Intelligent Control Agent for Autonomous UAS

Suradet Tantrairatn

Department of Automatic Control and Systems Engineering

The University of Sheffield

Thesis

submitted to the University of Sheffield
in partial fulfilment of the requirements
for admission to the degree of

Doctor of Philosophy

July 2016

I would like to dedicate this thesis to my loving parents and wife for their unconditional love and support.

Acknowledgements

I am grateful to Prof Sandor Veres for introducing me to the topic of agent supervised autonomous control systems, which now forms the subject of this thesis containing my investigations and solutions to the problem of agent supervised learning of autopilots in unmanned aerial vehicles. I would like to thank for valuable discussions, suggestions and for technical advice given to me by Dr Owen McAree and Imjith Nagawahatte and postgraduate colleagues and staff at the Department of Automatic Control and Systems Engineering. Not the least I am grateful to my parents and my wife for all their support and belief in me throughout my postgraduate studies.

Abstract

A self reconfiguring autopilot system is presented, which is based on a rational agent framework that integrates decision making with abstractions of sensing and actions for next generation unmanned aerial vehicles. The objective of the new intelligent control system is to provide advanced capabilities of self-tuning control for a new UAS airframe or adaptation for an old UAS in the presence of failures in adverse flight conditions. High-level system performance is achieved through on-board dynamical monitoring and estimation associated with controller switching and tuning by the agent. The agent can handle an untuned autopilot or retune the autopilot when dynamical changes occur due to aerodynamic and on-board system changes. The system integrates dynamical modelling, hybrid adaptive control, model validation, flight condition diagnosis, control performance evaluation through software agent development. An important feature of the agent is its abstractions from real-time measurements and also its abstractions from model based on-board simulation. The agent, while tuning and supervising the autopilot, also performs real-time evaluations on the effects of its actions.

Contents

List of Figures

List of Tables

Part I

Background and Literature Review

Chapter 1

Introduction

Modern technology of unmanned aerial system (UAS) relies heavily on sophisticated control systems to enhance capability for reliability, survivability and safety. A conventional control system for a complicated system may come out with an unsatisfactory performance or instability, in the situation of malfunctions in actuators or other aircraft components and unexpected change of aircraft characteristics including de-icing or minor trim component missing. Additionally, the progressively complex mission requirements for high-level autonomy enforces UAS to re-tune its control capabilities during flight without external intervention. To achieve this, a control system with an advanced capability of self-tuning is needed to cope with different mission and environments, failures, and to address adverse flight conditions.

Advanced control tools are to support reconfiguration and achieve the aforementioned requirements. In practice, gain-scheduling approaches [121] have been outstanding solutions to flight control design. However, sometimes it has proved difficult for gain scheduling controllers to guarantee robust stability and good performance for the full flight envelope and under unexpected operational conditions [80]. Consequently, there have been a good number of papers published on reconfigurable control systems to overcome the deficiency of gain scheduling techniques [80], for example, dynamic inversion control (DIC) [115], back-stepping based on a Lyapunov function [78], feedback linearization with online parameter identification [137], sliding mode control methodology [50], and neural network techniques for direct adaptive tracking control [16, 71, 112]. Recently, reconfigurable flight

control laws for stability and control recovery of a damaged aircraft have been investigated such as $\mathcal{L}_1$ adaptive control [44] and hybrid adaptive flight control [96].

Adaptive control, however, is often limited by a fixed controller structure or switches between a small set of structural options. To enhance reliability, autonomous systems usually require additional adaptivity for operation, such as controlling damaged aircraft, flying in variety manoeuvres and manoeuvring in variable environments [134]. Therefore, issues of how to provide adaptability through switching and tuning of suitable controllers under uncertain dynamics are considered in this thesis.

To operate aircraft safely and to permit some autonomy, the US Federal Aviation Administration (FAA) and UK Civil Aviation Authority (CAA), are likely to permit the application of software of intelligence systems. Intelligent systems can potentially provide similar "analyze-and-decide" capabilities to human [13]. Therefore, software implementations based on agent-oriented approaches have been developed in combination with modern control techniques [36]. The advantage of these methodologies is transparent decision making that can be easily communicated to a human supervisor. Developments and designs of intelligent agents controlling sophisticated systems have been carried out in several works reported in [73, 128, 134]. These software techniques not only allow for portable design but also provide suitable schemes for practical implementation where decision making can also be formally verified [35].

There are several agent architectures available [62, 134, 136] to augment control systems of aircraft. A well-known agent architecture with considerable advantages, in term of its ability to combine reactivity with long-term planning, is the belief-desire-intention (BDI) agent approach, which parallels with decision making to follow intentions and pursue goals based on beliefs [14, 139]; not a negligible practical aspect when the agent needs to explain its decisions. Therefore, this thesis proposes a rational BDI agent system that integrates some decision-making rules with adaptive feedback controllers for fixed wing UAVs. The agent is developed in an extension of the AgentSpeak/Jason languages [14] using Natural Language Programming (NPLr) by sEnglish publication software and makes decisions using abstractions from flight data and from prediction of the antic-

ipated physical environment. The agent not only performs adaptive control of flight but also carries out aircraft dynamical identification in the frequency domain in a real-time manner. The agent can use a model validation approach in the frequency domain for flight monitoring and diagnosis of flight dynamics in order to achieve safety and robust improvement during flight.

This thesis aims to develop an intelligent software agent for control (ISAC) that is able to record aircraft data, analyse them, develop dynamical models of various complexity and finally propose robust controllers and flight path planning systems for the aircraft. The output of the result of this agent is presented in a format to be useful for other software agents who are parts of the mission management of the autonomous UAS. The coordinator agent makes decisions using abstractions from flight data and from prediction of the future physical environment. Additionally, this control system for small and low-cost UAVs is designed to work with a limited number of standard UAV actuators and sensors, including a pressure sensor for measuring airspeed and altitude, GPS, inertial measurement units (IMUs) and compass and primarily control surfaces, respectively.

The control agent is demonstrated in simulation on a benchmark performance of an Aerosonde UAV on MATLAB/SIMULINK environment. The intelligent control system for UAS uses rules-based-reasoning with an ability to abstract discrete events and evaluate the impact of its actions. This control system brings improvements upon traditional adaptive or reconfigurable control schemes by providing more adaptability via dynamical modelling and flight controller tuning under changing dynamics, including deficiencies in an initial controller and also in an event of system degradation due to accidental control surface damages.

1.1 Structure

The thesis is organized into four main parts:

Part I contains a literature review relevant to flight control systems already developed for aircraft and UAS in Chapter 2. Part I concludes with an overview of the problem to be invested in further chapters.

Part II encompasses mathematical material relating to flight dynamics, autopilot architecture, and agent theory. It does not contain any novel material, but serves as a foundation for material presented in Part III. In Chapter 3, the nonlinear flight dynamic model used for simulation and control design is presented. Chapter 4 presents an autopilot architecture including inner & outer flight control, guidance, and Dubins path planning methodologies. Chapter 5 details the design and the implementation of an agent architecture.

Part III constitutes the main contributions of the thesis, primarily developing the agent skills of state estimation, control action, model validation, and decision methods. Part III draws upon information presented in Part I and II, and is concluded with an analysis of the developed agent skills and a discourse of considerations relating to agent system design. Chapter 6 describes the process of how to apply the agent to UAV autopilot system. In Chapter 7, a new methodology to estimate the air flow angles and gravity acceleration is proposed. Chapter 8 is dedicated to a hybrid adaptive control technique that combines indirect and direct adaptive control schemes. Indirect adaptive control bases on a new parameter selection and estimation technique that modifies the orthogonal least square approach in the form of discrete Fourier transform to compute in the frequency domain. Model reference adaptive control is employed for direct adaptive control. Chapter 9 describes the real-time evaluation of UAV agent abstraction including on-line model validation in the frequency domain, flight trim condition monitoring, and control performance evaluation. Chapter 10 presents knowledge-based decision making for control agent on Jason. This development relies on natural language programming on sEnglish publication software. Finally, the thesis concludes with an outlook that discusses potential improvement of the method presented and possible future research development in the area of intelligent control agent systems for UAS.

Part IV contains appendix and other materials relating to the matter.

1.2 Publications During Work Undertaken

During the undertaking of the work presented within this thesis, key components have been presented at international conferences and in an internationally leading journal.

Journal publication:

- Tantrairatn, S. and Veres, S. M. (2016). An Intelligent Agent Supervised Reconfigurable Autopilot System, In *Journal of AIAA Guidance, Control and Dynamic*. (Submitted on 5th Dec 2015)

Refereed conference publications:

- Tantrairatn, S. and Veres, S. M. (2015). Onboard System Identification for Improved Flight Control of UAS, In *Proceeding of 8th IFAC Symposium on Robust Control Design-ROCOND'15.*, Bratislava, Slovak Republic, July 8-11 ,2015
- Tantrairatn, S. and Veres, S. M. (2015). A Rational Agent Framework for Adaptive Flight Control of UAVs, In *Proceeding of International Conference on Unmanned Aircraft System 2015-ICUAS'15.*, Denvor, CO, USA, June 9-12 ,2015

1.3 Contributions of the Thesis

Contributions of the thesis are related to state determination, model validation, control methodology and decision making. This control system will be implemented with small fixed-wing UAVs that have a limitation of the quantities of standard sensors and actuators. It means the aircraft will consist of only IMU, GPS, magnetic compass, pressure sensor and have only primary control surfaces.

State Estimation The state estimation proposed uses a technique to estimate air flow angles such as Angle-Of-Attack (AOA) and Sideslip Angle (SSA) of a small fixed-wing UAV and a value of gravitational acceleration during flight, using only kinematic relationships with an Extended Kalman Filter (EKF). This method does not need to know aerodynamic models, other aircraft parameters, airdata sensors, or any extra sensors.

Model Validation A model validation approach called frequency-dependent model validation is applied, which is calculated in real-time in recursive or iterative batch formats. This method is to check whether or not the aerodynamic models are still valid for inner loop NDI control of the autopilot system on the frequency domain. This technique relies on a discrete Fourier transform and a statistical hypothesis test on a residual as the difference between the measured output and the estimated model output in a format called normalised magnitude spectrum for each frequency with χ^2 distribution. This algorithm is a key to abstracting the continuous information into discrete abstractions for validating the aerodynamic model in the inner loop of the NDI controller. Furthermore, this technique can not just eliminate some essential variables such as angular accelerations in the calculation without any extra sensor or state determination but can also remove noise by calculating in an interesting frequency range.

Indirect Adaptive Control This new methodology is proposed to select and estimate the significant aerodynamic parameters of small fixed-wing UAV from flight data to improve the dynamical qualities of an indirect adaptive flight control system. Parameter estimation and selection of significant aerodynamic parameters are performed on linear regression model structures with forward Orthogonal Least Square (OLS) and Error Reduction Ratio (ERR) methods and calculated in the frequency domain. Additionally, this approach, based on recursive Fourier Transform, can also remove noise and some essential variables such as angular accelerations without any extra sensors or state determinations.

Human Readable Control Code for Desicion An issue discussed in this thesis is to find out how to synchronize or integrate reconfigurable control and all additional components in the overall structure of control system using adaptive control, model validation, flight monitoring, and control performance evaluation into knowledge based agent architecture. The rational agent supervises control system in order to provide advanced capabilities for self-tuning and adaptation in the presence of failure and adverse flight conditions. This process is to transfer the knowledge of control engineering from the literature review to agent reasoning with the aid of natural language programming (NLPr) in sEnglish Publication software and Agent Executive Matlab Toolbox. The sEnglish programming that represents meaningful sentences and conceptual structures allow us to demonstrate our work in NLPr for interpretation and adoption by development engineers.

Chapter 2

Flight Control Design for UAS - Literature Review

A literature review of methods published on the design of generic flight control systems for UAVs is presented in this chapter. Classical feedback control is introduced in Section 2.1. Modern control methods are reviewed in Section 2.2. Reconfigurable flight control approaches are considered in Section 2.3. The intelligent flight control issues are reported in Section 2.4. Finally a summary is provided and the approach taken in this thesis is discussed in Section 2.5.

2.1 Classical Control Designs Applied to UAS

The classical feedback control approach to flight, which is based on linear system theory, is to design a local controller for a specific trim condition to track the desired reference command in the presence of external disturbances. One of the most popular conventional control method is called Proportional-Integral-Derivative (PID) control design technique. PID has been extensively utilised and by now is well understood due to its simplicity and ease of implementation. This method depends on three controller gains to adjust and improve the system response: a proportional gain (K_P) is commonly employed to decrease the rise time of response or improve a sensitivity of system; an integral gain (K_I) is typically used to to improve the steady state response; and a derivative gain

(K_D) is useful for increasing damping in the closed-loop system, thereby leading to a more stable response. Therefore, most commercial autopilot systems, such as Cloud Cap Piccolo, MicroPilot MP series, Procerus Kestrel autopilot, UNAV 3500, Pixhawk and Ardupilot Mega use this traditional PID control approach. Tuning of PID controllers is carried out in various loops and layers manually, mostly based on rule of thumb method [47]. However, this PID control technique has limitation when it comes to obtaining optimal performance and robustness. It also requires a certain level of skill and experience from a user of a UAV. Also, it is sometimes difficult to tune the control loops under some conditions.

Other methods of controller design and tuning are based on an accurate mathematical model of flight dynamics. Controller development is carried out in order to reach more optimal performance according to the design requirements. Jodeh [56] presented a USAF Stability and Control Digital Datcom program which relies on estimating basic stability and control derivative coefficients in order to create a nonlinear simulation model in MATLAB, followed by simulations to determine a controller. The Athena Vortex Lattice (AVL) software, which is a program for the aerodynamic and flight-dynamic analysis of rigid aircraft of most configurations, generates the aircraft aerodynamic derivative and control coefficients for simulation in order to tune the Piccolo autopilot's gains [12, 43].

Furthermore, an iterative optimization technique [12] to tune gains of the pseudo-derivative feedback controller, which is similar algorithmic simplicity to the PID controller while in flight is verified with the simulation built on aerodynamic coefficient from AVL software and real aircraft of the Trainer 60. However, this method requires a runtime over 10 minutes to complete the controller tuning test for one flight condition. Moreover, it is hard to define an initial point to guarantee the global optimisation convergence and implement with the practical flight limit concerning long flight periods of maintaining stability under a constant condition.

2.2 Modern Control Theory Applied to UAS

Due to a wider flight envelope and varying aircraft configurations, not the least due to load variations, system non-linearities, coupling and model uncertain-

ties, the classical control approach requires adaptation in flight until the system performs satisfactory. In practice, this process demands time and money to be spend on controller tuning. Consequently, modern control methods based on linear-synthesis were proposed to deal with errors, which occurred from an imperfect approximation of the mathematical model and disturbances, without loss of control performance and stability from the trial and error technique of tuning controller gain by real-time flight observation requiring an expert operator. Moreover, this method uses the multiple input-output representation of the linear system to solve the coupling problem.

This modern control approach relies on several methods that are applicable to aircraft control design. For instance, Nelson [92] demonstrated how to apply pole placement and Linear Quadratic Regular (LQR) to autopilot systems. In addition, an eigenstructure assignment technique, which is an extension of pole placement, was suggested by Patt [103], Shapiro and Chung [110] for modern flight control systems.

All mathematical models of a physical system suffer from inaccuracies occurring from imprecise measurements or from the general failure to capture all appearances related to the dynamics of the considered system. Although it is possible to model a system accurately, the obtained result are often too complex to support for following analysis including the design of a controller. Consequently, methods to consider a simple model and a certain error between the simplified and more complex model are utilised with mathematical analysis such as linear algebra or optimal theory to solve the mentioned problem. Based on these principles, AlSwailem [4] and Patt [103] introduced H_∞ loop-shaping, based on H_∞ criterion optimisation, for use in flight control systems of aircraft and rotorcraft. Moreover, Paw [101] used H_∞ and μ controller synthesis methods, based on mathematical models, and applied it to some UAV autopilot systems.

2.3 Reconfigurable Control Systems for UAS

The conventional control approaches in Subsections 2.1 and 2.2 are able to design local controllers for specific trim/equilibrium conditions using linear system theory. However, the aircraft needs to perform under various operating condi-

tions. Consequently, a controller of a closed-loop system will change when the operating conditions vary. In addition, the increasing complexity of missions and requirements on high level of survivability dictate that future UAS work in a more sophisticated manner. Therefore various approaches have been proposed for reconfigurable flight control systems as follows.

2.3.1 Gain Scheduling

The local controllers that use the methods mentioned in Section 2.1 and 2.2 are designed in many different flight operating points for the whole flight envelope to guarantee a desired performance. Then gain-scheduling is used to provide a set of linear controllers. For example, Chumalee [30] proposed a gain-scheduled technique, namely linear parameter-varying control, that depends on robust H_∞ control theory to deal with uncertainties and nonlinearities of a UAV.

However, sometimes it proves difficult for gain-schedule controllers to guarantee robust stability and good performance for the full flight envelope and under unexpected operations conditions [80]. Therefore, control schemes, which have the capability of self-adjusting the control parameters in real-time, are proposed in the next section to improve the aircraft flying quality over the gain-scheduling technique.

2.3.2 Nonlinear Dynamic Inversion

The nonlinear dynamic inversion (NDI) technique relies on a nonlinear model of aircraft dynamics for the process of designing a control law. This nonlinear model, which is carefully inverted the input-output model in state-space form, is transformed to an equivalent linear system, where linear control theory can then be utilized for synthesis. In other words, NDI is a feedback linearisation method used to design nonlinear controllers for the full flight envelope without gain-scheduling.

Dynamic inversion is normally applied to an autopilot system with two time-scale loops assumption that separates the fast dynamical loop of angular rates from the slow dynamic loop of bank angles, angle of attack and angle of sideslip [69]. The dynamic inversion technique requires an accurate nonlinear mathemat-

ical model of aircraft for the full flight envelope which has to be estimated as discussed next.

2.3.3 Parameter Identification for Nonlinear Flight Control Systems

Aircraft parameter identification (APID) can be applied to estimate aircraft parameters in real-time in order to synthesise controllers on-line. Due to the requirement of real-time computation, there are a few APID methods that can be divided into two categories: time-domain based and frequency-domain based approaches. Time-domain based methodologies consist of recursive least squares (RLS), RLS with a forgetting factor, modified sequential least squares (MSLS), real time batch least squares, extended Kalman filter (EKF) [27, 32] and unscented Kalman filter [27]. The Fourier transform regression (FTR) [85–88] is typical for calculations in frequency domain.

Using the time-domain approach, a self-designing flight control system (SDFCS), which integrated model-following receding-horizon optimal control with on-line MSLS PID algorithms was tested with the VISTA/F-16 [54]. The MSLS is an iterative batch parameter estimation algorithm used with temporal and spatial constraints to identify parameters smoothly, without reducing the ability to track rapidly varying parameters during periods of low excitation and correlated inputs. Furthermore, the RLS algorithm was evaluated for real-time application in flight control and tested by Kamali [61]. This RLS technique, based on the equation error principle, employed a forgetting factor and a stabilising parameter in reconfigurable control and online stability margin estimation. Furthermore, Yu et al. [142] presented application of RLS identification algorithm for an indirect adaptive flight controller based on dynamic inversion with simulation of a six degree-of-freedom (DOF) nonlinear aircraft model.

In addition, Lombaerts et al. [76] proposed a joint aerodynamic model identification approach, sometimes called a two steps method, to identify a physical model of damaged aircraft in real time. This model was utilized for a model-based adaptive mechanism of non-linear dynamic inversion to reconfigure a flight controller in flight. This method consists of, first, aircraft state estimation and,

second, the aerodynamic model identification in order to attend the unfiltered aircraft state and unknown aerodynamic parameters equally in one single process.

Moreover, Lombaerts et al. [77] proposed an adaptive recursive orthogonal least squares algorithm to select a model structure and also estimate parameters for indirect fault-tolerant flight control. This procedure is an extension and modification of the classical recursive orthogonal least-squares procedure with new steps to produce three crucially different aspects. Firstly, subset selection stopping criterion using the normalize residual sum of square is replaced with other criterion such as Schwarz criterion or Bayesian information criterion to prevent over-fitting. Secondly, an additional condition to check the necessary to extend the size of the sliding time window is applied to the procedure to handle with collinearity problem between the several regressor. Finally, more criterions are added in the last routine step of the process to check changes in the dynamics of the actual system by comparing to the previous situation.

The well-known frequency-domain based approach, called Fourier transform regression (FTR), has been widely implemented for online APID and consequently applied to reconfigurable flight control systems [29]. For instance, this algorithm was used to program an intelligent flight control system (IFCS) [48, 84]. This method is based on equation error method in frequency domain and it can be formulated as a standard LS regression problem with complex data in recursive form of the discrete Fourier transform to estimate non-dimensional stability and control derivatives in real-time [86, 88]. These methods, presented in [26, 34, 100, 114] have been successfully tested in flight simulation and real flight data for real-time parameter identification under nominal and structurally damaged operations.

There are some publications, which can be compared with APID for reconfigurable flight control [60, 116, 117]. Both time-domain based and frequency-domain based techniques exist, where RLS and recursive FTR are popular as applied to reconfigurable flight control to achieve similar results in terms of their capability of estimating aircraft aerodynamic parameters. RLS is normally simpler and requires less computation time than recursive FTR. Also, the convergence of estimates using RLS is less oscillatory than using recursive FTR during

the initial phase of estimation. However, RLS techniques rely on the use of digital filters that produce time lags to remove the unwanted frequency bands. Otherwise, there are some advantages of recursive FTR methods, which topple RLS approach in practice: 1) ability to suppress noise effects and 2) state reduction for aircraft parameter estimation. These advantages will be further elaborated in section 8.1.

2.3.4 Neural Network Technology for Reconfiguration Control System

Neural network has been found in many successful applications of the aerospace industry such as modelling [3, 89, 106], system identification [49, 105] and control [17, 18, 58] due to its capabilities in non-linear mapping, learning, and adaptation. Especially in control application, neural networks can be employed to design adaptive control laws that can deal with uncertainties and non-linearities in system dynamics and the environment without the requirement of modelling and system identification of the aircraft platform.

Neural network techniques have been successfully implemented to augment approximated model dynamic inversion controllers for various fixed-wing and rotary wing aircraft by Calise el at [18–20]. In addition, neural network control techniques has been applied to various aircraft [17, 111] and small UAVs [52, 58]. Furthermore, the above mentioned neural network controllers have been extended by integrating pseudo-control hedging (PCH) for attitude control of X-33 [57].

As mentioned, adaptive control based on neural networks has been employed with some success in many applications. However an issue of high-gain control due to fast adaptation is critical. Fast adaptation is required to improve tracking performance in the presence of a large source of uncertainties such as aircraft structural damage that can lead to large changes in aerodynamic derivatives and physical properties. In these cases tracking error can decrease rapidly through using a large adaptive gain or learning rate of adaptive control. However fast adaptation can result in high-frequency oscillations that could adversely affect the robustness and stability of adaptive control law, especially model reference adaptive control (MRAC) law [97]. Therefore, issues of balance between stability

and adaptation are significantly considered.

Therefore, to increase stability robustness of MRAC by fast adaptation, robust adaptive laws have been widely developed. Robust adaptation effectively introduces an additional term and mechanism into an adaptive law so as to ensure that the adapted weights remain bounded. Two well-known robust modification adaptive laws, which have been utilized extensively in adaptive control, are the σ modification [51] and ε modification [90] by adding damping term into the weight update law. Furthermore, an adaptive law based on an optimal control theory that minimizes the $\mathcal{L}_2$ norm of the tracking error bounded away from some lower bound is introduced by [97] and addressed that optimal control modification (OCM) adaptive law.

2.3.5 Model Predictive Control

Model predictive control (MPC), sometimes called receding horizon control (RHC), is a model-based optimal control methodology based on numerical optimization. The control signals and future plant responses are calculated using the predicted outputs of the systems relying on a current system model and an assumed control sequence. The optimal control sequence is determined by optimizing a cost function that penalizes the deviation between the predicted outputs and the desired outputs [68].

This methodology has also been applied to flight control systems. For instance, Kale and Chipperfield introduced formulations and experimental evaluations of various MPC schemes applied to a full flight envelope with a non-linear model of a fighter aircraft in [59]. Furthermore, a nonlinear MPC technique that can be employed to a general nonlinear plant by using Taylor expansion of the plant output and control in a multi-input-multi-output (MIMO) setting was presented by Sleger et al. [113] for two UAV systems including parafoil and glider aircraft. In addition, Richard et al. [108] proposed an adaptive receding horizon optimal controller to implement and evaluate with Calspan Learjet 3 in-flight simulator. This single-input-single-output (SISO) controller combined receding horizon optimal strategy with MSLS to aid with estimating the online model parameter.

2.3.6 New Trends in UAS Control

Recently, the three most popular approaches to UAS control included Adaptive Backstepping Control [79, 118], $\mathcal{L}_1$ Adaptive Control [41, 83, 141], and Hybrid Adaptive Control [94, 98, 99] have been investigated to modern flight control design in order to achieve better stability guarantees and improve flying performance in case of significant changes such as structural damage of aircraft during flight.

2.3.6.1 Adaptive Backstepping Control

Back-stepping control is a recursive design method, which is mostly based on a Lyapunov function construction to synthesize a controller. This control theory can overcome the problem of nonlinear systems with parametric uncertainties and wider range of controlled system. Moreover in [118], the author proposed on-line model parameter estimation in combination with nonlinear backstepping control design, called adaptive backstepping control, in order to deal with unexpected faults or changes during flight.

2.3.6.2 $\mathcal{L}_1$ Adaptive Control

To begin with, $\mathcal{L}_1$ adaptive control, which consists of a state predictor, fast adaptive laws and a control law with low pass filter elements (as shown in Figure 2.1), adapts fast and robustly, leading to desired transient tracking. Its low pass filter $C(s)$ can aid to prevent high-frequency oscillations in the control signal of the reference model of other adaptive techniques. This control scheme depends on system identification for the state predictor, and then the adaptive law will adjust control parameters from the error signal between estimated states of the system and the predictor.

There are some publications which investigated the $\mathcal{L}_1$ adaptive architecture for aircraft. For instance, You el at. [141] proposed a technique to employ $\mathcal{L}_1$ adaptive control to augment flight control system based on the linear baseline controller in the presence of flight regime changes during take-off and landing in order to maintain the robustness of the control performance without the need for conventional gain scheduling. Furthermore, Gregory el at. [44] evaluated the

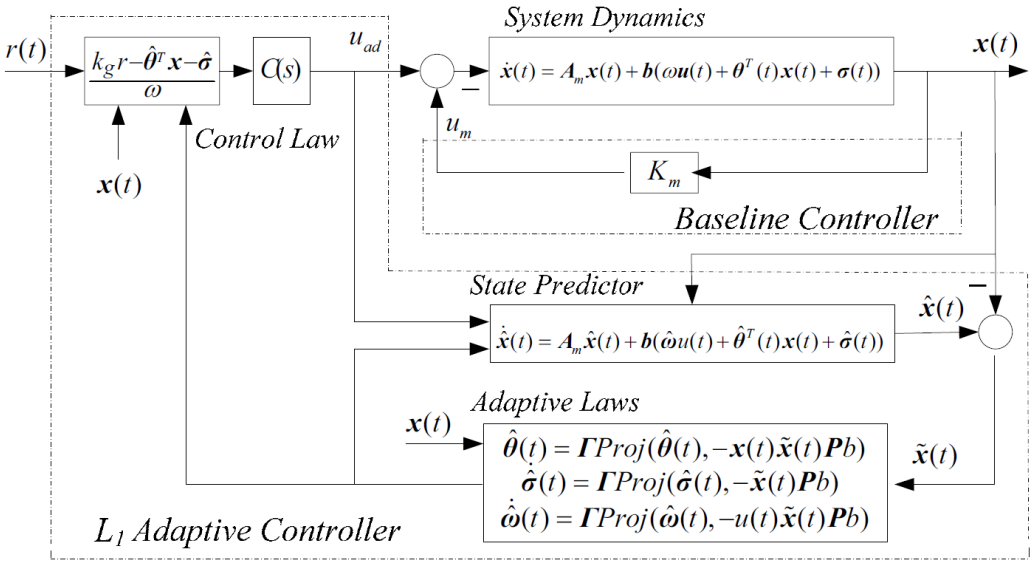


Figure 2.1: Diagram of $\mathcal{L}_1$ Flight Adaptive Control

$\mathcal{L}_1$ adaptive control law with rapid prototyping and testing of control laws in the Airborne Subscale Transport Aircraft Research system at the NASA Langley Research Center in nominal and damaged aircraft including rudder missing, left outboard trailing edge flap missing, loss of outboard (approx 25% semispan) left wing tip, loss of entire elevator from left stabilizer, and loss of entire left stabilizer.

2.3.6.3 Hybrid Adaptive Control

Hybrid adaptive control combines direct and indirect adaptive control with an NDI based flight control architecture as shown in Figure 2.2. The indirect adaptive control part is utilized to compensate the aircraft parameters (including inertial and aerodynamic terms) of model inversion control with parameter estimation techniques [21, 75] to reduce the model inversion error. This part directly leads to the decrease of a tracking error. Then, any remaining tracking errors can be further reduced by the direct adaptive control part that could be manipulated by a neural network. Wherewith the direct adaptive controller can produce a reinforced reference command signal that depends on the remaining tracking error. Because the direct adaptive controller only demands to adapt to residual uncertainty, its adaptive gain can be reduced to improve stability robustness [93].

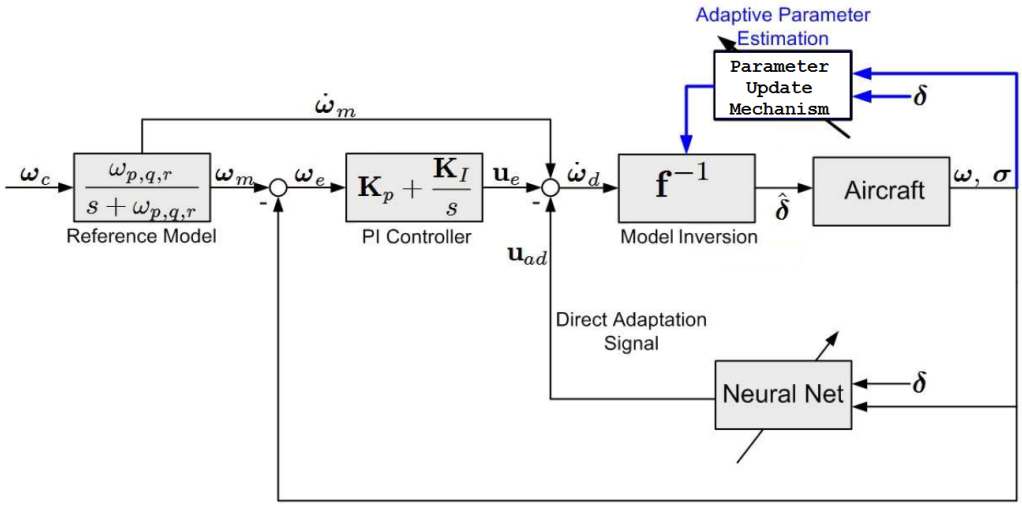


Figure 2.2: Diagram of Hybrid Flight Adaptive Control Approach

Recent studies of aircraft operating in off-nominal flight conditions under damages and or failures were presented in [96, 99, 144]. Nguyen [93] investigated hybrid adaptive control methodology for stability recovery. He proposed a hybrid adaptive control based on a model inversion flight control architecture for damaged aircraft. Two indirect adaptive laws have been examined: 1) a Lyapunov-based indirect adaptive law with neural net based model augmentation, and 2) an RLS indirect adaptive law for online parameter estimation and one neural net direct adaptive control augmentation with the e-modification adaptive weight update law. Consequently, hybrid adaptive flight control, especially with RLS indirect adaptive law, is able to improve the control performance potentially when operating in adverse events such as with damages and or failures.

A comparison study of several adaptive control strategies, including direct adaptive control strategy, indirect adaptive control strategy, combined direct and indirect (hybrid) adaptive control strategy, $\mathcal{L}_1$ adaptive control, output error feedback strategy, and combinations of strategies was presented by Boskovic and Knoebel [15]. They utilised Genetic Algorithm to find out the best gains including PID controllers and learning rates of the adaptive control methods to minimise adaptive control performance metrics criterion (a weighted sum of transient performance) under constraints by running a large number of simulation. Fur-

thermore, these adaptive control algorithms were implemented and evaluated on the tail-sitter UAV and F/A-18 simulation in cases of nominal, loss-of-effective actuator failure, cross-coupling effects and time delay. The results demonstrated that the hybrid adaptive controller with output error feedback can outclass other algorithms in many cases. For example, in a case of tail-sitter UAV with the time delay of 0.08 second and actuator failure of 70 percent effective, hybrid adaptive controller outperforms direct adaptive and $\mathcal{L}_1$ adaptive controllers with a minimum fitness (i.e. the integral of square of the tracking and input errors, norm of the tracking and input errors, number of oscillations, etc). Moreover, the results illustrated that hybrid adaptive controllers are robust to actuator failure, cross-coupling effect, and with time delay and combinations of direct and indirect adaptive algorithms can achieve excellent overall performance in term of both transient and steady state response due to the integration of advantages of each algorithm.

2.4 Intelligent Autonomous Flight Control System

Autonomous control systems are defined as systems that are "designed to perform well under significant uncertainties in the system and environment for extended periods of time, and they must be able to compensate for significant system failures without external intervention"in [7]. Techniques from the field of artificial intelligence (AI) are applied to such control systems in order to achieve autonomy where this control system is called "**Intelligent Autonomous Control System**"[6]. Such control systems are developed from conventional control systems by additional intelligent components which are able to perform a number of interdisciplinary functions such as compensated control, identification, estimation, communication theory, computer science, especially artificial intelligence, and operation research in order to achieve autonomy.

For instance, Stengel [120] presented the concept of an intelligent flight control system with three categories of control functions including declarative, procedural and reflexive functions. Declarative actions relate to decision making, providing

models or system monitoring, goal planning, and system/scenario identification in the outer loop of the control system. Reflex actions are performed by the control system's inner loops which are relevant to control and estimation based compensation. Procedural actions involve skilled behaviours and responsible in guidance, navigation, and adaptation in an intermediate level.

2.4.1 Fault-Tolerant Flight Control Systems

Furthermore, one common kind of the intelligent control systems, namely Fault Tolerant Control System (FTCS), is designed to improve reliability, maintainability, and survivability with its capability of tolerating potential faults in the system [119]. An overview of FTCS usually consists of at least two additional essential components related to fault detection and diagnosis (FDD) schemes and reconfigurable control techniques. An FDD scheme consists of three tasks: (1) *fault detection* indicates that something is wrong in the system, i.e., the occurrence of a fault and the time of the fault occurrence; (2) *fault isolation* determines the location and the type of the fault (which component has failed); (3) *fault identification* determines the magnitude of the fault [145]. Fault isolation and identification are jointly called *fault diagonosis*.

Additionally, the FDD approaches can be typically classified into two categories: (1) data-based (model-free) and (2) model-based schemes. A model-free approach mostly depends on data to analyse without model requirement according to residual evaluation strategies including threshold test on the residuals, statistical methods (Hypothesis test on whiteness), methods based on fuzzy logic symptom assessment and neural network pattern classification. On the other hand, a model-based method relies on a mathematical model to implement FDD in real-time with state estimation, parameter estimation, or combination technique [145]. Many methodologies have been proposed to address model-based FDD. An overview can be found in survey papers [42, 53, 130–132] and a survey on aerospace systems [81].

A model-based fault diagnosis method [81] is typically worked out by incorporating a residual generator and a residual evaluation strategy to detect whether faults have occurred by providing boolean decisions as illustrated in Fig. 2.3.

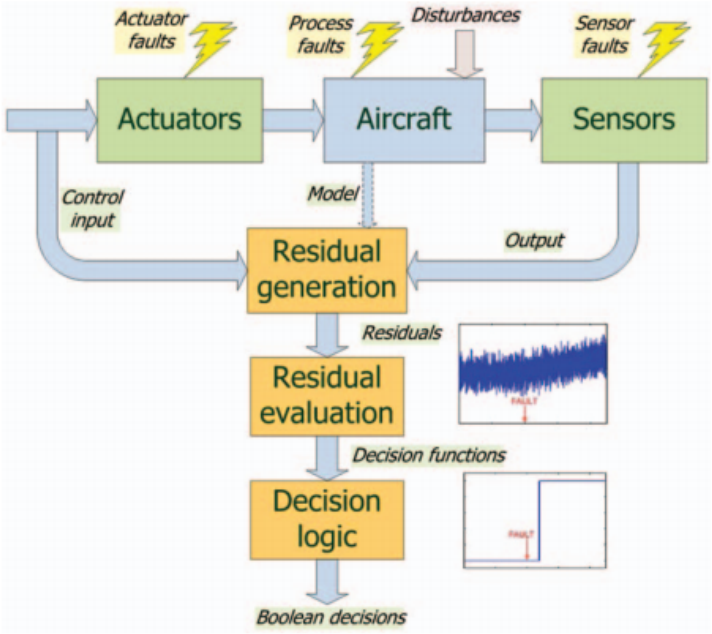


Figure 2.3: Typical FDD Scheme [81]

Residual generation employs a mathematical model of the system where the actuator control inputs and the system responses that are measured by the sensors are delivered to predict the behaviour of the system and then compare this predicted response with the actual behaviour. This procedure aims to calculate quantitative indices of fault occurrence in a format of the residuals. The residuals should converge to zero in a condition of no fault and deviate from zero when failure occurred. Next, a residual evaluation strategy is required to translate the time history of residual behaviour into a logic decision function.

In order to design FTCS successfully, the balance among various design objectives and interaction among FDD and reconfigurable control have to be considered to perform in real-time. Therefore, issues of how to integrate FDD and reconfigurable control in FTCS pose significant challenges in practice and deserve further investigation. An excellent and comprehensive review on the development of fault tolerant control system was presented in [145], which illustrated the development trend in the future. However, the literature review in this paper focuses on applications of FTCS in aerospace research community. Fault tolerant control tech-

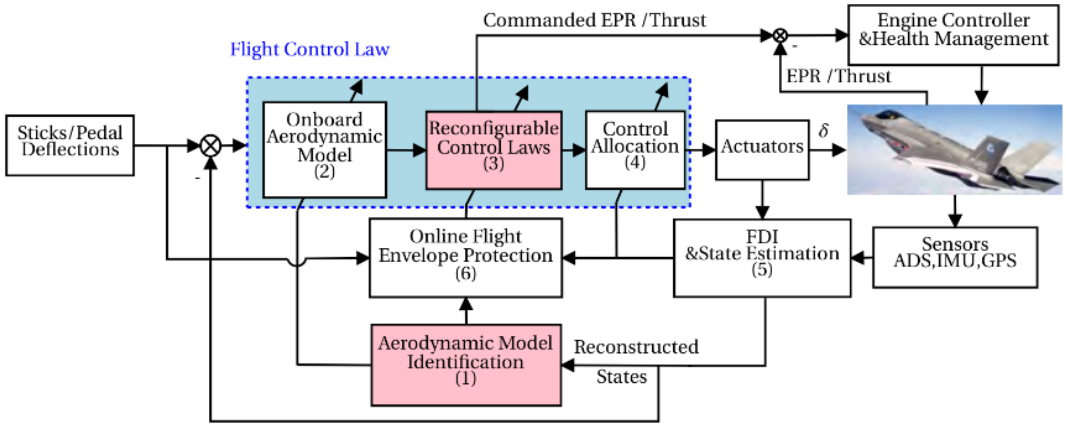


Figure 2.4: Block diagram of a conceptual modern model-based flight control system [122].

niques have been designed for flight control systems to meet the increasing safety requirement in various situations such as sensor & actuator failure [39, 55, 140], mass value and centre variations & damaged wing [125, 126], or/and emergency landing [123].

Typically, a fault tolerant flight control system comprises three sub-systems: (1) a reconfigurable control scheme, (2) an FDD, and (3) a reconfiguration mechanism as presented in [39, 140]. One of the strategies, similar to the FTC architecture for aircraft under adverse flight condition, was studied by Fekih and Pilla [39]. This flight control system incorporates passive and active control, such as robust control and adaptive control respectively, to work together according to a triggering function to achieve the best performance. Controller switching depends on the magnitude of the failure and the impairment severity evaluated by the FDI subsystem. In addition, Xingjian et al. [140] proposed a fault-tolerant control approach for civil aircraft under elevator failures. Trimmable horizontal stabilizers (THS) are considered to produce further pitch moments instead of a faulty elevator. A switching mechanism using performance-improvement-coefficients is employed to decide when it is suitable to use THS. This control system relies on LQR control method and model following technique to choose a suitable controller based switching control strategy.

Recently, Sun [122] introduced the increasing complex architecture of a model-based fault tolerant flight control system. This fault tolerant flight control system contains six sub-systems: 1) an aerodynamic model identification (AMI) element; 2) an online aerodynamic model element; 3) a reconfigurable control laws element; 4) a control allocation element; 5) a fault detection & isolation (FDI) element; and 6) an online flight envelope protection to work together. The AMI element based on physics can provide an accurate aircraft model for an indirect adaptive controller in a control block and detect failures occurring in the structure of the control surfaces by monitoring changes in meaningful aerodynamic parameters. Additionally, the online flight envelope protection element that receives failure information from FDI element can be used to predict safe flight envelope in case of failure, and then modify the reference command before feed it to the flight control law.

Moreover, research studies on FTCS methodology developed for autopilot system of UAVs can be found in [55, 67, 123]. These methods do not only considered fault-tolerant control techniques on flight control systems but also developed reconfiguration mechanisms for guidance control systems combined with a new simple adaptive path planning algorithm. Suzuki and Yanagida [123] presented the development of an intelligent flight control systems that could perform adaptive control and guide an aircraft in case of emergency situations. This intelligent control system consists of a fault tolerant control system unit using augmented neural network technology based on NDI controller and fault tolerant guidance system using online trajectory optimization to navigate the aircraft to a safety area in case of emergency.

Additionally, Ducard [55] proposed an autopilot system designed the algorithm modules to compute efficiently. This control system comprises of a nonlinear FDI system module, a control allocation module, a reconfigurable control module, and an adaptive and reconfigurable guidance module. The FDI module is based on a multiple model scheme with an auxiliary signal excitation in order to monitor any suspicious behaviours of the aircraft. An efficient control allocation module, which relies on the output of an FDI system, takes a responsible to distribute actuator control actions over the different control effectors available with some optimal method such as quadratic programming with magnitude and

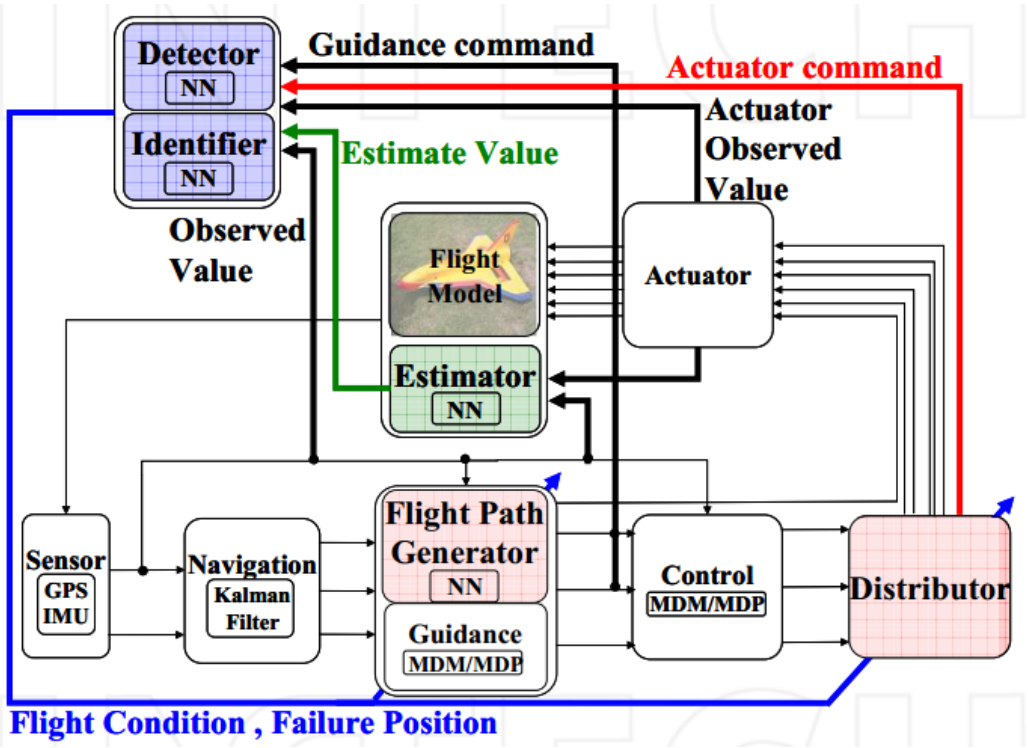


Figure 2.5: Block diagram of fault-tolerant autopilot system in [67].

rate constants on the control effectors. The flight control module depends on a combined technique of explicit model following, nonlinear dynamic inversion, and nonlinear transformations of selected state variables and an analysis of the stability and robustness in the presence of model parameter and sensor uncertainties. The novel adaptive and reconfigurable guidance module integrates a nonlinear guidance control law with a new simple adaptive path planning algorithm. This guidance system is used to autonomously avoid any known obstacles and calculate new trajectories online.

In [67], the intelligent flight control system consists of four additional advanced components and one input switching distributor unit as illustrated in Fig. 2.5. The four advanced components are: 1) next state estimator component; 2) flight condition detector component; 3) fault actuator location identifier component; and 4) flight path generator component. Each component is based on a neural network. The distributor unit is introduced in this control system to switch the input distribution matrix by using the outputs of the detector and identifier

components. All components and one distribution unit are employed to work together with the autopilot system in order to discriminate between faults and natural disturbances. They assess and adapt to the circumstances and a learning-based systematic methodology is used.

2.4.2 Agent Technology in Aerospace Systems

From the previous literature review, the flight control systems have become to be intensively a complicated system due to a number of additional subsystems. The issues of how to integrate all subsystem into the flight control system are significant challenges and deserve further research. Furthermore, a recent NASA report [13] supported that the next generation of aircraft will necessarily combine new complex algorithms (likely to be artificial intelligent) and non-traditional software components with adaptive control algorithms to provide enhanced safety, autonomy, and robustness under adverse flight conditions. This unmanned aircraft will operate with intelligent software that performs the high-level decision-making ("analyze-and-decide"capability) functions similar to human pilots and engineers.

Consequently, the agent-oriented paradigm is considered as one of the efficient patterns for large-scale distributed systems to handle dynamic uncertainties in the environment. Additionally, the agent-oriented methodology possesses the properties of autonomy, proactiveness, reactiveness and social ability. It means that the agent can sense its environment or other agents and perform autonomously actions or plans by independently making a decision in complex situations [139]. An extensive literature survey on applications of agent technology in control engineering applications can be found in [33], which shows that the techniques and methods in the field of agent and multi-agent systems have been applied to many aspects of road , railway, and air transportation. Furthermore, there are some publications to develop and design intelligent agents controlling sophisticated systems [73, 134], such as satellites [36, 74] and rovers [9].

The problem of a theoretical agent-based framework for adaptive control was studied by Lincoln et al [73]. This agent framework which is an extension of Belief-Desire-Intension (BDI) agent integrates rational decision making with real-

time evaluation of abstractions on the effect of future actions and planning for the future of the physical environment. This new framework aims to decrease the complexity of logical inference of agents controlling autonomous vehicles and robots. Natural language programming (NLPr) is utilized to facilitate how to program abstractions and unified system ontology in sEnglish. The motivating example implementation presented the development of this new agent framework for satellite control focuses on the need of dynamic adaptivity due to actuator changes and disturbances.

In addition, Veres and Luo [135] presented a multi-agent scheme on control systems with a high degree of autonomy. This architecture consists of agents for various components of the system, including modelling and controller optimization, implementation and performance monitoring. This new agent architecture called cautiously optimistic control agents (COCA) applies new modelling results with caution for control while using current model settings until a certain threshold exceeded a margin. For implementation, agent-oriented programming (AOP) which allows actions to be triggered by events was used. COCA is a multilayer architecture with a central unit acting as a coordinator or supervisor of the entire system. Plans and tasks are distributed among multiple agents. Agents such as a physical modeller agent and experimenter agent have specific tasks to complete and must communicate the results to other agents. These results could be employed as inputs for other agents.

There are, furthermore, some publications on the application of agent approach for high-level control of aircraft related to mission command control, such as collision and object avoidance. However, there is still a lack of research study on agent-oriented approach applied directly to intelligent flight control system. Details of agent theory will be explained in Chapter 5.

2.5 Chapter Summary and Thesis Direction

2.5.1 Chapter Summary

This chapter has presented literature associated with flight control methods for fixed-wing aircraft and UAVs. The PID algorithm is a popular feedback controller implementation in both commercial and academic autopilots. But the PID algorithm requires gain tuning, with manual trial and error technique that relies on the experience of a tuner. Consequently, modern control methods based on linear synthesis and/or optimisation are considered in aircraft control design to deal with uncertainties occurring from an imperfect approximation of the mathematical model and disturbances in order to achieve robust performance and stability in the presence of bounded modelling errors. However, there are some considerable drawbacks. The satisfying modern controller is not guaranteed in events of unanticipated or multiple failures, resulting in models outside the stability bound. Furthermore, suitable model of certain structural failures representing an uncertainty description is necessary to solve the modern controller. Therefore, the different reconfigurable control methods including gain-scheduling, NDI, parameter identification for indirect adaptive control, neural networks, and MPC have been studied and applied to the flight control system under adverse flight operating due to a larger flexibility to deal with failures.

Recently, advanced adaptive control schemes such as adaptive back-stepping control, $\mathcal{L}_1$ adaptive control, and hybrid adaptive control have been investigated for stability and control recovery of a damaged aircraft in order to guarantee the stability and improve the performance under adverse flight conditions. Furthermore, new generations of flight control systems have not only relied on either one of the controller, but they will feature complete and integrated systems that reconfigure flight controllers by considering variation of flight operational condition or fault that might occur in real-time. Therefore, the increasing complex features of the fault-tolerant flight control schemes were proposed with multiple intelligent subsystems including a reconfigurable control element, a fault detection and isolation element, an aerodynamic model identification element, a control allocation, a flight path generator and an on-line flight envelope protection in order

to meet with the increased performance and demand on reliability, safety, fault tolerance, and autonomy. The fault-tolerant flight control systems possess the ability to support failure component automatically by a collaboration of all such intelligent components to maintain aircraft stability and acceptable performance in events of failures.

Moreover, an agent-oriented paradigm that possesses a capability of high-level decision-making and deals with a large-scale distributed system is considered as an efficient technique to join all various interdisciplinary components of the system to achieve the ultimate goal. There are several agent architectures available. A well-known agent architecture with considerable advantages, in term of its ability to combine reactivity with long-term planning, is the belief-desire-intention (BDI) agent approach, that parallels with decision making to follow intentions and pursue goals based on beliefs; which is not a negligible practical aspect when the agent needs to explain its decisions. Furthermore, the agent can be developed in an extension of the AgentSpeak/Jason languages with aids of Natural Language Programming (NPLr) by sEnglish publication software and makes decisions using abstractions from flight data and from predictions of the anticipated physical environment.

2.5.2 Chosen Method and Thesis Direction

Adaptive control, which possesses its inherent flexibility to adapt to changes in system parameters, has been introduced in this research project. More clearly, the combination of adaptive nonlinear dynamic inversion augmented with a real-time aerodynamic model identification and neural networks has been chosen here as a control approach to be followed, which focuses on the use of mathematical representations based on flight dynamics. Such control, sometimes called hybrid adaptive control, combines the advantages of three reconfigurable control techniques including NDI (as explained in Section 2.3.2), indirect adaptive control based on on-line parameter estimation (as explained in Section 2.3.3), and direct adaptive control based on neural network (as explained in Section 2.3.4).

The ability to handle changes of operating point naturally without the requirement of gain scheduling is a major attraction of NDI control. Additionally,

another advantage is its property of decoupling the control axes. It means that no coupling effects remain between steering control channels and the different degree of freedom. Furthermore, every quantity and variable appearing in the model has a physical meaning and thus are interpretable in this method. Therefore, this is a transparent approach that allows designers and engineers to interpret data in each step. It is assumed that these physical models will facilitate certification for real-life applications in the future since monitoring data is more meaningful. For example, these physical meaning variables can be used to monitor system health by observing changes in quantities and variables. Satisfactory performance of NDI depends on an assumption of highly accurate known dynamic model. However, in practice, the plant dynamics for NDI is not realistic to assume to be accuracy, not only in the aspect of system uncertainties but also unable to account for unanticipated failures.

One successful solution to over the weakness of classical NDI, specifically its sensitivity to modelling errors, is the application of a real-time identification algorithm, which provides updated model information to the dynamic inversion. The disadvantage of this algorithm is that no formal stability proof can be provided since it is not based on Lyapunov's Theorem like adaptive backstepping or slide mode control. However, relying on the certainty equivalence principle [75], the stability proof can be implicitly removed in this thesis. Furthermore, neural networks also have been introduced in the literature to augment the control signal as compensation for the inverted dynamics, as explained in Section 2.3.4. An optimal control modification law based on an optimal control formulation that minimizes the $\mathcal{L}_2$ norm of the tracking error bounded away from some lower bound is considered to be an adaptive law for neural networks control in this thesis due to the ability to deal with fast adaptation without loss of robustness.

Therefore, the main benefits of hybrid adaptive control approach are 1) to provide stability guarantee and improve flight control performance in cases of significant change such as structure damage of aircraft in real-time and 2) to rely on a mathematical model that provides physical meanings of aerodynamic stability and control coefficients. $\mathcal{L}_1$ adaptive approach that implements with a low-pass filter on the adaptive control signal effectively can also suppress the problem of high-frequency oscillations that cause fast adaptation. However, the

$\mathcal{L}_1$ adaptive method has a limitation in providing a time delay margin bounded away from zero.

To achieve advanced capability of maintaining aircraft stability and an acceptable performance in the event of system failure or damage, the issues of how to integrate control system with increasing intelligent components have been considered. A BDI agent architecture has been chosen in this thesis due to its ability to combine reactivity with long-term planning. the BDI agent approach, which parallels with human decision making to follow intentions and pursue goals based on beliefs, has a non-negligible practical benefit when the agent needs to explain its decisions. Furthermore, the agent can be developed in an extension of the AgentSpeak/Jason languages with the use of Natural Language Programming (NPLr) by an sEnglish software development platform [124] using abstractions from flight data and prediction of the anticipated physical environment. A rational agent system, which integrates some knowledge based decision-making rules with hybrid adaptive controllers for small fixed wing UAVs, has been developed in this thesis. This agent will contain NDI based control with various additional elements including aircraft dynamic identification, neural networks, model validation, flight trim condition monitoring, and control performance evaluation in order to work together under adverse flight conditions.

In this concept, the agent consists of seven main computation components for perceptions and actions. Frequency Dependent Model Validation (FDMV), firstly, is chosen in this thesis as one component of the intelligent control system to find out whether or not the aircraft aerodynamic parameters are good enough for the inner NDI control of the autopilot system as explain in Chapter 9.1. This method can check if each spectrum component of the frequency domain residual has statistical properties of white a noise signal. This advantage of FDMV is to remove unwanted noise by the calculation in the interesting frequency range. Secondly, a method monitors the aircraft whether it is in trim condition from flight data. This method is called the real-time wavelet flight evaluator as to be explained in Chapter 9.2.1. Thirdly, FDMV is also applied to evaluate control performance by observing tracking error as explained in Chapter 9.3. Fourthly, the real-time aircraft parameter estimation component uses a method called forward OLS in frequency domain in this thesis as explained in Chapter

8.1. This approach computes in real-time and reduces the number of states for estimation such as angular acceleration. Fifthly, this component bases on neural networks control to augment the control signal for NDI control as explained in Chapter 8.2. Another component, sixthly, utilizes to calculate guidance control law that is based on L_1 line of sight guidance law as explained in Chapter 4.4. The final component has the responsibility to generate a flight path following desired waypoints via Dubins path planing algorithm as explain in Chapter 4.5.

In the beginning, the agent uses NDI controllers to maintain stability and track the desired waypoints. The agent will perform some actions based on decision-making rules with discrete perception abstractions (discrete symbols by filtering continuous flight data) that result from the computation of the model validation element, the control performance evaluation element, and the flight monitoring element. The agent will take a responsibility for the five main tasks to augment the performance of NDI control;

1. indirect adaptive control based on parameter estimation
2. neural networks based direct adaptive control
3. re-identification process to update new aircraft parameters for the inverse model of the inner flight control loop
4. reset mechanism for adaptive gain adjustment of direct adaptive control
5. hybrid adaptive control of the outer loop flight control

Part II

Mathematical Models

Chapter 3

UAV Dynamics

3.1 Reference Frames

It is importance to introduce reference frames before studying aircraft dynamic and flight control. All reference frames [31] are right-handed systems.

Inertial axes. Its origin is the center of the Earth. The Z-axis points to the North Pole of the Earth. The X-axis steers towards the **Vernal Equinox**. The Y-axis is perpendicular to both axes according to the right-hand rule.

Earth axes $O_{X_E Y_E Z_E}$. Its origin is at an arbitrary location on the ground. The O_{Z_E} axis points towards the center of the Earth. The O_{X_E} axis is directed North. The O_{Y_E} axis that can be determined by using the right-hand rule points to the direction of East.

Body axes $O_{X_B Y_B Z_B}$. The origin of this body axes is at the center of gravity (c.g.) of the aircraft. The O_{X_B} axis lies in the symmetry plane of the aircraft and points forward through the nose of the aircraft. The O_{Z_B} axis also lies in the symmetry plane, but points downwards. (It is perpendicular to the O_{X_B} axis.) The O_{Y_B} axis points out towards the right wing according to the right-hand rule.

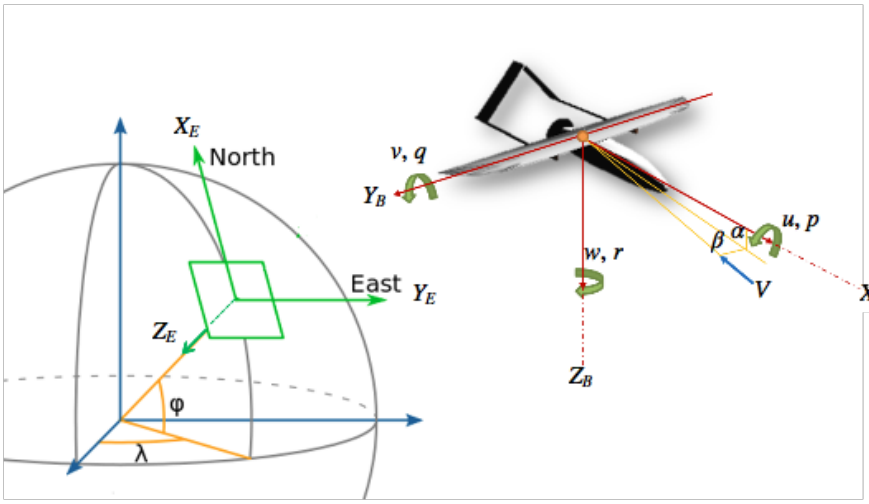


Figure 3.1: Reference Frames.

Stability axes $O_{X_S Y_S Z_S}$ is similar to the body axes. It is rotated by an angle α about the O_{Y_B} axis. The α is related to the relative wind vector (V_a) and the body axes. This relative wind vector can be projected onto plane of symmetry of the aircraft. Then this projection is the direction of the O_{X_S} axis. (The O_{Z_S} axis still lies in the plane of symmetry. Also, the O_{Y_S} axis is still equal to the O_{Y_B} axis.) So, the relative wind vector lies in the $O_{X_S Y_S}$ plane.

Wind axes $O_{X_W Y_W Z_W}$ is similar to the stability axes. It is rotated by an angle β about the O_{Z_S} axis. This is done, such that the O_{X_W} axis points in the direction of the relative wind vector V_a. (So the O_{X_W} axis generally does not lie in the symmetry plane.) The O_{Z_W} axis is still equal to the O_{Z_S} axis. The O_{Y_W} axis can now be found using the right-hand rule.

3.2 Flight Equations of Motion

3.2.1 Translational and Rotational Dynamics

To consider the effect of c.g. shifting, the general equations of aircraft motion can be modified to create the effect in the body-fixed reference frame of the aircraft

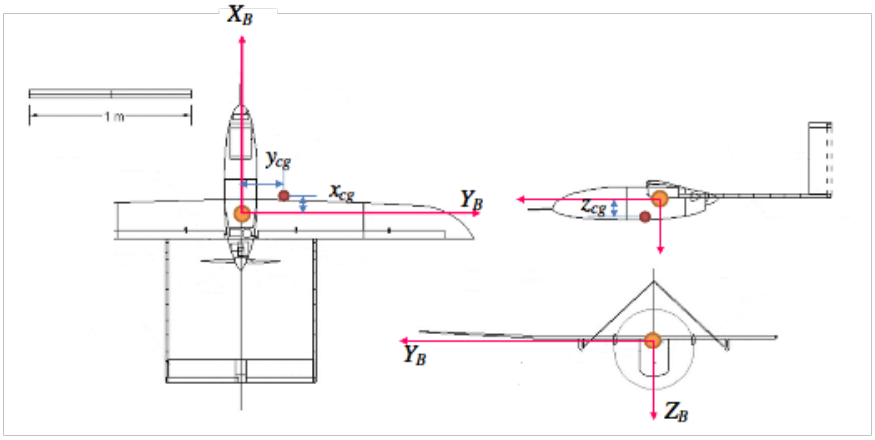


Figure 3.2: Center of gravity shifts to reference point.

as shown in translational and rotational forms [96]:

$$\begin{aligned} \vec{\mathbf{F}}_B &= \frac{d}{dt}(m\bar{v} + m\bar{\omega} \times \Delta\bar{r}) \\ \vec{\mathbf{M}}_B &= \frac{d}{dt}(\vec{\mathbf{H}}_B) \;=\; \frac{d}{dt}(\mathbf{I}\bar{\omega} + m\Delta\bar{r} \times \bar{v}) \end{aligned} \tag{3.1}$$

where $\vec{\mathbf{F}}_B$ is the applied force vector in body axes, $\vec{\mathbf{M}}_B$ is the applied moment vector in body axes, $m\bar{v}$ is the linear momentum , $\vec{\mathbf{H}}_B$ is the angular momentum vector, $\mathbf{I}$ is the inertia tensor, $\bar{\omega}$ is the angular velocity of aircraft, and $\Delta r = [x_{cg}\ y_{cg}\ z_{cg}]^T$ is the shifted distance vector of c.g. from the origin of the body frame as shown in Fig. 3.2.

However, the general motion equations based on Newton's second law are valid in an inertial frame where the earth frame system is assumed to be fixed in inertial space.

$$\tfrac{d(.)}{dt}|_I = \tfrac{d(.)}{dt}|_B + \bar{\omega} \times (.) \tag{3.2}$$

Eq. (3.1) can be transformed from the body-fixed reference frame to the inertial reference frame as

$$\begin{aligned} \vec{\mathbf{F}} &= \vec{\mathbf{F}}_B + \bar{\omega} \times (m\bar{v} + m\bar{\omega} \times \Delta\bar{r}) \\ \vec{\mathbf{M}} &= \frac{d}{dt}(\vec{\mathbf{H}}_B) + \bar{\omega} \times \vec{\mathbf{H}}_B \end{aligned} \tag{3.3}$$

Then expanding Eq. (3.3)

$$\begin{aligned}\vec{\mathbf{F}} = &\ m\dot{\bar{v}} + m\dot{\bar{\omega}} \times \Delta r + m\bar{\omega} \times \Delta\dot{\bar{r}} + \Delta\dot{m}(\bar{v} + \bar{\omega} \times \Delta\bar{r}) + m\bar{\omega} \times (\bar{v} + \bar{\omega} \times \Delta\bar{r}) \\ \vec{\mathbf{M}} = &\ \mathbf{I}\dot{\bar{\omega}} + \dot{\mathbf{I}}\bar{\omega} + m\Delta\bar{r} \times \dot{\bar{v}} + m\Delta\dot{\bar{r}} \times \bar{v} + \Delta\dot{m}\Delta\bar{r} \times \bar{v} + \bar{\omega} \times \mathbf{I}\bar{\omega} + m\bar{\omega} \times (\Delta\bar{r} \times \bar{v})\end{aligned} \tag{3.4}$$

When ignore some terms due to time-varying mass, inertia, and c.g. position, Eq. (3.4) become

$$\begin{aligned}\vec{\mathbf{F}} = &\ m\dot{\bar{v}} + m\dot{\bar{\omega}} \times \Delta r + m\bar{\omega} \times (\bar{v} + \bar{\omega} \times \Delta\bar{r}) \\ \vec{\mathbf{M}} = &\ \mathbf{I}\dot{\bar{\omega}} + m\Delta\bar{r} \times \dot{\bar{v}} + \bar{\omega} \times \mathbf{I}\bar{\omega} + m\bar{\omega} \times (\Delta\bar{r} \times \bar{v})\end{aligned} \tag{3.5}$$

Therefore, Eq. (3.5) can be expanded into the following force and moment equations where it can be proved the dynamics of the fixed-wing aircraft can be described with the following 6-DOF non-linear model as follows:

Force equations:

$$\begin{aligned}F_x = &\ m(\dot{u} - vr + wq - x_{cg}(q^2 + r^2) + y_{cg}(pq - \dot{r}) + z_{cg}(pr + \dot{q})) \\ F_y = &\ m(\dot{v} - wp + ur - x_{cg}(r^2 + p^2) + y_{cg}(qr - \dot{p}) + z_{cg}(qp + \dot{r})) \\ F_z = &\ m(\dot{w} - uq + vp - x_{cg}(p^2 + q^2) + y_{cg}(rp - \dot{q}) + z_{cg}(rq + \dot{p}))\end{aligned} \tag{3.6}$$

Moment equations:

$$\begin{aligned}L = &\ I_x\dot{p} + (I_z - I_y)qr - (\dot{r} + pq)I_{xz} + (r^2 - q^2)I_{yz} + (pr - \dot{q})I_{xy} \\ &+m[x_{cg}(vq + wr) + y_{cg}(\dot{w} - uq) - z_{cg}(\dot{v} + ur)] \\ M = &\ I_y\dot{q} + (I_x - I_z)rp - (\dot{p} + qr)I_{xy} + (p^2 - r^2)I_{zx} + (qp - \dot{r})I_{yz} \\ &+m[-x_{cg}(\dot{w} + vp) + y_{cg}(up + wr) + z_{cg}(\dot{u} - vr)] \\ N = &\ I_z\dot{r} + (I_y - I_x)pq - (\dot{q} + rp)I_{yz} + (q^2 - p^2)I_{xy} + (rq - \dot{p})I_{zx} \\ &+m[x_{cg}(\dot{v} - wp) - y_{cg}(\dot{u} + wq) + z_{cg}(up + qv)]\end{aligned} \tag{3.7}$$

where:

	m	Total mass of the aircraft
$\bar{v} =$	$[u\ v\ w]^T$	Linear velocities decomposed in the body-frame.

$\bar{\omega} =$	$[p\ q\ r]^T$	Angular velocities decomposed in the body-frame. The angular velocities p, q and r are commonly known as *roll*, *pitch* and *yaw* respectively.
$\vec{\mathbf{F}} =$	$[F_x\ F_y\ F_z]^T$	External forces decomposed in the body-frame.
$\vec{\mathbf{M}} =$	$[L\ M\ N]^T$	External momentums decomposed in the body-frame.
$\Delta\bar{r} =$	$[x_{cg}\ y_{cg}\ z_{cg}]^T$	Shifted position of the centre of gravity in the body-frame as shown in Fig. 3.2
$\mathbf{I} =$	$\begin{bmatrix} I_x & I_{xy} & I_{xz} \\ I_{yx} & I_y & I_{yz} \\ I_{zx} & I_{zy} & I_z \end{bmatrix}$	Inertia tensor

The left hand side of Eq. (3.5) represents all the external forces and moments applied to the aircraft, respectively. In the dynamical model presented in [31, 66], the external forces and moments vector can be identified as the sum of three components: *aerodynamic* ($\vec{\mathbf{F}}_A, \vec{\mathbf{M}}_A$), *propulsion* ($\vec{\mathbf{F}}_P, \vec{\mathbf{M}}_P$) and *gravity* ($\vec{\mathbf{F}}_G, \vec{\mathbf{M}}_G$) :

$$\vec{\mathbf{F}} = \vec{\mathbf{F}}_A + \vec{\mathbf{F}}_G + \vec{\mathbf{F}}_P \tag{3.8}$$

$$\vec{\mathbf{M}} = \vec{\mathbf{M}}_A + \vec{\mathbf{M}}_G + \vec{\mathbf{M}}_P \tag{3.9}$$

The total applied forces and moments in turn can be expressed as:

$$\begin{aligned} F_x &= \bar{q}SC_x(\alpha, \beta, \tfrac{pb}{2V}, \tfrac{q\bar{c}}{2V}, \tfrac{rb}{2V}, ...) && -mgsin\theta && +F_{P_x} \\ F_y &= \bar{q}SC_y(\alpha, \beta, \tfrac{pb}{2V}, \tfrac{q\bar{c}}{2V}, \tfrac{rb}{2V}, ...) && +mgsin\phi cos\theta && +F_{P_y} \\ F_z &= \bar{q}S\underbrace{C_z(\alpha, \beta, \frac{pb}{2V}, \frac{q\bar{c}}{2V}, \frac{rb}{2V}, ...)}_{\substack{\text{aerodynamic force coefficients} \\ \text{discussed in Section 3.2.2}}} && +mgcos\phi cos\theta && +F_{P_z} \end{aligned} \tag{3.10}$$

$$\begin{aligned} L &= \bar{q}SbC_l(\alpha, \beta, \tfrac{pb}{2V}, \tfrac{q\bar{c}}{2V}, \tfrac{rb}{2V}, ...) && +F_{P_x}x_e \\ M &= \bar{q}S\bar{c}C_m(\alpha, \beta, \tfrac{pb}{2V}, \tfrac{q\bar{c}}{2V}, \tfrac{rb}{2V}, ...) && +F_{P_y}y_e \\ N &= \bar{q}Sb\underbrace{C_n(\alpha, \beta, \frac{pb}{2V}, \frac{q\bar{c}}{2V}, \frac{rb}{2V}, ...)}_{\substack{\text{aerodynamic moment coefficients} \\ \text{discussed in Section 3.2.2}}} && +F_{P_z}z_e \end{aligned} \tag{3.11}$$

Note that Eqs. (3.6) and (3.7) simplify consistently in case the c.g. shifted from the origin of the body-frame, or in other words, in case the c.g. is located at the origin of body-frame , as in that case $x_{cg} = y_{cg} = z_{cg} = 0$. Furthermore, it is clear that for a rigid body with symmetry relative to the $O_{X_B Y_B Z_B}$ in body axes, therefore we can define that $I_{xy} = I_{yx} = I_{yz} = I_{zy} = 0$

Force equations:

$$\begin{aligned} F_x &= m(\dot{u} - vr + wq) \\ F_y &= m(\dot{v} - wp + ur) \\ F_z &= m(\dot{w} - uq + vp) \end{aligned} \tag{3.12}$$

Moment equations:

$$\begin{aligned} L &= I_x\dot{p} + (I_z - I_y)qr - (\dot{r} + pq)I_{xz} \\ M &= I_y\dot{q} + (I_x - I_z)rp + (p^2 - r^2)I_{zx} \\ N &= I_z\dot{r} + (I_y - I_x)pq + (rq - \dot{p})I_{zx} \end{aligned} \tag{3.13}$$

Using Eq. (3.12) and Eq. (3.13) through Eq. (3.10) and Eq. (3.11), the following six differential equations describe the symmetric aircraft motion with assumption that the thrust from the propulsion performs along the x body axis and through the c.g.

Force equations:

$$\begin{aligned} m\dot{u} &= m(vr - wq) + \bar{q}SC_x - mg\sin\theta + T \\ m\dot{v} &= m(wp - ur) + \bar{q}SC_y - mg\cos\theta\sin\phi \\ m\dot{w} &= m(uq - vp) + \bar{q}SC_z - mg\cos\theta\cos\phi \end{aligned} \tag{3.14}$$

Moment equations:

$$\begin{aligned} I_x\dot{p} - I_{xz}\dot{r} &= \bar{q}SbC_l - (I_z - I_y)qr + I_{xz}qp \\ I_y\dot{q} &= \bar{q}S\bar{c}C_m - (I_x - I_z)pr - I_{xz}(p^2 - r^2) \\ I_z\dot{r} - I_{xz}\dot{p} &= \bar{q}SbC_n - (I_y - I_x)pq - I_{xz}qr \end{aligned} \tag{3.15}$$

3.2.2 Aerodynamic Modelling

Before studying aerodynamic modelling, there are significant factors in wind axes system to investigate which consist of airspeed (V), the angle of attack (α) and sideslip angle (β) as illustrated in Fig 3.1. Consequently, these variables can be defined in term of u, v, and w as follows [66]:

$$\begin{aligned} V &= \sqrt{u^2 + v^2 + w^2} \\ \alpha &= \tan^{-1} \frac{w}{u} \\ \beta &= \sin^{-1} \frac{v}{V} \end{aligned} \tag{3.16}$$

Aerodynamic model in this thesis relied on quasi-steady flow with time-invariant parameters. Therefore, this form of the aerodynamic equation can be provided by a linear Taylar series expansion of the aerodynamic forces and moments about a reference condition. For simplicity, the aerodynamic model equations represent use of non-dimensional derivatives of the non-dimensional aerodynamic force and moment coefficients including C_D, C_L, C_Z, C_l, C_m and C_n. These parameters depend on the reference airspeed and altitude condition [66]. From general aerodynamic principle following the literature [65, 77, 109], the regular aerodynamic forces and moments are not only dependent on the usual linear independent variables but also reliant on non-linear symmetrical regressor candidates in case of aggressive manoeuvering. In case of asymmetric aircraft structure damage, asymmetrical non-linear regressor candidates also require considering the following parameter:

- Conventional linear independent variables:
  - In longitudinal dynamics: $1, \alpha, \frac{q\bar{c}}{2V}$ and δ_e
  - In lateral & directional dynamics: $1, \beta, \frac{pb}{2V}, \frac{rb}{2V}, \delta_a$ and δ_r

Therefore, aerodynamic force and moment coefficients in six degree-of-freedom can be expressed in terms of the mentioned independent variables in linear re-

gression form as:

$$\begin{aligned}
C_L &= C_{L_0} + C_{L_\alpha}\alpha + C_{L_q}(\tfrac{q\bar{c}}{2V}) + C_{L_{\delta_e}}\delta_e \\
C_D &= C_{D_0} + C_{D_\alpha}\alpha + C_{D_q}(\tfrac{q\bar{c}}{2V}) + C_{D_{\delta_e}}\delta_e \\
C_Y &= C_{Y_0} + C_{Y_\beta}\beta + C_{Y_p}(\tfrac{pb}{2V}) + C_{Y_r}(\tfrac{rb}{2V}) + C_{Y_{\delta_a}}\delta_a + C_{Y_{\delta_r}}\delta_r \\
C_l &= C_{l_0} + C_{l_\beta}\beta + C_{l_p}(\tfrac{pb}{2V}) + C_{l_r}(\tfrac{rb}{2V}) + C_{l_{\delta_a}}\delta_a + C_{l_{\delta_r}}\delta_r \\
C_m &= C_{m_0} + C_{m_\alpha}\alpha + C_{m_q}(\tfrac{q\bar{c}}{2V}) + C_{m_{\delta_e}}\delta_e \\
C_n &= C_{n_0} + C_{n_\beta}\beta + C_{n_p}(\tfrac{pb}{2V}) + C_{n_r}(\tfrac{rb}{2V}) + C_{n_{\delta_a}}\delta_a + C_{n_{\delta_r}}\delta_r
\end{aligned} \tag{3.17}$$

where

$$\begin{aligned}
C_L &= -C_Z\cos\alpha + C_X\sin\alpha \\
C_D &= -C_X\cos\alpha - C_Z\sin\alpha
\end{aligned} \tag{3.18}$$

- Non-linear symmetrical regressor candidates
  - In longitudinal dynamic : $\alpha^2, \alpha^m, \alpha\frac{q\bar{c}}{2V}, \alpha\delta_e \; m = 3, ..., 8$
  - In lateral & directional dynamic : $\alpha\beta, \alpha\beta^2, \alpha^2\beta, \alpha\beta^3, \alpha^2\beta^3, \alpha\frac{pb}{2V}, \alpha\frac{rb}{2V}, \alpha^2\frac{pb}{2V}, \alpha^2\frac{rb}{2V}, \beta^n \; n = 2, ..., 5$
- Asymmetrical non-linear regressor candidates:
  - In longitudinal dynamic : $\beta, \frac{pb}{2V}, \frac{rb}{2V}, \alpha\beta, \alpha\beta^2, \alpha^2\beta, \alpha\beta^3, \alpha^2\beta^3, \alpha\frac{pb}{2V}, \alpha\frac{rb}{2V}, \alpha^2\frac{pb}{2V}, \alpha^2\frac{rb}{2V}, \beta^n \; n = 2, ..., 5$
  - In lateral & directional dynamic : $1, \alpha, \frac{q\bar{c}}{2V}, \alpha^2, \alpha^m, \alpha\frac{q\bar{c}}{2V}, \alpha\delta_e \; m = 3, ..., 8$

Note that: For linear regression, aerodynamic modelling functions that can be linear or nonlinear functions of the regressor candiates are considered at the trim condition as a specific airspeed and altitude (at one point within the flight envelope). For large amplitude, rapid excursions, flight profiles with high angle of attack, and deficiency from aircraft damage about the reference flight condition, it is essential to extend the linear models by adding nonlinear terms, such as higher order and coupling terms as above mentioned. It means that only valid models are obtained in this study. Consequently, to find out the aerodynamic models for

covering the entire operational envelope of the aircraft, a number of test cases of flight manoeuvres need to be performed at various velocities (Mach number), angles of attack and altitudes with multiple local models with crip transitions at the boundaries between them [66]. An issue To find aerodynamic models which are locally as well as globally valid over the entire flight envelope, without crisp transition is outside the scope of the thesis.

3.2.3 Effects of Mass Property Changes Due to Damage

Before studying this section, it is necessary to discuss the translational acceleration measured by accelerometers. The equation for the translational acceleration is

$$\vec{a} = \dot{\bar{v}} + \omega \times \bar{v} - \frac{\vec{F_G}}{m} \tag{3.19}$$

In scalar form,

$$\begin{aligned} a_x &= \dot{u} - rv + qw + g\sin\theta \\ a_y &= \dot{v} - pw + ru - g\cos\theta\sin\phi \\ a_z &= \dot{w} - qu + pv - g\cos\theta\cos\phi \end{aligned} \tag{3.20}$$

From Eq. (3.5), the aircraft mass and inertia are assumed to undergo a change so that

$$m = m^* + \Delta m \tag{3.21}$$

$$\mathbf{I} = \mathbf{I}^* + \Delta\mathbf{I} = \begin{bmatrix} I_x^* + \Delta I_x & I_{xy}^* + \Delta I_{xy} & I_{xz}^* + \Delta I_{xz} \\ I_{yx}^* + \Delta I_{yx} & I_y^* + \Delta I_y & I_{yz}^* + \Delta I_{yz} \\ I_{zx}^* + \Delta I_{zx} & I_{zy}^* + \Delta I_{zy} & I_z^* + \Delta I_z \end{bmatrix} \tag{3.22}$$

where m^* is the original mass of the aircraft, Δm is the negative mass change due to damage, $\mathbf{I}^*$ is the original inertia matrix of the aircraft, and $\Delta\mathbf{I}$ is the

change in the inertia matrix due to damage.

$$\begin{aligned}\vec{\mathbf{F}} = &\ (m^* + \Delta m)\dot{\bar{v}} + (m^* + \Delta m)\dot{\bar{\omega}} \times \Delta r + (m^* + \Delta m)\bar{\omega} \times (\bar{v} + \bar{\omega} \times \Delta\bar{r}) \\ \vec{\mathbf{M}} = &\ (\mathbf{I}^* + \Delta\mathbf{I})\dot{\bar{\omega}} + (m^* + \Delta m)\Delta\bar{r} \times \dot{\bar{v}} + \bar{\omega} \times (\mathbf{I}^* + \Delta\mathbf{I})\bar{\omega} \\ &+(m^* + \Delta m)\bar{\omega} \times (\Delta\bar{r} \times \bar{v})\end{aligned} \tag{3.23}$$

From the appendix A, Eq. (3.23) can be expanded as:

Force equations:

$$\begin{aligned} m^*(\dot{u} - vr + wq + g\sin\theta) - F^*_{A_x} - F_{P_x} = &\ \Delta F_{A_x} - \Delta F_{M_x} \\ m^*(\dot{v} - wp + ur - g\cos\theta\sin\phi) - F^*_{A_y} - F_{P_y} = &\ \Delta F_{A_y} - \Delta F_{M_y} \\ m^*(\dot{w} - uq + vp - g\cos\theta\cos\phi) - F^*_{A_z} - F_{P_z} = &\ \Delta F_{A_z} - \Delta F_{M_z} \end{aligned} \tag{3.24}$$

where

$$\begin{aligned}\Delta F_{M_x} = &\ \Delta m a_x - (m^* + \Delta m)x_{cg}q^2 - (m^* + \Delta m)x_{cg}r^2 + (m^* + \Delta m)y_{cg}pq \\ &-(m^* + \Delta m)y_{cg}\dot{r} + (m^* + \Delta m)z_{cg}pr + (m^* + \Delta m)z_{cg}\dot{q} \\ = &\ f_{M_x}(a_x, \dot{q}, \dot{r}, q^2, r^2, pq, pr)\end{aligned} \tag{3.25}$$

$$\begin{aligned}\Delta F_{M_y} = &\ \Delta m a_y - (m^* + \Delta m)x_{cg}r^2 - (m^* + \Delta m)x_{cg}p^2 + (m^* + \Delta m)y_{cg}qr \\ &-(m^* + \Delta m)y_{cg}\dot{p} + (m^* + \Delta m)z_{cg}qp + (m^* + \Delta m)z_{cg}\dot{r} \\ = &\ f_{M_y}(a_y, \dot{p}, \dot{r}, p^2, r^2, qp, qr)\end{aligned} \tag{3.26}$$

$$\begin{aligned}\Delta F_{M_z} = &\ \Delta m a_z - (m^* + \Delta m)x_{cg}p^2 - (m^* + \Delta m)x_{cg}q^2 + (m^* + \Delta m)y_{cg}rp \\ &-(m^* + \Delta m)y_{cg}\dot{q} + (m^* + \Delta m)z_{cg}rq + (m^* + \Delta m)z_{cg}\dot{p} \\ = &\ f_{M_z}(a_z, \dot{p}, \dot{q}, p^2, q^2, rp, rq)\end{aligned} \tag{3.27}$$

Then Eq. (3.24) can be rearranged in matrix form as:

$$m^*\dot{\bar{v}} + m^*\bar{\omega} \times \bar{v} - \vec{\mathbf{F}}^*_G - \vec{\mathbf{F}}^*_A - \vec{\mathbf{F}}^*_P = \ \Delta\vec{\mathbf{F}}_A - \Delta\vec{\mathbf{F}}_M \tag{3.28}$$

where

$$\begin{aligned}\Delta\vec{\mathbf{F}}_A &= \underbrace{f_{F_A}(\alpha, \beta, \frac{pb}{2V}, \frac{q\bar{c}}{2V}, \frac{rb}{2V}, ...)}_{\substack{\text{aerodynamic force functions}\\ \text{discussed in Section 3.2.2}}} \\ \Delta\vec{\mathbf{F}}_M &= \underbrace{f_{F_M}(a_x, a_y, a_z, \dot{p}, \dot{q}, \dot{r}, p^2, q^2, r^2, pq, pr, qp, qr)}_{\substack{\text{mass and inertia force functions}\\ \text{discussed in Equation 3.25, 3.26, and 3.27}}}\end{aligned} \tag{3.29}$$

Moment equations:

$$\begin{aligned} I_x^*\dot{p} + (I_z^* - I_y^*)qr - (\dot{r} + pq)I_{xz}^* + (r^2 - q^2)I_{yz}^* + (pr - \dot{q})I_{xy}^* - L_A^* &= \Delta L_A - \Delta L_M \\ I_y^*\dot{q} + (I_x^* - I_z^*)rp - (\dot{p} + qr)I_{xy}^* + (p^2 - r^2)I_{zx}^* + (qp - \dot{r})I_{yz}^* &= \Delta M_A - \Delta M_M \\ I_z^*\dot{r} + (I_y^* - I_x^*)pq - (\dot{q} + rp)I_{yz}^* + (q^2 - p^2)I_{xy}^* + (rq - \dot{p})I_{zx}^* &= \Delta N_A - \Delta N_M \end{aligned} \tag{3.30}$$

where

$$\begin{aligned}\Delta L_M = \;& \Delta I_x\dot{p} + (\Delta I_z - \Delta I_y)qr - (\dot{r} + pq)\Delta I_{xz} + (r^2 - q^2)\Delta I_{yz} + (pr - \dot{q})\Delta I_{xy} \\ & + (m^* + \Delta m)x_{cg}vq - (m^* + \Delta m)y_{cg}vp + (m^* + \Delta m)x_{cg}wr \\ & - (m^* + \Delta m)z_{cg}wp + (m^* + \Delta m)y_{cg}a_z - (m^* + \Delta m)z_{cg}a_y \\ = \;& f_{M_L}(\dot{p}, \dot{q}, \dot{r}, q^2, r^2, pq, pr, qr, vq, vp, wr, wp, a_y, a_z)\end{aligned} \tag{3.31}$$

$$\begin{aligned}\Delta M_M = \;& \Delta I_y\dot{q} + (\Delta I_x - \Delta I_z)rp - (\dot{p} + qr)\Delta I_{xy} + (p^2 - r^2)\Delta I_{zx} + (qp - \dot{r})\Delta I_{yz} \\ & - (m^* + \Delta m)x_{cg}a_z + (m^* + \Delta m)y_{cg}up - (m^* + \Delta m)x_{cg}uq \\ & + (m^* + \Delta m)y_{cg}wr - (m^* + \Delta m)z_{cg}wq + (m^* + \Delta m)z_{cg}a_x \\ = \;& f_{M_M}(\dot{p}, \dot{q}, \dot{r}, p^2, r^2, rp, qr, qp, up, uq, wr, wq, a_x, a_z)\end{aligned} \tag{3.32}$$

$$\begin{aligned}\Delta N_M = \;& \Delta I_z\dot{r} + (\Delta I_y - \Delta I_x)pq - (\dot{q} + rp)\Delta I_{yz} + (q^2 - p^2)\Delta I_{xy} + (rq - \dot{p})\Delta I_{zx} \\ & + (m^* + \Delta m)x_{cg}a_y - (m^* + \Delta m)y_{cg}a_x + (m^* + \Delta m)z_{cg}up \\ & + (m^* + \Delta m)z_{cg}qv - (m^* + \Delta m)x_{cg}ur - (m^* + \Delta m)y_{cg}rv \\ = \;& f_{M_N}(\dot{p}, \dot{q}, \dot{r}, p^2, q^2, pq, rp, rq, up, qv, ur, rv, a_y, a_z)\end{aligned} \tag{3.33}$$

Similarly, Eq. (3.30) can be rearranged in matrix form as:

$$\mathbf{I}^*\dot{\bar{\omega}} + \bar{\omega} \times \mathbf{I}^*\bar{\omega} + \vec{\mathbf{M}}_A^* = \quad \Delta\vec{\mathbf{M}}_A - \Delta\vec{\mathbf{M}}_M \tag{3.34}$$

where

$$\begin{aligned} \Delta\vec{\mathbf{M}}_A = & \underbrace{f_{M_A}(\alpha, \beta, \frac{pb}{2V}, \frac{q\bar{c}}{2V}, \frac{rb}{2V}, ...)}_{\substack{\text{aerodynamic moment functions} \\ \text{discussed in Section 3.2.2}}} \\ \Delta\vec{\mathbf{M}}_M = & \underbrace{f_{M_M}(a_x, a_y, a_z, \dot{p}, \dot{q}, \dot{r}, p^2, q^2, r^2, pq, pr, qp, qr, up, uq, ur, vp, vq, vr, wp, wq, wr)}_{\substack{\text{mass and inertia moment functions} \\ \text{discussed in Equations 3.31, 3.32, and 3.33}}} \end{aligned} \tag{3.35}$$

3.3 Rotational Kinematic Equations & Navigation Equations

Rotational kinematic equations are a relationship between the rate of change of the Euler angles and the body-axis components of angular velocity. The relationship can be found in [66] as shown in Eq. (3.36)

$$\begin{aligned} \dot{\phi} = & \quad p + \tan\theta(q\sin\phi + r\cos\phi) \\ \dot{\theta} = & \quad q\cos\phi - r\cos\phi \\ \dot{\psi} = & \quad \tfrac{q\sin\phi + r\cos\phi}{\cos\theta} \end{aligned} \tag{3.36}$$

The navigation equations [66] can be written by expressing a relationship between the aircraft velocity components in earth axes and body-axis components of velocity:

$$\begin{aligned} \dot{x_E} = & \quad u\cos\psi\cos\theta + v(\cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi) \\ & + w(\cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi) \\ \dot{y_E} = & \quad u\sin\psi\cos\theta + v(\sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi) \\ & + w(\sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi) \\ \dot{h} = & \quad u\sin\theta - v\cos\theta\sin\phi - w\cos\theta\cos\phi \end{aligned} \tag{3.37}$$

where defining h = altitude (height above the ground).

Since air flow angle sensors base on wind reference system rather than body axes, the navigation equations can be considered to be related to V, α, and β in wind reference system by:

$$\begin{aligned} u &= V\cos\alpha\cos\beta \\ v &= V\sin\beta \\ w &= V\sin\alpha\cos\beta \end{aligned} \tag{3.38}$$

Consequently, the navigation equations are

$$\begin{aligned} \dot{x_E} = \; & V\cos\alpha\cos\beta\cos\psi\cos\theta + V\sin\beta(\cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi) \\ & + V\sin\alpha\cos\beta(\cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi) \end{aligned} \tag{3.39}$$

$$\begin{aligned} \dot{y_E} = \; & V\cos\alpha\cos\beta\sin\psi\cos\theta + V\sin\beta(\sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi) \\ & + V\sin\alpha\cos\beta(\sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi) \end{aligned} \tag{3.40}$$

$$\dot{h} = V\cos\alpha\cos\beta\sin\theta - V\sin\beta\cos\theta\sin\phi - V\sin\alpha\cos\beta\cos\theta\cos\phi \tag{3.41}$$

Chapter 4

NDI Control Based Architecture of Autopilot

The autopilot system consists of a guidance block and two loops (inner and outer loop) of flight control system as shown in Fig.4.1. The adaptive control based inner loop allows rate control in roll, pitch and yaw steering. The outer loop also adds adaptive control for heading, pitch and sideslip angle. Furthermore, the guidance system is based on an L_1 line of sight guidance law to calculate roll angle command to track the desired waypoints. The autopilot architecture is based on NDI control that represents "inversion"of the non-linear model of flight dynamics. In an sEnglish based encoding of this control scheme, all quantities and variables that appear in the model have a physical meaning expressed in professional English and hence are interpretable. This allows designers and maintenance engineers to interpret data in each step in order to potentially facilitate legal certification of these processes.

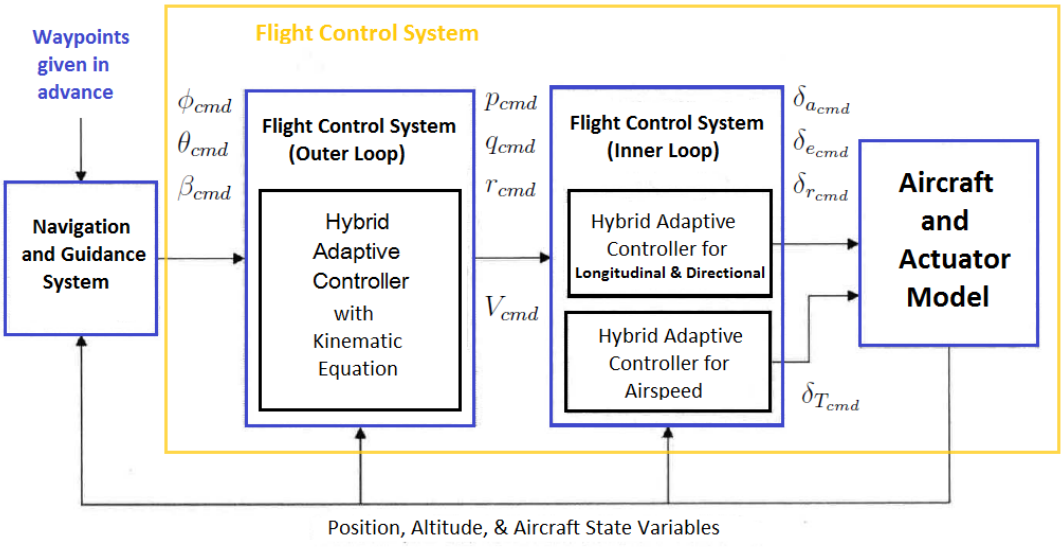


Figure 4.1: Autopilot Control Architecture

4.1 Inner Loop of Autopilot Laws

From aircraft moment equations (3.15) described by Eqs. (4.1-4.2)

$$\begin{aligned} I_x\dot{p} - I_{xz}\dot{r} &= \bar{q}SbC_l - (I_z - I_y)qr + I_{xz}qp \\ I_y\dot{q} &= \bar{q}S\bar{c}C_m - (I_x - I_z)pr - I_{xz}(p^2 - r^2) \\ I_z\dot{r} - I_{xz}\dot{p} &= \bar{q}SbC_n - (I_y - I_x)pq - I_{xz}qr \end{aligned} \tag{4.1}$$

where

$$\begin{aligned} C_l &= C_{l_0} + C_{l_\beta}\beta + C_{l_p}(\tfrac{pb}{2V}) + C_{l_r}(\tfrac{rb}{2V}) + C_{l_{\delta_a}}\delta_a + C_{l_{\delta r}}\delta_r + \dots \\ C_m &= C_{m_0} + C_{m_\alpha}\alpha + C_{m_q}(\tfrac{q\bar{c}}{2V}) + C_{m_{\delta e}}\delta_e + \dots \\ C_n &= C_{n_0} + C_{n_\beta}\beta + C_{n_p}(\tfrac{pb}{2V}) + C_{n_r}(\tfrac{rb}{2V}) + C_{n_{\delta_a}}\delta_a + C_{n_{\delta_r}}\delta_r + \dots \end{aligned} \tag{4.2}$$

With the body rotational rates $\omega = \begin{bmatrix} p & q & r \end{bmatrix}^T$, angular acceleration $\begin{bmatrix} \dot{p} & \dot{q} & \dot{r} \end{bmatrix}^T$, the moment coefficients $C_M = \begin{bmatrix} C_l & C_m & C_n \end{bmatrix}^T$, V as the airspeed, $\bar{q}$ as the dynamic pressure, S as the wing surface area, b as the wing span, $\bar{c}$ as the mean

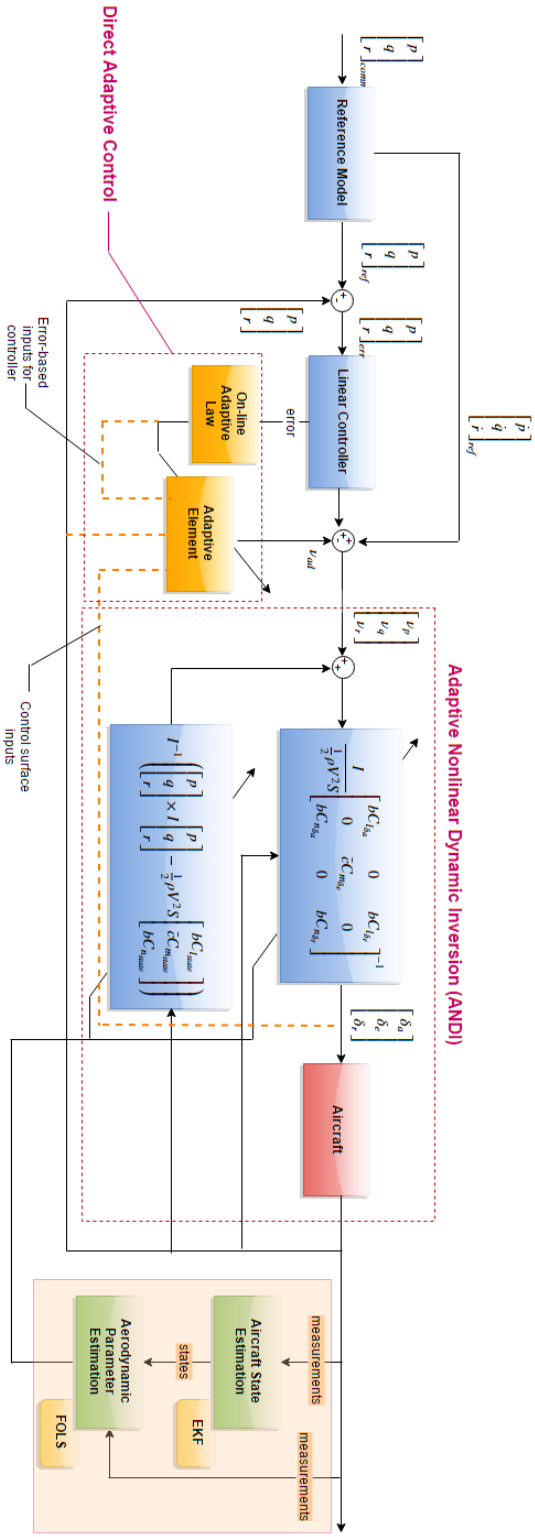


Figure 4.2: Hybrid adaptive angular rate control inner loop ASE=aircraft state estimation; AMI=aerodynamic model identification

aerodynamic chord, α as the angle of attack and β as the side slip angle. I is the moment of inertial matrix. Eq. (4.1) can be rewritten in matrix form as

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = I^{-1} \begin{bmatrix} \bar{q}SbC_l \\ \bar{q}S\bar{c}C_m \\ \bar{q}SbC_n \end{bmatrix} - I^{-1} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \left(I \begin{bmatrix} p \\ q \\ r \end{bmatrix} \right) \tag{4.3}$$

where

$$I = \begin{bmatrix} I_x & -I_{xy} & -I_{xz} \\ -I_{yx} & I_y & -I_{yz} \\ -I_{zx} & -I_{zy} & I_z \end{bmatrix} \tag{4.4}$$

Eq. (4.2) can also be rearranged in the matrix form as

$$\begin{bmatrix} C_l \\ C_m \\ C_n \end{bmatrix} = \begin{bmatrix} C_{l_{states}} \\ C_{m_{states}} \\ C_{n_{states}} \end{bmatrix} + \begin{bmatrix} C_{l_{\delta_a}} & 0 & C_{l_{\delta_r}} \\ 0 & C_{m_{\delta_e}} & 0 \\ C_{n_{\delta_a}} & 0 & C_{n_{\delta_r}} \end{bmatrix} \begin{bmatrix} \delta_a \\ \delta_e \\ \delta_r \end{bmatrix} \tag{4.5}$$

where

$$\begin{aligned} C_{l_{states}} &= C_{l_0} + C_{l_\beta}\beta + C_{l_p}(\tfrac{pb}{2V}) + C_{l_r}(\tfrac{rb}{2V}) + \dots \\ C_{m_{states}} &= C_{m_0} + C_{m_\alpha}\alpha + C_{m_q}(\tfrac{q\bar{c}}{2V}) + \dots \\ C_{n_{states}} &= C_{n_0} + C_{n_\beta}\beta + C_{n_p}(\tfrac{pb}{2V}) + C_{n_r}(\tfrac{rb}{2V}) + \dots \end{aligned} \tag{4.6}$$

Inserting Eq. (4.5) into Eq. (4.3) and solving for the control input $\begin{bmatrix} \delta_a & \delta_e & \delta_r \end{bmatrix}^T$, results in

$$\begin{bmatrix} \delta_a \\ \delta_e \\ \delta_r \end{bmatrix} = \begin{bmatrix} bC_{l_{\delta_a}} & 0 & bC_{l_{\delta_r}} \\ 0 & \bar{c}C_{m_{\delta_e}} & 0 \\ bC_{n_{\delta_a}} & 0 & bC_{n_{\delta_r}} \end{bmatrix}^{-1} \cdot \left\{ \frac{I}{\frac{1}{2}\rho V^2 S} \left(\begin{bmatrix} v_p \\ v_q \\ v_r \end{bmatrix} + I^{-1} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times I \begin{bmatrix} p \\ q \\ r \end{bmatrix} \right) - \begin{bmatrix} bC_{l_{states}} \\ \bar{c}C_{m_{states}} \\ bC_{n_{states}} \end{bmatrix} \right\} \tag{4.7}$$

where the virtual inputs $\begin{bmatrix} v_p & v_q & v_r \end{bmatrix}^T$ are the time derivatives of the rotational rates of aircraft.

A reference model is defined as a second order dynamics

$$\frac{X_m(s)}{R(s)} = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{4.8}$$

where the natural frequency of the response can be defined as $\omega_n = 2rad/s$ and its damping coefficient $\zeta = 0.8$, for instance [97] suggests that a linear PID controller can be assigned with ω_n and ζ as $K_p = 2\zeta\omega_n$ and $K_i = \omega_n^2$ in cascade with the compensated non-linear dynamics.

Classical NDI's weakness is its sensitivity to modelling errors which results in erroneous inversion. However, this possibly unstable result is solved by using the real-time identified physical model, which has greater accuracy than a priori-based model, or compensating error signal with an adaptive control mechanism.

Here, the principle of hybrid flight adaptive control has been proposed to apply in each of the inner and outer loops of control to overcome the problem of modelling error. The OLS (Orthogonal Least Squares) method is operating and supplying the real-time identified model parameters, including failure characteristics when relevant. Model reference adaptive control is applied for direct adaptive control. More details of hybrid adaptive control are described in Chapter 8.

Furthermore, in cases of aircraft structural damage, the aircraft model structure of NDI changes due to the effect of the center of gravity shifted and the inertia properties varied as discussed in Eq. 3.34 of Chapter 3. As a result, the stability and control derivative matrices ($\Delta\vec{\mathbf{M}}_A$) and the mass and inertia derivative matrices ($\Delta\vec{\mathbf{M}}_M$) are usually unknown. Consequently, a flight control system is required to be able to compensate for the uncertain model dynamics of the damaged aircraft by using real-time identified physical model that depends on parameters as mentioned in Section 3.2.2 for aerodynamic terms and in Eq. 3.35 for additional inertia terms. Moreover, these parameters are also used to construct adaptive control mechanism for directive adaptive control.

4.2 Outer Loop of Autopilot Laws

Non-linear dynamic inversion control is used in the outer loop to regulate Euler roll (ϕ), pitch (θ) angle, side slip (β) angles and gravity acceleration (g) response

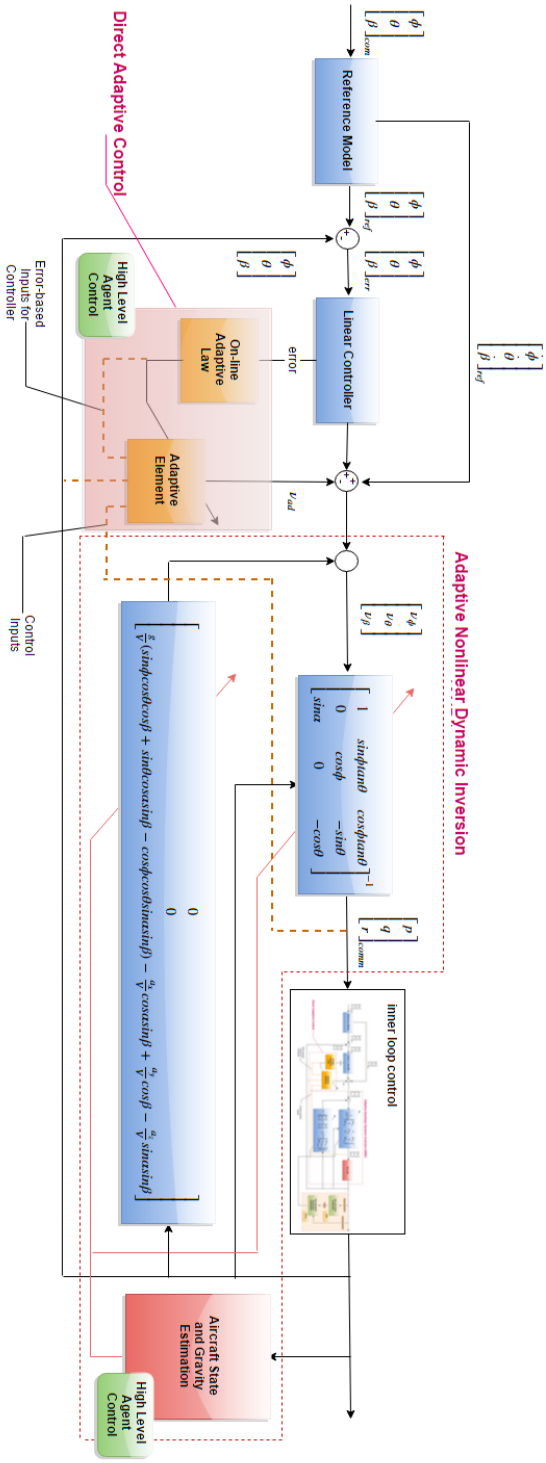


Figure 4.3: Hybrid adaptive Euler angle and sideslip angle control outer loop ASE=aircraft state estimation; AMI=aerodynamic model identification

following an input command from the guidance system. From the rotational kinematic equations (3.36) and the reconstructed equation of sideslip angle are discussed in Appendix B (IV).

$$\begin{aligned}
\dot{\phi} = &\quad p + \tan\theta(q\sin\phi + r\cos\phi) \\
\dot{\theta} = &\quad q\cos\phi - r\sin\theta \\
\dot{\beta} = &\quad psin\alpha - rcos\alpha + \tfrac{g}{V}(sin\phi cos\theta cos\beta + sin\theta cos\alpha sin\beta - cos\phi cos\theta sin\alpha sin\beta) - ... \\
&\quad \tfrac{a_x}{V}cos\alpha sin\beta + \tfrac{a_y}{V}cos\beta - \tfrac{a_z}{V}sin\alpha sin\beta
\end{aligned} \tag{4.9}$$

Rearranging Eq. (4.9) in the matrix form as:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\beta} \end{bmatrix} = \begin{bmatrix} 1 & sin\phi tan\theta & cos\phi tan\theta \\ 0 & cos\phi & -sin\theta \\ sin\alpha & 0 & -cos\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \beta_{states} \end{bmatrix} \tag{4.10}$$

where

$$\begin{aligned}
\beta_{states} = &\quad \tfrac{g}{V}(sin\phi cos\theta cos\beta + sin\theta cos\alpha sin\beta - cos\phi cos\theta sin\alpha sin\beta) - ... \\
&\quad \tfrac{a_x}{V}cos\alpha sin\beta + \tfrac{a_y}{V}cos\beta - \tfrac{a_z}{V}sin\alpha sin\beta
\end{aligned} \tag{4.11}$$

Solving the angular rate input $\begin{bmatrix} p & q & r \end{bmatrix}^T$ in form of inversion results in Eq. (4.12).

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & sin\phi tan\theta & cos\phi tan\theta \\ 0 & cos\phi & -sin\theta \\ sin\alpha & 0 & -cos\theta \end{bmatrix}^{-1} . \left(\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\beta} \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ \beta_{states} \end{bmatrix} \right) \tag{4.12}$$

In a similar way, the reference model in this case is defined to be second order model in Eq. (4.8). However, this loop has a slower response than the inner loop. Therefore, $\omega_{n_\theta} = 0.4\ rad/s$, $\omega_{n_\phi}, \omega_{n_\beta} = 1\ rad/s$ and $\zeta = 0.8$ are chosen. Linear controllers can be assigned with $K_p = 2\zeta\omega_n$ and $K_i = \omega_n^2$.

4.3 Altitude Control Laws

Altitude control enables an aircraft to maintain its required altitude. Here the NDI controller is proposed to send an angle θ demand to the outer flight control loop. A relation between the flight path angle (γ), the pitch angle (θ), the angle of attack (α), and the altitude or height (h) is given in Eq. (4.13)

$$\gamma = \theta - \alpha \tag{4.13}$$

The navigation equation is derived by expressing the aircraft velocity vector in earth z axes. The axes relate to body-axis components in Eq. (3.37) is shown in Eq. (4.14).

$$\dot{h} = usin\theta - vcos\theta sin\phi - wcos\theta cos\phi \tag{4.14}$$

in which body-axis velocity components are related to the airspeed (V), the angle of attack (α) and the sideslip angle (β) by

$$\begin{aligned} u = &\quad Vcos\alpha cos\beta \\ v = &\quad Vsin\beta \\ w = &\quad Vsin\alpha cos\beta \end{aligned} \tag{4.15}$$

Inserting Eq. (4.15) into Eq. (4.14) results in the dynamics of altitude.

$$\dot{h} = Vcos\alpha cos\beta sin\theta - Vsin\beta cos\theta cos\phi - Vsin\alpha cos\beta cos\theta cos\phi \tag{4.16}$$

Assuming that β and $\phi = 0$, due to turn coordination and considering only θ, therefore results in Eq. (4.17).

$$\dot{h} = Vcos\alpha sin\theta - Vsin\alpha cos\theta \tag{4.17}$$

Assuming that both θ and α are small, the equation can be approximated to Eq. (4.18).

$$\dot{h} = V(\theta - \alpha) \tag{4.18}$$

Rearranging this equation in form of inversion in order to send the theta command to flight control system results in Eq. (4.19).

$$\theta = \frac{\dot{h}}{V} + \alpha \tag{4.19}$$

4.4 Guidance Control Laws

Lateral guidance system for trajectory tracking is based on a regular waypoint tracking algorithm in [37] to send a roll angle command to the flight control system. Normally aircraft will perform coordinated turn to minimise undesirable aerodynamic loading of aircraft structure and payload considerations. Therefore, the sideslip angle command (β) is commonly defined as zero.

In order to calculate a desired roll command, it is assumed that the aircraft turns to coordinated. Therefore, the aircraft maintains sufficient lift to balance its weight, even though banked at an angle ϕ. This gives

$$L\cos\phi = W = mg, \qquad L\sin\phi = ma_{lat} \tag{4.20}$$

and

$$\phi_{des} = \tan^{-1}\left(\tfrac{a_{lat}}{g}\right) \tag{4.21}$$

Therefore, the desired roll command (ϕ_{des}) depends on the lateral acceleration (a_{lat}). This thesis utilizes L_1 lateral guidance control law for trajectory tracking [37] that can be written as

$$a_{lat} = \frac{2V^2}{L_1}\sin\eta \approx 2\frac{V}{L_1}\left(\dot{y_{L1}} + \tfrac{V}{L_1}y_{L1}\right) \tag{4.22}$$

where L_1 is the segment that binds the center of aircraft (O) to the point P on the desired path and y_{L1} is the perpendicular distance between the center of aircraft (O) and waypoint segment on the desired path as shown in Fig. 4.4.

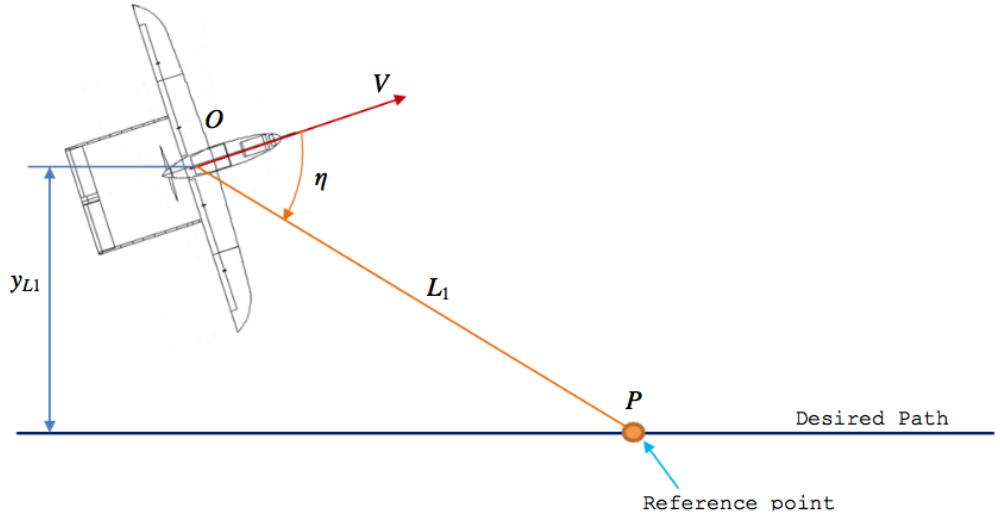


Figure 4.4: Lateral Guidance Law Geometry

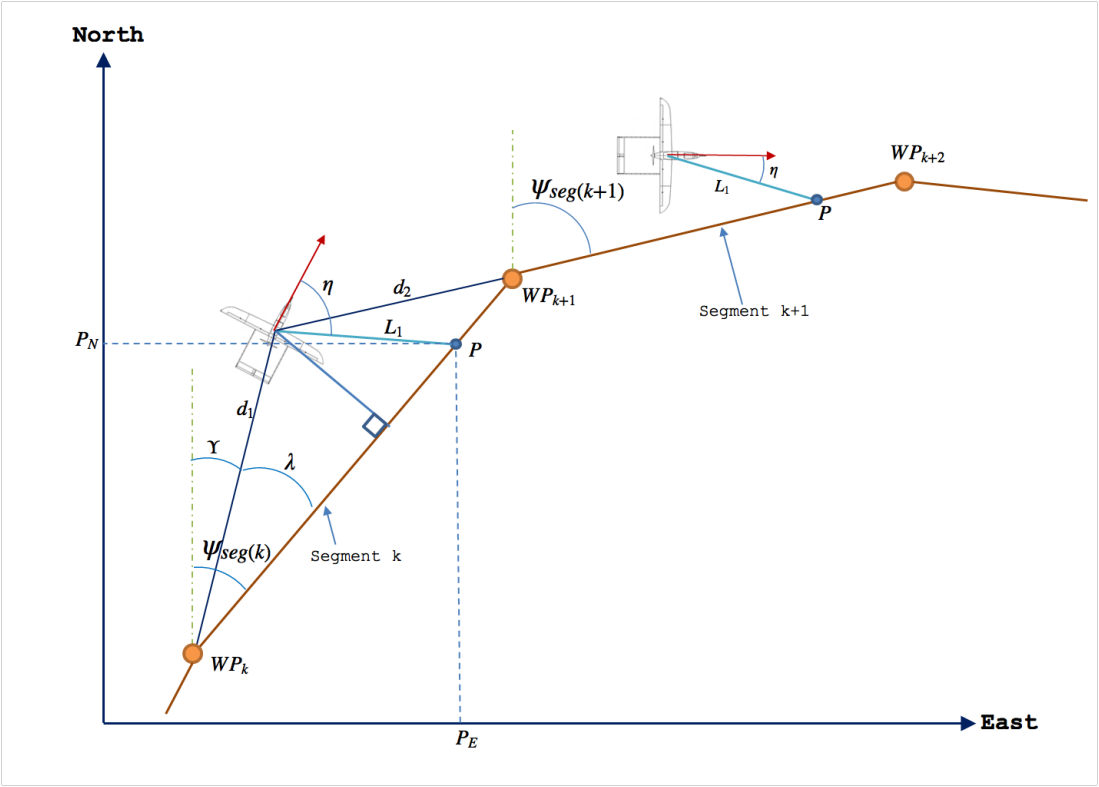


Figure 4.5: Waypoint Tracking

4.4.1 Computation of the vertical distance (y_{L1})

The position of the center of the aircraft is at (P_{A_E},P_{A_N}) on the north-east plane. The angle $\psi_{seg(k)}$, Υ and λ, and the distance d_1, y_{L1} can be determined with

$$\psi_{seg(k)} = \tan^{-1}\left(\frac{WP(k+1)_E - WP(k)_E}{WP(k+1)_N - WP(k)_N}\right) \in [-\pi;\pi] \tag{4.23}$$

$$\begin{aligned} \Upsilon &= \tan^{-1}\left(\frac{P_{A_E} - WP(k)_E}{P_{A_N} - WP(k)_N}\right) \in [-\pi;\pi] \\ \lambda &= |\psi_{seg(k)}| - |\Upsilon| \\ d_1 &= \sqrt{(P_{A_E} - WP(k)_E)^2 + (P_{A_N} - WP(k)_N)^2} \\ y_{L1} &= d_1 \tan\lambda \end{aligned} \tag{4.24}$$

Furthermore, one method to estimate $\dot{y}$ is to employ a backward finite difference method

$$\dot{y_{L1}} = \frac{y_{L1} - y_{L1_0}}{t_s} \tag{4.25}$$

where y_{L1_0} is the previous distance y_{L1} and t_s is the sampling time. However, this approach can result in an important error if t_s is either too small or too large. Another method is to utilize them to generate a smooth curve using a cubic, B-spline,or the polynomial least square method. This curve can be then differentiated at specific time to determine the derivative value. Also, the value of distance L_1 is a significant factor that acts as the gain of the controller. In this study, the distance L_1 is assigned be equal to 150m by default as follows in [37].

4.4.2 Logic for flight path switching

While the aircraft is flying, the reference point (P) also goes along the desired trajectory path. To switch the segment, the relationship between L_1 and d_2 is considered. In cases where the distance L_1 is longer than d_2, this means that the end of the current segment (Segment k) has been reached, and a new reference point has to be selected for the next segment (Segment k+1).

There are two cases to select L_1:

- Case of $|\lambda| > \frac{\pi}{2}$, then the distance L_1 is assigned as $L_1 = max(L_1, d_1)$.
- Case of $|\lambda| < \frac{\pi}{2}$, firstly, check the distance $y_{L1} = d_1 \tan\lambda$.
  - if $y_{L1} > L_1$ then the distance L_1 is assigned as $L_1 = y_{L1} * 1.1$.

4.5 Flight Path Planner

There are several planning techniques presented in a comprehensive summary of existing approaches that can be found in [70]. One of the popular planing methods applied to UAV is Dubins' algorithm [5, 23, 38]. 2D Dubins' curve [38] have been employed in this work due to simple computation and short calculate time. Dubins path is constructed with connection among particular part of circle curves and straight tangent lines having the form:

$$\{L \quad R \quad S\}, \tag{4.26}$$

where L and R is left and right turns (in geometry form of a part of the circumference of a circle) at a bank angle and S is a straight tangent line. The radius of arcs can be calculated by

$$R_{s,e} = \frac{V_{TAS}^2}{g \tan(\phi_{s,e})} \tag{4.27}$$

in which $R_{s,e}$ are the start and nearby radii of the arcs, V_{TAS} is the true airspeed of aircraft, g is the gravity acceleration constant, and $\phi_{s,e}$ are the start and nearby bank angles.

Consider the example illustrated in Fig. 4.6, the UAV is initially located at Point z_s and demanded to arrive at Point z_e. In the beginning, two circles ($C_{rs} : R$ and $C_{re} : R$), with the radius from Eq. (4.27), are drawn to be tangent at Point z_s. Next, the similar procedure is repeated at Point z_e as illustrated in Fig. 4.6(a). Then tangent lines are created to join the circumferences of these circles as shown in Fig. 4.6(b). Consequently, there are four paths (K_1, K_2, K_3,

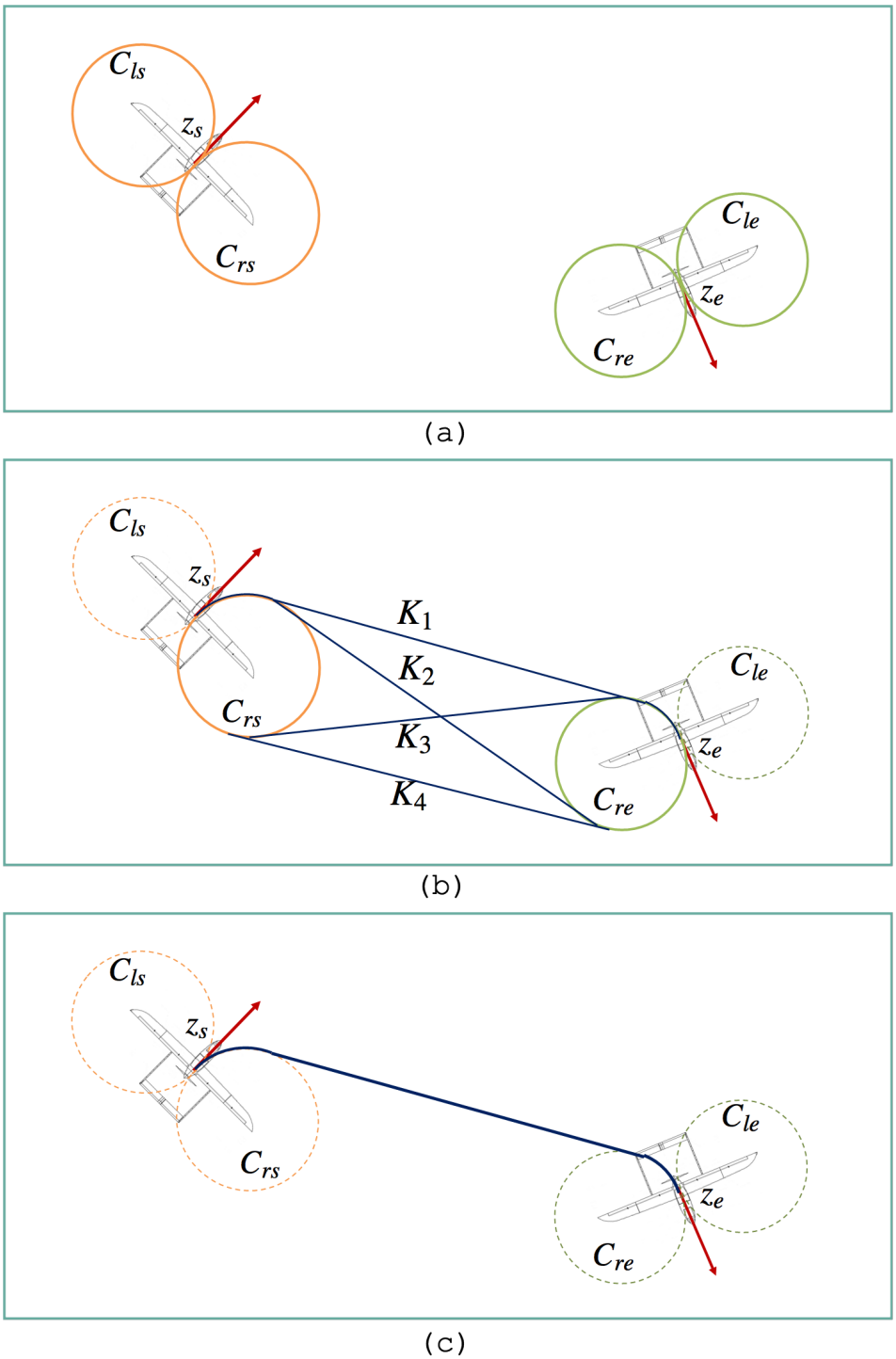


Figure 4.6: Step of Generating 2-D Dubins Path

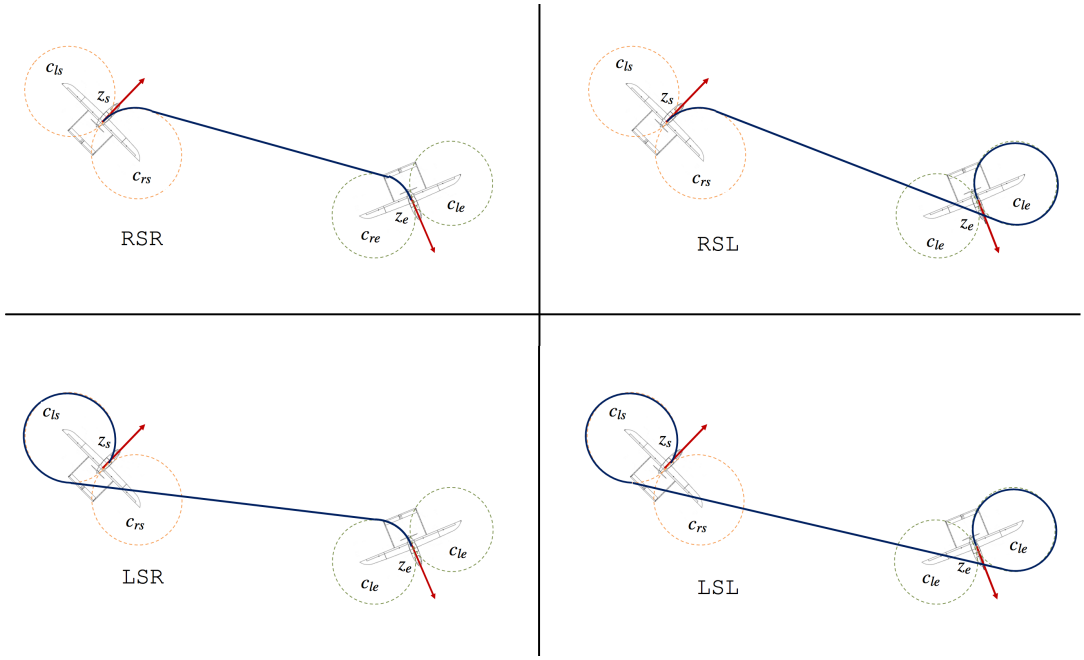


Figure 4.7: Dubin Curve Path Generating Step

and K_4) to link each circle pairs of center at (z_s, z_e) on each right side. However, only one of these paths, here called RSR, is accordant with the start and nearby headings of UAV [Fig. 4.6(c)]. In a similar process, three other paths (RSL, LSR, and LSR) can be obtained as demonstrated in Fig. 4.7. In addition, an example of MATLAB result of Dubins' curve is depicted in Fig. 4.8. Finally, the optimal path that considers in term of the shortest distance is selected from paths of RSR, RSL, LSR, and LSR.

4.6 Chapter Summary

This chapter has presented the fundamental autopilot architecture and mathematical formulations applied to this work. The structure of the attitude controller comprises the inner-loop and outer-loop design based on dynamic inversion. The inner-loop control functions to provide body-axis angular rate tracking and the outer loop portion is designed with commands being in the form of Euler angle and sideslip angle. Furthermore, L_1 lateral guidance system based on a waypoint

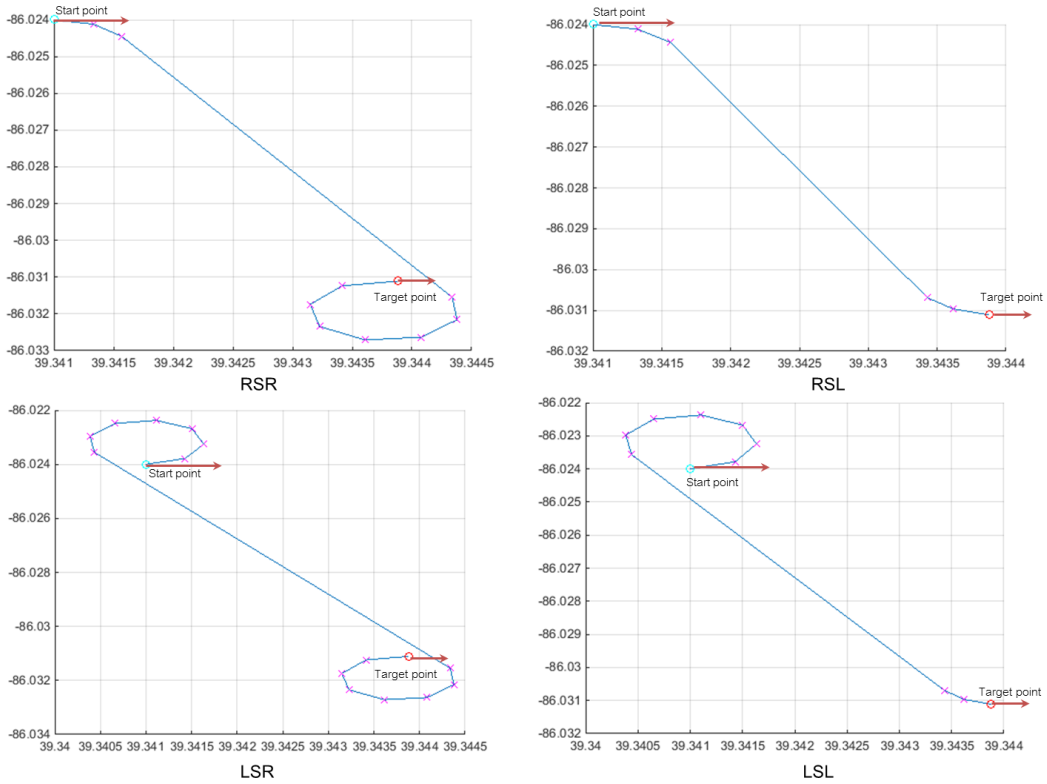


Figure 4.8: MATLAB Result Example of Dubins' Curve Path Generating

tracking is utilized to determine the roll angle command to the outer-loop loop of flight control system. The altitude control portion is also based on dynamic inversion in the form of a simplified kinematic equation to provide the pitch angle command for outer loop control. Moreover, the Dubins' curve algorithm has been considered to generate a flight path for guidance loop.

After explaining the detailed NDI set-up with lateral guidance law for autopilot control and adaptive control configurations that have been developed to compensate the inaccuracy of modelling of dynamic inversion will be discussed in the later chapter in a form of the action plans of an agent.

Chapter 5

Agent Theory

5.1 Intelligent Agent

There are various definitions of software agents due to the different aspects of agents. In this context an *agent* is a software system installed on a suitable computing hardware in a variable environment, which possesses an ability, through its sensors and actuators, to act autonomously without human intervention to achieve its designed goals in its environment [82].

A formal definition of agent [134, 139] is as follows. Let $S_{AG} = \{s_1, s_2, \ldots\}$ be the set of states of the environment in which the agent is placed, and $A_{AG} = \{a_1, a_2, \ldots\}$ be the set of actions that presents the capability of the agent. For any set X let X^* denote the set of all finite sequences of elements in X, and $\wp(X)$ the set of subsets of X. A common definition of agent is given by a model called "**standard agent**".

$$action : S^*_{AG} \rightarrow A_{AG} \tag{5.1}$$

which maps sequences of environment states to actions. The non-deterministic behaviour of the environment can be modelled as a function:

$$env : S_{AG} \times A_{AG} \rightarrow \wp(S_{AG}) \tag{5.2}$$

which maps the state of the environment and the agent's actions to a set of states that could result from the actions.

Therefore, a history sequence of the interaction between agent and environment can be presented in:

$$h : s_0 \text{ — } a_0 \rightarrow s_1 \text{ — } a_1 \rightarrow s_2 \text{ — } a_2 \rightarrow s_3 \ldots \tag{5.3}$$

Furthermore, the description of this standard agent can be developed by breaking the *action* into two functions that include *see* and *act*, and by entering a non-empty set of percepts and action maps sequences of percepts P_{AG} to actions and a set of states K_{AG} of the agent. Consequently, the agent's decision can be modelled as follows:

$$\begin{aligned} see : &\quad S_{AG} \rightarrow P_{AG} \\ next : &\quad K_{AG} \times P_{AG} \rightarrow K_{AG} \\ act : &\quad K_{AG} \rightarrow A_{AG} \end{aligned} \tag{5.4}$$

There are several agent architectures such as reactive agents, behavioural agents, logics-based agents, layered architectures and Belief-Desire-Intention (BDI) agents. They are distinct architectures but not exclusive, as most approaches to autonomous vehicles development can rely on more than one theme. This work has considered the BDI agent due to its capability to combine reactivity with long-term planning. BDI agent approach, which corresponds with decision making to serve intentions and pursue goals based on beliefs. The detail of a formal description of the BDI agent can be found in [139].

Let Bel, Des and Int denote large sets representing all possible beliefs, desires and intentions, respectively in which the agent could take. The state of a BDI agent can be indicated at any time by a triple (B, D, I), where $B \subseteq Bel$, $D \subseteq Des$ and $I \subseteq Int$.

The **beliefs** set represents the informational state of the agent (the world including itself and other agents). The reasoning cycle of a BDI agent starts with an update of the belief set by considering a mapping of a belief set and the perception of the environment. This can be expressed with a function called 'belief revision function' (brf):

$$brf : \wp(Bel) \times P_{AG} \rightarrow \wp(Bel) \tag{5.5}$$

The next step is an update of the desire set. The **desires** set denotes the mo-

tivational state of the agent that represents objectives or situations where the agent would accomplish by taking into account the belief set and the intention set from a previous cycle. The update is performed by a function called 'options generation function' (*options*):

$$options : \wp(Bel) \times \wp(Int) \rightarrow \wp(Des) \tag{5.6}$$

Once the agent stores a set of options, the next step is a deliberation process to decide what to do. The **intentions** set represents a deliberative set of options that the agent has decided to work towards. The update of the intentions set is performed with a 'filter' function, which maps all the knowledge from previous steps in which the agent has at current with the set of intentions:

$$filter : \wp(Bel) \times \wp(Des) \times \wp(Int) \rightarrow \wp(Int) \tag{5.7}$$

Finally, the function 'execute' is utilized to choose an intention that corresponds to a directly executable action, and then performs the action:

$$execute : \wp(Int) \rightarrow A_{AG} \tag{5.8}$$

There is no comprehensive agreement on how to define or measure an intelligent system. For the objective of this study, "intelligence"is defined as "the ability of a system to act appropriately in a dynamic environment, where an appropriate action is that which increases the probability of achievement and the achievement is the fulfilment of behavioural sub-goal that supports the system's ultimate goal " [82].

An autonomous aerial vehicle is placed in an urban environment where it has to deal with unexpected events that might happen, as well as consider highly complex realities such as aircraft health, and safety. To achieve the capability of adaptation, the agent designed for flight control systems will consist of a hierarchy of logic-connected procedures for improving flight performance, reliability and safety.

5.2 Agent-Oriented Programming

The concept of Agent Oriented Programming (AOP), and for instance the main characteristics of an AOP Jason programming language, are based on the concepts of the BDI approach to agency. An engineering agent software, fundamentally, can be abstracted into two interacting components: rational agent logic, or reasoning cycle, and abilities, or skills, that the agent can perform. Skills refer to particular routines that initiate action in connected subsystems, which can be either hardware or software based. The key idea of AOP is to program agents reasoning cycle in high-level terms and mostly with symbolic information regarding some ontology languages in order to implement the BDI framework.

Some of the most popular AOP languages available are Goal, PRS, and AgentSpeak/Jason. There is also natural language Programming (NLPr) that enables users to write code in natural language sentences, for instance, English sentences, using a predefined ontology. An example of this is sEnglish that is a part of an agent development environment call Cognitive Agent Toolbox (CAT) features in sEnglsih^{TM} software that can compile both the agent reasoning and the skills in different programming languages (MATLAB or C++) to interface with Jason BDI agent Architecture.

5.2.0.1 Jason

Jason is the extension of AgentSpeak, a logic-based AOP language based on the BDI agent paradigm. An agent's architecture is the software structure of parts and interactions which make it functions. One component of the Jason architecture is its belief base. The agent can continuously receive perceptions and communications from the environment and update the belief components accordingly. Another important element is the agent's goal set, which is archived by the execution of plans listed in the Jason program. The agent operates in reasoning cycles, which can typically run several times per second. And during a cycle it considers all new beliefs, performs logical implications, decides sub-goals and intentions, and starts executing some of the pre-written plans if they are triggered and their context is applicable.

A formal description of the *AgentSpeak using Jason* ($\mathcal{R}$) *based on rational BDI agent* is a tuple:

$$\mathcal{R} = \{\mathcal{B}, \mathcal{G}, \mathcal{M}, \mathcal{L}, \mathcal{E}, \mathcal{A}\} \tag{5.9}$$

where

- $\mathcal{B}$ is a total atomic belief set, of which $\mathcal{B}_t \subset \mathcal{B}$ is the currently 'true' belief set at any given time t and $\mathcal{B}_0 \subset \mathcal{B}$ is the set of initial beliefs. Belief b can be enhanced with internal variables as $b(x, y, \ldots)$ for a richer description. By the end of each reasoning cycle, the agent associates a value of either 'true' or 'false' or 'unknown' with every predicate $b \in \mathcal{B}$.
- $\mathcal{G}$ is a total atomic goal set, of which $\mathcal{G}_t \subset \mathcal{G}$ is the current goal set at any given time t and $\mathcal{G}_0 \subset \mathcal{G}$ is a set of initial goals. A goal g can be enhanced with internal variables as $g(x, y, \ldots)$ for a richer description. Goals can be created or erased during reasoning cycles.
- $\mathcal{M}$ is a set of possible messages generated by human supervisors and other agents as incoming and outgoing communications by the agent. The effect of messages is interpreted into beliefs in $\mathcal{B}_t$ during reasoning cycles.
- $\mathcal{L}$ is a set of logic-based implication rules.
- $\mathcal{E}$ is an ordered list of executable plans $\pi_1, \pi_2, \ldots$
- $\mathcal{A}$ is a set of executable actions, of which $\mathcal{A}_I \subset \mathcal{A}$ is the set of initial actions.

Each executable plan $\pi_i \in \mathcal{E}$ is of the format

$$\mathit{triggering\_event} : \mathit{content} \rightarrow \mathit{body}$$

where the plan can be potentially activated by an addition or a deletion of a triggering predicate b_i from the current belief base $\mathcal{B}_t$ (namely $+b_i$ or $-b_i$) or g_i from the current goal base $\mathcal{G}_t$ (namely $+!g_i$ or $-!g_i$) , if the *context* is also satisfied, which is a propositional logic formula of predicates from $\mathcal{B}$ or $\mathcal{G}$. *body* is a sequence of actions, predicates (added or taken away from the belief base using '+' or '−') and goals to be achieved in order to handle the event, i.e. one

of the following actions can stand for an a_i in a body of $a_1, a_2, ..., a_n$:

$+b(x, y, \ldots)$	where $b \in \mathcal{B}\backslash\mathcal{A}$, means an addition of a predicate to $\mathcal{B}$.
$-b(x, y, \ldots)$	where $b \in \mathcal{B}\backslash\mathcal{A}$, means an elimination of a predicate from $\mathcal{B}$.
$+!g(x, y, \ldots)$	where $g \in \mathcal{G}\backslash\mathcal{A}$, means an addition of an achievement goal to $\mathcal{G}$.
$-!g(x, y, \ldots)$	where $g \in \mathcal{G}\backslash\mathcal{A}$, means an elimination of an achievement goal from $\mathcal{G}$.
$+?g(x, y, \ldots)$	where $g \in \mathcal{G}\backslash\mathcal{A}$, means an addition of a test goal to $\mathcal{G}$.
$-?g(x, y, \ldots)$	where $g \in \mathcal{G}\backslash\mathcal{A}$, means an elimination of a test goal from $\mathcal{G}$.

The aforementioned reasoning cycle as shown in Fig. 5.1 is executed in 10 main steps as:

1. **Perceiving the environment**
   The first thing that the software agent does within a Jason reasoning cycle is to sense the environment in order to update its beliefs about the state of the environment. There is a component in a symbolic form as a list of *literals* that contains capability of perceiving the environment in the overall agent architecture. Each literal is a *percept* that represents a symbol of a particular feature of the current state of the environment.

2. **Updating the belief base**
   Once the list of percepts has been achieved, the belief base is updated to echo perceived changes in the environments using a function called *Belief Update Function* (**buf**).

3. **Receiving communication from other agents**
   At this step, the agent checks for incoming messages from other agents by *checkMail* method. Then the *message selection function* (S_M) is used to select, among all previously obtained messages with prioritisation, then selects the one that will be processed in current reasoning cycle.

4. **Selecting 'socially acceptable' messages**
   Before the messages are processed, they are delivered to a selection process to determine whether they can be accepted. This step can be done by *social acceptance function* (**SocAcc**).

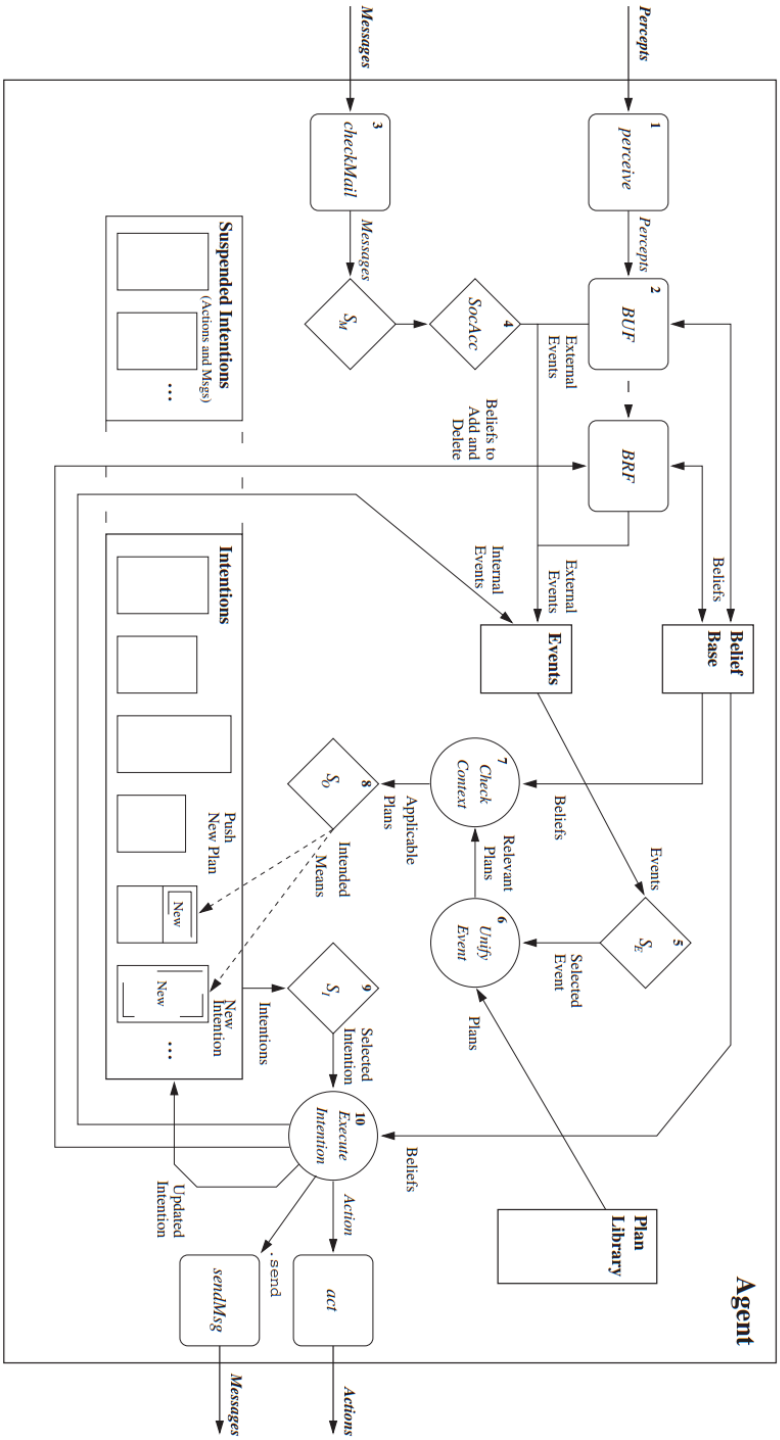


Figure 5.1: The *Jason* reasoning cycle [14]

5. **Selecting an event**
   In each reasoning cycle, the interpretor can handle only a single pending event. Therefore, if there is more than one pending event, a function called *Event Selection Function* (S_E), that may be customised, will give priority to certain events to define what it is believed to be more important at the current time. By default, the queue of events is first-in-first-out (FIFO) structure.

6. **Retrieving all relevant plans**
   This step is to find a plan that allows the agent to deal with that event. All plans triggered by the current event are retrieved from the **plan library**.

7. **Determining the applicable plans**
   From the previous stage, plans have a context part given information of agent. In this stage, the context of all the relevant plans are checked and the applicable ones are selected.

8. **Selecting one applicable plan**
   From all applicable plans, a function called *applicable plan selection function* (S_O) selects the one that the agent intends to execute, which is called *intended means*. The plan is stored in the *Set of Intentions*.

9. **Selecting an intention for further execution**
   At every reasoning cycle, the agent is only able to execute one action of the intended means stored in the set of intentions. It means that although the agent intends to perform a plan, it is not able to do it in one reasoning cycle. At this stage then a function called *intention selection function* (S_I) will choose one particular intention among those currently ready for execution.

10. **Executing one step of an intention**
    Finally, the agent executes one of the courses of action from the selected intention, which could be an internal or external action or a message to send to another agent.

Programming details can be found out in [14]. Furthermore, actions a_k can also be conditioned with logical statements such as:

`if` X `then` $a_1, a_2, \ldots$ `end`

`while` X `then` $a_1, a_2, \ldots$ `end`

where a_i can be one of the actions described above and X is a propositional logic formula of predicates from $\mathcal{B}$ and $\mathcal{G}$. Extension by '`elseif`' is also possible in the usual manner. In our enhancement of Jason, which we call *Jason+*, the action predicates $a(x, y, \ldots)$, where $a \in \mathcal{A}$, can be of type '`runOnce`' or '`runRepeated`'. When using '`runOnce`', the activated action thread is guaranteed to close itself after the action is completed and '`runRepeated`' needs the agent to issue the following action $a_{stop}(x, y, \ldots)$ in order to stop the current one. *Jason+* also permits the use of external actions in context formulae [72].

5.2.0.2 Natural Language Programming

Natural Language Programming (NLPr) of agents [72] is a way of programming using natural human language, e.g. English. sEnglish, which stands for system-English, is an NLPr language designed to make agent reasoning more anthropomorphic and enable thought sharing processes between agents and humans. An agent's reasoning is described in a readable sEnglish document with title, contents, sections and subsections and references, which compiles into Jason. sEnglish also allows the definition (i.e. programming) of domain specific agent skills in MATLAB, which can be compiled into C++ under ROS and other lower level programming languages.

When a *Jason+* program is written in sEnglish language sentences, then sentences defined with meaning (in terms of executable code) are encapsulated within square brackets. There are however two types of sEnglish sentences used: executable actions and mental notes of the agent to itself. The use of sentences is bound together by the constraint of a single ontology to be defined within the sEnglish document. The advantages of using this kind of language for programming is that the designers are enforced to write programs in simple and elegant way which lends itself to easy interpretation by non programmers such as lawyers investigating autonomous system decision making in the near future. This also means that even people who are inexperienced in logic programming can understand and perhaps modify a NLPr Jason+ program. The main characteristics

of the simple sEnglish syntax for Jason+ (there is no "grammar" as there is in other programming languages), can be described as follows:

- Sentences within square brackets '`[...]`' represent actions/skills that the agent can execute. These actions must be clearly defined within the sEnglish document of the agent.
- Sentences within square brackets preceded by the 'hat' operand '`^[...]`' represent internal literals for Jason+, i.e. *mental notes* of the agent.
- The 'tilde' symbol '`~`' represents a negation of the sentence that it precedes.
- The symbols '`+`' and '`-`' represents an *addition* or a *deletion* of a literal from the belief base.
- Logical rules are defined statements using if, *then*, *implies*, *and*, *or*, ... and sentences of mental notes.

The relevance of sEnglish in the context of the topic of this chapter is that programming systems like this sEnglish based Jason+ will eventually help to achieve legal certification of *human-equivalent* software based intelligent autopilots by aviation lawyers. This chapter is a precursor to such an effort. More detailed descriptions of the sEnglish development processes can be found in [72].

5.3 Chapter Summary

This chapter has defined the fundamental terms and theory related to agent systems, especially BDI agents. It has discussed agent-oriented programming explained how Jason is based on the BDI agent paradigm with the AgentSpeak language. In addition, the methodology using the NLPr software sEnglish has been formally presented. This method aids us to move away from the low-level programming to a more user-friendly abstraction. sEnglish sentences are used in Jason programming in a way as described in this chapter.

Part III

Development of Control Agents for UAVs

Chapter 6

Agents for UAV Autopilot Systems

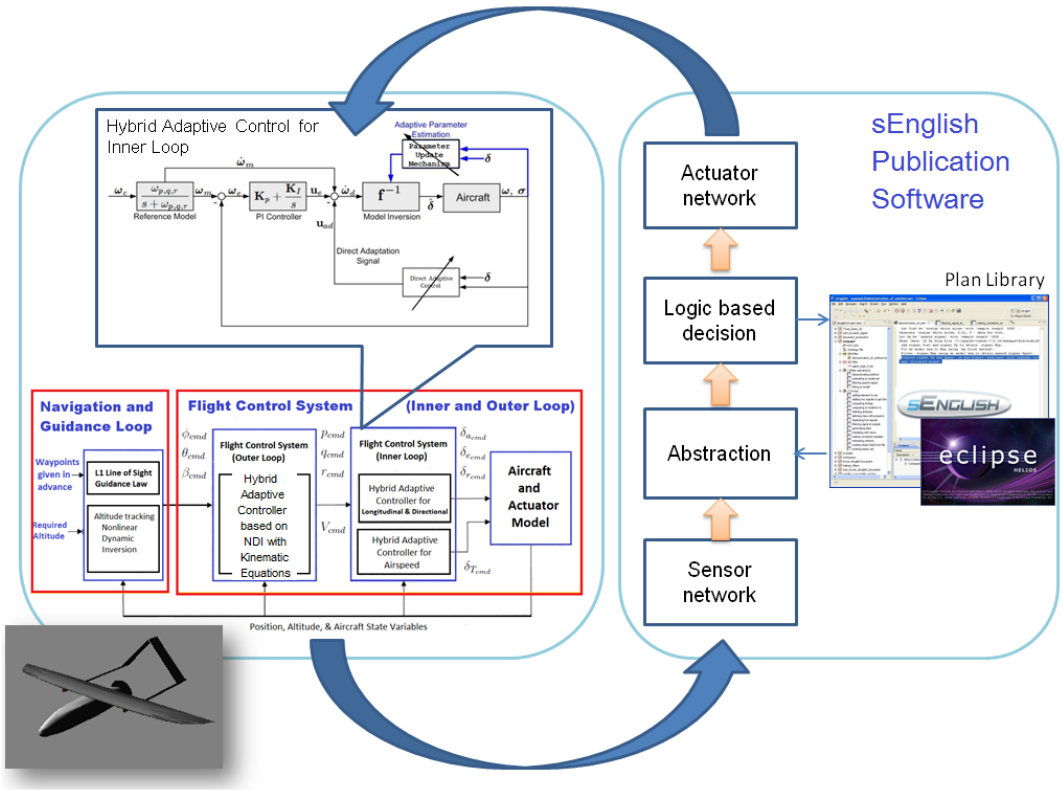


Figure 6.1: Concept Diagram for Developing Agent with sEnglish Publication Software.

A block diagram, which describes the application of an intelligent agent for UAV supervision using an sEnglish publication software, is illustrated in Fig. 6.1. The sEnglish publication software enables users or engineers to develop a rational agent including high abstraction level objects and relationships in the world model and agent logic reasoning rules in terms of English sentences. The agent can provide more adaptable behaviours for autopilot systems in complex and adverse situations, for example, deficiencies in an initial controller and system degradation due to accidental control surface damage. The programming language paradigm and procedure in sEnglish publication software splits into three layers. NLPr in terms of English sentences compiles into embedded MATLAB code that is able to compile into standard C/C++ for the robot operating system (ROS). At the top level, abstractions are expressed in natural language programming (sEnglish) layers to define decision operations in the format of an sEnglish document. Then the sEnglish document can be used to compile into declarative rational agent code which runs on a Java-based interpreter that is an extension of AgentSpeak (*Jason*).

A functional architecture of the agent in sEnglish, which is implemented in this thesis, contains a high-level reasoning system connected through an abstraction layer to low-level sensor and control systems as illustrated in Fig. 6.1. More precisely, the system structure in sEnglish consists of a physical engine, an abstraction layer , a continuous engine, and a reasoning engine as demonstrated in Fig. 6.2. The physical engine comprises of features that can sense and produce changes in the environment that may be real or simulated with the real-time sensing and control processes. The physical engine interfaces with an abstraction engine. In abstraction layer, perception information, that is sampled from the physical engine, is filtered for sensing abstraction as the belief based that belongs to the reasoning engine. Then the rule-based rational engine is responsible for dictating processes occurred with the physical engine. The reasoning engine, which is the highest level layer within the system and stands on traditional BDI agent (*Jason*), contains a Sense-Reason-Act loop within itself. Sensing here is associated with the perception of changes within the belief base modified by the abstraction layer. Additionally, sensing in the ration engine may lead to reasoning over new events and result in actions necessitating communication with either the

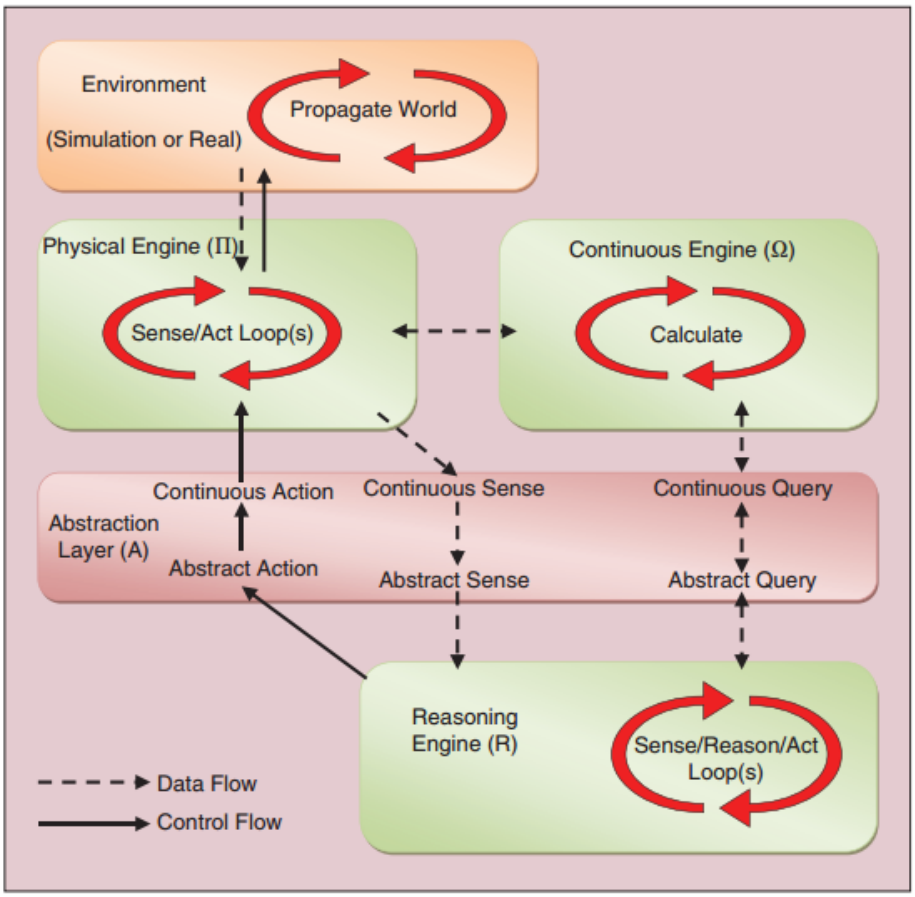


Figure 6.2: System Strucute Diagram [74].

physical engine or the continuous engine. The continuous engine, which supports the reasoning engine, is employed to execute sophisticated numerical procedures that may be utilised to support reasoning processes in the rational engine or produce data required for a physical process within the physical engine. Moreover, a separate connection channel occurring between the physical engine and the continuous engine is used to enable direct information transferred between both engines without intervention from the abstraction engine. All actions, which are executed by the reasoning engine, are transferred into the abstraction engine for reification. In this structure, the reasoning engine relied on Jason deals with discrete information. The physical engine and continuous engine are conventional systems, while the abstraction layer provides the vital interconnection with all

these elements via hosting primary communication channels and translating between continuous and abstract data. The connection between the components in the architecture is encapsulated by a language independent operational semantics such as sEnglish language [36, 72, 74].

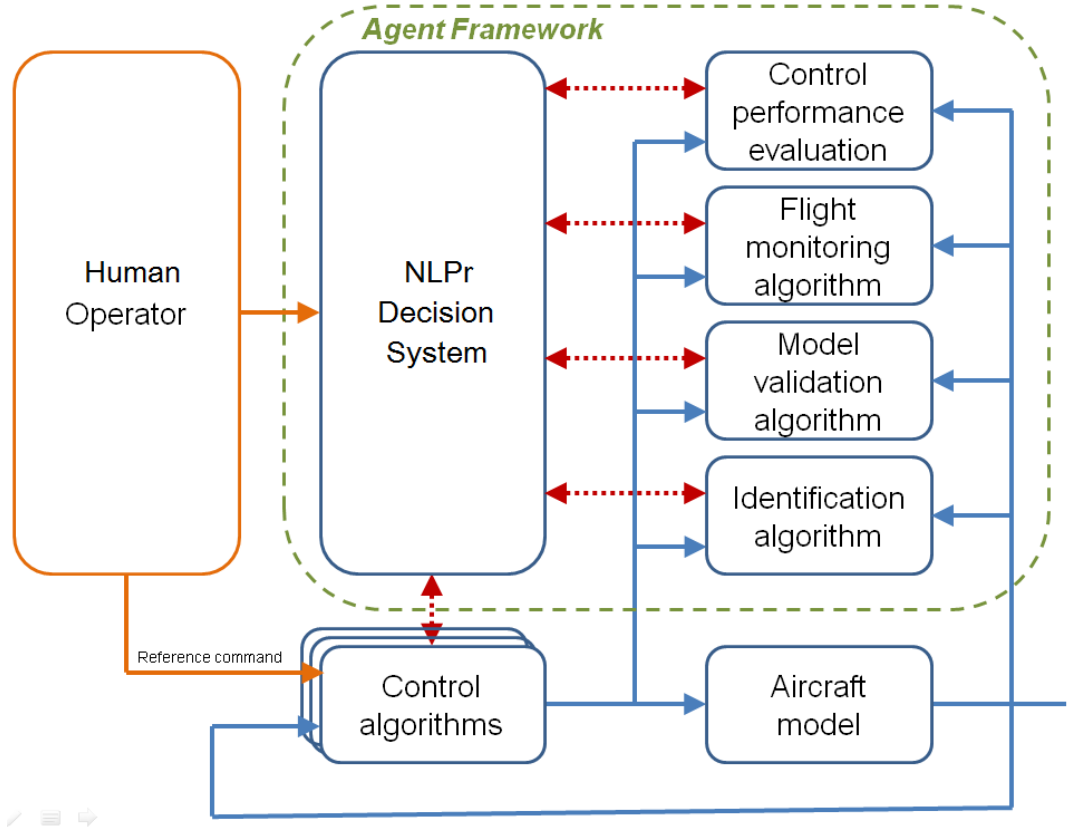


Figure 6.3: Concept Diagram for Applying Agent-Oriented Approach to Autopilot System.

Fig. 6.3 describes a block diagram of a more specific agent application scheme for an intelligent autopilot. A human operator including pilot or ground control station can directly command to the control system in manual flight mode or switch to use an intelligent control agent mode by assigning the desired waypoints to control agent system. In the intelligent agent mode, the agent cooperates with a conventional control system based on NDI controller in order to provide the advanced capability of maintaining stability and acceptable performance in the event of deficiencies in controllers and system degradation. The agent with multi-

threaded execution integrates signal processing, including system identification, model based indirect adaptive control, direct adaptive control, model validation, flight state monitoring, and control performance evaluation. The Jason+ based BDI agent is capable of making decisions using reasoning by logic rules (hence, it can be described by the adjective "rational"), executing actions as well as calculating the workload for each of the processes to be executed, way beyond adaptive/intelligent control schemes used in avionics.

In order to ensure proper adjustment of the control system, the agent performs a real-time identification/monitoring of abnormal dynamics, where it validates against the current model and decides whether aircraft aerodynamic parameters for the NDI controller have changed significantly. It computes non-linear dynamic inversion control for the autopilot system and then executes a reasoning-based action to either compensate or update the parameters of a nonlinear dynamic inversion controller to trigger reconfiguration of the flight controller. Furthermore, the agent has a capability to monitor error signals in angular rates and angular angle, whether or not the value of the error grows using frequency dependent model validation approach, in order to evaluate the performance of the flight control system. The agent may then decide, base on rules, to execute a plan to stabilise the aircraft by triggering a reset mechanism of direct adaptive control or activating a parameter compensation algorithm for the NDI controller of both inner and outer loop of flight control in adverse flight conditions as shown in Fig. 4.2 and 4.3, respectively.

A key aspect to develop the agent is to define suitable discrete abstractions for sensing and action of the continuous time/value signals of the aircraft dynamical system in order to supply discrete abstractions for decision making. The perceptual abstraction process is to filter any discrete information from the environment and on-board system to supply the inference rules in the rational engine. As shown in Fig 6.2, the perception data, which is sampled from the physical engine, is sent to the abstraction engine. Then the abstraction engine, which might call on the continuous engine to do calculations by using functions as be explained in Chapter 9, is responsible for discrete information as the beliefs belonging to the rational engine. To express statements and decisions for system reconfiguration in a readable format for development engineers, and also to support real-time

reasoning feedback by agents to human supervisors, signal abstractions and decision rules in the rational engine have been developed in terms of sEnglish in NLPr [72, 133] as described in Chapter 10. Moreover, the rational engine of the rational agent is a rule-based decision-making process about both the system configuration and its parameters to achieve goals. Based on agent rules, the rational engine can generate data that are transmitted to the physical engine or also call the continuous engine (via the abstraction engine) to perform complex numerical procedures /functions as described in Chapter 8, for instance, reconfigure or trigger new controllers or can send instructions directly to the physical engine.

Chapter 7

UAV State Estimation

7.1 Problem Formulation

Air flow angles including the Angle Of Attack (AOA) and Side-Slip Angles (SSA) are the most relevant variables of the aircraft state vector which are necessary for aircraft parameter estimation and flight control system. However, AOA and SSA sensors are usually available for only commercial aircraft or large UAVs which have enough space and relative size for installation of the angle of attack and sideslip sensors. AOA and SSA sensors can produce a significant effect on the dynamics of smaller aircraft. Furthermore, air flow angle sensors require extensive calibration for good accuracy in practical terms because the air flow angle sensors cannot be fitted at the center of gravity of an aircraft [8, 22]. Moreover, some supersonic aircraft are unsuitable to carry air flow angle sensors on their platforms due to the heat caused by surface friction with the air [63].

From the literature review, some methods have been published to reconstruct air flow angles from other sensor measurements such as (1) data reconstruction analysis [66], (2) filtering techniques, and (3) numerical integration of air flow angle reconstruction equations with small perturbation assumption and high-pass filtering [87]. However, the methods in (1) and (3) can only be used for short periods because of the drift in the reconstructed quantities over time due to the integrated effects as clearly stated in [66]. In (2) the filtering method relies on various expensive measuring devices for airspeed and tracking radar, inertial

measurement unit (IMU) and GPS to estimate the airflow angles for supersonic aircraft [63]. Alternative filtering approach utilizes an Extended Kalman-Filter (EKF) based on the nonlinear kinematics and measurement model in combination with an aerodynamic model [24, 25, 107].

Due to the deficiencies of past methods in terms of heavy reliance on sensors, our objective is to propose a new technique of air flow angle estimation, which is based on a general dynamical principle from inertial data and using sensors only for (1) magnetic compass data, (2) GPS data, (3) IMU data and, (4) Euler angle data from full GPS/INS EKF-based aircraft state estimation [11].

7.2 Methodology Proposal

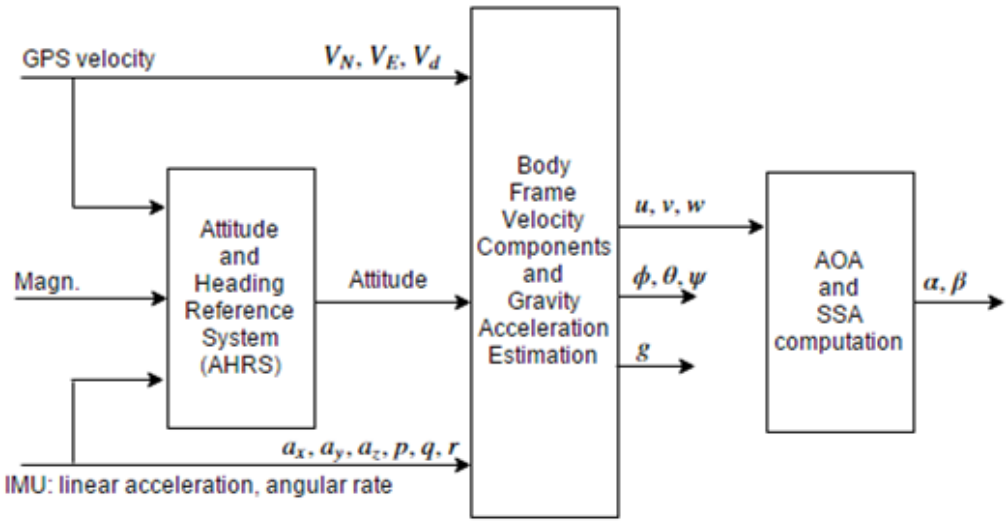


Figure 7.1: Diagram of Estimator.

This method is presented to estimate AOA and SSA of a fixed-wing UAV using only kinematic relationships with an Extended Kalman Filter (EKF), but avoiding the requirement to know aerodynamic models or other aircraft parameters as illustrated in Fig. 7.1. It has the following modules

- AHRS estimates the attitude (roll, pitch and yaw angles ϕ, θ, ψ) based on the kinematic relationships. This could be based on the EKF, e.g. [11, 64] or non-linear observers, e.g. [46]

- The "Body Frame Velocity Components and Gravitational Acceleration Estimation Module"estimates the body frame relative aircraft velocity and gravitational acceleration based on aircraft kinematics relationships equation in Section 3, see the next section where it is discussed in detail, such aircraft body frame relative velocity can be input to the computation of AOA and SSA.

- The AOA and SSA computation module applies with Eq.(7.4).

A diagram depicting the principle of air flow angle estimation is shown in Fig. 7.1. Inputs include sensor measuring signals, such as linear accelerations, and angular rates while outputs are ground speed, and Euler angle. Together they are sent to an air flow angle estimation module in order to estimate body velocity components and gravity acceleration via extended Kalman filtering using a general dynamical equation and then use body velocity components to construct an angle of attack and sideslip angle. It is assumed that the Euler roll, pitch, and yaw angle can be estimated using state estimation algorithms according to [11]. In our work the estimated angle of attack and sideslip angles will be utilized in conjunction with control surface deflections, airspeed and inertial data from IMUs to estimate aircraft aerodynamic stability and control derivatives and coefficients (e.g. C_{L_0}, C_{L_α},..., $C_{n_{\delta_a}}$, $C_{n_{\delta_r}}$) for non-linear aircraft dynamic equations in real time and also be employed as feedback states for NDI control as descried in the next chapter.

7.3 Air Flow Angle and Gravitational Acceleration Estimation Using Extended Kalman Filtering

This filtering technique is not utilized only for estimating the aircraft states but also for predicting the desired parameter. Parameter estimation through the filtering approach is an indirect procedure. The main idea is to transform the parameter estimation problem into a state estimation problem. Additional state variables, which are artificially defined with the unknown parameters, is expanded the system state vector. The generic system dynamics are presented
(Note that such method which utilizes a continuous model for prediction with a discrete filtering algorithm is known as the continuous-discrete filtering problem)

$$\begin{aligned} \dot{x}(t) &= f[x(t), u(t), \beta_p] + F\varepsilon(t), \quad x(t_0) = x_0 \\ y(t) &= g[x(t), u(t), \beta_p] \\ z(k) &= y(k) + G\eta(k) \qquad\qquad k = 1, .., N \end{aligned} \tag{7.1}$$

where

x is the state vector with initial value x_0 at time t_0.

u is the input vector.

z is the measurement vector.

β_p is the unknown system parameters.

f and g are the general non-linear real-valued function.

F and G represent the process and measurement noise matrices.

ε and η are the process and measurement noise vectors (assumed to be zero mean white Gaussian noise), respectively.

The unknown parameter vector Θ that consists of unknown system parameters β_p is assumed to be constant. Therefore, the derivative of the unknown parameter vector is defined as

$$\dot{\Theta} = 0 \tag{7.2}$$

Then, the extended state vector is defined as

$$x_e = \begin{bmatrix} x \\ \Theta \end{bmatrix} \tag{7.3}$$

Consequently, the extended system can be presented as:

$$\begin{aligned} \dot{x}_e(t) &= f_e[x_e(t), u(t)] + F_e \varepsilon_e(t) \\ &= \begin{bmatrix} f[x(t), u(t), \beta_p] \\ 0 \end{bmatrix} + \begin{bmatrix} F & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \varepsilon(t) \\ 0 \end{bmatrix} \\ y(t) &= g_e[x_e(t), u(t)] \\ z(k) &= y(k) + G\eta(k) \end{aligned} \tag{7.4}$$

where the extended variables are denoted by subscript e.

7.3.1 Extended Kalman Filtering

The EKF algorithm [28, 45], which can be applied to non-linear problems by using local linearisation at each iteration to approximate the non-linearities, is represented in Eq. (7.5) - (7.11) as follows (here the notation "bar" and "hat" to denote the predicted and corrected variables respectively):

Extrapolation:

$$\bar{x}_e(k) = \hat{x}_e(k-1) + \int_{t(k-1)}^{t(k)} f_e[\hat{x}_e(t), \bar{u}]\, dt \tag{7.5}$$

$$\bar{P}_e(k) \approx \Phi_e(k) \hat{P}_e(k-1) \Phi_e^T(k) + \Delta t . F_e F_e^T \tag{7.6}$$

with the initial conditions

$$\bar{x}_e(1) = x_{e0}, \quad \bar{P}_e(1) = P_{e0} \tag{7.7}$$

$$A_e(k) = \left. \frac{\partial f_e}{\partial x_e} \right|_{x_e = \hat{x}_e(k-1)} = \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial \Theta} \\ 0 & 0 \end{bmatrix}_{x_e = \hat{x}_e(k-1)} \tag{7.8}$$

$$\Phi_e(k) = exp(A_e \Delta t) \tag{7.9}$$

Update:

$$\begin{aligned}
\bar{y}k) &= g_e[\bar{x}_e(k), u(k)] \\
K_e(k) &= \bar{P}_e(k) C_e^T(k) [C_e(k) \bar{P}_e(k) C_e^T(k) + GG^T]^{-1} \\
\hat{x}_e(k) &= \bar{x}_e(k) + K_e(k)[z(k) - \bar{y}(k)] \\
\hat{P}_e(k) &= [I - K_e(k) C_e(k)] \bar{P}_e(k) \\
&= [I - K_e(k) C_e(k)] \bar{P}_e(k) [I - K_e(k) C_e(k)]^T + K_e(k) GG^T K_e^T(k)
\end{aligned} \tag{7.10}$$

where $C_e(k)$ is the linearised measurement matrix as represented

$$C_e(k) = \left. \frac{\partial g_e}{\partial x_e} \right|_{x_e = \hat{x}_e(k)} = \left[\frac{\partial g}{\partial x} \quad \frac{\partial g}{\partial \Theta} \right]_{x_e = \hat{x}_e(k)} \tag{7.11}$$

7.3.2 Application to flight vehicles: Fixed-wing platform

This method bases on the algorithm of [127] to estimate gravitational acceleration as follows. From the non-linear aircraft kinematics equations:

$$\begin{aligned}
\dot{u} &= rv - qw + \frac{\bar{q}S}{m} C_X - g \sin\theta + \frac{T_x}{m} \\
\dot{v} &= pw - ru + \frac{\bar{q}S}{m} C_Y + g \cos\theta \sin\phi \\
\dot{w} &= qu - pv + \frac{\bar{q}S}{m} C_Z + g \cos\theta \cos\phi + \frac{T_z}{m}
\end{aligned} \tag{7.12}$$

Using $\bar{v} = [u, v, w]^T$ in the dynamical equations for the measurable acceleration vector $\bar{a} = [a_x, a_y, a_z]^T$ in the body frame are

$$\bar{a} = \dot{\bar{v}} + \bar{\omega} \times \bar{v} - \frac{\bar{F}_G}{m} = \frac{1}{m} (\bar{F}_A + \bar{F}_T) \tag{7.13}$$

Substituting translational acceleration measurements for the applied forces results in the translational kinematic equations in body axes as shown in Eq. 3.20:

$$\begin{aligned} \dot{u} &= rv - qw - g\sin\theta + a_x \\ \dot{v} &= pw - ru + g\cos\theta\sin\phi + a_y \\ \dot{w} &= qu - pv + g\cos\theta\cos\phi + a_z \end{aligned} \tag{7.14}$$

From definition in Eq. 3.16, sideslip angle, and angle of attack can be computed from u, v, and w using

$$\begin{aligned} \beta &= \sin^{-1}(\frac{v}{V}) \\ \alpha &= \tan^{-1}(\frac{w}{u}) \end{aligned} \tag{7.15}$$

The standard kinematic equations in Section 3.3 are as follows:

$$\begin{aligned} \dot{\phi} = &\quad p + \tan\theta(q\sin\phi + r\cos\phi) \\ \dot{\theta} = &\quad q\cos\phi - r\sin\theta \\ \dot{\psi} = &\quad q\sin\phi\sec\theta + r\cos\phi\sec\theta \end{aligned} \tag{7.16}$$

Furthermore, gravity (g) is considered as a constant:

$$\dot{g} = \quad 0 \tag{7.17}$$

The body axes translational kinematic equations in Eq. (7.14) and kinematic equations in Eq. (7.16-7.17) defined as f_e are applied with filtering technique in order to estimate a state. The state vector is formed as $x = [u,\ v,\ w,\ \phi,\ \theta,\ \psi]^T$, the unknown parameter vector as $\Theta = [g]$ and the input vector is defined as $u = [p,\ q,\ r,\ a_x,\ a_y,\ a_z]^T$.

Therefore, the extended state vector presents as

$$x_e = [u \quad v \quad w \quad \phi \quad \theta \quad \psi \quad g]^T \tag{7.18}$$

where the observation equations defined as $g_e[x_e(t), u(t)]$ are:

$$\begin{aligned}
V_N &= u\cos\theta\cos\psi + v(\sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi) + w(\cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi) \\
V_E &= u\cos\theta\sin\psi + v(\sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi) + w(\cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi) \\
V_d &= -u\sin\theta + v\sin\phi\cos\theta + w\cos\phi\cos\theta \\
\phi_m &= \phi \\
\theta_m &= \theta \\
\psi_m &= \psi
\end{aligned} \tag{7.19}$$

The Runge-Kutta algorithm can be employed in Eq. (7.5) and the state transition matrix ($\Phi(k)$) can be approximated using Pade approximation. P_0 represents the confidence in the initial state estimates. The value of P_0 can be initialised with high values in the absence of any priori knowledge. The value of the measurement noise covariance matrix (GG^T) can be specified using laboratory measurements of sensors in order to ensure good noise filtering. However, the value of the process noise covariance matrix (FF^T) is more difficult to determine and a trial and error method based on engineering judgement is employed if no other suitable method is found. Adaptive filtering, which has the capability to adapt to unknown noise characteristics, is not considered in this work.

7.4 Simulation Results

7.4.1 Aerosonde UAVs

Flight data from a non-linear simulation of the Aerosonde UAVs was used for real-time aircraft system identification with air flow angle estimators. A 3-2-1-1 sequence input on the elevator, aileron and rudder were applied to the simulation, and Extended Kalman filter with the translational kinematic equation in body frame and kinematic equations in (7.14) and (7.16) were employed to estimate the body axis velocity component using simulated measurement states of the Aerosonde model. And then the air flow angle data is calculated from (7.15)

with the velocity component in body frame from the estimation. Furthermore, in this simulation experiment approximately five percent Gaussian random noise was inserted to all outputs of Aerosonde simulation.

Figure 7.2 illustrates a comparison between simulated-measured and estimated sideslip angle and angle of attack in the time domain from the Aerosonde model and extended Kalman filtering, respectively with Gaussian random noise added to all simulation output.

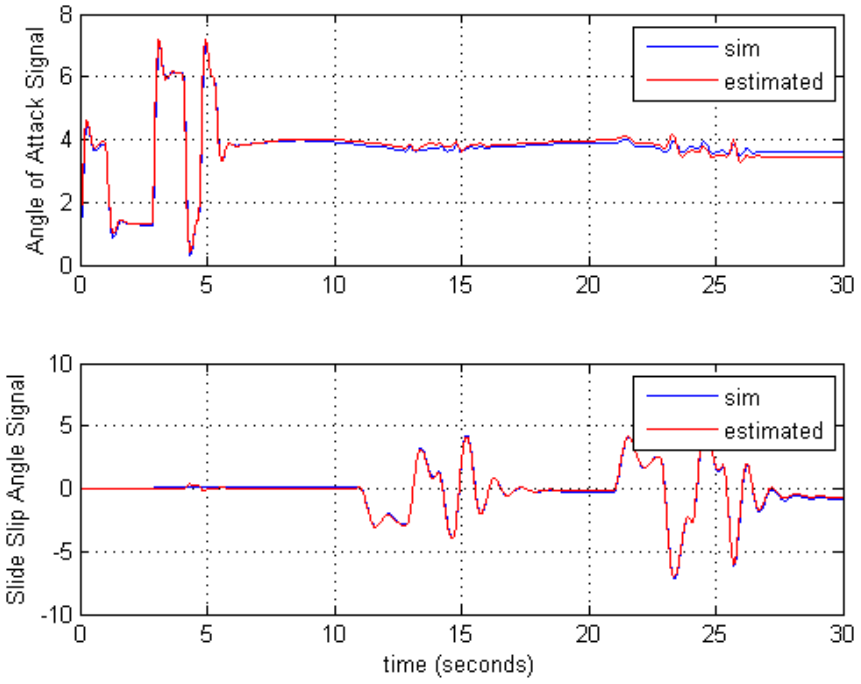


Figure 7.2: Air Flow Angles of Simulated and Estimated Data via the Proposed Method

Figure 7.3 demonstrates the comparison of magnitudes of simulated and estimated air flow angles each frequency using flight data with Gaussian random noise added. The two graphs are very similar, and there is almost no difference which indicates that good aircraft parameter results could be obtained using frequency domain system identification.

Figure 7.4 shows the comparison of gravitational acceleration from Aerosonde simulation and estimation method in the time domain using flight data with

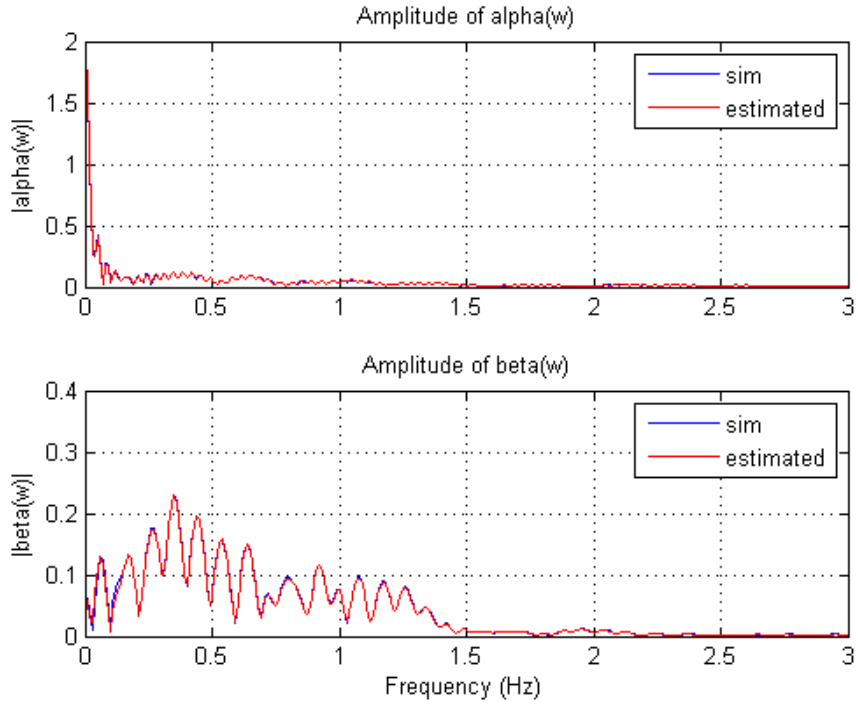


Figure 7.3: Magnitude of Reconstructed Air Flow Angles in Frequency Domain Between Aerosonde Simulation and The Proposed Estimation

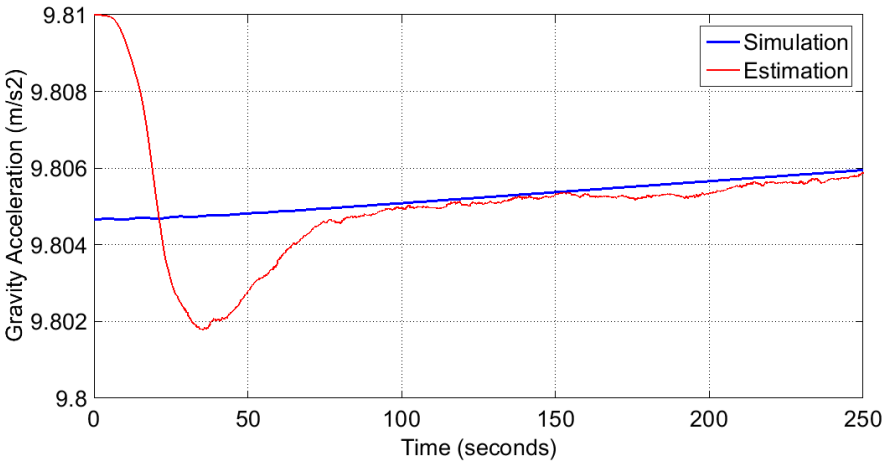


Figure 7.4: Comparison between Simulated and Estimated Gravitational Acceleration Data via Extended Kalman Filtering

Gaussian noise added. This figure demonstrates that the estimated gravitational acceleration converges to the simulated value as time progresses.

Therefore, it means that the estimated gravitational acceleration is closer to the real value of the environment; the air flow angle from the estimation will be more accurate. This method is suitable for real implementation because the value of gravitational acceleration is not constant in the atmosphere.

7.4.2 NASA Twin Otto Aircraft

Moreover, this air flow angle estimation method was also verified with flight data measurements of NASA Twin Otter aircraft. The flight data measurements can be obtained from an example of the System Identification Program for Aircraft (SIDPAC) of Klein and Morelli [66].

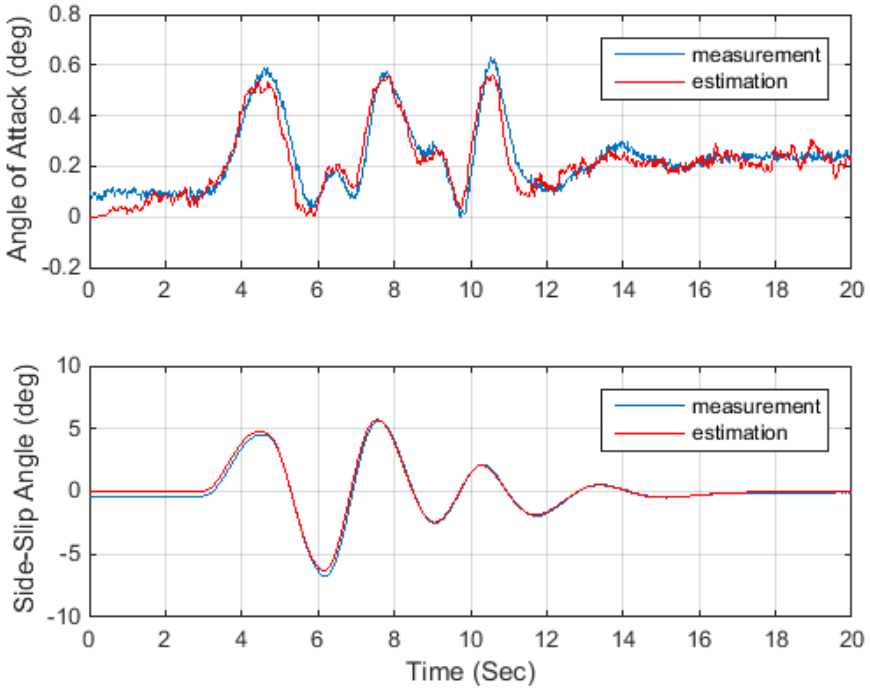


Figure 7.5: Air Flow Angles of Measured and Estimated Data of NASA Twin Otter aircraft via the Proposed Method

Figure 7.5 illustrates a comparison between measured and estimated sideslip

angle and angle of attack in the time domain from the NASA Twin Otter aircraft and the extended Kalman filtering, respectively with Gaussian random noise added to all simulation output.

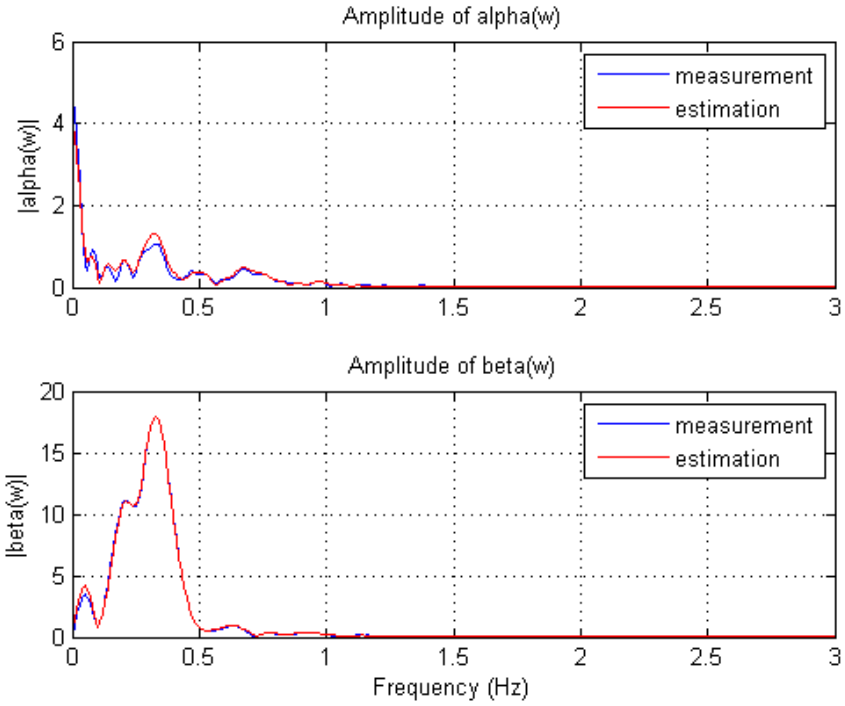


Figure 7.6: Magnitude of Reconstructed Air Flow Angles in Frequency Domain Between NASA Twin Otter Aircraft Measuement Data and The Proposed Estimation

Figure 7.6 demonstrates the comparison of magnitudes of measured and estimated air flow angles of NASA Twin Otter aircraft each frequency using flight data with Gaussian random noise added. The two graphs are also very similar and there is almost no difference which indicates that good aircraft parameter results could be obtained using frequency domain system identification.

Figure 7.7 shows the result of estimated gravitational acceleration while NASA Twin Otter aircraft is operating.The estimated gravitational acceleration remain near a known constant value (9.80665 m/s^2) of the environment.

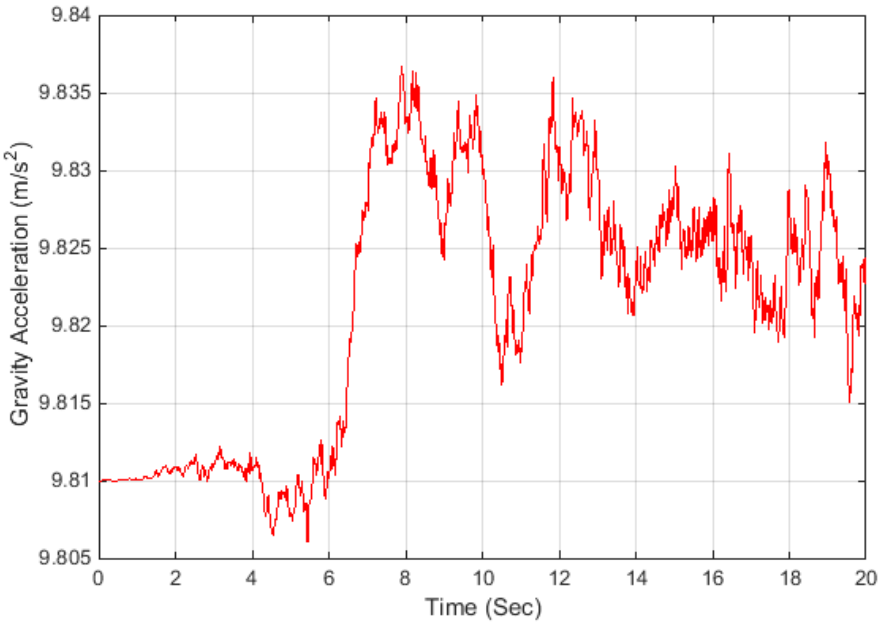


Figure 7.7: Simulation Result of Estimated Gravitational Acceleration Data from NASA Twin Otter Aircraft Flight Data via Extended Kalman Filtering

7.5 Chapter Summary

New techniques have been presented to obtain estimates of light aircraft aerodynamic parameters without airflow angle sensors. The results demonstrated good quality estimates of the angles of attack and sideslip using extended Kalman filtering techniques. The new methods are based on general kinematic equations. Furthermore, the proposed estimation can predict a value of gravitational acceleration which leads to an increment in the accuracy of calculation in terms of implementation as the gravitational acceleration is not constant in the atmosphere and difficult to determine in real-time. Moreover, the estimated gravitational acceleration will be used to update parameter in the outer loop of flight control system.

Chapter 8

Control Action of UAV Agents

As mentioned in Chapter 4, dynamic inversion control (DIC) is chosen to be the control laws for inner and outer flight control system of autopilot architecture, since this method based on a physical model where each parameter can give a physical meaning of system. However, the performance of DIC technique is sensitive to an accurate mathematical non-linear aircraft model. Consequently, direct, indirect, or combine adaptive control methodology, which is based on a model inversion flight control architecture, are proposed as agent plans to compensate the deficiency of DIC approach.

8.1 Indirect Adaptive Control Law Proposal

The indirect adaptive control law, based on the physical flight dynamic model, relies on real-time aircraft parameter estimation algorithm in order to calculate the compensated or new parameters for adapting the inner loop NDI controller in real time. A method, sometimes called orthogonal-least-square in frequency-domain [127], is employed here due to the need for real-time computation and for the ability to easily eliminate noise effects and unnecessary state reduction such as angular acceleration for aircraft parameter estimation. Furthermore, this method can also easily select the significant aerodynamic parameters of UAV.

$$\mathcal{F}[f(t)] \equiv \tilde{f}(\omega) \equiv \int_0^T f(t)e^{-j\omega t}dt \tag{8.1}$$

The equation can be approximated as a discrete term by

$$\tilde{f}(\omega) \approx \Delta t \sum_{i=0}^{N-1} f(i) e^{-j\omega i \Delta t} \tag{8.2}$$

Therefore, the discrete Fourier transform can be arranged as

$$\mathcal{A}(\omega) \equiv \sum_{i=0}^{N-1} f(i) e^{-j\omega i \Delta t} \tag{8.3}$$

From Eq. (8.2) and Eq. (8.3), the finite Fourier transform can be approximated as

$$\tilde{f}(\omega) \approx \mathcal{A}(\omega) \Delta t \tag{8.4}$$

Time domain linear regression can be rearranged and transformed into a formulation of a standard complex linear regression problem in the frequency domain in order to apply the least squares method to estimate unknown parameters. The general form of the complex linear regression is

$$\tilde{Z} = \tilde{X}\Theta + \tilde{e} \tag{8.5}$$

For example, using the pitching moment equation in Eq. (4.2),

$$\tilde{Z} = \begin{bmatrix} \tilde{C}_m(\omega_1) \\ \tilde{C}_m(\omega_2) \\ \vdots \\ \tilde{C}_m(\omega_M) \end{bmatrix} \tag{8.6}$$

$$\tilde{X} = \begin{bmatrix} \mathcal{F}[1](\omega_1) & \tilde{\alpha}(\omega_1) & \tilde{q}(\omega_1) & \tilde{\delta}_e(\omega_1) \\ \mathcal{F}[1](\omega_2) & \tilde{\alpha}(\omega_2) & \tilde{q}(\omega_2) & \tilde{\delta}_e(\omega_2) \\ \vdots & \vdots & \vdots & \vdots \\ \mathcal{F}[1](\omega_M) & \tilde{\alpha}(\omega_M) & \tilde{q}(\omega_M) & \tilde{\delta}_e(\omega_M) \end{bmatrix} \tag{8.7}$$

$$\Theta = \begin{bmatrix} C_{m_0} \\ C_{m_\alpha} \\ C_{m_q} \\ C_{m_{\delta_e}} \end{bmatrix} \tag{8.8}$$

where

$\tilde{Z} = Mx1$ vector of transformed dependent variable

$\Theta = N_p x1$ vector of unknown parameters

$\tilde{X} = MxN_p$ vector of transformed regressors

$\tilde{e} = Mx1$ vector of complex measurement errors

$M =$ the number of selected frequencies in the frequency band with fixed frequency spacing

$N_p =$ the number of unknown parameter elements

The cost function of the least squares computation is

$$J(\Theta) = \frac{1}{2}(\tilde{Z} - \tilde{X}\Theta)^\dagger(\tilde{Z} - \tilde{X}\Theta) \tag{8.9}$$

and the estimate is

$$\tilde{\Theta} = [Re(\tilde{X}^\dagger \tilde{X})]^{-1} Re(\tilde{X}^\dagger \tilde{Z}) \tag{8.10}$$

The covariance matrix of estimated parameter vector is

$$Cov(\tilde{\Theta}) \equiv E[(\tilde{\Theta} - \Theta)(\tilde{\Theta} - \Theta)^T] = \sigma^2 Re(\tilde{X}^\dagger \tilde{X})]^{-1} \tag{8.11}$$

where variance σ^2 is approximated as

$$\sigma^2 = \frac{1}{(M - N_p)}[(\tilde{Z} - \tilde{X}\Theta)^\dagger(\tilde{Z} - \tilde{X}\Theta)] \tag{8.12}$$

Recursive calculation by Fourier transform can be achieved by rearranging the discrete Fourier transform in Eq. (8.2) and exploiting the relation between times $i\Delta t$ and $(i-1)\Delta t$:

$$\mathcal{A}_i(\omega) = \mathcal{A}_{i-1}(\omega) + \alpha(i)e^{-j\omega i\Delta t} \tag{8.13}$$

where

$$e^{-j\omega i\Delta t} = e^{-j\omega\Delta t}e^{-j\omega(i-1)\Delta t} \tag{8.14}$$

The quantity of $e^{-j\omega\Delta t}$ is constant because a given frequency ω and sampling time Δt is fixed. Here we assign a frequency spacing of 0.04 Hz on the interval [0.0-2.0] Hz so $\omega_1, \omega_2, ..., \omega_M = 2\pi[0.0, 0.04, ..., 2.0]$.

Aircraft parameter estimation with this aerodynamic modelling depends on various states, which can be directly and indirectly measured from sensors including airspeed (V : pressure sensor), air flow angles (note α and β estimates used in this study), translation acceleration (a_x, a_y and a_z : accelerometers), angular rate (p, q and r : rate gyro) and (indirectly measured) angular acceleration ($\dot{p}$, $\dot{q}$ and $\dot{r}$). The angular acceleration states are directly unavailable from sensors. However, with properties of Fourier transform, the angular accelerations in Eq. (8.15) are unnecessary because the derivative term in the frequency domain can be transformed by multiplying $j\omega$ to p, q and r. For example, from Eq. (4.2), the pitching moment coefficient can be calculated as

$$\tilde{C_m}(\omega) \equiv j\omega\mathcal{F}[\frac{I_y q}{\bar{q}S\bar{c}}] + \mathcal{F}[\frac{I_{xz}(p^2 - r^2) + (I_x - I_z)pr]}{\bar{q}S\bar{c}}] \tag{8.15}$$

Most importantly, however, this method removes the unwanted high-frequency components of signals by considering only the interested frequency range in the computations according to the properties of infinite Fourier transform. Generally, a "good" frequency band of $0.01 - 3.0$ Hz is used in our calculations for aircraft dynamics.

The OLS algorithm and the ERR (Error Reduction Ratio) approach [138] have been extensively studied and widely applied. From Eq. (8.5), assuming that the regression matrix $\tilde{X}$ is full rank in columns and could be orthogonally transformed as

$$\tilde{X} = WT \tag{8.16}$$

where N_o is the data length, M_o is the number of estimated parameters, T is an M_o x M_o unit upper triangular matrix and W is an N_o x M_o orthogonal matrix with columns $w_1, w_2, ..., w_M$, thus W^TW is a diagonal matrix (D) which equals to $diag[d_1, d_2, ..., d_M]$ can show as $d_i = < w_i, w_i > = \sum_{t=1}^{N} w_i(t)w_i(t)$, where $t =$

$1, ..., N_o$ and $<,>$ denotes the inner product of two vectors. Therefore, in OLS, Eq. (8.5) can be expressed as

$$\tilde{Z} = (\tilde{X}T^{-1})(T\Theta) + \tilde{e} = WG + \tilde{e} \tag{8.17}$$

where $G = [g_1, g_2, ..., g_M]$ is an auxiliary parameter vector calculated directly from $\tilde{Z}$ and W as

$$g_i = \frac{< \tilde{Z}, w_i >}{< w_i, w_i >} \tag{8.18}$$

To calculate W and G matrix, they can be solved by using a classical and modified Gram-Schmidt algorithm. Furthermore, ERR indicates the importance of each regressor term and can be utilised with forward selection criteria. The ERR is defined as

$$ERR_i = \frac{g_i^2 < w_i, w_i >}{< \tilde{Z}, \tilde{Z} >} \times 100, i = 1, 2, ..., M \tag{8.19}$$

However, computing ERR_i in the frequency domain will result in complex numbers, therefore, this can be modified as the ratio of magnitude energy of each term $g_i^2 < w_i, w_i >$ to the magnitude energy of output $< Y, Y >$ illustrated as

$$ERR_i = \frac{abs(g_i^2 < w_i, w_i >)}{abs(< \tilde{Z}, \tilde{Z} >)} \times 100, i = 1, 2, ..., M \tag{8.20}$$

This ratio is an effective indication for searching for the order of significant terms. The selection criteria can use Eq. (8.21) to stop the procedure.

$$1 - \sum_{i=1}^{n_r} ERR_i < \rho_i \tag{8.21}$$

where n_r is the number of selected regressor and ρ_i is the desired tolerance.

8.2 Model Reference Direct Adaptive Control Laws

8.2.1 Inner Loop of Flight Control System

Assume that a non-linear equation of angular motion of an aircraft can be described by

$$\dot{x} = f(x, u, z) \tag{8.22}$$

where, in this section, $x = \begin{bmatrix} p & q & r \end{bmatrix}^T$ is the angular rate vector, $u = \begin{bmatrix} \delta_a & \delta_e & \delta_r \end{bmatrix}^T$ is the control surface deflection vector, and $z = \begin{bmatrix} \alpha & \beta & \delta_t \end{bmatrix}^T$ is the state vector. (δ_t is engine throttle level)

Assuming $\dot{x_d}$ is a desired rate;

$$\dot{x_d} = \dot{x_m} + K_P(x_m - x) + K_I \int_0^t (x_m - x)d\tau - u_{ad} \tag{8.23}$$

where $u_{ad} = W^T\Phi$ with $\Phi = \begin{bmatrix} x^T & u^T & z^T \end{bmatrix}^T$

Assuming that there is uncertainty (ε) in system, then;

$$\dot{x_d} = \dot{x} + \varepsilon \tag{8.24}$$

Inserting Eq. (8.24) into Eq. (8.23), and rearranging to compute the acceleration error yields:

$$\dot{x_e} = K_P x_e + K_I \int_0^t x_e d\tau + u_{ad} - \varepsilon \tag{8.25}$$

where $x_e = x_m - x$, $K_p = diag(K_{P,p}, K_{P,q}, K_{P,r}) > 0$, and $K_I = diag(K_{I,p}, K_{I,q}, K_{I,r}) > 0$ are matrices of the proportional and integral gain for roll, pitch, and yaw.

Rearranging the Eq. (8.25) into the tracking error equation matrix form:

$$\dot{e} = A_c e + b(u_{ad} - \varepsilon) \tag{8.26}$$

where $e = \begin{bmatrix} \int_0^t x_e d\tau \\ x_e \end{bmatrix}$, $A_c = \begin{bmatrix} 0 & I \\ -K_I & -K_P \end{bmatrix}$, $b = \begin{bmatrix} 0 \\ I \end{bmatrix}$

The weight W are updated by an adaptation law with an optimal control modification according to:

$$\dot{W} = -\mathcal{T}(\Phi e^T P B - \mathcal{L}\Phi\Phi^T W B^T P {A_c}^{-1} B) \, , \; \mathcal{L} > 0 \tag{8.27}$$

where the matrix P solves the Lyapunov equation $A^T P + P^T A = Q$ by defining $Q = 2I$, then the solution of the equation is:

$$P = \begin{bmatrix} {K_I}^{-1} K_P + {K_P}^{-1}(K_I + I) & {K_I}^{-1} \\ {K_I}^{-1} & {K_P}^{-1}(I + {K_I}^{-1}) \end{bmatrix} > 0 \tag{8.28}$$

${A_c}^{-1}$ is calculated as

$${A_c}^{-1} = \begin{bmatrix} -{K_I}^{-1} K_P & -{K_I}^{-1} \\ I & 0 \end{bmatrix} \tag{8.29}$$

Determining the term $b^T P {A_c}^{-1} b = -{K_I}^{-2} < 0$ can be done by applying the term $b^T P {A_c}^{-1} b$ to the adaptive law in Eq. (8.30), then the weight update law is given by

$$\dot{W} = -\mathcal{T}(\Phi e^T P B - \mathcal{L}\Phi^T W {K_I}^{-2}) \tag{8.30}$$

Furthermore, modifying Derivative-Free Adaptive Law [143] with optimal control modification, the new update weight law can be given by

$$W(t) = \Omega W(t - \tau) - \mathcal{T}(\Phi e^T P B - \mathcal{L}\Phi^T W {K_I}^{-2}) \tag{8.31}$$

8.2.2 Outer Loop of Flight Control System

Similar to the direct adaptive mechanism in Section 8.2.1, assuming that a non-linear equation Euler angle kinematic equation can be described by

$$\dot{x} = f(x, u, z) \tag{8.32}$$

where, in this section, $x = \begin{bmatrix} \phi & \theta & \beta \end{bmatrix}^T$ is the Euler angle and sideslip angle vector, $u = \begin{bmatrix} p & q & r \end{bmatrix}^T$ is the angular rate vector, and $z = \begin{bmatrix} a_x & a_y & a_z \end{bmatrix}^T$ is the linear acceleration vector. Then the adaptive mechanism procedure is the same process as in the Section 8.2.1.

8.3 Simulation Results

8.3.1 Evaluation of Aircraft Parameter Estimation with OLS in time and frequency domain

8.3.1.1 Test 1: Evaluation with Linear regression equation

The performance of OLS was investigated. The pitching moment coefficient model in linear regression form was simulated for structure selection and parameter estimation using OLS algorithm in time and frequency domain. The investigated model was:

Model 1 :

$$C_m(t) = C_{m_0} + C_{m_\alpha}\alpha(t) + C_{m_q}(\frac{q\bar{c}}{2V})(t) + C_{m_{\delta_e}}\delta_e(t) + e(t) \tag{8.33}$$

where

$$C_{m_0} = 0.1, C_{m_\alpha} = -2.75, C_{m_q} = -20.0, C_{m_{\delta_e}} = -0.75 \tag{8.34}$$

and $e(t)$ is a random white noise. A sequence of 750 output data points was collected from the state of Aerosonade Simulation while disturbing with 3-2-1-1 input similar to the data of air flow angle estimation in the previous section. Then all the mentioned output data points were used to determine the pitch

Table 8.1: Estimated parameters for Model 1 with fitting structure

Terms	OLS in time estimated value	ERR_i	OLS in frequency estimated value	ERR_i
C_{m_0}	0.1	0.1464	0.1	1.7923
C_{m_α}	-2.75	16.5430	-2.75	35.9476
C_{m_q}	-20.0	2.2563	-20.0	8.0643
$C_{m_{\delta_e}}$	-0.75	81.0544	-0.75	54.1958
	sum of ERR_i	100.00	sum of ERR_i	100.00

Table 8.2: Estimated parameters for Model 1 with under-fitting structure by cutting q state

Terms	OLS in time estimated value	ERR_i	OLS in frequency estimated value	ERR_i
C_{m_0}	0.0860	0.1464	0.0696	1.7923
C_{m_α}	-2.2572	16.5430	-1.8553	35.9476
$C_{m_{\delta_e}}$	-0.5401	65.6706	-0.4471	42.6725
	sum of ERR_i	82.3599	sum of ERR_i	80.4124

moment coefficient in Model 1. The fitting, under-fitting and over-fitting model structure was defined to depend on coefficient parameters according to Table 8.1-8.3, respectively. The OLS in time and frequency domain were applied to compute the stability and control pitching moment coefficients according to the model structure in Table 8.1-8.3 to monitor the ERR indication performance. The result is summarised in Table 8.1-8.3.

Table 8.1 illustrates that the estimated values of pitching moment stability and control coefficients which calculated from OLS in time and frequency domain, are equal in case of fitting structure. The ERR indications computing in the frequency domain can utilize to prioritize variables similar to the order of significant term computing in the time domain.

The algorithm is repeated for the Model 1 with under-fitting structure. The result is summarised in Table 8.2. Both the sum of ERR value in time and frequency domain are less that 85 percent. It means that the model structure still lacks some significant variables.

Table 8.3 shows that the estimated parameters using OLS in the frequency

Table 8.3: Estimated parameters for Model 1 with over-fitting structure by adding α^2 state

Terms	OLS in time estimated value	ERR_i	OLS in frequency estimated value	ERR_i
C_{m_0}	0.1	0.1464	0.1	1.7923
C_{m_α}	-2.75	16.5430	-2.75	35.9476
$C_{m_{\alpha^2}}$	-4.294e-17	0.0081	-7.442e-16	0.2157
C_{m_q}	-20.0	2.2939	-20.0	8.0804
$C_{m_{\delta_e}}$	-0.75	81.0087	-0.75	53.9640
	sum of ERR_i	100.00	sum of ERR_i	100.00

domain are close to the values using OLS in the time domain. Moreover, the value of ERR using either OLS approach in time and frequency domain for the term $C_{m_0}, C_{m_\alpha}, C_{m_q}$ and $C_{m_{\delta_e}}$ are significantly higher that the term of $C_{m_{\dot{\alpha}}}$. Thus, it indicates that frequency OLS approach can select the significant terms of the model with a slight error.

8.3.1.2 Test 2: Evaluation with aircraft system identification problem

OLS in time and frequency domain were proposed to solve the aircraft system identification problem according to the equation below as shown in:

Model 2 :

$$C_m(t) = C_{m_0} + C_{m_\alpha}\alpha(t) + C_{m_{\dot{\alpha}}}(\frac{\dot{\alpha}\bar{c}}{2V})(t) + C_{m_q}(\frac{q\bar{c}}{2V})(t) + C_{m_{\delta_e}}\delta_e(t) + e(t) \quad (8.35)$$

where

$$C_m(t) = [\dot{q(t)}I_y + p(t)r(t)(I_x - I_y) + (p(t)^2 - r(t)^2)I_{xz}]/\bar{q(t)}S\bar{c} \quad (8.36)$$

A sequence of 750 pitching moment coefficients output data was generated from state input of $\dot{q}(t)$, $\dot{p}(t)$, $\dot{r}(t)$ for time domain and $p(t)$, $q(t)$, $r(t)$ for frequency domain as mention in Eq. (8.5). Next, C_{m_0}, C_{m_α}, $C_{m_{\dot{\alpha}}}$, C_{m_q} and $C_{m_{\delta_e}}$ were computed using the least square algorithm and OLS in time and frequency domain as illustrated in Table 8.4-8.5.

Table 8.4: Estimated parameters for Model 2 with parameter estimation in time domain

Terms	OLS in time Estimated value	ERR_i	Least Square Estimated value
C_{m_0}	0.1159	1.766e-04	0.1159
C_{m_α}	-3.0769	17.7901	-3.0769
$C_{m_{\dot{\alpha}}}$	-7.8875	4.5079	-7.8875
C_{m_q}	-19.7372	0.1073	-19.7372
$C_{m_{\delta_e}}$	-0.8072	59.7849	-0.8072
	sum of ERR_i	82.1904	

Table 8.5: Estimated parameters for Model 2 with parameter estimation in frequency domain

Terms	OLS in frequency Estimated value	ERR_i	Least Square Estimated value
C_{m_0}	0.1061	0.3122	0.1234
C_{m_α}	-2.91	28.5571	-3.3070
$C_{m_{\dot{\alpha}}}$	-9.1849	28.4477	-16.2841
C_{m_q}	-19.7759	6.5876	-23.4229
$C_{m_{\delta_e}}$	-0.7984	27.8456	-0.8810
	sum of ERR_i	91.7503	

Table 8.4 and 8.5 show that the estimated parameters that were calculated from OLS in the frequency domain are close to the calculated values from OLS in the time domain and with the least square algorithm. Furthermore, the ERR indications using frequency OLS can be employed to rank the significant order of variables similar to the ERR indications using OLS in the time domain.

8.3.2 Performance of Direct Adaptive Flight Control for Inner Loop

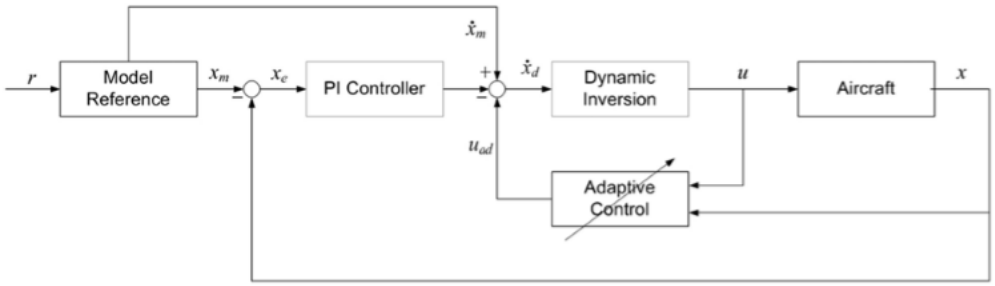


Figure 8.1: Direct Adaptive Control Architecture [97].

Firstly, the direct adaptive flight control based on NDI control architecture with a free derivative law, an optimal control modification, and a combined adaptive law was performed to evaluate with Aerosonde UAV simulation in the MATLAB/SIMULINK environment. The inner loop adaptive flight control architecture that used this study is shown in Fig. 8.1. This control architecture consists of: (1) a reference model for desired rate command, (2) a proportional-integral feedback control, (3) a non-linear dynamic inversion controller for computing actuator command, and (4) a direct neural networks adaptive augmentation which is a single-layer sigma-pi neural network with adaptive law such as the free derivative law, the optimal control modification, and the combination of both adaptive laws. The inner loop rate feedback control is applied to improve the aircraft angular rate response.

The reference pitch rate command is stimulated to the simulation with a series of step input longitudinal stick command doublets by pilot. The tracking performance of the inner loop adaptive flight controller with three adaptive control laws is demonstrated in Fig. 8.2 in normal flight condition. An accurate aircraft aerodynamic parameter model, in this case, obtained from Table. 8.5. The simulation results show that the inner loop adaptive controller with three adaptive control laws offers good tracking performance in case of normal flight and accurate model due to no difference in the aircraft pitch rate response. Thus, it can be seen that the aircraft aerodynamic parameters in Table. 8.5, which were

calculated with OLS in the frequency domain, are accepted for dynamic inversion controller as it offers good tracking performance. To be clear, the inner loop adaptive controller with three adaptive control laws has slightly better tracking performance that the baseline controller (NDI).

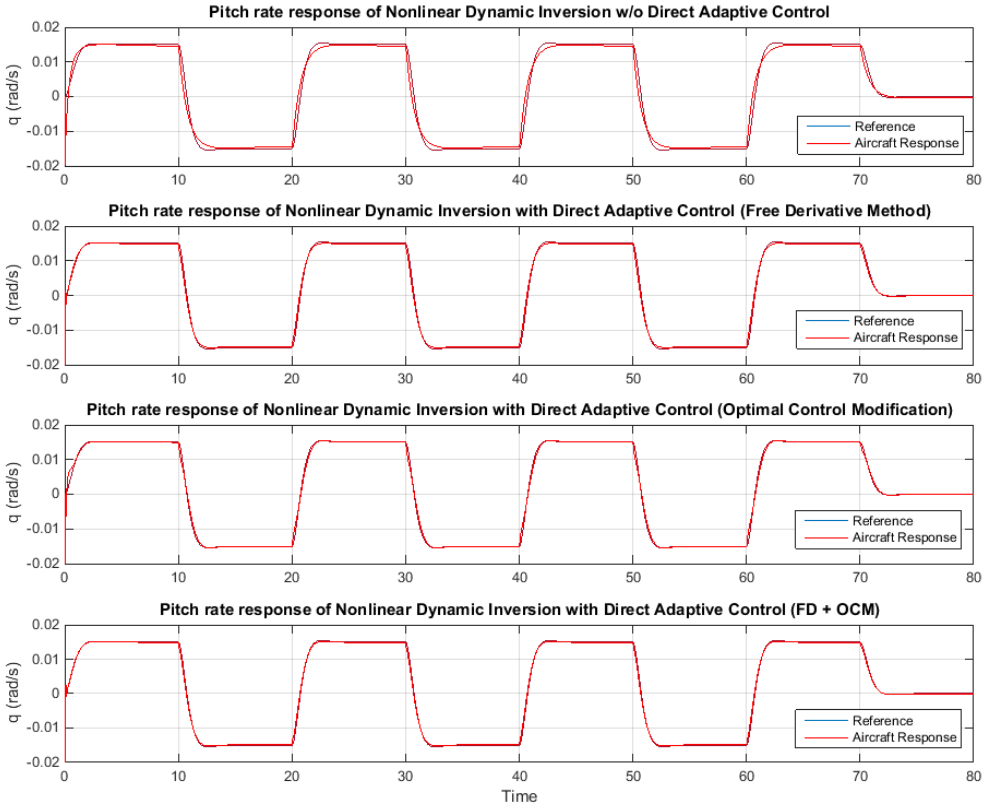


Figure 8.2: Control Performance Comparison of Nonlinear Dynamic Inversion Control with Three Direct Adaptive Control Mechanisms using estimated parameter from Table 8.5

In addition, a falsity of elevator parameter model configuration corresponding to a mistake of 50% of a pitching moment coefficient depending on elevator is selected. The architecture of the inner loop adaptive flight control and input pattern command of pitch rate are similar to the previous case. The Fig. 8.3 illustrates the tracking control performance of the the inner loop adaptive flight controller with three direct adaptive control laws in case of the mentioned falsity.

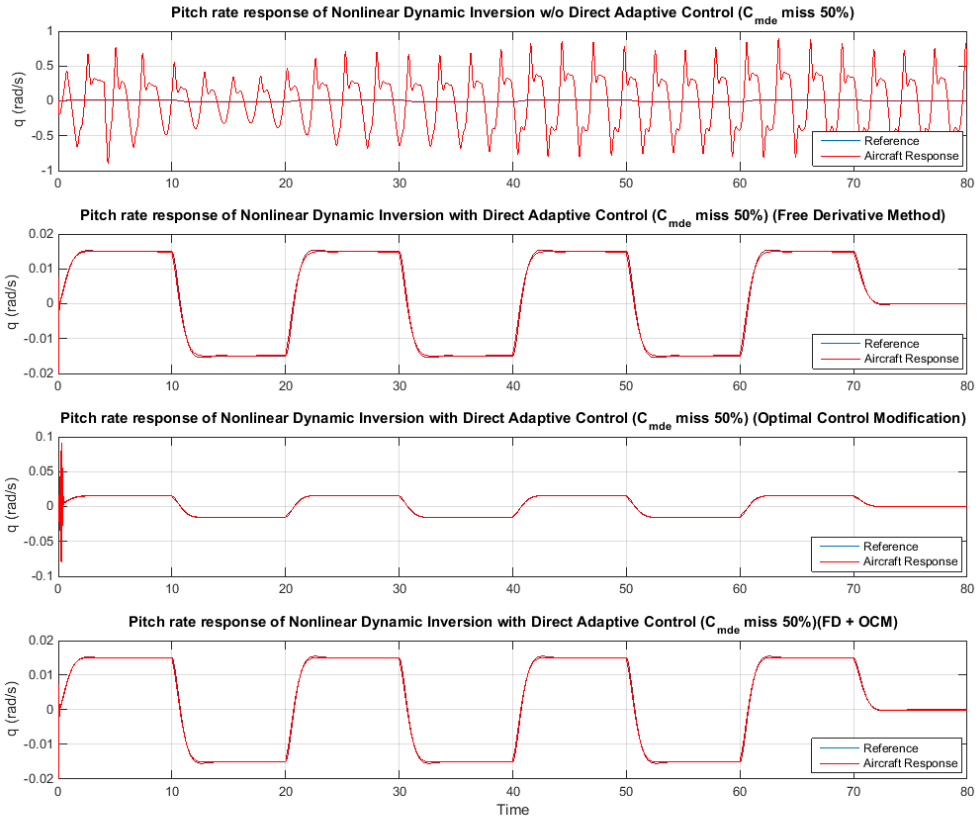


Figure 8.3: Control Performance Comparison of Non-linear Dynamic Inversion Control with Direct Adaptive Control Mechanisms using estimated parameter from Table 8.5 in case of defining $C_{m_{\delta_e}}$ of NDI control missing 50%

It can be seen that the direct adaptive control based on NDI control architecture with three adaptive control laws invests some degree of improvement in tracking performance as compared with the baseline NDI control scheme in case of falsity in elevator parameter model configuration. In contrast, the direct adaptive control method with free derivative and combined laws seem to perform better than both the NDI and direct adaptive control approaches with optimal control modification law. Theoretically, the baseline NDI controller is unable to track the reference pitch rate commend because the performance of NDI controller relies on an accuracy of the model as change of aircraft dynamic which is unmatched

with model of NDI. Therefore, the direct adaptive controllers have the ability to augment the control signal as the compensation for the inverted dynamic model of baseline NDI controller.

For precisely, the result of the inner loop adaptive flight controller with the adaptive law combining mechanisms between optimal control modification and the free derivative law seem to provide trivially better tracking performance than one with free derivative law. From the result in Fig. 8.3, the optimal modification adaptive law offers the best tracking performance and robust adaptation of with large adaptive gain without high-frequency oscillation problem but it has high overshot in transient response. Alternatively, the controller with combined adaptive laws has no overshot in transient response due to a combination of an advantage of both optimal control modification and free derivative laws. It means that the inner loop adaptive flight controller with combination with the free derivative law and the optimal modification law offers good tracking control performance in steady state error and transient state. Therefore, the inner loop adaptive flight control with the adaptive law combining the free derivative and optimal modification law has been chosen as a direct adaptive control element of hybrid adaptive flight control.

8.3.3 Performance of Hybrid Adaptive Control for Inner Loop

Furthermore, to evaluate the neural network hybrid adaptive flight control with the OLS method in the frequency domain, the Aerosonde UAV simulation was performed in the MATLAB/SIMULINK environment. The hybrid adaptive flight control architecture used in this study is illustrated in Fig 2.2 and 4.2. This control framework comprise: (1) the reference model, (2) the proportional-integral feedback control, (3) the non-linear dynamic inversion controller, (4) the direct neural networks adaptive augmentation being single-layer sigmal-pi neural network with the adaptive control law that combines the free derivative law and the optimal modification law, and (5) parameter update mechanism (indirect adaptive control) using recursive OLS in frequency domain. An elevator damage configuration corresponding to a 50% loss of elevator control surface for Aerosonde

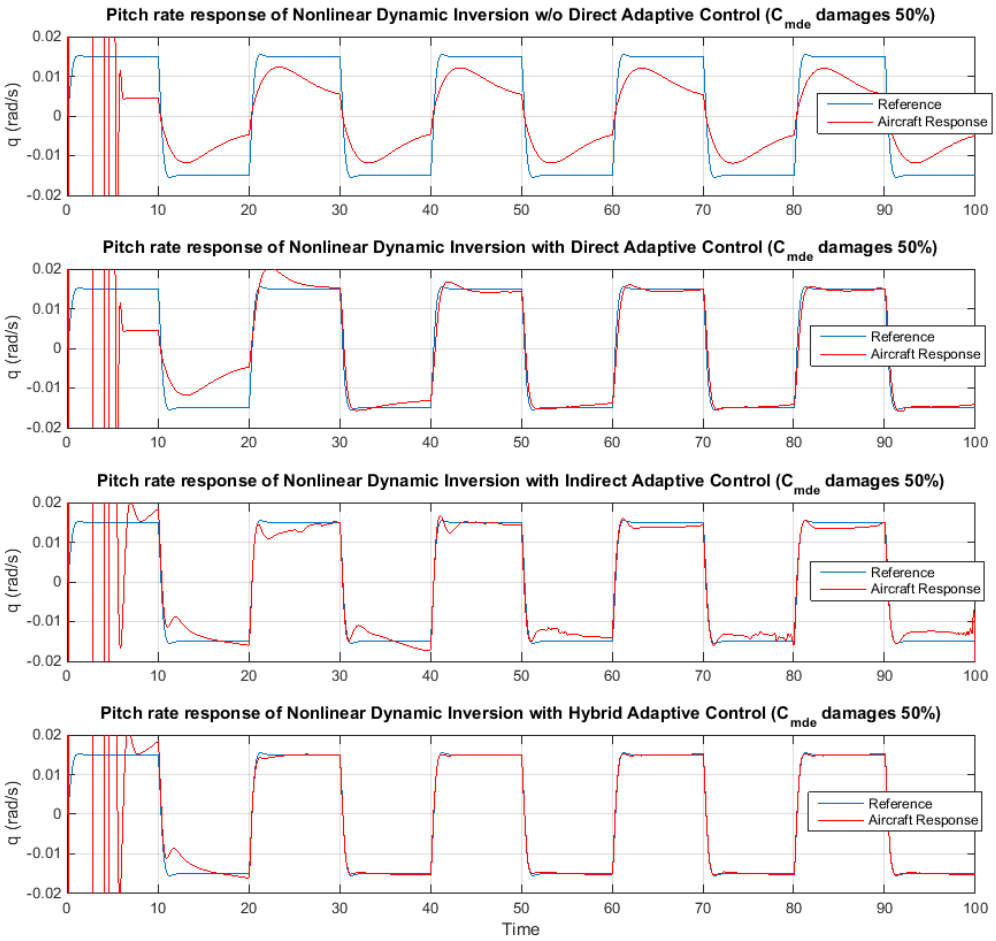


Figure 8.4: Control Performance Comparison of Non-linear Dynamic Inversion Control with Direct Adaptive Control Method and the OLS method in the frequency domain in case of a 50% loss of the elevator that effects a 50% loss of $C_{m_{\delta_e}}$ in simulation model (Note: Adaptive control approach is active at 10 sec.)

UAV model was chosen. Similar to the previous section, a series of step input longitudinal stick command doublets were applied to the simulation.

The tracking performance comparison of four control methodologies such as (1) a baseline NDI control, (2) a direct adaptive control based on NDI control architecture with the adaptive control law combining the free derivative and opti-

mal modification laws, (3) indirect adaptive control with recursive OLS approach in frequency domain, and (4) hybrid adaptive control schemes as mentioned, is shown in Figure 8.4.

As presented in Figure 8.4, the three adaptive control methods provide some degree of betterment in the tracking performance compare with the baseline NDI control approach with no adaptation. Obviously, simulation results present that the hybrid adaptive flight control can offer a significant improvement in the tracking performance in the pitch channel command over a direct and indirect adaptive control approaches alone as it provides the best tracking performance in longitudinal channel. Additionally, the tracking performance of direct adaptive control approach is improving with time as the tracking error reduces notably when time increases. With indirect adaptive control, the performance of the flight control is worse than two adaptive schemes as significant overshoot and steady state error occur due to insufficient input oscillation. Without adaptations, the performance of the baseline NDI flight control is very poor as large tracking error happen due to inaccurate model of inverted control.

Moreover, the hybrid adaptive flight control with the OLS method in the frequency domain was also evaluated in the MATLAB/SIMULINK simulation environment. In this case, insufficient initial parameter set-up configuration relating to aircraft aerodynamic parameter model for NDI control is considered. The initial aircraft aerodynamic parameters are provided by digital DATCAOM software [104] which based on aircraft geometry with numerical computation. In Figure 8.5, the pitch rate responses during the pitch doublet manoeuvre resulted from computation by four control schemes as aforementioned. The tracking performance of indirect control scheme is poor as large tracking error because of insufficient oscillation input. Due to inaccurate model, the worst tracking performance was obtained by the baseline NDI control.

The results are similar to the outcomes in the previous case. The hybrid adaptive flight controller using recursive OLS in the frequency domain can provide the best improvement in the tracking performance over a direct and indirect adaptive controller that work alone. With adaptive neural network control, the performance of the flight control is worse than one using hybrid adaptive control as significant steady state error but it is better than ones that use the baseline

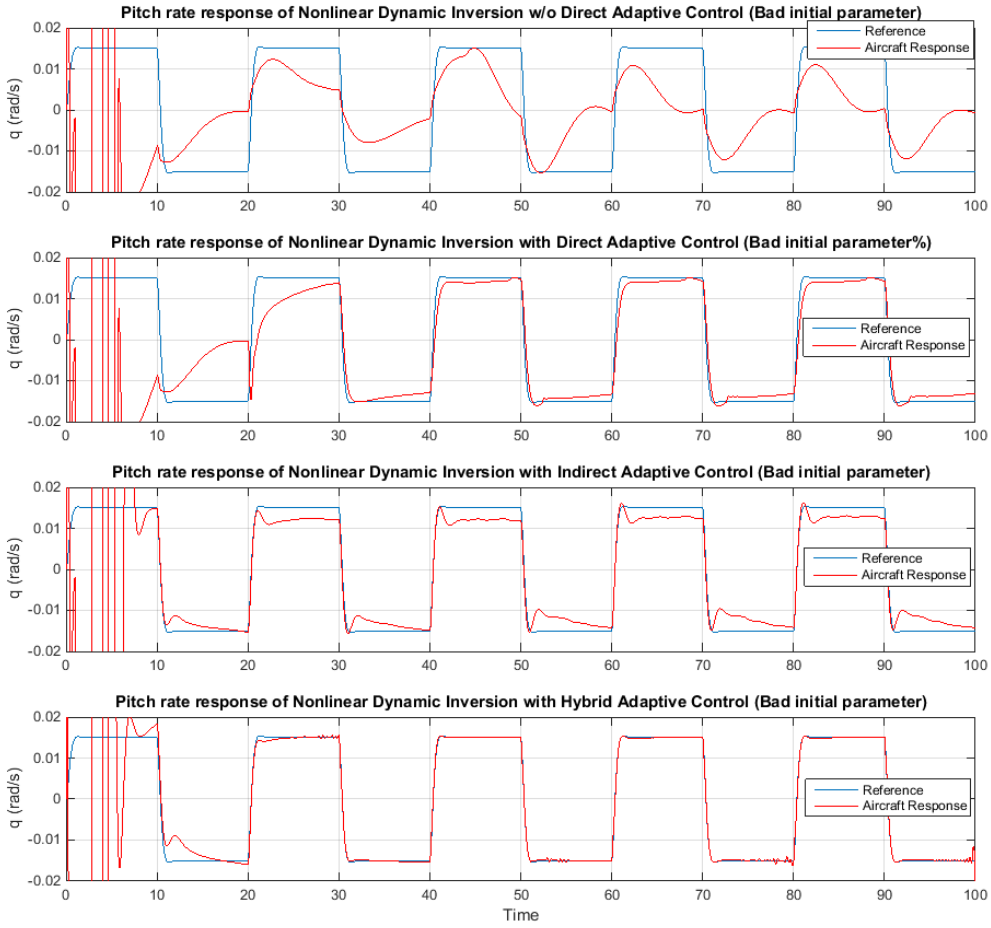


Figure 8.5: Control Performance Comparison of Non-linear Dynamic Inversion Control with direct adaptive control method and the OLS approach in the frequency domain in case of defining insufficient initial aerodynamic parameter for NDI control (Adaptive control approach is active at 10 sec.)

NDI control and indirect adaptive control as smaller steady state error.

Finally, a performance comparison of four control schemes, including two baseline NDI controls using estimated parameters from OLS in time and frequency domain (Table 8.4 and 8.5, respectively), indirect adaptive control with recursive OLS in frequency domain, and hybrid adaptive control, is shown in Fig. 8.6. These control schemes were assessed in the MATLAB/SIMULIN simulation envi-

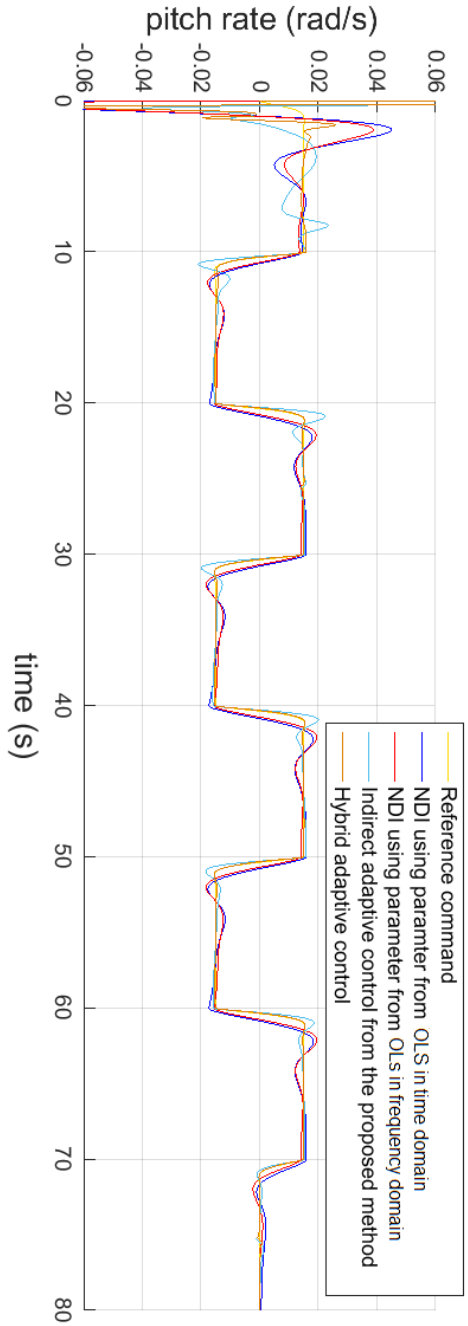


Figure 8.6: Control Performance Comparison of Non-linear Dynamic Inversion Control between parameter from time and frequency domain method and Hybrid Adaptive Control

ronment with Aerosonde UAV model. Similarly, a series of step input longitudinal stick command doublets was activated to the simulation.

The pitch rate responses implemented with such control approaches are shown in the Fig. 8.6. Firstly, this result indicates that control performance using parameters from OSL in frequency domain can attain the same quality of flight control performance that was obtained by calculating in the time domain. With indirect adaptation, the performance of flight control is better than two baseline NDI control scheme in term of faster transient response but wore than the hybrid adaptive control as larger tracking error. Certainly, the hybrid adaptive control can give the best tracking performance over three control approaches due to the advantage of using combination between direct and indirect adaptive control. Namely, the indirect adaptive control can compensate the aircraft parameters for NDI control to reduce the model error and any remaining tracking errors can be handled by direct adaptive control. Therefore, the hybrid adaptive control has been selected in this thesis to maintain the stability and performance of flight under adverse flight condition, especially deficiencies of the initial controller and system degradation due to accidental control surface damages.

For this simulation, actuator dynamics are not included. In this study, the small UAV only has primary control effectors including aileron, elevator, and rudder to stabilize and maintain the aircraft. If UAV has more redundancy control surface inputs, a control allocation strategy can employ other possible control effectors. But the control allocation is beyond the scope of this study. Additionally, in this adaptive flight control study, only failure of control input effectors are considered. However, it is not only occurred in cases of control surface failure but also in cases of serious damage situations, such as aircraft structure damages, might happen. Damage effects can present a serious challenge to the flight control system because a damaged aircraft would no longer function normally because its stability, control, and inertia parameter characteristics had changed significantly as mentioned in Section 3.2.3.

Therefore, in case of structural damaged aircraft, several affected variables from c.g. shifting, mass change, and aerodynamic terms should be considered in flight dynamic modelling. The inverted modelling of NDI control has to be changed according to Section 3.2.3 with on-line estimation in order to be able

to provide a significant improvement in the control performance in case of damaged aircraft. Furthermore, number and variable of inputs of the neural network with learning capability can also be increased following the variables that effect the damaged aircraft (in Section 3.2.3) to improve flight control performance. However, this topic is also beyond the scope of this study.

8.3.4 Performance of Hybrid Adaptive Control for Outer Loop

The outer loop flight control based on NDI control architecture with adaptive control law is explained in Section 7.3 and 8.2.2 was implemented to evaluate with Aerosonde UAV simulation in the MATLAB/SIMULINK environment. This outer loop hybrid adaptive flight control architecture used in this study is presented in Fig 4.3. This control framework contents: (1) the reference model, (2) the proportional-integral feedback control, (3) the non-linear dynamic inversion controller based in kinematic equations (as clarified in Section 4.2), (4) the direct neural networks adaptive augmentation being single-layer sigmal-pi neural network with the adaptive control law that combines the free derivative law and the optimal modification law (as explained in Section 8.2.2), and (5) parameter update mechanism (indirect adaptive control) to update the quantity of gravitational acceleration as described in Chapter 7. A reference pilot command was simulated by step input of angle for 5 degree at 5 sec in longitudinal dynamic and 10 degree at 50 sec for bang manoeuvre.

The tracking performance of the outer loop hybrid adaptive flight control was demonstrated in Fig. 8.7 in normal flight condition with accurate parameter model. The figure 8.7 illustrated the Euler roll, pitch angle, and sideslip angle response for outer loop of hybrid adaptive flight control system based on NDI architecture. As presented, the outer loop hybrid adaptive flight control provides a satisfied tracking performance as a slight error in both pitch and bank angle channel in normal flight condition. The UAV can do a coordinated turn flight at zero sideslip angle. There are some overshot in pitch and sideslip channel when aircraft coordinate turn.

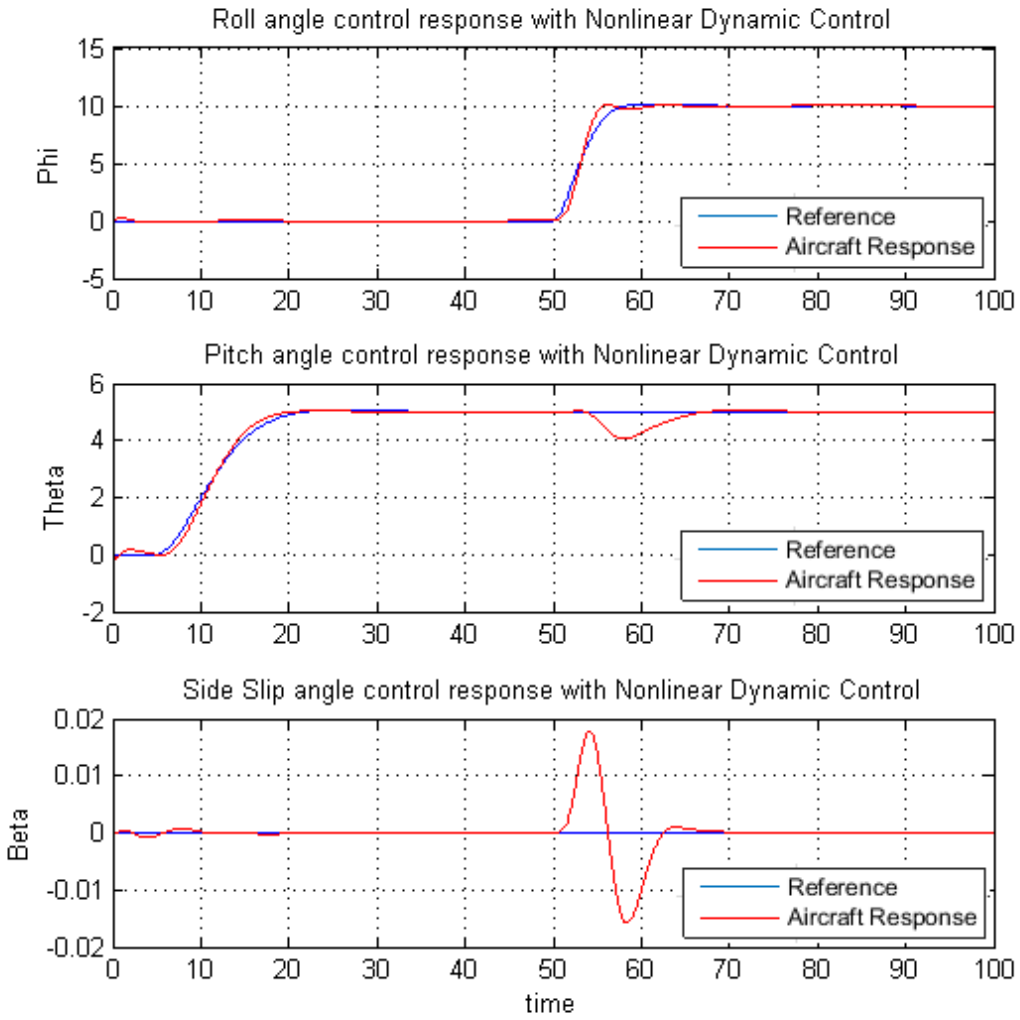


Figure 8.7: Tracking Performance of Outer Loop Flight Control System using initial parameter estimated from Table 8.5 in case of normal flight and accurate model of NDI

Next, an erroneousness of parameter model configuration relating to a mistake of a pitch moment coefficient that depends on elevator by 50% for NDI control is considered. The Euler roll, pitch angle, and sideslip angle response for outer loop hybrid adaptive flight control system based on NDI architecture were illustrated in Fig. 8.8. It can be seen that the control performance in this case can reach

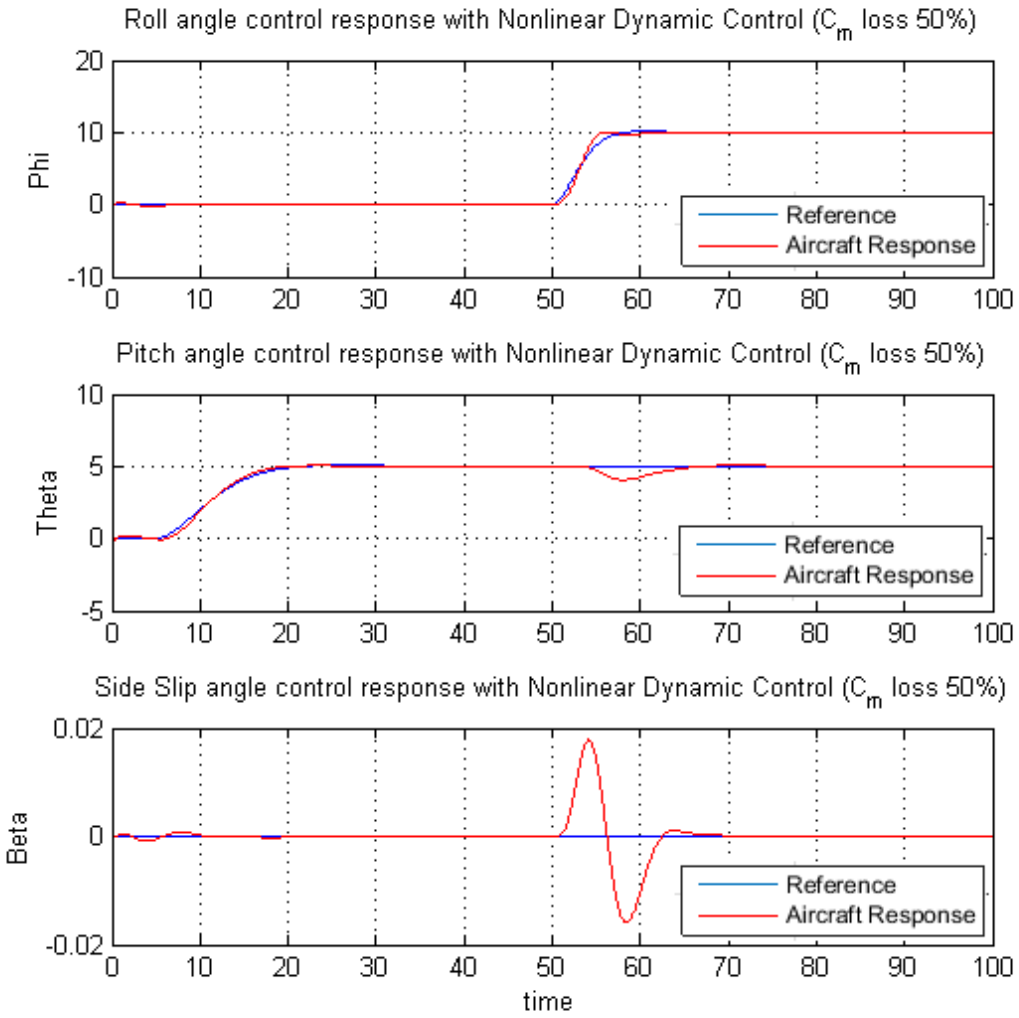


Figure 8.8: Tracking Performance of Outer Loop Flight Control System in case of defining a mistak of pitching moment coefficients depending on elevator by 50% from Table 8.5 for NDI control

the same quality as flight control performance in normal flight condition with no difference in response pattern. It means that the inner loop hybrid adaptive flight control can compensate the model error for inner loop NDI control and the falsity does not effect the outer loop flight control.

8.4 Chapter Summary

For aircraft system identification, OLS algorithm in the frequency domain can be applied to a linear-regression aircraft-model using estimated air flow angle states in order to select the model structure and compute aerodynamic coefficients of a fixed wing aircraft. The advantage of using discrete Fourier transforms is the ability to filter state signals over a frequency range of interest. Furthermore, the calculation in the frequency domain can aid to reduce the number of states in order to identify aircraft aerodynamic parameters via eliminating the derivative states through properties of the Fourier transform. For on-board implementation, the discrete Fourier transform can be rearranged in a recursive form in discrete time as a recursive Fourier transform. Finally, the performance comparison of the system identification methods shows that the proposed technique can obtain the same quality of flight performance as OLS in time domain and when airflow sensors are available. Therefore, for all the above mentioned reasons, this practical technique can be effectively used to estimate aircraft aerodynamic parameters in real-time during flight.

Furthermore, OLS in the frequency domain that works in combination with a direct adaptive control strategy has been applied as indirect adaptive learning for the neural network hybrid adaptive control scheme. The simulation results illustrate that the hybrid frequency OLS adaptive control strategy can contribute the significant improvement in the tracking performance over the direct and indirect adaptive control.

Chapter 9

Real-Time UAV Agent Perceptual Abstractions

In the development of intelligent control system operating functionally within a specified domain and implementing procedures based on declarative, procedural and heterogeneous knowledge, the concise representation and employment of relevant knowledge in abstraction perception are one important part. The perception abstraction evaluation is a method to receive a perception stream and subsequently filter it to generate the belief base that belongs to the rational agent as aforementioned in Chapter 6.

This study is about the development of system identification of non-linear aircraft dynamical models and robust control methods in combination with agent supervised autopilot on-board a UAS. Therefore, this agent-based control system requires real-time calculations to 1) evaluate a control performance, 2) investigate whether a set of aerodynamic coefficients is validated for NDI control, and 3) monitor flight trim condition before performing system re-identification in the physical engine for perceptual abstraction process. These boolean outcomes of all these computations are represented by predicates of beliefs in format of sEnglish as illustrated the relations between boolean outcomes and beliefs in Fig. 9.1.

9.1 On-line Model Validation for Reconfiguration Proposal

This method is used to validate whether or not the aircraft's aerodynamic parameters are required to be updated over a certain frequency range for the inner loop NDI control of the autopilot system. With this method, the time domain residual of each dimensionless aerodynamic force and moment coefficients $[C_L\ C_D\ C_Z\ C_l\ C_m\ C_n]^T$ are transformed into frequency domain by using discrete Fourier transformation and checking if each spectrum component of the frequency domain residual has the statistical properties of a white noise signal. The validation step is based on a hypothesis test applied to each frequency component of the normalised spectrum with χ^2-distribution.

The residual of magnitude of the estimated aerodynamic model in each frequency can be calculated as follows:

$$\Delta|C_l(\omega_i)| = |C_{l_1}(\omega_i)| - |C_{l_2}(\omega_i)| \tag{9.1}$$

in which $\Delta|C_l(\omega_i)|$ is the residual of magnitude of each pair parameter in each frequency, $|C_{l_1}(\omega_i)|$ is the example signal data that generates from Eq. (4.1) , and $|C_{l_2}(\omega_i)|$ is the example signal data that generates from Eq. (4.2). The faults, which change the system dynamics, also change the characteristics of $\Delta|C_l(\omega_i)|$ and make it increase.

Furthermore, the statistical test on the residual in Eqs. (9.1) can be applied to validate the parameters in order to confirm the parameter and model. This study proposed the frequency model validation approach [10] for monitoring the aircraft aerodynamic parameter of model inversion of inner flight control loop. The frequency model validation procedure is as follows:

1. Calculate the discrete Fourier transform of the residual $\xi_k \approx \Delta|.|$ as the difference of the real output in Eq. (4.1) and the model estimated output in Eq. (4.2). in the window size

2. Decompose each frequency component on its real part and imaginary part $\xi_k = R_k + jI_k$

3. Calculate distribution parameters of the Real (R_k) and Imaginary (I_k) part of each frequency of component k, that is:

   - Real part (R_k): Calculate μ_{R_0} and $\sigma^2_{R_k}$ for $k \in \{0, 1, 2, ..., N-1\}$

$$\mu_{R_k} = \mu_\xi \frac{1}{N} \sum_{n=0}^{N-1} cos(\Omega_0 kn) \tag{9.2}$$

$$\sigma^2_{R_k} = \sigma^2_\xi \frac{1}{N} \sum_{n=0}^{N-1} cos^2(\Omega_0 kn) \tag{9.3}$$

   - Imaginary part (I_k): Calculate μ_{I_0} and $\sigma^2_{I_k}$ for $k \in \{0, 1, 2, ..., N-1\}$

$$\mu_{I_k} = \mu_\xi \frac{1}{N} \sum_{n=0}^{N-1} sin(\Omega_0 kn) \tag{9.4}$$

$$\sigma^2_{I_k} = \sigma^2_\xi \frac{1}{N} \sum_{n=0}^{N-1} sin^2(\Omega_0 kn) \tag{9.5}$$

4. Calculate the normalized magnitude spectrum for each frequency $k \in \{0, 1, 2, ..., N-1\}$ as follows:

$$M^2_k = (\frac{R_k - \mu_{R_k}}{\sigma_{R_k}})^2 + (\frac{I_k - \mu_{I_k}}{\sigma_{I_k}})^2 \tag{9.6}$$

5. Perform a hypothesis test over each of the normalized magnitude spectrum M^2_k, indexed by $k \in \{0, 1, 2, ..., N-1\}$, as follows:

$$\begin{aligned} H_0 &: M^2_k \in \chi^2_2 \\ H_1 &: M^2_k \notin \chi^2_2 \end{aligned} \tag{9.7}$$

The probability of rejecting H_0 when it is true is set by choosing the confidence limit. For example, if the confidence limit is chosen to be 10.6 then the 99.5 percent of the sample of a χ^2_2 distribution fall within the limit. The Boolean outputs of Eq. (9.7) are represented by predicates of beliefs (sEnglish sentences) as shown in Fig. 9.1. Therefore, frequency dependent model validation [10] is applied to the inner loop non-linear dynamic inversion control of an autopilot system

in order to confirm the model. Therefore, this algorithm is a key computation to abstract the continuous information to discrete abstractions for validating aircraft aerodynamic parameters in the inner-loop of the NDI controller.

9.2 Flight Trim Condition Monitoring

A dedicated algorithm is used to monitor flight trim conditions from flight data to detect when and if the agent is ready to perform a re-identification in order to update the set of aerodynamic parameters for the inner loop NDI controller. The agent can't execute a system identification procedure immediately since the accuracy of the estimated aircraft parameters results hinges on flight conditions and adequate excitation of motion.

9.2.1 Using Wavelet Transform and Multi Resolution Analysis

The method called real-time wavelet flight data evaluator [91] was utilized in this process by monitoring input and output response in the form of Wavelet transform and multi-resolution analysis.

Wavelet transform has been one of the most important and powerful tools for signal representation. Wavelet transform, commonly, decomposes a time variant function $s(t)$ into time-frequency information $\mathcal{W}(a,b)$ as follows

$$\mathcal{W}(a,b) \equiv \int_{-\infty}^{\infty} s(t) \frac{1}{\sqrt{a}} \Psi^* \left(\frac{t-b}{a} \right) dt \tag{9.8}$$

where

t is time space.

a is scale parameter, which corresponds to the frequency band.

b is shift parameter, which corresponds to time space

Ψ^* is a mother wavelet function, which expresses localized oscillation. (Note: this study utilizes the Bump wavelets)

Table 9.1: Thresholds for Determination of the Trimmed States in [91]

Item	Symbol	Threshold(Δ_{trim})	Unit
Roll rate	p	5×10^2	deg^2/s^2
Pitch rate	q	5×10^2	deg^2/s^2
Heading rate	r	5×10^2	deg^2/s^2
Elevator Deflection Angle	δ_e	1×10^3	deg^2
Aileron Deflection Angle	δ_a	1×10^3	deg^2
Rudder Deflection Angle	δ_r	5×10^2	deg^2

Furthermore, Multi Resolution Analysis (MRA), which is a chart that shows a square of transformed values correlated to signal strength $\mathcal{W}^2(a, b)$ over frequency-time axis, is a powerful tool to analyse the spectrum and time solution of discrete data.

The analysis of the flight in the frequency domain (with signal strength obtained by MRA) determines whether the aircraft is operating in trim condition by checking the following conditions:

$$\sum_{a \in a^*_{z_x}} \mathcal{W}^2(a, b_1)[z_x] < \Delta_{trim,z_x} \tag{9.9}$$

$$\sum_{a \in a^*_{z_u}} \mathcal{W}^2(a, b_1)[z_u] < \Delta_{trim,z_u} \tag{9.10}$$

where

z_x is observed values correlated with state value x. (Note that: $z_x = p, q, r.$)

z_u is observed values correlated with inputs u. (Note that: $z_u = \delta_e, \delta_a, \delta_r.$)

Δ represents a threshold to determine whether data is almost settled. (The threshold values in each state and input illustrate in Table 9.1.)

a^* is certain frequency bands. (Note that: ranging from 10 Hz to approximate 0.05 Hz.)

9.2.2 Using Frequency Dependent Model Validation Approach

This method is similar to the one in Section 9.1. The residual is replaced by each angular rate compared with zero instead of each dimensionless aerodynamic force and moment coefficients, and then calculated by the frequency model validation procedure in Section 9.1.

Similar to previous section, the Boolean outcomes of this method are also represented by predicates of beliefs (sEnglish sentences) as shown in Fig. 9.1. Therefore, frequency dependent model validation [10] and wavelet transform with multi resolution analysis [91] have been applied to monitor a flight trim condition state. Therefore, this approach is an essential computation to abstract the continuous information to discrete abstractions for observing the flight trim condition state during a flight.

9.3 Control Performance Evaluation Approach

The main objective of our agent is to maintain control performance in the outer and inner loop of the flight control system. Firstly, this evaluation approach is required to assign a quantitative bound to the control system output error under flight operating conditions. This means that the abstractions of control performance is to observe whether or not the tracking error is within bounds in predicate form. According to [95], this abstraction can be used to trigger a resetting mechanism of a direct adaptive controller by a pre-defined threshold of the tracking error. When this threshold is exceeded, the adaptive gain is re-initialized with a large suitable value. However, the threshold should be chosen deliberately so that the trigger would execute appropriately to prevent false triggering. Therefore, the condition for monitoring control performance is [95]

$$\texttt{if}\ (\ |\mathcal{E}| < \Delta_{err}\ )\ \texttt{and}\ (\ |\int \mathcal{E}dt| < \Delta_{\int errdt}\ )$$

$$\texttt{then}\ \mathcal{B}(Control\ \ Performance\ is\ \ Good.)$$

$$\texttt{else if}\ (\ |\mathcal{E}| \geqslant \Delta_{err}\ )\ \texttt{or}\ (\ |\int \mathcal{E}dt| \geqslant \Delta_{\int errdt}\ )$$

$$\texttt{then}\ \mathcal{B}(Control\ \ Performance\ is\ \ Bad.)$$

where

$\mathcal{E}$ is observed values correlated with error between angular rate (p, q, r) response and reference command.

$\int \mathcal{E}dt$ is observed values correlated with integrated term of error between angular rate (p, q, r) response and reference command.

Δ_{err} represents a threshold depended on error to determine whether tracking performance is good.

$\Delta_{\int errdt}$ represents a threshold depended on integration term of error to determine whether tracking performance is good or not.

$\mathcal{B}(\ldots)$ represents an updated belief base of the agent.

However, the selection of the proper limits or thresholds is difficult due to various aircraft characteristics and flight operating conditions. Consequently, the frequency dependent model validation approach [10] can also be applied to this problem. Similarly, the computed procedure is similar in Section 9.1. However, the residual in this case is replaced by the error and the integral term of error of angular rate (p, q, r) instead of the residuals of each dimensionless aerodynamic force and moment coefficients $(C_L, C_D, C_Y, C_l, C_m, C_n)$. This method has been chosen in this thesis due to the advantage of reliability by using statistical hypothesis test on a residual in interesting frequency ranges. Furthermore, this method can remove the unwanted noise by calculating in the interesting frequecy ranges. Example implementation are illustrated in Fig 9.17-9.19.

Similarly, the Boolean outputs of the control performance evaluation method are also represented by predicates of beliefs (sEnglish sentences) as shown in Fig. 9.1. Therefore, frequency dependent model validation [10] has also been utilized to evaluate the performance of the control system during a flight.

9.4 Connection with Agent Framework

As aforementioned in Chapter 6, the perception abstraction process is a process to discrete a continuous information as predicates of beleifs in format of sEnglish sentences. The perception stream, which sampled from the physical engine, is

Mathematical Functions	sEnglish Sentence (Action Abstraction)	Boolean Outputs as Beliefs
On-line Model Validation using FDMV as shown in Section 9.1	[Executing a frequency model validation method to monitor aircraft aerodynamic parameter for inner NDI control.]	(~) Required to udpate aircraft aerodynamic parameters in longitudinal dynamic for inner NDI loop. (~) Required to udpate aircraft aerodynamic parameters in lateral and directional dynamic for inner NDI loop. (~) Required to compensate aircraft aerodynamic parameters in longitudinal dynamic for inner NDI loop. (~) Required to compensate aircraft aerodynamic parameters in lateral dynamic for inner NDI loop. (~) Required to compensate aircraft aerodynamic parameters in directional dynamic for inner NDI loop. (~) Required to trigger a direct adaptive control for inner loop.
Flight Trim Condition Monitoring using Wavelet Transform and MRA as Shown in Section 9.2	[Executing a monitoring algorithm to check aircraft condition (trim) status.]	(~) Aircraft manoeuvres in the trim condition.
Control Performance Evaluation using FDMV in Section 9.3	[Executing a control performance evaluation.]	(~) Bad Control System performs in the inner loop. (~) Bad Control System performs in the outer loop.

Figure 9.1: Illustration of perception processes which contribute to the belief base during each reasoning cycle of the BDI agent. These mathematical functions are utilized for numberical procedure in the physical engine [Appendix IV].

delivered to the abstraction engine. The abstraction engine, which might call on the continuous engine to make computation by using calculated funtions as explained in previous sections.

For implementation, sEnglish sentence can compile into embedded MATLAB code for computation routine. Lincoln [74] explains that "each sEnglish sentence is matched with a routine call in MATLAB as well as with a similar looking predicate for logic operations that abstracts away the code based meaning behind the predicates. Basic predicate abstractions from sentences are applied to both the world environment and the UAV model in perception abstracton process."These abstractions are passed by the physical engine to the abstraction engine. The information provided to the abstraction engine includes the following as demonstrated in Fig. 9.1.

9.5 Simulation Result

9.5.1 Results of Model Validation for Reconfigurable Control

The frequency dependent model validation method was evaluated with Aerosonde UAV simulation in the MATLAB/SIMULINK environment. Elevator surface deflection command input is stimulated by a pattern of doublet 3-2-1-1 from 0s. In the first case, the aerodynamic stability and control coefficients from frequency OLS estimation from Table 8.5 were defined to validate by calculating pitching moment coefficient C_m with Eq.(3.17) in red colour line illustrated in Fig. 9.2. And the blue line was computed C_m from Eq.(3.15). Both mentioned pitch moment coefficients were compared in time and frequency domain as shown in Figure 9.2. Visually, both signals are similar in all time history and spectrum. It means the aerodynamic parameters obtained from estimation are sufficient for aircraft model.

Then the residual of both pitch moment coefficients was calculated by following the procedure described in Section 9.1. Therefore, values of R_k and I_k in each frequency component calculated with Eq. (9.4) and (9.5) are shown in Fig. 9.3. And the normalized magnitude spectrum is calculated according to

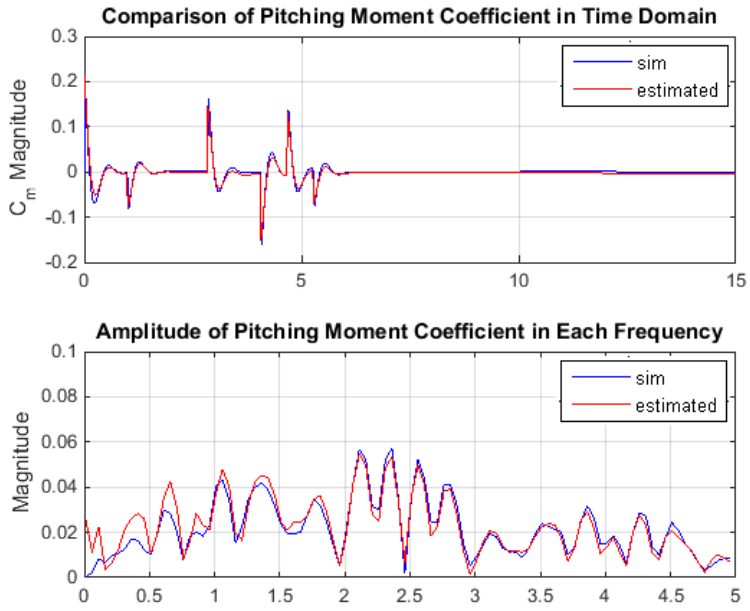


Figure 9.2: Comparison of Pitching Moment Coefficient (C_m) in time and frequency domain in case of accurate model from the estimation

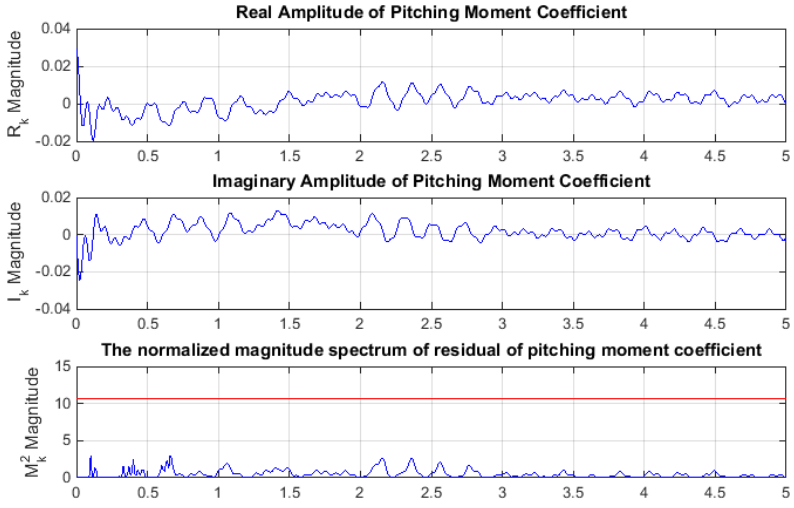


Figure 9.3: The real (R_k) and imaginary (I_k) parts of discrete Fourier transform of residual and normalized magnitude spectrum (M_k^2) in case of aerodynamic parameter from frequency OLS estimation in Table 8.5

Eq. (9.6) with mean (μ_ξ) of 0 and variance (σ_ξ^2) of 0.001. It can be seen that all the magnitude frequency components remain below the confidence limit (the 99.5% confidence limit of the χ^2-distribution is between 0 and 10.6). It means that the aerodynamic parameter is also validated for NDI based inner loop of flight control system. Therefore, this approach can provide the similar solution as visual inspection.

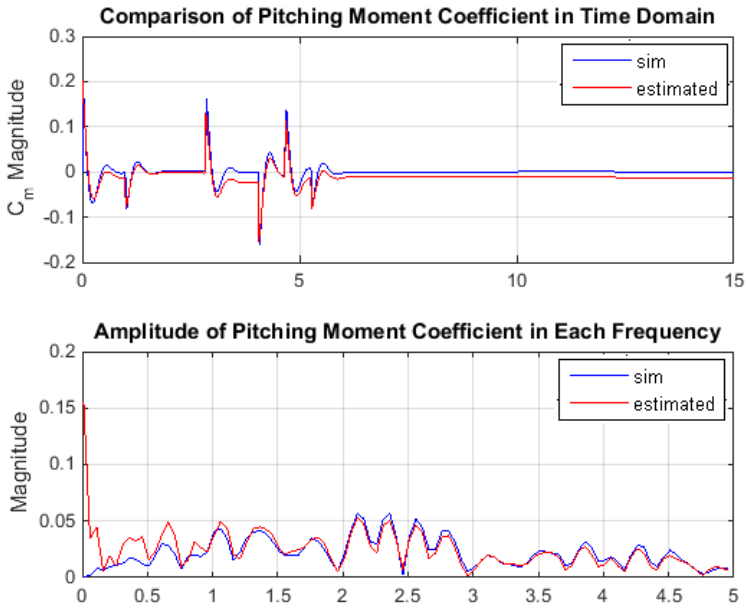


Figure 9.4: Comparison of Pitching Moment Coefficient (C_m) in time and frequency domain in case of inaccurate model for NDI control ($C_{m_{\delta_e}}$ 10% missing).

Similarly, the same flight data used in the first instance was also used for evaluation in the second case. In this case, the aerodynamic stability and control coefficients from frequency OLS estimation from Table 8.5 except $C_{m_{\delta_e}}$ having an error of 10% were defined to validate by calculating pitching moment coefficient C_m with Eq.(3.17) in red colour line illustrated in Fig. 9.4. And the blue line was computed C_m from Eq.(3.15). Both mentioned pitch moment coefficients were compared in time and frequency domain as shown in Figure 9.4. The magnitude of the two lines has some differences in some case of time period and the frequency component due to model errors.

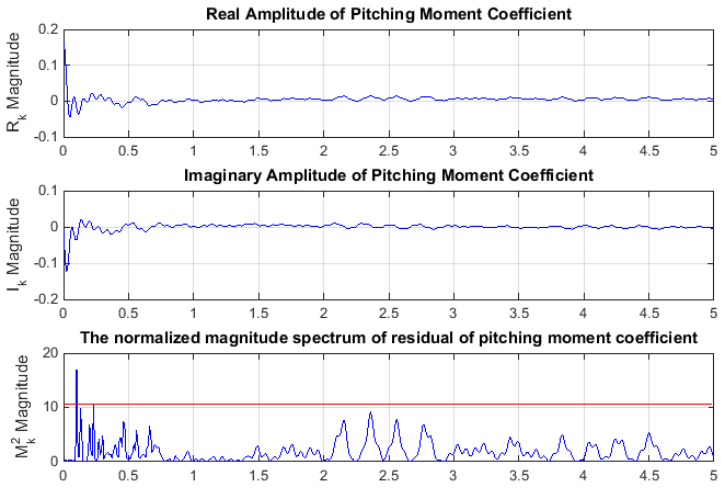


Figure 9.5: The real (R_k) and imaginary (I_k) parts of discrete Fourier transform of residual and normalized magnitude spectrum (M_k^2) in case of aerodynamic parameter from frequency OLS estimation in Table 8.5 but with 10% of $C_{m_{\delta_e}}$ missing

Then the residual of both pitch moment coefficients were calculated according to the procedure in Section 9.1. Therefore, values of R_k, I_k, and M_k in each frequency component calculated with Eq. (9.4), (9.5), and (9.6), respectively with mean (μ_ξ) of 0 and variance (σ_ξ^2) of 0.001 are depicted in Fig. 9.5. It can be seen that some of the magnitude of low frequency components exceed the confidence limit (the 99.5% confidence limit of the χ^2-distribution is between 0 and 10.6 : red line). It means that the aerodynamic parameter is invalidated for NDI based inner loop of flight control system. The agent will execute a plan to compensate the error of the model.

The same flight data in the first case was also implemented to evaluate in the third case. The estimated aerodynamic stability and control coefficients from frequency OLS approach in Table 8.5 except $C_{m_{\delta_e}}$ that has error of 20% were assigned for model validation like in the first case. Figure 9.6 demonstrated the magnitude of both pitching moment coefficients in time and frequency domain. Obviously, the estimated pitching moment coefficient was shifted from the simulated pitching moment coefficient in the time domain and in the low frequency

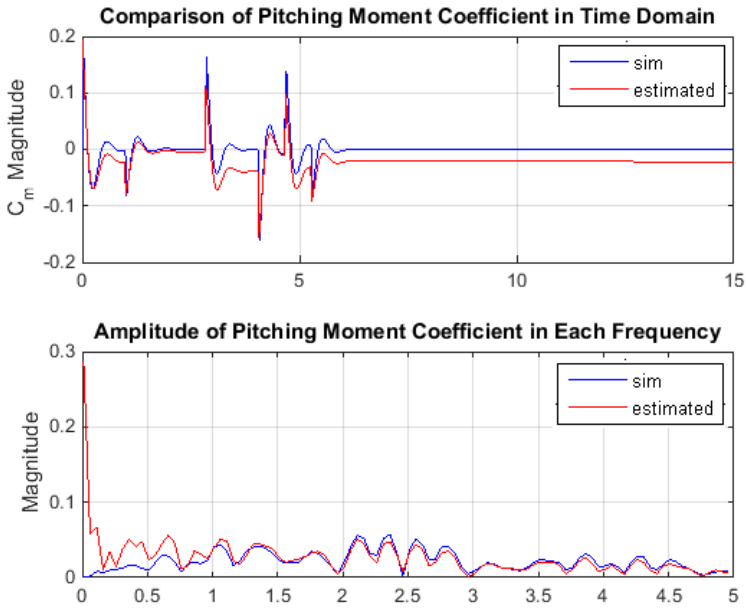


Figure 9.6: Comparison of Pitching Moment Coefficient (C_m) in time and frequency domain in case of inaccurate model for NDI control ($C_{m_{\delta_e}}$ 20% missing).

component.

Then the residual of both pitch moment coefficients in time domain were transformed into the frequency domain and then calculated according to the procedure in Section 9.1. Therefore, values of R_k, I_k, and M_k in each frequency component calculated with Eq. (9.4), (9.5), and (9.6), respectively with mean (μ_ξ) of 0 and variance (σ_ξ^2) of 0.001 are depicted in Fig. 9.7. It can be seen that some of the magnitude in the low frequency components exceeds the confidence limit (the 99.5% confidence limit of the χ^2-distribution is between 0 and 10.6 : red line). All the magnitude in each frequency components grows greater than in the second case.

Finally, in case of elevator failure which occurred in Aerosonade simulation (10% of $C_{m_{\delta_e}}$ loss), the frequency dependent model validation approach was implemented to evaluate the aerodynamic stability and control coefficients in Table 8.5. Figure 9.8 illustrated the magnitude of both pitching moment coefficients in

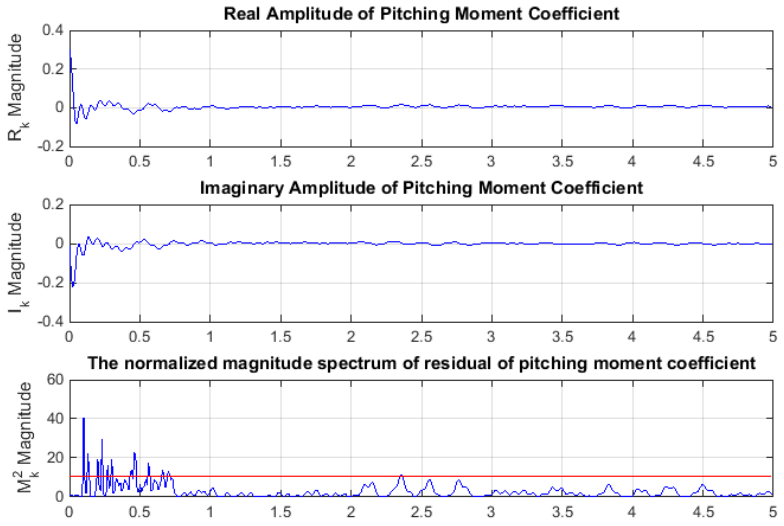


Figure 9.7: The real (R_k) and imaginary (I_k) parts of discrete Fourier transform of residual and normalized magnitude spectrum (M_k^2) in case of aerodynamic parameter from frequency OLS estimation in Table 8.5 but with 20% of $C_{m_{\delta_e}}$ missing

time and frequency domain.

Then the residual of both pitch moment coefficients in time domain were also transformed into the frequency domain and then calculated according to the procedure defined in Section 9.1. Therefore, values of R_k, I_k, and M_k in each frequency component calculated with Eq. (9.4), (9.5), and (9.6) with mean (μ_ξ) of 0 and variance (σ_ξ^2) of 0.001 and 0.002 are depicted in Figure 9.9 and 9.10, respectively. These results show that this method can detect the failure occurred and the difference of variance value can be utilized to compare the level of failure. For instance, some of the magnitude in low frequency components exceed the confidence limit as shown in Fig. 9.9 but all the magnitude in low frequency components remain in the confidence limit as shown in Fig. 9.10. It means we can use the quantity of variance to observe the quantity of failure.

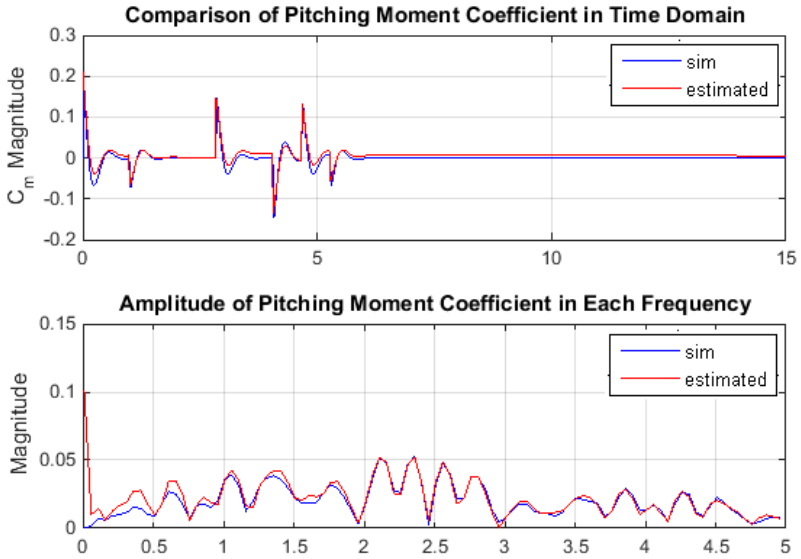


Figure 9.8: Comparison of Pitching Moment Coefficient (C_m) in time and frequency domain in case of elevator failure in simulation model ($C_{m_{\delta_e}}$ 10% loss).

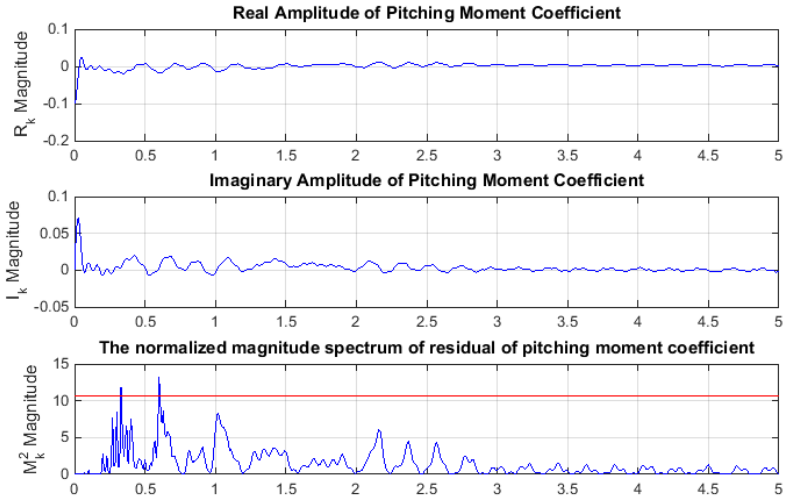


Figure 9.9: The real (R_k) and imaginary (I_k) parts of discrete Fourier transform of residual and normalized magnitude spectrum (M_k^2) with variance ($\sigma_\xi^2 = 0.001$) in case of elevator failure in Aerosonde simulation (10% of $C_{m_{\delta_e}}$ loss)

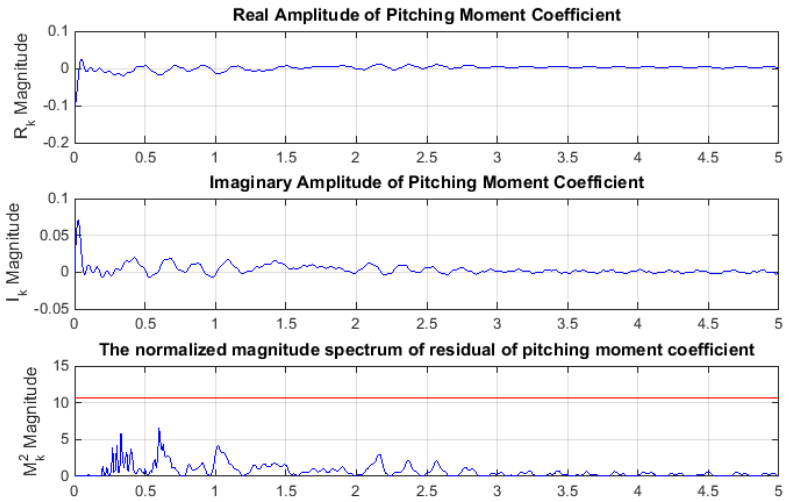


Figure 9.10: The real (R_k) and imaginary (I_k) parts of discrete Fourier transform of residual and normalized magnitude spectrum (M_k^2) with variance ($\sigma_\xi^2 = 0.002$) in case of elevator failure in Aerosonde simulation (10% of $C_{m_{\delta_e}}$ loss)

9.5.2 Results of Flight Trim Condition Monitoring

9.5.2.1 Wavelet Transform Analysis Technique

To evaluate the flight trim condition monitoring approach using wavelet transform analysis, an 3-2-1-1 doublet elevator deflection input and pitch rate response were implemented in MATLAB environment as shown in Figure 9.11 to 9.14.

The elevator deflection input signal was transformed to a time-frequency information and then calculated with $\sum_{a \in a^*_{z_{\delta_e}}} \mathcal{W}^2(a, b_1)[\delta_e]$. Finally, this summation of $\mathcal{W}^2(a, b_1)[\delta_e]$ at each specific time was compared with thresholds value ($\Delta_{trim\_\delta_e}$) in Table 9.1 with condition in Eq. 9.10 to check the flight trim condition status as shown in Fig. 9.11 and 9.12. With visual inspection, these results shown that the elevator 3-2-1-1 doublet elevator deflection input was activated from 11 sec to 15.5 sec as well as no flight trim condition status using this wavelet technique was active from 11 sec to 15.5 sec.

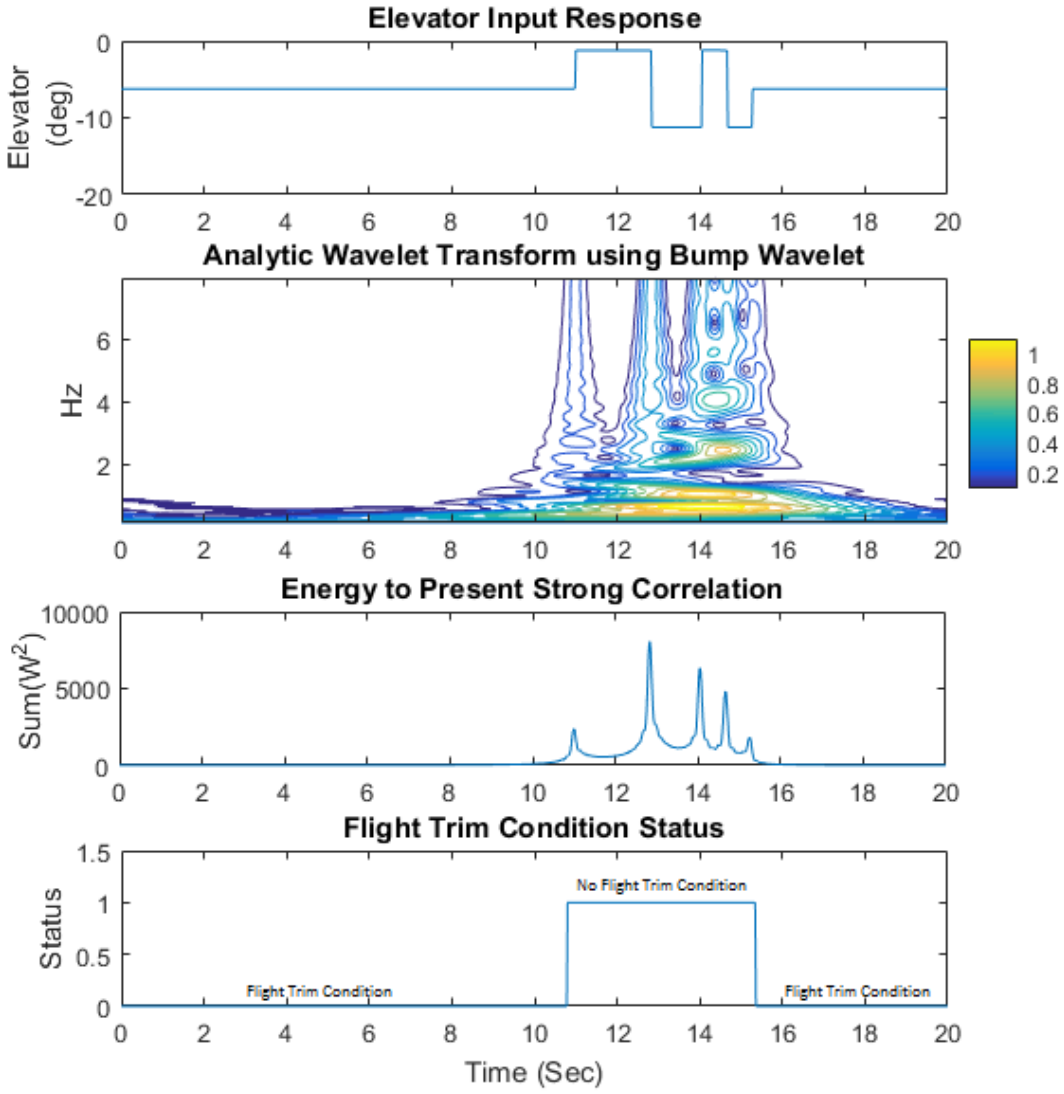


Figure 9.11: Flight Trim Condition Monitoring Technique using Bump Wavelet Transform Analysis with Elevator Deflection Input (δ_e)

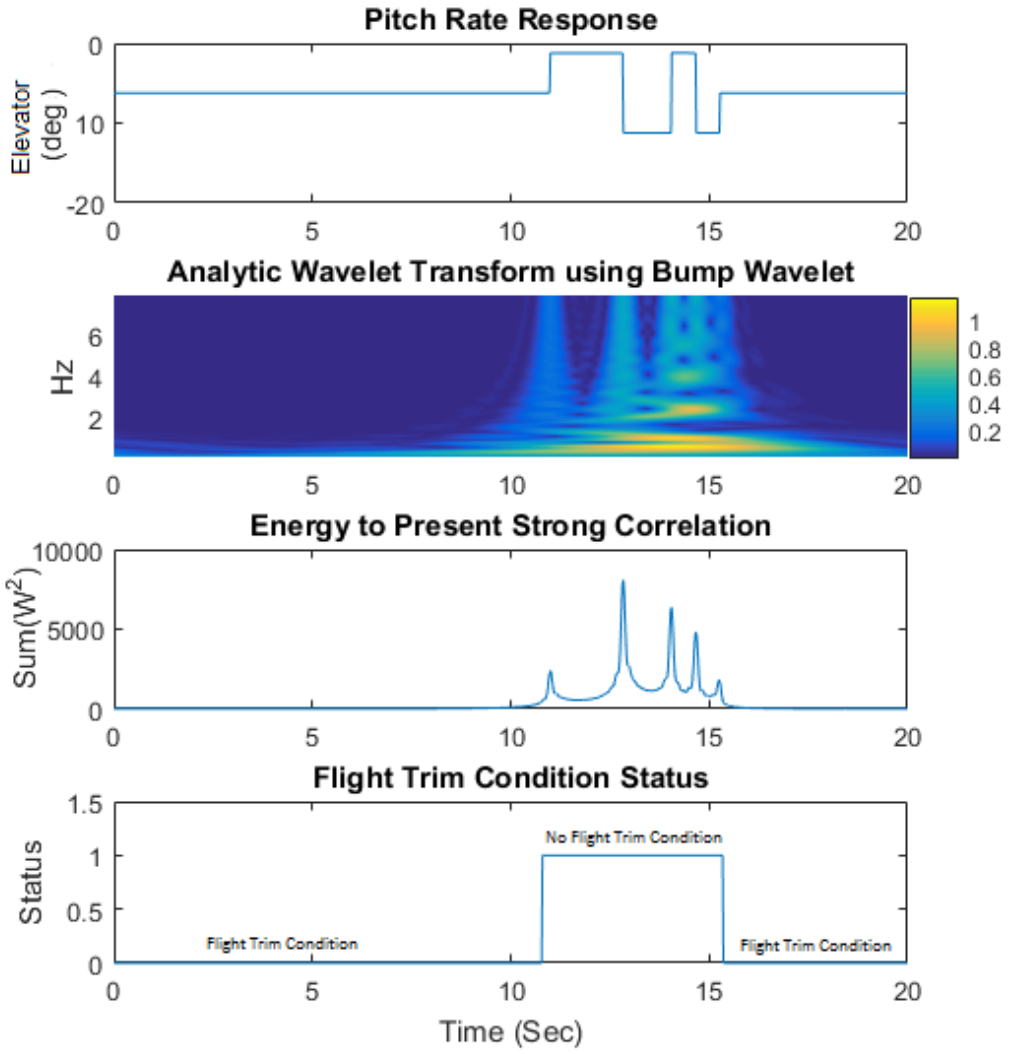


Figure 9.12: Flight Trim Condition Monitoring Technique using Bump Wavelet Transform Analysis with Elevator Deflection Input (δ_e)

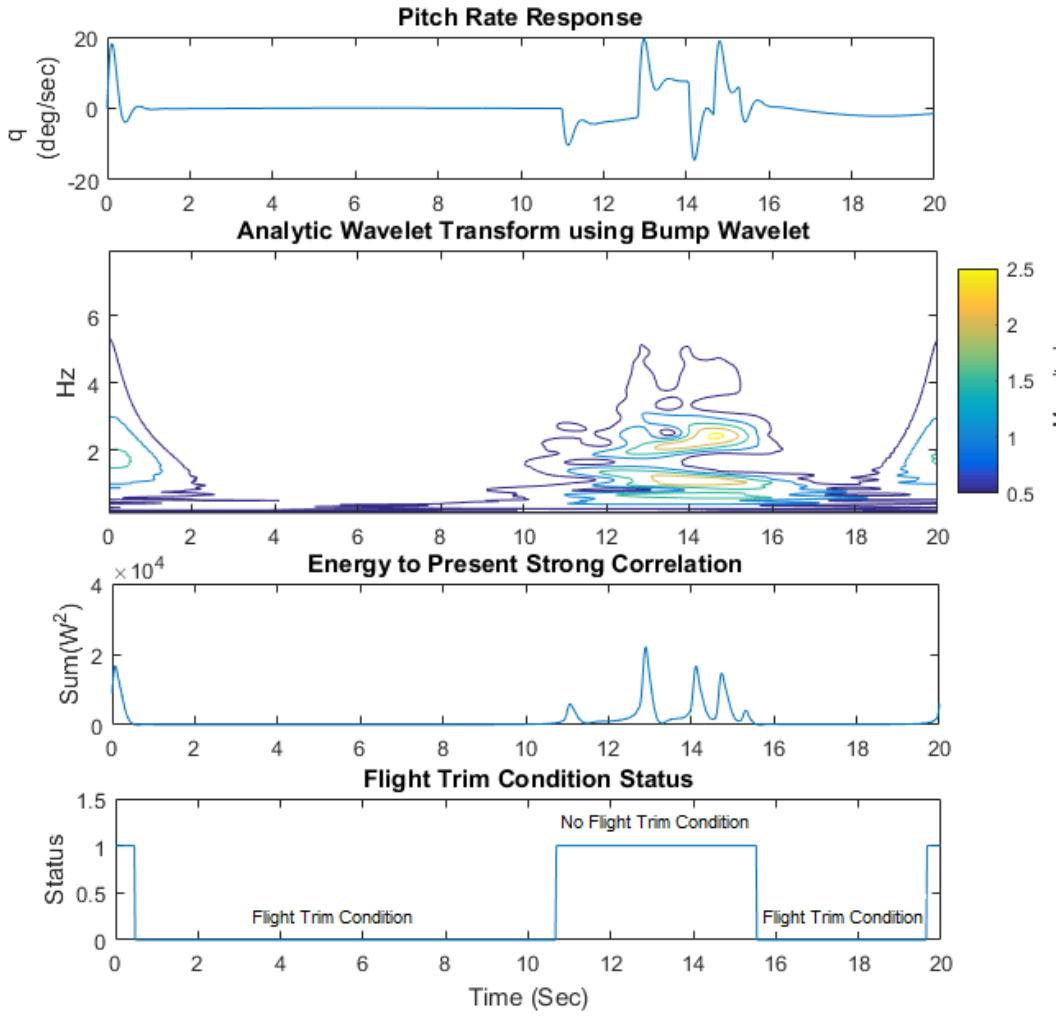


Figure 9.13: Flight Trim Condition Monitoring Technique using Bump Wavelet Transform Analysis with Pitch Rate Response (q)

Furthermore, the pitch rate response was transformed into a time-frequency information with the Bump Wavelet transforms and MRA and then calculated as $\sum_{a \in a^*_{z_q}} \mathcal{W}^2(a, b_1)[z_q]$. Finally, this summation of $\mathcal{W}^2(a, b_1)[z_q]$ at each specific time was compared with thresholds value (Δ_{trim_q}) in Table 9.1 with condition in Eq. 9.9 to check the flight trim condition status as illustrated in Fig. 9.13 and 9.14. Visually, the pitch rate response relied on oscillation of input. Therefore, a quantity of pitch rate was diverged from zero between 11.0 sec and 16 sec which corresponded to the status of no flight trim condition happened at the same time

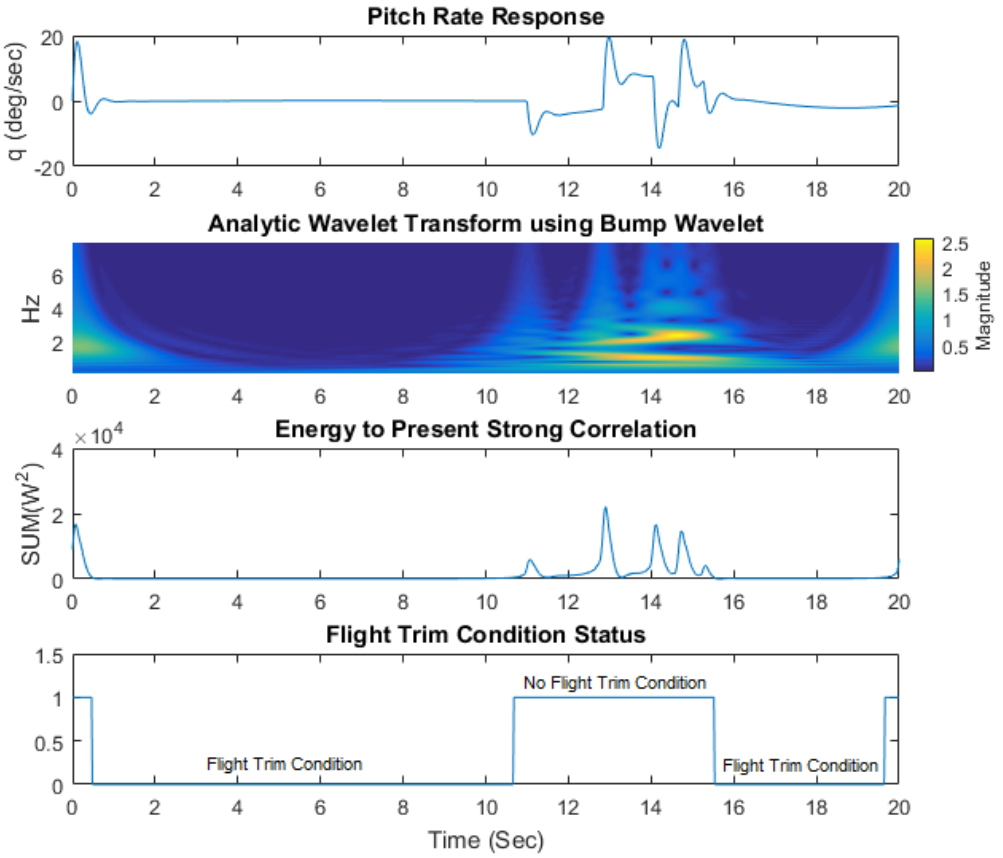


Figure 9.14: Flight Trim Condition Monitoring Technique using Bump Wavelet Transform Analysis with Pitch Rate Response (q)

periods.

9.5.2.2 Frequency Dependent Model Validation Approach

The flight trim condition monitoring method using model validation in frequency domain technique was evaluated here with the pitch rate response as illustrated in Fig. 9.15. In this implementation, the window moving size is assigned as 1 second in order to compute in real-time. And the variance is equal to 0.0005 rad^2/s^2. With visual observation, this method can predict the status of no flight trim condition while the pitch rate response was oscillating. It means that the

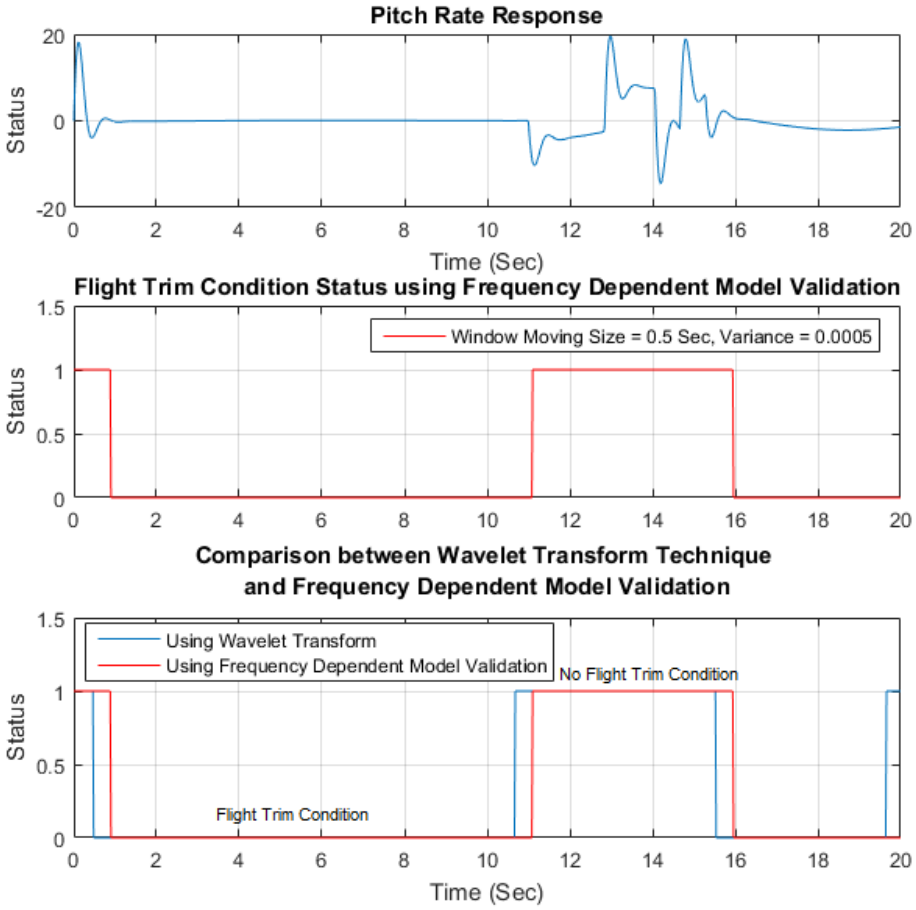


Figure 9.15: Flight Trim Condition Monitoring Technique using Frequency Dependent Model Validation with Pitch Rate Response (q)

frequency dependent model validation can also be utilized to monitor that the aircraft is operating in flight trim condition or not. The key factors of this technique are the window moving size and the variance.

Furthermore, a comparison between both methods was illustrated in Fig 9.15. Both methods can be evaluated efficiently with visual inspection. However, the frequency dependent model validation has a delay about 0.3 sec for monitoring the flight trim condition. This logic status of flight trim condition is a discrete predicate as a belief base.

9.5.3 Results of Control Performance Evaluation

Pitch rate response based on feedback control was implemented in the MATLAB environment to evaluate the control performance evaluation technique with the frequency dependent model validation approach as shown in Fig. 9.16 to 9.17. Fig. 9.16 illustrates that this pitch rate response tried to track the reference command in a form of doublet input where the NDI control with insufficient initial aerodynamic parameter was performed at the beginning and then the hybrid adaptive control was activated at 40s, which oscillating transition was experienced.

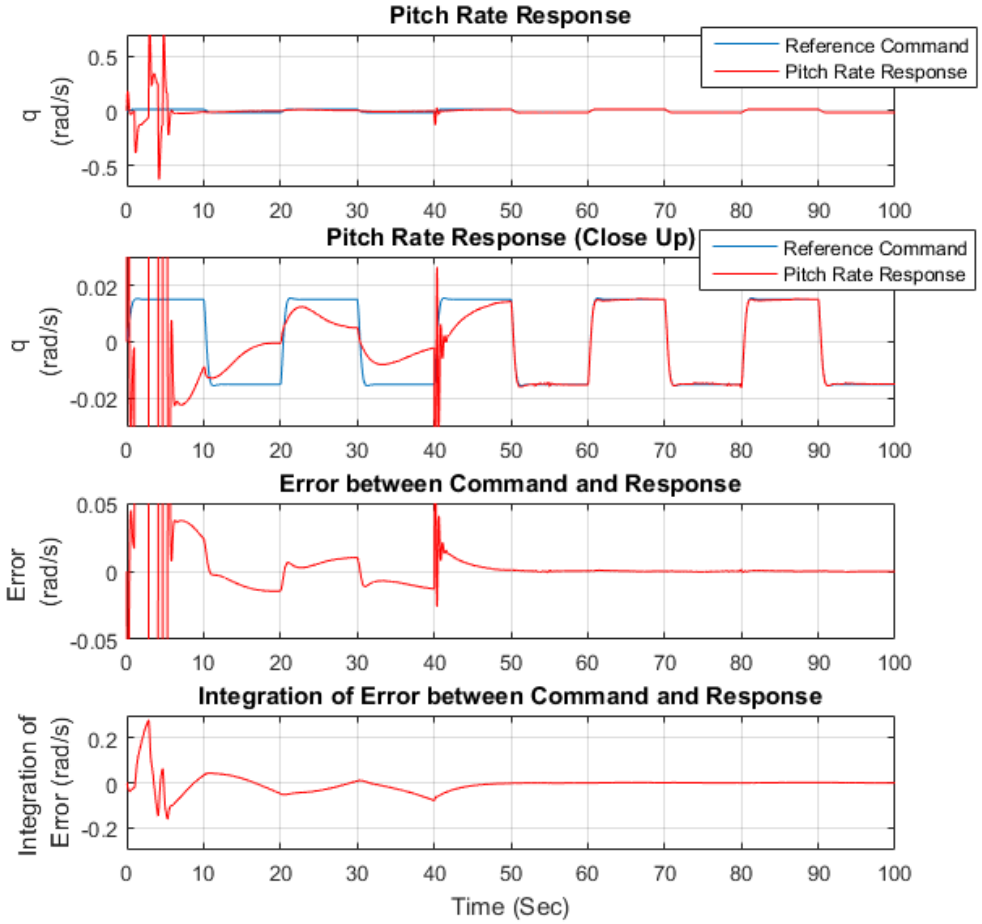


Figure 9.16: Pitch Rate Response (q) compared with reference command. At the beginning, the NDI controller with bad initial aerodynamic parameter was executed. And then hybrid adaptive controller started to perform after 40 sec.

The error of pitch rate response and reference command was utilized to assess the tracking performance of the control system with model validation in the frequency domain as demonstrated in Fig. 9.17 (Note: the window moving size is 2 second and variance is set to 0.000001). Furthermore, this approach, at the same condition, was also evaluated with the integration of error as presented in Fig. 9.18.

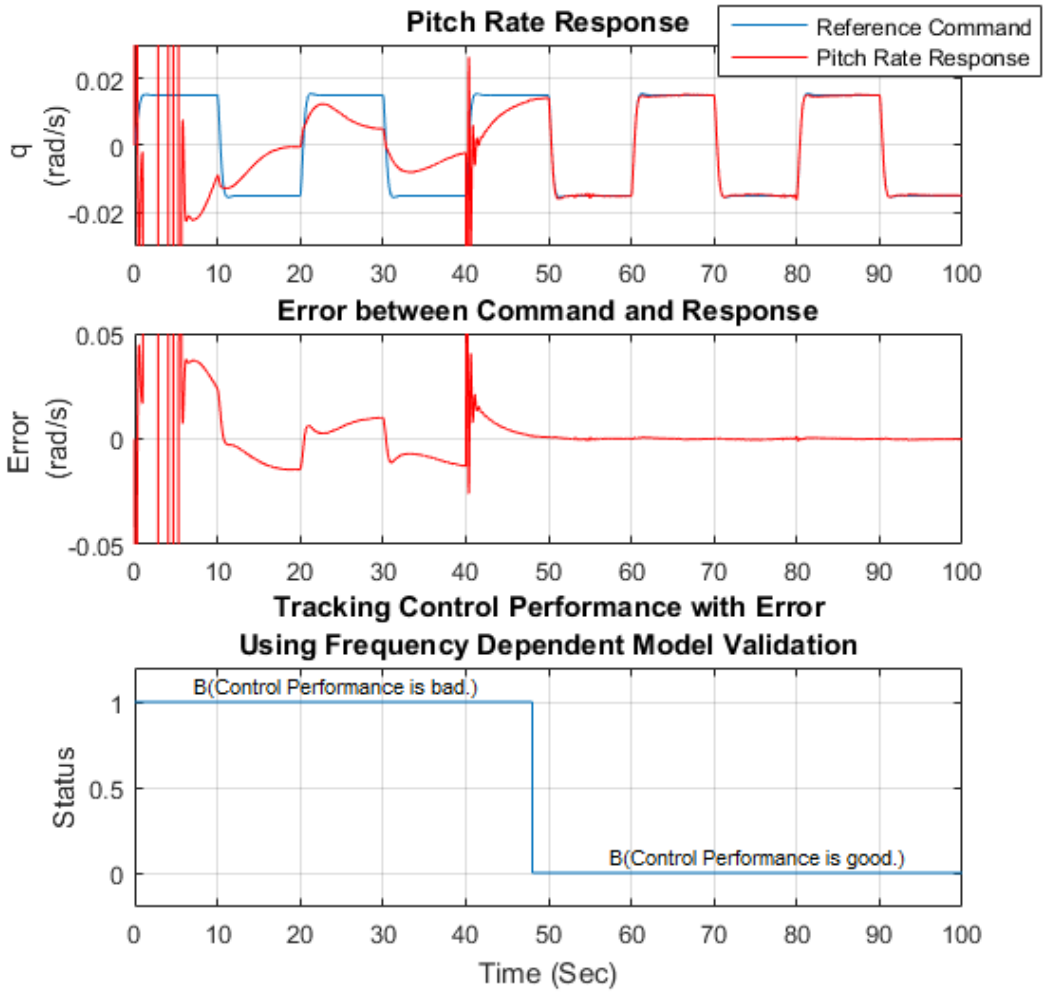


Figure 9.17: Tracking Control Performance Technique using Frequency Dependent Model Validation with Error between Pitch Rate Command and Response ($\mathcal{E}_q$)

Finally, to consider both errors and errors integration, control performance status with errors and integration of errors were combined with OR logical condition. The result of combined control performance status displays in Fig. 9.19. Up to time 50.5 sec control performance is evaluated as unsatisfactory, and after that it is acceptable. Therefore, the procedure of validating a model in the frequency domain has been proven for control performance evaluation. And this status is an updated belief base for an agent to evaluate the control performance.

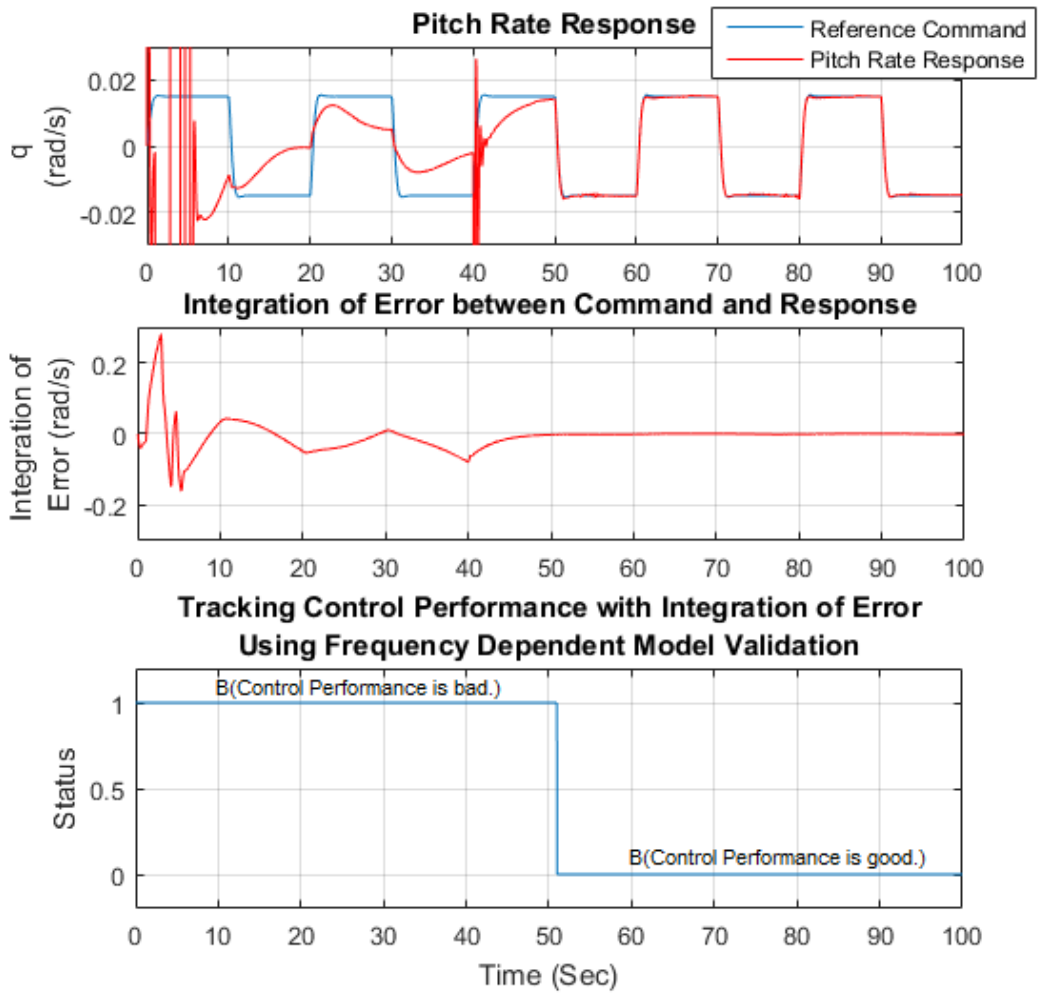


Figure 9.18: Tracking Control Performance Technique using Frequency Dependent Model Validation with Integration of Error between Pitch Rate Command and Response ($\int \mathcal{E}_q dt$)

9.6 Chapter Summary

This chapter has presented real-time evaluations for UAV agent abstractions. The abstraction evaluation method is the approach to filter the continuous perception to the discrete information in order to update the belief base for the rational agent. This study has proposed the development of model validation for NDI control, flight trim condition monitoring, and control performance evaluation.

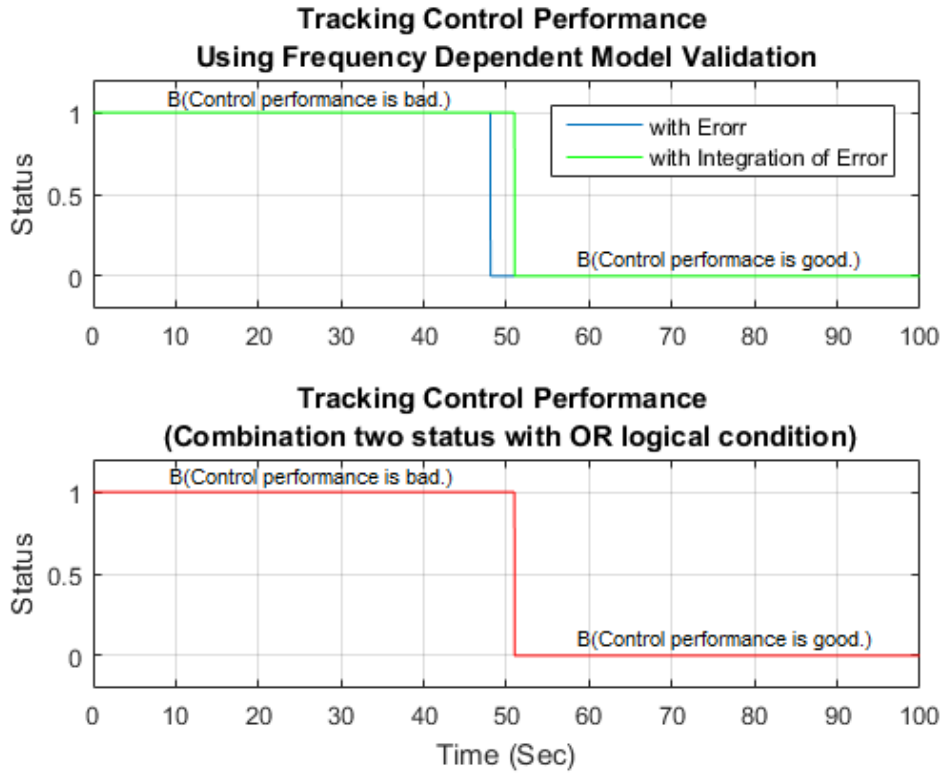


Figure 9.19: Tracking Control Performance Technique with Combination of Two Status with OR Logical Condition

The frequency dependent model validation approach based on a discrete Fourier transform and a hypothesis test of the normalised spectrum with χ^2 distribution have been considered. This algorithm is the key computation to abstract the continuous information to discrete abstractions for 1) validating aerodynamic stability and control coefficients in the inner loop of the NDI controller, 2) monitoring flight trim condition, and 3) assessing the tracking control performance of the flight control system. Therefore, the procedure for validating a model in the frequency domain has been proven in various cases for abstraction evaluation of UAV autopilot agent.

Chapter 10

Decision Methods for UAV Agents

10.1 Agent Development for Reconfiguration

As mentioned, the intelligent control system is designed to reconfigure automatically in different events including various environments, flight conditions, and system degradations. It means that the control system is increasingly demanded to work in the various situations, when the circumstances require a distinct changes in behaviour and often require to switch to the utilization of alternative controllers. Thus, this hybrid control system clearly desires to integrate some decision-making system with the feedback controllers.

Developing autonomous decision making in the hybrid control system can be simplified by an approach that involves choosing abstractions relating the continuous world with discrete decision states. Subsequently, basic rules of behaviour are defined by using these abstractions with goals formulated to maintain the system within constraints and objectives. Therefore, an agent-based approach where goals, plans, and logical inferences are all captured within a rational agent is considered in this work.

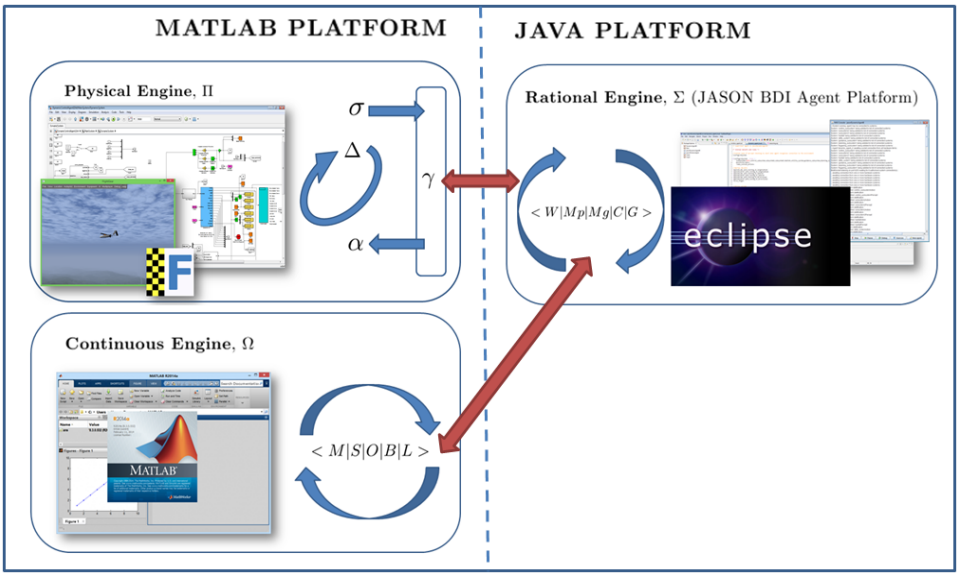


Figure 10.1: Jason to MATLAB (J2M) Interface Diagram

10.1.1 Agent Computational Architecture

sEnglish/Jason with AET (Agent Executive Toolbox, [124], based on the Jason+ agent architecture) has been used to program a high-level rational agent to provide increased capability of intelligent adaptation for the autopilot system. This rational-agent architecture is shown in Figure 10.1 by analogy of the system outlined in [73]. This architecture of the agent consists of three main constituents including (1) physical engine (Π), (2) continuous engine (Ω) and (3) reasoning engine (Σ). The physical engine is a process to receive signals from sensors, to generate signals for actuators and to form symbolic logic abstraction for discrete event decision making of the agent. A continuous engine is a computational unit to calculate functions for continuous signal processing of sensing and control and to abstract symbolic logic expressions for discrete hypothetical events generated by simulation for a future time horizon. The reasoning engine, finally, is a rule-based decision making process to decide upon actions for the physical engine in order to achieve goals, which in this case relates to smooth control of the aircraft to locations despite the atmospheric disturbances.

Definition. A rational behaviour engine, γ, is a tuple, $\gamma =< W|M_p|M_g|C_s|G >$, which consists of a granulated multi-resolution and multi-domain symbolic world

model W, abstract physical skills memory M_p, goal achievement memory (problem solving memory) M_g, abstract formulation of behaviour constraints C_s, abstract formulation of short and long term goals G.

Definition. A continuous engine, Ω, is a tuple, $\Omega =< M|S|O|B|L >$, where M is a set of approximately continuous models of the world, S is a continuous time simulator which uses analytical and empirical data based dynamic models to predict future state of the world, O is an optimizer which can optimize continuous time planning of actions, B is a Boolean evaluator of propositions in terms of σ statements and L is a library of useful numerical computations in terms of continuous variables.

For clarity of development, the process of abstraction design for the agent system is aided using NLPr in sEnglish to link abstraction of the continuous phenomena and agent deliberation. Furthermore, this agent architecture is supported by the Cognitive Agent Toolbox (CAT) [74, 124], which is an integration tool for autonomous control systems. CAT contains the Agent Executive Toolbox (AET) for MATLAB/Simulink, which is suitable for developing the continuous engine for the rational agent. AET acts as a bridge to link MATLAB/Simulink and Jason based on Belief-Desire-Intention (BDI) agent architecture. All real-time computations in this chapter are done in MATLAB/SIMULINK and virtual reality displaying a flight.

10.1.2 Abstraction using NLPr Implementation

The methodological knowledge from publications is formulated regarding human engineering concepts that are easily expressed in NLPr to aid agent development. This speeds up the development process and makes the connection between abstraction and agent deliberation [133] clear. There are abstractions (σ) of precepts which are Boolean, and there are objects creating abstractions with reference to the object by a variable in Jason+, this forms a predicate with an argument which can be used in logic based reasoning in Jason+ . Each Boolean type abstraction of perception is monitored during reasoning cycles and is defined by an sEnglish sentence that returns type boolean result that is its single output.

The abilities (or skills) of an agent are tasks such as data manipulation, feed-

back control or other interactions with hardware devices in order to achieve a particular result. In developing agent skills in this manner, abilities are abstracted into clear functional components thus making subsequent usage of each component more intuitive through NLPr text abstractions in the form of sentences in sEnglish. Then such sentences can be compiled into an executable file (MATLAB or C++/ROS).

The skill abstractions in this work are divided into three main categories: communication abstractions (γ), open-loop action abstractions (α), and closed-loop action abstractions (Δ). First, some communication abstractions are used in connection with communication externally with another agent system or hardware. Open-loop control abstraction of an action of type '`runOnce`' runs the executable sentence once within the executive cycle. Closed-loop control abstraction is an action of type '`runRepeated`' for which the corresponding process runs continuously until the agent decides to terminate the action by a command `stopRepeated`'.

As an illustration, the following tables list some of the $\sigma, \gamma, \alpha, \Delta$ abstractions available to our UAV autopilot system in NLPr as produced in sEnglish. The sentences listed are in fact the sEnglish code that are unambiguously compiled into MATLAB and ultimately to lower level languages. In our project, we have used Simulink and FlightGear simulation for 3D visualization and did not yet compile into C++/ROS, which will be the next stage of our work.

Some communications abstractions (γ) of a UAV control agent:

γ_1	Sending message M to ground control station Co.
γ_2	Receiving message Mr from ground control station Co.

Some open-loop abstractions (α) of a UAV control agent:

α_1	Applying 3-2-1-1 input to elevator of UAV for period T_1.
α_2	Applying 3-2-1-1 sequence input to aileron and rudder of UAV for period T_2.
α_3	Updating to new control configuration (update a new aircraft aerodynamic parameter for the inner NDI loop).
α_4	Switching to reset an initial state for adaptive learning rate of direct adaptive control of inner loop.

Some closed-loop abstractions (Δ) of a UAV control agent:

Δ_1	Executing a frequency model validation method to monitor aircraft aerodynamic parameter for inner NDI control.
Δ_2	Using NDI feedback control to track a required Position Pa.
Δ_3	Using hybrid adaptive feedback control to track a required position Pa.
Δ_4	Using indirect adaptive learning to compensate aircraft parameters in longitudinal dynamic for inner NDI loop (Minor NDI Compensation).
Δ_5	Using indirect adaptive learning to compensate aircraft parameters in lateral dynamic for inner NDI loop (Minor NDI Compensation).
Δ_6	Using indirect adaptive learning to compensate aircraft parameters in directional dynamic for inner NDI loop (Minor NDI Compensation).
Δ_7	Using indirect adaptive learning to compensate aircraft parameters in all axes dynamic for inner NDI loop (Minor NDI Compensation).
Δ_8	Estimating new aircraft aerodynamic parameter in longitudinal dynamic for inner NDI control loop (Major NDI Adjustment).
Δ_9	Estimating new aircraft aerodynamic parameter in lateral and directional dynamic for inner NDI control loop (Major NDI Adjustment).
Δ_{10}	Executing a monitoring algorithm to check aircraft condition (trim) status and error indicator.
Δ_{11}	Trigger hybrid adaptive feedback control in outer loop

Some perception abstractions (σ) of a UAV control agent:

σ_1	Required to update aircraft aerodynamic parameters in longitudinal dynamic for inner NDI loop. (Re-identification)
σ_2	Required to update aircraft aerodynamic parameters in lateral and directional dynamic for inner NDI loop. (Re-identification)
σ_3	Required to compensate aircraft aerodynamic parameters in longitudinal dynamic for inner NDI loop. (Minor adjustment of NDI with indirect adaptive control)
σ_4	Required to compensate aircraft aerodynamic parameters in lateral dynamic for inner NDI loop. (Minor adjustment of NDI with indirect adaptive control)
σ_5	Required to compensate aircraft aerodynamic parameters in directional dynamic for inner NDI loop. (Minor adjustment of NDI with indirect adaptive control)
σ_6	Required to trigger a direct adaptive control for the inner loop.
σ_7	Required to reset an adaptive learning rate mechanism of direct adaptive control for the inner loop.
σ_8	Aircraft manoeuvres in the trim condition.
σ_9	Bad Control system performs in the inner loop.
σ_{10}	Bad Control system performs in the outer loop.

10.2 Overall Diagram of Agent Reasoning

The presented agent framework as shown in Figure 6.3 and Figure 10.2 has been fully developed using the sEnglish Publisher and the AET toolbox [124]. The rational agent controlling the UAV is assigned to track a sequence of required waypoints in terms of latitude/longitude/altitude while being able to handle abnormal dynamic flight conditions by its reconfigurable UAV control abilities which relies on system identification, adaptive non-linear control system, model validation and control performance evaluation to improve flight performance.

Furthermore, MATLAB/SIMULINK has the capability to provide multi-threading of the physical and continuous engines of our agent. In the implementation, computational workloads in this work were divided into eight executive processes as

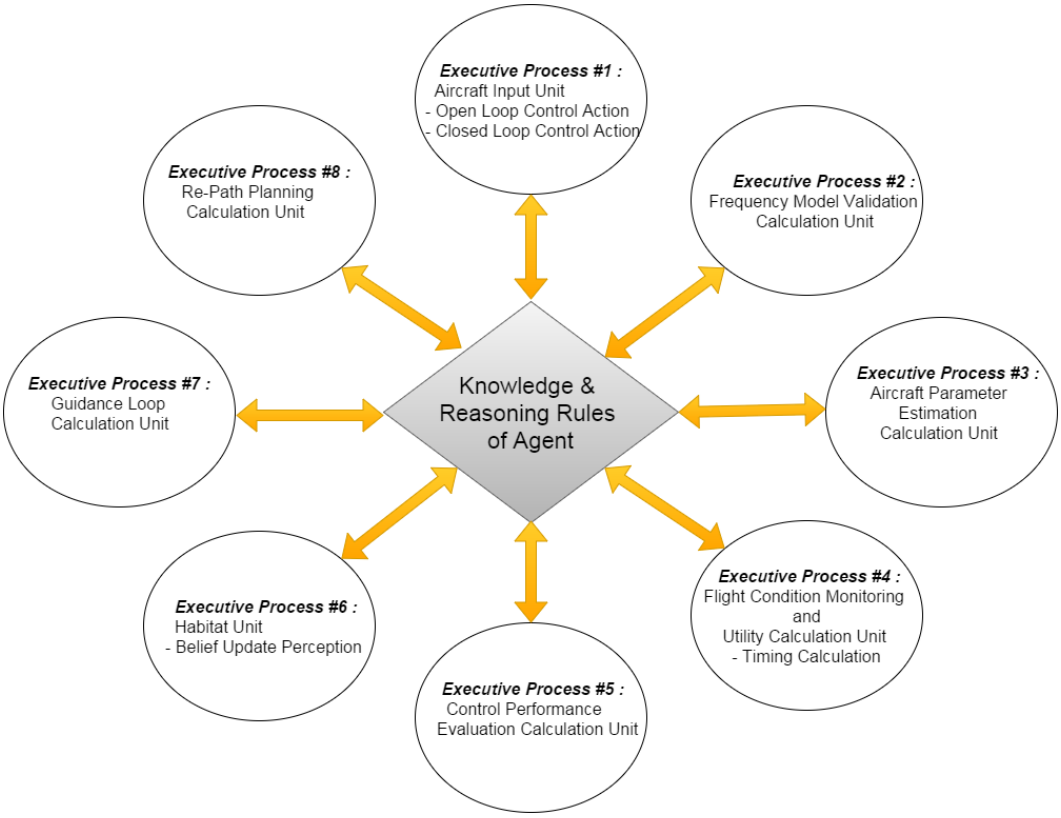


Figure 10.2: Working Diagram of Execute Processes

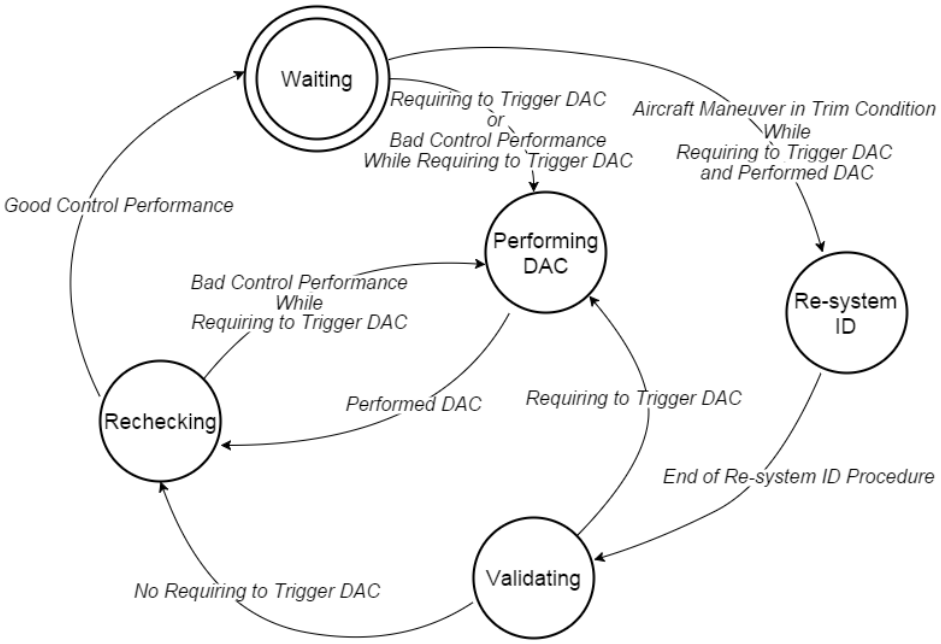


Figure 10.3: Diagram of Knowledge Based Rules for Triggering DAC

illustrated in Figure 10.2. An executive process is defined as a Simulink block with the associated skill item which can also perform perception abstractions communicated to a Jason belief base or action execution as requested by a Jason agent. Each executive process is responsible for different calculations or jobs the agent needs to be able to perform.

Agent programming supports designers or engineers to express decisions in terms of what the agent wants to achieve and how the agent will deal with any unusual events. The key feature of deliberation within agent programs permits the decision making part of the system to adapt intelligently to situations and system degradation. To make it more reliable and autonomous, the autopilot system requires suitable adaptivity for the operation. Therefore, concepts of reasoning rules of how to provide adaptability through switching and tuning of appropriate controllers under uncertain dynamics and situations are proposed here:

1) *Reasoning for Triggering Direct Adaptive Control in the Inner Loop*

Upon activation, our rational agent system based on the NDI control archi-

tecture in the inner and outer loops achieves its goal of stabilising the aircraft and tracking specified waypoints smoothly. Meanwhile, each executive process of the agent system performs real-time tasks of model validation and control performance evaluation in parallel. The executive process of model validation unit decides whether or not the aircraft aerodynamic parameters are required to be compensated or updated over a certain frequency range for the inner loop NDI control of autopilot system. Furthermore, another executive process does the computation to observe control performance of the autopilot system.

As illustrated in Figure 10.3, in the case that the agent perceives a belief abstraction of "requiring to trigger direct adaptive control" or "bad control performance in the inner loop" while beliefs of "requiring to trigger DAC" is still active, the agent system performs DAC to compensate model deficiencies in inner loop NDI control. Then the agent generates a belief base of "performed DAC" and re-performs the computing task to re-check the control performance and validate parameters in inner loop NDI control again. If an agent achieves a goal of "good control performance" that agent will wait to listen to another belief else, the agent will repeat to perform DAC.

Furthermore, if the agent receives a perception abstraction of "aircraft is flying in trim flight condition" while agent still perceives beliefs of "requiring to trigger DAC" and "performed DAC", the agent will carry out a system identification procedure to determine new aircraft aerodynamic parameters and then update them for inner loop NDI control of autopilot system. After that, the agent will execute model validation and control performance evaluation, respectively. Then the agent will be back to waiting until hearing any perception abstractions in any changes.

2) *Reasoning for Triggering a Minor Adjustment Mechanism in the Inner Loop*

Similarly, as shown in Figure 10.4, while the agent is waiting to listen for perceptions, if the agent senses belief abstractions of "requiring minor compensation" or "bad control performance in the inner loop" while beliefs of "requiring minor compensation" and "performed DAC" also activate, the agent system performs a calculating task of indirect adaptive control and reset parameter for direct adaptive control to reduce tracking error. Then agent defines a belief base of "per-

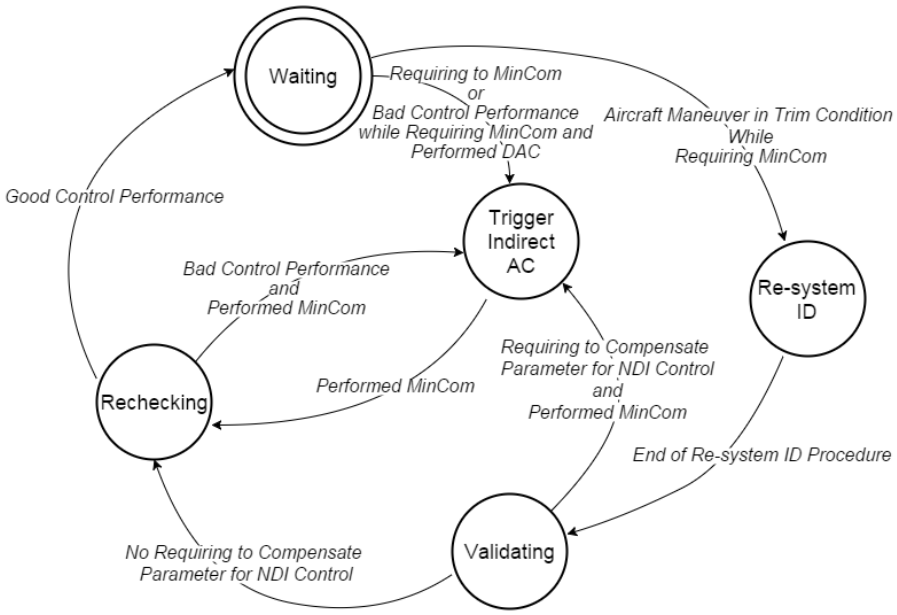


Figure 10.4: Diagram of Knowledge Based Rules for Minor Compensation : MinCom

formed minor compensation" and re-performs the computing task to re-check the control performance in inner loop NDI control again. If the agent achieves a goal of "good control performance" that agent will wait to listen to another belief else, the agent will repeat to perform indirect adaptive control.

In addition, if the agent percepts a perception abstraction of "aircraft is flying in trim flight condition" while the agent still has abstractions of "requiring minor compensation" and "performed minor compensation"as beliefs, the agent will carry out a system identification procedure to determine new aircraft aerodynamic parameters and then update them for inner loop NDI control of autopilot system. After that, the agent will execute model validation and control performance evaluation, respectively.

3) *Reasoning for Triggering a Major Adjustment Mechanism in the Inner Loop*

This case is considered as the aircraft is heavily damaged, so the aircraft aerodynamic parameter changes significantly. The hybrid adaptive control that

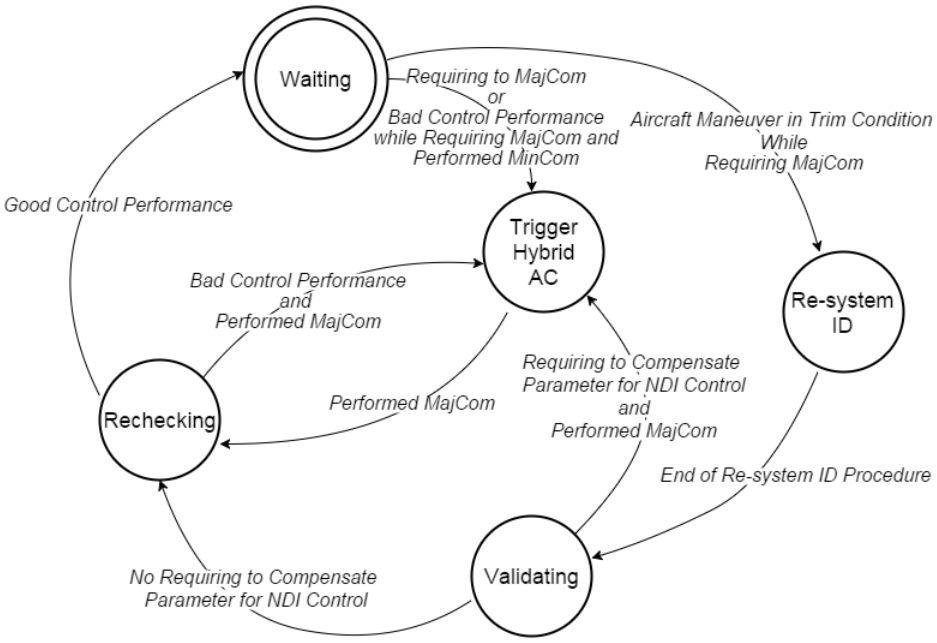


Figure 10.5: Diagram of Knowledge Based Rules for Major Compensation : MajCom

is possible to decrease the effect of high-gain control is applied to meet a good tracking performance according to the diagram in Fig. 10.5. In this case, the agent will perceive perception abstractions of "requiring major compensation " or "bad control performance"while the boolean logic of belief abstraction of "requiring major compensation" is still true. Then the agent will execute a task of hybrid adaptive control and create a belief base of "performed major compensation". Next, the agent will re-perform the computing task to re-check the control performance and validate parameters in inner loop NDI control again. If the agent attains the goal of "good control performance" , the agent will wait to listen to another belief or else, the agent will resort to hybrid adaptive control to improve flight performance.

Similar to previous rule concepts, if the agent percepts a perception abstraction of "aircraft is flying in trim flight condition" while agent still has abstractions of "requiring major compensation" and "performed major compensation"as beliefs, the agent will react system identification procedure to find out new aircraft aerodynamic parameters and then update them for inner loop NDI control of au-

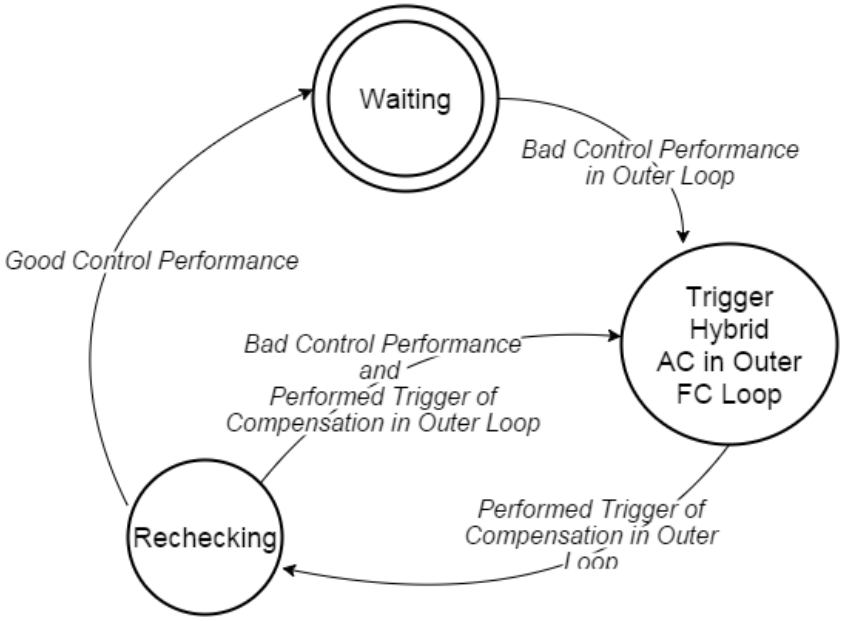


Figure 10.6: Diagram of Knowledge Based Rules for Compensation in Outer Loop of Autopilot System

topilot system. After that, the agent will implement model validation and control performance evaluation, respectively.

4) *Reasoning for Adjustment Mechanism in the Outer Loop*

The outer loop NDI flight control architecture, which is based on the dynamic kinematic equations, does not depend on aircraft parameters. Therefore, the model validation technique is inessential in this reasoning process. If the agent perceives a perception abstraction of "bad control performance in the outer loop", hybrid adaptive control will be executed in the outer loop of autopilot system by the agent. Then the agent produces a belief base of "performed trigger of compensation in outer loop" and act the calculating task to re-check the control performance in outer loop NDI control. If the agent reaches a goal of good control performance, then the agent will wait to catch another belief or else, the agent will repeatedly execute hybrid adaptive control in the outer loop.

5) *Reasoning for Executing a New Plan via Re-Path Planning Algorithm for Emergency Landing Situation :* This is an optional reasoning rule for an emergency situation. Based on this the agent can utilise physical information such as aerodynamic stability and control coefficient to diagnose the health of the aircraft. For instance, after the agent selects an appropriate control method for the autopilot system, the agent will do system re-identification to estimate a new set of aerodynamic parameters while the aircraft is operating in trim flight condition or the agent is trying to compensate the aircraft parameter with indirect adaptive control. The new aircraft parameters can be compared with the old set of aircraft aerodynamic information and analysed to detect and identify if a failure occurred [75].

This failure information can be associated with an algorithm to select an emergency landing site using computer vision [40], which can be used to make a decision for the agent where to land in case of emergency. Then the agent can execute a task to generate a new flight path for UAV forced landing [102] and keep tracking the path to the landing site. However, the details of an emergency landing are beyond the scope of this thesis and it will be investigated in the future.

The development of the agent, to select an appropriate control methodology, depends on NLPr abstractions and ontology developed within sEnglish Publisher [124, 133]. A single sEnglish document contains high level code for agent reasoning in Jason(AgentSpeak) as well as the definitions of all signal processing and control processes which compile into MATLAB, which is used in Simulink or ultimately compiled into C++/ROS for embedded systems, is not considered in this thesis.

Figure 10.7 and 10.8 show a simple example of the use of sEnglish in reasoning (Figure 10.7) with corresponding Jason code (Figure 10.8). The "invoke()" function call execute process named "subsystem2" and "utility_system" to perform "validate_model_in_frequency_domainV2" and "monitor_aircraft_status", respectively. These two functions receive the updated aircraft states to evaluate if the agent is required to update aircraft aerodynamic parameters in longitudinal dynamics for the inner NDI control loop, expressed by the listed σ_1 abstraction as "Requiring to update aircraft aerodynamic parameters in longitudinal dynamics

INITIAL BELIEFS AND GOALS
-Applied dac auto tunning in longitudinal.
-Applied minor auto tuning in longitudinal.
-Applied dac auto tunning in lateral.
-Applied minor auto tuning in lateral.
-Applied dac auto tunning in directional.
-Applied minor auto tuning in directional.
+Applying hac control.
+First flight.

INITIAL ACTIONS
Apply Force and set the initial parameter of aircraft for agent.
Calculate Pathpoint using dubin flight path planning from Aircraft_position, Aircraft_heading, Target_position, Finaltarget_direction and Option.
Calculate Input for flight control system using line of sight guidance algorithm from P, V and Required_altitude.
Apply Control_surface_input using hybrid adaptive control for flight control loop from Input_command, State_ndi and State_ndi_m.
if (~first_flight) {
Check Update_state_of_model in term of model validation for nonlinear dynamic inversion control in frequency domain from State_ndi.};
Receive Perceptions from environment and other processor like agent adminssion.
Monitor aircraft manoeuvre condition or status from Altitude and send back output of NoUtilityFucn.

PERCEPTION PROCESSES

Monitor the following Booleans :

Monitor the following objects :

REASONING
If ~^[First flight.] then +^[Applied revalidation.].
If ~^[Compensate parameter.] then +^[Switch off compensate parameter.]

EXECUTABLE PLANS
If +^[First flight.] under the condition of ^[True.]
then do the following: +^[Applying minor auto tuning in lateral.]
+^[Applying minor auto tuning in directional.]
+^[Applying minor auto tuning in longitudinal.]
[Calculate Theta that is the compensator makeing use and compensate of model-based adaptive control routines for first flight from State_ndi.]
[Stopping monitor aircraft manoeuvre condition or status from Altitude and send back output of NoUtilityFucn.]
[Set the terminated time of thirty second and send output back with NoUtilityFunc.]
[Start to record the clock timer and send NoUtilityFunc back.]
...
...

Figure 10.7: Illustration of the definition of an agent reasoning processes in format of sEnglish sentences in NLP defined within sEnglish document that define the *.sej file, Appendix IV

```
22 //INITIAL ACTION
23 +!take_initial_actions <-
24     invoke(control_subsystem,runOnce,initial_parameter_for_agent,[],[]);
25     invoke(planning_subsystem,runRepeated,plan_flight_path,[Aircraft_position,
26           Aircraft_heading,Target_position,FinalTarget_direction,Option],[]);
27     invoke(guidance_subsystem,runRepeated,apply_guidance_control12,[Aircraft_position,
28           Aircraft_airspeed,Required_altitude],[]);
29     invoke(control_subsystem,runRepeated,apply_hac_control,[Input_command,
30           State_ndi,State_ndi_m],[]);
31     invoke(subsystem2,runRepeated,validate_model_in_frequency_domainV2,[State_ndi],[]);
32     invoke(habitat,runRepeated,update_percept,[],[]);
33     invoke(utility_system,runRepeated,monitor_aircraft_status,[Altitude],[]);
34     +applied_revalidation;
35     +applying_hac_control;
```

Figure 10.8: The Example of Jason/AgentSpeak Language for Abstraction Process that define the *.asl file, Appendix IV

```
356 +aircraft_in_trim_status : auto_tuning & requiring_major_adjustment_in_longitudinal
357                           & not applying_major_auto_tuning_in_longitudinal
358                           & not applying_minor_auto_tuning_in_longitudinal
359                           & not applying_dac_auto_tuning_in_longitudinal
360                           & not applying_major_auto_tuning_in_lateral
361                           & not applying_minor_auto_tuning_in_lateral
362                           & not applying_dac_auto_tuning_in_lateral
363                           & not applying_major_auto_tuning_in_directional
364                           & not applying_minor_auto_tuning_in_directional
365                           & not applying_dac_auto_tuning_in_directional <-
366       +applying_major_auto_tuning_in_longitudinal;
367       invoke(utility_system,stopRepeated,monitor_aircraft_status,[],[]);
368       invoke(utility_system,runOnce,set_speed_command_to_cruise_speed,[],[]);
369       invoke(control_subsystem,stopRepeated,apply_hac_control);
370       invoke(utility_system,runOnce,set_terminated_time_of_ten_second,[],[]);
371       invoke(control_subsystem,runOnce,
372             set_initial_value_for_3_2_1_1_input_in_longitudinal,[],[]);
373       invoke(subsystem3,runOnce,
374             set_initial_value_before_reidentification_in_longitudinal,[],[]);
375       invoke(utility_system,runRepeated,reckon_time,[],[]);
376       invoke(control_subsystem,runRepeated,apply_3_2_1_1_for_pitch_force,[],[]);
377       invoke(subsystem3,runRepeated,
378             estimate_new_aircraft_aerodynamic_parameters_for_longitudinal,[State_ndi],[]);
379       +applying_major_auto_tuning_with_hac_in_longitudinal.
```

Figure 10.9: Illustration of the definition of an agent reasoning processes in Jason/AgentSpeak that define the *.asl file, Appendix IV.

for inner NDI loop.". The sub-system then sends the abstraction back to the execute process named "habitat" with "update_percept" function for making a decision.

In addition, implementation of agent reasoning, for instance, the execution of an open-loop control-plan, which links to the α_1 abstraction "Applying 3-2-1-1 input to elevator of UAV for period T1",is shown in line 376 of Figure 10.9. The code example in Figure 10.9 may be read as: given the condition of "Aircraft executes a manoeuvre in the trim condition.", under the condition that it is believed that the agent requires to update the aircraft aerodynamic parameters in longitudinal dynamics for the inner NDI loop, then do the following: stop monitoring

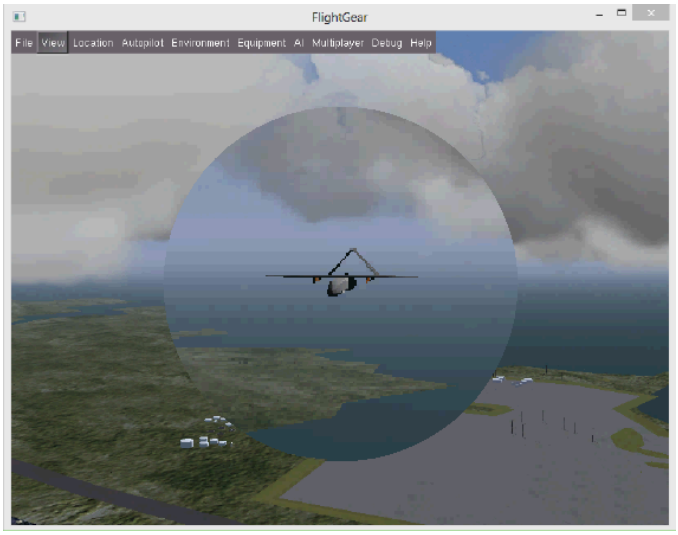


Figure 10.10: 3D Visualization with FlightGear Flight Simulation [1]

and closed-loop control. Then set an initial values before re-identification. Next activate the 3-2-1-1 sequence elevator input and then estimate new aerodynamic parameters for the inner (longitudinal) NDI control loop. More detail of sEnglish development and implementation can be found in [133].

10.3 UAV Simulation Environment

Non-linear Aerosonde UAV [129] simulation was used to evaluate agent performance in the MATLAB/SIMULINK environment. FlightGear flight simulation software [1] was employed to aid in 3D visualisation to monitor the aircraft dynamics via a UDP communication interface as shown in Figure 10.10. The DATCOM [104] or TORNADO [2] software was used to provide numerically calculated aerodynamic stability and control coefficients from aircraft geometry for the initial parameter of the inner NDI control Loop.

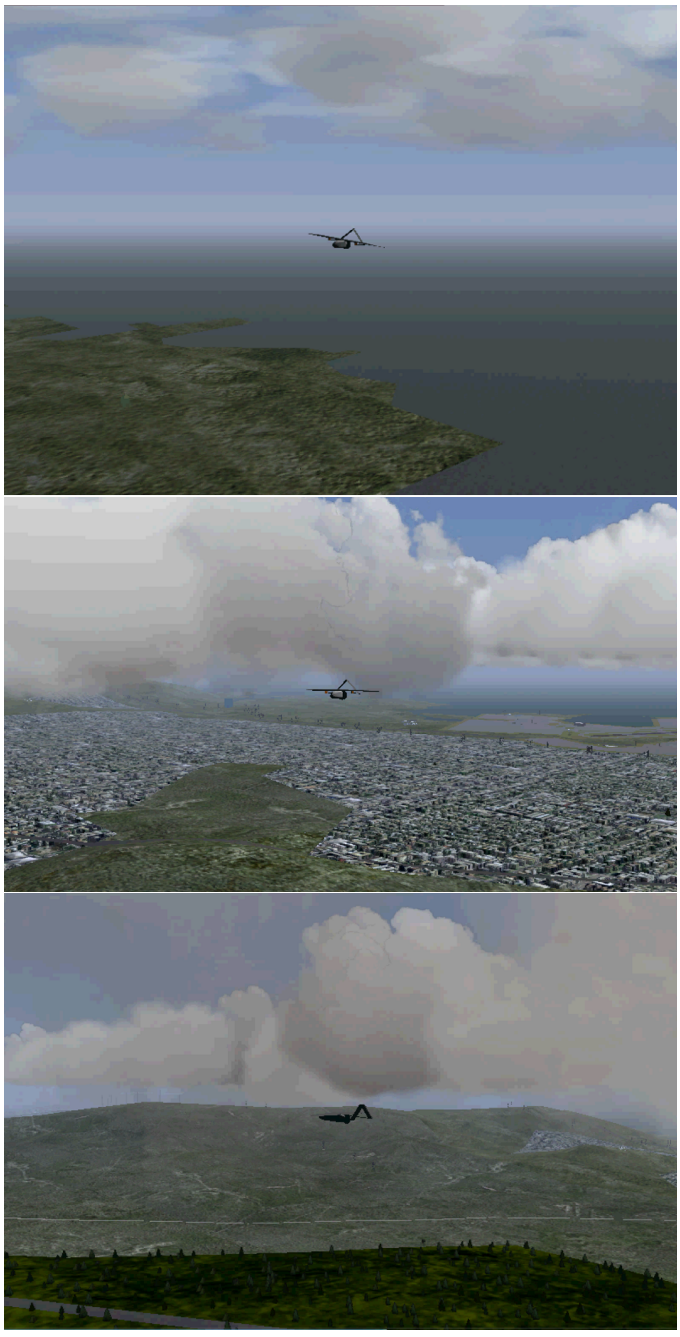


Figure 10.11: Real-Time Flight Simulation of the Agent Controlled at 3 Stages of the Flight with 3D Visualization (FlightGear Flight Simulation [1]) depicting activation of agent based control system.

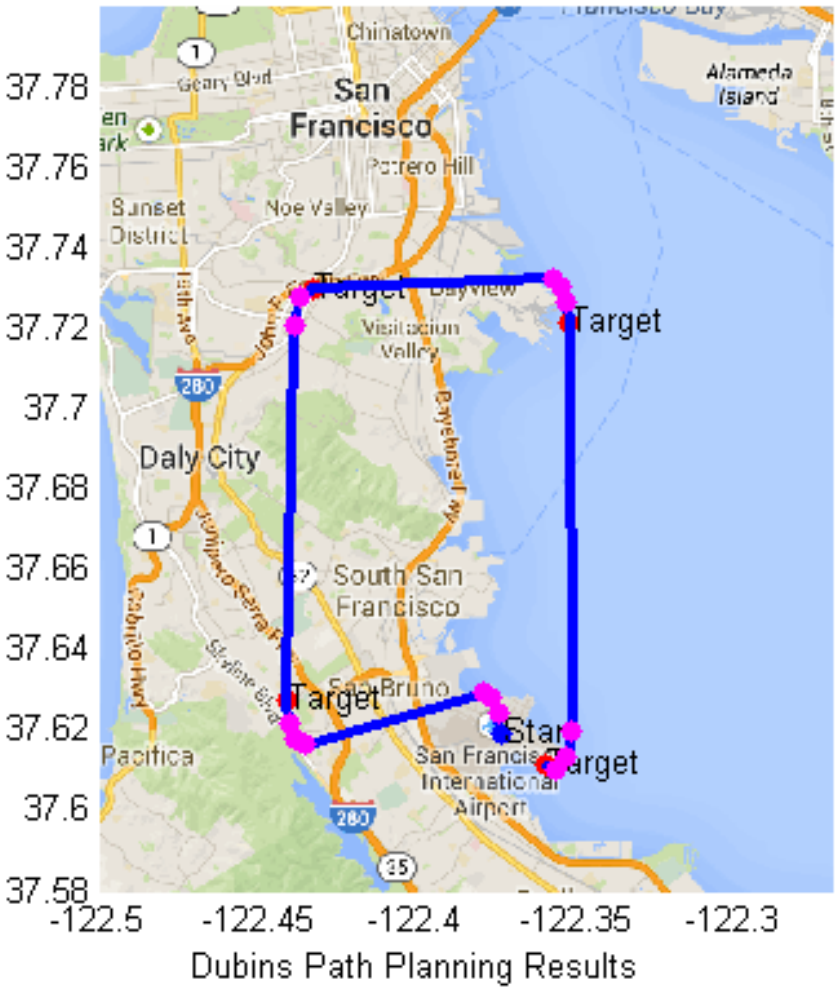


Figure 10.12: 2D Google Map to Illustrate the Desired Flight Path. [1]

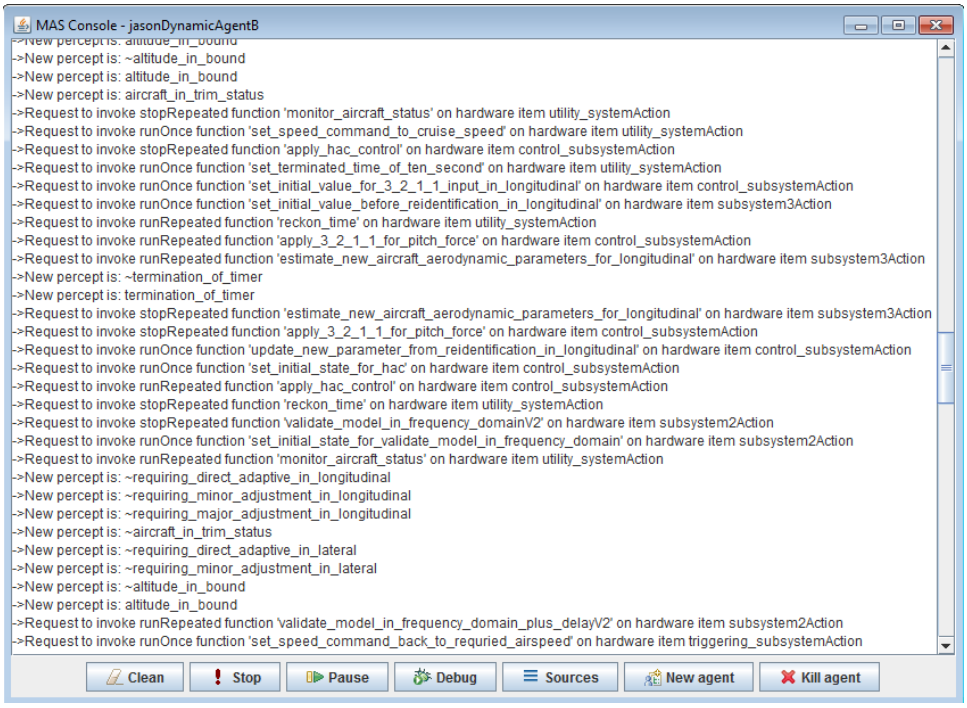


Figure 10.13: Example Result of Decision Making based on Jason Reasoning (Case Study I)

10.4 Computational Experiments

The agent based methodology presented has been evaluated in a non-linear Aerosonde simulation environment which belongs to AeroSim Blockset and its decision making ability was assessed.

10.4.1 Case Study I : Insufficient Initial Parameters and First Flight Tuning

In this case study, our agent system was placed in a difficult situation where the initial controller gains were insufficient to stabilise the aircraft and provide smooth tracking of the waypoint in order to evaluate its performance. Therefore, our agent system took over the responsibility to reconfigure the autopilot system by intelligently tuning and switching the controller of the inner loop of the autopilot system based on reasoning rules mentioned.

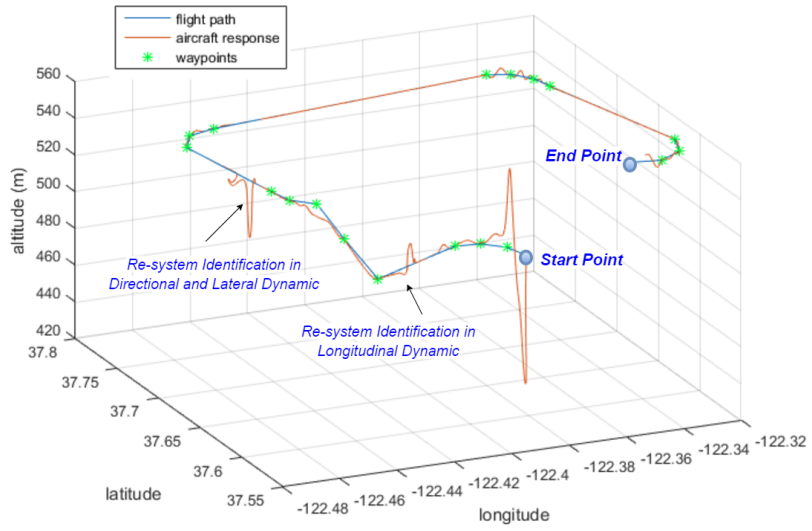


Figure 10.14: Overall 3D Simulated Result demonstrating operation using agent based control system (Case I)

The example output of a Jason BDI agent, based on prescribed reasoning under insufficiency of initial parameter accuracy, is demonstrated in Figure 10.13. The figure shows that the agent triggered a plan for re-identification of the longitudinal dynamics under the belief of needing to update new aircraft aerodynamic parameters of inner NDI control loop after the agent believes that aircraft manoeuvres in the trim flight condition and it requires updating the aircraft aerodynamic parameters in longitudinal dynamics for the inner NDI loop. And then the agent execute a task to validate aircraft aerodynamic parameters.

Figure 10.14 illustrates an overview of mission in 3D view and aircraft response states while operating. In addition, Figure 10.15 and 10.16 show the aircraft while tracking the latitude-longitude waypoints and altitude, respectively. At the beginning of the mission, the agent made a decision to select a hybrid adaptive flight control mode to stabilise the aircraft for compensating local deficiencies of initial aerodynamic parameters. In a second decision, the agent activated a re-identification process, including plane testing, in order to update the aerodynamic parameters for the inner NDI control-loop in longitudinal axes. Finally, it generated a decision to re-identify new aerodynamic parameters of the inner NDI

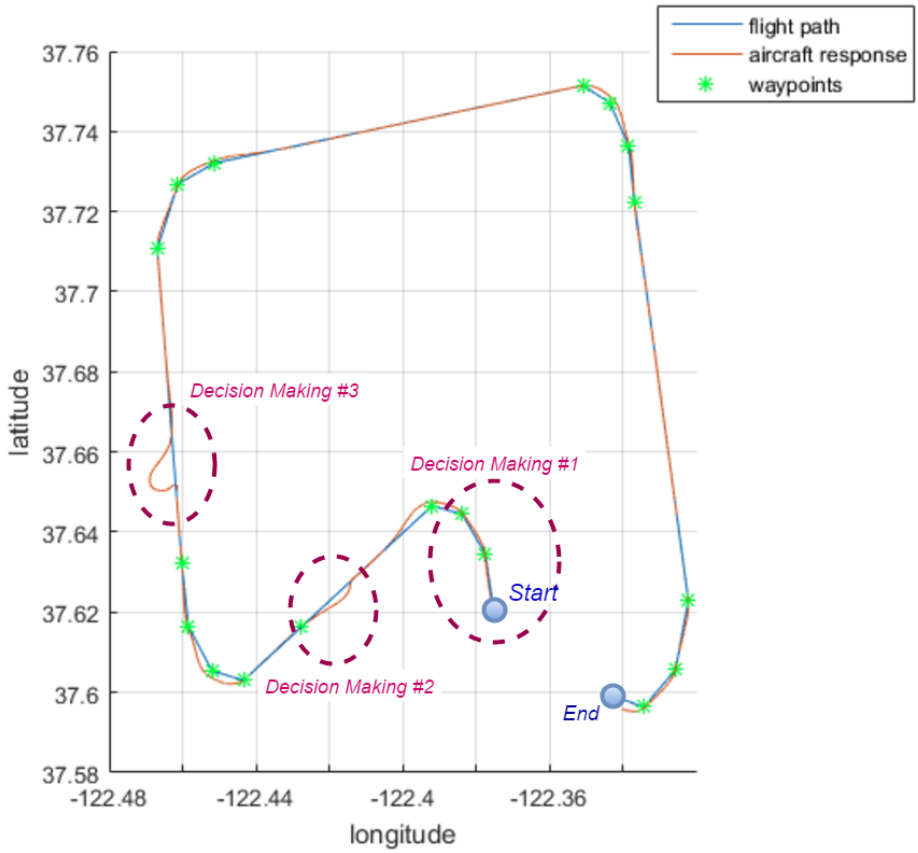


Figure 10.15: Simulated Output of Waypoint Tracking (Case I)

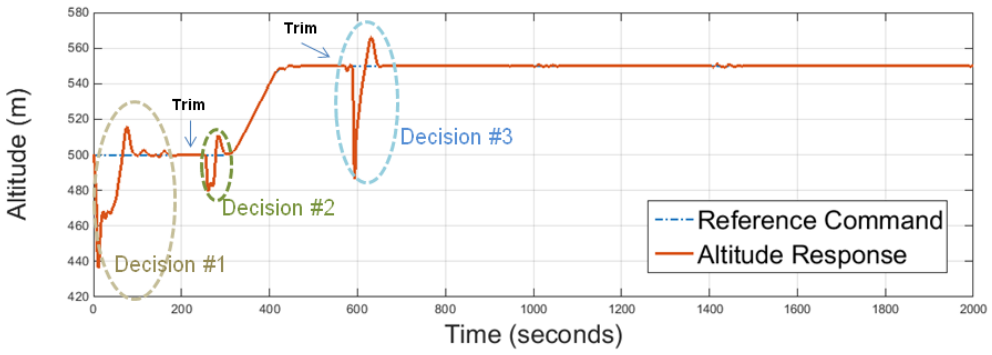


Figure 10.16: Time Histories of Simulated Output States of Altitude Tracking (Case I)

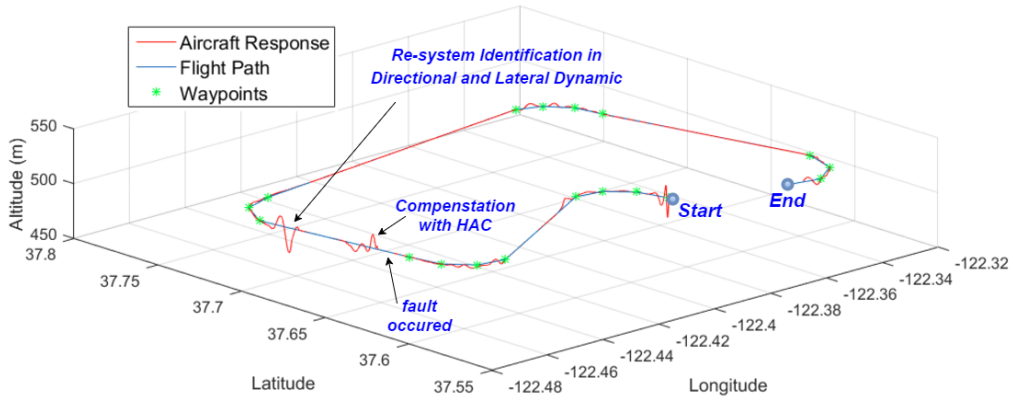


Figure 10.17: Overall 3D Simulated Result demonstrating operation using agent based control system (Case II)

control-loop in lateral dynamics.

10.4.2 Case Study II : Elevator Failure

To evaluate the reliability and robustness of the system, our agent system was made to face a problem of control surface impairment while stabilising and smoothly tracking waypoints. Damage of a control surface was presented in the simulation as a degradation in the control effectiveness of the aileron by 50%. The time of fault occurrence in all control channels was set to be 580s. Our agent system made a decision of tuning and switching the appropriate controller for the autopilot system in order to continue operating until the end of the mission.

Similarly, Figure 10.17 demonstrates a summation of mission in 3D view and aircraft position states whilst in flight. In addition, Figure 10.18 and 10.19 show the aircraft position output of tracking the latitude-longitude waypoints and altitude, respectively. At the beginning of the mission, the agent monitored the control performance to validate a model of the inner NDI control loop while tackling waypoints to complete the mission. Eventually, the agent detected that the control system performed poorly and believed that the system required updating of the aircraft aerodynamic parameter in lateral dynamics for inner NDI loop. The agent made a decision to select a hybrid adaptive flight control mode to stabilise the aircraft and for compensating local deficiencies of initial aerodynamic

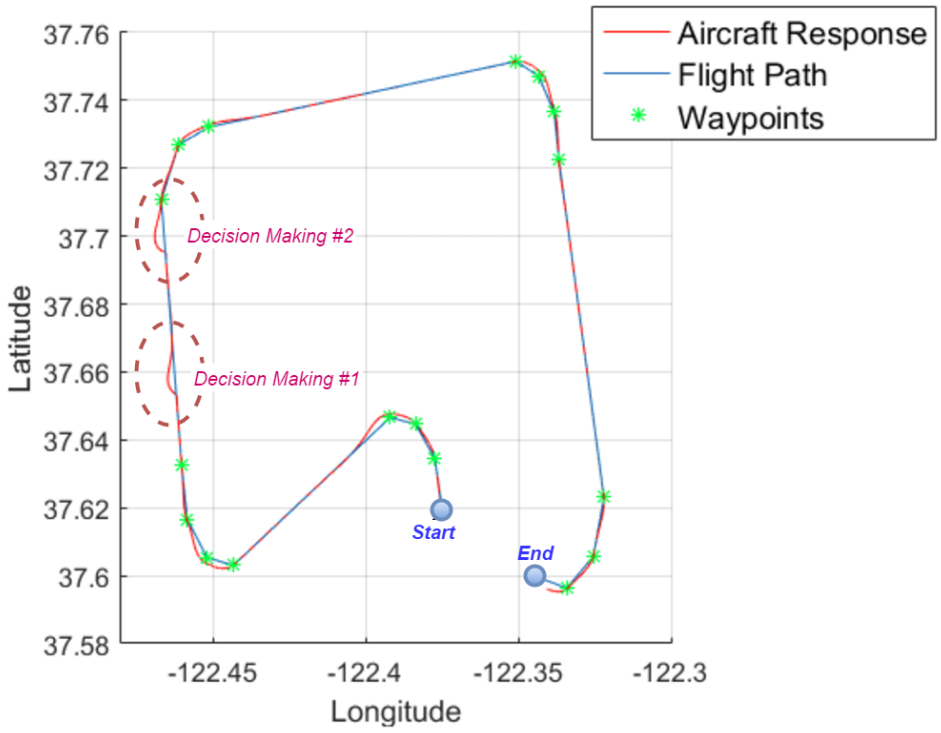


Figure 10.18: Simulated Output of Waypoint Tracking (Case II)

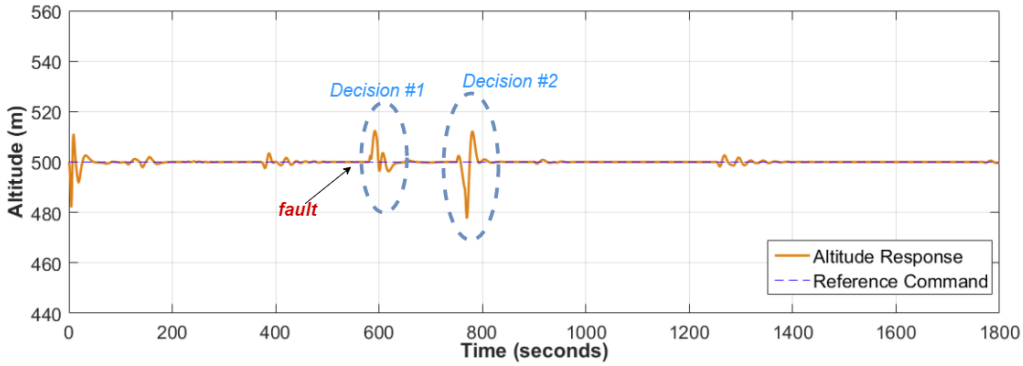


Figure 10.19: Time Histories of Simulated Output States of Altitude Tracking (Case II)

parameters. Then, in a second decision, the agent stimulated a re-identification process, including validation methodology, in order to update the aerodynamic parameters for the inner NDI control-loop in lateral and directional axes. Finally, upon stabilizing and tracking the desired waypoints, the agent continued to monitor the flight system.

Throughout the implementation, the agent reasoning engine performed actions which were dependent on abstractions formed via the interface of the physical engine while abstractions were evaluated by complex processes with the continuous engine. These abstractions were employed with a rational framework to generated appropriated decisions.

10.5 Chapter Summary

This chapter has presented the implementations of a theoretical agent-based-framework for an intelligent UAV control system that use rule-based reasoning with the ability to abstract events. This new framework brings improvements upon traditional adaptive or reconfigurable control schemes currently in aviation by providing more adaptability via on-board dynamical modelling and flight controller tuning under changing dynamics. As MATLAB/SIMULINK computations of the agent were fast enough for the real-time simulation of the flight, the C/C++ implementation of the agent on-board of a UAS will be even faster.

Deficiencies in an initial controller and system degradation can be detected due to accidental damage or ageing. In the future, this agent framework can be augmented to provide an even more capable supervision of UAV's mission and to ultimately result in increased operational safety and longer term preservation of autonomous UAV assets for their operators. sEnglish Publisher was used for demonstration and to facilitate the compilation of the Jason based supervisor-agent with SIMULINK-based executive processes for sensing and simulation for foresight and feedback-control-based action.

Chapter 11

Conclusions and Future Work

11.1 Conclusion

This thesis presents an agent-supervised control-systems of UAV autopilots. The agent consists of rule-based reasoning with the abstractions of events and the ability to act upon decisions. This agent framework for the intelligent autopilot system brings an improvement upon traditional adaptive or reconfigurable control schemes currently used in aviation by providing more adaptability via on-board dynamical modelling and flight controller tuning under changing dynamics. Deficiencies in an initial controller and system degradation can be detected due to accidental damage or ageing.

The primary focus is on the development of key agent skills, including airflow angle estimation, adaptive control, and decision-making methods for control switching to stabilize and track the desired waypoints. Furthermore, the development of real-time abstraction evaluation method results in the ability to receive a perception stream and subsequently filter it to generate the belief base that belongs to the rational agent. This study proposes the development of model validation for NDI control, flight trim condition monitoring, and control performance evaluation.

In terms of integrating agent skills set and agent reasoning development, sEnglish Publisher was used for demonstration and to facilitate the compilation of the Jason based supervisor-agent with Simulink-based executive processes for sensing and simulation for foresight and feedback-control-based action. The agent with

multi-thread execution integrates signal processing including system identification, model based indirect and direct adaptive control, model validation, flight trim condition status monitoring, and control performance evaluation. The Jason+ based BDI agent makes decisions using reasoning by logic to execute actions with requirements of intelligent control schemes.

Furthermore, this intelligent control system is considered to be applied on small and low-cost UAVs that are equipped with a limited number of standard UAV actuators and sensors such as pressure sensor for measuring airspeed and altitude, GPS, IMU, and compass. Therefore, airflow angle estimation is developed instead of airflow angle sensor which requires extensive calibration for good accuracy in practice.

Additionally, OLS in the frequency domain that works in combination with a direct adaptive control strategy has been applied as indirect adaptive learning for the neural network hybrid adaptive control scheme. This hybrid adaptive control strategy can contribute a significant improvement in tracking performance over direct and indirect adaptive control. Moreover, this frequency OLS can be utilised to estimate aircraft aerodynamic parameter for NDI inner loop of flight control system.

A frequency dependent model validation approach is used, which is based on a discrete Fourier transform and a hypothesis test of the normalised spectrum with χ^2 distribution. This algorithm is a key procedure to abstract the continuous information to discrete abstractions for 1) validating aerodynamic stability and control coefficients in the inner loop of the NDI controller, 2) monitoring flight trim condition, and 3) assessing tracking control performance of the flight control system.

11.2 Future Work

In the future, this agent framework can be augmented to provide an even more capable supervision of UAV's mission and ultimately result in increased operational safety and the preservation of the autonomous UAV assets for their operators. For instance, another problem that should be considered about agent framework

is the issue of structural damaged aircraft. There are various relevant variables that are affected from c.g. shifting, mass and inertia change, and aerodynamic characteristics as mentioned in Section 3.2.3. Therefore, in response to this problem, the agent can perform an action of switching to a suitable structure of direct and indirect adaptive control by considering the mentioned effect to provide a significant improvement in the control performance in this case.

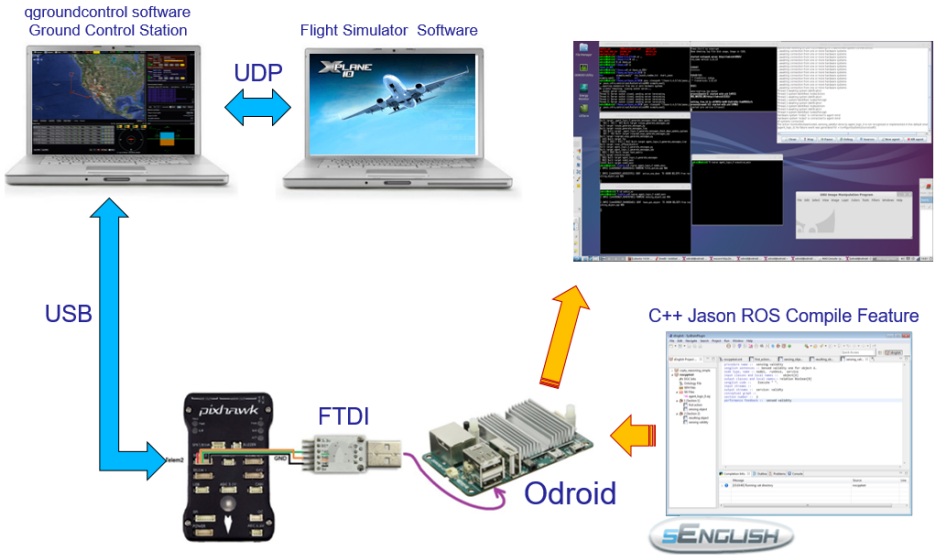
Another extension of the current work is to test the designed intelligent control system in real world experiments. This experiment addresses the issue of verifying a complete intelligent autopilot agent from a practical point of view. The model that is used in our work is based on a small aircraft, such as 2.1 meter-wingspan flying wing and 1.8 meter-wingspan conventional fixed-wing aircraft, built in our autonomous control laboratory as shown in Figure 11.1.

a) Skywalker X8 Flying Wing Aircraft

b) Cirrus SR22 Fixed-Wing Aircraft

Figure 11.1: Aircraft models built on our autonomous control laboratory

The aircraft models are equipped with Pixhawk autopilot for low-level programming that consists of 32bit Cortex M4 core as processor, 3-axis gyroscope, 3-axis accelerometer, magnetometer, barometer, and pressure sensors to collect all flight sensor data. Odroid board that is a high performance single computer board is also equipped on the aircraft for high-level Jason programming. Furthermore, the developed sEnglish document that contains high-level code for agent reasoning in Jason (AgentSpeak) can ultimately compile into C++/ROS for embedded systems such as Linux-based Odroid board. Before real flight experiment, the C++/ROS software and (Pixhawk autopilot and Odroid board) can be verified with X-Plane Flight Simulator where this technique is called "Hardware-in-the-Loop implementation"as illustrated in Figure 11.2.



a) Diagram of Hardware-in-the-loop implementation

b) Hardware-in-the-loop Implementation Experiment

Figure 11.2: Concept and Experiment of Hardware-in-the-loop Implementation

Part IV

Appendices

Appendix A: Flight Equation of Motion with Effect of Mass Change

Considering the translational and rotational flight equation of motion from Eq.(3.5) in Chapter 3.

$$\begin{aligned} \vec{\mathbf{F}} = &\quad m\dot{\bar{v}} + m\dot{\bar{\omega}} \times \Delta r + m\bar{\omega} \times (\bar{v} + \bar{\omega} \times \Delta\bar{r}) \\ \vec{\mathbf{M}} = &\quad \mathbf{I}\dot{\bar{\omega}} + m\Delta\bar{r} \times \dot{\bar{v}} + \bar{\omega} \times \mathbf{I}\bar{\omega} + m\bar{\omega} \times (\Delta\bar{r} \times \bar{v}) \end{aligned} \tag{1}$$

where:

	m	Total mass of the aircraft
$\bar{v} =$	$[u\ v\ w]^T$	Linear velocities decomposed in the body-frame.
$\bar{\omega} =$	$[p\ q\ r]^T$	Angular velocities decomposed in the body-frame. The angular velocities p, q and r are commonly known as *roll*, *pitch* and *yaw* respectively.
$\vec{\mathbf{F}} =$	$[F_x\ F_y\ F_z]^T$	External forces decomposed in the body-frame.
$\vec{\mathbf{M}} =$	$[L\ M\ N]^T$	External momentums decomposed in the body-frame.
$\Delta\bar{r} =$	$[x_{cg}\ y_{cg}\ z_{cg}]^T$	Shifted position of the centre of gravity in the body-frame as shown in Fig. 3.2

$$\mathbf{I} = \begin{bmatrix} I_x & I_{xy} & I_{xz} \\ I_{yx} & I_y & I_{yz} \\ I_{zx} & I_{zy} & I_z \end{bmatrix} \quad \text{Inertia tensor}$$

The aircraft mass and inertia are assumed to undergo a change so that

$$m = m^* + \Delta m \tag{2}$$

$$\mathbf{I} = \mathbf{I}^* + \Delta\mathbf{I} = \begin{bmatrix} I_x^* + \Delta I_x & I_{xy}^* + \Delta I_{xy} & I_{xz}^* + \Delta I_{xz} \\ I_{yx}^* + \Delta I_{yx} & I_y^* + \Delta I_y & I_{yz}^* + \Delta I_{yz} \\ I_{zx}^* + \Delta I_{zx} & I_{zy}^* + \Delta I_{zy} & I_z^* + \Delta I_z \end{bmatrix} \tag{3}$$

where m^* is the original mass of the aircraft, Δm is the negative mass change due to damage, $\mathbf{I}^*$ is the original inertia matrix of the aircraft, and $\Delta\mathbf{I}$ the change in the inertia matrix due to damage.

Therefore, the translational equation of Eq. (1) can be expanded into the force equations as follows:

$$\begin{aligned} F_x &= (m^* + \Delta m)(\dot{u} - vr + wq - x_{cg}(q^2 + r^2) + y_{cg}(pq - \dot{r}) + z_{cg}(pr + \dot{q})) \\ F_y &= (m^* + \Delta m)(\dot{v} - wp + ur - x_{cg}(r^2 + p^2) + y_{cg}(qr - \dot{p}) + z_{cg}(qp + \dot{r})) \\ F_z &= (m^* + \Delta m)(\dot{w} - uq + vp - x_{cg}(p^2 + q^2) + y_{cg}(rp - \dot{q}) + z_{cg}(rq + \dot{p})) \end{aligned} \tag{4}$$

The left hand side of Eq. (4) represents all the external forces applied to the aircraft, respectively. In the dynamical model presented in [31, 66], the external forces vector can be identified as the sum of three components: *aerodynamic* ($\vec{\mathbf{F}}_A$), *propulsion* ($\vec{\mathbf{F}}_P$) and *gravity* ($\vec{\mathbf{F}}_G$). Then considering aerodynamic forces in

each axis as shown:

$$\begin{aligned}
F_{A_x} = \; & (m^* + \Delta m)(\dot{u} - vr + wq - x_{cg}(q^2 + r^2) + y_{cg}(pq - \dot{r}) + z_{cg}(pr + \dot{q}) \\
& + g\sin\theta - \tfrac{F_{P_x}}{m^* + \Delta m}) \\
F_{A_y} = \; & (m^* + \Delta m)(\dot{v} - wp + ur - x_{cg}(r^2 + p^2) + y_{cg}(qr - \dot{p}) + z_{cg}(qp + \dot{r}) \\
& - g\cos\theta\sin\phi - \tfrac{F_{P_y}}{m^* + \Delta m}) \\
F_{A_z} = \; & (m^* + \Delta m)(\dot{w} - uq + vp - x_{cg}(p^2 + q^2) + y_{cg}(rp - \dot{q}) + z_{cg}(rq + \dot{p}) \\
& - g\cos\theta\cos\phi - \tfrac{F_{P_z}}{m^* + \Delta m})
\end{aligned} \tag{5}$$

Rewriting Eq. (5),

$$\begin{aligned}
F_{A_x} = \; & m^*(\dot{u} - vr + wq - x_{cg}(q^2 + r^2) + y_{cg}(pq - \dot{r}) + z_{cg}(pr + \dot{q}) + g\sin\theta) \\
& + \Delta m(\dot{u} - vr + wq - x_{cg}(q^2 + r^2) + y_{cg}(pq - \dot{r}) + z_{cg}(pr + \dot{q}) + g\sin\theta) \\
& - F_{P_x} \\
F_{A_y} = \; & m(\dot{v} - wp + ur - x_{cg}(r^2 + p^2) + y_{cg}(qr - \dot{p}) + z_{cg}(qp + \dot{r}) - g\cos\theta\sin\phi) \\
& + \Delta m(\dot{v} - wp + ur - x_{cg}(r^2 + p^2) + y_{cg}(qr - \dot{p}) + z_{cg}(qp + \dot{r}) - g\cos\theta\sin\phi) \\
& - F_{P_y} \\
F_{A_z} = \; & m^*(\dot{w} - uq + vp - x_{cg}(p^2 + q^2) + y_{cg}(rp - \dot{q}) + z_{cg}(rq + \dot{p}) - g\cos\theta\cos\phi) \\
& + \Delta m(\dot{w} - uq + vp - x_{cg}(p^2 + q^2) + y_{cg}(rp - \dot{q}) + z_{cg}(rq + \dot{p}) - g\cos\theta\cos\phi) \\
& - F_{P_z}
\end{aligned} \tag{6}$$

Alternatively, rewriting Eq. (6) into simplify form.

$$\begin{aligned}
F^*_{A_x} + \Delta F_{A_x} = \; & m^*(\dot{u} - vr + wq + g\sin\theta) + \Delta F_{M_x} - F_{P_x} \\
F^*_{A_y} + \Delta F_{A_y} = \; & m^*(\dot{v} - wp + ur - g\cos\theta\sin\phi) + \Delta F_{M_y} - F_{P_y} \\
F^*_{A_z} + \Delta F_{A_z} = \; & m^*(\dot{w} - uq + vp - g\cos\theta\cos\phi) + \Delta F_{M_z} - F_{P_z}
\end{aligned} \tag{7}$$

Where

$$\begin{aligned}
\Delta F_{M_x} = \; & -m^* x_{cg} q^2 - m^* x_{cg} r^2 + m^* y_{cg} pq - m^* y_{cg} \dot{r} + m^* z_{cg} pr + m^* z_{cg} \dot{q} \\
& + \Delta m \dot{u} - \Delta m v r + \Delta m w q - \Delta m x_{cg} q^2 - \Delta m x_{cg} r^2 + \Delta m y_{cg} pq - \Delta m y_{cg} \dot{r} \\
& + \Delta m z_{cg} pr + m z_{cg} \dot{q} + \Delta m g \sin\theta \\
\Delta F_{M_y} = \; & -m^* x_{cg} r^2 - m^* x_{cg} p^2 + m^* y_{cg} qr - m^* y_{cg} \dot{p} + m^* z_{cg} qp + m^* z_{cg} \dot{r} \\
& + \Delta m \dot{v} - \Delta m w p + \Delta m u r - \Delta m x_{cg} r^2 - \Delta m x_{cg} p^2 + \Delta m y_{cg} qr - \Delta m y_{cg} \dot{p} \\
& + \Delta m z_{cg} qp + \Delta m z_{cg} \dot{r} - \Delta m g \cos\theta \sin\phi \\
\Delta F_{M_z} = \; & -m^* x_{cg} p^2 - m^* x_{cg} q^2 + m^* y_{cg} rp - m^* y_{cg} \dot{q} + m^* z_{cg} rq + m^* z_{cg} \dot{p} \\
& + \Delta m \dot{w} - \Delta m u q + \Delta m v p - \Delta m x_{cg} p^2 - \Delta m x_{cg} q^2 + \Delta m y_{cg} rp - \Delta m y_{cg} \dot{q} \\
& + \Delta m z_{cg} rq + \Delta m z_{cg} \dot{p} - \Delta m g \cos\theta \cos\phi
\end{aligned} \tag{8}$$

From the relative equation in Eq. (3.20), Eq. (8) can be rearranged into:

$$\begin{aligned}
\Delta F_{M_x} = \; & \Delta m a_x - (m^* + \Delta m) x_{cg} q^2 - (m^* + \Delta m) x_{cg} r^2 + (m^* + \Delta m) y_{cg} pq \\
& - (m^* + \Delta m) y_{cg} \dot{r} + (m^* + \Delta m) z_{cg} pr + (m^* + \Delta m) z_{cg} \dot{q} \\
= \; & f_{M_x}(a_x, \dot{q}, \dot{r}, q^2, r^2, pq, pr) \\
\Delta F_{M_y} = \; & \Delta m a_y - (m^* + \Delta m) x_{cg} r^2 - (m^* + \Delta m) x_{cg} p^2 + (m^* + \Delta m) y_{cg} qr \\
& - (m^* + \Delta m) y_{cg} \dot{p} + (m^* + \Delta m) z_{cg} qp + (m^* + \Delta m) z_{cg} \dot{r} \\
= \; & f_{M_y}(a_y, \dot{p}, \dot{r}, p^2, r^2, qp, qr) \\
\Delta F_{M_z} = \; & \Delta m a_z - (m^* + \Delta m) x_{cg} p^2 - (m^* + \Delta m) x_{cg} q^2 + (m^* + \Delta m) y_{cg} rp \\
& - (m^* + \Delta m) y_{cg} \dot{q} + (m^* + \Delta m) z_{cg} rq + (m^* + \Delta m) z_{cg} \dot{p} \\
= \; & f_{M_z}(a_z, \dot{p}, \dot{q}, p^2, q^2, rp, rq)
\end{aligned} \tag{9}$$

Therefore, Eq. (7) can be written:

$$\begin{aligned}
m^*(\dot{u} - vr + wq + g\sin\theta) - F^*_{A_x} - F_{P_x} = \; & \Delta F_{A_x} - \Delta F_{M_x} \\
m^*(\dot{v} - wp + ur - g\cos\theta\sin\phi) - F^*_{A_y} - F_{P_y} = \; & \Delta F_{A_y} - \Delta F_{M_y} \\
m^*(\dot{w} - uq + vp - g\cos\theta\cos\phi) - F^*_{A_z} - F_{P_z} = \; & \Delta F_{A_z} - \Delta F_{M_z}
\end{aligned} \tag{10}$$

Can be rearranged in matrix form:

$$m^* \dot{\bar{v}} + m^* \bar{\omega} \times \bar{v} - \vec{\mathbf{F}}^*_G - \vec{\mathbf{F}}^*_A - \vec{\mathbf{F}}^*_P = \; \Delta \vec{\mathbf{F}}_A - \Delta \vec{\mathbf{F}}_M \tag{11}$$

where

$$\begin{aligned}\Delta\vec{\mathbf{F}}_A = & \underbrace{f_{F_A}(\alpha, \beta, \frac{pb}{2V}, \frac{q\bar{c}}{2V}, \frac{rb}{2V}, ...)}_{\substack{\text{aerodynamic force functions} \\ \text{discussed in Section 3.2.2}}} \\ \Delta\vec{\mathbf{F}}_M = & \underbrace{f_{F_M}(a_x, a_y, a_z, \dot{p}, \dot{q}, \dot{r}, p^2, q^2, r^2, pq, pr, qp, qr)}_{\substack{\text{aerodynamic force functions} \\ \text{discussed in Equation 9}}}\end{aligned} \tag{12}$$

Next, the rotational equation of Eq. (1) can be expanded into the moment equations as follows:

$$\begin{aligned}L = \; & I_x^*\dot{p} + (I_z^* - I_y^*)qr - (\dot{r} + pq)I_{xz}^* + (r^2 - q^2)I_{yz}^* + (pr - \dot{q})I_{xy}^* \\ & + \Delta I_x\dot{p} + (\Delta I_z - \Delta I_y)qr - (\dot{r} + pq)\Delta I_{xz} + (r^2 - q^2)\Delta I_{yz} + (pr - \dot{q})\Delta I_{xy} \\ & + m^*[x_{cg}(vq + wr) + y_{cg}(\dot{w} - uq) - z_{cg}(\dot{v} + ur)] \\ & + \Delta m[x_{cg}(vq + wr) + y_{cg}(\dot{w} - uq) - z_{cg}(\dot{v} + ur)] \\ M = \; & I_y^*\dot{q} + (I_x^* - I_z^*)rp - (\dot{p} + qr)I_{xy}^* + (p^2 - r^2)I_{zx}^* + (qp - \dot{r})I_{yz}^* \\ & + \Delta I_y\dot{q} + (\Delta I_x - \Delta I_z)rp - (\dot{p} + qr)\Delta I_{xy} + (p^2 - r^2)\Delta I_{zx} + (qp - \dot{r})\Delta I_{yz} \\ & + m^*[-x_{cg}(\dot{w} + vp) + y_{cg}(up + wr) + z_{cg}(\dot{u} - vr)] \\ & + \Delta m[-x_{cg}(\dot{w} + vp) + y_{cg}(up + wr) + z_{cg}(\dot{u} - vr)] \\ N = \; & I_z^*\dot{r} + (I_y^* - I_x^*)pq - (\dot{q} + rp)I_{yz}^* + (q^2 - p^2)I_{xy}^* + (rq - \dot{p})I_{zx}^* \\ & + \Delta I_z\dot{r} + (\Delta I_y - \Delta I_x)pq - (\dot{q} + rp)\Delta I_{yz} + (q^2 - p^2)\Delta I_{xy} + (rq - \dot{p})\Delta I_{zx} \\ & + m^*[x_{cg}(\dot{v} - wp) - y_{cg}(\dot{u} + wq) + z_{cg}(up + qv)] \\ & + \Delta m[x_{cg}(\dot{v} - wp) - y_{cg}(\dot{u} + wq) + z_{cg}(up + qv)]\end{aligned} \tag{13}$$

Before move on, considering the left hand side of Eq. (1) represents all the external moments applied to the aircraft, respectively. In the dynamical model presented in [31, 66], the external moments vector can be identified as the sum of three components: *aerodynamic* ($\vec{\mathbf{M}}_A$), *propulsion* ($\vec{\mathbf{M}}_P$) and *gravity* ($\vec{\mathbf{M}}_G$). Then consider the left hand side term only aerodynamic moment term:

$$\vec{\mathbf{M}}_A = \; \mathbf{I}\dot{\bar{\omega}} + m\Delta\bar{r} \times \dot{\bar{v}} + \bar{\omega} \times \mathbf{I}\bar{\omega} + m\bar{\omega} \times (\Delta\bar{r} \times \bar{v}) - \Delta r \times \vec{\mathbf{F}}_G \tag{14}$$

Then, the rotational equation of Eq. (14) can be expanded into:

$$
\begin{aligned}
L_A = \;& I_x^*\dot{p} + (I_z^* - I_y^*)qr - (\dot{r} + pq)I_{xz}^* + (r^2 - q^2)I_{yz}^* + (pr - \dot{q})I_{xy}^* \\
& + \Delta I_x\dot{p} + (\Delta I_z - \Delta I_y)qr - (\dot{r} + pq)\Delta I_{xz} + (r^2 - q^2)\Delta I_{yz} + (pr - \dot{q})\Delta I_{xy} \\
& + m^*[x_{cg}(vq + wr) + y_{cg}(\dot{w} - uq - g\cos\phi\cos\theta) - z_{cg}(\dot{v} + ur - g\sin\phi\cos\theta)] \\
& + \Delta m[x_{cg}(vq + wr) + y_{cg}(\dot{w} - uq - g\cos\phi\cos\theta) - z_{cg}(\dot{v} + ur - g\sin\phi\cos\theta)] \\
M_A = \;& I_y^*\dot{q} + (I_x^* - I_z^*)rp - (\dot{p} + qr)I_{xy}^* + (p^2 - r^2)I_{zx}^* + (qp - \dot{r})I_{yz}^* \\
& + \Delta I_y\dot{q} + (\Delta I_x - \Delta I_z)rp - (\dot{p} + qr)\Delta I_{xy} + (p^2 - r^2)\Delta I_{zx} + (qp - \dot{r})\Delta I_{yz} \\
& + m^*[-x_{cg}(\dot{w} + vp - g\cos\phi\cos\theta) + y_{cg}(up + wr) + z_{cg}(\dot{u} - vr + g\sin\theta)] \\
& + \Delta m[-x_{cg}(\dot{w} + vp - g\cos\phi\cos\theta) + y_{cg}(up + wr) + z_{cg}(\dot{u} - vr + g\sin\theta)] \\
N_A = \;& I_z^*\dot{r} + (I_y^* - I_x^*)pq - (\dot{q} + rp)I_{yz}^* + (q^2 - p^2)I_{xy}^* + (rq - \dot{p})I_{zx}^* \\
& + \Delta I_z\dot{r} + (\Delta I_y - \Delta I_x)pq - (\dot{q} + rp)\Delta I_{yz} + (q^2 - p^2)\Delta I_{xy} + (rq - \dot{p})\Delta I_{zx} \\
& + m^*[x_{cg}(\dot{v} - wp - g\sin\phi\cos\theta) - y_{cg}(\dot{u} + wq + g\sin\theta) + z_{cg}(up + qv)] \\
& + \Delta m[x_{cg}(\dot{v} - wp - g\sin\phi\cos\theta) - y_{cg}(\dot{u} + wq + g\sin\theta) + z_{cg}(up + qv)]
\end{aligned}
\tag{15}
$$

Rewriting Eq. (15) as follows:

$$
\begin{aligned}
L_A = \;& I_x^*\dot{p} + (I_z^* - I_y^*)qr - (\dot{r} + pq)I_{xz}^* + (r^2 - q^2)I_{yz}^* + (pr - \dot{q})I_{xy}^* \\
& + \Delta I_x\dot{p} + (\Delta I_z - \Delta I_y)qr - (\dot{r} + pq)\Delta I_{xz} + (r^2 - q^2)\Delta I_{yz} + (pr - \dot{q})\Delta I_{xy} \\
& + (m^* + \Delta m)x_{cg}vq - (m^* + \Delta m)y_{cg}vp + (m^* + \Delta m)x_{cg}wr \\
& - (m^* + \Delta m)z_{cg}wp + (m^* + \Delta m)y_{cg}a_z - (m^* + \Delta m)z_{cg}a_y \\
M_A = \;& I_y^*\dot{q} + (I_x^* - I_z^*)rp - (\dot{p} + qr)I_{xy}^* + (p^2 - r^2)I_{zx}^* + (qp - \dot{r})I_{yz}^* \\
& + \Delta I_y\dot{q} + (\Delta I_x - \Delta I_z)rp - (\dot{p} + qr)\Delta I_{xy} + (p^2 - r^2)\Delta I_{zx} + (qp - \dot{r})\Delta I_{yz} \\
& - (m^* + \Delta m)x_{cg}a_z + (m^* + \Delta m)y_{cg}up - (m^* + \Delta m)x_{cg}uq \\
& + (m^* + \Delta m)y_{cg}wr - (m^* + \Delta m)z_{cg}wq + (m^* + \Delta m)z_{cg}a_x \\
N_A = \;& I_z^*\dot{r} + (I_y^* - I_x^*)pq - (\dot{q} + rp)I_{yz}^* + (q^2 - p^2)I_{xy}^* + (rq - \dot{p})I_{zx}^* \\
& + \Delta I_z\dot{r} + (\Delta I_y - \Delta I_x)pq - (\dot{q} + rp)\Delta I_{yz} + (q^2 - p^2)\Delta I_{xy} + (rq - \dot{p})\Delta I_{zx} \\
& + (m^* + \Delta m)x_{cg}a_y - (m^* + \Delta m)y_{cg}a_x + (m^* + \Delta m)z_{cg}up \\
& + (m^* + \Delta m)z_{cg}qv - (m^* + \Delta m)x_{cg}ur - (m^* + \Delta m)y_{cg}rv
\end{aligned}
\tag{16}
$$

Then, Eq. (16) can be simplified into:

$$\begin{aligned} L_A = &\ I_x^*\dot{p} + (I_z^* - I_y^*)qr - (\dot{r} + pq)I_{xz}^* + (r^2 - q^2)I_{yz}^* + (pr - \dot{q})I_{xy}^* + \Delta L_M \\ M_A = &\ I_y^*\dot{q} + (I_x^* - I_z^*)rp - (\dot{p} + qr)I_{xy}^* + (p^2 - r^2)I_{zx}^* + (qp - \dot{r})I_{yz}^* + \Delta M_M \\ N_A = &\ I_z^*\dot{r} + (I_y^* - I_x^*)pq - (\dot{q} + rp)I_{yz}^* + (q^2 - p^2)I_{xy}^* + (rq - \dot{p})I_{zx}^* + \Delta N_M \end{aligned} \tag{17}$$

where

$$\begin{aligned} \Delta L_M = &\ \Delta I_x\dot{p} + (\Delta I_z - \Delta I_y)qr - (\dot{r} + pq)\Delta I_{xz} + (r^2 - q^2)\Delta I_{yz} + (pr - \dot{q})\Delta I_{xy} \\ &+(m^* + \Delta m)x_{cg}vq - (m^* + \Delta m)y_{cg}vp + (m^* + \Delta m)x_{cg}wr \\ &-(m^* + \Delta m)z_{cg}wp + (m^* + \Delta m)y_{cg}a_z - (m^* + \Delta m)z_{cg}a_y \\ = &\ f_{M_L}(\dot{p}, \dot{q}, \dot{r}, q^2, r^2, pq, pr, qr, vq, vp, wr, wp, a_y, a_z) \end{aligned} \tag{18}$$

$$\begin{aligned} \Delta M_M = &\ \Delta I_y\dot{q} + (\Delta I_x - \Delta I_z)rp - (\dot{p} + qr)\Delta I_{xy} + (p^2 - r^2)\Delta I_{zx} + (qp - \dot{r})\Delta I_{yz} \\ &-(m^* + \Delta m)x_{cg}a_z + (m^* + \Delta m)y_{cg}up - (m^* + \Delta m)x_{cg}uq \\ &+(m^* + \Delta m)y_{cg}wr - (m^* + \Delta m)z_{cg}wq + (m^* + \Delta m)z_{cg}a_x \\ = &\ f_{M_M}(\dot{p}, \dot{q}, \dot{r}, p^2, r^2, rp, qr, qp, up, uq, wr, wq, a_x, a_z) \end{aligned} \tag{19}$$

$$\begin{aligned} \Delta N_M = &\ \Delta I_z\dot{r} + (\Delta I_y - \Delta I_x)pq - (\dot{q} + rp)\Delta I_{yz} + (q^2 - p^2)\Delta I_{xy} + (rq - \dot{p})\Delta I_{zx} \\ &+(m^* + \Delta m)x_{cg}a_y - (m^* + \Delta m)y_{cg}a_x + (m^* + \Delta m)z_{cg}up \\ &+(m^* + \Delta m)z_{cg}qv - (m^* + \Delta m)x_{cg}ur - (m^* + \Delta m)y_{cg}rv \\ = &\ f_{M_N}(\dot{p}, \dot{q}, \dot{r}, p^2, q^2, pq, rp, rq, up, qv, ur, rv, a_y, a_z) \end{aligned} \tag{20}$$

In addition, Eq. 17 can be rewritten to:

$$\begin{aligned} I_x^*\dot{p} + (I_z^* - I_y^*)qr - (\dot{r} + pq)I_{xz}^* + (r^2 - q^2)I_{yz}^* + (pr - \dot{q})I_{xy}^* - L_A^* = &\ \Delta L_A - \Delta L_M \\ I_y^*\dot{q} + (I_x^* - I_z^*)rp - (\dot{p} + qr)I_{xy}^* + (p^2 - r^2)I_{zx}^* + (qp - \dot{r})I_{yz}^* - M_A^* = &\ \Delta M_A - \Delta M_M \\ I_z^*\dot{r} + (I_y^* - I_x^*)pq - (\dot{q} + rp)I_{yz}^* + (q^2 - p^2)I_{xy}^* + (rq - \dot{p})I_{zx}^* - N_A^* = &\ \Delta N_A - \Delta N_M \end{aligned} \tag{21}$$

Therefore, Eq. (21) Can be rearranged in matrix form:

$$\mathbf{I}^*\dot{\bar{\omega}} + \bar{\omega} \times \mathbf{I}^*\bar{\omega} + \vec{\mathbf{M}}_A^* = \quad \Delta\vec{\mathbf{M}}_A - \Delta\vec{\mathbf{M}}_M \tag{22}$$

where

$$\begin{aligned} \Delta\vec{\mathbf{M}}_A = & \underbrace{f_{M_A}(\alpha, \beta, \frac{pb}{2V}, \frac{q\bar{c}}{2V}, \frac{rb}{2V}, ...)}_{\substack{\text{aerodynamic moment functions} \\ \text{discussed in Section 3.2.2}}} \\ \Delta\vec{\mathbf{M}}_M = & \underbrace{f_{M_M}(a_x, a_y, a_z, \dot{p}, \dot{q}, \dot{r}, p^2, q^2, r^2, pq, pr, qp, qr, up, uq, ur, vp, vq, vr, wp, wq, wr)}_{\substack{\text{aerodynamic moment functions} \\ \text{discussed in Equations 18, 19, and 20}}} \end{aligned} \tag{23}$$

Appendix B: Air Flow Angle Reconstruction

From the definition of airspeed, angle of attack and side-slipe angle:

$$\begin{aligned} V &= \sqrt{u^2 + v^2 + w^2} \\ \alpha &= \tan^{-1} \frac{w}{u} \\ \beta &= \sin^{-1} \frac{v}{V} \end{aligned} \tag{24}$$

Then, differentiating Eqs 24 with respect to time gives

$$\begin{aligned} \dot{V} &= \frac{1}{V}(u\dot{u} + v\dot{v} + w\dot{w}) \\ \dot{\alpha} &= \left(\frac{u\dot{w} - w\dot{u}}{u^2 + w^2}\right) \\ \dot{\beta} &= \frac{(u^2 + w^2)\dot{v} - v(u\dot{u} + w\dot{w})}{v^2\sqrt{u^2 + w^2}} \end{aligned} \tag{25}$$

The translational acceleration in Eq. (3.20) can be rearranged to

$$\begin{aligned} \dot{u} = &\ rv - qw - g\sin\theta + a_x \\ \dot{v} = &\ pw - ru + g\cos\theta\sin\phi + a_y \\ \dot{w} = &\ qu - pv + g\cos\theta\cos\phi + a_z \end{aligned} \tag{26}$$

The body-axis components are related to V, α, and β in wind axes by

$$\begin{aligned} u &= V\cos\alpha\sin\beta \\ v &= V\sin\beta \\ w &= V\sin\alpha\cos\beta \end{aligned} \tag{27}$$

Substituting in Eqs. 25 for $\dot{u}$, $\dot{v}$, and $\dot{w}$ from Eqs. 26 and for u, v, and w from Eqs. 27

$$\begin{aligned} \dot{V} = \;& g(\cos\phi\cos\theta\sin\alpha\cos\beta + \sin\phi\cos\theta\sin\beta - \sin\theta\cos\alpha\cos\beta) \\ & + a_x\cos\alpha\cos\beta + a_y\sin\beta + a_z\sin\alpha\cos\beta \\ \dot{\alpha} = \;& q - \tan\beta(p\cos\alpha + r\sin\alpha) + \frac{g}{V\cos\beta}(\cos\phi\cos\theta\cos\alpha + \sin\theta\sin\alpha) \\ & - a_x\frac{\sin\alpha}{V\cos\beta} + a_z\frac{\cos\alpha}{V\cos\beta} \\ \dot{\beta} = \;& p\sin\alpha - r\cos\alpha + \frac{g\sin\beta}{V}(\cos\alpha\sin\theta - \sin\alpha\cos\phi\cos\theta) \\ & + \frac{g}{V}\cos\beta\sin\phi\cos\theta - a_x\frac{\cos\alpha\sin\beta}{V} + a_y\frac{\cos\beta}{V} - a_z\frac{\sin\alpha\sin\beta}{V} \end{aligned} \tag{28}$$

Appendix C: Aircraft Geometry, Mass Properties, and Aerodynamic Characteristics

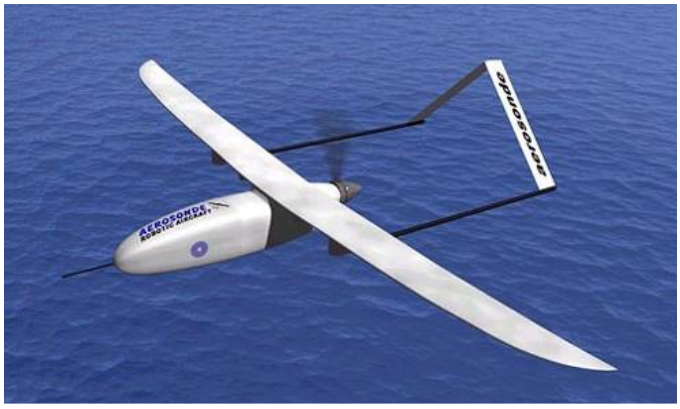


Figure 3: Aerosonde UAV.

Table 2: Aircraft geometry and mass properties

	Aerosonde UAV	NASA Twin Otter Aircraft
mean chord $\bar{c}$, m	0.18994	1.9812
wing span b, m	2.8956	19.812
wing area S, m^2	0.55	39.252
mass, kgs	13.5	4961.93
I_x, kg-m^2	0.8244	28,336.60
I_y, kg-m^2	1.135	32,893.50
I_z, kg-m^2	1.759	52,156.96
I_xz, kg-m^2	0.1204	1,529.36

Table 3: Performance Comparison of Parameter Estimation with Aerosonde Simulation in Longitudinal Dynamic.

Parameter θ	Aerosonde Model in Aerosim Blockset	Least Square in Time Domain Method	Least Square in Frequency Domain Method	OLS in Time Domain Method	OLS in Frequency Domain Method
C_{m_o}	0.1350	0.1159	0.1234	0.1159	0.1061
C_{m_α}	-2.7397	-3.0769	-3.3070	-3.0769	-2.91
$C_{m_{\dot{\alpha}}}$	-10.3796	-7.8875	-16.2841	-7.8875	-9.1849
C_{m_q}	-38.2067	-19.7372	-23.4229	-19.7372	-19.7759
$C_{m_{\delta_e}}$	0.9918	-0.8072	-0.8810	-0.8072	-0.7984

Table 4: Parameter estimates for NASA Twin Otter aircraft measurement data for lateral maneuver

Parameter θ	Measurement α, β Real-Time RFT Method	Estimated α, β Real-Time RFT Method	Publication [Klein:06] Output-error Method
C_{Y_β}	-0.8819	-0.89	-0.866
C_{Y_r}	6.649	-0.777	0.931
$C_{Y_{\delta r}}$	0.3425	0.346	0.375
C_{l_β}	-0.1087	-0.1096	-0.119
C_{l_p}	-0.5765	-0.5776	-0.584
C_{l_r}	0.1736	0.1498	0.188
$C_{l_{\delta a}}$	-0.2283	-0.2289	-0.228
$C_{l_{\delta r}}$	-0.0186	0.01904	0.0384
C_{n_β}	0.08831	0.08932	0.0865
C_{n_p}	-0.05613	-0.05422	-0.0639
C_{n_r}	-0.0205	-0.1853	-0.192
$C_{n_{\delta a}}$	0.000229	0.0009684	-0.00273
$C_{n_{\delta r}}$	-0.1373	-0.1373	-0.136

Table 5: Parameter Estimates for Simulated Data of Aerosonde Model

Parameter θ	Simulated α, β Real-Time Frequency Domain Final Value of Ideal Signal	Estimated α, β Real-Time Frequency Domain Estimated $\hat{\theta}\pm\hat{\sigma}$
C_{L_0}	0.2999	0.300±0.008
C_{L_α}	7.124	6.911±0.194
C_{L_q}	9.026	8.520±1.727
$C_{L_\delta e}$	0.1342	0.073±0.050
C_{D_0}	0.03931	0.043±0.0009
C_{D_α}	0.2606	0.158±0.0202
C_{D_q}	11.16	10.057±0.2638
$C_{D_\delta e}$	0.06779	0.0895±0.0067
C_{Y_0}	-0.0001229	-0.0006±0.0018
C_{Y_β}	-1.13	-1.0742±0.0329
C_{Y_p}	-0.02373	0.2280±0.1009
C_{Y_r}	0.005276	-0.220±0.0242
$C_{Y_\delta a}$	-0.1011	-0.0226±0.0355
$C_{Y_\delta r}$	0.2435	0.252±0.0034
C_{l_0}	-0.001003	-0.00107±0.0002
C_{l_β}	-0.1377	-0.1324±0.0036
C_{l_p}	-0.4824	-0.4563±0.0109
C_{l_r}	0.2362	0.2102±0.0033
$C_{l_\delta a}$	-0.1617	-0.1537±0.0038
$C_{l_\delta r}$	-0.001492	-0.000298±0.00037262
C_{m_0}	0.0083	0.08627±0.0002
C_{m_α}	-2.3764	-2.385±0.0045
C_{m_q}	-26.2375	-26.43±0.0465
$C_{m_\delta e}$	-0.6513	-0.6714±0.0013
C_{n_0}	1.309e-5	0.000041±0.000114
C_{n_β}	0.06973	0.066±0.0023
C_{n_p}	-0.08816	-0.1047±0.0072
C_{n_r}	-0.07067	-0.05648±0.0013
$C_{n_\delta a}$	0.004077	-0.0011±0.0025
$C_{n_\delta r}$	-0.06834	-0.068865±0.0002350

Appendix D: Source Code in Format of sEnglish Sentences for Agent Decision

```
INITIAL BELIEFS AND GOALS
-Applied dac auto tunning in longitudinal.
-Applied minor auto tuning in longitudinal.
-Applied dac auto tunning in lateral.
-Applied minor auto tuning in lateral.
-Applied dac auto tunning in directional.
-Applied minor auto tuning in directional.
+Applying hac control.
+First flight.

INITIAL ACTIONS
Apply Force and set the initial parameter of aircraft for
   agent.
Calculate Pathpoint using dubin flight path planning from
   Aircraft_position, Aircraft_heading, Target_position,
   Finaltarget_direction and Option.
```

```
Calculate Input for flight control system using line of
   sight guidance algorithm from P, V and Required_altitude.
Apply Control_surface_input using hybrid adaptive control
   for flight control loop from Input_command, State_ndi and
    State_ndi_m.
if (~first_flight) {
Check Update_state_of_model in term of model validation for
   nonlinear dynamic inversion control in frequency domain
   from State_ndi.
};
Receive Perceptions from environment and other processor
   like agent adminssion.
Monitor aircraft manoeuvre condition or status from Altitude
    and send back output of NoUtilityFucn.

PERCEPTION PROCESSES

Monitor the following Booleans  :

Monitor the following objects  :

REASONING
If ~^[First flight.] then +^[Applied revalidation.].
If ~^[Compensate parameter.] then +^[Switch off compensate
   parameter.]

EXECUTABLE PLANS
If +^[First flight.] under the condition of ^[True.]
then do the following: +^[Applying minor auto tuning in
   lateral.]
```

```
+^[Applying minor auto tuning in directional.]
+^[Applying minor auto tuning in longitudinal.]
[Calculate Theta that is the compensator makeing use and
   compensate of model-based adaptive control routines for
   first flight from State_ndi.]
[Stopping monitor aircraft manoeuvre condition or status
   from Altitude and send back output of NoUtilityFucn.]
[Set the terminated time of thirty second and send output
   back with NoUtilityFunc.]
[Start to record the clock timer and send NoUtilityFunc back
   .]
if (applying_ndi_control) {
[Stopping apply Control_surface_input using nonlinear
   dynamic inversion control for flight control loop from
   Input_command, State_ndi and State_ndi_m.]
~^[Applying ndi control.]
[Set the initial parameter for hybrid adaptive control from
   State_ndi and hold Control_surface_input.]
[Apply Control_surface_input using hybrid adaptive control
   for flight control loop from Input_command, State_ndi and
    State_ndi_m.]
+^[Applying hac control.]
+^[Applying minor auto tuning with hac for first flight.]
};
if (applying_hac_control) {
+^[Apply hac control.]
+^[Applying minor auto tuning with hac for first flight.]
};
~^[First flight.].
```

```
If +^[Termination of timer.] under the condition of ^[
   Applying minor auto tuning with hac for first flight.]
then do the following:
[Stopping calculate Theta that is the compensator makeing
   use and compensate of model-based adaptive control
   routines for first flight from State_ndi.]
[Stopping start to record the clock timer and send
   NoUtilityFunc back.]
[Check Update_state_of_model in term of model validation for
    nonlinear dynamic inversion control in frequency domain
   from State_ndi.]
[Monitor aircraft manoeuvre condition or status from
   Altitude and send back output of NoUtilityFucn.]
~^[Applying minor auto tuning in lateral.]
~^[Applying minor auto tuning in directional.]
~^[Applying minor auto tuning in longitudinal.]
~^[Applying minor auto tuning with hac for first flight.]
+^[Apply hac control.]
+^[Applied revalidation.]
+^[Applied minor auto tuning in lateral.]
+^[Applied minor auto tuning in longitudinal.]
+^[Applied minor auto tuning in directional.].

If +^[Bad control performance.] under the condition of ^[
   Auto tuning.] & not ^[Applying major auto tuning in
   longitudinal.] & not ^[Applying minor auto tuning in
   longitudinal] & not ^[Applying dac auto tuning in
   longitudinal.] & not ^[Applying major auto tuning in
   lateral.] & not ^[Applying minor auto tuning in lateral.]
    & not ^[Applying dac auto tuning in lateral.] & not ^[
```

```
   Applying major auto tuning in directional.] & not ^[
   Applying minor auto tuning in directional.] & not ^[
   Applying dac auto tuning in directional.]
then do the following:
+^[Applying minor auto tuning in lateral.]
+^[Applying minor auto tuning in directional.]
+^[Applying minor auto tuning in longitudinal.]
if (requiring_revalidation_plus_delay) {
~^[Requiring revalidation plus delay.]
[Check Update_state_of_agent in term of model validation for
    nonlinear dynamic inversion control in frequency domain
   with delayed time from State_ndi.]
[Set speed command back to required airspeed and send
   NoTriggerFunc.]
+^[Applied revalidation plus delay.]
};
if (applying_ndi_control) {
-^[Applying ndi control.]
[Stopping apply Control_surface_input using nonlinear
   dynamic inversion control for flight control loop from
   Input_command, State_ndi and State_ndi_m.]
[Set the initial parameter for direct adaptive control from
   State_ndi and hold Control_surface_input.]
[Apply Control_surface_input using hybrid adaptive control
   for flight control loop from Input_command, State_ndi and
    State_ndi_m.]
+^[Applying hac control.]
};
if (applying_hac_control) {
```

```
[Reset adaptive gain adjustment for direct adaptive control
   and send back the output of NoTriggerFunc.]
[Set the initial parameter for control compensation using
   frequency least square algorithm in bad control
   performance condition and send Theta output back.]
[Calculate Theta that is the compensator making use and
   compensate of model-based adaptive control routines in
   bad control performance from State_ndi.]
[Stopping monitor aircraft manoeuvre condition or status
   from Altitude and send back output of NoUtilityFucn.]
[Set the terminated time of thirty second and send output
   back with NoUtilityFunc.]
[Start to record the clock timer and send NoUtilityFunc back
   .]
+^[Apply hac control.]
+^[Applying minor auto tuning with hac for bad control
   performance.]
};
+^[Applied hac with compensation method in bad control
   performance.].

If +^[Termination of timer.] under the condition of ^[
   Applying minor auto tuning with hac for bad control
   performance.]
then do the following:
[Stopping calculate Theta that is the compensator making use
    and compensate of model-based adaptive control routines
   in bad control performance from State_ndi.]
[Stopping start to record the clock timer and send
   NoUtilityFunc back.]
```

```
[Monitor aircraft manoeuvre condition or status from
   Altitude and send back output of NoUtilityFucn.]
~^[Applying minor auto tuning in lateral.]
~^[Applying minor auto tuning in directional.]
~^[Applying minor auto tuning in longitudinal.]
~^[Applying minor auto tuning with hac for bad control
   performance.]
+^[Applied minor auto tuning in lateral.]
+^[Applied minor auto tuning in longitudinal.]
+^[Applied minor auto tuning in directional.]
+^[Requiring revalidation.].

If +^[Control via hac.] under the condition of ~^[Control
   via ndi.] & ~^[Auto tuning]
then do the following:
if (applying_ndi_control) {
[Stopping apply Control_surface_input using nonlinear
   dynamic inversion control for flight control loop from
   Input_command, State_ndi and State_ndi_m.]
~^[Applying ndi control.]
};
if (applying_trim_force) {
[Stopping hold the Control_surface_input of aircraft with a
   previous constant value.]
-^[Applying trim force.]
};
[Set the initial parameter for hybrid adaptive control from
   State_ndi and hold Control_surface_input.]
[Apply Control_surface_input using hybrid adaptive control
   for flight control loop from Input_command, State_ndi and
```

```
    State_ndi_m.]
+^[Applying hac control.].

If +^[Compensate parameter.] under the condition of ~^[Auto
   tuning]
then do the following:
[Set the initial parameter for control compensation using
   frequency orthogonal least square algorithm and send
   Theta output back.]
[Calculate Theta that is the compensator making use and
   compensate of model-based adaptive control routines using
    frequency orthogonal least sqaure from State_ndi.].

If +^[Switch off compensate parameter.] under the condition
   of ~^[Auto tuning]
then do the following:
[Stopping calculate Theta that is the compensator making use
    and compensate of model-based adaptive control routines
   using frequency orthogonal least sqaure from State_ndi.].

If +^[Control via ndi.] under the condition of not ^[
   Applying ndi control.] & ~^[Auto tuning]
then do the following:
if (applying_hac_control) {
[Stopping apply Control_surface_input using hybrid adaptive
   control for flight control loop from Input_command,
   State_ndi and State_ndi_m.]
~^[Applying hac control.]
};
```

```
[Set the initial parameter for nonlinear dynamic inversion
   control from State_ndi and hold Control_surface_input.]
[Apply Control_surface_input using nonlinear dynamic
   inversion control for flight control loop from
   Input_command, State_ndi and State_ndi_m.]
+^[Applying ndi control.].

If +^[Validate parameter.] under the condition of ^[True.]
then do the following:
[Set initial parameter for validating model in frequency
   domain and send State output back.]
[Check Update_state_of_model in term of model validation for
    nonlinear dynamic inversion control in frequency domain
   from State_ndi.].

If +^[Requiring revalidation.] under the condition of ^[True
   .]
then do the following:
if (applied_revalidation) {
[Stopping check Update_state_of_model in term of model
   validation for nonlinear dynamic inversion control in
   frequency domain from State_ndi.]
};
if (applied_revalidation_plus_delay) {
[Stopping check Update_state_of_agent in term of model
   validation for nonlinear dynamic inversion control in
   frequency domain with delayed time from State_ndi.]
};
[Set initial parameter for validating model in frequency
   domain and send State output back.]
```

```
[Check Update_state_of_model in term of model validation for
    nonlinear dynamic inversion control in frequency domain
   with delayed time from State_ndi.]
[Monitor aircraft manoeuvre condition or status from
   Altitude and send back output of NoUtilityFucn.]
~^[Applied revalidation plus delay.]
+^[Applied revalidation.]
~^[Requiring revalidation].

If +^[Requiring revalidation plus delay.] under the
   condition of ^[True.]
then do the following:
if (applied_revalidation) {
[Stopping check Update_state_of_model in term of model
   validation for nonlinear dynamic inversion control in
   frequency domain from State_ndi.]
};
if (applied_revalidation_plus_delay) {
[Stopping check Update_state_of_model in term of model
   validation for nonlinear dynamic inversion control in
   frequency domain with delayed time from State_ndi.]
};
[Set initial parameter for validating model in frequency
   domain and send State output back.]
[Monitor aircraft manoeuvre condition or status from
   Altitude and send back output of NoUtilityFucn.].

If +^[Altitude in bound.] under the condition of ^[Requiring
    revalidation plus delay.]
then do the following:
```

~^[Requiring revalidation plus delay.]
[Check Update_state_of_model in term of model validation for nonlinear dynamic inversion control in frequency domain with delayed time from State_ndi.]
[Set speed command back to required airspeed and send NoTriggerFunc.]
+^[Applied revalidation plus delay].

If +^[Termination of timer.] under the condition of ^[Applying major auto tuning with hac in longitudinal.]
then do the following:
[Stopping calculate new aerodynamic parameter of Theta for model-based adaptive control routines in longitudinal axes from State_ndi.]
[Stopping apply a 3-2-1-1 sequence input of Force on elevator.]
[Replace a new aerodynamic parameter for nonlinear dynamic inversion control routines in longitudinal from reidentification process and hold Control_surface_input.]
[Set the initial parameter for hybrid adaptive control from State_ndi and hold Control_surface_input.]
[Apply Control_surface_input using hybrid adaptive control for flight control loop from Input_command, State_ndi and State_ndi_m.]
[Stopping start to record the clock timer and send NoUtilityFunc back.]
~^[Applying major auto tuning in longitudinal.]
~^[Applying major auto tuning with hac in longitudinal.]
~^[Applied dac auto tuning in longitudinal.]
~^[Applied minor auto tuning in longitudinal.]

```
+^[Applying hac control]
+^[Requiring revalidation plus delay].

If +^[Termination of timer.] under the condition of ^[
   Applying major auto tuning with hac in lateral.]
then do the following:
[Stopping calculate new aerodynamic parameter of Theta for
   model-based adaptive control routines in lateral axes
   from State_ndi.]
[Stopping apply a 3-2-1-1 sequence input of Force on aileron
    and rudder repectively.]
[Replace a new aerodynamic parameter for nonlinear dynamic
   inversion control routines in lateral from
   reidentification process and hold Control_surface_input.]
[Set the initial parameter for hybrid adaptive control from
   State_ndi and hold Control_surface_input.]
[Apply Control_surface_input using hybrid adaptive control
   for flight control loop from Input_command, State_ndi and
    State_ndi_m.]
[Stopping start to record the clock timer and send
   NoUtilityFunc back.]
~^[Applying major auto tuning in lateral.]
~^[Applying major auto tuning with hac in lateral.]
~^[Applied dac auto tuning in lateral.]
~^[Applied minor auto tuning in lateral.]
+^[Applying hac control]
+^[Requiring revalidation plus delay].

If +^[Termination of timer.] under the condition of ^[
   Applying major auto tuning with hac in directional.]
```

```
then do the following:
[Stopping calculate new aerodynamic parameter of Theta for
   model-based adaptive control routines in directional axes
    from State_ndi.]
[Stopping apply a 3-2-1-1 sequence input of Force on aileron
    and rudder repectively.]
[Replace a new aerodynamic parameter for nonlinear dynamic
   inversion control routines in directional from
   reidentification process and hold Control_surface_input.]
[Set the initial parameter for hybrid adaptive control from
   State_ndi and hold Control_surface_input.]
[Apply Control_surface_input using hybrid adaptive control
   for flight control loop from Input_command, State_ndi and
    State_ndi_m.]
[Stopping start to record the clock timer and send
   NoUtilityFunc back.]
~^[Applying major auto tuning in directional.]
~^[Applying major auto tuning with hac in directional.]
~^[Applied dac auto tuning in directional.]
~^[Applied minor auto tuning in directional.]
+^[Applying hac control]
+^[Requiring revalidation plus delay].

If +^[Termination of timer.] under the condition of ^[
   Applying major auto tuning with hac in coupling axes.]
then do the following:
[Stopping calculate new aerodynamic parameter of Theta for
   model-based adaptive control routines in both lateral and
    directional axes from State_ndi.]
```

[Stopping apply a 3-2-1-1 sequence input of Force on aileron and rudder repectively.]

[Replace a new aerodynamic parameter for nonlinear dynamic inversion control routines in both lateral and directional from reidentification process and hold Control_surface_input.]

[Set the initial parameter for hybrid adaptive control from State_ndi and hold Control_surface_input.]

[Apply Control_surface_input using hybrid adaptive control for flight control loop from Input_command, State_ndi and State_ndi_m.]

[Stopping start to record the clock timer and send NoUtilityFunc back.]

~^[Applying major auto tuning in directional.]

~^[Applying major auto tuning in lateral.]

~^[Applying major auto tuning with hac in coupling axes.]

~^[Applied dac auto tuning in lateral.]

~^[Applied dac auto tuning in directional.]

~^[Applied minor auto tuning in lateral.]

~^[Applied minor auto tuning in directional.]

+^[Applying hac control]

+^[Requiring revalidation plus delay].

If +^[Requiring major adjustment in longitudinal.] under the condition of ^[Auto tuning.] & ^[Aircraft in trim status.] & not ^[Applying major auto tuning in longitudinal.] & not ^[Applying minor auto tuning in longitudinal.] & not ^[Applying dac auto tuning in longitudinal.] & not ^[Applying major auto tuning in lateral.] & not ^[Applying minor auto tuning in lateral.] & not ^[Applying dac auto

```
   tuning in lateral.] & not ^[Applying major auto tuning in
    directional.] & not ^[Applying minor auto tuning in
   directional.] & not ^[Applying dac auto tuning in
   directional.]
then do the following:
+^[Applying major auto tuning in longitudinal.]
if (applying_hac_control) {
[Stopping monitor aircraft manoeuvre condition or status
   from Altitude and send back output of NoUtilityFucn.]
[Set a speed command of cruise speed value and send
   NoUtilityFunc back.]
[Stopping apply Control_surface_input using hybrid adaptive
   control for flight control loop from Input_command,
   State_ndi and State_ndi_m.]
[Set the terminated time of ten second and send output back
   with NoUtilityFunc.]
[Set an initial vaule for a 3-2-1-1 sequence input of Force
   on elevator.]
[Set an initial value before reidentification process in
   pitching moment and send Theta output back.]
[Start to record the clock timer and send NoUtilityFunc back
   .]
[Apply a 3-2-1-1 sequence input of Force on elevator.]
[Calculate new aerodynamic parameter of Theta for model-
   based adaptive control routines in longitudinal axes from
    State_ndi.]
+^[Applying major auto tuning with hac in longitudinal.]
};
if (applying_hac_control) {
```

```
[Stopping monitor aircraft manoeuvre condition or status
   from Altitude and send back output of NoUtilityFucn.]
[Set a speed command of cruise speed value and send
   NoUtilityFunc back.]
[Stopping apply Control_surface_input using nonlinear
   dynamic inversion control for flight control loop from
   Input_command, State_ndi and State_ndi_m.]
[Set the terminated time of ten second and send output back
   with NoUtilityFunc.]
[Set an initial vaule for a 3-2-1-1 sequence input of Force
   on elevator.]
[Set an initial value before reidentification process in
   pitching moment and send Theta output back.]
[Start to record the clock timer and send NoUtilityFunc back
   .]
[Apply a 3-2-1-1 sequence input of Force on elevator.]
[Calculate new aerodynamic parameter of Theta for model-
   based adaptive control routines in longitudinal axes from
    State_ndi.]
+^[Applying major auto tuning with ndi in longitudinal.]
};
+^[Waiting termination.].

If +^[Aircraft in trim status.] under the condition of ^[
   Auto tuning.] & ^[Requiring major adjustment in
   longitudinal.] & not ^[Applying major auto tuning in
   longitudinal.] & not ^[Applying minor auto tuning in
   longitudinal.] & not ^[Applying dac auto tuning in
   longitudinal.] & not ^[Applying major auto tuning in
   lateral.] & not ^[Applying minor auto tuning in lateral.]
```

```
    & not ^[Applying dac auto tuning in lateral.] & not ^[
    Applying major auto tuning in directional.] & not ^[
    Applying minor auto tuning in directional.] & not ^[
    Applying dac auto tuning in directional.]
then do the following:
+^[Applying major auto tuning in longitudinal.]
if (applying_hac_control) {
[Stopping monitor aircraft manoeuvre condition or status
    from Altitude and send back output of NoUtilityFucn.]
[Set a speed command of cruise speed value and send
    NoUtilityFunc back.]
[Stopping apply Control_surface_input using hybrid adaptive
    control for flight control loop from Input_command,
    State_ndi and State_ndi_m.]
[Set the terminated time of ten second and send output back
    with NoUtilityFunc.]
[Set an initial vaule for a 3-2-1-1 sequence input of Force
    on elevator.]
[Set an initial value before reidentification process in
    pitching moment and send Theta output back.]
[Start to record the clock timer and send NoUtilityFunc back
    .]
[Apply a 3-2-1-1 sequence input of Force on elevator.]
[Calculate new aerodynamic parameter of Theta for model-
    based adaptive control routines in longitudinal axes from
     State_ndi.]
+^[Applying major auto tuning with hac in longitudinal.]
};
if (applying_hac_control) {
```

```
[Stopping monitor aircraft manoeuvre condition or status
   from Altitude and send back output of NoUtilityFucn.]
[Set a speed command of cruise speed value and send
   NoUtilityFunc back.]
[Stopping apply Control_surface_input using nonlinear
   dynamic inversion control for flight control loop from
   Input_command, State_ndi and State_ndi_m.]
[Set the terminated time of ten second and send output back
   with NoUtilityFunc.]
[Set an initial vaule for a 3-2-1-1 sequence input of Force
   on elevator.]
[Set an initial value before reidentification process in
   pitching moment and send Theta output back.]
[Start to record the clock timer and send NoUtilityFunc back
   .]
[Apply a 3-2-1-1 sequence input of Force on elevator.]
[Calculate new aerodynamic parameter of Theta for model-
   based adaptive control routines in longitudinal axes from
    State_ndi.]
+^[Applying major auto tuning with ndi in longitudinal.]
};
+^[Waiting termination.].

If +^[Aircraft in trim status.] under the condition of ^[
   Auto tuning.] & ^[Requiring major adjustment in lateral.]
    & not ^[Applying major auto tuning in longitudinal.] &
   not ^[Applying minor auto tuning in longitudinal.] & not
   ^[Applying dac auto tuning in longitudinal.] & not ^[
   Applying major auto tuning in lateral.] & not ^[Applying
   minor auto tuning in lateral.] & not ^[Applying dac auto
```

```
   tuning in lateral.] & not ^[Applying major auto tuning in
    directional.] & not ^[Applying minor auto tuning in
   directional.] & not ^[Applying dac auto tuning in
   directional.]
then do the following:
+^[Applying major auto tuning in lateral.]
if (applying_hac_control) {
[Stopping monitor aircraft manoeuvre condition or status
   from Altitude and send back output of NoUtilityFucn.]
[Set a speed command of cruise speed value and send
   NoUtilityFunc back.]
[Stopping apply Control_surface_input using hybrid adaptive
   control for flight control loop from Input_command,
   State_ndi and State_ndi_m.]
[Set the terminated time of twelve second and send output
   back with NoUtilityFunc.]
[Set an initial vaule for a 3-2-1-1 sequence input of Force
   on aileron and rudder repectively from State_ndi.]
[Set an initial value before reidentification process in
   rolling moment and send Theta output back.]
[Start to record the clock timer and send NoUtilityFunc back
   .]
[Apply a 3-2-1-1 sequence input of Force on aileron and
   rudder repectively.]
[Calculate new aerodynamic parameter of Theta for model-
   based adaptive control routines in lateral axes from
   State_ndi.]
+^[Applying major auto tuning with hac in lateral.]
};
+^[Waiting termination.].
```

```
If +^[Aircraft in trim status.] under the condition of ^[
   Auto tuning.] & ^[Requiring major adjustment in
   directional.] & not ^[Applying major auto tuning in
   longitudinal.] & not ^[Applying minor auto tuning in
   longitudinal.] & not ^[Applying dac auto tuning in
   longitudinal.] & not ^[Applying major auto tuning in
   lateral.] & not ^[Applying minor auto tuning in lateral.]
    & not ^[Applying dac auto tuning in lateral.] & not ^[
   Applying major auto tuning in directional.] & not ^[
   Applying minor auto tuning in directional.] & not ^[
   Applying dac auto tuning in directional.]
then do the following:
+^[Applying major auto tuning in directional.]
if (applying_hac_control) {
[Stopping monitor aircraft manoeuvre condition or status
   from Altitude and send back output of NoUtilityFucn.]
[Set a speed command of cruise speed value and send
   NoUtilityFunc back.]
[Stopping apply Control_surface_input using hybrid adaptive
   control for flight control loop from Input_command,
   State_ndi and State_ndi_m.]
[Set the terminated time of twelve second and send output
   back with NoUtilityFunc.]
[Set an initial vaule for a 3-2-1-1 sequence input of Force
   on aileron and rudder repectively from State_ndi.]
[Set an initial value before reidentification process in
   yawing moment and send Theta output back.]
[Start to record the clock timer and send NoUtilityFunc back
   .]
```

```
[Apply a 3-2-1-1 sequence input of Force on aileron and
   rudder repectively.]
[Calculate new aerodynamic parameter of Theta for model-
   based adaptive control routines in directional axes from
   State_ndi.]
+^[Applying major auto tuning with hac in directional.]
};
+^[Waiting termination.].

If +^[Aircraft in trim status.] under the condition of ^[
   Auto tuning.] & ^[Applied minor auto tuning in
   longitudinal.] & not ^[Applying major auto tuning in
   longitudinal.] & not ^[Applying minor auto tuning in
   longitudinal.] & not ^[Applying dac auto tuning in
   longitudinal.] & not ^[Applying major auto tuning in
   lateral.] & not ^[Applying minor auto tuning in lateral.]
    & not ^[Applying dac auto tuning in lateral.] & not ^[
   Applying major auto tuning in directional.] & not ^[
   Applying minor auto tuning in directional.] & not ^[
   Applying dac auto tuning in directional.]
then do the following:
+^[Applying major auto tuning in longitudinal.]
if (applying_hac_control) {
[Stopping monitor aircraft manoeuvre condition or status
   from Altitude and send back output of NoUtilityFucn.]
[Set a speed command of cruise speed value and send
   NoUtilityFunc back.]
[Stopping apply Control_surface_input using hybrid adaptive
   control for flight control loop from Input_command,
   State_ndi and State_ndi_m.]
```

```
[Set the terminated time of ten second and send output back
   with NoUtilityFunc.]
[Set an initial vaule for a 3-2-1-1 sequence input of Force
   on elevator.]
[Set an initial value before reidentification process in
   pitching moment and send Theta output back.]
[Start to record the clock timer and send NoUtilityFunc back
   .]
[Apply a 3-2-1-1 sequence input of Force on elevator.]
[Calculate new aerodynamic parameter of Theta for model-
   based adaptive control routines in longitudinal axes from
    State_ndi.]
+^[Applying major auto tuning with hac in longitudinal.]
};
+^[Waiting termination.].

If +^[Aircraft in trim status.] under the condition of ^[
   Auto tuning.] & ^[Applied minor auto tuning in lateral.]
   & not ^[Applied minor auto tuning in directional.] & not
   ^[Applying major auto tuning in longitudinal.] & not ^[
   Applying minor auto tuning in longitudinal.] & not ^[
   Applying dac auto tuning in longitudinal.] & not ^[
   Applying major auto tuning in lateral.] & not ^[Applying
   minor auto tuning in lateral.] & not ^[Applying dac auto
   tuning in lateral.] & not ^[Applying major auto tuning in
    directional.] & not ^[Applying minor auto tuning in
   directional.] & not ^[Applying dac auto tuning in
   directional.]
then do the following:
+^[Applying major auto tuning in lateral.]
```

```
if (applying_hac_control) {
[Stopping monitor aircraft manoeuvre condition or status
   from Altitude and send back output of NoUtilityFucn.]
[Set a speed command of cruise speed value and send
   NoUtilityFunc back.]
[Stopping apply Control_surface_input using hybrid adaptive
   control for flight control loop from Input_command,
   State_ndi and State_ndi_m.]
[Set the terminated time of twelve second and send output
   back with NoUtilityFunc.]
[Set an initial vaule for a 3-2-1-1 sequence input of Force
   on aileron and rudder repectively from State_ndi.]
[Set an initial value before reidentification process in
   rolling moment and send Theta output back.]
[Start to record the clock timer and send NoUtilityFunc back
   .]
[Apply a 3-2-1-1 sequence input of Force on aileron and
   rudder repectively.]
[Calculate new aerodynamic parameter of Theta for model-
   based adaptive control routines in lateral axes from
   State_ndi.]
+^[Applying major auto tuning with hac in lateral.]
};
+^[Waiting termination.].

If +^[Aircraft in trim status.] under the condition of ^[
   Auto tuning.] & ^[Applied minor auto tuning in
   directional.] & not ^[Applied minor auto tuning in
   lateral.] & not ^[Applying major auto tuning in
   longitudinal.] & not ^[Applying minor auto tuning in
```

```
   longitudinal.] & not ^[Applying dac auto tuning in
   longitudinal.] & not ^[Applying major auto tuning in
   lateral.] & not ^[Applying minor auto tuning in lateral.]
    & not ^[Applying dac auto tuning in lateral.] & not ^[
   Applying major auto tuning in directional.] & not ^[
   Applying minor auto tuning in directional.] & not ^[
   Applying dac auto tuning in directional.]
then do the following:
+^[Applying major auto tuning in directional.]
if (applying_hac_control) {
[Stopping monitor aircraft manoeuvre condition or status
   from Altitude and send back output of NoUtilityFucn.]
[Set a speed command of cruise speed value and send
   NoUtilityFunc back.]
[Stopping apply Control_surface_input using hybrid adaptive
   control for flight control loop from Input_command,
   State_ndi and State_ndi_m.]
[Set the terminated time of twelve second and send output
   back with NoUtilityFunc.]
[Set an initial vaule for a 3-2-1-1 sequence input of Force
   on aileron and rudder repectively from State_ndi.]
[Set an initial value before reidentification process in
   yawing moment and send Theta output back.]
[Start to record the clock timer and send NoUtilityFunc back
   .]
[Apply a 3-2-1-1 sequence input of Force on aileron and
   rudder repectively.]
[Calculate new aerodynamic parameter of Theta for model-
   based adaptive control routines in directional axes from
   State_ndi.]
```

```
+^[Applying major auto tuning with hac in directional.]
};
+^[Waiting termination.].

If +^[Aircraft in trim status.] under the condition of ^[
   Auto tuning.] & ^[Applied minor auto tuning in
   directional.] & ^[Applied minor auto tuning in lateral.]
   & not ^[Applying major auto tuning in longitudinal.] &
   not ^[Applying minor auto tuning in longitudinal.] & not
   ^[Applying dac auto tuning in longitudinal.] & not ^[
   Applying major auto tuning in lateral.] & not ^[Applying
   minor auto tuning in lateral.] & not ^[Applying dac auto
   tuning in lateral.] & not ^[Applying major auto tuning in
    directional.] & not ^[Applying minor auto tuning in
   directional.] & not ^[Applying dac auto tuning in
   directional.]
then do the following:
+^[Applying major auto tuning in lateral.]
+^[Applying major auto tuning in directional.]
if (applying_hac_control) {
[Stopping monitor aircraft manoeuvre condition or status
   from Altitude and send back output of NoUtilityFucn.]
[Set a speed command of cruise speed value and send
   NoUtilityFunc back.]
[Stopping apply Control_surface_input using hybrid adaptive
   control for flight control loop from Input_command,
   State_ndi and State_ndi_m.]
[Set the terminated time of twelve second and send output
   back with NoUtilityFunc.]
```

```
[Set an initial vaule for a 3-2-1-1 sequence input of Force
   on aileron and rudder repectively from State_ndi.]
[Set an initial value before reidentification process in
   rolling and yawing moment and send Theta output back.]
[Start to record the clock timer and send NoUtilityFunc back
   .]
[Apply a 3-2-1-1 sequence input of Force on aileron and
   rudder repectively.]
[Calculate new aerodynamic parameter of Theta for model-
   based adaptive control routines in both lateral and
   directional axes from State_ndi.]
+^[Applying major auto tuning with hac in coupling axes.]
};
+^[Waiting termination.].

If +^[Termination of timer.] under the condition of ^[
   Applying minor auto tuning with hac in longitudinal.]
then do the following:
[Stopping calculate Theta that is the compensator makeing
   use and compensate of model-based adaptive control
   routines for pitching maneuver from State_ndi.]
[Stopping start to record the clock timer and send
   NoUtilityFunc back.]
~^[Applying minor auto tuning in longitudinal.]
~^[Applying minor auto tuning with hac in longitudinal.]
+^[Applying hac control.]
+^[Applied minor auto tuning in longitudinal.]
+^[Requiring revalidation.].
```

```
If +^[Termination of timer.] under the condition of ^[
   Applying minor auto tuning with ndi in longitudinal.]
then do the following:
[Stopping calculate Theta that is the compensator makeing
   use and compensate of model-based adaptive control
   routines for pitching maneuver from State_ndi.]
[Stopping start to record the clock timer and send
   NoUtilityFunc back.]
~^[Applying minor auto tuning in longitudinal.]
~^[Applying minor auto tuning with ndi in longitudinal.]
+^[Applying ndi control.]
+^[Applied minor auto tuning in longitudinal.]
+^[Requiring revalidation.].

If +^[Termination of timer.] under the condition of ^[
   Applying minor auto tuning with hac in lateral.]
then do the following:
[Stopping calculate Theta that is the compensator makeing
   use and compensate of model-based adaptive control
   routines for rolling maneuver from State_ndi.]
[Stopping start to record the clock timer and send
   NoUtilityFunc back.]
~^[Applying minor auto tuning in lateral.]
~^[Applying minor auto tuning with hac in lateral.]
+^[Applying hac control.]
+^[Applied minor auto tuning in lateral.]
+^[Requiring revalidation.].

If +^[Termination of timer.] under the condition of ^[
   Applying minor auto tuning with hac in directional.]
```

```
then do the following:
[Stopping calculate Theta that is the compensator makeing
   use and compensate of model-based adaptive control
   routines for yawing maneuver from State_ndi.]
[Stopping start to record the clock timer and send
   NoUtilityFunc back.]
~^[Applying minor auto tuning in directional.]
~^[Applying minor auto tuning with hac in directional.]
+^[Applying hac control.]
+^[Applied minor auto tuning in directional.]
+^[Requiring revalidation.].

If +^[Auto tuning.] under the condition of ^[Requiring minor
    adjustment in longitudinal.] & ^[Applied dac auto tuning
    in longitudinal.] & not ^[Applying major auto tuning in
   longitudinal.] & not ^[Applying minor auto tuning in
   longitudinal.] & not ^[Applying dac auto tuning in
   longitudinal.] & not ^[Applying major auto tuning in
   lateral.] & not ^[Applying minor auto tuning in lateral.]
    & not ^[Applying dac auto tuning in lateral.] & not ^[
   Applying major auto tuning in directional.] & not ^[
   Applying minor auto tuning in directional.] & not ^[
   Applying dac auto tuning in directional.]
then do the following:
+^[Applying minor auto tuning in longitudinal.]
if (applying_hac_control) {
[Set the initial parameter for control compensation using
   frequency least square algorithm for pitching maneuver
   and send Theta output back.]
```

```
[Calculate Theta that is the compensator makeing use and
   compensate of model-based adaptive control routines for
   pitching maneuver from State_ndi.]
[Stopping monitor aircraft manoeuvre condition or status
   from Altitude and send back output of NoUtilityFucn.]
[Set the terminated time of sixty second and send output
   back with NoUtilityFunc.]
[Start to record the clock timer and send NoUtilityFunc back
   .]
+^[Apply hac control.]
+^[Applying minor auto tuning with hac in longitudinal.]
};
if (applying_ndi_control) {
[Set the initial parameter for control compensation using
   frequency least square algorithm for pitching maneuver
   and send Theta output back.]
[Calculate Theta that is the compensator makeing use and
   compensate of model-based adaptive control routines for
   pitching maneuver from State_ndi.]
[Stopping monitor aircraft manoeuvre condition or status
   from Altitude and send back output of NoUtilityFucn.]
[Set the terminated time of sixty second and send output
   back with NoUtilityFunc.]
[Start to record the clock timer and send NoUtilityFunc back
   .]
+^[Apply ndi control.]
+^[Applying minor auto tuning with ndi in longitudinal.]
};
+^[Waiting termination.].
```

```
If +^[Requiring minor adjustment in lateral.] under the
   condition of ^[Auto tuning.] & ^[Applied dac auto tuning
   in lateral.] & not ^[Applied minor auto tuning in lateral
   .] & not ^[Applying major auto tuning in longitudinal.] &
    not ^[Applying minor auto tuning in longitudinal.] & not
    ^[Applying dac auto tuning in longitudinal.] & not ^[
   Applying major auto tuning in lateral.] & not ^[Applying
   minor auto tuning in lateral.] & not ^[Applying dac auto
   tuning in lateral.] & not ^[Applying major auto tuning in
    directional.] & not ^[Applying minor auto tuning in
   directional.] & not ^[Applying dac auto tuning in
   directional.]
then do the following:
+^[Applying minor auto tuning in lateral.]
if (applying_hac_control) {
[Reset adaptive gain adjustment for direct adaptive control
   and send back the output of NoTriggerFunc.]
[Set the initial parameter for control compensation using
   frequency least square algorithm for rolling maneuver and
    send Theta output back.]
[Calculate Theta that is the compensator makeing use and
   compensate of model-based adaptive control routines for
   rolling maneuver from State_ndi.]
[Stopping monitor aircraft manoeuvre condition or status
   from Altitude and send back output of NoUtilityFucn.]
[Set the terminated time of thirty second and send output
   back with NoUtilityFunc.]
[Start to record the clock timer and send NoUtilityFunc back
   .]
+^[Apply hac control.]
```

```
+^[Applying minor auto tuning with hac in lateral.]
};
if (applying_ndi_control) {
[Set the initial parameter for control compensation using
   frequency least square algorithm for rolling maneuver and
    send Theta output back.]
[Calculate Theta that is the compensator makeing use and
   compensate of model-based adaptive control routines for
   rolling maneuver from State_ndi.]
[Stopping monitor aircraft manoeuvre condition or status
   from Altitude and send back output of NoUtilityFucn.]
[Set the terminated time of thirty second and send output
   back with NoUtilityFunc.]
[Start to record the clock timer and send NoUtilityFunc back
   .]
+^[Apply ndi control.]
+^[Applying minor auto tuning with ndi in lateral.]
};
+^[Waiting termination.].

If +^[Requiring minor adjustment in directional.] under the
   condition of ^[Auto tuning.] & ^[Applied dac auto tuning
   in directional.] & not ^[Applied minor auto tuning in
   directional.] & not ^[Applying major auto tuning in
   longitudinal.] & not ^[Applying minor auto tuning in
   longitudinal.] & not ^[Applying dac auto tuning in
   longitudinal.] & not ^[Applying major auto tuning in
   lateral.] & not ^[Applying minor auto tuning in lateral.]
    & not ^[Applying dac auto tuning in lateral.] & not ^[
   Applying major auto tuning in directional.] & not ^[
```

```
   Applying minor auto tuning in directional.] & not ^[
   Applying dac auto tuning in directional.]
then do the following:
+^[Applying minor auto tuning in directional.]
if (applying_hac_control) {
[Reset adaptive gain adjustment for direct adaptive control
   and send back the output of NoTriggerFunc.]
[Set the initial parameter for control compensation using
   frequency least square algorithm for yawing maneuver and
   send Theta output back.]
[Calculate Theta that is the compensator makeing use and
   compensate of model-based adaptive control routines for
   yawing maneuver from State_ndi.]
[Stopping monitor aircraft manoeuvre condition or status
   from Altitude and send back output of NoUtilityFucn.]
[Set the terminated time of thirty second and send output
   back with NoUtilityFunc.]
[Start to record the clock timer and send NoUtilityFunc back
   .]
+^[Apply hac control.]
+^[Applying minor auto tuning with hac in directional.]
};
if (applying_ndi_control) {
[Set the initial parameter for control compensation using
   frequency least square algorithm for yawing maneuver and
   send Theta output back.]
[Calculate Theta that is the compensator makeing use and
   compensate of model-based adaptive control routines for
   yawing maneuver from State_ndi.]
```

```
[Stopping monitor aircraft manoeuvre condition or status
   from Altitude and send back output of NoUtilityFucn.]
[Set the terminated time of thirty second and send output
   back with NoUtilityFunc.]
[Start to record the clock timer and send NoUtilityFunc back
   .]
+^[Apply ndi control.]
+^[Applying minor auto tuning with ndi in directional.]
};
+^[Waiting termination.].

If +^[Requiring minor adjustment in longitudinal.] under the
    condition of ^[Auto tuning.] & ^[Applied dac auto tuning
    in longitudinal.] & not ^[Applied minor auto tuning in
   longitudinal.] & not ^[Applying major auto tuning in
   longitudinal.] & not ^[Applying minor auto tuning in
   longitudinal.] & not ^[Applying dac auto tuning in
   longitudinal.] & not ^[Applying major auto tuning in
   lateral.] & not ^[Applying minor auto tuning in lateral.]
    & not ^[Applying dac auto tuning in lateral.] & not ^[
   Applying major auto tuning in directional.] & not ^[
   Applying minor auto tuning in directional.] & not ^[
   Applying dac auto tuning in directional.]
then do the following:
+^[Applying minor auto tuning in longitudinal.]
if (applying_hac_control) {
[Reset adaptive gain adjustment for direct adaptive control
   and send back the output of NoTriggerFunc.]
[Set the initial parameter for control compensation using
   frequency least square algorithm for pitching maneuver
```

```
   and send Theta output back.]
[Calculate Theta that is the compensator makeing use and
   compensate of model-based adaptive control routines for
   pitching maneuver from State_ndi.]
[Stopping monitor aircraft manoeuvre condition or status
   from Altitude and send back output of NoUtilityFucn.]
[Set the terminated time of sixty second and send output
   back with NoUtilityFunc.]
[Start to record the clock timer and send NoUtilityFunc back
   .]
+^[Apply hac control.]
+^[Applying minor auto tuning with hac in longitudinal.]
};
if (applying_ndi_control) {
[Set the initial parameter for control compensation using
   frequency least square algorithm for pitching maneuver
   and send Theta output back.]
[Calculate Theta that is the compensator makeing use and
   compensate of model-based adaptive control routines for
   pitching maneuver from State_ndi.]
[Stopping monitor aircraft manoeuvre condition or status
   from Altitude and send back output of NoUtilityFucn.]
[Set the terminated time of sixty second and send output
   back with NoUtilityFunc.]
[Start to record the clock timer and send NoUtilityFunc back
   .]
+^[Apply ndi control.]
+^[Applying minor auto tuning with ndi in longitudinal.]
};
+^[Waiting termination.].
```

```
If +^[Auto tuning.] under the condition of ^[Requiring
   direct adaptive in longitudinal.] & not ^[Applied dac
   auto tuning in longitudinal.] & not ^[Applying major auto
    tuning in longitudinal.] & not ^[Applying minor auto
   tuning in longitudinal.] & not ^[Applying dac auto tuning
    in longitudinal.] & not ^[Applying major auto tuning in
   lateral.] & not ^[Applying minor auto tuning in lateral.]
    & not ^[Applying dac auto tuning in lateral.] & not ^[
   Applying major auto tuning in directional.] & not ^[
   Applying minor auto tuning in directional.] & not ^[
   Applying dac auto tuning in directional.]
then do the following:
+^[Applying dac auto tuning in longitudinal.]
if (applying_ndi_control) {
[Stopping apply Control_surface_input using nonlinear
   dynamic inversion control for flight control loop from
   Input_command, State_ndi and State_ndi_m.]
[Clear the compensated parameter and send Theta output.]
[Set the initial parameter for hybrid adaptive control from
   State_ndi and hold Control_surface_input.]
[Apply Control_surface_input using hybrid adaptive control
   for flight control loop from Input_command, State_ndi and
    State_ndi_m.]
~^[Applying ndi control.]
+^[Applying hac control.]
};
+^[Requiring revalidation.]
~^[Applying dac auto tuning in longitudinal.]
+^[Applied dac auto tuning in longitudinal.].
```

```
If +^[Requiring direct adaptive in lateral.] under the
   condition of ^[Auto tuning.] & not ^[Applied dac auto
   tuning in lateral.] & not ^[Applying major auto tuning in
    longitudinal.] & not ^[Applying minor auto tuning in
   longitudinal.] & not ^[Applying dac auto tuning in
   longitudinal.] & not ^[Applying major auto tuning in
   lateral.] & not ^[Applying minor auto tuning in lateral.]
    & not ^[Applying dac auto tuning in lateral.] & not ^[
   Applying major auto tuning in directional.] & not ^[
   Applying minor auto tuning in directional.] & not ^[
   Applying dac auto tuning in directional.]
then do the following:
+^[Applying dac auto tuning in lateral.]
if (applying_ndi_control) {
[Stopping apply Control_surface_input using nonlinear
   dynamic inversion control for flight control loop from
   Input_command, State_ndi and State_ndi_m.]
[Clear the compensated parameter and send Theta output.]
[Set the initial parameter for hybrid adaptive control from
   State_ndi and hold Control_surface_input.]
[Apply Control_surface_input using hybrid adaptive control
   for flight control loop from Input_command, State_ndi and
    State_ndi_m.]
~^[Applying ndi control.]
+^[Applying hac control.]
};
+^[Requiring revalidation.]
~^[Applying dac auto tuning in lateral.]
+^[Applied dac auto tuning in lateral.].
```

```
If +^[Requiring direct adaptive in directional.] under the
   condition of ^[Auto tuning.] & not ^[Applied dac auto
   tuning in directional.] & not ^[Applying major auto
   tuning in longitudinal.] & not ^[Applying minor auto
   tuning in longitudinal.] & not ^[Applying dac auto tuning
    in longitudinal.] & not ^[Applying major auto tuning in
   lateral.] & not ^[Applying minor auto tuning in lateral.]
    & not ^[Applying dac auto tuning in lateral.] & not ^[
   Applying major auto tuning in directional.] & not ^[
   Applying minor auto tuning in directional.] & not ^[
   Applying dac auto tuning in directional.]
then do the following:
+^[Applying dac auto tuning in directional.]
if (applying_ndi_control) {
[Stopping apply Control_surface_input using nonlinear
   dynamic inversion control for flight control loop from
   Input_command, State_ndi and State_ndi_m.]
[Clear the compensated parameter and send Theta output.]
[Set the initial parameter for hybrid adaptive control from
   State_ndi and hold Control_surface_input.]
[Apply Control_surface_input using hybrid adaptive control
   for flight control loop from Input_command, State_ndi and
    State_ndi_m.]
~^[Applying ndi control.]
+^[Applying hac control.]
};
+^[Requiring revalidation.]
~^[Applying dac auto tuning in directional.]
+^[Applied dac auto tuning in directional.].
```

```
If +^[Requiring direct adaptive in longitudinal.] under the
   condition of ^[Auto tuning.] & not ^[Applied dac auto
   tuning in longitudinal.] & not ^[Applying major auto
   tuning in longitudinal.] & not ^[Applying minor auto
   tuning in longitudinal.] & not ^[Applying dac auto tuning
    in longitudinal.] & not ^[Applying major auto tuning in
   lateral.] & not ^[Applying minor auto tuning in lateral.]
    & not ^[Applying dac auto tuning in lateral.] & not ^[
   Applying major auto tuning in directional.] & not ^[
   Applying minor auto tuning in directional.] & not ^[
   Applying dac auto tuning in directional.]
then do the following:
+^[Applying dac auto tuning in longitudinal.]
if (applying_ndi_control) {
[Stopping apply Control_surface_input using nonlinear
   dynamic inversion control for flight control loop from
   Input_command, State_ndi and State_ndi_m.]
[Clear the compensated parameter and send Theta output.]
[Set the initial parameter for hybrid adaptive control from
   State_ndi and hold Control_surface_input.]
[Apply Control_surface_input using hybrid adaptive control
   for flight control loop from Input_command, State_ndi and
    State_ndi_m.]
~^[Applying ndi control.]
+^[Applying hac control.]
};
+^[Requiring revalidation.]
~^[Applying dac auto tuning in longitudinal.]
+^[Applied dac auto tuning in longitudinal.].
```

```
If +^[Aircraft in trim status.] under the condition of ^[
   Auto tuning.] & not ^[Applying major auto tuning in
   longitudinal.] & not ^[Applying minor auto tuning in
   longitudinal.] & not ^[Applying dac auto tuning in
   longitudinal.] & not ^[Applying major auto tuning in
   lateral.] & not ^[Applying minor auto tuning in lateral.]
    & not ^[Applying dac auto tuning in lateral.] & not ^[
   Applying major auto tuning in directional.] & not ^[
   Applying minor auto tuning in directional.] & not ^[
   Applying dac auto tuning in directional.]
then do the following:
if (applied_revalidation){
[Stopping check Update_state_of_model in term of model
   validation for nonlinear dynamic inversion control in
   frequency domain from State_ndi.]
};
if (applied_revalidation_plus_delay) {
[Stopping check Update_state_of_agent in term of model
   validation for nonlinear dynamic inversion control in
   frequency domain with delayed time from State_ndi.]
};
[Set initial parameter for validating model in frequency
   domain and send State output back.]
[Check Update_state_of_model in term of model validation for
    nonlinear dynamic inversion control in frequency domain
   from State_ndi.]
~^[Applied revalidation plus delay.]
+^[Applied revalidation.]
~^[Applied dac auto tuning in longitudinal.]
```

```
~^[Applied minor auto tuning in longitudinal.]
~^[Applied dac auto tuning in lateral.]
~^[Applied minor auto tuning in lateral.]
~^[Applied dac auto tuning in directional.]
~^[Applied minor auto tuning in directional.].

If -^[Control via ndi.] under the condition of ~^[Control
   via hac]
then do the following:
[Stopping apply Control_surface_input using nonlinear
   dynamic inversion control for flight control loop from
   Input_command, State_ndi and State_ndi_m.]
~^[Applying ndi control.].

If -^[Control via ndi.] under the condition of ^[Applying
   ndi control.]
then do the following:
[Stopping apply Control_surface_input using nonlinear
   dynamic inversion control for flight control loop from
   Input_command, State_ndi and State_ndi_m.]
~^[Applying ndi control.].

If -^[Control via ndi.] under the condition of ^[Applying
   ndi control.] & ^[Control via hac]
then do the following:
[Stopping apply Control_surface_input using nonlinear
   dynamic inversion control for flight control loop from
   Input_command, State_ndi and State_ndi_m.]
~^[Applying ndi control.]
```

```
[Set the initial parameter for hybrid adaptive control from
   State_ndi and hold Control_surface_input.]
[Apply Control_surface_input using hybrid adaptive control
   for flight control loop from Input_command, State_ndi and
    State_ndi_m.]
+^[Applying hac control.].

If -^[Validate parameter.] under the condition of ^[True.]
then do the following:
[Stopping check Update_state_of_model in term of model
   validation for nonlinear dynamic inversion control in
   frequency domain from State_ndi.].

If -^[Auto tuning.] under the condition of ^[True.]
then do the following:
~^[Applying auto tuning.].
```

Appendix E: Source Code in Format of Jason/AgentSpeak for Agent Decision

```
//  INITIAL GOAL AND BELIEFS

!configureSystem.


// PERCEPTION PROCESSES

// Note: Perception Boolean tag names have to be identical
   to their sentence in lower case.

+!configureSystem:true <-
linkSystems(6253,control_subsystem,planning_subsystem,
   guidance_subsystem,subsystem2,habitat,utility_system,
   subsystem3,triggering_subsystem);
!take_initial_actions.
// INITIAL BELIEFS
```

```
-applied_dac_auto_tunning_in_longitudinal .
-applied_minor_auto_tuning_in_longitudinal .
-applied_dac_auto_tunning_in_lateral .
-applied_minor_auto_tuning_in_lateral .
-applied_dac_auto_tunning_in_directional .
-applied_minor_auto_tuning_in_directional .
+applying_hac_control .
+first_flight .
// INITIAL ACTIONS
+!take_initial_actions <-
invoke(control_subsystem,runOnce,initial_parameter_for_agent
    ,[],["Force"]);
invoke(planning_subsystem,runRepeated,plan_flight_path,["
    Aircraft_position","Aircraft_heading","Target_position","
    Finaltarget_direction","Option"],["Pathpoint"]);
invoke(guidance_subsystem,runRepeated,
    apply_guidance_control12,["P","V","Required_altitude"],["
    Input"]);
invoke(control_subsystem,runRepeated,apply_hac_control,["
    Input_command","State_ndi","State_ndi_m"],["
    Control_surface_input"]);
if (~first_flight) {
invoke(subsystem2,runRepeated,
    validate_model_in_frequency_domainv2,["State_ndi"],["
    Update_state_of_model"]);
};
invoke(habitat,runRepeated,update_percept,[]);
invoke(utility_system,runRepeated,monitor_aircraft_status,["
    Altitude"],["NoUtilityFucn"]).
```

```
// UPDATING PERCEPTION
+!linkSystems(complete):true <- checkPercepts.
@cd
!checkPercepts <-
updateSystems(control_subsystem,planning_subsystem,
    guidance_subsystem,subsystem2,habitat,utility_system,
    subsystem3,triggering_subsystem);
!checkPercepts.

// REASONING
applied_revalidation :- ~first_flight .
switch_off_compensate_parameter :- ~compensate_parameter .
// EXECUTABLE PLANS
// executable plan :
+first_flight :
true <-
+applying_minor_auto_tuning_in_lateral;
+applying_minor_auto_tuning_in_directional;
+applying_minor_auto_tuning_in_longitudinal;
invoke(subsystem3,runRepeated,
    compensate_aircraft_parameter_v2_for_first_flight,["
    State_ndi"],["Theta"]);
invoke(utility_system,stopRepeated,monitor_aircraft_status,[
    "Altitude"],["NoUtilityFucn"]);
invoke(utility_system,runOnce,
    set_terminated_time_of_thirty_second,[],["NoUtilityFunc"
    ]);
invoke(utility_system,runRepeated,reckon_time,[],["
    NoUtilityFunc"]);
if (applying_ndi_control) { invoke(control_subsystem,
```

```
    stopRepeated,apply_ndi_control,["Input_command","
    State_ndi","State_ndi_m"],["Control_surface_input"]);
~applying_ndi_control;
invoke(control_subsystem,runOnce,
    set_initial_parameter_for_hac,["State_ndi"],["
    Control_surface_input"]);
invoke(control_subsystem,runRepeated,apply_hac_control,["
    Input_command","State_ndi","State_ndi_m"],["
    Control_surface_input"]);
+applying_hac_control;
+applying_minor_auto_tuning_with_hac_for_first_flight;
}; if (applying_hac_control) { +apply_hac_control;
+applying_minor_auto_tuning_with_hac_for_first_flight;
}; ~first_flight .
// executable plan :
+termination_of_timer :
applying_minor_auto_tuning_with_hac_for_first_flight <-
invoke(subsystem3,stopRepeated,
    compensate_aircraft_parameter_v2_for_first_flight,["
    State_ndi"],["Theta"]);
invoke(utility_system,stopRepeated,reckon_time,[],["
    NoUtilityFunc"]);
invoke(subsystem2,runRepeated,
    validate_model_in_frequency_domainv2,["State_ndi"],["
    Update_state_of_model"]);
invoke(utility_system,runRepeated,monitor_aircraft_status,["
    Altitude"],["NoUtilityFucn"]);
~applying_minor_auto_tuning_in_lateral;
~applying_minor_auto_tuning_in_directional;
~applying_minor_auto_tuning_in_longitudinal;
```

```
~applying_minor_auto_tuning_with_hac_for_first_flight;
+apply_hac_control;
+applied_revalidation;
+applied_minor_auto_tuning_in_lateral;
+applied_minor_auto_tuning_in_longitudinal;
+applied_minor_auto_tuning_in_directional .
// executable plan :
+bad_control_performance :
auto_tuning & not applying_major_auto_tuning_in_longitudinal
    & not applying_minor_auto_tuning_in_longitudinal & not
   applying_dac_auto_tuning_in_longitudinal & not
   applying_major_auto_tuning_in_lateral & not
   applying_minor_auto_tuning_in_lateral & not
   applying_dac_auto_tuning_in_lateral & not
   applying_major_auto_tuning_in_directional & not
   applying_minor_auto_tuning_in_directional & not
   applying_dac_auto_tuning_in_directional <-
+applying_minor_auto_tuning_in_lateral;
+applying_minor_auto_tuning_in_directional;
+applying_minor_auto_tuning_in_longitudinal;
if (requiring_revalidation_plus_delay) { ~
   requiring_revalidation_plus_delay;
invoke(subsystem2,runRepeated,
   validate_model_in_frequency_domain_plus_delay_v2,["
   State_ndi"],["Update_state_of_agent"]);
invoke(triggering_subsystem,runOnce,
   set_speed_command_back_to_required_airspeed,[],["
   NoTriggerFunc"]);
+applied_revalidation_plus_delay;
}; if (applying_ndi_control) { -applying_ndi_control;
```

```
invoke(control_subsystem,stopRepeated,apply_ndi_control,["
    Input_command","State_ndi","State_ndi_m"],["
    Control_surface_input"]);
invoke(control_subsystem,runOnce,
    set_initial_parameter_for_direct_adaptive_control,["
    State_ndi"],["Control_surface_input"]);
invoke(control_subsystem,runRepeated,apply_hac_control,["
    Input_command","State_ndi","State_ndi_m"],["
    Control_surface_input"]);
+applying_hac_control;
}; if (applying_hac_control) { invoke(triggering_subsystem,
    runOnce,reset_adaptive_gain_adjustment,[],["NoTriggerFunc
    "]);
invoke(subsystem3,runOnce,
    set_initial_parameter_for_control_compensation_in_bad_control_condition
    ,[],["Theta"]);
invoke(subsystem3,runRepeated,
    compensate_aircraft_parameter_v2_for_bad_control_performance
    ,["State_ndi"],["Theta"]);
invoke(utility_system,stopRepeated,monitor_aircraft_status,[
    "Altitude"],["NoUtilityFucn"]);
invoke(utility_system,runOnce,
    set_terminated_time_of_thirty_second,[],["NoUtilityFunc"
    ]);
invoke(utility_system,runRepeated,reckon_time,[],["
    NoUtilityFunc"]);
+apply_hac_control;
+
    applying_minor_auto_tuning_with_hac_for_bad_control_performance
    ;
```

```
}; +
    applied_hac_with_compensation_method_in_bad_control_performance
     .
// executable plan :
+termination_of_timer :
applying_minor_auto_tuning_with_hac_for_bad_control_performance
     <-
invoke(subsystem3,stopRepeated,
    compensate_aircraft_parameter_v2_for_bad_control_performance
    ,["State_ndi"],["Theta"]);
invoke(utility_system,stopRepeated,reckon_time,[],["
    NoUtilityFunc"]);
invoke(utility_system,runRepeated,monitor_aircraft_status,["
    Altitude"],["NoUtilityFucn"]);
~applying_minor_auto_tuning_in_lateral;
~applying_minor_auto_tuning_in_directional;
~applying_minor_auto_tuning_in_longitudinal;
~
    applying_minor_auto_tuning_with_hac_for_bad_control_performance
    ;
+applied_minor_auto_tuning_in_lateral;
+applied_minor_auto_tuning_in_longitudinal;
+applied_minor_auto_tuning_in_directional;
+requiring_revalidation .
// executable plan :
+control_via_hac :
~control_via_ndi & ~auto_tuning <-
if (applying_ndi_control) { invoke(control_subsystem,
    stopRepeated,apply_ndi_control,["Input_command","
    State_ndi","State_ndi_m"],["Control_surface_input"]);
```

```
~applying_ndi_control;
}; if (applying_trim_force) { invoke(control_subsystem,
    stopRepeated,apply_trim_force,[],["Control_surface_input"
    ]);
-applying_trim_force;
}; invoke(control_subsystem,runOnce,
    set_initial_parameter_for_hac,["State_ndi"],["
    Control_surface_input"]);
invoke(control_subsystem,runRepeated,apply_hac_control,["
    Input_command","State_ndi","State_ndi_m"],["
    Control_surface_input"]);
+applying_hac_control .
// executable plan :
+compensate_parameter :
~auto_tuning <-
invoke(subsystem3,runOnce,
    set_initial_parameter_of_orthogonal_least_square_for_control_compensati
    ,[],["Theta"]);
invoke(subsystem3,runRepeated,
    compensate_aircraft_parameter_using_orthogonal_least_square_for_ndi
    ,["State_ndi"],["Theta"]) .
// executable plan :
+switch_off_compensate_parameter :
~auto_tuning <-
invoke(subsystem3,stopRepeated,
    compensate_aircraft_parameter_using_orthogonal_least_square_for_ndi
    ,["State_ndi"],["Theta"]) .
// executable plan :
+control_via_ndi :
applying_ndi_control & ~auto_tuning <-
```

```
if (applying_hac_control) { invoke(control_subsystem,
    stopRepeated,apply_hac_control,["Input_command","
    State_ndi","State_ndi_m"],["Control_surface_input"]);
~applying_hac_control;
}; invoke(control_subsystem,runOnce,
    set_initial_parameter_for_ndi,["State_ndi"],["
    Control_surface_input"]);
invoke(control_subsystem,runRepeated,apply_ndi_control,["
    Input_command","State_ndi","State_ndi_m"],["
    Control_surface_input"]);
+applying_ndi_control .
// executable plan :
+validate_parameter :
true <-
invoke(subsystem2,runOnce,
    set_initial_parameter_for_validating_model_in_frequency_domain
    ,[],["State"]);
invoke(subsystem2,runRepeated,
    validate_model_in_frequency_domainv2,["State_ndi"],["
    Update_state_of_model"]) .
// executable plan :
+requiring_revalidation :
true <-
if (applied_revalidation) { invoke(subsystem2,stopRepeated,
    validate_model_in_frequency_domainv2,["State_ndi"],["
    Update_state_of_model"]);
}; if (applied_revalidation_plus_delay) { invoke(subsystem2,
    stopRepeated,
    validate_model_in_frequency_domain_plus_delay_v2,["
    State_ndi"],["Update_state_of_agent"]);
```

```
}; invoke(subsystem2,runOnce,
    set_initial_parameter_for_validating_model_in_frequency_domain
    ,[],["State"]);
invoke(subsystem2,runRepeated,
    validate_model_in_frequency_domain_plus_delay_v2,["
    State_ndi"],["Update_state_of_model"]);
invoke(utility_system,runRepeated,monitor_aircraft_status,["
    Altitude"],["NoUtilityFucn"]);
~applied_revalidation_plus_delay;
+applied_revalidation;
~requiring_revalidation .
// executable plan :
+requiring_revalidation_plus_delay :
true <-
if (applied_revalidation) { invoke(subsystem2,stopRepeated,
    validate_model_in_frequency_domainv2,["State_ndi"],["
    Update_state_of_model"]);
}; if (applied_revalidation_plus_delay) { invoke(subsystem2,
    stopRepeated,
    validate_model_in_frequency_domain_plus_delay_v2,["
    State_ndi"],["Update_state_of_model"]);
}; invoke(subsystem2,runOnce,
    set_initial_parameter_for_validating_model_in_frequency_domain
    ,[],["State"]);
invoke(utility_system,runRepeated,monitor_aircraft_status,["
    Altitude"],["NoUtilityFucn"]) .
// executable plan :
+altitude_in_bound :
requiring_revalidation_plus_delay <-
~requiring_revalidation_plus_delay;
```

```
invoke(subsystem2,runRepeated,
    validate_model_in_frequency_domain_plus_delay_v2,["
    State_ndi"],["Update_state_of_model"]);
invoke(triggering_subsystem,runOnce,
    set_speed_command_back_to_required_airspeed,[],["
    NoTriggerFunc"]);
+applied_revalidation_plus_delay .
// executable plan :
+termination_of_timer :
applying_major_auto_tuning_with_hac_in_longitudinal <-
invoke(subsystem3,stopRepeated,
    estimate_new_aircraft_aerodynamic_parameter_for_longitudinal
    ,["State_ndi"],["Theta"]);
invoke(control_subsystem,stopRepeated,
    apply_3_2_1_1_for_pitch_force,[],["Force"]);
invoke(control_subsystem,runOnce,
    update_new_parameter_from_reidentification_in_longitudinal
    ,[],["Control_surface_input"]);
invoke(control_subsystem,runOnce,
    set_initial_parameter_for_hac,["State_ndi"],["
    Control_surface_input"]);
invoke(control_subsystem,runRepeated,apply_hac_control,["
    Input_command","State_ndi","State_ndi_m"],["
    Control_surface_input"]);
invoke(utility_system,stopRepeated,reckon_time,[],["
    NoUtilityFunc"]);
~applying_major_auto_tuning_in_longitudinal;
~applying_major_auto_tuning_with_hac_in_longitudinal;
~applied_dac_auto_tuning_in_longitudinal;
~applied_minor_auto_tuning_in_longitudinal;
```

```
+applying_hac_control;
+requiring_revalidation_plus_delay .
// executable plan :
+termination_of_timer :
applying_major_auto_tuning_with_hac_in_lateral <-
invoke(subsystem3,stopRepeated,
    estimate_new_aircraft_aerodynamic_parameters_for_lateral
    ,["State_ndi"],["Theta"]);
invoke(control_subsystem,stopRepeated,
    apply_3_2_1_1_for_lateral_and_directional_force,[],["
    Force"]);
invoke(control_subsystem,runOnce,
    update_new_parameter_from_reidentificaiton_in_lateral
    ,[],["Control_surface_input"]);
invoke(control_subsystem,runOnce,
    set_initial_parameter_for_hac,["State_ndi"],["
    Control_surface_input"]);
invoke(control_subsystem,runRepeated,apply_hac_control,["
    Input_command","State_ndi","State_ndi_m"],["
    Control_surface_input"]);
invoke(utility_system,stopRepeated,reckon_time,[],["
    NoUtilityFunc"]);
~applying_major_auto_tuning_in_lateral;
~applying_major_auto_tuning_with_hac_in_lateral;
~applied_dac_auto_tuning_in_lateral;
~applied_minor_auto_tuning_in_lateral;
+applying_hac_control;
+requiring_revalidation_plus_delay .
// executable plan :
+termination_of_timer :
```

```
applying_major_auto_tuning_with_hac_in_directional <-
invoke(subsystem3,stopRepeated,
    estimate_new_aircraft_aerodynamic_parameter_for_directional
    ,["State_ndi"],["Theta"]);
invoke(control_subsystem,stopRepeated,
    apply_3_2_1_1_for_lateral_and_directional_force,[],["
    Force"]);
invoke(control_subsystem,runOnce,
    update_new_parameter_from_reidentification_in_directional
    ,[],["Control_surface_input"]);
invoke(control_subsystem,runOnce,
    set_initial_parameter_for_hac,["State_ndi"],["
    Control_surface_input"]);
invoke(control_subsystem,runRepeated,apply_hac_control,["
    Input_command","State_ndi","State_ndi_m"],["
    Control_surface_input"]);
invoke(utility_system,stopRepeated,reckon_time,[],["
    NoUtilityFunc"]);
~applying_major_auto_tuning_in_directional;
~applying_major_auto_tuning_with_hac_in_directional;
~applied_dac_auto_tuning_in_directional;
~applied_minor_auto_tuning_in_directional;
+applying_hac_control;
+requiring_revalidation_plus_delay .
// executable plan :
+termination_of_timer :
applying_major_auto_tuning_with_hac_in_coupling_axes <-
invoke(subsystem3,stopRepeated,
    estimate_new_aircraft_aerodynamic_parameters_for_coupling
    ,["State_ndi"],["Theta"]);
```

```
invoke(control_subsystem,stopRepeated,
    apply_3_2_1_1_for_lateral_and_directional_force,[],["
    Force"]);
invoke(control_subsystem,runOnce,
    update_new_parameter_from_reidentification_in_coupling
    ,[],["Control_surface_input"]);
invoke(control_subsystem,runOnce,
    set_initial_parameter_for_hac,["State_ndi"],["
    Control_surface_input"]);
invoke(control_subsystem,runRepeated,apply_hac_control,["
    Input_command","State_ndi","State_ndi_m"],["
    Control_surface_input"]);
invoke(utility_system,stopRepeated,reckon_time,[],["
    NoUtilityFunc"]);
~applying_major_auto_tuning_in_directional;
~applying_major_auto_tuning_in_lateral;
~applying_major_auto_tuning_with_hac_in_coupling_axes;
~applied_dac_auto_tuning_in_lateral;
~applied_dac_auto_tuning_in_directional;
~applied_minor_auto_tuning_in_lateral;
~applied_minor_auto_tuning_in_directional;
+applying_hac_control;
+requiring_revalidation_plus_delay .
// executable plan :
+requiring_major_adjustment_in_longitudinal :
auto_tuning & aircraft_in_trim_status & not
    applying_major_auto_tuning_in_longitudinal & not
    applying_minor_auto_tuning_in_longitudinal & not
    applying_dac_auto_tuning_in_longitudinal & not
    applying_major_auto_tuning_in_lateral & not
```

```
    applying_minor_auto_tuning_in_lateral & not
    applying_dac_auto_tuning_in_lateral & not
    applying_major_auto_tuning_in_directional & not
    applying_minor_auto_tuning_in_directional & not
    applying_dac_auto_tuning_in_directional <-
+applying_major_auto_tuning_in_longitudinal;
if (applying_hac_control) { invoke(utility_system,
    stopRepeated,monitor_aircraft_status,["Altitude"],["
    NoUtilityFucn"]);
invoke(utility_system,runOnce,
    set_speed_command_to_cruise_speed,[],["NoUtilityFunc"]);
invoke(control_subsystem,stopRepeated,apply_hac_control,["
    Input_command","State_ndi","State_ndi_m"],["
    Control_surface_input"]);
invoke(utility_system,runOnce,
    set_terminated_time_of_ten_second,[],["NoUtilityFunc"]);
invoke(control_subsystem,runOnce,
    set_initial_value_for_3_2_1_1_input_in_longitudinal,[],["
    Force"]);
invoke(subsystem3,runOnce,
    set_initial_value_before_reidentification_in_longitudinal
    ,[],["Theta"]);
invoke(utility_system,runRepeated,reckon_time,[],["
    NoUtilityFunc"]);
invoke(control_subsystem,runRepeated,
    apply_3_2_1_1_for_pitch_force,[],["Force"]);
invoke(subsystem3,runRepeated,
    estimate_new_aircraft_aerodynamic_parameter_for_longitudinal
    ,["State_ndi"],["Theta"]);
+applying_major_auto_tuning_with_hac_in_longitudinal;
```

```
}; if (applying_hac_control) { invoke(utility_system,
    stopRepeated,monitor_aircraft_status,["Altitude"],["
    NoUtilityFucn"]);
invoke(utility_system,runOnce,
    set_speed_command_to_cruise_speed,[],["NoUtilityFunc"]);
invoke(control_subsystem,stopRepeated,apply_ndi_control,["
    Input_command","State_ndi","State_ndi_m"],["
    Control_surface_input"]);
invoke(utility_system,runOnce,
    set_terminated_time_of_ten_second,[],["NoUtilityFunc"]);
invoke(control_subsystem,runOnce,
    set_initial_value_for_3_2_1_1_input_in_longitudinal,[],["
    Force"]);
invoke(subsystem3,runOnce,
    set_initial_value_before_reidentification_in_longitudinal
    ,[],["Theta"]);
invoke(utility_system,runRepeated,reckon_time,[],["
    NoUtilityFunc"]);
invoke(control_subsystem,runRepeated,
    apply_3_2_1_1_for_pitch_force,[],["Force"]);
invoke(subsystem3,runRepeated,
    estimate_new_aircraft_aerodynamic_parameter_for_longitudinal
    ,["State_ndi"],["Theta"]);
+applying_major_auto_tuning_with_ndi_in_longitudinal;
}; +waiting_termination .
// executable plan :
+aircraft_in_trim_status :
auto_tuning & requiring_major_adjustment_in_longitudinal &
    not applying_major_auto_tuning_in_longitudinal & not
    applying_minor_auto_tuning_in_longitudinal & not
```

```
    applying_dac_auto_tuning_in_longitudinal & not
    applying_major_auto_tuning_in_lateral & not
    applying_minor_auto_tuning_in_lateral & not
    applying_dac_auto_tuning_in_lateral & not
    applying_major_auto_tuning_in_directional & not
    applying_minor_auto_tuning_in_directional & not
    applying_dac_auto_tuning_in_directional <-
+applying_major_auto_tuning_in_longitudinal;
if (applying_hac_control) { invoke(utility_system,
    stopRepeated,monitor_aircraft_status,["Altitude"],["
    NoUtilityFucn"]);
invoke(utility_system,runOnce,
    set_speed_command_to_cruise_speed,[],["NoUtilityFunc"]);
invoke(control_subsystem,stopRepeated,apply_hac_control,["
    Input_command","State_ndi","State_ndi_m"],["
    Control_surface_input"]);
invoke(utility_system,runOnce,
    set_terminated_time_of_ten_second,[],["NoUtilityFunc"]);
invoke(control_subsystem,runOnce,
    set_initial_value_for_3_2_1_1_input_in_longitudinal,[],["
    Force"]);
invoke(subsystem3,runOnce,
    set_initial_value_before_reidentification_in_longitudinal
    ,[],["Theta"]);
invoke(utility_system,runRepeated,reckon_time,[],["
    NoUtilityFunc"]);
invoke(control_subsystem,runRepeated,
    apply_3_2_1_1_for_pitch_force,[],["Force"]);
invoke(subsystem3,runRepeated,
    estimate_new_aircraft_aerodynamic_parameter_for_longitudinal
```

```
    ,["State_ndi"],["Theta"]);
+applying_major_auto_tuning_with_hac_in_longitudinal;
}; if (applying_hac_control) { invoke(utility_system,
    stopRepeated,monitor_aircraft_status,["Altitude"],["
    NoUtilityFucn"]);
invoke(utility_system,runOnce,
    set_speed_command_to_cruise_speed,[],["NoUtilityFunc"]);
invoke(control_subsystem,stopRepeated,apply_ndi_control,["
    Input_command","State_ndi","State_ndi_m"],["
    Control_surface_input"]);
invoke(utility_system,runOnce,
    set_terminated_time_of_ten_second,[],["NoUtilityFunc"]);
invoke(control_subsystem,runOnce,
    set_initial_value_for_3_2_1_1_input_in_longitudinal,[],["
    Force"]);
invoke(subsystem3,runOnce,
    set_initial_value_before_reidentification_in_longitudinal
    ,[],["Theta"]);
invoke(utility_system,runRepeated,reckon_time,[],["
    NoUtilityFunc"]);
invoke(control_subsystem,runRepeated,
    apply_3_2_1_1_for_pitch_force,[],["Force"]);
invoke(subsystem3,runRepeated,
    estimate_new_aircraft_aerodynamic_parameter_for_longitudinal
    ,["State_ndi"],["Theta"]);
+applying_major_auto_tuning_with_ndi_in_longitudinal;
}; +waiting_termination .
// executable plan :
+aircraft_in_trim_status :
auto_tuning & requiring_major_adjustment_in_lateral & not
```

```
    applying_major_auto_tuning_in_longitudinal & not
    applying_minor_auto_tuning_in_longitudinal & not
    applying_dac_auto_tuning_in_longitudinal & not
    applying_major_auto_tuning_in_lateral & not
    applying_minor_auto_tuning_in_lateral & not
    applying_dac_auto_tuning_in_lateral & not
    applying_major_auto_tuning_in_directional & not
    applying_minor_auto_tuning_in_directional & not
    applying_dac_auto_tuning_in_directional <-
+applying_major_auto_tuning_in_lateral;
if (applying_hac_control) { invoke(utility_system,
    stopRepeated,monitor_aircraft_status,["Altitude"],["
    NoUtilityFucn"]);
invoke(utility_system,runOnce,
    set_speed_command_to_cruise_speed,[],["NoUtilityFunc"]);
invoke(control_subsystem,stopRepeated,apply_hac_control,["
    Input_command","State_ndi","State_ndi_m"],["
    Control_surface_input"]);
invoke(utility_system,runOnce,
    set_terminated_time_of_twelve_second,[],["NoUtilityFunc"
    ]);
invoke(control_subsystem,runOnce,
    set_initial_value_for_3_2_1_1_input_in_lateral_and_directional
    ,["State_ndi"],["Force"]);
invoke(subsystem3,runOnce,
    set_initial_value_before_reidentification_in_lateral,[],[
    "Theta"]);
invoke(utility_system,runRepeated,reckon_time,[],["
    NoUtilityFunc"]);
invoke(control_subsystem,runRepeated,
```

```
    apply_3_2_1_1_for_lateral_and_directional_force,[],["
    Force"]);
invoke(subsystem3,runRepeated,
    estimate_new_aircraft_aerodynamic_parameters_for_lateral
    ,["State_ndi"],["Theta"]);
+applying_major_auto_tuning_with_hac_in_lateral;
}; +waiting_termination .
// executable plan :
+aircraft_in_trim_status :
auto_tuning & requiring_major_adjustment_in_directional &
    not applying_major_auto_tuning_in_longitudinal & not
    applying_minor_auto_tuning_in_longitudinal & not
    applying_dac_auto_tuning_in_longitudinal & not
    applying_major_auto_tuning_in_lateral & not
    applying_minor_auto_tuning_in_lateral & not
    applying_dac_auto_tuning_in_lateral & not
    applying_major_auto_tuning_in_directional & not
    applying_minor_auto_tuning_in_directional & not
    applying_dac_auto_tuning_in_directional <-
+applying_major_auto_tuning_in_directional;
if (applying_hac_control) { invoke(utility_system,
    stopRepeated,monitor_aircraft_status,["Altitude"],["
    NoUtilityFucn"]);
invoke(utility_system,runOnce,
    set_speed_command_to_cruise_speed,[],["NoUtilityFunc"]);
invoke(control_subsystem,stopRepeated,apply_hac_control,["
    Input_command","State_ndi","State_ndi_m"],["
    Control_surface_input"]);
invoke(utility_system,runOnce,
    set_terminated_time_of_twelve_second,[],["NoUtilityFunc"
```

```
    ]);
invoke(control_subsystem,runOnce,
    set_initial_value_for_3_2_1_1_input_in_lateral_and_directional
    ,["State_ndi"],["Force"]);
invoke(subsystem3,runOnce,
    set_initial_value_before_reidentification_in_directional
    ,[],["Theta"]);
invoke(utility_system,runRepeated,reckon_time,[],["
    NoUtilityFunc"]);
invoke(control_subsystem,runRepeated,
    apply_3_2_1_1_for_lateral_and_directional_force,[],["
    Force"]);
invoke(subsystem3,runRepeated,
    estimate_new_aircraft_aerodynamic_parameter_for_directional
    ,["State_ndi"],["Theta"]);
+applying_major_auto_tuning_with_hac_in_directional;
}; +waiting_termination .
// executable plan :
+aircraft_in_trim_status :
auto_tuning & applied_minor_auto_tuning_in_longitudinal &
    not applying_major_auto_tuning_in_longitudinal & not
    applying_minor_auto_tuning_in_longitudinal & not
    applying_dac_auto_tuning_in_longitudinal & not
    applying_major_auto_tuning_in_lateral & not
    applying_minor_auto_tuning_in_lateral & not
    applying_dac_auto_tuning_in_lateral & not
    applying_major_auto_tuning_in_directional & not
    applying_minor_auto_tuning_in_directional & not
    applying_dac_auto_tuning_in_directional <-
+applying_major_auto_tuning_in_longitudinal;
```

```
if (applying_hac_control) { invoke(utility_system,
    stopRepeated,monitor_aircraft_status,["Altitude"],["
    NoUtilityFucn"]);
invoke(utility_system,runOnce,
    set_speed_command_to_cruise_speed,[],["NoUtilityFunc"]);
invoke(control_subsystem,stopRepeated,apply_hac_control,["
    Input_command","State_ndi","State_ndi_m"],["
    Control_surface_input"]);
invoke(utility_system,runOnce,
    set_terminated_time_of_ten_second,[],["NoUtilityFunc"]);
invoke(control_subsystem,runOnce,
    set_initial_value_for_3_2_1_1_input_in_longitudinal,[],["
    Force"]);
invoke(subsystem3,runOnce,
    set_initial_value_before_reidentification_in_longitudinal
    ,[],["Theta"]);
invoke(utility_system,runRepeated,reckon_time,[],["
    NoUtilityFunc"]);
invoke(control_subsystem,runRepeated,
    apply_3_2_1_1_for_pitch_force,[],["Force"]);
invoke(subsystem3,runRepeated,
    estimate_new_aircraft_aerodynamic_parameter_for_longitudinal
    ,["State_ndi"],["Theta"]);
+applying_major_auto_tuning_with_hac_in_longitudinal;
}; +waiting_termination .
// executable plan :
+aircraft_in_trim_status :
auto_tuning & applied_minor_auto_tuning_in_lateral & not
    applied_minor_auto_tuning_in_directional & not
    applying_major_auto_tuning_in_longitudinal & not
```

```
    applying_minor_auto_tuning_in_longitudinal & not
    applying_dac_auto_tuning_in_longitudinal & not
    applying_major_auto_tuning_in_lateral & not
    applying_minor_auto_tuning_in_lateral & not
    applying_dac_auto_tuning_in_lateral & not
    applying_major_auto_tuning_in_directional & not
    applying_minor_auto_tuning_in_directional & not
    applying_dac_auto_tuning_in_directional <-
+applying_major_auto_tuning_in_lateral;
if (applying_hac_control) { invoke(utility_system,
    stopRepeated,monitor_aircraft_status,["Altitude"],["
    NoUtilityFucn"]);
invoke(utility_system,runOnce,
    set_speed_command_to_cruise_speed,[],["NoUtilityFunc"]);
invoke(control_subsystem,stopRepeated,apply_hac_control,["
    Input_command","State_ndi","State_ndi_m"],["
    Control_surface_input"]);
invoke(utility_system,runOnce,
    set_terminated_time_of_twelve_second,[],["NoUtilityFunc"
    ]);
invoke(control_subsystem,runOnce,
    set_initial_value_for_3_2_1_1_input_in_lateral_and_directional
    ,["State_ndi"],["Force"]);
invoke(subsystem3,runOnce,
    set_initial_value_before_reidentification_in_lateral,[],[
    "Theta"]);
invoke(utility_system,runRepeated,reckon_time,[],["
    NoUtilityFunc"]);
invoke(control_subsystem,runRepeated,
    apply_3_2_1_1_for_lateral_and_directional_force,[],["
```

```
    Force"]);
invoke(subsystem3,runRepeated,
    estimate_new_aircraft_aerodynamic_parameters_for_lateral
    ,["State_ndi"],["Theta"]);
+applying_major_auto_tuning_with_hac_in_lateral;
}; +waiting_termination .
// executable plan :
+aircraft_in_trim_status :
auto_tuning & applied_minor_auto_tuning_in_directional & not
     applied_minor_auto_tuning_in_lateral & not
    applying_major_auto_tuning_in_longitudinal & not
    applying_minor_auto_tuning_in_longitudinal & not
    applying_dac_auto_tuning_in_longitudinal & not
    applying_major_auto_tuning_in_lateral & not
    applying_minor_auto_tuning_in_lateral & not
    applying_dac_auto_tuning_in_lateral & not
    applying_major_auto_tuning_in_directional & not
    applying_minor_auto_tuning_in_directional & not
    applying_dac_auto_tuning_in_directional <-
+applying_major_auto_tuning_in_directional;
if (applying_hac_control) { invoke(utility_system,
    stopRepeated,monitor_aircraft_status,["Altitude"],["
    NoUtilityFucn"]);
invoke(utility_system,runOnce,
    set_speed_command_to_cruise_speed,[],["NoUtilityFunc"]);
invoke(control_subsystem,stopRepeated,apply_hac_control,["
    Input_command","State_ndi","State_ndi_m"],["
    Control_surface_input"]);
invoke(utility_system,runOnce,
    set_terminated_time_of_twelve_second,[],["NoUtilityFunc"
```

```
    ]);
invoke(control_subsystem,runOnce,
    set_initial_value_for_3_2_1_1_input_in_lateral_and_directional
    ,["State_ndi"],["Force"]);
invoke(subsystem3,runOnce,
    set_initial_value_before_reidentification_in_directional
    ,[],["Theta"]);
invoke(utility_system,runRepeated,reckon_time,[],["
    NoUtilityFunc"]);
invoke(control_subsystem,runRepeated,
    apply_3_2_1_1_for_lateral_and_directional_force,[],["
    Force"]);
invoke(subsystem3,runRepeated,
    estimate_new_aircraft_aerodynamic_parameter_for_directional
    ,["State_ndi"],["Theta"]);
+applying_major_auto_tuning_with_hac_in_directional;
}; +waiting_termination .
// executable plan :
+aircraft_in_trim_status :
auto_tuning & applied_minor_auto_tuning_in_directional &
    applied_minor_auto_tuning_in_lateral & not
    applying_major_auto_tuning_in_longitudinal & not
    applying_minor_auto_tuning_in_longitudinal & not
    applying_dac_auto_tuning_in_longitudinal & not
    applying_major_auto_tuning_in_lateral & not
    applying_minor_auto_tuning_in_lateral & not
    applying_dac_auto_tuning_in_lateral & not
    applying_major_auto_tuning_in_directional & not
    applying_minor_auto_tuning_in_directional & not
    applying_dac_auto_tuning_in_directional <-
```

```
+applying_major_auto_tuning_in_lateral;
+applying_major_auto_tuning_in_directional;
if (applying_hac_control) { invoke(utility_system,
    stopRepeated,monitor_aircraft_status,["Altitude"],["
    NoUtilityFucn"]);
invoke(utility_system,runOnce,
    set_speed_command_to_cruise_speed,[],["NoUtilityFunc"]);
invoke(control_subsystem,stopRepeated,apply_hac_control,["
    Input_command","State_ndi","State_ndi_m"],["
    Control_surface_input"]);
invoke(utility_system,runOnce,
    set_terminated_time_of_twelve_second,[],["NoUtilityFunc"
    ]);
invoke(control_subsystem,runOnce,
    set_initial_value_for_3_2_1_1_input_in_lateral_and_directional
    ,["State_ndi"],["Force"]);
invoke(subsystem3,runOnce,
    set_initial_value_before_reidentification_for_coupling
    ,[],["Theta"]);
invoke(utility_system,runRepeated,reckon_time,[],["
    NoUtilityFunc"]);
invoke(control_subsystem,runRepeated,
    apply_3_2_1_1_for_lateral_and_directional_force,[],["
    Force"]);
invoke(subsystem3,runRepeated,
    estimate_new_aircraft_aerodynamic_parameters_for_coupling
    ,["State_ndi"],["Theta"]);
+applying_major_auto_tuning_with_hac_in_coupling_axes;
}; +waiting_termination .
// executable plan :
```

```
+termination_of_timer :
applying_minor_auto_tuning_with_hac_in_longitudinal <-
invoke(subsystem3,stopRepeated,
    compensate_aircraft_parameter_v2_for_longitudinal_axes,["
    State_ndi"],["Theta"]);
invoke(utility_system,stopRepeated,reckon_time,[],["
    NoUtilityFunc"]);
~applying_minor_auto_tuning_in_longitudinal;
~applying_minor_auto_tuning_with_hac_in_longitudinal;
+applying_hac_control;
+applied_minor_auto_tuning_in_longitudinal;
+requiring_revalidation .
// executable plan :
+termination_of_timer :
applying_minor_auto_tuning_with_ndi_in_longitudinal <-
invoke(subsystem3,stopRepeated,
    compensate_aircraft_parameter_v2_for_longitudinal_axes,["
    State_ndi"],["Theta"]);
invoke(utility_system,stopRepeated,reckon_time,[],["
    NoUtilityFunc"]);
~applying_minor_auto_tuning_in_longitudinal;
~applying_minor_auto_tuning_with_ndi_in_longitudinal;
+applying_ndi_control;
+applied_minor_auto_tuning_in_longitudinal;
+requiring_revalidation .
// executable plan :
+termination_of_timer :
applying_minor_auto_tuning_with_hac_in_lateral <-
invoke(subsystem3,stopRepeated,
    compensate_aircraft_parameter_v2_for_lateral_axes,["
```

```
    State_ndi"],["Theta"]);
invoke(utility_system,stopRepeated,reckon_time,[],["
    NoUtilityFunc"]);
~applying_minor_auto_tuning_in_lateral;
~applying_minor_auto_tuning_with_hac_in_lateral;
+applying_hac_control;
+applied_minor_auto_tuning_in_lateral;
+requiring_revalidation .
// executable plan :
+termination_of_timer :
applying_minor_auto_tuning_with_hac_in_directional <-
invoke(subsystem3,stopRepeated,
    compensate_aircraft_parameter_v2_for_directional_axes,["
    State_ndi"],["Theta"]);
invoke(utility_system,stopRepeated,reckon_time,[],["
    NoUtilityFunc"]);
~applying_minor_auto_tuning_in_directional;
~applying_minor_auto_tuning_with_hac_in_directional;
+applying_hac_control;
+applied_minor_auto_tuning_in_directional;
+requiring_revalidation .
// executable plan :
+auto_tuning :
requiring_minor_adjustment_in_longitudinal &
    applied_dac_auto_tuning_in_longitudinal & not
    applying_major_auto_tuning_in_longitudinal & not
    applying_minor_auto_tuning_in_longitudinal & not
    applying_dac_auto_tuning_in_longitudinal & not
    applying_major_auto_tuning_in_lateral & not
    applying_minor_auto_tuning_in_lateral & not
```

```
    applying_dac_auto_tuning_in_lateral & not
    applying_major_auto_tuning_in_directional & not
    applying_minor_auto_tuning_in_directional & not
    applying_dac_auto_tuning_in_directional <-
+applying_minor_auto_tuning_in_longitudinal;
if (applying_hac_control) { invoke(subsystem3,runOnce,
    set_initial_parameter_for_control_compensation_in_longitudinal
    ,[],["Theta"]);
invoke(subsystem3,runRepeated,
    compensate_aircraft_parameter_v2_for_longitudinal_axes,["
    State_ndi"],["Theta"]);
invoke(utility_system,stopRepeated,monitor_aircraft_status,[
    "Altitude"],["NoUtilityFucn"]);
invoke(utility_system,runOnce,
    set_terminated_time_of_sixty_second,[],["NoUtilityFunc"])
    ;
invoke(utility_system,runRepeated,reckon_time,[],["
    NoUtilityFunc"]);
+apply_hac_control;
+applying_minor_auto_tuning_with_hac_in_longitudinal;
}; if (applying_ndi_control) { invoke(subsystem3,runOnce,
    set_initial_parameter_for_control_compensation_in_longitudinal
    ,[],["Theta"]);
invoke(subsystem3,runRepeated,
    compensate_aircraft_parameter_v2_for_longitudinal_axes,["
    State_ndi"],["Theta"]);
invoke(utility_system,stopRepeated,monitor_aircraft_status,[
    "Altitude"],["NoUtilityFucn"]);
invoke(utility_system,runOnce,
    set_terminated_time_of_sixty_second,[],["NoUtilityFunc"])
```

```
    ;
invoke(utility_system,runRepeated,reckon_time,[],["
    NoUtilityFunc"]);
+apply_ndi_control;
+applying_minor_auto_tuning_with_ndi_in_longitudinal;
}; +waiting_termination .
// executable plan :
+requiring_minor_adjustment_in_lateral :
auto_tuning & applied_dac_auto_tuning_in_lateral & not
    applied_minor_auto_tuning_in_lateral & not
    applying_major_auto_tuning_in_longitudinal & not
    applying_minor_auto_tuning_in_longitudinal & not
    applying_dac_auto_tuning_in_longitudinal & not
    applying_major_auto_tuning_in_lateral & not
    applying_minor_auto_tuning_in_lateral & not
    applying_dac_auto_tuning_in_lateral & not
    applying_major_auto_tuning_in_directional & not
    applying_minor_auto_tuning_in_directional & not
    applying_dac_auto_tuning_in_directional <-
+applying_minor_auto_tuning_in_lateral;
if (applying_hac_control) { invoke(triggering_subsystem,
    runOnce,reset_adaptive_gain_adjustment,[],["NoTriggerFunc
    "]);
invoke(subsystem3,runOnce,
    set_initial_parameter_for_control_compensation_in_lateral
    ,[],["Theta"]);
invoke(subsystem3,runRepeated,
    compensate_aircraft_parameter_v2_for_lateral_axes,["
    State_ndi"],["Theta"]);
invoke(utility_system,stopRepeated,monitor_aircraft_status,[
```

```
    "Altitude"],["NoUtilityFucn"]);
invoke(utility_system,runOnce,
    set_terminated_time_of_thirty_second,[],["NoUtilityFunc"
    ]);
invoke(utility_system,runRepeated,reckon_time,[],["
    NoUtilityFunc"]);
+apply_hac_control;
+applying_minor_auto_tuning_with_hac_in_lateral;
}; if (applying_ndi_control) { invoke(subsystem3,runOnce,
    set_initial_parameter_for_control_compensation_in_lateral
    ,[],["Theta"]);
invoke(subsystem3,runRepeated,
    compensate_aircraft_parameter_v2_for_lateral_axes,["
    State_ndi"],["Theta"]);
invoke(utility_system,stopRepeated,monitor_aircraft_status,[
    "Altitude"],["NoUtilityFucn"]);
invoke(utility_system,runOnce,
    set_terminated_time_of_thirty_second,[],["NoUtilityFunc"
    ]);
invoke(utility_system,runRepeated,reckon_time,[],["
    NoUtilityFunc"]);
+apply_ndi_control;
+applying_minor_auto_tuning_with_ndi_in_lateral;
}; +waiting_termination .
// executable plan :
+requiring_minor_adjustment_in_directional :
auto_tuning & applied_dac_auto_tuning_in_directional & not
    applied_minor_auto_tuning_in_directional & not
    applying_major_auto_tuning_in_longitudinal & not
    applying_minor_auto_tuning_in_longitudinal & not
```

```
    applying_dac_auto_tuning_in_longitudinal & not
    applying_major_auto_tuning_in_lateral & not
    applying_minor_auto_tuning_in_lateral & not
    applying_dac_auto_tuning_in_lateral & not
    applying_major_auto_tuning_in_directional & not
    applying_minor_auto_tuning_in_directional & not
    applying_dac_auto_tuning_in_directional <-
+applying_minor_auto_tuning_in_directional;
if (applying_hac_control) { invoke(triggering_subsystem,
    runOnce,reset_adaptive_gain_adjustment,[],["NoTriggerFunc
    "]);
invoke(subsystem3,runOnce,
    set_initial_parameter_for_control_compensation_in_directional
    ,[],["Theta"]);
invoke(subsystem3,runRepeated,
    compensate_aircraft_parameter_v2_for_directional_axes,["
    State_ndi"],["Theta"]);
invoke(utility_system,stopRepeated,monitor_aircraft_status,[
    "Altitude"],["NoUtilityFucn"]);
invoke(utility_system,runOnce,
    set_terminated_time_of_thirty_second,[],["NoUtilityFunc"
    ]);
invoke(utility_system,runRepeated,reckon_time,[],["
    NoUtilityFunc"]);
+apply_hac_control;
+applying_minor_auto_tuning_with_hac_in_directional;
}; if (applying_ndi_control) { invoke(subsystem3,runOnce,
    set_initial_parameter_for_control_compensation_in_directional
    ,[],["Theta"]);
invoke(subsystem3,runRepeated,
```

```
    compensate_aircraft_parameter_v2_for_directional_axes,["
    State_ndi"],["Theta"]);
invoke(utility_system,stopRepeated,monitor_aircraft_status,[
    "Altitude"],["NoUtilityFucn"]);
invoke(utility_system,runOnce,
    set_terminated_time_of_thirty_second,[],["NoUtilityFunc"
    ]);
invoke(utility_system,runRepeated,reckon_time,[],["
    NoUtilityFunc"]);
+apply_ndi_control;
+applying_minor_auto_tuning_with_ndi_in_directional;
}; +waiting_termination .
// executable plan :
+requiring_minor_adjustment_in_longitudinal :
auto_tuning & applied_dac_auto_tuning_in_longitudinal & not
    applied_minor_auto_tuning_in_longitudinal & not
    applying_major_auto_tuning_in_longitudinal & not
    applying_minor_auto_tuning_in_longitudinal & not
    applying_dac_auto_tuning_in_longitudinal & not
    applying_major_auto_tuning_in_lateral & not
    applying_minor_auto_tuning_in_lateral & not
    applying_dac_auto_tuning_in_lateral & not
    applying_major_auto_tuning_in_directional & not
    applying_minor_auto_tuning_in_directional & not
    applying_dac_auto_tuning_in_directional <-
+applying_minor_auto_tuning_in_longitudinal;
if (applying_hac_control) { invoke(triggering_subsystem,
    runOnce,reset_adaptive_gain_adjustment,[],["NoTriggerFunc
    "]);
invoke(subsystem3,runOnce,
```

```
    set_initial_parameter_for_control_compensation_in_longitudinal
    ,[],["Theta"]);
invoke(subsystem3,runRepeated,
    compensate_aircraft_parameter_v2_for_longitudinal_axes,["
    State_ndi"],["Theta"]);
invoke(utility_system,stopRepeated,monitor_aircraft_status,[
    "Altitude"],["NoUtilityFucn"]);
invoke(utility_system,runOnce,
    set_terminated_time_of_sixty_second,[],["NoUtilityFunc"])
    ;
invoke(utility_system,runRepeated,reckon_time,[],["
    NoUtilityFunc"]);
+apply_hac_control;
+applying_minor_auto_tuning_with_hac_in_longitudinal;
}; if (applying_ndi_control) { invoke(subsystem3,runOnce,
    set_initial_parameter_for_control_compensation_in_longitudinal
    ,[],["Theta"]);
invoke(subsystem3,runRepeated,
    compensate_aircraft_parameter_v2_for_longitudinal_axes,["
    State_ndi"],["Theta"]);
invoke(utility_system,stopRepeated,monitor_aircraft_status,[
    "Altitude"],["NoUtilityFucn"]);
invoke(utility_system,runOnce,
    set_terminated_time_of_sixty_second,[],["NoUtilityFunc"])
    ;
invoke(utility_system,runRepeated,reckon_time,[],["
    NoUtilityFunc"]);
+apply_ndi_control;
+applying_minor_auto_tuning_with_ndi_in_longitudinal;
}; +waiting_termination .
```

```
// executable plan :
+auto_tuning :
requiring_direct_adaptive_in_longitudinal & not
    applied_dac_auto_tuning_in_longitudinal & not
    applying_major_auto_tuning_in_longitudinal & not
    applying_minor_auto_tuning_in_longitudinal & not
    applying_dac_auto_tuning_in_longitudinal & not
    applying_major_auto_tuning_in_lateral & not
    applying_minor_auto_tuning_in_lateral & not
    applying_dac_auto_tuning_in_lateral & not
    applying_major_auto_tuning_in_directional & not
    applying_minor_auto_tuning_in_directional & not
    applying_dac_auto_tuning_in_directional <-
+applying_dac_auto_tuning_in_longitudinal;
if (applying_ndi_control) { invoke(control_subsystem,
    stopRepeated,apply_ndi_control,["Input_command","
    State_ndi","State_ndi_m"],["Control_surface_input"]);
invoke(subsystem3,runOnce,clear_the_compensated_parameter
    ,[],["Theta"]);
invoke(control_subsystem,runOnce,
    set_initial_parameter_for_hac,["State_ndi"],["
    Control_surface_input"]);
invoke(control_subsystem,runRepeated,apply_hac_control,["
    Input_command","State_ndi","State_ndi_m"],["
    Control_surface_input"]);
~applying_ndi_control;
+applying_hac_control;
}; +requiring_revalidation;
~applying_dac_auto_tuning_in_longitudinal;
+applied_dac_auto_tuning_in_longitudinal .
```

```
// executable plan :
+requiring_direct_adaptive_in_lateral :
auto_tuning & not applied_dac_auto_tuning_in_lateral & not
    applying_major_auto_tuning_in_longitudinal & not
    applying_minor_auto_tuning_in_longitudinal & not
    applying_dac_auto_tuning_in_longitudinal & not
    applying_major_auto_tuning_in_lateral & not
    applying_minor_auto_tuning_in_lateral & not
    applying_dac_auto_tuning_in_lateral & not
    applying_major_auto_tuning_in_directional & not
    applying_minor_auto_tuning_in_directional & not
    applying_dac_auto_tuning_in_directional <-
+applying_dac_auto_tuning_in_lateral;
if (applying_ndi_control) { invoke(control_subsystem,
    stopRepeated,apply_ndi_control,["Input_command","
    State_ndi","State_ndi_m"],["Control_surface_input"]);
invoke(subsystem3,runOnce,clear_the_compensated_parameter
    ,[],["Theta"]);
invoke(control_subsystem,runOnce,
    set_initial_parameter_for_hac,["State_ndi"],["
    Control_surface_input"]);
invoke(control_subsystem,runRepeated,apply_hac_control,["
    Input_command","State_ndi","State_ndi_m"],["
    Control_surface_input"]);
~applying_ndi_control;
+applying_hac_control;
}; +requiring_revalidation;
~applying_dac_auto_tuning_in_lateral;
+applied_dac_auto_tuning_in_lateral .
// executable plan :
```

```
+requiring_direct_adaptive_in_directional :
auto_tuning & not applied_dac_auto_tuning_in_directional &
    not applying_major_auto_tuning_in_longitudinal & not
    applying_minor_auto_tuning_in_longitudinal & not
    applying_dac_auto_tuning_in_longitudinal & not
    applying_major_auto_tuning_in_lateral & not
    applying_minor_auto_tuning_in_lateral & not
    applying_dac_auto_tuning_in_lateral & not
    applying_major_auto_tuning_in_directional & not
    applying_minor_auto_tuning_in_directional & not
    applying_dac_auto_tuning_in_directional <-
+applying_dac_auto_tuning_in_directional;
if (applying_ndi_control) { invoke(control_subsystem,
    stopRepeated,apply_ndi_control,["Input_command","
    State_ndi","State_ndi_m"],["Control_surface_input"]);
invoke(subsystem3,runOnce,clear_the_compensated_parameter
    ,[],["Theta"]);
invoke(control_subsystem,runOnce,
    set_initial_parameter_for_hac,["State_ndi"],["
    Control_surface_input"]);
invoke(control_subsystem,runRepeated,apply_hac_control,["
    Input_command","State_ndi","State_ndi_m"],["
    Control_surface_input"]);
~applying_ndi_control;
+applying_hac_control;
}; +requiring_revalidation;
~applying_dac_auto_tuning_in_directional;
+applied_dac_auto_tuning_in_directional .
// executable plan :
+requiring_direct_adaptive_in_longitudinal :
```

```
auto_tuning & not applied_dac_auto_tuning_in_longitudinal &
    not applying_major_auto_tuning_in_longitudinal & not
    applying_minor_auto_tuning_in_longitudinal & not
    applying_dac_auto_tuning_in_longitudinal & not
    applying_major_auto_tuning_in_lateral & not
    applying_minor_auto_tuning_in_lateral & not
    applying_dac_auto_tuning_in_lateral & not
    applying_major_auto_tuning_in_directional & not
    applying_minor_auto_tuning_in_directional & not
    applying_dac_auto_tuning_in_directional <-
+applying_dac_auto_tuning_in_longitudinal;
if (applying_ndi_control) { invoke(control_subsystem,
    stopRepeated,apply_ndi_control,["Input_command","
    State_ndi","State_ndi_m"],["Control_surface_input"]);
invoke(subsystem3,runOnce,clear_the_compensated_parameter
    ,[],["Theta"]);
invoke(control_subsystem,runOnce,
    set_initial_parameter_for_hac,["State_ndi"],["
    Control_surface_input"]);
invoke(control_subsystem,runRepeated,apply_hac_control,["
    Input_command","State_ndi","State_ndi_m"],["
    Control_surface_input"]);
~applying_ndi_control;
+applying_hac_control;
}; +requiring_revalidation;
~applying_dac_auto_tuning_in_longitudinal;
+applied_dac_auto_tuning_in_longitudinal .
// executable plan :
+aircraft_in_trim_status :
auto_tuning & not applying_major_auto_tuning_in_longitudinal
```

```
    & not applying_minor_auto_tuning_in_longitudinal & not
    applying_dac_auto_tuning_in_longitudinal & not
    applying_major_auto_tuning_in_lateral & not
    applying_minor_auto_tuning_in_lateral & not
    applying_dac_auto_tuning_in_lateral & not
    applying_major_auto_tuning_in_directional & not
    applying_minor_auto_tuning_in_directional & not
    applying_dac_auto_tuning_in_directional <-
if (applied_revalidation){ invoke(subsystem2,stopRepeated,
    validate_model_in_frequency_domainv2,["State_ndi"],["
    Update_state_of_model"]);
}; if (applied_revalidation_plus_delay) { invoke(subsystem2,
    stopRepeated,
    validate_model_in_frequency_domain_plus_delay_v2,["
    State_ndi"],["Update_state_of_agent"]);
}; invoke(subsystem2,runOnce,
    set_initial_parameter_for_validating_model_in_frequency_domain
    ,[],["State"]);
invoke(subsystem2,runRepeated,
    validate_model_in_frequency_domainv2,["State_ndi"],["
    Update_state_of_model"]);
~applied_revalidation_plus_delay;
+applied_revalidation;
~applied_dac_auto_tuning_in_longitudinal;
~applied_minor_auto_tuning_in_longitudinal;
~applied_dac_auto_tuning_in_lateral;
~applied_minor_auto_tuning_in_lateral;
~applied_dac_auto_tuning_in_directional;
~applied_minor_auto_tuning_in_directional .
// executable plan :
```

```
+control_via_ndi :
~control_via_hac <-
invoke(control_subsystem,stopRepeated,apply_ndi_control,["
    Input_command","State_ndi","State_ndi_m"],["
    Control_surface_input"]);
~applying_ndi_control .
// executable plan :
+control_via_ndi :
applying_ndi_control <-
invoke(control_subsystem,stopRepeated,apply_ndi_control,["
    Input_command","State_ndi","State_ndi_m"],["
    Control_surface_input"]);
~applying_ndi_control .
// executable plan :
+control_via_ndi :
applying_ndi_control & control_via_hac <-
invoke(control_subsystem,stopRepeated,apply_ndi_control,["
    Input_command","State_ndi","State_ndi_m"],["
    Control_surface_input"]);
~applying_ndi_control;
invoke(control_subsystem,runOnce,
    set_initial_parameter_for_hac,["State_ndi"],["
    Control_surface_input"]);
invoke(control_subsystem,runRepeated,apply_hac_control,["
    Input_command","State_ndi","State_ndi_m"],["
    Control_surface_input"]);
+applying_hac_control .
// executable plan :
+validate_parameter :
true <-
```

```
invoke(subsystem2,stopRepeated,
    validate_model_in_frequency_domainv2,["State_ndi"],["
    Update_state_of_model"]) .
// executable plan :
+auto_tuning :
true <-
~applying_auto_tuning .
```

Bibliography

[1] *FlightGear Flight Simulation.* [Online] Available from http://www.flightgear.org/. xiii, 158, 159, 160

[2] *Tornado: A Vortex Lattice Method Implemented in MATLAB.* [Online]. Available from http://tornado.redhammer.se/index.php. 158

[3] M. J. Allen and R. P. Dibley. Modeling aircraft wing loads from flight data using neural networks. Technical Report NASA/TM-2003-212032, NASA Dryden Flight Research Center, Septempber 2003. 15

[4] S. I. AlSwailem. *Application of Robust Control in Unmanned Vehicle Flight Control System Design.* PhD thesis, College of Aeronautics, Cranfield University, 2004. 11

[5] G. Ambrosino, M. Ariola, U. Ciniglio, F. Corraro, A. Pironti, and M. Virgilio. Algorithms for 3d uav path generation and tracking. In *In Proceeding of 45th IEEE Conference on Decision & Control*, pages pp.5275–5280, San Diego, CA, 2006. 58

[6] P. J. Antsaklis, K. M. Passino, and S. J. Wang. Towards intelligent autonomous control systems : Architecture and fundamental issues. *Journal of Intelligent and Robotic Systems*, **1**:315–342, 1989. 20

[7] P. J. Antsaklis, K. M. Passino, and S. J. Wang. An introduction to autonomous control systems. *Journal and Magazines of Control Systems, IEEE*, **11**[4]:5–13, June 1991. 20

[8] J. P. Appel. *Online System Identification for Fault Tolerant Control of Unmanned Aerial Vehicles*. Master's thesis, Stellenbosch University, 2013. 80

[9] B. A. Bakar. *Autonomous Multi-agent Reconfigurable Control System*. PhD thesis, University of Southampton, March 2013. 26

[10] P Balaguer and R Vilanova. A new frequency dependent approach to model validation. *Frontiers in Adaptive Control*, 2009. Shuang Cong, InTech. 118, 119, 122, 123

[11] J. D. Barton. Fundamentals of small unmanned aircraft flight. *Journal of Johns Hopkins APL Technical Digest*, **31**:pp. 132–149, 2012. 81, 82

[12] M. Bennett. *Development of Technologies for Low-Cost Oceanographic Unmanned Aeronautical Vehicle*. PhD thesis, School of Electronics and Computer Science, 2009. 10

[13] S. Bhattacharyya, D. Cofer, D. J. Musliner, J. Mueller, and E. Engstrom. Certification considerations for adaptive systems. Technical report, NASA Langley Research Center, March 2015. 3, 26

[14] R. H. Bordini, J. F. Hbner, and M. Wooldridge. *Programming Multi-Agent Systems in AgentSpeak using Jason*. Wiley, Chichester, 2007. ix, 3, 69, 70

[15] J. D. Boskovic and N. Knoebel. A comparison study of several adaptive control strategies for resilient flight control. In *AIAA Guidance, Navigation, and Control Conference*, Chicago, Illinois, August 2009. 19

[16] J. T. BoswJohn and P. S. Willliams-Hayes. Flight test results from the nf-15b intelligent flight control system (ifcs) project with adaptation to a simulated stabilator failure. Technical report, NASA Dryden Flight Research Center, Edwards, California, USA, Decempber 2007. 2

[17] J. J. Burken, P. Williams-Hayes, J. T. Kaneshige, and S. J. Stachowiak. Adaptive control using neural network augmentation for a

modified f-15 aircraft. In *Control and Automation, 2006. MED '06. 14th Mediterranean Conference on*, pages 1–6, 2006. 15

[18] A. J. Calise, S. Lee, and M. Sharma. Direct adaptive reconfigurable control of a tailless fighter aircraft. In *Proceeding of the AIAA Guidance, Navigation, and Control Conference*, pages 88–97, 1998. 15

[19] A. J. Calise and R. T. Rysdyk. Adaptive model inversion flight control for tiltrotor aircraft. In *Proceeding of AIAA Guidance, Navigation and Control Conference*, pages 1633–1639, 1997.

[20] A. J. Calise and R. T. Rysdyk. Nonlinear adaptive flight control using neural networks. *Journal of Control Systems, IEEE*, **18**[6]:14–25, December 1998. 15

[21] S. F. Campbell, N. T. Nguyen, J. Kaneshige, and K. Krishnakumar. Parameter estimation for a hybrid adaptive flight controller. In *AIAA Infotech@Aerospace Conference*, Seatle, Washington, April 2009. 18

[22] X. Q. Chen, Q. Ou, D. R. Wong, Y. J. Li, M. Sinclair, and A. Marburg. *Flight Dynamics Modelling and Experimental Validation for Unmanned Aerial Vehicles*. Sate of the Art in Land, Sea, Air, and Collaborative Missions. InTech, 2009. ISBN: 978-953-307-001-8. 80

[23] H. Chitsaz and S. LaValle. Time-optimal paths for a dubins airplane. In *In proceeding of 46th IEEE Conference on Decision and Control*, pages pp.2379–2384, New Orleans, LA, December 2007. 58

[24] A. Cho, Y. S. Kang, B. J. Park, and C. S. Yoo. Airflow angle and wind estimation using gps/ins navigation data and airspeed. In *In Proceeding of 13th International Conference of Control Automation and System (ICCAS)*, Gwangja, Korea, 2013. 81

[25] A. Cho, J. Kim, S. Lee, and C. Kee. Wind estimation and airspeed calibration using a uav with a single-antenna gps receiver and pitot tube. *Journal of IEEE Transactions on Aeorspace and Electronic Systems*, **47**:pp. 109–117, 2011. 81

[26] G. Chowdhary, W. M. DeBusk, and E. N. Johnson. Real-time system identification of a small multi-engine aircraft with structure damage. In *AIAA Infotech@Aerospace 2010*, Atlanta, Georgia, 20-22 April 2010. 14

[27] G. Chowdhary and R. Jategaonkar. Aerodynamic parameter estimation from flight data applying extended and unscented kalman fiter. In *Proceeding of AIAA Atmospheric Flight Mechanics Conference and Exhibit*, Keystone, Colorado, 2006. 13

[28] G. Chowdhary and R. Jategaonkar. Aerodynamic parameter estimation from flight data applying extended and unscented kalman filter. *Journal of Aerospace Science and Technology*, **14**:pp. 106–117, 2010. 84

[29] G. Chowdhary and E. Johnson. Recursive updated least squares based modification term for adaptive control. In *2010 American Control Conference*, Marriott Waterfront, Baltimore, MD, USA, June 30 - July 2 2010. 14

[30] S. Chumalee. *Robust Gain-Scheduled H-infinity Control for Unmanned Aerial Vehicles*. PhD thesis, School of Engineering, Cranfield University, 2010. 12

[31] M. V. Cook. *Flight Dynamic Principles*. Elsevier Ltd., Oxford, 2007. 34, 38, 173, 176

[32] M. Curvo. Estimation of aircraft aerodynamic derivatives using extended kalman filter. *Journal of the Brazilian Society of Mechanical Sciences*, **22(2)**:133–148, 2000. 13

[33] F. Daneshfar and H. Bevrani. Multi-agent systems in control engineering: A survey. *Journal of Control Science and Engineering*, **2009**[531080]:pp. 1–12, 2009. 26

[34] W. M. DeBusk, G. Chowdhary, and E. N. Johnson. Real-time system identification of a small multi-engine aircraft. In *Proceeding of the AIAA Atmosheric Flight Mechanics Conference*, pages 1–15, 2009. 14

[35] L. Dennis, M. Fisher, N. Lincoln, A. Lisitsa, and S. M. Veres. Practical verification of decision-making in agent-based autonomous systems. *Automated Software Engineering*, 2014. 3

[36] L. Dennis, M. Fisher, A. Lisitsa, N. Lincoln, and S. M. Veres. Satellite control using rational agent programming. *IEEE Intelligent Systems*, **25**[3], April 2010. 3, 26, 77

[37] G. Ducard, K. C. Kulling, and H. P. Geering. A simple and adaptive on-line path planing system for a uav. In *Proceeding of the 15th Mediterranean Conference on Control and Automation*, Athen, Greece, July 2007. 55, 57

[38] P. Eng, L. Mejias, R. Walker, and D. Fitzgerald. Guided chaos - path planning and control for a uav-forced landing. *IEEE Robotics & Automation Magazine*, **17**[2]:90–98, June 2010. 58

[39] A. Fekih and P. Pilla. A new fault tolerant control strategy for aircraft systems under adverse flying conditions. *Jounrnal of Automation & Systems Engineering*, **3**[2]:1–15, 2009. 23

[40] D. L. Fitzgerald. *Landing Site Selection for UAV Forced Landing using Machine Vision*. PhD thesis, Queensland University of Technology, 2007. 155

[41] L. Gao and S. Zhou. Application of $\mathcal{L}_1$ adaptive control augmentation to the flying wing unmanned aerial vehicle. In *Knowledge Acquisition and Modeling (KAM), 2011 Fourth International Symposium on*, pages 222–225, 2011. 17

[42] Z. Gao, C. Cecati, and S. X. Ding. A survey of fault diagnosis and fault-tolerant techniques part i: Fault diagnosis with model based and signal-based approaches. *Journal of IEEE Transactions on Industrial Electronics*, **62**[6]:pp. 3757–3767, 2015. 21

[43] E. G. Gracia and J. Becker. Uav stability derivatives estimation for hardware-in-the-loop simulation of piccolo autopilot by qualitative flight

testing. In *in 1st Latin American UAV Conference*, Panama, August 2007. 10

[44] I. M. Gregory, C. Cao, E. Xargay, N. Havakimyan, and X. Zou. $\mathcal{L}_1$ adaptive control design for nasa airstar flight test vehicle. In *AIAA Guidance, Navigation, and Control Conference*, Chicago, Ilinois, USA, August 2010. 3, 17

[45] M. S. Grewal and A. P. Andrews. *Kalman Filtering: Theory and Practice*. John Wiley and Sons, New York, USA, 2001. 84

[46] H. F. Grip, T. I. Fossen, T. A. Johansen, and A. Saberi. Nonliear observer for gnss-aided inertial navigation with quaternion-based attitude estimation. In *In Proceeding of American Control Conference*, Washington DC, 2013. 82

[47] Y. Cao H. Chao and Y. Chen. Autopilots for small unmanned aerial vehicles: A survey. *International Journal of Control, Automation, and Systems*, **8**[1]:36–44, 2010. 10

[48] J. J. Hageman, M. S. Smith, and S. Stachowiak. Integration of online parameter identification and neural network for in-flight adaptive control. Technical Report NASA/TM-2003-212028, NASA Dryden Flight Research Center, October, 2003. 14

[49] J. J. Hageman, M. S. Smith, and S. Stachowiak. Integration of online parameter identification and neural network for in-flight adaptive control. Technical Report NASA/TM-2003-212028, NASA Dryden Flight Research Center, October 2003. 15

[50] R. A. Hess and S. R. Wells. Sliding mode control applied to reconfigurable flight control design. *Journal of Guidance, Control, and Dynamics*, **26**[3]:452–462, 2003. 2

[51] P. A. Ioannou and P. V. Kokotovic. Instability analysis and improvement of robust of adaptive control. *Automatica*, **20**[5]:583–594, 1984. 16

[52] C. Ippolito, Y. H. Yeh, and J. Kaneshige. Neural adaptive flight control testing on an unmanned experimental aerial vehicle. In *AIAA Infotech@Aerospace 2007 Conference and Exhibit*, Rohner Park, Califonia, May 2007. 15

[53] R. Isermann. Model-based fault-detection and dianosis - status and applications. *Journal of Annual Reviews in Control*, **29**:pp. 71–85, 2005. 21

[54] D. Ward J. Monaco and R. Bird. Implementation and flight test assessment of an adaptive, reconfigurable flight control system. In *Proceeding of the AIAA Guidance, Navigation, and Control Conference*, pages 1443–1454, 1997. 13

[55] G. Jacques and J. Ducard. *Fault-Tolerant Flight Control and Guidance Systems for a Small Unmanned Aerial Vehicle*. PhD thesis, ETH ZURICH, 2007. 23, 24

[56] N. M. Jodeh. *Development of Autonomous Unmanned Aerial Vehicle Platform: Modeling, Simulating, and Flight Testing*. Master's thesis, Department of the Air Force Air University, Air Force Institute of Technology, 2006. 10

[57] E. N. Johnson, A. J. Calise, H. A. El-Shirbiny, and R. T. Rysdyk. Feedback linearilinear with neural network augmentation applied to x-33 attitude control. In *Proceeding of the AIAA Guidance, Navigation, and Control Conference*, pages 1–11, 2000. 15

[58] A. D. Kahn. Adaptive control for small fixed-wing unmanned air vehicles. In *AIAA Guidance, Navigation, and Control Conference*, Toronto, Ontario Cannada, August 2010. 15

[59] M. M. Kale and A. J. Chipperfield. Robust and stabilized mpc formulation for fault tolerant and reconfigurable flight control. In *Proceeding of the IEEE International Symposium on Intelligent Control*, pages 222–227, 2004. 16

[60] C. Kamali., A. A. Pashikar, and J. R. Raol. Real-time parameter estimation for reconfigureable control of unstable aircraft. *Defence Science Journal*, **57**[4]:381–391, July 2007. 14

[61] C. Kamali, A. A. Pashikar, and J. R. Raol. Evaluation of recursive least square algorithm for parameter estimation in aircraft real time application. *Journal of Aerospace Science and Technology*, **15**:165–174, 2011. 13

[62] R. Kamyar and E. Taheri. Aircraft optimal terrain/threat-based trajectory planning and control. *Journal of Guidance, Control, and Dynamics*, **37**[2]:466–483, 2014. doi: 10.2514/1.61339. 3

[63] C. D. Karlgaard, P. V. Tartabini, R. C. Blanchard, M. Kirsch, and M. D. Toniolo. Hyper-x post-flight trajectory reconstruction. *Journal of Spacecraft and Rockets*, **43**[1]:pp. 105–115, 2006. DOI: 10.2514/1.12733. 80, 81

[64] D. B. Kingston and R. W. Beard. Real-time attitude and position estimation for small uavs using low-cost sensors. In *In Proceeding of AIAA 3rd Unmanned Unlimited Technical Conference, Workshop and Exhibit*, Chicago, Illinois, September 2004. 82

[65] V. Klein, J. Batterson, and P. Murphy. Determination of airplane model structure from flight data by using modified stepwise regression. Technical Report NASA TP 1916, October 1981. 40

[66] V. Klein and E. A. Morelli. *Aircraft System Identification Theory and Pratice*. AIAA (American Institute of Aeronautics & Astronautics), 2006. 38, 40, 42, 45, 80, 90, 173, 176

[67] Y. Kobayashi and M. Takahashi. *Design of Intelligent Fault-Tolerant Flight Control System for Unmanned Aerial Vehicles*. Number ISBN:978-953-307-218-0. InTech, 2011. ix, 24, 25

[68] W. H. Kwon and S. Han. *Receding Horizon Control : Model Prodictive Control for State Models*. Springer London Ltd, 2005. ISBN: 1846280249. 16

[69] S. H. Lane and R. F. Stengel. Flight control design using nonlinear inverse dynamics. In *In Proceeding of American Control Conference*, pages 587–596, Seattle, WA, USA, June 1986. 12

[70] S. M. Lavalle. *Planning Algorithms*. New York: Cambridge Univ. Press, 2006. 58

[71] J. Leitner, A. Calise, and J. V. R. Prasad. Analysis of adaptive neural networks for helicopter flight control. *Journal of Guidance, Control, and Dynamics*, **20**[5]:972–979, 1997. doi: 10.2514/2.4142. 2

[72] N. K. Lincoln and S. M. Veres. Natural language programming of complex robotic bdi agents. *Journal of Intelligent and Robotic Systems*, **71**[2]:211–230, 2013. 71, 72, 77, 79

[73] N. K. Lincoln, S. M. Veres, L. Dennis, M. Fisher, and A. Lisitsa. An agent based framework for adaptive control and decision making of autonomous vehicles. In Proceeding of IFAC Workshop on Adaptation and Learning in Control and Signal Processing (ALCOSP), 2010. 3, 26, 144

[74] N. K. Lincoln, S. M. Veres, L. Dennis, M. Fisher, and A. Lisitsa. Autonomous asteroid exploration by rational agents. *IEEE Computational Intelligence Magazine*, **8**[4]:25–38, 2013. x, 26, 76, 77, 125, 145

[75] T. J. J. Lombaerts. *Fault Tolerant Flight Control A Physical Model Approach*. PhD thesis, Technische Unversiteit Delft, 2010. 18, 30, 155

[76] T. J. J. Lombaerts, H. O. Huisman, Q. P. Chu, J. A. Mulder, and D. A. Joosten. Nonlinear reconfiguring flight control based on online physical model identification. *Journal of Guidance, Control, and Dynamics*, **32**[3]:727–748, 2009. 13

[77] T. J. J. Lombaerts, E. R. van Oort, Q. P. Chu, J. A. Mulder, and D. A. Joosten. Online aerodynamic model structure selection and parameter estimation for fault-tolerant control. *Journal of Guidance, Control, and Dynamics*, **33**[3]:707–723, 2010. DOI: 10.2514/1.47256. 14, 40

[78] M. Lungu. Stabilization and control of a uav flight attitude angles using the backstepping method. *Jounal of World Academy of Science, Engineering and Technology*, **61**:290–297, 2012. 2

[79] M. Lungu and R. Lungu. Adaptive backstepping flight control for a mini-uav. *International Journal of Adaptive Control and Signal Processing*, **27**:635–650, August 2013. 17

[80] X. Lv, B. Jiang, R. Qi, and J. Zhao. Survey on nonlinear reconfigurable flight control. *Journal of Systems Engineering and Electronics*, **24**[6], December 2013. 2, 12

[81] J. Marzat, H. Piet-Lahanier, F. Damongeot, and E. Walter. Model-based fault diagnosis for aerospace systems: a survey. *In Proceeding of the Institution of Mechanical Engineers, Part G: Journal of Aerosapce Engineering*, **226**[10]:pp. 1329–1360, January 2012. ix, 21, 22

[82] A. Meystel and J. Albus. *Intelligent System: architecture, design and control*. Wiley, 2002. 63, 65

[83] B. Michini and J. P. Howy. $\mathcal{L}_1$ adaptive control for indoor autonomous vehicles: Design process and flight testing. In *AIAA Guidance, Navigation, and Control Conference*, Chicago, Illinois, August 2009. AIAA 2009-5754. 17

[84] T. R. Moes, M. S. Smith, and E. A. Morelli. Flight investigation of prescribed simultaneous independent surface excitations for real-time parameter identification. Nasa/tm-2003-212029, NASA Dryden Flight Research Center and NASA Langley Research Center, October 2003. 14

[85] E. A. Morelli. In-flight system identification. In *AIAA Atmospheric Flight Mechanics Conference*, Boston, Massachusette, August 1998. 13

[86] E. A. Morelli. Real-time parameter estimation in the frequency domain. *Journal of Guidance, Control, and Dynamics*, **23**[5]:812–818, 2000. 14

[87] E. A. Morelli. Real-time aerodynamic parameter estimation witwith air flow angle measurements. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, number AIAA-2010-7951, Toronto, August 2010. 80

[88] E. A. Morelli and M. S. Smith. Real-time dynamic modmodel: Data information requirements and flight-test results. *Journal of Aircraft*, **46(6)**:1894–1904, 2009. 13, 14

[89] R. S. M. Munoz, C. Rossi, and A. B. Cruz. *Modelling and identification of Flight Dynamics in Mini-Helicopters using Neural Networks*. Number ISBN:978-953-7619-41-1. InTech, 2009. 15

[90] K. S. Narendra and A. M. Annaswamy. A new adaptive law for robust adaptation without persistent excitation. *IEEE Transactions on Automatic Control*, **AC-32**[2]:134–145, February 1987. 16

[91] M Nauroka. Real-time wavelet flight data evaluation for system identification of flight characteristics. In *28th International Congress of the Aeronautical Sciences*, pages 23–28, Brisbane, AUS, September 2012. xv, 120, 121, 122

[92] R. C. Nelson. *Flight Stability and Automatic Control*. McGRAW-Hill companies, Inc, 1998. 11

[93] N. Nguyen. Hybrid adaptive flight control with model inversion adaptation. *Advances in Flight Control Systems*, ISBN: 978-953-307-218-0 2011. 18, 19

[94] N. Nguyen. *Hybrid Adaptive Flight Control with Model Inversion Adaptation, Advances in Flight Control Systems*, chapter 3, pages 53–76. InTech, 2011. ISBN 978-953-307-218-0. 17

[95] N. Nguyen, J. Burken, and C. Hanson. Optimal control modification adaptive law with covariance adaptive gain adjustment and normalization. In *In Proceeding of AIAA Guidance, Navigation, and Control Conference*, 2011. 122

[96] N. Nguyen, K. Krishnakumar, J. Kaneshige, and P. Nespeca. Flight dynamics and hybrid adaptive control of damaged aircraft. *Journal of Guidance, Control, and Dynamics*, **31**[3]:751–764, May-June 2008. 3, 19, 36

[97] N. T. Nguyen. Optimal control modification for robust adaptive control with large adaptive gain. *Journal of System & Control Letters*, **61**:485–494, 2012. x, 15, 16, 51, 104

[98] N. T. Nguyen, M. Bakhtiari-Nejad, and Y. Huang. Hybrid adaptive flight control with bounded linear stability analysis. In *Proceeding of AIAA Guidance, Navigation and Control Conference and Exhibit*, Hilton Head, South Carolina, August 2007. 17

[99] N. T. Nguyen and J. D. Boskovic. Bounded linear stability margin analysis of nonlinear hybrid adaptive control. In *American Control Conference*, Seattle, Washington, USA, June 2008. 17, 19

[100] W. J. Park, E. T. Kim, Y. K. Song, and B. J. Ko. A study on the real-time parameter estimation of durumi-ii for control surface fault using flight test data (longitudinal motion). *International Journal of Control, Automation, and Systems*, **5**[4]:410–418, August 2007. 14

[101] Y. C. Paw. *Synthesis and Validation of Flight Control for UAV*. PhD thesis, The University of Minnesota, 2009. 11

[102] C S. Pillar. *Path Planning , Guidance and Control for a UAV Forced Landing*. PhD thesis, Queensland University of Technology, 2011. 155

[103] R. W. Pratt, editor. *FLIGHT CONTROL SYSTEMS: Practical Issues in Design and Implementation*. Series 57. IEE Control Engineering, 2000. 11

[104] Public Domain Computer Programs for the Aeronautical Engineer., [Online]. Available from http://www.pdas.com/datcom.html. *Digital Datcom*. 109, 158

[105] V. Puttige and S. Anavatti. Real-time system identification of unmanned aerial vehicles: A multi-network approach. *Journal of Computers*, **3**[7]:31–38, July 2008. 15

[106] V. R. Puttige and S. G. Anavatti. Comparison of real-time online and offline neural network mmodel for a uav. In *Proceeding of International Joint Conference on Neural Networks*, Orlando, Florida, USA, August 12-17 2007. 15

[107] C. Ramprasadh and H. Arya. Multi-stage fusion algorithm for estimation of aerodynamic angles in mini aerial vehicle. In *In Proceeding of 49th AIAA Aerospace Science Meeting*, Orlando, Florida, 2011. 81

[108] N. D. Richards, R. J. Adams, D. H. Klyde, and B. Cogan. Flight test evaluation of an adaptive controller for flying qualites specification and protection. *Journal of Guidance, Control, and Dynamics*, **38**[12]:2241–2256, December 2015. 16

[109] G. Shah. Aerodynamic effects and modeling of damage to transport aircraft. In *In proceedinf of AIAA Atmospheric Flight Mechanics Conference and Exhibit*, Honolulu, Hawaii, 2008. 40

[110] E. Y. Shapiro and J. C. Chung. Flight control system synthesis using eigenstructure assignment. *Journal of Optimization Theory and Application*, **43**[3]:415–429, 1984. 11

[111] Y. Shin, A. J. Calise, and M. D. Johnson. Adaptive control of advanced fighter aircraft in nonlinear flight regimes. *Journal of Guidance, Control, and Dynamics*, **31**[5]:1464–1477, September-October 2008. 15

[112] Y. Shin, A. J. Calise, and M. A. Motter. Application of adaptive autopilot designs for an unmmaned aerial vehicle. In *In proceeding of AIAA*

Guidance, Navigation, and Control Conference and Exhibit, San Francisco, California, USA, August 2005. 2

[113] N. Slegers, J. Kyle, and M. Costello. Nonlinear model predictive control technique for unmanned air vehicles. *Journal of Guidance, Control, and Dynamics*, **29**[6]:1179–1188, September-October 2006. 16

[114] M. S. Smith, T. R. Moes, and E. A. Morelli. Real-time stability and control derivative extraction from f-15 flight data. Technical Report NASA/TM-2003-212027, NASA Dryden Flight Research Center and NASA Langley Research Center, September 2003. 14

[115] S. A Snell, D. F. Nns, and W. L. Arrard. Nonlinear inversion flight control for a supermaneuverable aircraft. *Journal of Guidance, Control, and Dynamics*, **15**[4]:976–984, 1992. doi: 10.2514/3.20932. 2

[116] Y. Song, G. Campa, M. Napolitano, B. Seanor, and M. G. Perhinschi. Online parameter estimation techniques comparison within a fault tolerant flight control system. *Journal of Guidance, Control, and Dynamics*, **25**[3]:528–537, May-June 2002. 14

[117] Y. Song, B. Song, B. Seanor, and M. R. Napolitano. On-line aircraft parameter identification using fourier transform regression with an application to nasa f/a-18 harv flight data. *KSME International Journal*, **16**[3]:327–337, 2002. 14

[118] L. Sonneveldt. *Adaptive Backstepping Flight Control for Modern Fighter Aircraft*. Wohrmann Print Service, 2010. ISBN 978-90-8570-573-4. 17

[119] R. F. Stengel. Intelligent failure tolerant control. *Journal and Magazines of Control Systems, IEEE*, **11**[4]:14–23, June 1991. 21

[120] R. F. Stengel. Toward intelligent flight control. *IEEE Transactions on Systems, Man and Cybernetics*, **23**[6]:1699–1717, Novermber/December 1993. 20

[121] D. J. Stilwell. State-space interpolation for a gain-scheduled autopilot. *Journal of Guidance, Control, and Dynamics*, **24**[3]:460–465, 2001. doi: 10.2514/2.4766. 2

[122] L. G. Sun. *Model and Sensor Based Nonlinear Adaptive Flight Control with Online System Identification*. PhD thesis, Delft University of Technology, 2014. ix, 23, 24

[123] S. Suzuki and A. Yanagida. Research and development for fault tolerant flight control system - part 1. intelligent flight control system. In *26th International Congress of the Aeronautical Sciences (ICAS2008)*, pages 1–7, 2008. 23, 24

[124] SysBrain Ltd, [Online]. Available from http://www.sysbrain.com/cognitive_agent_toolbox. *sEnglish Publiser*. 31, 144, 145, 148, 155

[125] Y. Tang. *Fault Tolerant Control for Nonlinear Aircraft based on Feedback Linearization*. PhD thesis, University of Hull, 2013. 23

[126] Y. Tang and R. J. Patton. Fault-tolerant flight control for nonlinear-uav. In *2012 20th Mediterranean Conference on Control & Automation (MED)*, 2012. 23

[127] S. Tantrairatn and S. M. Veres. Onboard system identification for improved flight control of uas. In *In Proceeding of 8th IFAC Symposium on Robust Control Design (ROCOND'15)*, **48**, pages 368–375, Bratislava, Slovak Republic, July 2015. 85, 93

[128] D. Thakur, S. Hernandez, and M. R. Akella. Space swarm finite-thrust cooperative control for common orbit convergence. *Journal of Guidance, Control, and Dynamics*, **38**[3]:478–488, 2015. doi: 10.2514/1.G000621. 3

[129] Unmanned Dynamic, [Online]. Available from http://www.u-dynamics.com/aerosim/default.htm. *AereSim Aeronautical Simulation Blockset*. 158

[130] V. Venkatasubramanian, R. Rengaswamy, and S. N. Kavuri. A review of process fault dectection and diagnosis: Part ii: Qualitative models and search strategies. *Journal of Computers and Chemical Engineering*, **27**[3]:pp. 313–326, March 2003. 21

[131] V. Venkatasubramanian, R. Rengaswamy, S. N. Kavuri, and K. Yin. A review of process fault dectection and diagnosis: Part iii: Process history based methods. *Journal of Computers and Chemical Engineering*, **27**[3]:pp. 327–346, March 2003.

[132] V. Venkatasubramanian, R. Rengaswamy, K. Yin, and S. N. Kavuri. A review of process fault detection and diagnosis: Part i: Quantitative model-based methods. *Journal of Computers and Chemical Engineering*, **27**[3]:pp. 293–311, 2003. 21

[133] S. M. Veres. Natural language programming of agents and robotic devices. SysBrain, 2008. 79, 145, 155, 158

[134] S. M. Veres. Knowledge of machines: Review and forward look. *Journal of Systems and Control Engineering*, **225**, April 2011. 3, 26, 63

[135] S. M. Veres and J. Luo. A class of bdi agent architecture for autonomous control. In *In proceeding of IEEE Conference on Decision and Control*, **5**, pages pp. 4746–4751, December 2004. 27

[136] X. Wang and N. Hovakimyan. Predicting the performance of uncertain multi-agent system using event-triggering and $\mathcal{L}_1$ adaptation. In *Proceeding of AIAA Guidance, Navigation, and Control Conference*, pages 1–15, Toronto, Ontarlo, Canada, August 2010. 3

[137] D G. Ward, R L. Barron, M P. Carley, and T J. Curtis. Real-time parameter identification for self-designing flight control. In *In Proceeding of National Aerospace and Electronics Conference (NAECON 1994)*, pages 526–531, May 1994. 2

[138] H L. Wei, S A. Billings, and J Liu. Term and variable selection for nonlinear system identification. *International Journal of Control*, **77**[1]:86–110, 2004. 96

[139] M. Wooldridge. *An Introduction to MultiAgent Systems*. Wiley, 2002. 3, 26, 63, 64

[140] W. Xingjian, W. Shaoping, Y. Zhongwei, and Z. Chao. Active fault-tolerant control strategy of large civil aircraft under elevator failures. *Chinese Journal of Aeronautics*, **28**[6]:1658–1666, 2015. 23

[141] D. I. You, Y. D. Jung, and S. W. Cho. A guidance and control law design for precision automatic take-off and landing of fixed-wing uavs. In *AIAA Guidance, Navigation, and Control Conference*, Minneapolis, Minnesota, August 2012. AIAA 2012-4674. 17

[142] Z. Yu, G. Fan, and J. Yi. Indirect adaptive flight control based on nonlinear inversion. In *Proceeding of the 2009 IEEE International Conference on Mechatronics and Automation*, Changchun, China, August 2009. 13

[143] T. Yucelen. *Advances in Adaptive Control Theory: Gradient and Derivative-Free Approaches*. PhD thesis, Aerospace Engineering, Georgia Institute of Technology, May 2012. 99

[144] J. Zhang, L. Yang, and G. Shen. New hybrid adaptive control approach for aircraft with centre of gravity variation. *Journal of IET Control Theory and Applications*, **6**[14]:2179–2187, 2012. 19

[145] Y. Zhang and J. Jiang. Bibliogapproach reviw on reconfigurable fault-tolerant control systems. *Journal of Annual Reviews in Control*, **32**:229–252, 2008. 21, 22