UAV Modeling and Simulation at Normal and Abnormal Conditions

Mofetoluwa Fagbemi

Follow this and additional works at: https://researchrepository.wvu.edu/etd

Recommended Citation
Fagbemi, Mofetoluwa, "UAV Modeling and Simulation at Normal and Abnormal Conditions" (2018). Graduate Theses, Dissertations, and Problem Reports. 5564.
https://researchrepository.wvu.edu/etd/5564

This Thesis is protected by copyright and/or related rights. It has been brought to you by the The Research Repository @ WVU with permission from the rights-holder(s). You are free to use this Thesis in any way that is permitted by the copyright and related rights legislation that applies to your use. For other uses you must obtain permission from the rights-holder(s) directly, unless additional rights are indicated by a Creative Commons license in the record and/or on the work itself. This Thesis has been accepted for inclusion in WVU Graduate Theses, Dissertations, and Problem Reports collection by an authorized administrator of The Research Repository @ WVU. For more information, please contact researchrepository@mail.wvu.edu.
UAV Modeling and Simulation at Normal and Abnormal Conditions

Mofetoluwa Fagbemi

Thesis submitted to the
Benjamin M. Statler College of Engineering and Mineral Resources
at West Virginia University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Aerospace Engineering

Mario Perhinschi, Ph.D., Chair
Patrick Browning, Ph.D.
Peter Gall, Ph.D.

Department of Mechanical and Aerospace Engineering

Morgantown, West Virginia
2018

Keywords: Unmanned Aerial Vehicles; Flight Control Laws;
Abnormal Conditions; Sensor Failure Model

Copyright 2018 Mofetoluwa Fagbemi
ABSTRACT

UAV Modeling and Simulation at Normal and Abnormal Conditions

Mofetoluwa Fagbemi

The main objective of this thesis is to develop new capabilities within the West Virginia University (WVU) unmanned aerial systems (UAS) simulation environment for the design and analysis of fault tolerant control laws on small sized unmanned aerial vehicles (UAVs). An aerodynamic model for an electric powered UAV is developed using a vortex lattice method implemented within the computational design package Tornado. One-dimensional look-up tables are developed for the main stability and control derivatives, which are then used to calculate linear aerodynamic forces and moments for the nonlinear aircraft equations of motion. Flight data are used for model verification and tuning. The characteristics under normal and abnormal operation of various types of sensors typically used for UAV control are classified under nine functional categories. A general and comprehensive framework for sensor modeling is defined as a sequential alteration of the exact value of the measurand corresponding to each functional category. Simple mathematical and logical algorithms are formulated and used in this process. Each functional category is characterized by several parameters, which may be maintained constant or may vary during simulation. The user has maximum flexibility in selecting values for the parameters within and outside sensor design ranges. These values can be set to change at pre-defined moments, such that permanent and intermittent scenarios can be simulated. The aircraft and sensor models are then integrated with the WVU UAS simulation environment, which is created using MATLAB/Simulink for the computational part and FlightGear for the visualization of the aircraft and scenery. A simple user-friendly graphical interface is designed to allow for detailed simulation scenario setup.

The functionality of the developed models is illustrated through a limited analysis of the effects of sensor abnormal operation on the trajectory tracking performance of autonomous UAV. A composite metric is used for aircraft performance assessment based on both trajectory tracking errors and control activity. The targeted sensors are the gyroscopes providing angular rate measurements and the global positioning system providing position and velocity information. These sensors are instrumental in the inner and outer control loops, respectively, which characterize the typical control architecture for autonomous trajectory tracking.

Due to its generality and flexibility, the proposed sensor model provides detailed insight into the dynamic implications of sensor functionality on the performance of control algorithms. It facilitates the investigation of the synergistic interactions between sensors and control systems and may lead to improvements in both areas.
DEDICATION

To God, my family, friends, professors and all loved ones who have helped me this far. I know I could not have done it without all the love and support I freely received – even when I was undeserving of it. Thank you all very much. Words cannot describe how grateful I am.
ACKNOWLEDGEMENT

I would like to appreciate Dr. Mario Perhinschi, my advisor for all the help, support, patience and guidance shown to me during my Master’s program. I started the program as a novice, but with his help, I have learned so much in and out of my field of study. Thank you for helping me achieve this feat. I am forever grateful.

To Dr. Patrick Browning, thank you for showing me Phastball when I was looking for a new UAV to model. To Dr. Peter Gall, thanks for your help with the modeling of Phastball when I ran into problems that I did not know how to solve. Also, thank you all for agreeing to work with me and be on my committee.

To my parents and sister, thank you very much for your support with kind words, prayers and love. You always encouraged me whenever I was discouraged and helped me carry on. To my girlfriend, thank you so much for your love and patience in the trying times of my Master’s program, you were always full of love, support and care and I truly appreciate you.

To Stephane D’Urso, I am very grateful to you for providing me with more than I needed to model Phastball. You were always willing to help me regardless of how busy you were. Thank you very much.

To Ghassan and Jessica, you guys are a huge part of the successful completion of my Master’s, from showing me the ropes, to helping with my models, I say a very big thank you.

Finally, to everyone that has been an influence or shown support in one way or another, thank you. It would have been much harder to do this without you all.
# TABLE OF CONTENTS

## CHAPTER 1: INTRODUCTION
1.1 OBJECTIVES ........................................................................................................... 1
1.2 BACKGROUND .......................................................................................................... 1
1.3 THESIS LAYOUT ....................................................................................................... 2

## CHAPTER 2: LITERATURE REVIEW
2.1 GENERAL UAV MODELING .......................................................................................... 4
2.2 ABNORMAL CONDITIONS ............................................................................................. 5
2.2.1 General Abnormal Conditions ................................................................................ 5
2.2.2 Sensor Abnormal Conditions .................................................................................. 6
2.3 GPS SENSOR FAILURE ............................................................................................... 6
2.4 TORNADO VORTEX LATTICE METHOD ................................................................. 7

## CHAPTER 3: WVU UAS SIMULATION ENVIRONMENT
3.1 GENERAL ARCHITECTURE .......................................................................................... 9
3.2 GRAPHICAL USER INTERFACE (GUI) ......................................................................... 9
3.2.1 WVU UAS Simulation Environment GUI .................................................................. 10
3.2.2 Sensor Model GUI .................................................................................................. 11
3.3 MATLAB/SIMULINK INTERFACE ............................................................................. 11
3.3.1 Aircraft Model ........................................................................................................ 12
3.3.2 Sensor Failure Model ............................................................................................ 12
3.4 TRAJECTORY ALGORITHMS ...................................................................................... 13
3.4.1 Trajectory Generation ............................................................................................ 13
3.4.2 Trajectory Tracking Controllers ............................................................................. 15
3.5 PERFORMANCE METRICS ......................................................................................... 16
3.5.1 Trajectory Tracking Performance Index .................................................................... 16
3.5.2 Control Activity Performance Index ...................................................................... 17
3.5.3 Total Performance Index ....................................................................................... 18
3.6 FLIGHTGEAR ............................................................................................................. 20
3.7 UAVDASHBOARD ...................................................................................................... 20

## CHAPTER 4: AIRCRAFT MODELING
4.1 AIRCRAFT DYNAMICS ............................................................................................... 22
4.1.1 Coordinate System ............................................................................................... 22
4.1.2 Equations of Motion ............................................................................................. 23
4.2 TORNADO VORTEX LATTICE METHOD .................................................................. 24
4.2.1 Coordinate System ............................................................................................... 25
4.2.2 Phastball Geometry Modeling .............................................................................. 26
4.2.3 Phastball Modeling Results .................................................................................. 30
4.3 MATLAB/SIMULINK IMPLEMENTATION .................................................................. 34
4.4 MODEL VERIFICATION AND ANALYSIS ............................................................... 35

## CHAPTER 5: SENSOR FAILURE MODEL
5.1 DEFINITIONS, NOTATIONS, NOMENCLATURE .......................................................... 40
5.1.1 Measurand ............................................................................................................. 40
5.1.2 Sensor Output ....................................................................................................... 40
LIST OF FIGURES

Figure 3.1: WVU UAS Simulation Environment GUI ................................................................. 10
Figure 3.2: Abnormal Conditions Setup GUI ......................................................................... 10
Figure 3.3: Sensor Failure Model GUI Example ................................................................. 11
Figure 3.4: Phastball UAV Simulink Model ........................................................................ 12
Figure 3.5: Sensor Failure Simulink Model ........................................................................ 13
Figure 3.6: Phastball Square Trajectory ............................................................................. 14
Figure 3.7: Phastball Oval Trajectory ............................................................................... 14
Figure 3.8: Phastball Figure 8 Trajectory ............................................................................ 15
Figure 3.9: Phastball Climbing S-Turns Trajectory ............................................................... 15
Figure 3.10: Position PID Block Diagram ............................................................................ 16
Figure 3.11: FlightGear Interface – Phastball ................................................................. 20
Figure 3.12: WVU UAVDashboard .................................................................................... 21
Figure 3.13: San Francisco Bay Area – UAVDashboard Map [10] ........................................ 21
Figure 4.1: Aircraft Body Axes [52] .................................................................................. 22
Figure 4.2: Tornado’s Main Menu [16] .............................................................................. 25
Figure 4.3: Tornado’s Axes Definition [53] ......................................................................... 25
Figure 4.4: Phastball Wing Orthographic Projection .......................................................... 28
Figure 4.5: 3-D Wing and Partition Layout ......................................................................... 28
Figure 4.6: Phastball Wing Panels 3-D Visualization .......................................................... 29
Figure 4.7: 3-D Lift Curve Slope from Tornado ................................................................. 32
Figure 4.8: 2-D Lift Curve Slope [55] ............................................................................... 33
Figure 4.9: Corrected 3-D Lift Curve Slope ....................................................................... 33
Figure 4.10: Roll Angular Rate vs Time ............................................................................ 35
Figure 4.11: Pitch Angular Rate vs Time ........................................................................... 36
Figure 4.12: Yaw Angular Rate vs Time ........................................................................... 36
Figure 4.13: Sideslip Angle Time History ........................................................................ 37
Figure 5.1: Block Diagram of a Measurement System ..................................................... 40
Figure 5.2: Measurement System Example ...................................................................... 40
Figure 5.3: Proposed Sensor Model .................................................................................. 43
Figure 5.4: Dead Band FC Example .................................................................................. 45
Figure 5.5: Sensor Dynamics FC (Case 1) Example .............................................................. 46
Figure 5.6: Sensor Dynamics FC (Case 2) Example .............................................................. 47
Figure 5.7: Sensor Dynamics FC (Case 3) Example .............................................................. 47
Figure 5.8: Scaling FC Example ......................................................................................... 48
Figure 5.9: Bias FC Example ............................................................................................ 49
Figure 5.10: Drift FC Example .......................................................................................... 50
Figure 5.11: Saturation FC Example .................................................................................. 51
Figure 5.12: Dropout FC Example .................................................................................... 52
Figure 5.13: Noise FC Example ......................................................................................... 53
Figure 5.14: Intermittent Failure Example for a Dropout FC ........................................................ 56
Figure 5.15: Varied Magnitude Example for a Scaling FC ............................................................. 57
Figure 5.16: Intermittent and Varied Failure Example for a Bias FC ............................................ 57
Figure 7.1: Square Trajectory – Nominal Conditions .................................................................... 61
Figure 7.2: Oval Trajectory – Nominal Conditions ........................................................................ 62
Figure 7.3: Figure 8 Trajectory – Nominal Conditions ................................................................. 62
Figure 7.4: Climbing S Turns Trajectory – Nominal Conditions .................................................... 63
Figure 7.5: Angular Rates for Square Trajectory – Nominal Conditions ..................................... 63
Figure 7.6: Angular Rates for Oval Trajectory – Nominal Conditions ........................................ 64
Figure 7.7: Angular Rates for Figure 8 Trajectory – Nominal Conditions ..................................... 64
Figure 7.8: Angular Rates for Climbing S Turns Trajectory – Nominal Conditions ...................... 65
Figure 7.9: Angular Rates for a Scaling FC Affecting the Roll Rate Sensor (Oval Trajectory) ... 66
Figure 7.10: Angular Rates for a Scaling FC Affecting the Pitch Rate Sensor (Oval Trajectory) ... 67
Figure 7.12: GPS Position Dropout for 5 Seconds (Figure 8) ........................................................ 68
Figure 7.13: GPS Position Dropout for 10 Seconds (Figure 8) ....................................................... 68
Figure 7.14: GPS Position Dropout for 13.8 Seconds (Figure 8) .................................................... 69
Figure 7.15: GPS Position Dropout for 13.9 Seconds (Figure 8) .................................................... 69
Figure 7.16: GPS Velocity Dropout for 5 Seconds (Figure 8) ........................................................ 70
Figure 7.17: GPS Velocity Dropout for 10 Seconds (Figure 8) ....................................................... 70
Figure 7.18: GPS Velocity Dropout for 45.4 Seconds (Figure 8) .................................................... 71
Figure 7.19: GPS Velocity Dropout for 45.5 Seconds (Figure 8) .................................................... 71
Figure 7.20: GPS Position and Velocity Dropout for 5 Seconds (Figure 8) ..................................... 72
Figure 7.21: GPS Position and Velocity Dropout for 10 Seconds (Figure 8) ................................. 72
Figure 7.22: GPS Position and Velocity Dropout for 33.2 Seconds (Figure 8) ............................... 73
Figure 7.23: GPS Position and Velocity Dropout for 33.3 Seconds (Figure 8) ............................... 73
Figure 7.24: GPS Position Dropout for 5 Seconds (Oval) ............................................................... 74
Figure 7.25: GPS Position Dropout for 10 Seconds (Oval) ............................................................. 74
Figure 7.26: GPS Position Dropout for 13.8 Seconds (Oval) ........................................................ 75
Figure 7.27: GPS Position Dropout for 13.9 Seconds (Oval) ........................................................ 75
Figure 7.28: GPS Velocity Dropout for 5 Seconds (Oval) ............................................................... 76
Figure 7.29: GPS Velocity Dropout for 10 Seconds (Oval) ............................................................. 76
Figure 7.30: GPS Velocity Dropout for 28.3 Seconds (Oval) .......................................................... 77
Figure 7.31: GPS Velocity Dropout for 28.4 Seconds (Oval) .......................................................... 77
Figure 7.32: GPS Position and Velocity Dropout for 5 Seconds (Oval) .......................................... 78
Figure 7.33: GPS Position and Velocity Dropout for 10 Seconds (Oval) ......................................... 78
Figure 7.34: GPS Position and Velocity Dropout for 34.8 Seconds (Oval) .................................... 79
Figure 7.35: GPS Position and Velocity Dropout for 34.9 Seconds (Oval) .................................... 79
Figure 7.36: Square Trajectory PI – Nominal Conditions ............................................................ 80
Figure 7.37: Oval Trajectory PI – Nominal Conditions ................................................................. 81
Figure 7.38: Figure 8 Trajectory PI – Nominal Conditions ............................................................ 81
Figure 7.39: Climbing S Turns PI – Nominal Conditions ................................................................. 82
Figure 7.40: Trajectory Tracking Index for Square Trajectory (Roll Rate Sensor Failure) .......... 83
Figure 7.41: Control Activity Index for Square Trajectory (Roll Rate Sensor Failure) ................. 84
Figure 7.42: Trajectory Tracking Index for Square Trajectory (Pitch Rate Sensor Failure) ........... 84
Figure 7.43: Control Activity Index for Square Trajectory (Pitch Rate Sensor Failure) ............... 84
Figure 7.44: Trajectory Tracking Index for Oval Trajectory (Roll Rate Sensor Failure) ............... 85
Figure 7.45: Control Activity Index for Oval Trajectory (Roll Rate Sensor Failure) .................... 85
Figure 7.46: Trajectory Tracking Index for Oval Trajectory (Pitch Rate Sensor Failure) ............... 86
Figure 7.47: Control Activity Index for Oval Trajectory (Pitch Rate Sensor Failure) .................. 86
Figure 7.48: Trajectory Tracking Index for Figure 8 Trajectory (Roll Rate Sensor Failure) ......... 87
Figure 7.49: Control Activity Index for Figure 8 Trajectory (Roll Rate Sensor Failure) ................. 87
Figure 7.50: Trajectory Tracking Index for Figure 8 Trajectory (Pitch Rate Sensor Failure) ........ 88
Figure 7.51: Control Activity Index for Figure 8 Trajectory (Pitch Rate Sensor Failure) ............... 88
Figure 7.52: Trajectory Tracking Index for Climbing S Trajectory (Roll Rate Sensor Failure) ....... 89
Figure 7.53: Control Activity Index for Climbing S Trajectory (Roll Rate Sensor Failure) ............ 89
Figure 7.54: Trajectory Tracking Index for Climbing S Trajectory (Pitch Rate Sensor Failure) ..... 90
Figure 7.55: Control Activity Index for Climbing S Trajectory (Pitch Rate Sensor Failure) .......... 90
Figure 7.56: Oval Trajectory for a High Severity Drift AC affecting the Pitch Rate Gyro Sensor .. 91
Figure 7.57: Trajectory Tracking Index for Increasing Drift Value (Oval Trajectory) ..................... 92
Figure 7.58: Control Surface Deflections for High Severity Time Delay AC affecting Roll Rate Sensor (Figure 8 Trajectory) ................................................................................................................. 93
Figure 7.59: Control Surface Deflections for High Severity Noise AC affecting Roll Rate Sensor (Figure 8 Trajectory) ................................................................................................................. 93
Figure 7.60: Trajectory Tracking PI (Figure 8 Position Dropout) .................................................. 94
Figure 7.61: Control Activity PI (Figure 8 Position Dropout) ....................................................... 95
Figure 7.62: Throttle Saturation PI (Figure 8 Position Dropout) ..................................................... 95
Figure 7.63: Trajectory Tracking PI (Figure 8 Velocity Dropout) ................................................... 96
Figure 7.64: Control Activity PI (Figure 8 Velocity Dropout) ....................................................... 96
Figure 7.65: Throttle Saturation PI (Figure 8 Velocity Dropout) ................................................... 96
Figure 7.66: Trajectory Tracking PI (Figure 8 Position and Velocity Dropout) ............................. 97
Figure 7.67: Control Activity PI (Figure 8 Position and Velocity Dropout) ................................... 98
Figure 7.68: Throttle Saturation PI (Figure 8 Position and Velocity Dropout) .................. 98
Figure 7.69: Trajectory Tracking PI (Oval Position Dropout) ........................................................ 99
Figure 7.70: Control Activity PI (Oval Position Dropout) ............................................................. 99
Figure 7.71: Throttle Saturation PI (Oval Position Dropout) .......................................................... 100
Figure 7.72: Trajectory Tracking PI (Oval Velocity Dropout) ....................................................... 100
Figure 7.73: Control Activity PI (Oval Velocity Dropout) ............................................................. 101
Figure 7.74: Throttle Saturation PI (Oval Velocity Dropout) ....................................................... 101
Figure 7.75: Trajectory Tracking PI (Oval Position and Velocity Dropout) ................................ 102
Figure 7.76: Control Activity PI (Oval Position and Velocity Dropout) ...................................... 102
Figure 7.77: Throttle Saturation PI (Oval Position and Velocity Dropout) .................................. 103
LIST OF TABLES

Table 3.1: Trajectory Tracking and Control Activity Weight Vectors ............................................ 19
Table 3.2: PI Cutoff Values ............................................................................................................ 20
Table 4.1: Phastball Geometry Data for Tornado ......................................................................... 27
Table 4.2: Phastball Flight Condition Setup in Tornado ............................................................... 29
Table 4.3: Phastball Force Results from Tornado ......................................................................... 30
Table 4.4: Angular Moments Coefficients .................................................................................... 30
Table 4.5: Force Coefficients for all Wing Surfaces ..................................................................... 30
Table 4.6: α and β Stability Derivatives ...................................................................................... 31
Table 4.7: p, q and r Stability Derivatives ..................................................................................... 31
Table 4.8: Control Surface Derivatives.......................................................................................... 31
Table 4.9: Phastball Characteristics [53] ....................................................................................... 34
Table 6.1: Angular Rate Limits for All Trajectories [deg/s] ........................................................... 58
Table 6.2: Sensor Abnormal Conditions Parameters .................................................................... 59
Table 6.3: Saturation Parameters for Square Trajectory .............................................................. 59
Table 6.4: Saturation Parameters for Oval Trajectory ................................................................. 59
Table 6.5: Saturation Parameters for Figure 8 Trajectory ............................................................ 59
Table 6.6: Saturation Parameters for Climbing S Turns Trajectory ............................................ 59
Table 7.1: Sensor Failure Sensitivity Analysis ............................................................................. 104
# LIST OF SYMBOLS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>English</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( b )</td>
<td>Wingspan</td>
<td>m</td>
</tr>
<tr>
<td>( \bar{c} )</td>
<td>Mean Aerodynamic Chord</td>
<td>m</td>
</tr>
<tr>
<td>( C_D )</td>
<td>Drag Coefficient</td>
<td>-</td>
</tr>
<tr>
<td>( C_L )</td>
<td>3-D Lift Coefficient</td>
<td>-</td>
</tr>
<tr>
<td>( C_I )</td>
<td>Rolling Moment Coefficient</td>
<td>-</td>
</tr>
<tr>
<td>( C_{L\infty} )</td>
<td>2-D Lift Coefficient</td>
<td>-</td>
</tr>
<tr>
<td>( C_m )</td>
<td>Pitching Moment Coefficient</td>
<td>-</td>
</tr>
<tr>
<td>( C_n )</td>
<td>Yawing Moment Coefficient</td>
<td>-</td>
</tr>
<tr>
<td>( C_X )</td>
<td>X Force Coefficient</td>
<td>-</td>
</tr>
<tr>
<td>( C_Y )</td>
<td>Y Force Coefficient</td>
<td>-</td>
</tr>
<tr>
<td>( C_Z )</td>
<td>Z Force Coefficient</td>
<td>-</td>
</tr>
<tr>
<td>( g )</td>
<td>Acceleration Due to Gravity</td>
<td>m/s²</td>
</tr>
<tr>
<td>( h )</td>
<td>Cruise Altitude</td>
<td>m</td>
</tr>
<tr>
<td>( m )</td>
<td>Mass</td>
<td>kg</td>
</tr>
<tr>
<td>( M_{Tx} )</td>
<td>Thrust Moment X Component</td>
<td>Nm</td>
</tr>
<tr>
<td>( M_{Ty} )</td>
<td>Thrust Moment Y Component</td>
<td>Nm</td>
</tr>
<tr>
<td>( M_{Tz} )</td>
<td>Thrust Moment Z Component</td>
<td>Nm</td>
</tr>
<tr>
<td>( p )</td>
<td>Roll Angular Velocity</td>
<td>rad/s</td>
</tr>
<tr>
<td>( q )</td>
<td>Pitch Angular Velocity</td>
<td>rad/s</td>
</tr>
<tr>
<td>( \dot{q} )</td>
<td>Dynamic Pressure</td>
<td>Pa</td>
</tr>
<tr>
<td>( r )</td>
<td>Yaw Angular Velocity</td>
<td>rad/s</td>
</tr>
<tr>
<td>( Re )</td>
<td>Reynolds Number</td>
<td>-</td>
</tr>
<tr>
<td>( S_{ref} )</td>
<td>Reference Area</td>
<td>m²</td>
</tr>
<tr>
<td>( T )</td>
<td>Thrust</td>
<td>N</td>
</tr>
<tr>
<td>( T_X )</td>
<td>Thrust Force X Component</td>
<td>N</td>
</tr>
<tr>
<td>( T_Y )</td>
<td>Thrust Force Y Component</td>
<td>N</td>
</tr>
<tr>
<td>( T_Z )</td>
<td>Thrust Force Z Component</td>
<td>N</td>
</tr>
<tr>
<td>( V )</td>
<td>Velocity</td>
<td>m/s</td>
</tr>
<tr>
<td>( x_a )</td>
<td>Actual Aircraft x Position</td>
<td>m</td>
</tr>
<tr>
<td>( X_B )</td>
<td>Longitudinal Axis</td>
<td>-</td>
</tr>
<tr>
<td>( x_c )</td>
<td>Commanded Trajectory x Coordinates</td>
<td>m</td>
</tr>
<tr>
<td>( y_a )</td>
<td>Actual Aircraft y Position</td>
<td>m</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>------------------------------------------------------------------</td>
<td>-----</td>
</tr>
<tr>
<td>$y_B$</td>
<td>Lateral Axis</td>
<td>-</td>
</tr>
<tr>
<td>$y_c$</td>
<td>Commanded Trajectory y Coordinates</td>
<td>m</td>
</tr>
<tr>
<td>$z_a$</td>
<td>Actual Aircraft z Position</td>
<td>m</td>
</tr>
<tr>
<td>$Z_B$</td>
<td>Vertical Axis</td>
<td>-</td>
</tr>
<tr>
<td>$z_c$</td>
<td>Commanded Trajectory z Coordinates</td>
<td>m</td>
</tr>
</tbody>
</table>

### Greek

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>Angle of Attack</td>
<td>deg</td>
</tr>
<tr>
<td>$\dot{\alpha}$</td>
<td>Rate of Change of $\alpha$</td>
<td>deg/s</td>
</tr>
<tr>
<td>$\beta$</td>
<td>Sideslip Angle</td>
<td>deg</td>
</tr>
<tr>
<td>$\dot{\beta}$</td>
<td>Rate of Change of $\beta$</td>
<td>deg/s</td>
</tr>
<tr>
<td>$\theta$</td>
<td>Pitch Angle</td>
<td>deg</td>
</tr>
<tr>
<td>$\mu$</td>
<td>Dynamic Viscosity</td>
<td>N·s/m²</td>
</tr>
<tr>
<td>$\rho$</td>
<td>Density</td>
<td>kg/m³</td>
</tr>
<tr>
<td>$\phi$</td>
<td>Roll Angle</td>
<td>deg</td>
</tr>
<tr>
<td>$\psi$</td>
<td>Yaw Angle</td>
<td>deg</td>
</tr>
</tbody>
</table>

### Subscripts

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$i$</td>
<td>index</td>
<td>-</td>
</tr>
<tr>
<td>$a$</td>
<td>aileron</td>
<td>-</td>
</tr>
<tr>
<td>$e$</td>
<td>elevator</td>
<td>-</td>
</tr>
<tr>
<td>$r$</td>
<td>rudder</td>
<td>-</td>
</tr>
<tr>
<td>$t$</td>
<td>throttle</td>
<td>-</td>
</tr>
<tr>
<td>$B$</td>
<td>body axes</td>
<td>-</td>
</tr>
<tr>
<td>$E$</td>
<td>earth axes</td>
<td>-</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>---------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>2-D</td>
<td>2 Dimensional</td>
<td></td>
</tr>
<tr>
<td>3-D</td>
<td>3 Dimensional</td>
<td></td>
</tr>
<tr>
<td>AC</td>
<td>Abnormal Condition</td>
<td></td>
</tr>
<tr>
<td>AIS</td>
<td>Artificial Immune System</td>
<td></td>
</tr>
<tr>
<td>AVL</td>
<td>Athena Vortex Lattice</td>
<td></td>
</tr>
<tr>
<td>CA</td>
<td>Control Activity</td>
<td></td>
</tr>
<tr>
<td>CAI</td>
<td>Control Activity Index</td>
<td></td>
</tr>
<tr>
<td>CAS</td>
<td>Calibrated Air Speed</td>
<td></td>
</tr>
<tr>
<td>CFD</td>
<td>Computational Fluid Dynamics</td>
<td></td>
</tr>
<tr>
<td>CG</td>
<td>Center of Gravity</td>
<td></td>
</tr>
<tr>
<td>CS</td>
<td>Coordinate System</td>
<td></td>
</tr>
<tr>
<td>EAS</td>
<td>Equivalent Air Speed</td>
<td></td>
</tr>
<tr>
<td>FC</td>
<td>Functional Category</td>
<td></td>
</tr>
<tr>
<td>FDI</td>
<td>Fault Detection and Identification</td>
<td></td>
</tr>
<tr>
<td>FOS</td>
<td>First Order System</td>
<td></td>
</tr>
<tr>
<td>GPS</td>
<td>Global Positioning System</td>
<td></td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
<td></td>
</tr>
<tr>
<td>HUD</td>
<td>Head Up Display</td>
<td></td>
</tr>
<tr>
<td>MAC</td>
<td>Mean Aerodynamic Chord</td>
<td></td>
</tr>
<tr>
<td>MC</td>
<td>Modified Condition</td>
<td></td>
</tr>
<tr>
<td>NACA</td>
<td>National Advisory Committee for Aeronautics</td>
<td></td>
</tr>
<tr>
<td>NLDI</td>
<td>Nonlinear Dynamic Inversion</td>
<td></td>
</tr>
<tr>
<td>PI</td>
<td>Performance Index</td>
<td></td>
</tr>
<tr>
<td>PID</td>
<td>Proportional Integral Derivative</td>
<td></td>
</tr>
<tr>
<td>PPID</td>
<td>Position Proportional Integral Derivative</td>
<td></td>
</tr>
<tr>
<td>PV</td>
<td>Performance Vector</td>
<td></td>
</tr>
<tr>
<td>RMS</td>
<td>Root Mean Square</td>
<td></td>
</tr>
<tr>
<td>SE</td>
<td>Simulation Environment</td>
<td></td>
</tr>
<tr>
<td>SFDI</td>
<td>Sensor Fault Detection and Identification</td>
<td></td>
</tr>
<tr>
<td>SI</td>
<td>International Systems</td>
<td></td>
</tr>
<tr>
<td>SOS</td>
<td>Second Order System</td>
<td></td>
</tr>
<tr>
<td>TAS</td>
<td>True Air Speed</td>
<td></td>
</tr>
<tr>
<td>TPR</td>
<td>Transient Peak Ratio</td>
<td></td>
</tr>
</tbody>
</table>

xiii
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>TT</td>
<td>Trajectory Tracking</td>
</tr>
<tr>
<td>TTI</td>
<td>Trajectory Tracking Index</td>
</tr>
<tr>
<td>UAS</td>
<td>Unmanned Aerial System</td>
</tr>
<tr>
<td>UAV</td>
<td>Unmanned Aerial Vehicle</td>
</tr>
<tr>
<td>VLM</td>
<td>Vortex Lattice Method</td>
</tr>
<tr>
<td>WVU</td>
<td>West Virginia University</td>
</tr>
</tbody>
</table>
Chapter 1: INTRODUCTION

1.1 Objectives

The main purpose of this thesis is to develop additional simulation tools within the West Virginia University (WVU) unmanned aerial system (UAS) simulation environment [1] that are expected to support the design and advancement of autonomous flight control laws capable of handling subsystem normal and abnormal operational conditions. The implementation of advanced fault tolerant control laws is key to ensuring unmanned aerial vehicle (UAV) safety and preventing aviation incidents and accidents [2], [3]. A general and comprehensive sensor model at normal and abnormal conditions and a new aircraft model are developed using different modeling tools. This thesis is aimed at providing UAV designers with valuable additional design tools that would aid the detection, identification and evaluation of faults affecting UAV sensors. These tools will also help with the development of control laws capable of accommodating these faults and improving the safety and usability of the UAV. The WVU UAS simulation environment is used to test several flight scenarios under nominal and abnormal conditions including failures on the sensor subsystem of the aircraft. The functionality of the developed models is illustrated through an analysis of the effects of sensor abnormal operation on the trajectory tracking and control activity performance of autonomous UAV.

1.2 Background

The growth and relevance of UAS has increased significantly over the past few years and is expected to increase a lot more with the FAA estimating a growth in the purchases of drones for hobbyists from 1.9 million to 4.3 million between years 2016 and 2020 [4]. UAS are used for an ever expanding variety of both military and civilian purposes. The military uses them for missions such as intelligence, security, search and rescue, monitoring and even combat [5]. Civilian applications include industrial, agricultural and environmental monitoring, public safety, image acquisition, mapping, leisure and sport. Companies like Amazon and Google plan to start using UAS to deliver commercial packages in the foreseeable future [6].

In order for UAS to carry out required tasks effectively, it is necessary for all subsystems to perform normally, as expected by design. One very important subsystem on a UAS is the sensor subsystem as all the information regarding the UAS speed, location, orientation are provided by unique high quality sensors that need to be well mounted, calibrated and tested in order to get good and useful outputs. These outputs are used to implement autonomous trajectory tracking controllers that allow the UAS fly without any pilot input for a given trajectory or a set number of trajectories. They can also be used when controlling the UAV manually, where the sensors will provide feedback to the pilot regarding the state of the UAV.
Whenever a failure occurs on the sensor subsystem, the failed sensors produce incorrect information, which is detrimental to either the implemented autonomous controllers on the UAS or the pilot who is flying the system manually. The failures can lead to unwanted circumstances such as failed missions and accidents; therefore, the necessity for pilots to be able to understand and tolerate failures on the sensor subsystem as well as the ability of the autonomous control laws to accommodate the aforementioned failures becomes critical.

There are many UAV simulation tools available including Simdrone [7], Simlat [8], FlightGear [9] and many others. Simdrone is a real-time UAV flight simulator that helps reduce the expensive cost and high risks of UAV trainings and operations. Simdrone implements an autopilot (weControl) [7]. Simlat is similar to Simdrone but it implements control station as well as exploitation software in conjunction with its autopilot implementation [8]. FlightGear is an open-source flight simulation software that supports several platforms such as Windows and Linux [9]. It features well-detailed graphics and has many available aircraft models but little capability to implement flight control system design [10]. FlightGear also allows users to create and add their own aircraft model making it a very useful visual tool for flight simulations.

With the availability of the simulation tools mentioned above, the ability to implement failures on the sensor subsystem of the UAS is not possible, thereby underlining the need for a comprehensive sensor failure model for the very purpose of understanding and tolerating sensor failures on UAS.

1.3 Thesis Layout

This thesis is organized as follows. Chapter 1: highlights the objectives, motivation towards the formulation of the thesis and its layout. Chapter 2: presents the literature review, which includes the concepts of modeling UAVs, abnormal conditions occurring on UAVs in general as well as abnormal conditions occurring on the sensor subsystem. This chapter also discusses failures on the global positioning system (GPS) sensor of UAVs. Finally, it addresses the vortex lattice method (VLM) to be used to acquire the control and stability derivatives of the modeled UAV. Chapter 3: describes the WVU UAS simulation environment, its general architecture, algorithms for trajectory tracking and performance evaluation, how the graphical user interface (GUI) is connected to the developed Simulink aircraft and sensor models, as well as the visuals provided by FlightGear and the UAV dashboard. Chapter 4: explains how the modeling of the UAV was carried out, it expands on the VLM method used and how the aircraft geometry was modeled with this tool. It shows how the model was implemented in Simulink and how the model was verified and analyzed to evaluate its usefulness. Chapter 5: illustrates the proposed sensor failure model and how it works, the types of failures implementable and its MATLAB and Simulink implementation. Chapter 6: presents all the abnormal conditions simulated. Chapter 7: shows all the results obtained from the performed simulations and finally, Chapter 8:
discusses all conclusions drawn from the modeling, simulations and acquired results within this thesis. It also proposes future recommendations for improvements.
CHAPTER 2: LITERATURE REVIEW

The modeling of an aircraft and its sensor subsystem requires a thorough understanding of aircraft dynamics and sensor functionalities and how all the different parameters involved affect the response of the aircraft. This chapter briefly presents the general state-of-the-art to the modeling of UAVs, their abnormal operational conditions in general and with respect to the sensor subsystem (including GPS) and the VLM tool used in this thesis.

2.1 General UAV Modeling

UAVs can be modeled using different techniques including wind tunnel testing, state space identification, computational fluid dynamics (CFD) and vortex lattice methods. These different techniques amongst many others need to provide accurate results as they determine the dynamics of the UAV’s motion.

Wind tunnel testing is a widely used technique by aerodynamicists to test aircraft and engine components of proposed models. For this to be performed, the proposed model is carefully positioned in the wind tunnel test section and air is flown past it. The aerodynamic forces and moments can be determined by using various instruments or techniques [11]. One major drawback to wind tunnel testing is the cost, as it can be very expensive to set up and maintain a wind tunnel. Another drawback is the large area needed for the wind tunnel and the adjacent complex facilities.

State space identification is a mapping method that solves modeled parameters using specified equations. Data used to perform this technique has to be previously provided from an available source such as flight tests [12]. The technique is used to identify the coefficients of the state matrix for a set of state variables. Before state space identification can be performed, the equations to be modeled will have to be defined.

CFD is a tool that provides a satisfactory aerodynamic model in the linear region of the aircraft aerodynamic forces and moments, without considering the effects that spaces and gaps will have on the control surfaces, wings, horizontal and vertical tails of the aircraft. Also, CFD codes have difficulty predicting the drag force or coefficient acting on the aircraft [13] especially potential flow CFD codes which do not predict viscous drag effects at all. Most CFD codes only account for the drag force generated by the pressure distribution about all wing surfaces and the fuselage neglecting other contributions to the drag such as the skin friction drag.

VLM is a method that provides a linear approximation of the aerodynamic derivatives of the aircraft with respect to its geometry. This method is more cost-efficient and easier to implement compared to the other methods mentioned above. Some examples of available VLM software packages are DATCOM [14], Athena Vortex Lattice (AVL) [15] and
Tornado [16]. DATCOM implements a semi-empirical method to predict the aerodynamic characteristics of the aircraft being modeled. One major drawback to using DATCOM is its applicability for full-scale aircraft. Another is that it does not provide control derivatives. AVL and Tornado are more suitable for small sized UAVs. AVL and Tornado are based off of linear CFD codes and just like CFD, are not the most suitable tools for finding the drag force or coefficient acting on the aircraft [13]. AVL is a very simple tool that uses the approximate geometry of an aircraft to generate reasonable results. Its input is a text file which defines the geometry and Mach number and it outputs the stability and control derivatives. AVL and DATCOM were used to model some of the aircraft in the WVU UAS simulation environment and more details can be found in [10].

2.2 Abnormal Conditions

This section discusses abnormal conditions that occur on aircraft subsystems, such as actuators and structural components. It then addresses abnormal conditions due to failures of the sensor subsystems with some references made to small-sized UAVs.

2.2.1 General Abnormal Conditions

Failures on any of the aircraft subsystems occur primarily due to abnormal conditions affecting the subsystem and its corresponding components. As a consequence, it becomes more difficult for pilots or automatic flight control laws to perform required maneuvers, which eventually may lead to aborted missions, aviation incidents and accidents. There have been several technologies developed over the years in order to prevent loss of control and improve post-failure flight safety of manned aircraft [17], [18]. Most of these technologies include implementing high-performance real-time fault detection and identification (FDI) schemes [19]–[21] for the main aircraft subsystems in order to increase pilot situational awareness and/or support automatic control laws recovery and successful mission completion. However, there are not so many available significant research efforts directly addressing these abnormal conditions and failures on UAVs. Artificial immune system (AIS) [22] is one recent technique that has been used to combat abnormal conditions on UAVs. It works comparable to the biological immune system in accordance with the self–nonself discrimination principle. Its FDI scheme uses a hierarchical multiself strategy. One advantage of this strategy is its ability to significantly improve the failure detection while reducing the necessary computational requirements. Abnormal conditions occurring on the actuator subsystem of a UAV has been previously modeled for failures occurring on all control surfaces [21] affecting the roll, pitch and yawing moments accordingly for actuator blockages and a missing fragment of the control surface. AIS has been used successfully for FDI of this class of abnormal conditions for UAVs [23], [24]. The methodology has also been demonstrated with actual flight data [25]. Abnormal conditions on the structural and propulsion subsystems of an aircraft can also lead to mishaps. Structural failures signify a missing or damaged part of an aircraft structure. It affects the stability of the aircraft and can affect its controllability too, depending on the failed
structure. Most simulation environments model this type of failure by mathematically altering the stability or control derivatives corresponding to the failed structural component [26]. For example, ice formation on a wing’s leading edge corresponds to an alteration of the wing mass and aerodynamic stability derivatives, which can be modified mathematically to implement the failure [27]. Aircraft health management simulation environments [28] have also been developed to increase knowledge on detecting and accommodating abnormal conditions.

2.2.2 Sensor Abnormal Conditions

Sensors play a very significant role in the operations of UAS. The degree of autonomy of a system is dependent on the number of sensors used on the system [29]. Unlike the control surfaces, multiple sensors can be mounted on an aircraft in order to mitigate failures through redundancy [21] with the use of a voting logic [30] but this technique is not completely fool-proof as it could fail in instances where multiple sensors in the voting scheme are producing similar abnormal values. Unfortunately, due to cost, weight and space constraints, it is not always feasible to mount multiple sensors on a small UAV, with most having one sensor for each task or measurement, making the pilot or automatic flight control laws solely depend on the available information from the single sensor regardless of its accuracy. Training pilots thoroughly to combat all abnormal conditions occurring due to sensor failures is not realistic due to the complexity and ambiguity of the failures.

The use of FDI schemes is one of the techniques that can help improve the reaction time of the pilot and enhance the probability of success for the aircraft post-failure recovery [19]. An example of this scheme implements a main set of neural networks and another set of three decentralized neural networks. For the sensor FDI (SFDI) scheme, one major assumption is that SFDI can be performed by monitoring the aircraft response [19]. This technique can be applied to other sensors on the aircraft as well.

Some abnormal conditions that have previously been modeled on the sensor subsystem is the presence of a bias and drift affecting the angular rates of the aircraft. The bias was modeled as an instantaneous step and the drifting bias as a transient. Different biases and different transients may be possible [31], but abnormal conditions on sensors are not limited to biases or drifts. Other modeled sensor abnormal conditions include scaling (gain failure), noise and loss of signals [29]. These abnormal conditions can be modeled mathematically and are explained in more detail in Chapter 5:

2.3 GPS Sensor Failure

The GPS is a fully functional Global Navigation Satellite System that is used to determine the current location, orientation and velocity of a system. This is performed by communicating with GPS satellites that orbit the earth and transmit signals continuously to the GPS receiver. Since the launch of the first ever experimental satellite in the year 1978,
the GPS has grown to be an integral system necessary for tasks like navigation and cartography [32].

For a UAV to carry out its given tasks successfully, it requires a continuous informational update regarding its position and velocity. A UAV flying manually or automatically depends on this information for effective navigation and control. This information is provided by a GPS sensor. A failure occurring on a GPS sensor can be catastrophic to the performance of autonomous UAV trajectory tracking. Some GPS system errors include satellite clock errors, ephemeris errors, multipath effect and receiver noise. Some other error sources which affect the accuracy of GPS include delays which occur on the hardware both on the satellite and user level and saganc interference which represents the error due to the rotation of the earth during signal transmission. The accuracy of a GPS is determined by checking if the signal in space user range error are within the standardized limits. GPS failure has been previously modeled using mathematical equations based off of data collected from actual GPS measurements using MATLAB/Simulink for its implementation [33], [34]. The sensor model proposed in this thesis can also be implemented on GPS outputs.

GPS errors have also been modeled in the frequency domain [35] to improve accuracy with the use of a shaping filter [36] and reduce the root mean square (RMS) position and velocity errors. The shaping filter is used to model a white noise that correlates noise of the GPS measurement with its output being the system’s measurement error. Errors can also be modeled in the time domain using a Kalman filter. In [35], the RMS position and velocity errors were found to be less than 0.52 m for the frequency domain modeling as compared to the time domain.

2.4 Tornado Vortex Lattice Method

Tornado is a free educational tool developed for linear aerodynamic wing design applications in conceptual aircraft design. It is a 3-D VLM that is based off of linear CFD codes [13]. It models all lifting surfaces as thin plates and is good at solving for most aerodynamic derivatives for a vast majority of aircraft geometries. It is very user-friendly and has a neat GUI interfaced with MATLAB for ease of use. Tornado provides results in a timely manner and also provides well-formatted plots as outputs as well as data files with all the stored data from every simulated run. Tornado can be run in either a text based interface or batch mode [37]. Tornado is a very useful tool for estimating aerodynamic forces and moments on small UAVs and has been used on several research efforts successfully to estimate aerodynamic forces and moments [38]–[41].

In [38], Tornado was used to solve for the aerodynamic properties of winglet geometry to analyze the effects of winglets on small sized UAVs to aid optimizing winglet design on UAV platforms. Tornado was used in [39] to calculate aerodynamic derivatives that were validated using experimental flight test data for Vector-P, the modeled UAV. In
[40], an analysis of lift induced drag was performed using Tornado for a P51D Mustang. Tornado can also be used to perform wing loading analysis [41], in order to determine the maximum loads on the wing for specific operating conditions.
CHAPTER 3: WVU UAS SIMULATION ENVIRONMENT

In this chapter, the WVU UAS simulation environment will be discussed in depth along with its implemented algorithms for creating commanded trajectories, tracking them and evaluating UAV performance. The GUI framework in the environment would also be discussed as well as the interfacing between MATLAB and Simulink.

3.1 General Architecture

The WVU UAS simulation environment is a tool developed in-house used to simulate autonomous flight and fault-tolerant control laws for a generated or pre-recorded trajectory [10], [26]. It can also simulate manual flight providing adequate feedback to a pilot where necessary. It takes either the pilot’s input commands or generated trajectory commands as its input. The typical 4-channel control commands are the aileron, elevator, rudder and throttle signals. These inputs are then converted to forces and moments in the Simulink model for whichever aircraft is being simulated. As shown in Figure 3.1, there are currently six aircraft models in the simulation environment, with Phastball being the newest addition as a part of this thesis effort.

A commanded trajectory generation module includes a variety of different approaches, such as Voronoi [42], clothoid and Dubins [43], [44]. Both classical fixed parameter trajectory tracking algorithms as well as adaptive control laws are implemented [24], [45] covering distinct classes of approaches such as position proportional, integral and derivative (PID), nonlinear dynamic inversion (NLDI) and immunity based adaptive methods [46].

A detailed model of atmospheric conditions is available including constant wind in any direction, turbulence, wind gusts and wind shear [47], [48]. Visual cues are provided by FlightGear [9] and the UAVDashboard, a customized visualization tool for mission scenario definition and trajectory tracking monitoring [46]. A GPS model is also implemented which provides real-time aircraft position and velocity under normal and abnormal operational scenarios [34].

This thesis was made possible because of the availability of the WVU UAS simulation environment which allows users to perform various simulations including normal and abnormal conditions with failures occurring on different aircraft subsystems at set times.

3.2 Graphical User Interface (GUI)

There are many GUIs developed within the WVU UAS simulation environment to facilitate ease of use. They are very user-friendly and easy to navigate. Some of the tasks of the GUIs include selecting the aircraft to be simulated, defining simulation conditions or parameters and showing plots. Chapters 3.2.1 and 3.2.2 discuss the WVU UAS and sensor model graphical interfaces, respectively.
3.2.1 WVU UAS Simulation Environment GUI

The WVU UAS simulation environment GUI is the first level a user has to go through before running any simulations. The desired vehicle, map, trajectory planning and controller options are defined in the main WVU UAS simulation environment GUI, which is shown in Figure 3.1. The next interfaces are used to define abnormal conditions if desired, if not, the simulation can be launched. The failure conditions available are control surface and sensor failures as shown in Figure 3.2.
3.2.2 Sensor Model GUI
The developed sensor model interface is used to setup an abnormal condition scenario on the aircraft’s sensor subsystem. It allows the user select the failure type from one of the nine functional categories available, the affected sensor and to define the parameters respective to the failure. Depending on the failure type, the occurrence of the failure can be set to be permanent or intermittent and the magnitude can be constant or variable. Figure 3.3 shows an example of the sensor failure GUI for a bias affecting the pitch rate. The magnitude of the bias is 0.45 rad/s and it will occur 20 seconds after the simulation has begun. Finally, the failure is set to be permanent with a constant magnitude.

![Phastball Sensor Failure GUI](image)

Figure 3.3: Sensor Failure Model GUI Example

3.3 MATLAB/Simulink Interface
MATLAB is a programming platform developed by MathWorks to be used specifically by engineers and scientists [49]. It runs on the MATLAB language which is matrix-based and allows the implementation of computational mathematics. Its uses include data analysis, algorithm development and modeling [49]. Simulink is also a tool by MathWorks used to design and simulate systems [50]. In the WVU UAS simulation environment, MATLAB and Simulink work together; MATLAB is used to define all variables used in the simulation environment and Simulink implements a mathematical model of all aircraft systems and the conventional and adaptive controllers used to fly the aircraft autonomously.
3.3.1 Aircraft Model
The Simulink model for the Phastball UAV is shown in Figure 3.4. The model has several blocks that serve different purposes. The manual flight, follow leader and follow trajectory blocks are the inputs to the UAV. These are selected depending on the simulation to be run by the user. The conventional and adaptive controllers’ blocks contain the trajectory tracking algorithms developed within the simulation environment. More information on their development and implementation can be found in [46]. The Phastball UAV block contains all the aircraft dynamics of the Phastball UAV. The amount of turbulence simulated is set using the turbulence block. There are different turbulence severities available. The turbulence model is developed using the Dryden model [10]. In Figure 3.4, there is no turbulence simulated. The wind scenario block is a recent addition to the WVU UAS simulation environment. It is used to set up an atmospheric condition of either constant wind, wind gust or wind shear [48]. The sensor failure block contains the model that is discussed in section 3.2.2. The GPS block is used to set up a GPS failure scenario. It statistically generates errors on the position and velocity signals and simulates the satellite configuration effect on the accuracy of the measured position and velocity [34].

![Figure 3.4: Phastball UAV Simulink Model](image)

3.3.2 Sensor Failure Model
The sensor failure modeling is implemented using MATLAB/Simulink as well. Its development incorporates the use of Simulink mathematical operations and some of its other library properties including discrete functions and logical operations. Each block in the model corresponds to an individual sensor functional category (FC). The Simulink model is presented in Figure 3.5.
3.4 Trajectory Algorithms

There are several algorithms developed within the WVU UAS simulation environment for tasks such as trajectory generation, trajectory tracking and controller performance evaluation. This section addresses the specific algorithms, which were used for the completion of this thesis effort.

3.4.1 Trajectory Generation

The trajectory generation algorithm function is a very useful tool within the simulation environment because of its ability to create several trajectories as an array of varying headings and flight times. The algorithm function requires some initial parameters to be provided by the user to define the trajectory motion. These parameters include the commanded bank angle of the aircraft, its speed and rates of climb and descent. The initial aircraft position is also defined in the algorithm function. The function inputs include the heading and its corresponding flight time for 2-D trajectories. For 3-D trajectories the altitude, a third input, is required. Multiple commands can be combined and separated using a semi-colon and the commands must be provided in brackets. An example of the command to create ‘square’ trajectory is shown in equation 3.1. The command will initially set the aircraft heading to $270^\circ$ and then fly in that direction for 30 seconds at an altitude of 100 meters, then the heading will be set to $180^\circ$ and flown in that heading for 30 seconds but at an altitude of 200m and so on. Figure 3.12 shows a 2-D representation of a square trajectory. For this thesis effort, the trajectories created to be used for simulations are shown in figures Figure 3.6-Figure 3.9.

$$\text{Command} = [270 30 100; 180 30 200; 090 30 300; 360 30 100] \quad (3.1)$$
Figure 3.6: Phastball Square Trajectory

Figure 3.7: Phastball Oval Trajectory
3.4.2 Trajectory Tracking Controllers

There are several trajectory tracking controllers available within the WVU UAS simulation environment including PID controllers, position PID (PPID) controllers and NLDI controllers. Both fixed parameter and adaptive versions of the control laws are implemented [46]. In this thesis, only the position PID controller is used.
The position PID controller uses linear proportional, integral and derivative control laws to minimize the distance error between the reference and actual trajectory. The algorithm is made up of three distinct controllers namely, the forward, lateral and vertical distance controllers [46].

The position PID controller works as follows. Firstly, it determines the trajectory tracking errors of the actual with respect to the reference trajectory in terms of the distances mentioned above. It then calculates the errors and their corresponding time derivatives which it uses to produce bank angle, pitch angle and throttle commands. Finally, the angle commands are converted to deflections on the control surfaces of the aircraft to get the aircraft on the desired trajectory track [46]. Figure 3.10 provides a block diagram representation of how the position PID controller works.

![Figure 3.10: Position PID Block Diagram](image)

### 3.5 Performance Metrics

The evaluation of the trajectory tracking controllers available within the WVU UAS simulation environment is defined by two criteria; the accuracy of the trajectory tracking algorithm with respect to the desired trajectory and the amount of activity on the aircraft’s control surfaces and throttle [46]. These criteria are discussed in sections 3.5.1 and 3.5.2. Section 3.5.3 explains how the criteria are used to determine the overall performance index (PI) of the controller.

#### 3.5.1 Trajectory Tracking Performance Index

The accuracy of the trajectory tracking controller is evaluated using nine indices. The maximum, mean and standard deviation of the tracking errors in the X-Y plane, the Z-axis and the 3-D XYZ space [34]. The tracking errors are defined using equations 3.2-3.4.

\[
e_{xy}(t) = \sqrt{(x_a(t) - x_c(t))^2 + (y_a(t) - y_c(t))^2} \tag{3.2}
\]

\[
e_z(t) = \sqrt{(z_a(t) - z_c(t))^2} \tag{3.3}
\]

\[
e_{xyz}(t) = \sqrt{(x_a(t) - x_c(t))^2 + (y_a(t) - y_c(t))^2 + (z_a(t) - z_c(t))^2} \tag{3.4}
\]

Where:

\[x_a, y_a, z_a - actual \ aircraft \ position\]
Based off equations 3.2-3.4, the metrics for the selected trajectory tracking algorithm can be found for the nine indices mentioned. Equations 3.5-3.7 are used to find the maximum tracking errors in the XY plane, Z axis and XYZ 3-D space.

\[ e_{xy}^{\text{max}} = \max(|e_{xy}|) \]  
(3.5)

\[ e_{z}^{\text{max}} = \max(|e_z|) \]  
(3.6)

\[ e_{xyz}^{\text{max}} = \max(|e_{xyz}|) \]  
(3.7)

Equations 3.8-3.10 are used to find the average tracking errors in the XY plane, Z axis and XYZ 3-D space.

\[ \bar{e}_{xy} = \text{mean}(|e_{xy}|) \]  
(3.8)

\[ \bar{e}_{z} = \text{mean}(|e_z|) \]  
(3.9)

\[ \bar{e}_{xyz} = \text{mean}(|e_{xyz}|) \]  
(3.10)

Finally, equations 3.11-3.13 are used to find the standard deviation of the tracking errors in the XY plane, Z axis and XYZ 3-D space.

\[ \hat{e}_{xy} = \text{STD}(|e_{xy}|) \]  
(3.11)

\[ \hat{e}_{z} = \text{STD}(|e_z|) \]  
(3.12)

\[ \hat{e}_{xyz} = \text{STD}(|e_{xyz}|) \]  
(3.13)

Using equations 3.5-3.13, a performance vector can be defined for the trajectory tracking metrics as follows:

\[ PV_{TT} = [tt_i | i = 1,2,...,9] = [e_{xy}^{\text{max}}, e_{z}^{\text{max}}, e_{xyz}^{\text{max}}, \bar{e}_{xy}, \bar{e}_{z}, \bar{e}_{xyz}, \hat{e}_{xy}, \hat{e}_{z}, \hat{e}_{xyz}]^T \]  
(3.14)

Where:

\( tt_i \) – trajectory tracking index

3.5.2 Control Activity Performance Index

The performance of the control activity is evaluated using two parameters. The first is the integral of the rate of change of the signal sent to a control surface or throttle. This parameter is used to assess how gradual or sudden the control surfaces and throttle are exerted. Control saturation, the second parameter is defined to be the percentage of time steps when a control surface is operating close to or at its maximum allowable range as defined by the user [46]. It is important to note that different missions may require
different limits. The three control surfaces, aileron (a), elevator (e), rudder (r) and the throttle (t) lever are evaluated using this method. Equations 3.15-3.18 show the definitions used for the integration of their rates of change.

\[
I_{\delta_a} = \frac{1}{T} \int_0^T |\delta_a(t)| \, dt \tag{3.15}
\]

\[
I_{\delta_e} = \frac{1}{T} \int_0^T |\delta_e(t)| \, dt \tag{3.16}
\]

\[
I_{\delta_r} = \frac{1}{T} \int_0^T |\delta_r(t)| \, dt \tag{3.17}
\]

\[
I_{\delta_t} = \frac{1}{T} \int_0^T |\delta_t(t)| \, dt \tag{3.18}
\]

The control saturation index for the control surfaces and throttle lever is defined as follows:

\[
S_{\delta_a} = \frac{100}{T} \int_0^T \tilde{\delta}_a \, dt \tag{3.19}
\]

\[
S_{\delta_e} = \frac{100}{T} \int_0^T \tilde{\delta}_e \, dt \tag{3.20}
\]

\[
S_{\delta_r} = \frac{100}{T} \int_0^T \tilde{\delta}_r \, dt \tag{3.21}
\]

\[
S_{\delta_t} = \frac{100}{T} \int_0^T \tilde{\delta}_t \, dt \tag{3.22}
\]

Where:

\[
\tilde{\delta}_x = \begin{cases} 
0, & \text{if } \delta_x > \delta_{x\text{min}} \land \delta_x < \delta_{x\text{max}} \\
1, & \text{if } \delta_x \leq \delta_{x\text{min}} \land \delta_x \geq \delta_{x\text{max}}
\end{cases} \quad x \in [a, e, r, t] \tag{3.23}
\]

Finally, the performance vector for the control activity of a selected controller is defined using equation 3.24 below.

\[
P_{VA} = \begin{bmatrix} c_{a_i} \end{bmatrix} \quad i = 1, 2, ..., 8 = [I_{\delta_e} I_{\delta_a} I_{\delta_r} S_{\delta_e} S_{\delta_a} S_{\delta_r} S_{\delta_t}]^T \tag{3.24}
\]

Where:

\[
c_{a_i} - \text{control activity index}
\]

3.5.3 Total Performance Index
The total performance index of a controller is determined by combining the trajectory tracking and control activity indices discussed in sections 3.5.1 and 3.5.2. This is done using a normalized and weighted sum process for the 9 trajectory tracking indices and 8 control activity indices in equations 3.14 and 3.24. All chosen weights were based off of the importance of the particular index to the total performance index [34].
The total trajectory tracking index (TTI) has a performance value between 0 and 1. An index value of 0 means the controller is performing poorly and an index of 1 means it is performing excellently [48]. The total control activity index (CAI) is also similar to that of the TTI, where its values range from 0 to 1 also, with a value of 0 corresponding to a lot of exertion on the actuators and operation close to saturation while a value of 1 corresponds to minimal exertions on the actuators within range in order to follow the commanded trajectory [34]. The weight vectors for both the trajectory tracking and control activity indices are shown in Table 3.1 below. The weights are chosen based off the deemed importance of the defined indices.

<table>
<thead>
<tr>
<th></th>
<th>Trajectory Tracking and Control Activity Weight Vectors</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_{TT}$</td>
<td>[0.06 0.08 0.06 0.12 0.12 0.16 0.12 0.16 0.12]</td>
</tr>
<tr>
<td>$w_{CA}$</td>
<td>[0.10 0.10 0.10 0.20 0.10 0.10 0.10 0.20]</td>
</tr>
</tbody>
</table>

Using the weights defined above, the performance index for the trajectory tracking and control activity can be calculated using equations 3.25 and 3.26.

\[
PI_{TT} = w_{TT} \overline{PV}_{TT} \\
PI_{CA} = w_{CA} \overline{PV}_{CA}
\]

Where:

$\overline{PV}_{TT} - normalized \ trajectory \ tracking \ performance \ vector$

$\overline{PV}_{CA} - normalized \ control \ activity \ performance \ vector$

The total performance index is determined using a weighted sum method. A set of normalized weights are chosen for the trajectory tracking and the control activity performance vectors and then are multiplied by their corresponding weights and summed resulting in the total performance index. Equation 3.27 explains how the total PI is calculated. A value of 0.7 and 0.3 are used for the overall trajectory tracking and control activity weights, respectively. This is because higher accuracy in the trajectory tracking is desired and deemed more important compared to the control activity of the control surfaces. Hence, a value of 0 outlines poor overall performance and vice versa. A total PI value of 0.6 is deemed good. A value between 0.3 and 0.6 is typically acceptable as this will usually provide reasonable tracking and control activity, where the aircraft may sometimes be off course. A value below 0.3 is very poor as the aircraft is way off its desired course.

\[
PI_{Total} = \overline{w}_{TT} PV_{TT} + \overline{w}_{CA} PV_{CA}
\]

Where:

$\overline{w}_{TT} - overall \ trajectory \ tracking \ weight$

$\overline{w}_{CA} - overall \ control \ activity \ weight$
The trajectory tracking and control activity PIs can also have a value of zero if the aircraft is uncontrollable or if the corresponding indices exceed their corresponding thresholds. The thresholds for all indices are shown in Table 3.2. Note the trajectory tracking PI cutoff values are in meters.

<table>
<thead>
<tr>
<th>Table 3.2: PI Cutoff Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trajectory Tracking PI Cutoff Values</td>
</tr>
<tr>
<td>Control Activity PI Cutoff Values</td>
</tr>
</tbody>
</table>

3.6 FlightGear

FlightGear is an open-source software package [9] used within the WVU UAS simulation environment. It is a 3-dimensional (3-D) visualization tool allowing for a lot of user flexibility such as setting the scenery, selecting the aircraft and view. The inputs to FlightGear are received from the selected aircraft’s state variables which are used to generate the 3-D representation of the aircraft simulation. FlightGear also has a simple head up display (HUD) which aids visualization of key parameters such as the aircraft’s pitch and bank angles, heading, speed, altitude [10]. The FlightGear interface for Phastball can be seen in Figure 3.11.

![FlightGear Interface – Phastball](image)

3.7 UAVDashboard

The UAV dashboard is another tool that aids simulation visualization. It is a custom software developed for mission configuration and visualization [46]. It provides a 2
dimensional (2-D) visualization of the flight environment. The software was developed using Microsoft Visual C# and it allows the user to specify the position of the aircraft, its altitude and orientation. It also allows the placement of threat zones of different sizes for obstacle avoidance tests [46]. The map used in the software is that of the San Francisco bay area. Typical views of the software interface are shown in Figure 3.12 and Figure 3.13.

Figure 3.12: WVU UAVDashboard

Figure 3.13: San Francisco Bay Area – UAVDashboard Map [10]
CHAPTER 4: AIRCRAFT MODELING

This chapter explains the general nomenclature used in flight dynamics for aircraft modeling, the method used to model the Phastball aircraft, its implementation and verification using MATLAB and Simulink and an analysis of its dynamic parameters.

4.1 Aircraft Dynamics

4.1.1 Coordinate System

For modeling purposes, an aircraft is considered to be a rigid body of constant mass. Its coordinate system (CS) also known as its body axes is symbolized as $CS_B$ or $CX_B Y_B Z_B$. Its origin is defined to be at the center of its mass with its longitudinal axis ($X_B$) being along its fuselage, defined positive pointing forward. Its lateral axis ($Y_B$) defined positive to the right of the pilot, perpendicular to ($X_B$) and its vertical axis ($Z_B$) is positive downwards according to the right hand rule [51]. The coordinate system can be seen in Figure 4.1

![Figure 4.1: Aircraft Body Axes](image.png)

The conversion from the aircraft’s ($CS_B$) to the earth’s coordinate system ($CS_E$) is done using a transformation matrices that is dependent on the Euler angles, $\phi$, $\theta$ and $\psi$. Where $\phi$ is the roll angle, $\theta$ is the pitch angle and $\psi$ is the yaw angle. The transformation matrices to convert from the earth to body CS and vice versa are shown in equations 4.1 and 4.2, respectively. The transformation matrix in equation 4.2 is the transpose of equation 4.1 because transformation is orthonormal [51].

$$
H^B_E = \begin{bmatrix}
\cos \theta \cos \psi & \cos \theta \sin \psi & - \sin \theta \\
- \cos \phi \sin \psi + \sin \phi \sin \theta \cos \psi & \cos \phi \cos \psi + \sin \phi \sin \theta \sin \psi & \sin \phi \cos \theta \\
\sin \phi \sin \psi + \cos \phi \sin \theta \cos \psi & - \sin \phi \cos \psi + \cos \phi \sin \theta \sin \psi & \cos \phi \cos \theta
\end{bmatrix}
$$ (4.1)

$$
H^E_B = \begin{bmatrix}
\cos \theta \cos \psi & \cos \theta \sin \psi & - \sin \theta \\
- \cos \phi \sin \psi + \sin \phi \sin \theta \cos \psi & \cos \phi \cos \psi + \sin \phi \sin \theta \sin \psi & \sin \phi \cos \theta \\
\sin \phi \sin \psi + \cos \phi \sin \theta \cos \psi & - \sin \phi \cos \psi + \cos \phi \sin \theta \sin \psi & \cos \phi \cos \theta
\end{bmatrix}^T
$$ (4.2)
4.1.2 Equations of Motion

Based off the rigid body assumption, aircraft motion can be modeled using the following equations [51]:

\[
\dot{u} = rv - qw + g_x + \left(\frac{F_x}{m}\right) \\
\dot{v} = pw - ru + g_y + \left(\frac{F_y}{m}\right) \\
\dot{w} = qu - pv + g_z + \left(\frac{F_z}{m}\right)
\]

\[
p = \frac{\left[I_{zz} L + I_{xz} N - \left[I_{xx}(I_{yy} - I_{xx} - I_{zz})\right]pq + \left[I_{zz}^2 + I_{zz}(I_{zz} - I_{yy})\right]rq\right]}{\left(I_{xx}I_{zz} - I_{zz}^2\right)}
\]

\[
\dot{q} = \frac{1}{I_{yy}} \left[M - (I_{xx} - I_{zz})pr - I_{xz}(p^2 - r^2)\right]
\]

\[
\dot{\phi} = p + (q \sin \phi + r \cos \phi) \tan \theta
\]

\[
\dot{\theta} = q \cos \phi - r \sin \phi
\]

\[
\dot{\psi} = (q \sin \phi + r \cos \phi) \sec \theta
\]

Where:

\(\dot{u}, \dot{v}\) and \(\dot{w}\) – rate of change of aircraft translational velocity

\(\dot{p}, \dot{q}\) and \(\dot{r}\) – rate of change aircraft rotational velocity

\(\dot{\phi}, \dot{\theta}\) and \(\dot{\psi}\) – rate of change of aircraft rotational position

In the equations listed above, \(m\) is the mass of the aircraft, \(p, q\) and \(r\) are the angular rates, \(u, v\) and \(w\), the translational velocity components and \(q, \theta\) and \(\psi\) are the Euler angles as defined in section 4.1.1 above. \(F_x, F_y\) and \(F_z\) are the force components in the \(x, y\) and \(z\) directions whereas \(L, M\) and \(N\) are the aerodynamic moments in the same directions respectively. \(I_{xx}, I_{yy}, I_{zz}, I_{xz}\) are the moments of inertia about the specified axes and \(g_x, g_y\) and \(g_z\) are the gravitational accelerations in the \(x, y\) and \(z\) directions. The gravitational vector is shown in equation 4.11 below.

\[
\mathbf{\bar{g}} = \begin{bmatrix} g_x \\ g_y \\ g_z \end{bmatrix} = \begin{bmatrix} -g \sin \theta \\ g \sin \phi \cos \theta \\ g \cos \phi \cos \theta \end{bmatrix}
\]

The aerodynamic and propulsion forces and moments governing the aircraft motion are denoted as follows:

\[
F_x = C_x \bar{q} S_{ref} + T_x
\]
\[ F_y = C_y \bar{q} S_{ref} + T_y \] 
\[ F_z = C_z \bar{q} S_{ref} + T_z \] 
\[ L = C_l \bar{q} S_{ref} + M_{T_x} \] 
\[ M = C_m \bar{q} S_{ref} + M_{T_y} \] 
\[ N = C_n \bar{q} S_{ref} + M_{T_z} \]

Where:
\[ \bar{q} = \frac{1}{2} \rho V^2 \]

In equations 4.12-4.17, \( C_x, C_y \) and \( C_z \) are the force coefficients and \( C_l, C_m \) and \( C_n \) are the moment coefficients. \( T_x, T_y, T_z \) and \( M_{T_x}, M_{T_y}, M_{T_z} \) are the thrust force and moment effects in the \( x, y \) and \( z \) axes, respectively. In equation 4.18, \( \bar{q} \) is the dynamic pressure and \( V \) is the aircraft velocity.

4.2 Tornado Vortex Lattice Method

Tornado is a 3-D vortex lattice program with a flexible wake used for aerodynamic modeling. It is a free tool available publicly and very user friendly with its implementation in MATLAB. This section expands on Tornado, how it was used to model the Phastball aircraft, its downsides and how they were counteracted. Tornado is run in the command window of MATLAB and Figure 4.2 shows its main menu GUI.

One major assumption made to implement Tornado’s vortex lattice theory is the presence of small angles of attack. So, Tornado cannot be trusted to provide useful results with large angles of attacks or large rotational speeds. It is also important to note that Tornado does not consider fuselage, engine or landing gear effects, so it can not completely find the drag results as stated in Chapter 2.1 above. Also, compressibility effects are not considered in Tornado [16], although this is not a concern when modeling small low-speed UAVs.
4.2.1 Coordinate System

Tornado uses the Cartesian coordinate system with the aircraft’s positive X-axis defined backwards along its body, its positive Y-axis defined to the right of the aircraft and its positive Z-axis defined upwards according to the right-hand rule [16]. Figure 4.3 shows Tornado’s axes definition.
According to Tornado’s definition, the first wing defined is taken to be the main wing and the direction of the X-axis is defined to be in line with the wings root chord. Also, all flat surfaces are considered to be wings. In other words, the main wing, horizontal and vertical tails are all considered to be wings with the vertical tail having a dihedral angle of 90°. All wing surfaces are split into partitions and increasing the number of partitions on a surface increases its complexity. Partitions may be increased to account for swept and tapered wings. Tornado also allows the user to input control surfaces on the wing. The chord of the control surface with respect to that of the wing is used to define the control surface. For symmetric wing surfaces, the control surfaces can then be defined to deflect symmetrically or not.

4.2.2 Phastball Geometry Modeling

For the modeling of Phastball using Tornado, the first step was to define the aircraft geometry. The geometry definition included the information listed below. The main wing was split into two partitions because the control surface was not on the whole wing and the wing geometry varies across the wing semi-span. The horizontal and vertical tails only had one partition. It is important to note that some of these pieces of information would have to be provided more than once if a wing has multiple partitions.

- Number of wings
- Number of semi-span-wise partitions on the wing
- x, y and z coordinates the wing
- Reference x, y and z coordinates (only specified once)
- Wing mirroring about the x-axis
- Wing root chord
- Wing partition root chord airfoil
- Number of panels chord-wise on the wing partition
- Dihedral angle of the wing partition
- Number of panels semi-span-wise on the wing
- Wing partition span
- Wing partition taper ratio
- Wing partition tip chord airfoil
- Wing quarter chord line sweep angle
- Wing outboard twist
- Mesh distribution type
- Presence of control surfaces on the wing partition
- Control surface chord as a fraction of the wing chord
- Number of chord-wise panels on the control surface
- Control surface deflection definition (symmetric or differential)
The number of wings was specified to be three: the main wing and the horizontal and vertical tails. The values used for the remaining parameters for the main wing, horizontal and vertical tails are shown in Table 4.1. The source of all information about the Phastball aircraft is reference [53].

Table 4.1: Phastball Geometry Data for Tornado

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Main Wing</th>
<th>Horizontal Tail</th>
<th>Vertical Tail</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Global Entries</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Semi-span-wise partitions</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Reference x, y, z coordinates</td>
<td>(0, 0, 0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Center of gravity position</td>
<td>(0.0775, 0, 0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Partition Specific Entries</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Partition number</td>
<td>Partition 1</td>
<td>Partition 2</td>
<td>Partition 1</td>
</tr>
<tr>
<td>Wing x, y, z coordinates</td>
<td>(0, 0, 0)</td>
<td>(1.1042, 0, 0.3)</td>
<td>(1.1042, 0, 0)</td>
</tr>
<tr>
<td>Wing symmetry about the x axis</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Wing root chord</td>
<td>0.3520 m</td>
<td>0.2660 m</td>
<td>0.2720 m</td>
</tr>
<tr>
<td>Wing airfoil</td>
<td>NACA 2410</td>
<td>NACA 2410</td>
<td>NACA 0009</td>
</tr>
<tr>
<td>Chord-wise panels</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Dihedral angle</td>
<td>0°</td>
<td>0°</td>
<td>0°</td>
</tr>
<tr>
<td>Semi-span-wise panels</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Wing partition span</td>
<td>0.6085</td>
<td>0.61</td>
<td>0.381</td>
</tr>
<tr>
<td>Taper ratio</td>
<td>1</td>
<td>0.6477</td>
<td>0.7707</td>
</tr>
<tr>
<td>Tip chord airfoil</td>
<td>NACA 2410</td>
<td>NACA 2410</td>
<td>NACA 0009</td>
</tr>
<tr>
<td>Quarter chord sweep angle</td>
<td>0°</td>
<td>0°</td>
<td>0°</td>
</tr>
<tr>
<td>Outboard twist</td>
<td>0°</td>
<td>0°</td>
<td>0°</td>
</tr>
<tr>
<td>Mesh distribution type</td>
<td>Linear</td>
<td>Linear</td>
<td>Linear</td>
</tr>
<tr>
<td>Control surface on the partition</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Control surface to wing chord ratio</td>
<td>N/A</td>
<td>0.2532</td>
<td>0.3440</td>
</tr>
<tr>
<td>Control surface chord wise panels</td>
<td>N/A</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Symmetric control surface deflection</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

An orthographic projection of the wings on the aircraft using the values shown in Table 4.1 as inputs to Tornado is plotted in Figure 4.4, while the 3-D wing and partition
layout can be seen in Figure 4.5 with specified center of gravity (CG), reference point and mean aerodynamic chord (MAC).

Figure 4.4: Phastball Wing Orthographic Projection

Figure 4.5: 3-D Wing and Partition Layout

Figure 4.5: 3-D Wing and Partition Layout
Figure 4.6 shows a 3-D visualization of the panels created, their connection points and normals for all wing surfaces. Note that this figure is a general representation of the geometry mesh and not the VLM results.

After defining the geometry, the user will have to define the flight condition state. The information required by Tornado are the angle of attack ($\alpha$), sideslip angle ($\beta$), roll, pitch and yaw angular velocities ($p$, $q$ and $r$) and the rate of change of $\alpha$ and $\beta$. After providing these information, Tornado then allows the user determine the information they wish to enter regarding the speed and altitude. The user can provide the true air speed (TAS), equivalent air speed (EAS) or calibrated air speed (CAS) and altitude. This information can be entered in either the International System (SI) units or in Imperial units. Tornado also allows the user provide of the Mach number in place of the speed. The parameters used to define the flight condition state are shown in Table 4.2.

**Table 4.2: Phastball Flight Condition Setup in Tornado**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>0°</td>
</tr>
<tr>
<td>$\beta$</td>
<td>0°</td>
</tr>
<tr>
<td>$p$</td>
<td>0 deg/s</td>
</tr>
<tr>
<td>$q$</td>
<td>0 deg/s</td>
</tr>
<tr>
<td>$r$</td>
<td>0 deg/s</td>
</tr>
<tr>
<td>$\dot{\alpha}$</td>
<td>0 deg/s</td>
</tr>
<tr>
<td>$\dot{\beta}$</td>
<td>0 deg/s</td>
</tr>
<tr>
<td>TAS</td>
<td>25 m/s</td>
</tr>
<tr>
<td>Altitude</td>
<td>100 m</td>
</tr>
</tbody>
</table>
After setting up the geometry and state, the vortex lattice can be generated. Tornado allows the user to select between a freestream following wake (Tornado’s method) and a fixed wake which is the standard vortex lattice method. Tornado’s method was used to generate the lattice and the results will be discussed in section 4.2.3.

4.2.3 Phastball Modeling Results
Tornado provides its results in a “.mat” file which is readable by MATLAB. This file contains the provided geometry, the generated vortex lattice, the reference values of the aircraft such as reference span and area, the results including forces and moments acting on all wings, pressure distribution and stability and control derivatives and lastly, the defined flight state. The results and the generated lattice data are the actual output from Tornado as the other “outputs” were actually inputted to Tornado. All results presented were generated using the values listed in Table 4.2. Table 4.3 shows the force results from Tornado for Phastball.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lift (N)</td>
<td>49.19</td>
</tr>
<tr>
<td>Induced Drag (N)</td>
<td>0.3588</td>
</tr>
<tr>
<td>Side Force (N)</td>
<td>0</td>
</tr>
<tr>
<td>$C_l$</td>
<td>0.1659</td>
</tr>
<tr>
<td>$C_D$</td>
<td>0.001210</td>
</tr>
<tr>
<td>$C_Y$</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4.3: Phastball Force Results from Tornado

Table 4.4 shows the results for the rolling, pitching and yawing moment coefficients.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_l$</td>
<td>0</td>
</tr>
<tr>
<td>$C_m$</td>
<td>-0.06388</td>
</tr>
<tr>
<td>$C_n$</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4.4: Angular Moments Coefficients

Table 4.5 shows the force coefficient results for all the wing surfaces.

<table>
<thead>
<tr>
<th>Coefficient</th>
<th>Main Wing</th>
<th>Vertical Tail</th>
<th>Horizontal Tail</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_{l_0}$</td>
<td>0.1747</td>
<td>0</td>
<td>-0.008881</td>
</tr>
<tr>
<td>$C_{D_0}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4.5: Force Coefficients for all Wing Surfaces
Table 4.6 shows the stability derivatives with respect to changes in $\alpha$ and $\beta$.

**Table 4.6: $\alpha$ and $\beta$ Stability Derivatives**

<table>
<thead>
<tr>
<th>Coefficient</th>
<th>$C_{L\alpha}$</th>
<th>$C_{L\beta}$</th>
<th>$C_{D\alpha}$</th>
<th>$C_{D\beta}$</th>
<th>$C_{Z\alpha}$</th>
<th>$C_{Z\beta}$</th>
<th>$C_{m\alpha}$</th>
<th>$C_{m\beta}$</th>
<th>$C_{l\alpha}$</th>
<th>$C_{l\beta}$</th>
<th>$C_{n\alpha}$</th>
<th>$C_{n\beta}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_{L\alpha}$</td>
<td>5.277</td>
<td>0</td>
<td>0.06733</td>
<td>0</td>
<td>5.277</td>
<td>0</td>
<td>0.02693</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-3.239</td>
<td>0.001281</td>
</tr>
<tr>
<td>$C_{L\beta}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-0.1288</td>
</tr>
</tbody>
</table>

Table 4.7 shows the stability derivatives with respect to changes in the angular rates, $p$, $q$ and $r$.

**Table 4.7: $p$, $q$ and $r$ Stability Derivatives**

<table>
<thead>
<tr>
<th>Coefficient</th>
<th>$C_{p\alpha}$</th>
<th>$C_{p\beta}$</th>
<th>$C_{q\alpha}$</th>
<th>$C_{q\beta}$</th>
<th>$C_{r\alpha}$</th>
<th>$C_{r\beta}$</th>
<th>$C_{l\alpha}$</th>
<th>$C_{l\beta}$</th>
<th>$C_{m\alpha}$</th>
<th>$C_{m\beta}$</th>
<th>$C_{n\alpha}$</th>
<th>$C_{n\beta}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_{p\alpha}$</td>
<td>0</td>
<td>0</td>
<td>11.11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.08891</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-0.02308</td>
</tr>
<tr>
<td>$C_{p\beta}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$C_{q\alpha}$</td>
<td>-0.02308</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-0.1814</td>
</tr>
<tr>
<td>$C_{q\beta}$</td>
<td>-0.5125</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-23.10</td>
</tr>
<tr>
<td>$C_{r\alpha}$</td>
<td>0</td>
<td>0</td>
<td>-23.10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$C_{r\beta}$</td>
<td>0.003616</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-0.1310</td>
</tr>
</tbody>
</table>

Finally, the control surface derivatives are shown in Table 4.8 for all control surfaces: aileron, elevator and rudder.

**Table 4.8: Control Surface Derivatives**

<table>
<thead>
<tr>
<th>Coefficient</th>
<th>Control Surface</th>
<th>Aileron</th>
<th>Rudder</th>
<th>Elevator</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_{\delta}$</td>
<td>Aileron</td>
<td>0</td>
<td>0</td>
<td>0.5587</td>
</tr>
<tr>
<td>$C_{\delta}$</td>
<td>Rudder</td>
<td>0</td>
<td>0</td>
<td>0.001708</td>
</tr>
<tr>
<td>$C_{\gamma}$</td>
<td>Elevator</td>
<td>0.02345</td>
<td>-0.1814</td>
<td>0</td>
</tr>
<tr>
<td>$C_{\delta}$</td>
<td>Elevator</td>
<td>0.3326</td>
<td>0.01683</td>
<td>0</td>
</tr>
<tr>
<td>$C_{\delta}$</td>
<td>Elevator</td>
<td>0</td>
<td>0</td>
<td>-2.0607</td>
</tr>
<tr>
<td>$C_{\delta}$</td>
<td>Elevator</td>
<td>0.001803</td>
<td>-0.09364</td>
<td>0</td>
</tr>
</tbody>
</table>

Note that sign convention of all of the results from Tornado correspond to its coordinate system definition shown in Figure 4.3.
Most of the results from Tornado were useful for the model development in MATLAB/Simulink apart from the drag coefficient and lift coefficients at high angles of attack. This was expected as Tornado is not a very suitable tool to estimate the drag coefficient. This parameter is also affected by the fact that the fuselage effects are not accounted for in the model. The lift coefficient results were only useful for small angles of attack. The lift curve slope result from Tornado is shown in Figure 4.7. From Figure 4.7, it is noticeable that due to Tornado being a linear VLM method, it assumes the lift curve slope is “linear” from -90 to 90° signifying the stall angle of attack (α_{stall}) will occur close to 90°, which is not realistic. For this reason, the lift curve slope had to be corrected.

![Figure 4.7: 3-D Lift Curve Slope from Tornado](image)

In order to correct the lift curve slope, the wing’s NACA 2410 2-D lift curve slope was needed, but before it could be gotten the Reynolds number had to be calculated first as the 2-D lift curve slope depends on it. The Reynolds number was calculated using equation 4.19.

\[
Re = \frac{\rho V \bar{c}}{\mu}
\]  

(4.19)

Where:

\(\rho\) – Density

\(V\) – Aircraft velocity

\(\bar{c}\) – Mean aerodynamic chord (MAC)

\(\mu\) – Dynamics viscosity

The Reynolds number was found to be 494,236 for a density of 1.123 kg/m³, a velocity of 25 m/s, an MAC of 0.31 m [53] and a dynamic viscosity of 1.761x10^{-5} N·s/m² [54]. Using this Reynolds number, the 2-D lift curve slope can be found in Figure 4.8.
Using Figure 4.8, the maximum 2-D lift coefficient ($C_{L\infty}$) was seen to be approximately 1.5. This can be converted to a 3-D lift coefficient ($C_L$) using equation 4.20 in [56]. The ratio $\left(\frac{C_{L\max}}{C_{L\infty\max}}\right)$ shown in the equation was found to be 0.9 from Figure 9.9 in [56] as the Phastball leading edge sweep angle is 0°. The maximum 3-D lift coefficient ($C_{L\max}$) was then found to be approximately 1.35.

$$C_{L\max} = \left[\frac{C_{L\max}}{C_{L\infty\max}}\right]C_{L\infty\max} \tag{4.20}$$

Knowing $C_{L\max}$, the zero lift angle of attack ($\alpha_0$) and the rate of change of $C_L$ with respect to $\alpha$ ($C_{L\alpha}$), a new 3-D lift curve slope can be generated using mathematical relationships with the understanding of how the aforementioned parameters are related to the slope. The corrected lift curve slope is shown in Figure 4.9.

**Figure 4.9: Corrected 3-D Lift Curve Slope**
The drag coefficient ($C_D$) gotten from Tornado was 0.001210 as shown in Table 4.3. This value was found to be much lower than expected and needed to be corrected. To do this, the drag coefficient was calculated using equation 4.21.

$$C_D = \frac{D}{\bar{q} S_{\text{ref}}} \quad (4.21)$$

Where:

- $D$ – total drag
- $\bar{q}$ – dynamic pressure (4.18)
- $S_{\text{ref}}$ – wing reference area

The total drag was equated to the aircraft’s thrust at cruise, 30 N and $\bar{q}$ and $S_{\text{ref}}$ were found to be 7.995 Pa and 0.75 m² [53] respectively. A value of 0.1045 was gotten for the $C_D$, a significant difference compared to the result from Tornado.

After completing the modeling of Phastball, the next step was to implement the results in MATLAB/Simulink. This process is discussed in section 4.3.

### 4.3 MATLAB/Simulink Implementation

The WVU UAS simulation environment supports the addition of new aircraft models using an already available model as a guide. For the new model, properties and characteristics including its mass, moments of inertia, speed and much more needed to be updated. All these parameters were defined for the Phastball model in the MATLAB workspace and can be found in Table 4.9 [53]. The modeled Phastball results from section 4.2.3 were also defined in the workspace.

#### Table 4.9: Phastball Characteristics [53]

<table>
<thead>
<tr>
<th>Phastball Characteristics</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mass ($m$)</td>
<td>12 kg</td>
</tr>
<tr>
<td>Velocity ($V$)</td>
<td>25 m/s</td>
</tr>
<tr>
<td>Cruise Altitude ($h$)</td>
<td>100 m</td>
</tr>
<tr>
<td>Wing Span ($b$)</td>
<td>2.437 m</td>
</tr>
<tr>
<td>Mean Aerodynamic Chord ($\bar{c}$)</td>
<td>0.31 m</td>
</tr>
<tr>
<td>Wing Reference Area ($S_{\text{ref}}$)</td>
<td>0.75 m²</td>
</tr>
<tr>
<td>Moment of Inertia About x Body Axis ($I_{xx}$)</td>
<td>0.96 kg/m²</td>
</tr>
<tr>
<td>Moment of Inertia About y Body Axis ($I_{yy}$)</td>
<td>2.91 kg/m²</td>
</tr>
<tr>
<td>Moment of Inertia About z Body Axis ($I_{zz}$)</td>
<td>3.64 kg/m²</td>
</tr>
<tr>
<td>Moment of Inertia About x-z Body Plane ($I_{xz}$)</td>
<td>0.20 kg/m²</td>
</tr>
<tr>
<td>Maximum Thrust ($T$)</td>
<td>60 N</td>
</tr>
</tbody>
</table>
The Simulink model was based off an already available template within the WVU UAS simulation environment, which implements the aircraft dynamic equations shown in section 4.1, although aircraft characteristics and some other parameters such as the position PID controller gains had to be updated to correspond to the characteristics of Phastball.

4.4 Model Verification and Analysis

For the verification of the aircraft model, previously available flight data [53] was obtained and used as input commands to the model and the output from the flight test compared to that of the simulation environment. The flight data inputs provided were doublets on the three input channels and the throttle commands were also inputted to the model. Figure 4.10–Figure 4.12 show the comparison between the flight test data and the simulation environment results for the provided roll, pitch and yaw doublets. It can be seen that the result from the WVU UAS simulation environment and the flight test data match up fairly well for the roll and pitch doublets, but not well enough for the yaw doublet. This could be due to the fact that VLM tools do not generally model the rudder dynamics well enough and the rudder is not the most used control surface for performing maneuvers. Also, it is important to note that the flight data available was not intended for verification purposes and data on atmospheric conditions or perturbations were not available. Only the control surface and throttle commands were the inputs to the WVU UAS simulation environment.

![Figure 4.10: Roll Angular Rate vs Time](image-url)
With the yaw verification having a poor match, an analysis was performed on the Dutch roll of the aircraft using the acquired flight simulation data. The sideslip angle \( \beta \) was the parameter used for the analysis. The oscillatory motion of the sideslip angle representing the Dutch roll can be described as follows [57]:

\[
\beta(t) = A_{DR}e^{-\zeta_{DR} \omega_{dDR} t} \sin(\omega_{dDR} t + \mu_{DR})
\]  

(4.22)
Where $\omega_{nDR}$ is the Dutch roll undamped or natural frequency, $\omega_{dDR}$ is its damped frequency, $\zeta_{DR}$ is its damping coefficient, $A_{DR}$ is its amplitude and $\mu_{DR}$ is its phase angle. Note that the values of amplitude and phase angle are dependent on the initial conditions. So, $\omega_{dDR}$ can be represented as follows:

$$\omega_{dDR} = \omega_{nDR} \sqrt{1 - \zeta_{DR}^2}$$ \hspace{1cm} (4.23)

A Peak-to-Peak method [57] was used to solve for the Dutch roll’s characteristics. For this method, the time to reach the peaks of the sinusoid as well as the corresponding peak values needs to be known. The time period, $T_P$ can be defined as the time it takes to go from one peak to another as in equation 4.24, where $t_1$ is the time to reach one peak and $t_2$ is the time to reach the successive peak. For an impulse on the directional channel, the aircraft response can be seen in Figure 4.13.

$$T_P = t_2 - t_1$$ \hspace{1cm} (4.24)

The first and second peaks shown in Figure 4.13 can be expressed mathematically as follows:

$$\beta_1(t) = A_{DR}e^{-\zeta_{DR}\omega_{nDR}t_1} \sin(\omega_{dDR}t_1 + \mu_{DR})$$ \hspace{1cm} (4.25)
\[
\beta_2(t) = A_{DR}e^{-\zeta_{DR}\omega_{nDR}t} \sin(\omega_{dDR}t + \mu_{DR}) 
\]

(4.26)

Their ratios can then be defined as:

\[
\frac{\beta_2}{\beta_1} = \frac{A_{DR}e^{-\zeta_{DR}\omega_{nDR}t_2} \sin(\omega_{dDR}t_2 + \mu_{DR})}{A_{DR}e^{-\zeta_{DR}\omega_{nDR}t_1} \sin(\omega_{dDR}t_1 + \mu_{DR})} 
\]

(4.27)

Knowing the period of an oscillation is inversely proportional to its frequency and that a circular revolution equals \(2\pi\) radians, the following expression can be derived for the time period.

\[
T_P = \frac{1}{f \text{requency (Hz)}} = \frac{2\pi}{\text{circular frequency (rad/sec)}} 
\]

(4.28)

Where \(f\) is the frequency of the oscillation. So, equation 4.24 can be rewritten as follows:

\[
t_2 = t_1 + T_P = t_1 + \frac{2\pi}{\omega_{dDR}} 
\]

(4.29)

The ratios of the peaks can then be rewritten as:

\[
\frac{\beta_2}{\beta_1} = \frac{A_{DR}e^{-\zeta_{DR}\omega_{nDR}t_2} \sin(\omega_{dDR}t_2 + \mu_{DR})}{A_{DR}e^{-\zeta_{DR}\omega_{nDR}t_1} \sin(\omega_{dDR}t_1 + \mu_{DR})} = \frac{e^{-\zeta_{DR}\omega_{nDR}t_2} \sin(\omega_{dDR}(t_1 + T_P))}{e^{-\zeta_{DR}\omega_{nDR}t_1} \sin(\omega_{dDR}t_1)} = e^{-\zeta_{DR}\omega_{nDR}T_P} 
\]

(4.30)

Knowing the Dutch roll mode is made up of consecutive oscillations, a more particular ratio can be defined for all peaks using their average value over a set number of peaks, \(N\). This is called the Transient Peak Ratio (TPR) and is defined as follows:

\[
TPR = \frac{\left(\frac{\beta_2}{\beta_1} + \frac{\beta_3}{\beta_2} + \frac{\beta_4}{\beta_3} + \cdots\right)}{N} = \frac{1}{N} \sum_{i=1}^{N} \beta_{i+1} \beta_i 
\]

(4.31)

From the TPR expression, a logarithmic decrement can be defined as follows:

\[
\delta = \ln(TPR) = -\zeta_{DR}\omega_{nDR} T_P = -\zeta_{DR}\omega_{nDR} \left(\frac{2\pi}{\omega_{dDR}}\right) 
\]

(4.32)

With the damped and natural frequencies related by equation 4.23, \(\delta\) can be rewritten as:
\[
\delta = -\zeta_{DR} \omega_{n_{DR}} \left( \frac{2\pi}{\omega_{n_{DR}} \sqrt{1 - \zeta_{DR}^2}} \right) = -\frac{2\pi \zeta_{DR}}{\sqrt{1 - \zeta_{DR}^2}} \Rightarrow \delta^2
\]

\[
= \frac{(2\pi)^2 \zeta_{DR}^2}{|1 - \zeta_{DR}^2|}
\]

The damping coefficient \( \zeta_{DR} \) can be solved for as follows:

\[
\zeta_{DR}^2 = \frac{\delta^2}{(2\pi)^2 + \delta^2} \Rightarrow \zeta = \frac{[\ln(TPR)]}{\sqrt{(2\pi)^2 + (\ln^2(TPR))}}
\]

After finding the damping coefficient, the natural frequency can be found using the following equations:

\[
T_{P_{AVG}} = \frac{1}{N} \sum_{i=1}^{N} T_{P_{i}}
\]

\[
\omega_{d_{DR_{AVG}}} = \frac{2\pi}{T_{P_{AVG}}} \Rightarrow \omega_{n_{DR_{AVG}}} = \omega_{n_{DR}} = \frac{\omega_{d_{DR_{AVG}}}}{\sqrt{1 - \zeta_{DR}^2}}
\]

After performing the analysis of the Dutch Roll, \( \zeta_{DR} \) was found to be 0.1039 and the natural frequency, \( \omega_{n_{DR}} \) was found to be 5.901 rad/s.
CHAPTER 5: SENSOR FAILURE MODEL

This chapter focuses on the newly developed sensor failure model, its definitions, notations, assumptions made, the failure categories developed and its implementation in MATLAB/Simulink [58].

5.1 Definitions, Notations, Nomenclature

The notations and nomenclature used in the sensor model are defined as follows:

5.1.1 Measurand

The measurand, $x(t)$ is the physical variable to be measured by the sensor. Examples include: angular rate, altitude, temperature, ground velocity and acceleration. It will also be used to denote the “actual value” of the physical variable. Note that the Laplace transform of $x(t)$ will also be referred to as “measurand” as shown in equation 5.1.

$$\mathcal{L}\left[ x(t) \right] = X(s) \quad (5.1)$$

5.1.2 Sensor Output

The sensor output denoted as $y(t)$ represents the direct response of the sensor when exposed to the measurand. Just like the measurand, the Laplace transform of the sensor output may also be referred to as the sensor output.

5.1.3 Measurement

The measurement, $m(t)$ represents the output from the sensor converted to units of the measurand. Its Laplace transform is also referred to as the measurement.

Figure 5.1 illustrates a measurement system’s block diagram showing the measurand, sensor output and the converted measurement signal. Figure 5.2 shows an example of a measurement system where the measurand is the temperature in degrees Celsius, the sensor is a thermocouple, the sensor output is the voltage, the conversion is carried out using non-linear voltage to temperature conversion tables and the measurement is the measured temperature in degrees Celsius.
5.1.4 Sensor Perfect Condition
Sensor perfect condition or operation represents the ideal situation when the measurement is equal to the measurand. This is expressed in equation 5.2 below. Typically, this cannot be achieved in practice, but can be simulated for system analysis and evaluation purposes.

\[ m(t) = x(t) \]  

(5.2)

5.1.5 Sensor Normal Condition
Sensor normal or nominal condition represents the situation when the sensor works within its design specifications. Note that “imperfection” is present and acceptable as “normal condition” within certain limits/ranges, such that:

\[ m(t) = F[x(t)] \]  

(5.3)

\[ F[x(t)] \approx x(t) \]  

(5.4)

The situation when parameters are varied within normal ranges will be referred to as sensor modified condition (MC). Note that it is still a normal condition.

5.1.6 Sensor Abnormal Condition
An abnormal condition (AC) on a sensor or abnormal operation represents any deviation from or alteration of \( F[x(t)] \). Note that an AC may be caused by internal or external factors. A physical damage of the sensor is categorized as an internal factor whereas examples of external factors may include poor calibration, electro-magnetic interference, temperature outside bounds, or the measurand exceeding design range. In these instances, the sensor is “healthy”, but the measurement does not match the measurand beyond acceptable inaccuracies.

5.1.7 Sensor Abnormal Condition Consistency
Sensor AC consistency represents those characteristics of the AC related to its behavior with respect to time and magnitude (or severity).

Time related sensor AC behavior can be:

- Permanent: once an AC occurs, normal operation of the sensor is never re-instated.
- Intermittent: AC occurs and disappears alternately, or in other words, nominal and abnormal operation succeed each other after random or defined intervals.

Magnitude related sensor AC behavior can be:

- Constant: all parameters that characterize the AC are constant over the duration of the AC.
- Variable: at least one AC-defining parameter varies randomly or in a determined manner.
Some aspects need to be considered when modeling and simulating sensor failures. Firstly, there is no feedback control without sensors. This implies that sensor models are needed for control system design and evaluation, as well as for monitoring purposes. Also, redundancy is used on a large scale with sensors; however, it is sometimes too costly or impractical, sometimes it even fails. Therefore, effects of sensor AC on the operation of feedback systems must be well understood and analyzed, hence the need for adequate simulation tools. It is also important to note that sensor failure recordings are not widely available and injecting sensor failures in controlled experiments for data acquisition is very challenging. Therefore, validation of sensor models under AC is difficult in a comprehensive manner. Sensor AC may be produced by a very large set of physical circumstances and phenomena that can materialize in a diversity of AC internal and external causes. From a modeling and simulation point of view and for monitoring and control purposes, the physical circumstances of sensor damages are, in most instances, of little concern and the focus is on the AC consequences, that is on their effects on and how they equivalently alter the measurement \( F[x(t)] \).

5.2 Functional Failure Categories

Sensor functional categories represent elements that affect the measurement, \( F[x(t)] \) and model the mismatch between it and the measurand, \( x(t) \). Some of these elements may be present under normal and abnormal operation while others may only be present under AC. The ones present under both normal and abnormal operation are marked (a) in the sensor functional categories list, while the ones only occurring during AC are marked (b) in the list.

Under normal conditions, the parameters that define sensor functional categories may take values inside acceptable ranges, typically provided by the sensor manufacturers. If these ranges are exceeded, then the situation becomes an AC. For example, certain levels of noise are acceptable in terms of distribution and standard deviation. Noise outside the design parameters would represent an AC.

Sensor functional categories are:

- Dead Band (a)
- Sensor Dynamics (a)
- Scaling (b)
- Bias (b)
- Other Error-producing Phenomena (a)
- Drift (a)
- Saturation (b)
- Dropout (b)
- Noise (a)
The proposed sequence of the sensor functional categories (FC) is shown in Figure 5.3. Note that the general sequence may not be unique, although some sequences of particular individual blocks are.

Sensor FCs typically depend on one or more specific parameters $p$ that may take different constant values over random or determined time intervals. Generally, for a parameter $p$ and a given sequence of time instances $t_1, t_2, \ldots, t_n$, the variation of $p$ may be expressed as follows:

$$p(t) = \begin{cases} p_1 & \text{for } 0 \leq t < t_1 \\ p_2 & \text{for } t_1 \leq t < t_2 \\ \vdots \\ p_n & \text{for } t_{n-1} \leq t \end{cases}$$

(5.5)

Note that $p_1$ is the initial or reference value of the parameter and $t_1$ represents the moment of occurrence of a MC or AC when the initial value of the sensor parameter is
modified. \( p_2 \) through \( p_n \) represent modified parameter values that can be inside or outside normal ranges.

The simulation setup will assign default values for all parameters. The user is able to change these initial values within or outside normal ranges. For simulating normal conditions or MC, values can only be assigned within normal ranges. For AC simulation, initial values must be outside normal ranges. However, the user may choose to start with default initial values or other normal values and then inject a MC or AC at \( t_1 \).

The modeling of all FCs is explained in sections 5.2.1-5.2.9.

5.2.1 Dead Band (FC #1)

The dead band or dead zone is a small range of input values for which the output is zero. Once the border value of the dead band is reached, that is equivalent to a step input to the system. For a sensor, assuming symmetric input and symmetric sensor behavior about zero, a dead band can be defined by one parameter \( x_{db} \geq 0 \), such that:

\[
y_1(t) = \begin{cases} 
0 & \text{for } -x_{db} < x(t) < x_{db} \\
C[x(t)] & \text{otherwise}
\end{cases}
\]

(5.6)

Where \( C[x(t)] \) transforms the measurand into direct sensor output. For example, for a thermocouple, this function represents the sensor internal phenomena that converts the temperature into its corresponding voltage. Note that \( x_{db} \) may take different values at different instances of time and so equation 5.6 can be written more generally as:

\[
y_1(t) = \begin{cases} 
0 & \text{for } -x_{db}(t) < x(t) < x_{db}(t) \\
C[x(t)] & \text{otherwise}
\end{cases}
\]

(5.7)

Where \( x_{db}(t) \) is similar to parameter \( p \) that varies according to equation 5.5. Figure 5.4 shows an example of an implemented dead band FC on a sinusoidal input for an \( x_{db} \) of 0.25 occurring after 25 seconds.
5.2.2 Sensor Dynamics (FC #2)

The internal dynamics of the sensor are typically modeled as a first or second order transfer function, possibly affected by a pure time delay ($T$). Note that sensor dynamics may be neglected for analysis purposes, case in which the transfer function is reduced to a gain (a zero-order transfer function). Therefore, in general, sensor dynamics may depend on five parameters and be expressed as:

$$G(s) = \frac{Y_2(s)}{Y_1(s)} = \frac{k}{a_1s^2 + a_2s + a_3}e^{-Ts} \quad (5.8)$$

By properly selecting the five parameters ($a_1, a_2, a_3, k, T$), all possible cases can be implemented as follows:

**Case 1:** No dynamics, only a gain, $K$:

$$\begin{cases} k = K \\ a_1 = a_2 = 0 \quad \text{and} \quad K \neq 0 \\ a_3 = 1 \end{cases} \quad (5.9)$$

**Case 2:** First order system (FOS) dynamics with gain, $K$ and time constant, $\tau > 0$:

$$\begin{cases} k = K \\ a_1 = 0 \\ a_2 = \tau \quad \text{and} \quad K \neq 0, \tau > 0 \\ a_3 = 1 \end{cases} \quad (5.10)$$
Case 3: Second order system (SOS) dynamics with gain $K$, damping $\zeta$ and natural frequency, $\omega_n$, for $0 < \zeta < 1$ and $\omega_n > 0$:

\[
\begin{align*}
    k &= K \omega_n^2 \\
    a_1 &= 1 \\
    a_2 &= 2\zeta \omega_n \\
    a_3 &= \omega_n^2 \\
\end{align*}
\]

and $K \neq 0$, $0 < \zeta < 1$, $\omega_n > 0$ (5.11)

For all cases, a pure time delay may be added as follows:

\[
\begin{align*}
    T &= 0 \quad \text{no delay} \\
    T &> 0 \quad \text{delay in seconds} \\
\end{align*}
\]

(5.12)

Figure 5.5-Figure 5.7 show an example of the system response of all three cases to a step input at 10 seconds. Case 1 has a gain of 1.5 and no delay ($T = 0$). Case 2 has unitary gain and a time constant of 1.25 seconds with a 5 second delay. Case 3 has a gain of 4, a damping factor of 0.25 and a natural frequency of 2 rad/s.
Figure 5.6: Sensor Dynamics FC (Case 2) Example

Figure 5.7: Sensor Dynamics FC (Case 3) Example
5.2.3 Scaling (FC #3)

The third FC affects the sensor output by a scale factor $\sigma$, which may be constant or variable in time according to equation 5.5. An example with a constant scaling factor of 1.2 injected at $t = 12$ seconds is presented in Figure 5.8.

$$y_3(t) = \sigma(t) \cdot y_2(t) \tag{5.13}$$

![Figure 5.8: Scaling FC Example](image)

5.2.4 Bias (FC #4)

Sensor bias represents a sensor output offset with respect to the measurand. Typically, it is the constant output produced when the input is zero [29]. Bias may be eliminated through calibration. However, note that over larger periods of time, bias may vary (see equation 5.5) and re-calibration is necessary. Between calibrations, a certain amount of bias may be considered acceptable as “normal operation” in some instances. A bias can be modeled as shown in equation 5.14, where $\beta$ represents the bias value. An example can be found in Figure 5.9 with a bias value of 0.5 starting at 20 seconds.

$$y_4(t) = y_3(t) + \beta(t) \tag{5.14}$$
5.2.5 Other Error-producing Phenomena (FC #5)

Sensor specification may identify particular sources of measurement errors and provide information regarding the magnitude of these errors, such as non-linearity errors and hysteresis errors. Considering this FC requires customized modeling of an additive error $e(t)$, with a binary switch, $k_e$, such that:

$$y_5(t) = y_4(t) + k_e e(t)$$  \hspace{1cm} (5.15)

Where:

$$k_e = \begin{cases} 
0 & \text{no additional errors considered} \\
1 & \text{additional errors modeled and considered}
\end{cases}$$

5.2.6 Drift (FC #6)

The drift FC represents the variation with constant low rate $\delta$ of the sensor output when sensor is exposed to constant input [29]. This variation is additive with respect to input, such that:

$$y_6(t) = y_5(t) + \delta \cdot t$$  \hspace{1cm} (5.16)

In some instances, the drift, $\delta$ may change at certain moments in time, such that, more generally:
\[ y_6(t) = y_5(t) + \Delta(t) \]  

(5.17)

Where:

\[ \Delta(t) = \begin{cases} 
\delta_1 \cdot t & \text{for } 0 \leq t < t_1 \\
\delta_2 \cdot (t - t_1) & \text{for } t_1 \leq t < t_2 \\
\vdots & \\
\delta_n \cdot (t - t_{n-1}) & \text{for } t_{n-1} \leq t 
\end{cases} \]  

(5.18)

Note that equation 5.18 is for a linear drift case and a non-linear drift case may also be possible [59]. An example of a linear drift is shown in Figure 5.10 for a constant drift of 0.01 starting 15 seconds into the simulation.

5.2.7 Saturation (FC #7)

A sensor’s output is limited to a certain range due to circumstances like the amount of available power supply or other physical constraints. For an implementation that allows the user to remove saturation for analysis purposes, a binary trigger \( k_s \) must be introduced. The logical opposite of \( k_s \) can be denoted as \( \tilde{k}_s \), so when \( k_s = 0 \), then \( \tilde{k}_s = 1 \) and when \( k_s = 1 \), \( \tilde{k}_s = 0 \). Therefore, the scenario that considers saturation will set \( k_s = 1 \), \( \tilde{k}_s = 0 \) and the scenario that ignores saturation will have \( k_s = 0 \), \( \tilde{k}_s = 1 \). With these considerations, the general formulation of the saturation FC can be expressed using
equation 5.19. An example of this FC is shown in Figure 5.11 for an upper and lower limit of -0.5 and 0.5 respectively.

\[
y_7(t) = \begin{cases} 
    k_s \cdot y_6(t) + k_s \cdot y_{\text{max}} & \text{if } y_6(t) \geq y_{\text{max}} \\
    y_6(t) & \text{if } y_{\text{min}} < y_6(t) < y_{\text{max}} \\
    k_s \cdot y_6(t) + k_s \cdot y_{\text{min}} & \text{if } y_6(t) \leq y_{\text{min}} 
\end{cases}
\] (5.19)

\[
y_8(t) = \begin{cases} 
    y_7(t) & \text{for } t < t_D \\
    k_D \cdot D + \k_D \cdot y_7(t) & \text{for } t \geq t_D 
\end{cases}
\] (5.20)

Where:

5.2.8 Dropout (FC #8)

A dropout occurs when a sensor output is stuck at certain constant value, \(D\). Loss of signal is one particular case when sensor output is stuck at zero. The case when the output is stuck at either limit of the output range is different from saturation because of two aspects. First, dropout may occur for any actual value of the input, while saturation is triggered by the input exceeding the range limit. Secondly, with dropout, the output will remain at the extreme value regardless of the input, while with saturation, output will follow the input once the input is back within the operational range, except for the case when saturation breaks down the sensor. If \(t_D\) is the time when dropout occurs and \(k_D\) is a dropout failure binary trigger, then the dropout mathematical model can be defined as follows:

\[
y_8(t) = \begin{cases} 
    y_7(t) & \text{for } t < t_D \\
    k_D \cdot D + \k_D \cdot y_7(t) & \text{for } t \geq t_D 
\end{cases}
\] (5.20)
\[ k_D = \begin{cases} 
0 \text{ (and } \bar{k_D} = 1) & \text{no dropout failure} \\
1 \text{ (and } \bar{k_D} = 0) & \text{dropout failure is simulated} 
\end{cases} \quad (5.21) \]

The value of the dropout, \( D \), may vary with respect to time and can be modeled using equation 5.22. A dropout example is simulated in Figure 5.12 where the simulated dropout starts at \( t = 25 \) seconds.

A dropout example is simulated in Figure 5.12 where the simulated dropout starts at \( t = 25 \) seconds.

\[
y(t) = \begin{cases} 
y_7(t) & \text{for } t < t_{D1} \\
D_1 & \text{for } t_{D1} \leq t < t_{D2} \\
D_2 & \text{for } t_{D2} \leq t < t_{D3} \\
\vdots & \\
D_n & \text{for } t_{Dn} \leq t 
\end{cases} \quad (5.22) \]

\[
\text{System Input}
\]

\[
\text{System Output for a Dropout Functional Category}
\]

**Figure 5.12: Dropout FC Example**

### 5.2.9 Noise (FC #9)

Measurement noise, \( \nu(t) \) can be produced by a variety of sensor internal and external factors. It can typically be modeled as a random variable with zero mean Gaussian or normal distribution. A Gaussian distribution is expressed in terms of its probability density function (PDF) as:

\[
PDF(x, \mu, \sigma_D) = \frac{1}{\sqrt{2\pi\sigma_D^2}} e^{-\frac{(x-\mu)^2}{2\sigma_D^2}} \quad (5.23)
\]

Where:

\[
\begin{align*}
\mu & \text{ mean of the Gaussian distribution} \\
\sigma_D & \text{ standard deviation of the Gaussian distribution} \\
\end{align*}
\]

52
\[ \mu - \text{mean} \]
\[ \sigma_D - \text{standard deviation of random variable, } x \]

Note that a non-zero mean is equivalent to a bias.

Let \( k_n \) be a binary switch that allows the user to consider sensor noise (\( k_n = 1 \)) or ignore it (\( k_n = 0 \)). Then a mathematical model for the noise FC can be expressed as in equation 5.24. An example of a sensor measurement affected by noise with \( \mu = 0 \) (zero mean) and \( \sigma_D = 0.1 \) occurring throughout is presented in Figure 5.13.

\[ y_9(t) = y_8(t) + k_n \cdot v(t) \] (5.24)

5.2.10 Conversion

Conversion to units of the measurand is expected to model the inverse internal relationship between sensor input and output. This can be expressed as in equation 5.25.

\[ m(t) = C^{-1}[y_9(t)] \] (5.25)

Note that, for modeling and simulation purposes, this conversion cancels the one at the beginning of the block diagram. As a consequence, one can simply consider that \( C[x(t)] = x(t) \) and throughout the model all \( y_i(t) \) formally become \( m_i(t) \).
5.3 MATLAB/Simulink Implementation

For the sensor failure modeling in MATLAB/Simulink, all FCs are modeled in subsystems using Simulink’s mathematical operations and other built-in functions. A matrix of flags were pre-defined in MATLAB to set up the simulation scenario as expressed in equation 5.26.

\[
F_{\text{flags}} = [F_{\text{AC}} \ F_{\text{time}} \ F_{\text{mag}} \ F_{\text{FC}} \ F_{\text{case}_2}] \quad (5.26)
\]

The \(F_{\text{AC}}\) flag is used to define the occurrence of a failure at MC, AC or reference (default) values. In other words, this flag makes the difference between sensors that are at initial conditions within normal ranges and those that are not. The \(F_{\text{AC}}\) flag is defined as follows:

\[
F_{\text{AC}} = \begin{cases} 
0 & \text{All parameters at "reference" or default values} \\
1 & \text{Parameters are modified within normal ranges and constant} \\
2 & \text{Parameters are modified (AC or MC)}
\end{cases} \quad (5.27)
\]

The \(F_{\text{time}}\) flag establishes the occurrence of the failure. It determines whether or not the initial or modified failure values occur throughout the failure simulation. For failures not occurring throughout the simulation, the failures are re-instated at times pre-defined by the user. Note that the \(F_{\text{time}}\) flag is only active if \(F_{\text{AC}} = 2\). The flag is defined as follows:

\[
F_{\text{time}} = \begin{cases} 
0 & \text{for permanent AC or MC} \\
1 & \text{for intermittent AC or MC}
\end{cases} \quad (5.28)
\]

\(F_{\text{mag}}\) determines whether or not initial values are maintained constant or they change at certain instants during the simulation. In other words, these flags make the difference between constant and variable ACs and MCs. \(F_{\text{mag}}\) is set up as follows:

\[
F_{\text{mag}} = \begin{cases} 
0 & \text{for constant values of all parameters} \\
1 & \text{for variable values of at least one parameter}
\end{cases} \quad (5.29)
\]

The \(F_{\text{FC}}\) flag is used to select which functional category is implemented for the failed sensor. Note that for the general case where all sensors may have all FCs affected, the \(F_{\text{FC}}\) flag forms an \(n \times 9\) matrix. For the case where only one FC may be affected for any given sensor, the flag forms an \(n\)-element column vector. The flag is defined as:

\[
F_{\text{FC}} = \begin{cases} 
0 & \text{no FC affected (no AC or MC)} \\
1 & \text{FC#1 (Dead Band) affected} \\
2 & \text{FC#2 (Sensor Dynamics) affected} \\
\vdots \\
9 & \text{FC#9 (Noise) affected}
\end{cases} \quad (5.30)
\]
The $F_{case_2}$ flag is used to select the case to be simulated by the user in the sensor dynamics functional category and it is defined as follows:

$$F_{case_2} = \begin{cases} 1 & K \ (pure \ gain) \\ 2 & FOS \\ 3 & SOS \end{cases} \quad (5.31)$$

The initial reference values for all functional categories are defined as follows:

- $x_{db} = 0$ No dead band
- $k = 1, a_1 = a_2 = 0, a_3 = 1$ No dynamics
- $T = 0$ No time delay
- $\beta = 0$ No bias
- $k_e = 0, e(t) = 1$ No other errors
- $\sigma = 0$ No scaling
- $\delta = 0$ No drift
- $k_s = 1, y_{min} = -\infty, y_{max} = \infty$ No saturation
- $k_D = 0$ No dropout
- $k_n = 0$ No noise

### 5.3.1 Modeling of Sensor Abnormal Condition Consistency

The permanent or intermittent occurrence of a failure is modeled as a repeating sequence in Simulink. The repeating sequence consists of two vectors; the time and output vectors. Both are defined by the user and each time value has a corresponding output value. The output vector is binary, so its values can only be 0 or 1. It determines whether a failure is off or not, where:

$$Output \ vector \ value = \begin{cases} 0 & Failure \ present \\ 1 & Failure \ absent \end{cases} \quad (5.32)$$

An example of an intermittent failure is shown in Figure 5.14 for a dropout FC. A permanent dropout FC can be found in Figure 5.12 for comparison. Compared to Figure 5.12, the dropout can be seen to occur between 25 and 42 seconds, 65 and 122 seconds and from 145 seconds onwards. At other time intervals, the dropout is not present.
The failure magnitude is also modeled using a repeating sequence block in Simulink. The output vector for the failure magnitude represents an additive modification of the already defined FC parameter. A negative output vector value will result in a subtraction from the user-provided parameter value and vice versa. An example of a varied magnitude failure is shown in Figure 5.15 for a scaling FC. Figure 5.8 can be used as a reference for the response of a constant magnitude scaling FC. In Figure 5.8, the scale factor of 1.2 can be seen to yield a constant measurement after the failure kicks in, but in Figure 5.15, the measurement varies depending on the additional scale factor modification present.
The time and magnitude of a FC can also be intermittent and varied respectively and an example of this is shown for a bias FC in Figure 5.16. Compared to the constant bias example in Figure 5.9, it can be seen in Figure 5.16 that the magnitude of the bias is varied between 40 and 60 seconds. One can also notice the bias FC not being present between the 85 and 90 second intervals.
CHAPTER 6: DESIGN OF ABNORMAL CONDITIONS SIMULATION TESTS

The WVU UAS simulation environment supports the simulation of abnormal conditions affecting different aircraft components and subsystems. The abnormal conditions simulated in this thesis are explained in this chapter. The purpose of these tests is to illustrate the use of the sensor failure model and investigate the effects of the functional categories on the performance of the trajectory tracking algorithm. The generated results can be useful to the improvement of fault tolerant control laws for UAVs. For this research effort, abnormal conditions on the sensor and GPS subsystems are considered and are explained in sections 6.1 and 6.2 below. The performance of the trajectory tracking controller will be evaluated using the performance metrics discussed in section 3.5.

6.1 Sensor Abnormal Conditions

Using the developed aircraft and sensor model, newer sensor abnormal conditions can be simulated in the WVU UAS simulation environment. All eight FCs developed are used to simulate abnormal conditions affecting the measured angular rates of the aircraft \((p, q\) and \(r))\), for the four trajectories shown in section 3.4.1, at low and high severities. Before the FC parameters can be defined, the commanded angular rate limits for all trajectories need to be known. This is necessary because they will be used as a guideline for the selection of the failure parameters. The angular rate limits for all trajectories can be found in Table 6.1 where all values are in deg/s.

<table>
<thead>
<tr>
<th>Trajectory</th>
<th>Square</th>
<th>Oval</th>
<th>Figure 8</th>
<th>Climbing S Turns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Roll rate ((p))</td>
<td>-30 28</td>
<td>-30 29</td>
<td>-30 30</td>
<td>-30 29</td>
</tr>
<tr>
<td>Pitch rate ((q))</td>
<td>0 7</td>
<td>0 7</td>
<td>0 7</td>
<td>0 12</td>
</tr>
<tr>
<td>Yaw rate ((r))</td>
<td>-13 5</td>
<td>-13 5</td>
<td>-13 13.5</td>
<td>-13 12</td>
</tr>
</tbody>
</table>

Knowing the limits, the failure parameters can be defined. For the sensor dynamics FC, two cases are simulated; a time constant, \(\tau\) and time delay, \(T\). All but the saturation simulation parameters are defined in Table 6.2 for low and high FC severities. Note that the dropout only has one severity where the output goes to 0. The parameters used for the saturation FC are dependent on the angular rate limits of the simulated trajectories and are defined in Table 6.3-Table 6.6 for all trajectories. The upper and lower saturation limits are defined to be equal and opposite because the angular rates are also equal and opposite depending on the direction of the performed maneuver.
### Table 6.2: Sensor Abnormal Conditions Parameters

<table>
<thead>
<tr>
<th>Functional Category</th>
<th>Low Severity</th>
<th>High Severity</th>
<th>Failure Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dead Band</td>
<td>$x_{db} = 4$ deg/s</td>
<td>$x_{db} = 10$ deg/s</td>
<td>5 seconds</td>
</tr>
<tr>
<td>Sensor Dynamics (Case 1)</td>
<td>$\tau = 0.5$ s</td>
<td>$\tau = 2.5$ s</td>
<td>5 seconds</td>
</tr>
<tr>
<td>Sensor Dynamics (Case 2)</td>
<td>$T = 0.05$ s</td>
<td>$T = 0.2$ s</td>
<td>5 seconds</td>
</tr>
<tr>
<td>Scaling</td>
<td>$\sigma = 1.5$</td>
<td>$\sigma = 4$</td>
<td>5 seconds</td>
</tr>
<tr>
<td>Bias</td>
<td>$\beta = 4$ deg/s</td>
<td>$\beta = 10$ deg/s</td>
<td>5 seconds</td>
</tr>
<tr>
<td>Drift</td>
<td>$\delta = 0.15$</td>
<td>$\delta = 0.5$</td>
<td>5 seconds</td>
</tr>
<tr>
<td>Dropout</td>
<td>Only one severity</td>
<td>Only one severity</td>
<td>5 seconds</td>
</tr>
<tr>
<td>Noise</td>
<td>$\sigma_p = 1.6$</td>
<td>$\sigma_p = 6$</td>
<td>0 seconds</td>
</tr>
</tbody>
</table>

### Table 6.3: Saturation Parameters for Square Trajectory

<table>
<thead>
<tr>
<th>Saturation</th>
<th>Low Severity</th>
<th>High Severity</th>
<th>Failure Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Roll rate (p)</td>
<td>$y_{lim} = \pm 25$ deg/s</td>
<td>$y_{lim} = \pm 10$ deg/s</td>
<td>0 seconds</td>
</tr>
<tr>
<td>Pitch rate (q)</td>
<td>$y_{lim} = \pm 2$ deg/s</td>
<td>$y_{lim} = \pm 3$ deg/s</td>
<td>0 seconds</td>
</tr>
<tr>
<td>Yaw rate (r)</td>
<td>$y_{lim} = \pm 10$ deg/s</td>
<td>$y_{lim} = \pm 2$ deg/s</td>
<td>0 seconds</td>
</tr>
</tbody>
</table>

### Table 6.4: Saturation Parameters for Oval Trajectory

<table>
<thead>
<tr>
<th>Saturation</th>
<th>Low Severity</th>
<th>High Severity</th>
<th>Failure Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Roll rate (p)</td>
<td>$y_{lim} = \pm 25$ deg/s</td>
<td>$y_{lim} = \pm 10$ deg/s</td>
<td>0 seconds</td>
</tr>
<tr>
<td>Pitch rate (q)</td>
<td>$y_{lim} = \pm 2$ deg/s</td>
<td>$y_{lim} = \pm 3$ deg/s</td>
<td>0 seconds</td>
</tr>
<tr>
<td>Yaw rate (r)</td>
<td>$y_{lim} = \pm 8$ deg/s</td>
<td>$y_{lim} = \pm 2$ deg/s</td>
<td>0 seconds</td>
</tr>
</tbody>
</table>

### Table 6.5: Saturation Parameters for Figure 8 Trajectory

<table>
<thead>
<tr>
<th>Saturation</th>
<th>Low Severity</th>
<th>High Severity</th>
<th>Failure Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Roll rate (p)</td>
<td>$y_{lim} = \pm 25$ deg/s</td>
<td>$y_{lim} = \pm 10$ deg/s</td>
<td>0 seconds</td>
</tr>
<tr>
<td>Pitch rate (q)</td>
<td>$y_{lim} = \pm 1$ deg/s</td>
<td>$y_{lim} = \pm 2$ deg/s</td>
<td>0 seconds</td>
</tr>
<tr>
<td>Yaw rate (r)</td>
<td>$y_{lim} = \pm 8$ deg/s</td>
<td>$y_{lim} = \pm 2$ deg/s</td>
<td>0 seconds</td>
</tr>
</tbody>
</table>

### Table 6.6: Saturation Parameters for Climbing S Turns Trajectory

<table>
<thead>
<tr>
<th>Saturation</th>
<th>Low Severity</th>
<th>High Severity</th>
<th>Failure Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Roll rate (p)</td>
<td>$y_{lim} = \pm 25$ deg/s</td>
<td>$y_{lim} = \pm 10$ deg/s</td>
<td>0 seconds</td>
</tr>
<tr>
<td>Pitch rate (q)</td>
<td>$y_{lim} = \pm 4$ deg/s</td>
<td>$y_{lim} = \pm 1$ deg/s</td>
<td>0 seconds</td>
</tr>
<tr>
<td>Yaw rate (r)</td>
<td>$y_{lim} = \pm 8$ deg/s</td>
<td>$y_{lim} = \pm 2$ deg/s</td>
<td>0 seconds</td>
</tr>
</tbody>
</table>
6.2 GPS Abnormal Conditions

For the GPS subsystem, an analysis is performed on the effectiveness of the trajectory tracking controller used when a dropout occurs affecting the GPS position, velocity and both of them at the same time. The GPS abnormal condition is only simulated for the oval and figure 8 trajectories. The dropout is injected 5 seconds from the start of the simulation. A study is performed on the maximum threshold the dropout can accommodate while still returning to the commanded trajectory. One important thing to note is the output signal from the GPS when the dropout occurs is not zero, but corresponds to the last known GPS position or velocity.
CHAPTER 7: RESULTS

In this chapter, the performance of the UAV under nominal and the abnormal conditions discussed in Chapter 6: is analyzed. The P PID controller discussed in section 3.4.2 is used for the trajectory tracking of the UAV in all simulations. To evaluate the controller’s performance, the metrics discussed in section 3.5 are used and the UAV used is the modeled Phastball. Turbulence, wind gusts and other adverse atmospheric conditions were not considered in these simulations.

7.1 Simulation Results

7.1.1 Nominal Conditions

The first set of simulations were performed under nominal conditions to understand how well the controller used would perform in tracking the commanded trajectory and how its generated commands would affect the UAV angular rates. Under nominal conditions, the trajectory tracking algorithm was able to generate commands to the UAV to follow the commanded trajectory well enough. Figure 7.1-Figure 7.4 show the commanded and actual trajectories under nominal conditions for all simulated trajectories.

It is also important to know the response of the UAV angular rates under nominal conditions in order to understand the response of the UAV to the P PID controller under both nominal and abnormal conditions. The angular rates for all trajectories are shown in Figure 7.5-Figure 7.8.

Figure 7.1: Square Trajectory – Nominal Conditions
Figure 7.2: Oval Trajectory – Nominal Conditions

Figure 7.3: Figure 8 Trajectory – Nominal Conditions
Figure 7.4: Climbing S Turns Trajectory – Nominal Conditions

Figure 7.5: Angular Rates for Square Trajectory – Nominal Conditions
Figure 7.6: Angular Rates for Oval Trajectory – Nominal Conditions

Figure 7.7: Angular Rates for Figure 8 Trajectory – Nominal Conditions
7.1.2 Sensor Failure

The gyro sensor failure was simulated affecting the roll, pitch and yaw gyros, using all trajectories. After performing the simulations, it was noticed that the failure parameters defined in Chapter 6 had some effects on the roll and pitch gyro sensor failure simulation results, but did not have a significant effect when the yaw gyro sensor was failed. This could be due to several factors including the failure parameters, the class of the controller used and even the commanded trajectory. For example, the scaling FC with the same high severity parameters using the oval trajectory had more of an effect when the roll rate sensor was failed compared to when same failure was implemented for the pitch and yaw rate sensors. Although a bit of an effect can be seen on the angular rate for the pitch rate gyro failure 5 seconds into the simulation, there is no noticeable change in the actual angular rates of the aircraft when the yaw sensor gyro is failed. The nominal and failed angular rates are shown in Figure 7.9–Figure 7.11. In Figure 7.9, it can be seen that the measured angular rates depart significantly from the nominal values in the case of the roll gyro failure, but in Figure 7.10 and Figure 7.11 for the pitch and yaw gyros, the nominal and failed angular rates case match up fairly well. With the yaw gyro failure being showing the least effect, a sensitivity analysis is performed in section 7.3 to provide more understanding on the effects of the FCs on the simulations and controller performance.
Figure 7.9: Angular Rates for a Scaling FC Affecting the Roll Rate Sensor (Oval Trajectory)

Figure 7.10: Angular Rates for a Scaling FC Affecting the Pitch Rate Sensor (Oval Trajectory)
All other simulations displayed a trend similar to that seen above apart from the dropout FC which did not affect the actual trajectory regardless of the failure affecting any of the three sensors. The sensor failure results are discussed more in-depth using the PIs in section 7.2.2.

7.1.3 GPS Failure

For the GPS abnormal conditions, the figure 8 and oval trajectories were used. For all simulations, the failure was implemented at the 5 second mark and a threshold was defined to perform an analysis on how long the aircraft could recover to the commanded trajectory. Note that the case of the dropout affecting one signal at a time is for analysis purposes to identify the corresponding effects and not for simulating actual scenarios.

For the figure 8, aircraft was able to recover to the commanded trajectory for a maximum position dropout threshold of 13.8 seconds. Figure 7.12 and Figure 7.13 show the commanded vs actual trajectories for a dropout lasting 5 and 10 seconds. It can be seen that the position error increases as the threshold of the dropout increases. Although the UAV knows its velocity in this instance, this information is actually detrimental to its completion of the commanded trajectory, as the known velocity does not correspond to the known position causing the generated commands to the UAV to be inaccurate. Figure 7.14 and Figure 7.15 show the actual and failed trajectories for the position dropout lasting...
for 13.8 and 13.9 seconds. For a dropout lasting 13.9 seconds, the aircraft cannot return to the commanded trajectory anymore.

*Figure 7.12: GPS Position Dropout for 5 Seconds (Figure 8)*

*Figure 7.13: GPS Position Dropout for 10 Seconds (Figure 8)*
When the GPS’ velocity dropout was implemented, the aircraft was able to return to the commanded trajectory for a longer dropout threshold than in the case of the position dropout. The maximum recoverable threshold was found to be 45.4 seconds, at least 3 times longer than that of the position dropout. Figure 7.16-Figure 7.19 show the commanded and actual trajectories for the velocity dropout case for 5, 10, 45.4 and 45.5 seconds.
Figure 7.16: GPS Velocity Dropout for 5 Seconds (Figure 8)

Figure 7.17: GPS Velocity Dropout for 10 Seconds (Figure 8)
For the position and velocity dropout, the aircraft was able to return to the commanded trajectory after a longer dropout duration of 33.2 seconds as compared to the position-only dropout. However, this threshold value was lower than that reached in the velocity-only dropout case. The maximum recoverable threshold was found to be 33.2 seconds. Position and velocity dropouts lasting 5, 10, 33.2 and 33.3 seconds can be seen in Figure 7.20-Figure 7.23. The aircraft is unable to return to the commanded trajectory for a dropout lasting longer than 33.2 seconds.
Figure 7.20: GPS Position and Velocity Dropout for 5 Seconds (Figure 8)

Figure 7.21: GPS Position and Velocity Dropout for 10 Seconds (Figure 8)
The results for the oval trajectory dropout scenario followed the trend seen in the case of the figure 8 trajectory. For the case where the position signal was dropped out, the aircraft was able to recover after a maximum dropout duration of 13.8 seconds, exactly the same as the corresponding case for the figure 8 trajectory. Figure 7.24-Figure 7.27 show the commanded and actual trajectories for the dropout lasting 5, 10, 13.8 and 13.9 seconds.
**Figure 7.24: GPS Position Dropout for 5 Seconds (Oval)**

**Figure 7.25: GPS Position Dropout for 10 Seconds (Oval)**
For the velocity dropout, the aircraft was able to return to the trajectory after a maximum dropout duration of 28.3 seconds, much lower than the corresponding result for the case for the figure 8 trajectory. The aircraft also lost altitude as the dropout duration increased. Figure 7.28-Figure 7.31 show the velocity dropout profile for a 5, 10, 28.3 and 28.4 seconds duration.
Figure 7.28: GPS Velocity Dropout for 5 Seconds (Oval)

Figure 7.29: GPS Velocity Dropout for 10 Seconds (Oval)
Just as in the case of the figure 8 trajectory, the aircraft was able to return to the commanded trajectory after the longest dropout duration when the GPS position and velocity signals were dropped out. The aircraft could recover after a maximum dropout lasting 34.8 seconds, much higher than when the position signal was dropped out and slightly higher than the case of the velocity dropout. The commanded vs actual trajectories for the position and velocity dropouts lasting 5, 10, 34.8 and 34.9 seconds can be seen in Figure 7.32-Figure 7.35.
Figure 7.32: GPS Position and Velocity Dropout for 5 Seconds (Oval)

Figure 7.33: GPS Position and Velocity Dropout for 10 Seconds (Oval)
7.2 Performance Evaluation

The performance of the PPID controller under nominal, sensor failure and GPS dropout conditions using the metrics defined in section 3.5 for the trajectory tracking algorithm accuracy and control surface activity is evaluated in this section. In some instances for the abnormal conditions, some performance indices were analyzed individually to understand how they are affected by the simulated failure.
7.2.1 Nominal Conditions

Under nominal conditions, the performance of the PPID controller was evaluated for all four trajectories. The controller was found to perform well in tracking all the square, oval and figure 8 trajectories and it performed fairly well in tracking the climbing S turns trajectory. The square trajectory had a trajectory tracking PI of about 0.82 and a control activity PI of approximately 1. The total PI was calculated to be just under 0.9, illustrating good performance by the controller. The controller was penalized for the mean XY and mean XYZ errors between the commanded and actual trajectory being out of the specified limits, although this error was not too large and did not hurt the PI a lot. The PI for the square trajectory is shown in Figure 7.36.

![Trajectory Tracking Performance Index Contributions](image1)

![Control Activity Performance Index Contributions](image2)

![Overall Performance Index Contributions](image3)

*Figure 7.36: Square Trajectory PI- Nominal Conditions*

The oval trajectory performed similarly to the PI of the square trajectory with them having similar performance indices, although, the mean XY and XYZ errors were slightly larger in the oval trajectory than in the square trajectory. This can be said to be due to more turns performed in this trajectory than for the square trajectory. The oval trajectory’s PI under nominal conditions can be seen in Figure 7.37 with a trajectory tracking PI of 0.8 and a total PI of just below 0.9, similar to that of the square trajectory.
For the figure 8 trajectory, the mean XY and XYZ errors were higher compared to that of the square trajectory, but similar to that of the oval trajectory, with both of them having similar PIs. The trajectory tracking PI was approximately 0.8, with the control activity PI being slightly less than 1 and the total PI just below 0.9, as shown in Figure 7.38.
The climbing S turn’s trajectory had the worst PI under nominal conditions with a trajectory tracking PI just above 0.4. The PI of its control activity was high, having a value close to 1, but with the trajectory tracking PI regarded as more important in the weighting criteria, it brought down the total PI to about 0.6, much lower than that of the other three trajectories. The controller was penalized significantly for large XY and XYZ mean and standard deviation errors which were way off with the controller having a PI of 0 for the aforementioned indices. The climbing S turns PI can be seen in Figure 7.39.

![Trajectory Tracking Performance Index Contributions](image1)

![Control Activity Performance Index Contributions](image2)

![Overall Performance Index Contributions](image3)

*Figure 7.39: Climbing S Turns PI – Nominal Conditions*

### 7.2.2 Sensor Failure

To evaluate the sensor failure simulations, the trajectory tracking and control activity indices are used. The first index is chosen because it shows the controller’s ability to follow the commanded trajectory, while being within the specified thresholds. The second one represents the overall activity on the control surfaces and throttle lever due to the commands generated by the controller. The performance evaluation is performed for all trajectories, but only for the roll and pitch angular rate sensors. The PIs for the yaw rate sensors can be found in Appendix A. All functional categories simulated are compared at low and high severities with respect to the nominal case. Note that for the dropout one severity is applicable (signal drops to 0).

For the square trajectory, the high severity drift FC had the worst trajectory tracking PI with a value close to 0.77. Interestingly, the dropout case had a trajectory tracking PI equal to that of the square trajectory under nominal conditions. This can be said to be due to the
Few banks performed by the aircraft and a dropout in this case means the output equals 0, for a square trajectory having straight segments for the most part, this should not be a problem for the controller. The dead band, time constant, bias, saturation and noise functional categories all had a trajectory tracking index equal or slightly higher than nominal for both low and high severities showing the controllers ability to handle to abnormal condition. The control activity PI was not significantly affected for the failure affecting the roll rate sensor apart from the sensor dynamics FC time delay case, which affected this index by about 15% for the low severity cases and a bit over 20% for the high severity case. The other FC that affected the control activity index somewhat the noise. This FC dropped the index by about 5% for the low severity case and about 3% more for the high severity case. The trajectory tracking and control activity indices for the square trajectory affecting the roll rate sensor is shown in Figure 7.40 and Figure 7.41.

![Figure 7.40: Trajectory Tracking Index for Square Trajectory (Roll Rate Sensor Failure)](image-url)
For the presence of abnormal conditions on pitch rate sensor gyro, the drift FC was found to have the worst effect on the trajectory tracking index, with the index dropping from about 0.82 (nominal and low severity) to almost 0 for the high severity case. The high severity time delay, scaling and bias failures affected the trajectory tracking index a bit, but their effects were not as significant as that of the high severity drift. For the control activity, the high severity time delay had the most effect, dropping the index by almost 20%. The high severity noise and drift also had a mild effect on this index. The trajectory tracking and control activity indices for the square trajectory for the pitch rate sensor failure is in Figure 7.42 and Figure 7.43.
For the oval trajectory, the high severity time delay and scaling FC had the worst trajectory tracking index dropping by less than 0.5 from the nominal value. All other abnormal conditions had an index equal to or slightly better than the nominal value, with the high severity bias and drift FCs having a higher index than the low severity case, although the difference was not as much. The dropout FC also had a trajectory tracking index equal to that of the nominal case. The trajectory tracking index for the oval trajectory for a roll rate sensor failure is presented in Figure 7.44. For the control activity PI (Figure 7.45), the low and high severity time delay case again had the worst index, dropping it by about 15 and 20% from nominal. The noise also dropped the index a bit.

Figure 7.43: Control Activity Index for Square Trajectory (Pitch Rate Sensor Failure)

Figure 7.44: Trajectory Tracking Index for Oval Trajectory (Roll Rate Sensor Failure)
Figure 7.45: Control Activity Index for Oval Trajectory (Roll Rate Sensor Failure)

When the pitch rate sensor gyro was failed for the oval trajectory, the high severity drift FC once again had the worst effect on the trajectory tracking index dropping it from a nominal value of about 0.82 to 0.2. The high severity time delay and bias also had a bit of an effect on this index as seen in Figure 7.46. The control activity performance index (Figure 7.47) was the worst once again when the high severity time delay was simulated and dropped a bit when noise was added to the measurement.

Figure 7.46: Trajectory Tracking Index for Oval Trajectory (Pitch Rate Sensor Failure)
When the roll sensor was failed for the figure 8 trajectory, the high severity drift AC had the worst trajectory tracking index dropping it by approximately 15%. The high severity time delay, scaling and bias failures dropped the index a bit but not as much as the high severity drift AC. The trajectory tracking indices for the figure 8 trajectory for the roll rate sensor failure can be seen in Figure 7.48. On the other hand, for the control activity index, the low and high severity time delay cases still exhibited the approximate 15 and 20% decrease in the index and the addition of noise also affected the control activity index somewhat. The control activity PI is shown in Figure 7.49.
Figure 7.49: Control Activity Index for Figure 8 Trajectory (Roll Rate Sensor Failure)

For the same trajectory, trajectory tracking index when the pitch rate sensor gyro failure was implemented was most sensitive to the high severity drift AC. This AC dropped the index from a nominal value of about 0.82 to a value of 0.1. The high severity time delay also had a small effect on the index as well as the high severity scaling and bias ACs. The high severity time delay AC affected the control activity index the most with the presence of noise in the measurement also having a bit of an effect on the index. The trajectory tracking and control activity indices can be found in Figure 7.50 and Figure 7.51.

Figure 7.50: Trajectory Tracking Index for Figure 8 Trajectory (Pitch Rate Sensor Failure)
Finally, for the climbing S trajectory, no trend was found for the trajectory tracking index, as all the values had an index similar to the nominal case. The trajectory tracking index is shown in Figure 7.52. For the control activity, the time delay and noise ACs still followed the same trend, dropping the index by approximately 15 and 20% for the low and high severity time delay cases and 5 and 8% for the respective noise AC. The control activity index is shown in Figure 7.53.
When the pitch rate sensor gyro was failed for the same trajectory, the high severity drift AC once again dropped the trajectory tracking index to about 0.05, from a nominal value of about 0.43. The high severity time delay, scaling and bias ACs also affected the index a bit but lightly compared to the effect of the aforementioned drift. The control activity index exhibited the same trend as before and was affected by the time delay and noise ACs, but only dropped for the high severity time delay AC while dropping a bit for the low and high noise severities. The trajectory tracking and control activity indices can be seen in Figure 7.54 and Figure 7.55.
Upon further analysis of the drift AC affecting the pitch rate sensor, it was found that one the drift gets too large, the aircraft loses altitude which causes the drop in the trajectory tracking PI. The failed oval trajectory can be seen in Figure 7.56 for a high severity drift AC affecting the pitch rate gyro.
The drift AC was found to have a threshold where the trajectory tracking controller could tolerate the failure and beyond the threshold, the AC would affect the trajectory tracking index significantly. The maximum drift threshold the controller could handle was found to be about 0.41 deg/sec² for the oval trajectory. The deterioration of the trajectory tracking index for a drift value from 0.15 to 0.5 deg/sec² is shown in Figure 7.57. It can be seen that the index does not change by much as the drift value increases from 0.15 to 0.41 deg/sec², but at 0.42 deg/sec², a significant decline occurs and this decline continues up till the 0.5 deg/sec² drift value.

![Figure 7.57: Trajectory Tracking Index for Increasing Drift Value (Oval Trajectory)](image)

Longer trajectory flight times was also found to have a negative impact on the trajectory tracking index. The square trajectory had the longest flight time of about 130 seconds, followed by the climbing S trajectory, which flew for about 97 seconds, then the figure 8 lasted about 95 seconds and the oval had the shortest flight time of about 85 seconds. From the trajectory tracking indices for the pitch rate gyro sensor failure for all four trajectories it can be seen that the index for the square trajectory was the least and that of the oval trajectory was the highest.

The time delay and noise ACs were also analyzed for their effects on the generated control surface commands and it as found that the time delay caused an erratic behavior on the generated commands for all control surfaces, while the noise affected the surface corresponding to the affected sensor, so noise affecting the roll rate sensor gyro measurement would affect the generated aileron command of the controller. The control surface deflections for a high severity time delay and noise AC affecting the roll rate for the figure 8 trajectory are shown in Figure 7.58 and Figure 7.59.
Figure 7.58: Control Surface Deflections for High Severity Time Delay AC affecting Roll Rate Sensor (Figure 8 Trajectory)

Figure 7.59: Control Surface Deflections for High Severity Noise AC affecting Roll Rate Sensor (Figure 8 Trajectory)
7.2.3 GPS Failure

The trajectory tracking, control activity and throttle saturation are the PIs used to evaluate the GPS failure. The trajectory tracking PI is used to evaluate the ability of the controller to return the aircraft to the commanded trajectory - the primary objective of the task. The control activity is also analyzed as too much activity and saturation on the control surfaces is not desired. The throttle saturation index is relevant because it exhibited a decreasing trend as the dropout duration increased.

For the figure 8 trajectory, the trajectory tracking index reduced to about 0.5 for a position dropout threshold lasting 5 seconds. This index reduced more for the 10 second threshold and was very similar to that of the 13.8 second threshold, the maximum recoverable threshold. For the dropout lasting 13.9 seconds, the aircraft was unrecoverable and had a trajectory tracking PI of 0. The trajectory tracking PI is shown in Figure 7.60. The control activity index did not decline much, having a value of about 0.8 for the non-recoverable threshold. This shows the control surfaces were not over saturated and their overall activity was not excessive, although, that does not make up for the aircraft not being able to return to the trajectory. The control activity PI for the position dropout is shown in Figure 7.61. After inspecting the control surface PI, a trend was seen with the throttle saturation PI that this index decreased significantly as the threshold of the dropout increased. This can be seen in Figure 7.62.

![Figure 7.60: Trajectory Tracking PI (Figure 8 Position Dropout)](image_url)
For the case of the velocity dropout, the trajectory tracking PI decreased by about 75% but stayed within this range for all dropout durations, even for the unrecoverable threshold case, as shown in Figure 7.63. This can be said to be due to the aircraft position being known by the controller. The control activity and throttle saturation PIs followed the same trend as the case of the position dropout case, but the throttle was slightly less saturated for the unrecoverable case (45.5 seconds). The control activity and throttle saturation PIs can be seen in Figure 7.64 and Figure 7.65.
Figure 7.63: Trajectory Tracking PI (Figure 8 Velocity Dropout)

Figure 7.64: Control Activity PI (Figure 8 Velocity Dropout)
The trajectory tracking PI was the worst when the position and velocity signals were dropped out for the figure 8 trajectory with the unrecoverable case having an index of 0. The control activity and throttle saturation PIs also followed the similar trend mentioned for the position dropout case. Although, the trajectory tracking and control activity PIs were slightly higher for the 33.2 second dropout duration (maximum recoverable threshold) when compared to the 10 second dropout duration, the throttle saturation exhibited a consistent decreasing trend. The trajectory tracking, control activity and throttle saturation PIs can be seen in Figure 7.66-Figure 7.68.
For the dropout case using the oval trajectory, a trend similar to that of the figure 8 trajectory was found. For the 5 second position dropout duration case, the trajectory tracking index dropped by almost 50%, whereas the control activity index dropped by less than 5%. The trajectory tracking index was fairly constant for the oval trajectory even at
the unrecoverable threshold (13.9 seconds) compared to that of the figure 8 trajectory which had an index of 0 for the same dropout duration. The throttle saturation index trend was consistent with that of the figure 8 dropout cases, with it reducing exponentially. The trajectory tracking, control activity and throttle saturation indices for the position dropout are presented in Figure 7.69 through Figure 7.71.

![Figure 7.69: Trajectory Tracking PI (Oval Position Dropout)](image)

![Figure 7.70: Control Activity PI (Oval Position Dropout)](image)
For the case where the velocity was dropped, the trajectory tracking index as shown in Figure 7.72 was lowest for the 5 second dropout duration for the recoverable cases and was 0 for the unrecoverable case (28.4 second threshold). The control activity and throttle saturation indices (Figure 7.73 and Figure 7.74) followed the same trend as earlier mentioned. It is important to note that the trajectory tracking index shown in Figure 7.72 for the velocity dropout shows a similar trend as Figure 7.63, the corresponding dropout case for the figure 8 trajectory, highlighting the importance of the position signal for trajectory tracking purposes.
Finally, when the position and velocity signals were dropped out for the oval trajectory, the values for the trajectory tracking index were very close to 0 (Figure 7.75), comparable to the trend seen in the case of the figure 8 trajectory (Figure 7.66). The control activity index (Figure 7.76) decreased as the dropout duration increased from 5 to 10 seconds but did not decrease significantly onwards although the throttle saturation (Figure 7.77) index followed its similar trend.
Figure 7.75: Trajectory Tracking PI (Oval Position and Velocity Dropout)

Figure 7.76: Control Activity PI (Oval Position and Velocity Dropout)
7.3 Sensor Failure Analysis

This section presents a sensitivity analysis on for the yaw sensor failure simulations. Due to the controller not being significantly affected by abnormal conditions on this sensor, the analysis is performed in order to understand how altering the several FC parameters would affect the trajectory tracking and control activity indices.

For the dead band FC, increasing the dead band region did not have a significant effect on the yaw rate sensors even for a huge dead band region of 10 deg/s. The time constant case was also found to not have a significant effect on the yaw rate sensor with extreme values not having any significant effect on the controllers PI. The time delay case had a significant effect on the yaw rate sensor with a pure delay of 0.22 seconds affecting the controllers’ indices. The control activity index reduced due to the delay causing too much activity on the rudder surface. The trajectory tracking index also decreased for the yaw rate sensor failure case. For the bias FC, a continuous increase of the bias value on the yaw rate sensor gyro did not have any significant effect on the controller’s performance. The saturation FC, did not have any effect on the PIs even for an extreme case of a sensor saturated a 0, which is comparable to a dropout. Similarly, the dropout (signal drops to 0), was found to not have any significant effect on the PIs. Even when a dropout occurs at a non-zero value, no effect was found on the PI for the dropout affecting the yaw rate sensor. Finally, for the noise FC, a huge standard deviation noise, has to be achieved for the yaw rate sensor gyro AC to reduce the controllers control activity index by a lot, but leaves the trajectory tracking index indifferent.

Table 7.1 shows the parameters that were tested to achieve a significant drop in the PIs of the controller for failures affecting the yaw rate sensor. Note that the analysis is
performed using the oval trajectory and some of the values found may be out of failure operational modes, but provide a better understanding of the sensitivity of the controller to the yaw rate sensor failures. N/A is used in cases where no significant effect was found.

Table 7.1: Sensor Failure Sensitivity Analysis

<table>
<thead>
<tr>
<th>Functional Category</th>
<th>Yaw Rate Sensor Failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dead Band</td>
<td>N/A</td>
</tr>
<tr>
<td>Time Constant</td>
<td>N/A</td>
</tr>
<tr>
<td>Time Delay</td>
<td>$T = 0.22$</td>
</tr>
<tr>
<td>Scaling</td>
<td>$\sigma = 20$</td>
</tr>
<tr>
<td>Bias</td>
<td>N/A</td>
</tr>
<tr>
<td>Drift</td>
<td>N/A</td>
</tr>
<tr>
<td>Saturation</td>
<td>N/A</td>
</tr>
<tr>
<td>Dropout</td>
<td>N/A</td>
</tr>
<tr>
<td>Noise</td>
<td>$\sigma_D = 40$</td>
</tr>
</tbody>
</table>
CHAPTER 8: CONCLUSIONS

The main objective of this research effort was to add more capabilities within the WVU UAS simulation environment to facilitate the improvement of autonomous flight handling normal and abnormal conditions on the aircraft subsystems. This was achieved through the development and implementation of a comprehensive sensor failure model for 9 FCs which can be implemented on different sensor subsystems.

A new aircraft model, Phastball was also added to the WVU UAS simulation environment which was used for the completion of thesis effort. The aircraft modeling was made possible with the use of Tornado [16] VLM to get the stability and control derivatives corresponding to the aircraft geometry. The major drawback found with using Tornado was it having a low angle of attack limitation; therefore, the lift coefficient at higher angles of attack had to be solved for, which was done using aerodynamic equations. The derivatives obtained were used to model the aircraft in the WVU UAS simulation environment and available flight data was used to verify the model comparing its response with that of the simulation environment. One recommendation to improve the aircraft modeling would be to tune the model using flight data acquired from specifically designed tests for validation purposes with known atmospheric conditions and tightly executed commands.

For the gyro sensor abnormal conditions, simulations were performed successfully using the position PID controller for four trajectories with the failures implemented at low and high severities affecting the aircraft angular rates individually. The simulated gyro sensor abnormal conditions were found to have some effect on the roll and pitch rate sensors more than the yaw rate sensor for all trajectories with some FCs having more of a noticeable effect than others. A sensitivity analysis was performed to study the effects of the failure parameters for failures affecting the yaw rate sensors using one of the trajectories which showed how excessive some parameters will need to be to see the effect of abnormal conditions due to failures affecting the yaw sensor gyro. The high severity time delay and drift ACs were found to have the worst effect on the trajectory tracking performance of the controller for failures affecting the roll rate sensor gyro. The delay and the noise ACs showed the most effects on the control activity index of the controller. The trajectory tracking PI dropped a lot for the high severity drift AC when the pitch rate sensor gyro was failed. This was found to be due to a significant change in altitude due to the failure. The control activity index for the pitch rate sensor gyro failure followed the same trend as the roll rate gyro failure with the time delay and noise ACs having the most effect on the index. The climbing S trajectory was the least affected of all the trajectories with only its control activity index being affected slightly by some of the ACs.
The developed comprehensive sensor model proved to be a valuable tool for investigating the combined interaction between sensors, control laws and aircraft dynamics. It may also be used to reveal the important and sometime unexpected effects of sensor abnormal operation on the performance of autonomous flight control laws.

From the simulations, it was observed that the high severity drift and time delay ACs were the most detrimental to the trajectory tracking performance of the controller and the time delay and noise ACs affected the control activity performance the most. One interesting finding was that the dropout FC did not have the worst effect on the controller performance, with the dropout FC having the same PIs as the nominal case for all trajectories. Upon further inspection, the controller was able to account for the dropout by commanding a bank angle which accommodated the abnormal condition. When a dropout was tested for the YF22, another aircraft in the WVU UAS simulation environment, using the same controller class, the aircraft was unstable and had a lot of control activity on the affected surfaces.

An effective study was performed accurately on the effects of a GPS signal being dropped out, where the last known GPS information is held constant over the duration of the dropout. The study was performed using two trajectories for GPS position, velocity and position and velocity signals being dropped individually which aided the analysis of the individual effects of the dropout on the signals. The maximum recoverable time was also found for the several simulation scenarios. For the GPS dropout simulations, the worst trajectory tracking performance was found when the position and velocity signals were dropped. The control activity performance was exhibiting a decreasing trend as the duration of the dropout increased. The throttle saturation index was also analyzed for the dropout and was found to show an exponential decrease for increasing dropout durations. With respect to the maximum recoverable time for the dropout, the position dropout case had the worst recoverable time with a maximum 13.8 second threshold for both figure 8 and oval trajectories. For the velocity-only and position and velocity dropout cases, the aircraft was able to recover after a much longer dropout duration than in the position-only dropout case, but did not show a noticeable trend regarding the ability of the aircraft to return to the commanded trajectory.

The limited analysis performed in this research effort leads to the conclusion that the effects of sensor abnormal operations on the autonomous flight control laws may vary significantly depending on the sensor FC, the structure of the control laws, the dynamic characteristics of the aircraft and the characteristics of the commanded trajectory.

Some recommendations for future work include, but are not limited to:

- Investigation of accommodation schemes for the ACs
- Consideration of other aircraft models for the sensor failure simulation scenarios
• Consideration of other trajectories that are heavily dependent on elevator and rudder commands to investigate the effects on the controller’s performance
• Consideration of other classes of controllers including other fixed and adaptive controllers
• Consideration of simultaneous ACs such as adverse atmospheric conditions and other subsystem failures.
REFERENCES


Appendix A - Performance Indices

This appendix contains the performance indices for all trajectories for the yaw rate sensor failure for all simulated cases.

Figure A-1: Trajectory Tracking Index for Square Trajectory (Yaw Rate Sensor Failure)

Figure A-2: Control Activity Index for Square Trajectory (Yaw Rate Sensor Failure)
Figure A-3: Trajectory Tracking Index for Oval Trajectory (Yaw Rate Sensor Failure)

Figure A-4: Control Activity Index for Oval Trajectory (Yaw Rate Sensor Failure)
Figure A-5: Trajectory Tracking Index for Figure 8 Trajectory (Yaw Rate Sensor Failure)

Figure A-6: Control Activity Index for Figure 8 Trajectory (Yaw Rate Sensor Failure)
Figure A-7: Trajectory Tracking Index for Climbing S Trajectory (Yaw Rate Sensor Failure)

Figure A-8: Control Activity Index for Climbing S Trajectory (Yaw Rate Sensor Failure)