions are included:

`Accelerations.m`: Calculates total acceleration acting on a spacecraft.

`AccelDrag.m`: Calculates acceleration due to atmospheric drag.

`DensityHarrisPriester.m`: Calculates atmospheric density.

`AccelSolRad.m`: Calculates acceleration due to solar radiation pressure.

`AccelSolRadCyl.m`: Calculates the illumination factor for eclipse conditions.

`AccelThirdBody.m`: Calculates the acceleration due to third-body effects.

E.1 Spacecraft Accelerations

```
function [r_ddot] = Accelerations(R_Sat,V_Sat,Constants,PropOptions)
%% Orbit Propagation Force Model ==========================================
% ========================================================================
% Description: This function calculates the accelerations acting on a
% spacecraft due to various perturbing forces, including the effects of
% two-body gravity, J2, atmospheric drag, solar radiation pressure, and
% third body gravity from the Sun and Moon.
%
% Inputs:
%   R_Sat        - Position vector of the spacecraft in ECI frame [m]
%   V_Sat        - Velocity vector of the spacecraft in ECI frame [m/s]
%   Constants.   - Structure containing astrodynamic constants
%   PropOptions. - Structure containing orbit propagation options
%
% Outputs:
%   r_ddot - Acceleration of the spacecraft [m/s^2]
%
% Other Function Calls:
%   AccelThirdBody        - Calculates acceleration due to point mass
%   AccelSolRad           - Calculates acceleration due to radiation
%   AccelDrag             - Calculates acceleration due to drag
%   DensityHarrisPriester - Calculates atmospheric density with HP-model
%
% Reference: Montenbruck & Gill, Chapter 3, 2011
%
% Created by: Cory Fraser - JUN 20, 2018
% Last Edit : Cory Fraser - JUL 27, 2018
% Copyright(c) 2018 by Cory Fraser
% ========================================================================
%% Astrodynamic Constants

mu_Earth = Constants.mu_Earth;
R_Earth  = Constants.R_Earth;

% ========================================================================
%% Two-Body Acceleration w/ J2 Option

r_ddot = -(mu_Earth/(norm(R_Sat))^3)*R_Sat;

if (PropOptions.J2)
    J2 = Constants.J2;
    delta_AccelJ2 = (mu_Earth/(norm(R_Sat))^3) * ...
    [ R_Sat(1)*(3/2)*J2*(R_Earth/norm(R_Sat))^2*(5*R_Sat(3)^2/norm(R_Sat)^2 - 1)
      R_Sat(2)*(3/2)*J2*(R_Earth/norm(R_Sat))^2*(5*R_Sat(3)^2/norm(R_Sat)^2 - 1)
      R_Sat(3)*(3/2)*J2*(R_Earth/norm(R_Sat))^2*(5*R_Sat(3)^2/norm(R_Sat)^2 - 3)];

  r_ddot = r_ddot + delta_AccelJ2;
end

% ========================================================================
%% Luni-Solar Third-Body Perturbations

%Assuming constant position of Sun/Moon over simulation duration
r_Sun = Constants.Rvec_Sun;
r_Moon = Constants.Rvec_Moon;

```

```
57 if (PropOptions.Sun)
58     delta_ThirdBodySun = AccelThirdBody(R_Sat,r_Sun,Constants.mu_Sun);
59     r_ddot = r_ddot + delta_ThirdBodySun;
60 end
61
62 if (PropOptions.Moon)
63     delta_ThirdBodyMoon = AccelThirdBody(R_Sat,r_Moon,Constants.mu_Moon);
64     r_ddot = r_ddot + delta_ThirdBodyMoon;
65 end
66
67 % ================================================================================
68 %% Atmospheric drag
69
70 if (PropOptions.Drag)
71
72     % Atmospheric density [kg/m^3]
73     rho_atm = DensityHarrisPriester(R_Sat, r_Sun, Constants);
74
75     %Angular Velocity of Earth [rad/sec]
76     Omega = Constants.w_Earth;
77
78     delta_Drag = AccelDrag(rho_atm,R_Sat,V_Sat, ...
79         PropOptions.A_Drag,PropOptions.M_Sat,PropOptions.Cd,Omega);
80     r_ddot = r_ddot + delta_Drag;
81 end
82
83 % ================================================================================
84 %% Solar Radiation Pressure
85
86 if (PropOptions.SolarRad)
87     delta_SolarRad = AccelSolRad(R_Sat,r_Sun,...
88         PropOptions.A_Solar,PropOptions.M_Sat,PropOptions.Cr,...
89         Constants.P_Solar, Constants.AU, Constants.R_Earth);
90
91     r_ddot = r_ddot + delta_SolarRad;
92 end
93
94 % ================================================================================
95 end
```

E.2 Acceleration Due to Drag

```
function [r_ddot] = AccelDrag(rho,R_Sat,V_Sat,Ad,m,Cd,w_Earth)
%% Atmospheric Drag Model ==================================================
% =========================================================================
% Description: This function calculates the accelerations acting on a
% spacecraft due to atmospheric drag.
%
% Inputs:
%   rho     - Density of the atmoshpere [kg/m^3]
%   R_Sat   - Position vector of the spacecraft in ECI frame [m]
%   V_Sat   - Velocity vector of the spacecraft in ECI frame [m/s]
%   Ad      - Area exposed to drag forces [m^2]
%   m       - Spacecraft mass [kg]
%   Cd      - Coefficient of Drag
%   w_Earth - Earth's angular velocity [rad/s]
%
% Outputs:
%   r_ddot - Acceleration of the spacecraft [m/s^2]
%
% Other Function Calls:
%   None
%
% Reference: Montenbruck & Gill, Chapter 3, 2011, pages 83-104
%
% Created by: Cory Fraser - JUN 25, 2018
% Last Edit : Cory Fraser - JUN 25, 2018
% Copyright(c) 2018 by Cory Fraser
% =========================================================================

% Angular velocity vector of Earth
omega_Earth = [0; 0; w_Earth];

% Relative velocity vector
v_rel = V_Sat - cross(omega_Earth, R_Sat);
e_v = v_rel/norm(v_rel); %Unit Vector

r_ddot = -(1/2)*rho*norm(v_rel)^2*(Cd*Ad/m)*e_v;

end
```

E.3 Harris-Priester Atmospheric Density Model

```
function [rho_atm] = Density_HarrisPriester(R_Sat, R_Sun, Constants)
% Atmospheric Density Model ===============================================
% =========================================================================
% Description: This function calculates the density of the atmosphere using
% the Harris-Priester model.
%
% Inputs:
%   R_Sat - Position vector of the spacecraft in ECI frame [m]
%   r_Sun - Position vector of the Sun in the ECI frame [m]
%
% Outputs:
%   rho_atm - Density of the atmosphere [kg/m^3]
%
% Other Function Calls:
%   ECEF_to_gLLA - Determines satellite geodetic lat, long, and height
%   CalcPolarAngles - Determines right ascension and declination angles
%
% Reference: Montenbruck & Gill, Chapter 3, 2011, pages 83-104
%
% Created by: Cory Fraser - JUN 30, 2018
% Last Edit : Cory Fraser - JUN 30, 2018
% Copyright(c) 2018 by Cory Fraser
% =========================================================================

% Convert Satellite Position from ECI Frame to ECEF Frame
t = 0; %Neglecting Earth's rotation during simulation
R_Sat_ECEF = ECI_to_ECEF(R_Sat, t);
e_Sat = R_Sat_ECEF/norm(R_Sat_ECEF);

% Calculate satellite altitude [m]
geoLLA = ECEF_to_gLLA(R_Sat_ECEF, Constants);
alt = geoLLA(3);
alt = alt/1000;  %Convert to [km] for HP Model

% Check if the height is within HP Model Limits
if alt >= 1000 || alt <= 100
    rho_atm = 0;
    return
end

% Calculate the right ascension and declination of the Sun [rads]
[SunAngles] = PolarAngles(R_Sun);
DEC_Sun = SunAngles(1);
RA_Sun  = SunAngles(2);

% Calculate a unit vector to the diurnal bulge
RA_lag      = 0.523599; % Right ascension lag ~ 30 degrees [rads]

e_bulge(1) = cos(DEC_Sun) * cos(RA_Sun + RA_lag);
e_bulge(2) = cos(DEC_Sun) * sin(RA_Sun + RA_lag);
e_bulge(3) = sin(DEC_Sun);
```

```
% ================================================================================
%% Extract coefficients for HP Model

%Altitude [km]   %Min Density [g/km^3]  %Max Density [g/km^3]
HP_Coefficients = ...
    [  100      497400.0      497400.0
       120       24900.0       24900.0
       130        8377.0        8710.0
       140        3899.0        4059.0
       150        2122.0        2215.0
       160        1263.0        1344.0
       170         800.8         875.8
       180         528.3         601.0
       190         361.7         429.7
       200         255.7         316.2
       210         183.9         239.6
       220         134.1         185.3
       230          99.49        145.5
       240          74.88        115.7
       250          57.09         93.08
       260          44.03         75.55
       270          34.30         61.82
       280          26.97         50.95
       290          21.39         42.26
       300          17.08         35.26
       320          10.99         25.11
       340           7.214        18.19
       360           4.824        13.37
       380           3.274         9.955
       400           2.249         7.492
       420           1.558         5.684
       440           1.091         4.355
       460           0.7701        3.362
       480           0.5474        2.612
       500           0.3916        2.042
       520           0.2819        1.605
       540           0.2042        1.267
       560           0.1488        1.005
       580           0.1092        0.7997
       600           0.08070       0.6390
       620           0.06012       0.5123
       640           0.04519       0.4121
       660           0.03430       0.3325
       680           0.02632       0.2691
       700           0.02043       0.2185
       720           0.01607       0.1779
       740           0.01281       0.1452
       760           0.01036       0.1190
       780           0.008496      0.09776
       800           0.007069      0.08059
       840           0.004680      0.05741
       880           0.003200      0.04210
       920           0.002210      0.03130
       960           0.001560      0.02360
       1000          0.001150      0.01810  ];

h         = HP_Coefficients(:,1);
CoEff_min = HP_Coefficients(:,2);
CoEff_max = HP_Coefficients(:,3);

```

```
%Find the correct height index for interpolation
h_id = 0;
i = 1;
while ~h_id
    if alt >= h(i) && alt< h(i+1)
        h_id = i;
        %break
    end
    i = i+1;
end

% Calculate the minimum and maximum scale heights [km]
H_min = (h(h_id) - h(h_id+1)) / log( CoEff_min(h_id+1) / CoEff_min(h_id) );
H_max = (h(h_id) - h(h_id+1)) / log( CoEff_max(h_id+1) / CoEff_max(h_id) );

% Calculate the minimum and maximum density values [g/km^3]
rho_min = CoEff_min(h_id) * exp( (h(h_id) - alt)/H_min);
rho_max = CoEff_max(h_id) * exp( (h(h_id) - alt)/H_max);

% Calculate angles between the satellite and the apex of the diurnal bulge
%n = 2; %(Low inclination orbits)
%n = 3; %(PROBA-3 Formation)
n = 5; %(PRISMA Formation)
%n = 5; %(PEOinLEO Formation)
%n = 6; %(Polar orbits)

cos_npsi = ( 0.5 + 0.5*dot(e_Sat, e_bulge) )^(n/2);

% Calculate the density, accounting for the diurnal variations [g/km^3]
rho_atm = rho_min + ( rho_max - rho_min ) * cos_npsi;

% Convert to [kg/m^3]
rho_atm = rho_atm*1e-12;

end
```

E.4 Acceleration Due to Solar Radiation Pressure

```
function r_ddot = AccelSolRad(r_Sat,r_Sun,A_Solar,M_Sat,Cr,P_Solar,AU,R_Earth)
%% Solar Radiation Pressure ========================================================
% Description: This function calculates the worst-case acceleration caused
% by solar radiation pressure, assumed acting on the full-frontal area of
% the spacecraft .
%
% Inputs:
%   r_sat       - Spacecraft position vector (ECI) [m]
%   r_Sun       - Sun position vector (ECI) [m]
%   A_solar     - Cross-sectional area exposed to sunlight [m^2]
%   M_sat       - Spacecraft mass [kg]
%   Cr          - Coefficient of radiation pressure [unitless]
%   P_Solar     - Nominal Solar radiation pressure at 1 AU [W/m^2]
%   AU          - Astronomical unit of length [m]
%   R_Earth     - Radius of the Earth [m]
% Outputs:
%   r_ddot - Acceleration of the spacecraft [m/s^2]
%
% References:
%   Montenbruck, 'Satellite Orbits', pages 77-83, 2001
%
% Created by:  Cory Fraser - JUN 23, 2018
% Latest Edit: Cory Fraser - JUN 23, 2018
% Copyright(c) 2018 by Cory Fraser
% =================================================================================

%Unit vector pointing from the Sun to the Satellite
e_SatSun = (r_Sat - r_Sun)/norm(r_Sat - r_Sun);

%Unit vector pointing from the Satellite to the Sun
e_SunSat = -e_SatSun;

%Evaluate shadow effects
nu = AccelSolRadCyl(r_Sat,r_Sun, R_Earth);

% Acceleration
r_ddot = -nu * P_Solar * Cr * (A_Solar/M_Sat) * AU * AU *...
            (r_Sun-r_Sat)/norm(r_Sun-r_Sat)^3;

end
```

E.5 Solar Illumination Factor for SRP

```
function nu = AccelSolRadCyl(r_Sat,r_Sun, R_Earth)
%% Solar Radiation Pressure ================================================
% Description: This function calculates the fraction of the spacecraft in
% the shadow of the Earth, assuming that the shadow created by the Earth
% extends as a cylinder.
%
% Inputs:
%   r_Sat       - Spacecraft position vector (ECI) [m]
%   r_Sun       - Sun position vector (ECI) [m]
%   R_Earth     - Radius of the Earth [m]

% Outputs:
%   nu - Illumination Factor (1 = full illumination, 0 = eclipse)
%
% References:
%   Montenbruck, 'Satellite Orbits', pages 80-81, 2001
%
% Created by:  Cory Fraser - JUN 23, 2018
% Latest Edit: Cory Fraser - JUL 25, 2018
% Copyright(c) 2018 by Cory Fraser
% =========================================================================

%Unit vector pointing from the Earth to the Sun
e_Sun = r_Sun/norm(r_Sun);

%Scalar & vector projections of the spacecraft position on the Sun vector
s_proj = dot(r_Sat, e_Sun);
v_proj = s_proj*e_Sun;

%Illuminated Case 1: Spacecraft is on the Sun-side of Earth
Case1 = s_proj > 0;

%Illuminated Case 2: Spacecraft is outside Earth's cylindrical shadow
Case2 = norm(r_Sat-v_proj) > R_Earth;

if ( Case1 || Case2)
    nu = 1;
else
    nu = 0;
end

end
```

E.6 Acceleration Due to Thirdy Body Gravity

```
function r_ddot = AccelThirdBody(R_sat,R_mass,mu)
%% Third-body Acceleration ================================================
% Description: This function calculates the acceleration caused on a
% spacecraft due a large point mass.
%
% Inputs:
%   R_sat  - Spacecraft position vector [m]
%   R_mass - Point mass position vector [m]
%   mu     - Gravitational coefficient of point mass [m^3/s^2]
%
% Outputs:
%   r_ddot - Acceleration of the spacecraft
%
% References:
%   Montenbruck, 'Satellite Orbits', pages 69-70, 2001
%
% Created by:  Cory Fraser - JUN 20, 2018
% Latest Edit: Cory Fraser - JUN 20, 2018
% Copyright(c) 2018 by Cory Fraser
% ========================================================================

% Relative position of satellite with respect to the mass
R_rel = R_sat - R_mass;

% Acceleration
r_ddot = -mu*(R_rel/(norm(R_rel)^3)+R_mass/(norm(R_mass)^3));

end
```

Appendix F

EKF MATLAB Code

The MATLAB script used to run the formation flying navigation simulation is contained in this appendix, and is referred to as the *run script*. Additional functions are included here as well, which complete the propagation and correction phases of the EKF algorithm as summarized in Chapter 3. The run script initializes the spacecraft formation, the orbit propagator, and the EKFs, and calls the Simulink model that contains the formation flying navigation routines. The sample MATLAB function presented here include:

`FFNAV_EKF_RunSim.m`: Run the formation flying navigation simulations.

`FFNAV_EKF_Propagation.m`: Propagate the state estimate and error covariance within the EKF.

`FFNAV_EKFdot.m`: Calculates the state derivative vector.

`FFNAV_EKF_STM.m`: Calculates the linearized dynamics state transition matrix.

`FFNAV_EKF_Correction.m`: Correct the state estimate and error covariance within the EKF.

`Observability.m`: Compute the nonlinear observability matrix for the proposed EKF system.

F.1 Simulation Run

```
%% FFNAV_RunSim ============================================================
% FFNAV Extended Kalman Filter Simulations ================================
% Description: This script executes an extended kalman filtering algorithm
% for relative navigation of a spacecraft formation. The exact nonlinear
% equations of relative motion are utilized in the dynamics model, and GPS
% measurements are simulated using an orbit propagator with various
% perturbations and noise sources.
%
% Inputs:
%   Parameter_filename  - Spacecraft and formation parameters filename
%   EKF_flag            - Select filter (EKF, MLE-EKF, or FAEKF)
%   Q_adapt             - Choose Q-adaptations (ON/OFF)
%   R_adapt             - Choose R-adaptations (ON/OFF)
%   Smooth_flag         - Choose residual fixed-window smoothing (ON/OFF)
%   N_window            - Fixed-window size for smoothing
%   PropOptions.        - Structure for orbit propagator settings
%   time_start          - Start time for the simulation
%   time_step           - Time step of the simulation
%   orbit_num           - Number of orbits to simulate
%   time_end            - End time of the simulation
%   save_flag           - Choose to save output to a .mat file
%   post_flag           - Choose to post-process the data
%   n_start             - Starting point for post-processing
%   n_end               - Ending point for post-processing
%   plot_flag           - Choose to create output plots
%   print_flag          - Choose to print plots to .eps files
%   profiler_flag       - Choose to show performance profiler
%
% Outputs:
%   Results.mat file
%   Figures & animations (optional)
%   RMS Error Analysis (optional)
%
% Other Functions Called:
%   FFNAV_EKF_MakeParams.m      - Create FAEKF parameters (if none found)
%   OrbitalMechanicsConstants   - Initialize astrodynamics constants
%   FFNAV_EKF_Sim.slx           - Simulink simulation of the EKFs
%
% Created by:  Cory Fraser - OCT 31, 2017
% Latest Edit: Cory Fraser - OCT 21, 2018
% Copyright(c) 2018 by Cory Fraser
% ========================================================================

%% Initialization
clear
close all

fprintf('\n-------------------------------------------------------------------------\n')
fprintf(' \t\t FFNAV EXTENDED KALMAN FILTER SIMULATION \t\t              \n')
fprintf('-------------------------------------------------------------------------\n')

% Define path to parameters/output data
pathname = fileparts('C:\Users\Cory\Desktop\FFNAV Data\');

% Check for parameter files
if isempty(dir([pathname, '\*Params.mat']))
```

```
57      fprintf('Initialization parameters were not found in this directory...\n')
58      pause(2)
59      fprintf('Generating initial parameters... \n')
60
61      mydir  = pwd;                          %Get current directory
62      idcs   = strfind(mydir,filesep);       %Index file separations
63      newdir = mydir(1:idcs(end)-1);         %Path to higher folder (1-level up)
64      addpath(newdir)                        %Add path to location of MakeParams
65      FFNAV_EKF_MakeParams;                  %Create a parameter file
66  end
67
68  % ======================================================================
69  %% Select Formation Configuration Parameter File
70
71  Parameter_filename = 'PRISMA';
72
73  %Thesis Test Cases
74      % PRISMA      = PRISMA Mission (LEO, low eccentricty)
75      % PROBA-3     = PROBA-3 Mission (HEO, high eccentricity)
76      % PEOinLEO    = Modified PRISMA, with inclination change (LEO,low e,delta i)
77
78  file_in = fullfile(pathname, ['FFNAV_', Parameter_filename, '_Params.mat']);
79  load(file_in)
80  fprintf('\nParameter file loaded : %s  \n', [Parameter_filename, '_Params.mat'])
81
82  % ======================================================================
83  %% Select Orbit Propagator Options (1 = ON, 0 = OFF)
84  PropOptions.J2              = 1;
85  PropOptions.Sun             = 1;
86  PropOptions.Moon            = 1;
87  PropOptions.Drag            = 1;
88  PropOptions.SolarRad        = 1;
89
90  %Add path to orbit propagator, load orbital mechanics constants
91  addpath('C:\Users\Cory\Desktop\FFNAV Code\Orbit Propagator')
92  OrbitalMechanicsConstants
93
94  fprintf('Astrodynamic constants: %s \n', ConstantsModel)
95  fprintf('Orbital Perturbations : ')
96  if (PropOptions.J2 || PropOptions.GravEl_Earth || PropOptions.Sun || ...
97          PropOptions.Moon ||  PropOptions.Planets || PropOptions.Relativity...
98          || PropOptions.Drag || PropOptions.SolarRad )
99      fprintf('On \n')
100 else
101     fprintf('None (Two-Body Motion) \n')
102 end
103 if (PropOptions.J2)
104     fprintf('    - Earth''s Oblateness (J2) \n')
105 end
106 if (PropOptions.Sun)
107     fprintf('    - Solar Third-Body Gravity\n')
108 end
109 if (PropOptions.Moon)
110     fprintf('    - Lunar Third-Body Gravity \n')
111 end
112 if (PropOptions.SolarRad)
113     fprintf('    - Solar Radiation Pressure \n')
114 end
115 if (PropOptions.Drag)
116     fprintf('    - Atmospheric Drag \n')
```

```
end

% Create PropOptions for target spacecraft (same for both spacecraft)
PropOptions_target.J2              = PropOptions.J2;
PropOptions_target.Sun             = PropOptions.Sun;
PropOptions_target.Moon            = PropOptions.Moon;
PropOptions_target.Drag            = PropOptions.Drag;
PropOptions_target.SolarRad        = PropOptions.SolarRad;

% Create PropOptions for chaser spacecraft (same for both spacecraft)
PropOptions_chaser.J2              = PropOptions.J2;
PropOptions_chaser.Sun             = PropOptions.Sun;
PropOptions_chaser.Moon            = PropOptions.Moon;
PropOptions_chaser.Drag            = PropOptions.Drag;
PropOptions_chaser.SolarRad        = PropOptions.SolarRad;

% ======================================================================
%% Select Kalman Filter Options

EKF_flag = 0;
% 0 = EKF           Extended Kalman Filter
% 1 = MLE-EKF       MLE Adaptive Extended Kalman Filter
% 2 = FAEKF         Fuzzy Adaptive Extended Kalman Filter

% Select Adaptation Options - MLE-AEKF & FAEKF (1 = ON, 0 = OFF)
Q_adapt = 0;
R_adapt = 0;

% Select Residuals Smoothing Options - MLE-AEKF & FAEKF (1 = ON, 0 = OFF)
Smooth_flag = '1';
N_window    = 30;  %Running-Average Window Size

% ======================================================================
%% Select Simulation Options
time_start   = 0;                              % Start time of the simulation
time_step    = 1;                              % Time step for the simulation
orbit_num    = 2;                              % Number of orbits to simulate
time_end     = ceil(orbit_num*T_target);       % End time of the simulation

% Post Processing Options
save_flag     = 'on';     % Save data to a .mat file
post_flag     = 'on';     % Post-process the simulation data
    n_start   = floor(time_end*0.5);
    n_end     = time_end+1;
    cov_flag  = 'on';     % Perform analysis of covariances
plot_flag     = 'on';     % Create output plots
print_flag    = 'off';    % Print plots to .eps files
profiler_flag = 'off';    % Show profile simulink performance

% ======================================================================
% Open the Simulink Model, set simulation parameters

Sim_filename = 'FFNAV_EKF_Sim';
load_system(Sim_filename)
if strcmp(get_param(Sim_filename, 'shown'), 'off')
    open_system(Sim_filename);
end
set_param(Sim_filename, 'Solver', 'ode4')
set_param(Sim_filename, 'SolverType', 'Fixed-step')
set_param(Sim_filename, 'FixedStep', 'time_step')
```

```
177 set_param(Sim_filename, 'RelTol','1e-6')
178 set_param(Sim_filename, 'AbsTol','1e-6')
179 set_param(Sim_filename, 'StartTime', 'time_start')
180 set_param(Sim_filename, 'StopTime', 'time_end')
181 set_param(Sim_filename, 'Profile', profiler_flag)
182 fprintf('\nSimulink model loaded : %s.slx \n', Sim_filename)
183
184 % ========================================================================
185 %% Run the EKF Simulation
186
187 %Add path to EKF Functions
188 addpath('C:\Users\Cory\Desktop\FFNAV Code\EKF Algorithms')
189
190 switch EKF_flag
191
192     %Extended Kalman Filter
193     case 0
194         EKF_name = 'EKF';
195         fprintf('Filtering algorithm   : EKF \n')
196         File_specifier = 'NOadapt.mat';
197
198     %MLE Adaptive Extended Kalman Filter
199     case 1
200         EKF_name = 'MLE_EKF';
201         fprintf('Filtering algorithm   : MLE Adaptive EKF \n')
202         fprintf('RTS Smoother          : On, N = %i \n', N_window)
203
204         %Set Q and R Adaptation Switches
205         if (Q_adapt)
206             fprintf('Q-adaptations         : On \n')
207         else
208             fprintf('Q-adaptations         : Off \n')
209         end
210         if (R_adapt)
211             fprintf('R-adaptations         : On \n')
212         else
213             fprintf('R-adaptations         : Off \n')
214         end
215
216     %Fuzzy Adaptive Extended Kalman Filter
217     case 2
218         EKF_name = 'FAEKF';
219         fprintf('Filtering algorithm   : Fuzzy Adaptive EKF \n')
220
221         % Initialize Fuzzy Inference System
222         fprintf('Fuzzy system loaded   :')
223         MSFplot_flag = 0; % 1 = Plot Membership Functions
224         FUZZY_props = FFNAV_FAEKF_FuzzyInitialize(MSFplot_flag);
225
226         %Set Smoothing Settings
227         if Smooth_flag == '1'
228             fprintf('Residual smoothing    : On, N = %i \n', N_window)
229         else
230             fprintf('Residual smoothing    : Off \n')
231         end
232         set_param('EKF_Sim_FAEKF/Kalman Filter/FAEKF/Adaptations/Switch: ...
               Smoothing On//Off', 'sw', Smooth_flag)
233
234         %Set Q and R Adaptation Switches
235         if (Q_adapt)
```

```
236             fprintf('Q-adaptations          : On \n')
237         else
238             fprintf('Q-adaptations          : Off \n')
239         end
240         if (R_adapt)
241             fprintf('R-adaptations          : On \n')
242         else
243             fprintf('R-adaptations          : Off \n')
244         end
245 end
246
247 %Set output filename based on adaptation scheme
248 if (EKF_flag~=0) && (Q_adapt) && (R_adapt)
249     File_specifier = 'QRadapt.mat';
250 elseif (EKF_flag~=0) && (Q_adapt)
251     File_specifier = 'Qadapt.mat';
252 elseif (EKF_flag~=0) && (R_adapt)
253     File_specifier = 'Radapt.mat';
254 else
255     File_specifier = 'NOadapt.mat';
256 end
257
258 % Start the simulation
259 fprintf('Simulation duration   : Orbital period x %i \n', orbit_num)
260 fprintf('                      : %.0f seconds \n', time_end)
261 fprintf('.............................................\n')
262 fprintf('....... running the simulation ..............\n')
263 fprintf('.............................................\n')
264
265 % Turn on the profiler
266 if strcmp(profiler_flag,'on')
267     profile on
268 end
269
270 % Execute the simulation
271 tic;
272 sim(Sim_filename);
273 t_run = toc;
274
275 % Shut off the profiler
276 if strcmp(profiler_flag,'on')
277     profile off
278 end
279
280 fprintf('EKF simulation complete \n')
281 fprintf('Runtime = %f \n', t_run)
282 fprintf('.............................................\n')
283
284 % ========================================================================
285 %% Clean-up and save data
286 clearvars -except   EKF_output   GPS_output     NERM_output             ...
287                     EKF_flag     EKF_name       time        t_run       ...
288                     r_clean      v_clean        R_Earth     mu          ...
289                     r_target_ECI v_target_ECI                           ...
290                     r_chaser_ECI v_chaser_ECI                           ...
291                     theta_target rt_clean       rt_dot_clean            ...
292                     theta_clean  theta_dot_clean                        ...
293                     Pv_obs       Pv_smoothed    Pr_theo                 ...
294                     P_error      Pdot_error     P0                      ...
295                     Q_output     R_output       Q0          R0          ...
```

```
                    lambda       NIS            deltaDOD    DOD_true    ...
                    save_flag    post_flag      plot_flag   print_flag  ...
                    pathname     Parameter_filename         Sim_filename...
                    File_specifier ...
                    profiler_flag  cov_flag Q_adapt R_adapt n_start n_end;

if strcmp(save_flag, 'on') %(1:end-10) for line below
    file_out = strcat(Parameter_filename, ['_', EKF_name,...
                            '_',File_specifier]);
    clear save_output
    file_out = fullfile(pathname, file_out);
    save(file_out);
end

% ======================================================================
%% Post Processing

if strcmp(post_flag, 'on')
    tic;
    FFNAV_EKF_PostProcess(file_out)
    t_process = toc;
    fprintf('Post-processing Time = %f \n', t_process)
end

load gong
sound(y,Fs)

% Plotting
if strcmp(plot_flag, 'on')
    FFNAV_EKF_Plotter(file_out)
end

% Computation Profile Viewer
if strcmp(profiler_flag,'on')
    profile viewer
end

fprintf('-------------------------------------------------------------------------\n')
% ======================================================================
```

F.2 EKF Propagation Phase

```
function [output] = FFNAV_EKF_Propagation(input_pre, time_step, mu, Q)
% FFNAV Extended Kalman Filter Propagation ================================
% Description: This function completes the state and covariance propagation
% (time update) of the EKF algorithm. The resulting a priori data is
% then passed to the correction phase of the EKF algorithm.
%
% Inputs:
%   input_pre   - Vector of previous states and covariances
%   time_step   - Time step of the simulation
%   mu          - Earth's gravitational parameter
%   Q           - Process noise covariance matrix
%
% Outputs:
%   output      - Vector of a priori states and covariances
%
% Other Functions Called:
%   FFNAV_EKFdot.m  - Calculates the state vector derivative components
%   FFNAV_EKF_STM.m - Calculates the state transition matrix
%
% Created by:  Cory Fraser - OCT 31, 2017
% Latest Edit: Cory Fraser - JUL 04, 2018
% Copyright(c) 2018 by Cory Fraser
% =========================================================================

%% Initialize and assign data
    T = time_step;              %Time step for integration

% Previous State Vector (10x1 Matrix)
    x            = input_pre(1);
    y            = input_pre(2);
    z            = input_pre(3);
    theta        = input_pre(4);
    rt           = input_pre(5);
    x_dot        = input_pre(6);
    y_dot        = input_pre(7);
    z_dot        = input_pre(8);
    theta_dot    = input_pre(9);
    rt_dot       = input_pre(10);

    state_pre =  [   x; y; z; theta; rt; ...
                        x_dot; y_dot; z_dot; ...
                        theta_dot; rt_dot ];

% Previous State Error Covariance (10x10 Matrix)
    P_pre = [ input_pre(11:20)'
              input_pre(21:30)'
              input_pre(31:40)'
              input_pre(41:50)'
              input_pre(51:60)'
              input_pre(61:70)'
              input_pre(71:80)'
              input_pre(81:90)'
              input_pre(91:100)'
              input_pre(101:110)' ];


```

```
% ========================================================================
%% EKF Propagation Step

% Runge-Kutta-Merson integration to propagate state vector (10x1 Matrix)
    k1 = FFNAV_EKFdot([state_pre], mu);
    k2 = FFNAV_EKFdot([state_pre + (1/3)*k1*T], mu);
    k3 = FFNAV_EKFdot([state_pre + (1/6)*k1*T + (1/6)*k2*T], mu);
    k4 = FFNAV_EKFdot([state_pre + (1/8)*k1*T + (3/8)*k3*T], mu);
    k5 = FFNAV_EKFdot([state_pre + (1/2)*k1*T - (3/2)*k3*T + 2*k4*T], mu);

    state_priori = state_pre + (1/6)*(k1 + 4*k4 + k5)*T;

% Discrete-time state transition matrix to propagate the covariance (10x10 Matrix)
    phi = FFNAV_EKF_STM(input_pre(1:10), T, mu);
    P_priori = phi*P_pre*phi' + Q;

% ========================================================================
%% Convert data into output vector format (1x110 Matrix)

P_priori = [ P_priori(1,:) P_priori(2,:) P_priori(3,:) P_priori(4,:)...
    P_priori(5,:) P_priori(6,:) P_priori(7,:) P_priori(8,:)...
    P_priori(9,:) P_priori(10,:) ];

output = [ state_priori' P_priori]';

end
% ========================================================================
```

F.3 Nonlinear Dynamics Equations

```
function [f] = FFNAV_EKFdot(state, mu)
% FFNAV EKF Dot ===========================================================
% Description: This function defines the nonlinear equations of relative
% motion for formation flying spacecraft. When passed a state vector, this
% function calculates the respective accelerations for the x, y, z, rt and
% theta functions, and returns them to the calling function in a vector.
%
% Inputs:
%   state - The current state vector
%   mu    - Earth's gravitational parameter
%
% Outputs:
%   f       - Diffential equation output accelerations
%
% Created by: Cory Fraser - FALL... 2015
% Last Edits: Cory Fraser - JUL 04, 2018
% Copyright(c) 2018 by Cory Fraser
% =========================================================================

%% State Vector
 x           = state(1);
 y           = state(2);
 z           = state(3);
 theta       = state(4);
 rt          = state(5);
 x_dot       = state(6);
 y_dot       = state(7);
 z_dot       = state(8);
 theta_dot   = state(9);
 rt_dot      = state(10);

  rc         = sqrt((rt + x)^2 + y^2 + z^2);

%% Derivatives of the state variables
 %x_dot      = x_dot_priori;
 %y_dot      = y_dot_priori;
 %z_dot      = z_dot_priori;
 %theta_dot  = theta_dot_priori;
 %rt_dot     = rt_dot_priori;
 x_ddot      = x*theta_dot^2 + ...
     2*theta_dot*(y_dot-y*rt_dot/rt)+mu/rt^2-mu*(rt+x)/(rc^3);
 y_ddot      = y*theta_dot^2 - 2*theta_dot*(x_dot-x*rt_dot/rt)-mu*y/(rc^3);
 z_ddot      = -mu*z/rc^3;
 theta_ddot  = -2*rt_dot*theta_dot/rt;
 rt_ddot     = rt*theta_dot^2-mu/rt^2;

%% Assembling the derivative of the state vector
f = [ x_dot; y_dot; z_dot; theta_dot; rt_dot; x_ddot; y_ddot; z_ddot; ...
    theta_ddot; rt_ddot ];

end
% =========================================================================
```

F.4 EKF Linearized State Transition Matrix

```
function [Phi, F] = FFNAV_EKF_STM(state_pre, T, mu)
% FFNAV State Transition Matrix ==========================================
% Description: This function calculates the linearized state matrix
% (Jacobian) and the discrete-time state transition matrix, given the
% current state and the time step.
%
% Inputs:
%   state_pre - The previous state vector
%   T         - Time step of the simulation
%   mu        - Earth's gravitational parameter

% Outputs:
%   Phi - State Transition Matrix
%   F   - Jacobian of the Dynamic Model
%
% Created by: Cory Fraser - AUG 31, 2017
% Last Edits: Cory Fraser - JUL 04, 2018
% Copyright(c) 2018 by Cory Fraser
% ========================================================================

%% Initialize the states

 x          = state_pre(1);
 y          = state_pre(2);
 z          = state_pre(3);
 theta      = state_pre(4);
 rt         = state_pre(5);
 x_dot      = state_pre(6);
 y_dot      = state_pre(7);
 z_dot      = state_pre(8);
 theta_dot  = state_pre(9);
 rt_dot     = state_pre(10);

 rc        = sqrt((rt + x)^2 + y^2 + z^2);

% ========================================================================
%% Partial derivatives of the 5 non-linear equations

%Derivatives of x-acceleration equation
    dxddot_dx           = (mu/rc^3)*(3*((rt+x)/rc)^2 - 1) + theta_dot^2;
    dxddot_dy           = 3*mu*((rt+x)/(rc^5))*y - ...
                          2*theta_dot*rt_dot/rt;
    dxddot_dz           = 3*mu*((rt+x)/rc^5)*z;
    dxddot_dtheta       = 0;
    dxddot_drt          = 2*theta_dot*rt_dot*y/rt^2 - ...
                          2*mu/rt^3 + (mu/rc^3)*(3*((rt+x)/rc)^2 - 1);
    dxddot_dx_dot       = 0;
    dxddot_dy_dot       = 2*theta_dot;
    dxddot_dz_dot       = 0;
    dxddot_dtheta_dot   = 2*(theta_dot*x + y_dot - ...
                          (rt_dot/rt)*y);
    dxddot_drt_dot      = -2*theta_dot*y/rt;

%Derivatives of y-acceleration equation
    dyddot_dx           = 2*theta_dot*rt_dot/rt + ...
                          3*mu*((rt+x)/rc^5)*y;
```

```
    dyddot_dy           = (mu/rc^3)*(3*(y/rc)^2-1) + theta_dot^2;
    dyddot_dz           = 3*mu*(y/rc^5)*z;
    dyddot_dtheta       = 0;
    dyddot_drt          = 3*mu*((rt+x)/rc^5)*y - ...
                          2*theta_dot*rt_dot*x/rt^2;
    dyddot_dx_dot       = -2*theta_dot;
    dyddot_dy_dot       = 0;
    dyddot_dz_dot       = 0;
    dyddot_dtheta_dot   = 2*(theta_dot*y - x_dot + ...
                          rt_dot*x/rt);
    dyddot_drt_dot      = 2*theta_dot*x/rt;

%Derivatives of z-acceleration equation
    dzddot_dx           = 3*mu*((rt+x)/rc^5)*z;
    dzddot_dy           = 3*mu*y*z/rc^5;
    dzddot_dz           = (mu/rc^3)*(3*(z/rc)^2-1);
    dzddot_dtheta       = 0;
    dzddot_drt          = 3*mu*((rt+x)/rc^5)*z;
    dzddot_dx_dot       = 0;
    dzddot_dy_dot       = 0;
    dzddot_dz_dot       = 0;
    dzddot_dtheta_dot   = 0;
    dzddot_drt_dot      = 0;

%Derivatives of theta-acceleration equation
    dtheta_ddot_dx          = 0;
    dtheta_ddot_dy          = 0;
    dtheta_ddot_dz          = 0;
    dtheta_ddot_dtheta      = 0;
    dtheta_ddot_drt         = 2*rt_dot*theta_dot/rt^2;
    dtheta_ddot_dx_dot      = 0;
    dtheta_ddot_dy_dot      = 0;
    dtheta_ddot_dz_dot      = 0;
    dtheta_ddot_dtheta_dot  = -2*rt_dot/rt;
    dtheta_ddot_drt_dot     = -2*theta_dot/rt;

%Derivatives of rt-acceleration equation
    drt_ddot_dx             = 0;
    drt_ddot_dy             = 0;
    drt_ddot_dz             = 0;
    drt_ddot_dtheta         = 0;
    drt_ddot_drt            = theta_dot^2 + 2*mu/rt^3;
    drt_ddot_dx_dot         = 0;
    drt_ddot_dy_dot         = 0;
    drt_ddot_dz_dot         = 0;
    drt_ddot_dtheta_dot     = 2*theta_dot*rt;
    drt_ddot_drt_dot        = 0;

% ========================================================================
%% Assembling the Jacobian (10x10 Matrix)
    F11 = zeros(5,5);
    F12 = eye(5,5);

    F21 = [dxddot_dx dxddot_dy dxddot_dz dxddot_dtheta dxddot_drt];
    F22 = [dxddot_dx_dot dxddot_dy_dot dxddot_dz_dot dxddot_dtheta_dot ...
       dxddot_drt_dot];

    F31 = [dyddot_dx dyddot_dy dyddot_dz dyddot_dtheta dyddot_drt];
    F32 = [dyddot_dx_dot dyddot_dy_dot dyddot_dz_dot dyddot_dtheta_dot ...
       dyddot_drt_dot];
```

```
115
116     F41 = [dzddot_dx dzddot_dy dzddot_dz dzddot_dtheta dzddot_drt];
117     F42 = [dzddot_dx_dot dzddot_dy_dot dzddot_dz_dot dzddot_dtheta_dot ...
           dzddot_drt_dot];
118
119     F51 = [dtheta_ddot_dx dtheta_ddot_dy dtheta_ddot_dz dtheta_ddot_dtheta ...
           dtheta_ddot_drt];
120     F52 = [dtheta_ddot_dx_dot dtheta_ddot_dy_dot dtheta_ddot_dz_dot ...
121            dtheta_ddot_dtheta_dot dtheta_ddot_drt_dot];
122
123     F61 = [drt_ddot_dx drt_ddot_dy drt_ddot_dz drt_ddot_dtheta drt_ddot_drt];
124     F62 = [drt_ddot_dx_dot drt_ddot_dy_dot drt_ddot_dz_dot drt_ddot_dtheta_dot ...
125            drt_ddot_drt_dot];
126
127     F = [   F11 F12
128             F21 F22
129             F31 F32
130             F41 F42
131             F51 F52
132             F61 F62 ];
133
134 % ========================================================================
135 %% Calculate the approximate state transition matrix, Phi = expm(F_k*T)
136     Phi = (F*T)*(((F*T)/2)*(((F*T)/3)*((F*T)/4 + eye(10,10))+eye(10,10))...
137           +eye(10,10))+eye(10,10);
138 end
139 % ========================================================================
```

F.5 EKF Correction Phase

```
function [output] = FFNAV_EKF_Correction(input_priori, sensors, R)
% FFNAV Extended Kalman Filter ========================================
% Description: This function completes the state and covariance correction
% (measurement update) of the EKF algorithm. The resulting a posteriori
% data is used as the state estimate for the next time step.
%
% Inputs:
%   input_priori – Vector of a priori states and covariances
%   sensors      – Vector of measurements
%   R            – Measurement noise covariance matrix
%
% Outputs:
%   output      – Vector of a priori states and covariances
%
% Created by: Cory Fraser – OCT 31, 2017
% Last Edit : Cory Fraser – JUL 04, 2018
% Copyright(c) 2018 by Cory Fraser
% ========================================================================

%% Initialize and assign data
    coder.extrinsic('chol');

% Propagated State Vector (10x1 Matrix)
x_priori            = input_priori(1);
y_priori            = input_priori(2);
z_priori            = input_priori(3);
theta_priori        = input_priori(4);
rt_priori           = input_priori(5);
x_dot_priori        = input_priori(6);
y_dot_priori        = input_priori(7);
z_dot_priori        = input_priori(8);
theta_dot_priori    = input_priori(9);
rt_dot_priori       = input_priori(10);

state_priori =  [   x_priori; y_priori; z_priori; theta_priori; rt_priori; ...
                    x_dot_priori; y_dot_priori; z_dot_priori; ...
                    theta_dot_priori; rt_dot_priori ]; %(10 x 1)

% Propagated State Error Covariance (10x10 Matrix)
P_priori = [    input_priori(11:20)'
                input_priori(21:30)'
                input_priori(31:40)'
                input_priori(41:50)'
                input_priori(51:60)'
                input_priori(61:70)'
                input_priori(71:80)'
                input_priori(81:90)'
                input_priori(91:100)'
                input_priori(101:110)' ];

% Measurement Vector (7x1 Matrix)
x_m       = sensors(1);
y_m       = sensors(2);
z_m       = sensors(3);
theta_m   = sensors(4);
x_dot_m   = sensors(5);
```

```
y_dot_m   = sensors(6);
z_dot_m   = sensors(7);

Z_m = [     x_m
            y_m
            z_m
            theta_m
            x_dot_m
            y_dot_m
            z_dot_m ];
% ========================================================================
%% Defining the measurement model (7 x 10 matrix)

%Relative position, velocity, and theta
H = [   1 0 0 0 0 0 0 0 0 0
        0 1 0 0 0 0 0 0 0 0
        0 0 1 0 0 0 0 0 0 0
        0 0 0 1 0 0 0 0 0 0
        0 0 0 0 0 1 0 0 0 0
        0 0 0 0 0 0 1 0 0 0
        0 0 0 0 0 0 0 1 0 0 ];

% ========================================================================
%% EKF Correction Step

% Calulating the estimated measurements (7x1 matrix)
Z_est = H*state_priori;

%Calculating the innovations (7x1 matrix)
innovations   = Z_m - Z_est;

%Calculating the theoretical covariance of the innovations (7x7 Matrix)
Pr_theo = H*P_priori*H'+R;

%Pr_inv  = inv(Pr_theo);        %This inversion gives numerical issues
Pr_inv = Pr_theo\eye(7);        %Seems sufficient for EKF, FAEKF

%Covariance Inversion using Cholesky Decomposition, PSDS Check
%{
    Pr_inv = zeros(7);
    PSDS_flag = 0;    %Flag for Positive Semi-Definite Symmetric
    [M_upper, PSDS_flag] = chol(Pr_theo);

    if PSDS_flag~=0
        fprintf('Pr_theo is not PDS \n')
    end
    M_inv = M_upper \ eye(7);
    Pr_inv = M_inv * M_inv';
%}

%Calculating the Kalman Gain (10x7 Matrix)
%K = (P_priori*H')*inv(H*P_priori*H'+R_k);
K = (P_priori*H')*Pr_inv;

%Correcting the state estimate (10x1 Matrix)
correction  = K*innovations;
state_post  = state_priori + correction;

%Correct the state error covariance matrix - Joseph's Form (10x10 Matrix)
P_post = (eye(10)-K*H)*P_priori*(eye(10)-K*H)' + K*R*K';
```

```
117
118 %% ======================================================================
119 % Converting data into output vector format
120 P_post = [ P_post(1,:) P_post(2,:) P_post(3,:) P_post(4,:) P_post(5,:) ...
       P_post(6,:)...
121             P_post(7,:) P_post(8,:) P_post(9,:) P_post(10,:) ]; %(1x100)
122
123 Pr_theo = [ Pr_theo(1,:) Pr_theo(2,:) Pr_theo(3,:) Pr_theo(4,:) Pr_theo(5,:) ...
       Pr_theo(6,:)...
124             Pr_theo(7,:)]; %(1x49)
125
126 K = [ K(1,:) K(2,:) K(3,:) K(4,:) K(5,:) K(6,:) K(7,:) K(8,:) K(9,:) K(10,:)]; ...
       %(1x70)
127
128 output = [ state_post' P_post Z_est' Pr_theo innovations' K]'; %(1x243)
129
130 end
```

F.6 Observability Analysis

```
% FFNAV Extended Kalman Filter Observability ==============================
% Description: This script computes the nonlinear observability test matrix
% for the formation flying EKF scenario. The MATLAB Symbolic Tooblox is
% necessary to perform the analytic derivations. The first three orders of
% the observability matrix are sufficient to demonstrate local
% observability, but up to the ninth order can be calculated here; note
% that the observability matrix becomes increasingly complex to compute,
% and evalute up to the ninth-order requires roughly
%
% Inputs:
%   None
%
% Outputs:
%   Observability_results.mat file
%
% Other Functions Called:
%   jacobian - Calculates the symbolic Jacobian of a matrix
%
% Created by:  Cory Fraser - OCT 31, 2016
% Latest Edit: Cory Fraser - DEC 21, 2016
% Copyright(c) 2018 by Cory Fraser
% ========================================================================

%% Initialization
close all; clear; clc;

% Define symbols
syms x      y       z       theta       rt      mu
syms x_dot y_dot  z_dot  theta_dot  rt_dot
syms x_ddot y_ddot  z_ddot  theta_ddot  rt_ddot

% Define the state vector
state_vec = [ x y z theta rt x_dot  y_dot   z_dot   theta_dot   rt_dot].';

% Define the dynamics equations
x_ddot = @(x,y,z,theta,rt,x_dot,y_dot,z_dot,theta_dot,rt_dot) ...
            theta_dot^2*x + 2*theta_dot*y_dot - 2*theta_dot*rt_dot*y/rt ...
            + mu/rt^2 - mu*(rt+x)*((rt+x)^2+y^2+z^2)^(-3/2);

y_ddot = @(x,y,z,theta,rt,x_dot,y_dot,z_dot, theta_dot,rt_dot) ...
            theta_dot^2*y - 2*theta_dot*x_dot + 2*theta_dot*rt_dot*x/rt...
            - mu*y*((rt+x)^2+y^2+z^2)^(-3/2);

z_ddot = @(x,y,z,theta,rt,x_dot,y_dot,z_dot,theta_dot,rt_dot) ...
            - mu*z*((rt+x)^2+y^2+z^2)^(-3/2);

theta_ddot = @(x,y,z,theta,rt,x_dot,y_dot,z_dot,theta_dot,rt_dot)...
            - 2*rt_dot*theta_dot/rt;

rt_ddot = @(x,y,z,theta,rt,x_dot,y_dot,z_dot,theta_dot,rt_dot)...
            theta_dot^2*rt - mu/rt^2;

% Calculate the linearized dynamics equations (i.e., the Jacobian)
part_ddot_x = jacobian(x_ddot, [x, y, z, theta, rt, ...
                        x_dot, y_dot, z_dot, theta_dot, rt_dot]);

```

```
part_ddot_y = jacobian(y_ddot, [x, y, z, theta, rt, ...
                        x_dot, y_dot, z_dot, theta_dot, rt_dot]);

part_ddot_z = jacobian(z_ddot, [x, y, z, theta, rt, ...
                        x_dot, y_dot, z_dot, theta_dot, rt_dot]);

part_ddot_theta = jacobian(theta_ddot, [x, y, z, theta, rt,...
                           x_dot, y_dot, z_dot, theta_dot, rt_dot]);

part_ddot_rt = jacobian(rt_ddot, [x, y, z, theta, rt,...
                        x_dot, y_dot, z_dot, theta_dot, rt_dot]);

% Define the linearized measurement matrix
H = [   eye(4,4)    zeros(4,1) zeros(4,3) zeros(4,2)
        zeros(3,4)  zeros(3,1) eye(3,3)   zeros(3,2) ];

% Assemble the derivative of the state vector
f = [ x_dot y_dot z_dot theta_dot rt_dot ...
            x_ddot y_ddot  z_ddot  theta_ddot  rt_ddot].';

%=========================================================================
%% Constructing the Observability Matrix
tic;

%Zeroth-Order Lie Derivative Gradient
G0 = H;
toc

%First-Order Lie Derivative Gradient
L1 = H*f;
G1 = jacobian(L1, state_vec);
toc

%Second-Order Lie Derivative Gradient
L2 = G1*f;
G2 = jacobian(L2, state_vec);
toc

%Third-Order Lie Derivative Gradient
L3 = G2*f;
G3 = jacobian(L3, state_vec);
toc

%Fourth-Order Lie Derivative Gradient
L4 = G3*f;
G4 = jacobian(L4, state_vec);
toc

%Fifth-Order Lie Derivative Gradient
L5 = G4*f;
G5 = jacobian(L5, state_vec);
toc

%Sixth-Order Lie Derivative Gradient
L6 = G5*f;
G6 = jacobian(L6, state_vec);
toc

%Seventh-Order Lie Derivative Gradient
L7 = G6*f;
```

```
G7 = jacobian(L7, state_vec);
toc

%Eighth-Order Lie Derivative Gradient
L8 = G7*f;
G8 = jacobian(L8, state_vec);
toc

%Nineth-Order Lie Derivative Gradient
L9 = G8*f;
G9 = jacobian(L9, state_vec);
toc

%Assemble the observability matrix
OM      = [G0; G1; G2; G3; G4; G5; G6; G7; G8];% G9];
OM_rank = rank(OM)

time = toc

filename = 'Observability_results.mat';
save(filename)
%=========================================================================
```

Appendix G

MLE-AEKF MATLAB Code

Contained in this appendix is the MATLAB script that implements the Maximum Likelihood Adaptive EKF presented in Chapter 4. This function is contained within an embedded MATLAB function block in the formation flying navigation Simulink model, and is activated when using the MLE-AEKF algorithm in the simulation. Whereas the standard EKF uses separate functions for the propagation and correction phases of the EKF, the MLE-AEKF algorithm is completed in a single function. The following source code is presented in this appendix:

`FFNAV_EKF_MLE.m`: Run the formation flying navigation simulations.

`findPSDS.m`: Calculates the nearest positive semi-definite symmetric matrix.

G.1 MLE Adaptive Extended Kalman Filter

```
function [EKF_out, Storage_out, L, Q_update, R_update] = ...
    FFNAV_EKF_MLE(Storage_in, time_step, mu, data_pre, sensors, ...
    Q, R, Q_adapt_flag,R_adapt_flag)
% FFNAV Maximum Likelikelihood Estimation EKF ============================
% Description: This function completes both the propagation and correction
% phases of the EKF, along with adapting the process and measurement noise
% covariances Q and R using an MLE method.
%
% Inputs:
%   Storage_in   - Matrix of past data needed for the RTS Smoother
%   time_step    - Time step of the simulation
%   mu           - Earth's gravitational parameter
%   data_pre     - EKF state and covariance data from previous time step
%   sensors      - Vector of measurements
%   Q            - Process noise covariance matrix
%   Q_adapt_flag - Flag to adapt Q (0 = no adaptation)
%   R            - Measurement noise covariance matrix
%   R_adapt_flag - Flag to adapt R (1 = adapt)
%
% Outputs:
%   EKF_out     - Vector of EKF output data, to be passed to next EKF step
%   Storage_out - Vector of data to be stored in memory for RTS Smoother
%   L           - Value of the pseudo-likelihood cost function
%   Q_update    - Updated process noise covariance matrix
%   R_update    - Updated measurement noise covariance matrix
%
% Other Functions Called:
%   FFNAV_EKFdot      - Calculates the state vector derivative components
%   FFNAV_EKF_STM     - Calculates the state transition matrix
%   findPSDS (option) - Finds the nearest PSDS matrix
%   chol (option)     - Performs a Cholesky decomposition on a PDS matrix
%
% Notes:
%   1) Data_pre comes in as a single column vector
%   2) Storage_in comes in as a matrix, with each row corresponding to the
%           full set of output data from the previous filtering iterations
%
% Created by:  Cory Fraser - AUG 22, 2017
% Latest Edit: Cory Fraser - JUL 04, 2018
% Copyright(c) 2018 by Cory Fraser
% ========================================================================

%% Initialize Parameters

coder.extrinsic('legend');
coder.extrinsic('fprintf');
coder.extrinsic('chol');

% Previous State Estimates
x         = data_pre(1);
y         = data_pre(2);
z         = data_pre(3);
theta     = data_pre(4);
rt        = data_pre(5);
x_dot     = data_pre(6);
y_dot     = data_pre(7);
```

```
z_dot     = data_pre(8);
theta_dot = data_pre(9);
rt_dot    = data_pre(10);

state_pre = zeros(10,1);
state_pre =  [   x; y; z; theta; rt; ...
    x_dot; y_dot; z_dot; ...
    theta_dot; rt_dot ]; %(10 x 1)

% Unpropagated State Error Covariance (10x10)
P_pre = [   data_pre(11:20)'
            data_pre(21:30)'
            data_pre(31:40)'
            data_pre(41:50)'
            data_pre(51:60)'
            data_pre(61:70)'
            data_pre(71:80)'
            data_pre(81:90)'
            data_pre(91:100)'
            data_pre(101:110)' ];

% ========================================================================
%% EKF Step 2: Propagation Phase

%Time step for integration
T = time_step;

%Dynamics Propagation using RK-Merson Method
k1 = FFNAV_EKFdot(state_pre, mu);
k2 = FFNAV_EKFdot(state_pre + (1/3)*k1*T, mu);
k3 = FFNAV_EKFdot(state_pre + (1/6)*k1*T + (1/6)*k2*T, mu);
k4 = FFNAV_EKFdot(state_pre + (1/8)*k1*T + (3/8)*k3*T, mu);
k5 = FFNAV_EKFdot(state_pre + (1/2)*k1*T - (3/2)*k3*T + 2*k4*T, mu);

state_priori = state_pre + (1/6)*(k1 + 4*k4 + k5)*T; %(10 x 1)

%Calculating the a priori state error covariance (10 x 10)
Phi = FFNAV_EKF_STM(state_pre, T, mu);
P_priori = Phi*P_pre*Phi' + Q;

% ========================================================================
%% EKF Step 3: Correction Phase

% Measured position and velocity parameters
x_m      = sensors(1);
y_m      = sensors(2);
z_m      = sensors(3);
theta_m  = sensors(4);
x_dot_m  = sensors(5);
y_dot_m  = sensors(6);
z_dot_m  = sensors(7);

%Assembling the measured outputs (8 x 1 matrix)
Z_m = [ x_m
        y_m
        z_m
        theta_m
        x_dot_m
        y_dot_m
        z_dot_m ];
```

```
117
118 %Defining the measurement model (7 x 10 matrix)
119 H = [   1 0 0 0 0 0 0 0 0 0
120         0 1 0 0 0 0 0 0 0 0
121         0 0 1 0 0 0 0 0 0 0
122         0 0 0 1 0 0 0 0 0 0
123         0 0 0 0 0 1 0 0 0 0
124         0 0 0 0 0 0 1 0 0 0
125         0 0 0 0 0 0 0 1 0 0];
126
127 %Calulating the Estimated Measurements (7x1 matrix)
128 Z_est = H*state_priori;
129
130 %Calculating the residuals (7x1 matrix)
131 residuals = (Z_m - Z_est);
132
133 %Calculating the theoretical Residual Covariance (7x7 Matrix)
134 Pr_theo = H*P_priori*H' + R;
135
136 %Pr_inv  = inv(Pr_theo);        %This inversion gives numerical issues
137 Pr_inv = Pr_theo\eye(7);       %Not sufficient for MLE-AEKF
138
139 %Inverting innovations covariance using Cholesky Decomposition (optional)
140 %{
141 Pr_inv = zeros(7);
142 PSDS_flag = 0;    %Flag for Positive Semi-Definite Symmetric
143 [M_upper, PSDS_flag] = chol(Pr_theo);
144
145 if PSDS_flag~=0
146     fprintf('Pr_theo is not PDS - finding nearest PSDS Matrix \n')
147     [M_upper, PSDS_flag] = chol(findPSDS(Pr_theo));
148 end
149 M_inv = M_upper \ eye(7);
150 Pr_inv = M_inv * M_inv';
151 %}
152
153 %Calculating the Kalman Gain (10x7 matrix)
154 K = (P_priori*H')*Pr_inv;
155
156 %Correcting the State Estimate (10x1 matrix)
157 correction  = K*residuals;
158 state_post  = state_priori + correction;
159
160 %Correct the state error covariance matrix (10 x 10 matrix)
161 P_post = (eye(10)-K*H)*P_priori*(eye(10)-K*H)' + K*R*K';
162
163 %Calculate Negative-Log Likelihood Psuedo-cost function
164 L = log(abs(det(Pr_theo))) + residuals'*Pr_inv*residuals;
165
166 Q_update = Q;
167 R_update = R;
168 % ========================================================================
169
170 %% Maximum Likelihood Estimation - Adaptation Algorithm
171 if (Q_adapt_flag) || (R_adapt_flag)
172
173     N     = size(Storage_in,1)+1;   %Size based on fixed window size
174     N_use = nnz(Storage_in(:,1))+1; %Useful size (# of non-zero rows)
175
176     if N_use > 2
```

```
177
178         % Organizing the stored data
179         state_post_mem      = zeros(10,1,N);
180         state_priori_mem    = zeros(10,1,N);
181         P_post_mem          = zeros(10,10,N);
182         P_priori_mem        = zeros(10,10,N);
183         Phi_mem             = zeros(10,10,N);
184         Zm_mem              = zeros(7,1,N);
185         Zest_mem            = zeros(7,1,N);
186
187         state_post_mem(:,1,N_use)    = state_post;
188         state_priori_mem(:,1,N_use)  = state_priori;
189         Phi_mem(:,:,N_use)           = Phi;
190         P_post_mem(:,:,N_use)        = P_post;
191         P_priori_mem(:,:,N_use)      = P_priori;
192         Zm_mem(:,1,N_use)            = Z_m;
193         Zest_mem(:,1,N_use)          = Z_est;
194
195         % Arranging data - Flipping, so N(ewest) is at last index
196         for j = 1:N_use-1
197             state_post_mem(:,1,N_use-j)      = [Storage_in(j,1:10)]';
198             state_priori_mem(:,1,N_use-j)    = [Storage_in(j,11:20)]';
199
200             P_post_mem(:,:,N_use-j) =...
201               [ Storage_in(j,21:30) %Note that the rows in Storage_in
202                 Storage_in(j,31:40) %correspond to each filter step
203                 Storage_in(j,41:50) % (different than output vector)
204                 Storage_in(j,51:60)
205                 Storage_in(j,61:70)
206                 Storage_in(j,71:80)
207                 Storage_in(j,81:90)
208                 Storage_in(j,91:100)
209                 Storage_in(j,101:110)
210                 Storage_in(j,111:120) ];
211
212             P_priori_mem(:,:,N_use-j) =...
213               [ Storage_in(j,121:130)
214                 Storage_in(j,131:140)
215                 Storage_in(j,141:150)
216                 Storage_in(j,151:160)
217                 Storage_in(j,161:170)
218                 Storage_in(j,171:180)
219                 Storage_in(j,181:190)
220                 Storage_in(j,191:200)
221                 Storage_in(j,201:210)
222                 Storage_in(j,211:220) ];
223
224             Phi_mem(:,:,N_use-j) = ...
225               [ Storage_in(j,221:230)
226                 Storage_in(j,231:240)
227                 Storage_in(j,241:250)
228                 Storage_in(j,251:260)
229                 Storage_in(j,261:270)
230                 Storage_in(j,271:280)
231                 Storage_in(j,281:290)
232                 Storage_in(j,291:300)
233                 Storage_in(j,301:310)
234                 Storage_in(j,311:320) ];
235
236             Zm_mem(:,1,N_use-j)   = [Storage_in(j,321:327)]';
```

```
237             Zest_mem(:,1,N_use-j) = [Storage_in(j,328:334)]';
238         end
239
240     % Extended Kalman Smoother (for k = N-1, N-2,..., 1)
241     state_smooth    = zeros(10,1,N);
242     P_smooth        = zeros(10,10,N);
243     G_smooth        = zeros(10,10,N);
244     Zest_smooth     = zeros(7,1,N);
245
246     k = N_use;
247     state_smooth(:,:,k) = state_post_mem(:,:,k);
248     P_smooth(:,:,k)     = P_post_mem(:,:,k);
249     Zest_smooth(:,:,k)  = H*state_smooth(:,:,k);
250
251     for k = N_use:-1:2
252         P_priori_mem_inv      = P_priori_mem(:,:,k)\eye(10,10);
253
254         G_smooth(:,:,k-1)     = P_post_mem(:,:,k-1)*Phi_mem(:,:,k-1)'...
255                                                         *P_priori_mem_inv;
256
257         state_smooth(:,:,k-1) = state_post_mem(:,:,k-1) + ...
258                                 G_smooth(:,:,k-1)*( state_smooth(:,k) - ...
259                                 state_priori_mem(:,:,k) );
260
261         Zest_smooth(:,:,k-1)  = H*state_smooth(:,:,k-1);
262
263         P_smooth(:,:,k-1)     = P_post_mem(:,:,k-1) + G_smooth(:,:,k-1)*...
264                                 (P_smooth(:,:,k) - P_priori_mem(:,:,k))*...
265                                                     G_smooth(:,:,k-1)';
266     end
267     state_smooth(:,:,1) = state_post_mem(:,:,1);
268     Zest_smooth(:,:,1)  = H*state_smooth(:,:,1);
269     P_smooth(:,:,1)     = P_post_mem(:,:,1);
270     %================================================
271
272     % Adaptation Equations
273     N_start = 3; %Loop from k >= 2 because of 1-Lag
274     BQ = zeros(7,7,N);
275
276     %Q-Adaptations
277     if (Q_adapt_flag)
278         for  k = N_start:1:N_use
279             BQ(:,:,k) = ( Zm_mem(:,:,k) - Zest_smooth(:,:,k) )*...
280                         ( Zm_mem(:,:,k) - Zest_smooth(:,:,k) )';
281         end
282
283         BQ_sum = sum(BQ(:,:,N_start:N_use),3);
284         Q_update = 1/(N_use-N_start+1)*K*BQ_sum*K';
285     end
286
287     %R-Adaptations
288     if (R_adapt_flag)
289         BR = zeros(7,7,N);
290         if ~Q_adapt_flag %BQ was not calculated in Q-adaptations
291             for  k = N_start:1:N_use
292                 BQ(:,:,k) = ( Zm_mem(:,:,k) - Zest_smooth(:,:,k) )*...
293                                 (Zm_mem(:,:,k)-Zest_smooth(:,:,k))';
294                 BR(:,:,k) = BQ(:,:,k) + H*P_smooth(:,:,k)*H';
295             end
296         else %BQ can be used from Q-adaptations
```

```
297             for  k = N_start:1:N_use
298                 BR(:,:,k) = BQ(:,:,k) + H*P_smooth(:,:,k)*H';
299             end
300         end
301         BR_sum = sum(BR(:,:,N_start:N_use),3);
302         R_update = 1/(N_use-N_start+1)*BR_sum;
303     end
304 
305     % ====================================================================
306     %Diagonalize (optional)
307     Q_update = diag(diag(Q_update));
308     R_update = diag(diag(R_update));
309 
310     % ====================================================================
311     %Positive Semidefinite Symmetric Check & Correction (optional)
312     %Q_update = findPSDS(Q_update);
313     %R_update = findPSDS(R_update);
314 
315     end
316 end
317 
318 % ========================================================================
319 %% Converting data into output vector format
320 P_post_out = [ P_post(1,:) P_post(2,:) P_post(3,:) P_post(4,:)...
321                P_post(5,:) P_post(6,:) P_post(7,:) P_post(8,:)...
322                P_post(9,:) P_post(10,:) ]; %(1 x 100)
323 
324 Pr_theo_out = [ Pr_theo(1,:) Pr_theo(2,:) Pr_theo(3,:) Pr_theo(4,:)...
325                 Pr_theo(5,:) Pr_theo(6,:) Pr_theo(7,:)]; %(1 x 49)
326 
327 K_out = [ K(1,:) K(2,:) K(3,:) K(4,:) K(5,:) ...
328           K(6,:) K(7,:) K(8,:) K(9,:) K(10,:)]; %(1 x 70)
329 
330 EKF_out = [ state_post' P_post_out Z_est'...
331             Pr_theo_out residuals' K_out]'; %(1x243)
332 
333 % ========================================================================
334 %% Preparing data vector for Memory Storage
335 P_priori_out = [ P_priori(1,:) P_priori(2,:) P_priori(3,:)...
336                  P_priori(4,:) P_priori(5,:) P_priori(6,:) P_priori(7,:)...
337                  P_priori(8,:) P_priori(9,:) P_priori(10,:) ]; %(1 x 100)
338 
339 Phi_out = [ Phi(1,:) Phi(2,:) Phi(3,:) Phi(4,:) Phi(5,:) Phi(6,:)...
340            Phi(7,:) Phi(8,:) Phi(9,:) Phi(10,:) ]; %(1 x 100)
341 
342 
343 Storage_out = [ state_post' state_priori' P_post_out P_priori_out ...
344                 Phi_out Z_m' Z_est']'; %(1 x 334)
345 
346 end
347 % ========================================================================
```

G.2 Positive Semi-definite Symmetric Matrix Check

```
function A_hat = findPSDS(A)
% FFNAV PSDS Check ========================================================
% Description: This function takes a square matrix and determines the
% nearest positive semi-definite symmetric matrix, based on the Frobenius
% norm.
%
% Inputs:
%   A - a square matrix, which will be converted to the nearest PSDS matrix
%
% Outputs:
%   A_hat - the nearest PSDS matrix to A
%
% References:
%   Higham   - Computing a Nearest Symmetric Positive Semidefinite Matrix
%   D'Errico - "nearestSPD", MathWorks File Exchange
%
% Created by: Cory Fraser  - JUN 06, 2018
% Latest Edit: Cory Fraser - JUN 07, 2018
% Copyright(c) 2018 by Cory Fraser
% =========================================================================

%% Initialize Parameters
coder.extrinsic('fprintf');
coder.extrinsic('eig');

% Test if A is square
[r,c] = size(A);
if r ~= c
  error('A must be a square matrix.')
elseif (r == 1) && (A <= 0)
% A was scalar and non-positive, so just return eps (small value)
  A_hat = eps;
  return
end

% =========================================================================
%% Nearest PSDS Matrix Algorithm

% Step 1: Construct the symmetric B matrix, and skew-symmetric C matrix
B = (A + A')/2;
%C = (A - A')/2; %(Not needed for calculations)

% Step 2: Compute the symmetric polar factor of B
[U,Sigma,V] = svd(B);
H = V*Sigma*V';

% Step 3: Compute the positive approximant matrix, symmetricize
A_hat = (B+H)/2;
A_hat = (A_hat + A_hat')/2;

% Step 4: Testing for PSD, adjusting A_hat if not
p = 1;
k = 0;

while p ~= 0
  [~,p] = chol(A_hat);
```

```
57
58    %Restrict number of iterations
59    k = k + 1;
60    if k > 5
61       fprintf('Forced break after 5 iterations \n')
62       break
63    end
64
65    %If A_hat is not PSDS, perturb it a bit
66    if p ~= 0
67      min_eig = 0;
68      min_eig = min(eig(A_hat));
69
70      A_hat = A_hat + (-min_eig*k.^2 + eps(min_eig))*eye(size(A));
71    end
72  end
73
74  end
```

Appendix H

FAEKF MATLAB Code

The final appendix of the thesis houses the MATLAB codes developed for the Fuzzy Logic System used to adaptive the EKF, following the theory presented in Chapter 5. The fuzzy system parameters, membership functions, and linguistics are defined within a specific initialization script called by the formation flying simulation run script. During the simulation, the SISO fuzzy logic system is called within the EKF via an extrinsic MATLAB function, and adaptations to the noise covariance matrices are then computed using an adaptation function. The three operations described above are presented here:

`FFNAV_FAEKF_FuzzyInitialize.m`: Initialize the fuzzy system, linguistic variables, rule bases, membership functions, etc.

`FFNAV_FAEKF_Fuzzy_Type1_SISO.m`: Calculate the normalized crisp outputs of the SISO fuzzy system, given normalized crisp inputs.

`FFNAV_FAEKF_FuzzyAdapt.m`: Re-dimensionalizes the normalized crisp outputs from the fuzzy system, and adapts the EKF noise covariances.

H.1 Initialization of the FLS

```
function [FUZZY_props] = FFNAV_FAEKF_FuzzyInitialize(Plot_flag)
% FFNAV Fuzzy Adaptation Initialization ========================================
% Description: This script defines the linguistic variables, membership
% functions, universes of discourse, and implication methods for the FLS.
% Plots of the I/O membership functions can be created based on user input.
%
% The single-input, single-output FLS is used within the FAEKF. As a potential
% method for improving on the FAEKF, a multiple-input, single-output (MISO)
% FLS can also be initizlized within this script.
%
% Inputs:
%   Plot_flag    - Flag for membership function plots (1 = make plots)
%   implied_flag - Flag for implication method (1 = implied sets, 0 = aggregate)
%
% Outputs:
%   FUZZY_props - structure containing various properties of the FLS
%
% Other Function Calls:
%   None
%
% Created by: Cory Fraser  - NOV 15, 2017
% Latest Edit: Cory Fraser - JUL 04, 2018
% Copyright(c) 2018 by Cory Fraser
%===============================================================================
%% Linguistic Values

%Inputs NEG HI, NEG LO, ZERO, POS LO, POS HI
LingValsIn = {'NH', 'NL','ZERO', 'PL', 'PH'};
n_LingValsIn = length(LingValsIn);

%SISO Outputs
LingValsOut_SISO = {'NMAX', 'NMIN','ZERO', 'PMIN', 'PMAX'};
n_LingValsOut_SISO = length(LingValsOut_SISO);

%MISO Outputs ZERO, PL, MED, PH, MAX
LingValsOut_MISO = {'ZERO', 'LOW', 'MED', 'HIGH', 'MAX'};
n_LingValsOut_MISO = length(LingValsOut_MISO);

%===============================================================================
%% Rule Bases

%SISO System (1 = NEG HI, 2 = NEG LO, 3 = ZERO, 4 = POS LO, 5 = POS HI)
rules_SISO    = [5 4 3 2 1];
n_Rules_SISO  = length(rules_SISO);

%MISO System (1 = ZERO, 2 = LOW, 3 = MED, 4 = HIGH, 5 = MAX)
rules_MISO   = [ 5 4 3 4 5
                 4 3 2 3 4
                 3 2 1 2 3
                 4 3 2 3 4
                 5 4 3 4 5 ];
n_Rules_MISO  = numel(rules_MISO);

%===============================================================================
%% Centers of the Membership Functions

```

```
%         [  NH     NL   ZERO   PL    PH ]
%c_IN =   [  -1   -0.55   0    0.55   1  ];    %Spread - Outwards
 c_IN =   [  -1   -0.5    0    0.5    1  ];    %Spread - Equal
%c_IN =   [  -1   -0.25   0    0.25   1  ];    %Spread - Inwards

%           [  NH       NL   ZERO   PL     PH   ]
%c_SISO =   [  -1     -0.75    0   0.75     1   ];   %Spread - Outwards
%c_SISO =   [  -1     -0.5     0   0.5      1   ];   %Spread - Equal
c_SISO =   [  -1     -0.25    0   0.25     1   ];    %Spread - Inwards

%           [  ZERO    PL      M       PH    MAX  ]
%c_MISO =   [   0     0.5    0.75     0.9    1    ];   %Spread - Higher
 c_MISO =   [   0     0.25   0.5     0.75    1    ];   %Spread - Equal
%c_MISO =   [   0     0.1    0.25     0.5    1    ];   %Spread - Lower

%=========================================================================
%% Membership Function Definitions

n_pts      = 101;                      %Number of points
X_pts      = linspace(-1,1,n_pts);     %Universe of discourse - Inputs
Y_pts      = linspace(-1,1,n_pts);     %Universe of discourse - Outputs

sigma_IN   = 1/12;                     %Gaussian Curvature - Inputs
sigma_SISO = 1/12;                     %Gaussian Curvature - Output
sigma_MISO = 1/12;                     %Gaussian Curvature - Output

sigmoid_a   = 25;                      %Sigmoidal Curvature
sigmoid_c   = 0.75;                    %Sigmoidal Center

%=========================================================================
% Input Membership Functions
in_mf = zeros(n_LingValsIn,n_pts);

% Outer Functions - Sigmoidal for Saturation
in_mf(1,1:n_pts) = 1./(1+exp(sigmoid_a*(X_pts+sigmoid_c)));;
in_mf(n_LingValsIn,1:n_pts) = 1./(1+exp(-sigmoid_a*(X_pts-sigmoid_c)));

% Inner Functions - Gaussian
for i = 2:n_LingValsIn-1
    in_mf(i,1:n_pts) = exp(-((X_pts-c_IN(i)).^2)/(2*sigma_IN^2));
end

%=========================================================================
% SISO System Output Membership Functions
out_mf_SISO = zeros(n_LingValsOut_SISO,n_pts);

% Gaussian MSFs
for i = 1:n_LingValsOut_SISO
    out_mf_SISO(i,1:n_pts) = exp(-((Y_pts-c_SISO(i)).^2)/(2*sigma_SISO^2));
end

%=========================================================================
% MISO System Output Membership Function
out_mf_MISO = zeros(n_LingValsOut_MISO,n_pts);

% Gaussian MSFs
for i = 1:n_LingValsOut_MISO
    out_mf_MISO(i,1:n_pts) = exp(-((Y_pts-c_MISO(i)).^2)/(2*sigma_MISO^2));
end

```

```
%=========================================================================
%% Initialize Output Fuzzy Sets

% 1 = Implied Fuzzy Sets (Faster)
% 0 = Overall (Aggregated) Fuzzy Set

implied_flag = 1;

if implied_flag == 1
    implied_mf_MISO = zeros(1,n_pts);
    implied_mf_SISO = implied_mf_MISO;
    aggregate_mf = 0;
    fprintf(' Implied Fuzzy Sets \n')
else
    implied_mf_MISO = zeros(n_LingValsIn^2,n_pts);
    implied_mf_SISO = zeros(n_LingValsIn,n_pts);
    aggregate_mf = zeros(1,n_pts);
    fprintf(' Aggregated Fuzzy Sets \n')
end

%=========================================================================
%% Form a structure for Fuzzy Properties

FUZZY_props.rules_MISO    = rules_MISO;
FUZZY_props.rules_SISO    = rules_SISO;
FUZZY_props.n_pts         = n_pts;
FUZZY_props.X_pts         = X_pts;
FUZZY_props.Y_pts         = Y_pts;

% Membership Function Definitions
FUZZY_props.c_IN           = c_IN;
FUZZY_props.c_SISO         = c_SISO;
FUZZY_props.c_MISO         = c_MISO;
FUZZY_props.sigmoid_a      = sigmoid_a;
FUZZY_props.sigmoid_c      = sigmoid_c;

FUZZY_props.implied_flag   = implied_flag;
FUZZY_props.sigma_IN       = sigma_IN;
FUZZY_props.sigma_SISO     = sigma_SISO;
FUZZY_props.sigma_MISO     = sigma_MISO;

% Membership Functions
FUZZY_props.out_mf_SISO     = out_mf_SISO;
FUZZY_props.out_mf_MISO     = out_mf_MISO;
FUZZY_props.implied_mf_SISO = implied_mf_SISO;
FUZZY_props.implied_mf_MISO = implied_mf_MISO;
FUZZY_props.aggregate_mf    = aggregate_mf;

% Number of rules and lingustic variables
FUZZY_props.n_LingValsIn        = n_LingValsIn;
FUZZY_props.n_LingValsOut_SISO  = n_LingValsOut_SISO;
FUZZY_props.n_LingValsOut_MISO  = n_LingValsOut_MISO;

save('FAEKF_FuzzyProps.mat', 'FUZZY_props');

end
```

H.2 SISO Fuzzy Logic System

```
function [lambda] = FFNAV_FAEKF_Fuzzy_Type1_SISO(x, FUZZY_props)
% FFNAV Fuzzy Controller Type 1 - SISO ===================================
% Description: This function completes the fuzzification, implication, and
% defuzzification step of the FLS. Given a single normalized input, the
% function returns the corresponding normalized output of the fuzzy system.
%
% Inputs:
%   x            - Normalized input to the fuzzy system (one input)
%   FUZZY_props. - Structure containing properties of the FLS
%
% Outputs:
%   lambda       - Normalized output of the fuzzy system
%
% Other Function Calls:
%   None
%
% Created by:  Cory Fraser - NOV 15, 2017
% Latest Edit: Cory Fraser - JUL 04, 2018
% Copyright(c) 2018 by Cory Fraser
% ========================================================================
%% Initialize Parameters
rules_SISO      = FUZZY_props.rules_SISO;
n_pts           = FUZZY_props.n_pts;
X_pts           = FUZZY_props.X_pts;
Y_pts           = FUZZY_props.Y_pts;
out_mf_SISO     = FUZZY_props.out_mf_SISO;
implied_mf_SISO = FUZZY_props.implied_mf_SISO;
aggregate_mf    = FUZZY_props.aggregate_mf;
c_IN            = FUZZY_props.c_IN;
c_SISO          = FUZZY_props.c_SISO;
implied_flag    = FUZZY_props.implied_flag;
n_LingValsIn    = FUZZY_props.n_LingValsIn;
n_LingValsOut   = FUZZY_props.n_LingValsOut_SISO;
sigma_IN        = FUZZY_props.sigma_IN;
sigmoid_a       = FUZZY_props.sigmoid_a     ;
sigmoid_c       = FUZZY_props.sigmoid_c;

% ========================================================================
%% Fuzzification

u_x1 = zeros(1,n_LingValsIn);

u_x1(1) = 1./(1+exp(sigmoid_a*(x+sigmoid_c)));
for i = 2:n_LingValsIn-1
    u_x1(i) = exp(-((x-c_IN(i)).^2)/(2*sigma_IN^2))';
end
u_x1(n_LingValsIn) = 1./(1+exp(-sigmoid_a*(x-sigmoid_c)));

% ========================================================================
%% Fuzzy Inference Mechanism

if implied_flag == 1
    area = 0;
    num = 0;
    den = 0;

```

```
57      for i=1:n_LingValsIn
58
59          % Implication (MIN Operator - Mamdani)
60          implied_mf_SISO = min(  out_mf_SISO(rules_SISO(i),:),  u_x1(i)  );
61
62          % Find area under all implied fuzzy sets
63          area = sum( (1/(n_pts))*(implied_mf_SISO(1:n_pts)));
64
65          % Defuzzification (COG Numerator)
66          num = num + c_SISO(rules_SISO(i))*area;
67
68          % Defuzzification (COG Denominator)
69          den = den + area;
70      end
71
72      % Crisp Output
73      lambda = num/den;
74
75  % ========================================================================
76  else % Use Aggregated Implied Fuzzy Set
77
78      for i=1:n_LingValsOut
79
80          % Implication (MIN - Mamdani)
81          implied_mf_SISO(i,1:n_pts) = min(out_mf_SISO(rules_SISO(i),:),u_x1(i));
82      end
83
84      % Aggregation (MAX)
85      for i=1:n_pts
86          aggregate_mf(1,i) = max(implied_mf_SISO(:,i));
87      end
88
89      % Defuzzification (COA Numerator)
90      Y_aggregate_mf = Y_pts.*aggregate_mf;
91      num = sum((1/n_pts)*(Y_aggregate_mf(1:n_pts)));
92
93      % Defuzzification (COA Denominator)
94      den = sum((1/n_pts)*(aggregate_mf(1:n_pts)));
95
96      % Crisp Output
97      lambda = num/den;
98  end
99
100 end
```

H.3 Fuzzy Adaptations

```
function [Q_update, R_update, NIS,deltaDOD,DOD_true,epsilon_Q,epsilon_R] = ...
FFNAV_FAEKF_FuzzyAdapt(state_pre,Pr_theo, Pr_smooth,residuals,...
time_step, mu, Q0, R0, Q_pre, R_pre, time, FUZZY_props,Q_adapt_flag, R_adapt_flag)
% FFNAV Fuzzy Adaptations ================================================
% Description: This function adapts the process noise covariance Q and
% measurement noise covariance R matrices using a Fuzzy Logic System.
%
% Inputs:
%   state_pre    - Previous state estimate
%   Pr_theo      - Theoretical covariance of the innovations
%   Pr_smooth    - Smoothed sampled covariance of the innovations
%   residuals    - Original innovations from the current time step
%   time_step    - Time step of the simulation
%   mu           - Earth's gravitational parameter
%   Q0           - Initial process noise covariance matrix
%   R0           - Initial measurement noise covariance matrix
%   Q_pre        - Previous time step process noise covariance matrix
%   R_pre        - Previous time step measurement noise covariance matrix
%   time         - Current simulation time
%   Fuzzy_props. - Structure containing data for the Fuzzy System
%   Q_adapt_flag - Flag to adapt Q (0 = no adaptation)
%   R_adapt_flag - Flag to adapt R (0 = no adaptation)
%
% Outputs:
%   Q_update    - Updated process noise covariance matrix
%   R_update    - Updated measurement noise covariance matrix
%   NIS         - Normalized innovation squared
%   deltaDOD    - Difference in Degree of Divergence
%   DOD_true    - True degree of Divergence
%   epsilonQ    - Scaling parameter for Q matrix
%   epsilonR    - Scaling parameters for R matrix (7x7 matrix)
%
% Other Functions Called:
%   FFNAV_FAEKF_Fuzzy_Type1_SISO - Computes fuzzy output from single input
%   FFNAV_FAEKF_Fuzzy_Type1_MISO - Computes fuzzy output from multiple inputs
%   findPSDS - Finds nearest positive semidefinite symmetric matrix
%
% Created by:  Cory Fraser - JAN 13, 2018
% Latest Edit: Cory Fraser - SEP 11, 2018
% Copyright(c) 2018 by Cory Fraser
% ========================================================================
%% EKF PERFORMANCE METRICS

%Innovations Covariances
Pr_theo = diag(Pr_theo);              %Theoretical Covariance of Residuals
Pr_smooth = diag(Pr_smooth);          %Smoothed Covariance of Residuals
DOM = Pr_theo - Pr_smooth;            %Degree of Mismatch

%DOD_true = residuals'*residuals;     %Degree of Divergence - True
DOD_true = trace(Pr_smooth);          %Degree of Divergence - Smoothed True
DOD_theo = trace(Pr_theo);            %Degree of Divergence - Theoretical
deltaDOD  = DOD_true - DOD_theo;      %Difference in DOD

% ========================================================================
%% Scalar Q-Adaptations, using Degree of Divergence
if (Q_adapt_flag)
```

```
57
58      %PRISMA Gains
59      g_DOD  = -5e-3; %Input Scaling Gain
60      h_DOD  = 1e-2;  %Output Scaling Gain
61
62      %PROBA-3 Gains
63      %g_DOD  = -3e-3;  %Input Scaling Gain
64      %h_DOD  = 1e-2;   %Output Scaling Gain
65
66      %FLS Inputs
67      u_DOD       = g_DOD*deltaDOD;   % FIS Inputs
68
69      % Single-Input Single-Output Fuzzy Inference System
70      lambda_DOD  = FFNAV_FAEKF_Fuzzy_Type1_SISO(u_DOD, FUZZY_props);
71
72      epsilon_Q   = 1+lambda_DOD*h_DOD; % Q-Scale Factor (ensure > 0)
73      Q_update    = epsilon_Q*Q_pre;    % Updated Q
74
75  else
76      Q_update    = Q_pre;
77      epsilon_Q   = 1;
78  end
79
80  %=========================================================================================
81  %% Scalar R-Adaptations, using Degree of Mismatch
82
83  if (R_adapt_flag)
84
85      %PRISMA Gains
86      g_r     = 5e-2;     %Input Gain for Position
87      g_theta = 1e1;      %Input Gain for True Anomaly
88      g_v     = 1e0;      %Input Gain for Velocity
89
90      h_r     = 1e-4;     %Output Gain for Position
91      h_theta = 1e-4;     %Output Gain for True Anomaly
92      h_v     = 1e-3;     %Output Gain for Velocity
93
94      %PROBA-3 Gains
95      %{
96      g_r     = 1e-2;     %Input Gain for Position
97      g_theta = 5e2;      %Input Gain for True Anomaly
98      g_v     = 5e-1;     %Input Gain for Velocity
99
100     h_r     = 5e-5;     %Output Gain for Position
101     h_theta = 5e-5;     %Output Gain for True Anomaly
102     h_v     = 5e-4;     %Output Gain for Velocity
103     %}
104
105     G = [g_r g_r g_r g_theta g_v g_v g_v]';
106     H = [h_r h_r h_r h_theta h_v h_v h_v]';
107
108     % FLS Inputs
109     u_DOM = G.*diag(DOM);
110
111     % Single-Input Single-Output Fuzzy Inference System
112     lambda = zeros(7,1);
113     for i = 1:length(u_DOM)
114         lambda(i) = FFNAV_FAEKF_Fuzzy_Type1_SISO(u_DOM(i), FUZZY_props);
115     end
116
```

```
117     %Scale Factor and Update
118     epsilon_R = eye(7)+diag(H.*lambda); % R Scaling Factor
119     R_update = epsilon_R.*R_pre;    % Updated R (ensure > 0)
120 
121 else
122     R_update = R_pre;
123     epsilon_R = eye(7,7);
124 end
125 
126 %=========================================================================
127 %% Check Positive Semidefinite Symmetry (if desired)
128 
129 %Q_update = findPSDS(Q_update);
130 %R_update = findPSDS(R_update);
131 
132 end
```