Modeling, Simulation and Control of Very Flexible Unmanned Aerial Vehicle

by

Zi Yang Pang

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Aerospace Engineering)
in the University of Michigan
2018

Doctoral Committee:

Professor Carlos E. S. Cesnik, Chair
Professor Ella M. Atkins
Professor Bogdan I. Epureanu
Professor Ilya V. Kolmanovsky

Zi Yang Pang

pziyang@umich.edu

ORCID iD: 0000-0003-4614-5857

This dissertation is dedicated to my family, my friends and colleagues.

ACKNOWLEDGEMENTS

First, I would like to thank the committee for reviewing and providing valuable feedback on this dissertation.

To my advisor, Prof. Cesnik, for his guidance and support over the years. I might not have been his best student, but he is certainty my best teacher.

To Prof. Atkins, who guided me from day one, when I know next to nothing about hardware and software. I would not have achieve large part of the dissertation without her.

To Prof. Kolmanovsky, for his valuable guidance on control aspects, and for the interesting discussions outside of work.

To past and present members of A^2SRL research lab, for the companionship, and for all the help rendered over the years (especially for the flight tests).

To everybody who contributed to X-HALE project: special thanks to Keith Shaw for being our magnificent pilot; shout out to my partner in crime, Dr. Jessica Jones: We did it!; to Dylan Davis, and Ying Ying for the help with the stereo vision system.

To my parents and sister, who always supported what I did, I love you all very much. To my dearest wife Cheerin, who brought love and joy into my life, and for dragging me to travel around the world.

Finally, I would like to thank DSO National Laboratories for the Post-Graduate Scholarship Award which allowed me to purse this PhD.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDICES

LIST OF ABBREVIATIONS

ATV	Aeroelastic Testbed Vehicle
BEC	Battery Elimination Circuit
CAD	Computer Aided Design
CCD	Charged Coupled Devices
CFD	Computation Fluid Dynamics
CSD	Computational Structure Dynamics
DAQ	Data Acquisition Unit
DATCOM	United States Air Force Data Compendium
DOF	Degree of Freedom
ESC	Electronic Speed Controller
FBG	Fiber Braggs Grating
FEM	Finite Element Method
FIFO	First In, First Out
FG	Fast Gradient
HALE	High Altitude Long Endurance
HDF5	Hierarchal File Format 5
INS	Inertial Navigation System
IMU	Inertial Measurement Unit
KF	Kalman Filter
LED	Light Emitting Diode
LQG	Linear Quadratic Gaussian

LQR	Linear Quadratic Regulator
NLS	Nonlinear Least Square
MPC	Model Predictive Control
OLTF	Open Loop Transfer Function
PCB	Printed Circuit Board
PI	Proportional Integral
PWM	Pulse Width Modulation
RRV	Risk Reduction Vehicle
RTOS	Real-Time Operating System
SBC	Single Board Computer
SNR	Signal-to-Noise Ratio
SISO	Single Input Single Output
SSC	Servo Switch Controller
SVD	Singular Value Decomposition
TIC	Theil Inequality Coefficient
UAS	Unmanned Aerial System
UM/NAST	Nonlinear Aeroelastic Simulation Toolbox
VFA	Very Flexible Aircraft
XML	eXtensible Markup Language
RSSI	Received Signal Strength Indication

LIST OF SYMBOLS

Control Design

A_0	Chirp amplitude
$C_{d\Delta}$	Select rows of output matrix C_d corresponding to measurements
e	Prediction error
e_F	Filtered prediction error
F	Nonlinear least squares function
$F(q)$	Prefilter written using delay operator q
f_0	Chirp minimum frequency
f_1	Chirp maximum frequency
f_k	Frequency point
G	Kalman filter output noise covariance
$G_{uu}(s)$	Auto/cross-spectral of input
$G_{yu}(s)$	Cross-spectra of input and output
$G_{yy}(s)$	Cross-spectra of output
$H(q,\theta)$	Equivalent state-space for control input written using delay operator q
$H(s)$	Transfer function, matrix form
$H_0(q,\theta)$	Equivalent state-space for noise written using delay operator q
H_{k+1}	Kalman filter output matrix
J	Cost functional
K_{k+1}	Kalman filter gain
$L(.)$	Scalar-valued cost function

n	State dimension
N_p	MPC prediction horizon
N_u	MPC control horizon
P	Terminal cost
p	Control dimension
Q	State cost
q	Delay operator
Q_{cov}	Process noise covariance matrix
R	Control cost
$R_{cov,\Delta}$	Sub-matrix of noise covariance matrix R corresponding to measurements
R_{cov}	Measurement noise covariance matrix
s	Laplace operator
S_u	Affine control constraints
S_x	Affine state constraints
t_1	Chirp maximum duration
t_k	k_{th} time instant
U	Predicted control trajectory
$U(s)$	Fourier representation of input data, matrix form
U^*	Optimal control trajectory
u^*	Optimal control
u_k	Control states at k^{th} time step
u_{max}	Control maximum allowable magnitude
u_{min}	Control minimum allowable magnitude
v_k	Measurement noise
w_k	Process noise
X	Predicted state trajectory

x_k	System states at k^{th} time step
x_{max}	State maximum allowable magnitude
x_{min}	State minimum allowable magnitude
y	Measurements/output
$Y(s)$	Fourier representation of output/measurement data, matrix form
y_{meas}	Measured output at the sensor
y_{pred}	Predicted output given a fit model
Z^N	Set of input and output over N time points
$\delta(t)$	Chirp signal
γ	Coherence function
$\mathcal{M}$	Model structure
θ	Model parameters
θ^*	Optimal model parameters
$\mathbb{U}$	Set of admissible control inputs
$\mathbb{X}$	Set of admissible states

UM/NAST

A	Continuous time state-space plant matrix
A_d	Discrete time state-space plant matrix
a_i	i^{th} column of A
a_{sensor}	Sensor acceleration
B	Body frame
B	Continuous time state-space control matrix
b	Integral of body velocity and angular rate states β
b_c	Half chord
B_d	Discrete time state-space control matrix
b_i	i^{th} column of B
b_n	Inflow state coefficients

B_w	Continuous time state-space disturbance matrix
B_{wd}	Discrete time state-space disturbance matrix
b_{wi}	i^{th} column of B_w
C^{Bw}	Rotation matrix from beam frame w to body frame B
C^{SG}	Rotation matrix from sensor frame S to inertial frame G
c_d	Sectional drag coefficient
c_l	Sectional lift coefficient
c_m	Sectional moment coefficient
C_{BB}	Damping matrix due to rigid body states
C_{BF}	Damping matrix due to coupling between rigid body and flexible states
C_{FB}	Damping matrix due to coupling between flexible and rigid body states
C_{FF}	Damping matrix due to flexible element
c_{l_α}	Lift curve slope
c_{l_δ}	Lift derivative due to control surface
d	Distance between aerodynamic center and beam reference axis
F_1	Inflow matrix due to acceleration
F_2	Inflow matrix due to velocity
F_3	Inflow matrix due to inflow states
G	Inertial frame
h	12×1 vector of position and directrix vector including rigid body translation
h_w	12×1 vector of relative position and directrix vector
$h_{BC,w}$	h_w vector of the boundary node
$I_{3\times3}$	Identity matrix
I_{xx}	Moment of inertia
I_{xy}	Product moment of inertia
$J_{\theta b}$	Structural jacobian relating rotation to rigid body motion

$J_{\theta\varepsilon}$	Structural jacobian relating rotation to strain
$J_{h\varepsilon}$	Structural jacobian relating h to strain
J_{hb}	Structural jacobian relating h to rigid body motion
K	12×12 stiffness matrix of each strain element
K_{FF}	Stiffness matrix
M	Mach number
M_{BB}	Mass matrix due to rigid body states
M_{BF}	Mass matrix due to coupling between rigid body and flexible states
M_{FB}	Mass matrix due to coupling between flexible and rigid body states
M_{FF}	Mass matrix due to flexible states
nq	Dimensions of first order states
P_B	Inertial position of body origin (expressed in inertial frame)
p_B	Position body origin (expressed in body frame)
p_c	Position vector of point c from inertial origin
p_w	Position vector of point w on beam reference line from body origin
$p_{c,r}$	Relative position vector of point c from body origin
q	Generalized state vector
R_B	Generalized load for rigid body equations
R_F	Generalized load for structural equations
Re	Reynolds number
S	Sensor frame
s	Distance along beam reference line
u	Control vector
v_B	Linear velocity of the body frame
v_g	Gust velocity
w_x	Directrix vector of local beam x direction
w_y	Directrix vector of local beam y direction

w_z	Directrix vector of local beam z direction
x	UM/NAST frame ordinates, direction positive towards beam right
y	UM/NAST frame ordinates, direction positive towards beam front
y_S	Offset from beam reference line in the local beam frame y direction
z	UM/NAST frame ordinates, positive in right hand triad with x and y
z_S	Offset from beam reference line in the local beam frame z direction
α_{eff}	Effective angle of attack
$\dot{\alpha}$	Time derivative of angle of attack
β	6×1 linear and angular velocity vector
Δs	Distance from the previous boundary node to position of sensor projected onto the beam reference line
ε	Strain state vector
ε_x	Extensional strain
κ_x	Curvature in x direction (twist)
κ_y	Curvature in y direction (out-of-plane bending)
κ_z	Curvature in z direction (in-plane-bending)
λ	Inflow state vector
λ_0	Downwash due to inflow
ω_B	Angular velocity of the body frame
ω_c	Angular velocity of sensor frame to inertia frame (expressed in body frame)
ω_{sensor}	Angular velocity of sensor frame to inertia frame (expressed in sensor frame)
ϕ	Euler angle about UM/NAST x direction
ψ	Euler angle about UM/NAST z direction
ρ	Density
θ	Euler angle about UM/NAST y direction
ζ	Quaternion vector

Vision Processing

C	Circularity
c_x	Offset from optical origin
c_y	Offset from optical origin
dx	Displacement in x direction
dy	Displacement in y direction
dz	Displacement in z direction
e_{cb}	Re-projection error of chessboard
e_{sep}	Re-projection error in separation
f	Focal length
f_x	Focal Length in x direction
f_y	Focal Length in y direction
k_1	Radial distortion parameter 1
k_2	Radial distortion parameter 2
k_3	Radial distortion parameter 3
M	Intrinsic parameter matrix
p_1	Tangential distortion parameter 1
p_2	Tangential distortion parameter 2
R	Rotational matrix
t_v	Translational vector
$\hat{X}$	Object space coordinates
X	Physical space coordinates
x	Image space coordinates
$\hat{Y}$	Object space coordinates
Y	Physical space coordinates
y	Image space coordinates
$\hat{Z}$	Object space coordinates
Z	Physical space coordinates

ABSTRACT

This dissertation presents research on modeling, simulation and control of very flexible aircraft. This work includes theoretical and numerical developments, as well as experimental validations. On the theoretical front, new kinematic equations for modeling sensors are derived. This formulation uses geometrically nonlinear strain-based finite elements developed as part of University of Michigan Nonlinear Aeroelastic Simulation Toolbox (UM/NAST). Numerical linearizations of both the flexible vehicle and the sensor measurements are developed, allowing a linear time invariant model to be extracted for control analysis and design. Two different algorithms to perform sensor fusion from different sensor sources to extract elastic deformation are investigated. Nonlinear Least Square (NLS) method uses geometry and nonlinear beam strain-displacement kinematics to reconstruct the wing shape. Detailed information such as material properties or loading conditions are not required. The second method is the Kalman Filter (KF), implemented in a multi-rate form. This method requires a dynamical system representation to be available. However, it is more robust to noise corruption in sensor measurements.

In order to control maneuver loads, Model Predictive Control (MPC) is applied to maneuver load alleviation of a representative very flexible aircraft (X-HALE). Numerical studies are performed in UM/NAST for pitch up and roll maneuvers. Both control and state constraints are successfully enforced, while reference commands are still being tracked. MPC execution is also timed and current implementation is capable of almost real-time operation.

On the experimental front, two aeroelastic testbed vehicles (ATV-6B and RRV-

6B) are instrumented with sensors. On ATV-6B, an extensive set of sensors measuring structural, flight dynamic, and aerodynamic information are integrated on-board. A novel stereo-vision measurement system mounted on the body center looking towards the wing tip measures wing deformation. High brightness LEDs are used as target markers for easy detection and to allow each view to be captured with fast camera shutter speed. Experimental benchmarks are conducted to verify the accuracy of this methodology.

RRV-6B flight test results are presented. System identification is applied to the experimental data to generate a SISO description of the flexible aircraft. System identification results indicate that the UM/NAST X-HALE model requires some tuning to match observed dynamics. However, the general trends predicted by the numerical model are in agreement with flight test results.

Finally, using this identified plant, a stability augmentation autopilot is designed and flight tested. This augmentation autopilot utilizes a cascaded two-loop proportional integral control design, with the inner loop regulating angular rates and outer loop regulating attitude. Each of the three axes is assumed to be decoupled and designed using SISO methodology. This stabilization system demonstrates significant improvements in the RRV-6B handling qualities.

This dissertation ends with a summary of the results and conclusions, and its main contributions to the field. Suggestions for future work are also presented.

CHAPTER 1

Introduction

1.1 Motivation

Due to cost and operational considerations, there is a relentless drive for more efficient aircraft. In civilian aerospace applications, a more efficient aircraft reduces direct operating cost (e.g., due to lower fuel burn). Increasing useful payload weight fraction by optimizing aircraft structure also increases operating profits. In military applications, a more efficient aircraft allows longer mission endurance, particularly important for intelligence, surveillance and reconnaissance (ISR) applications. Such aircraft designs often employ high-aspect-ratio wings for aerodynamic efficiency and light weight structures for weight efficiency. As a result, these designs tend to exhibit large wing deformations within their normal operating envelope. In addition, structural modes interact with classical flight dynamic modes, making the design, analysis, and control radically different from conventional rigid aircraft. Again, it should be emphasized that this is not the design *goal* but the design *consequence*. Although there is no formal definition found in literature, aircraft which exhibit above characteristics to a significant degree are termed VFA.

There have been numerous incidents involving VFA, highlighting the lack of understanding of flight physics and safe operations of a VFA. In 2003, NASA Helios, an experimental flying wing VFA with wingspan of 75 m encountered turbulence during

flight. It morphed into an unexpected, persistent, high dihedral configuration, leading to a divergent pitch mode. High dynamic pressure caused structural failure of the wing, resulting in a crash and total loss of the prototype [3]. More recently in 2015, Titan Aerospace (acquired by Google) was flight testing a solar powered VFA named Solara 50 when it crashed shortly after takeoff. NTSB Report DCA15CA117 [4] concluded that a significant thermal air activity caused Solara 50 to exceed its design speed, inducing significant wing deformation. The consequence is an uncontrollable rapid bank and descent, culminating in structural failure and eventual crash. Finally, in 2016, Facebook Aquila, also a solar powered VFA meant to operate as an atmospheric satellite for telecommunication relay, experienced an in-flight structural failure during landing approach. NTSB Report DCA16CA197 [5] concluded that during approach, a wind gust caused the autopilot to command a nose down maneuver, leading to an increased airspeed. Upon re-establishing the glide path, the autopilot commanded the elevon upwards. This combination of overspeed and control maneuver caused structural failure of the right wing, resulting in a loss of the vehicle.

It is apparent that coupled aeroelastic-flight dynamic simulation tools are required to analyze and better predict the VFA characteristics. In order to establish confidence in these numerical tools, they need to be validated against experimental flight results. Novel control strategies for maneuver load or gust load alleviation specifically for VFA should also be explored. Finally, sensors capable of monitoring structural information have to be integrated with flight control system. However, it remains an open question on the type of sensor information and the sensor fusion algorithm needed for feedback control.

1.2 Literature Review

This section presents a literature survey of selected topics pertinent to the modeling, simulation, and control of VFA. Section 1.2.1 reviews existing works and key observations on aeroelasticity coupled with flight dynamics. Section 1.2.2 reviews flight test design and system identification as applied to Unmanned Aerial System (UAS) and VFA. Section 1.2.3 reviews measurement techniques for structural deformation as well as reconstruction techniques to obtain overall shape from discrete measurements. Section 1.2.4 reviews the control design considerations and techniques for control of VFA. Finally, Section 1.2.5 reviews application of MPC to control VFA and summarizes the numerical techniques for solving MPC problems.

1.2.1 Coupled Aeroelasticity and Flight Dynamics

Pioneering work by van Schoor and von Flotow [6] presented aeroelastic analysis of the MIT's human powered airplane named Michelob Light Eagle (MTE). Similarities between MTE and VFA include high-aspect-ratio, flexible wings with low structural frequencies, as well as high apparent mass effects. Therefore, conclusions from their work can be applied to modeling and simulation of VFA. The authors showed that inclusion of unsteady aerodynamic effects via Theordorsen's function and elastic structural response are mandatory for accurate modeling. More recently, Cesnik et al. [7] re-examined aircraft structural design procedure as applied to VFA. They concluded that structural rigid body interactions are important due to lack of frequency separation between structural modes and rigid body modes. In addition, geometrically non-linear effects about the steady state condition significantly affects the accuracy of the model. This is apparent in long term numerical simulations where significant variations in the final position and attitude can develop without proper consideration of structural nonlinearities. Lastly, linearized models may not yield a

conservative estimate of internal loads in the presence of gust.

Many researchers have proposed various analytical methods to model this strong coupling between aerodynamics, structural dynamics and rigid body dynamics. In this section, the review is focused on formulations without resorting to high fidelity Computational Structure Dynamics (CSD)/Computation Fluid Dynamics (CFD) coupling with millions of degrees of freedom, requiring hours of computation per load case. Fast computational speed requires simplifications in structural dynamics and/or aerodynamics, while retaining full nonlinear 6-degree-of-freedom flight dynamics. Patil and co-workers [8] uses a mix variational beam formulation coupled with induced flow theory to model a complete VFA. They compared the theory using experimental flutter results from a Goland Wing. Drela [9] created ASWING, aimed at performing fast aeroelastic analysis as well as rapid design iteration for preliminary aircraft design. ASWING employs nonlinear beams, with compressible vortex and source-lattice aerodynamic model.

Cesnik and Brown [10, 11] started development of 1-D nonlinear strain-based beams for modeling piezoelectric embedded active structure and control design for very flexible wings. The underlying theory eventually became UM/NAST. Different solver options for different aerodynamics (e.g., strip theory for quasi-steady with finite inflow for unsteady aerodynamics [12, 13], unsteady vortex lattice methods, and convolution integrals), and nonlinear 6-Degree of Freedom (DOF) rigid body equation of motion [14,15] were incorporated. Shearer and Cesnik [16] subsequently derived an implicit modified Newmark method for time integration. UM/NAST is well validated with numerical results from other solvers. Su and Cesnik [14] compared UM/NAST with MSC NASTRAN for simple, joint and split beam test cases, and showed negligible error. Hallissy and Cesnik [17] showed good time domain correlation with high fidelity CFD/CSD coupled solution employing large mesh deformation for a hypothetical High Altitude Long Endurance (HALE) wing. Ritter et al. [18] compared

UM/NAST with proposed enhanced modal methods coupled with Unsteady Vortex Lattice Method (UVLM) and both simulations show good agreement.

Meirovitch and Tuzcu [19] introduced a structural formulation using generalized shape functions for spatial discretization of beam torsion and bending. They successfully applied that to time domain dynamic simulation of a representative HALE aircraft.

Time domain aeroelastic analysis tools are critical for analyzing aeroelastic phenomenon. However, most numerical tools have been validated only for interactions between aerodynamics and structural dynamics focusing on flutter or limit cycle oscillations phenomenon (e.g., Refs. [20, 21]). There is a scarcity of experimental data for validation of coupled aeroelastic interactions with flight dynamics. Therefore, experimental flight data collected with a specially instrumented aeroelastic UAS testbed will be valuable for the field.

1.2.2 Experimental Flight Test and System Identification

Full scale flight testing is used for evaluating actual aircraft performance and control characteristics. For manned aircraft, this is mandatory for flight-worthiness certification (FAR Part 21.35). Although advances in computing power allow higher fidelity models and simulations, experimental results are still required to corroborate analytical or numerical predictions. Ground vibration and wind tunnel tests can provide good insights into the aerodynamics and structural response at relatively low cost. However, limitations of scaling effects (e.g., Mach number, Reynolds Number, and aeroelastic matching) as well as inability to fully replicate free flight boundary conditions mean that experimental flight testing can never be fully eliminated. [22, 23] A natural extension is then to use the flight test data to build or improve a mathematical model of the physical aircraft.

System identification is the science of determining a suitable system model given

observations of input and output measurements. Hamel and Jategaonkar [24] provided a historical view of system identification applied to flight vehicles. Morelli and Klein [25] at NASA Langely, and Wang and Iliff [26] at NASA Dryden detailed the methodology developed at their respective research centers. They also provided case studies where system identification were successfully applied during the aircraft development process. The best practices and mathematical background of system identification are covered extensively by Ljung [27] and Tischler [28]. In grey-box identification, dynamical equations can be written but the magnitude of the parameters are unknown and to be estimated (e.g., I_{xx}, C_{l_α}, and C_{l_δ}). In black-box identification, no *a priori* assumptions are made about system dynamics. In general, each state in a n-order black-box model does not have physical interpretation. System identification can also be classified into time domain and frequency domain methods. Time domain methods use time history of input and output measurements directly, typically minimizing error between actual measured and predicted outputs [27]. There are also different model structures, and locations where measurement and process noise are introduced. Hamel and Jategaonkar [24] argued that time domain identification techniques excel in their simplicity, and most are extensible to non-linear models. Frequency domain methods, on the other hand, use Fourier transform representation of input and output measurements. To their advantage lies numerical stability even for identifying unstable systems, as well as their ability to emphasize frequency bands of accurate data through use of coherence functions. Frequency domain analysis and control theories (e.g., Bode, and Nyquist plots) are also well established. However, the disadvantage is that frequency methods are restricted to linear analysis. Pintelon and Schoukens [29] provided suggestions on testing linearity of the system under question, as well as a best linear estimate in presence of nonlinear dynamics.

Theodore et al. [30] postulated that system identification for UAS is especially attractive for the following reasons: First, it is often difficult to develop and validate

analytical flight dynamics models based on first principles modeling due to short UAS design cycle. Next, most semi-empirical estimation methods like United States Air Force Data Compendium (DATCOM) [31] and Roskam [32] are developed for full scale aircraft, and cannot be scaled easily for UAS. Inviscid assumptions to simplify aerodynamic computations are not applicable in the low Reynolds number regime UAS operate in. System identification allows researchers to circumvent these issues. Hoffer [33] summarized a large body of literature demonstrating successful application of system identification to UAS. Dorobantu et al. [34] performed frequency domain system identification on a small fixed wing UAS, as well as detailing the flight test design considerations using insights from analytical models. In that work, they identified a 4^{th}-order longitudinal and 5^{th}-order lateral model in CIFER [35], and their predicted model correlated very well with validation flight data. Scheper et al. [36] used time domain Output Error method to estimate a 10-state linear plant for a fixed wing UAS. At the same time, they cautioned that special considerations must be made when identifying small and agile UAS as assumptions in identification of large aircraft may not hold.

There are also recent works on system identification of flexible structures and UAS. Schulze et al. [37] demonstrated a real time identification of flexible structural modes in an UAS. In that work, they implemented a curve-fitting frequency domain decomposition method which utilizes accelerometer data on the wings to obtain mode shapes and damping ratios. Silva and Mönnich [38] identified an instrumented flexible sailplane using five structural modes (two anti-symmetric and three symmetric) coupled with rigid body flight dynamics. The flexibility of aircraft extremities (tail and wingtips) clearly contributed to the aircraft dynamics. Inclusion of structurally flexible modes improved the predictive capability of their model, decreasing Theil Inequality Coefficient (TIC) across all measured parameters (e.g., angular rates, and linear accelerations). Garrec and Kubica [39] used Eigensystem Realization Algo-

rithm to identify the structural modes of an Airbus A340-300 for control law design refinements to improve ride comfort. Kukreja and Brenner [40] presented a non-linear NARMAX identification of aeroelastic pitch-plunge model. They suggested NARMAX as a useful tool in identification of severely nonlinear system.

In order to collect high-fidelity aeroelastic data from flight tests, accurate measurements of structural, aerodynamic, and flight dynamics are required. In particular, the UAS needs to be specially instrumented to measure structural deformations.

1.2.3 Measurements of Structural Deformation

Measuring deformation in wing structures, especially for VFA, is important for safety, performance, and controllability. Potential applications include structural health monitoring, wing shape control and active aeroelastic tailoring, among others. From published literature, there are three classes of sensors used for this purpose, namely: vision-based, foil strain-gauges and Fiber Braggs Grating (FBG) systems. However, in-situ flight vision measurement systems are still rare. There exist significant engineering challenges in sensor placement and accommodating supporting electronics on a UAS due to weight, volume, and power constraints.

Vision based systems generally involve taking a series of photographic images and post-processing them to recover time history of a tracked object in physical space. Being a non-intrusive measurement method, they can be used in an environment where interference to the flow is not permitted. Hence, it is not surprising to find their deployment in flow field measurements, shape recovery or model attitude measurements for wind tunnel tests [41–44]. The earliest in-flight measurement system is probably the HiMAT Aeroelastic Tailored Wing study commissioned by NASA in 1980s. In that study [45], infra-red LEDs were mounted in aerodynamically shaped fixtures on the wing. They were focused and captured using a light sensitive diode array mounted on the fuselage. The wing deformation was recovered by comparing

the image with a calibration of known displacement. Allen et al. [46] employed a similar system in the Active Aeroelastic Wing study on a modified F/A-18. By careful selection of two receivers employing different focal lengths for different segments on the wing, they reported accuracy of less than 0.08 inch for displacement and 0.13° for twist. Kurita et al. [47] instrumented a JAXA Beechcraft Queen Air low wing research aircraft with a stereo-vision rig. The setup included two high resolution Charged Coupled Devices (CCD) cameras looking out through its cabin windows, capturing fiducial markers painted on the wing.

Foil strain gauges are electro-mechanic devices consisting of a metallic foil bonded on a flexible backing. The foil resistance changes due to mechanical deformation, which can be recorded on a voltage sensing device. FBG systems employ illumination sources which transmit light through optical fiber cables mounted on or embedded in the object to be measured. Each FBG sensor located within the optic fiber will reflect light of a specific wavelength, also known as the Bragg's wavelength, back to the transmitter. When external mechanical or thermal strain is applied, it alters the optical refractive index of the fiber core. This results in a shift in the reflected wavelength from the FBG sensor. By measuring this shift using a wavelength detector, the external disturbance can be recovered [48]. Since both strain gauges and FBGs measure strain data, reconstruction techniques for both sensors are similar. Ko et al. [49] derived kinematics relationship between strain and deflection. That work showed good comparison with numerical Finite Element Method (FEM) models ranging from simple canonical geometry (tubes and plates) to complex 3D FEM representation of wing box. Derkevorkian et al. [50] demonstrated the use of FBG sensors on an aluminum swept wing in a wind tunnel experiment. They developed a displacement transfer function (DTF) approach and compared it against a more traditional FEM modal approach. In DTF, each local segment is approximated using linear beam-curvature equation with some modification for numerical stability. The advantage is that DTF

does not require knowledge of the modal properties. Pak [51] proposed a novel two-step algorithm to predict the deflection and slope of a wing structure using measured strain at discrete locations. The first step consists of piecewise least-squares fitting of curvature using splines to reduce measurement noise. The curvature is integrated twice to obtain deflection. In the second step, computed deflection along the optical fiber is combined with a FE model using system equivalent reduction and expansion process (SEREP) to interpolate/extrapolate the deflection and slope of the entire structure. The accuracy of this new two-step method is excellent when compared with numerical results from MSC/NASTRAN.

Typically, there will be different sensor sources on a UAS. In addition, vision-based system have lower update rate compare to IMUs or accelerometers. It is desirable to implement a multi-rate sensor estimation algorithm to fuse all the available measurements. Baraniello et al. [52] numerically investigated the fusion of camera-LED system with distributed accelerometers to obtain mode shape estimation. The authors implemented a Kalman filter using pixel coordinates of the LED directly, incorporating a projection step from 3-D coordinates to 2-D image coordinates implicitly in the formulation. Armesto et al. [53] proposed a multi-rate Kalman filter update algorithm to fuse inertia measurements with vision data. They presented experimental results for a Kalman filter running at 5 ms with inertia data input at 10 ms and vision input at 35 ms. Smyth and Wu [54] also used multi-rate Kalman filter to fuse accelerometer data and GPS displacement data for large civil engineering structure. They proposed an additional smoothing step to obtain better estimation by utilizing forward filtering over the entire sequence of available measurements. This is done at the expense of online estimation as the filter is no longer causal.

In addition to measuring structural deformation for monitoring purposes, control laws need to be tailored to handle the effects of flexibility. This usually entails using state estimators or sensor fusion algorithms to recover structural state variables for

feedback control.

1.2.4 Aeroelastic Control of Very Flexible Aircraft

Maneuver control of VFA present interesting challenges which are not present in traditional aircraft or UAS. For conventional aircraft, control designs are based on rigid aircraft dynamics, with notch filters at the natural structural vibration frequencies to prevent excitation of structural motion. With VFA, structural frequencies interact with rigid-body dynamics, therefore such filtering techniques can no longer be applied [55, 56]. In certain situations, change in vehicle geometry due to wing deformation can change the system characteristics from a stable system to an unstable system. A notable example is the Helios mishap [3].

Pedro and Bigg [57] applied Linear Quadratic Gaussian (LQG) techniques to a longitudinal control of flexible aircraft. In that study, they focused on the handling qualities in the presence of wind turbulence and shear. Schirrer et al. [58] employed Linear Quadratic Regulator (LQR) techniques to shaping lateral dynamics of a flexible blended-wing-body (BWB) type aircraft. Although flexible aircraft is modeled, the flexible states are not regulated. The primary concern was with lateral rigid body control (e.g., roll, and side-slip angle). Tuzcu et al. [59] used full state LQR feedback and applied it to a HALE aircraft in three different configuration (quasi-rigid, restrained, and free flight). By performing eigenvalue analysis of the three configurations, they showed that coupling between rigid body and elastic motions is reflected in all of the eigen-solutions. It is no longer possible to distinguish "rigid body" modes or "elastic" modes in free flight. They also explicitly showed that a LQR control designed using quasi-rigid model actually destabilized the fully flexible aircraft. Gonzalez et al. [60] devised a loop separation technique for linear control of VFA. The inner loop consists of a stabilization control to transform the flexible aircraft into a slightly flexible one. Then, the outer loop can be designed with conventional rigid-body considerations.

Other modern techniques attempted by researchers include H_∞ control and adaptive control. Schmidt and Chavez [61] developed longitudinal controllers based upon μ-synthesis. Their work focused on developing a systematic framework to incorporate unsteady aerodynamic uncertainties, structural uncertainties and model stiffness uncertainties. Qu et al. [62] formulated an output adaptive feedback controller. They assumed bounded uncertainties in both plant as well as actuators dynamics. The baseline LQG controller is augmented with an adaptive term which is updated online. Patil [63] developed a Static Output Feedback (SOF) controller used for flutter suppression of HALE type aircraft. The resulting controller is of a much lower order than LQR/LQG type controllers. It is noted that with the proper choice of sensors, SOF can achieve almost identical performance compared to LQR/LQG controllers. Shearer and Cesnik [64, 65] developed a heuristic approach for trajectory control mimicking the actions of a human pilot. The outer loop comprises of a Proportional Integral Derivative (PID) controller for flight path angle and roll angle tracking. A dynamic inversion was implemented in the inner loop. Longitudinal and lateral dynamics were assumed decoupled and designed separately. Dillsaver et al. [66] implemented command and extended command governors for gust load alleviation. The governor uses maximal output set theory to determine if system state will exceed some user defined bounds and adjust the reference setpoint command accordingly.

It is important for the control laws to handle the effects of flexibility. At the same time, structural limits need to enforced to ensure the safe operation of VFAs. Model Predictive Control (MPC) is a modern control law capable of handling such constraints.

1.2.5 Model Predictive Control

Maneuvering loads or external gust disturbances can have a large impact on the internal loads in VFA. Traditional aircraft can mitigate this by increasing structural

strength at the expense of structural weight. This is usually not feasible for VFA due to weight and endurance requirements. One way to limit structural loads is for an active controller to monitor and adhere to state constraints (e.g., wing root strain, and wing acceleration). In addition, VFA usually do not have high control authority in order to minimize actuator weights. Control saturation can be a real issue if control authority is low. Hence, the ability to impose both state and control constraints are advantageous, if not critical, in VFA control.

Model Predictive Control (MPC) is a promising control strategy to achieve these requirements. It was first employed in the chemical process industry in 1980s. It has been very successful due to its ability to simultaneously handle constraints and provide optimal performance in an elegant optimization based framework. Mayne [67] and Eren et al. [68] summarized contemporary challenges and future growth areas for MPC. Although MPC is a powerful tool, both works highlighted high computational complexity as a disadvantage of online MPC. Typically, MPC can only be implemented in embedded systems with significant computational resources. Many researchers have worked on addressing real-time operation of MPC. Nesterov Gradient or Fast Gradient (FG) method is a popular projected gradient method to the optimization subproblem. Kögel and Findeisen [69] showed that it can be easily applied to input constrained MPC formulated in the condensed form. Zometa et al. [70] remarked that for such methods, the most costly computation is the quadratic cost term with complexity $O((Np)^2)$ where N is the prediction horizon and p is the control dimension. Generally, it is non-trivial to compute projected gradient in the presence of constraints. Analytical expression can only be written for simple box constraints of the form $u_{min} \leq u \leq u_{max}$. One might try to compute projected gradient by first reducing constraint equations to a tractable form using Fourier-Motzkin elimination. However, this is not typically feasible as Kelber [71] showed that this elimination algorithm takes non-polynomial time and the memory required to store additional

equations can grow dramatically. Kögel et al. [72, 73] extended FG with augmented Lagrange multipliers and system states, solving for the dual solution instead. This allows system state constraints to be written directly using box constraints instead of affine constraints when written in the condensed form. Jerez et al. [74], however, found that such dual formulation is not strongly concave. This will adversely affect the convergence speed. Instead, they worked with non-condensed form using Alternating Direction Method of Multipliers (ADMM). Ferreau et al. [75] developed an online active set method which exploits the structure of the MPC problem. In MPC, a control sequence is to be determined. This can be viewed as a multi-parametric optimization problem. The open-source qpOASES C++ MPC library [76] implements the theory developed in that work. The numerical results generated in this thesis employs that library.

While MPC is widely used in the chemical process industry, researchers have also applied MPC to control of UAS and VFA. Simpson et al. [77] applied MPC to a reduced order model obtained from balanced truncation of a cantilevered wing. They employed μAO-MPC [78] auto-generated C-code and managed real-time control suppressing wing response to a "1–cosine" gust within a numerical simulation environment. Giesseler et al. [79] attempted to reduce gust loads by using MPC and previewed gust measurements using LIDAR. They showed that the predictive nature of MPC can reduce wing bending loads significantly while constraints on control inputs and control input rates are respected. Haghighat et al. [80] applied MPC to a generic HALE UAS. The proposed MPC design performed better than LQR at regulating the maximum stress and the rigid-body parameters as a result of control saturation. Wang and Palacios [81] performed nonlinear MPC using a nonlinear reduced order model with linear and quadratic states and compared its performance with a linear MPC. They noted that linear MPC could lead to a different state trajectory in nonlinear plant, though the corresponding control trajectory differs only

slightly. Using nonlinear MPC yields lower cumulative cost compared to linear MPC.

1.2.6 Summary and Findings

From the literature review, it can be concluded that significant work has been done on modeling, simulation, and control of VFA. Figure 1.1 illustrates three potential areas of improvements and contributions.

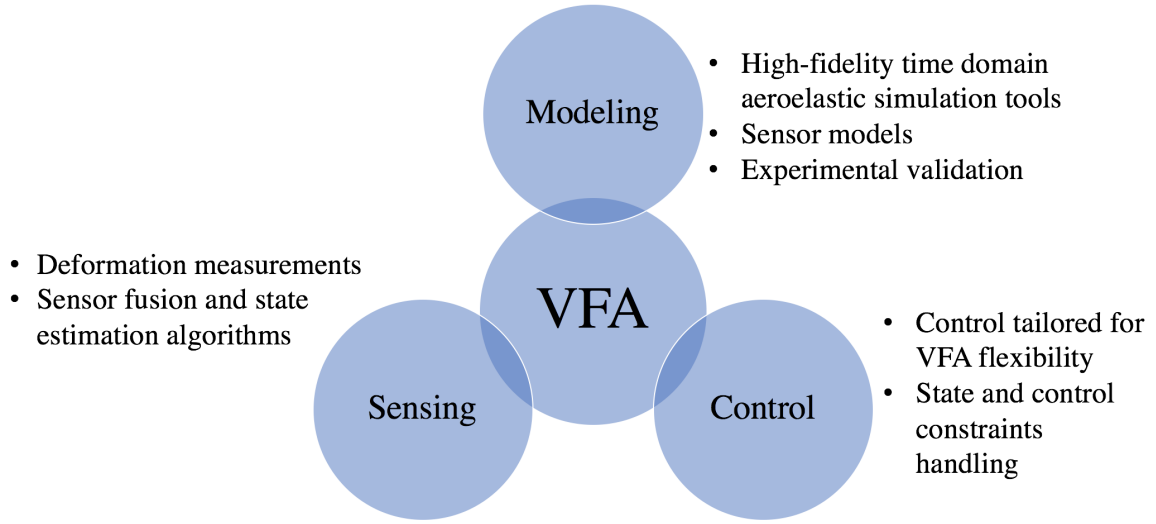

Figure 1.1: Aspects of VFA analysis and operation

There remains a need for numerical tools to analyze and predict VFA flight characteristics. In addition, numerical sensors models should be developed for evaluating structural monitoring or control feedback strategies. However, there is a clear lack of experimental data to corroborate numerical results. Therefore, building, flying, and collecting data on a specially instrumented aeroelastic UAS testbed would be a valuable contribution to the field of VFA aeroelasticity.

Obtaining high quality data on small UAS is in itself not a trivial task due to stringent restrictions on weight, power, and volume. Vision-based deformation measurement methods seem to be good candidates, but the implementation and overall accuracy need to be validated.

Researchers have applied different control techniques to control of VFA. Usually, they require structural feedback which is assumed to be available. Therefore, development of estimation or sensor fusion algorithms to provide such information will be very useful from a control design perspective.

Finally, even assuming full state feedback is available, interesting challenges in terms of control saturation (due to low control authority) and the need for state constraints (to prevent over-stress of flexible wings) remains. MPC handles both type of constraints naturally, making it a good candidate for VFA maneuver load control. An evaluation of MPC compared to traditional LQR will be insightful.

1.3 Dissertation Outline

This dissertation aims to answer some of the motivating questions raised in the previous section. Chapter 2 first summarizes past theoretical development of Nonlinear Aeroelastic Simulation Toolbox (UM/NAST). Next, the chapter presents the derivation of the kinematic equations for modeling sensors and its computational and implementation aspects within UM/NAST. A new numerical linearization procedure is also developed to allow linearized dynamics of both the plant and sensor measurements to be easily obtained. Chapter 3 presents the theoretical background of computer stereo-vision. The developed stereo vision-based measurement system is then benchmarked experimentally to ascertain its accuracy. Chapter 4 details the theoretical background and formulation of MPC for control of VFA. Next, given a multitude of measurement sources, two sensor fusion algorithms for estimation of structural states are proposed. The last section in this chapter presents system identification procedure to obtain plant dynamics from experimental data. Chapter 5 details the design, modeling and system integration of an aeroelastic testbed (X-HALE). Two variants are produced: a fully instrumented aeroelastic testbed ATV-6B

and Risk Reduction Vehicle RRV-6B. Chapter 6 presents numerical studies validating the developed virtual sensor kinematics. Next, sensor fusion algorithms using different sensor combinations are investigated with and without noise. Finally, numerical simulations of MPC demonstrating maneuver load alleviation are presented. Chapter 7 presents experimental flight results obtained from RRV-6B. First, the plant transfer functions are identified from flight segments where suitable excitation signals are injected. Based on this identified plant, a stabilization autopilot is designed. Improvements in handling qualities are demonstrated in subsequent flights. Chapter 8 concludes this dissertation and outlines future areas of studies and improvements.

CHAPTER 2

UM/NAST Theoretical and Numerical Developments

In this chapter, the underlying theoretical foundation behind UM/NAST is presented. First, the strain-based finite element aeroelastic formulation, coupled with flight dynamic equations of motion is presented in Section 2.1. A new numerical linearization scheme is developed. Next, kinematic relationship of virtual sensors (e.g., accelerometer, orientation sensor) are developed in Section 2.2. Linearization of sensor kinematic equations are also performed numerically. Finally, Section 2.3 details the port and reorganization of UM/NAST to C++ for computational efficiency and future extensibility.

2.1 Flight Dynamic Coupled Aeroelastic Equations of Motion

Strain-based formulation in UM/NAST have been developed by Cesnik and coworkers [65, 82, 83]. A brief summary of their work is presented to aid understanding of further theoretical development in subsequent sections.

2.1.1 Strain-based Formulation

Each beam element has three nodes with four degrees of freedom: extensional, twist and two orthogonal bending curvature of the beam reference line. The strain vector within each beam element is assumed constant and is denoted by:

$$\varepsilon_e = \begin{bmatrix} \varepsilon_x & \kappa_x & \kappa_y & \kappa_z \end{bmatrix}^T \tag{2.1}$$

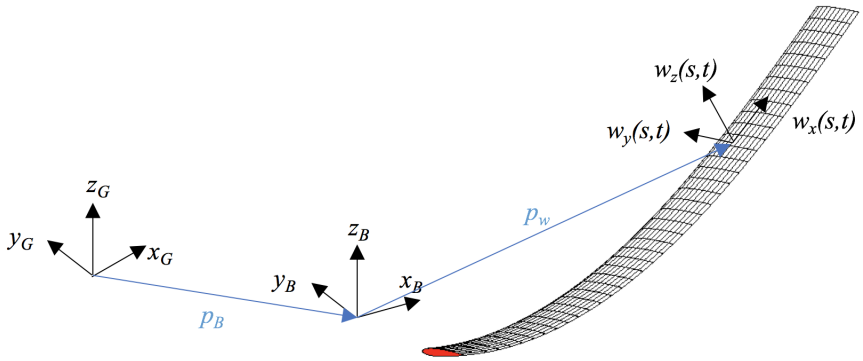

Figure 2.1: UM/NAST beam coordinates

The position and orientation of each point can be described by a position vector and three directrix vectors, resulting in 12-component h vector. The relative position of a point from the body origin is useful for expressing structural information. This will be denoted as h_w. For a point at location s from the origin shown in Fig. 2.1:

$$h(s) = \begin{bmatrix} p_B + p_w(s) & w_x(s) & w_y(s) & w_z(s) \end{bmatrix}^T \tag{2.2}$$

$$h_w(s) = \begin{bmatrix} p_w(s) & w_x(s) & w_y(s) & w_z(s) \end{bmatrix}^T \tag{2.3}$$

The rigid body motion of the body frame is described by the three linear and three angular velocities expressed in the body frame as:

$$\beta = \begin{bmatrix} v_B & \omega_B \end{bmatrix}^T \tag{2.4}$$

The orientation and inertial displacement of the body frame are given by quaternions ζ and displacement vector P_B as:

$$\zeta = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \end{bmatrix}^T \tag{2.5}$$

$$P_B = \begin{bmatrix} x_B & y_B & z_B \end{bmatrix}^T \tag{2.6}$$

Given nodal directrix vectors, the direction cosine matrix from the local beam frame w to the body frame B can be expressed as:

$$C^{Bw} = \begin{bmatrix} w_x & w_y & w_z \end{bmatrix} \tag{2.7}$$

Coordinate transformation between body and beam frames can be computed by:

$$\left\{ . \right\}_B = C^{Bw} \left\{ . \right\}_w \tag{2.8}$$

The rotation matrix from body frame B to inertial frame G can be computed from quaternions ζ as:

$$C^{BG} = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1 q_2 + q_0 q_3) & 2(q_1 q_3 - q_0 q_2) \\ 2(q_1 q_2 - q_0 q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2 q_3 + q_0 q_1) \\ 2(q_1 q_3 + q_0 q_2) & 2(q_2 q_3 - q_0 q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \tag{2.9}$$

The position and orientation of a point on the beam reference line can be recovered from the strain and boundary node $h_{BC,w}$ using kinematic relation:

$$h_w(s) = e^{K(s-s_0)} h_{BC,w} = e^{G(s)} h_{BC,w} \tag{2.10}$$

where

$$K = \begin{bmatrix} 0 & 1+\varepsilon_x & 0 & 0 \\ 0 & 0 & \kappa_z & -\kappa_y \\ 0 & -\kappa_z & 0 & \kappa_x \\ 0 & \kappa_y & -\kappa_x & 0 \end{bmatrix}_{12\times 12} \tag{2.11}$$

Nodal displacement can then be computing by marching Eq. (2.10) from root to tip between connecting beam elements.

Unsteady wake effects are modeled using 6^{th}-order finite inflow states for each lifting surface element [13], i.e.,

$$\lambda = \begin{bmatrix} \lambda_1 & \lambda_2 & \lambda_3 & \lambda_4 & \lambda_5 & \lambda_6 \end{bmatrix}^T \tag{2.12}$$

Unsteady aerodynamics comprise of apparent mass effect, lift/moment/drag due to local effective angle of attack and control surface contributions [15, 84] and are given by:

$$l_{ac} = \pi\rho b_c^2 \left(-\ddot{z} + \dot{y}\dot{\alpha} - d\ddot{\alpha}\right) + \rho b_c \dot{y}^2 \left[c_l(\alpha_{eff}) + c_{l_\delta}\delta\right] \tag{2.13}$$

$$m_{ac} = \pi\rho b_c^3 \left[\frac{1}{2}\ddot{z} - \dot{y}\dot{\alpha} - \left(\frac{1}{8}b_c - \frac{1}{2}d\right)\ddot{\alpha}\right] + 2\rho b_c^2 \dot{y}^2 \left[c_m(\alpha_{eff}) + c_{m_\delta}\delta\right] \tag{2.14}$$

$$d_{ac} = -\rho b_c \dot{y}^2 \left[c_d(\alpha_{eff}) + c_{d_\delta}\delta\right] \tag{2.15}$$

where the local effective angle of attack α_{eff} is:

$$\alpha_{eff} = -\frac{\dot{z}}{\dot{y}} + \left(\frac{1}{2}b_c - d\right)\frac{\dot{\alpha}}{\dot{y}} - \frac{\lambda_0}{\dot{y}} \tag{2.16}$$

and:

$$\lambda_0 = \frac{1}{2}\sum_{n=1}^{6} b_n \lambda_n \tag{2.17}$$

This effective angle of attack comprises of the pitching motion, plunging motion,

local airfoil inclination, and unsteady wake effects. b_n terms are computed based on geometry of the airfoil and details are given in Ref. [13].

2.1.2 Full Coupled Aeroelastic EOM

Finally, the coupled aeroelastic equations of motion can be expressed as [15, 16]:

$$\begin{bmatrix} M_{FF}(\varepsilon) & M_{FB}(\varepsilon) \\ M_{BF}(\varepsilon) & M_{BB}(\varepsilon) \end{bmatrix} \begin{Bmatrix} \ddot{\varepsilon} \\ \dot{\beta} \end{Bmatrix} + \begin{bmatrix} C_{FF}(\varepsilon, \dot{\varepsilon}, \beta) & C_{FB}(\varepsilon, \dot{\varepsilon}, \beta) \\ C_{BF}(\varepsilon, \dot{\varepsilon}, \beta) & C_{BB}(\varepsilon, \dot{\varepsilon}, \beta) \end{bmatrix} \begin{Bmatrix} \dot{\varepsilon} \\ \beta \end{Bmatrix} + \begin{bmatrix} K_{FF} & 0 \\ 0 & 0 \end{bmatrix} \begin{Bmatrix} \varepsilon \\ b \end{Bmatrix} = \begin{Bmatrix} R_F\left(\varepsilon, \dot{\varepsilon}, \ddot{\varepsilon}, \beta, \dot{\beta}, \lambda, \zeta, u\right) \\ R_B\left(\varepsilon, \dot{\varepsilon}, \ddot{\varepsilon}, \beta, \dot{\beta}, \lambda, \zeta, u\right) \end{Bmatrix} \tag{2.18}$$

$$\dot{\zeta} = -\frac{1}{2}\Omega_\zeta(\beta)\zeta \tag{2.19}$$

$$\dot{P}_B = \begin{bmatrix} C^{GB}(\zeta) & 0 \end{bmatrix} \beta \tag{2.20}$$

$$\dot{\lambda} = F_1(\varepsilon, \dot{\varepsilon}, \beta) \begin{Bmatrix} \ddot{\varepsilon} \\ \dot{\beta} \end{Bmatrix} + F_2(\varepsilon, \dot{\varepsilon}, \beta) \begin{Bmatrix} \dot{\varepsilon} \\ \beta \end{Bmatrix} + F_3\lambda \tag{2.21}$$

The above set of equations of motion can be integrated using various time marching schemes. Brown [82] developed a trapezoidal scheme, which is computationally simple. However, long term numerical stability may pose issues. Shearer and Cesnik [85] developed a modified generalized-α scheme which is robust numerically at the expense of computation cost. Theoretical developments for UM/NAST not covered here include skin wrinkling effect on structural stiffness [86], inclusion of gust excitation [87], and displacement constraints for joint wing configurations [83].

Finally, $2 \times 4n$ elastic states, 13 rigid-body states and $6m$ inflow states (assuming six inflow states are used per lifting element) make up the dimension of the problem, where n is the number of flexible strain elements and m is the number of lifting elements.

Brown [82] showed that Eqs. (2.18) – (2.21) can be re-written as a first order set of ordinary differential equation of the form:

$$Q_1(q)\dot{q} = Q_2(q)q + R(q, \dot{q}, u, v_g) \tag{2.22}$$

with:

$$q = \begin{bmatrix} \varepsilon & \dot{\varepsilon} & \beta & \zeta & P_B & \lambda \end{bmatrix}^T \tag{2.23}$$

$$u = \begin{bmatrix} u_1 & u_2 & u_3 & \cdots \end{bmatrix}^T \tag{2.24}$$

$$v_g = \begin{bmatrix} v_{g_1} & v_{g_2} & \cdots \end{bmatrix}^T \tag{2.25}$$

where q is the first order system states, u is the control states and v_g is the nodal gust velocity. Integrating using trapezoidal integration scheme, one has:

$$q_{k+1} = \left(Q_1 - \frac{1}{2}\Delta t Q_2\right)^{-1} \left[\left(Q_1 + \frac{1}{2}\Delta t Q_2\right) q_k + \Delta t R\right] \tag{2.26}$$

2.1.3 Linearization of Full Coupled EOM

The linearized equations of motion can be expressed as:

$$\begin{Bmatrix} \dot{\tilde{q}}_1 \\ \dot{\tilde{q}}_2 \\ \vdots \\ \dot{\tilde{q}}_{nq} \end{Bmatrix} = \begin{Bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \vdots \\ \dot{q}_{nq} \end{Bmatrix}_{perturbed} - \begin{Bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \vdots \\ \dot{q}_{nq} \end{Bmatrix}_{ref} \tag{2.27}$$

$$\dot{\tilde{q}} = \tilde{Q}_1^{-1}\tilde{Q}_2\tilde{q} + \tilde{Q}_1^{-1}\frac{\partial R}{\partial u}\tilde{u} + \tilde{Q}_1^{-1}\frac{\partial R}{\partial v_g}\tilde{v}_g \tag{2.28}$$

The tilde symbol indicates perturbation from some reference condition. Linearization matrices $\tilde{Q}_1$, $\tilde{Q}_2$, $\dfrac{\partial R}{\partial u}$, and $\dfrac{\partial R}{\partial v_g}$ are derived by Cesnik and co-workers in Refs. [14, 16, 87].

A new numerical linearization is developed in this work. Performing Fourier series expansion on Eqs. (2.18) – (2.21):

$$\dot{\tilde{q}} = A\tilde{q} + B\tilde{u} + B_w\tilde{v}_g + H.O.T \tag{2.29}$$

Each state can be perturbed separately and the state velocity (LHS of Eqs. (2.18) – (2.21)) can be computed. This state velocity will correspond to the columns of the A state matrix. For example, perturbing the first state yields:

$$\begin{Bmatrix} \dot{\tilde{q}}_1 \\ \dot{\tilde{q}}_2 \\ \vdots \\ \dot{\tilde{q}}_{n_q} \end{Bmatrix} = \begin{bmatrix} | & | & & | \\ a_1 & a_2 & \dots & a_{n_q} \\ | & | & & | \end{bmatrix} \begin{Bmatrix} \tilde{q}_1 \\ 0 \\ \vdots \\ 0 \end{Bmatrix} + \begin{bmatrix} | & | & & | \\ b_1 & b_2 & \dots & b_{n_u} \\ | & | & & | \end{bmatrix} \begin{Bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{Bmatrix} + \begin{bmatrix} | & | & & | \\ b_{w1} & b_{w2} & \dots & b_{n_{v_g}} \\ | & | & & | \end{bmatrix} \begin{Bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{Bmatrix} \tag{2.30}$$

Therefore, one has:

$$a_1^T = \frac{1}{\tilde{q}_1} \begin{bmatrix} \dot{\tilde{q}}_1 & \dot{\tilde{q}}_2 & \cdots & \dot{\tilde{q}}_{n_q} \end{bmatrix}^T \tag{2.31}$$

The procedure is similar for all other states, including control state u and disturbances v_g. A one-step forward difference scheme is implemented in this work. Other difference schemes (e.g., backwards, or central difference) can also be used.

The numerical linearization scheme for quaternion states ζ require special consid-

erations. Quaternion states cannot be perturbed separately since $\|\zeta\|_2 = 1$. Therefore, the quaternion states are perturbed jointly to give four sets of equations. Each set of quaternion should be linearly independent from other sets. By selecting the quaternions in this fashion, a system of equations with $4n_q$ unknowns (from columns of the A matrix corresponding to $\zeta_0, \zeta_1, \zeta_2$ and ζ_3) with $4n_q$ equations (corresponding to four sets of state velocity vector $\dot{\tilde{q}}$) can be written. Let $\dot{\tilde{q}}^i$ denote the state velocity vector corresponding to quaternion set ζ^i. Perturbing only the quaternions states and setting the remaining states and control to zeros:

$$\begin{Bmatrix} \dot{\tilde{q}}_1 \\ \dot{\tilde{q}}_2 \\ \vdots \\ \dot{\tilde{q}}_{n_q} \end{Bmatrix}^i = \begin{bmatrix} | & | & | & | \\ a_{\zeta_0} & a_{\zeta_1} & a_{\zeta_2} & a_{\zeta_3} \\ | & | & | & | \end{bmatrix} \begin{Bmatrix} \zeta_0 \\ \zeta_1 \\ \zeta_2 \\ \zeta_3 \end{Bmatrix}^i \tag{2.32}$$

Rearranging Eq. (2.32) as a linear system of equations:

$$\Theta_{4\times n_q} = \Phi_{4\times 4_q} \Psi_{4\times n_q} \tag{2.33}$$

where

$$\Theta_{4\times n_q} \equiv \begin{bmatrix} \left(\dot{\tilde{q}}^a\right)^T \\ \left(\dot{\tilde{q}}^b\right)^T \\ \left(\dot{\tilde{q}}^c\right)^T \\ \left(\dot{\tilde{q}}^d\right)^T \end{bmatrix} \quad \Phi_{4\times 4} \equiv \begin{bmatrix} \left(\tilde{\zeta}^a\right)^T \\ \left(\tilde{\zeta}^b\right)^T \\ \left(\tilde{\zeta}^c\right)^T \\ \left(\tilde{\zeta}^d\right)^T \end{bmatrix} \quad \Psi_{4\times n_q} \equiv \begin{bmatrix} \left(a_{\zeta_0}\right)^T \\ \left(a_{\zeta_1}\right)^T \\ \left(a_{\zeta_2}\right)^T \\ \left(a_{\zeta_3}\right)^T \end{bmatrix} \tag{2.34}$$

If all quaternion sets are linearly independent, the matrix $\Phi_{4\times 4}$ is invertible. Thus, the columns of the A matrix can be read off rows of $\Psi_{4\times n_q}$, computed from:

$$\Psi_{4\times n_q} = \left(\Phi_{4\times 4}\right)^{-1} \Theta_{4\times n_q} \tag{2.35}$$

The continuous time state-space form can be converted to discrete time state-space

by selecting a time step and numerical integration scheme, so that:

$$\tilde{q}_{k+1} = A_d\tilde{q}_k + B_d\tilde{u}_k + B_{wd}\tilde{v}_{gk} \tag{2.36}$$

In this dissertation, the discrete time state-space plant is used for control design and Kalman filter implementation (Chapter 4).

2.2 Theoretical Development of Virtual Sensors

A new sensor frame denoted by S (Fig. 2.2) is defined to allow virtual sensors to be placed in any orientation with respect to the local structure. It is usual for a physical sensor to be rigidly mounted to the local structure, but not aligned with the local beam structural axis.

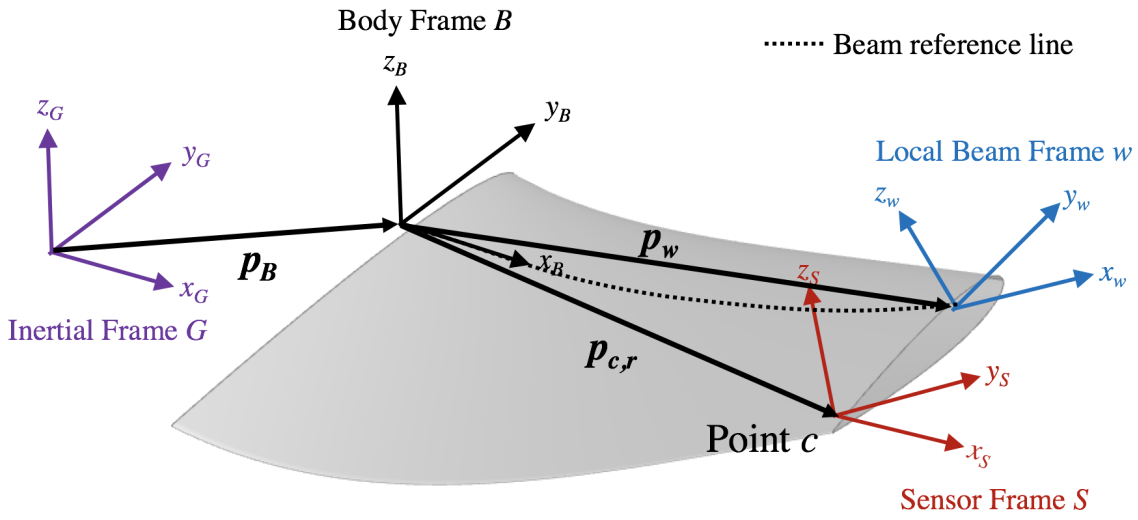

Figure 2.2: Sensor frame definition

2.2.1 Kinematics

First, consider an arbitrary point c in 3-D volume, some distance away from the beam reference line. It is the sum of the distance p_B of the body origin from inertia origin, and the relative distance $p_{c,r}$ of the point c from the body frame origin:

$$\begin{aligned} p_c &= p_B + p_{c,r} \\ &= p_B + p_w + y_S w_y + z_S w_z \end{aligned} \tag{2.37}$$

It is assumed that cross section does not deform, but is free to displace and rotate. For generality, assume the body frame is rotating with respect to the inertial frame at angular speed ω_B. Taking the time derivative to obtain the inertial velocity expressed in body frame, Shearer [65] derived it to be:

$$\begin{aligned} \frac{d}{dt}(p_c) &= (\dot{p}_B + \tilde{\omega}_B p_B) + (\dot{p}_w + y_S \dot{w}_y + z_S \dot{w}_z) + \tilde{\omega}_B (p_w + y_S w_y + z_S w_z) \\ &= v_B + \dot{p}_w + y_S \dot{w}_y + z_S \dot{w}_z + \tilde{\omega}_B (p_w + y_S w_y + z_S w_z) \end{aligned} \tag{2.38}$$

where v_B is the velocity of the body frame origin with respect to the inertial frame origin, expressed in the body frame. The tilde notation represents the cross product matrix, i.e.,

$$a \equiv \begin{Bmatrix} a_1 \\ a_2 \\ a_3 \end{Bmatrix} \quad \tilde{a} \equiv \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \tag{2.39}$$

Taking the derivative again to obtain inertial acceleration, expressed in the body frame, one gets [65]:

$$\begin{aligned} \frac{d^2}{dt^2}(p_c) = \dot{v}_B + \tilde{\omega}_B v_B + \ddot{p}_w + y_S \ddot{w}_y + z_S \ddot{w}_z + \dot{\tilde{\omega}}_B (p_w + y_S w_y + z_S w_z) \\ + 2\tilde{\omega}_B (\dot{p}_w + y_S \dot{w}_y + z_S \dot{w}_z) + \tilde{\omega}_B \tilde{\omega}_B (p_w + y_S w_y + z_S w_z) \end{aligned} \tag{2.40}$$

The angular rotation rate of a local sensor frame attached to point c with respect

to the inertia frame is the vector sum of: 1) the angular rate between body frame B and the inertia frame G, 2) the angular rate between the local beam frame w and the body frame B resulting from elastic structural deformation and, 3) the angular rate between local user defined sensor frame S and the local beam frame w. This can be expressed in the body frame as:

$$\omega_c = \omega_B + \omega_w + \omega_S \tag{2.41}$$

If the local sensor frame is fixed with respect to the local beam frame, $\omega_S = 0$.

Next, the attitude of the local sensor frame with respect to the inertia frame can be written using rotational matrices. This total rotation matrix can be decomposed as consecutive rotations between intermediate frames as: 1) from inertial frame G to body frame B, 2) body frame B to local beam frame w and, 3) local beam frame w to sensor frame S:

$$C^{SG}_{sensor} = C^{Sw} C^{wB} C^{BG} \tag{2.42}$$

If the local sensor frame is identical to the local beam frame, $C^{Sw} = I_{3\times3}$. Also, note that this last rotation matrix is not dependent on the elastic deformation or rigid body motion, instead, it is only dependent on mounting geometry.

Therefore, inertial acceleration of point c expressed in the local sensor frame can be written as:

$$a_{sensor} = C^{SB} \ddot{p}_c = C^{Sw} C^{wB} \ddot{p}_c \tag{2.43}$$

Similarly, the sensor frame angular rate expressed in the local sensor frame can be written as:

$$\omega_{sensor} = C^{SB} \omega_c = C^{Sw} C^{wB} \omega_c \tag{2.44}$$

The *relative* position of point c with respect to the body frame origin expressed

in the local sensor frame S can be written as:

$$p_{sensor,r} = C^{SB} p_{c,r} = C^{Sw} C^{wB} p_{c,r} \tag{2.45}$$

The attitude of the body frame B with respect to the inertial frame G in Euler angle representation can be recovered from quaternions:

$$\begin{aligned} \phi_B &= \arctan\left(\frac{2q_2q_3 + 2q_0q_1}{q_0^2 - q_1^2 - q_2^2 + q_3^2}\right) \\ \theta_B &= -\arcsin\left(2q_1q_3 - 2q_0q_2\right) \\ \psi_B &= \arctan\left(\frac{2q_1q_2 + 2q_0q_3}{q_0^2 + q_1^2 - q_2^2 - q_3^2}\right) \end{aligned} \tag{2.46}$$

Similarly, Euler angles measured at the sensor can be converted back to quaternions as:

$$\zeta = \begin{Bmatrix} \cos\left(\psi_B/2\right) \\ 0 \\ 0 \\ \sin\left(\psi_B/2\right) \end{Bmatrix} \begin{Bmatrix} \cos\left(\theta_B/2\right) \\ 0 \\ \sin\left(\theta_B/2\right) \\ 0 \end{Bmatrix} \begin{Bmatrix} \cos\left(\phi_B/2\right) \\ \sin\left(\phi_B/2\right) \\ 0 \\ 0 \end{Bmatrix} \tag{2.47}$$

Finally, attitude of the sensor frame S with respect to the inertial frame G using Euler angle representation can be recovered from the rotation matrix C^{SG}:

$$\begin{aligned} \phi_S &= \arctan\left(\frac{C_{23}^{SG}}{C_{33}^{SG}}\right) \\ \theta_S &= -\arcsin\left(C_{13}^{SG}\right) \\ \psi_S &= \arctan\left(\frac{C_{12}^{SG}}{C_{11}^{SG}}\right) \end{aligned} \tag{2.48}$$

2.2.2 Computation

This subsection employs structural Jacobians derived by Cesnik and co-workers [15, 65] for computing virtual sensor kinematics. Recall that:

$$dh_w = J_{h\varepsilon} d\varepsilon \tag{2.49}$$

$$dh = J_{h\varepsilon} d\varepsilon + J_{hb} db \tag{2.50}$$

$$d\theta = J_{\theta\varepsilon} d\varepsilon + J_{\theta b} db \tag{2.51}$$

where

$$J_{h\varepsilon} \equiv \frac{\partial h}{\partial \varepsilon},\ J_{hb} \equiv \frac{\partial h}{\partial b}, J_{\theta\varepsilon} \equiv \frac{\partial \theta}{\partial \varepsilon},\ J_{\theta b} \equiv \frac{\partial \theta}{\partial b} \tag{2.52}$$

Rewriting Eqs. (2.43) – (2.48) in terms of strain ε, strain rate $\dot{\varepsilon}$, rigid body motion β, and body attitude ζ using structural Jacobians:

$$\begin{aligned} \omega_c &= \omega_B + \omega_w \\ &= J_{\theta b}\beta + J_{\theta\varepsilon}\dot{\varepsilon} \end{aligned} \tag{2.53}$$

$$\begin{aligned} \frac{d}{dt}(p_c) &= \begin{bmatrix} I_{3\times3} & 0_{3\times3} & y_S I_{3\times3} & z_S I_{3\times3} \end{bmatrix} \left(\begin{Bmatrix} \dot{p}_w \\ \dot{w}_x \\ \dot{w}_y \\ \dot{w}_z \end{Bmatrix} + \begin{bmatrix} I_{3\times3} & \tilde{p}_w^T \\ 0_{3\times3} & \tilde{w}_x^T \\ 0_{3\times3} & \tilde{w}_y^T \\ 0_{3\times3} & \tilde{w}_z^T \end{bmatrix} \begin{Bmatrix} v_B \\ \omega_B \end{Bmatrix} \right) \\ &= \begin{bmatrix} I_{3\times3} & 0_{3\times3} & y_S I_{3\times3} & z_S I_{3\times3} \end{bmatrix} (J_{h\varepsilon}\dot{\varepsilon} + J_{hb}\beta) \end{aligned} \tag{2.54}$$

$$\frac{d^2}{dt^2}(p_c) = \begin{bmatrix} I_{3\times3} & 0_{3\times3} & y_S I_{3\times3} & z_S I_{3\times3} \end{bmatrix} \left(\left(\begin{Bmatrix} \ddot{p}_w \\ \ddot{w}_x \\ \ddot{w}_y \\ \ddot{w}_z \end{Bmatrix} + \begin{bmatrix} I_{3\times3} & \tilde{p}_w^T \\ 0_{3\times3} & \tilde{w}_x^T \\ 0_{3\times3} & \tilde{w}_y^T \\ 0_{3\times3} & \tilde{w}_z^T \end{bmatrix} \begin{Bmatrix} \dot{v}_B \\ \dot{\omega}_B \end{Bmatrix} \right.$$
$$\left. + \begin{bmatrix} \tilde{\omega}_B & \tilde{\omega}_B \tilde{p}_w^T \\ 0_{3\times3} & \tilde{\omega}_B \tilde{w}_x^T \\ 0_{3\times3} & \tilde{\omega}_B \tilde{w}_y^T \\ 0_{3\times3} & \tilde{\omega}_B \tilde{w}_z^T \end{bmatrix} \begin{Bmatrix} v_B \\ \omega_B \end{Bmatrix} + 2 \begin{bmatrix} 0_{3\times3} & \dot{\tilde{p}}_w^T \\ 0_{3\times3} & \dot{\tilde{w}}_x^T \\ 0_{3\times3} & \dot{\tilde{w}}_y^T \\ 0_{3\times3} & \dot{\tilde{w}}_z^T \end{bmatrix} \begin{Bmatrix} v_B \\ \omega_B \end{Bmatrix} \right)$$
$$= \begin{bmatrix} I_{3\times3} & 0_{3\times3} & y_S I_{3\times3} & z_S I_{3\times3} \end{bmatrix} \left(J_{h\varepsilon}\ddot{\varepsilon} + \dot{J}_{h\varepsilon}\dot{\varepsilon} + J_{h\beta}\dot{\beta} + H_{hb}\beta + 2H_{h\dot{\varepsilon}\dot{\beta}}\beta \right) \tag{2.55}$$

Finally, expressing the measurements in local sensor frame S,

$$p_{sensor,r} = C^{SB} \begin{bmatrix} I_{3\times3} & 0_{3\times3} & y_S I_{3\times3} & z_S I_{3\times3} \end{bmatrix} h_w \tag{2.56}$$

$$\Theta_{sensor} = \begin{bmatrix} \arctan\left(\frac{C_{23}^{SG}}{C_{33}^{SG}}\right) & -\arcsin\left(C_{13}^{SG}\right) & \arctan\left(\frac{C_{12}^{SG}}{C_{11}^{SG}}\right) \end{bmatrix} \tag{2.57}$$

$$\omega_{sensor} = C^{SB}\left(J_{\theta\varepsilon}\dot{\varepsilon} + J_{\theta b}\beta\right) \tag{2.58}$$

$$v_{sensor} = C^{SB} \begin{bmatrix} I_{3\times3} & 0_{3\times3} & y_S I_{3\times3} & z_S I_{3\times3} \end{bmatrix} \left(J_{h\varepsilon}\dot{\varepsilon} + J_{h\beta}\beta\right) \tag{2.59}$$

$$a_{sensor} = C^{SB} \begin{bmatrix} I_{3\times3} & 0_{3\times3} & y_S I_{3\times3} & z_S I_{3\times3} \end{bmatrix} \left(J_{h\varepsilon}\ddot{\varepsilon} + \dot{J}_{h\varepsilon}\dot{\varepsilon} + J_{h\beta}\dot{\beta} + H_{hb}\beta + 2H_{h\dot{\varepsilon}\dot{\beta}}\beta \right) \tag{2.60}$$

One can compute the structural Jacobians at point w by first principles. However, by solving the equation of motion for the entire model, the h vectors at the FE nodes of a strain beam element have already been obtained. Structural Jacobians at the FE nodes are also computed as part of the solution process. Therefore, one can leverage on existing results to reduce computational complexity.

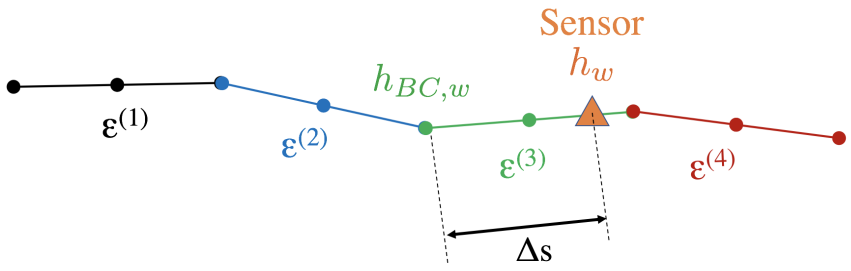

Figure 2.3: Sensor kinematics and geometry

For example, consider the procedure to compute the structural Jacobian at point w for a simple beam (Fig. 2.3). In this case, point w is on the third element, at a distance Δs away from the boundary node (first node of the third element). $J_{h\varepsilon}$ has the following structure:

$$\begin{aligned} h_w &= e^{K\left(\varepsilon^{(3)}\right)\Delta s} h_{BC,w} \\ h &= h_w + p_B \\ J_{h\varepsilon} &= \frac{\partial h}{\partial \varepsilon} = \left[e^{K\left(\varepsilon^{(3)}\right)\Delta s} \frac{\partial h_{BC}}{\partial \varepsilon^{(1)}} \quad e^{K\left(\varepsilon^{(3)}\right)\Delta s} \frac{\partial h_{BC}}{\partial \varepsilon^{(2)}} \quad \frac{\partial e^{K\left(\varepsilon^{(3)}\right)\Delta s}}{\partial \varepsilon^{(3)}} h_{BC} \quad 0 \right] \end{aligned} \tag{2.61}$$

Note that $\frac{\partial h_{BC}}{\partial \varepsilon^{(1)}}$ and $\frac{\partial h_{BC}}{\partial \varepsilon^{(2)}}$ are known nodal values as part of the solution process. New terms which needs to be computed are $e^{K(e^{\varepsilon^{(3)}})\Delta s}$ and $\frac{\partial e^{K\left(\varepsilon^{(3)}\right)\Delta s}}{\partial \varepsilon^{(3)}}$. The rest of the structural Jacobians are computed from components of the h_w vector. Readers are directed to Refs. [15, 65] for more information.

2.2.3 Linearization

The linearized output relation for Eqs. (2.56) – (2.60) can be obtain analytically or numerically. Partial derivatives of sensor measurement equations are taken with respect to the system states q. Following linearizing procedure proposed by Refs. [83, 88], analytical linearization can be made more tractable by assuming the structural Jacobians are constant. In this dissertation, numerical linearization is carried out

instead.

Let $y = [y_1, y_2, \ldots, y_m]^T$ denote some virtual sensor outputs. Performing Fourier series expansion on the sensor measurement equations yields:

$$\tilde{y} = y_{perturbed} - y_{ref} \tag{2.62}$$

$$\tilde{y} = C\tilde{q} + H.O.T \tag{2.63}$$

By perturbing the system state $\tilde{q}$ individually, the columns of output C matrix can be computed. For example, by perturbing the first state yields:

$$\begin{Bmatrix} \tilde{y}_1 \\ \tilde{y}_2 \\ \vdots \\ \tilde{y}_m \end{Bmatrix} = \begin{bmatrix} | & | & & | \\ c_1 & c_2 & \ldots & c_m \\ | & | & & | \end{bmatrix} \begin{Bmatrix} \tilde{q}_1 \\ 0 \\ \vdots \\ 0 \end{Bmatrix} \tag{2.64}$$

and therefore,

$$c_1^T = \frac{1}{\tilde{q}_1} \begin{bmatrix} \tilde{y}_1 & \tilde{y}_2 & \cdots & \tilde{y}_m \end{bmatrix}^T \tag{2.65}$$

Special treatment is also applied to quaternion states using Eq. (2.35), replacing Φ, Θ, and Ψ variables with the following definitions:

$$\Theta_{4\times m} \equiv \begin{bmatrix} (\tilde{y}^a)^T \\ (\tilde{y}^b)^T \\ (\tilde{y}^c)^T \\ (\tilde{y}^d)^T \end{bmatrix} \quad \Phi_{4\times 4} \equiv \begin{bmatrix} \left(\tilde{\zeta}^a\right)^T \\ \left(\tilde{\zeta}^b\right)^T \\ \left(\tilde{\zeta}^c\right)^T \\ \left(\tilde{\zeta}^d\right)^T \end{bmatrix} \quad \Psi_{4\times m} \equiv \begin{bmatrix} (c_{\zeta_0})^T \\ (c_{\zeta_1})^T \\ (c_{\zeta_2})^T \\ (c_{\zeta_3})^T \end{bmatrix} \tag{2.66}$$

2.3 UM/NAST C++ Software Architecture

The original UM/NAST theory plus additions developed above are completely rewritten in C++ from the original MATLAB implementation. This is done primarily

for computational efficiency and extensibility, as well as future maintainability of the code. Open-source libraries used in the new code are listed in Table 2.1.

Table 2.1: UM/NAST external libraries

Library	Version	License	Purpose
Eigen	3.2.5	MPL2 [89]	Matrix manipulation and linear algebra
HDF5	1.8.19	BSD [90]	Data storage library
PugiXML	1.6	MIT [91]	XML parser
qpOASES	3.2	LGPLv2 [92]	MPC solver
Eigen-HDF5	–	MIT [91]	Interfacing library for Eigen and HDF5

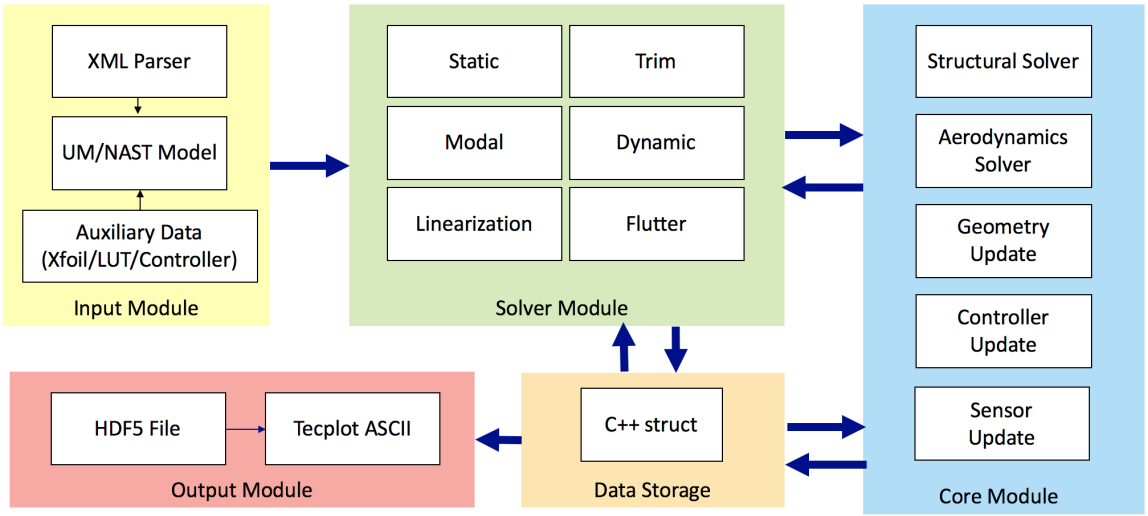

Figure 2.4: UM/NAST software architecture

The code consist of four main parts: input, core, solver, and output modules (Fig. 2.4). The input module parses the XML-based input file, processes the model properties, and saves into `Nast_Model` C++ class object. The core module consists of common functions used by all the solvers. They include geometry, load, aerodynamic, controller and sensor update subroutines (Fig. 2.5). The solver module consists of functions used to build and solve the equations of motion for the specific simulation type. The solver and core modules access and update variables into `NAST_Simulation` C++ struct. This database struct is used (passing by reference) whenever possible to reduce unnecessary copying and repeated allocation of memory. The output module

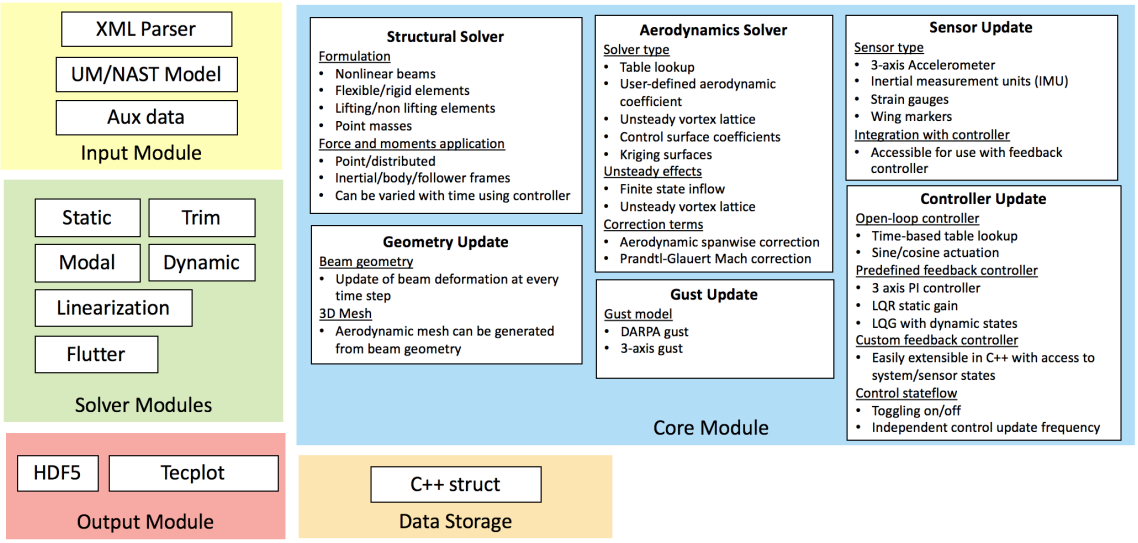

Figure 2.5: UM/NAST core module

then accesses and saves variables from `NAST_Simulation` to Hierarchal File Format 5 (HDF5) output file format.

2.3.1 Input

UM/NAST input is defined using an XML file. Usage of XML allows some run-time validation of input parameters. An added advantage is that the input file is easily readable (Fig. 2.6). The input file is divided into six main sections and described here:

`Aerodynamics` sets the aerodynamic options (e.g., compressibility corrections, and aerodynamic stall response). The number of inflow states per lifting element can be changed (default: 6) or it can disabled entirely.

`FlightConditions` sets the flight speed, altitude, and load factor.

`Structure` sets the key points containing the model geometry. Member connectivity information are defined by "radiating" outwards from the body center. Member structural properties (e.g., mass, inertia, stiffness, and non-structural masses) as well as cross-sectional properties (e.g., reference axis location, size, and inci-

dence angle) are defined. For lifting surfaces, the airfoil data input types (e.g., XFOIL lookup table, user defined coefficients, or kriging lookup) are specified.

`Loads` sets the manipulations of the control effectors (e.g., applied loads, control surfaces, and propellers). Open or closed-loop controls can be defined to control the magnitude of control effectors during simulation.

`Sensors` sets the virtual sensor measurement outputs, placement location, and "mounting" geometry.

`Simulation` sets the simulation type to be performed. Internal numerical tolerances and solver specific parameters are defined.

```
<!-- Distributed inertia values in CSYS w [kgm]-->
<inertia>
  <Iyy type="constant"> 1.0e-8 </Iyy>
  <Iyz type="constant"> 0 </Iyz>
  <Izz type="constant"> 1.0e-8 </Izz>
</inertia>
<!-- Stiffness property -->
<stiffness>
  <K11 type="constant"> 5.390E+07 </K11>
  <K12 type="constant"> 0 </K12>
  <K13 type="constant"> 0 </K13>
  <K14 type="constant"> 0 </K14>
  <K22 type="constant"> 5.390E+07 </K22>
  <K23 type="constant"> 0 </K23>
  <K24 type="constant"> 0 </K24>
  <K33 type="constant"> 5.390E+07 </K33>
  <K34 type="constant"> 0 </K34>
  <K44 type="constant"> 5.390E+07 </K44>
</stiffness>
```

Figure 2.6: Snippet of UM/NAST XML input file

2.3.2 Solvers

Six different solver types are implemented. Refs. [65, 83] present the detailed theoretical formulation of the solvers and they are summarized here:

`nast_staticsolver` computes the steady state structural deflection given a specified loading condition. It implements sub-iterations in pseudo time step to provide better numerical stability at large model deflection.

`nast_modalsolver` computes the modal shapes and frequencies either around the undeformed configuration, or after static simulation has been performed.

`nast_trimsolver` computes the control actuation (control surface deflections or applied forces) as well as the aircraft trim states (angle of attack and angle of side-slip) at the defined operating condition. Newton-Rhapson root finding algorithm calls upon `nast_staticsolver` to compute the deformed shape and overall residual forces at the vehicle center of gravity in free flight at each sub-iteration step. Within the sub-iteration, the full gradient is recomputed at each step using central differences. The cost of gradient computation high and grows linearly with the number of trim parameters. Broyden's method [93], which performs an approximate update of the gradient, is also implemented. The latter method is computationally cheaper, but this approximate gradient update can diverge from its true value, resulting in convergence problems.

`nast_dynamicsolver` performs time marching simulation. First, the initial shape is computed via `nast_staticsolver`. The equations of motion are propagated using user specified integration scheme. Trapezoidal integration is used for numerical simulation throughout this dissertation. Control surface deflections and applied loads can be prescribed by the user in the input file. Otherwise, their magnitudes can also be modified by a closed-loop feedback controller.

`nast_linsolver` computes the linearized continuous time state-space model (A, B). If virtual sensors are specified, the linearized output matrix C is also provided. Numerical linearization perturbs the states using `nast_dynamicsolver`.

`nast_fluttersolver` computes the flutter speed and mode. First, it linearizes the vehicle using `nast_linsolver` over a range of flight speed. Eigenvalue analysis is then carried out to determine the onset of instability.

2.3.3 Core

Sparse matrix manipulations are used when necessary. Careful design of the computation subroutines ensures that only relevant portions of the code are executed based on the physics of the elements. For example, for rigid elements, only rigid kinematic information are updated and portions of the code pertaining to flexible structural dynamics are not executed. For non-lifting surfaces, portions of the code pertaining to aerodynamic computation are not executed. A brief summary of the core functions are shown here in execution order:

`UpdateKinematics` first computes the nodal kinematic information (e.g., h vectors, and rotational matrices between different frames) using strain-displacement relationship.

`UpdateJacobian` computes the structural Jacobians from nodal information.

`UpdateSensors` builds sensor Jacobians from nodal Jacobians and computes the virtual sensor measurements at the current time step. Sensors are computed first to allow the measurements to be used in feedback control.

`UpdateController` computes the control action at the current time step. Open-loop control can be scheduled by simulation time using a lookup table, or it can be simply held constant at defined values. Closed-loop control actions are computed based on implemented controller type and gains.

`UpdateGust` computes the gust excitation at element nodes (if any) before the aerodynamics forces are calculated.

`UpdateLoads` computes the total applied loads to the structure, including gravity loads, user applied point/distributed loads, and aerodynamic loads.

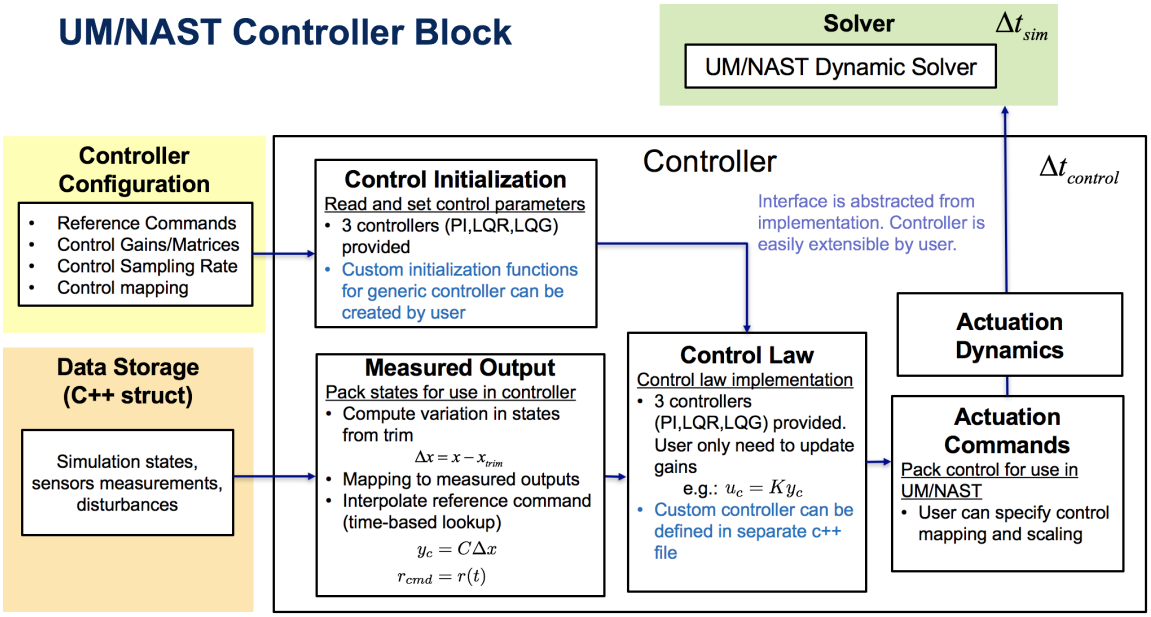

Figure 2.7: UM/NAST controller architecture

Among various functions, redesigned controller module allows more flexibility in control design and implementation. The control loop can run at a different rate compared to the main simulation loop. This new framework is shown in Fig. 2.7 and it consists of an initialization routine, input and output state mapping to UM/NAST simulation states, and control update routine. The input map allows the user to map individual or linear combinations of simulation states q to measured variables y_c used in the controller. The controller output can then be distributed to the control surfaces or applied loads within UM/NAST. Custom controller written in C++ can easily be integrated, as the interface to the UM/NAST is abstracted from the implementation of the control laws.

2.3.4 Executables

UM/NAST is compiled into two main executables (Fig. 2.8): 1) `nast` executes the simulation and saves it in an output HDF5 file, and 2) `nvis` loads the output HDF5 file and outputs a Tecplot® plotting data file for easy data visualization.

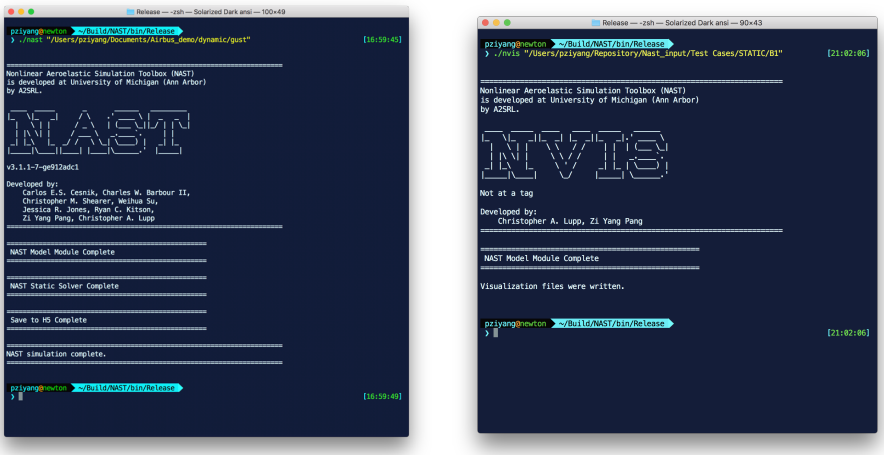

Figure 2.8: `nast` and `nvis` executables

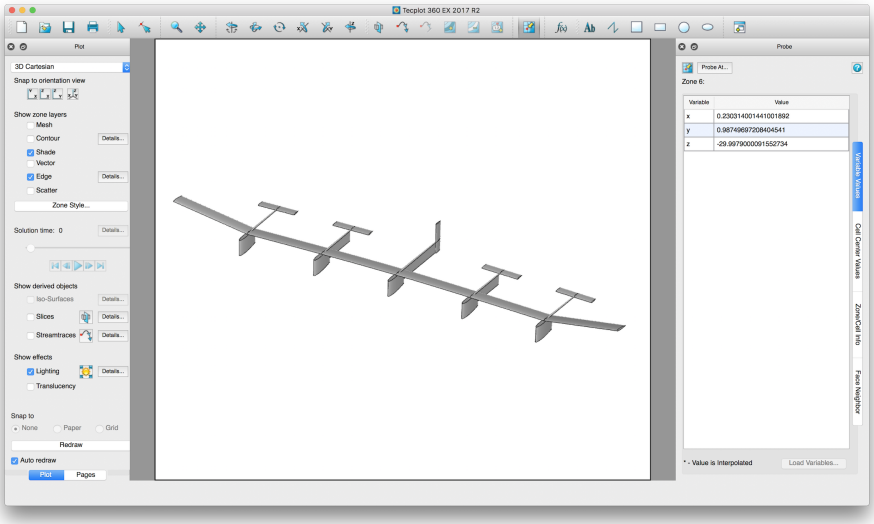

Figure 2.9: Sample results visualization in Tecplot®

CHAPTER 3

Stereo Vision Deformation Measurement System Development

This chapter presents the theoretical foundation and experimental validation of a stereo vision based deformation measurement system. Section 3.1 introduces the principles of stereo vision. The workflow includes camera calibration, stereo calibration, and stereo rectification. The above procedures allow one to triangulate a target from a stereo view of the scene. OpenCV computer vision library is used for all image processing tasks. Section 3.2 reports accuracy obtained from experimental results, employing chessboard targets and LED markers. Finally, Section 3.3 presents a full scale test of X-HALE using Vicon as the external reference benchmark. This is done to validate the overall accuracy of the proposed measurement system.

3.1 Principles of Stereo vision

As its name suggest, stereo vision compares a scene from two cameras placed at different vantage points and extracts three dimensional information from the two dimensional pictures. This dissertation focuses on left-right stereo vision arrangement (as opposed to up-down).

3.1.1 Pinhole Camera Model

The pinhole camera is an idealized model where a single ray of light from a scene is projected onto an imaging surface as shown in Fig. 3.1. By geometric arguments, the relationship between image coordinates and physical coordinates can be written as:

$$\begin{aligned} x &= f_x \left(\frac{X}{Z} + c_x \right) \\ y &= f_y \left(\frac{Y}{Z} + c_y \right) \end{aligned} \tag{3.1}$$

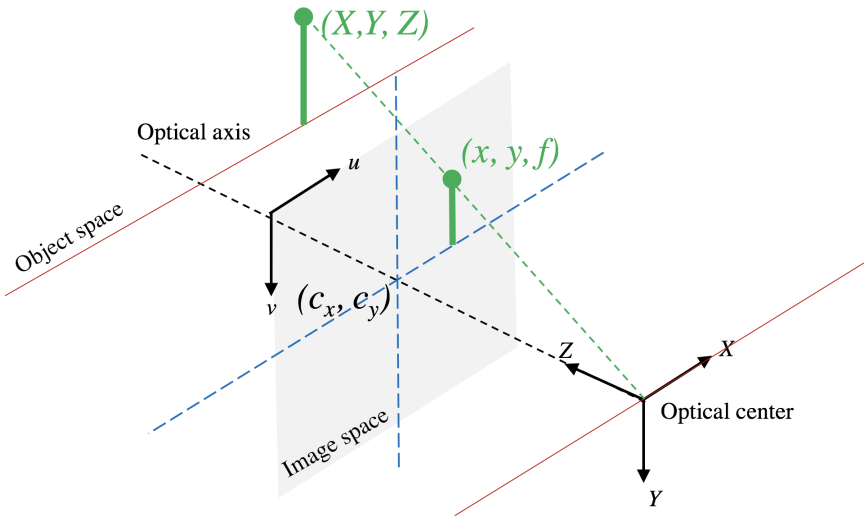

Figure 3.1: Pinhole camera model

Parameters c_x and c_y are offsets from the origin of image coordinates to the optical origin of the camera. The optical origin is located approximately at center of the sensor, with some variations due to manufacturing tolerance. Image origin is usually centered at top left corner in computer graphics. Parameters f_x and f_y are focal lengths in units of pixel. In computer vision, it is conventional to differentiate focal lengths as sensor elements can be rectangular, especially for low-cost CCD cameras. The skew parameter (non-rectangular sensor elements) is assumed to be zero. Let Q denote a point in the physical space with coordinates (X, Y, Z) while q denote a

corresponding projection in the image space with coordinates (x, y, f). The projective transform from Q to q can be expressed as:

$$\begin{Bmatrix} x \\ y \\ w \end{Bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} X/Z \\ Y/Z \\ 1 \end{Bmatrix} = M \begin{Bmatrix} X/Z \\ Y/Z \\ 1 \end{Bmatrix} \tag{3.2}$$

where f_x, f_y, c_x, and c_y are also known as camera intrinsic coefficients.

Camera lens can cause distortion to the perfect pin-hole model described above. There exist two forms of lens distortions: radial and tangential [94]. Radial distortion arises from the use of spherical lens instead more optically perfect parabolic lens. This manifest as a "fish-eye" effect due to light rays bending more further from the optical center. Tangential distortion occurs due to misalignment between the lens glass elements and the sensor plane. The points on the image plane as produced by a perfect pin hole camera (x, y) corrected from measurements from imperfect lens (x_d, y_d) can be expressed as:

$$\begin{aligned} x &= x_d \left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) + 2p_1 x_d y_d + p_2 \left(r^2 + 2x_d^2\right) \\ y &= y_d \left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) + p_1 \left(r^2 + 2y_d^2\right) + 2p_2 x_d y_d \\ r^2 &= x_d^2 + y_d^2 \end{aligned} \tag{3.3}$$

Therefore, there are five distortion coefficients k_1, k_2, k_3, p_1, and p_2 and four intrinsic camera parameters f_x, f_y, c_x, and c_y that must be obtained in order to transform between q and Q.

An object coordinate system (Fig. 3.2) is defined for subsequent derivation. The object coordinates can be viewed as a rotation $R \equiv [r_1, r_2, r_3]^T$ and translation t_v from the camera origin. Let $\tilde{Q} = (\tilde{X}, \tilde{Y}, \tilde{Z})$ denotes a point in the object space. In

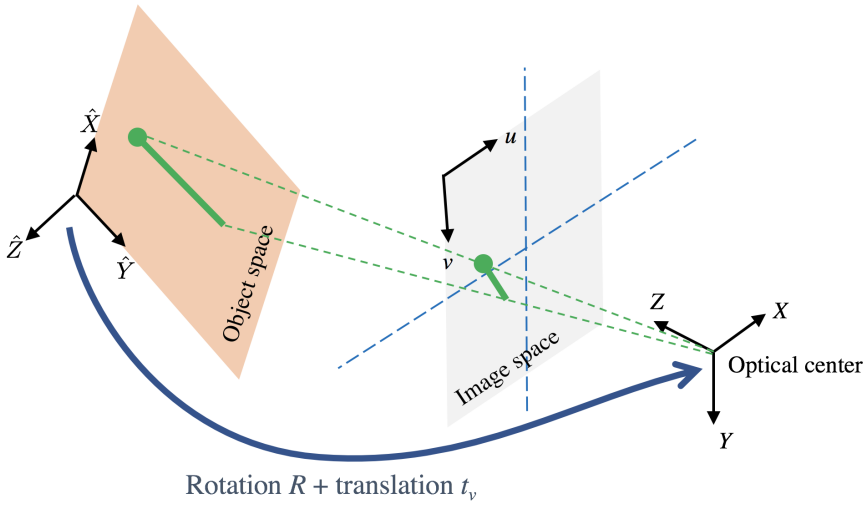

Figure 3.2: Homography between object space and image space

homogenous coordinates, this can be expressed as

$$\begin{Bmatrix} x \\ y \\ 1 \end{Bmatrix} = sM \begin{bmatrix} R & t_v \end{bmatrix} \begin{Bmatrix} \tilde{X} \\ \tilde{Y} \\ \tilde{Z} \\ 1 \end{Bmatrix} \tag{3.4}$$

where s is an arbitrary scale factor. Without loss in generality, the object space can be defined such that $\tilde{Z} = 0$. The homography between the image plane and object plane is therefore written as:

$$\begin{Bmatrix} x \\ y \\ 1 \end{Bmatrix} = sM \begin{bmatrix} r_1 & r_2 & t_v \end{bmatrix} \begin{Bmatrix} \tilde{X} \\ \tilde{Y} \\ 1 \end{Bmatrix} = s \begin{bmatrix} h_1 & h_2 & h_3 \end{bmatrix} \begin{Bmatrix} \tilde{X} \\ \tilde{Y} \\ 1 \end{Bmatrix} \tag{3.5}$$

This results in six additional unknowns (three rotation and three translation variables) and are usually referred to as camera extrinsic parameters [94].

3.1.2 Single Camera Calibration

Multiple views of a well-characterized object are used to compute the intrinsic and extrinsic parameters. This procedure is known as camera calibration. Chessboards (Fig. 3.3) are widely used as calibration objects (e.g., Refs. [95, 96]) since they are i) widely available and easily manufactured, and ii) possess high contrast corners amenable to sub-pixel detection, yielding better accuracy. OpenCV [97], an

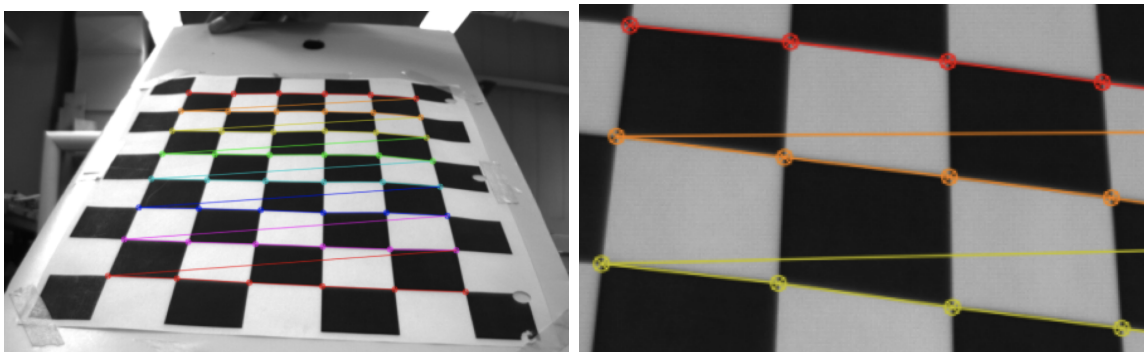

Figure 3.3: Chessboard used for camera calibration

open source computer vision library, is used for all numerical vision computations in this work. Bradiski and Kaehler [98] provide a detailed theory and implementation overview of OpenCV's algorithms. The following method used for single camera calibration implemented in `cv::calibrateCamera` is first proposed by Zhang [95], and shown in Appendix A.1. This step will produce camera intrinsic parameters $[f_x, f_y, c_x, c_y]$ and lens distortion coefficients $[k_1, k_2, k_3, p_1, p_2]$.

3.1.3 Stereo Calibration

Frontal parallel refers to the case where the left and right image plane are parallel (i.e., the epipoles are at infinty) and the image pixels are row aligned. In this case, the physical coordinates can be easily recovered by simple geometrical reasoning. However, in practice, a frontal parallel camera arrangement will never be achieved. Instead, it is achieved by mathematically transforming the physical arrange-

ment (Fig. 3.4) into frontal parallel. By taking joint views of chessboard from the perspectives of both cameras, `cv::stereoCalibrate` implements global Levenberg-Marquardt minimization to compute the rotation R and translation t_v relating the left and right cameras. Details are given in Appendix A.2.

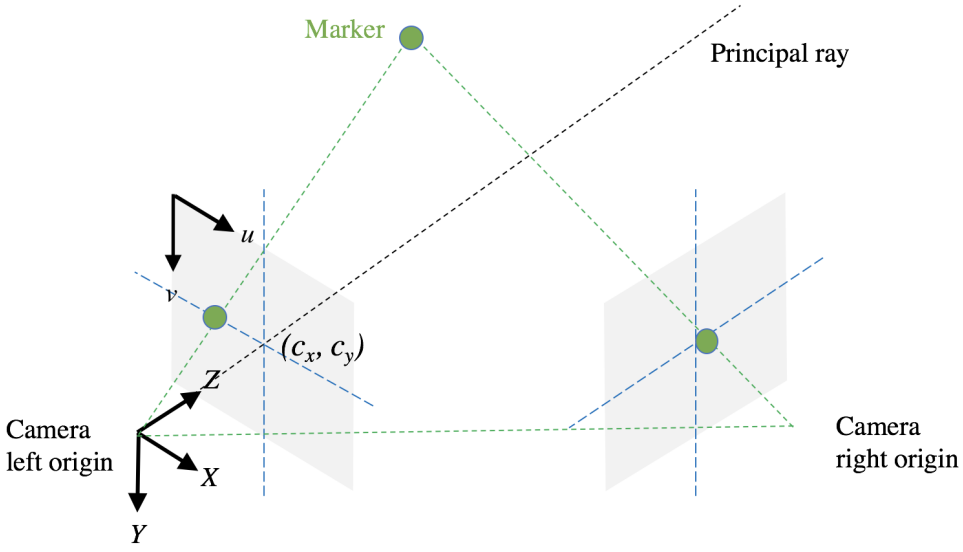

Figure 3.4: Physical camera arrangement

3.1.4 Stereo Rectification

From the stereo-calibration step, the rotation R and translation t_v between the cameras are known. To make them frontal parallel, `cv::stereoRectify` employs Bouguet's algorithm [99] where each camera is rotated by half of the computed rotation in opposite directions so that their principal rays are parallel. Then, the image planes are rotated in the z-direction to achieve row alignment. Details are given in Appendix A.3. The cameras are now frontal parallel. At this juncture, all the information required to solve a stereo vision triangulation problem are found.

3.1.5 Marker Detection

High-power LEDs can be used as markers for triangulation (Fig. 3.5). The camera aperture, gain, and shutter speed are controlled to obtain images of active markers as bright circular orbs on dark background. This is done to aid background rejection in image processing. Brightness thresholding is used to convert the greyscale image to a black and white image. Each filled contour represents a LED marker and the centroid can be easily found. Built-in OpenCV function `cv::SimpleBlobDetector` is used for this purpose. In this procedure, circularity C is used as a check to eliminate filled contours which do not correspond to a LED marker (e.g., surface reflection, or non-marker sources of light), i.e.,

$$C \equiv 4\pi \frac{\text{marker area}}{\text{marker perimeter}} \tag{3.6}$$

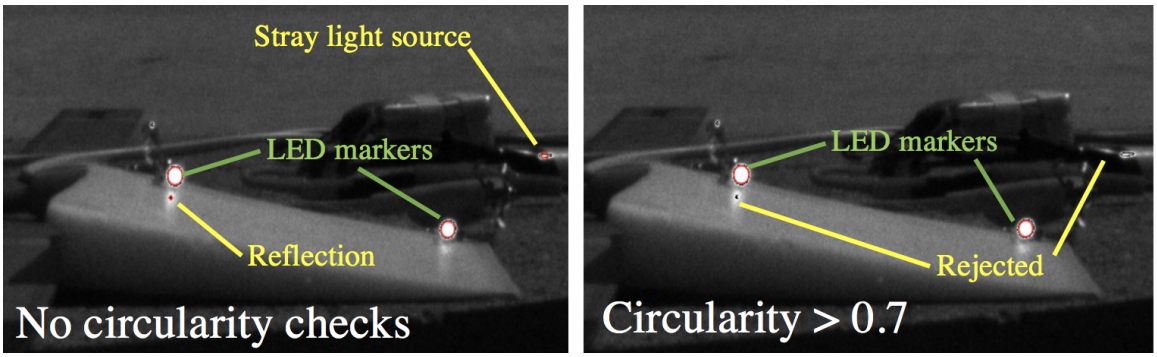

Figure 3.5: Thresholding and `SimpleBlob` detector centroid detection of LED markers circled in red

3.1.6 Triangulation

Image coordinates from both camera views must correspond to the same physical point. The projection of the physical point on to the image plane can be described by a projection matrix. Hartley and Zimmerman [94] developed a linear triangulation

algorithm which is implemented in `cv::TriangulatePoints`. Details are given in Appendix A.4. In summary, from a corresponding pair of image coordinates, using information about the stereo vision system, the physical point is reconstructed. The workflow is illustrated in Fig. 3.6.

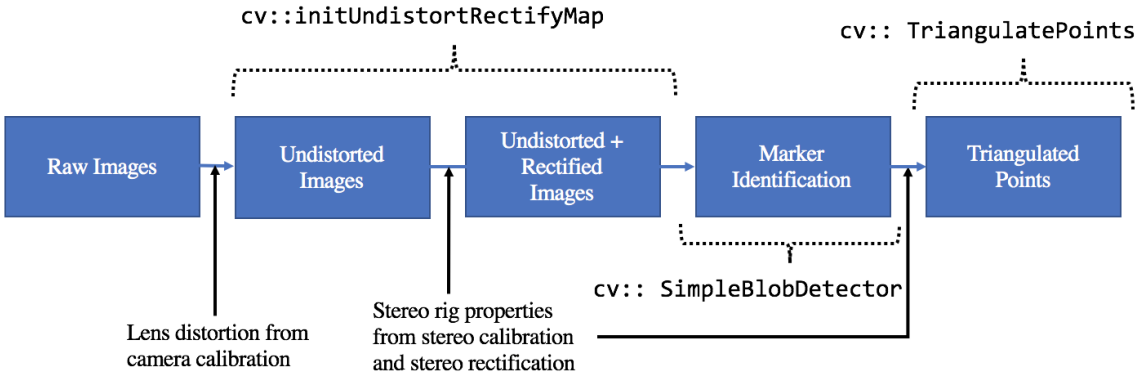

Figure 3.6: Workflow for stereo vision processing to obtain physical points

3.2 Preliminary Experimental Stereo Vision Characterization

PointGrey Flea3 USB3 cameras with Kowa lenses (described in Section 5.3.1.5) are used for experimental validation in this section. Images of a 10×8 chessboard are used to compute the intrinsic parameters of each camera independently. Then, joint views of the chessboard are used to perform stereo-calibration with the intrinsic parameters fixed. Three sets (of 20 views each) of images are used to experimentally characterize the parameters of the stereo vision setup. Table 3.1 reports the intrinsic and distortion parameters. The cameras are labelled as left and right looking, and numbering is from front to back.

The stereo relationship between the camera pair looking towards the right wing

Table 3.1: Camera intrinsic calibration values for $f = 3.5\,\text{mm}$ (units: pixel)

	Left 1	Left 2	Right 1	Right 2
f_x	1434.06	1428.58	1511.45	1427.25
f_y	1434.06	1428.58	1511.45	1427.25
c_x	719.163	747.283	764.296	771.687
c_y	1025.91	996.373	1039.36	1047.36
k_1	-0.3569	-0.3746	-0.383	-0.3856
k_2	0.1863	0.14078	0.32331	0.22591
p_1	0.00614	0.00396	0.00859	0.00191
p_2	-0.0016	0.00038	-0.0019	0.00032
k_3	-0.0705	0.11396	-0.2368	-0.0852

on ATV-6B (vehicle is described in Chapter 5) is identified as:

$$R = \begin{bmatrix} 0.992 & -0.009 & 0.125 \\ 0.011 & 0.998 & -0.013 \\ -0.125 & 0.014 & 0.992 \end{bmatrix} \quad t_v = \begin{Bmatrix} 259.901 \\ -14.593 \\ 11.171 \end{Bmatrix} \text{mm} \tag{3.7}$$

3.2.1 Chessboard Benchmark

The stereo vision system is mounted on an optical table for this benchmark. The mounting arrangement is identical to that on ATV-6B. A chessboard is placed perpendicular to the optical table, at different locations relative to the stereo vision rig (Fig. 3.7). Three different focal lengths are tested. They correspond to the minimum, middle, and maximum focal length of the Kowa lens.

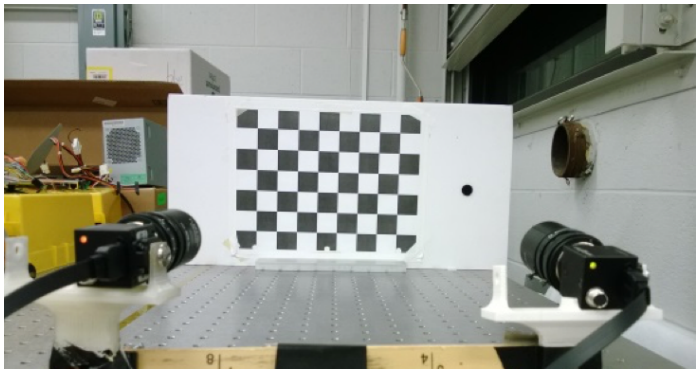

Figure 3.7: Chessboard placed at various distance from stereo vision rig

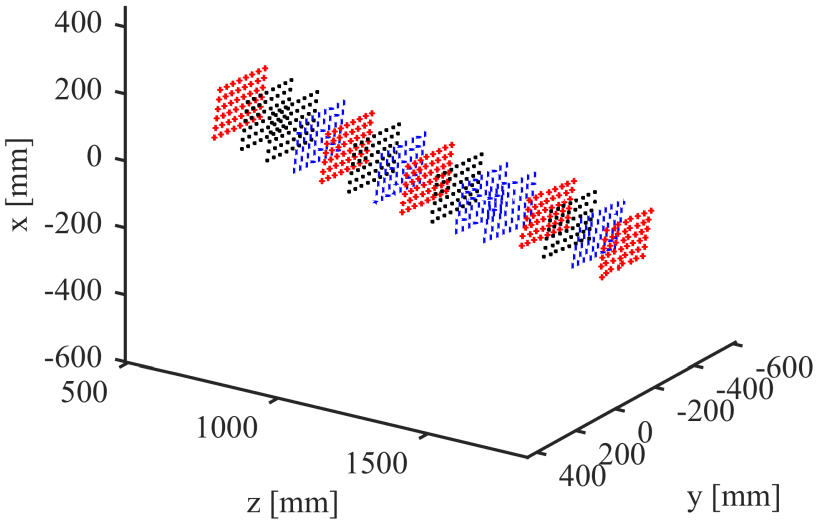

Figure 3.8: Re-constructed chessboard corners using stereo vision procedure

Physical coordinates of the chessboard corners are reconstructed using stereo vision from the images (Fig. 3.8). Since the chessboard dimensions are accurately known in the plane of the chessboard, the error metric e_{cb} is defined as the error between the known coordinates (x_{cb}, y_{cb}) and the recovered physical coordinates of the chessboard corners, re-projected into the chessboard frame (object space), denoted by (x_{rp}, y_{rp}). It can be written as:

$$e_{cb} = \frac{1}{N} \sum_{i=1}^{N} \sqrt{(x_{cb,i} - x_{rp,i})^2 + (y_{cb,i} - y_{rp,i})^2} \tag{3.8}$$

Table 3.2 shows the re-projection error increases as the chessboard is placed further away from the cameras. At the maximum tested distance (2.0 m), the re-projection error is 3.4 mm. Increasing the focal length should reduce the re-projection error at the expense of viewing area, as seen from $f = 3.5$ mm and $f = 10$ mm columns in Table 3.2 and Fig. 3.9. Using a focal length of 10 mm, the narrow field of view limits coverage to targets placed greater than 1.5 m away. Experimental results unexpectedly show that intermediate focal length ($f = 6.75$ mm) is actually worse than the shortest focal length.

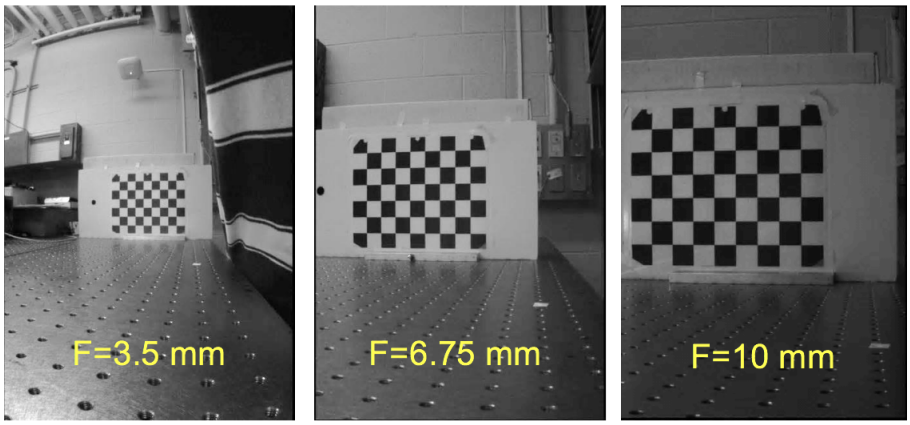

Figure 3.9: Field of view comparison for different focal lengths

Table 3.2: Re-projection error for chessboard target (units: mm)

Set	Z Distance	$f = 3.5\,\text{mm}$	$f = 6.75\,\text{mm}$	$f = 10\,\text{mm}$
1	500	0.22 ± 0.21	–	–
2	750	0.34 ± 0.25	–	–
3	1000	-0.99 ± 0.49	-0.4 ± 0.28	–
4	1250	1.55 ± 0.76	-1.4 ± 0.67	–
5	1500	-2.30 ± 1.12	-2.5 ± 1.15	-0.90 ± 0.48
6	1750	-2.88 ± 1.37	-3.5 ± 1.59	-1.98 ± 0.91
7	2000	-3.42 ± 1.68	-4.4 ± 2.04	-2.82 ± 1.30

– indicates chessboard not in field of view

It is well known that the accuracy of depth recovery decreases as the distance of the target increases. Figure 3.10 shows the experimental results for $f = 3.5\,\text{mm}$ when recovering depth information (displacement away from camera). The actual ground displacement z_{true} is measured, while the stereo vision processing computes recovered displacements z_{rp}. The error plot is normalized by the actual displacement using:

$$e_z = \frac{z_{rp} - z_{true}}{z_{true}} \tag{3.9}$$

At the furthest distance tested $(1.9\,\text{m})$[1], the error is 92 mm or 4.8%. Depth recovery is significantly worse than coordinate recovery parallel to the image plane.

[1]Data point at 2 m was not recorded due to oversight

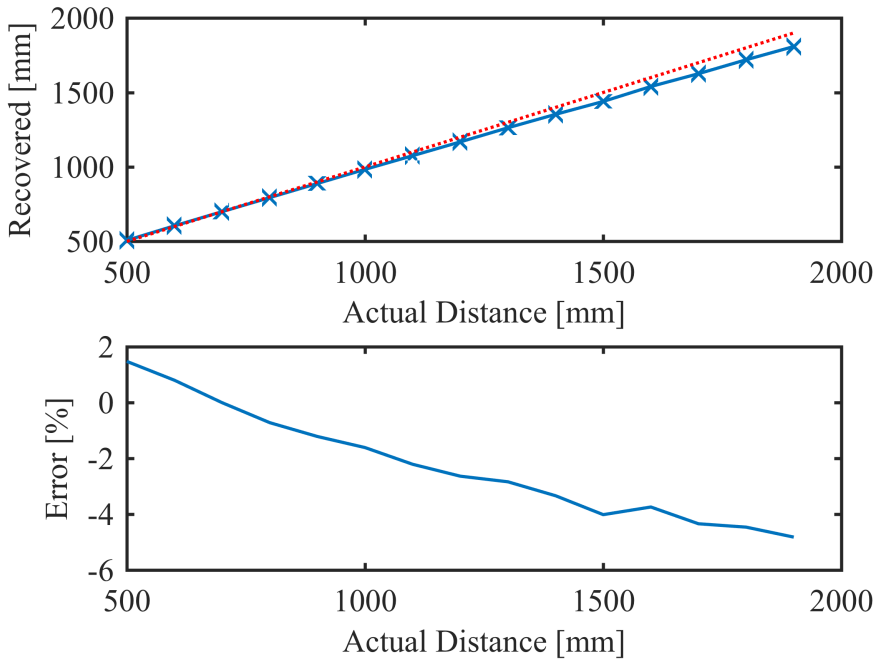

Figure 3.10: Depth recovery error between known and re-projected chessboard points reconstructed by stereo vision system for $f = 3.5\,\text{mm}$

3.2.2 LED Benchmark

The previous test is repeated, now replacing chessboard target with LED markers shown in Fig. 3.11. Two high-power LEDs identical to those used on ATV-6B (Chapter 5) are mounted 101.6 mm apart on a breadboard. Since the separation between the LEDs is accurately known, the error metric in this case is defined as the difference of reconstructed separation and the actual separation between the LEDs, i.e.,

$$e_{sep} = r_{rp} - 101.6 \tag{3.10}$$

Similar conclusions can be drawn from both chessboard and LED benchmark tests. Table 3.3 confirms that error metric increases as the LED pair is placed further away from the camera. Compared to the chessboard, the error is higher for the LED markers given the same the distance away from the camera. In addition, stereo vision recovery seems to consistently under-predict the physical coordinate of the LED mark-

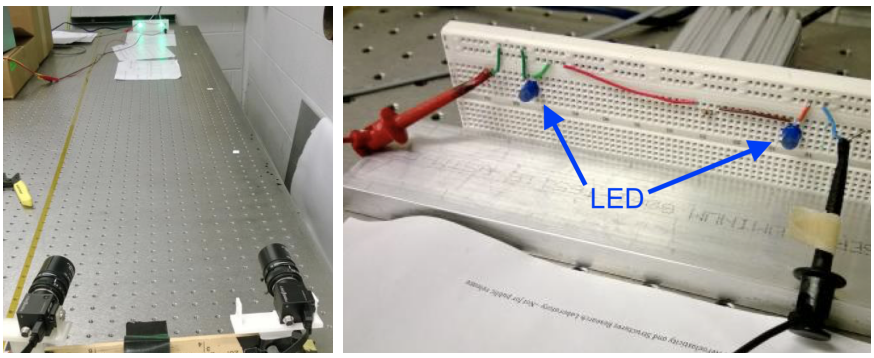

Figure 3.11: LED markers mounted on a breadboard are displaced perpendicular to lens assembly

ers. Figure 3.12 shows similar error magnitude in depth reconstruction. Although theoretically it is advantageous to use chessboard patterns for accuracy reasons, it is not feasible to use them on VFA wing surface as the view will be very shallow. This means that corner recognition will be difficult unlike the current benchmark test. In addition, a huge advantage in using LED markers is that the shutter speed remains consistently fast at 1 ms regardless of ambient lighting conditions. For chessboard and other fiducial patterns, even under bright sunlight, a shutter speed of 20 ms or longer is required. The former allows a small aperture (for large depth of view) and fast shutter speed (to minimize blurring due to target motion).

Table 3.3: Separation error for LED marker (units: mm)

Set	Z Distance	$f = 3.5\,\text{mm}$	$f = 6.75\,\text{mm}$	$f = 10\,\text{mm}$
1	500	−3.41	–	–
2	750	−3.86	–	–
3	1000	−4.29	−4.07	–
4	1250	−4.78	−4.83	–
5	1500	−5.19	−5.61	−4.02
6	1750	−5.61	−6.40	−4.76
7	2000	−6.27	−7.13	−5.52

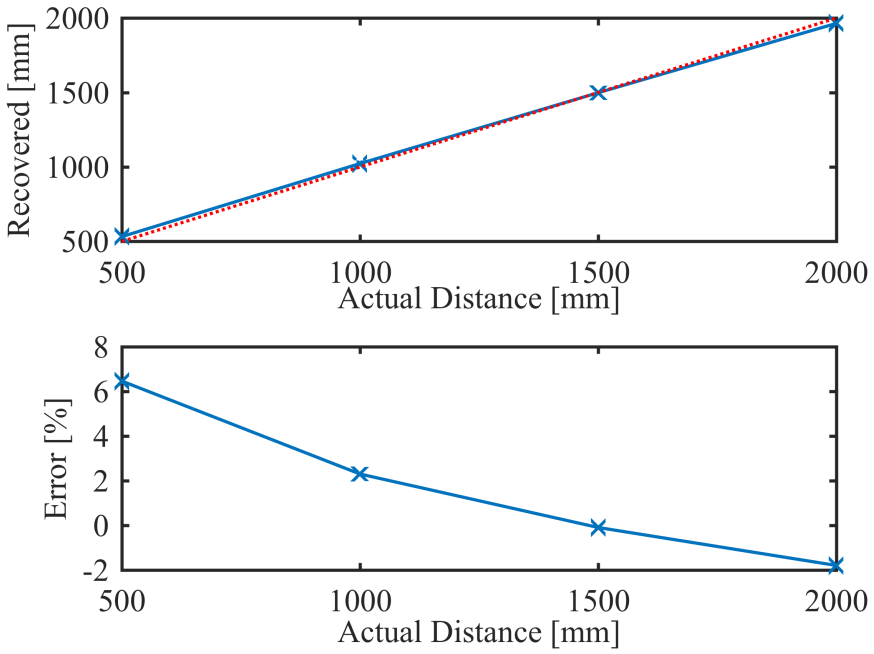

Figure 3.12: Depth recovery error between known and re-projected LEDs reconstructed by stereo vision system for $f = 3.5\,\text{mm}$

3.3 Vicon Full-Scale Benchmark

A very flexible wing test with external Vicon system is used to verify the accuracy of overall system. It is realized with the full scale ATV-6B aircraft described in Chapter 5. Vicon Infra-red (IR) markers are placed at known locations on the top wing surface as close as possible to the LED markers. A set of 12 Vicon T-40[1] cameras are placed around the ATV-6B to provide good reconstruction of the IR markers at all times (Fig. 3.13). IR marker displacements are post-processed using Vicon Blade[2]. The wing is deformed by displacing the wing tip by hand (Fig. 3.14). Various excitation types (bending/twisting), amplitudes, and frequencies are used.

Figure 3.15 shows an instantaneous snapshot view as seen by the Flea3 stereo vision cameras mounted at the center looking out towards the left wing. The picture is artificially brightened to show the wing surface and background. Identified markers

[1] http://www.vicon.com/Software/TSeries
[2] http://www.vicon.com/Software/Blade

Figure 3.13: Vicon T-40 cameras mounted on tripods around the testing area

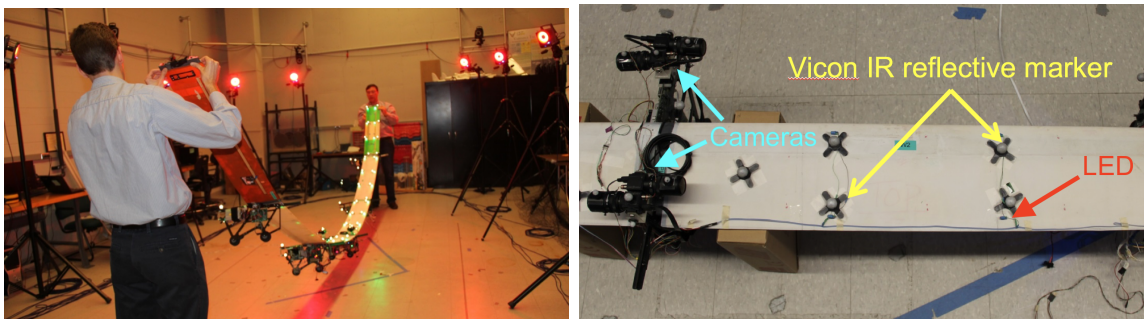

Figure 3.14: Deforming the wing by displacing the wing tip, with IR markers tracked by Vicon

are shown in red crosses. When correct exposure is used, only round orbs of LED markers are seen. The post-processed Vicon IR markers are also shown using Vicon's Blade software.

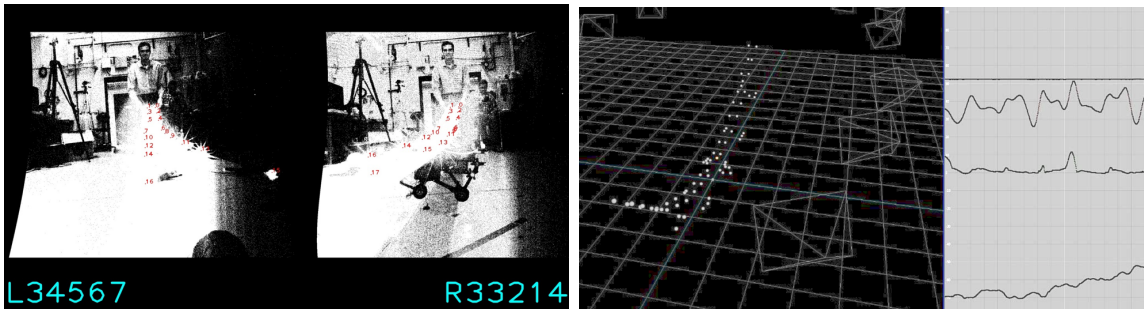

Figure 3.15: View from the camera, artificially brightened (left) and Vicon IR markers viewed in Blade software (right)

3.3.1 Initial Camera Mount Design

The displacement of Vicon IR markers are compared against stereo vision measurements for markers on the wing tip. Figure 3.16 shows the displacement comparison in dx (positive towards wing right), dy (positive towards wing leading edge) and dz (positive with right hand triad with dx and dy, approximately in the vertical direction). Out-of-plane bending generally compares well with Vicon measurements. Geometric beam shortening (stereo vision depth recovery) is captured but is visibly more noisy. There are some discrepancies in the wing in-plane bending direction with maximum error of about 20 mm or 7.6% of maximum out-of-plane displacement. This level of error is not seen in the benchmarks, which suggests that there is rotation of the camera views relative to each other. Upon further inspection of data where large bending motions (both frequency and amplitude) are present, stereo vision measurements show significant under-prediction of the vertical displacement (dz) of about 25.7 mm or about 12.9% of maximum out-of-plane displacement (Fig. 3.17). This is attributed to vibration of the entire camera assembly. The 3D printed plastic mount that supports the cameras has insufficient stiffness to keep the camera assembly (which weighs approximately 0.5 kg) steady. When examining video playback, during large upward deformations, the background scene is also shifted downwards, in phase with the wing displacements, indicating a upward tilt at the camera assembly. This is consistent with the reduced stereo vision measurements compared to Vicon reference measurements. Therefore, the camera mount had to be redesigned and further verification tests had to be conducted.

3.3.2 Reinforced Camera Mount Design

The new mount is re-designed to have a tight "collar" around the main wing in addition to the pod attachment to decrease any possible movement of the camera assembly relative to the body center (Fig. 3.18). In addition, the support structure is

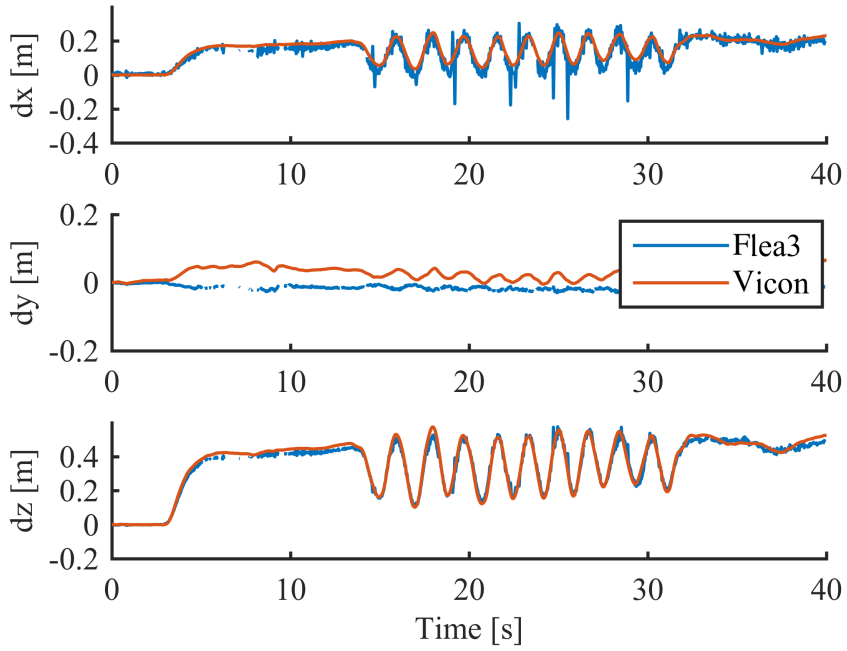

Figure 3.16: Comparison of displacement for a marker at the wing tip

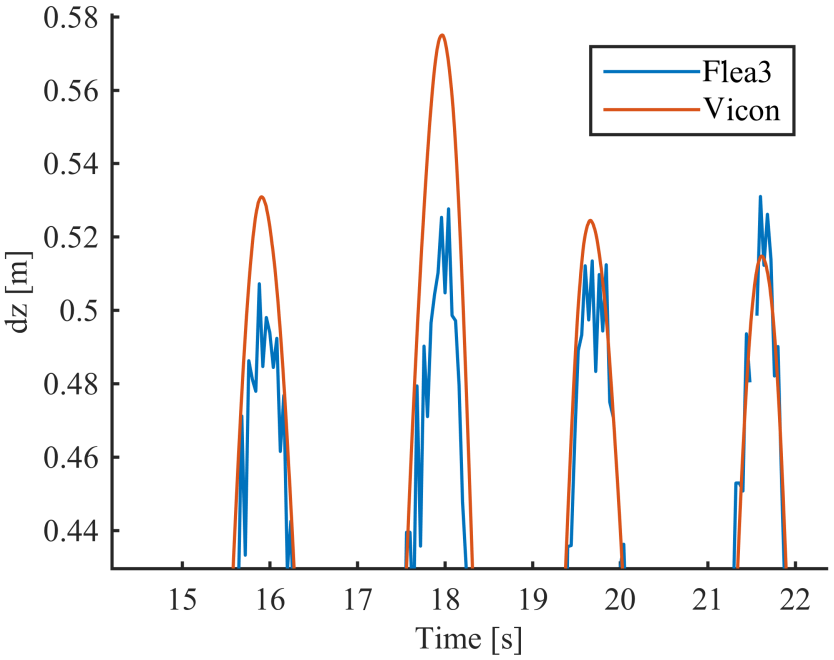

Figure 3.17: Zoomed in view shows under-prediction of LED markers due to support vibrations

thickened and printed with more infill to increase overall stiffness. Figure 3.19 shows the stereo vision measurements taken with the reinforced mount. The maximum in-plane bending measurement error decreased to 16.3 mm or 4.08% of maximum displacement. The maximum out-of-plane measurement error decreased to 2.2 mm or 0.55% of maximum displacement. Figure 3.20 shows that even at large frequency excitation, the camera assembly did not vibrate relative to body center. No appreciable increase of error in both out-of-plane and in-plane displacement is observed.

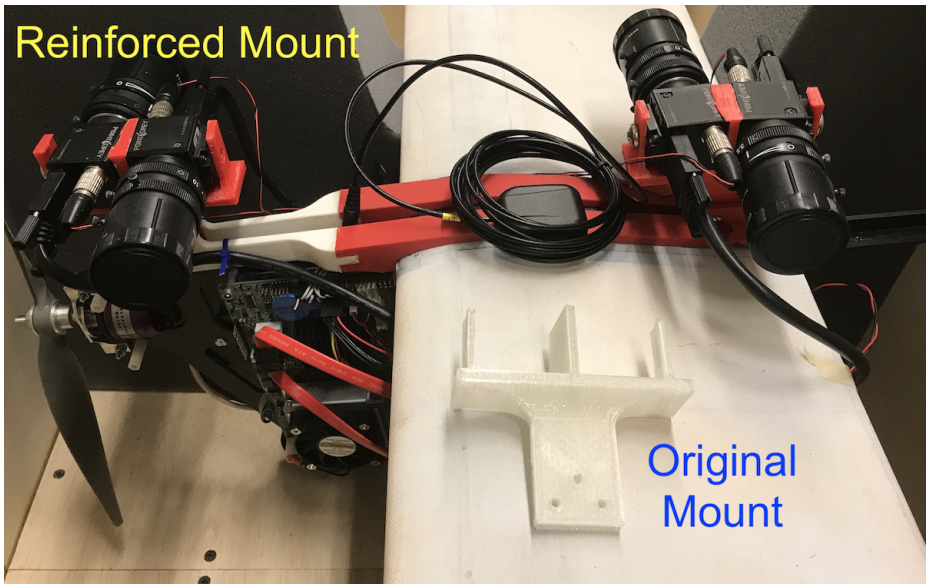

Figure 3.18: Initial and reinforced camera mounting designs

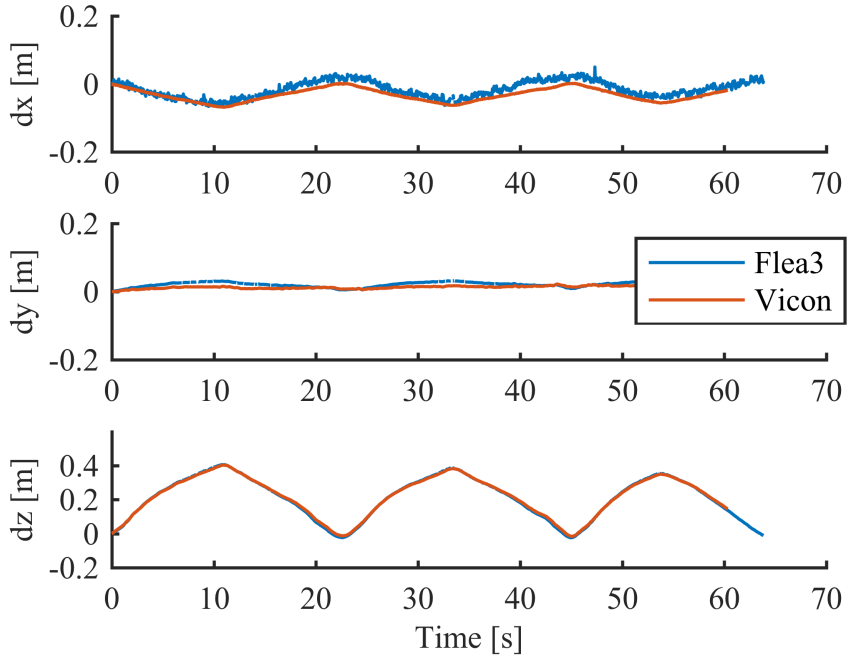

Figure 3.19: Comparison of displacement for a marker at the wing tip with reinforced mount

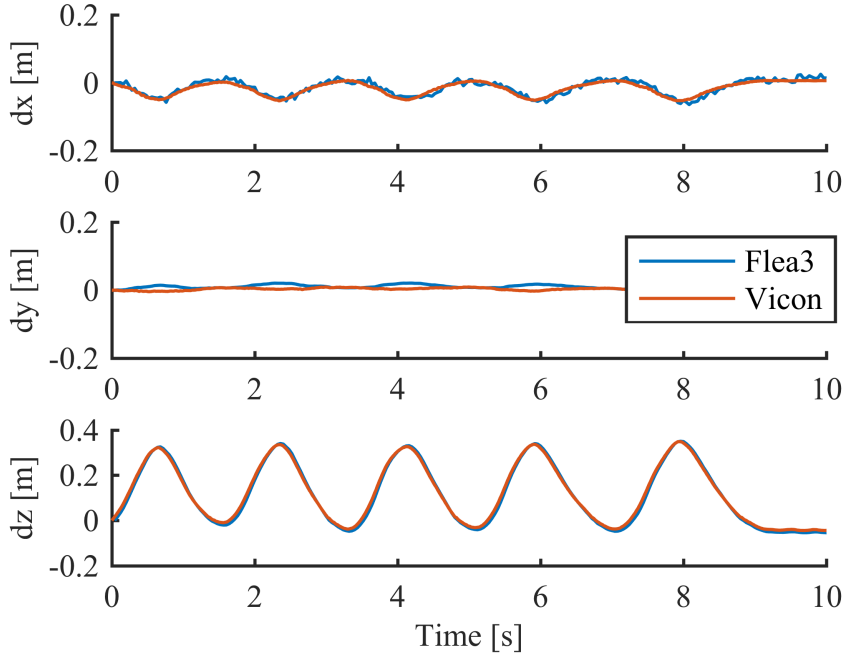

Figure 3.20: Comparison of displacement for a marker at the wing tip with reinforced mount under large frequency excitations

CHAPTER 4

Framework for VFA Maneuver Load Control

This chapter presents required components for designing a control law for VFA: a system model, a control law, and an observer. Section 4.1 tackles VFA control problem using using MPC, incorporating both control and state constraints in the formulation. Section 4.2 tackles an estimation problem so that the developed control law can be implemented. Two methods are presented: i) static shape reconstruction by using a nonlinear least square fitting based on beam kinematics, and ii) full state estimation using Kalman filter. A multi-rate Kalman filter formulation is also presented. An accurate model of the vehicle needed for control design. Therefore, Section 4.3 presents the system identification theory to identify a parametric state-space model and a non-parametric frequency description of the system using input and output data. Finally, experimental system identification results and state reconstruction results (after post-processing sensor data) allow the accuracy of UM/NAST to be evaluated.

4.1 Model Predictive Control of VFA

From the literature survey, the ability to apply control constraints (due to the low control authority of VFA) and apply state constraints (to prevent over-stress of

the lightweight wing structure) are important for successful control of VFA. Model predictive control is an ideal candidate as it is a modern optimal control technique capable of handling state and control constraints. It exploits knowledge of a system model to predict its future response and selects the optimal control trajectory. It is formulated as a constrained optimization problem, minimizing a user-defined objective cost function over a finite horizon. Again, consider a linear time-invariant (LTI) discrete time plant:

$$x_{k+1} = A_d x_k + B_d u_k \tag{4.1}$$

$$y_k = C_d x_k + D_d u_k \tag{4.2}$$

where state $x_k \in R^n$, input $u_k \in R^p$, and the LTI system matrices $A_d \in R^{n\times n}, B_d \in R^{n\times p}$. (A_d, B_d) has to be stabilizable. Moreover, r control and s state constraints at each time step are modeled as affine inequalities, forming a closed, convex set containing the origin, i.e.,

$$\mathbb{U} = \{u_{min} \leq S_u u_k \leq u_{max}\} \tag{4.3}$$

$$\mathbb{X} = \{x_{min} \leq S_x x_k \leq x_{max}\} \tag{4.4}$$

where $S_u \in R^{r\times p}$ and $S_x \in R^{s\times n}$. The objective cost in quadratic form can be written as:

$$J\left(x, u, N_p\right) = \sum_{i=0}^{N_p-1} \left(\frac{1}{2} x_{k+i}^T Q x_{k+i} + \frac{1}{2} u_{k+i}^T R u_{k+i}\right) + \frac{1}{2} x_{k+N_p}^T P x_{k+N_p} \tag{4.5}$$

where $N_p \geq 1$ is the prediction horizon, Q, R are weights on the states and controls, respectively, and P is the terminal cost. The weights are chosen such that the

following properties are satisfied:

$$\begin{aligned} Q &= Q^T \geq 0 \\ R &= R^T > 0 \\ P &= P^T > 0 \\ &(A, Q^{\frac{1}{2}}) \text{ detectable} \end{aligned} \tag{4.6}$$

The standard MPC problem can be written as:

$$u^* = \arg\min_{u,x} J(x, u, N_p) \tag{4.7}$$

subject to:

$$\begin{aligned} x_{k+1} &= A_d x_k + B_d u_k \\ x_k &\in \mathbb{X} \\ u_k &\in \mathbb{U} \end{aligned} \tag{4.8}$$

The terminal cost P can be chosen to be the cost of the unconstrained, infinite horizon optimal control problem. The solution to this standard linear quadratic problem is given by the algebraic Riccati equation:

$$A_d^T P + P A_d - (P B_d) R^{-1} (B_d^T P) + Q = 0 \tag{4.9}$$

The advantage of choosing P in this way ensures exponential stability for the closed-loop system if constraints are not active [100]. This constrained optimization problem can be solved using standard QP solvers. For MPC with control horizon of one step, the optimal control at current time step u_k^* is extracted from the control trajectory u^* obtained from Eq. (4.7).

4.1.1 Condensed MPC

From Eqs. (4.7) – (4.8), there are $N_p(n+p)$ optimization variables with $N_p n$ equality constraints from the LTI system state propagation. The equality constraints can be eliminated by rewriting the problem in condensed form. In this case, the number of optimization variables is reduced to $N_p p$. First, define the following vectors:

$$X \equiv \begin{bmatrix} x_k^T & x_{k+1}^T & x_{k+2}^T & \cdots & x_{k+N_p}^T \end{bmatrix}^T \tag{4.10}$$

$$U \equiv \begin{bmatrix} u_k^T & u_{k+1}^T & \cdots & u_{k+N_p-1}^T \end{bmatrix}^T \tag{4.11}$$

The predicted state trajectory $X \in R^{(N_p+1)n}$ can be expressed using the control trajectory $U \in R^{N_p p}$ and current initial state x_k, assuming the control horizon N_u is identical to the prediction horizon N_p as:

$$X = HU + Gx_k \tag{4.12}$$

where $H \in R^{(N_p+1)n \times N_p p}$ lower triangular matrix can be viewed as the forced response and $G \in R^{(Np+1)n \times n}$ as the free response. They are given by:

$$H = \begin{bmatrix} 0_{n \times N_p p} \\ H_{N_p} \end{bmatrix} = \begin{bmatrix} 0 & & & \\ B & & & \\ AB & B & & \\ \vdots & & \ddots & \\ A^{N_p-1}B & A^{N_p-2}B & \cdots & B \end{bmatrix} \tag{4.13}$$

$$G = \begin{bmatrix} I_{n\times n} \\ G_{N_p} \end{bmatrix} = \begin{bmatrix} I \\ A \\ A^2 \\ \vdots \\ A^{N_p} \end{bmatrix} \tag{4.14}$$

The quadratic cost can be written solely in terms of the control trajectory U as:

$$J\left(U, x_k\right) = \frac{1}{2} U^T \left(H^T \hat{Q} H + \hat{R}\right) U + U^T H^T \hat{Q} G x_k \tag{4.15}$$

$$\hat{Q} = \begin{bmatrix} I_{N_p} \otimes Q & 0 \\ 0 & P \end{bmatrix} \tag{4.16}$$

$$\hat{R} = \left[I_{N_p} \otimes R\right] \tag{4.17}$$

where $\otimes$ denotes the Kronecker product, and $\hat{Q} \in R^{(N_p+1)n\times(N_p+1)n}, \hat{R} \in R^{N_p p \times N_p p}$. To complete the formulation, the state x and control u constraints in Eq. (4.3) and Eq. (4.4) can be written as:

$$\begin{Bmatrix} U_{min} \\ X_{min} - S_{x_k} x_k \end{Bmatrix} \leq \begin{bmatrix} S_U \\ S_X \end{bmatrix} U \leq \begin{Bmatrix} U_{max} \\ X_{max} - S_{x_k} x_k \end{Bmatrix} \tag{4.18}$$

where

$$S_U = \left[I_{N_p} \otimes S_u\right] \tag{4.19}$$

$$S_X = \left[\left(I_{N_p} \otimes S_x\right) H_{N_p}\right] \tag{4.20}$$

$$S_{x_k} = \left[\left(I_{N_p} \otimes S_x\right) G_{N_p}\right] \tag{4.21}$$

$$X_{max} = 1_{N_p} \otimes x_{max} \tag{4.22}$$

$$U_{max} = 1_{N_p} \otimes u_{max} \tag{4.23}$$

$$X_{min} = 1_{N_p} \otimes x_{min} \tag{4.24}$$

$$U_{min} = 1_{N_p} \otimes u_{min} \tag{4.25}$$

Finally, using Eqs. (4.10) – (4.25), the condensed MPC problem becomes:

$$U^* = \arg\min_U J(U, x_k) \tag{4.26}$$

subject to:

$$U_{min} \leq S_U U \leq U_{max} \tag{4.27}$$

$$X_{min} - S_{x_k} x_k \leq S_X U \leq X_{max} - S_{x_k} x_k \tag{4.28}$$

There are many numerical techniques which can solve this constrained optimization problem. Online Active Set Strategy [75] implemented in open-source C++ library qpOASES [101] is used in this work. Ferreau [76] demonstrated superior performance of this strategy on small to medium-scale convex test examples. Moreover, qpOASES is self-contained, which allows easy integration in UM/NAST and facilitates implementation in embedded hardware platforms. Online active set strategy is also amenable to early termination, giving sub-optimal results, but usually without adverse effects. In fact, the library allows a user-specified time limit.

4.1.2 Condensed MPC with Reference Tracking

To perform offset free tracking, error states are augmented into plant dynamics:

$$e_{k+1} = e_k + (r_k - y_k)\Delta t \tag{4.29}$$

where $y \in \mathbb{R}^{ne\times 1}$ are the outputs which had to track commands, $r \in \mathbb{R}^{ne\times 1}$ are the external reference commands, and $e \in \mathbb{R}^{ne\times 1}$ is the accumulated error. Therefore, the augmented state-space model can be written as:

$$\begin{Bmatrix} x_{k+1} \\ e_{k+1} \end{Bmatrix} = \begin{bmatrix} A_d & 0 \\ -C_d\Delta t & I_{ne\times ne} \end{bmatrix} \begin{Bmatrix} x_k \\ e_k \end{Bmatrix} + \begin{bmatrix} B_d \\ 0 \end{bmatrix} \begin{Bmatrix} u_k \end{Bmatrix} + \begin{bmatrix} 0 \\ I_{ne\times ne}\Delta t \end{bmatrix} \begin{Bmatrix} r_k \end{Bmatrix} \tag{4.30}$$

where $B_r \equiv [0, I\Delta t]^T$ is the new control input matrix relating to the external reference commands r_k. Since future knowledge of r_{k+m} is usually not available, r_{k+m} is assumed to be unchanging from current value r_k over the entire prediction horizon. The new cost function, using the augmented plant, can be written as:

$$J(U, x_k) = \frac{1}{2}U^T\left(H^T\hat{Q}H + \hat{R}\right)U + U^T H^T\hat{Q}Gx_k - U^T H^T\hat{Q}\tilde{H}R_{cmd} \tag{4.31}$$

with:

$$\tilde{H} = \begin{bmatrix} 0 & & & \\ B_r & & & \\ AB_r & B_r & & \\ \vdots & & \ddots & \\ A^{N_p-1}B_r & A^{N_p-2}B_r & \cdots & B_r \end{bmatrix} \tag{4.32}$$

$$R_{cmd} = I_{Np} \otimes r_k \tag{4.33}$$

The solution procedure is identical to a standard MPC problem. Alternative formulations for offset free tracking are shown below.

One can choose to penalize the control action deviation from the new required steady state control computed from the tracked command given by:

$$u_{ss} = C_d \left[(I - A_d)B_d\right]^{-1} y_{ss} \tag{4.34}$$

$$\Delta u_k = u_k - u_{ss} \tag{4.35}$$

This requires a computationally expensive matrix inversion to find the new steady state control action u_{ss}.

Rate-based MPC is formulated using change of state Δx_k and change of control Δu_k, and error in tracked variables e_k given by:

$$x_{k+1} = x_k + \Delta x_k \tag{4.36}$$

$$u_{k+1} = u_k + \Delta u_k \tag{4.37}$$

$$e_{k+1} = C\Delta x_k + e_k \tag{4.38}$$

The state-space model has to be augmented with additional states to recover the instantaneous states x_k from the rate-based optimization variables Δu, given by:

$$\begin{Bmatrix} \Delta x_{k+1} \\ e_{k+1} \\ x_{k+1} \\ u_{k+1} \end{Bmatrix} = \begin{bmatrix} A_d & 0 & 0 & 0 \\ C_d & I_{ne\times ne} & 0 & 0 \\ I_{s\times s} & 0 & I_{s\times s} & 0 \\ 0 & 0 & 0 & I_{r\times r} \end{bmatrix} \begin{Bmatrix} \Delta x_k \\ e_k \\ x_k \\ u_k \end{Bmatrix} + \begin{bmatrix} B_d \\ 0 \\ 0 \\ I_{r\times r} \end{bmatrix} \left\{ \Delta u_k \right\} \tag{4.39}$$

The computational cost of the latter two methods are higher (matrix inversion, and higher problem dimensions, respectively). Therefore, the first method is selected for implementation and further analysis.

4.2 Sensor Fusion and Observer Design

From the literature review, the control of VFA requires feedback of structural information. Therefore, state estimation using observers is usually required since sensors typically do not measure the actual state. Two different methods for estimating structural deformation based on sensor measurements at discrete points are now presented. The first method employs nonlinear least square fitting using static snapshots. It is formulated using geometric arguments from beam kinematics. However, computations involved are comparatively expensive due to the need to solve a nonlinear least square problem. The second method is the well-known Kalman filter. A multi-rate Kalman filter formulation is presented. The computational cost is relative small and widely used in estimation problems.

4.2.1 Nonlinear Least Square Fitting

In this formulation, strain states ε and quaternions ζ are estimated based on instantaneous "static" snapshots of sensor measurements. Dynamic states will not be estimated. This is done entirely using kinematic strain-displacement relationship without knowledge of externally applied loads (e.g., aerodynamic, and gravity loads). Only geometric properties (e.g., location and offset from the beam reference axis) are required. Mass and stiffness properties of the structure need not be known.

First, the observed sensor measurements (from center mounted Inertial Navigation System (INS), M outboard IMU, and N wing markers) are stacked into a column vector, that is:

$$y_{meas} = \begin{bmatrix} \phi_B & \theta_B & \psi_B & \phi_1 & \theta_1 & \cdots & \phi_M & \theta_M & p_{1,r} & \cdots & p_{N,r} \end{bmatrix}^T \tag{4.40}$$

From the kinematic equations developed in Section 2.2, the predicted measurements are functions of strains ε and quaternions ζ only. Table 4.1 summarizes the

sensor measurements, kinematic equations and dependent variables.

Table 4.1: Sensor kinematic relationship for least square problem

Sensor	Measurement	Kinematics	Dependence
Center INS	θ_B, ϕ_B, ψ_B	Eq. (2.46)	ζ
i^{th} Outboard IMU	θ_i, ϕ_i, ψ_i	Eq. (2.48)	ε, ζ
j^{th} Marker rel. position	$p_{j,r}$	Eq. (2.45)	ε

For a center mounted INS, there will not be any dependence on the strain states since it is by definition fixed to the body frame. For the wing markers, relative position expressed in the body frame (relative to body origin) is dependent on deformation of the structure but not the rigid body rotation of the body frame. For the multiple IMUs, the local attitude measurements comprise both structural deformation and rigid body rotations. Note that for strain dependent measurements, they are only affected by strain states of the flexible elements leading to the point of measurement from the origin of the body B frame (in a root to tip sense). For example, intuitively, the motion of flexible members on the right wing relative to the origin of the body B frame will not affect an IMU mounted on the left wing. The predicted measurement values given dependent states ε and ζ can be written as a nonlinear function F:

$$y_{pred} = F(\varepsilon, \zeta) \tag{4.41}$$

This state estimation is formulated as a nonlinear least square problem where the 2-norm error between predicted measurements y_{pred} and observed measurements y_{meas} is minimized:

$$\min_{\varepsilon,\zeta} \| W (y_{meas} - F(\varepsilon, \zeta)) \|_2 \tag{4.42}$$

where W is a user-specified weighting matrix. This matrix can be used to tune the sensitivity of the least square solution to each sensor type. For example, one may wish to place less emphasis on a noisy low-grade sensor compared to a less noisy one from

a high-grade sensor. From past experiences on UAS, the drift variation of IMU yaw angle measurement is much higher than pitch and roll angle measurements as yaw is corrected only via magnetometer. Due to the use of brushless DC electric motor and close proximity of other onboard electronics, the magnetic interference will be high. Hence, one can either discard the yaw measurement from the IMU or assign a very large uncertainty to it. To avoid solving a constrained optimization problem to enforce $\|\zeta\|_2 = 1$ for the quaternion estimation, Eq. (4.42) can be converted to Euler angle representation and solved for three rigid body Euler angles $(\phi_B, \theta_B, \psi_B)$ directly.

When the dimension of the measurement equations is larger than the dimension of states to be estimated, this system is overdetermined. Such a system will be better at noise rejection. However, one must also take note of the distribution of the sensors to ensure redundancy in measurements. As explained, each sensor is only affected by preceding flexible elements and not by subsequent flexible elements. For example, putting many sensors at the wing root will not improve the estimation of strain near the wing tips.

For gradient based optimizers, the gradient to Eq. (4.42) can be supplied using linearized equations obtained either numerically or analytically. They are similar in form to the ones in Section 2.2.3.

4.2.2 Kalman Filter

Kalman filter [102] is an algorithm which combines noisy measurements to give an optimal estimate of the states of a dynamical system. Consider the following discrete time linear model with process and measurement noise:

$$\begin{aligned} x_{k+1} &= A_d x_k + B_d u_k + w_k \\ y_k &= C_d x_k + v_k \end{aligned} \tag{4.43}$$

where $w_k \sim \mathcal{N}(0, Q_{cov})$ is the process noise, and $v_k \sim \mathcal{N}(0, R_{cov})$ is the measurement noise. $\mathcal{N}(0, \sigma^2)$ denotes a Gaussian distribution with mean zero and variance σ^2. The optimal estimate of the system states is given by the prediction and update steps. In the prediction step, the *a priori* state estimate $\hat{x}_{k+1|k}$ is propagated using linear system dynamics:

$$\hat{x}_{k+1|k} = A_d \hat{x}_k + B_d u_k \tag{4.44}$$

$$P_{k+1|k} = A_d P_{k|k} {A_d}^T + Q_{cov} \tag{4.45}$$

In the update step, the *a posteriori* estimate of state $\hat{x}_{k+1|k+1}$, corrected by the measurements, is given by:

$$v_{k+1} = y_{k+1} - H_{k+1} x_{k+1|k} \tag{4.46}$$

$$K_{k+1} = P_{k+1|k} H_{k+1}^T \left[H_{k+1} P_{k+1|k} {H_{k+1}}^T + G \right]^{-1} \tag{4.47}$$

$$\hat{x}_{k+1|k+1} = \hat{x}_{k+1|k} + K_{k+1} v_{k+1} \tag{4.48}$$

$$P_{k+1|k+1} = \left[I - K_{k+1} H_{k+1} \right] P_{k+1|k} \tag{4.49}$$

where

$$\begin{aligned} H_{k+1} &= C_d \\ G &= R_{cov} \end{aligned} \tag{4.50}$$

For the traditional Kalman filter, all measurements are assumed to be available at the same rate as the filter update rate. However, update rates typically differ for different types of sensor. For example, update rate for IMUs and accelerometers are typically much higher than vision-based sources. Armesto et al. [53] modified Eq. (4.50) to allow multi-rate update while retaining the usual filter equations. This

is done by dynamically changing the size of the filter matrix:

$$\begin{aligned} H_{k+1} &= C_{d\Delta} \\ G &= R_{cov,\Delta} \end{aligned} \tag{4.51}$$

In this new formulation, $C_{d\Delta}$ only contains the rows of the full output matrix C_d corresponding to available measurements. Similarly, $R_{cov,\Delta}$ contains only the sensor noise covariance matrix of available measurements. Therefore, the update of estimated states can be done with any subset of measurement signals. To obtain the output matrix C_d, one has to linearize the output measurement equation. Table 4.2 summarizes the kinematic equations to be linearized.

Table 4.2: Sensor kinematic relationship for Kalman filter

Sensor	Measurement	Kinematics	Dependence
Center INS attitude	θ_B, ϕ_B, ψ_B	Eq. (2.46)	ζ
Center INS ang. rate	ω_B	Eq. (2.58)	β
Center INS velocity	v_B	Eq. (2.59)	β
i^{th} Outboard IMU attitude	$(\Theta_{sensor})_i$	Eq. (2.57)	ε, ζ
i^{th} Outboard IMU ang. rate	$(\omega_{sensor})_i$	Eq. (2.58)	$\dot{\varepsilon}, \dot{\beta}$
i^{th} Outboard IMU acceleration	$(a_{sensor})_i$	Eq. (2.60)	$\ddot{\varepsilon}, \ddot{\beta}, \dot{\varepsilon}, \dot{\beta}, \varepsilon$
j^{th} Wing marker rel. position	$p_{j,r}$	Eq. (2.45)	ε

4.3 System Identification Theory

System identification serves two purposes in this dissertation. To design control laws, the designer must first obtain an accurate model of aircraft dynamics. This can be achieved experimentally via system identification. Secondly, although UM/NAST aeroelastic models are created, they need to be corroborated with flight data. Comparing system identification data with UM/NAST predictions allows both validation of UM/NAST aeroelastic code, and fine-tuning of numerical model for control design.

In support of above objectives, this section presents the theoretical background for

system identification using input and output measurement data. Time and frequency domain methods are presented. Both identifications are based on distinct methods but offer complementary information.

4.3.1 Non-Parametric Frequency Domain Identification

A multiple-input-multiple-output (MIMO) system can be represented in the frequency domain in matrix form as:

$$\begin{aligned} Y(s) &= H(s)U(s) \\ H_{ij}(s) &= h(s), \quad i = 1, \ldots, n_o;\ j = 1, \ldots, n_i \end{aligned} \tag{4.52}$$

where each component entry is given by a single-input-single-output (SISO) transfer function $h_{ij}(s)$ for all n_i input and n_o output channels. The estimate (denoted by the hat symbol) of the MIMO transfer function $\hat{H}(s)$ can be obtained using:

$$\hat{H}(s) = \hat{G}_{uu}^{-1}(s)\,\hat{G}_{yu}(s) \tag{4.53}$$

where G_{uu} is a $n_i \times n_i$ auto/cross spectrum of the control inputs and G_{yu} is a $n_o \times n_i$ cross spectrum of the output and control inputs. If the off diagonal terms in the input spectrum are zero, the inverse is simplified to a reciprocal, reducing the problem to SIMO identification. This only occurs when all the controls are fully uncorrelated. For a discrete-time case, the spectra estimates can be expressed as:

$$\hat{H}(f_k) = \hat{H}(k\Delta f) = \hat{G}_{uu}^{-1}(f_k)\,\hat{G}_{yu}(f_k) \tag{4.54}$$

where f_k is a discrete frequency point.

To obtain a smooth spectral estimate, overlapped windowing techniques can be applied. In this dissertation, a Hanning window is used. For detailed treatment

on obtaining spectra estimate and windowing techniques, refer to standard signal processing or system identification text (Refs. [27, 28, 103]). Coherence function is a measure of the linear correlation between signals. It is defined as:

$$\hat{\gamma}^2_{yu}(f_k) \equiv \frac{\hat{G}^*_{yu}(f_k)\,\hat{G}^{-1}_{uu}(f_k)\,\hat{G}_{yu}(f_k)}{\hat{G}_{yy}(f_k)} \tag{4.55}$$

where superscript $*$ represents complex conjugate transpose. For SISO or uncorrelated MIMO cases, Eq. (4.55) reduces to:

$$\hat{\gamma}^2_{yu}(f_k) = \frac{|G_{yu}(f_k)|^2}{|G_{yy}(f_k)|\,|G_{uu}(f_k)|} \tag{4.56}$$

In practice, $\gamma > 0.6$ is desired for accurate frequency response [28]. In addition, the coherence level should not vary rapidly over a small frequency band. A low coherence value may be attributed to noise contamination in measured signals, nonlinearities in input-output relationship, and unknown disturbances. In this work, since only one control channel is altered at any time with the rest held constant, all identification can be done using SISO methodology. This frequency response computation is evaluated using the `spa` function in MATLAB System Identification Toolbox.

4.3.2 Parametric Time Domain Identification

Another relevant method is to identify a parametric model from the experimental data. This method is advantageous as it estimates a complete algebraic description of the dynamical system as opposed to discrete points on the frequency response obtained from Section 4.3.1. Given some model structure $\mathcal{M}$ parameterized by parameter vector θ, the prediction error e is the difference between measured output y_{meas} and predicted output y_{pred} at each time instant t_k and given by:

$$e(t_k, \theta, \mathcal{M}) = y_{meas}(t_k) - y_{pred}(t_k, \theta, \mathcal{M}) \tag{4.57}$$

The objective is to find a parameter vector θ^* given a fixed model structure $\mathcal{M}$ which minimizes the prediction error over the time history of the outputs and inputs $Z^N = [y_{meas}(t_1), u(t_1), \cdots, y_{meas}(t_N), (t_N)]^T$. Mathematically, it can be expressed as [27]:

$$\theta^* = \arg\min_{\theta} V\left(\theta, Z^N\right) \tag{4.58}$$

with

$$V\left(\theta, Z^N\right) = \frac{1}{N}\sum_{i=1}^{N} L\left(e\left(t_i, \theta\right)\right) \tag{4.59}$$

where $L(.)$ is a scalar-valued quadratic cost function. The predicted output y_{pred} can be computed based on the assumed model structure $\mathcal{M}$ (e.g., polynomial, state-space, or transfer function). In the following derivation, a state-space model structure is assumed. This class of identification method is known as subspace identification. The parameter vector in this case is the (A, B, C, D) state-space matrices. Writing the model in state-space form using a delay operator q, corrupted with sensor noise $v(t)$:

$$y_{pred}(t) = H(q, \theta)\, u(t) + H_0(q, \theta)\, v(t) \tag{4.60}$$

The prediction error e_F with pre-filtering $F(q)$ is:

$$e_F(t_k, \theta) = F(q)\, e(t_k, \theta) = \frac{1}{H_0(q, \theta)}\left[F(q)\, y_{meas}(t) - F(q)\, H(q, \theta)\, u(t)\right] \tag{4.61}$$

The optimization problem in Eq. (4.58) can be solved using e_F to affect the frequency range of the fit [27]. For instance, a bandpass filter can be implemented to focus the prediction fit over a certain frequency band. A high-pass filter might be implemented to remove any residual low-frequency disturbance outside the frequency of interest. Pre-filtering will not have an effect on the input-output relationship as well as parameter estimation for a linear system. In this dissertation, MATLAB System Identification Toolbox function `ssest` is used to solve this nonlinear optimization

problem.

4.3.3 System Identification Signal

One of the challenges in system identification is to design the input disturbance to excite the modes of interest. In turn, this requires some *a priori* knowledge of the system. Frequency sweep is a common excitation signal used for system identification. Frequency sweeps are sinusoids whose frequencies vary with time. In this dissertation, the frequency is varied logarithmically as illustrated in Fig. 4.1. This signal $\delta(t)$ is completely parameterized by the signal amplitude A_0, start frequency f_0, stop frequency f_1, signal duration t_1, and current time t:

$$\delta(t) = A_0 \sin\left(f_0 \left(\frac{f_1}{f_0}\right)^{\frac{t}{t_1}} t\right) \tag{4.62}$$

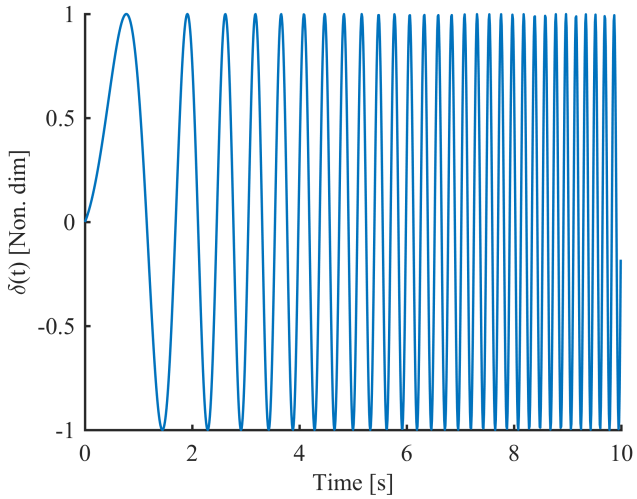

Figure 4.1: Example chirp signal from 0.1 Hz to 3 Hz for duration of 10 s

CHAPTER 5

X-HALE Aeroelastic Testbed Vehicle

In this chapter, the X-HALE aeroelastic testbed vehicle is introduced. Two configurations are presented: Aeroelastic Testbed Vehicle (ATV-6B) and Risk Reduction Vehicle (RRV-6B). A brief design history of the X-HALE is described in Section 5.1. Features common to both vehicles are reported in Section 5.2. Reported properties include structural, aerodynamic, servo, and propulsion characteristics. ATV-6B and RRV-6B specific hardware and software components are described in Section 5.3 and Section 5.4, respectively. Section 5.5 details the ground telemetry architecture. Finally, Section 5.6 describes the sensor selection process and the rationale for the final chosen configuration.

Figure 5.1: X-HALE aeroelastic testbed vehicle takeoff sequence [1]

5.1 X-HALE Design and History

X-HALE aeroelastic VFA was first designed by Cesnik and co-workers [104]. The primary objective for the development of this vehicle is to collect experimental aeroe-

lastic data in support of validation of coupled nonlinear aeroelastic-flight mechanics simulation tools, and to serve as a testbed to evaluate VFA control strategies. To demonstrate aeroelastic instability, the X-HALE was designed to exhibit an unstable but controllable aeroelastic coupled flight dynamic behavior excitable under finite disturbance. To capture nonlinear structural effects, X-HALE should exhibit large wing deformation of more than 25% semi-span, where linear assumptions usually start to break down. To enable validation of numerical simulations with experimental flight data, X-HALE needs to be instrumented with a comprehensive suite of sensors to measure various structural, aerodynamic, and flight dynamic information.

Initial design published by Cesnik and co-workers produced two configurations: one with the 6-meter span, and the other with a 8-meter span wing-boom-tail type aircraft. This initial design had a total weight of 11 kg to 12 kg with aspect ratio of 30 to 40, respectively, and flight speed ranging from $10\,\mathrm{m\,s^{-1}}$ to $19\,\mathrm{m\,s^{-1}}$. The primary sensor suite consisted of four Athena-II Data Acquisition Unit (DAQ) recording 128 strain gauges configured as balanced bridges embedded in the wings. A Risk Reduction Vehicle (RRV) was built, and it was structurally and aerodynamically identical to the published design, except that the sensor suite was replaced with ballast mass. This RRV was used for pilot familiarization and preliminary flight tests. The lower cost (absence of expensive computers and DAQ systems) and complexity (lower number of wirings and software requirements) allowed rapid prototyping and turn-around time. In parallel, the fully instrumented Aeroelastic Testbed Vehicle (ATV) was also manufactured. A 4-meter span design was initially used to evaluate the design and operations. The pilot reported good handling characteristics with limited aeroelastic effects. The 6-meter span design was chosen for the final vehicle once it confirmed significant elastic deformation in flight. This 6-meter span design exhibited required level of coupled aeroelastic responses with reasonable handling qualities, though it still proved very sensitive to pilot induced oscillations during landing. Moreover,

the additions of the aerodynamic fairings with air scoops reduced aerodynamic yaw damping, and produced an over-powering yawing moment. Differential thrust was unable to counteract this asymmetry. Hence, the fairings were kept out for the current design iteration. Finally, the original strain gauge-based sensor suite was deemed to be operationally infeasible. Every strain gauge bridge needed to be balanced prior to flight, and balancing all 128 proved to be an extremely time-consuming operation. Even after balancing, bias will still set in due to temperature differential between top and bottom wing surfaces when exposed to the sun in flight. Therefore, a radical re-design of the sensor suite was necessary.

In this dissertation, the new design based primarily around a stereo-vision system is presented. The main structure and aircraft shape/layout is carried over from the initial 6-meter span design. Jones [1] provided a complete history of the design modifications and experience gleaned from flight test conducted from 2011 to 2017. For the current design, the roll spoilers locations and dimensions are slightly different from reported values in that work. Otherwise, the aircraft are structurally and aerodynamically identical, hence the vehicle is referred to as ATV-6B (aeroelastic testbed) and RRV-6B (risk reduction). Slight variations between RRV-6B and ATV-6B are expected due to manufacturing and assembly variances. However, quality control steps are taken to minimize this variability to ensure reproducible results on both vehicles.

5.2 Airframe Layout

X-HALE (both ATV-6B and RRV-6B) consist of six 1-meter long wing sections, with the tip sections set at a dihedral of 10°. The wings employ EMX-07 airfoil profile with 0.2 m chord length. The wing joiners are designed for load transfer and made from machined aluminum. Roll spoilers are situated on the dihedral wing segments

(Fig. 5.2). For the outboard sections, carbon fiber booms connect all movable tails (acting as elevators) to the main wings (Fig. 5.3). The center tail can only be flipped from the horizontal to vertical orientation to add aerodynamic yaw damping. It is not used for flight control. All tails have NACA 0012 airfoil profile with chord length of 0.12 m.

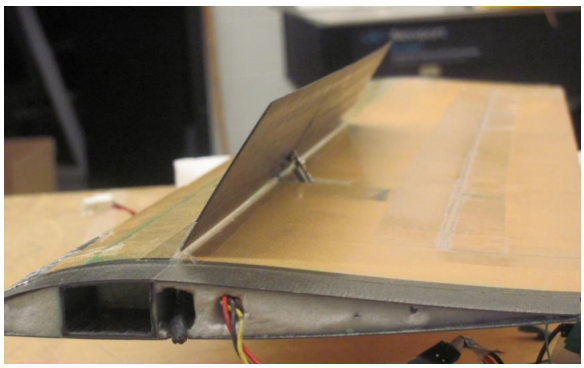

Figure 5.2: X-HALE roll spoiler on wing dihedral employing indirect drive mechanism

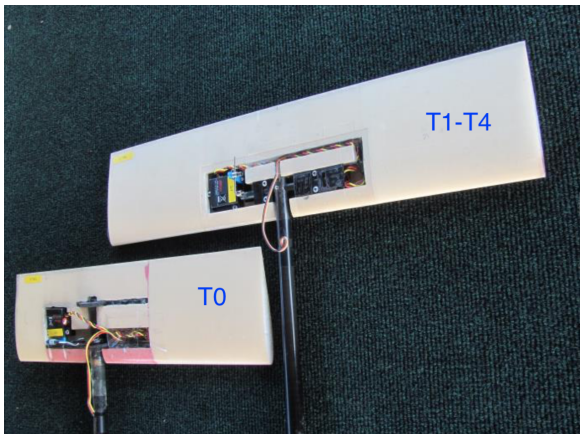

Figure 5.3: X-HALE center and outboard tails employing indirect drive mechanism

Five pods are mounted below the main wing (Figs. 5.4 – 5.6). They house batteries, motors with propellers, sensors, as well as supporting electronics. In RRV-6B, most of the sensor and electrical components are replaced by ballast metal plates. Propellers attached to electric DC motors at the front of each pod provides thrust. 5400 mA h 11.1 V Lithium-Polymer (LiPo) motor batteries power the motors, and are fixed in place by 3-D printed plastic holders. Sensors, supporting electronics, and bal-

last weights (if any) are bolted on the carbon fiber spines at predetermined mounting locations. Flat plates are attached to the center, left, and right inner most booms, acting as ventral fins for lateral stability.

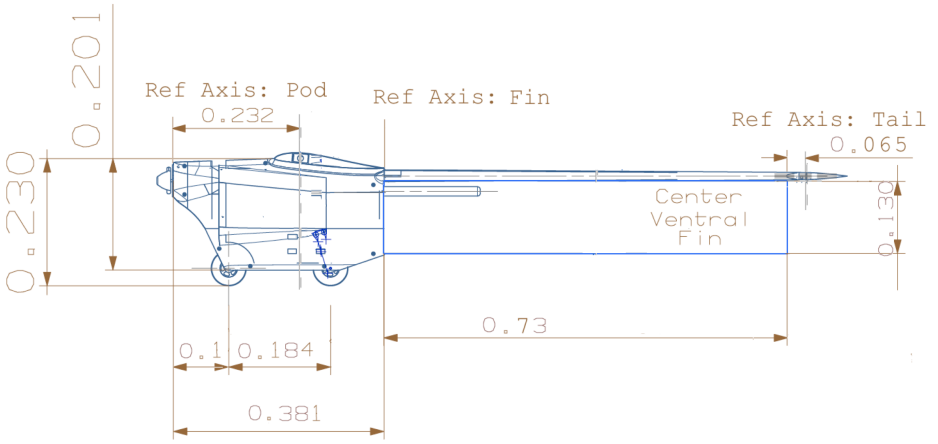

Figure 5.4: X-HALE center pod dimensional drawing (units: meter)

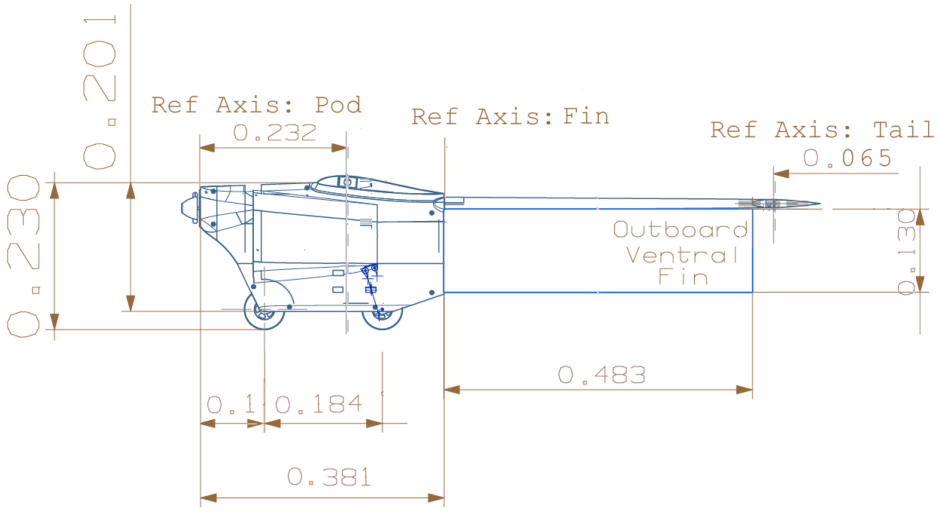

Figure 5.5: X-HALE outboard pod dimensional drawing (units: meter)

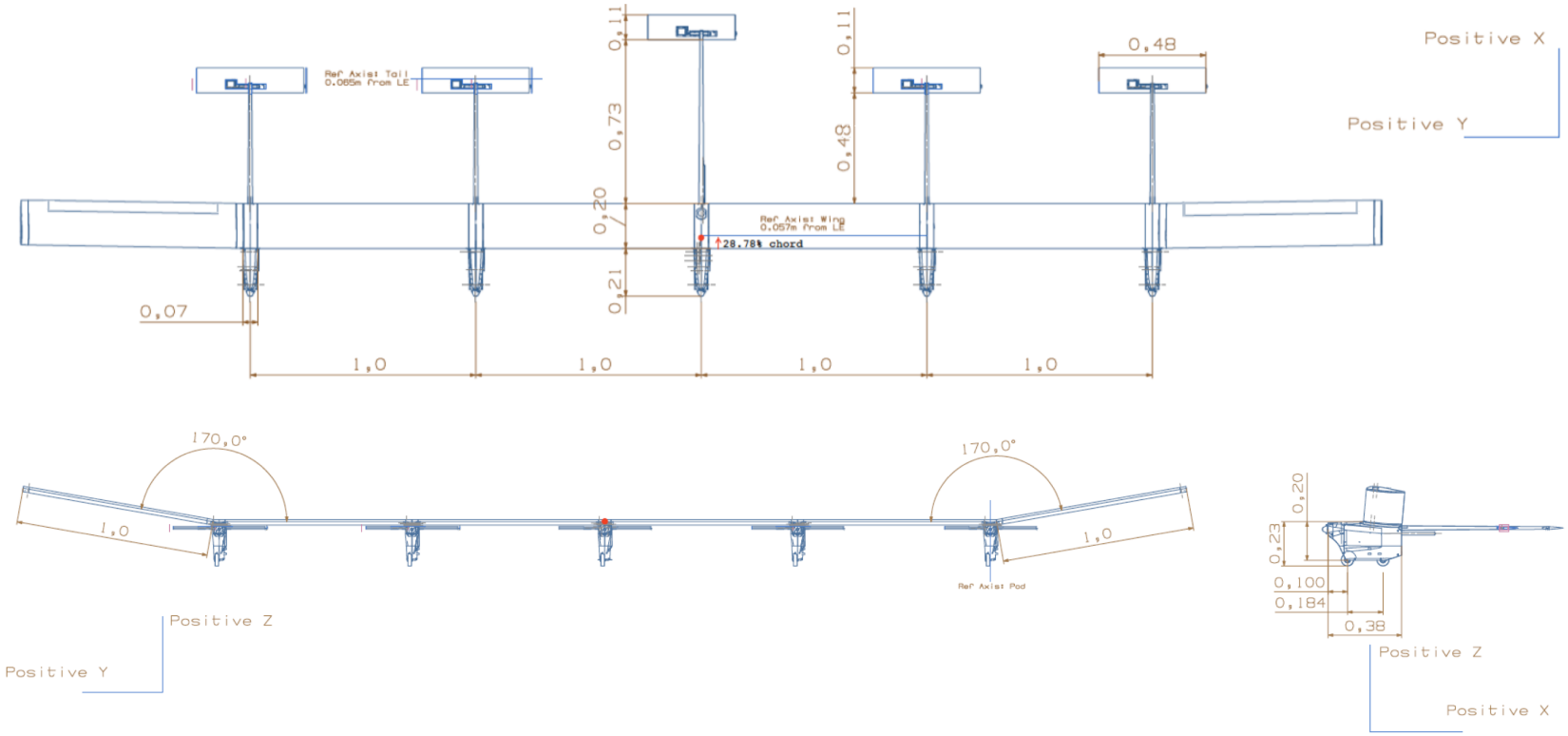

Figure 5.6: X-HALE airframe dimensional drawings (units: meter)

5.2.1 Nomenclature

For ease of referencing structural components for the remainder of the dissertation, X-HALE will be labelled as follows:

- The main wing sections from right tip to left tip are W6, W4, W2, W1, W3, W5.
- The ventral fins from right to left are V2, V0, V1.
- The pods from right to left are F4, F2, F0, F1, F3.
- The propellers from right to left are P4, P2, P0, P1, P3 following pod convention.
- Outboard tails acting as elevators from right to left are T4, T2, T1, T3.
- The center tail which does not act as an elevator is T0.
- The booms from right to left are B4, B2, B0, B1, B3 following tail convention.
- Roll spoilers are RS2 and RS1 for right and left surface, respectively.

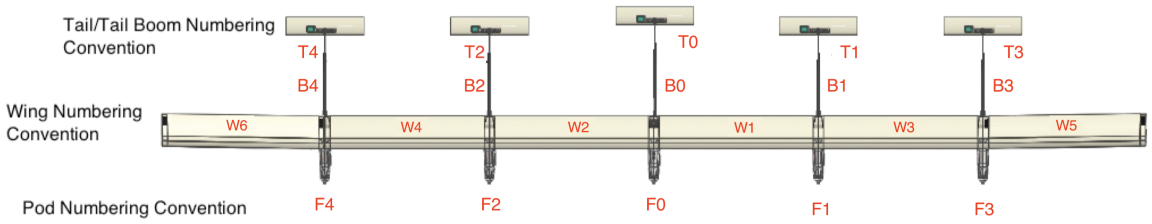

Figure 5.7: X-HALE nomenclature (shown with center tail horizontal)

5.2.2 Structural Properties

The vehicle is characterized using distributed mass and inertias for vehicle wings, booms, and tails while concentrated mass and inertias are used to characterize the

pods and ventral fins. Some components are assumed rigid and stiffness properties will not be reported (Table 5.1). Appendix B.1 lists the mass, inertias and stiffness of X-HALE.

Table 5.1: Structural component properties

Component	Stiffness Properties	Mass and Inertia Properties
Main Wings	Y	Distributed
Tails	N	Distributed
Booms	N	Distributed
Pod	N	Concentrated
Ventral Fins	N	Concentrated

5.2.3 Aerodynamic Properties

The lifting surfaces and their corresponding airfoil shapes are given in Table 5.2. XFOIL [105] data are generated at $Re = 1 \times 10^5$ and $M = 0.03$ with viscous effects enabled. Wind tunnel tests (Ref. [1]) were conducted for the pods without aerodynamic fairings due to its complex geometry. All aerodynamic properties are reported in Appendix B.2.

Table 5.2: Component airfoil profile and data source

Component	Airfoil	Data Source
Main Wings	EMX-07	XFOIL numerical simulation
Booms	–	–
Tails	NACA 0012	XFOIL numerical simulation
Pod (no fairings)	–	Experimental wind tunnel
Ventral Fins	NACA 0010	XFOIL numerical simulation

5.2.4 Servo Properties

All aerodynamic control surfaces employ indirect drive construction. This means that the digital servo is connected to a link rod which push/pull the control surface

given any servo arm movement. The ratio between the servo arm and linkage arms will determine the deflection magnitude of the control surface. Hitec HS-5125MG[1] slim digital servos are used on X-HALE. These servos are controlled via Pulse Width Modulation (PWM) and have an operating range of 0.9 ms to 2.1 ms. The response characteristic is identified experimentally in Appendix B.3. A second order transfer function is fitted to the response and given below:

$$H_{servo} = K_{linkratio}\left[\frac{-55.112s + 697.1}{s^2 + 40.77s + 699.7}\right] \tag{5.1}$$

Table 5.3: Table of $K_{linkratio}$

Servo	$K_{linkratio}$	Units
Elevators	1.6101	rad/ms
Roll spoilers	1.5581	rad/ms

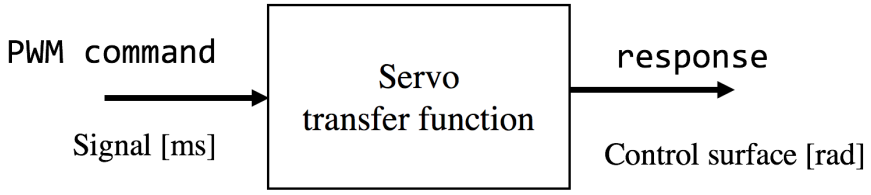

Figure 5.8: Servo transfer function for tails and spoilers

5.2.5 Propulsion Properties

ART-1200 brushless electric DC motors[2] drive APC 12×6E/12×6EP propellers[3]. Castle Creations Phoenix ICE50 Electronic Speed Controllers (ESC)[4] are used to control the DC motors. A PWM command of 1.2 ms corresponds to zero throttle and 1.8 ms corresponds to full throttle.

[1] http://hitecrcd.com/
[2] http://www.arthobby.com/
[3] https://www.apcprop.com/product/12x6e/
[4] http://www.castlecreations.com/en

The propeller response is experimentally verified at sea level static conditions in Appendix B.4. A second order transfer function is fitted to the response and given by:

$$H_{prop} = 178.3 \left[\frac{5.773s + 4.908}{s^2 + 5.721s + 4.908} \right] \tag{5.2}$$

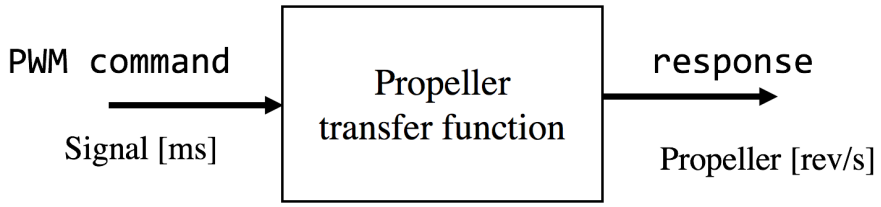

Figure 5.9: Propeller transfer function

5.3 ATV-6B

This section describes component hardware and software specific to ATV-6B. This is a major design change compared to Ref. [104].

5.3.1 Hardware

ATV-6B employs a complete suite of sensors to collect aerodynamic, structural, and flight dynamic information (Fig. 5.10). Pilot has full manual RC control of ATV-6B. Telemetry with the ground station is accomplished using UHF radio. Control surfaces and propellers are controlled using a dedicated servo control card. Aerodynamic data are gathered by three sets of 5-hole probes. Structural data are gathered by four IMUs distributed on the outboard pods. Four cameras take images of wing markers and provide more detailed measurements of the wing shape. Rigid body data are gathered by the inertial navigation system on the center pod. Custom designed PCBs are used for mounting and wire breakout purposes (Fig. 5.11). The following section presents a detailed description of each individual component. Technical

drawings are presented in Appendix D.1 – D.2.

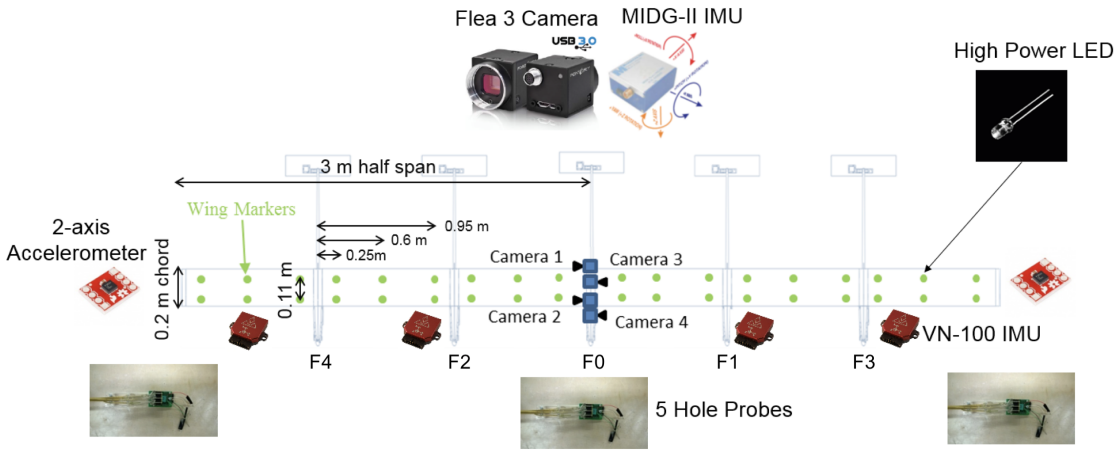

Figure 5.10: ATV-6B sensor placement schematic

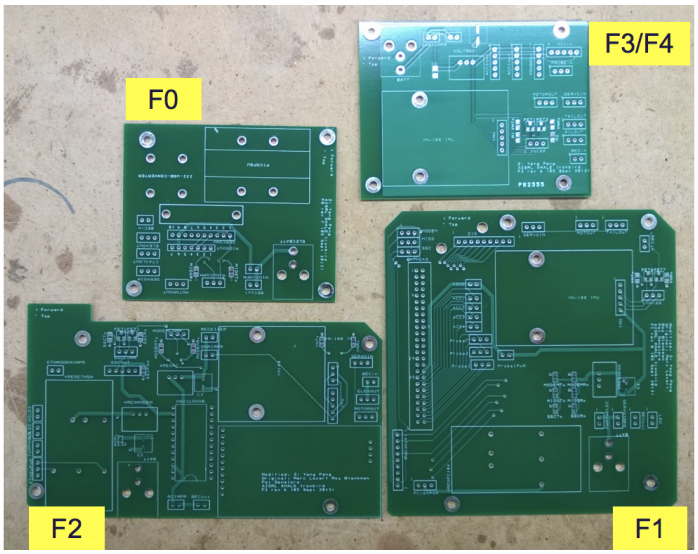

Figure 5.11: Custom design PCB for ATV-6B

5.3.1.1 Onboard Computers

Diamond Systems Athena-II[1] is a ruggedized PC-104 family Single Board Computer (SBC) with integrated DAQ, and mounted in the left inner pod F1 (Fig. 5.12). It is socketed with Intel Pentium 800MHz CPU with 256MB RAM and 1GB compact

[1] http://www.diamondsystems.com/products/athenaii

flash storage. It supports four RS232 serial ports for interfacing with peripherals. The integrated DAQ supports 16 analog inputs utilizing 16-bit analog-to-digital converters, capable of up to 100 kHz sample rate. It also has 24 programmable digital input/output (I/O) ports. Analog voltages are recorded using the integrated DAQ in Single-Ended (SE) mode. One digital I/O port is configured as a 25 Hz digital pulse output for camera triggering. Athena-II acts as the primary flight computer on ATV-6B.

An IEI Nano QM-770 EPIC family SBC[1] is mounted in pod F0. This SBC is socketed with an Intel i5 2.1 GHz CPU with 8 GB RAM. QM-770 was selected in 2013 when the current design was finalized. It was one of the few commercially available SBC which supported four USB3.0 (for the cameras) and four additional USB/RS232 ports (for the distributed IMU). QM-770 primarily acts as a recording device for saving images recorded by the Flea3 cameras and orientation information by the VN-100 IMU. Due to the substantial weight of the QM-770, it is designed to be carried in F0 to prevent asymmetry in the span-wise mass distribution. This resulted in an unusual configuration where the primary flight computer Athena-II is mounted outboard. Two 512 GB Samsung Pro 850 Solid State Drives (SSDs)[2] are used for storage. This is necessitated by the large through-put of image data captured.

5.3.1.2 Five-Hole Probes

For aerodynamic information, three sets of five-hole probes are mounted forward of the main wing, two sets on each wing tip (W5 and W6), and one set in the middle of W1. It is designed such that the pitot openings are far forward of the wing leading edge to measure free-stream pressures, undisturbed by the propeller wash. Each five-hole probe consists of three Allsensor Dx-4V-MINI-series pressure transducers[3]

[1] https://www.ieiworld.com/en/product/model.php?II=151
[2] http://www.samsung.com/semiconductor/minisite/ssd/product/consumer/850pro.html
[3] http://www.allsensors.com/products/mamp-series-prime-grade

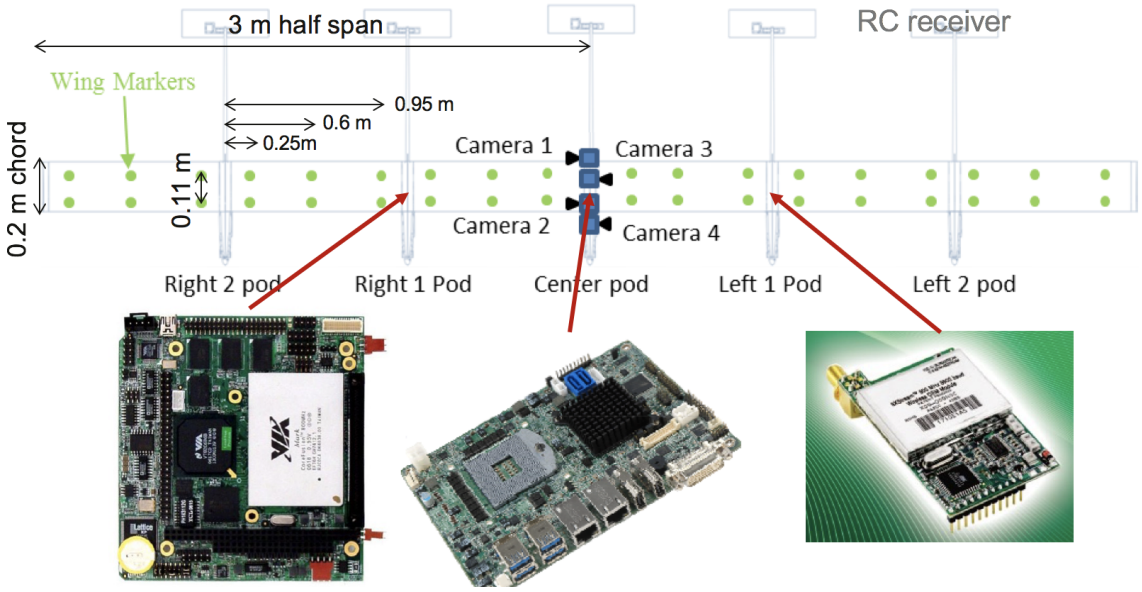

Figure 5.12: ATV-6B onboard computer placement

connected to a custom-made pitot tube assembly arranged in the shape of a cross. The transducer outputs electrical voltage, recorded by the onboard DAQ at 50 Hz, proportional to measured pressure. *A priori* calibration is performed using wind tunnel experiments to convert measured voltages into airspeed, angle of attack, and angle of sideslip information.

5.3.1.3 Accelerometers

Two pairs of Sparkfun 2-axis MEMS accelerometers[1] are mounted at the wingtips, fore and aft of the wing-box. The output voltage level corresponds to the measured acceleration, and recorded by onboard DAQ at 50 Hz.

5.3.1.4 Inertia Measurement Units

Microrobotics MIDG-II Inertial Navigation System (INS) with integrated Global Position System (GPS)[2] is mounted in F0. It provides flight dynamic information of the vehicle center. MIDG-II is connected (via RS422-to-RS232 converter chip) to the

[1] https://www.sparkfun.com/products/retired/844
[2] www.microboticsinc.com, company no longer in business since 2015

Athena-II serial port (baud rate 115,200) and recorded at 50 Hz. Four VectorNav VN-100 IMUs[1] are mounted in outboard pods (F1–F4). They act as orientation sensors measuring the attitude of the pods. IMU on F1 is configured as the master while the rest are configured as slaves. The master IMU will broadcast a 200 Hz sync pulse so that all VN-100 measurements are taken at the same time instant. VN-100 IMUs are connected to the QM-770 (baud rate 115,200) where they are recorded at 50 Hz.

5.3.1.5 Camera System

Four PointGrey Flea3 (FL3-U3-32S2M-CS)[2] computer vision cameras, together with Kowa lens (LMVZ3510-IR)[3] are mounted on F0 (Fig. 5.13), forming two pairs of stereo-vision cameras, looking towards the left and right wing. USB 3.0 connections are used for both data transfer and powering the cameras. To synchronize image capture across all four cameras, they are configured to be externally triggered via GPIO pin 1. A 25-Hz digital pulse from Athena-II is used for this purpose.

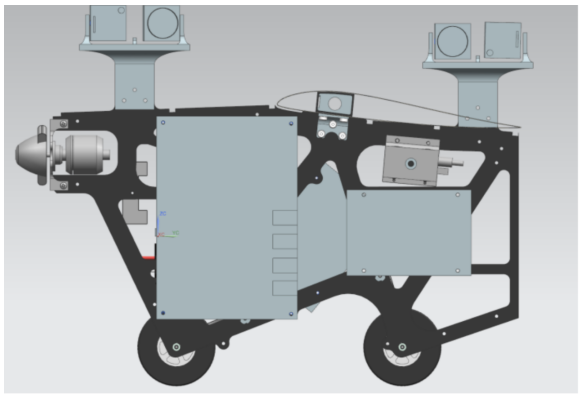

Figure 5.13: Flea3 cameras mounted on center pod

This lens model was selected for good resolution (200 lp/mm), a locking focus ring, and the ability to manually select aperture size (f-stop). A high lens resolution is needed to take advantage of the high pixel count on the Flea3 cameras. A locking

[1] https://www.vectornav.com/products/vn-100

[2] https://www.ptgrey.com/flea3-usb3-vision-cameras

[3] https://lenses.kowa-usa.com/varifocal-day-and-night-megapixel-lenses/554-lmvz3510-ir.html

screw ensures that vibrations in flight will not shift the focus ring and change camera stereo calibration values. Lastly, the aperture size should be set as small as possible to obtain a large depth of view to obtain sharp images for the entirety of the wingspan.

High intensity (10.000 mcd) LEDs[1] are used as tracking targets. They are mounted on the wing top surface in a rectangular 6×2 grid pattern on each 1-meter wing segment (Figs. 5.14 – 5.15).

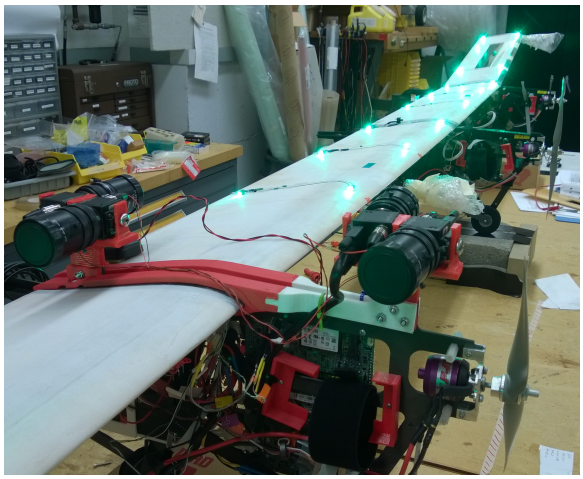

Figure 5.14: High intensity LED targets on wing surface (side view)

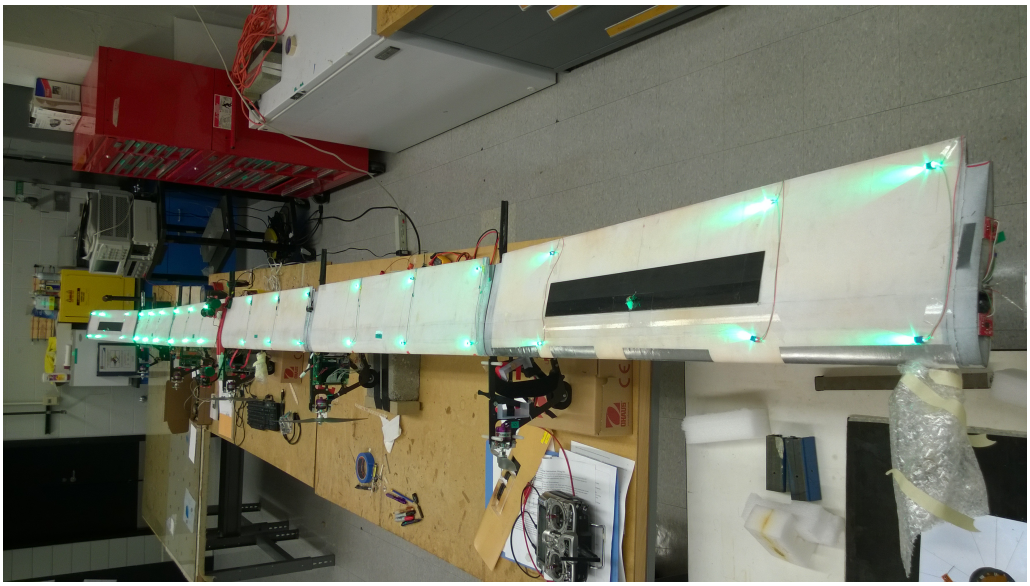

Figure 5.15: High intensity LED targets on wing surface (top view)

[1] https://www.sparkfun.com/products/8285

5.3.1.6 Radio Control, Onboard Communications and Ground Telemetry

JR921 radio receiver[1] is connected to the Servo Switch Controller (SSC)[2] which is then connected to Athena-II RS232 serial port. Internally, the SSC has a switch table to determine if servo outputs are direct passthrough from RC or generated by Athena-II. Onboard communications between Athena-II and QM-770 is done using TCP/IP via a Linksys router (EZXS55W)[3] mounted in F1. Ground telemetry is accomplished via XStream 900 MHz RF modem[4] connected to Athena-II serial port (baud rate 57,600) via 5 V TTL-RS232 converter.

5.3.1.7 Batteries and Power Distribution

A 14.8 V 2100 mA h LiPo electronics battery is carried on both F3 and F4. One is used to power the QM-770 while the other is used to power all other electronics (e.g., Athena-II, servos, and sensors). This is because the QM-770 and the Flea3 cameras have a much higher power draw compared to the rest of the system. The run time of both computers exceed 30 minutes under full load conditions (all sensors captured and saved to disk).

5.3.2 Software

Custom written C and C++ software adapted from Refs. [106, 107] are deployed on Athena-II and QM-770 to interface with sensors and other electronics. Athena-II runs QNX Neutrino Real-Time Operating System (RTOS)[5]. QNX software development and compilation is done using QNX Momentics Tools Suite[6]. QM-770 runs

[1] http://www.jramericas.com/179943/JRPR921X/?pcat=474
[2] www.microboticsinc.com, company no longer in business
[3] https://www.linksys.com/us/support-product?pid=01t80000003KRuBAAW
[4] https://www.digikey.com/product-detail/en/digi-international/X09-019NSC/X09-019NSC-ND/615848
[5] http://blackberry.qnx.com/en/products/neutrino-rtos/neutrino-rtos
[6] http://blackberry.qnx.com/en/products/tools/qnx-momentics

Ubuntu 12.04.3 LTS Server operating system[1]. The software stack consists of a main executable on both Athena-II and QM-770 running in the foreground on top of their respective operating systems.

5.3.2.1 Athena-II Software Stack

The primary flight computer Athena-II runs QNX 6.5.0 RTOS. This is to allow real-time, consistent execution of the software processes governing flight control and data measurements. The main executable spawns four threads, each running a submodule:

- Sensor acquisition
- Camera triggering
- Onboard communications
- Ground telemetry

The root process will monitor operator commands sent from the ground station (via ground telemetry module), broadcast the commands (via onboard communications module) and terminate all threads when termination command is received. The state flow diagram of possible operator commands is shown in Fig. 5.16. This flow allows toggling between states `CAPTURE` (data being recorded to disk) and `NOCAPTURE` (data not being recorded to disk). The operator can therefore selectively record only useful segments of the flight. Each time `INIT` state is entered, the current data file (if any) is finalized and closed, and a new data file is created. In the event of power failure during `CAPTURE` state, only the current data file will be corrupted while remaining files will be unaffected.

[1] https://www.ubuntu.com

Table 5.4: State-flow description

State name	Description
OFFLINE	Computers not connected
STANDBY	Computers connected via TCP Port 1333
INIT	Computers initializes onboard sensors and open new data file
CAPTURE	Computers saving sensor measurements to disk
NOCAPTURE	Computers stop recording, flushes all buffer
EXIT	Computers terminates threads
SHUTDOWN	Computers disconnects, root process exits

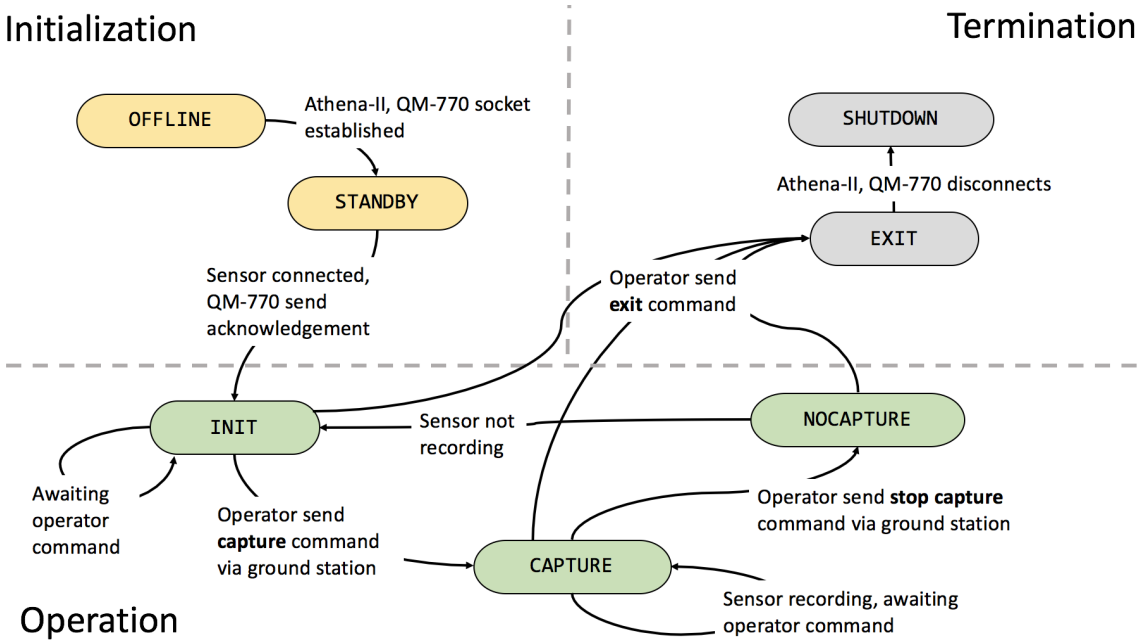

Figure 5.16: Stateflow diagram of Athena-II and QM-770

Sensor measurement module primarily interfaces with hardware DAQ and MIDG-II. It employs Diamond System Universal Driver (DSCUD) v6.02[1], which provides a high level, programmatic way of accessing DAQ board functions. Analog voltage channels are scanned and saved to disk. Flight information (vehicle attitude, attitude rates, GPS inertial position, and velocity) from MIDG-II, as well as RC pilot inputs from SSC are accessed and saved to disk. Camera trigger module uses DSCUD to generate a 25 Hz digital pulse on digital output port for Flea3 camera

[1] http://www.diamondsystems.com/products/software.php

triggering. Onboard communications module is tasked with socket communications between Athena-II and QM-770. TCP port 1333 is used for command and control using MAVLink[1] as the communication protocol. Athena-II acts as the server and QM-770 connects as a client. Ground telemetry module broadcasts collected information to the UHF radio. Precision Time Protocol v2.2[2] runs as a background daemon to synchronize time between Athena-II and QM-770.

5.3.2.2 QM-770 Software Stack

QM-770 runs Ubuntu 12.04.3 LTS for compatibility with PointGrey FlyCapture2 SDK[3]. Similar to the Athena-II software stack, the main executable spawns three threads:

- Camera control
- VN-100 control
- Onboard communications

The threads perform periodic task based on a software timer. The root process monitors operator commands (via onboard communications module), broadcast sensor readings/messages (via onboard communications module) and terminates all threads when termination command is received. The camera control module employs Flycapture2 API calls, which provide a high-level C++ interface for camera control and image transfer. Frame count and frame timer are embedded in the first 8 bits of the image. If Flea3 hardware buffer fills up faster than software retrieval, the oldest image is overwritten in a First In, First Out (FIFO) manner, resulting in dropped frames. Each camera image has dimensions of 2080×1552 pixels. It is saved in 8-bit

[1] http://www.qgroundcontrol.org/mavlink/start
[2] https://github.com/ptpd/ptpd
[3] https://www.ptgrey.com/flycapture-sdk

greyscale format (`MONO8`). As mentioned previously, the cameras are triggered externally, hence the capture frequency is controlled by Athena-II. VN-100 control module employs VectorNav C++ API[1] to configure and communicate with all four VN-100. An asynchronous callback function (provided by VectorNav API) will be used to retrieve latest sensor readings at 50 Hz. This will be written to disk and the thread will sleep until next cycle. Onboard communication module is responsible for receiving command and control messages from Athena-II primary flight computer on TCP Port 1333. It is also used to send sensor readings from QM-770 back to Athena-II to be down-linked for ground monitoring. The state flow diagram of possible operator commands is identical to Athena-II as shown in Fig. 5.16.

5.4 RRV-6B

This section describes hardware and software components specific to RRV-6B. A low-cost sensor suite is implemented in place of ATV-6B components.

5.4.1 Hardware

To allow preliminary data collection, as well as the ability to implement an stabilizing autopilot, Pixhawk was selected as a capable and low-cost replacement computer. In the following section, the components unique to the RRV are detailed.

5.4.1.1 Ballast Weights

Metal plates of various sizes and thickness are bolted on the pods to adjust the weight and center of gravity of each pod to match ATV-6B. On F0, 3D printed dummy shells with steel ballasts are mounted at the exact location of the actual cameras (Fig. 5.17). This serves to simulate both weight and aerodynamic effects.

[1] https://www.vectornav.com/support/downloads

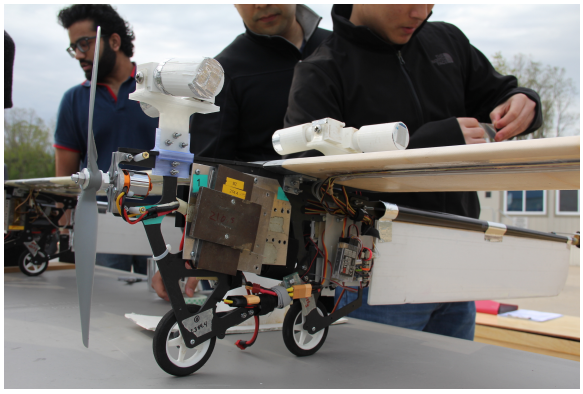

Figure 5.17: Ballast weights and dummy cameras on RRV-6B

5.4.1.2 Batteries and Power Distribution

Since there is a much lower power requirement, the RRV-6B is powered by a 11.1V 1100 mA h LiPo battery mounted in F0. The electronics batteries on ATV-6B (in F3 and F4) are replaced with ballast weights. Pixhawk does not provide a working 5 V power rail at the servo output ports. Therefore, a Castle Creations Battery Elimination Circuit (BEC)[1] is used to regulate power to drive the servos. This BEC is also used to power the radio receiver.

5.4.1.3 Pixhawk

Pixhawk[2] is an open-hardware autopilot jointly developed by 3DR Robotics, Ardupilot Group and ETH Zurich. It packs a 168 MHz ARM processor, with 3-axis accelerometers, gyroscope, magnetometer, and barometer. GPS and pitot-static pressure probe are also connected via Inter-Integrated Circuits (I2C) ports. Telemetry link with ground station is provided via 3DR 900 MHz radio[3]. Pilot commands are received from JR921 radio receiver and transmitted to the Pixhawk.

[1] http://www.castlecreations.com/en/fixed-wing/cc-bec-010-0004-00
[2] http://www.pixhawk.org
[3] https://store.3dr.com/products/915-mhz-telemetry-radio

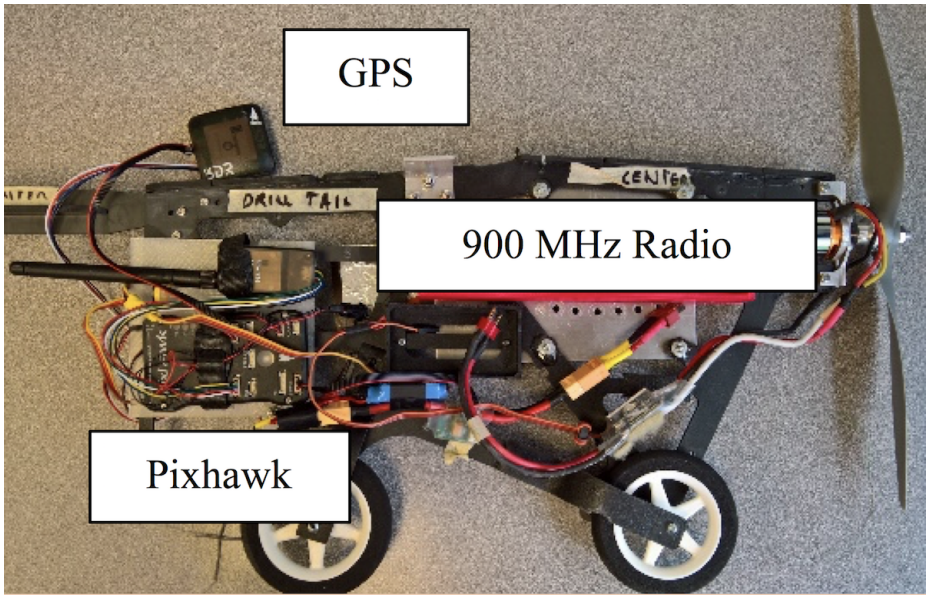

Figure 5.18: Pixhawk mounted on pod F0 in RRV-6B

5.4.2 Software

RRV-6B runs PX4[1] software on the Pixhawk. PX4 is an open-source full stack solution, consisting of PX4 flight stack (flight operations) and PX4 middleware (low-level drivers and data routing) running on top of NuttX RTOS[2]. For the complete overview of the PX4 software, the reader is directed to the PX4 user manual [108]. Code modifications, branched off release v1.4.1, were made to customize PX4 for RRV-6B operations and detailed below. Figure 5.19 shows the architecture overview with code modifications denoted with asterisks. `sdlog2` logger application provided by PX4 is used to record all sensors measurements at 100 Hz.

5.4.2.1 Modified `sensors` Module

Code modifications in `sensors` module allows a user defined excitation signal to be generated in software and added to the pilot stick command. This allows a clean, reproducible excitation during flight tests. The signal (e.g., frequency, amplitude,

[1] http://px4.io
[2] http://www.nuttx.org

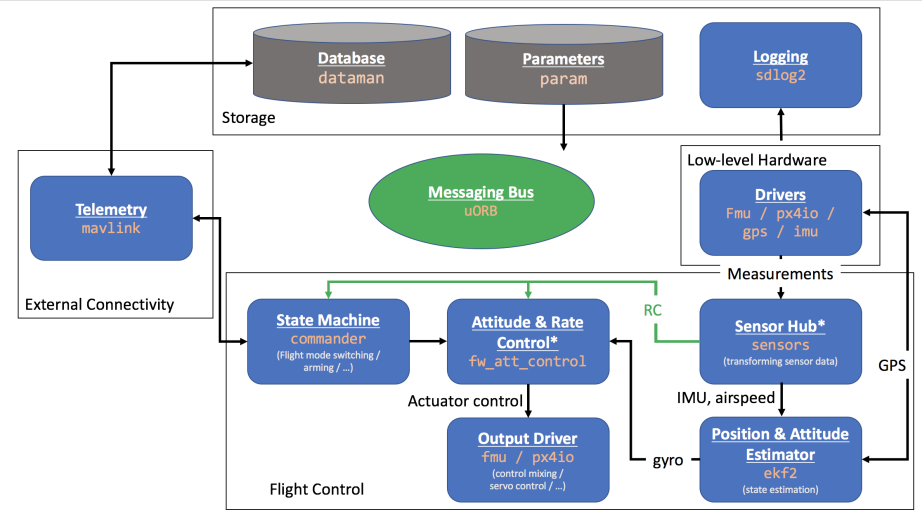

Figure 5.19: PX4 software modifications highlighted with asterisks

and channel) is selected via the ground station (Fig. 5.20). Test cards can also be pre-programmed and loaded on Pixhawk. The pilot activates the signal injection using a dedicated spring-loaded toggle switch on the radio transmitter. Details are shown in Appendix D.3.

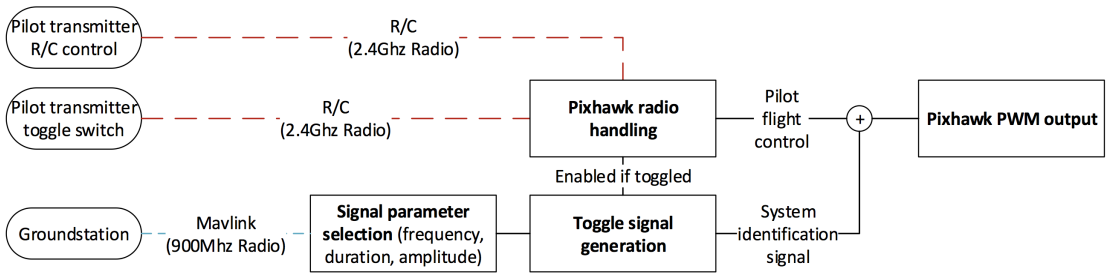

Figure 5.20: `sensors` module modification showing signal injection path

5.4.2.2 Modified `fw_att_control` Module

In the original PX4 code[1], the autopilot is implemented in `fw_att_control` and `ecl` modules. The provided autopilot consists of a cascaded two-loop architecture (Fig. 5.21). The inner feedback loop consists of a Proportional Integral (PI) controller

[1] https://github.com/PX4/Firmware/tree/v1.4.1

regulating the vehicle attitude rate to track a commanded rate setpoint value. The outer loop consists of a proportional controller regulating the vehicle attitude to track a commanded angle setpoint value. In the original implementation, in `MANUAL` flight mode, a direct pass through from pilot input (via `sensors`) is written directly into commanded output of `fw_att_control`. In `STABILIZED` flight mode, pilot input controls the pitch and roll angles while the yaw and throttle channels are direct pass through (no automated control).

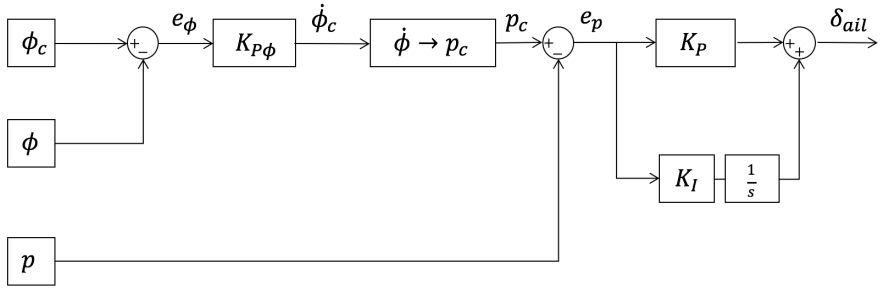

Figure 5.21: PX4 `fw_att_control` control architecture shown for roll control loop

On RRV-6B, the flight modes are modified and detailed in Table 5.5. `MANUAL` flight mode will be used in system identification maneuver to control the control surfaces directly. `STABILIZED` flight mode will be used to activate the autopilot for stabilization. `fw_att_control` and `ecl` modules are modified to include yaw rate control using proportional gain feedback. This autopilot is detailed in Section 7.3.

Table 5.5: Flight mode description

Flight Mode	RRV-6B
`MANUAL`	Direct passthrough + signal injection
`STABILIZED`	Pitch, roll angle controlled Yaw angular rate controlled Passthrough for throttle

The output of `fw_att_control` are pitch control command $\delta_{e,cmd}$, roll control command $\delta_{s,cmd}$, yaw control command $\delta_{r,cmd}$, and thrust control channel $\delta_{t,cmd}$. These values are non-dimensional between $[-1, 1]$. The commands then are mixed in `mixer`

module (Table 5.6) before being distributed to the servos. Differential thrust is limited to ±25% of the full throttle range to prevent aircraft stall due to insufficient airspeed.

Table 5.6: Pixhawk mixing table

Effector	Control Mixing
T1 – T4	$\delta_{e,cmd}$
P0	$\delta_{t,cmd}$
P1, P3	$\delta_{t,cmd} + 0.25\delta_{r,cmd}$
P2, P4	$\delta_{t,cmd} - 0.25\delta_{r,cmd}$
RS1	$-\delta_{s,cmd}{}^*$
RS2	$\delta_{s,cmd}{}^*$

Note: Negative roll spoiler commands are neglected

5.5 Ground Station

This section presents the ground station used to monitor and control X-HALE.

5.5.1 Hardware

Although both ATV-6B and RRV-6B use 900 MHz radio, it is not possible to standardize to a common radio model due to physical footprint and integration issues. For ATV-6B, Xstream 900 MHz radio is mounted on a Xstream X-BIR[1] board for convenient RS232 connection to ground station laptop. On the other hand, 3DR radio on RRV-6B has an inbuilt FDTI RS232 to USB chip and can be connected directly to ground station laptop USB port. Intermittent drops in telemetry link between the ground station and RRV-6B occur with stock omni-direction antenna. To solve this issue, an antenna tracker (Fig. 5.22) is added to improve the telemetry link quality. This tracker uses the Pixhawk hardware and runs Ardupilot antenna tracker firmware[2]. The ground antenna is also changed to a directional 8 dBi flat

[1] https://www.digikey.com/product-detail/en/digi-international/XIB-R/XIB-R-ND/765907

[2] http://ardupilot.org/antennatracker/index.html

patch antenna[1]. Received Signal Strength Indication (RSSI) improved by a huge margin and no telemetry drops are experienced when operating in excess of 400 m. Details are presented in Appendix D.4.

Figure 5.22: Antenna tracker with patch antenna

5.5.2 Software

Since both ATV-6B and RRV-6B employs MAVLink communication protocol, several open-source software and Graphical User Interface (GUI) clients can be used. After evaluating X-HALE flight operations, the following architecture is selected. MAVProxy[2] is used to broadcast the MAVlink stream from X-HALE to multiple ground station laptops (Figs. 5.23 – 5.24). Every ground station laptop has full control. This allows redundancies in the event of computer failure in one laptop. Different laptops are used for different purposes: One is used by the Test Director to coordinate with the Pilot, one laptop is used by the Flight Safety to monitor the vehicle for safety violations (e.g., breaching geofence), the last laptop is used by the Ground Station Operator for changing parameters on the UAV (e.g., switching on/off flight recording, changing autopilot gains, and loading of test cards). Qgroundcontrol[3]

[1] http://www.l-com.com/wireless-antenna-900-mhz-8-dbi-flat-patch-antennas
[2] http://ardupilot.github.io/MAVProxy/html/index.html
[3] http://qgroundcontrol.com

(QGC) recommended by Pixhawk development team is selected as the GUI.

Figure 5.23: Mavproxy broadcast to multiple ground stations

5.6 Sensor Selection Process

During the re-design of the sensor measurement system, a survey of existing technologies was first carried out. After weighing the feasibility, as well as technical advantages/disadvantages of various systems, a stereo vision-based measurement system was chosen. An attempt was made to rank performance metrics of each measurement type, although the exact implementation details may affect this ranking. Table 5.7 shows that the strain-gauge based system has serious limitation on data quality and noise rejection in the field, and has been eliminated from further considerations. On the other hand, FBG based systems have excellent performance metrics, but then commercial implementations were too heavy and too bulky for small UAV applications.

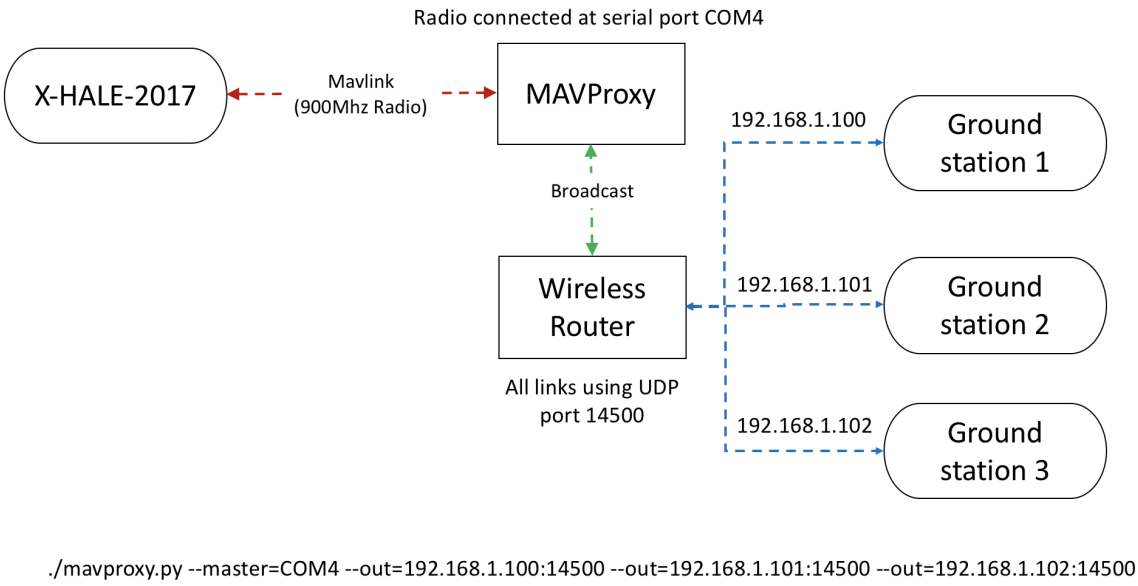

Figure 5.24: Mavproxy network diagram

Table 5.7: Relative performance metrics [2]

Performance Measure	Vision-Based	Strain Gauge-Based	FBG-based
Weight and Volume	+	++	-
Power Consumption	+	++	-
Data Quality	++	-	++
Noise Rejection	+	- -	++

Note: + indicates superior performance

CHAPTER 6

Numerical Results

In this chapter, numerical validation of theoretical development is presented. In Section 6.1, UM/NAST simulation of a virtual sensor mounted on an uniform cantilevered beam is compared against MSC NASTRAN nonlinear simulation (`SOL400`) to validate newly developed kinematics relationships. Section 6.2 details the creation of X-HALE model in UM/NAST for numerical simulations. The quality of the linearization is compared against UM/NAST nonlinear simulation. In Section 6.3, a numerical study using a cantilevered X-HALE main wing is performed to quantity the performance of the sensor fusion algorithms developed. Effects of noise on the accuracy of the shape recovery are also explored. Section 6.4 presents the design and analysis of MPC in maneuver load alleviation of X-HALE in UM/NAST.

6.1 Virtual Sensors Benchmark with Tip Loaded Clamped Beam

To verify the theoretical aspects of the sensor kinematics derived in Section 4.2, a numerical benchmark case is created. A simple tip loaded, uniform beam is created in UM/NAST and MSC NASTRAN. `SOL400` nonlinear analysis is used for results comparison. The beam is modeled using 20 strain elements in UM/NAST and 20 `CBEAM` elements in NASTRAN. The beam geometry is shown in Fig. 6.1 and the

beam properties are defined in Table 6.1. A `RBE2` rigid element is attached at 82.5% span location and have an offset of 0.2 m towards the rear of the beam. Linear and angular displacements, velocities, and accelerations are extracted at the tip of this `RBE2` element.

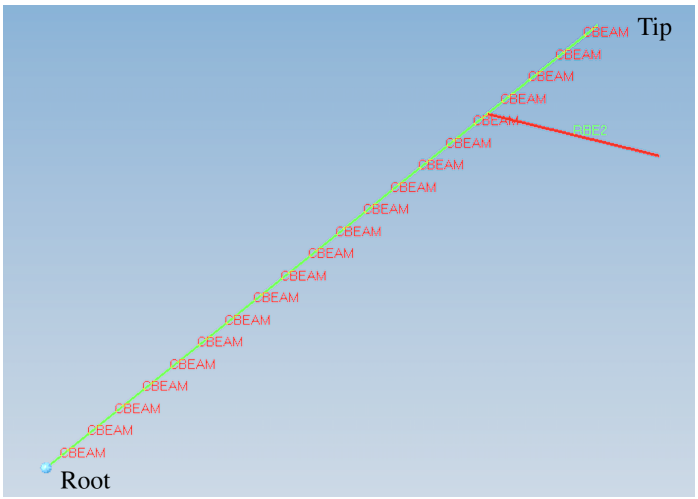

Figure 6.1: Benchmark beam test case in NASTRAN

Table 6.1: Beam properties

		Units
Ref. axis location (from L.E.)	50.0	% chord
Center of gravity (from L.E.)	50.0	% chord
Mass	10	$\mathrm{kg\,m^{-1}}$
Out-of-plane bending inertia (I_{yy})	5×10^{-4}	kg m
In-plane bending inertia (I_{zz})	1.25×10^{-2}	kg m
In/Out-of-plane bending inertia (I_{yz})	0	kg m
Extensional Stiffness (k_{11})	1×10^{6}	$\mathrm{N\,m^{-2}}$
Ext./Out-of-plane bending stiffness (k_{12})	0	$\mathrm{N\,m^{-2}}$
Ext./Out-of-plane bending stiffness (k_{13})	0	$\mathrm{N\,m^{-2}}$
Ext./In-plane bending stiffness (k_{14})	0	$\mathrm{N\,m^{-2}}$
Torsional stiffness (k_{22})	8×10^{1}	$\mathrm{N\,m^{-2}}$
Out-of-plane bending stiffness (k_{33})	5×10^{1}	$\mathrm{N\,m^{-2}}$
Out/In-plane bending stiffness (k_{34})	0	$\mathrm{N\,m^{-2}}$
In-plane bending stiffness (k_{44})	1.25×10^{3}	$\mathrm{N\,m^{-2}}$

In UM/NAST, a virtual sensor is placed such that its location is coincident with the tip of the `RBE2`. This allows direct comparison of the virtual sensor computed

quantities from Eqs. (2.56) – (2.60) against NASTRAN results. Note that this virtual sensor location is located in middle of a strain element. Since NASTRAN reports all results in the body fixed frame, the following comparison is reported using UM/NAST body B frame definition.

Two loading cases are simulated:

1. Body fixed, tip point load of $30 \sin 20t$ N
2. Body fixed, tip point moment of $30 \sin 20t$ N m

6.1.1 UM/NAST and NASTRAN Comparison

Figure 6.2 shows the computed virtual sensor output against NASTRAN nonlinear `SOL400` simulation for an applied tip force. The deformation is predominantly in the out-of-plane bending direction (dz). The changing curvature of the beam manifest as a change in the rotation of the local beam frame in the y direction ($d\theta$). The linear velocities (u, v, w) and angular rates (p, q, r) are results of the shortening and out-of-plane displacement response. All computed parameters match very well between NASTRAN and UM/NAST simulation. NASTRAN shows very small numerical oscillations $O(10^{-8})$ even though the beam is decoupled in twist, in-plane, and out-of-plane degrees of freedom. UM/NAST does not exhibit such numerical oscillations. However, the computed acceleration from UM/NAST is oscillatory compared to NASTRAN. Acceleration is computed from $\ddot{\varepsilon}$ and $\dot{\beta}$ in Eq. (2.60) using:

$$\dot{q}_k = {Q_1}^{-1} Q_2 q_k + {Q_1}^{-1} R(q_k, \dot{q}_{k-1}, u_k) \tag{6.1}$$

These oscillations are numerical artifacts from the explicit trapezoidal time integration scheme. The magnitude of oscillations depends on the relative size of the simulation time step and highest frequency of the model.

Figure 6.3 shows the computed response for an applied tip moment. The sole deformation is in the twist direction (x). There will not be any linear displacement of the beam reference line, instead any twist deformation manifest as tangential displacement and velocity at the sensor. There is excellent correlation between NASTRAN and UM/NAST numerical simulations. Again, NASTRAN shows very small numerical oscillations. On the other hand, UM/NAST correctly matches theoretical dynamics of a decoupled beam.

6.1.2 Linearized and Nonlinear Comparison

For the simple beam presented in Fig. 6.1, the virtual sensor is linearized about the undeformed configuration. The measurements of the virtual sensor are:

$$y = \begin{bmatrix} a_{sensor} & v_{sensor} & p_{sensor} & \omega_{sensor} & \Theta_{sensor} \end{bmatrix}^T \tag{6.2}$$

The beam consists of 20 elements, giving 80 strain and 80 strain rate states:

$$q = \begin{bmatrix} \varepsilon_{80\times 1} & \dot{\varepsilon}_{80\times 1} \end{bmatrix}^T \tag{6.3}$$

and consequently, the output matrix $C \in \mathbb{R}^{15\times 160}$. The linearized output measurements given system states q can be written as:

$$\tilde{y} = C\tilde{q} \tag{6.4}$$

Figure 6.4 compares the linearized virtual sensor measurements computed using Eq. (6.4) against nonlinear virtual sensor response computed from UM/NAST. The linearized response closely mirrors the nonlinear response in most measurements. Discrepancies are noted in the axial direction (displacements, velocities, and accelerations). The linearized axial response is much smaller in magnitude compared to the

actual nonlinear response. This is likely is due geometric nonlinear shortening effects, which is not captured by linearization.

6.2 UM/NAST X-HALE Model

This section presents the creation of the X-HALE model (Fig. 6.5) in UM/NAST for numerical studies.

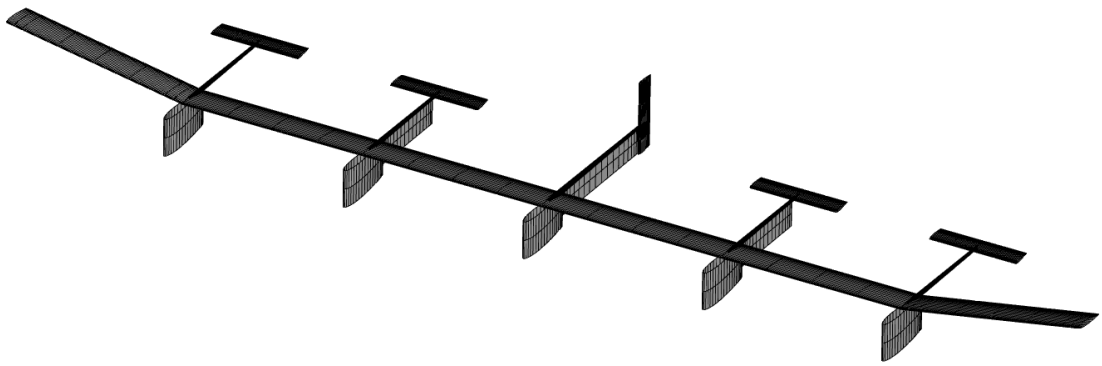

Figure 6.5: 3D mesh of X-HALE model

6.2.1 FEM Properties

The beam reference line is created using vehicle geometry information in Section 5.2. Each component is modeled using one or more member, and each member is discretized into one or more strain elements. Structural and aerodynamic properties of each element are taken from Appendix B.1 and Appendix B.2, respectively. Control surfaces and propellers are modeled using properties from Appendix B.3 and Appendix B.4, respectively. A stick plot of the beam reference axis is shown in Fig. 6.6. Key points used to create this beam reference axes are reported in Appendix C.1.

Only the main wing is modeled as flexible, and all other elements are rigid. All surfaces other than the booms are modeled as lifting surfaces (Fig. 6.7). Due to modeling restrictions, each tail comprises of a left and right tail surface. However,

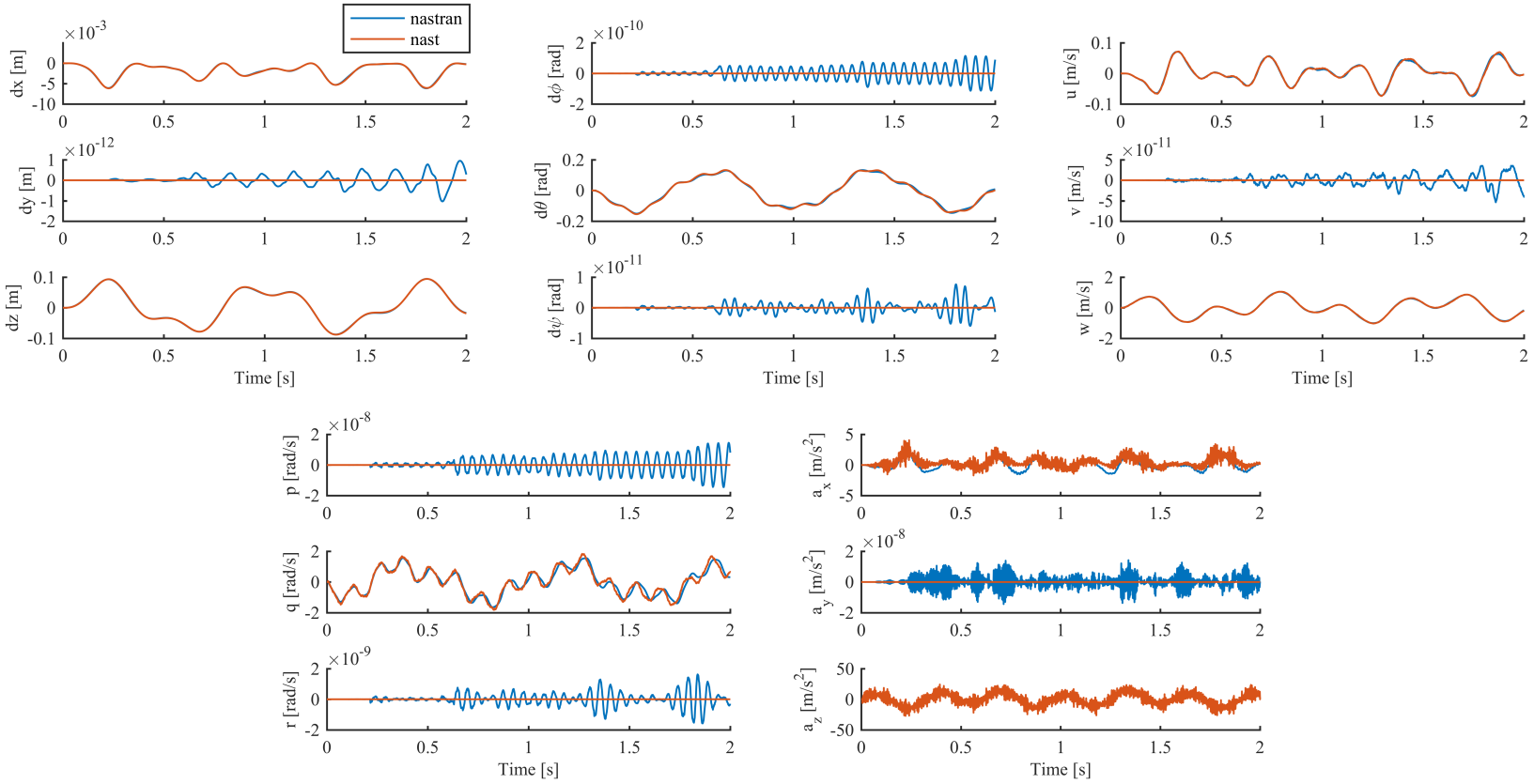

Figure 6.2: Nonlinear comparison of NASTRAN and UM/NAST at virtual sensor for applied tip force

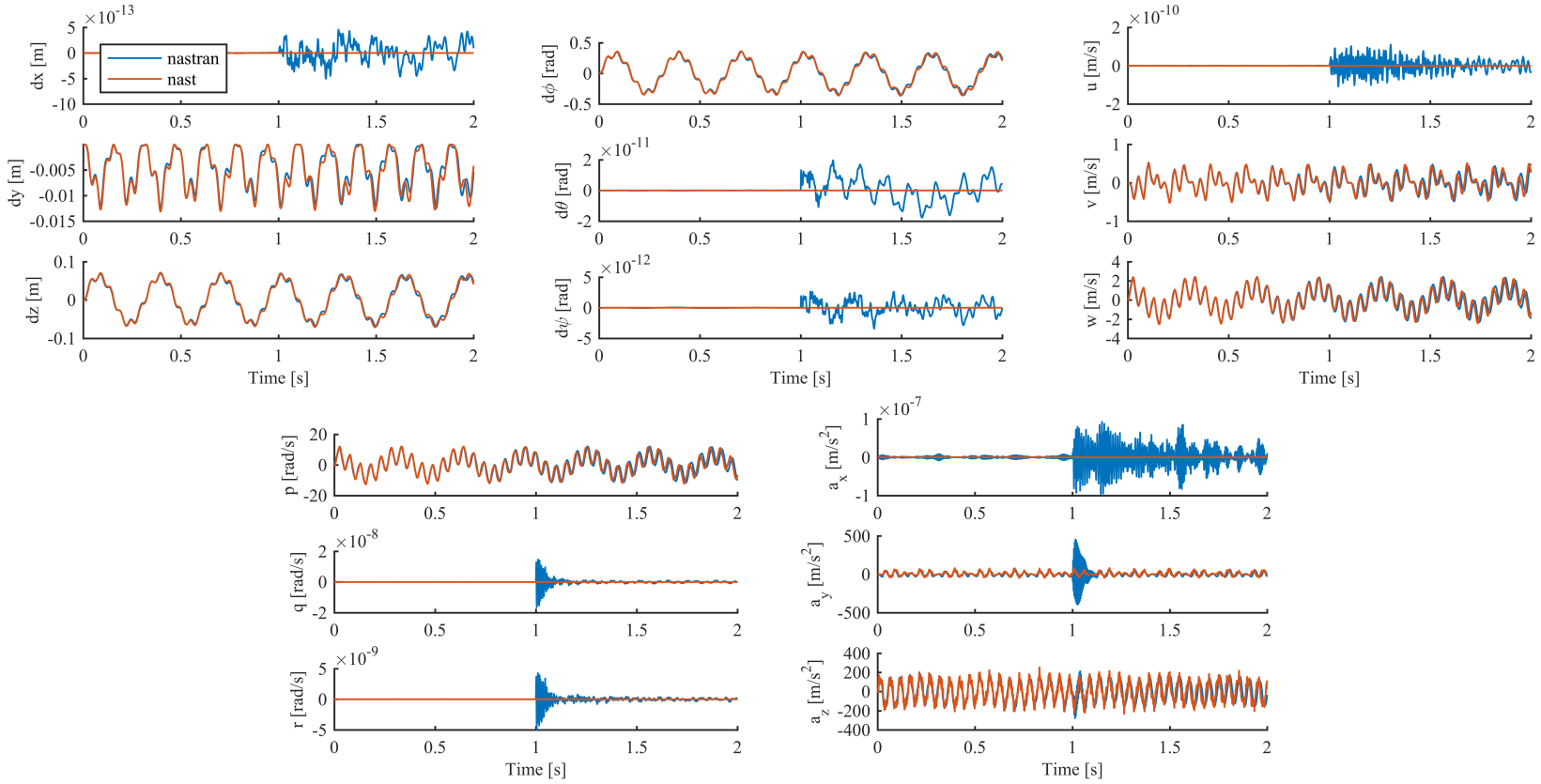

Figure 6.3: Nonlinear comparison of NASTRAN and UM/NAST at virtual sensor for applied tip moment

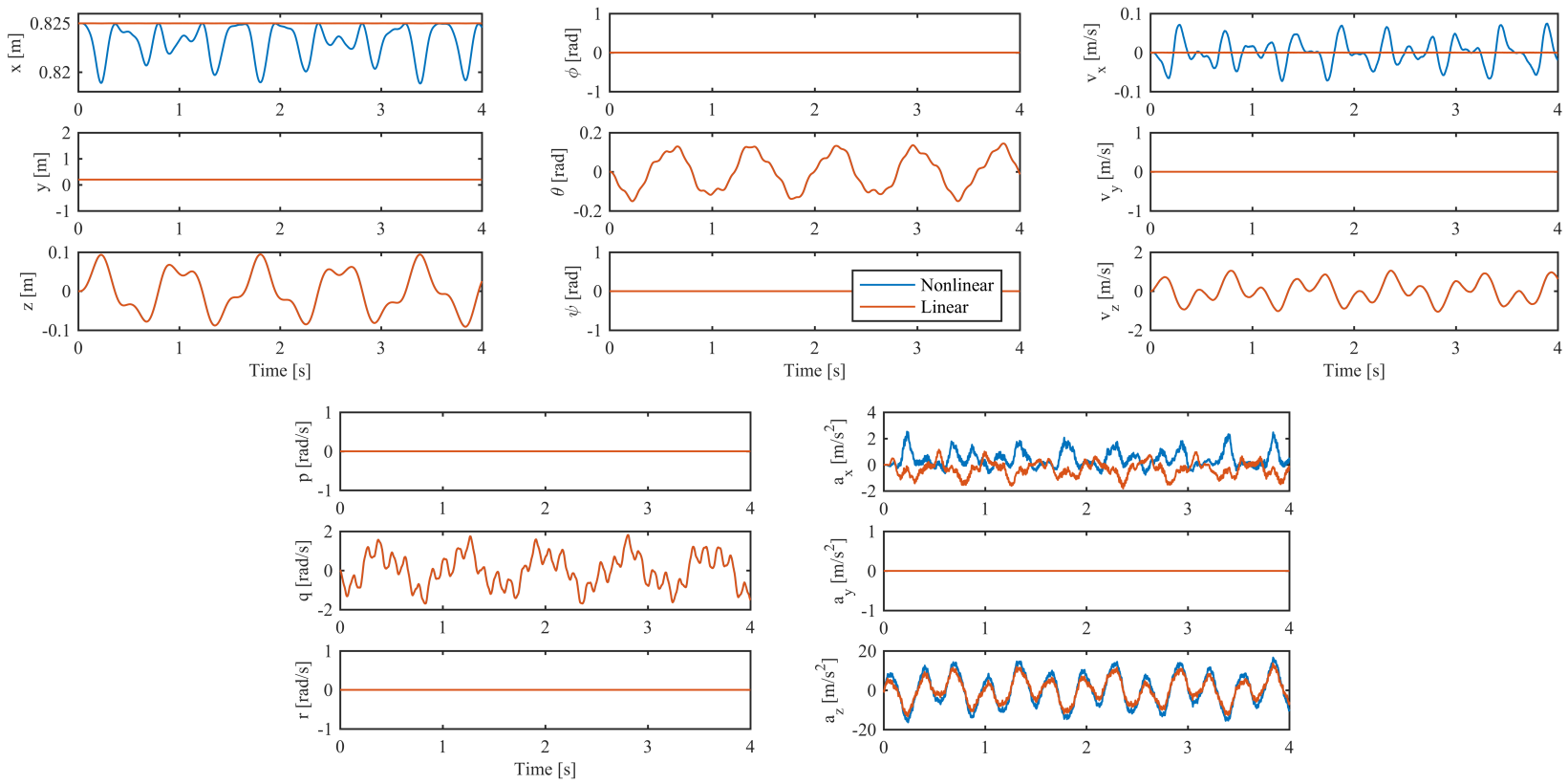

Figure 6.4: Comparsion of linearized and nonlinear sensor measurements of benchmark beam with applied tip force

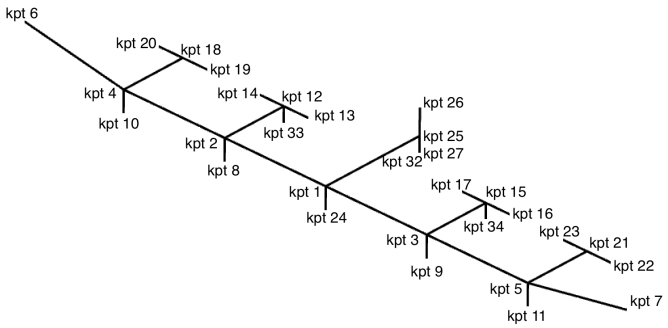

Figure 6.6: Key points definition of beam reference axis

they will not be allowed to actuate separately. The dihedrals wing members are discretized as four elements to accommodate the definition of a roll spoiler. The member discretization and properties are presented in Appendix C.2. From knowledge of the beam reference line and offset of each component, three-dimensional position can be reconstructed. This model has total of 16 flexible elements and 34 lifting elements.

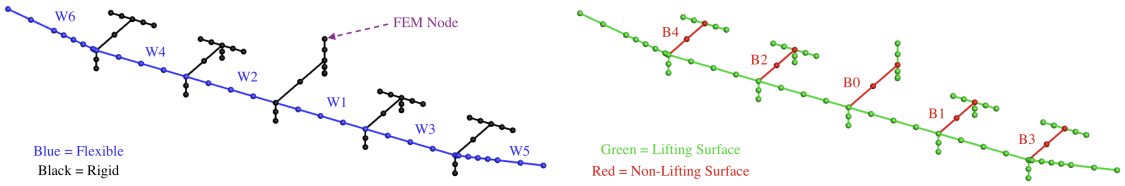

Figure 6.7: Flexible and lifting elements in UM/NAST model

6.2.2 Vehicle Trim

The X-HALE model is trimmed at steady level 1g flight with altitude of 30 m, at flight speed of $14\,\mathrm{m\,s^{-1}}$. For trim, elevators (T1 – T4) are deflected symmetrically. The propellers are actuated asymmetrically (P2,P4 vs P1,P3) to act as differential thrust to balance yawing moment. All five propellers (P0 – P5) are actuated symmetrically to balance total drag. Roll spoilers are deployed to balance rolling moment. The trim parameters are reported in Table 6.2 and the 1g static trimmed shape is

shown in Fig. 6.8.

Table 6.2: Trim parameters of X-HALE model without aerodynamic fairings

Parameters		Units
Speed	14	$\mathrm{m\,s^{-1}}$
Angle of attack	1.754	deg
Angle of sideslip	0.308	deg
Center Propeller (P0)	5956.2	rev/min
Left Propellers (P1,P3)	5893.71	rev/min
Right Propellers (P2,P4)	6018.64	rev/min
Elevators (T1–T4)	0.906	deg
Left roll spoiler (RS1)	0.0	deg
Right roll spoiler (RS2)	0.0430	deg
Tip deflection (W6)	14.7	% half span

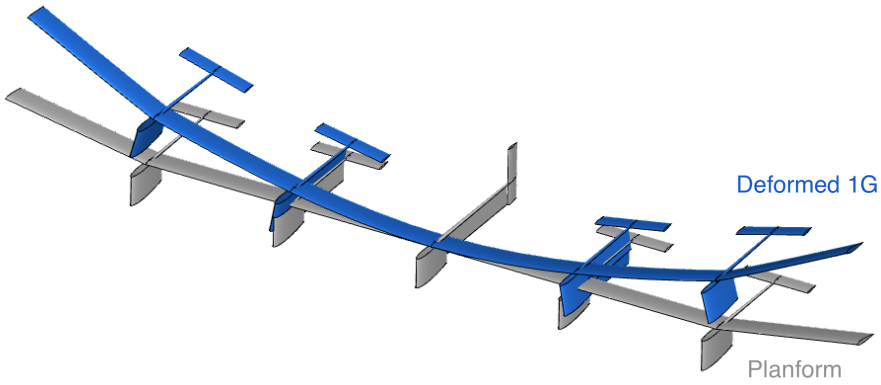

Figure 6.8: Deformed flight shape at $14\,\mathrm{m\,s^{-1}}$

6.2.3 Virtual Sensors

Virtual sensors corresponding to ATV-6B are modeled in UM/NAST. Table 6.3 summarizes the sensor locations and measured output available for control implementation.

Table 6.3: Virtual sensors on X-HALE UM/NAST FEM model

Sensor	Location	Sensor Type
MIDG	F0	INS
VN1	F1	IMU
VN2	F2	IMU
VN3	F3	IMU
VN4	F4	IMU
ML1 – ML6	W1	LED Wing Marker
ML7 – ML12	W3	LED Wing Marker
ML13 – ML18	W5	LED Wing Marker
MR1 – MR6	W2	LED Wing Marker
MR7 – MR12	W4	LED Wing Marker
MR13 – MR18	W6	LED Wing Marker

Note: LED are numbered front to back, inboard to outboard

6.2.4 Model Linearization

X-HALE is linearized about 1g steady level flight at $14\,\mathrm{m\,s^{-1}}$. System states q are detailed in Table 6.4 and control states u are detailed in Table 6.5. In usual state-space notation, dropping the tilde symbol for brevity:

$$q = \begin{bmatrix} \varepsilon & \dot{\varepsilon} & \beta & \zeta & P_B & \lambda \end{bmatrix}^T \tag{6.5}$$

$$u = \begin{bmatrix} \delta_{T1} & \delta_{T2} & \delta_{T3} & \delta_{T4} & \delta_{RS1} & \delta_{RS2} & \delta_{P0} & \delta_{P1} & \delta_{P2} & \delta_{P3} & \delta_{P4} \end{bmatrix}^T \tag{6.6}$$

$$y = \begin{bmatrix} y_{MIDG} & y_{VN1} & \cdots & y_{VN4} & y_{LM1} & \cdots & y_{LM18} & y_{RM1} & \cdots & y_{RM18} \end{bmatrix}^T \tag{6.7}$$

$$\dot{q} = Aq + Bu \tag{6.8}$$

$$y = Cq \tag{6.9}$$

where $A \in \mathbb{R}^{345\times345}$, $B \in \mathbb{R}^{345\times11}$, $C \in \mathbb{R}^{159\times345}$, $q \in \mathbb{R}^{345\times1}$, $u \in \mathbb{R}^{11\times1}$, and $y \in \mathbb{R}^{159\times1}$. Detailed numbering information is provided in Appendix C.3.

Table 6.4: Description of X-HALE system states in linearized A matrix

State	Symbol	Size	Units
Strain	ε	64	ext.: non-dim., curvature: m^{-1}
Strain rate	$\dot{\varepsilon}$	64	ext.: s^{-1}, curvature: $\mathrm{m}^{-1}\,\mathrm{s}^{-1}$
Body velocity and ang. rate	β	6	velocity: $\mathrm{m}\,\mathrm{s}^{-1}$, ang.: $\mathrm{rad}\,\mathrm{s}^{-1}$
Quaternions	ζ	4	non-dim.
Inertial position	P_B	3	m
Inflow	λ	204	$\mathrm{m}\,\mathrm{s}^{-1}$

Table 6.5: Description of X-HALE control states in linearized B matrix

Effector	Symbol	Actuation Range	Units
Elevator T1	δ_{T1}	$\left[-\frac{\pi}{4}, +\frac{\pi}{4}\right]$	rad
Elevator T2	δ_{T2}	$\left[-\frac{\pi}{4}, +\frac{\pi}{4}\right]$	rad
Elevator T3	δ_{T3}	$\left[-\frac{\pi}{4}, +\frac{\pi}{4}\right]$	rad
Elevator T4	δ_{T4}	$\left[-\frac{\pi}{4}, +\frac{\pi}{4}\right]$	rad
Roll Spoiler RS1	δ_{RS1}	$\left[0, \frac{\pi}{4}\right]$	rad
Roll Spoiler RS2	δ_{RS2}	$\left[0, \frac{\pi}{4}\right]$	rad
Propeller P0	δ_{P0}	$[0, 2000]$	rev/s
Propeller P1	δ_{P1}	$[0, 2000]$	rev/s
Propeller P2	δ_{P2}	$[0, 2000]$	rev/s
Propeller P3	δ_{P3}	$[0, 2000]$	rev/s
Propeller P4	δ_{P4}	$[0, 2000]$	rev/s

6.2.5 Validation of Model Linearization

The X-HALE model described in Chapter 5 is linearized about flight speed $14\,\mathrm{m\,s^{-1}}$ at altitude of 30 m. The virtual sensors are also linearized to provide a complete (A_d, B_d, C_d) description of the linearized plant and measured outputs. For comparison purposes, a nonlinear time marching simulation in UM/NAST is initiated from level trimmed flight. The elevators are actuated using a frequency sweep of 0.1 Hz to 3 Hz. Therefore, the response is predominantly in the longitudinal axis. Linearized plant is simulated using MATLAB `lsim` and compared with the nonlinear simulation.

Figure 6.9 shows that rigid body vertical velocity w and pitch rate p matches well with nonlinear simulation. The linearized response captures the initial drop in forward velocity v but exhibits deviations over the simulation horizon. Some discrepancies can be seen in the lateral states but the absolute magnitude are small. The linearized plant captures the response for wing root axial strain and bending curvatures well. This accuracy is important for designing controllers/observers for curvature limiting control.

Figure 6.10 shows a subset of the linearized and nonlinear sensor measurements. Since MIDG-II is mounted on F0, which is a rigid element attached to the body origin, the angular velocity states in β and the measured angular rates at the sensor should be identical, while the measured linear velocity is offset from linear velocity states in β by the moment arm multiplied by the angular rates in β. While X-HALE is pitching (ϕ_B), the wing flexes due to aerodynamic loading, manifesting as local varying roll angle (θ_S) and twist angle (ϕ_S) of the beam reference line. VN-100 measurements on F4 pod captures the orientation trends, but with some discrepancies as simulation time increases. The linearized response of the wing markers (on right wing tip W6) show good correlation with the nonlinear measurements.

In general, the correlation between the linearized measurements and nonlinear measurements follow closely the prediction capability of the linear plant (A_d, B_d).

Assuming systems states q are known, the accuracy of the linearization of sensor output matrix C_d is very good. The overall accuracy, however, depends on the state-space description (A_d, B_d, C_d).

6.3 Sensor Fusion Performance

UM/NAST numerical simulations are used to investigate the performance of the sensor fusion algorithms. X-HALE is chosen as the benchmark model as these algorithms will be eventually applied on ATV-6B flight data. The loads are applied at the tips of both X-HALE wings (Fig. 6.11) with three different load cases (Table 6.6). No gravity or aerodynamics loads are applied. The vehicle is clamped at the body center for this numerical study, therefore the true quaternion is known ($\zeta_{true} = [1, 0, 0, 0]$). Table 6.7 shows the different sensor combinations tested to assess their predictive performance.

Table 6.6: Load cases for sensor fusion evaluation

Case	Load Case
1	10 N, 1 Hz sine tip force in out-of-plane bending z
2	10 N m, 1 Hz sine tip moment in twist x
3	20 N, 1 Hz sine tip force in-plane bending y

Table 6.7: Tested sensor configurations

Configuration	Sensors Included
INS+IMU*	MIDG, and VN1–VN4
INS+Markers	MIDG, and ML1–MR18
INS+Markers+IMU*	MIDG, VN1–VN4, and ML1–MR18

Note: Yaw angle measurements from IMUs are discarded

The quality of the fit at each time step t_k is measured based on the residue e_ε

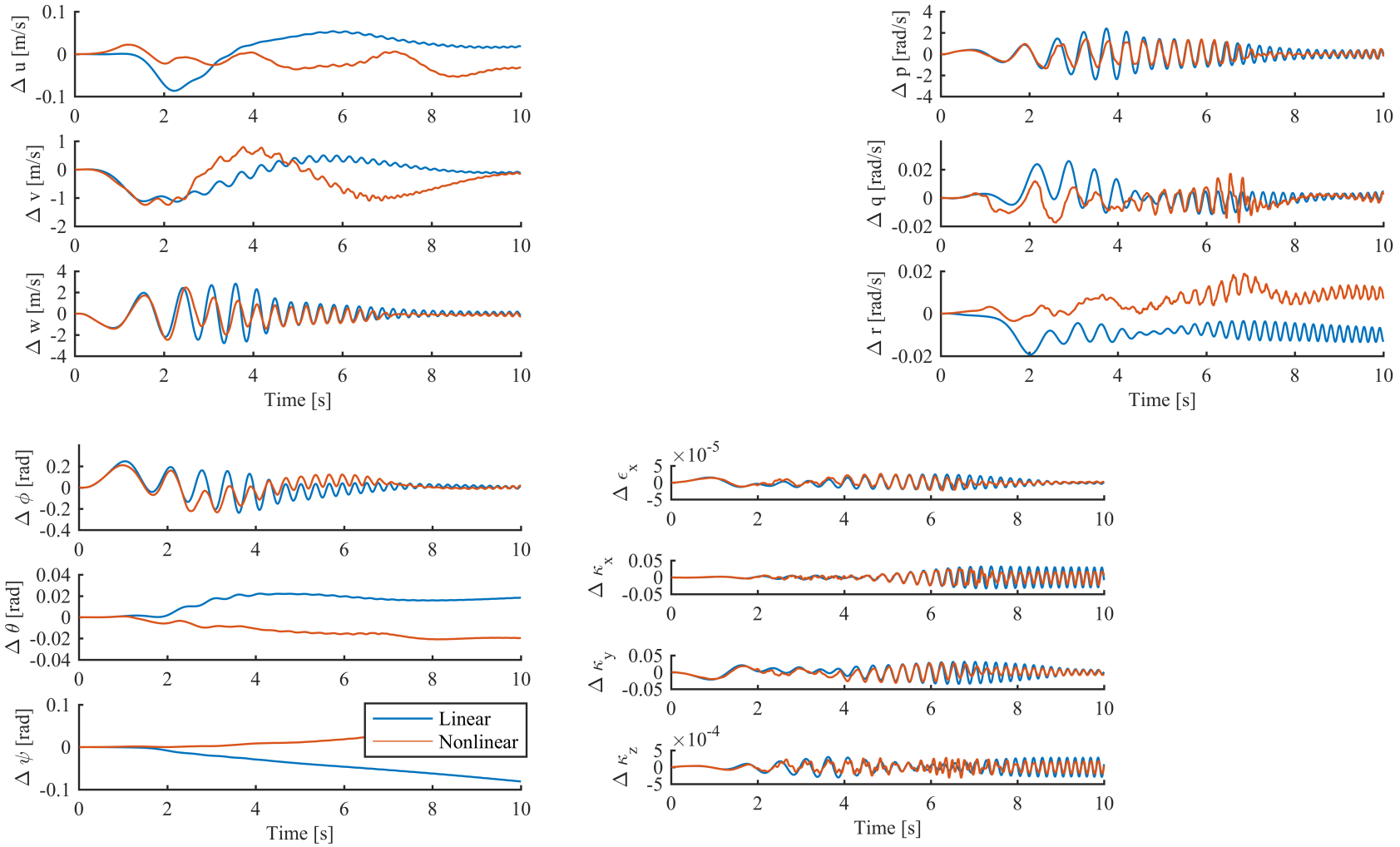

Figure 6.9: Time history comparison of linearized and nonlinear states for X-HALE elevator frequency sweep

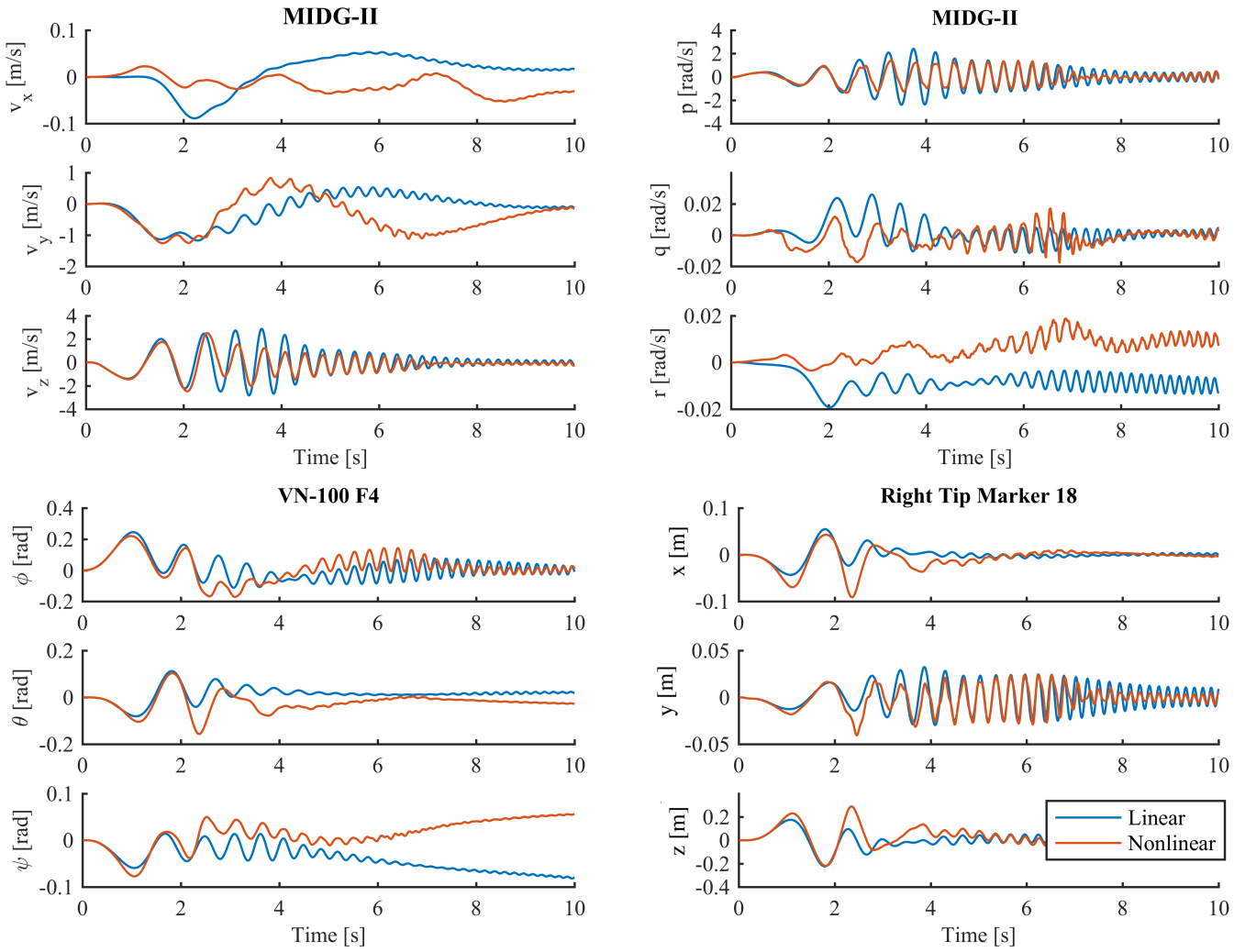

Figure 6.10: Time history comparison of linearized and nonlinear virtual sensor measurements for X-HALE elevator frequency sweep

defined by:

$$e_{\varepsilon}(t_k) = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{\kappa_x^{est}(t_k) - \kappa_x^{true}(t_k)}{\kappa^{true,max}} + \frac{\kappa_y^{est}(t_k) - \kappa_y^{true}(t_k)}{\kappa^{true,max}} + \frac{\kappa_z^{est}(t_k) - \kappa_z^{true}(t_k)}{\kappa^{true,max}}\right) \tag{6.10}$$

where $\kappa^{true,max}$ is the maximum curvature over the entire time history of the simulation, and N is the number of flexible elements to be estimated. For the X-HALE wing, there are $N = 8$ flexible elements for each left and right wing. The strain differences are normalized by the maximum value, making it the fractional fit error. To get a more physical interpretation of body attitude error, body attitude is solved directly in terms of Euler angles instead of quaternions. Thus, the body attitude error is defined as:

$$e_{att}(t_k) = \frac{1}{3}\left(\left[\phi_B^{est}(t_k) - \phi_B^{true}(t_k)\right] + \left[\theta_B^{est}(t_k) - \theta_B^{true}(t_k)\right] + \left[\psi_B^{est}(t_k) - \psi_B^{true}(t_k)\right]\right) \tag{6.11}$$

The average is then taken over the simulation duration to give strain and attitude residue metric. Note that INS+IMU combination results in an under-determined

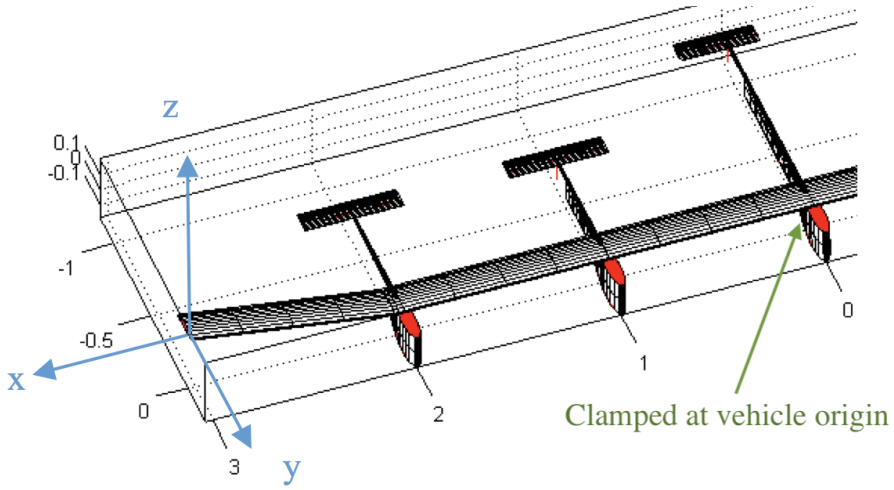

Figure 6.11: Load case geometry (shown for right wing tip)

system for Nonlinear Least Square (NLS) estimation. The solution is a least-norm solution for strains ε in Eq. (4.42). INS measurements are simply propagated to body Euler angle states. Similarly, elastic displacement and rigid body attitude are decoupled in INS+Markers combination. Therefore, the INS measurements are also passed through to estimated body Euler angle.

Artificial noise is added to sensor measurements to investigate the robustness of the sensor fusion algorithm. The noise levels (Table. 6.8) are chosen to be representative of the real system.

Table 6.8: Artificial noise level injected

Sensor Type	Noise Level
MIDG	2° measured angles
	$2\,^\circ\,\mathrm{s}^{-1}$ measured rate
	$0.5\,\mathrm{m\,s}^{-1}$ measured velocity
VN1–VN4	2° measured angles (pitch and roll)
	$2\,^\circ\,\mathrm{s}^{-1}$ measured rate
Markers*	1 mm measured displacement (nearest)
	9 mm measured displacement (furthest)

*Noise varies linearly with distance from camera

6.3.1 Nonlinear Least Square (NLS) Method

In this numerical study, yaw angle measurements from VN1 – VN4 are discarded and not used for estimation. Experimental characterization conducted as part of the full-scale test (Section 3.3) showed yaw measurements having large deviations from actual truth values. The extensional stiffness (K_{11}) of X-HALE wing is very high, therefore deformations are predominantly from the bending curvatures. Extensional strains of the elements ε_x are fixed at zero and not estimated for computational efficiency.

With no noise added, and giving equal weights to all measurements, unsurprisingly, INS+IMU+Markers combination which has the most number of redundant

measurements performs the best. The recovered strains and deformations are identical to that of the analytical solution. Figure 6.12 shows the recovered wing shape, computed from estimated strain, plotted against markers ML1–ML18. The wing markers appear right on top of the wing surface in their expected locations. Table 6.9 reports the strain residue for all the sensor combinations. INS+IMU has the highest residue and is poor at recovering wing in-plane bending deformation. This is expected as in-plane strain information are lost when yaw angle readings from VN1 – VN4 are discarded. Therefore, it is clear that INS+IMU combination with yaw measurement discarded is not sufficient for state estimation.

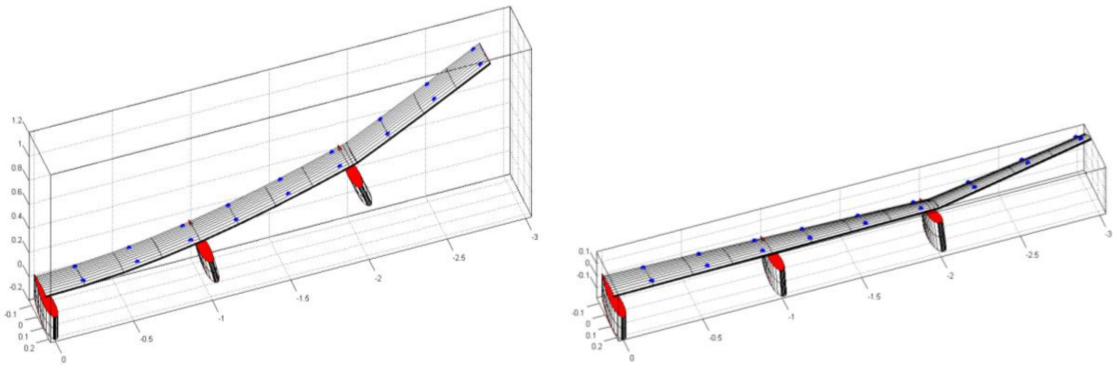

Figure 6.12: Good NLS performance for INS+IMU+Markers (measurements without noise)

Table 6.9: Strain residue e_ε for NLS with no measurement noise

Measurement	Case 1	Case 2	Case 3
INS+Markers	4.90×10^{-8}	1.91×10^{-8}	9.00×10^{-8}
INS+IMU	3.34×10^{-3}	2.09×10^{-3}	1.69×10^{-2}
INS+IMU+Markers	2.83×10^{-8}	1.40×10^{-8}	3.49×10^{-8}

Bold indicates best combination

In the presence of noise, the accuracy using INS+Markers combination drastically worsen. The recovered wing shape does not coincide with actual (noise-less) position of the maker plotted in Fig. 6.13. This suggests that strain recovery is very sensitive to noise in displacement measurements. Table 6.10 shows that combining all three

sensor types produces less strain residue than INS+SV or INS+IMU alone.

As explained previously, the Euler body angle residue will be identical for INS+Marker and INS+IMU since body attitude estimation is not affected by either IMU or marker measurements. Using the same noise seed for random noise generation across all three loading cases, the INS measurements will be identical and the residue reported is simply the mean of the generated noise profile.

Overall, it can be concluded that INS+IMU+Markers combination will provide the best estimation for NLS. Examining the span-wise error distribution in Fig. 6.14, the normalized error of each strain component seems to increase slightly from root to tip.

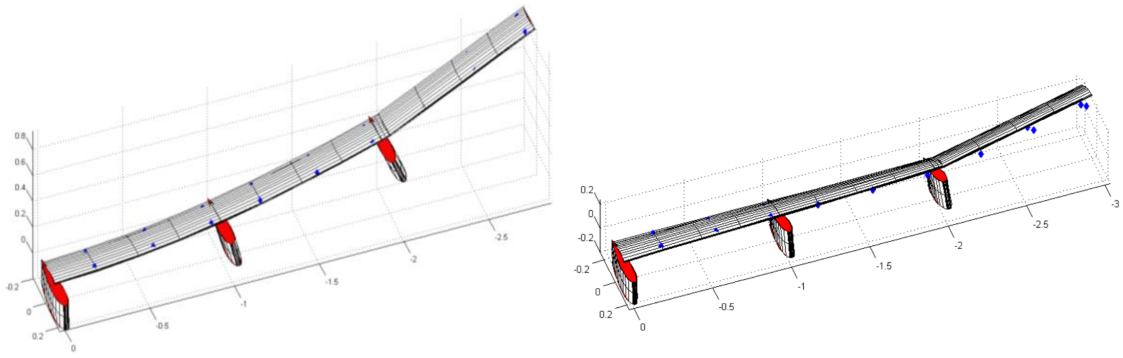

Figure 6.13: Degradation of NLS performance for INS+Markers (measurements with noise)

Table 6.10: Strain residue e_ε for NLS with measurement noise

Measurement	Case 1	Case 2	Case 3
INS+Markers	1.55×10^{-3}	2.59×10^{-3}	4.42×10^{-3}
INS+IMU	-3.77×10^{-3}	-5.78×10^{-4}	1.49×10^{-2}
INS+IMU+Markers	1.36×10^{-3}	2.23×10^{-3}	3.48×10^{-3}

Bold indicates best combination

The NLS algorithm typically converges in about 10 sub-iterations with numerical tolerance setting of 10^{-6}. The gradient of cost function is supplied to improve computation time. The number of iterations to convergence is also dependent on the

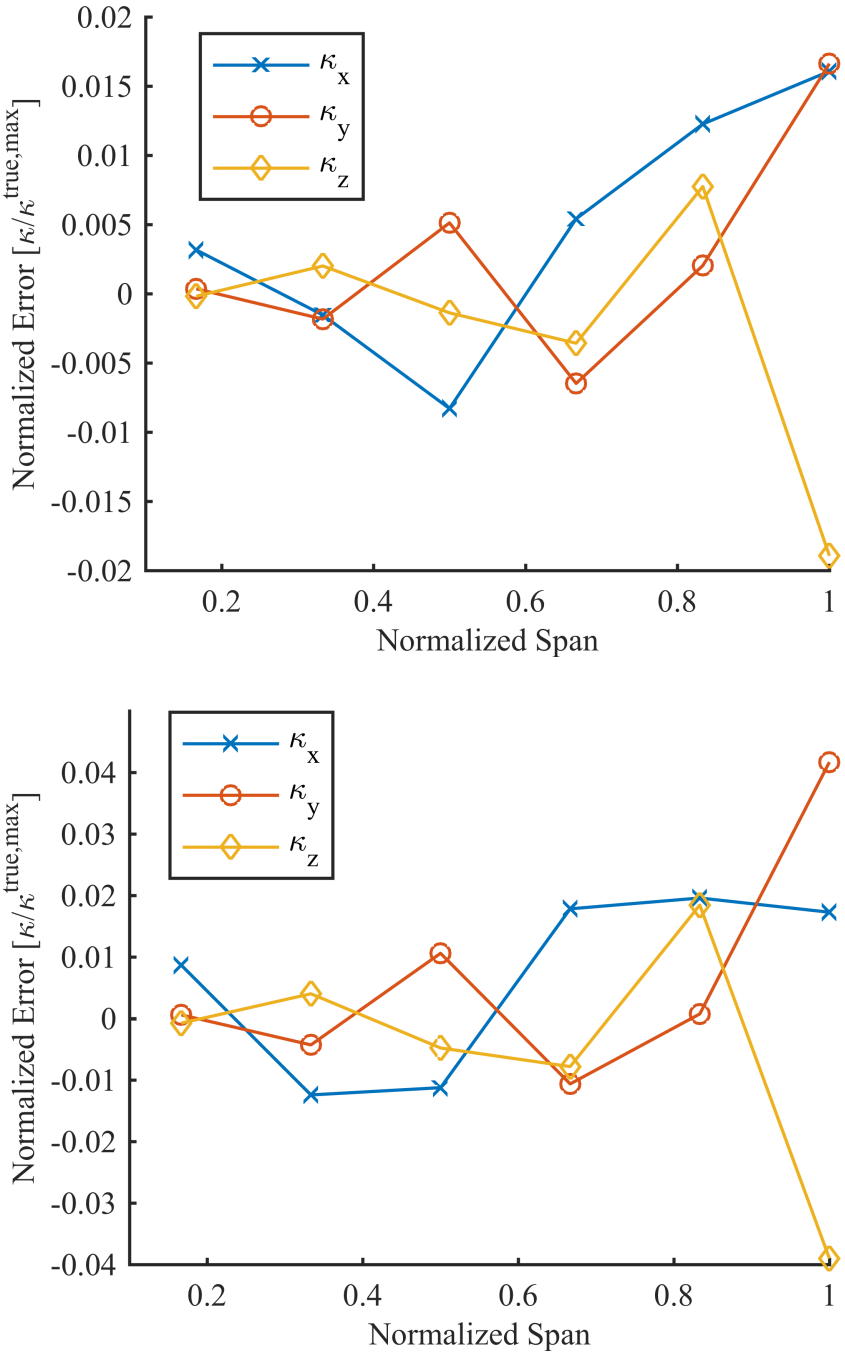

Figure 6.14: Span-wise error distribution for INS+Markers (measurements with noise) for load case 1 (top) and 2 (bottom)

Table 6.11: Euler angle residue e_{att} for NLS with measurement noise

Measurement	Case 1	Case 2	Case 3
INS+Markers	5.954×10^{-4}	5.954×10^{-4}	5.954×10^{-4}
INS+IMU	5.954×10^{-4}	5.954×10^{-4}	5.954×10^{-4}
INS+IMU+Markers	5.006×10^{-4}	5.957×10^{-4}	4.923×10^{-4}

Bold indicates best combination

initial supplied guess. If the time step between snapshots is sufficiently small, estimated strains from the previous time step should be used as the initial guess value as the strain value at the current step should not have differed much. This aids in significantly decreasing the required number of iterations. The time taken to solve each snapshot, using speedups described above, is approximately 0.5 s to 0.8 s on a desktop computer (Intel-Xeon 2.0 GHz) using MATLAB `lsqnonlin`.

6.3.2 Kalman Filter (KF)

Since no aerodynamic forces are applied, and there are no rigid body degrees of freedom (beam is clamped at the root), the inflow states λ and the rigid body states (β, ζ, P_B) are removed. The linearized plant used for this Kalman filter evaluation is truncated from the plant obtained in Section 6.2.4 and is written as:

$$q = \begin{bmatrix} \varepsilon & \dot{\varepsilon} \end{bmatrix}^T \tag{6.12}$$

$$u = \begin{bmatrix} u_l & u_r \end{bmatrix}^T \tag{6.13}$$

$$y = \begin{bmatrix} y_{MIDG} & y_{VN1} & \cdots & y_{VN4} & y_{LM1} & \cdots & y_{RM18} \end{bmatrix}^T \tag{6.14}$$

where u_l and u_r are the applied forces/moments at the left and right wing tip, respectively. For this problem, $q \in \mathbb{R}^{128\times1}$, $u \in \mathbb{R}^{2\times1}$ and $y \in \mathbb{R}^{144\times1}$. The sensor noise

covariance matrix is set to the variance of the sensor noise:

$$R = \begin{bmatrix} \sigma_{\phi}^2 & \cdots & 0 \\ 0 & \ddots & 0 \\ 0 & \cdots & \sigma_{MR18}^2 \end{bmatrix}_{144\times144} \tag{6.15}$$

The process covariance noise Q is tuned numerically to achieve best results. It is tricky to define an analytical form as this process noise is used to tune the mismatch between the linearized and nonlinear plant, i.e., it needs to account for higher order terms lost during the linearization process. For simplicity, the process noise is treated as uncertainties in the input u. From numerical simulations, this simple process noise covariance matrix gives good estimation results. The process noise is written as:

$$w_k = B_d I_{2\times2} \Delta u \tag{6.16}$$

with $\Delta u \sim \mathcal{N}(0, 1)$. Therefore,

$$Q = I_{2\times2} \tag{6.17}$$

Results in Tables 6.12 – 6.13 show that KF demonstrates comparable residues regardless of sensor combination types. This can be attributed to two factors. Firstly, a linearized model of the system response is available, so if the system is observable, the states can be reconstructed from any measurement data. Secondly, angular rates in three axes are measured by the outboard VN-100 gyroscopes. All information are retained unlike in the NLS formulation where yaw angle information is discarded. Figure 6.15 shows that, unlike NLS, there is no accumulation of error span-wise (root to tip). The time taken for one filter iteration is less than 1 ms as discrete KF computation only requires matrix operations without the need for sub-iterations.

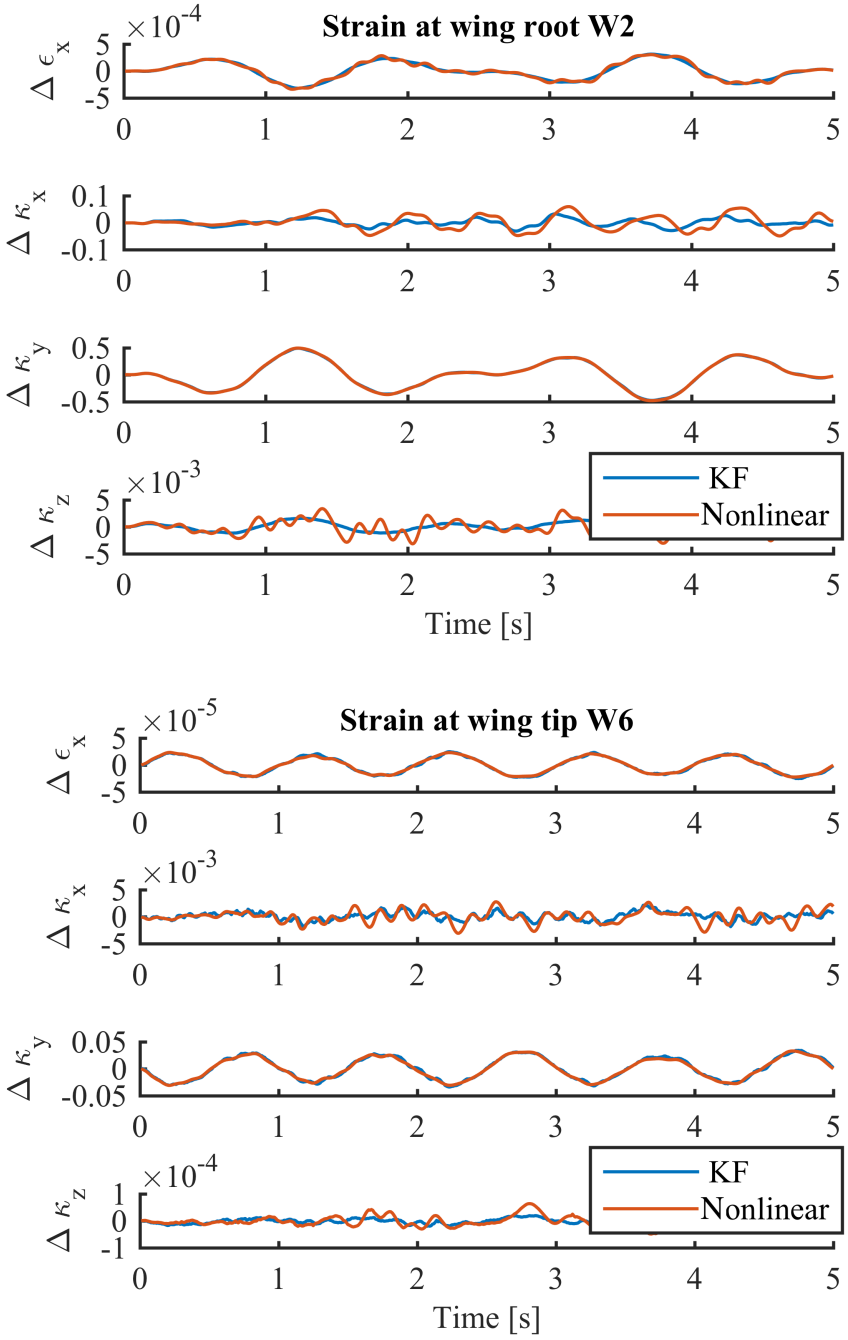

Figure 6.15: KF estimation of wing strain using Marker+IMU for case 1

Table 6.12: Strain residue for KF without measurement noise

Measurement	Case 1	Case 2	Case 3
Marker	-8.796×10^{-4}	-9.953×10^{-4}	-1.782×10^{-2}
IMU	-1.041×10^{-3}	-5.439×10^{-4}	-1.495×10^{-2}
Marker+IMU	-6.640×10^{-4}	-2.787×10^{-3}	-1.588×10^{-2}

Table 6.13: Strain residue for KF with measurement noise

Measurement	Case 1	Case 2	Case 3
Marker	-8.771×10^{-4}	-9.894×10^{-4}	-1.781×10^{-2}
IMU	-2.003×10^{-5}	1.835×10^{-3}	-1.265×10^{-2}
Marker+IMU	3.214×10^{-4}	-4.928×10^{-4}	-1.347×10^{-2}

6.3.3 Comparison of NLS and KF

Comparing Table 6.9 and Table 6.12, the strain residue is higher in the KF formulation compared to NLS when measurements are noiseless. This is due to the fact that NLS employs nonlinear relationship between strain and displacement while KF employs only a linearized relationship.

When noise is present in the measurements, the performance of NLS degrades significantly compared to noiseless case. On the other hand, measurement noise has far less impact for KF estimation. By proper tuning of the process noise Q and sensor noise covariance R matrices, knowledge of system dynamics from the embedded state-space model can be used to eliminate noise. However, this is both an advantage and disadvantage. To apply KF, the input forces have to be known or measured. This information may not be always available (e.g., VFA encountering an unknown wind gust in the atmosphere). The linearized plant also has to be obtained *a priori* and provide a good prediction of the actual response.

6.4 X-HALE Model Predictive Control

In this section, full-state feedback MPC is used for demonstrating maneuver load alleviation with both state and control constraints being enforced. Suppressing wing flexibility by imposing a large state weighting cost can reduce the deviation of the structural states from trimmed values, and achieve lower peak stress. However, it skews the control effort to minimizing the structural deformation even when none might be necessary (i.e., when deformations are within the allowable stress limits). This will degrade the tracking or regulation performance of other state variables. The goal of VFA control (typically) is not to transform the flexible aircraft to a rigid aircraft by suppressing all flexibility. By imposing state constraint instead of state suppression, MPC will only modify the control actions when state constraints are violated within the prediction horizon. Otherwise, the nominal controller performance is retained. Therefore, compared to state suppression method, state constraint MPC demonstrates superior performance while ensuring structural limits are respected.

6.4.1 Control Design

X-HALE control effector constraints (Table 6.5) are applied. X-HALE has low lateral control authority, and control saturation will happen during normal maneuvers. State constraints are also applied to the wing root curvature since the largest bending curvature during flight is experienced in this location. Moreover, since X-HALE has uniform wing construction, the point of maximum curvature directly relates to a point of highest bending stress, and therefore a point of possible structural failure.

Aerodynamic inflow states are not observable from any of the sensors implemented on X-HALE. Although a full state feedback control can be designed, without observability, the controller cannot be implemented outside of a numerical simulation environment where all the states are accessible. Examining the dynamic response

as well as the linearized response with and without inflow states, the discrepancies are minimal (Fig. 6.16). In addition, aerodynamic inflow states contributes a significant number of states to the overall plant dynamics (204 out of 345 total states). Therefore, for the control design, aerodynamic inflow states are removed. However, for accurate vehicle response, inflow states will still be used in UM/NAST nonlinear time marching simulation.

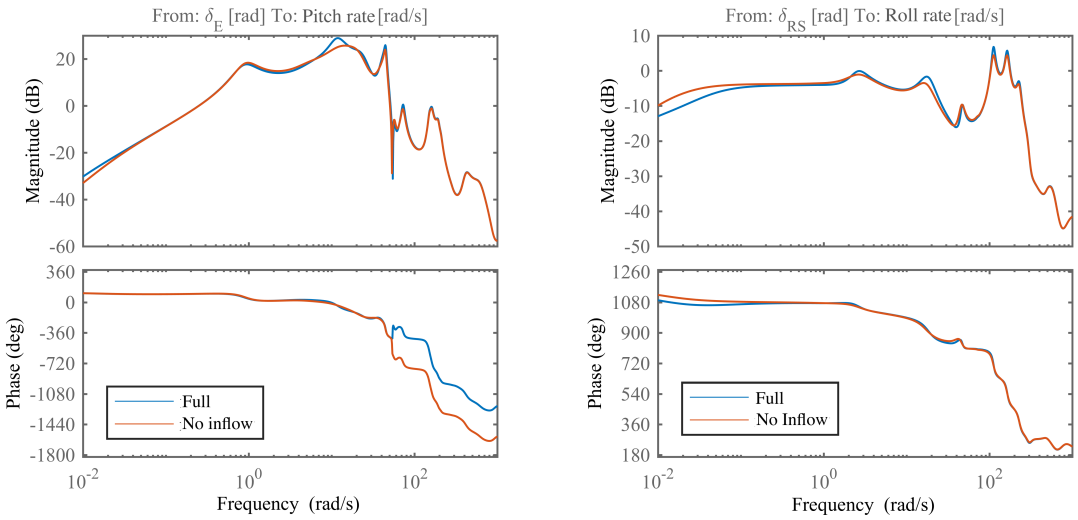

Figure 6.16: Differences with and without inflow states

MPC is formulated as a tracking problem. The tracked variables are pitch angle (θ_B) and roll angle (ϕ_B) reference commands. Yaw angle (ψ_B) is left unregulated. The rest of the states (e.g., strain, strain rate, and body velocities) are regulated. That is to say all structural strain ε and strain rate $\dot{\varepsilon}$ states are penalized with a small state cost. Similarly, rigid body linear and angular velocities β are penalized as well. The control effectors are grouped together into four control actuators as shown in Table 6.14. All four tail elevators are actuated together. Roll spoilers are actuated asymmetrically. Differential thrust uses both outboard propellers (P1,P3 vs P2,P4). Thrust actuates symmetrically for all propellers.

The augmented discrete time MPC control problem formulated as per Section 4.1.2 is defined with:

$$e_{\theta_B} = \theta_{cmd} - \theta_B \tag{6.18}$$

$$e_{\phi_B} = \phi_{cmd} - \phi_B \tag{6.19}$$

$$q \equiv \begin{bmatrix} \varepsilon & \dot{\varepsilon} & \beta & \phi_B & \theta_B & e_{\phi_B} & e_{\theta_B} \end{bmatrix}^T \tag{6.20}$$

$$u \equiv \begin{bmatrix} \delta_E & \delta_R & \delta_{RS} & \delta_T \end{bmatrix}^T \tag{6.21}$$

with:

$$A_{aug} = \begin{bmatrix} A_d & 0_{2\times 2} \\ [0_{2\times 136}, -\Delta t I_{2\times 2}] & I_{2\times 2} \end{bmatrix} \tag{6.22}$$

$$B_{aug} = \begin{bmatrix} B_d \\ 0_{2\times 4} \end{bmatrix} \tag{6.23}$$

$$B_r = \begin{bmatrix} 0_{136\times 2} \\ \Delta t I_{2\times 2} \end{bmatrix} \tag{6.24}$$

This system is open loop unstable. The control time step Δt is selected to be 0.02 s (compared to the simulation time step of 0.001 s). The prediction horizon N_p and constraint horizon N_c are set to 50 steps (equivalent to 1 s). The control horizon N_u

Table 6.14: MPC output variable mapping

X-HALE Control Surface	MPC Output Variable
$\delta_{T1} - \delta_{T4}$	δ_E
δ_{RS1}	$-\delta_{RS}$
δ_{RS2}	δ_{RS}
δ_{P0}	δ_T
δ_{P1}, δ_{P3}	$\delta_T + \delta_R$
δ_{P2}, δ_{P4}	$\delta_T - \delta_R$

Note: Negative roll spoiler commands are neglected

is set to 1. MPC state and control weighting matrix are set to:

$$Q = diag(I_{128\times128}, 1, 100, 1, 10, 10, 10, 1, 1, 50 \times 10^3, 50 \times 10^3) \tag{6.25}$$

$$R = diag(1, 1, 100, 1) \tag{6.26}$$

The state and control weight matrices are tuned using numerical simulations to achieve a rise time of approximately 1 s to 2 s without excessive control effort. Physical judgement also influences this selection. Small weights are placed on the strain ε and strain rate $\dot{\varepsilon}$ states to regulate but not to suppress structural flexibility. A large weight is placed on the rigid body forward velocity component in β to maintain flight speed in order to prevent aerodynamic stall. A moderate weights are placed on the rigid body angular rates in β to suppress attitude oscillations. Large weights are placed on the error states e_θ and e_ϕ to enforce zero tracking error.

Two maneuvers are examined in this study: The first is a commanded nose up maneuver of 0.1 rad or 5.73°. This approximates a pull up maneuver when changing altitude. The second maneuver is a commanded positive roll of 0.3 rad or 17.2°. This approximates a banking turn. No attempt at roll coordination was made in this study. During the maneuver, the wing flexes upwards due to increased aerodynamic loading. The bending curvature will peak, allowing the effectiveness of MPC to be evaluated.

A LQR controller with the same state and control weights is also be implemented in UM/NAST. This acts as the baseline control law for comparison. The LQR gain is computed from the Ricatti equation given in Eq. (4.9). Control saturation for LQR is applied externally to the control block in the main UM/NAST simulation code. Saturation is done to retain physical feasibility of the control action (e.g., roll spoilers are actuated within $[0, \frac{\pi}{4}]$).

The MPC problem is solved using `qpOASES` within UM/NAST. Four different configurations are tested. MPC1 solves the MPC problem but with arbitrarily relaxed

state constraints. Applied control constraints are identical to the LQR formulation. This converges to the LQR solution (with externally imposed control saturation) since state constraints are inactive. MPC2 solves the MPC problem with control and state constraints. This is done in `qpOASES` "default" mode. MPC3 is identical to MPC2, but solved in `qpOASES` "mpc" mode. The difference between MPC2 and MPC3 is in `qpOASES` internal numerical settings. MPC4 is similar to MPC3 with the addition of a user specified maximum iteration time. A function input argument in `qpOASES` solver activates this feature. Internally, a timer function checks the elapsed time and terminates active set computations prematurely when computation time is exceeded. In this case, this maximum time is set to 0.01 s.

At current solution time t_k, the state constraints are first evaluated at the next prediction time step t_{k+1} after one predicted control step u_k have been applied (Eq. 4.18). Therefore, for initial conditions outside the state constraint set $x_k \notin \mathbb{X}$ (due to external disturbances, or mismatched dynamics between supplied model and actual plant), if the next predicted state trajectory can be brought back to satisfy set $x_{k+1} \in \mathbb{X}$, `qpOASES` can return a solution. However, over the prediction horizon ($1 < i < N_p$), there is no guarantee that a feasible solution which satisfy both state constraints ($x_{k+i} \in \mathbb{X}$) and control constraints ($u_{k+i} \in \mathbb{U}$) always exist. In the current study, constraints are handled as handled as "hard" limits. `qpOASES` will return an error if no feasible solution is found. Infeasibility handling should be addressed in future studies.

6.4.2 Maneuver Load Alleviation

Both the MPC and LQR control laws are simulated using UM/NAST simulations. For the pitch maneuver, state constraints are applied to the wing root out-of-plane bending curvature (W1 and W2) such that $|\Delta\kappa_y| \leq 0.02$. This strain limit is selected to demonstrate the capabilities of MPC and not based on actual structural stress

requirements. Similarly, for the roll maneuver, the wing root strain constraints are selected to maintain $|\Delta\kappa_y| \leq 0.05$. Strain factor is defined as the maximum strain experienced at the wing root during the entire maneuver normalized by the maximum permissible strain. Therefore, a strain factor less than one indicates the maximum experienced strain is below maximum permissible. In the case of MPC2 – MPC4, this means that the state constraints should be respected. The control execution time is profiled within UM/NAST C++ environment and the mean and standard deviation of the time taken are recorded. This measure of computational performance is important for hardware implementation.

Table 6.15 shows that without state constraints but with control constraints, MPC1 and LQR indeed shows identical response. The strain factor is greater than one in both pitch (1.684) and roll (1.364) maneuvers since they are not constrained in the control formulation. Results of MPC2 and MPC3, which implement state constraints, show strain factor very close to one, illustrating that constraints are active and mostly respected. A reduction in the root out-of-plane bending curvature of 68% (pitch) and 29% (roll) are demonstrated. The adherence to curvature constraints comes at the expense of controller tracking performance as seen in Figs. 6.17 – 6.18. Examining the pitch up maneuver, MPC actually commanded the vehicle to pitch down momentarily around 0.5 s to 1 s to relieve root bending curvature. For the roll maneuver, the roll response is slowed around 0.5 s to 2 s. In both cases, the over-shoot due to the MPC controller is worse than the LQR case. Examining the control actions, there are fast adjustments of control effectors to keep the system within bounds, which is not seen in the LQR controller.

Table 6.16 summarizes the execution time for each controller type on a Macbook Pro laptop (Intel-i5 2.6 GHz). ATV-6B has a significantly lower computation speed (800 MHz Athena-II and 2.1 GHz QM-770), and further timing studies are required for real time implementation. LQR is extremely fast $\mathcal{O}(10^{-5})$ seconds as it only

requires matrix multiplication. The control and state trajectory of MPC1 is identical to LQR except that quadratic optimization is performed online. Even without active state constraints, the computation time of MPC1 is $\mathcal{O}(10^{-1})$ seconds, illustrating the computational disadvantage of MPC over non-optimization based control design. With active state constraints, the computational time taken by MPC2 is comparable to MPC1. This suggests that active set solution methodology is efficient at solving the constrained QP problem. By adjusting internal `qpOASES` solver settings, a speedup greater than 10 times is achieved by MPC3 over MPC2. Using the "mpc" option, the mean execution time is less than 0.014 s, which is less than the control sample frequency 50 Hz used for this numerical study. However, spikes in computational time is comparatively large (about 3 times as long in the worst case scenario) and not apparent by examining the standard deviation as shown in Fig. 6.19. Naturally, higher computation time corresponds to regions where state constraints are active.

MPC4 is an attempt to smoothen out the computational spikes by enforcing early termination in order to achieve real time control. Theoretically, early termination results in sub-optimal response, and by extension, constraints are not strictly enforced even though the controller does attempt to reduce to maximum excursion of the wing bending curvature compared to LQR case. Interestingly, by forcing early termination, the mean solution time actually increased. On further investigation, early solution termination resulted in infeasibility of the homotopy at subsequent solution step. `qpOASES` had to perform a cold-start in order to recompute homotopy for a feasible solution. Cold-starting and reconstructing the homotopy is computationally expensive compared to hot-restarting from a previous solution. This resulted in longer solution time as seen in the correlation between solution execution time and cold-start indicator (Fig. 6.20).

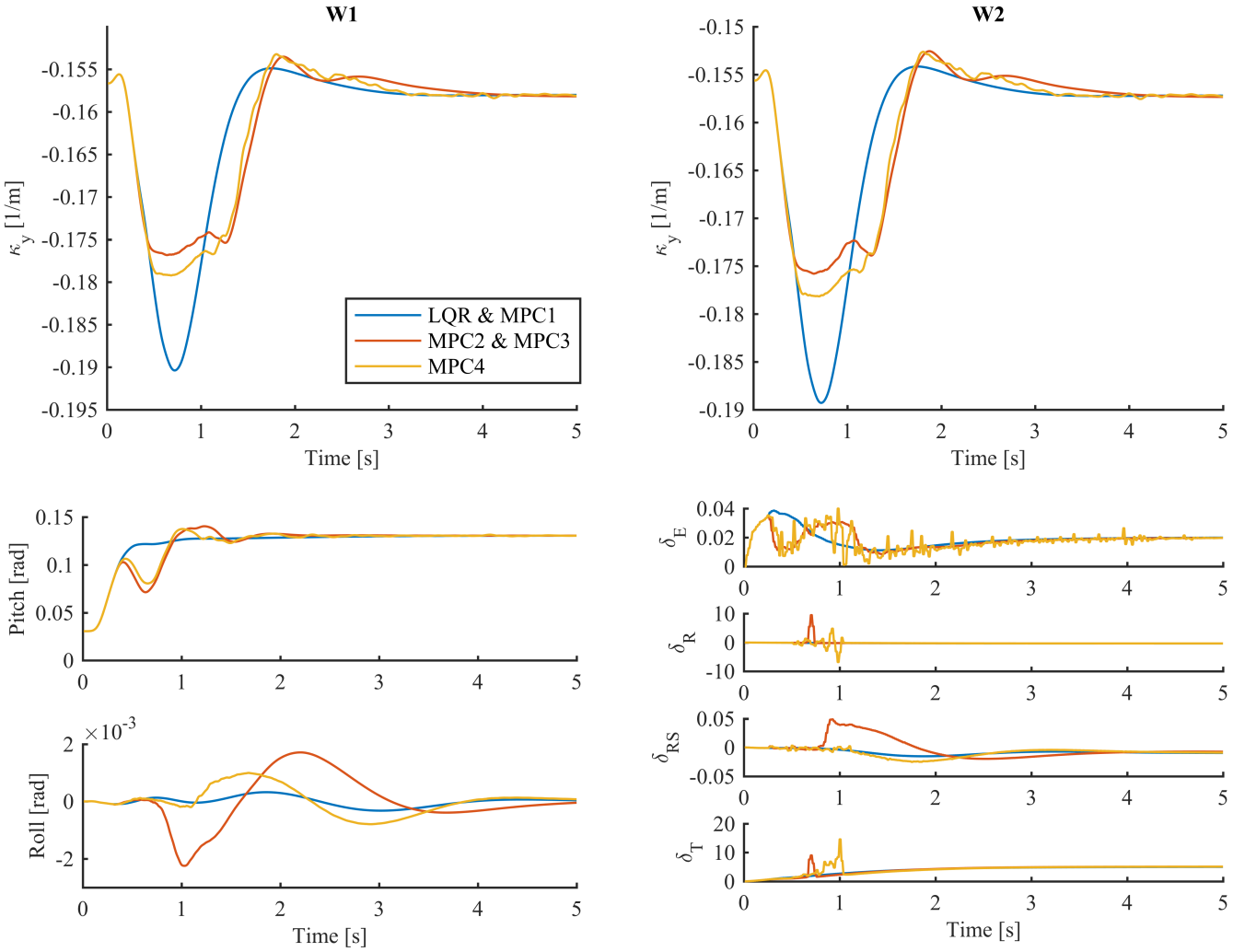

Figure 6.17: Pitch maneuver comparison of different controllers

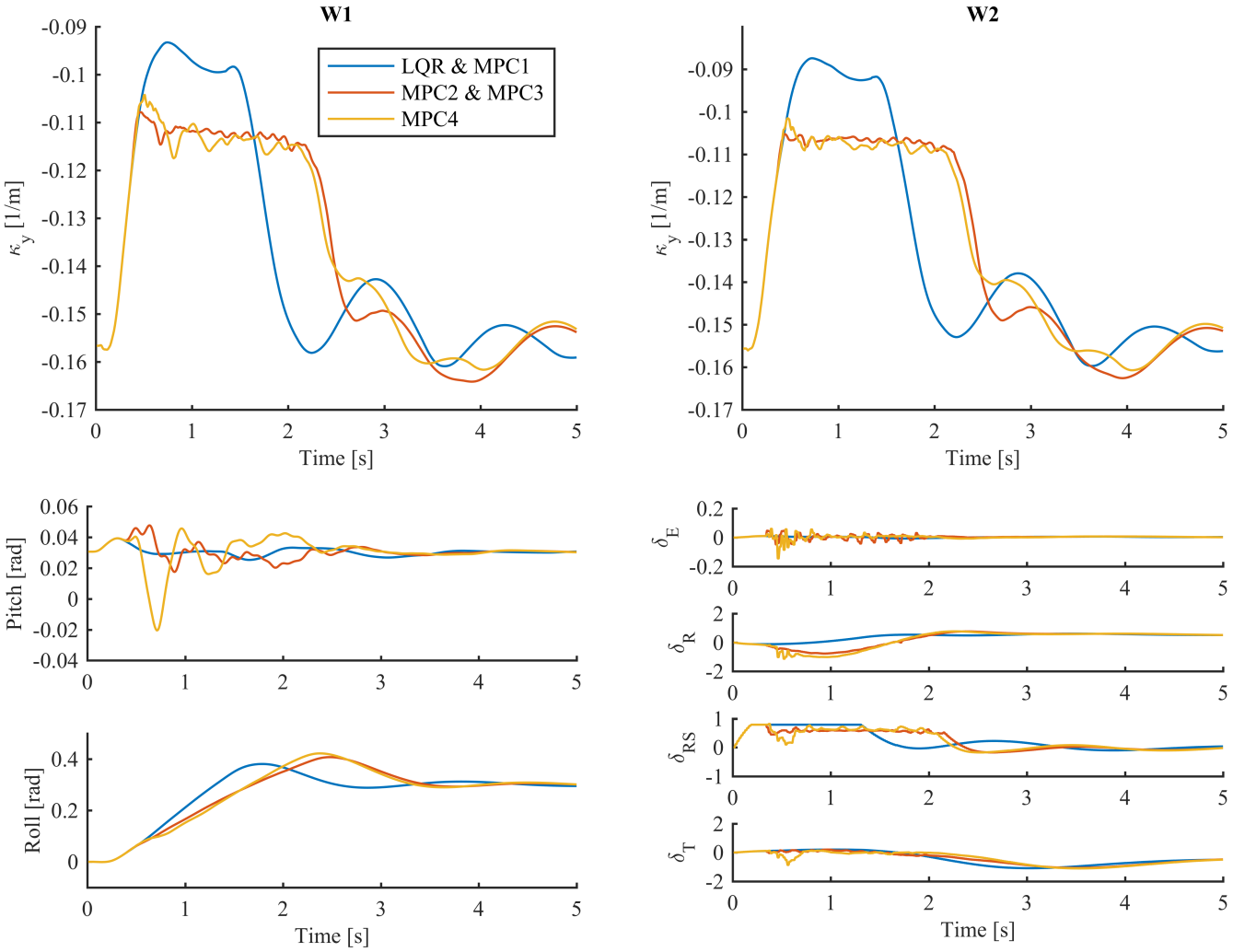

Figure 6.18: Roll maneuver comparison of different controllers

Table 6.15: Strain factor at wing right root for different controllers

Maneuvers	LQR	MPC1	MPC2	MPC3	MPC4
Pitch	1.687	1.6847	1.009	1.009	1.128
Roll	1.268	1.264	0.9761	0.9761	1.048

Table 6.16: Solution timing for different controllers (units: seconds)

Maneuvers	LQR		MPC1		MPC2		MPC3		MPC4	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
Pitch	2.81×10^{-5}	4.03×10^{-6}	0.0993	0.0026	0.1249	0.0134	0.0132	0.0051	0.0149	0.0070
Roll	3.03×10^{-5}	6.82×10^{-6}	0.1131	0.0100	0.1083	0.0116	0.0052	0.0033	0.0053	0.0035

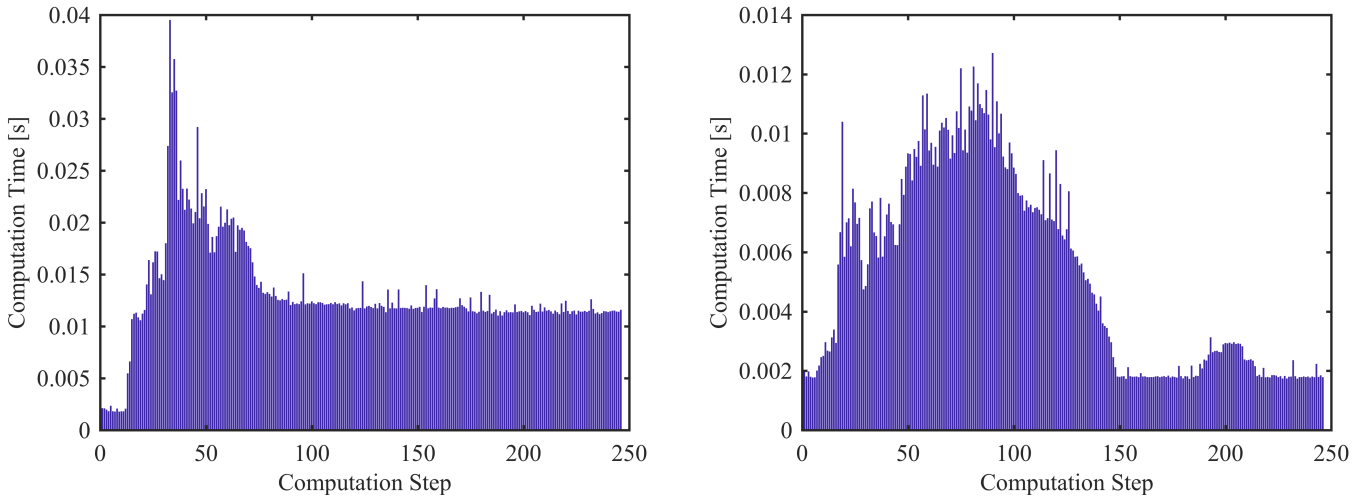

Figure 6.19: Computation time for MPC3 for pitch (left) and roll (right)

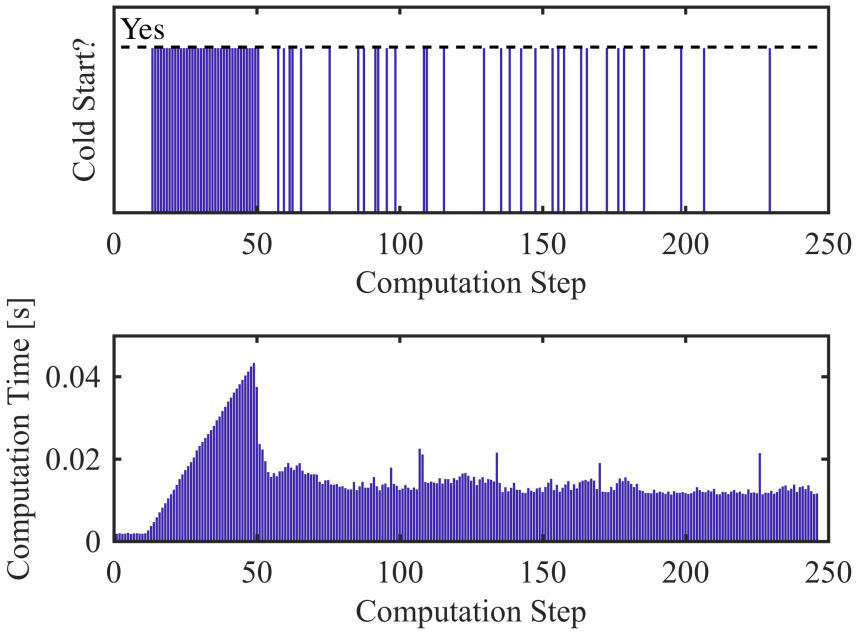

Figure 6.20: Computation time for MPC4 pitch maneuver with cold restart plot

CHAPTER 7

Experimental Results

Experimental flight tests using RRV-6B at Chelsea Proving Grounds in 2017 are presented in this chapter (Fig. 7.1). Section 7.1 presents the flight test design methodology including selection of excitation signal. In Section 7.2, non-parametric frequency domain identification and parametric time domain identification are applied to the collected flight data. Section 7.3 details the design of a stabilizing autopilot using the identified plant. This autopilot performs stabilization and set-point tracking of the pitch and roll angle, as well as regularization of the yaw rate. Before this controller is deployed on RRV-6B, a closed-loop numerical study is carried out in UM/NAST. Finally, the designed controller is implemented on the Pixhawk and flight tested. Section 7.4 presents the closed-loop flight performance and evaluation.

Figure 7.1: RRV-6B at Chelsea Proving Grounds

7.1 System Identification Flight Test Design

This section presents the system identification flight tests and the considerations behind the excitation signal selection to ensure good identification results.

7.1.1 Flight Test Procedure

The experimental flight test procedure can be divided into distinct segments:

1. Take-off and climb to target altitude;
2. Start test segment and perform system identification when ready;
3. End test segment;
4. Reorientation for next flight test point in the research segment;
5. Repeat items 2 and 4, until maximum flight time is reached;
6. Landing and recovery.

Every flight begins with a safety review on the current operational conditions (e.g., wind speed, wind gust, air temperature, and airspace clearance). Once clearance is obtained, the pilot will take off and climb to test altitude. When desired altitude is achieved, the pilot will fly a race track pattern, striving to perform steady level trimmed flight in the straight research segments. Once telemetry operators determine that trimmed conditions are achieved, the pilot is instructed to engage signal injection for system identification maneuver. In practice, it is extremely hard for the pilot to achieve steady level flight under manual control. Hence, the restriction is relaxed to allow some limited amplitude oscillations. Based on airspace limitation, as well as the natural sensitivity of the aircraft, the hands-off flight time of each straight segment is limited to about 10 seconds. Whenever possible, the dynamics are allowed to evolve naturally, unless prohibited by safety or attitude concerns. Once the current

test point is completed, the pilot disengages the signal injection and continues on the race track pattern. This procedure is repeated until flight termination. Once the aircraft has landed, data logs are downloaded from the Pixhawk and Electronic Speed Controller (ESC)s. The aircraft is inspected for structural integrity and prepared for the next flight.

7.1.2 System Identification Signal Selection

The pilot is instructed not to contaminate channels which are not currently being identified. This allows Single Input Single Output (SISO) identification to be carried out. There are several constraints in the design of the frequency sweep. Firstly, given that Pixhawk data is recorded at 100 Hz, Nyquist sampling theorem dictates that theoretical maximum frequency of the recoverable signal is 50 Hz. Tischler [28] noted that a more practical limit is about 20 Hz. Secondly, as the hands-off time is limited to 10 seconds, dynamics with frequencies below 0.1 Hz cannot be identified. Lastly, the closed-loop bandwidth of the Hitec servos is about 4 Hz or 25 rad/s. The bandwidth of propeller response is even lower at 1.4 Hz or 8.8 rad/s. Hence, the excitation signal is designed to be varied from 0.1 Hz (lowest possible) to 3.0 Hz.

Using the baseline UM/NAST model, one can obtain some insights into the dynamic characteristics of the X-HALE. Figure 7.2 highlights the identifiable frequency range (in bold lines) based on the UM/NAST linearization and the selected excitation frequency range. For elevator-to-pitch rate and differential-thrust-to-yaw rate response, the dominant poles are well captured. However, for spoiler-to-roll rate response, there are high frequency dominant poles outside the identifiable range. There are also differences between the fully flexible vehicle and rigidized vehicle. In the latter case, the vehicle flexible states are frozen at static trim values. Even in this relatively low frequency range, the flexible vehicle shows different responses compared to the rigidized vehicle.

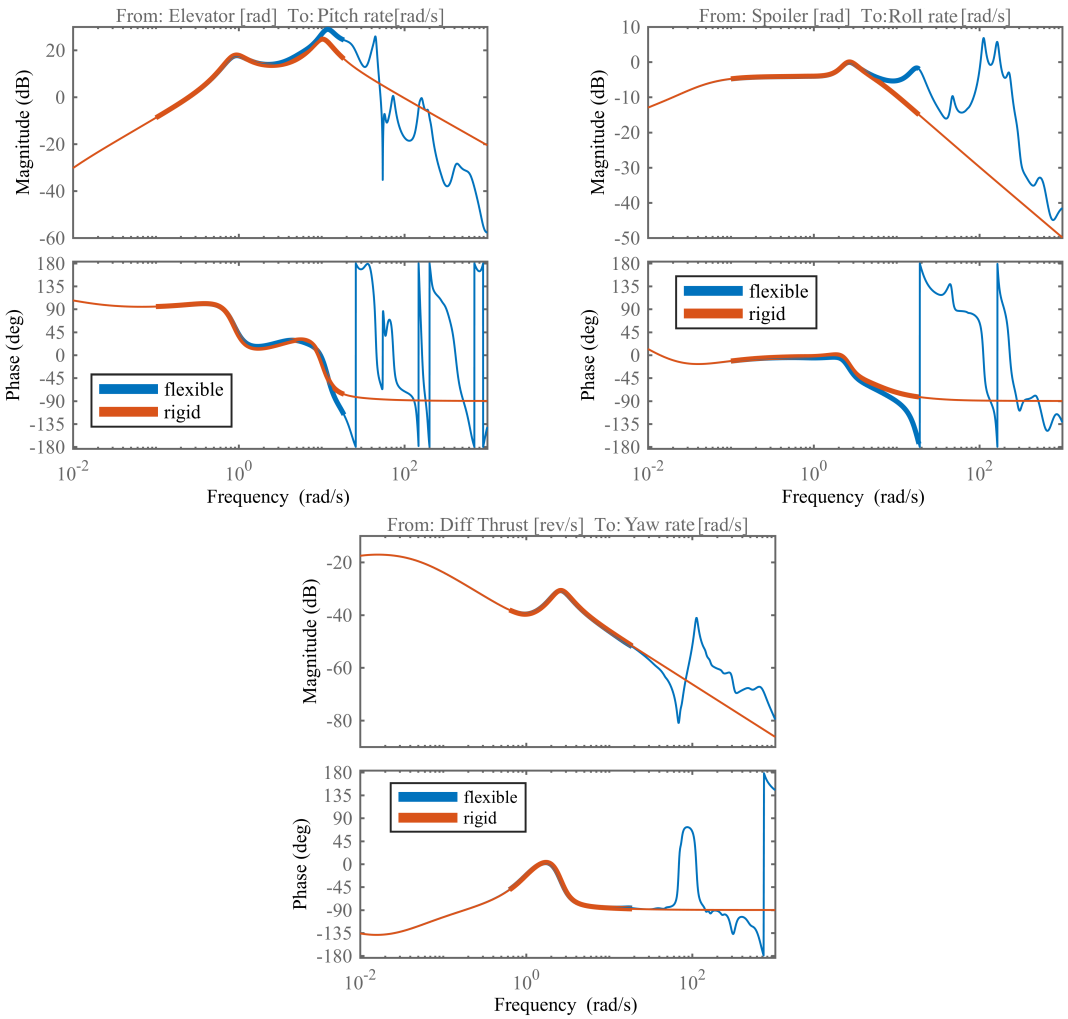

Figure 7.2: System response from UM/NAST linearized simulation, 0.1 Hz to 3.0 Hz in bold

There are two conflicting design considerations for excitation amplitude. Firstly, the angular rate response should be large to achieve a high Signal-to-Noise Ratio (SNR) ratio. On the other hand, the deviation from trim should be kept small to avoid nonlinearities. In addition, from safety considerations, large excursions from the trim condition may render the aircraft unrecoverable.

To establish the noise floor of the Pixhawk IMU, the RRV-6B is first characterized on the ground. The vehicle is suspended by supporting the main wings, and ensuring no part of RRV-6B is in contact with the ground. Next, acceleration and angular rate data are recorded while the motors (with the propellers attached) are set to

different throttle levels. This is done to simulate vibrations imparted by the spinning propeller and the impinging airflow on the aircraft structure during flight. Figure 7.3 established that for throttle in the RRV-6B operational range (80% to 100%), the noise amplitude is less than 2 °/s for angular rate and 0.6 m s^{-2} for acceleration.

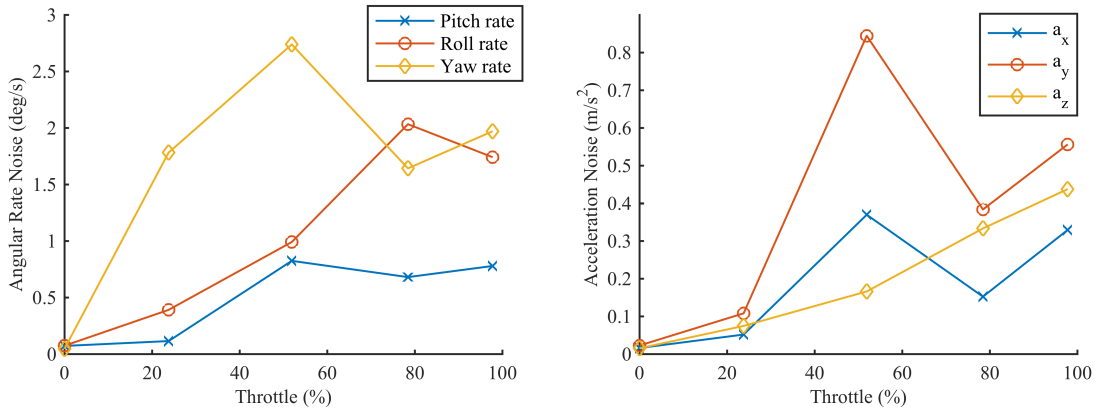

Figure 7.3: Noise floor characterization of Pixhawk IMU

Tischler [28] recommends SNR > 3, ideally SNR > 6, for best identification results. Therefore, an angular rate response of at least 6 °/s, ideally at least 12 °/s is required based on the noise level of the Pixhawk. Using nonlinear simulation as a guide, the following amplitudes are chosen: for the pitch axis, elevator sweep amplitude of 5° is used; for the roll axis, roll spoiler sweep amplitude of 20° is used; for the yaw axis, full differential thrust of 25% throttle (corresponding to 26.7 rev/s) is used. Note that the control authority in roll and yaw axis is very limited compared to traditional aircraft. This is due to the large lateral moment of inertia and the small control area (roll spoilers) or actuation range (differential thrust). This results in difficulties in increasing SNR. Three SISO models are identified from the flight tests. They are:

1. Elevator command (rad) to pitch rate (rad/s): $H_{\delta_{e,cmd} \to q}$
2. Roll spoiler command (rad) to roll rate (rad/s): $H_{\delta_{s,cmd} \to p}$
3. Differential thrust command (rev/s) to yaw rate (rad/s): $H_{\delta_{r,cmd} \to r}$

7.2 System Identification Experimental Flight

Figure 7.4 shows three sample flight segments of pitch, roll, and yaw excitation. In each case, the frequency sweep signal is injected into one channel and the rest of the controls are held constant. The pitch dynamic response is very large and closely tracks the pitch excitation. Even at higher frequencies (towards the end of the flight segment), a strong pitch rate signal of about 10 rad/s is observed. However, the dynamic response for lateral channels are much smaller than theoretical predictions. Although some response can still be seen, they are small in magnitude (<10 °/s), below the ideal SNR, and barely above the minimum SNR requirement. While the amplitude of the roll spoiler excitation can be increased, differential thrust is already at its maximum amplitude. This serves to highlight the low control authority in the lateral channels. This adds to difficulties in obtaining good system identification due to SNR issues. In addition, due to the natural instability of the RRV-6B, Fig. 7.4 shows residual low frequency roll oscillations at the start of the test segment, which the pilot was unable to cancel. A high-pass filter with bandpass of 0.25 Hz was used to filter out these oscillations during the identification process.

7.2.1 System Identification Results

Figures 7.5 – 7.7 show the Bode and coherence plot for parametric state-space identified model (blue), UM/NAST model (red), and non-parametric frequency domain identified model (yellow). The parametric state-space identified model is presented in equivalent discrete time transfer function form in Eqs. (7.1) – (7.4).

$$H_{\delta_{e,cmd}\to q}(z) = \frac{0.347z^3 - 1.144z^2 + 1.266z - 0.463}{z^4 - 3.414z^3 + 4.400z^2 - 2.545z + 0.560} \tag{7.1}$$

$$H_{\delta_{s,cmd}\to p}(z) = \frac{0.534z^6 - 3.154z^5 + 7.790z^4 - 10.305z^3 + 7.701z^2 - 3.083z + 0.516}{z^7 - 5.186z^6 + 14.588z^5 - 20.484z^4 + 17.421z^3 - 8.991z^2 + 2.613z - 0.330} \tag{7.2}$$

$$H_{\delta_{r,cmd}\to r}(z) = \frac{0.0468z^2 - 0.105z + 0.063}{z^3 - 2.728z^2 - 2.475z - 0.747} \tag{7.3}$$

$$T_s = 0.01\,\mathrm{s} \tag{7.4}$$

For the elevator-to-pitch rate response, all three models show good correlation from 1 rad/s to 25 rad/s. However, both experimentally identified models do not capture the 40 dB per decade slope at very low frequencies (outside the signal excitation frequencies) exhibited by the UM/NAST model. The peak at 10 rad/s from UM/NAST prediction is also not present in the flight data. The phase matches very well for all three models below 10 rad/s. Finally, input-output linearity is seen from

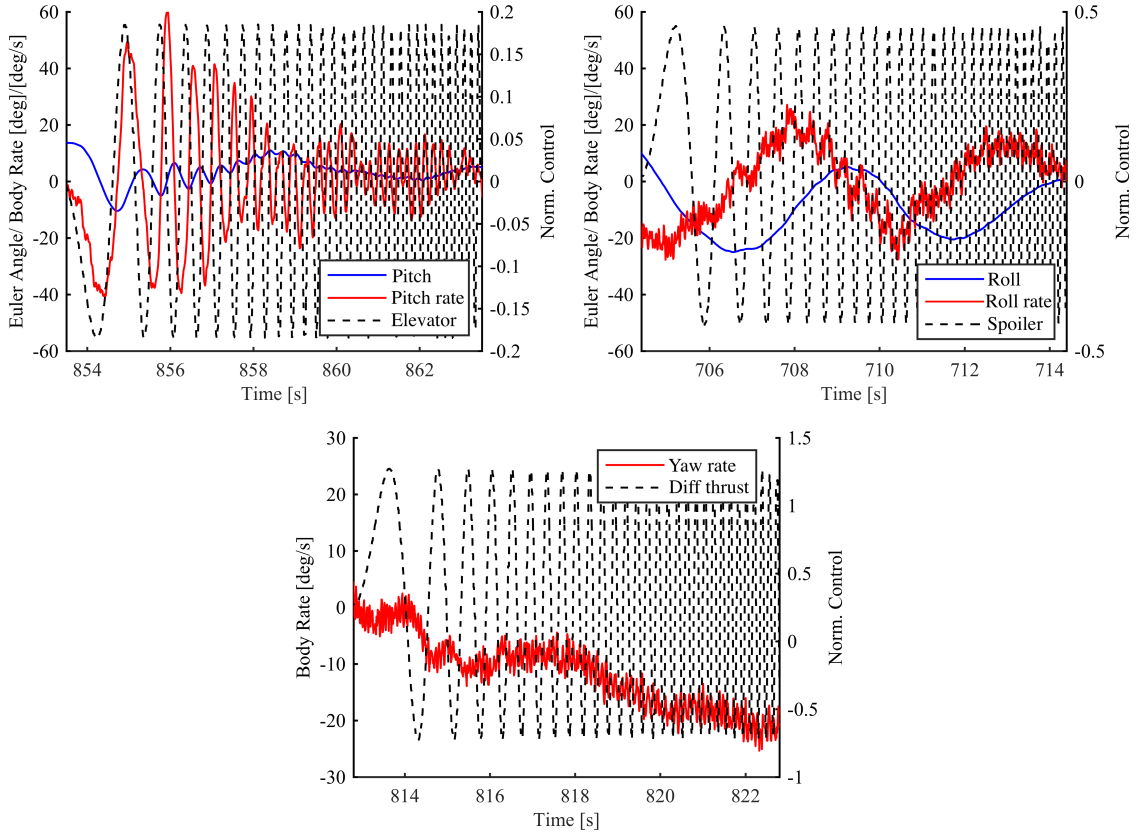

Figure 7.4: Vehicle response to injected excitation signal

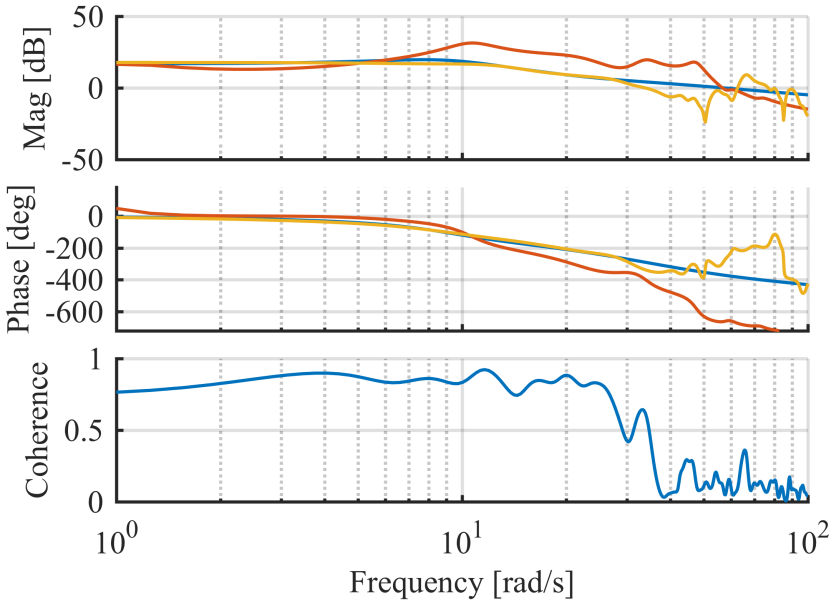

Figure 7.5: Bode plot of pitch rate models, identified state-space (blue), UM/NAST (red), and identified frequency domain (yellow) models

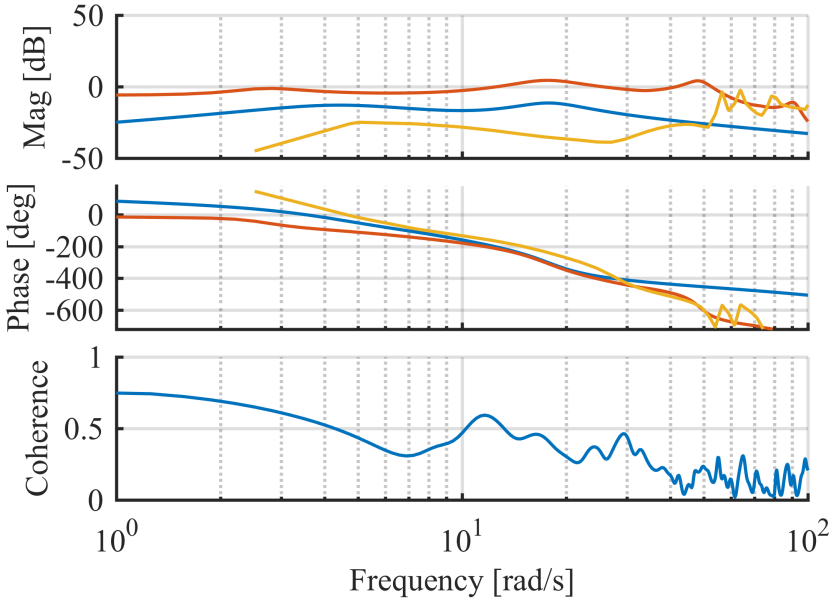

Figure 7.6: Bode plot of roll rate plants, identified state-space (blue), UM/NAST (red), and identified frequency domain (yellow) models

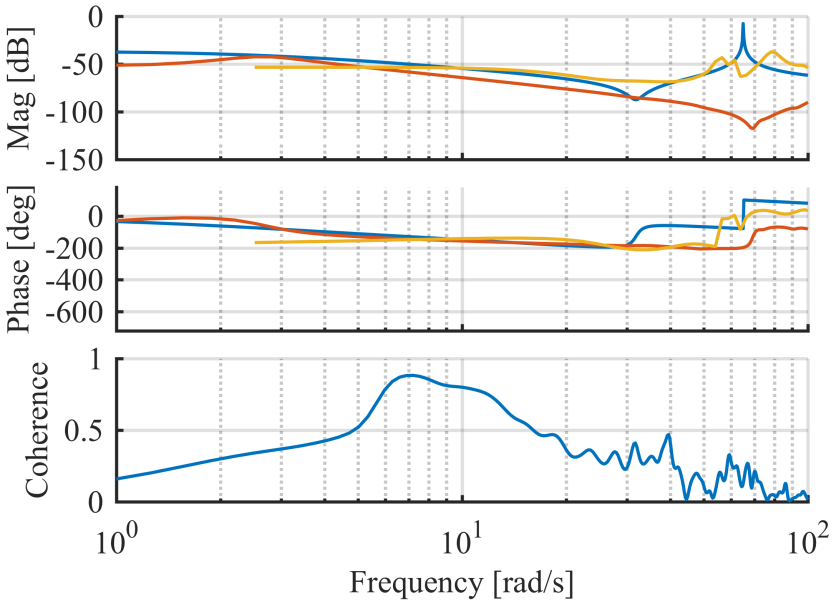

Figure 7.7: Bode plot of yaw rate plants, identified state-space (blue), UM/NAST (red), and identified frequency domain (yellow) models

the high coherence value ($\gamma > 0.8$).

For the spoiler-to-roll rate response, the coherence is generally low except for a narrow range of 10 rad/s to 15 rad/s. Identification improves at frequencies below 4 rad/s. Even in this range, the coherence is less than ideal ($0.5 < \gamma < 0.7$). The analytical model drastically over-predicts the effectiveness of the roll spoilers as seen from the magnitude plot. Also, the identified state-space model does not match the frequency points of the plant identified in frequency domain. This suggest either poor identification due to low SNR or strong nonlinearities giving different results.

For differential-thrust-to-yaw rate response, there is good coherence $\gamma > 0.8$ from 5 rad/s to 15 rad/s. The control authority of the propeller is slightly higher in flight compared to theoretical predictions, but both shows approximately -40 dB per decade roll off from 3 rad/s to 30 rad/s.

The differences between analytical and identified model may be attributed to the following reasons. Firstly, there exist some discrepancies between mass and inertia of

the actual vehicle and the numerical model in UM/NAST. Although, major structural and electronics components are placed using Computer Aided Design (CAD) software, miscellaneous components (e.g., wires, screws, and nuts) are not. The UM/NAST model also has not been tuned with Ground Vibration Test (GVT) to ensure a consistent mass, inertia, and stiffness representation.

The aerodynamics of the wings and tails are obtained numerically from XFOIL. The effects of propeller wake impinging on the wings and tails are not modeled in UM/NAST. Chadha et al. [109] showed that there is great variability in airfoil lift and drag behind a propeller slipstream. This has been recently confirmed that wing-tail, wing-fin, and wing-pod interference effects are also not accounted for [110, 111].

The low coherence in roll suggests this response is predominantly nonlinear. Firstly, roll spoilers are inherently nonlinear devices. Moreover, the roll spoiler is a flat thin carbon fiber plate attached to a servo at its center. The construction is relatively flexible and will likely deform under dynamic pressure, no longer acting as a rigid flat plate. This reduces the overall effectiveness compared to theoretical predictions, and more consistent with flight observations. Lastly, it cannot be ruled out that low SNR ratio degraded the identification results.

7.2.2 Time Domain Verification

The "acceptability" of the identified model is ultimately related to its ability to predict the dynamic response when subjected to control input. For the following time domain validation, a different set of flight data (not used in the estimation) is used.

Theil Inequality Coefficient (TIC) is used to assess the model prediction accuracy:

$$\text{TIC} = \frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(y_{meas} - y_{pred}\right)^2}}{\sqrt{\frac{1}{N}\sum_{i=1}^{N} y_{meas}^2} + \sqrt{\frac{1}{N}\sum_{i=1}^{N} y_{pred}^2}} \tag{7.5}$$

where y_{meas} is the measured output and y_{pred} is the model prediction. A value of TIC = 0 corresponds to model with perfect prediction capability while TIC = 1 corresponds to no predictive capability. Jateganonkar et al. [22] recommend as a guideline a TIC $\leq 0.25 - 0.3$. A 10-step prediction horizon is used to compute the predicted output y_{pred} for TIC computation. This selected horizon (0.1 s) is longer than the control sample time (0.02 s) and should prove sufficient to evaluate the accuracy of the identification.

Residual analysis is also employed to better understand the variance between predicted and measured response. The cross-correlation between the residual e and control u is given by:

$$\hat{R}_{eu}(\tau) = \frac{1}{N}\sum_{k=1}^{N} e(t_k)\, u(t_k - \tau) \tag{7.6}$$

where $e(t_k)$ is the prediction error at time instant t_k, and given by:

$$e(t_k) = y_{meas}(t_k) - y_{pred}(t_k) \tag{7.7}$$

The auto-correlation of the error is given by:

$$\hat{R}_{ee}(\tau) = \frac{1}{N}\sum_{k=1}^{N} e(t_k)\, e(t_k - \tau) \tag{7.8}$$

The auto-correlation $\hat{R}_{ee}$ should be small for $\tau \neq 0$, else the response can be better predicted by using past output measurements. Cross-correlation $\hat{R}_{eu}$ should also be small, otherwise, past control input history can be used to better predict response. Both indicate deficiencies in the identified model. Figure 7.8 shows the time domain prediction and the residual correlation. Identified pitch rate model has TIC = 0.203, indicating very good identification. Roll rate and yaw rate models have moderate TIC of 0.4 to 0.5, respectively, although, the lower frequency dynamics are well captured. Recall that previously, parametric system identification is weighted to regions with

higher coherence corresponding to low and moderate frequencies (4 rad/s to 15 rad/s). High frequency dynamics that are not fitted in the estimation process drives up TIC. All three axes have best autocorrelation at zero lag ($\tau = 0$), and the cross correlation are mostly within 99% confidence interval.

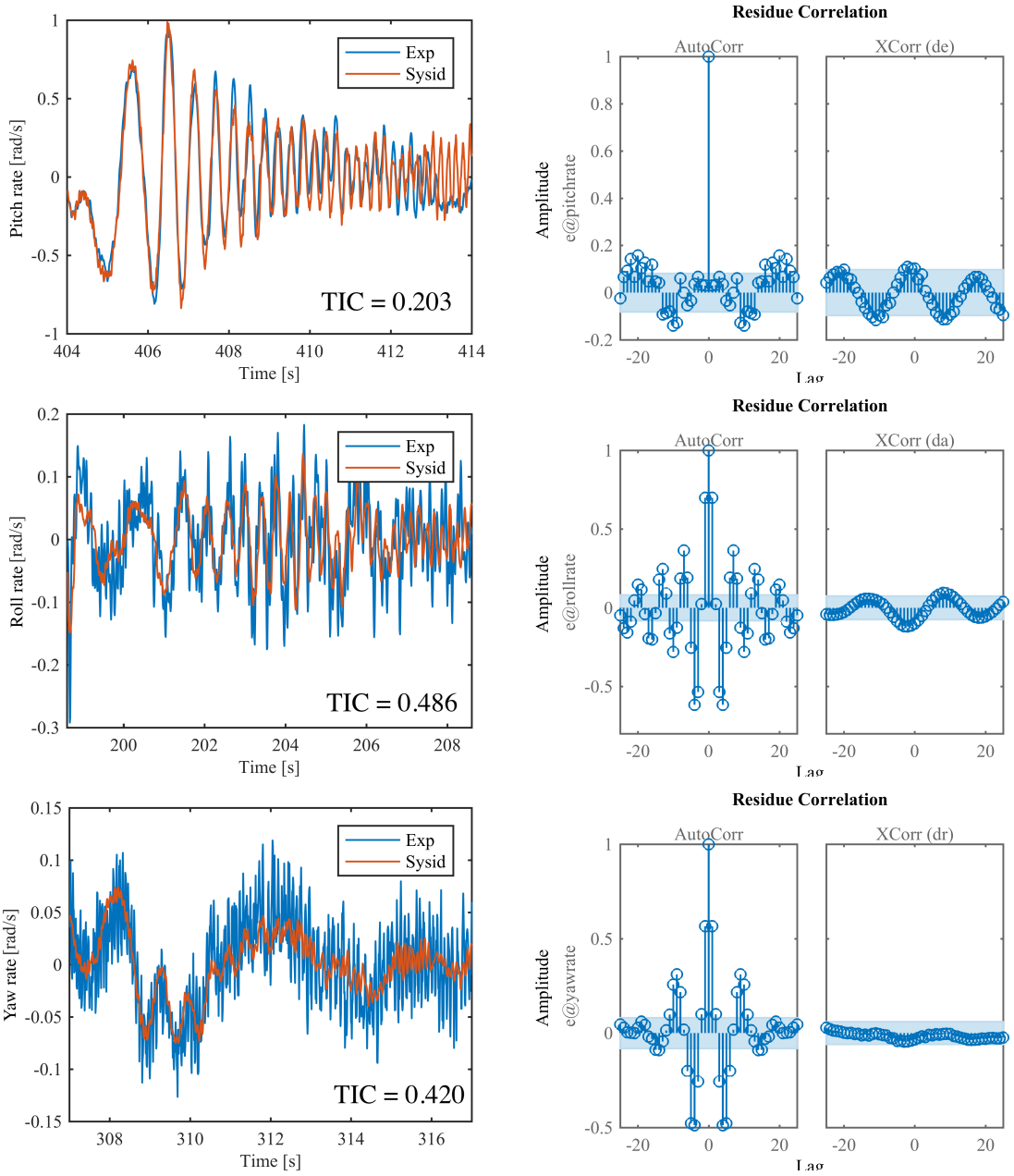

Figure 7.8: 10-step prediction and residual analysis of predicted and measured responses (Exp: measured experimental results, Sysid: simulated results with identified plant)

7.3 Stability Augmentation System

From previous open-loop flights, X-HALE exhibits an unstable Dutch-roll type oscillation, making the aircraft hard to control in a turn or approach for landing. A stability augmentation autopilot is designed using the identified model to improve its handling qualities.

7.3.1 Autopilot Design

A decoupled design strategy is employed. Each channel is designed independently using SISO methods, with the control allocation as given in Table 7.1. This control allocation is selected after initial study using UM/NAST numerical simulation with the RRV-6B model.

The design objective is to obtain gain margin of at least 6 dB and a phase margin of at least 60°. The inner loop crossover frequency is selected to be 3 rad/s while the outer loop crossover frequency is selected to be 1 rad/s. These crossover frequencies are selected to be well below the servo bandwidth. In addition, they give good performance (time constant $\tau \approx 2$ s) without being overly aggressive. This is confirmed in nonlinear simulations in UM/NAST.

Table 7.1: Control allocation for RRV-6B Pixhawk

Controlled Variable	Control Channel	Effector
Pitch angle	$\delta_{e,cmd}$	T1 – T4
Roll angle	$\delta_{s,cmd}$	RS1 vs RS2
Yaw rate	$\delta_{r,cmd}$	(P1,P3) vs (P2,P4)
Air Speed*	$\delta_{t,cmd}$	P0 – P5

*not controlled, direct pilot command pass through

7.3.1.1 Pitch Autopilot

Figure 7.9 shows the pitch loop control architecture. For the outer pitch angle control loop:

$$\dot{\theta}_{cmd} = K_{p,\theta} \left(\theta_{cmd} - \theta_{meas} \right) \tag{7.9}$$

where $K_{p,\theta}$ is the pitch proportional gain, $\dot{\theta}_{cmd}$ is the setpoint command to the inner rate loop, θ is the pitch angle in radians, the subscript cmd and $meas$ corresponds to the pilot input command and measured quantity, respectively. A saturation check is done to ensure the rate setpoint command is within user defined limits. The conversion from commanded Euler angle rate $\dot{\theta}_{cmd}$ to body fixed angular rate command q_{cmd} is given by:

$$q_{cmd} = (\cos \phi_{meas}) \dot{\theta}_{cmd} + (\sin \phi_{meas} \cos \theta_{meas}) \dot{\psi}_{cmd} \tag{7.10}$$

If the vehicle is in a bank, the lift vector is rotated, therefore it must generate additional lift to balance the weight in order to maintain level flight. From kinematics [112], the required offset is:

$$q_{cmd,offset} = \frac{g}{V_T} \tan \phi_{cmd} \sin \phi_{cmd} \tag{7.11}$$

where g is the gravity vector ($9.81\,\mathrm{m\,s^{-2}}$), V_T is the airspeed, and ϕ_{cmd} is the commanded roll angle. For the inner rate loop, accumulated pitch rate error e at time instant k is computed using forward Euler method:

$$e_k = e_{k-1} + \left(q_{cmd,k} - q_{meas,k} \right) \Delta t_k \tag{7.12}$$

where Δt_k is the elapsed time from the previous execution of the autopilot. The commanded servo output of the inner loop is given by:

$$\delta_{e,cmd} = K_{P,q}\, q_{err} + K_{I,q}\, e_q \tag{7.13}$$

A final saturation check is carried out to prevent controller saturation. The integral action is also suspended if the total accumulated error exceeds an user defined threshold. This is done to prevent integral windup.

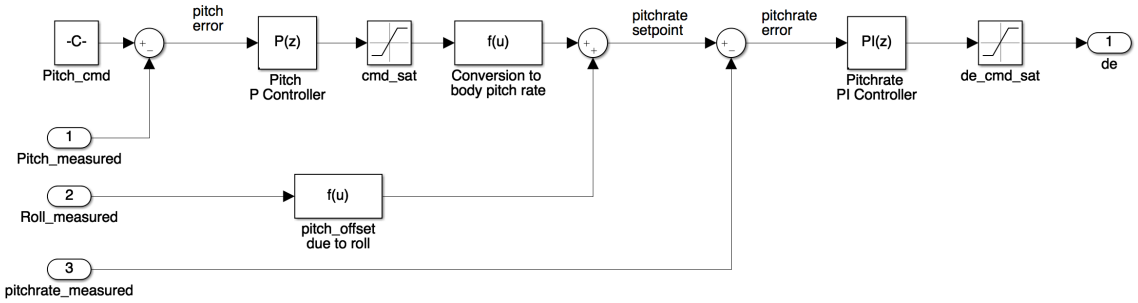

Figure 7.9: Pitch autopilot architecture

To design the pitch rate feedback loop, consider the Open Loop Transfer Function (OLTF):

$$H_{q,oltf}(z) = H_{\delta_e,cmd\to q}(z)\left(K_{P,q} + K_{I,q}\frac{T_s}{z-1}\right) \tag{7.14}$$

By careful selection of pitch rate gains $K_{P,q}$ and $K_{I,q}$, the control design objective of $GM \geq 6\,\text{dB}$ and $PM \geq 60^\circ$ with $\omega_{cr} = 3.0\,\text{rad/s}$ can be achieved. The closed-loop pitch rate transfer function $H_{q,cl}$ can be written as:

$$H_{q,cl} = \frac{H_{q,oltf}}{1 + H_{q,oltf}} \tag{7.15}$$

For the outer pitch feedback loop, the OLTF can be written as:

$$H_{\theta,oltf}(z) = K_{p,\theta} H_{q,cl}\left(\frac{T_s}{z-1}\right) \tag{7.16}$$

By careful selection of pitch gain $K_{P,\theta}$, the control design objective with $\omega_{cr} = 1.0\,\text{rad/s}$ can be achieved. Using MATLAB® `pidtune`, the pitch gains are designed and tabulated in Table 7.2. The closed-loop step response of pitch and pitch rate are shown in Fig. 7.10.

Table 7.2: Autopilot gain values and characteristics

	Crossover Freq (rad/s)	Gain Margin (dB)	Phase Margin (deg)	K_P	K_I
Pitch rate	3.0	6.57	67.8	0.2	0.004
Pitch	1.0	14.5	69.3	1.02	–
Roll rate	3.0	5.4	60.0	1.19	6.25
Roll	1.0	13.6	72.7	0.99	–
Yaw rate	3.0	13.0	60.0	108	–

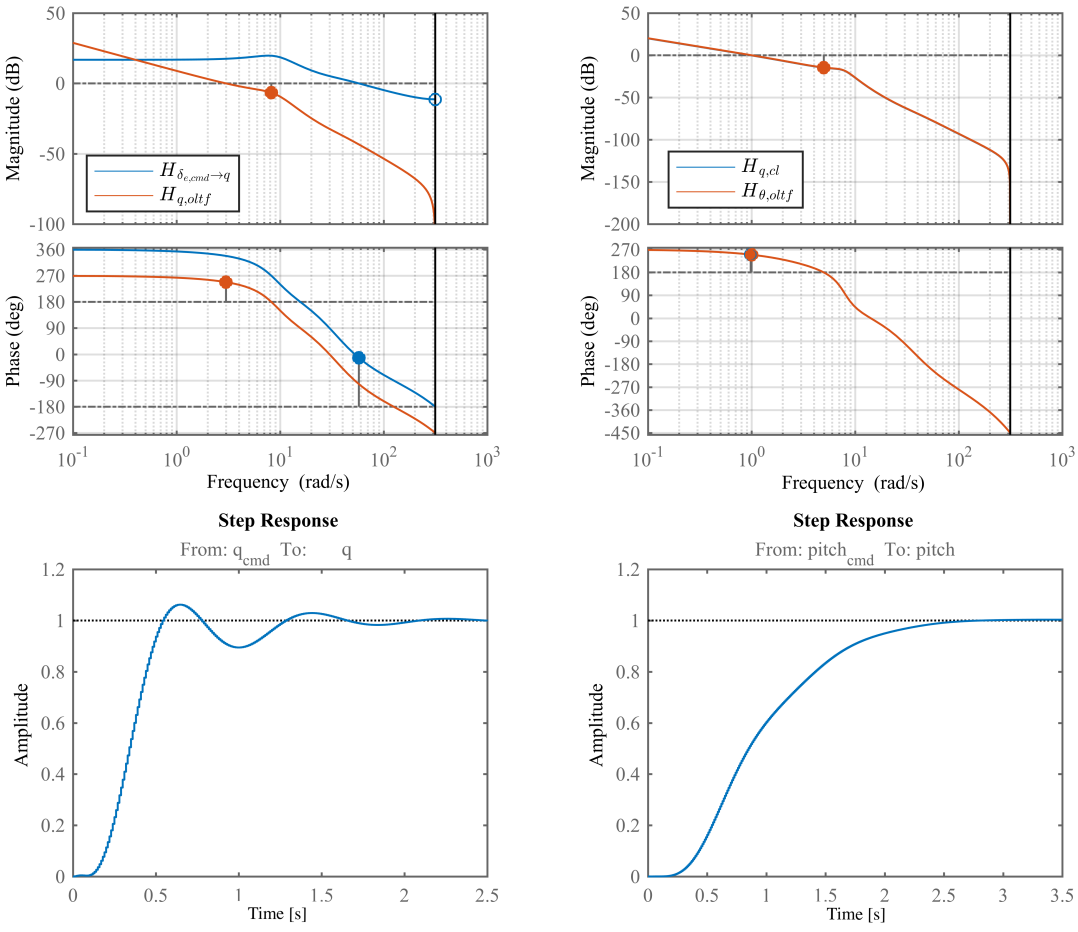

Figure 7.10: Bode plot and step response of pitch control

7.3.1.2 Roll Autopilot

Figure 7.11 illustrates the roll loop control architecture. It is very similar to the pitch angle control. For the outer roll angle control loop:

$$\dot{\phi}_{cmd} = K_{p,\phi}\left(\phi_{cmd} - \phi_{meas}\right) \tag{7.17}$$

where $K_{P,\phi}$ is the roll proportional gain, $\dot{\phi}_{cmd}$ is the setpoint command to the inner rate loop, ϕ is the roll angle in radians, the subscript *cmd* and *meas* corresponds to the pilot input command and measured quantity, respectively. Similar to the pitch loop, saturation block and conversion block from commanded Euler angle rate $\dot{\phi}_{cmd}$ to body fixed angular rate command, p_{cmd} is implemented as:

$$p_{cmd} = \dot{\phi}_{cmd} - (\sin\theta_{meas})\dot{\psi}_{cmd} \tag{7.18}$$

Accumulated roll rate error is computed using forward Euler method (Eq. (7.12)). An output saturation block and integral anti-windup described in the pitch rate controller are also implemented here. The commanded servo output of the inner loop is given by:

$$\delta_{s,cmd} = K_{P,p}\, p_{err} + K_{I,p}\, e_p \tag{7.19}$$

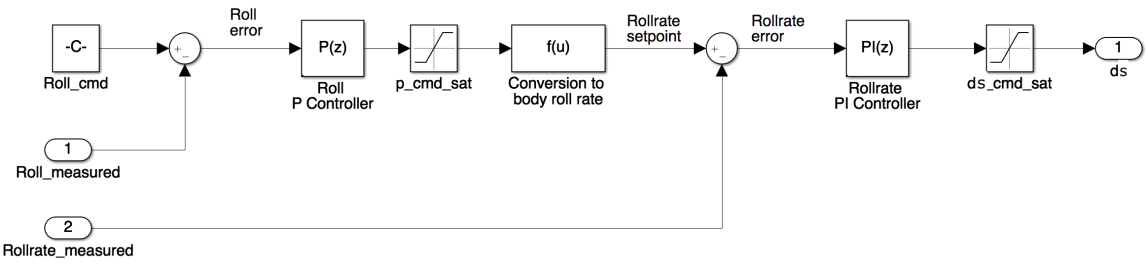

Figure 7.11: Roll autopilot architecture

The design procedure follows the pitch autopilot in Eqs. (7.14) – (7.16), replacing the plant with $H_{\delta_{s,cmd}\to p}$. The roll gains are reported in Table 7.2. Note that the gain

margin is slightly below the desired target of 6 dB. Without using additional control components (e.g., lead or lag compensators), it is not possible to satisfy both the margin and bandwidth requirement. The roll rate step response has a non-minimum phase behavior as shown in Fig. 7.12. However, the step response for roll angle is acceptable (both in margins and time domain response).

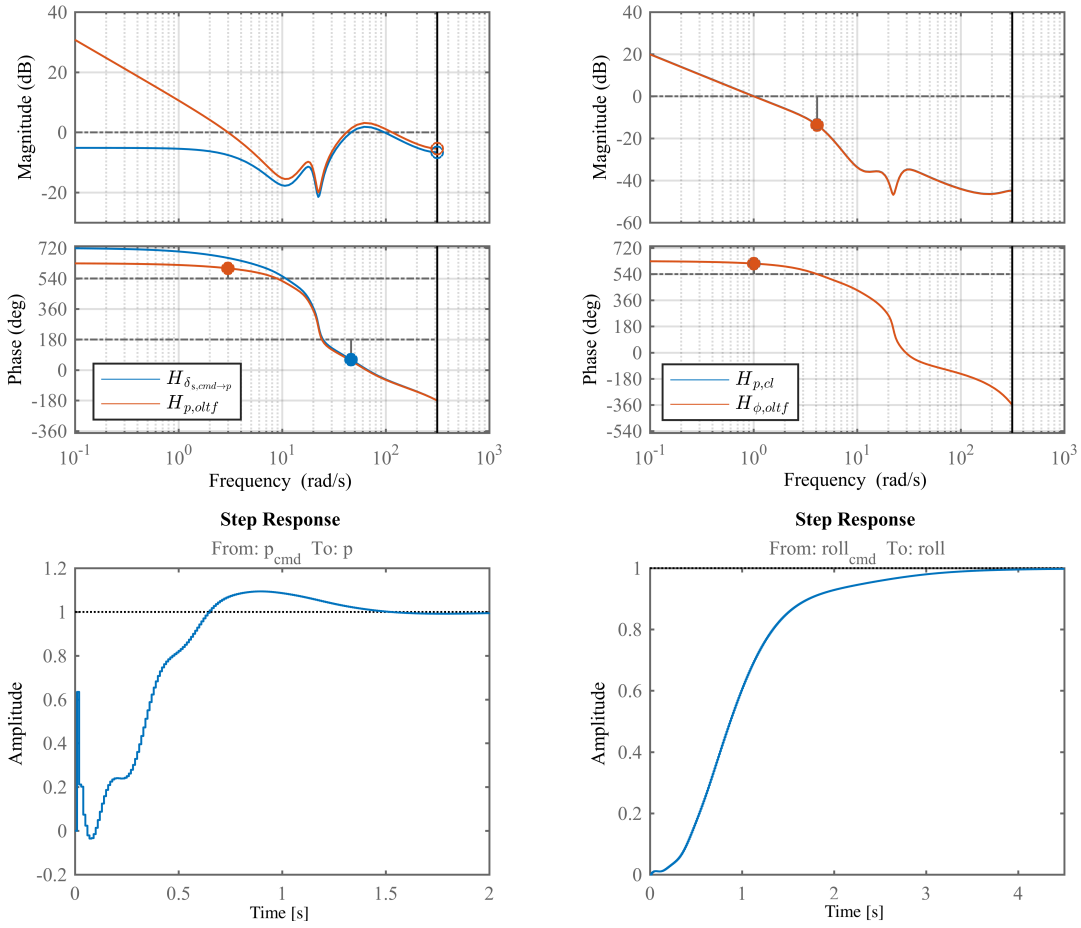

Figure 7.12: Bode plot and step response of roll control

7.3.1.3 Yaw rate Autopilot

Figure 7.13 illustrates the yaw rate loop control architecture. It is very similar to the inner rate loops for pitch and roll axes. The Euler rate command is given by the pilot. A saturation block prevents excessive large commands to be given. The

commanded Euler rate $\dot{\psi}_{cmd}$ is transformed to body fixed angular rate as:

$$r_{cmd} = -(\sin\phi_{meas})\dot{\theta}_{cmd} + (\cos\phi_{meas}\cos\theta_{meas})\dot{\psi}_{cmd} \tag{7.20}$$

A feedback loop with proportional gain is implemented to regulate the body yaw rate. The commanded servo output of the inner loop is given by:

$$\delta_{r,cmd} = K_{P,r}\, e_r \tag{7.21}$$

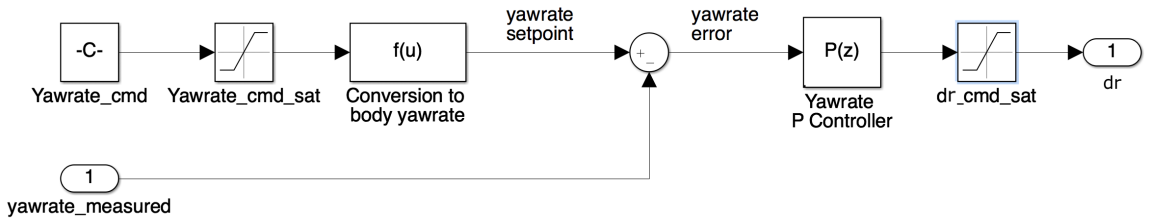

Figure 7.13: Yaw autopilot architecture

The design procedure is similar to the previous rate loops, with the exception that no integral action is used. The OLTF for the yaw rate loop is:

$$H_{r,oltf}(z) = K_{P,r} H_{\delta_{r,cmd}\to r}(z) \tag{7.22}$$

The gains and closed-loop responses are presented in Table 7.2 and Fig. 7.14.

7.3.2 Evaluation in UM/NAST Simulation

Before flying the designed controller, the autopilot is first implemented and simulated in UM/NAST. This numerical study evaluates the controller stability and performance. However, from Fig. 7.6, the UM/NAST model exhibits some discrepancies with the identified model, which was used to design the controller. In particular, the roll spoilers are too effective. Simulating the designed controller with the UM/NAST plant results in instability in the roll channel.

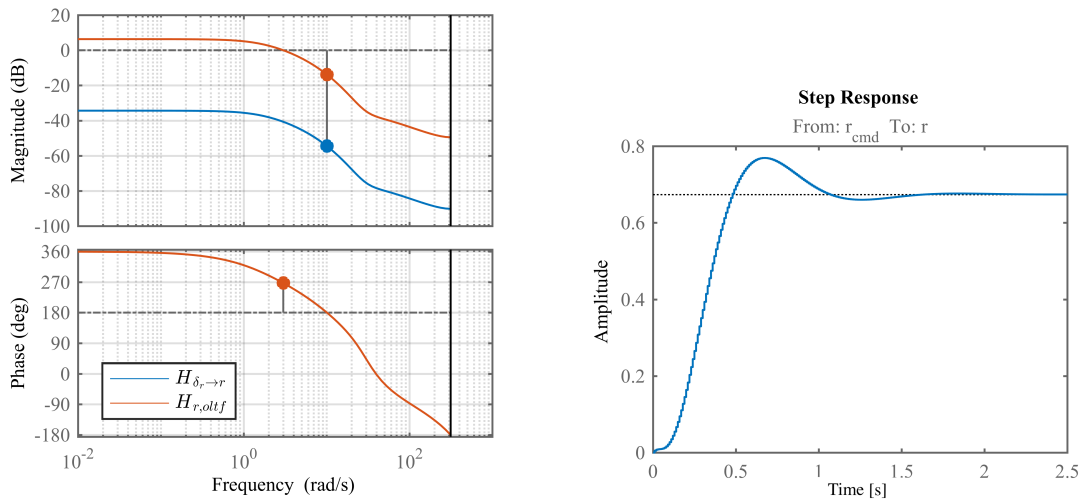

Figure 7.14: Bode plot and step response of yaw rate control

Therefore, the UM/NAST plant is tuned to match the experimental results more closely. The spoiler control derivative is reduced from $c_{l_{\delta_s}} = -1.55/\text{rad}$ to $c_{l_{\delta_s}} = -0.55/\text{rad}$ as shown in Fig 7.15. All other control derivatives are unchanged. No attempt was made to match experimental data in the pitch and yaw axes. This is a conscious attempt not to overfit flight results. In addition, this will serve as a check on the robustness of the designed controller.

Figure 7.16 shows the simulated response in UM/NAST when a step roll command $\phi_{cmd} = -0.2\,\text{rad}$ is given. Figure 7.17 shows the nonlinear response when a step pitch command $\theta_{cmd} = -0.3\,\text{rad}$ is given. In both cases, the controller performs tracking of the reference angle commands without oscillations in the outer angle loop. Some slight discrepancies are noted: the rise time of the pitch and roll angles are slightly longer than linear prediction; the pitch rate oscillations is greater in nonlinear simulation than linear design. Despite a mismatch between UM/NAST model and the experimentally identified plant, the fact that the controller works shows its robustness.

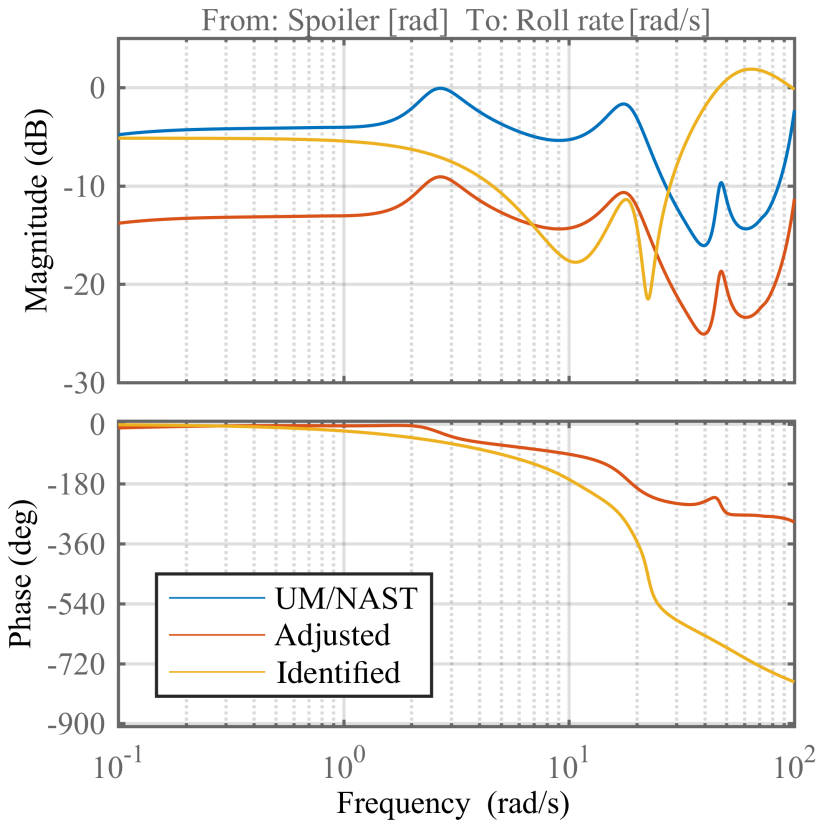

Figure 7.15: Bode plot of adjusted c_{l_δ} in the UM/NAST model

7.4 Autopilot Performance Flight Test

The designed controller is implemented and flight tested in the RRV-6B. Two flights were attempted. In the first flight, the objective was to qualitatively assess the performance of autopilot and if the stabilization reduces the pilot workload. Upon successful conclusion of the first flight and positive feedback from the pilot, the gains were finalized and no additional tuning was done. The second flight was conducted to quantitatively assess the stabilization property. The flight segments used for subsequent analysis are reported in Table 7.3.

7.4.1 Straight Level Flight

Figure 7.18 shows a flight segment where the pilot is attempting a straight level flight without engaging the autopilot. The plots are time history of Euler angles,

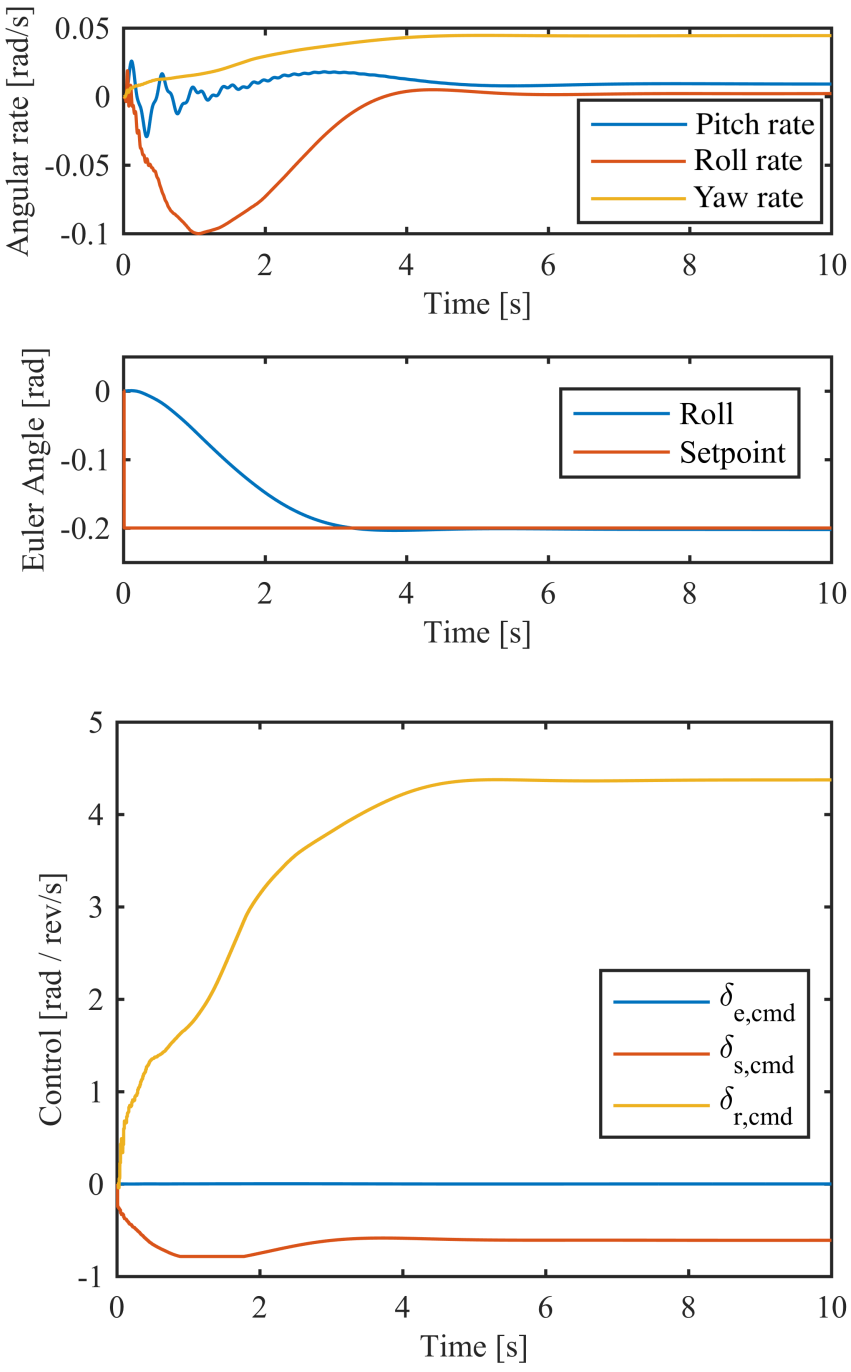

Figure 7.16: UM/NAST simulation of step roll response

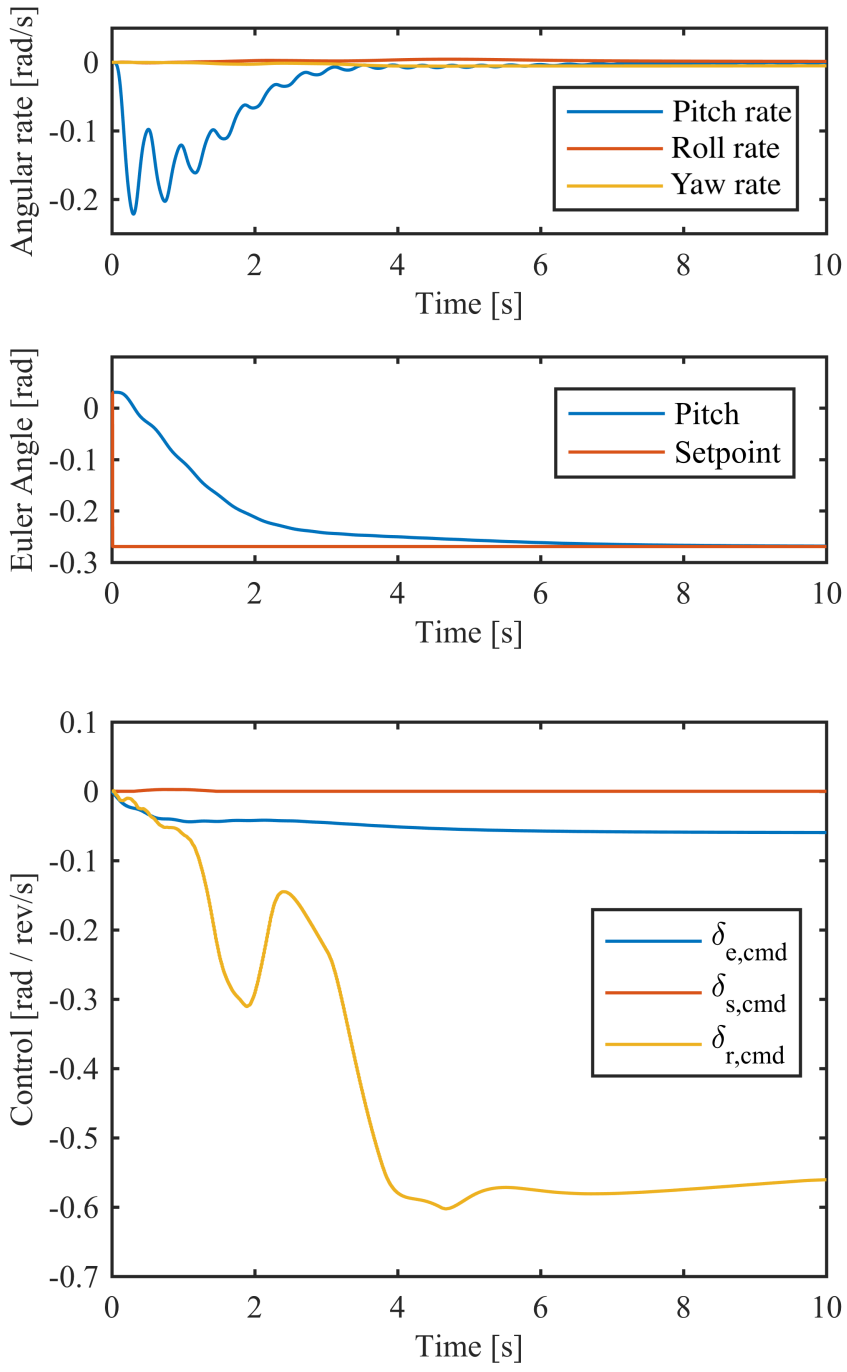

Figure 7.17: UM/NAST simulation of step pitch response

Table 7.3: Flight test sequence

Flight	Segment	Description	Flight Mode	Time
1	Turn 1	Left banking flight	`MANUAL`	690 s–702 s
1	Straight 1	Level flight	`MANUAL`	720 s–740 s
1	Turn 2	Left banking flight	`MANUAL`	735 s–755 s
1	Straight 2	Level flight	`STABILIZED`	840 s–850 s
1	Turn 5	Left banking flight	`STABILIZED`	855 s–875 s
1	Turn 6	Right banking flight	`STABILIZED`	920 s–940 s
2	Roll 1	Roll disturbance	`MANUAL`	180 s–193 s
2	Roll 3	Roll disturbance	`STABILIZED`	335 s–348 s
2	Pitch 3	Pitch disturbance	`STABILIZED`	370 s–383 s
2	Pitch 4	Pitch disturbance	`MANUAL`	407 s–420 s

body angular rates in solid lines (on the left axis) and the servo command normalized between $[-1, +1]$ (on the right axis). The flight path is also plotted with the current vehicle attitude superimposed. The axis for the aircraft are x (blue), y (red), and z (black) in usual aircraft flight dynamics frame. The magenta diamond indicates the position of the vehicle at the start of the flight sequence.

Although the flight path shows approximately straight ground track, there are significant oscillations in the roll angle and yaw rate. The pilot is attempting to correct oscillations using differential thrust. Figure 7.19 shows a straight flight segment with autopilot engaged. Roll angle and yaw rate are very effectively controlled about zero for most part of the flight. Examining Fig. 7.20, the pilot is not giving any input in all three control channels. The autopilot managed to regulate the controlled variables (pitch angle, roll angles, and yaw rate) to zero. There are some variations in height of about 10 m while the airspeed remains approximately constant at $17\,\mathrm{m\,s^{-1}}$. Since the pilot is flying by sight, and autopilot implements pitch control not altitude hold control, variations in vehicle height are unavoidable.

7.4.2 Banking Flight

Figures 7.21 – 7.22 show two banking flight segments without engaging the autopilot. When performing a left turn under manual control, the RRV-6B exhibits large oscillations in all three axes. The roll angle oscillates between ±20°, and reaching a maximum of 60° bank in one of the flight segments. This is due to both natural aircraft sensitivity to disturbance, and pilot induced oscillations due to pilot flying by sight. It clearly illustrates that it is hard to control the aircraft manually. Figures 7.23 – 7.24 show a left and right banking turn, respectively, now with the autopilot engaged. The oscillations, particularly about the lateral axis, are significantly damped. The pilot was also able to command a turn with a much smaller turn radius.

Qualitatively, the pilot gave excellent grade on the "feel" of the vehicle under autopilot stabilization. He commented that he is no longer fighting the RRV-6B's natural tendency to Dutch roll. Instead, the RRV-6B now performs like a normal but sluggish RC plane. Using coordinated roll and yaw control, he is able to achieve a smooth banking turn, which was not achievable in the past.

Examining the control loop in Figs. 7.25 – 7.26, there is little pitch command during this maneuver. To track the intended roll command, relatively large actuation of the roll spoiler (50 % to 100 %) and differential thrust (80 % to 100 %) is needed. In particular, for yaw rate control, the differential thrust command is saturated at 100 % for a significant portion of the segment. Again, this is due to limited control authority in yaw. There is a significant drop in airspeed of about $4\,\mathrm{m\,s^{-1}}$, exchanging for a height gain of about 20 m during the bank maneuver. A possible solution is to tune $q_{cmd,offset}$ in the pitch controller to minimize altitude deviations. Equation (7.11) is derived using rigid assumptions, where the lift vector rotates due to bank angle. For a flexible aircraft, vehicle attitude will affect the deformed shape, which ultimately affects lift generation. This relationship is likely to be nonlinear. By reducing the magnitude of airspeed decrease (preventing stall), the user imposed limit of 25 % for

differential thrust can be relaxed, allowing a faster response. Otherwise, the allowable and achievable range of pilot turn command have to be decreased.

7.4.3 Response to Disturbances from Trim

To clearly demonstrate the stabilizing quality of the autopilot, a second flight test was conducted. A doublet signal of 3-s length is injected into the elevators or roll spoilers to act as a disturbance to RRV-6B. After the disturbance terminates, data is collected for the next 10 seconds either in open-loop or closed-loop configuration. No pilot input is given to the vehicle in either case. In the open-loop case, the autopilot remains disengaged and control effectors remain fixed at their trimmed values. This emulates the open-loop response of the RRV-6B. In the closed-loop case, the autopilot is re-engaged, shown by the magenta line at the point of engagement. The autopilot attempts to bring the vehicle to a reference command, corresponding to aircraft trimmed state.

Figures 7.27 – 7.28 clearly show the vehicle roll angle and yaw rate are quickly regulated to setpoint values when autopilot is engaged compared to the open-loop response. For the pitch disturbance case, with the autopilot, the peak excursion in roll angle is reduced by 184% while the peak excursion in yaw rate is reduced by 49.5%. There is a slight increase in peak excursion in pitch angle by 7.6%, but it is regulated to zero in about 4 s. For the roll disturbance case, the peak excursion in pitch angle is reduced by 159% . The peak excursion in roll angle is reduced by 47.8% while the peak excursion in yaw rate is reduced by 49.4%.

The pitch closed-loop response is slower compared to the other two channels, and shows a much smaller elevator actuation range $|\delta_{e,cmd}| \leq 0.15$. The pitch autopilot loop performance can be improved by increasing the closed-loop bandwidth. The slow response of the pitch axis is apparent in the take-off and landing phases, where rapid setpoint changes are common.

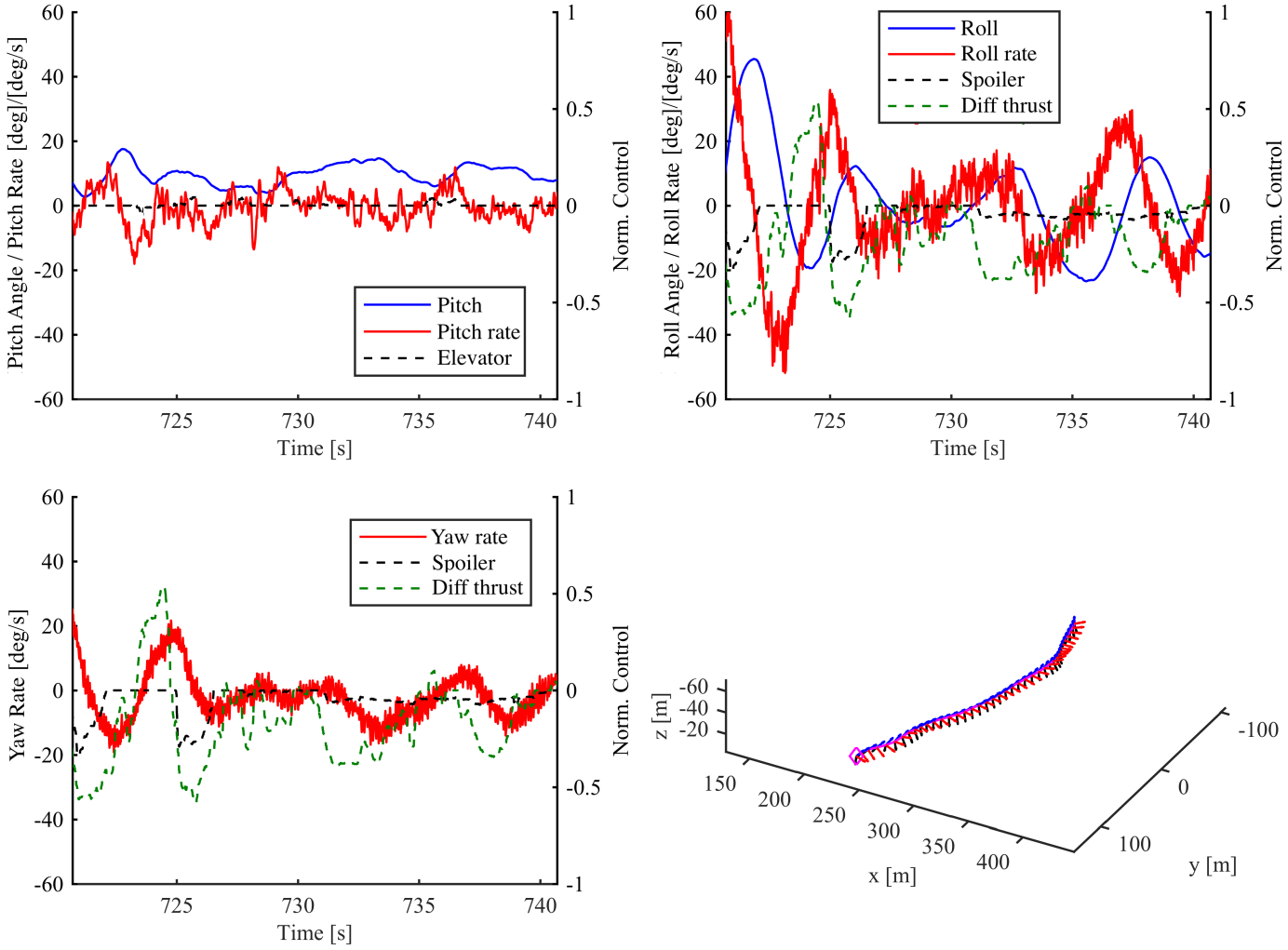

Figure 7.18: Vehicle response in straight level flight without autopilot (Straight 1, $t = 720\,\text{s}$ to $740\,\text{s}$)

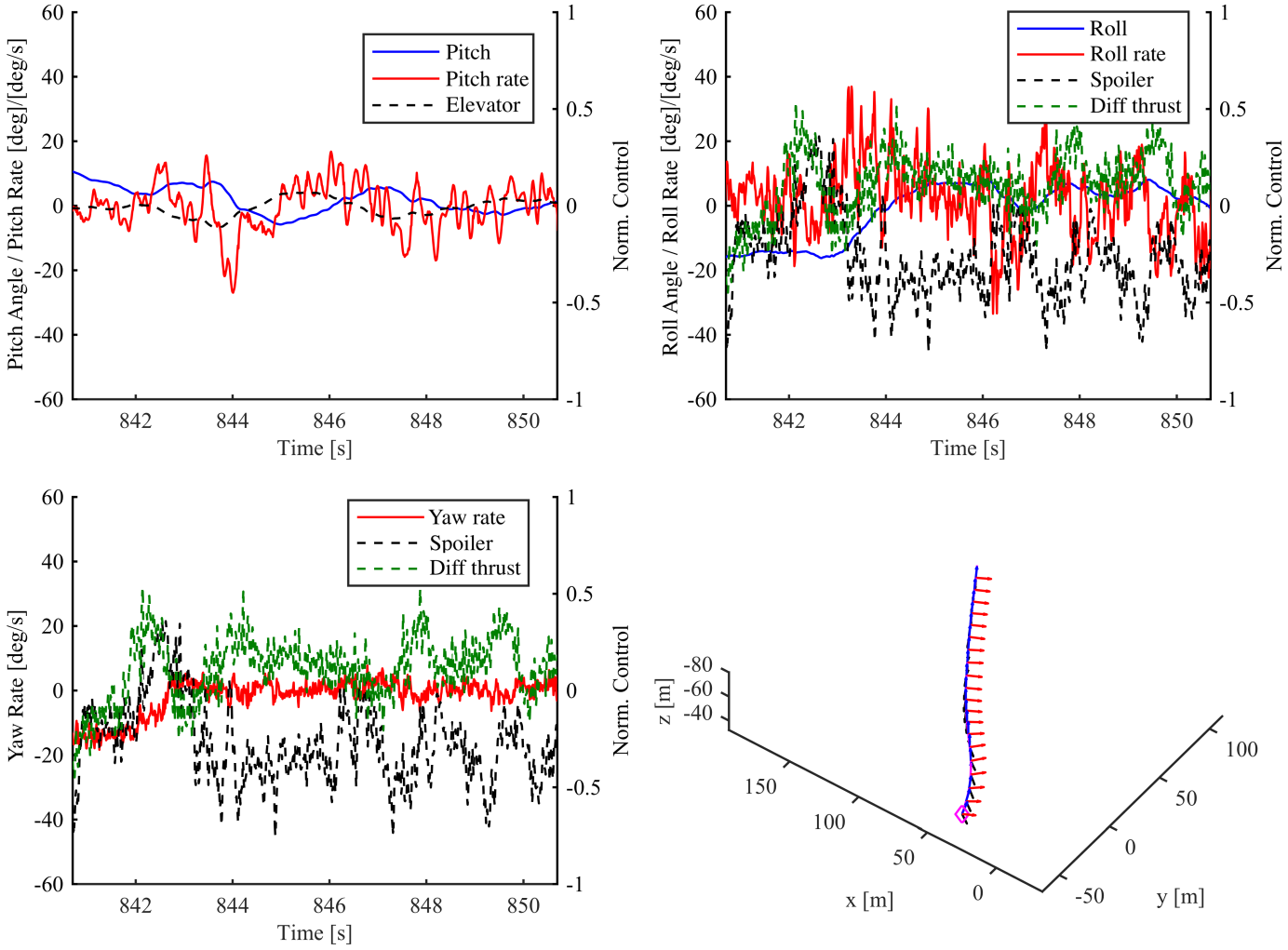

Figure 7.19: Vehicle response in straight level flight with autopilot (Straight 2, $t = 840\,\mathrm{s}$ to $850\,\mathrm{s}$)

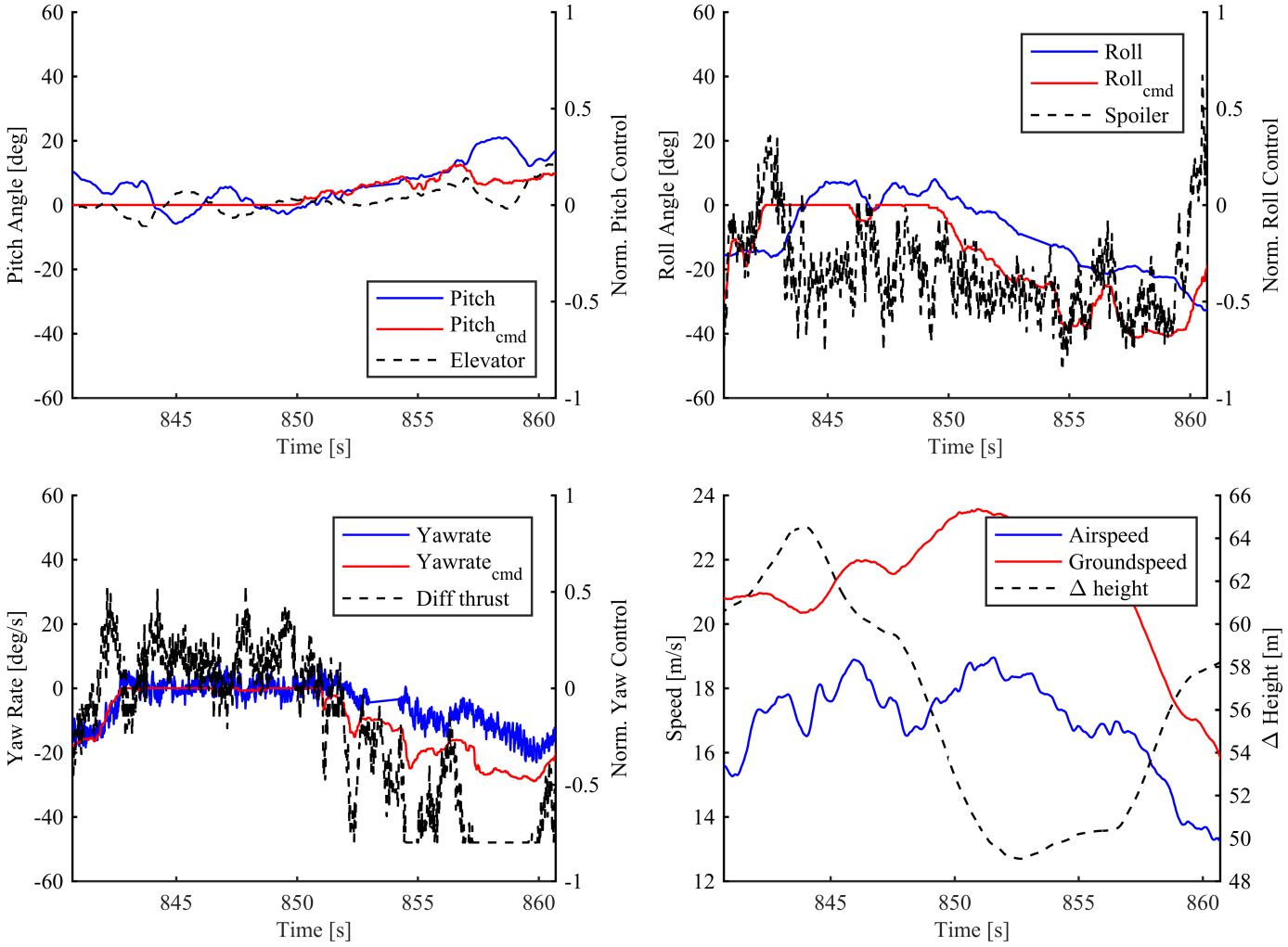

Figure 7.20: Control response in straight level flight with autopilot (Straight 2, $t = 840\,\mathrm{s}$ to $860\,\mathrm{s}$)

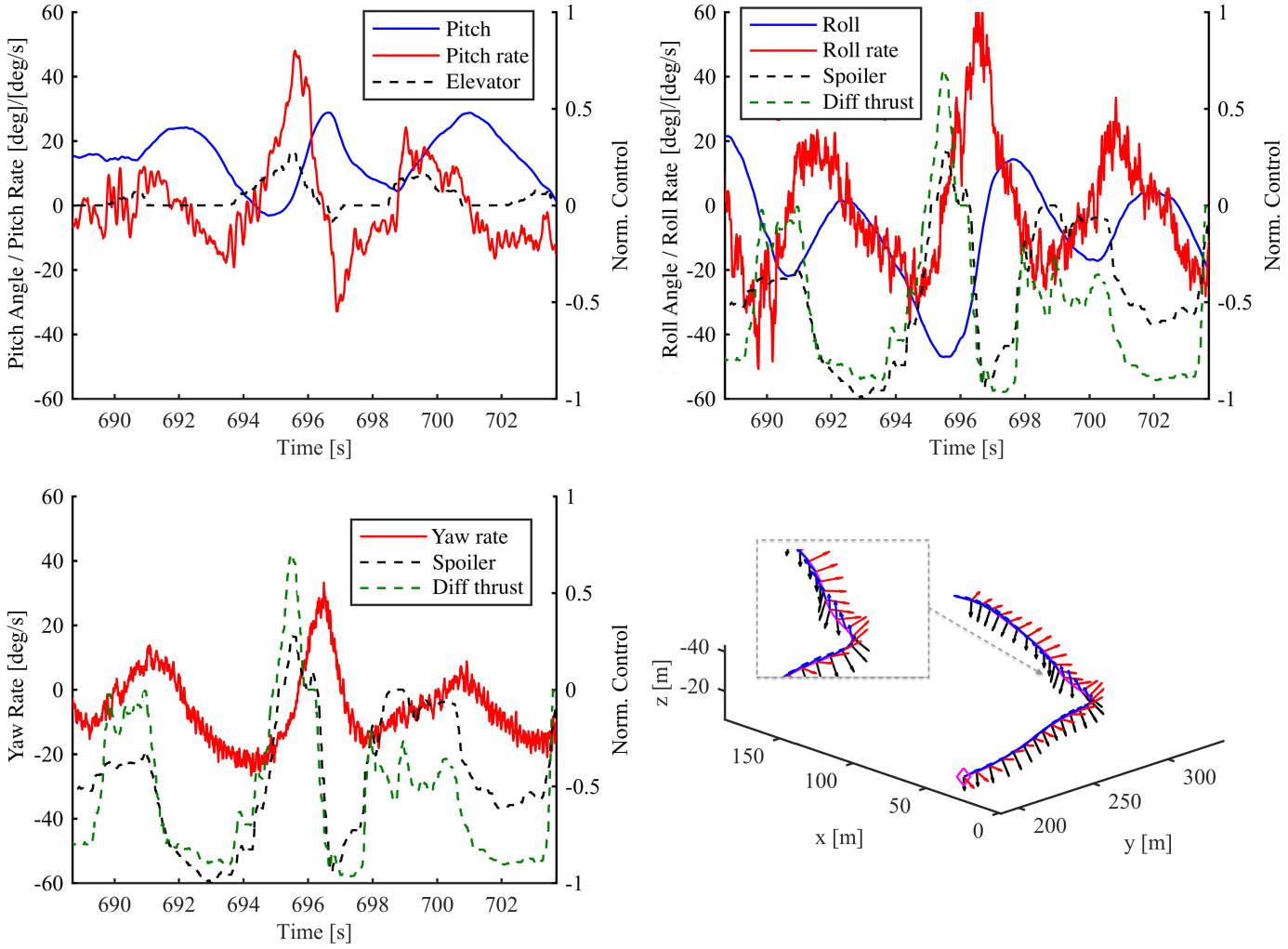

Figure 7.21: Vehicle response in banking flight without autopilot (Turn 1, $t = 690\,\mathrm{s}$ to $702\,\mathrm{s}$)

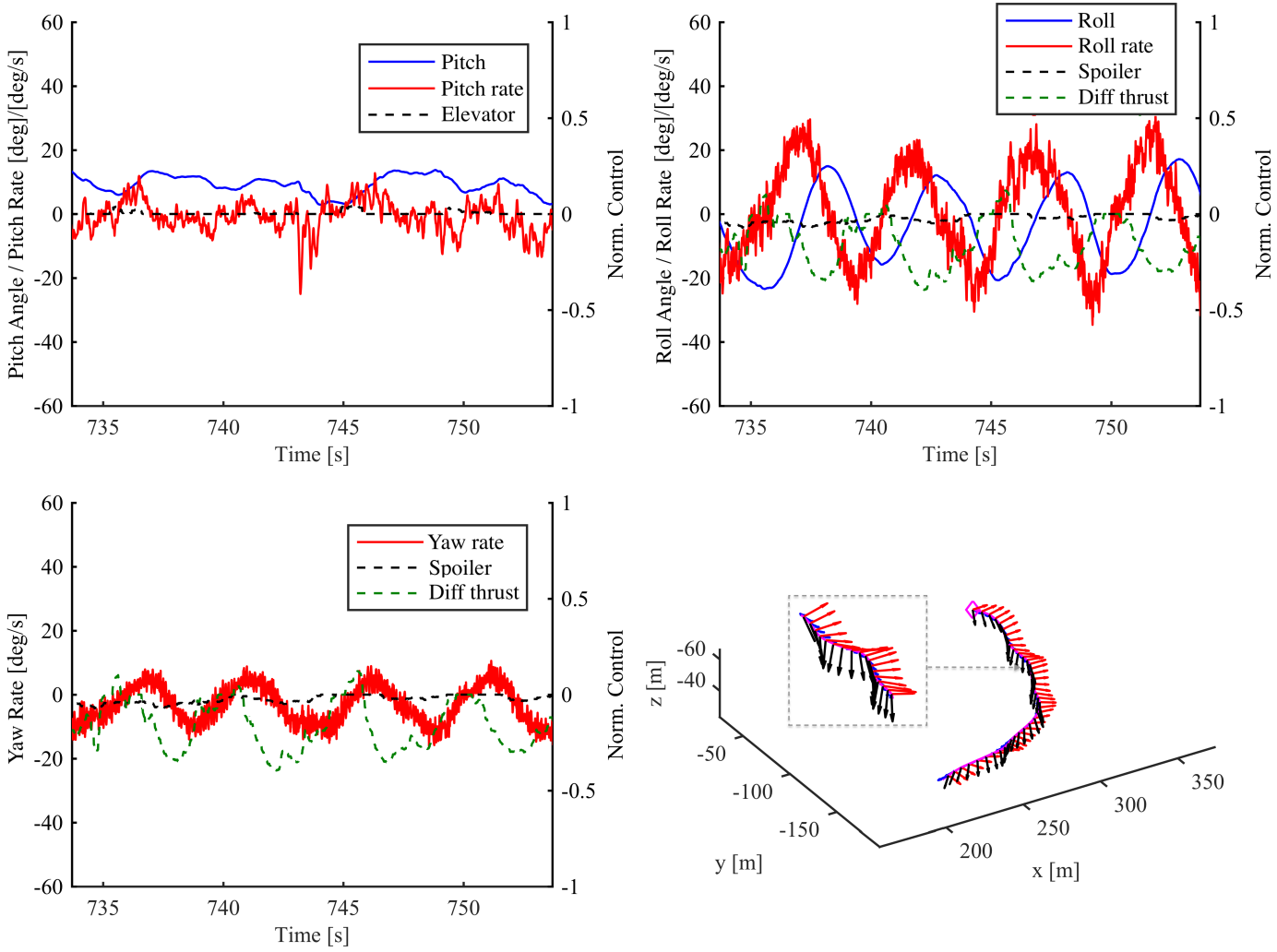

Figure 7.22: Vehicle response in banking flight without autopilot (Turn 2, $t = 735\,\mathrm{s}$ to $755\,\mathrm{s}$)

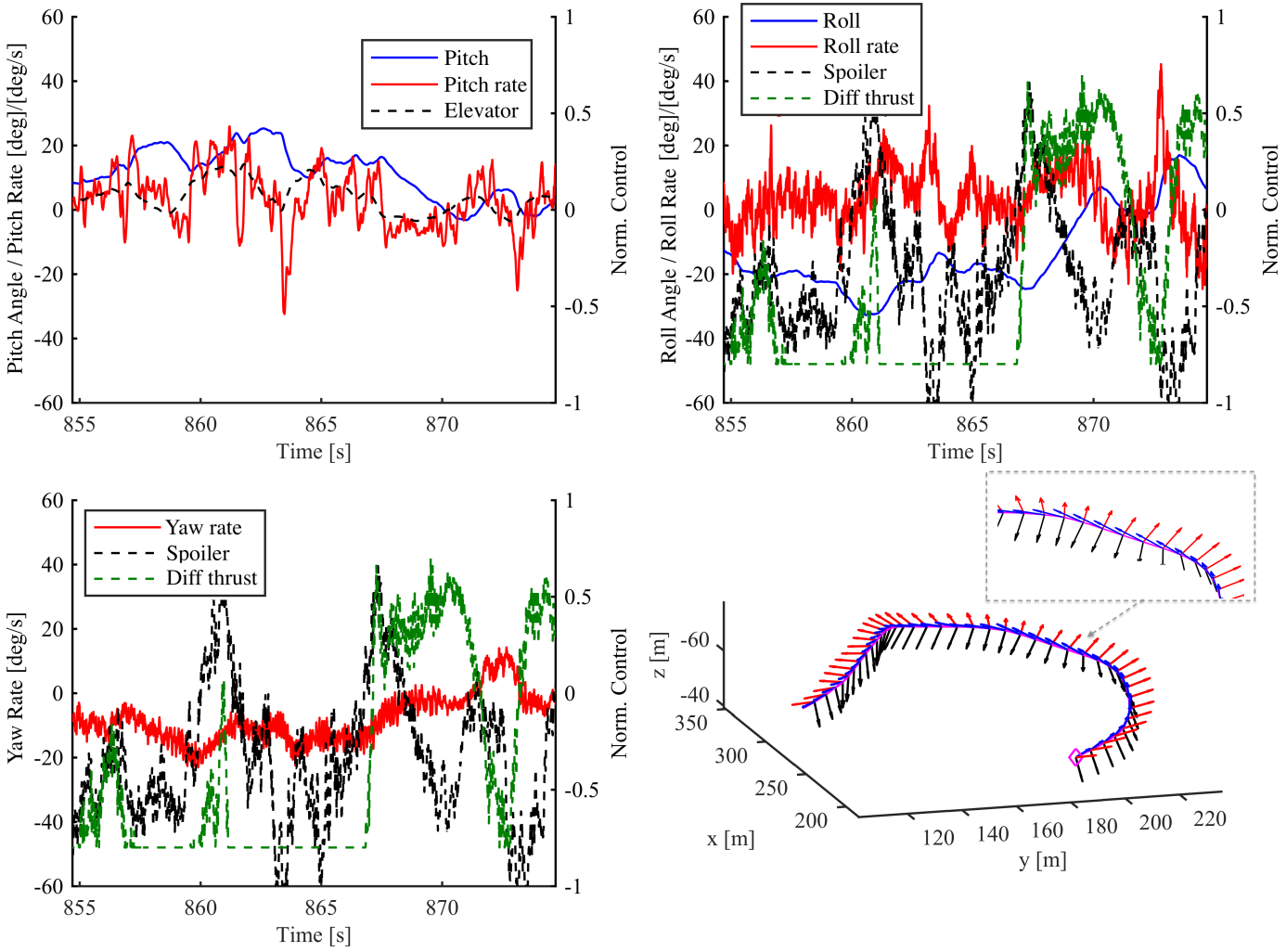

Figure 7.23: Vehicle response in banking flight with autopilot (Turn 5, $t = 855\,\mathrm{s}$ to $875\,\mathrm{s}$)

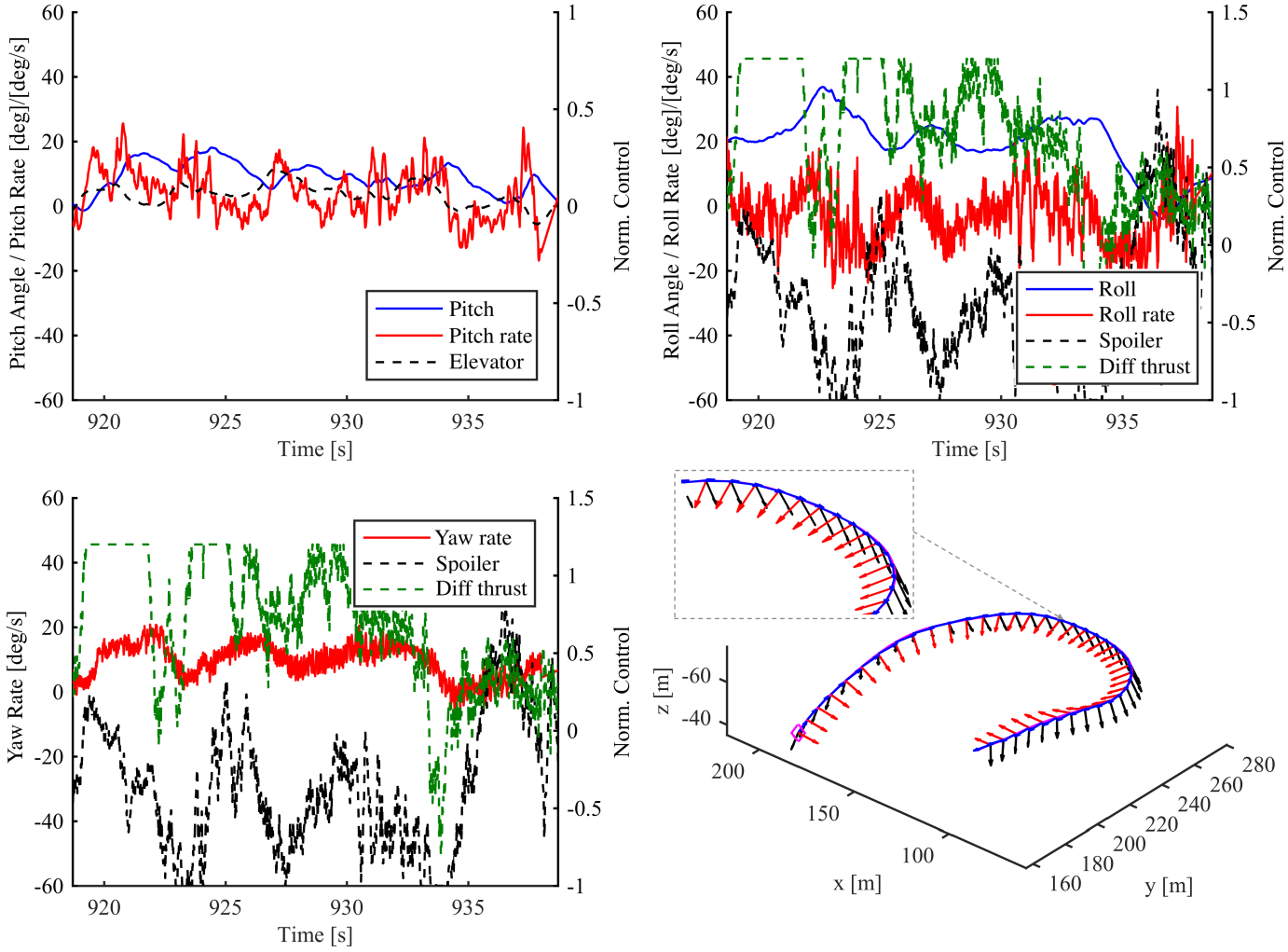

Figure 7.24: Vehicle response in banking flight with autopilot (Turn 6, $t = 920\,\mathrm{s}$ to $940\,\mathrm{s}$)

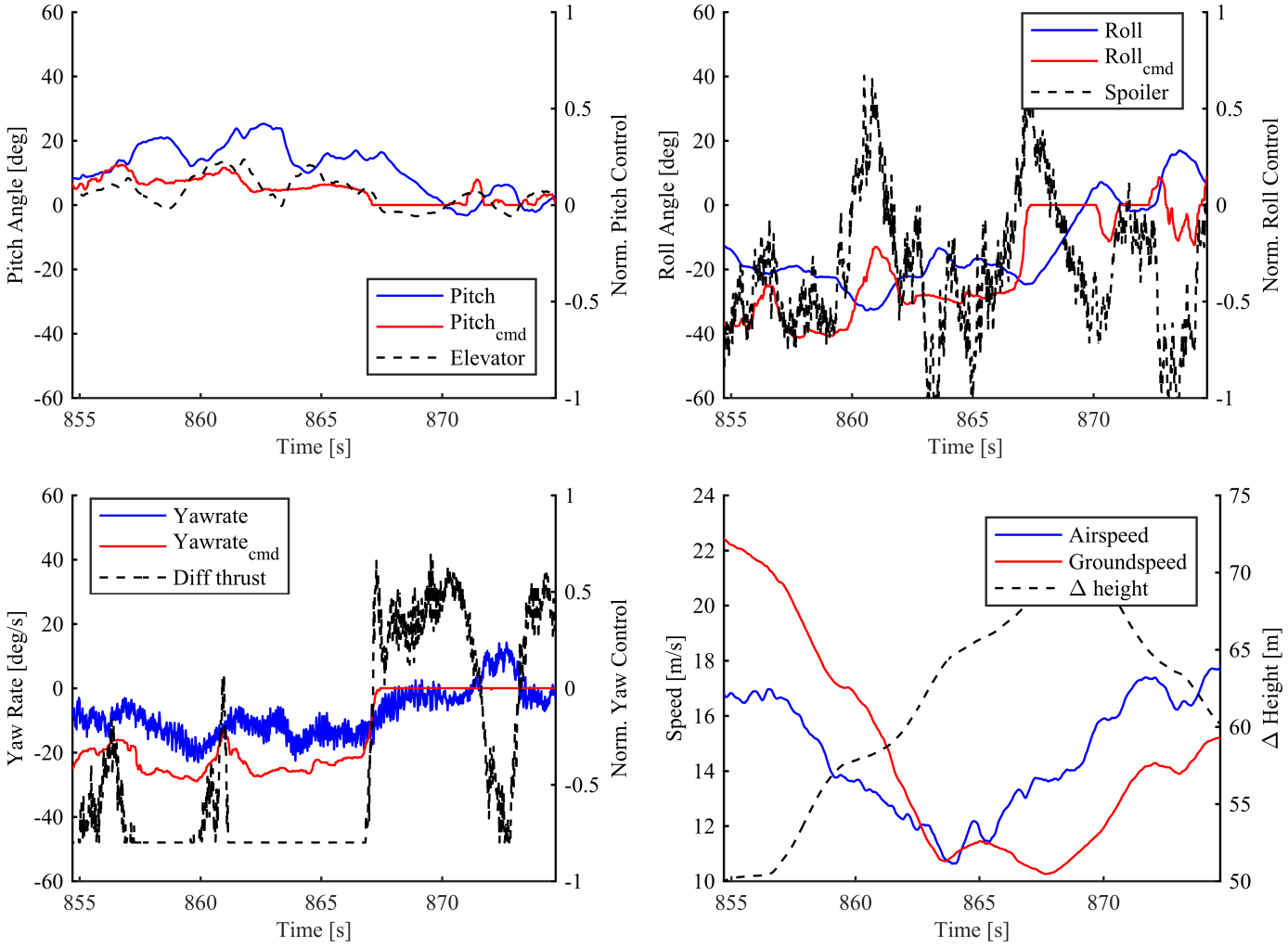

Figure 7.25: Control response in banking flight with autopilot (Turn 5, $t = 855\,\mathrm{s}$ to $875\,\mathrm{s}$)

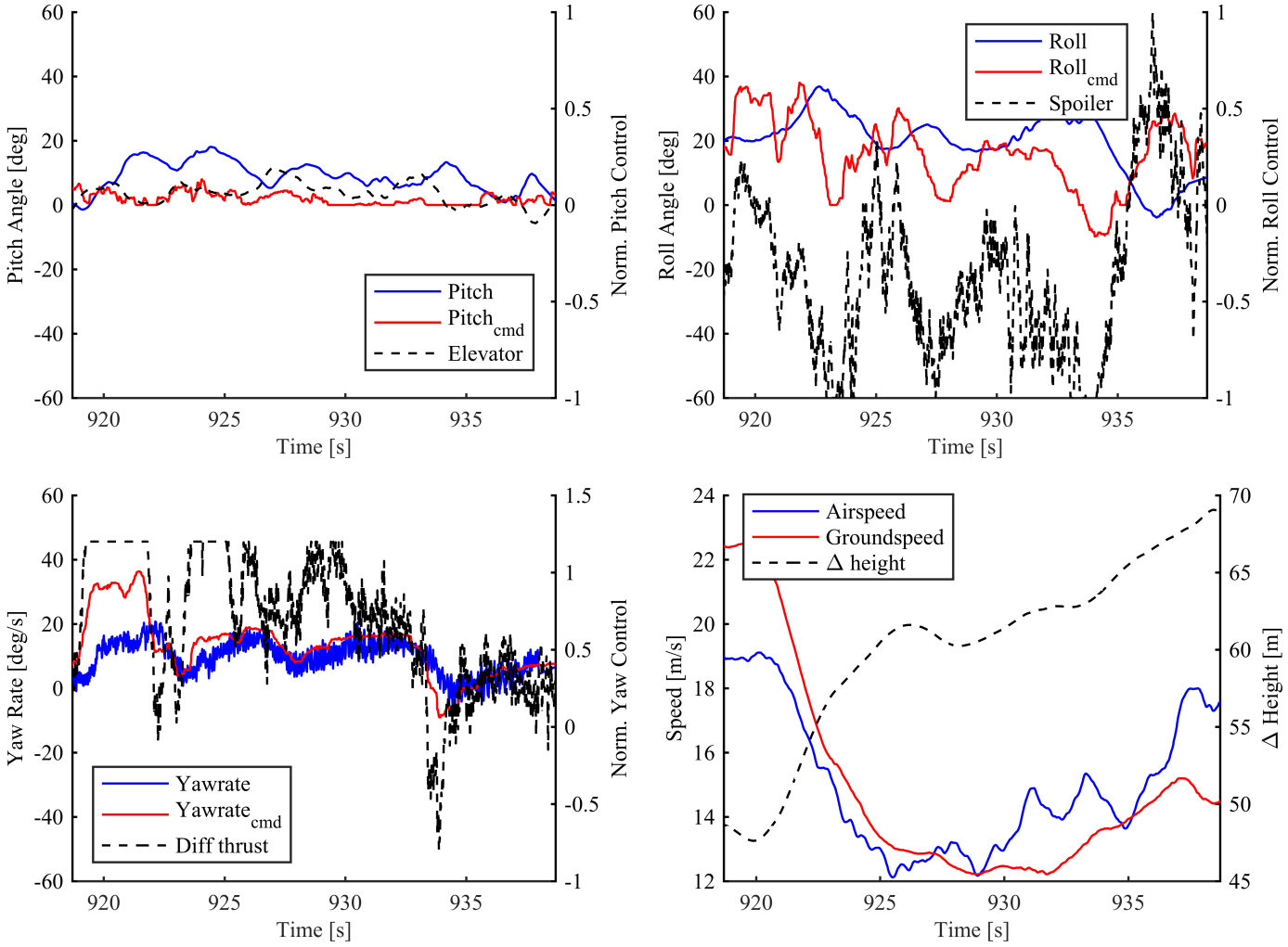

Figure 7.26: Control response in banking flight with autopilot (Turn 6, $t = 920\,\mathrm{s}$ to $940\,\mathrm{s}$)

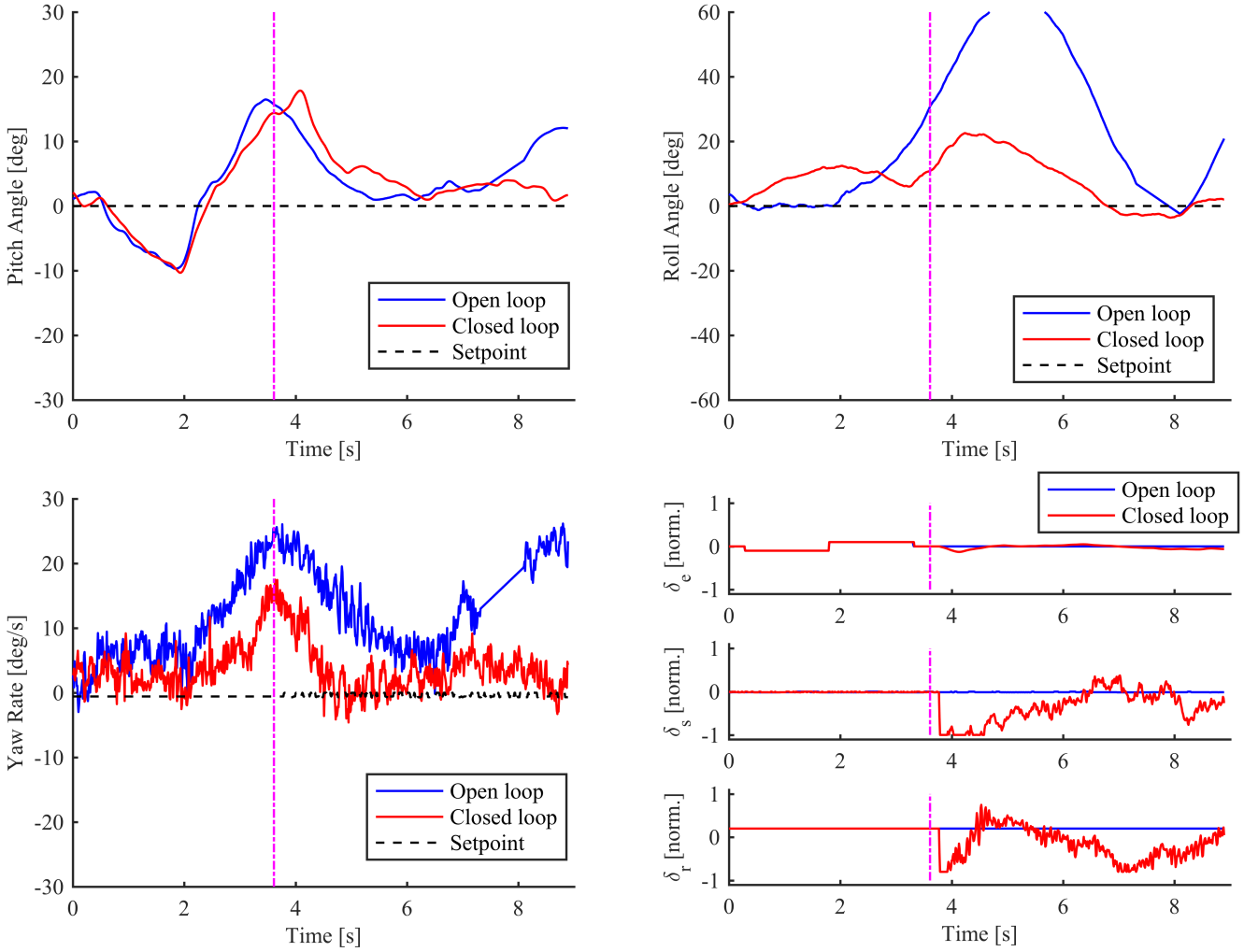

Figure 7.27: Vehicle response with pitch disturbance (point of autopilot engagement is denoted by magenta line in closed-loop case)

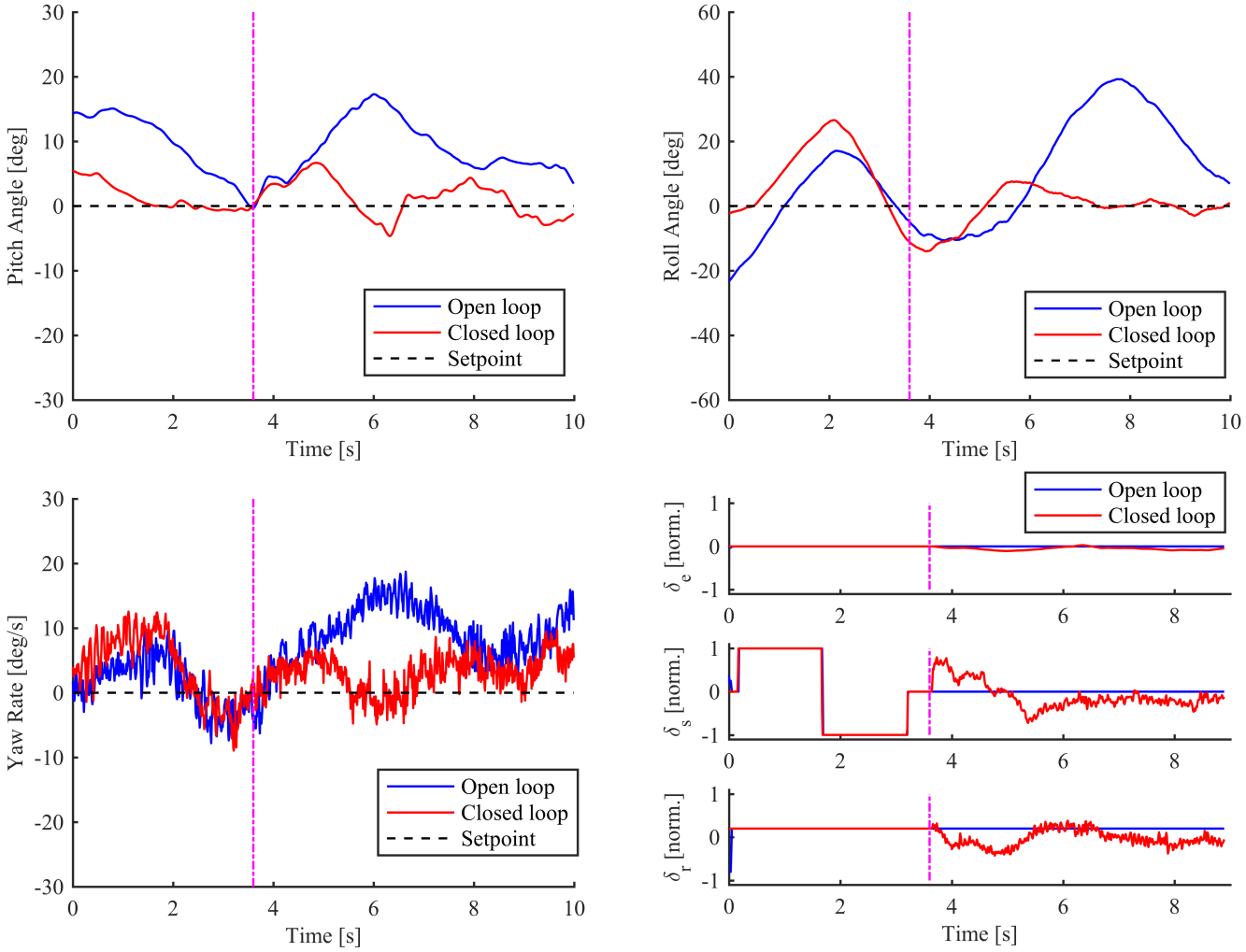

Figure 7.28: Vehicle response with roll disturbance (point of autopilot engagement is denoted by magenta line in closed-loop case)

CHAPTER 8

Concluding Remarks

This chapter summarizes the main contributions of this work. Recommendations of future work are also given.

8.1 Summary

The Nonlinear Aeroelastic Simulation Toolbox (UM/NAST) was completely rewritten in C++ for computational efficiency and code maintainability. New kinematic relations for modeling sensors placed at an arbitrary point in 3-D space away from the beam reference line were derived. Also, the implementation aspects within UM/NAST framework were discussed. Numerical validation using a simple cantilevered beam with NASTRAN showed excellent correlation. The computed acceleration values from UM/NAST were oscillatory due to numerical artifacts from explicit integration scheme. This can be mitigated by using a smaller time step size. A new numerical linearization framework was developed, allowing linear time invariant (A_d, B_d, C_d) description of a model to be easily computed. Quaternions states were treated in a special way to enforce unit magnitude during numerical perturbation. Numerical comparisons in UM/NAST showed that the linearized X-HALE state-space model is a good approximation of the nonlinear plant dynamics.

Two sensor fusion and state estimation algorithms were investigated. Each has its own advantages and disadvantages. Nonlinear Least Square (NLS) reconstruction

was based purely on geometric relationship, and useful when little information of the structure is available. However, measurement noise significantly degraded the NLS accuracy. Kalman Filter (KF) gave good estimation accuracy in the presence of noise. System dynamical properties were needed to implement this filter. Various combinations of sensors were compared in a numerical study to determine their reconstruction accuracy, with and without noise. A combination of INS+IMU+Markers sensor measurements gave the best estimation when tested on a X-HALE wing (representative of a very flexible structure).

A stereo vision-based measurement system was developed and implemented on the ATV-6B. Benchmark tests carried out using chessboards and LEDs showed good accuracy parallel to the camera sensor. Depth reconstruction was inferior and degrades as the distance from the camera increases. Subsequently, a full-scale test was realized using the ATV-6B with external Vicon system acting as the reference measurements. This test highlighted deficiencies in the camera mount. The stability of the camera mount was found to be critical since it was not possible to isolate the movement of the camera from the movement of the LED targets. The redesigned mount eliminated this issue by increasing the overall structural stiffness.

X-HALE maneuver load alleviation using MPC was also numerically demonstrated. State and control constraints were successfully imposed. Out-of-plane bending curvature was constrained at the wing roots (both left and right wings) for this study. MPC modified the vehicle response, adhering to the imposed curvature limits. MPC also successfully tracked pitch and roll angles with zero offset error. qpOASES showed promising results, approaching almost real-time performance. However, it is clear that MPC is computationally expensive compared to non-optimization based controller (e.g., LQR). Detailed profiling should be performed with ATV-6B flight hardware as part of future studies.

Experimental flight tests were carried out using the RRV-6B. SISO transfer func-

tions of i) elevator to pitch rate, ii) roll spoiler to roll rate, and iii) differential thrust to yaw rate were identified from experimental flight data. The identified model showed general correlation with UM/NAST model, but further tuning is required.

A stabilization autopilot was designed based on the identified plant. Flight validation showed good closed-loop performance and significantly improved X-HALE handling characteristics.

8.2 Key Contributions

This dissertation contributed to numerical modeling, estimation, and measurement of structural deformation of very flexible aircraft. It will be used for future ATV-6B flight experiments.

8.2.1 Theoretical Contributions

1. Developed a new version of UM/NAST in C++, maximizing computation speed and future extensibility;

2. Derived new kinematic equations of motion for modeling virtual sensors based on strain-based finite element;

3. Derived novel way of perturbing quaternion states in numerical linearization while enforcing unit quaternion magnitude;

4. Developed a nonlinear least square fitting method for recovering wing strain and body attitude from noisy sensor measurements;

5. Demonstrated multi-rate Kalman filter applicability for estimating system states from noisy sensor measurements;

6. Demonstrated use of MPC for maneuver load alleviation on X-HALE.

8.2.2 Experimental Contributions

1. Designed and implemented a stereo vision-based wing deformation measurement system on small UAS;
2. Validated stereo vision based measurement system with external benchmarks;
3. Designed and integrated sensors, computers, and supporting electronics on two aeroelastic testbed vehicles (RRV-6B and ATV-6B);
4. Experimentally characterized the dynamics of the RRV-6B via system identification;
5. Designed and validated a stabilization autopilot for the RRV-6B.

8.3 Future Work

There are different aspects of this study that warrants further investigations. They are arranged in groups and presented next.

8.3.1 UM/NAST

Propeller effects should be modeled to improve the accuracy of the simulation. Due to the location of the X-HALE tails directly behind the propellers, the tails are immersed entirely in the slipstream. Gyroscopic effects of the propellers should also be included.

Interference effects between the wings, pods, and ventral fins should be modeled. This can be done using experimental corrections factors, or numerical schemes (e.g., vortex lattice methods) which account for neighboring aerodynamic surfaces.

Sensor computations can potentially be parallelized for faster computation speed. This can be achieved relatively easily since the computation of measurement at each sensor (from the nodal FEM solution) are mutually independent.

8.3.2 MPC for Load Alleviation

Current work groups the control surfaces into "traditional" effectors (e.g, elevators for pitch, roll spoiler for roll, and differential thrust for yaw). A more fine-tuned allocation of control surfaces can potentially be better for shape control and load alleviation. For example, individual elevators can be actuated independently to allow localized control of the wing twist and hence aerodynamic distribution. Increased number of control channel affects the dimension of the MPC problem, a trade-off study will have to be performed to address this. Model reduction techniques should be attempted to reduce the problem dimension of the underlying MPC problem, enhancing the computational efficiencies for real-time implementation.

A larger flight envelope should be explored to ensure the MPC is robust. MPC may suffer from QP infeasibility due to large command changes or disturbances. For rapidly reference command, it can be coupled with a command governor to always ensure solution feasibility. For constraints handling, hard state constraints can be changed to slack variables such that the underlying problem always remains feasible. A rate-based MPC can also be used as an alternative for tracking.

Non-uniform prediction horizon time step size can also be used. A small step size can be used at the start of the prediction horizon to ensure fast dynamics (from structural deformation) are bounded. A larger step size can then be used at the end of the prediction horizon to capture the steady state behavior.

The MPC algorithm should be implemented on the ATV-6B using either Athena-II or QM-770 running the entire flight software stack concurrently to obtain more accurate asessment of computational viability. The control sample frequency (currently set at 50 Hz) should be varied to understand its implication on the control performance.

8.3.3 Shape Recovery and Sensor Fusion

The current study assumes a grid like distribution of stereo-vision makers for wing shape reconstruction. A study should be done to explore the optimal placement of sensors (including accelerometers, markers, and IMU) to obtain good reconstruction with lowest number of sensors.

Improvements should be made to the image processing algorithm to increase the computation efficiency and robustness. Although using very bright LED reduce background noise, occlusions and mis-matched identification can still occur. Other sources of light can also interfere with the marker recognition. Currently, the frame is discarded if unmatched number of LED markers are found, or if the recovered displacement exceeds a sanity check (too large a difference). A more robust marker tracking algorithm can be implemented to reduce false correspondence.

8.3.4 System Identification and Flight Testing

Nonlinear system identification can be attempted to improve the quality of the identified model. Avenues to increase the signal-to-noise ratio should also be attempted. From the dynamics of the RRV-6B, the roll and yaw control authority is low. Physical modifications (e.g., large and stiffer wing spoilers, and more powerful motor/propeller) may overcome control saturation issues. Sensors with better sensitivity and lower noise floor can also be used to pick up the low lateral response. Vibration isolation of the sensors (e.g., rubber mounts) may also help in reducing vibration experienced in flight.

BIBLIOGRAPHY

[1] Jones, J. R., *Development of a Very Flexible Testbed Aircraft for the Validation of Nonlinear Aeroelastic Codes*, Ph.D. thesis, The University of Michigan, 2017.

[2] Pang, Z. Y., Cesnik, C. E. S., and Atkins, E. M., "In-Flight Wing Deformation Measurement System for Small Unmanned Aerial Vehicles," *55th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, AIAA, National Harbor, Maryland, USA, Jan. 2014, pp. 1–13.

[3] Noll, T. E., Brown, J. M., Perez-Davis, M. E., Ishmael, S. D., Tiffany, G. C., and Gaier, M., "Investigation of the Helios Prototype Aircraft Mishap. Volume 1: Mishap Report," Tech. rep., NASA Langley Research Center, 2004.

[4] National Transportation Safety Board, "NTSB Identification: DCA15CA117," Tech. rep., Washington, District of Colombia, USA, 2015.

[5] National Transportation Safety Board, "NTSB Identification: DCA16CA197," Tech. rep., Washington, District of Colombia, USA, 2016.

[6] van Schoor, M. C. and von Flotow, A. H., "Aeroelastic Characteristics of A Highly Flexible Aircraft," *Journal of Aircraft*, Vol. 27, No. 10, 1990, pp. 901–908.

[7] Cesnik, C. E. S., Palacios, R., and Reichenbach, E. Y., "Reexamined Structural Design Procedures for Very Flexible Aircraft," *Journal of Aircraft*, Vol. 51, No. 5, 2014, pp. 1580–1591.

[8] Patil, M. J., Hodges, D. H., and Cesnik, C. E. S., "Nonlinear Aeroelastic Analysis of Complete Aircraft in Subsonic Flow," *Journal of Aircraft*, Vol. 37, No. 5, 2000, pp. 753–760.

[9] Drela, M., "Integrated Simulation Model for Preliminary Aerodynamic, Structural, and Control-law Design of Aircraft," *40th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference and Exhibit*, AIAA, St. Louis, Missouri, USA, April 1999, pp. 1644–1656.

[10] Cesnik, C. E. S. and Brown, E. L., "Modeling of High Aspect Ratio Active Flexible Wings for Roll Control," *43rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, AIAA, Denver, Colorado, USA, April 2002, pp. 1–15.

[11] Cesnik, C. E. S. and Brown, E. L., "Active Warping Control of a Joined Wing/-Tail Airplane Configuration," *44th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, AIAA, Norfolk, Virginia, USA, April 2003, pp. 1–15.

[12] Peters, D. A. and Karunamoorthy, S., "State-space Inflow Models for Rotor Aeroelasticity," *12th Applied Aerodynamics Conference*, AIAA, Colorado Springs, Colorado, USA, June 1994, pp. 828 – 837.

[13] Peters, D. A., Hsieh, M. A., and Torrero, A., "A State-Space Airloads Theory for Flexible Airfoils," *Journal of the American Helicopter Society*, Vol. 52, No. 4, 2007, pp. 329–342.

[14] Su, W. and Cesnik, C. E. S., "Strain-Based Geometrically Nonlinear Beam Formulation for Modeling Very Flexible Aircraft," *International Journal of Solids and Structures*, Vol. 48, No. 16-17, 2011, pp. 2349–2360.

[15] Su, W. and Cesnik, C. E. S., "Nonlinear Aeroelasticity of a Very Flexible Blended-Wing-Body Aircraft," *Journal of Aircraft*, Vol. 47, No. 5, 2010, pp. 1539–1553.

[16] Shearer, C. M. and Cesnik, C. E. S., "Nonlinear Flight Dynamics of Very Flexible Aircraft," *Journal of Aircraft*, Vol. 44, No. 5, 2007, pp. 1528–1545.

[17] Hallissy, B. and Cesnik, C. E. S., "High-fidelity Aeroelastic Analysis of Very Flexible Aircraft," *52nd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, AIAA, Denver, Colorado, USA, June 2012, pp. 1–22.

[18] Ritter, M., Jones, J., and Cesnik, C. E. S., "Enhanced Modal Approach for Free-flight Nonlinear Aeroelastic Simulation of Very Flexible Aircraft," *15th Dynamics Specialists Conference*, AIAA, San Diego, California, USA, Jan. 2016, pp. 1–17.

[19] Meirovitch, L. and Tuzcu, I., "Time Simulations of the Response of Maneuvering Flexible Aircraft," *Journal of Guidance, Control, and Dynamics*, Vol. 27, No. 5, 2004, pp. 814–828.

[20] Tang, D. and Dowell, E. H., "Experimental and Theoretical Study on Aeroelastic Response of High-Aspect-Ratio Wings," *AIAA Journal*, Vol. 39, No. 8, 2001, pp. 1430–1441.

[21] Su, W., Zhang, J., and Cesnik, C. E. S., "Correlations Between Um/Nast Nonlinear Aeroelastic Simulations And Experiments Of A Cantilever Slender Wing," *Proceedings of International Forum on Aeroelasticity and Structural Dynamics*, June 2009, pp. 1–15.

[22] Jategaonkar, R. V., Fischenberg, D., and Gruenhagen, W., "Aerodynamic Modeling and System Identification from Flight Data-Recent Applications at DLR," *Journal of Aircraft*, Vol. 41, No. 4, 2004, pp. 681–691.

[23] AIAA Flight Test Technical Committee, "Seeking the Proper Balance Between Simulation and Flight Test," Tech. rep., AIAA, 1999.

[24] Hamel, P. G. and Jategaonkar, R. V., "Evolution of Flight Vehicle System Identification," *Journal of Aircraft*, Vol. 33, No. 1, 1996, pp. 9–28.

[25] Morelli, E. A. and Klein, V., "Application of System Identification to Aircraft at NASA Langley Research Center," *Journal of Aircraft*, Vol. 42, No. 1, 2005, pp. 12–25.

[26] Wang, K. C. and Iliff, K. W., "Retrospective and Recent Examples of Aircraft Parameter Identification at NASA Dryden Flight Research Center," *Journal of Aircraft*, Vol. 41, No. 4, 2004, pp. 752–764.

[27] Ljung, L., *System Identification: Theory for the User*, Prentice-Hall, Upper Saddle River, NJ, USA, 1987.

[28] Tischler, M. B. and Remplr, R. K., *Aircraft and Rotorcraft System Identification*, AIAA Education Series, 2nd ed., 2012.

[29] Pintelon, R. and Schoukens, J., *System Identification: A Frequency Domain Approach*, John Wiley & Sons, Inc., Hoboken, NJ, USA, 2nd ed., 2012.

[30] Theodore, C. R., Tischler, M. B., and Colbourne, J. D., "Rapid Frequency-Domain Modeling Methods for Unmanned Aerial Vehicle Flight Control Applications," *Journal of Aircraft*, Vol. 41, No. 4, 2004, pp. 735–743.

[31] Finck, R. D., "USAF Stability and Control DATCOM (Data Compendium)," Tech. rep., USAF, 1978.

[32] Roskam, J., *Airplane Design Parts I Through VIII*, Dar Corporation, 2003.

[33] Hoffer, N. V., Coopmans, C., Jensen, A. M., and Chen, Y., "A Survey and Categorization of Small Low-Cost Unmanned Aerial Vehicle System Identification," *Journal of Intelligent & Robotic Systems*, Vol. 74, No. 1-2, 2014, pp. 129–145.

[34] Dorobantu, A., Murch, A., Mettler, B., and Balas, G., "System Identification for Small, Low-Cost, Fixed-Wing Unmanned Aircraft," *Journal of Aircraft*, Vol. 50, No. 4, 2013, pp. 1117–1130.

[35] "Comprehensive Identification from FrEquency Responses," http://nams.usra.edu/flight-control/cifer/, Accessed: 2018-01-04.

[36] Scheper, K. Y., Chowdhary, G., and Johnson, E. N., "Aerodynamic System Identification of Fixed-wing UAV," *AIAA Atmospheric Flight Mechanics Conference*, AIAA, Boston, Massachusetts, USA, Aug. 2013, pp. 1–10.

[37] Schulze, P. C., Danowsky, B. P., Yang, P., and Harris, C., "Real Time Modal Identification of a Flexible Unmanned Aerial Vehicle," *AIAA Atmospheric Flight Mechanics Conference*, AIAA, Grapevine, Texas, USA, Jan. 2017, pp. 1–13.

[38] Silva, B. G. d. O. and Mönnich, W., "System Identification of Flexible Aircraft in Time Domain," *AIAA Atmospheric Flight Mechanics Conference*, AIAA, Minneapolis, Minnesota, USA, Aug. 2012, pp. 1–26.

[39] Garrec, C. L. and Kubica, F., "In-Flight Structural Modes Identification for Comfort Improvement by Flight Control Laws," *Journal of Aircraft*, Vol. 42, No. 1, Jan. 2005, pp. 90–92.

[40] Kukreja, S. L. and Brenner, M. J., "Nonlinear Aeroelastic System Identification with Application to Experimental Data," *Journal of Guidance, Control, and Dynamics*, Vol. 29, No. 2, 2006, pp. 374–381.

[41] Liu, T., Cattafesta, L. N., Radeztsky, R. H., and Burner, A. W., "Photogrammetry Applied to Wind-Tunnel Testing," *AIAA Journal*, Vol. 38, June 2000, pp. 964–971.

[42] Albertani, R., Stanford, B., Hubner, J. P., and Ifju, P. G., "Aerodynamic Coefficients and Deformation Measurements on Flexible Micro Air Vehicle Wings," *Experimental Mechanics*, Vol. 47, No. 5, 2007, pp. 625–635.

[43] Lamiscarre, B. B., Sidoruk, B., Selvaggini, R., Castan, E., and Bazin, M., "Stereo-Optical System for High-Accuracy and High-Speed 3D Shape Reconstitution: Wind-Tunnel Applications for Model Deformation Measurements," *SPIE Proceedings*, edited by G. A. Kyrala and D. R. Snyder, Vol. 2273, SPIE, San Diego, California, USA, Oct. 1994, pp. 46–55.

[44] Burner, A. W. and Liu, T., "Videogrammetric Model Deformation Measurement Technique," *Journal of Aircraft*, Vol. 38, No. 4, 2001, pp. 745–754.

[45] DeAngelis, V. M., "In-flight Deflection Measurement of the HiMAT Aeroelastically Tailored Wing," *Journal of Aircraft*, Vol. 19, No. 12, 1982, pp. 1088–1094.

[46] Allen, M., Lizotte, A., Dibley, R., and Clarke, R., "Loads Model Development and Analysis for the F/A-18 Active Aeroelastic Wing Airplane," *AIAA Atmospheric Flight Mechanics Conference*, AIAA, San Francisco, California, USA, June 2012, pp. 1–29.

[47] Kurita, M., Koike, S., Nakakita, K., and Masui, K., "In-Flight Wing Deformation Measurement," *51st AIAA Aerospace Sciences Meeting Including The New Horizons Forum and Aerospace Exposition*, AIAA, Grapevine, Texas, USA, Jan. 2013, pp. 1–7.

[48] Kang, L., Kim, D., and Han, J., "Estimation of Dynamic Structural Displacements Using Fiber Bragg Grating Strain Sensors," *Journal of Sound and Vibration*, Vol. 305, No. 3, 2007, pp. 534–542.

[49] Ko, W. L., Richards, W. L., and Tran, V. T., "Displacement Theories for In-Flight Deformed Shape Predictions of Aerospace Structures," Tech. Rep. NASA/TP-2007-214612, NASA, Edwards, California, USA, 2007.

[50] Derkevorkian, A., Masri, S. F., Alvarenga, J., Boussalis, H., Bakalyar, J., and Richards, W. L., "Strain-Based Deformation Shape-Estimation Algorithm for Control and Monitoring Applications," *AIAA Journal*, Vol. 51, No. 9, 2013, pp. 2231–2240.

[51] Pak, C., "Wing Shape Sensing from Measured Strain," *AIAA Journal*, Vol. 54, No. 3, 2016, pp. 1068–1077.

[52] Baraniello, V. R., Cicala, M., and Corraro, F., "An Extension of Integrated Navigation Algorithms to Estimate Elastic Motions of Very Flexible Aircrafts," *2010 IEEE Aerospace Conference*, IEEE, Big Sky, Montana, USA, Dec. 2010, pp. 1–14.

[53] Armesto, L., Chroust, S., Vincze, M., and Tornero, J., "Multi-rate Fusion with Vision and Inertial Sensors," *IEEE International Conference on Robotics and Automation*, IEEE, New Orleans, Louisiana, USA, Aug. 2004, pp. 193–199.

[54] Smyth, A. and Wu, M., "Multi-Rate Kalman Filtering for the Data Fusion of Displacement and Acceleration Response Measurements in Dynamic System Monitoring," *Mechanical Systems and Signal Processing*, Vol. 21, No. 2, Feb. 2007, pp. 706–723.

[55] Theis, J., Pfifer, H., Balas, G., and Werner, H., "Integrated Flight Control Design for a Large Flexible Aircraft," *American Control Conference*, IEEE, July 2015, pp. 3830–3835.

[56] Cook, R. G., Palacios, R., and Goulart, P., "Robust Gust Alleviation and Stabilization of Very Flexible Aircraft," *AIAA Journal*, Vol. 51, No. 2, 2013, pp. 330–340.

[57] Pedro, J. and Bigg, C., "Development of a Flexible Embedded Aircraft/Control System Simulation Facility," *AIAA Modeling and Simulation Technologies Conference and Exhibit*, AIAA, San Francisco, California, USA, Aug. 2005, pp. 1–25.

[58] Schirrer, A., Westermayer, C., Hemedi, M., and Kozek, M., "LQ-Based Design of the Inner Loop Lateral Control for a Large Flexible BWB-Type Aircraft," *IEEE International Conference on Control Applications*, IEEE, Yokohama, Japan, July 2010, pp. 1850–1855.

[59] Tuzcu, I., Marzocca, P., Cestino, E., Romeo, G., and Frulla, G., "Stability and Control of a High-Altitude, Long-Endurance UAV," *Journal of Guidance, Control, and Dynamics*, Vol. 30, No. 3, 2007, pp. 713–721.

[60] Gonzalez, P. J., Silvestre, F. J., Paglione, P., Köthe, A., Pang, Z. Y., and Cesnik, C. E. S., "Linear Control of Highly Flexible Aircraft based on Loop Separation," *AIAA Atmospheric Flight Mechanics Conference*, AIAA, Washington, District of Colombia, USA, June 2016, pp. 1–26.

[61] Schmidt, D. and Chavez, F., "Systems Approach to Characterizing Aircraft Aeroelastic Model Variation for Robust Control Applications," *AIAA Guidance, Navigation, and Control Conference*, AIAA, Montreal, Canada, Aug. 2001, pp. 1–15.

[62] Qu, Z., Lavretsky, E., and Annaswamy, A. M., "An Adaptive Controller for Very Flexible Aircraft," *AIAA Guidance, Navigation, and Control Conference*, AIAA, Boston, Massachusetts, USA, Aug. 2013, pp. 1–11.

[63] Patil, M. J., *Nonlinear Aeroelastic Analysis, Flight Dynamics, Nonlinear Aeroelastic Analysis, Flight Dynamics, and Control of a Complete Aircraft*, Ph.D. thesis, Georgia Institute of Technology, 1999.

[64] Shearer, C. M. and Cesnik, C. E. S., "Trajectory Control for Very Flexible Aircraft," *Journal of Guidance, Control, and Dynamics*, Vol. 31, 2008, pp. 340–357.

[65] Shearer, C. M., *Coupled Nonlinear Flight Dynamics, Aeroelasticity, and Control of Very Flexible Aircraft*, Ph.D. thesis, The University of Michigan, 2006.

[66] Dillsaver, M. J., Kalabic, U. V., Kolmanovsky, I. V., and Cesnik, C. E. S., "Constrained Control of Very Flexible Aircraft Using Reference and Extended Command Governors," *American Control Conference*, IEEE, Washington, District of Colombia, USA, June 2013, pp. 1608–1613.

[67] Mayne, D. Q., "Model Predictive Control: Recent Developments and Future Promise," *Automatica*, Vol. 50, No. 12, 2014, pp. 2967–2986.

[68] Eren, U., Prach, A., Koçer, B. B., Raković, S. V., Kayacan, E., and Açkmee, B., "Model Predictive Control in Aerospace Systems: Current State and Opportunities," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 7, 2017, pp. 1–25.

[69] Kögel, M. and Findeisen, R., "A Fast Gradient Method for Embedded Linear Predictive Control," *IFAC Proceedings Volumes*, Vol. 44, No. 1, Jan. 2011, pp. 1362–1367.

[70] Zometa, P., Kögel, M., Faulwasser, T., and Findeisen, R., "Implementation Aspects of Model Predictive Control for Embedded Systems," *American Control Conference*, IEEE, Montreal, Quebec, Canada, May 2012, pp. 1205–1210.

[71] Kebler, C. W., "Parallel Fourier-Motzkin Elimination," *Euro-Par '96 - Parallel Processing*, Vol. 1123 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, Berlin, Heidelberg, Germany, 1996, pp. 66–71.

[72] Kögel, M. and Findeisen, R., "Fast Predictive Control of Linear, Time-Invariant Systems Using an Algorithm Based on the Fast Gradient Method and Augmented Lagrange Multipliers," *2011 IEEE International Conference on Control Applications)*, IEEE, Denver, Colorado, USA, Sept. 2011, pp. 780–785.

[73] Kögel, M. and Findeisen, R., "Fast Predictive Control of Linear Systems Combining Nesterov's Gradient Method and the Method of Multipliers," *IEEE Conference on Decision and Control and European Control Conference*, IEEE, Orlando, Florida, USA, Dec. 2011, pp. 501–506.

[74] Jerez, J. L., Goulart, P. J., Richter, S., Constantinides, G. A., Kerrigan, E. C., and Morari, M., "Embedded Online Optimization for Model Predictive Control at Megahertz Rates," *IEEE Transactions on Automatic Control*, Vol. 59, No. 12, 2014, pp. 3238–3251.

[75] Ferreau, H. J., Bock, H. G., and Diehl, M., "An Online Active Set Strategy to Overcome the Limitations of Explicit MPC," *International Journal of Robust and Nonlinear Control*, Vol. 18, No. 8, May 2008, pp. 816–830.

[76] Ferreau, H. J., Kirches, C., Potschka, A., Bock, H. G., and Diehl, M., "qpOASES: A Parametric Active-Set Algorithm for Quadratic Programming," *Mathematical Programming Computation*, Vol. 6, No. 4, Dec. 2014, pp. 327–363.

[77] Simpson, R. J., Palacios, R., Hesse, H., and Goulart, P., "Predictive Control for Alleviation of Gust Loads on Very Flexible Aircraft," *55th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, AIAA, National Harbor, Maryland, USA, Jan. 2014, pp. 1–25.

[78] Zometa, P., Kögel, M., and Findeisen, R., "muAO-MPC: a Free Code Generation Tool for Embedded Real-Time Linear Model Predictive Control," *American Control Conference*, IEEE, Washington, District of Colombia, USA, July 2013, pp. 5320–5325.

[79] Giesseler, H. G., Kopf, M., Varutti, P., Faulwasser, T., and Findeisen, R., "Model Predictive Control for Gust Load Alleviation," *IFAC Proceedings Volumes*, Vol. 45, No. 17, Aug. 2012, pp. 27–32.

[80] Haghighat, S., Liu, H. H. T., and Martins, J. R. R. A., "Model-Predictive Gust Load Alleviation Controller for a Highly Flexible Aircraft," *Journal of Guidance, Control, and Dynamics*, Vol. 35, No. 6, 2012, pp. 1751–1766.

[81] Wang, Y., Wynn, A., and Palacios, R., "Model-Predictive Control of Flexible Aircraft Dynamics using Nonlinear Reduced-Order Models," *57th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, AIAA, San Diego, California, USA, Jan. 2016, pp. 1–11.

[82] Brown, E. L., *Integrated Strain Actuation In Aircraft With Highly Flexible Composite Wings*, Ph.D. thesis, Massachusetts Institute of Technology, 2003.

[83] Su, W., *Coupled Nonlinear Aeroelasticity and Flight Dynamics of Fully Flexible Aircraft*, Ph.D. thesis, The University of Michigan, 2008.

[84] Hodges, D. H. and Pierce, G. A., *Introduction to Structural Dynamics and Aeroelasticity*, Cambridge University Press, 2nd ed., 2011.

[85] Shearer, C. M. and Cesnik, C. E. S., "Modified Generalized Alpha Method for Integrating Governing Equations of Very Flexible Aircraft," *47th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, AIAA, Newport, Rhode Island, USA, May 2006, pp. 1–21.

[86] Su, W. and Cesnik, C. E. S., "Dynamic Response of Highly Flexible Flying Wings," *AIAA Journal*, Vol. 49, No. 2, 2011, pp. 324–339.

[87] Dillsaver, M. J., *Gust Response and Control of Very Flexible Aircraft*, Phd thesis, The University of Michigan, 2013.

[88] Dillsaver, M., Cesnik, C. E. S., and Kolmanovsky, I., "Gust Load Alleviation Control for Very Flexible Aircraft," *AIAA Atmospheric Flight Mechanics Conference*, AIAA, Portland, Oregon, USA, Aug. 2011, pp. 2001–18.

[89] Mozilla Foundation, "Mozilla Public License," https://www.mozilla.org/en-US/MPL/2.0/, Accessed: 2018-01-04.

[90] Regents of the University of California, "The 2-Clause BSD License," https://opensource.org/licenses/BSD-2-Clause, Accessed: 2018-01-04.

[91] Massachusetts Institute of Technology, "The MIT License," https://opensource.org/licenses/MIT, Accessed: 2018-01-04.

[92] Free Software Foundation, "GNU Library General Public License, Version 2," https://www.gnu.org/licenses/old-licenses/lgpl-2.0.html, Accessed: 2018-01-04.

[93] Broyden, C. G., "A Class of Methods for Solving Nonlinear Simultaneous Equations," *Mathematics of Computation*, Vol. 19, No. 92, Oct. 1965, pp. 577.

[94] Hartley, R. and Zisserman, A., *Multiple View Geometry in Computer Vision*, Cambridge University Press, Cambridge, UK, 2nd ed., 2004.

[95] Zhang, Z., "A Flexible New Technique for Camera Calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, No. 11, 2000, pp. 1330–1334.

[96] Tsai, R. Y., "A Versatile Camera Calibration Technique for High-accuracy 3D Machine Vision Metrology Using Off-the-shelf TV Cameras and Lenses," *IEEE Journal on Robotics and Automation*, Vol. 3, No. 4, 1987, pp. 323–344.

[97] "OpenCV Computer Vision Library," https://opencv.org, Accessed: 2018-01-04.

[98] Bradski, G. R. and Kaehler, A., *Learning OpenCV: Computer Vision with the OpenCV Library*, O'Reilly Media, 1st ed., 2008.

[99] Bouguet, J., "Camera Calibration Toolbox for Matlab," http://www.vision.caltech.edu/bouguetj/calib_doc/, Accessed: 2018-01-04.

[100] Goodwin, G. C., Seron, M. M., and Doná, J. A. D., *Constrained Control and Estimation: an Optimisation Approach*, Springer London, London, 2010.

[101] "qpOASES Software," https://projects.coin-or.org/qpOASES, Accessed: 2018-01-04.

[102] Kalman, R. E. and Bucy, R. S., "New Results in Linear Filtering and Prediction Theory," *Journal of Basic Engineering*, Vol. 83, No. 1, 1961, pp. 95.

[103] Engelberg, S., *Digital Signal Processing*, Signals and Communication Technology, Springer London, London, UK, 2008.

[104] Cesnik, C. E. S., Senatore, P., Su, W., Atkins, E. M., and Shearer, C. M., "X-HALE: A Very Flexible UAV for Nonlinear Aeroelastic Tests," *AIAA Journal*, Vol. 50, 2012, pp. 2820–2833.

[105] Drela, M., "XFOIL," Accessed: 2018-01-04.

[106] Eubank, R., Atkins, E., and Meadows, G., "Unattended Operation Of An Autonomous Seaplane For Persistent Surface And Airborne Ocean Monitoring," *OCEANS 2010*, IEEE, Seattle, Washington, USA, Sept. 2010, pp. 1–8.

[107] Yeo, D., Atkins, E. M., Bernal, L. P., and Shyy, W., "Fixed-Wing Unmanned Aircraft In-Flight Pitch and Yaw Control Moment Sensing," *Journal of Aircraft*, Vol. 52, No. 2, 2015, pp. 403–420.

[108] "Pixhawk Developer Manual," http://dev.px4.io, Accessed: 2018-01-04.

[109] Chadha, S. A., Pomeroy, B. W., and Selig, M. S., "Computational Study of a Lifting Surface in Propeller Slipstreams," *34th AIAA Applied Aerodynamics Conference*, AIAA, Washington, District of Colombia, USA, June 2016, pp. 1–19.

[110] Teixeira, P. and Cesnik, C. E. S., "Propeller Effects On Aeroelastic Behavior Of Very Flexible Aircraft," *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, No. Jan., 2018, pp. 1–22.

[111] Teixeira, P. and Cesnik, C. E. S., "Propeller Effects on the Dynamic Response of HALE Aircraft," *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, No. Jan., 2018, pp. 1–22.

[112] Blakelock, J. H., *Automatic Control of Aircraft and Missiles*, Wiley-Interscience, 2nd ed., 1991.

[113] Brown, D. C., "Close-Range Camera Calibration," *Photogrammetric Engineering*, Vol. 37, No. 8, 1971, pp. 855–866.

APPENDIX A

Stereo-Vision Methodology

This appendix presents detailed theory behind stereo vision methodology.

A.1 Camera Calibration

Since columns of the rotation matrix are orthonormal, two constraint equations can be written as:

$$r_1^T r_2 = 0 \tag{A.1}$$

$$r_1^T r_1 = r_2^T r_2 \tag{A.2}$$

Equivalently, from Eq. (3.5), one has:

$$h_1^T M^{-T} M^{-1} h_2 = 0 \tag{A.3}$$

$$h_1^T M^{-T} M^{-1} h_1 = h_2^T M^{-T} M^{-1} h_2 \tag{A.4}$$

Defining matrix B as:

$$B = M^{-T} M^{-1} = \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \\ B_{31} & B_{32} & B_{33} \end{bmatrix} \tag{A.5}$$

The closed-form analytical solution in terms of intrinsic camera parameters is:

$$B = \begin{bmatrix} \frac{1}{f_x^2} & 0 & -\frac{c_x}{f_x^2} \\ 0 & \frac{1}{f_y^2} & -\frac{c_y}{f_y^2} \\ -\frac{c_x}{f_x^2} & -\frac{c_y}{f_y^2} & \frac{c_x}{f_x^2} + \frac{c_y}{f_y^2} + 1 \end{bmatrix} \tag{A.6}$$

Expanding Eq. (A.3) and rewriting in unknown coefficients in B:

$$h_i^T B h_j = v_{ij} b = \begin{bmatrix} h_{i1}^T h_{j1} \\ h_{i1}^T h_{j2} + h_{i2}^T h_{j1} \\ h_{i2}^T h_{j2} \\ h_{i3}^T h_{j1} + h_{i1}^T h_{j3} \\ h_{i3}^T h_{j2} + h_{i2}^T h_{j3} \\ h_{i3}^T h_{j3} \end{bmatrix} \begin{Bmatrix} B_{11} \\ B_{12} \\ B_{13} \\ B_{22} \\ B_{13} \\ B_{23} \\ B_{33} \end{Bmatrix} \tag{A.7}$$

Writing the constraint equations Eq. (A.3) and Eq. (A.4) using Eq. (A.7), one has:

$$\begin{bmatrix} v_{12}^T \\ (v_{11} - v_{22}) \end{bmatrix} b = 0 \tag{A.8}$$

By obtaining multiple views of the calibration chessboard, Eq. (A.8) can be stacked row-wise to obtain an overdetermined linear system of equations as:

$$Vb = 0 \tag{A.9}$$

From the analytical solution to B from Eq. (A.6),

$$\begin{aligned} f_x &= \sqrt{\frac{1}{sB_{11}}} \\ f_y &= \sqrt{\frac{B_{11}}{s\left(B_{11}B_{22} - B_{12}^2\right)}} \\ c_x &= -sB_{13}f_x^2 \\ c_y &= \frac{B_{12}B_{13} - B_{11}B_{23}}{B_{11}B_{22} - B_{12}^2} \\ s &= B_{33} - \frac{B_{13}^2 + c_y\left(B_{12}B_{13} - B_{11}B_{23}\right)}{B_{11}} \end{aligned} \tag{A.10}$$

The extrinsic calibration is given by

$$\begin{aligned} r_1 &= \frac{1}{s}M^{-1}h_1 \\ r_2 &= \frac{1}{s}M^{-1}h_2 \\ r_3 &= r_1 \times r_2 \\ t &= \frac{1}{s}M^{-1}h_3 \end{aligned} \tag{A.11}$$

Disregarding lens distortion effects, at least 2 views of 4 corners are required to completely estimate 10 parameters from Eq. (3.5).

Considering lens distortion, for stability and robustness of camera calibration algorithm, a larger number of chessboard views are required. The above computations are derived for a perfect pinhole camera. Coupling Eq. (3.5) and Eq. (3.3), the solution is no longer linear and cannot be solved analytically. In this case, a global Levenberg-Marquardt optimization algorithm minimizing the re-projection error d, is used to estimate camera intrinsic, extrinsic and lens distortion parameters simulatenously. Iterative computation approach to compute lens distortion coefficients is based on work by Brown [113].

$$(M, R, t) = \arg\min \sum d\left(q_d, HQ\right) \tag{A.12}$$

where re-projection error d is euclidean distance between the projected points HQ and the undistorted pin hole camera points q, computed from distorted points q_d based on lens distortion model in Eq. (3.3). The solution from Eq (A.10) and (A.11) can be used a initialization point for this nonlinear optimization problem.

A.2 Stereo Calibration

In this section, subscript l will be used to denote left camera and r to denote right camera. By taking joint views of chessboard, that is, views of the same chessboard from perspectives of both camera, each joint view produces one set of rotation R_l, R_r and translation t_l, t_r, relating the common object space to individual image space of the left and right camera. The image space of the left and right cameras are also related by a rotation R and translation t:

$$R = R_r R_l^T \tag{A.13}$$

$$t = t_r - R t_l \tag{A.14}$$

`cv::stereoCalibrate` implements a global Levenberg-Marquardt to compute R, t.

$$(R, t) = \arg\min \sum d\left(Q_l, R\left(Q_r - t\right)\right) \tag{A.15}$$

The optimization algorithm minimizes the difference between the chessboard corners expressed in the left camera frame and the reprojected chessboard corners of the right camera frame rotated and translate to left camera frame. Finally, the projection

matrix of the left and right cameras are:

$$P_l = \begin{bmatrix} f_{x_l} & 0 & c_{x_l} \\ 0 & f_{y_l} & c_{y_l} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{A.16}$$

$$P_r = \begin{bmatrix} f_{x_r} & 0 & c_{x_r} \\ 0 & f_{y_r} & c_{y_r} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{A.17}$$

A.3 Stereo Rectification

The resultant rotation matrix is denoted by r_l and r_r for left and right cameras, respectively. To transform the epipole of the left camera to infinity, define the following unit vector:

$$\begin{aligned} e_1 &= \frac{t}{\|t\|} \\ e_2 &= \frac{\begin{bmatrix} -t_y & t_x & 0 \end{bmatrix}^T}{\sqrt{t_x^2 + t_y^2}} \\ e_3 &= e_1 \times e_2 \end{aligned} \tag{A.18}$$

The rotation matrix which takes the epipole to infinity is given by:

$$R_{rect} = \begin{bmatrix} e_1^T \\ e_2^T \\ e_3^T \end{bmatrix} \tag{A.19}$$

To row align both images, the final camera rotation matrix is given by:

$$\begin{aligned} R_l &= R_{rect} r_l \\ R_r &= R_{rect} r_r \end{aligned} \tag{A.20}$$

A.4 Triangulation

Triangulated point $\hat{Q}$ in physical space must satisfy the projection matrix of both camera:

$$\begin{aligned} q_l &= P_l \hat{Q}_l \\ q_r &= P_r \hat{Q}_r \end{aligned} \tag{A.21}$$

However, due to noise from image measurements and camera calibration error, in general $\hat{Q}_l \neq \hat{Q}_r$. Using linear triangulation method [94], the cross product taken with itself to obtain 3 equations per image measurement. However, only 2 equations are linearly independent.

$$q \times P\hat{Q} = 0 \tag{A.22}$$

Expanding, there will be 4 equations from 2 images given by:

$$\begin{bmatrix} x\left(p_3^T\right) - \left(p_1^T\right) \\ y\left(p_3^T\right) - \left(p_2^T\right) \\ x\left(p_3^T\right) - \left(p_1^T\right) \\ y\left(p_3^T\right) - \left(p_2^T\right) \end{bmatrix} \left\{ \hat{Q} \right\} = 0 \tag{A.23}$$

This system of equation is over-determined but is linear in $\hat{Q}$. Therefore, Singular Value Decomposition (SVD) can be used to compute the solution. $\hat{Q}$ is the singular vector corresponding to the smallest singular value.

APPENDIX B

X-HALE Vehicle Properties

This appendix presents structural, aerodynamic, servo, and propulsion characteristics of X-HALE (both RRV-6B and ATV-6B). Experimental characterization for the structure is carried out in Ref. [1]. Experimental characterization for the servo and propulsion is carried out by the author.

B.1 Structural Properties of X-HALE

Components distributed properties are defined in the local beam frame w, while concentrated properties are defined in the aircraft body frame B. Details on the characterization on the structural and aerodynamic properties are given in Ref. [1].

Table B.1: Structural properties of booms and tails

	Center Boom B0	Side Boom (B1-B4)	Tail	Units
Ref. axis location (from component L.E.)	50.0	50.0	50.0	% chord
Center of gravity (from component L.E.)	50.0	50.0	50.0	% chord
Rod diameter (front)	0.024	0.024	–	m
Rod diameter (rear)	0.013	0.013	–	m
Chord length	–	–	0.11	m
Mass	0.07	0.07	0.24	$\mathrm{kg\,m^{-1}}$
Out-of-plane bending inertia (I_{yy})	1.79×10^{-3}	1.79×10^{-3}	1.10×10^{-4}	kg m
In-plane bending inertia (I_{zz})	1.79×10^{-3}	1.79×10^{-3}	5.97×10^{-3}	kg m
Out/In-plane bending inertia (I_{yz})	0	0	0	kg m

Note: Inertia values are reported about beam reference line
in the local beam coordinate system

Table B.2: Structural properties of ventral fins

	Center Fin (V0)	Side Fin (V1,V2)	Units
Ref. axis location (from component front edge)	1.0878	1.134	% length
Chord length	0.13	0.13	m
Fin length	0.73	0.48	m
Mass	0.074	0.04	kg
x_{cg}	0	0	m
y_{cg}	0.38	0.2415	m
y_{cg}	-0.075	-0.075	m
I_{xx}	4.01×10^{-4}	1.63×10^{-4}	$\mathrm{kg\,m^2}$
I_{yy}	5.98×10^{-3}	2.43×10^{-3}	$\mathrm{kg\,m^2}$
I_{zz}	1.27×10^{-2}	2.27×10^{-3}	$\mathrm{kg\,m^2}$
I_{xy}	0	0	$\mathrm{kg\,m^2}$
I_{xz}	0	0	$\mathrm{kg\,m^2}$
I_{yz}	0	0	$\mathrm{kg\,m^2}$

Note: Offset and inertias values are reported from the front and top of the fin in the body coordinate system

Table B.3: Non-structural mass properties for pods

	F0	F1	F2	F3	F4	Units
Ref. axis location (from leading edge)	60.9	60.9	60.9	60.9	60.9	% chord
Chord length	0.38	0.38	0.38	0.38	0.38	m
Pod height	0.23	0.23	0.23	0.23	0.23	m
Mass	3.09	0.989	0.959	1.265	1.324	kg
x_{cg}	-6.7×10^{-4}	-3.55×10^{-3}	5.95×10^{-3}	4.51×10^{-3}	4.51×10^{-3}	m
y_{cg}	5.4×10^{-2}	6.9×10^{-2}	7.0×10^{-2}	8.4×10^{-2}	8.7×10^{-2}	m
z_{cg}	-8.15×10^{-2}	-4.45×10^{-2}	-5.15×10^{-2}	-4.25×10^{-2}	-2.85×10^{-2}	m
I_{xx}	2.02×10^{-2}	1.34×10^{-2}	1.30×10^{-2}	1.73×10^{-2}	1.17×10^{-2}	$\mathrm{kg\,m^2}$
I_{yy}	3.27×10^{-3}	2.61×10^{-3}	1.56×10^{-3}	3.81×10^{-3}	3.61×10^{-3}	$\mathrm{kg\,m^2}$
I_{zz}	1.46×10^{-2}	1.35×10^{-2}	1.02×10^{-2}	8.91×10^{-3}	9.222×10^{-3}	$\mathrm{kg\,m^2}$
I_{xy}	2.32×10^{-4}	-1.21×10^{-3}	-1.21×10^{-3}	-1.21×10^{-3}	-1.21×10^{-3}	$\mathrm{kg\,m^2}$
I_{xz}	2.27×10^{-3}	1.06×10^{-5}	1.06×10^{-5}	1.06×10^{-5}	1.06×10^{-5}	$\mathrm{kg\,m^2}$
I_{yz}	4.5×10^{-4}	4.59×10^{-5}	4.59×10^{-5}	4.59×10^{-5}	4.59×10^{-5}	$\mathrm{kg\,m^2}$

Note: Offset and inertias values are reported from attachment point to the main wing in the body coordinate system

B.2 Aerodynamic properties of X-HALE

Table B.4: EMX-07 Polar

α (deg)	c_l	c_d	c_m	α (deg)	c_l	c_d	c_m
-20.0	-0.8596	0.26525	0.1092	0.0	0.1663	0.01252	-0.0019
-19.5	-0.8417	0.25902	0.1049	0.5	0.2178	0.0127	-0.0013
-19.0	-0.8237	0.25268	0.1006	1.0	0.2693	0.01291	-0.0007
-18.5	-0.8055	0.24632	0.0963	1.5	0.3208	0.01315	-0.0001
-18.0	-0.7878	0.24137	0.0905	2.0	0.3722	0.01345	0.0006
-17.5	-0.7768	0.23918	0.0834	2.5	0.4235	0.01381	0.0012
-16.5	-0.7369	0.21949	0.0795	3.0	0.4746	0.01426	0.0019
-16.0	-0.7198	0.21273	0.0763	3.5	0.5251	0.01472	0.0028
-15.5	-0.7033	0.206	0.0728	4.0	0.5763	0.01518	0.0034
-15.0	-0.6872	0.19916	0.0691	4.5	0.6271	0.01575	0.0041
-14.5	-0.672	0.1925	0.0648	5.0	0.6777	0.01636	0.0049
-14.0	-0.664	0.1879	0.0583	5.5	0.7283	0.01694	0.0057
-13.5	-0.6579	0.18248	0.052	6.0	0.7787	0.01763	0.0063
-13.0	-0.636	0.17059	0.0528	6.5	0.8287	0.01839	0.007
-12.5	-0.6194	0.16313	0.0517	7.0	0.8786	0.01892	0.0079
-12.0	-0.6077	0.15595	0.0486	7.5	0.9281	0.01902	0.0089
-11.5	-0.5996	0.14893	0.0439	8.0	0.9775	0.01936	0.0098
-11.0	-0.5996	0.14286	0.0359	8.5	1.0265	0.01958	0.0108
-10.5	-0.5976	0.13562	0.029	9.0	1.074	0.01958	0.0118
-10.0	-0.5804	0.12431	0.0327	9.5	1.1178	0.01998	0.0127
-9.5	-0.5688	0.11689	0.0314	10.0	1.1505	0.02236	0.0137
-9.0	-0.5625	0.10914	0.0278	10.5	1.1645	0.02685	0.0156
-8.5	-0.5609	0.10086	0.0212	11.0	1.1507	0.03305	0.0184
-8.0	-0.5706	0.09169	0.0072	11.5	1.1276	0.04114	0.0166
-7.5	-0.5712	0.08438	-0.0001	12.0	1.1235	0.04767	0.0157
-7.0	-0.5524	0.07363	0.0029	12.5	1.1226	0.05373	0.0155
-6.5	-0.4863	0.05421	-0.0078	13.0	1.1214	0.06005	0.0146
-6.0	-0.5053	0.05849	-0.0022	14.0	1.1063	0.07471	0.0126
-5.5	-0.4751	0.05056	-0.0036	14.5	1.0882	0.08442	0.0103
-5.0	-0.4363	0.04471	-0.0038	15.0	1.0646	0.09572	0.0054
-4.5	-0.3871	0.03851	-0.006	16.0	1.0439	0.11542	-0.0036
-4.0	-0.3411	0.03426	-0.0047	16.5	1.0132	0.13103	-0.0134
-3.5	-0.3054	0.03062	-0.0013	17.0	0.896	0.17653	-0.0392
-3.0	-0.265	0.02748	0.0015	17.5	0.6407	0.16909	-0.0193
-2.5	-0.1816	0.02116	0.0064	18.0	0.6377	0.17714	-0.022
-2.0	-0.1268	0.0188	0.0079	18.5	0.6356	0.1912	-0.0249
-1.5	-0.0737	0.01684	0.0095	19.0	0.6411	0.19982	-0.027
-1.0	-0.0228	0.01554	0.0111	19.5	0.6232	0.2049	-0.0331
-0.5	0.0202	0.01304	0.0135	20.0	0.6268	0.21093	-0.0359

Table B.5: NACA 0012 polar

α (deg)	c_l	c_d	c_m	α (deg)	c_l	c_d	c_m
-20.0	-0.5448	0.20725	0.0281	1.0	0.2073	0.01249	-0.0163
-19.5	-0.5367	0.20095	0.026	1.5	0.293	0.01218	-0.0212
-19.0	-0.5238	0.19584	0.0246	2.0	0.3609	0.01195	-0.0233
-18.5	-0.522	0.19115	0.0213	2.5	0.4041	0.01197	-0.0208
-18.0	-0.5238	0.18571	0.0179	3.0	0.4485	0.01215	-0.0187
-17.5	-0.5041	0.18034	0.0177	3.5	0.4926	0.01247	-0.0164
-17.0	-0.5167	0.17708	0.0129	4.0	0.5354	0.01292	-0.0141
-16.5	-0.5046	0.16992	0.0117	4.5	0.5768	0.01354	-0.0115
-16.0	-0.4886	0.16506	0.0111	5.0	0.6167	0.01434	-0.0087
-15.5	-0.5249	0.16297	0.0046	5.5	0.6549	0.01539	-0.0058
-15.0	-0.4872	0.15473	0.0065	6.0	0.692	0.01672	-0.0026
-14.5	-0.4845	0.15017	0.0048	6.5	0.729	0.01824	0.0006
-13.5	-0.4892	0.14049	0.0019	7.0	0.7669	0.01982	0.0036
-13.0	-0.5487	0.13265	-0.0069	7.5	0.8057	0.0216	0.0065
-12.5	-0.5129	0.12628	-0.0036	8.0	0.846	0.02388	0.009
-12.0	-0.5088	0.11978	-0.0046	8.5	0.8864	0.02681	0.0111
-11.0	-0.9913	0.03367	-0.0289	9.0	0.9263	0.02906	0.0137
-10.5	-0.9694	0.02954	-0.0261	9.5	0.9644	0.032	0.0156
-10.0	-0.9544	0.02474	-0.0218	10.0	0.9924	0.03644	0.0189
-9.5	-0.9272	0.02031	-0.0187	10.5	1.0121	0.04099	0.0227
-9.0	-0.8926	0.01754	-0.016	11.0	1.0318	0.04466	0.0259
-8.5	-0.8547	0.01456	-0.0137	11.5	1.0402	0.05009	0.029
-8.0	-0.8162	0.01209	-0.0112	12.0	1.0127	0.05758	0.034
-7.5	-0.7784	0.01007	-0.0084	12.5	0.9696	0.06542	0.0358
-7.0	-0.7424	0.0083	-0.0053	13.0	0.6569	0.14363	-0.0157
-6.5	-0.707	0.0068	-0.002	13.5	0.6913	0.14801	-0.0121
-6.0	-0.6921	0.01672	0.0027	14.0	0.6904	0.16007	-0.0156
-5.5	-0.655	0.0154	0.0058	14.5	0.6682	0.16872	-0.0262
-5.0	-0.6168	0.01434	0.0087	15.0	0.6845	0.17537	-0.0278
-4.5	-0.5769	0.01354	0.0115	15.5	0.5241	0.16262	-0.0044
-4.0	-0.5355	0.01292	0.0141	16.0	0.4874	0.16468	-0.011
-3.5	-0.4927	0.01247	0.0165	16.5	0.503	0.16951	-0.0115
-3.0	-0.4486	0.01215	0.0187	17.0	0.5161	0.17675	-0.0127
-2.5	-0.4042	0.01197	0.0208	17.5	0.5028	0.17992	-0.0175
-2.0	-0.361	0.01195	0.0233	18.0	0.5221	0.18523	-0.0178
-1.5	-0.2931	0.01218	0.0212	18.5	0.5211	0.19074	-0.021
-1.0	-0.2073	0.01249	0.0163	19.0	0.5224	0.19536	-0.0244
-0.5	-0.1107	0.01283	0.0096	19.5	0.5351	0.20044	-0.0258
0.0	0	0.01296	0	20.0	0.5443	0.20688	-0.0277
0.5	0.1107	0.01283	-0.0096				

Table B.6: Pod-without-fairing polar

β (deg)	c_y	c_d	c_r	β (deg)	c_y	c_d	c_r
-12.0	-0.2730	0.5373	-0.4168	0.5	0.0490	0.6098	0.0584
-11.5	-0.2527	0.5433	-0.3728	1.0	0.0594	0.6043	0.0614
-11.0	-0.2324	0.5492	-0.3288	1.5	0.0698	0.5988	0.0644
-10.5	-0.2122	0.5552	-0.2848	2.0	0.0801	0.5932	0.0675
-10.0	-0.1919	0.5611	-0.2408	2.5	0.0905	0.5877	0.0705
-9.5	-0.1716	0.5671	-0.1968	3.0	0.1009	0.5822	0.0735
-9.0	-0.1513	0.5731	-0.1528	3.5	0.1112	0.5767	0.0765
-8.5	-0.1311	0.5790	-0.1088	4.0	0.1216	0.5712	0.0796
-8.0	-0.1108	0.5850	-0.0648	4.5	0.1320	0.5656	0.0826
-7.5	-0.0905	0.5909	-0.0208	5.0	0.1424	0.5601	0.0856
-7.0	-0.0702	0.5969	0.0232	5.5	0.1495	0.5641	0.0936
-6.5	-0.0500	0.6028	0.0672	6.0	0.1567	0.5680	0.1017
-6.0	-0.0297	0.6088	0.1112	6.5	0.1638	0.5720	0.1097
-5.5	-0.0094	0.6147	0.1552	7.0	0.1710	0.5760	0.1177
-5.0	0.0109	0.6207	0.1993	7.5	0.1782	0.5800	0.1257
-4.5	0.0136	0.6201	0.1849	8.0	0.1853	0.5839	0.1338
-4.0	0.0164	0.6196	0.1705	8.5	0.1925	0.5879	0.1418
-3.5	0.0192	0.6191	0.1561	9.0	0.1996	0.5919	0.1498
-3.0	0.0220	0.6185	0.1417	9.5	0.2068	0.5958	0.1578
-2.5	0.0248	0.6180	0.1273	10.0	0.2140	0.5998	0.1659
-2.0	0.0275	0.6175	0.1129	10.5	0.2211	0.6038	0.1739
-1.5	0.0303	0.6169	0.0985	11.0	0.2283	0.6077	0.1819
-1.0	0.0331	0.6164	0.0841	11.5	0.2355	0.6117	0.1899
-0.5	0.0359	0.6159	0.0698	12.0	0.2426	0.6157	0.1980
0.0	0.0387	0.6153	0.0554				

Table B.7: Pod-with-fairing polar

β (deg)	c_y	c_d	c_r	β (deg)	c_y	c_d	c_r
-12.0	-0.2937	0.4155	-0.2348	0.5	0.0697	0.6033	0.0387
-11.5	-0.2781	0.4288	-0.2228	1.0	0.0811	0.6069	0.0479
-11.0	-0.2625	0.4421	-0.2108	1.5	0.0925	0.6104	0.0572
-10.5	-0.2468	0.4554	-0.1988	2.0	0.1039	0.6139	0.0664
-10.0	-0.2312	0.4687	-0.1868	2.5	0.1154	0.6175	0.0757
-9.5	-0.2156	0.4819	-0.1748	3.0	0.1268	0.6210	0.0849
-9.0	-0.2000	0.4952	-0.1628	3.5	0.1382	0.6246	0.0942
-8.5	-0.1844	0.5085	-0.1508	4.0	0.1496	0.6281	0.1035
-8.0	-0.1688	0.5218	-0.1388	4.5	0.1610	0.6316	0.1127
-7.5	-0.1531	0.5351	-0.1267	5.0	0.1724	0.6352	0.1220
-7.0	-0.1375	0.5484	-0.1147	5.5	0.1875	0.6294	0.1342
-6.5	-0.1219	0.5617	-0.1027	6.0	0.2026	0.6236	0.1464
-6.0	-0.1063	0.5749	-0.0907	6.5	0.2177	0.6178	0.1587
-5.5	-0.0907	0.5882	-0.0787	7.0	0.2328	0.6120	0.1709
-5.0	-0.0751	0.6015	-0.0667	7.5	0.2479	0.6063	0.1831
-4.5	-0.0617	0.6014	-0.0571	8.0	0.2630	0.6005	0.1953
-4.0	-0.0484	0.6012	-0.0475	8.5	0.2781	0.5947	0.2076
-3.5	-0.0350	0.6010	-0.0379	9.0	0.2932	0.5889	0.2198
-3.0	-0.0217	0.6008	-0.0283	9.5	0.3083	0.5831	0.2320
-2.5	-0.0084	0.6007	-0.0187	10.0	0.3234	0.5774	0.2442
-2.0	0.0050	0.6005	-0.0090	10.5	0.3385	0.5716	0.2565
-1.5	0.0183	0.6003	0.0006	11.0	0.3536	0.5658	0.2687
-1.0	0.0316	0.6001	0.0102	11.5	0.3687	0.5600	0.2809
-0.5	0.0450	0.6000	0.0198	12.0	0.3838	0.5542	0.2931
0.0	0.0583	0.5998	0.0294				

Table B.8: NACA 0010 polar

α (deg)	c_l	c_d	c_m	α (deg)	c_l	c_d	c_m
-10.0	-0.9852	0.03503	-0.0186	6.5	0.7157	0.01583	0.0023
-9.5	-0.9574	0.03136	-0.0157	7.0	0.7602	0.01714	0.0044
-9.0	-0.9252	0.02735	-0.0129	7.5	0.8011	0.01946	0.0068
-8.5	-0.8892	0.02314	-0.0106	8.0	0.846	0.02115	0.0087
-8.0	-0.8459	0.02115	-0.0087	8.5	0.8893	0.02315	0.0105
-7.5	-0.8011	0.01946	-0.0068	9.0	0.9253	0.02738	0.0128
-6.5	-0.7158	0.01583	-0.0023	9.5	0.9576	0.03136	0.0156
-6.0	-0.6733	0.01443	0.0001	10.0	0.9855	0.03504	0.0185
-5.5	-0.6297	0.01326	0.0024	10.5	0.9973	0.04052	0.022
-5.0	-0.5867	0.01203	0.0047	11.0	0.9832	0.04752	0.0263
-4.5	-0.5431	0.01093	0.0067	11.5	0.9297	0.05761	0.0273
-4.0	-0.4983	0.01003	0.0084	12.0	0.8395	0.08211	0.0074
-3.5	-0.4523	0.00933	0.01	14.0	0.4433	0.12375	-0.0025
-3.0	-0.4053	0.00878	0.0115	14.5	0.4503	0.12902	-0.0046
-2.5	-0.3569	0.00836	0.0128	15.0	0.4682	0.13351	-0.0049
-2.0	-0.3036	0.00804	0.0131	16.5	0.4722	0.15311	-0.0153
-1.0	-0.1461	0.00757	0.0046	17.0	0.4813	0.15852	-0.0177
-0.5	-0.0689	0.00749	0.0013	17.5	0.4977	0.16318	-0.0187
0.0	0	0.00747	0	18.5	0.5124	0.17676	-0.024
0.5	0.0689	0.00749	-0.0013	19.0	0.5149	0.18221	-0.0282
1.0	0.1461	0.00757	-0.0046	19.5	0.5246	0.18787	-0.0308
1.5	0.2245	0.00774	-0.0086	20.0	0.5356	0.19329	-0.033
2.0	0.3037	0.00804	-0.0132	20.5	0.5511	0.1995	-0.0342
2.5	0.3568	0.00836	-0.0127	21.0	0.5623	0.20775	-0.0364
3.0	0.4052	0.00877	-0.0114	21.5	0.5611	0.21221	-0.0412
3.5	0.4522	0.00933	-0.01	22.0	0.5694	0.21836	-0.0442
4.0	0.4982	0.01002	-0.0084	22.5	0.5787	0.22427	-0.0469
4.5	0.5429	0.01092	-0.0066	23.0	0.5887	0.22998	-0.0493
5.0	0.5866	0.01202	-0.0047	24.0	0.6088	0.24419	-0.0538
5.5	0.6296	0.01326	-0.0024	24.5	0.6131	0.24985	-0.0575
6.0	0.6732	0.01443	-0.0001	25.0	0.6205	0.25628	-0.0607

B.3 Servo Properties

The servo is connected to a potentiometer and oscilloscope to measure the dynamic response. As the servo rotates, the resistance of the potentiometer changes, which is then recorded as voltage on the oscilloscope. The readings are normalized and a 2^{nd} order transfer function fitted to it. The angular displacements are measured using a protractor. Linear regression is applied to find the relationship between PWM and deflection angle.

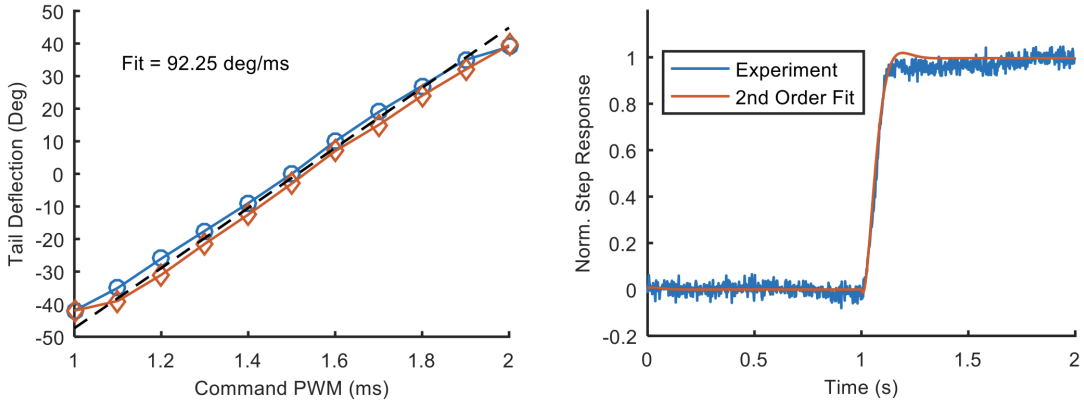

Figure B.1: Experimental characterization of tails

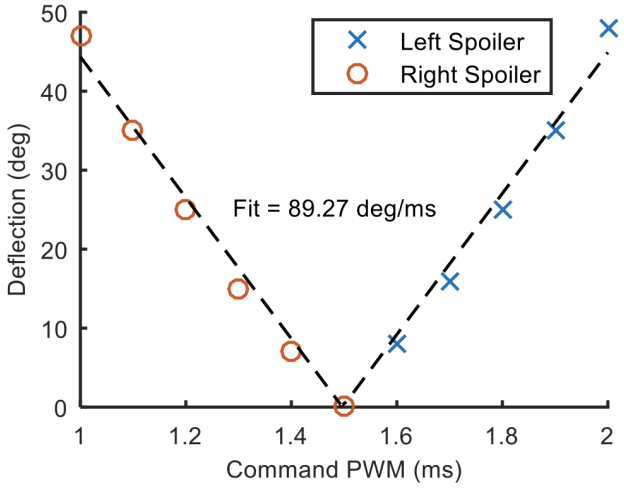

Figure B.2: Experimental characterization of roll spoiler

B.4 Propulsion Properties

An external PWM throttle command is given to the ESC. Within the ESC firmware, the time history motor speed is recorded at 10 Hz. A 2^{nd} order transfer function is fitted to the dynamic response. A linear regression is fitted for static relationship between throttle level and propeller speed.

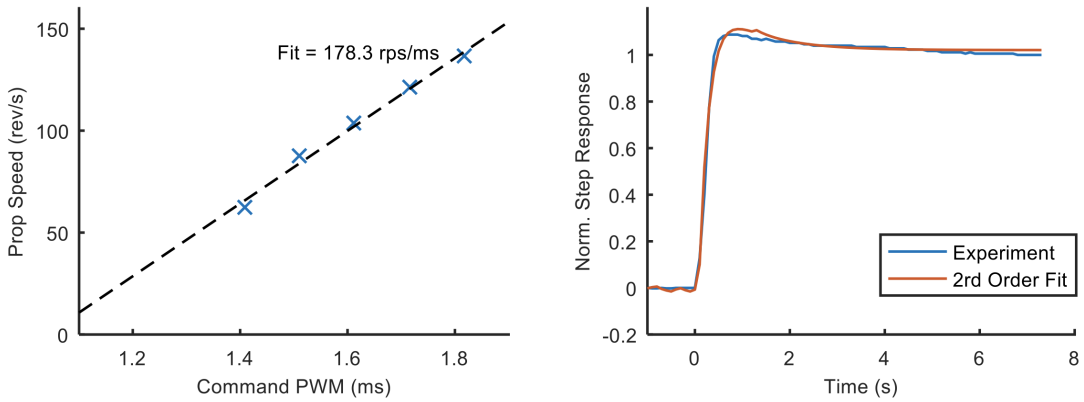

Figure B.3: Experimental characterization of propulsion unit

APPENDIX C

X-HALE UM/NAST Model Properties

Modeling details of X-HALE in UM/NAST is presented in this appendix. This model applies to both RRV-6B and ATV-6B as they are identical in geometry, structural and aerodynamic properties.

C.1 Keypoints Properties

Beam keypoints are extracted from X-HALE geometry. Member properties are decided based on judgement on relevant physics. State number are also provided for easy reference. Note that the numbering "radiates" from the body origin (e.g., the closest element will be at the start and furthest element at the end).

Table C.1: Beam reference axis key points (units: meters)

kpt	x	y	z	Description
1	0.0000	0.0000	0.0000	Body center
2	1.0000	0.0000	0.0000	Right tip of W2
3	−1.0000	0.0000	0.0000	Left tip of W1
4	2.0000	0.0000	0.0000	Right tip of W4
5	−2.0000	0.0000	0.0000	Left tip of W3
6	2.9848	0.0000	0.1737	Right tip of W6
7	−2.9848	0.0000	0.1737	Left tip of W5
8	1.0000	0.0000	−0.2010	Bottom of F2
9	−1.0000	0.0000	−0.2010	Bottom of F1
10	2.0000	0.0000	−0.2010	Bottom of F4
11	−2.0000	0.0000	−0.2010	Bottom of F3
12	1.0000	−0.697	0.0000	End of B2
13	0.7600	−0.697	0.0000	Left tip of T2
14	1.2400	−0.697	0.0000	Right tip of T2
15	−1.0000	−0.697	0.0000	End of B1
16	−0.7600	−0.697	0.0000	Left tip of T1
17	−1.2400	−0.697	0.0000	Right tip of T1
18	2.0000	−0.697	0.0000	End of B4
19	1.7600	−0.697	0.0000	Left tip of T4
20	2.2400	−0.697	0.0000	Right tip of T4
21	−2.0000	−0.697	0.0000	End of B3
22	−1.7600	−0.697	0.0000	Left tip of T3
23	−2.2400	−0.697	0.0000	Right tip of T3
24	0.0000	0.0000	−0.2010	Bottom of F0
25	0.0000	−0.944	0.0000	End of B0
26	0.0000	−0.944	0.2400	Top tip of T0
27	0.0000	−0.944	0.1480	Bottom tip of T0
28	2.0375	0.0000	0.006 616	Start of RS2
29	2.4721	0.0000	0.082 51	End of RS2
30	−2.0375	0.0000	0.006 616	Start of RS1
31	−2.4721	0.0000	0.082 51	End of RS1
32	0.0000	−0.944	−0.1400	End of V0
33	1.0000	−0.697	−0.1400	End of V2
34	−1.0000	−0.697	−0.1400	End of V1

C.2 FEM Properties

Table C.2: Member discretization

Member	Keypoints	No. of Elements	Flexible Element?	Lifting Surface?
F0	1, 24	1	No	Yes
B0	1, 25	1	No	No
T0,l	25, 26	1	No	Yes
T0,r	25, 27	1	No	Yes
W2	1, 2	2	Yes	Yes
F2	2, 8	1	No	Yes
B2	2,12	1	No	No
T2,l	12, 13	1	No	Yes
T2,r	12, 14	1	No	Yes
W4	2, 4	2	Yes	Yes
F4	4, 10	1	No	Yes
B4	4, 18	1	No	No
T4,l	18, 19	1	No	Yes
T4,r	18, 20	1	No	Yes
W6	4, 28, 29, 6	4	Yes	Yes
W1	1, 3	2	Yes	Yes
F1	3, 9	1	No	Yes
B1	3, 15	1	No	No
T1,r	15, 16	1	No	Yes
T1,l	15, 17	1	No	Yes
W3	3, 5	2	Yes	Yes
F3	5, 11	1	No	Yes
B3	5, 21	1	No	No
T3,r	21, 22	1	No	Yes
T3,l	21, 23	1	No	Yes
W5	5, 30, 31, 7	4	Yes	Yes
V0	25, 32	1	No	Yes
V2	12, 33	1	No	Yes
V1	15, 34	1	No	Yes

Note: Each tails are modeled as 2 separate members denoted by (l) and (r), but can only be actuated together

C.3 State Numbering

Table C.3: State numbering

Member	ε start	ε end	$\dot{\varepsilon}$ start	$\dot{\varepsilon}$ end	λ start	λ end
F0	–	–	–	–	142	147
B0	–	–	–	–	–	–
TL0	–	–	–	–	148	153
TR0	–	–	–	–	154	159
W2	1	8	65	72	160	171
F2	–	–	–	–	172	177
B2	–	–	–	–	–	–
TR2	–	–	–	–	178	183
TL2	–	–	–	–	184	189
W4	9	16	73	80	190	201
F4	–	–	–	–	202	207
B4	–	–	–	–	–	–
TL4	–	–	–	–	208	213
TR4	–	–	–	–	214	219
W6	17	32	81	96	220	243
W1	33	40	97	104	244	255
F1	–	–	–	–	256	261
B1	–	–	–	–	–	–
TL1	–	–	–	–	262	267
TR1	–	–	–	–	268	273
W3	41	48	105	112	274	285
F3	–	–	–	–	286	291
B3	–	–	–	–	–	–
TL3	–	–	–	–	292	297
TR3	–	–	–	–	298	303
W5	49	64	113	128	304	327
V0	–	–	–	–	328	333
V1	–	–	–	–	334	339
V2	–	–	–	–	340	345

APPENDIX D

X-HALE Hardware Drawings and Software Code

This appendix presents the hardware design drawings of the ATV-6B components (Section D.1 – D.2). The software modifications are shown in Section D.3. The hardware diagram for the antenna tracker is shown in Section D.4.

D.1 Custom PCB on ATV-6B

Custom designed PCBs are used to mount voltage regulators, resistors, and other required electronic conditioning circuits. In addition, they serve as wire breakout boards. The designs are shown in Fig. D.1 – Figs. D.4. Note that the ground planes are not shown for clarity.

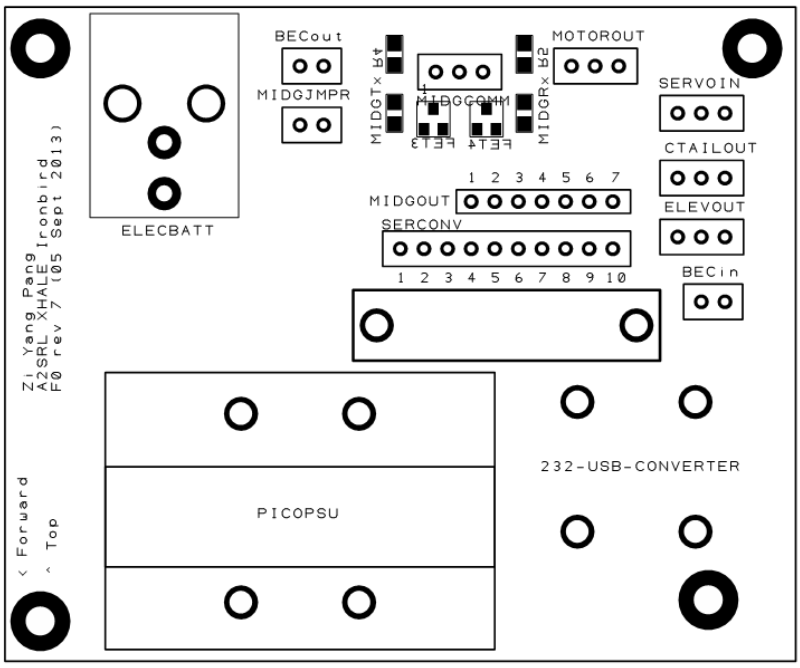

Figure D.1: PCB circuit design for F0

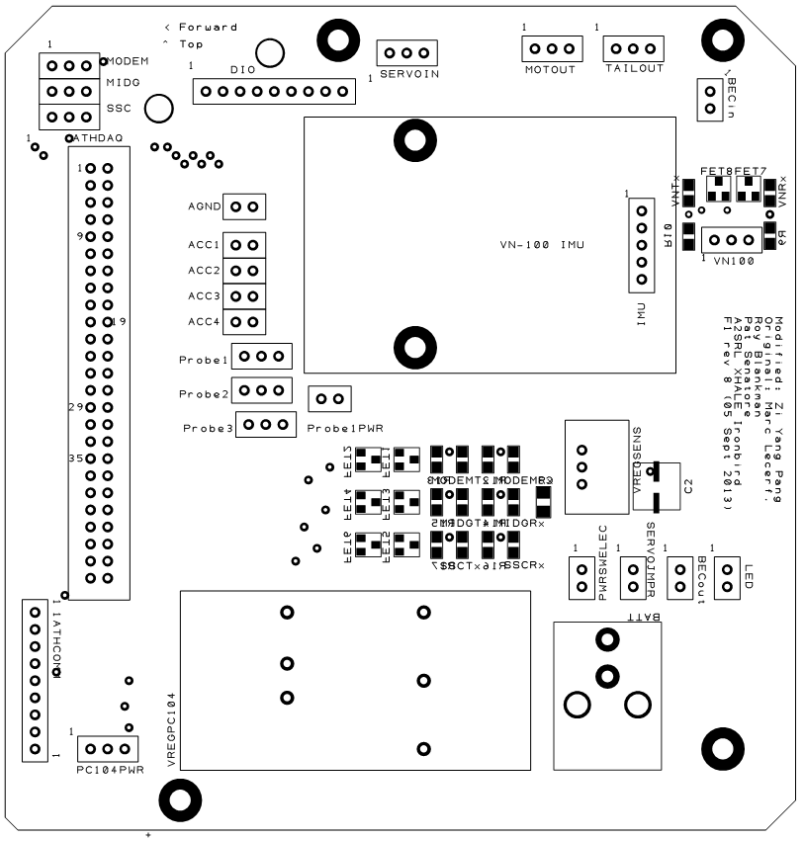

Figure D.2: PCB circuit design for F1

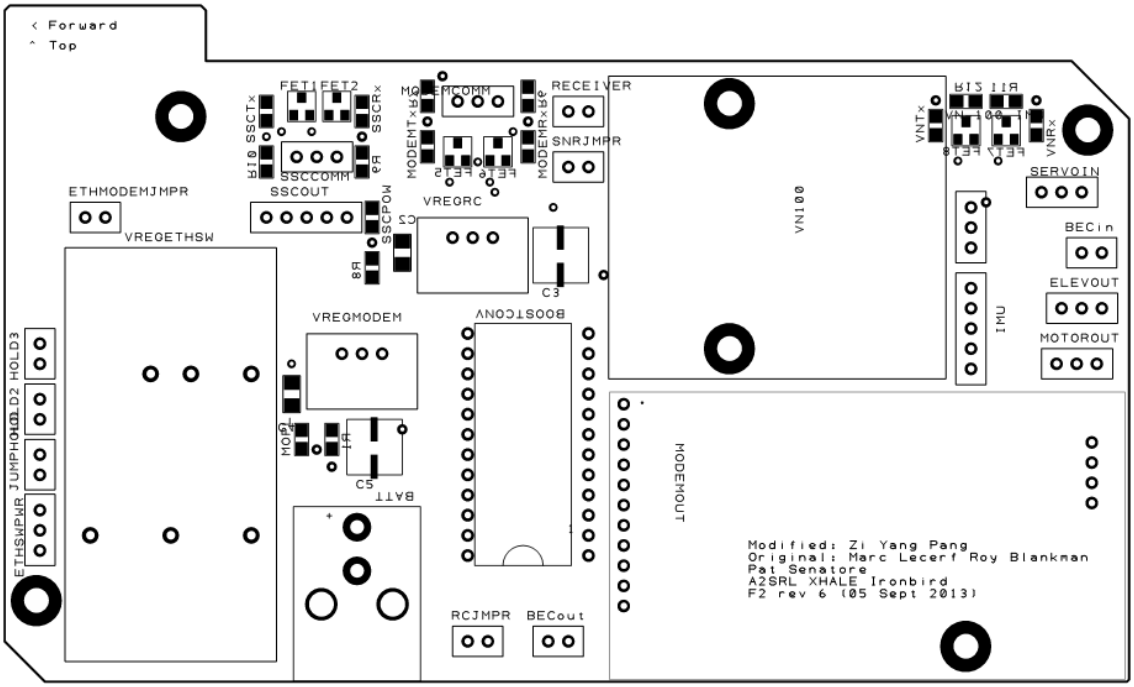

Figure D.3: PCB circuit design for F2

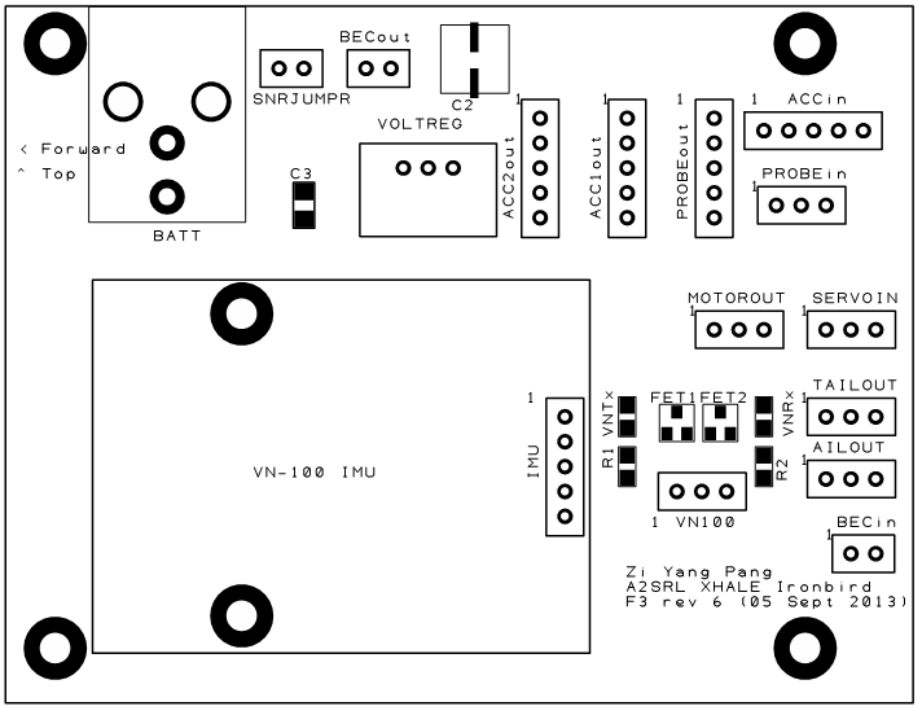

Figure D.4: PCB circuit design for F3 and F4 (identical)

D.2 Wiring diagram

The wiring diagram for ATV-6B is shown in Fig. D.5. ATV-6B is powered using two 14.8 V LiPo batteries in F3 and F4, denoted via power (PWR) and ground (GND) rail, indicated by the red and black wires, respectively. Green wires denote signal carrying wires (e.g., TX, RX, and PWM signals).

D.3 Pixhawk Modifications

The code modification based on Pixhawk release v1.4.1[1] is presented here. It is inspired from a similar pull request[2].

```
diff --git a/src/modules/fw_att_control/fw_att_control_main.cpp b/src/modules/fw_att_control/fw_att_control_main.
     cpp
index 83f50050f..e6a336d4e 100644
--- a/src/modules/fw_att_control/fw_att_control_main.cpp
+++ b/src/modules/fw_att_control/fw_att_control_main.cpp
@@ -215,6 +215,8 @@ private:

                float flaps_scale;                              /**< Scale factor for flaps */
                float flaperon_scale;                   /**< Scale factor for flaperons */
+
+               int disable_ground_check;               /**<This is to disable integrator reset for ground
     testing>*/

                int vtol_type;                                  /**< VTOL type: 0 = tailsitter, 1 = tiltrotor */

@@ -268,6 +270,8 @@ private:
                param_t flaperon_scale;

                param_t vtol_type;
+
+               param_t disable_ground_check;

        }               _parameter_handles;             /**< handles for interesting parameters */

@@ -458,6 +462,8 @@ FixedwingAttitudeControl::FixedwingAttitudeControl() :
        _parameter_handles.flaperon_scale = param_find("FW_FLAPERON_SCL");

        _parameter_handles.vtol_type = param_find("VT_TYPE");
+
+       _parameter_handles.disable_ground_check = param_find("NO_GNDCHCK");

        /* fetch initial parameter values */
        parameters_update();
@@ -549,6 +555,8 @@ FixedwingAttitudeControl::parameters_update()
        param_get(_parameter_handles.flaperon_scale, &_parameters.flaperon_scale);

        param_get(_parameter_handles.vtol_type, &_parameters.vtol_type);
+
+       param_get(_parameter_handles.disable_ground_check, &_parameters.disable_ground_check);

        /* pitch control parameters */
        _pitch_ctrl.set_time_constant(_parameters.p_tc);
@@ -972,7 +980,6 @@ FixedwingAttitudeControl::task_main()
                                float roll_sp = _parameters.rollsp_offset_rad;
                                float pitch_sp = _parameters.pitchsp_offset_rad;
                                float yaw_sp = 0.0f;
-                               float yaw_manual = 0.0f;
                                float throttle_sp = 0.0f;

                                // in STABILIZED mode we need to generate the attitude setpoint
@@ -982,7 +989,7 @@ FixedwingAttitudeControl::task_main()
```

[1] https://github.com/PX4/Firmware/releases/tag/v1.4.1

[2] https://github.com/PX4/Firmware/pull/5754

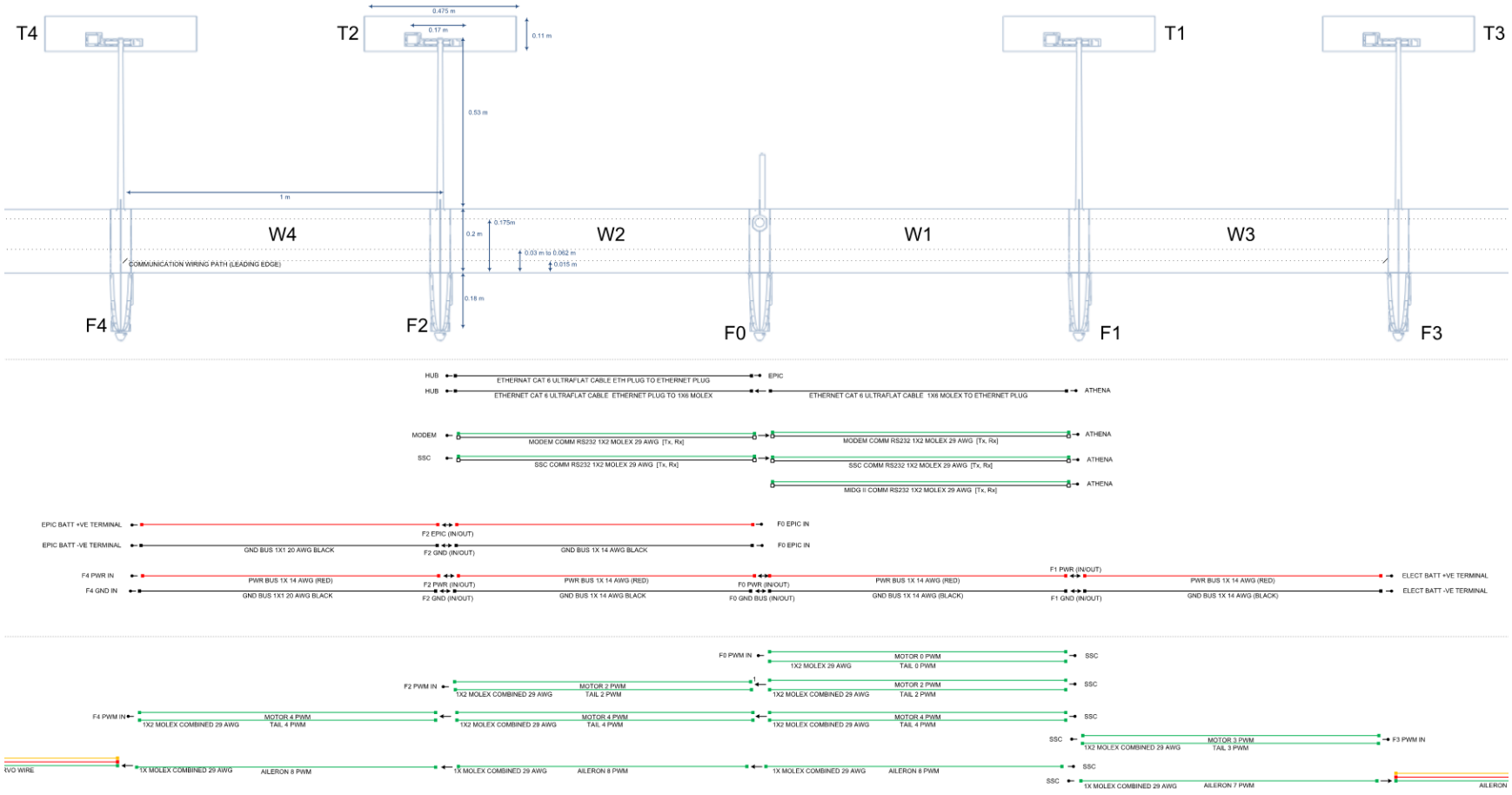

Figure D.5: Wiring diagram for X-HALE

```
                                        _att_sp.roll_body = math::constrain(_att_sp.roll_body, -_parameters.
                                             man_roll_max, _parameters.man_roll_max);
                                        _att_sp.pitch_body = -_manual.x * _parameters.man_pitch_max + _parameters
                                             .pitchsp_offset_rad;
                                        _att_sp.pitch_body = math::constrain(_att_sp.pitch_body, -_parameters.
                                             man_pitch_max, _parameters.man_pitch_max);
-                                       _att_sp.yaw_body = 0.0f;
+                                       _att_sp.yaw_body = _manual.r * _parameters.man_roll_max;
                                        _att_sp.thrust = _manual.z;
                                        int instance;
                                        orb_publish_auto(ORB_ID(vehicle_attitude_setpoint), &_attitude_sp_pub, &
                                             _att_sp, &instance, ORB_PRIO_DEFAULT);
@@ -993,11 +1000,6 @@ FixedwingAttitudeControl::task_main()
                                yaw_sp = _att_sp.yaw_body;
                                throttle_sp = _att_sp.thrust;

-                               /* allow manual yaw in manual modes */
-                               if (_vcontrol_mode.flag_control_manual_enabled) {
-                                       yaw_manual = _manual.r;
-                               }
-
                                /* reset integrals where needed */
                                if (_att_sp.roll_reset_integral) {
                                        _roll_ctrl.reset_integrator();
@@ -1013,7 +1015,9 @@ FixedwingAttitudeControl::task_main()
                                }

                                /* If the aircraft is on ground reset the integrators */
-                               if (_vehicle_land_detected.landed || _vehicle_status.is_rotary_wing) {
+                               /* also check if disable_ground_check is enabled */
+                               if ( (_vehicle_land_detected.landed || _vehicle_status.is_rotary_wing)
+                                       && !_parameters.disable_ground_check ) {
                                        _roll_ctrl.reset_integrator();
                                        _pitch_ctrl.reset_integrator();
                                        _yaw_ctrl.reset_integrator();
@@ -1112,9 +1116,6 @@ FixedwingAttitudeControl::task_main()

                                        _actuators.control[2] = (PX4_ISFINITE(yaw_u)) ? yaw_u + _parameters.
                                             trim_yaw : _parameters.trim_yaw;

-                                       /* add in manual rudder control */
-                                       _actuators.control[2] += yaw_manual;
-
                                        if (!PX4_ISFINITE(yaw_u)) {
                                                _yaw_ctrl.reset_integrator();
                                                _wheel_ctrl.reset_integrator();
@@ -1152,7 +1153,7 @@ FixedwingAttitudeControl::task_main()
                                 */
                                _rates_sp.roll = _roll_ctrl.get_desired_rate();
                                _rates_sp.pitch = _pitch_ctrl.get_desired_rate();
-                               _rates_sp.yaw = _yaw_ctrl.get_desired_rate();
+                               _rates_sp.yaw =_yaw_ctrl.get_desired_rate();

                                _rates_sp.timestamp = hrt_absolute_time();

@@ -1167,11 +1168,36 @@ FixedwingAttitudeControl::task_main()

                        } else {
                                /* manual/direct control */
-                               _actuators.control[actuator_controls_s::INDEX_ROLL] = _manual.y * _parameters.
     man_roll_scale + _parameters.trim_roll;
+
+                               // Important
+                               // =========
+                               // This section is customized for X-HALE.
+                               // Keith RC config:
+                               // x = pitch control, y = roll control, z = thrust control, r = yaw control
+                               // However, Keith uses differential thrust instead of roll spoilers for roll
     control
+                               // Therefore, the stick mapping (_manual.*) is swapped here.
+                               // Only in manual mode, y->differential thrust, r->rollspoilers
+                               // For stabilized mode and other AP control, the default configuration is used
+                               // which is r->differential thrust, y->rollspoilers
+                               //
+                               // Also note that _parameters.trim_* should be applied to the "original" control
     group
+                               // This is to allow QGC to set trim_param instead of trim tabs on the RC.
+                               // This is so that when changing to AP, even though the stick mapping has changed
     ,
+                               // the differntial thrust trim will still be applied.
+                               // If the aircraft is trimmed in RC, a constant offset will be set in the AP
     reference command
+                               // input and this is not what we want.
+
+                               _actuators.control[actuator_controls_s::INDEX_YAW] = _manual.y * _parameters.
     man_roll_scale + _parameters.trim_yaw;
                                _actuators.control[actuator_controls_s::INDEX_PITCH] = -_manual.x * _parameters.
                                     man_pitch_scale +
                                                _parameters.trim_pitch;
-                               _actuators.control[actuator_controls_s::INDEX_YAW] = _manual.r * _parameters.
     man_yaw_scale + _parameters.trim_yaw;
```

```
+                               _actuators.control[actuator_controls_s::INDEX_ROLL] = _manual.r * _parameters.
     man_yaw_scale + _parameters.trim_roll;
                                _actuators.control[actuator_controls_s::INDEX_THROTTLE] = _manual.z;
+
+                               //here we force a reset of the integator states to prevent problems when
     switching to other modes after MANUAL mode
+                               _roll_ctrl.reset_integrator();
+                               _pitch_ctrl.reset_integrator();
+                               _yaw_ctrl.reset_integrator();
+                               _wheel_ctrl.reset_integrator();
                        }

                        _actuators.control[actuator_controls_s::INDEX_FLAPS] = flaps_applied;
diff --git a/src/modules/fw_att_control/fw_att_control_params.c b/src/modules/fw_att_control/
     fw_att_control_params.c
index ebc6547c9..aa5b4c2cc 100644
--- a/src/modules/fw_att_control/fw_att_control_params.c
+++ b/src/modules/fw_att_control/fw_att_control_params.c
@@ -408,16 +408,16 @@ PARAM_DEFINE_FLOAT(FW_YCO_VMIN, 1000.0f);
  * Method used for yaw coordination
  *
  * The param value sets the method used to calculate the yaw rate
- * 0: open-loop zero lateral acceleration based on kinematic constraints
- * 1: closed-loop: try to reduce lateral acceleration to 0 by measuring the acceleration
- *
+ * 0: not used
+ * 1: yawrate autopilot active
+
  * @min 0
  * @max 1
  * @value 0 open-loop
  * @value 1 closed-loop
  * @group FW Attitude Control
  */
-PARAM_DEFINE_INT32(FW_YCO_METHOD, 0);
+PARAM_DEFINE_INT32(FW_YCO_METHOD, 1);

 /**
  * Roll Setpoint Offset
@@ -561,3 +561,18 @@ PARAM_DEFINE_FLOAT(FW_MAN_P_SC, 1.0f);
  * @group FW Attitude Control
  */
 PARAM_DEFINE_FLOAT(FW_MAN_Y_SC, 1.0f);
+
+/**
+ * Disable ground check flag
+ *
+ * Flag to disable integrator reset for ground testing purposes.
+ * DO NOT ENABLE in flight
+ *
+ * @min 0
+ * @max 1
+ * @value 0 integrator reset on ground, normal behaviour
+ * @value 1 integrator not reset on ground, lab testing behaviour
+ * @group FW Attitude Control
+ */
+PARAM_DEFINE_INT32(NO_GNDCHCK, 0);
+
diff --git a/src/modules/sensors/CMakeLists.txt b/src/modules/sensors/CMakeLists.txt
index a23f3c534..7f88e02e4 100644
--- a/src/modules/sensors/CMakeLists.txt
+++ b/src/modules/sensors/CMakeLists.txt
@@ -41,6 +41,7 @@ px4_add_module(
        SRCS
                sensors.cpp
                sensors_init.cpp
+               testcard_init.cpp

        DEPENDS
                platforms__common
diff --git a/src/modules/sensors/sensor_params.c b/src/modules/sensors/sensor_params.c
index d66fdb312..0646eee66 100644
--- a/src/modules/sensors/sensor_params.c
+++ b/src/modules/sensors/sensor_params.c
@@ -3188,3 +3188,154 @@ PARAM_DEFINE_INT32(PWM_AUX_MAX, 2000);
  * @group PWM Outputs
  */
 PARAM_DEFINE_INT32(PWM_AUX_DISARMED, 1000);
+
+/**
+ * System identification switch channel mapping
+ */
+PARAM_DEFINE_INT32(RC_MAP_SYSID_SW,0);
+
+/**
+ * Threshold for the system identification transition switch
+ */
+PARAM_DEFINE_FLOAT(RC_SYSID_TH, 0.25f);
+
+/**
+ * Define the amplitude of the sysID manoeuvre
+ */
```

```
+PARAM_DEFINE_FLOAT(SID_AMPLITUDE, 0.3f);
+
+/**
+ * Define the active time of the sysID manoeuvre
+ */
+PARAM_DEFINE_FLOAT(SID_ON_TIME, 3.0f);
+
+/**
+ * Define the trim time before the sysID manoeuvre
+ */
+PARAM_DEFINE_FLOAT(SID_TRIM_TIME_B, 1.0f);
+
+/**
+ * Define the trim time after the sysID manoeuvre
+ */
+PARAM_DEFINE_FLOAT(SID_TRIM_TIME_A, 1.0f);
+
+/* Define frequency for chirp
+ */
+PARAM_DEFINE_FLOAT(SID_START_FREQ, 0.1f);
+
+/**
+* Define the Stop frequency of the sysID manoeuvre
+*/
+PARAM_DEFINE_FLOAT(SID_STOP_FREQ, 5.0f);
+
+/**
+* Define the Manoever Input (Disabled/Testcard/Manual)
+*/
+PARAM_DEFINE_INT32(SID_MANOEUVRE, 0);
+
+/**
+* Define the Manoever Channel
++*/
+PARAM_DEFINE_INT32(SID_CHANNEL, 0);
+
+/**
+* Define the Testcard
++*/
+PARAM_DEFINE_INT32(SID_TESTCARD, 0);
\ No newline at end of file
diff --git a/src/modules/sensors/sensors.cpp b/src/modules/sensors/sensors.cpp
index 91096ed0d..e9545455b 100644
--- a/src/modules/sensors/sensors.cpp
+++ b/src/modules/sensors/sensors.cpp
@@ -64,6 +64,7 @@
 #include <errno.h>
 #include <math.h>
 #include <mathlib/mathlib.h>
+#include <vector>

 #include <drivers/drv_hrt.h>
 #include <drivers/drv_accel.h>
@@ -103,6 +104,7 @@
 #include <DevMgr.hpp>

 #include "sensors_init.h"
+#include "testcards.h"

 using namespace DriverFramework;

@@ -259,6 +261,7 @@ private:
        struct differential_pressure_s _diff_pres;
        struct airspeed_s _airspeed;
        struct rc_parameter_map_s _rc_parameter_map;
+       struct vehicle_control_mode_s vcontrol_mode;
        float _param_rc_values[rc_parameter_map_s::RC_PARAM_MAP_NCHAN]; /**< parameter values for RC control */

        math::Matrix<3, 3>      _board_rotation;        /**< rotation matrix for the orientation that the board
             is mounted */
@@ -318,7 +321,7 @@ private:
                int rc_map_param[rc_parameter_map_s::RC_PARAM_MAP_NCHAN];

                int rc_map_flightmode;
-
+
                int32_t rc_fails_thr;
                float rc_assist_th;
                float rc_auto_th;
@@ -352,6 +355,21 @@ private:

                float vibration_warning_threshold;

+               /* System identification additions */
+               int rc_map_sysid_sw;
+               float rc_sysid_th;
+               bool rc_sysid_inv;
+
+               int sid_manoeuvre;
+               int sid_testcard;
+               int sid_channel;
+               float sid_amplitude;
```

```
+               float sid_on_time;
+               float sid_trim_time_b;
+               float sid_trim_time_a;
+               float sid_start_freq;
+               float sid_stop_freq;
+
        }               _parameters;                    /**< local copies of interesting parameters */

        struct {
@@ -423,6 +441,20 @@ private:

                param_t vibe_thresh; /**< vibration threshold */

+               /* System identification */
+               param_t rc_map_sysid_sw;
+               param_t rc_sysid_th;
+
+               param_t sid_manoeuvre;
+               param_t sid_testcard;
+               param_t sid_channel;
+               param_t sid_amplitude;
+               param_t sid_on_time;
+               param_t sid_trim_time_b;
+               param_t sid_trim_time_a;
+               param_t sid_start_freq;
+               param_t sid_stop_freq;
+
        }               _parameter_handles;             /**< handles for interesting parameters */


@@ -552,6 +584,14 @@ private:
         * Main sensor collection task.
         */
        void            task_main();
+
+       /**
+        * Check if system id is performed
+        */
+       void check_sysid_manoeuvre(manual_control_setpoint_s *manual);
+
+       testcard_s* _testcards;
+       int _ncase;
 };

 namespace sensors
@@ -663,6 +703,20 @@ Sensors::Sensors() :
        _parameter_handles.rc_map_aux4 = param_find("RC_MAP_AUX4");
        _parameter_handles.rc_map_aux5 = param_find("RC_MAP_AUX5");

+       /* System identification */
+       _parameter_handles.rc_map_sysid_sw = param_find("RC_MAP_SYSID_SW");
+       _parameter_handles.rc_sysid_th = param_find("RC_SYSID_TH");
+
+       _parameter_handles.sid_manoeuvre = param_find("SID_MANOEUVRE");
+       _parameter_handles.sid_testcard = param_find("SID_TESTCARD");
+       _parameter_handles.sid_channel = param_find("SID_CHANNEL");
+       _parameter_handles.sid_amplitude = param_find("SID_AMPLITUDE");
+       _parameter_handles.sid_on_time = param_find("SID_ON_TIME");
+       _parameter_handles.sid_trim_time_b = param_find("SID_TRIM_TIME_B");
+       _parameter_handles.sid_trim_time_a = param_find("SID_TRIM_TIME_A");
+       _parameter_handles.sid_start_freq = param_find("SID_START_FREQ");
+       _parameter_handles.sid_stop_freq = param_find("SID_STOP_FREQ");
+
        /* RC to parameter mapping for changing parameters with RC */
        for (int i = 0; i < rc_parameter_map_s::RC_PARAM_MAP_NCHAN; i++) {
                char name[rc_parameter_map_s::PARAM_ID_LEN];
@@ -866,6 +919,11 @@ Sensors::parameters_update()
                PX4_WARN("%s", paramerr);
        }

+       /* System Identification */
+       if (param_get(_parameter_handles.rc_map_sysid_sw, &(_parameters.rc_map_sysid_sw)) != OK) {
+               PX4_WARN("%s", paramerr);
+       }
+
        param_get(_parameter_handles.rc_map_aux1, &(_parameters.rc_map_aux1));
        param_get(_parameter_handles.rc_map_aux2, &(_parameters.rc_map_aux2));
        param_get(_parameter_handles.rc_map_aux3, &(_parameters.rc_map_aux3));
@@ -910,6 +968,10 @@ Sensors::parameters_update()
        _parameters.rc_trans_inv = (_parameters.rc_trans_th < 0);
        _parameters.rc_trans_th = fabs(_parameters.rc_trans_th);

+       /* System Identication */
+       param_get(_parameter_handles.rc_sysid_th, &(_parameters.rc_sysid_th));
+       _parameters.rc_sysid_inv = (_parameters.rc_sysid_th < 0);
+       _parameters.rc_sysid_th = fabs(_parameters.rc_sysid_th);
        /* update RC function mappings */
        _rc.function[rc_channels_s::RC_CHANNELS_FUNCTION_THROTTLE] = _parameters.rc_map_throttle - 1;
        _rc.function[rc_channels_s::RC_CHANNELS_FUNCTION_ROLL] = _parameters.rc_map_roll - 1;
@@ -934,6 +996,8 @@ Sensors::parameters_update()
        _rc.function[rc_channels_s::RC_CHANNELS_FUNCTION_AUX_4] = _parameters.rc_map_aux4 - 1;
        _rc.function[rc_channels_s::RC_CHANNELS_FUNCTION_AUX_5] = _parameters.rc_map_aux5 - 1;
```

```
+       _rc.function[rc_channels_s::RC_CHANNELS_FUNCTION_SYSIDSWITCH] = _parameters.rc_map_sysid_sw - 1;
+
        for (int i = 0; i < rc_parameter_map_s::RC_PARAM_MAP_NCHAN; i++) {
                _rc.function[rc_channels_s::RC_CHANNELS_FUNCTION_PARAM_1 + i] = _parameters.rc_map_param[i] - 1;
        }
@@ -1028,6 +1092,44 @@ Sensors::parameters_update()

        _board_rotation = board_rotation_offset * _board_rotation;

+       //get system id
+       param_get(_parameter_handles.sid_manoeuvre, &(_parameters.sid_manoeuvre));
+       mavlink_and_console_log_info(&_mavlink_log_pub, "sid_manoeuver = #%i", _parameters.sid_manoeuvre);
+
+       if ((_parameters.sid_manoeuvre == 2) || (_parameters.sid_manoeuvre == 4)) //manual input from QGC
+       {
+               param_get(_parameter_handles.sid_amplitude, &(_parameters.sid_amplitude));
+               param_get(_parameter_handles.sid_channel, &(_parameters.sid_channel));
+               param_get(_parameter_handles.sid_on_time, &(_parameters.sid_on_time));
+               param_get(_parameter_handles.sid_trim_time_b, &(_parameters.sid_trim_time_b));
+               param_get(_parameter_handles.sid_trim_time_a, &(_parameters.sid_trim_time_a));
+               param_get(_parameter_handles.sid_start_freq, &(_parameters.sid_start_freq));
+               param_get(_parameter_handles.sid_stop_freq, &(_parameters.sid_stop_freq));
+       }
+
+       if ((_parameters.sid_manoeuvre == 1) || (_parameters.sid_manoeuvre == 3)) //load test cards
+       {
+               param_get(_parameter_handles.sid_testcard, &(_parameters.sid_testcard));
+
+               if(_parameters.sid_testcard >= 0 &&
+                       _parameters.sid_testcard < _ncase ) //check that input is valid
+               {
+                       _parameters.sid_channel = _testcards[_parameters.sid_testcard].channel;
+                       _parameters.sid_amplitude = _testcards[_parameters.sid_testcard].amp;
+                       _parameters.sid_on_time = _testcards[_parameters.sid_testcard].on_time;
+                       _parameters.sid_trim_time_b = _testcards[_parameters.sid_testcard].time_b;
+                       _parameters.sid_trim_time_a = _testcards[_parameters.sid_testcard].time_a;
+                       _parameters.sid_start_freq = _testcards[_parameters.sid_testcard].freq_start;
+                       _parameters.sid_stop_freq = _testcards[_parameters.sid_testcard].freq_stop;
+                       mavlink_and_console_log_info(&_mavlink_log_pub, "sid_testcard = %i", _parameters.
    sid_testcard);
+               }
+               else
+               {
+                       _parameters.sid_manoeuvre = 0;
+                       mavlink_and_console_log_info(&_mavlink_log_pub, "invalid sid_testcard #%i", _parameters.
    sid_testcard);
+               }
+       }
+
        /* update barometer qnh setting */
        param_get(_parameter_handles.baro_qnh, &(_parameters.baro_qnh));
        DevHandle h_baro;
         _parameters.rc_killswitch_th, _parameters.rc_killswitch_inv);
                        manual.transition_switch = get_rc_sw2pos_position(rc_channels_s::
                                RC_CHANNELS_FUNCTION_TRANSITION,
                                        _parameters.rc_trans_th, _parameters.rc_trans_inv);

+                       /* check for system identification */
+                       manual.sysid_switch = get_rc_sw2pos_position(rc_channels_s::
    RC_CHANNELS_FUNCTION_SYSIDSWITCH, _parameters.rc_sysid_th,
+                                                                                                       _parameters.
    rc_sysid_inv);
+
+                       check_sysid_manoeuvre(&manual);
+
                        /* publish manual_control_setpoint topic */
                        if (_manual_control_pub != nullptr) {
                                orb_publish(ORB_ID(manual_control_setpoint), _manual_control_pub, &manual);
@@ -2208,8 +2315,8 @@ Sensors::check_vibration()
        hrt_abstime cur_time = hrt_absolute_time();

        if (!_vibration_warning && (_gyro.voter.get_vibration_factor(cur_time) > _parameters.
            vibration_warning_threshold ||

+void
+Sensors::check_sysid_manoeuvre(manual_control_setpoint_s *manual)
+{
+       static bool is_doing_manoeuvre = false;
+       static uint64_t starting_time = 0;
+       static int _prev_sysid_sw_pos = manual_control_setpoint_s::SWITCH_POS_OFF;
+       static float signal_injection = 0.0f;
+
+       //check that system id is required
+       if (_parameters.sid_manoeuvre != 1 && _parameters.sid_manoeuvre != 2 &&
+               _parameters.sid_manoeuvre != 3 && _parameters.sid_manoeuvre != 4 )
+               return;
+
+       //check for toggle on from toggle off. This is to fix the starting time and start signal injection
+       if ((manual->sysid_switch == manual_control_setpoint_s::SWITCH_POS_ON)
+                       && (manual->sysid_switch != _prev_sysid_sw_pos)) {
+               is_doing_manoeuvre = true;
```

```
+               starting_time = hrt_absolute_time();
+               mavlink_and_console_log_info(&_mavlink_log_pub, "sid manoeuvre started");
+       }
+
+       //check if switch is held down. If switch is not held down for entire duration, signal interrupted
+       if ((manual->sysid_switch == manual_control_setpoint_s::SWITCH_POS_OFF)
+                       && (is_doing_manoeuvre)) {
+               is_doing_manoeuvre = false;
+               mavlink_and_console_log_info(&_mavlink_log_pub, "sid manoeuvre interrupted");
+       }
+
+       //check if switch is held down after system id duration has expired, signal finished
+       if ((manual->sysid_switch == manual_control_setpoint_s::SWITCH_POS_ON)
+                       && (!is_doing_manoeuvre)) {
+               mavlink_and_console_log_info(&_mavlink_log_pub, "sid manoeuvre finished");
+       }
+
+       //check if manual control is disabled
+       if (!vcontrol_mode.flag_control_manual_enabled) {
+               is_doing_manoeuvre = false;
+       }
+
+       if (is_doing_manoeuvre) {
+               float dt = static_cast<float>(hrt_absolute_time() - starting_time) / 1e6f; //calculate dt in
     seconds
+
+               if (dt > _parameters.sid_on_time + _parameters.sid_trim_time_b + _parameters.sid_trim_time_a) {
+                       is_doing_manoeuvre = false;
+
+               } else {
+
+                       if (dt < _parameters.sid_trim_time_b || dt > _parameters.sid_on_time + _parameters.
     sid_trim_time_b) {
+                               signal_injection = 0.0f;
+
+                               //signal input
+
+                       } else {
+
+                               if ((_parameters.sid_manoeuvre == 1) || (_parameters.sid_manoeuvre == 2))
+                               {
+                                       float progress = (dt - _parameters.sid_trim_time_b) / _parameters.
     sid_on_time;
+                                       float current_freq = _parameters.sid_start_freq + (_parameters.
    sid_stop_freq - _parameters.sid_start_freq) * progress;
+                                       signal_injection = _parameters.sid_amplitude * (float)sin(M_TWOPI_F *
    current_freq * (dt - _parameters.sid_trim_time_b));
+                               }
+                               if ((_parameters.sid_manoeuvre == 3 ) || (_parameters.sid_manoeuvre == 4))
+                               {
+                                       if (dt < (_parameters.sid_trim_time_b + 0.5f*_parameters.sid_on_time)) {
+                                               signal_injection = _parameters.sid_amplitude;
+                                       }
+                                       else {
+                                                       signal_injection = - _parameters.sid_amplitude;
+                                       }
+                               }
+                       }//end of the else for the signal input
+               }
+       }
+       else {
+               signal_injection = 0.0;
+       }
+
+       switch (_parameters.sid_channel)
+       {
+               case 1: //elevator excitation
+                       manual->x += signal_injection;
+                       break;
+               case 2: //differntal thrust excitation. Recall in manual mode y is mapped to INDEX::YAW
+                       manual->y += signal_injection;
+                       break;
+               case 3: //spoiler excitation. Recall in manual mode r is mapped to INDEX::ROLL
+                       manual->r += signal_injection;
+                       break;
+               case 4: //throttle excitation
+                       manual->z += signal_injection;
+                       break;
+               default:
+                       is_doing_manoeuvre = false;
+                       break;
+       }
+
+       _prev_sysid_sw_pos = manual->sysid_switch;
+}
+
 int
 Sensors::start()
 {
diff --git a/src/modules/sensors/testcard_init.cpp b/src/modules/sensors/testcard_init.cpp
new file mode 100644
index 000000000..4b0e599c9
```

```
--- /dev/null
+++ b/src/modules/sensors/testcard_init.cpp
@@ -0,0 +1,58 @@
+#include <stdio.h>
+#include <string.h>
+#include <unistd.h>
+#include <systemlib/err.h>
+
+#include <px4_defines.h>
+
+#include "testcards.h"
+
+#define MOUNTPOINT PX4_ROOTFSDIR"/fs/microsd/"
+static const char *log_root = MOUNTPOINT"testcards/testcards.txt";
+
+int testcard_init(testcard_s* &testcard_array, int& ncase)
+{
+       FILE            *fp;
+       char            line[120];
+
+       /* open the testcard file */
+       fp = fopen(log_root, "r");
+
+       if (fp == NULL) {
+               warnx("Unable to open testcard.txt");
+               return -1;
+       }
+
+       //skip first 3 lines of text file
+       fgets(line, sizeof(line), fp);
+       fgets(line, sizeof(line), fp);
+       fgets(line, sizeof(line), fp);
+
+       /* find number of testcards */
+       fgets(line, sizeof(line), fp);
+
+       /* allocate memory */
+       ncase = atoi(line);
+       testcard_array = new testcard_s[ncase];
+
+       for (int i = 0; i < ncase; i++) {
+
+               fgets(line, sizeof(line), fp);
+        int ret = sscanf(line, "%x %x %f %f %f %f %f %f",
+                                               &(testcard_array[i].id),
+                                               &(testcard_array[i].channel),
+                                               &(testcard_array[i].amp),
+                                               &(testcard_array[i].freq_start),
+                                               &(testcard_array[i].freq_stop),
+                                               &(testcard_array[i].on_time),
+                                               &(testcard_array[i].time_a),
+                                               &(testcard_array[i].time_b));
+
+               if (ret != 8) {
+                       warnx("Invalid number of parameters in test card");
+                       return -1;
+               }
+       }
+
+       return 1;
+}
\ No newline at end of file
diff --git a/src/modules/sensors/testcards.h b/src/modules/sensors/testcards.h
new file mode 100644
index 000000000..dde324ce2
--- /dev/null
+++ b/src/modules/sensors/testcards.h
@@ -0,0 +1,47 @@
+#pragma once
+
+struct testcard_s {
+       int id;
+       int channel;
+       float amp;
+       float freq_start;
+       float freq_stop;
+       float on_time;
+       float time_a;
+       float time_b;
+};
+
+int testcard_init(testcard_s* &testcard_array, int& ncase);
\ No newline at end of file
diff --git a/xhale additional/etc/extras.txt b/xhale additional/etc/extras.txt
new file mode 100644
index 000000000..f509351ed
--- /dev/null
+++ b/xhale additional/etc/extras.txt
@@ -0,0 +1,6 @@
+# set roll spoiler to have full deflection
+pwm min -c 67 -p 1000
+pwm max -c 67 -p 2000
```

```
+#set center tail to full deflection
+pwm min -c 8 -p 1030
+pwm max -c 8 -p 1880
\ No newline at end of file
```

D.4 Antenna Tracker

The antenna tracker computes the pointing orientation from the relative GPS coordinates of the vehicle and the antenna location (Fig. D.6). Position commands are passed to the servos to track the desired pointing orientation. The hardware connection diagram is shown in Fig. D.7.

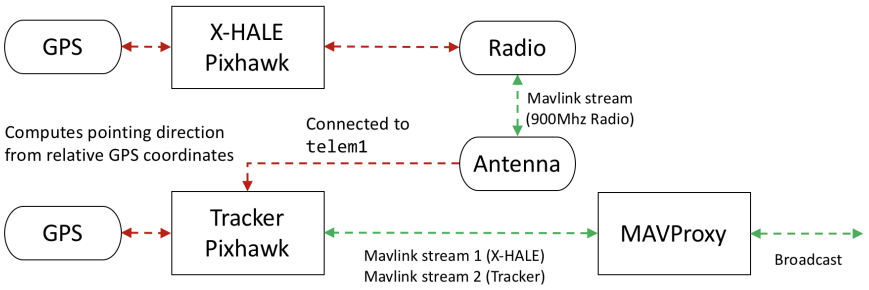

Figure D.6: Antenna tracker data flow

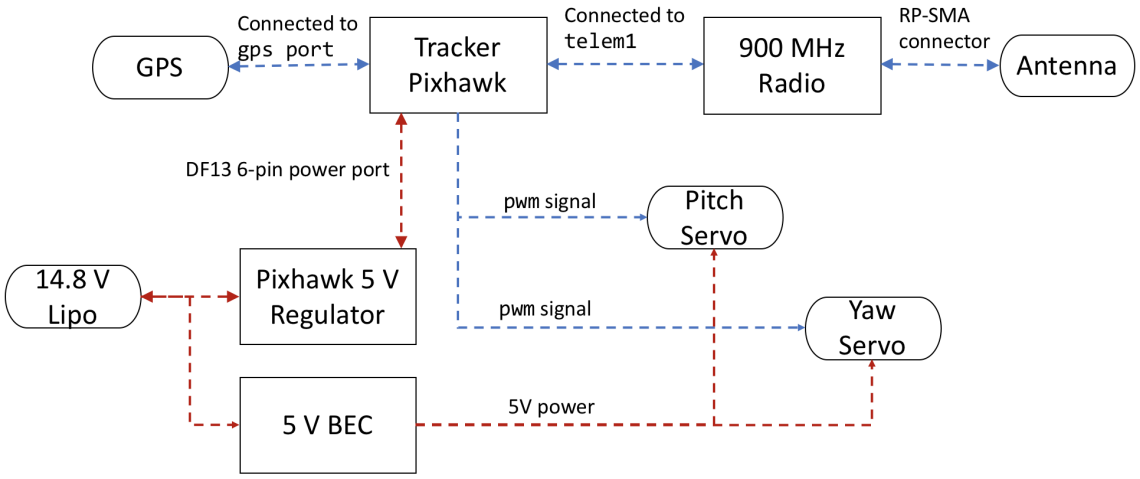

Figure D.7: Hardware connection diagram for antenna tracker