Design of a DDP Controller for Autonomous Autorotative Landing of RW UAV Following Engine Failure

Puseletso Matlala

A dissertation submitted to the Faculty of Engineering and the Built Environment, University of the Witwatersrand, Johannesburg, in partial fulfilment of the requirements for the degree of Master of Science in Engineering.

Johannesburg, April 2016

Declaration

I declare that this dissertation is my own, unaided work, except where otherwise acknowledged. It is being submitted for the degree of Master of Science in Engineering in the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination at any other university.

Signed this ___ day of _____ 20___

_____

Puseletso Matlala.

Acknowledgements

I would like to thank the Lord Almighty for the life, health and grace to carry this project. I would also like to thank the following people:

- My parents and family for their encouragement and support.
- My fiance for always pushing, encouraging and supporting me.
- Denel Aviation for the financial support
- Dr Renier van Rooyen for his technical guidance and asistance
- Prof J. O. Pedro for his deep insight and direction throughout this project.

Abstract

A Rotary Wing Unmanned Aerial Vehicle (RW UAV) as a platform and its payload consisting of sophisticated sensors would be costly items. Hence, a RW UAV in the 500 kg class designed to fulfil a number of missions would represent a considerable capital outlay for any customer. Therefore, in the event of an engine failure, a means should be provided to get the craft safely back on the ground without incurring damage or causing danger to the surrounding area. The aim of the study was to design a controller for autorotative landing of a RW UAV in the event of engine failure. In order to design a controller for autorotative landing, an acceleration model was used obtained from a study by Stanford University. FLTSIM helicopter flight simulation package yielded necessary RW UAV response data for the autorotation regimes. The response data was utilized in identifying the unknown parameters in the acceleration model. A Differential Dynamic Programming (DDP) control algorithm was designed to compute the main and tail rotor collective pitch and the longitudinal and lateral cyclic pitch control inputs to safely land the craft. The results obtained were compared to the FLTSIM flight simulation response data. It was noted that the mathematical model could not accurately model the pitch dynamics. The main rotor dynamics were modelled satisfactorily and which are important in autorotation because without power from the engine, the energy in main rotor is critical in a successful execution of an autorotative landing. Stanford University designed a controller for RC helicopter, XCell Tempest, which was deemed successful. However, the DDP controller was designed for autonomous autorotative landing of RW UAV weighing 560 kg, following engine failure. The DDP controller has the ability to control the RW UAV in an autorotation landing but the study should be taken further to improve certain aspects such as the pitch dynamics and which can possibly be achieved through online parameter estimation.

Published Work

Conference Paper under Review

- Matlala, P., Pedro, J.O.,"Design of DDP Controller for Autonomous Autorotative Landing of RWUAV Following Engine Failure",*Proceedings of the 2016 IEEE Multi-Conference on Systems and Control, Buenos Aires, Argentina, 19-22 September 2016.* Paper number 28

Journal Paper in Development

- Matlala, P., Pedro, J.O.,"Applying Differential Dynamic Programming Control algorithm to Autonomous Autorotative Landing of 560 kg RWUAV", *Nonlinear Dynamics.*

Contents

List of Figures

List of Tables

List of Symbols

C_u	Longitudinal acceleration derivative
C_v	Lateral acceleration derivative
C_w	Vertical acceleration derivative
C_p	Roll acceleration derivative vector
C_q	Pitch acceleration derivative vector
C_r	Yaw acceleration derivative vector
C_Ω	Main rotor acceleration derivative vector
F	Objective function
g	Gravity, m/s^2
I_{xx}	Roll moment of inertia, $[kg.m^2]$
I_{yy}	Pitch moment of inertia, $[kg.m^2]$
I_{zz}	Yaw moment of inertia, $[kg.m^2]$
I_{xz}	Product of inertia, $[kg.m^2]$
n	Index specifying a stage
N	Total number of time increments into which the time horizon has been divided
p	Roll rate, positive towards starboard, [rad/s]
P	Power, [kW]

P_0	Power before engine failure, [kW]
q	Number of decision variables
q	Pitch rate, positive nose up, [rad/s]
r	Yaw rate, positive towards starboard,[rad/s]
s(n)	m-dimensional state vector at stage n
S(n)	Admissible domain in the state space at stage n
t	Time
u(n)	q-dimensional decision vector at stage n-1
U(n)	Admissible domain in the decision space at stage n
u	Longitudinal velocity, positive forward, [m/s]
v	Lateral velocity, positive to the right, [m/s]
w	Vertical velocity, positive downward, [m/s]
τ_1	Time constant for the spool-down response, [s]
θ	Pitch angle, [rad]
θ_0	Main rotor collective pitch angle, [rad]
θ_{1s}	Longitudinal cyclic pitch angle, [rad]
θ_{1c}	Lateral cyclic pitch angle, [rad]
θ_{0T}	Tail rotor collective pitch, [rad]
ϕ	Roll angle, [rad]
ψ	Yaw angle, [rad]
Ω	Main rotor speed, [rad/s]

1 INTRODUCTION

1.1 Research Background

Unmanned aerial vehicles (UAVs) have potential applications in many areas. UAVs are capable of carrying out tasks in environments, which would be dangerous for human beings. UAV's ability to manoeuvre makes it versatile and ideal. UAVs can be used for surveillance and security, search and rescue, inspection and exploration. UAVs are developed to operate autonomously without human pilot. The difficult task is that they need to handle various situations that may arise in complicated and uncertain environments, such as unexpected obstacles, enemies attacking, device failures and communicating with technical personnel in the ground station. Failure of a device such as an engine can cause damage to the UAV and the sophisticated sensors forming part of the payload, which would be costly and dangerous. In such an instance, the UAV should be able to autonomously and safely execute an autorotation manoeuvre [Cai et al., 2010].

The Vertical Take-off and Landing Tactical Unmanned Aerial Vehicle (VTUAV) called the Fire Scout (see Figure 1.1) crashed and the investigation revealed human error associated with damage to onboard antennas during ground handling which led to the accident. An incorrect signal was emitted which caused the radar altimeter to incorrectly track the altitude. The antennas gave a false reading that indicated the Fire Scout was 2 ft above the ground when it was actually hovering at 500 ft. The "land" command was given and according to flight procedure, the engine shut down. Unique approaches to automation and procedures often lead to unforeseen and costly outcomes [Williams, 2004].

The A160T Hummingbird is an autonomously flown rotary-wing UAV (see Figure 1.2) making its own decisions to meet certain objectives rather than relying on human control. The Hummingbird incorporates optimum-speed rotor technology concept (OSRT) which improves the overall performance by adjusting the main rotor speed at different altitudes, gross weight and cruise speeds. This technology

Figure 1.1: Fire Scout

makes it unique from conventional rotor systems which tend to have a fixed rotor RPM regardless of altitude. [Daily, 2012]

Figure 1.2: Boeing A160 Hummingbird UAV

There is a wide interest in research on the control of helicopter UAVs but it is a challenging area. Military applications have dominated the UAV field but there is an increased interest in civilian and public domain applications [Alvarenga et al., 2015]. This was shown in a survey of the field [Kendoul, 2012]. Godbolt and Lynch (2013) provided new physical input models of rotor thrust and main rotor counter-torque. Different input models of varying complexity have been proposed to date but no experimentally validated physical input model which is suitable for control design has appeared [Godbolt and Lynch, 2013]. Godbolt and Lynch (2013) used

experimental data to demonstrate inaccuracies of certain input models. The experimental results they presented was collected using ANCL helicopter UAV platform which is described in [Godbolt et al., 2013] and [Godbolt and Lynch, 2013]. This ANCL helicopter UAV is shown in the Figure 1.3. The models were generalized to account for velocity dependence [Godbolt and Lynch, 2013]. They expect their work will prove useful for nonlinear model-based helicopter control [Godbolt and Lynch, 2013].

Figure 1.3: ANCL helicopter UAV

Autorotation landings are challenging to execute and when done incorrectly it can lead to severe damage or even complete loss of the helicopter. If the main rotor speed is too low, reliably controlling the helicopter becomes impossible and hence the helicopter will crash. A high horizontal speed when the helicopter touches down will cause the helicopter to tip over. There is only a single opportunity to execute an autorotation landing and if this is done with a poor controller, the helicopter may be destroyed [Abbeel et al., 2009].

Autorotation manoeuvre is split into three stages (see Figure 1.4) [Abbeel et al., 2009]:

- Autorotation glide: helicopter descends while maintaining a high rotor speed. The aim is to increase the translational kinetic energy of the helicopter to transfer it to the main rotor for flare. (Point 1 to 2, see Figure 1.4)
- Autorotation flare: at a certain altitude above the ground, the helicopter transitions from glide to flare. In which it slows down and ideally brings it to a zero velocity about a certain height above the ground. (Point 2 to 3)
- Autorotation landing: the helicopter lands using the remaining rotor speed to

maintain a level orientation and slowly descends until touch-down. (Point 4)

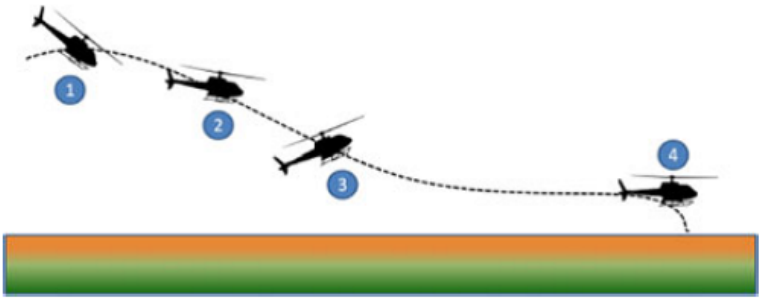


Figure 1.4: Autorotation manoeuvre [Santamaría et al., 2013]

1.2 Research Rationale and Motivation

A Rotary-Wing UAV (RW UAV) in the 500 kg class is envisaged that would fulfil a number of missions. Rotary-wing air vehicles offer capabilities such as vertical take-off and landing from confined spaces and hovering in close proximity to points of interest for surveillance and stores delivery. Such a craft would represent a considerable capital outlay for any customer, as both the RW UAV as a platform and its payload comprising sophisticated sensors, would be costly items. Therefore, in the event of an engine failure, a means should be provided to get the craft safely back on the ground without incurring damage or causing danger.

In comparison to regular landings, where a pilot could abort a landing attempt and try again, in autorotation landings there is only a single opportunity at the approach. This makes autorotation landings logistically a challenging problem.

1.3 Problem Statement

There is a requirement for a RW UAV with a mass of 560 kg, in the event of an engine failure to autonomously enter into autorotation, maintain steady autorotation to a height above ground level at which the flare for landing can be initiated. It has to autonomously execute the flare to reduce the forward speed and thereafter land without causing structural damage to the craft or damage to its payload.

1.4 Literature Review

1.4.1 History

An autorotation model was derived by Johnson in 1977, which included vertical and longitudinal movement [Johnson, 1977]. The optimal control used a cost function that was dependent on horizontal and vertical speed at touchdown. The control law was based on iterative numerical integration forwards and backwards between the two boundary points and updating using the steepest descent method. A linear and non-linear autorotation model was derived by Kensaku et al. (2004). A PI controller was used to land the small unmanned helicopter [Kensaku et al., 2004]. A method was formulated to optimize the trajectory and control inputs but the method pre-calculated the control inputs and trajectory before entering into autorotation [Aponso, 2005]. This method is not robust with regards to modelling errors and outside interference due to the pre-computation of the control inputs and trajectory, [Dalamagkidis, 2009].

1.4.2 Mathematical Model

Typically, when designing helicopter controllers, a model for the helicopter dynamics is firstly constructed and that model is used to design the controller. The model of the helicopter usually flies well in simulation but in real life, the performance deteriorates. This could be attributed to errors in the model of the helicopter as well as turbulence but building accurate helicopter models remains a challenge in autonomous flight, [Abbeel et al., 2005].

"CIFER (Comprehensive Identification from Frequency Responses) is the industry standard for learning helicopter models from data" [Abbeel et al., 2005]. It uses frequency response methods to identify a linear model. Disadvantage of CIFER is that it does not capture important aspects of the helicopter dynamics, such as the effects of inertia; this is evident from the models used in [Bagnell and Schneider, 2001], [Mettler et al., 1999] and [Abbeel et al., 2005]. However, this was due to the "naive body-coordinate" model that was used, which made it difficult for the for the learning algorithm to capture properties such as inertia [Abbeel et al., 2005].

Least-squares method produces maximum likelihood estimate of a parameter even if the probabilistic assumptions are not satisfied. Years of experience have shown that least-squares method produce useful results. Maximum likelihood estimation

method finds the most likely value for the parameter based on the data set. The methods are applicable to nonlinear systems but the presence of atmospheric turbulence often yields biased results [Jategaonkar et al., 2004]. Maximum likelihood method can handle measurement and process noise. However, it incorporates a Kalman filter which leads to the filter error method [Raol et al., 2004].

Locally weighted regression captures nonlinearities of a system. Locally weighted regression fits linear function locally, using the data close to the query. Points close to the query are given a large weight and points that are distant are given a small weight. To make a prediction at a new input, all the data will be reweighted again and a linear function will be fitted using those weights. The prediction will be computed by evaluating the linear function at the location of the new query. This procedure chooses a different linear function at every point. Plotting all the predictions as a function of the input query, the obtained function is not locally linear or locally affine, nor piecewise linear nor piecewise affine. Therefore, locally weighted regression is able to identify fully nonlinear models [Rahideh et al., 2007].

Abbeel et al. (2009) modelled a helicopter with thirteen dimensional states consisting of position, orientation, velocity, angular rate and main rotor speed. It was controlled via cyclic pitch controls, tail rotor rudder and main collective pitch control. The effects of inertia and gravity were subtracted and a model was learnt from data to predict accelerations. As a result, a number of parameters were estimated from flight data. The accelerations were integrated over time to obtain position, velocity, orientation, angular rate and main rotor speed [Abbeel et al., 2009].

Abbeel et al. (2005) used apprenticeship learning algorithm to control an aerobatic helicopter. Firstly, data was collected from a human pilot flying the desired manoeuvres with the aerobatic helicopter and the model was learnt from data. The formulated acceleration model did not include inertial coupling between different axes of rotation and the model did not include blade-flapping angles. The blade flapping angles and inertial coupling have been shown to improve the accuracy of the helicopter models but they believe the effects of inertial coupling to be very limited. This is because the flight regimes considered do not include fast rotation around more than one main axis simultaneously.

1.4.3 Control Techniques

Abbeel et al. (2009) recorded several autorotation manoeuvres from the expert pilot demonstrations. The autorotation demonstrations were split into three trajectories

and were used as the target trajectories. The target for the glide state was a steady-state rather than a trajectory, in particular velocity and rotor speed. The target for landing was level orientation and zero velocity. The best expert pilot demonstration was chosen for the flare target trajectory. Differential Dynamic Programming (DDP), an extension of Linear Quadratic Regulator (LQR) was used to design a controller for autonomous autorotation. The experiment was successful [Abbeel et al., 2009].

DDP approximately solves general continuous state-space Markov Decision Process (MDP) by computing the linear approximation to the dynamics and quadratic approximation to the reward function around the trajectory obtained when using the current policy. Computing the optimal policy for the LQR obtained and set the current policy equal to the optimal policy for the LQR problem [Abbeel et al., 2005].

Key problems in reinforcement learning and control are control of high-dimensional, continuous and nonlinear dynamical systems. Local methods, like DDP, are not directly subject to the curse of dimensionality, but do not model the value function or policy over the entire state space by focusing computational effort along likely trajectories. Receding Horizon DDP (RH-DDP) is a modification of the classic DDP algorithm. It allows stable and robust controllers to be constructed based on local control trajectories, in highly nonlinear, high-dimensional domains. This method is reminiscent of Model Predictive Control [Tassa et al., 2008].

In a study from [Abbeel et al., 2009], the algorithm approximately extracts an implicitly encoded optimal demonstration from multiple suboptimal expert demonstrations and builds a dynamic model in the vicinity of this trajectory suitable for high performance control. The algorithm learns a target trajectory and a model that allows the robot to mimic the behaviour of the expert. The assumption made was that the expert demonstrations were misaligned copies of the ideal trajectory corrupted by Gaussian noise. Prior knowledge can be incorporated to the learned ideal trajectory to enhance it. The algorithm was tested on the XCell Tempest, see Figure 1.5. The helicopter used receding-horizon differential dynamic programming controller. The algorithm successfully flew the following manoeuvres in rapid sequence: split-S, snap roll, stall turn, loop, loop with pirouette, stall-turn with pirouette, “hurricane” (fast backward funnel), knife edge, flips and rolls, tic-toc and inverted hover. The controller achieved RMS position error of 1.75 meters [Abbeel et al., 2009].

Reinforcement learning algorithm often includes learning of some form of system model while determining an optimal policy. These techniques face difficulties in the

Figure 1.5: XCell Tempest autonomous helicopter

application to robotics. Physical systems are high dimensional and it is difficult to have data of all the parts of state space. Any model used by the learning algorithm is not capable of capturing all of the subtlety of the real system dynamics and building optimal policies is complex [Bagnell and Schneider, 2001].

Model Predictive Control (MPC) also known as Receding Horizon Control (RHC) is a sophisticated control methodology which has been applied to linear system in the past but increasing interest has resulted for nonlinear system applications. The focus in MPC is to design a controller "where the inputs into the controller design are what to control, instead of how to control." Figure 1.6 illustrates the workings of MPC [Khan et al., 2011]. MPC controller has an internal model used to predict the behaviour of the plant over future prediction Horizon, Hp. The best input is to be selected such that the best predicted behaviour will be produced. The aim is to bring the predicted output as close as possible to the reference trajectory. This can be done by optimising the cost function [Khan et al., 2011].

Dalamagkidis (2009) derived a generic model of vertical autorotation. The frame of reference was based on a single axis because under vertical autorotation, no longitudinal or lateral movement is exhibited by the helicopter. The model was highly nonlinear and under-actuated. An analytical solution for such a problem involves solving the Hamilton-Jacobi-Bellman equation and which is possible for only a subset of nonlinear problems. An alternative would be to introduce a nonlinear feedback control law that linearises the system. Dalamagkidis (2009) used model predictive control as the control technique.

Dalamagkidis (2010) proposed the use of a nonlinear model predictive controller augmented by recurrent neural network for autonomous autorotation. With this

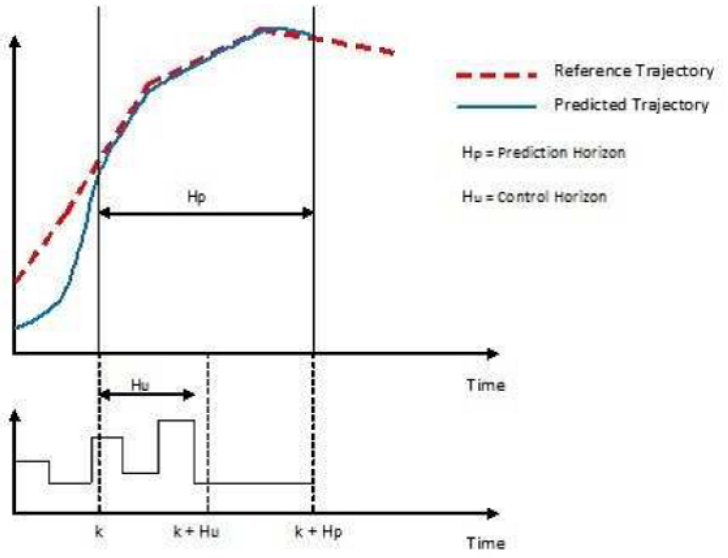


Figure 1.6: Workings of Model Predictive Control [Khan et al., 2011]

approach, on-board processing capacity was not limited and which did not put bounds on the computational complexity of the controller in real-time operation. The controller was tested using the model of the OH-58A helicopter with high-energy rotor system. The simulation performed showed that the helicopter accomplished the objective, [Dalamagkidis et al., 2010].

Tierney (2010) developed a methodology to enable computation of the set of all steady-state autorotation conditions, which are likely to result in a safe flare to landing. Safe and feasible meaning the controls and states stay within predefined allowable limits throughout the flight and touchdown occurs with descent rate, forward speed, position and pitch angle within acceptable limits. The safe landing set was determined by repeatedly solving a trajectory optimization problem from an initial state to a specified touchdown point. This method was tested for a limited number of initial states for a utility helicopter in steady-state autorotation. The results revealed regions of autorotation trim space which would lead to safe landing [Tierney, 2010].

Gavrilets et al. (2002) demonstrated an automatic axial roll with an X-Cell 60 helicopter. The control logic consists of steady-state trim trajectory controllers used prior to and upon exit from the manoeuvre and a manoeuvre logic inspired by human pilot strategies. The control laws designed for manoeuvre execution were tight angular rate tracking loops, experimentally determined reference trajectories for the angular rates, and a collective pitch modulation law. Low-order linear quadratic

regulator design was used for the trim trajectory tracking controllers. Flight tests with the control logic demonstrated smooth entry into the manoeuvre, automatic recovery to a steady-state trim trajectory and robustness of the trim trajectory control towards measurement and modelling errors [Gavrilets et al., 2002].

A study conducted by Santamaria et al. (2013) focused on specific autorotation patterns, where the initial altitude and velocity conditions for the emergency landing manoeuvre are fixed. The study entailed defining certain control guidelines and commands for each of the three stages of autorotation. The study was based on a set of experimental data of a small helicopter carried out by a skilled pilot. A set of guidelines to design a controller for autonomous autorotation landing of an UAV were presented by Santamaria et al. (2013).

Meng and Cheng (2013) in a trajectory optimization study, augmented a 6 degree of freedom (DOF) rigid body flight dynamics model of a helicopter which was described as a set of nonlinear differential algebraic equations. The trajectory optimization problem of the helicopter, UH-60, autorotation landing was formulated into nonlinear optimal control problem to take into account safety-related requirements and helicopter performance [Meng and Chen, 2013].

A comprehensive 8 DOF helicopter model to find the optimal control for all engines inoperative (AEI) autorotation landing, one engine inoperative (OEI) landing and OEI take-off for PZL Mi-2 Plus helicopter was formulated by Bibik et al. (2012). The main and tail rotor collective pitch angles, as well as the longitudinal and lateral cyclic pitch angles were determined by a discrete time adaptive optimal control algorithm for specified engine failure conditions [Bibik and Narkiewicz, 2012].

A direct optimal control method was applied to control an unmanned aerial vehicle helicopter executing autorotation [Taamallah, 2012]. This study also included obstacle avoidance capability by incorporationg a three-dimensional obstacle information path constraints [Taamallah, 2012].

Yomchinda (2013) objective was to develop a real-time system which provides full autonomous control for autorotation landing of a helicopter similar to UH-60 Blackhawk helicopter. Three different optimization algorithms were used to generate trajectories for entry into autorotation, descent and flare [Yomchinda, 2013]. The primary flight control was designed using a linear dynamic inversion control scheme and a path following control law was developed to track the autorotation trajectories [Yomchinda, 2013]. The results obtained by Yomchinda (2013) indicated the

feasibility of the capability of the algorithms to operate in real-time and of the integrated systems ability to provide safe autorotation landings. A complete feasibility solution from glide to touchdown was demonstrated by [Yomchinda et al., 2011].

Choudhury et al. (2013) designed a planning system that computes alternate routes (AR) in a rapid fashion. The algorithm builds upon the optimal sampling-based RRT* to generate AR in real-time while maintaining optimality guarantees and examines performance for simulated failures occurring in mountainous terrain [Choudhury et al., 2013]. RRT* alogorithm has been a significant contribution in planning problem of optimizing a cost function proposed by [Karaman and Frazzoli, 2010]. Choudhury et al. (2013) framed an optimization problem that returned as a solution a set of answers and these solutions have costs close to the optimal cost. Abraham et al. (2013) framed the same problem to solve for alternate routes in road networks. The problem was decomposed into various sub-problems to be linked via a state machine and framed as a multiple objective planning problem as in [Scherer and Singh, 2011]. The approach of unconstrained planning problem was adopted from Scherer et al. (2011). The proofs and the analysis of the results can be found in [Choudhury et al., 2012].

1.5 Identified Gaps

The gaps in the current literature that need to be addressed may be summarized as follows:

- There is limited research concerning DDP control algorithm applications on autonomous flight.
- DDP control algorithm is mostly applied to RC heclicopters and not on RW UAV's in the 500 kg class. These aircraft have different geometrical and physical properties which may alter the flight dynamics.

1.6 Research Question

Can the application of Differential Dynamic Programming methodology on a 500 kg class RW UAV be achieved for the design of an autonomous autorotative controller following engine failure?

1.7 Research Objectives

The research objectives are as follows:

- To formulate an acceleration model of the helicopter that will be suitable for control of the RW UAV in autorotation.
- To define a series of simulation runs to be performed with the available helicopter flight simulation package (FLTSIM) that would yield the necessary RW UAV response data for prescribed control inputs, covering the following regimes:
  - Entry into autorotation from forward flight
  - Maintaining autorotation and steering to a landing spot
  - Executing flare for landing
  - Executing landing

One flight condition will be considered, namely ISA sea level condition.

- To use FLTSIM software package populated with RW UAV geometric, kinematic and inertial parameters to generate RW UAV responses for use in the identification of the coefficients (parameters) of the nonlinear model through regression techniques for the different flight regimes.
- To identify the acceleration model coefficients for the various flight regimes.
- To investigate and determine if the DDP can be applied in the design of RW UAV controller.

1.8 Design Assumptions

The following assumptions were made due to the constraints imposed on the schedule and as well as the flight simulation software package to be used:

1. The RW UAV will initially be in steady level forward flight
2. ISA sea level conditions
3. The weather is ideal, therefore no turbulence
4. Landing coordinates will be provided by the ground station.

1.9 Envisaged Contribution to Knowledge

The envisaged contributions to knowledge from this research are as follows:

- Application of DDP control approach on a RW UAV weighing 560 kg.
- Autonomous autorotion of 560 kg RW UAV following engine failure.

1.10 Research Methodology

Figure 1.7 illustrates the roadmap to be followed in designing the DDP controller. An acceleration model of the RW UAV will be formulated. FLTSIM helicopter flight simulation package will be used to yield necessary RW UAV response data for the autorotation regimes. The parameters will then be identified. A DDP control algorithm will be designed. The results obtained from the control algorithm will be compared to the flight simulation response data.

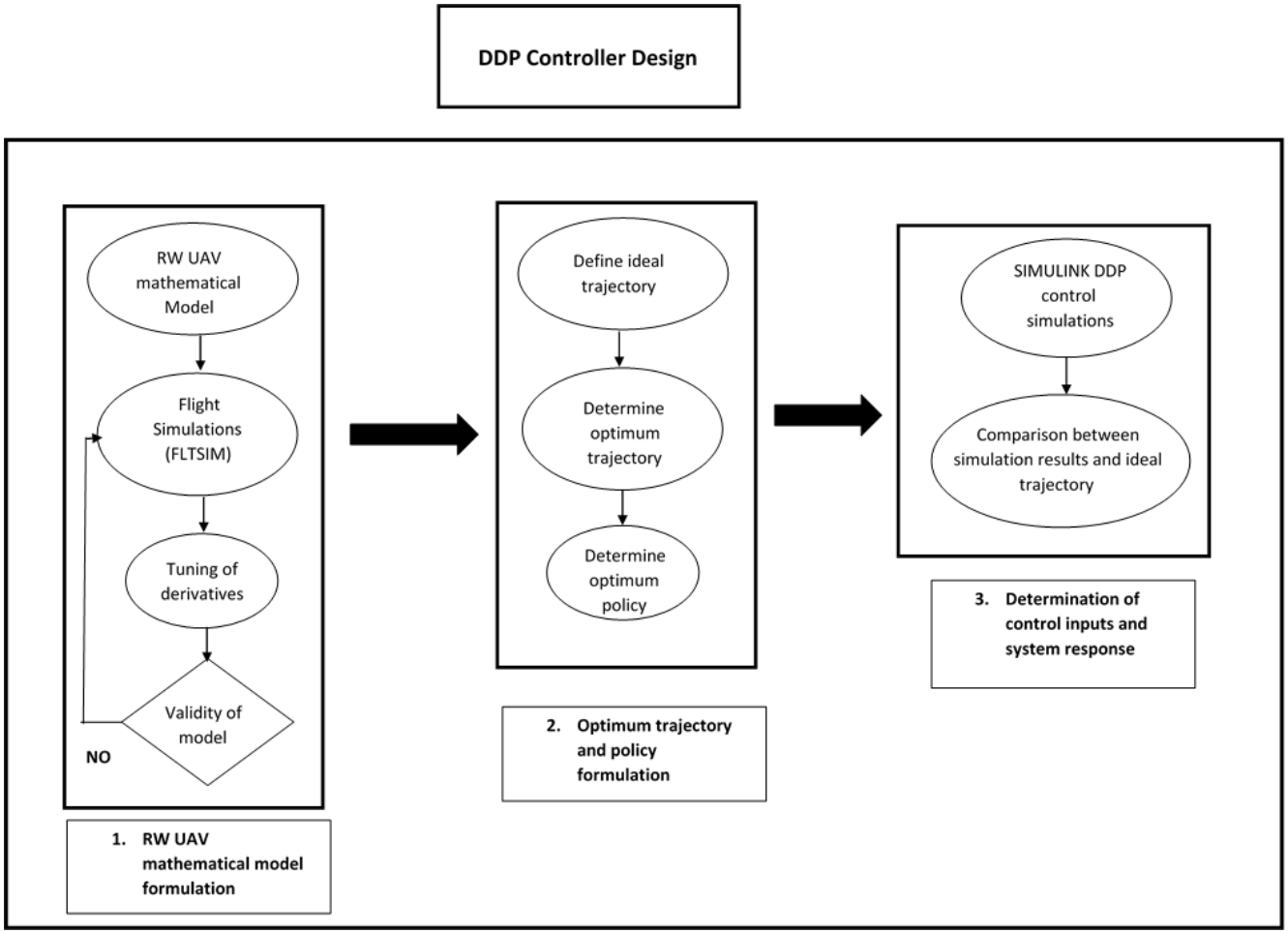


Figure 1.7: Roadmap to be followed in this research investigation

1.10.1 Research Helicopter

The aircraft proposed for the study is a rotary-wing UAV. The mass of the helicopter is approximately 500 kg and it has four blades with a diameter of 6 m, with conventional tail.

1.10.2 Mathematical Model

In order to obtain an understanding of the basic rotary-wing aircraft dynamics, a mathematical model of RW UAV will be established. The aerodynamic model will have the following subsystems:

- 6 degrees of freedom (DOF) equation of motion of the RWUAV with respect to the centre of gravity
- Main rotor model (3 DOF)
- Tail rotor model (3 DOF)
- Horizontal and vertical stabiliser aerodynamic model
- Fuselage aerodynamic model

From the understanding of the aerodynamic model, the acceleration model will be formulated. The acceleration model will be integrated with respect to time to obtain velocity, position, orientation, angular rate and main rotor speed. The helicopter model will be controlled via the cyclic pitch controls, the tail rotor collective and the main rotor collective pitch control.

1.10.3 Flight Simulation (FLTSIM)

A series of simulation runs will be performed with the available FLTSIM helicopter simulation package that yields RW UAV response data for prescribed control inputs covering the autorotation regimes for the flight condition mentioned. The responses obtained from the FLTSIM will used to identify the parameters of the acceleration model.

1.10.4 Offline Parameter Estimation Algorithms

FLTSIM helicopter software package will be populated with the RW UAV geometric, kinematic and inertial parameters to generate RW UAV responses to use for the identification of the parameters. MATLAB System Identification Toolbox will be used to identify the unknown parameters.

1.10.5 Autorotation Flight Controller

DDP will be applied for autorotation control design. The results will be compared to the FLTSIM response data.

1.10.6 Evaluation Criteria

The following criteria will be used to evaluate parameter estimation:

- Loss function of less than 0.001
- Akaike's Final Prediction Error (FPE) of less than 0.001

Attitude control will be the bases to validate the flight controller.

1.11 Proposed Control Methodologies

Optimal path planning strategies, presented by ([Johnson, 1977]), that minimize the vertical and horizontal velocities at ground contact using nonlinear optimal control have been tested in simulation and are computationally expensive and not suitable to be applied in real time. Machine learning approach in which the controller is trained with the pilot reference autorotation path has been successfully tested with a small UAV but it requires improvement since it does not perform well when trying to adapt itself to different initial conditions for autorotation.

Proposed automated autorotation methods have significant drawbacks such as: [Kensaku et al., 2004]

- The trajectory is not calculated on-line because the calculations cannot be carried out in real-time. This can result in discrepancies between the model

and the actual aircraft and any external disturbances can lead to accumulating error. This error may lead to catastrophic accident when the helicopter approaches touchdown.

- Autonomous autorotation is based on training using pre-recorded attempts by an expert. The limitations of the human pilot are incorporated into the design and the performance will be as good as the human pilot. It does not allow for different objectives and large deviations from the conditions under which the experiments were recorded.
- The controller is designed as a black box, trained through repeated, simulated trial and error and the controller needs to be repeatedly trained under all possible conditions.
- A fundamental problem in artificial intelligence and control is sequential decision making in stochastic systems.

Some reinforcement learning problems are [Jategaonkar et al., 2004]:

- The issue of high dimension, simple reinforcement learning algorithms based on discretization scale exponentially with the number of state variables.
- Reward function, the designer needs to specify a function that indicates when the helicopter is flying well or badly.
- Partial observability meaning the state of the system being controlled cannot be observed exactly, such as when a sensor on a helicopter measures the position approximately. Many standard reinforcement learning algorithms are inapplicable or become very difficult to apply.

1.11.1 Regulations and Certification

The acceptance of aerial vehicles by the certification authorities is a big challenge because they are limited by regulatory constraints. Regulations are put into place by national agencies (e.g, SACAA, FAA, National Air Traffic Services or Direction Generale de l'Aviation Civile) to maintain high levels of safety for air traffic. Safety is the primary objective of the regulations, and therefore pose a particular challenge for developers. Radio Technical Commission for Aeronautics (RTCA) has helped in establishing rules for the routine operation of aerial vehicles. [Ng, 2003]

The challenge for researchers is to develop the requirements and subsequent technology that meets the constraints set by the regulatory agencies or to propose and justify alternate constraints. Reliable components, defined maintenance procedures, formal training programs and the automation of emergency procedures (such as autorotation landing) are required for use of aerial vehicles in populated areas. Developing highly dependable systems and making such guarantees acceptable to the regulatory authorities is the challenge [Ng, 2003].

The principles of airworthiness need to be adhered to, which requires the behaviour of the control algorithm to remain deterministic for all possible sets of inputs and under all failure conditions.

1.12 Dissertation Outline

The layout of this dissertation is as follows: Chapter 2 describes the development of the RW UAV simulation model. Chapter 3 describes the flight simulation software package and it illustrates the manner in which it was applied. Chapter 4 describes the DDP methodology and design of the DDP controller for autonomous autorotative landing. Chapter 5 consists of the results and discussion of the results. Recommendations for this research are given in Chapter 6.

2 System Description and Mathematical Model

2.1 Description of the Unmanned Helicopter

The unmanned helicopter has a mass of 560 kg and a main rotor radius of 6 m. It has a conventional configuration with a fully articulated rotor.

2.2 Rotorcraft Modelling

Aircraft modelling falls within two categories, namely: first principles modelling and modelling using system identification. First principles modelling is a comprehensive analysis of the vehicle's physical features. Modelling using system identification focuses on developing compact and easier to understand models that capture the essence of the plant dynamics. This is achieved by estimating the system's responses from flight data collected during flight and ground experiments [Mettler, 2003].

The goal of the mathematical model is to develop a simulation valid for an autorotation manoeuvre Abbeel et al. (2009) have presented the first controller to successfully pilot a remotely controlled (RC) helicopter during an autorotation descent and landing. RW UAV mathematical model utilized is as proposed by Abbeel et al. (2009), as it has been demonstrated to be successful. A modular approach was undertaken which consists of a nonlinear dynamic model. Modelling using system identification was undertaken and hence the system's responses were estimated from flight simulation software package.

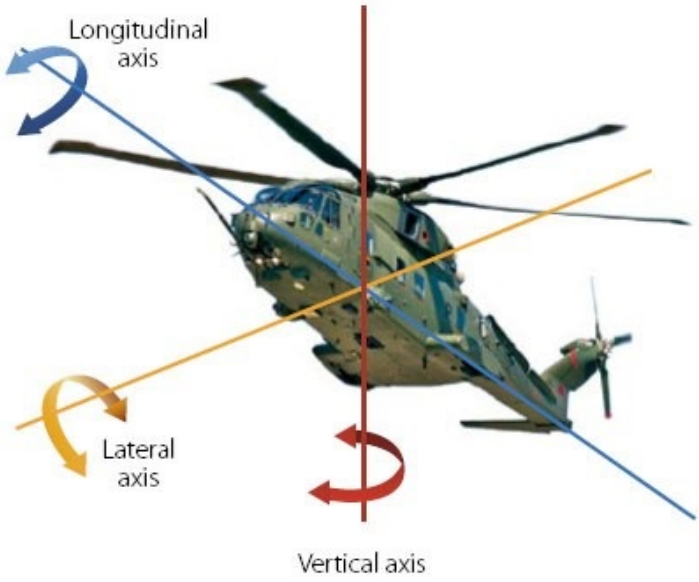


Figure 2.1: Helicopter axis

2.2.1 Nonlinear dynamic model

The application of Newton's laws of motion to a helicopter in flight leads to a set of nonlinear differential equations for the evolution of the aircraft response trajectory and attitude with time. The motion is referred to an orthogonal axes system fixed at the aircraft's c.g. The helicopter equations of motion in nonlinear form are given by:

$$\dot{x} = F(x, u, t) \tag{2.1}$$

In which the motion states are:

$$x = [u, v, w, p, q, r, \phi, \theta, \psi] \tag{2.2}$$

where u, v and w are the translational velocities along the three orthogonal directions of the fuselage fixed axes system described in Figure 2.1; p,q and r are the angular velocities about the x, y and z axes and θ, ϕ and ψ are the Euler angles defining the orientation of the body axes relative to the earth.

The control inputs are main rotor collective, longitudinal and lateral cyclic and tail

rotor collective as shown:

$$u = [\theta_0, \theta_{1s}, \theta_{1c}, \theta_{0T}] \tag{2.3}$$

The solution of equation 2.1 depends on the initial conditions of the motion state vector and the time variation of the vector function F(x,u,t), which includes the aerodynamic loads, gravitational forces and inertial forces and moments. The trajectory can be computed using various numerical integration schemes, achieving an approximate balance of the component accelerations with the applied forces and moments at every time step. However, numerical integration offers little insight into the physics of the aircraft behaviour. Analytic solutions provide a deeper understanding between the cause and effect. Unfortunately, the scope for deriving analytic solutions of general nonlinear differential equations as in Eq. (2.1) is extremely limited; only in special cases can functional forms be found and, even then, the range of validity is likely to be very small. [Padfield, 2008]

The expanded form of Eq (2.1) can be written as follows:

$$\dot{u} = vr - wq - gsin(\theta) + \frac{X}{m} \tag{2.4a}$$

$$\dot{v} = wp - ur + gcos(\theta)sin(\phi) + \frac{Y}{m} \tag{2.4b}$$

$$\dot{w} = uq - vp + gcos(\theta)cos(\phi) + \frac{Z}{m} \tag{2.4c}$$

$$\dot{p} = \frac{I_{yy} - I_{zz}}{I_{xx}}qr + \frac{I_{xz}}{I_{xx}}(\dot{r} - pq) + \frac{L}{I_{xx}} \tag{2.4d}$$

$$\dot{q} = \frac{I_{zz} - I_{xx}}{I_{yy}}rp + \frac{I_{xz}}{I_{yy}}(r^2 - p^2) + \frac{M}{I_{yy}} \tag{2.4e}$$

$$\dot{r} = \frac{I_{xx} - I_{yy}}{I_{zz}}pq + \frac{I_{xz}}{I_{zz}}(\dot{p} - qr) + \frac{N}{I_{zz}} \tag{2.4f}$$

$$\dot{\phi} = p + qsin(\phi)tan(\theta) + rcos(\phi)tan(\theta) \tag{2.4g}$$

$$\dot{\theta} = qcos(\phi) + rsin(\phi) \tag{2.4h}$$

$$\dot{\psi} = qsin(\phi)sec(\theta) + rcos(phi)sec(\theta) \tag{2.4i}$$

where:

$\dot{u}$ = longitudinal acceleration, positive forward, [m/s2];

$\dot{v}$ = lateral acceleration, positive to the right, [m/s2];

$\dot{w}$ = vertical acceleration, positive downwards, [m/s2];

u = longitudinal velocity, positive forward, [m/s];

v = lateral velocity, positive forward, [m/s];

w = vertical velocity, positive downward, [m/s];

$\dot{p}$ = roll angular acceleration, positive to the right, $[rad/s^2]$;

$\dot{q}$ = pitch angular acceleration, positive nose up,$[rad/s^2]$;

$\dot{r}$ = yaw angular acceleration, positive to the right, $[rad/s^2]$;

p = roll rate, positive to the right, [rad/s];

q = pitch rate, positive nose up, [rad/s];

r = yaw rate, positiveto the right, [rad/s];

I_{xx} = roll moment of inertia,$[kg.m^2]$;

I_{zz} = pitch moment of inertia, $[kg.m^2]$;

I_{xz} = product of inertia, $[kg.m^2]$;

Helicopters have highly complex dynamics and capturing the state of the "helicopter system" would include the state of the air around the helicopter into the dynamics model. But various works (see, eg., [Mettler et al., 1999], [Gavrilets et al., 2002], [Coates et al., 2008]) have shown it possible to build a sufficiently accurate model for control by treating the helicopter as a rigid body, perhaps including the blade

flapping dynamics and the main rotor speed [Abbeel et al., 2009].

The helicopter is modelled with thirteen-dimensional state consisting of position, orientation, velocity, angular rate and main rotor speed and with the helicopter inputs being the cyclic pitch controls, which prompt the helicopter to pitch forward/backward or sideways; the tail rotor which affects the tail rotor thrust and the yaw motion of the helicopter; the main rotor collective pitch which changed the main rotor thrust by changing the pitch of the rotor blades [Abbeel et al., 2009].

The effects of inertia have been subtracted from the dynamic model as shown in the set of Eqs. (2.5). The model was used to predict the accelerations in a coordinated frame attached to the helicopter.

$$\dot{u} = v * r - w * q + g_u + C_u * u \tag{2.5a}$$

$$\dot{v} = w * p - u * r + g_v + C_v * v \tag{2.5b}$$

$$\dot{w} = u * q - v * p + g_w + C'_w[1; w; \theta_0 * \Omega; \sqrt{u^2 + v^2}] \tag{2.5c}$$

$$\dot{p} = C'_p * [1; p; \theta_{1c} * \Omega] \tag{2.5d}$$

$$\dot{q} = C'_q * [1; q; \theta_{1s} * \Omega] \tag{2.5e}$$

$$\dot{r} = C'_r * [1; r; \theta_{0T} * \Omega] \tag{2.5f}$$

$$\dot{\Omega} = C'_\Omega * [1; \Omega; \theta_0; w; \sqrt{u^2 + v^2}; (\theta_{1c}^2 + \theta_{1s}^2)] \tag{2.5g}$$

where C_u, C_v, C_w, C_p, C_q, C_r and $C_{\dot{\Omega}}$ are the derivatives to be identified. Each derivative is made up of a contribution from the different aircraft components but the main rotor being dominant in helicopter flight dynamics.

2.2.2 Autorotation main rotor equation

In autorotation, instead of relying on the engine to drive the main rotor, the helicopter has to be controlled such that the potential energy from height above the ground is transferred to rotor speed. Management of the main rotor speed to maintain sufficient rotor speed throughout the events following engine failure up to and during execution of the landing is critical in order to prevent a crash of the craft. The equation below is specific to autorotation and it was obtained from Abbeel et al., (2009).

$$\dot{\Omega} = C'_{\Omega} * [1; \Omega; \theta_0; w; \sqrt{u^2 + v^2}; (\theta_{1c}^2 + \theta_{1s}^2) \tag{2.6}$$

where:

$\dot{\Omega}$ = main rotor acceleration, $[rad/s^2]$;

C_{Ω} = Main rotor derivative vector.

Equation (2.6) was used to model the main rotor dynamics.

2.2.3 Sensor model

The required instrumentations on the helicopter are the Inertial Measurement Unit (IMU), Global Positioning System (GPS), radio altimeter (RADALT) and tachometer.

Inertial Measurement Unit

The IMU uses gyros and accelerometers to measure the angular rates and translational accelerations. A GPS is often included to tack the helicopters relative position and velocity. Although the developed SIMULINK did not take into account rotational dynamics of the GPS box and feedback lags, it needs to be modelled when real flight data is collected.

Radio Altimeter

The RADALT measures the height above the ground of the aircraft by transmitting radio waves towards the ground and by measuring the time it takes the waves to be reflected back.

Tachometer

A tachometer measures the main rotor rotational speed.

The sensors mentioned above would be required to measure the flight parameters but have not been included in the mathematical model.

2.3 Simulation Model

A simulation facility for the helicopter model comprising nonlinear equations was formulated in MATLAB/Simulink. Simulink provides a graphical user interface (GUI) for building models as block diagrams. It has a comprehensive block diagram simplifying the modelling process and enabling realistic nonlinear models to be explored.

A modular approach was implemented in creating the simulation model. The simulation model consists of the helicopter model. Figure 2.2 illustrates the helicopter model.

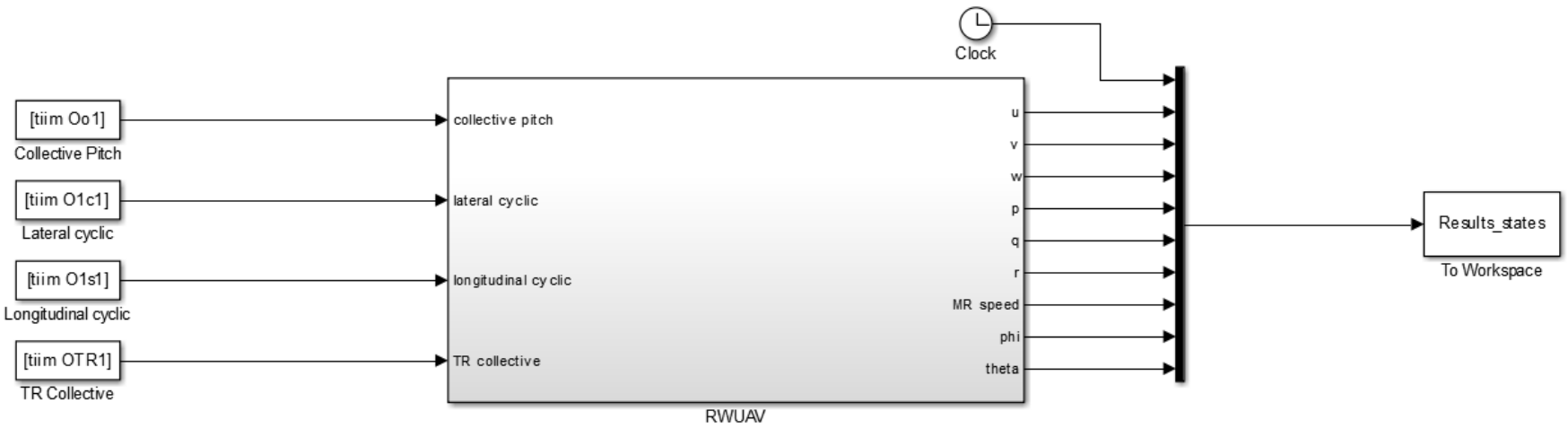


Figure 2.2: Helicopter model

3 Description and Application of Flight Simulation Software Package (FLTSIM)

3.1 Introduction

System identification is a procedure to determine an adequate mathematical model, usually containing differential equations with unknown parameters which have to be determined indirectly from measured data. The process includes performing suitable experiments and gathering system inputs and responses to determine the unknown parameters. The process of system identification involves the following fundamental assumptions [Jategaonkar, 2006]:

- The true state of the dynamic system is deterministic.
- The physical principles of the dynamics can be modelled.
- It is possible to conduct specific experiments.
- Measurements of the system inputs and outputs are available.

The parameters of a specified model are quantified by applying a numerical procedure. This part of the overall model building process is called "parameter estimation". Parameter estimation is followed by a step called "model validation" to assess the model fidelity [Jategaonkar, 2006].

A fundamental principle of empirical sciences suggests complimentary flight data not used in the parameters estimation to check the model predictive capability. However, there is another philosophy which advocates incorporating the maximum amount of available flight data in the model development and not dividing the flight data into two sets. The logic is that the identification process would automatically lead to a model with adequate fidelity, satisfying the validation tests [Jategaonkar, 2006].

To identify the control and stability derivatives in the mathematical model, a flight simulation package was used to collect flight simulation data that was used for parameter estimation. After the flight simulation data was obtained a MATLAB System Identification Toolbox was used to identify the control and stability derivatives.

3.2 Flight Simulation Software Package

The flight simulation package, FLTSIM, is a helicopter design tool used in the evaluation of various aspects of the aerodynamic design of helicopters. The program is a stand-alone software package written in FORTRAN that runs on a PC-based platform. The program is capable of analysing most conventional helicopter configurations.

FLTSIM has six main run time options, namely:

- Option 1 – Aerodynamic coefficients
- Option 2 – Trimmed flight analysis
- Option3 – Flying qualities analysis
- Option 4 – Nonlinear simulation
- Option 5 – Linear simulation
- Option 6 – Performance

When the program runs, the user is presented with the main menu from which one out of the six main options can be selected. Choosing one of the options causes the program to read the previously edited input file associated with that option and to then execute the relevant case. The program consists of an input file which contains information such as convergence factors, integration method and time step etc., which is used globally in the software. On completion of the run, a graphics management menu is displayed allowing the user to select the parameters to be plotted, axes scaling, line and marker types etc. depending on the option chosen, then the plot data will be written in particular output files.

Options 2 and 4 were used for the autorotation flight simulation of the RW UAV.

3.2.1 Trimmed flight analysis

When this option is run the software attempts to trim the aircraft based on the input parameters of the selected case using an iterative process.

3.2.2 Nonlinear simulation

This option uses a full non-linear simulation model to predict the dynamic behaviour of the selected helicopter when subjected to a set of predefined control inputs. The S80SIM.DTA has three time-based input tables as shown in Appendix A. The first table consists of a series of pilot control inputs defined as a percentage of full control movement. The second table defines the generic autopilot command values which are activated when the relevant autopilot channel is switched on. The third table is reserved for the definition of the stabilator pitch angle when the moving stabilator option is switched on.

3.3 Flight Simulation Flight Conditions

The parameters were identified for two flight conditions, ISA sea level and "Hot and High". "Hot and High" is at an altitude of 1524 m (5000 ft) and a temperature of 32 degrees Celsius.

3.3.1 Flight simulation

The autorotation flight simulation was split into two phases: entry into autorotation and glide; flare and land. It was assumed the RW UAV is initially in forward level flight and after 3 seconds the engine fails and the aircraft executes the autorotation manoeuvre. At a certain height above the ground and with a combination of main rotor speed and forward speed the aircraft has the ability to execute autorotation. This autorotation envelope was determined using Figure 3.1 and a trim simulation FLTSIM option. The aircraft attitude and speed was determined for forward level flight and for the engine failure case.

Upon engine failure, the power as the engine spools down was represented with Eq.

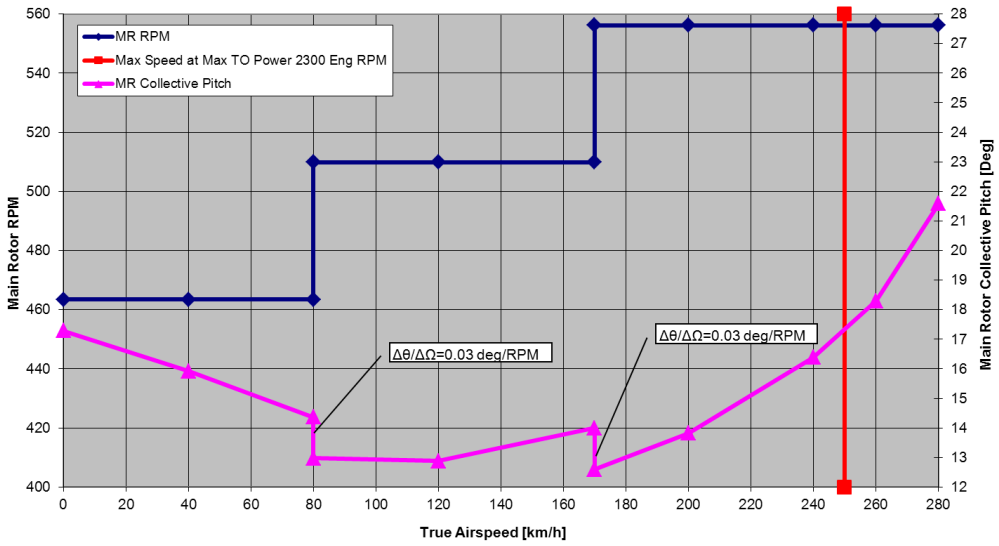


Figure 3.1: Main rotor RPM and collective pitch vs. true airspeed

(3.1).

$$P = P_0 e^{\frac{-t}{\tau_1}} \tag{3.1}$$

where:

P = power after 3 seconds, [kW];

P_0 = power before engine failure, [kW];

t = time, [s];

τ_1 = time constant for the spool-down response, [s].

The time constant for the spool-down response was taken as 0.5 s and it was obtained from the Rooivalk helicopter flight test data. The Rooivalk was flown to a certain height above the ground and the engines were shut-off to record or observe the spool-down response of the power and behaviour of the torque. Figure 3.2 illustrates the behaviour of the engine power when the engine fails.

When the engine fails, it is important to decrease the main rotor collective pitch

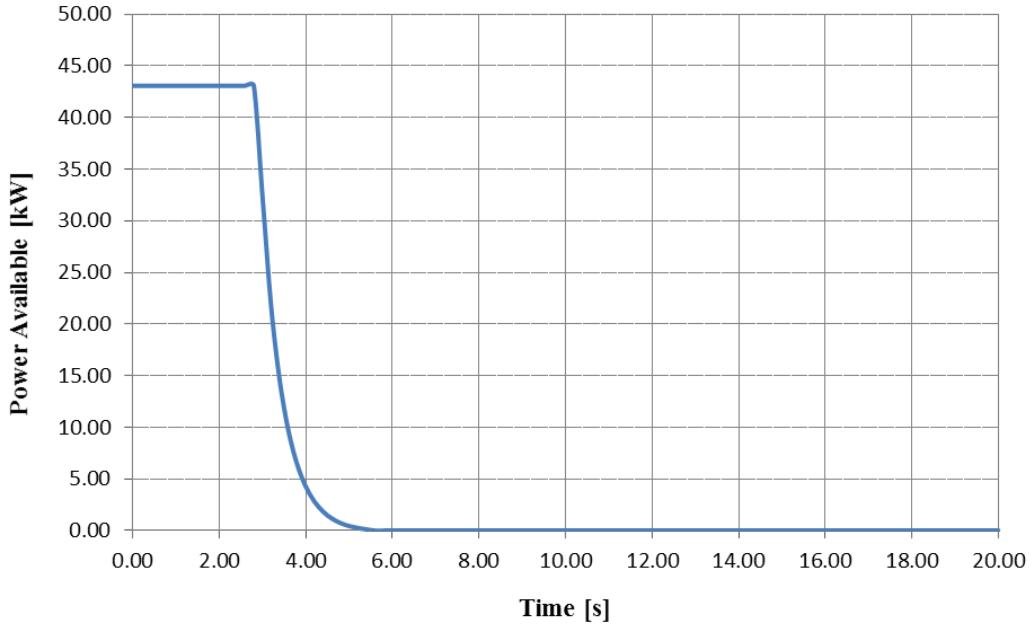


Figure 3.2: Engine power spool-down response following engine failure

angle to maintain the main rotor speed. The main rotor collective pitch angle was decreased a second after engine failure to account for the time it would take to detect engine failure. This is shown in Figure 3.3.

The plots that follow illustrate the behaviour of the RW UAV after engine failure for entry into autorotation and glide phase. The key parameters shown are the forward speed, vertical speed and the main rotor speed of the aircraft, as illustrated in Figures 3.4 to 3.6.

It can be seen from the plots that when the RW UAV enters into autorotation, it becomes untrimmed, shown by the oscillations. It recovers due to the decrease in main rotor collective pitch. The figures shown are for the sea level conditions.

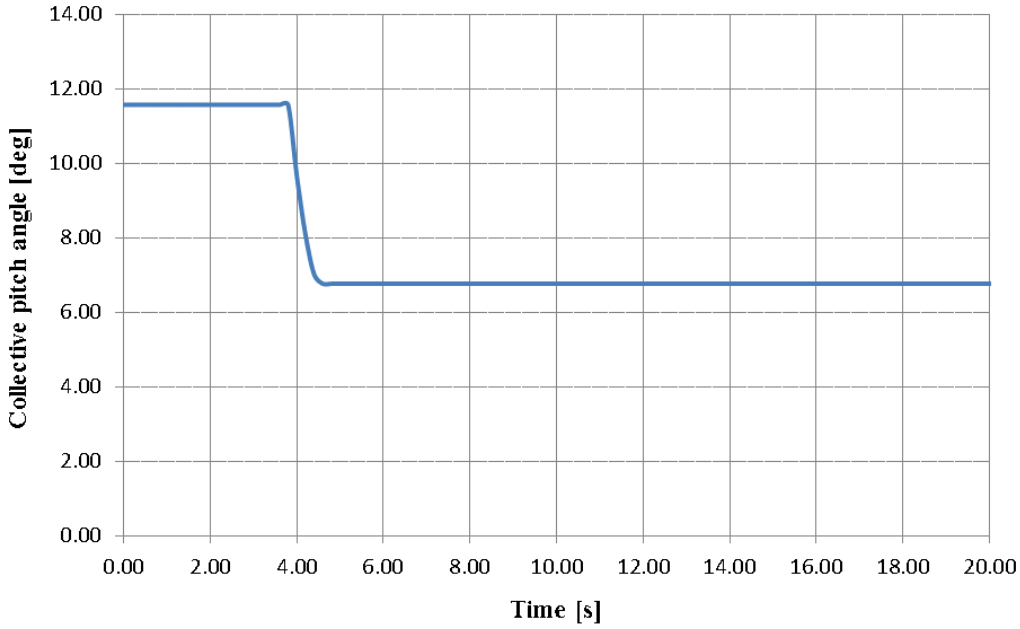


Figure 3.3: Collective pitch angle response following engine failure

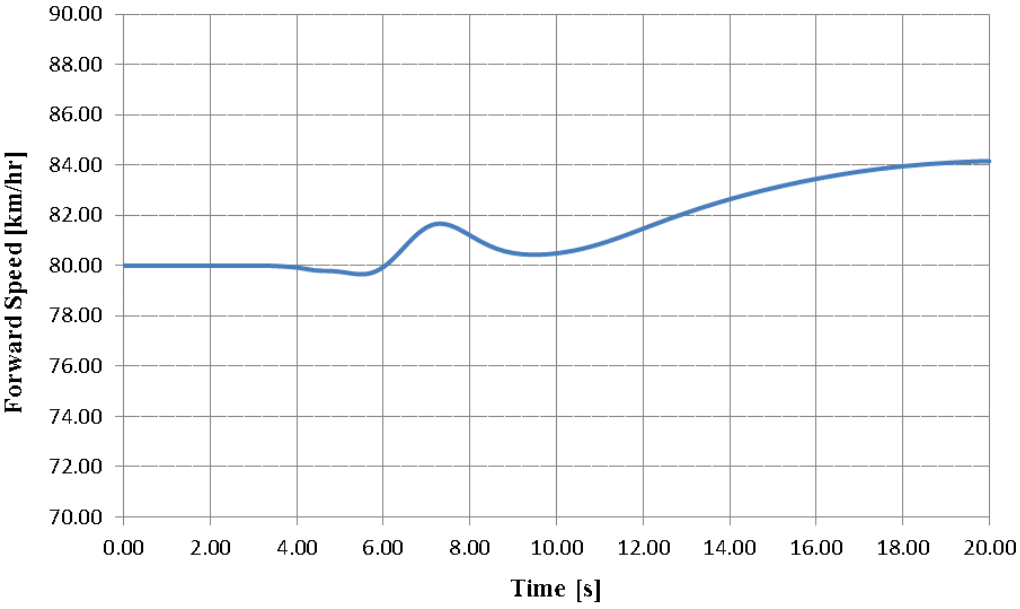


Figure 3.4: Forward speed response following engine failure

At a certain height above the ground, the RW UAV will flare. The main rotor blade inertia plays a very important role when the helicopter transitions from glide to flare during autorotation. Management of the main rotor speed to maintain

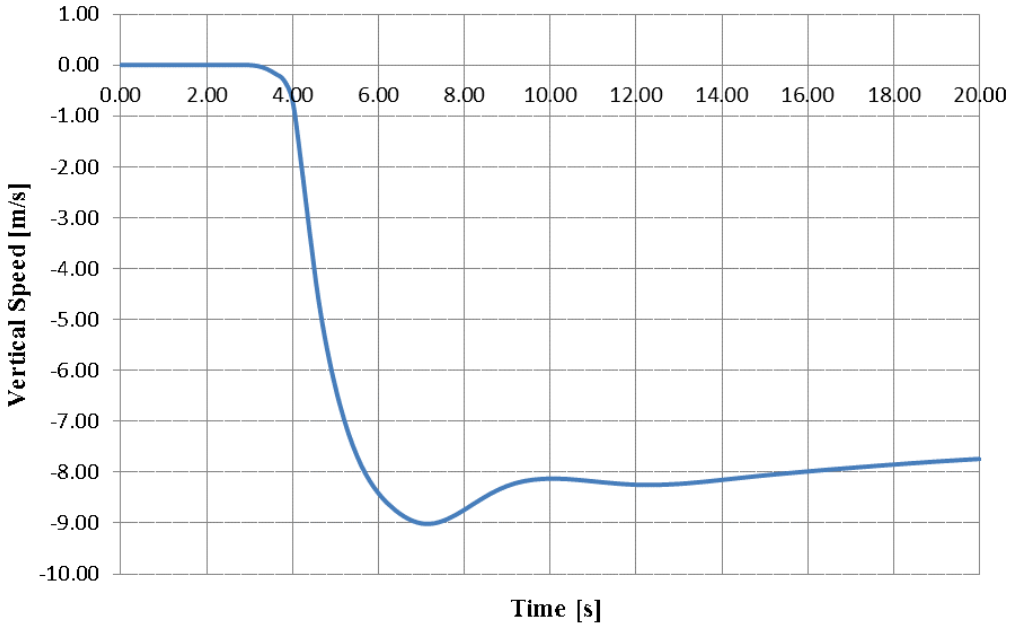


Figure 3.5: Vertical speed response following engine failure

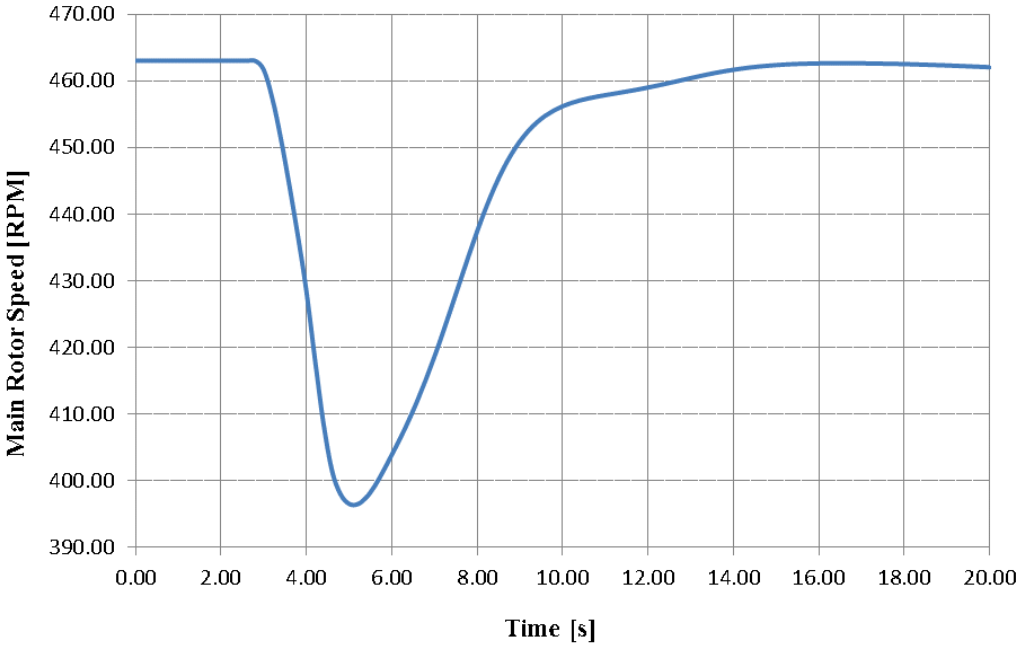


Figure 3.6: Main rotor speed response following engine failure

sufficient rotor speed throughout the events following engine failure up to and during execution of the landing is critical in order to prevent a crash of the helicopter.

The set of plots to follow illustrate the flare and land phase. Figure 3.7 shows forward speed vs. time, as the RW UAV approaches the ground the main rotor collective pitch angle was increased to decrease the forward speed and rate of descent, see Figure 3.9.

Figures 3.8 and 3.10 are the idealized flare manoeuvre taken from Prouty. An idealized flare manoeuvre starts with constant collective pitch in which increased rotor thrust and its aft tilt are used to decrease both the vertical and the horizontal velocity components. At the end of the flare, the aircraft should be near the ground with its vertical component zero [Prouty, 1995].

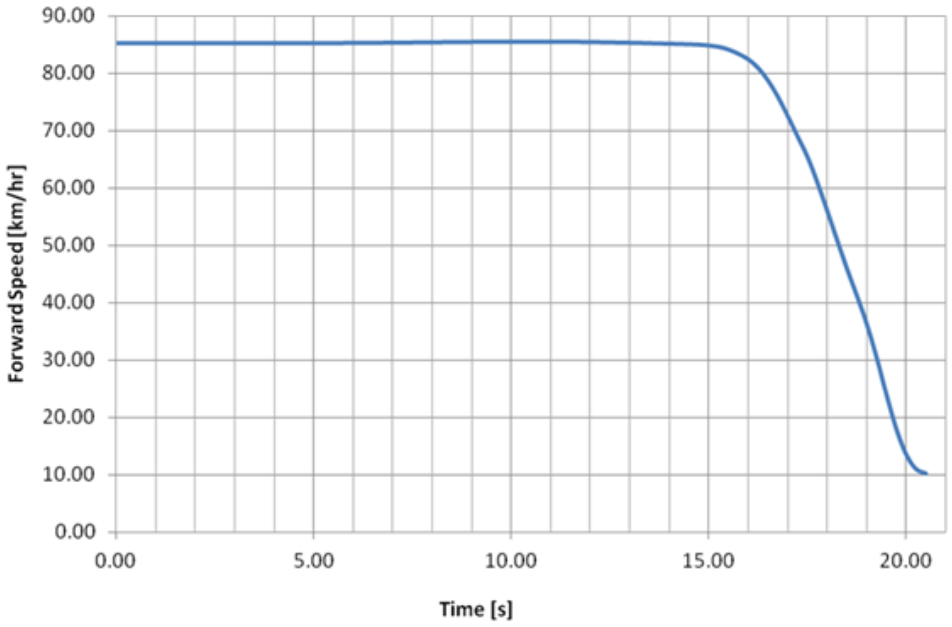


Figure 3.7: Forward speed response during flare and land

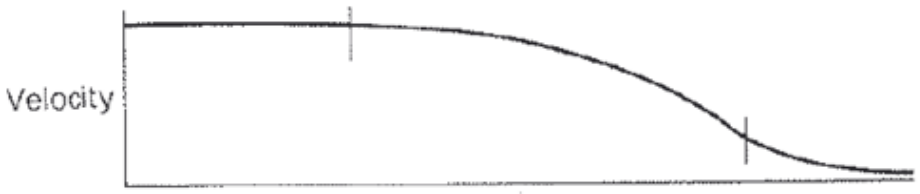


Figure 3.8: Autorotation velocity profile [Prouty, 1995]

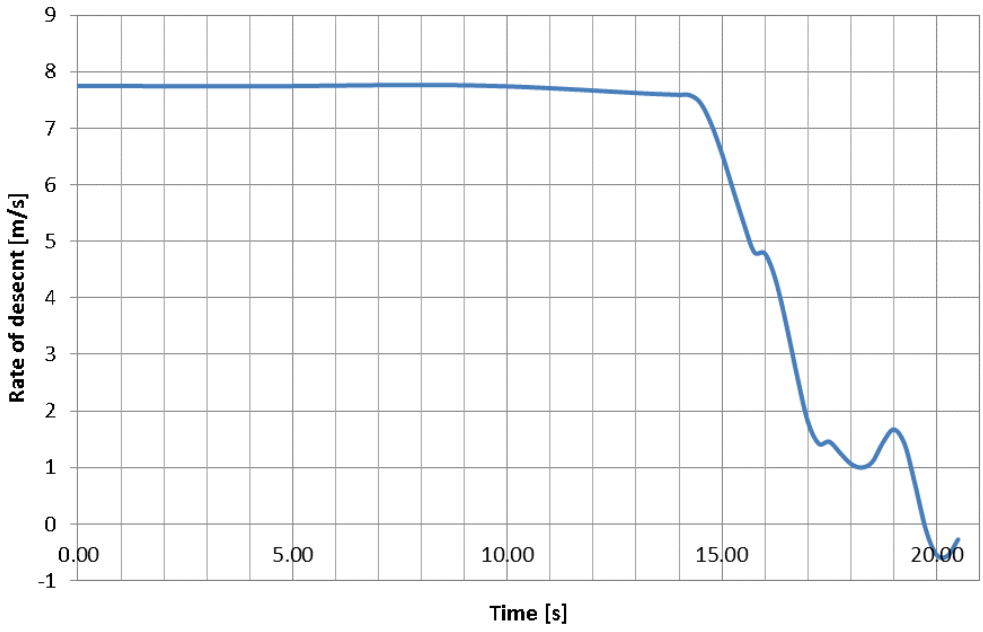


Figure 3.9: Rate of descent response during flare and landing

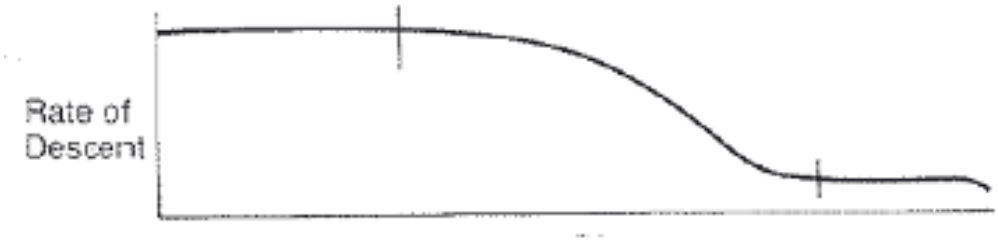


Figure 3.10: Ideal autorotation rate of descent profile [Prouty, 1995]

The main rotor collective pitch angle is shown in Figure 3.11. When the RW UAV was approaching the ground, collective pitch angle was increased then decreased to maintain main rotor speed. Just before touchdown, the main rotor collective pitch angle is increased to its maximum value to decrease the speed. The ideal autorotation collective pitch angle is shown in Figure 3.12.

The main rotor speed was maintained up until the main rotor collective pitch angle was increased, see Figure 3.13. The ideal autorotation main rotor speed is shown in Figure 3.14. Comparing flight simulation parameters with the ideal autorotation

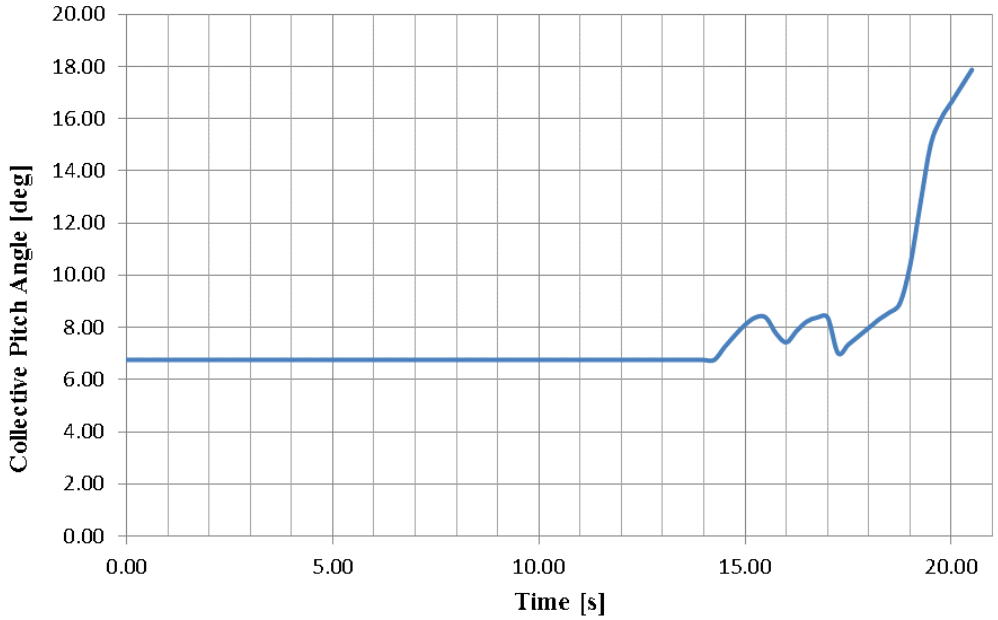


Figure 3.11: Collective pitch angle response during flare and landing

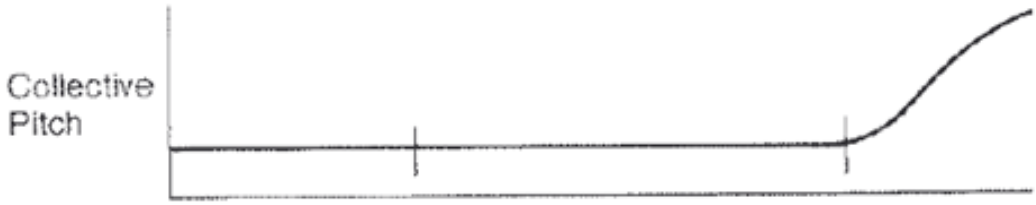


Figure 3.12: Ideal autorotation collective pitch angle profile [Prouty, 1995]

profile parameters, it can be seen that there is a lot of similarity.

3.4 Parameter Estimation

Several flight simulations of the RW UAV executing an autorotation manoeuvre in FLTSIM were run to obtain the best autorotation manoeuvre. The data was collected and utilised in the MATLAB System Identification Toolbox. A suitable model structure is prerequisite before estimation and the choice of the model structure is based on the understanding of the physical system. The black-box model, grey-box model and user-defined model are the three common types of models in

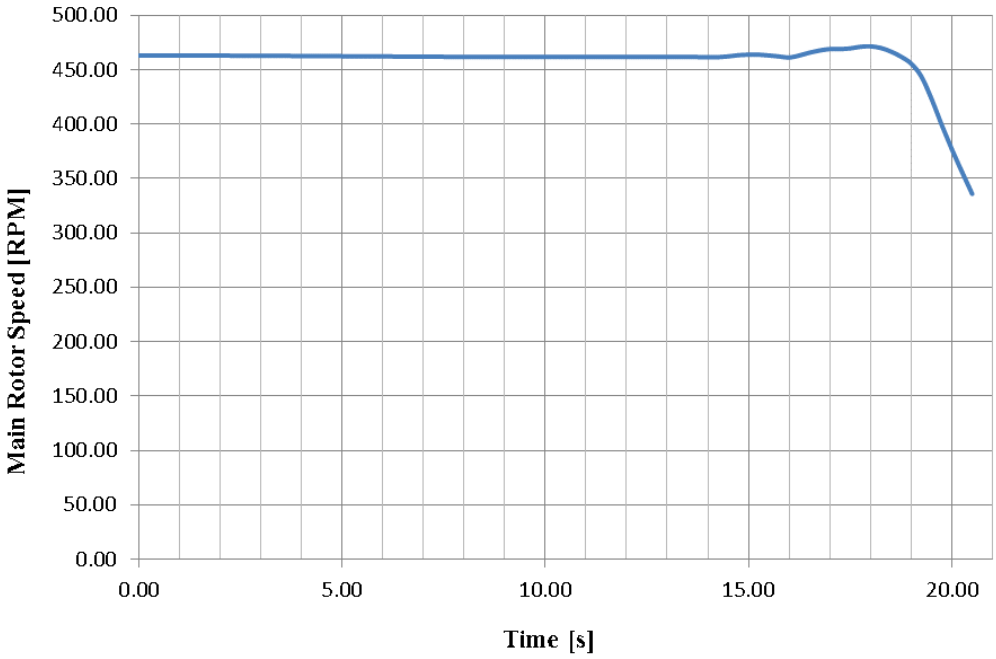


Figure 3.13: Main rotor speed response during flare and landing

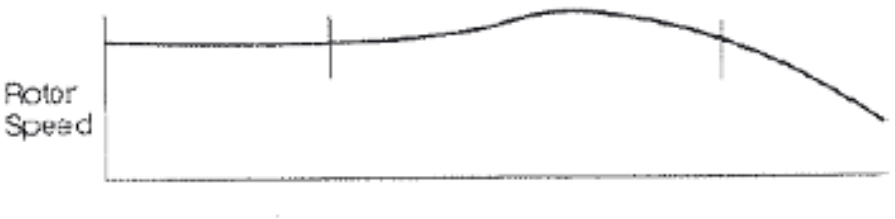


Figure 3.14: Ideal autorotation main rotor speed profile [Prouty, 1995]

system identification. The black-box model assumes the system is unknown and all the model parameters are adjustable without taking into account the physical background. The grey-box model assumes some of the information about the dynamics or physical parameters are known. The user-defined model assumes commonly used parametric models cannot represent the model that has to be estimated.

The grey box modelling tool within System Identification Toolbox was used because the physics of the system was understood and the system could be represented using ordinary differential equations with unknown parameters. The mathematical structure of the model was specified explicitly, including couplings between unknown

parameters and known parameter values.

The state-space model was setup in which there were 4 inputs, 9 states, 7 outputs and 21 parameters to be identified. The output variables are the translational velocity, angular rates and main rotor speed. The input variables are the main and tail rotor collective angle, longitudinal and lateral cyclic pitch angles. The state variables are the translational velocities, main rotor speed and roll, pitch and yaw angles. See Appendix B for the grey box modelling setup.

Figure 3.15 shows the height above ground vs. time for the autorotation flight simulation manoeuvre. In order to identify the derivatives, the flight simulation manoeuvre was divided into time intervals as illustrated by the red lines to increase the accuracy of the identified model and to ensure the dynamics of the model are captured at the phases of the manoeuvre.

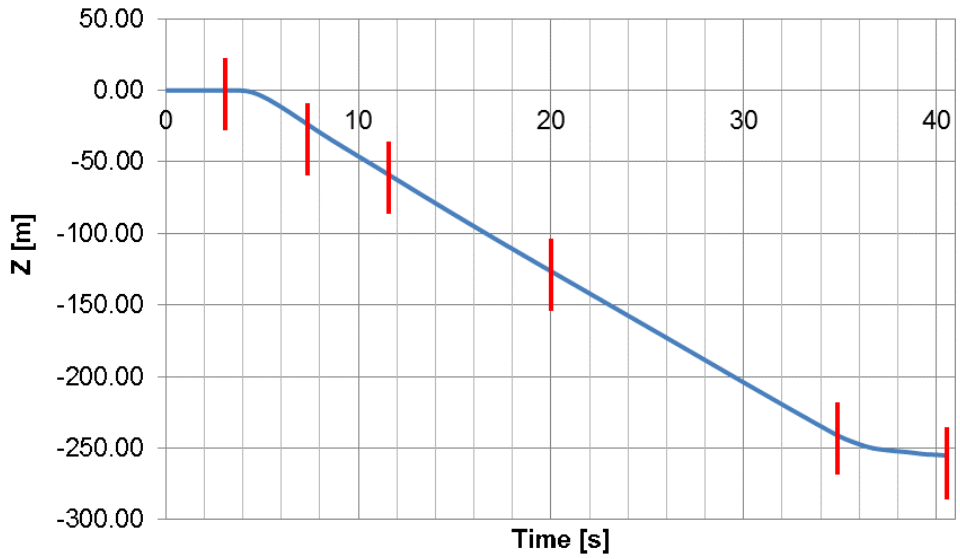


Figure 3.15: Change of height above the ground during autorotation

Figure 3.16 illustrates the control inputs vs. time. The first three seconds show the control inputs before engine failure and the fourth second was when the main and tail rotor collective pitch angle were dropped to maintain the main rotor speed after engine failure. Using the input and output data collected from the autorotation flight simulations in conjunction with the MATLAB System Identification Toolbox, the derivatives were identified.

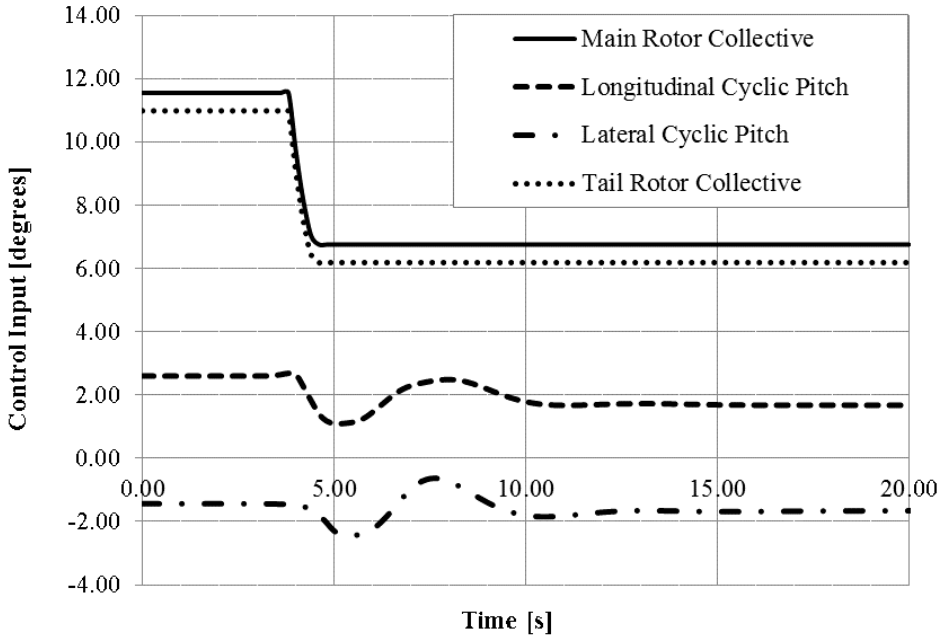


Figure 3.16: FLTSIM Control inputs for the first 20 seconds

3.5 Parameter Estimation Results

The figures that follow illustrate the fit obtained between the flight simulation and the identified model. The plots shown, illustrate the response following engine failure. As indicated in Figure 3.17, "Validation data" is the flight simulation data response and "Nonlinear Autorotation Simulation" is the estimated model. It should be noted that the Matlab solver integrated with smaller time steps and only sampled every 0.25 seconds. The parameter estimation results are shown for the last few seconds of the autorotation manoeuvre, the flare and land phase. In this phase of flight the RW UAV is dynamic compared to the glide phase. The other phases of flight results are not as good as compared to the flare and land phase because the inputs are small and the RW UAV is stable. The rest of the parameter estimation plots are illustrated in Appendix C.

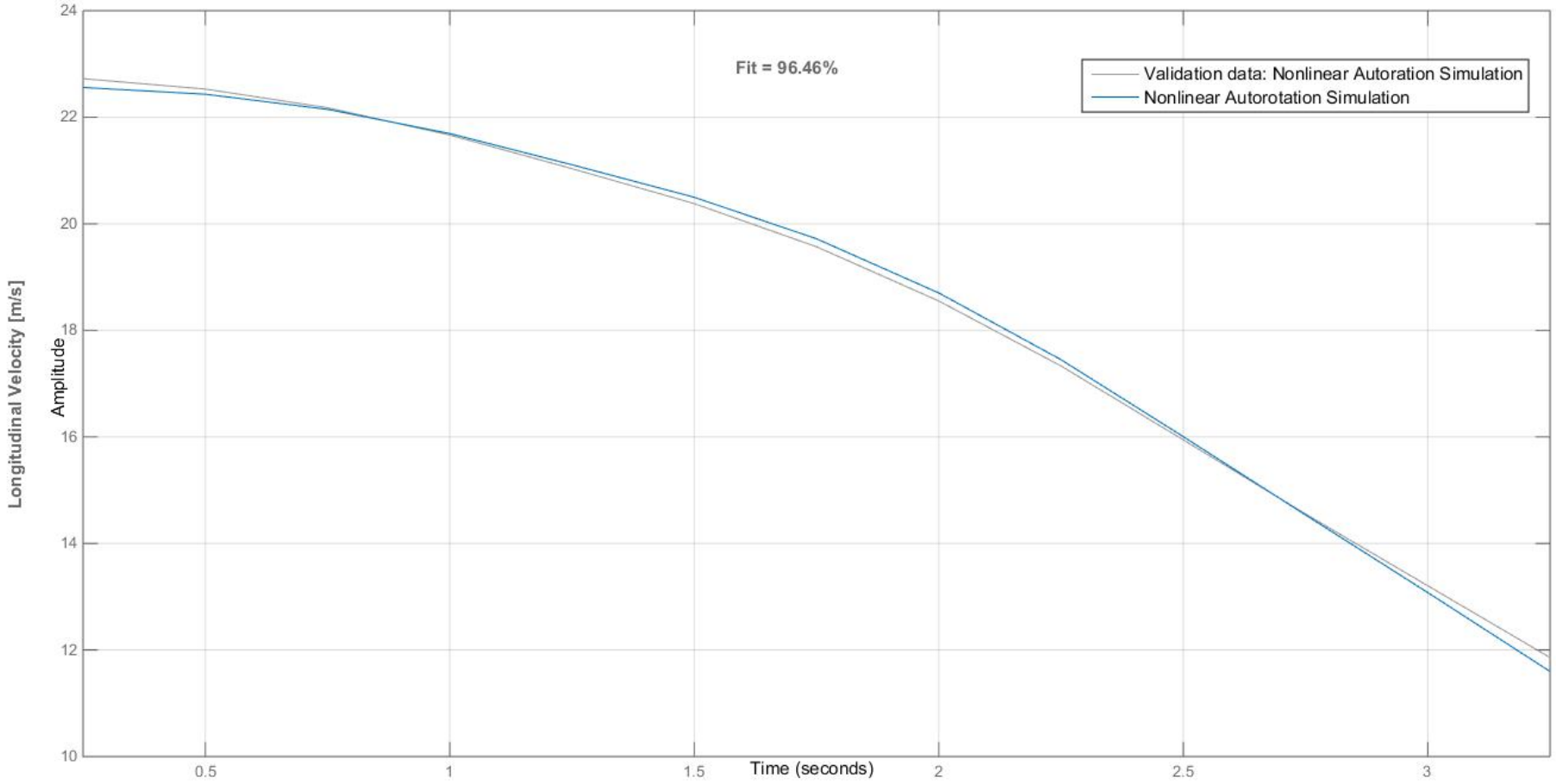


Figure 3.17: Longitudinal speed comparison between the validation data and identified data

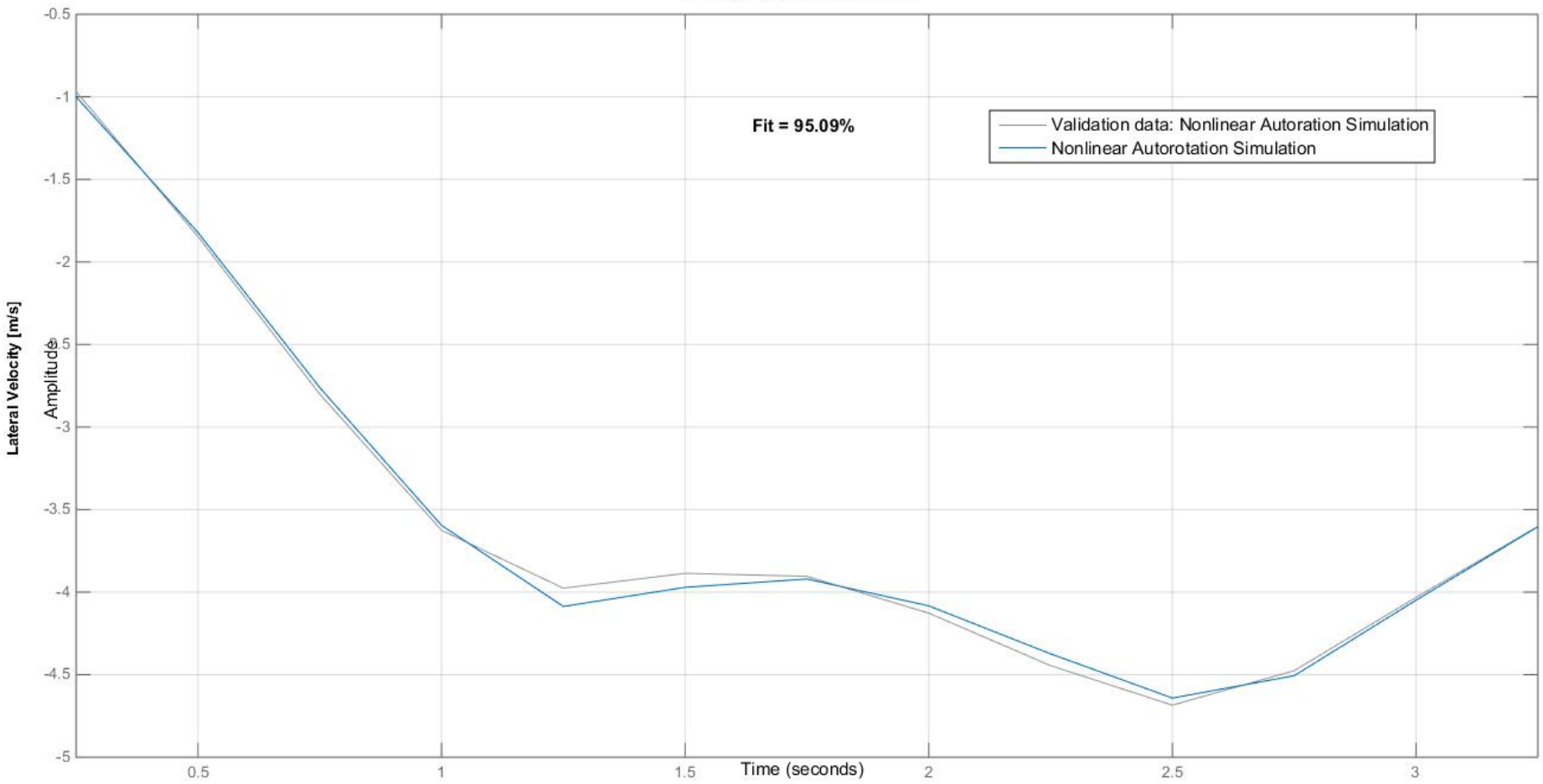


Figure 3.18: Lateral speed comparison between the validation data and identified data

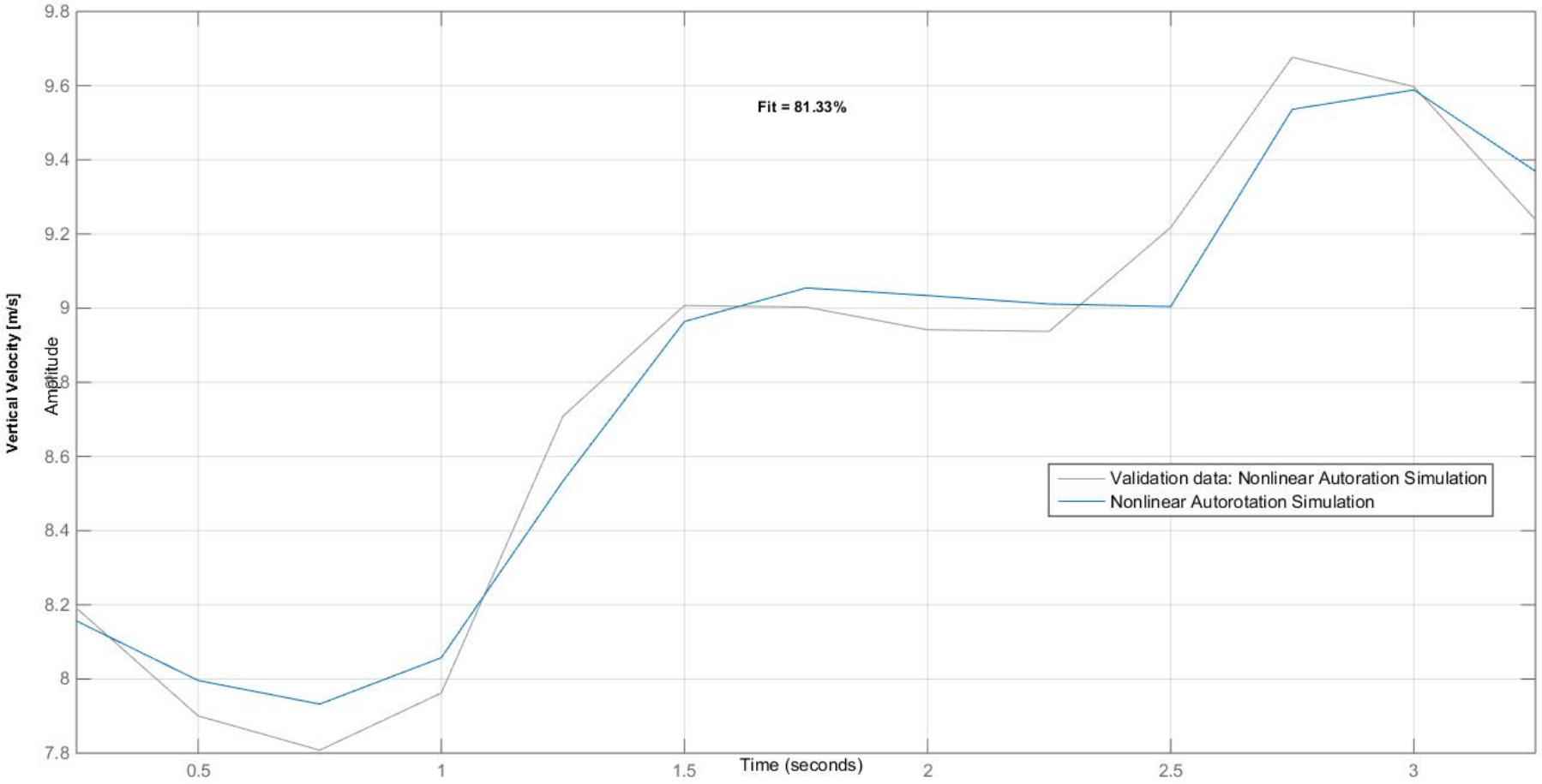


Figure 3.19: Vertical speed comparison between the validation data and identified data

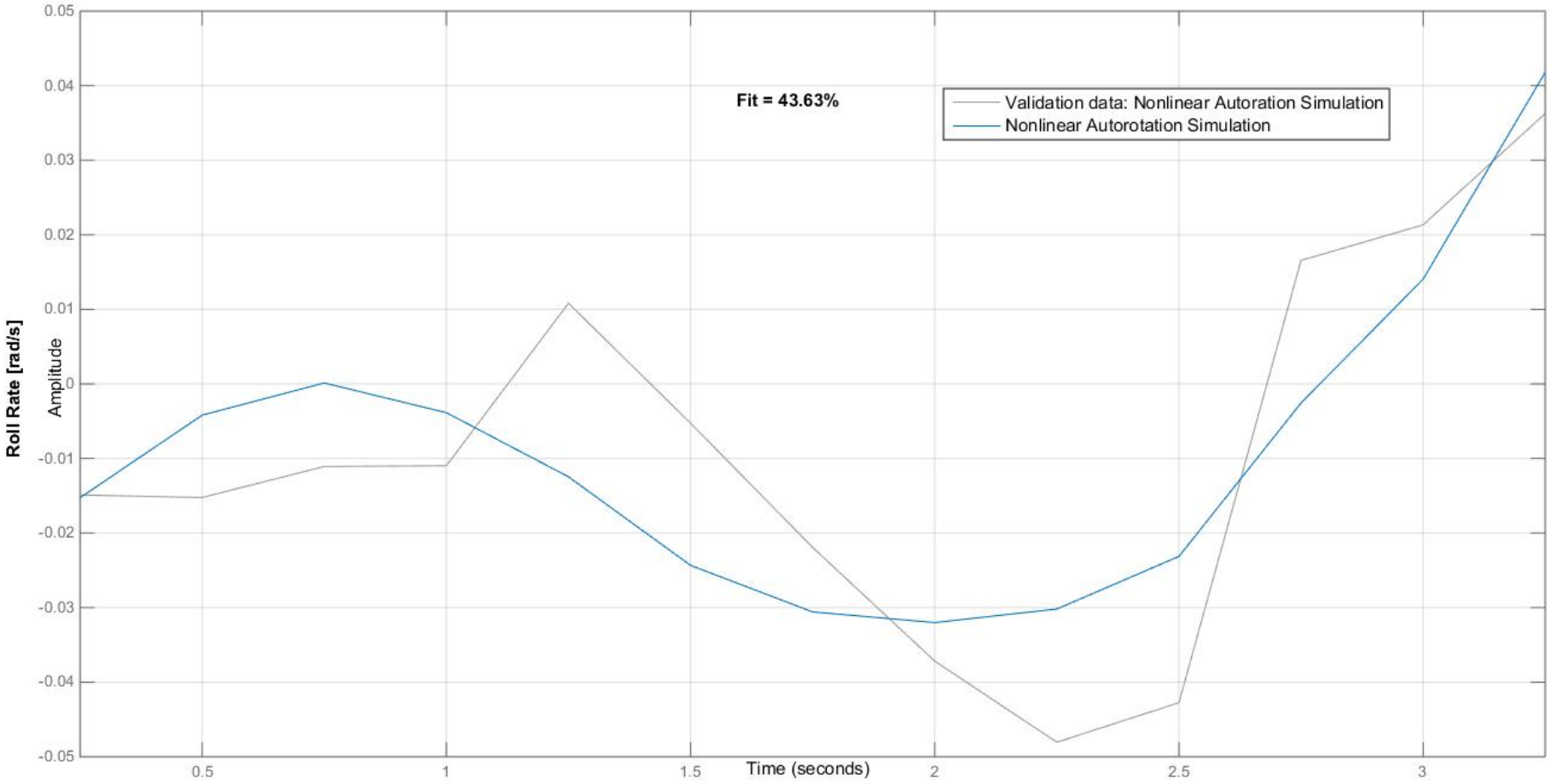


Figure 3.20: Roll rate comparison between the validation data and identified data

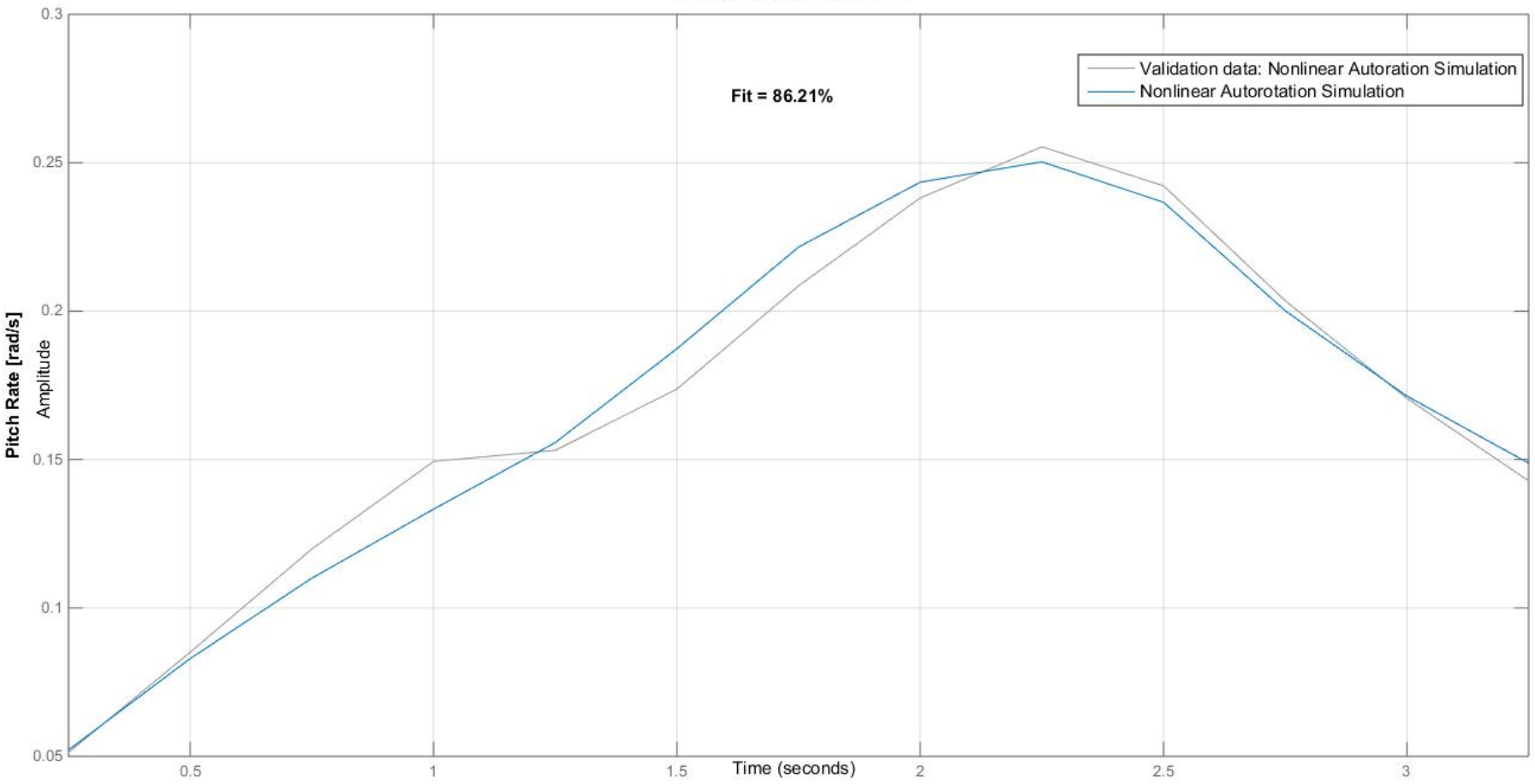


Figure 3.21: Pitch comparison between the validation data and identified data

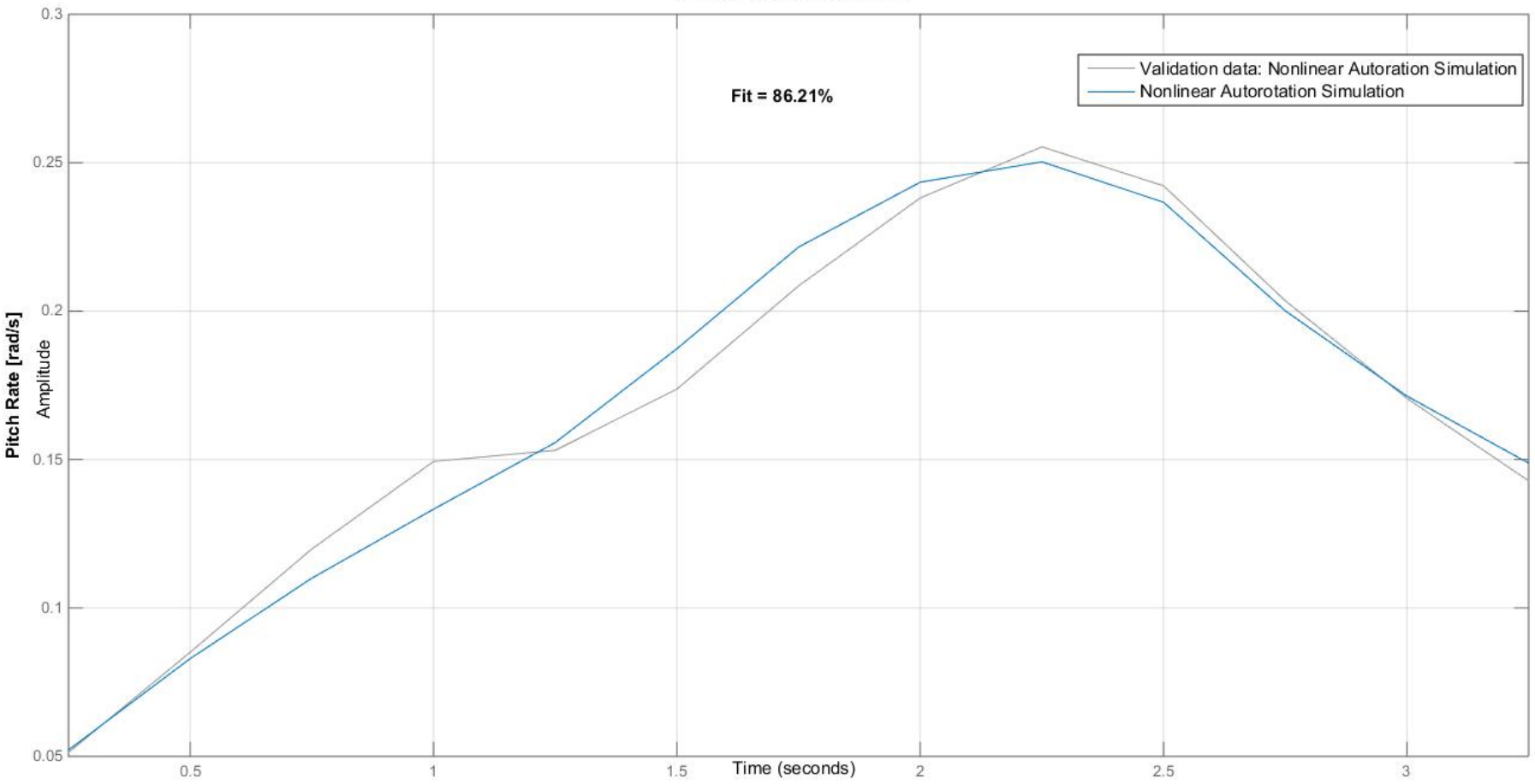


Figure 3.22: Yaw rate comparison between the validation data and identified data

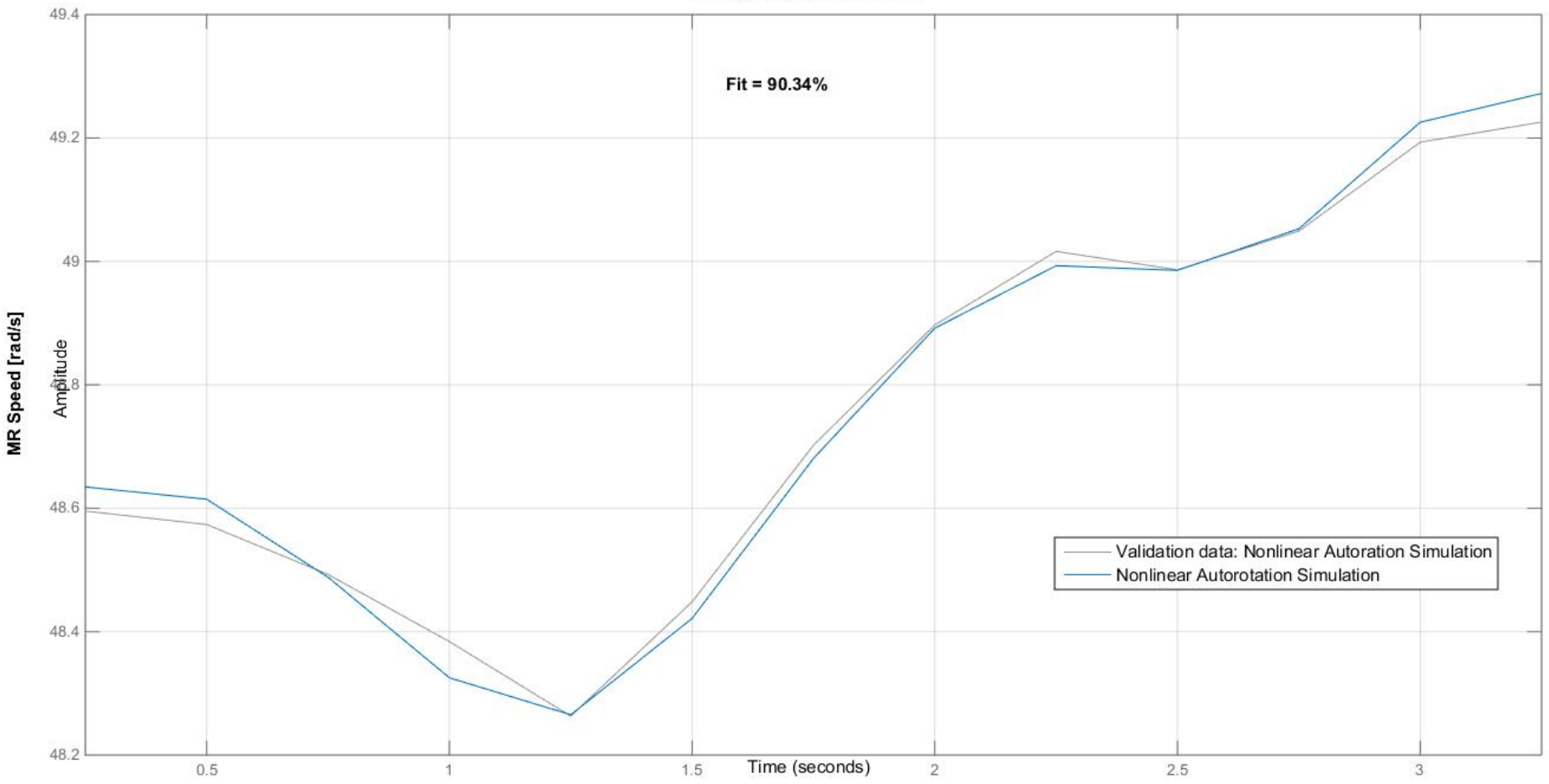


Figure 3.23: Main rotor speed comparison between the validation data and identified data

The main rotor speed vs. time, the main rotor decelerates because the main rotor collective was increased to its maximum value.

Akaike's Final Prediction Error criterion provides a measure of the model quality by simulating the situation where the model is tested on a different data set. According to the theory, the most accurate model has the smallest FPE.

Akaike's Final Prediction error is defined as follows:

$$FPE = V\left(\frac{1+\frac{d_1}{N_1}}{1-\frac{d_1}{N_1}}\right) \tag{3.2}$$

where:

V = Loss function;

d1 = Number of estimated parameters;

N1 = Number of values in estimation.

Loss function is defined as follows:

$$V = det\left(\frac{1}{N_1}\sum_{1}^{N_1}\epsilon(t,\theta'_{N_1})(\epsilon(t,\theta'_{N_1}))^T\right) \tag{3.3}$$

where:

θ'_{N_1} = estimated parameters.

The Akaike's FPE was 5.44 x 10^{-40} and loss function was 4.71 x 10^{-40}.

Similar figures were plotted for the entry into autorotation and glide. The identified derivatives were used in designing the controller and modelling the helicopter dynamics.

3.6 Validation of the RW UAV Mathematical Model

The validity of the RW UAV mathematical model with the identified stability and control derivatives has to be authenticated to ensure that the RW UAV flight dynamics have been captured.

3.6.1 Validity of the model

When the stability and control derivatives were identified, the flight simulation data was divided into the 3 phases of flight. These phases of flight are entry into autorotation, glide and flare and land. From each phase of flight a sample of data was used to identify the derivatives or unknown parameters. In order to check the validity of the mathematical model, the unused data, being the control inputs were utilized together with the identified derivatives to obtain the flight dynamics of RW UAV or output. A simulation was run to observe the response and to compare the response with the autorotation flight simulation. The response obtained was not good as the parameter estimation results. A process was undertaken to tune the identified derivatives to obtain an acceptable flight response.

Tuned Identified Parameters

Table 3.1 shows the values of the parameters that were tuned from the parameter identification process. From 0s to 34.75s the RW UAV enters autorotation and stabilises into the glide phase. From 34.75s to 40.5 the RW UAV flares and lands as shown in Figure 3.24.

Table 3.1: Identified parameters for the entire autorotation maneouvre

Parameters	Entry and glide time intervals [seconds]					Flare and Land time interval [seconds]
	0-3.8s	3.8 - 7.8s	7.8s - 11.8s	11.8s - 20s	20s - 34.7s	34.7s - 40s
C_u	0.02393	0.04262	0.04262	0.04425	0.04425	0.03736
C_v	-4.086	-0.1465	-0.1464	-3.147	-3.147	-0.1413
C_{w1}	-113.4	-36.57	-36.57	4.410	4.410	33.24
C_{w2}	-4.476	-1.588	-1.588	-1.126	-1.126	-2.387
C_{w3}	10.25	-1.413	-1.413	-0.2456	-0.2456	-2.783
C_{w4}	0.6722	2.193	2.193	-0.1111	-0.1111	-0.2819
C_{p1}	-0.01511	0.7518	0.7518	-0.1375	-0.1375	-0.2220
C_{p2}	-0.1585	-7.051	-7.051	0.2477	0.2477	-1.186
C_{p3}	-0.01002	0.6154	0.6154	-0.1061	-0.1061	-0.1889
C_{q1}	1.1570	0.1952	0.1952	0.08159	0.08159	0.9467
C_{q2}	0.96156	0.07500	0.1500	0.1111	0.1111	-1.127
C_{q3}	-0.8578	-0.1277	-0.1276	-0.05251	-0.05251	-0.5660
C_{r1}	-1.662	4.977	4.977	4.977	4.977	-0.9966
C_{r2}	-4.69355	-0.9289	-0.9289	-1.300	-1.300	0.03422
C_{r3}	0.1784	-0.9993	-0.9994	-0.9994	-0.9994	0.1541
$C_{\Omega 1}$	157.8	-15.45	-15.45	2.067	2.067	-33.59
$C_{\Omega 2}$	-2.446	-0.3778	-0.3778	-0.3099	-0.3099	-0.000241
$C_{\Omega 3}$	-68.39	8.301	8.301	-80.91	-80.91	12.42
$C_{\Omega 4}$	6.027	1.026	1.026	1.298	1.298	2.623
$C_{\Omega 5}$	-1.567	0.9777	0.9777	0.3563	0.3563	0.4168
$C_{\Omega 6}$	-5.153	693.3	693.3	1463	1463	792.3

Table 3.2: Original identified parameters for the entire autorotation maneouvre

Parameters	Entry and glide time intervals [seconds]					Flare and Land time interval [seconds]
	0-3.8s	3.8 - 7.8s	7.8s - 11.8s	11.8s - 20s	20s - 34.7s	34.7s - 40s
C_u	0.02393	-0.01066	0.049166	0.02	0.05	0.05
C_v	0.3947	-0.1465	-0.7866	-0.15	-0.79	-0.79
C_{w1}	-72.03	-36.57	4.411	9.00	4.410	4.410
C_{w2}	-5.539	-1.588	-1.126	-1.32	-1.13	-1.13
C_{w3}	-2.634	-1.413	-0.2456	-0.15	-0.25	-0.25
C_{w4}	4.435	2.193	-0.1112	-0.27	-0.1111	-0.1111
C_{p1}	-0.5116	-0.1482	-0.1375	-0.50	-0.14	-0.14
C_{p2}	0.7679	-7.051	0.9907	1.58	0.99	0.99
C_{p3}	0.4228	0.6154	-0.1061	-0.3490	-0.1061	-0.1061
C_{q1}	0.0167	0.1952	0.0816	-0.0935	0.0816	0.0816
C_{q2}	1.098	-0.375	0.2222	-2.535	0.2222	0.2222
C_{q3}	-0.0092	-0.1277	-0.05251	-0.06877	-0.05251	-0.05251
C_{r1}	-1.662	4.977	6.792	-4.861	4.977	4.977
C_{r2}	-469.4	-0.9289	1219	166.3	-0.9299	-0.9299
C_{r3}	0.1784	-0.9993	-1.330	-0.9279	-0.9994	-0.9994
$C_{\Omega 1}$	-52.59	-15.45	2.068	17.05	2.068	2.068
$C_{\Omega 2}$	1.469	-0.3778	-0.3099	-0.6403	-0.3099	-0.3099
$C_{\Omega 3}$	-117.1	8.301	-80.91	-79.14	-80.91	-80.91
$C_{\Omega 4}$	-2.606	1.026	1.298	1.773	1.298	1.298
$C_{\Omega 5}$	0.7090	0.9777	0.3563	0.3196	0.3563	0.3563
$C_{\Omega 6}$	-2394	693.3	1463	-6.392	1463	1463

Table 3.2 shows the original identified derivatives and Table 3.1 shows the tuned derivatives. The particular derivatives that were tuned are from time interval 7.8 seconds to 34.75 seconds. From 0 to 34.75 seconds the RW UAV enters autorotation and stabilises into the glide phase. From 34.75 seconds to 40 seconds the RW UAV flares and lands as shown in Figure 3.24. It was expected the derivatives from time interval 3.8 – 7.8 seconds and 7.8 – 11.8 seconds to be the same or similar because in this phase of flight the RW UAV shows unstable behaviour, refer to Table 3.2. Therefore, the original derivatives that were identified for time interval 3.8 – 7.8 seconds were utilized for time interval 7.8 -11.8 seconds as it showed a better fit from system identification. It should be noted that MATLAB System Identification Toolbox, will use the data provided and the model structure to obtain a good fit but it does not necessarily take into account the physical meaning.

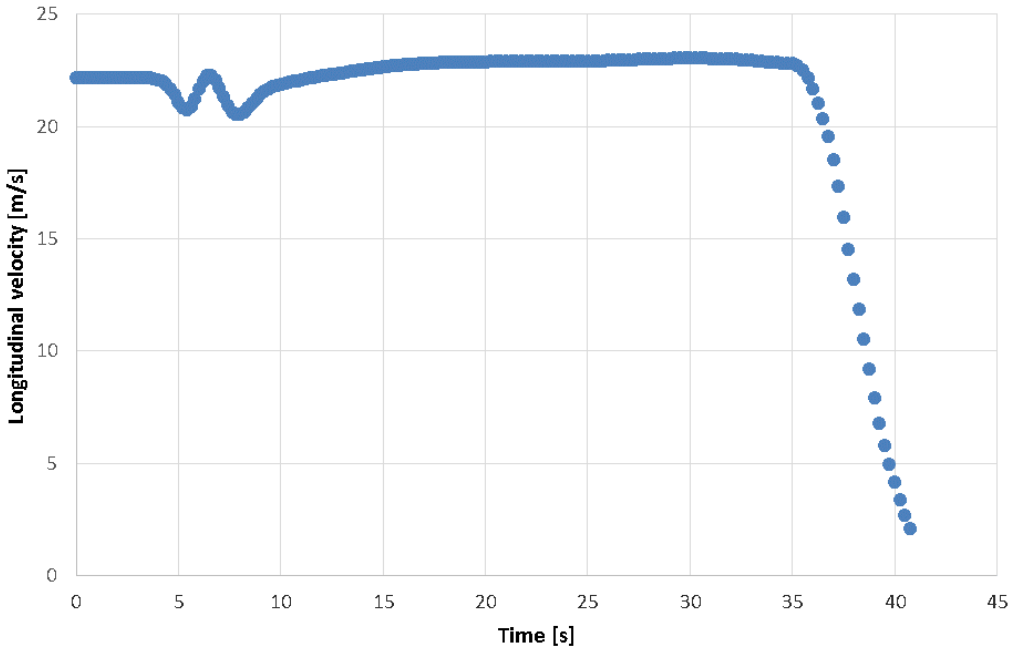


Figure 3.24: Longitudinal velocity vs time for all phases

The figures that follow were obtained using the tuned derivatives.

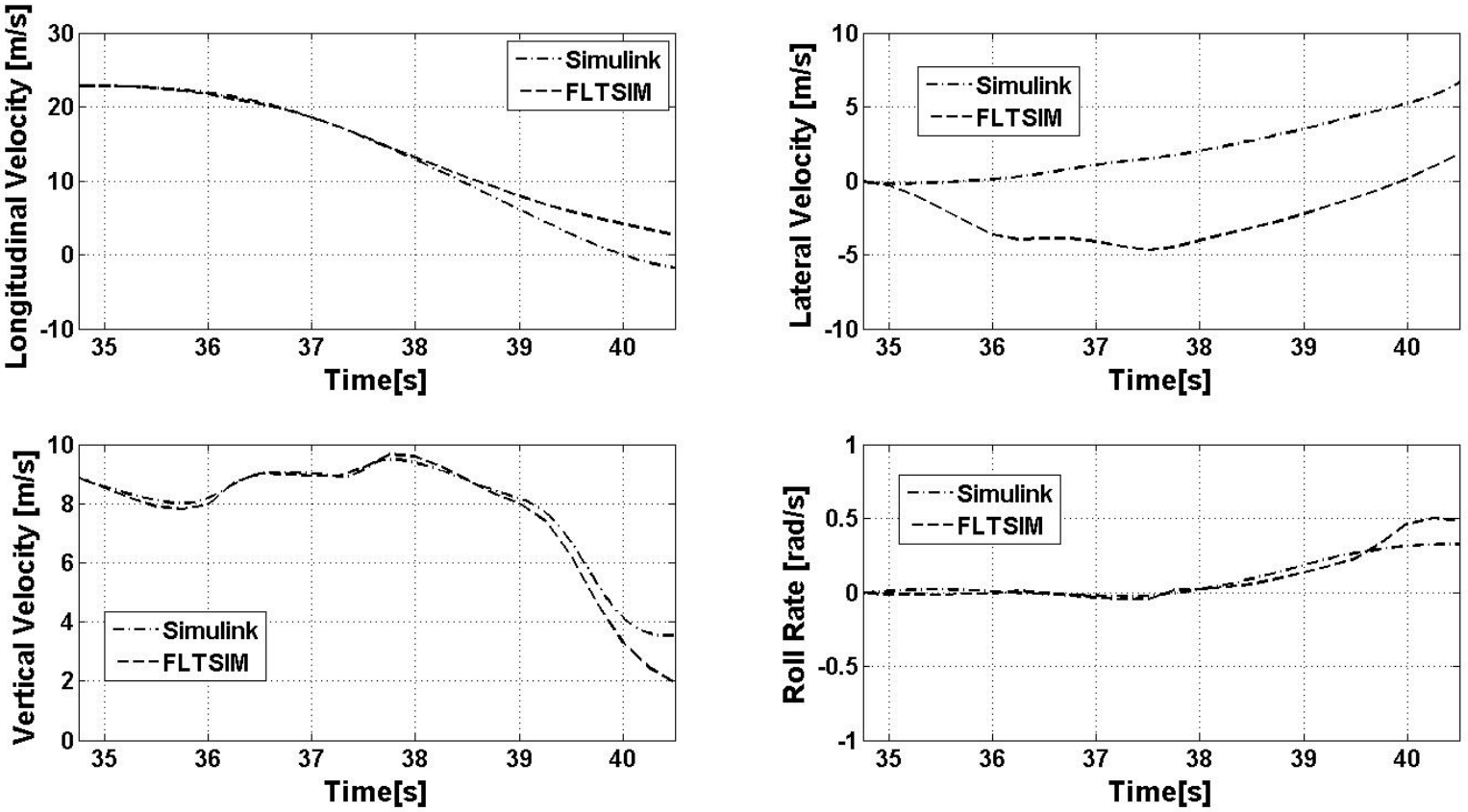


Figure 3.25: Validation of model using tuned derivatives

Referring to Figure 3.25, this is the flare and land phase. The data of the first 3 seconds of this phase of flight was utilized in identifying derivatives for this phase of flight and the rest of the data, the last 2 seconds or post 38 seconds, was used to distinguish the validity of the mathematical model. The longitudinal velocity, vertical velocity and the roll rate clearly illustrate that the mathematical model has a similar response as the flight simulation. There is a small discrepancy which is acceptable and expected as the derivatives may not necessarily capture the flight dynamics fully. The lateral velocity is of concern as the discrepancy is bigger in comparison to the flight parameters that have been mentioned. This could be an indication that the lateral velocity derivative has not captured the dynamics fully.

The main rotor speed dynamics, refer to Figure 3.26, have been captured by the model. The discrepancy is very small. The pitch rate also shows a good fit and the response is satisfactory. There is a divergence in yaw rate post 39 seconds. This divergence may be one of the reasons causing the error in the lateral velocity. From the responses obtained, the mathematical model is deemed valid but attention should be paid to the lateral velocity and yaw rate.

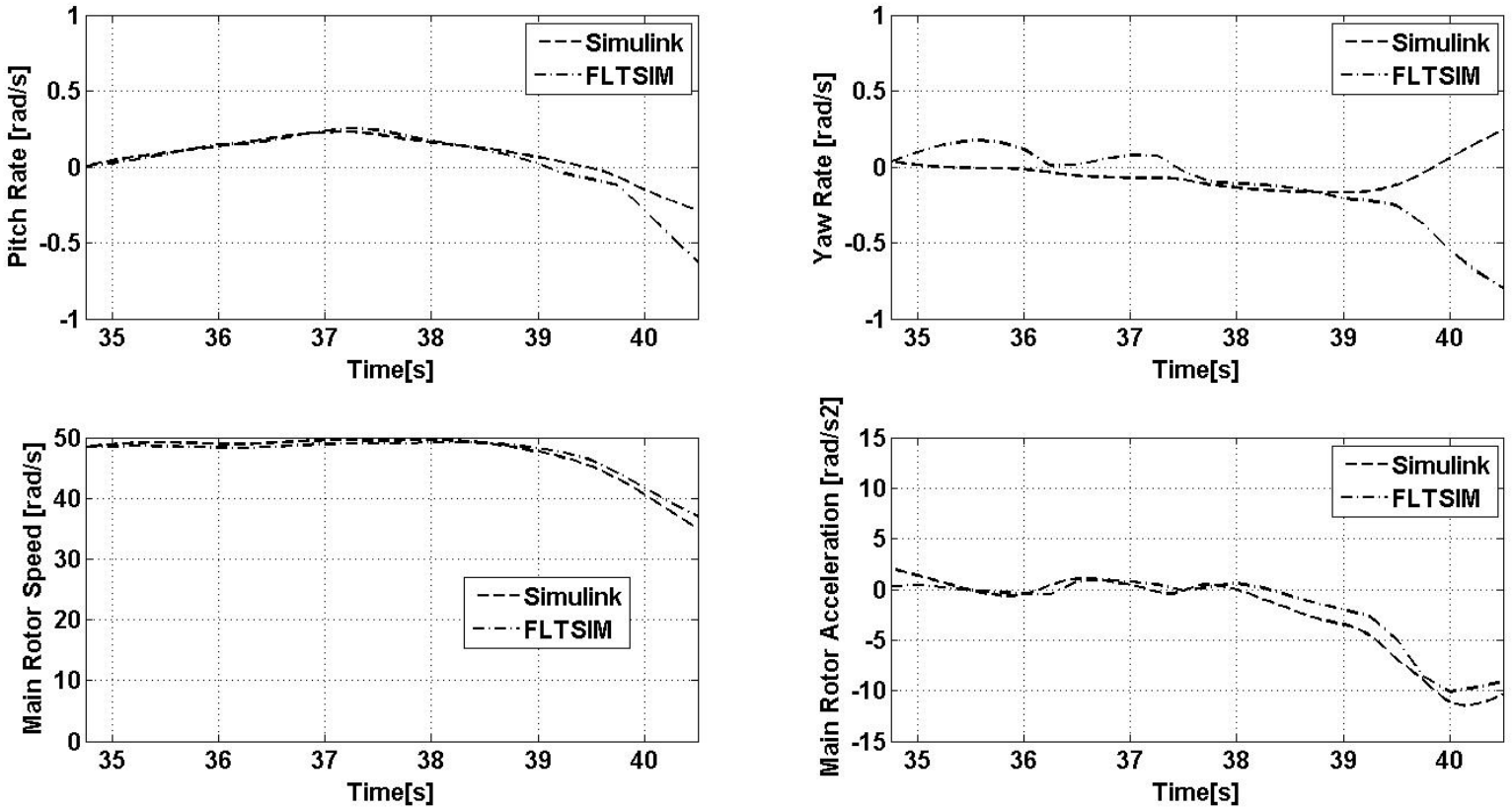


Figure 3.26: Validation of model using tuned derivatives

4 Design of the Differential Dynamic Programming Controller

4.1 Introduction

Differential Dynamic Programming (DDP) is a technique based on dynamic programming for determining the optimal control function of a nonlinear system. Optimal control defines the choice of actions that minimize future costs under the constraint of state-space dynamics, [Theodorou et al., 2010]. The DDP method uses successive approximations and expansions in differentials or increments to obtain a solution of optimal control problems. It is primarily used in deterministic problems in discrete time although there are many variations.

The principal of optimality states that an optimal set of decision rules have the property that regardless of the ith decision, the remaining decisions must be optimal with respect to the outcome that results from the ith decision. As a result, the optimal immediate decision depends only on the current state.

DDP has numerous advantages. The process entails breaking down a complex problem into a series of interrelated sub-problems and this provides insight into the nature of the problem. The computational procedure allows for a built-in form of sensitivity analysis based on state variables and on variables represented by stages. It is applicable to linear or nonlinear problems, discrete or continuous variables, deterministic or stochastic problems [Natarajan and Balasubramani, 2006]. The major disadvantage of DDP is dimensionality. This is prevalent in a problem characterized by multiple states.

The traditional dynamic programming can usually deal with two or three at most state variables, Discrete Differential Dynamic Programming (DDDP) can handle up to eight state variables [Chow, 1971].

4.2 The DDDP Technique

The DDDP procedure is an iterative method in which the recursive equation of dynamic programming is used to search for an improved trajectory among the discrete states in the neighbourhood of the trail trajectory [Chow, 1971].

Consider the dynamic system with state equation, [Chow, 1971]:

$$s(n) = \phi[s(n-1), u(n-1), n-1], n = 1, 2, ..., N \tag{4.1}$$

where:

n = the index specifying a stage;

N = total number of time increments into which the time horizon has been divided;

s(n) = m-dimensional state vector at stage n;

u(n) = q-dimensional decision vector at stage n-1;

q = the number of decision variables;

and

$$s(n)\epsilon S(n), u(n)\epsilon U(n) \tag{4.2}$$

where:

S(n) = admissible domain in the state space at stage n;

U(n) = admissible domain in the decision space at stage n.

The objective function to be minimized or maximized is:

$$F = \Sigma_{n=1}^{N} R[s(n-1), u(n-1), n-1] \tag{4.3}$$

where:

F = objective function.

The forward algorithm of dynamic programming may be used to optimize the objective function over n stages as follows:

$$F^*[s(n), n] = max_{u(n-1)\epsilon U(n-1)} R[s(n-1), u(n-1), n-1] + F^*[s(n-1), n-1] \tag{4.4}$$

where $F^*[s(n), n]$ is the maximum (or minimum) total from stage 0 to stage n when the state n is $s(n)$. Solving Eq. (4.1) for s(n-1):

$$s(n-1) = \theta[s(n), u(n-1), n-1] \tag{4.5}$$

Substituting Eq. (4.4) into Eq. (4.3), the recursive equation becomes:

$$F^*[s(n), n] = min_{u(n-1)\epsilon U(n-1)} R[\theta, u(n-1), n-1] + F^*[s(n-1), n-1] \tag{4.6}$$

The solution of Eq. (4.6) for a specific state provides an optimum decision that should be made for some stage to bring the system to the specific state at stage n.

Assuming the objective function for the system is to be optimized subject to the admissible domain in the state space and the m-dimensional state vectors at the initial and final stages are specified as follows:

$$s(0) = a(0); s(N) = a(N) \tag{4.7}$$

In the DDDP approach a trial sequence of admissible decision vectors, u'(n), n = 1, ..., N-1, is called the trial policy and it is assumed it satisfies Eq. (4.2). The state vectors at the different stages can be determined. The state vector satisfying Eq. (4.2) and (4.7) is called the trial trajectory, s'(n), n = 0,1, ..., N.

Substituting the trial policy, u'(n) and trial trajectory, s'(n) into Eq (4.3):

$$F\prime = \Sigma_{n=1}^{N} R[s^{'}(n-1), u\prime(n-1), n-1] \tag{4.8}$$

F' may not be the optimum objective function.

Consider a set of incremental m-dimensional vectors:

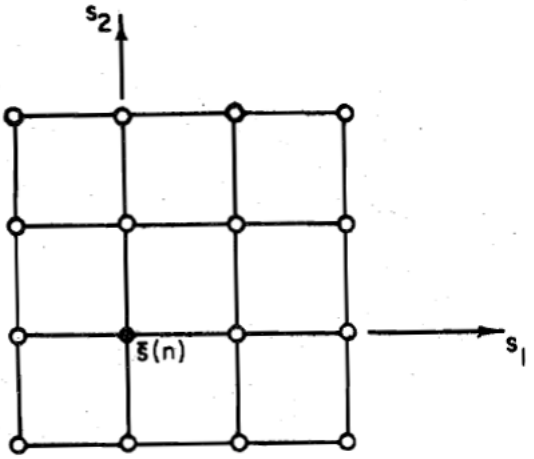
$$\Delta s_i(n) = \begin{pmatrix} \delta s_{i1}(n) \\ \delta s_{i2}(n) \\ \vdots \\ \delta s_{ij}(n) \\ \vdots \\ \delta s_{im}(n) \end{pmatrix}$$

with n = 0, 1, ..., N and i = 1, 2, ..., T^m and with the j-th component $\delta s_{ij}(n)$, can take any one value σ_t, t = 1, 2, ..., T, from a set of assumed incremental values of the state domain. The assumed value of σ_t is the t-th assumed increment from the state domain and T is the total number of assumed increments the state domain. When the incremental values are added to the trial trajectory at a stage, these vectors form an m-dimensional subdomain designated by D(n),

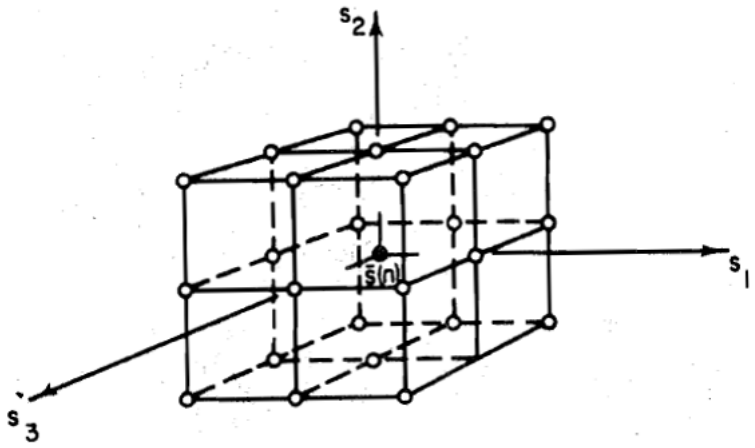
$$s^{'}(n) + \Delta s_i(n); i = 1, 2, ..., T^m \tag{4.9}$$

One of the values of t must be zero as the trial trajectory is always in the subdomain.

Figure 4.1 illustrates two subdomains for m = 2, T = 4 and m = 3, T = 3. D(n) , n = 0, 1, ..., N, is called a 'corridor' and designated by C as shown in Figure 4.2 by the space between two solid lines for a system with m = 1, T = 3, and n = 10.



A state sub-domain D(n) defined by 16 lattice points in the neighborhood of $\bar{s}(n)$ for a 2-dimensional state vector and $T = 4(\sigma_{j,1} = +2.0, \sigma_{j,2} = +1.0, \sigma_{j,3} = 0, \sigma_{j,4} = -1.0$ for $j = 1,2)$



A state sub-domain D(n) defined by 27 lattice points in the neighborhood of $\bar{s}(n)$ for a 3-dimensional state vector and $T = 3(\sigma_{j,1} = +1.0, \sigma_{j,2} = 0, \sigma_{j,3} = -1.0$ for $j = 1,2,3)$

Figure 4.1: Subdomains for m = 2, T = 4 and m = 3, T = 3 (Chow [1971], Figure 1)

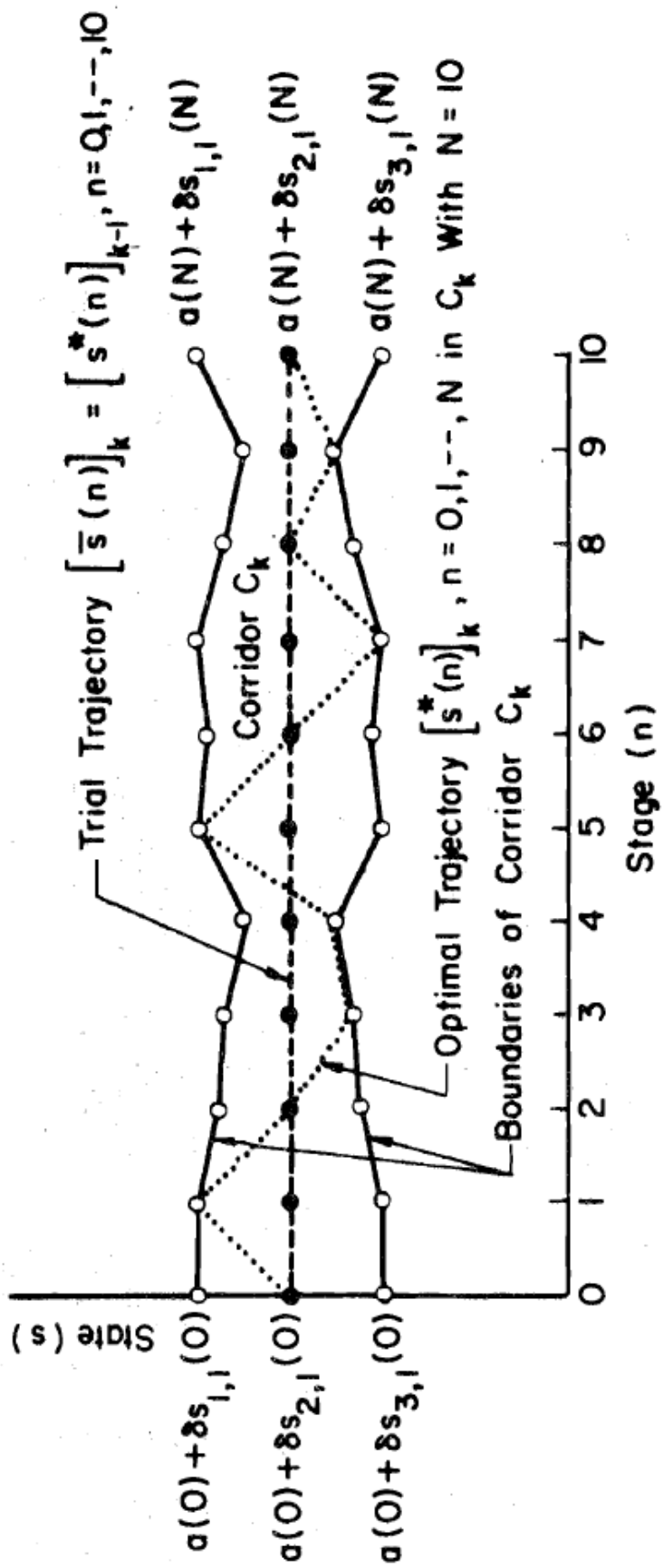


Figure 4.2: Trial trajectory, boundaries and defining corridor C_k (Chow [1971], Figure 2)

4.3 The Procedure

The corridor C is used as a set of admissible states and the optimization constrained to these states is performed by means of the recursive equation. The value of F obtained is at least equal to or less than F' in Eq. (4.7). If F is less than F', the corresponding trajectory and policy obtained from corridor C are used in the next iteration step as the trial trajectory and trial policy. The k-th iteration step that follows is:

1. The k-1)th iteration step results $[s^*(n)]_{k-1}$ and $[u^*(n)]_{k-1}$ can be used as the trial trajectory and policy for the k-th iteration step:
   $[s^{'}(n)]_k = [s^*(n)]_{k-1}$; $[u^{'}(n)]_k = [u^*(n)]_{k-1}$
2. The increment values $[\sigma_1]_k, [\sigma_2]_k, ..., [\sigma_T]_k$ should be selected to define the k-th corridor Ck and use equation 4.6 to minimize F subject to s(n) ϵ C_k.
3. Trace the optimum trajectory, $[s^*(n)]_{k-1}$ and corresponding $u_*(n)]_{k-1}$ within the corridor satisfying the boundary conditions
4. Determine F_k^*; if $F_k^* - F_{k-1}^* \leq \varepsilon$ where ε is a specified constant and the iteration stops when the criteria is met or step 1 is repeated.

The corridor size may be varied gradually by selecting different $[\sigma_t]_k, t = 1, 2, ..., T$, in step 2. If the corridor size is kept constant during the iterations and no improvement can be achieved after the k-th iteration, it is recommended that $[\sigma_t]_k, t = 1, 2, ..., T$ be reduced for the (k+1)th iteration. The process should be continued with the new corridor size until another iteration that behaves like the k-th iteration is reached. The corridor size is further reduced starting at the next iteration and the procedure is repeated until the condition in step 4 is met. It should be noted that possible to assume a different set of σ_t increments for each state variable.

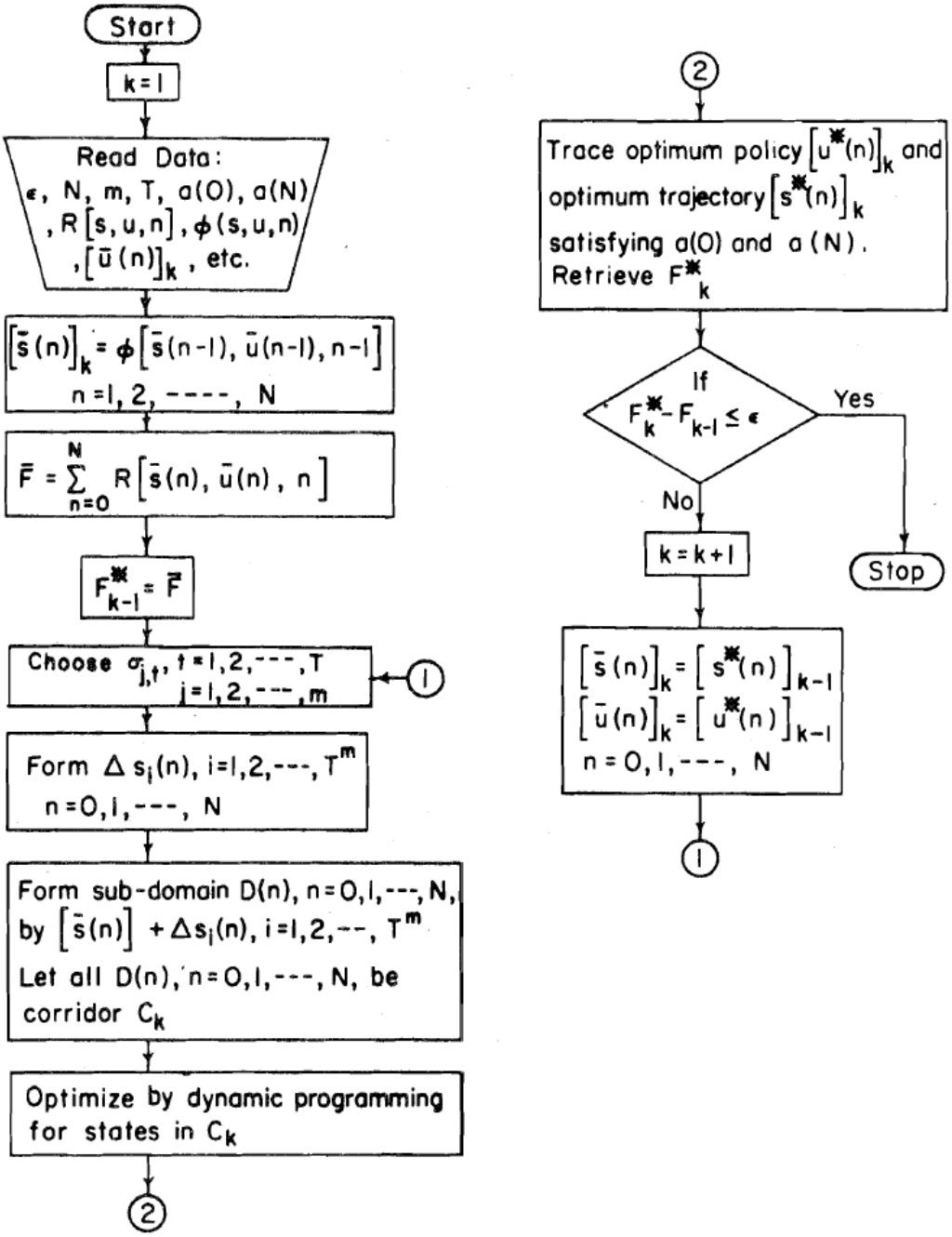


Figure 4.3: Flowchart showing steps of the DDDP approach (Chow [1971], Figure 3)

The above procedure was applied in designing the DDDP controller. The FLTSIM software data was used as the ideal trajectory and further more to determine the value of the objective function. To illustrate the controller is robust, the trial trajectory of the states (u, v, w, p, q, r,Ω) was generated randomly and each time an optimum trajectory was determined that met the criteria of the objective function. Please refer to Appendix D for the DDDP controller code. Figure 4.4 to Figure 4.10 illustrate the optimum trajectory that was determined for the states by the DDP controller and "ideal" is the trajectory obtained through flight simulations. A strategy to continuously decrease the increment constants, σ, at certain stages of the iteration aided in meeting the objective function criteria, because the plots were a perfect match.

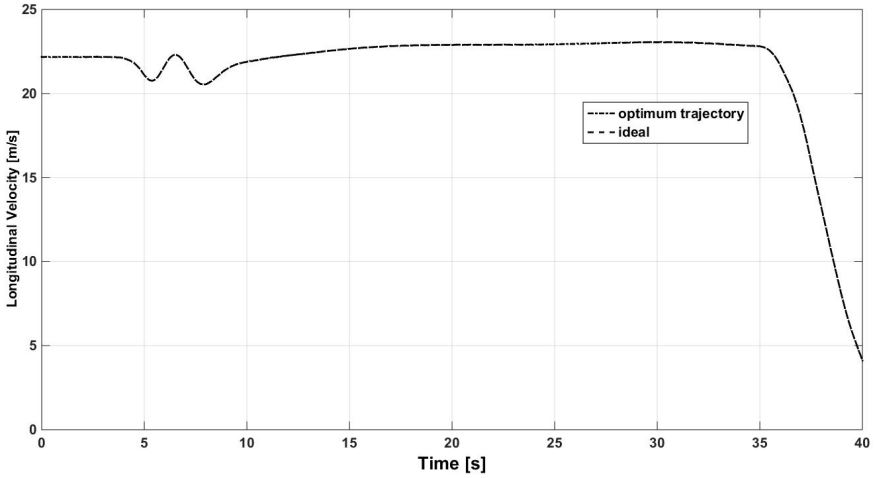


Figure 4.4: Longitudinal velocity, comparison between ideal and optimum trajectory

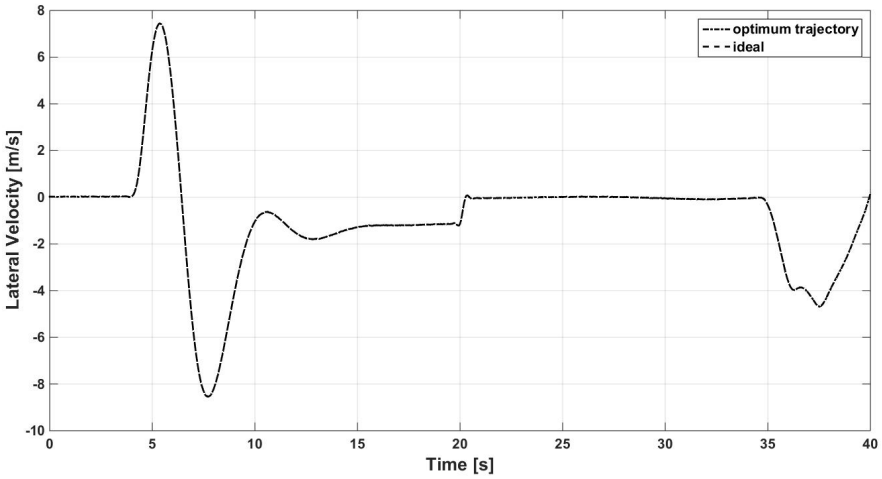


Figure 4.5: Lateral velocity, comparison between ideal and optimum

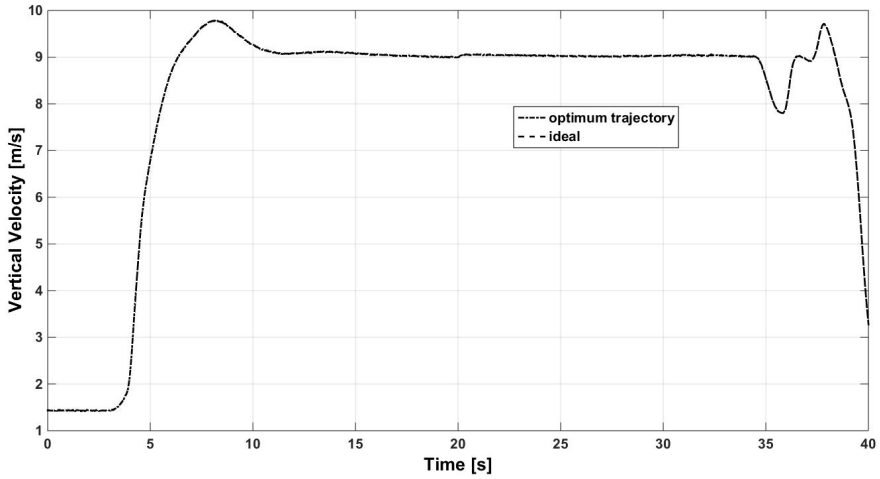


Figure 4.6: Vertical velocity, comparison between ideal and optimum trajectory

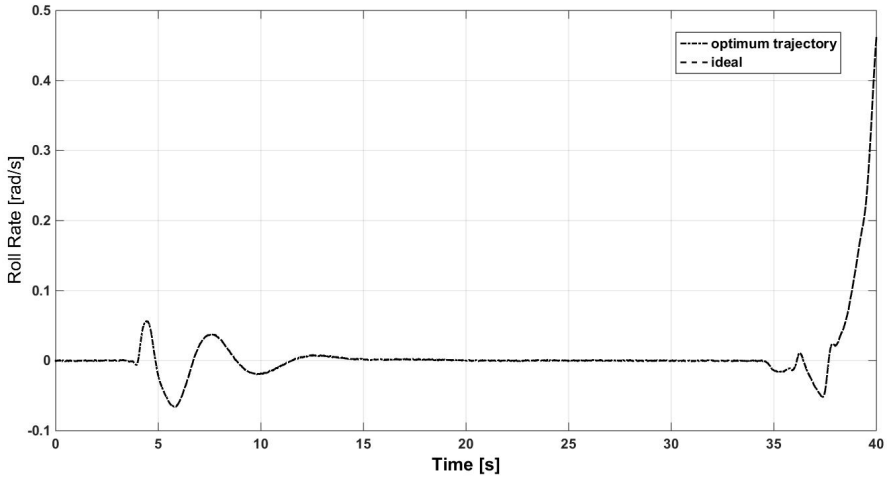


Figure 4.7: Roll rate, comparison between ideal and optimum trajectory

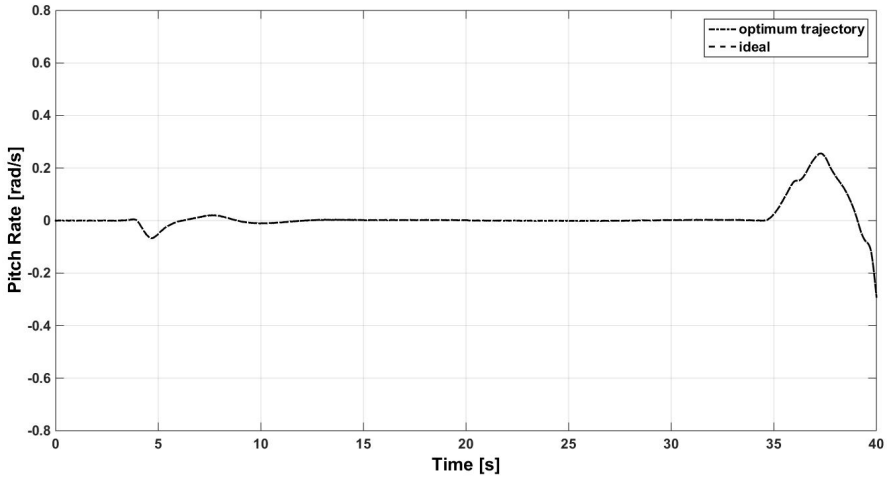


Figure 4.8: Pitch rate, comparison between ideal and optimum trajectory

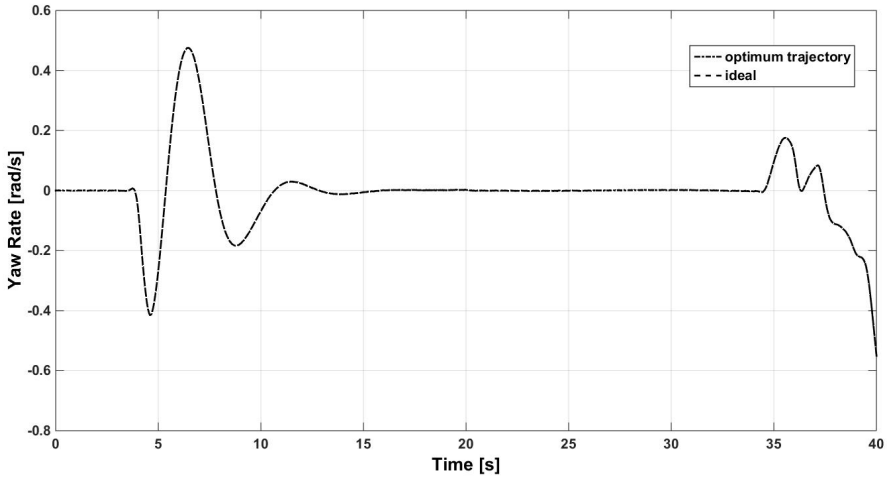


Figure 4.9: Yaw rate, comparison between ideal and optimum trajectory

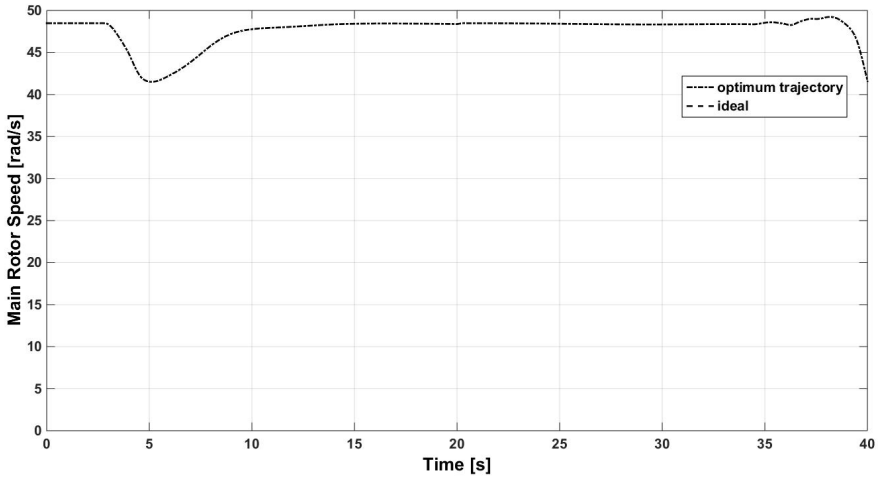


Figure 4.10: Main rotor speed: comparison between ideal and optimum trajectory

When the optimum trajectories of the states were determined, the DDP controller calculated the control inputs required as shown from Figure 4.11 to Figure 4.14. These were the predetermined inputs used to control the RW UAV. The trend of the DDP control inputs is similar to the FLTSIM control inputs. There is a spike at 4 seconds and just before 35 seconds in the main rotor collective pitch angle that was determined by the DDP controller, refer to Figure 4.11. The spike at 4 seconds could be attributed to entry into autorotation in which the flight dynamics are unstable and the controller is trying to stabilize the aircraft. This occurence is similar to the one just before 35 seconds. This is the point in which the aircraft starts to flare. It was expected that the tail rotor collective pitch angle towards touch-down to increase instead of decreasing as shown in Figure 4.14, as compared to the FLTSIM control inputs.

The control inputs were predetermined by the DDP controller before running the autorotation simulation manoeuvre because DDP has a major disadvantage of dimensionality. This was done to reduce the processing time and overall real-time simulation. By doing so, another disadvantage is introduced in that corrective actions cannot be provided to the RW UAV if it should be affected by turbulence. But turbulence was not included in the model.

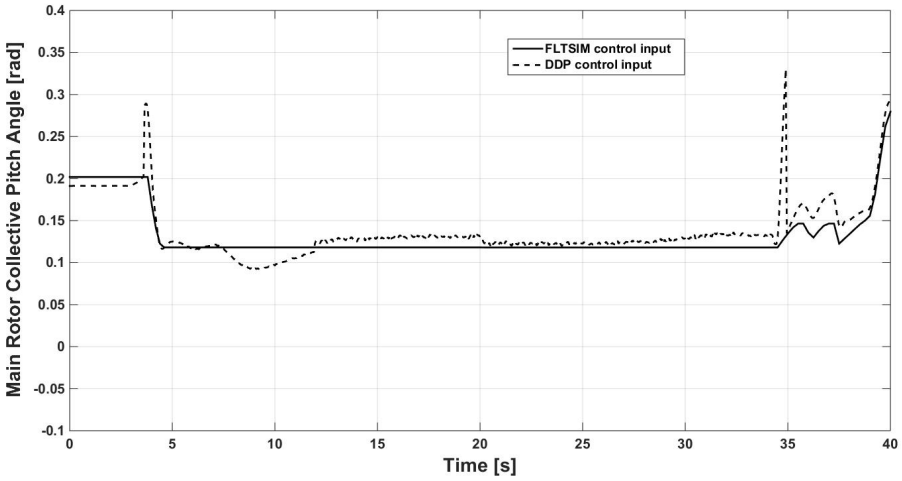


Figure 4.11: Main rotor collective pitch angle input comparison

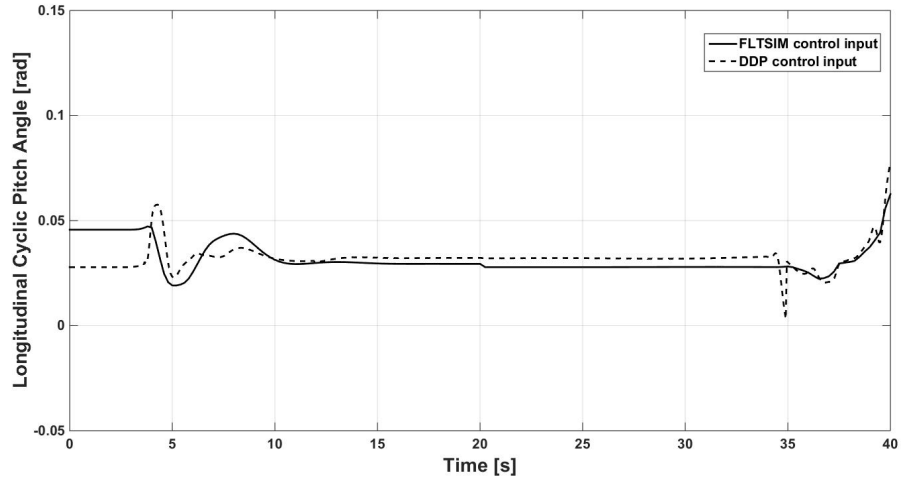


Figure 4.12: Longitudinal cyclic pitch angle input comparison

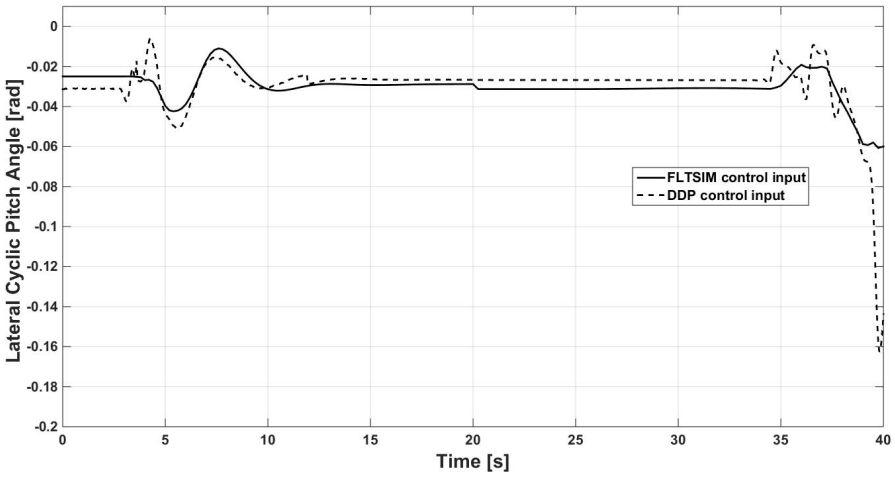


Figure 4.13: Lateral cyclic pitch angle input comparison

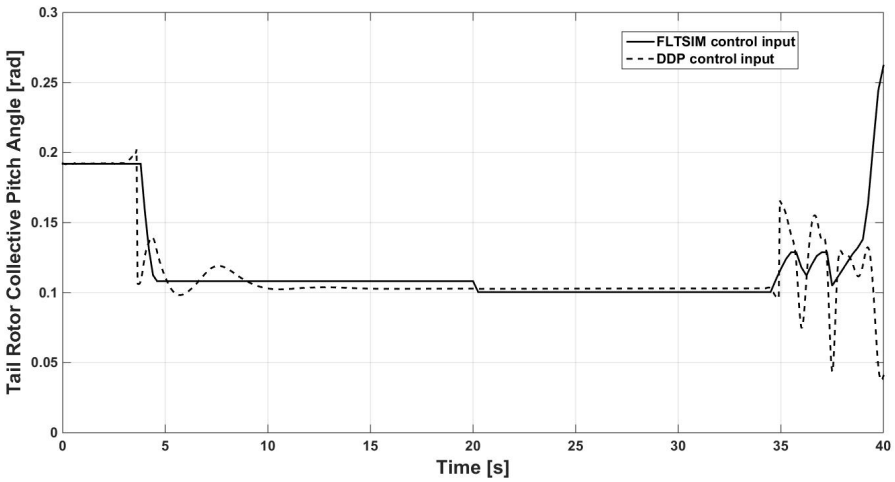


Figure 4.14: Tail rotor collective pitch angle input comparison

5 Results and Discussion

5.1 Autorotation Sea Level Manoeuvre

The figures that follow illustrate the ability of the DDP controller to safely land the RW UAV for sea level conditions. The longitudinal dynamics in autorotation are important, especially the longitudinal velocity, vertical velocity and the pitch attitude to ensure the RW UAV does not have a hard landing and the payload and RW UAV does not get damaged. The main rotor speed plays a significant role, in ensuring the RW UAV lands safely. The longitudinal dynamics will be discussed first, followed by the lateral dynamics.

5.1.1 Longitudinal direction and main rotor dynamics

Initially the RW UAV is in forward level flight and in equilibrium, as shown in Figure 5.3. Referring to the legend in Figure 5.3, "Simulink" is the results obtained from the RW UAV mathematical model and with inputs determined by the DDP controller. "FLTSIM" is the ideal trajectory, obtained from flight simulation software package. At 3 seconds, the engine fails and the engine power spools down. The longitudinal velocity is modelled well even though there is an error in the longitudinal acceleration but the error is small. There is slight diversion in the longitudinal acceleration post 3 seconds as illustrated in Figure 5.3 but this could be attributed to the change in dynamics of the RW UAV as in the system identification process the derivatives are identified to obtain a good fit for the majority of the ideal trajectory or flight simulation response data, which would be the first three seconds. The drift in the pitch attitude, refer to Figure 5.1, could be due to the stability derivative C_{q_2}, which is the rate of change of the pitch rate. Possibly, if it was smaller the error in the pitch rate and acceleration would decrease. The main rotor speed is modelled satisfactorily. At 3 seconds the main rotor acceleration diverges, however, this divergence has a small impact in the main rotor speed. Therefore this error does not have a dire effect in the main rotor speed.

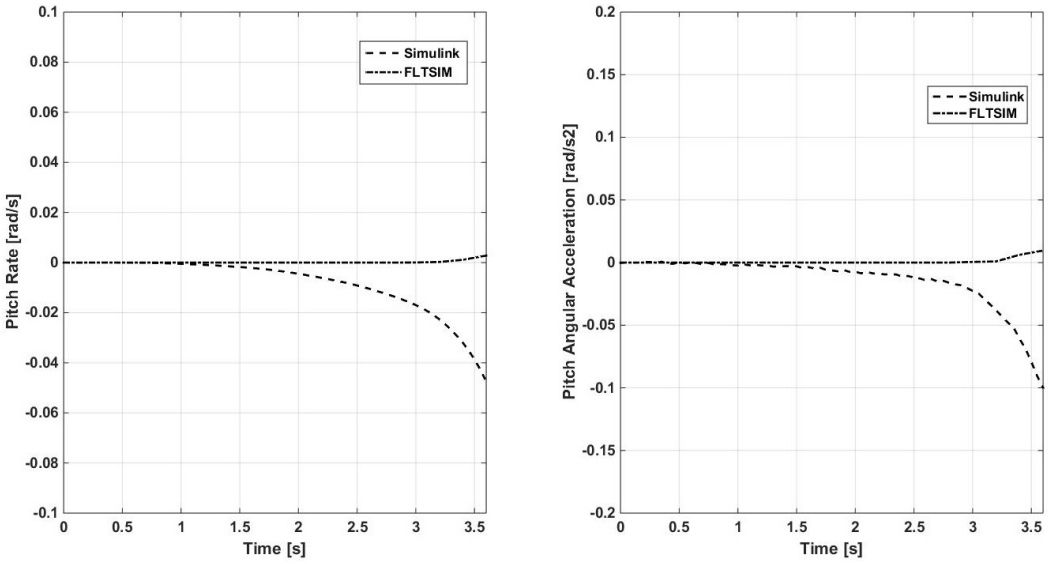


Figure 5.1: Pitch rate and acceleration from 0 s to 3.6 s

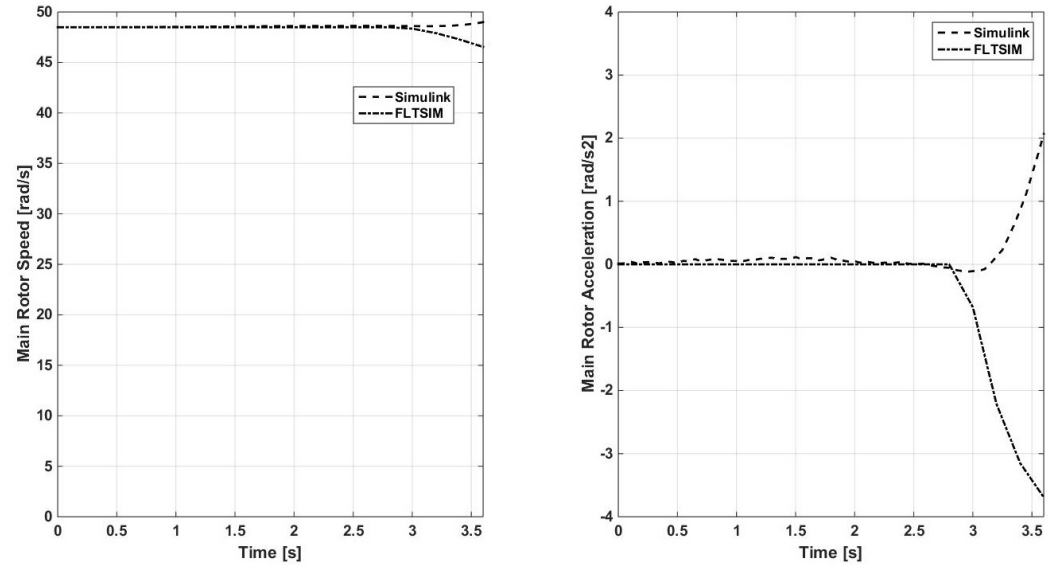


Figure 5.2: Main rotor from 0 s to 3.6 s

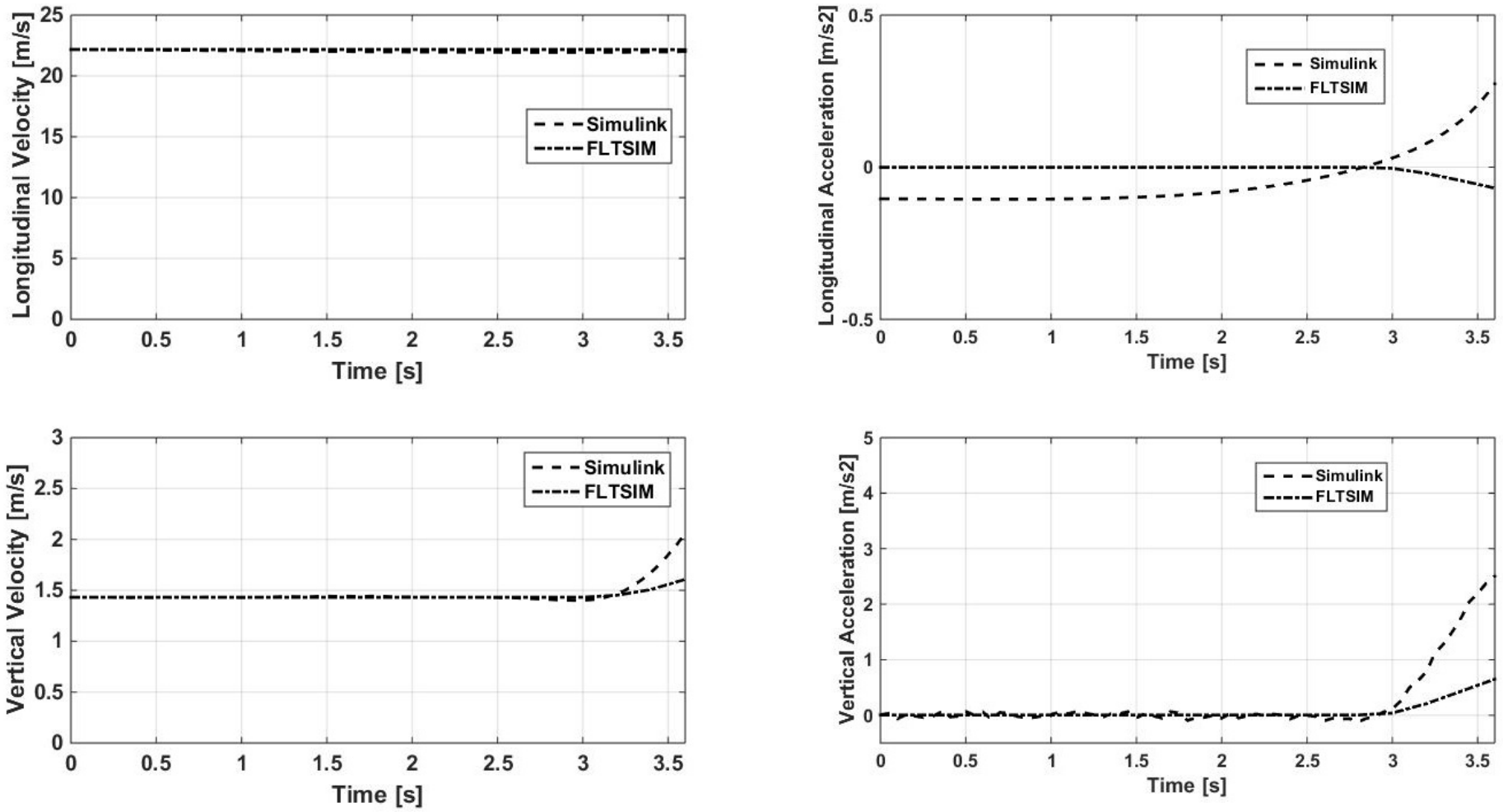


Figure 5.3: Time interval from 0 s to 3.6 s

Time interval 3.6s to 7.8s the engine power is zero and the DDP controller decreased the main rotor collective pitch angle to an optimum angle that will conserve the energy in the main rotor. In this time interval, the RW UAV is unstable as it is entering into the autorotation and the purpose of the controller at this stage is to stabilize the aircraft into the glide phase. The first two seconds of this time interval, the pitch rate is modelled adequately. In the last two seconds, the model does not reach the peak as shown in Figure 5.4, instead it seems the model does not accurately capture the pitch dynamics and could be possibly over-damped. The modelled main rotor speed follows the ideal trajectory trend. The main rotor speed decreases initially but by the controller decreasing the main rotor collective pitch angle, the main rotor speed increases. The obtained pitch rate response has a significant impact on the longitudinal acceleration, which can be seen in Figure 5.6, particularly the last two seconds. The error in the longitudinal acceleration between the model and the ideal trajectory is due to the inaccurate pitch rate. Probably by correcting the pitch rate, the longitudinal acceleration would be corrected as well. The vertical dynamics are modelled adequately.

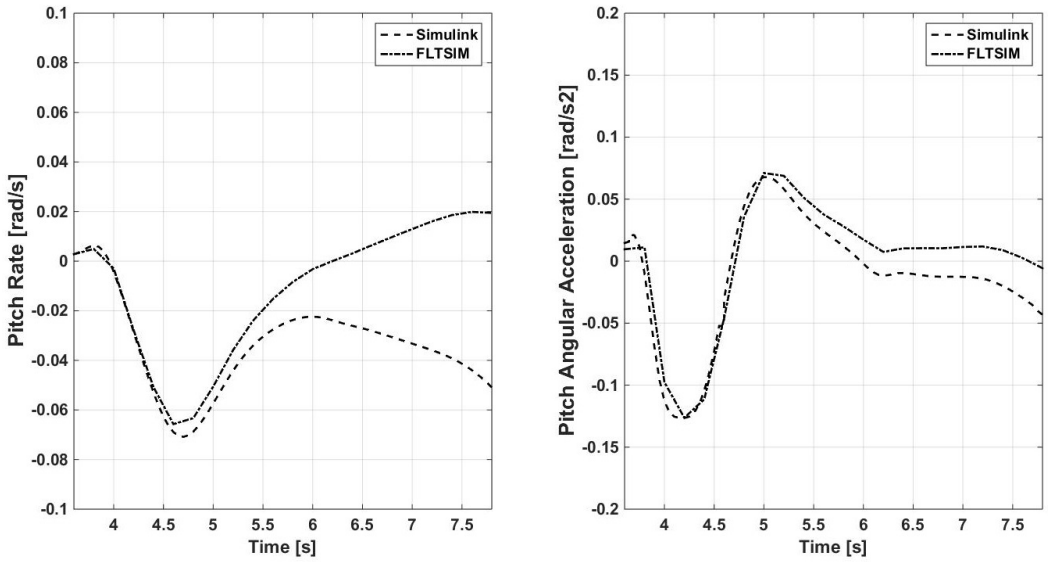


Figure 5.4: Pitch rate and acceleration from 3.6 s to 7.8 s

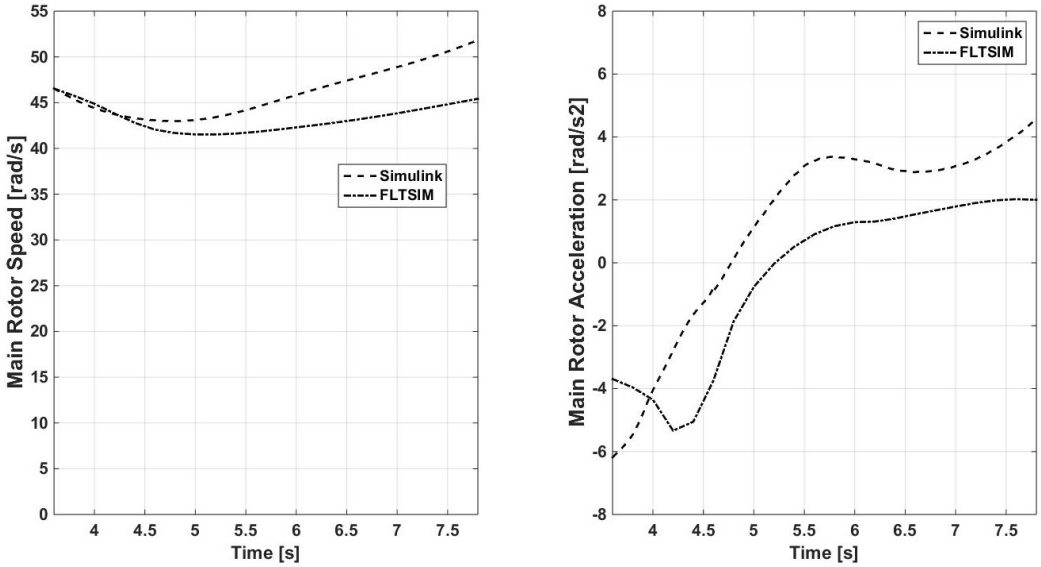


Figure 5.5: Main rotor from 3.6 s to 7.8 s

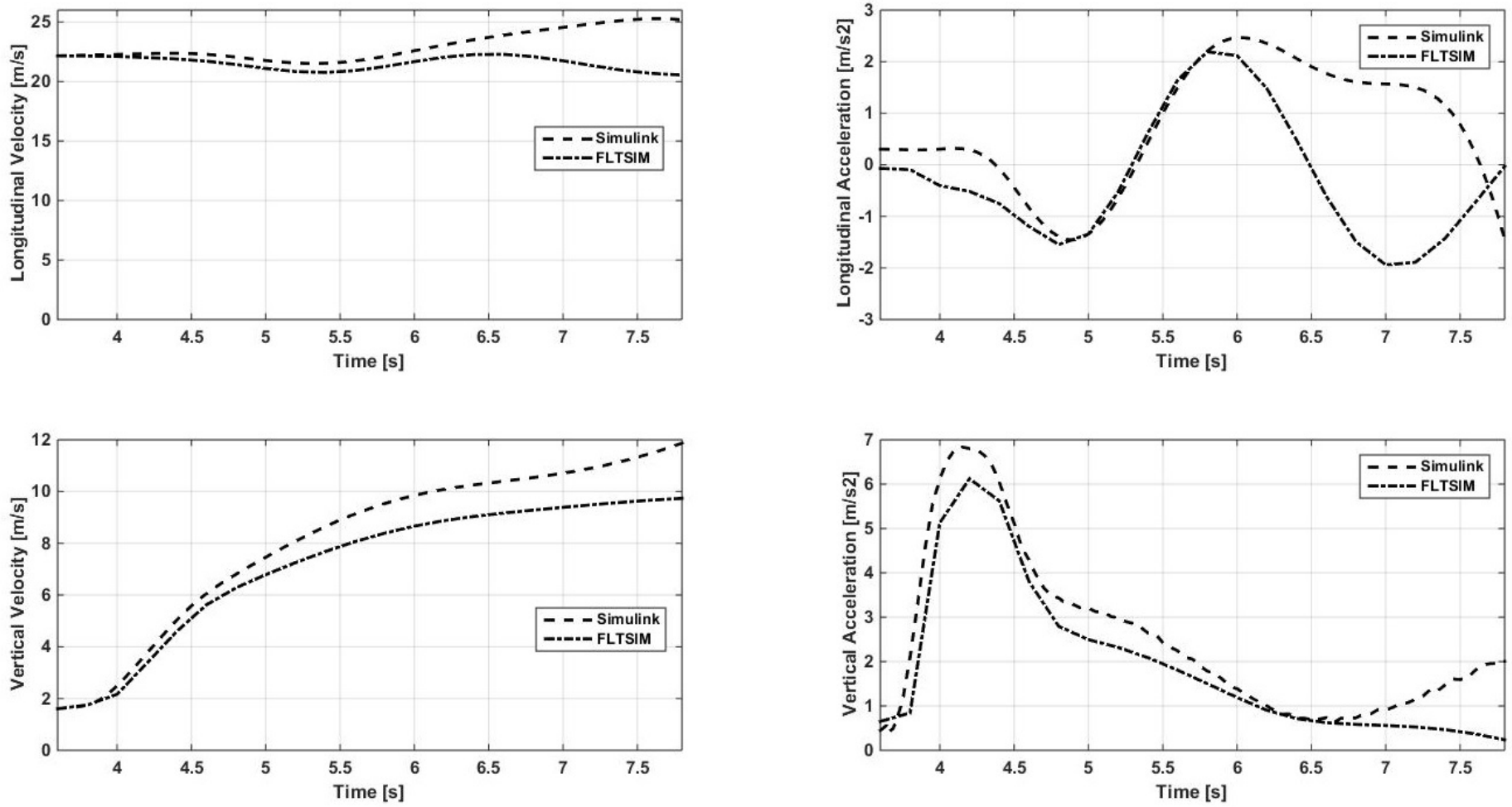


Figure 5.6: Time interval from 3.6 s to 7.8 s

The discussion that follows is with regards to Figures 5.7 to 5.9. The discrepancy between the modelled pitch rate and the ideal pitch rate is small. Therefore the modelled pitch rate and acceleration is satisfactory. This is also true for the main rotor speed. The modelled vertical acceleration is not a smooth curve because of the main rotor collective pitch angle input which shows the same behaviour (see Figure 5.11). However, the vertical speed is modelled satisfactory even though there is a slight discrepancy. The longitudinal velocity is modelled accurately.

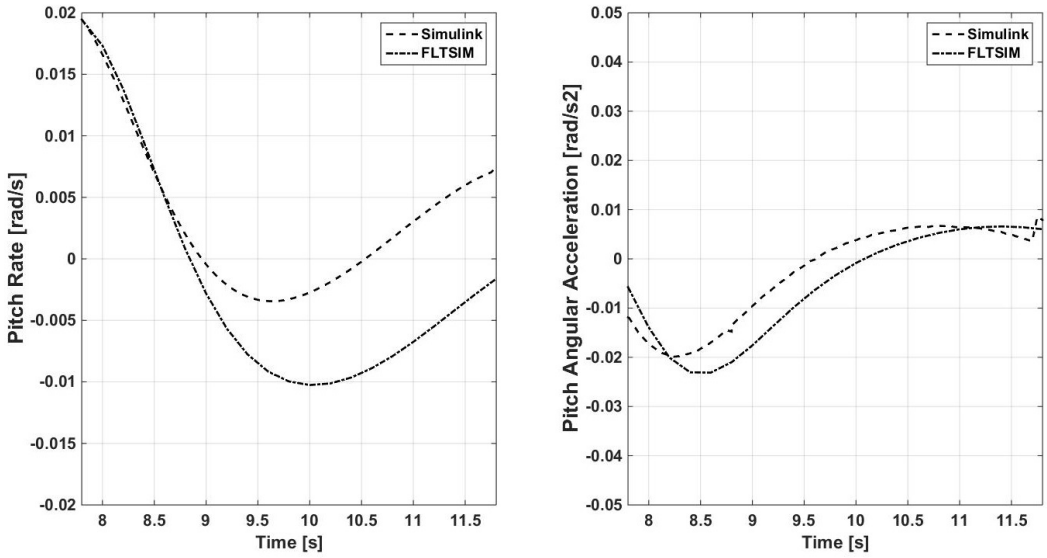


Figure 5.7: Pitch rate and acceleration from 7.8 s to 11.8 s

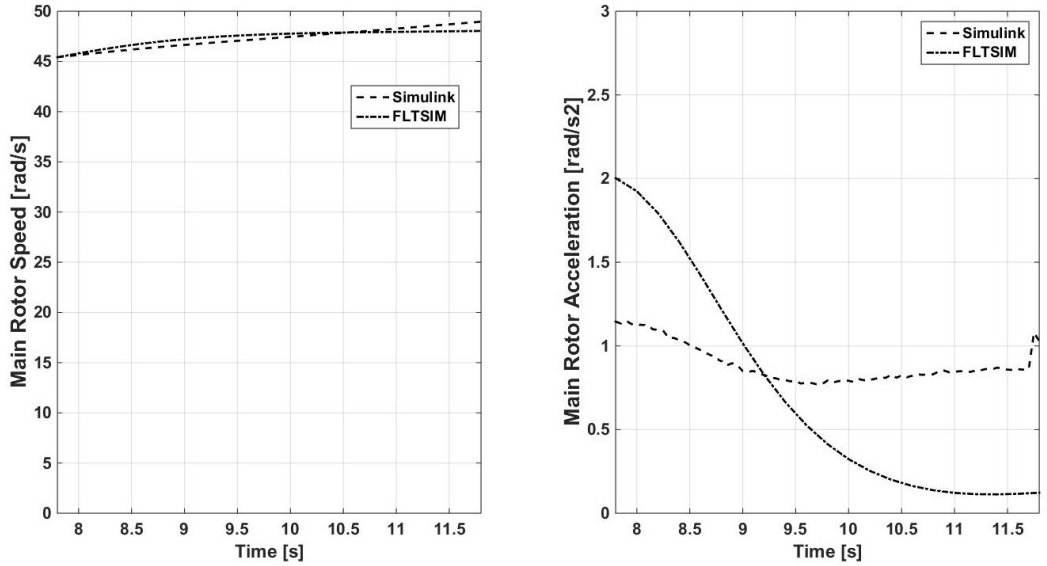


Figure 5.8: Main rotor from 7.8 s to 11.8 s

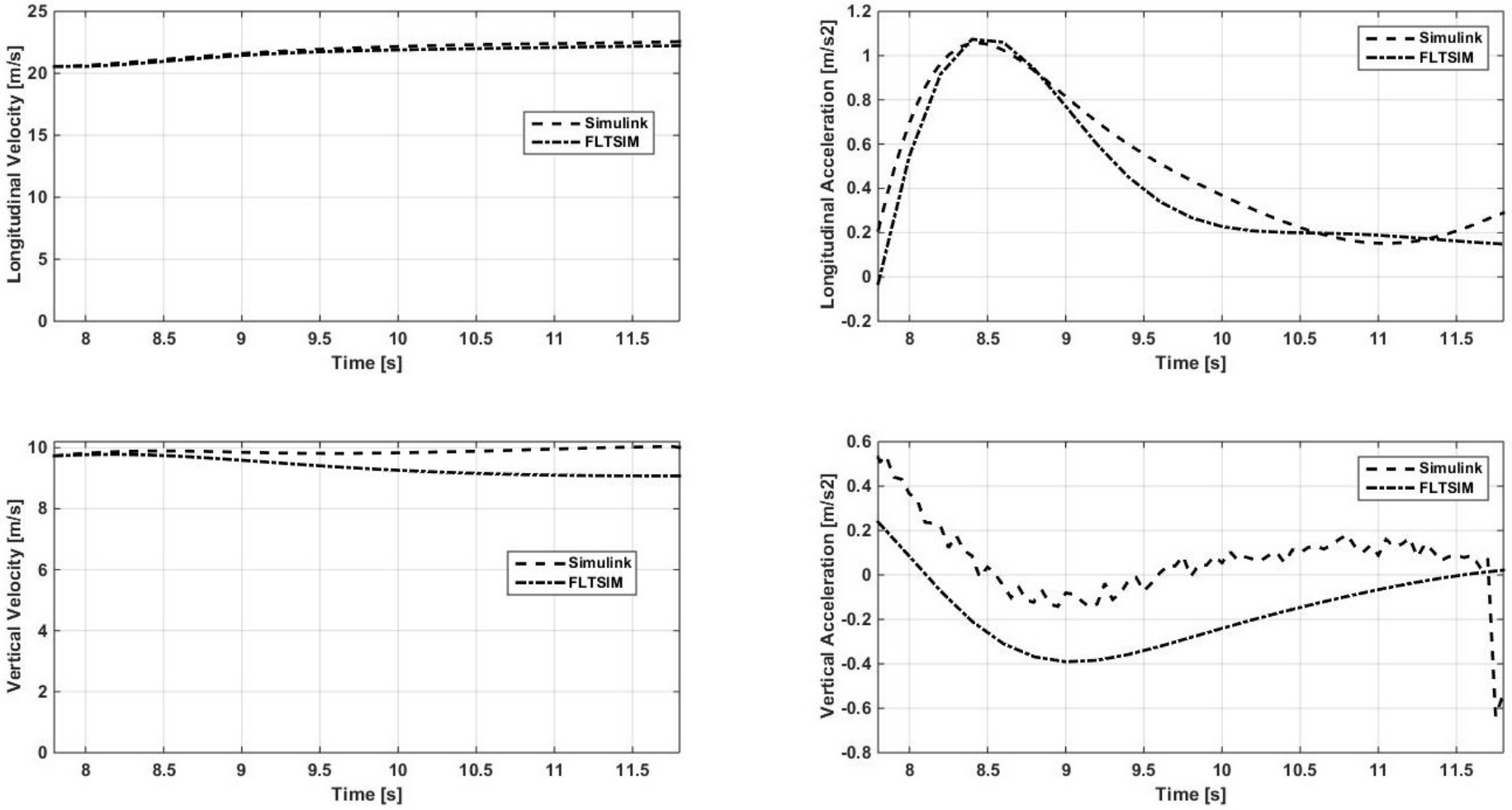


Figure 5.9: Time interval from 7.8 s to 11.8 s

There seems to be a trend in the pitch dynamics from zero seconds to this time interval, in that there is an error between the model and ideal trajectory. However, the error is very small, as can be seen in Figure 5.10. In this time interval the RW UAV is recovering from the instability and becoming stable or gliding as shown from Figures 5.10 to 5.12. This is evident in the model response particularly the main rotor speed, longitudinal velocity and vertical velocity. These parameters are vital in ensuring the autorotation manoeuvre is successful.

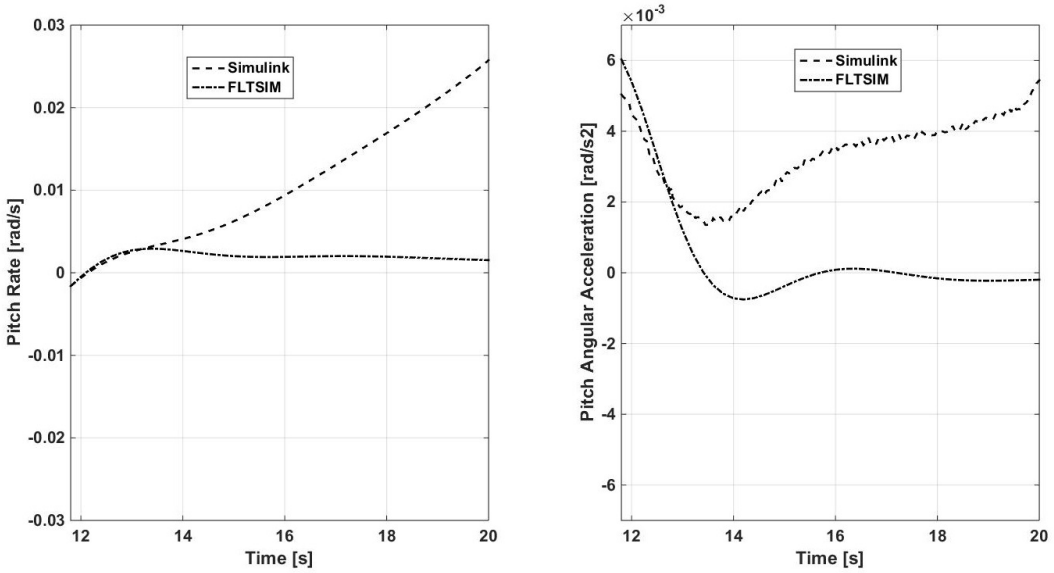


Figure 5.10: Pitch rate and acceleration from 11.8 s to 20 s

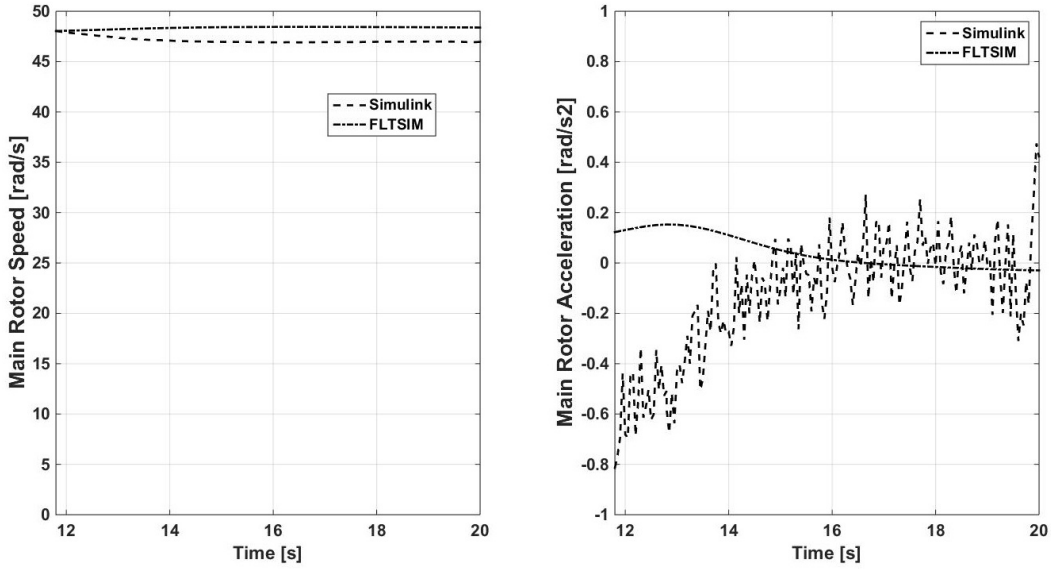


Figure 5.11: Main rotor from 11.8 s to 20 s

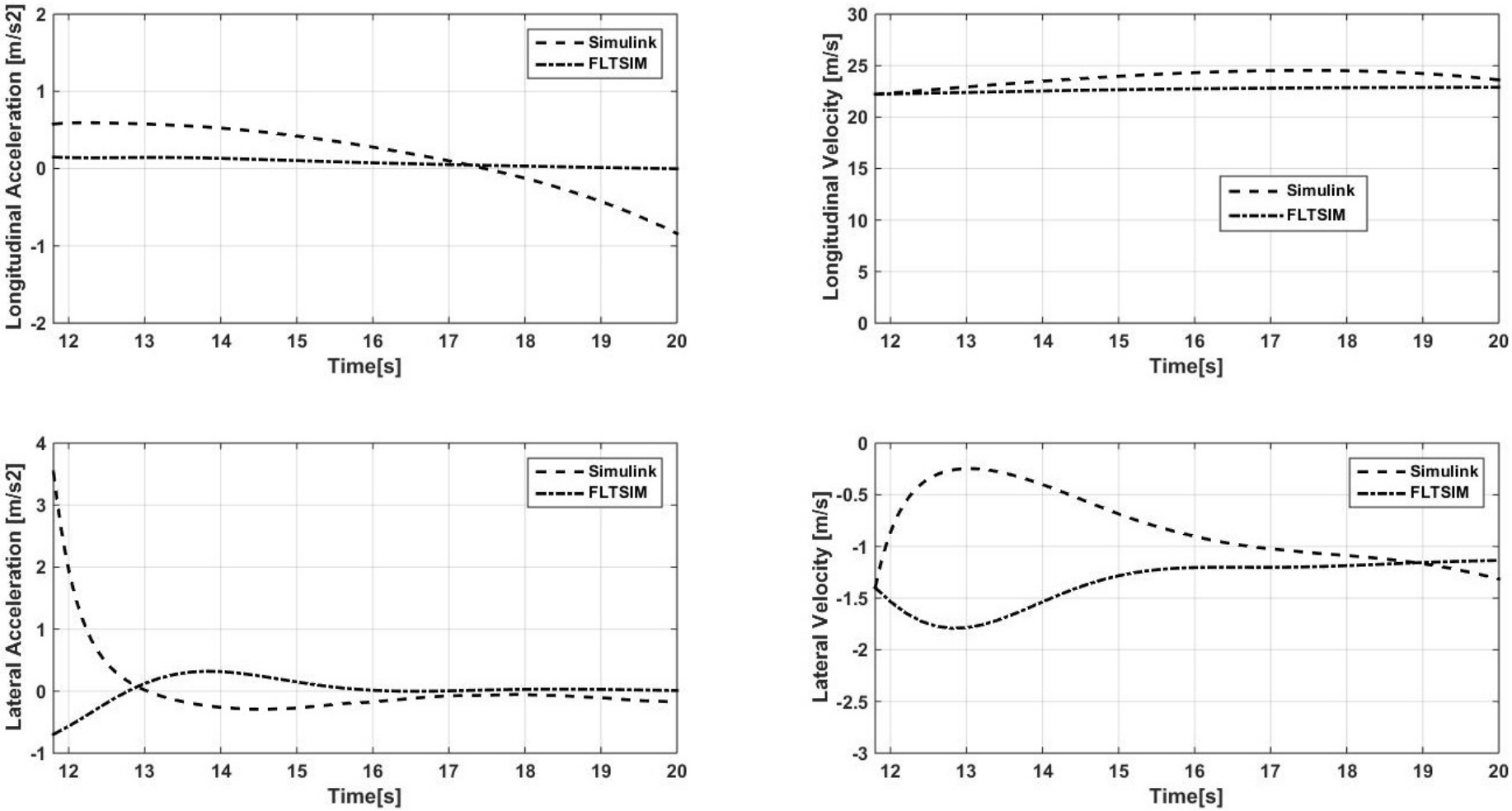


Figure 5.12: Time interval from 7.8 s to 11.8 s

In next 14.75 seconds the RW UAV is gliding to reach a height above the ground that is suitable to flare and land. In this time interval, the attitude of the RW UAV remains constants. Post 34 seconds there is a spike or peak, refer to Figures 5.13 to 5.15. The spike in the model pitch attitude is expected because there is a spike in the ideal trajectory and this could be due to the entry into the flare and land phase, the RW UAV becomes slightly unstable. The extreme drop in the main rotor acceleration post 34 seconds is due to the spike in the main rotor collective which is attributed to the change in flight phase, from glide to flare. This phenomenon is also evident in the vertical acceleration but it is not as severe. The mathematical model becomes unstable and this phenomenon should be investigated in future work.

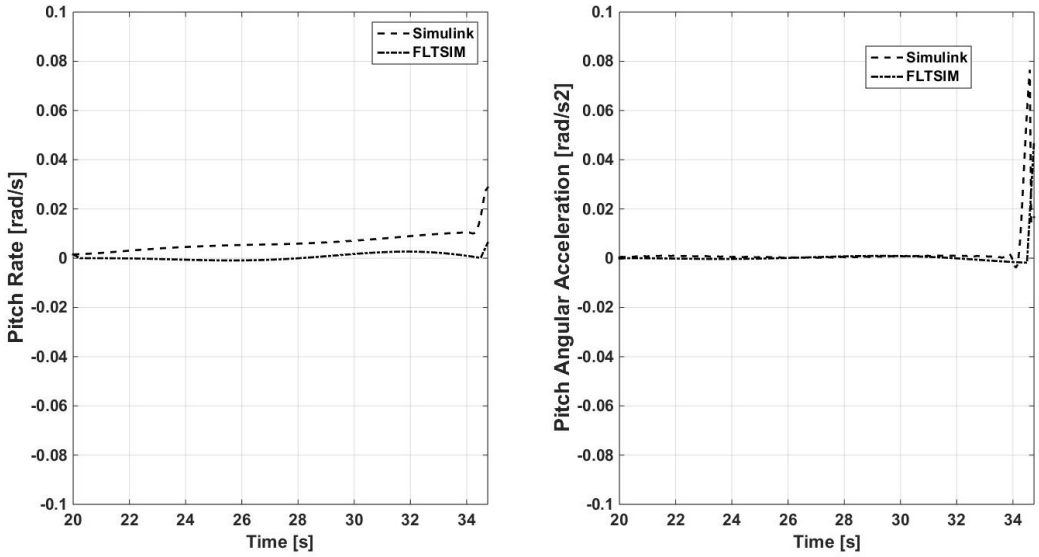


Figure 5.13: Pitch rate and acceleration from 20 s to 34.75 s

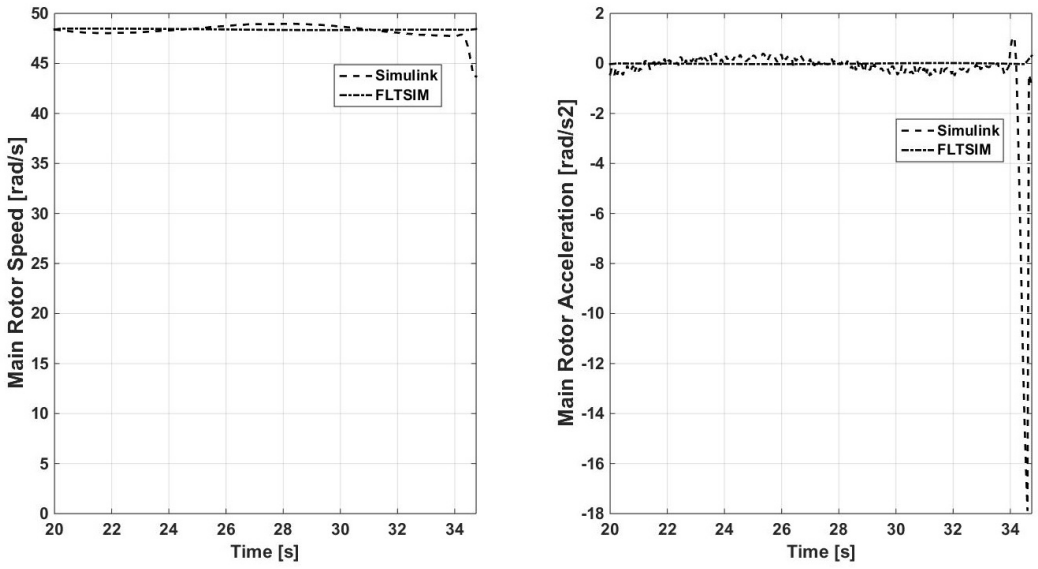


Figure 5.14: Main rotor from 20 s to 34.75 s

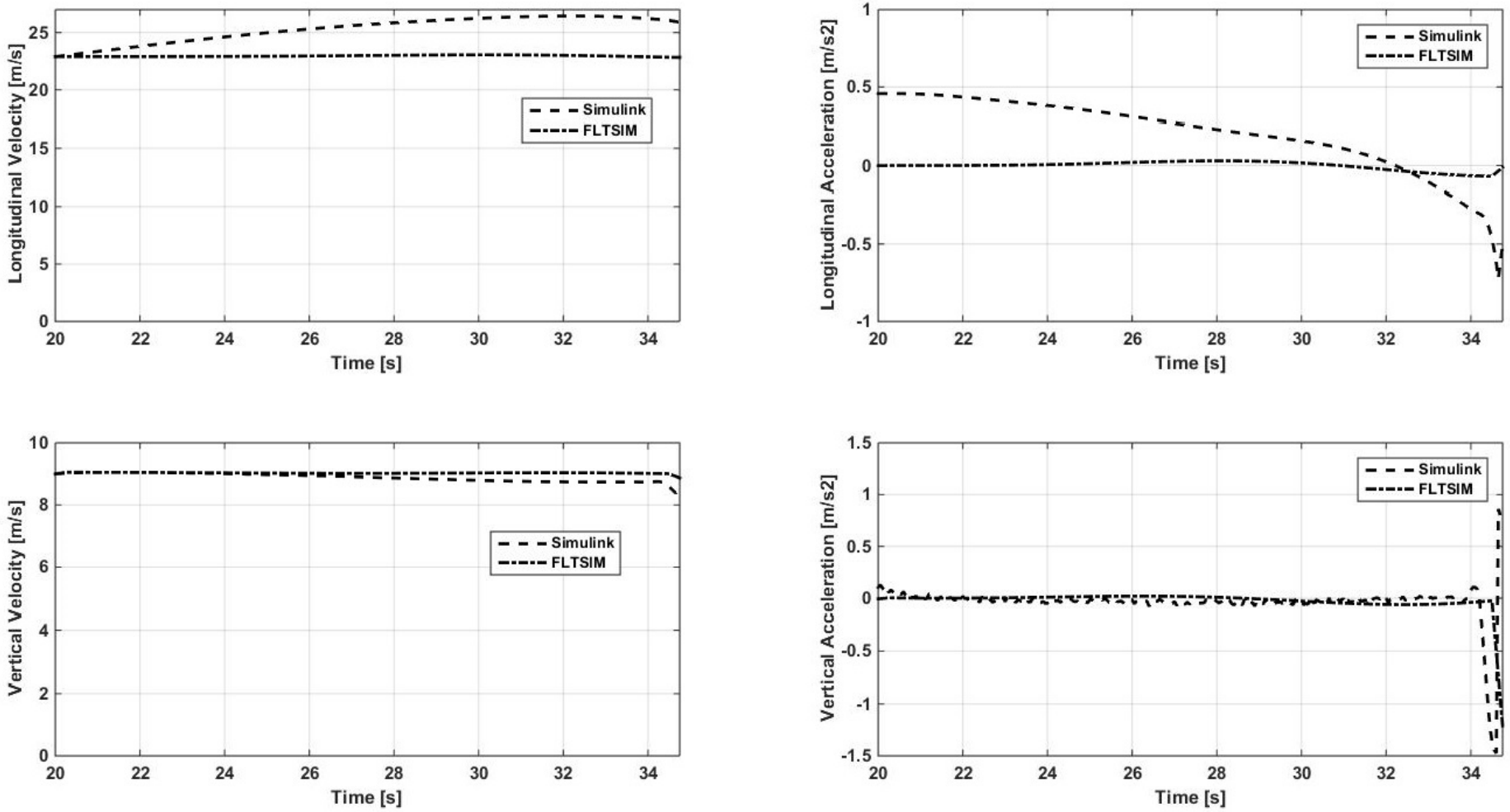


Figure 5.15: Time interval from 20 s to 34.75 s

The pitch rate of the mathematical model in the initial phase of flare and land illustrates a good trajectory, refer to Figure 5.16. From 39 seconds the RW UAV is nose down and the pitch angle is increasing. At touch down the nose down pitch angle should be small so that the nose of the aircraft does not hit the ground. The pitch derivatives should probably be tuned further to obtain a smaller pitch rate. As shown in Figure 5.17, the main rotor speed initially decays at a faster rate compared to the ideal trajectory but at touch down the two main rotor speeds are the same. The importance of flare is to decrease the longitudinal and vertical velocity so that the aircraft can have a soft landing to ensure that no damage to the aircraft occurs. The controller increases the main rotor collective pitch angle so that the longitudinal and vertical velocity decreases; this is evident in Figure 5.18. The longitudinal velocity at touch-down is approximately zero, which is excellent for landing. The vertical velocity is also small hence the landing would not be hard.

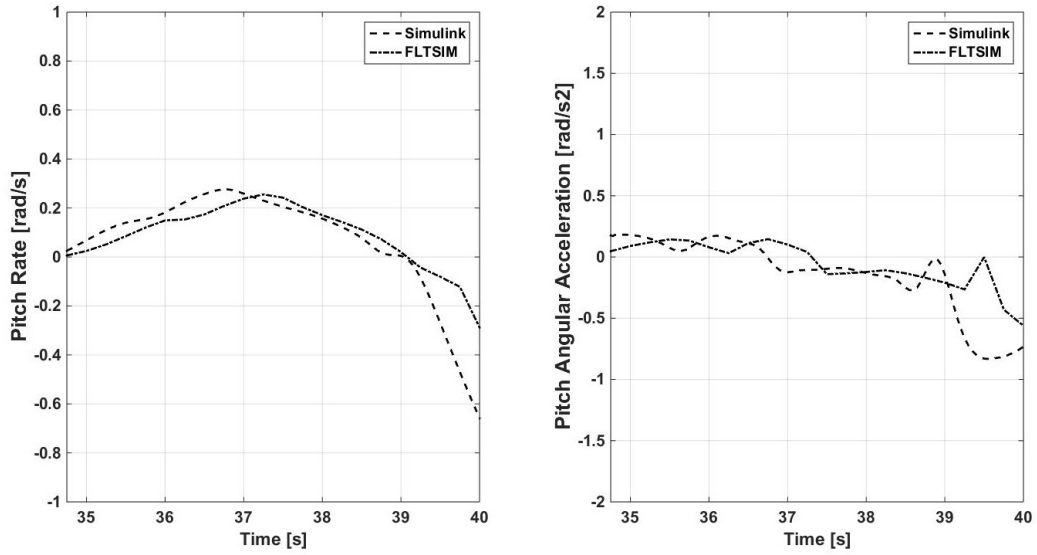


Figure 5.16: Pitch rate and acceleration, flare and land

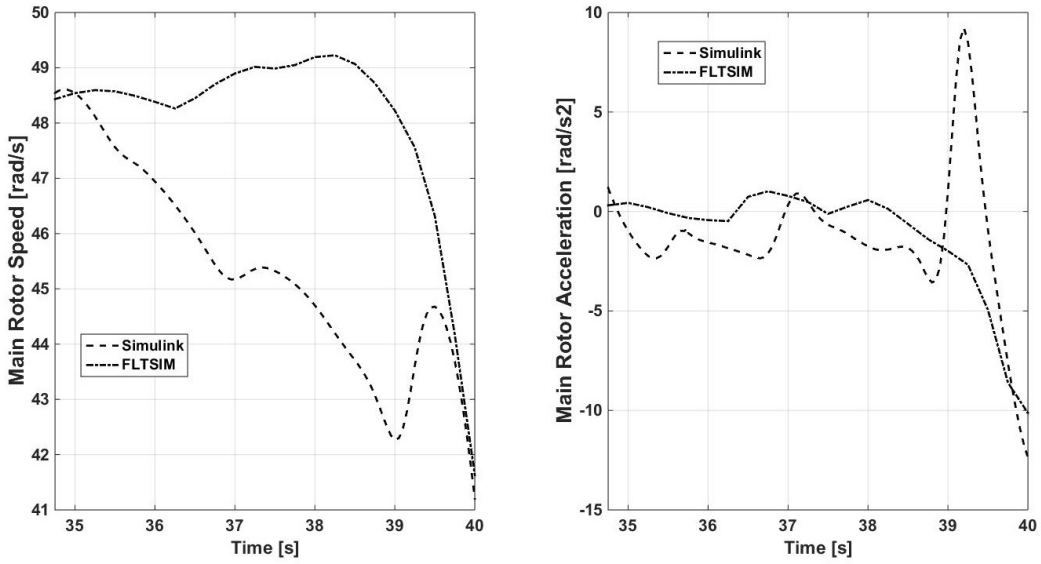


Figure 5.17: Main rotor flare and land

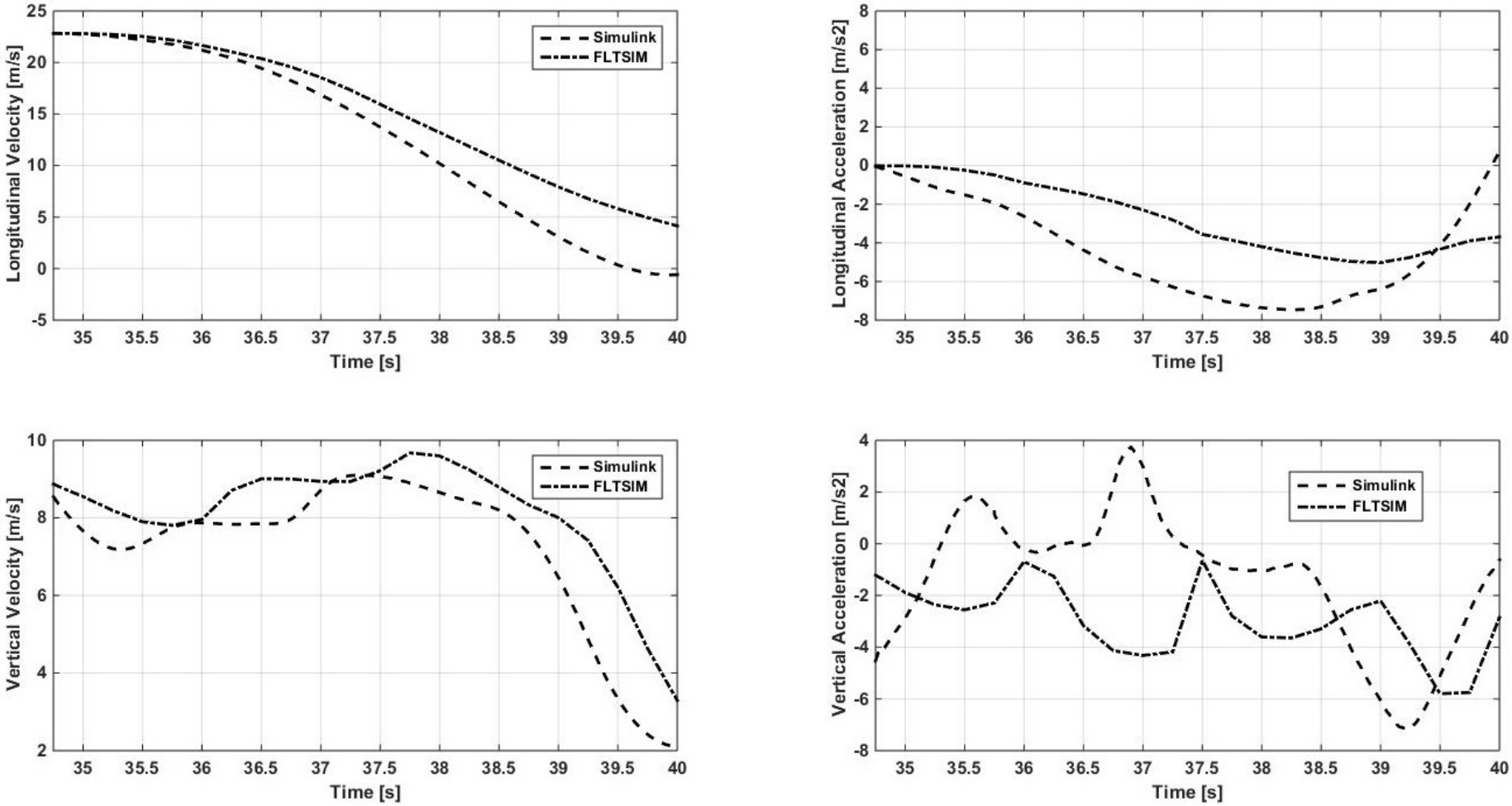


Figure 5.18: Flare and land phase

5.1.2 Lateral direction

In this section, the following time intervals were looked at: 0s to 7.8 and 34.5s to 40s; these are crucial because this is when the aircraft enters autorotation and flare and land phase. For the rest of the time interval plots refer to **Appendix D**.

As it has been mentioned in the previous section, the RW UAV is initially in steady level forward flight hence the lateral velocity should be zero. Referring to Figure 5.19, the combination of the model and the DDP controller, the lateral velocity is approximately 0.03 m/s which is small and negligible. The lateral acceleration follows a similar trend of the FLTSIM lateral acceleration trajectory. Both the roll rate and roll acceleration were modelled accurately. The yaw rate and yaw acceleration were modelled adequately, as shown in Figure 5.19.

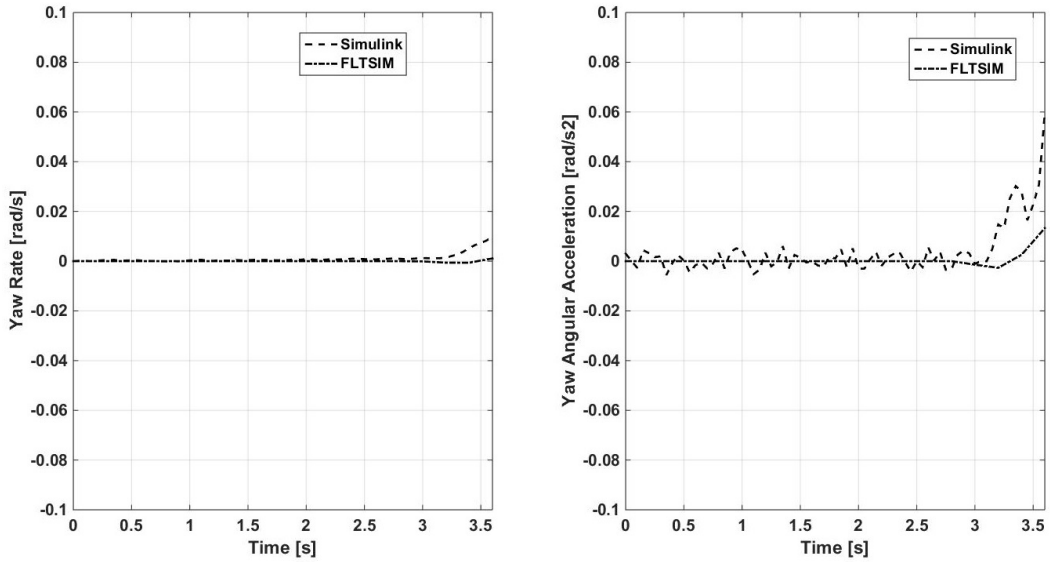


Figure 5.19: Yaw rate and yaw angular acceleration from 0 s to 3.6 s

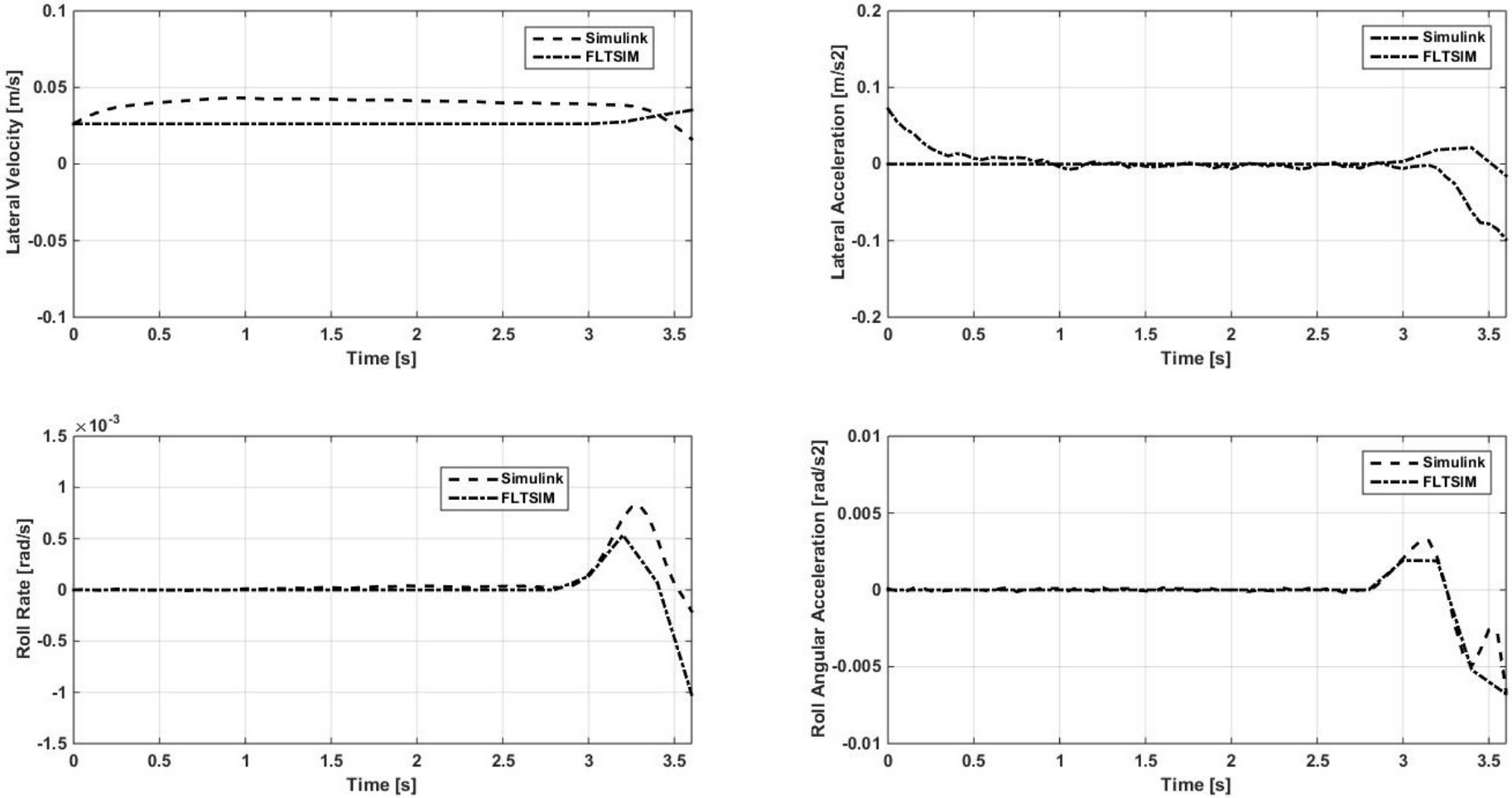


Figure 5.20: Time interval 0 s to 3.6 s

The lateral flight dynamics are not captured adequately in the last two seconds of this time interval, particularly the yaw dynamics and lateral velocity and acceleration, refer to Figures 5.21 and 5.22. The RW UAV has a side velocity which changes direction, hence the aircraft is moving from one side to the other. In other words, it is yawing from one side to the other. Possibly by correcting the lateral velocity, the yaw dynamics could also be corrected in the process. Focusing on the mathematical model of the RW UAV, specifically the lateral acceleration equation, there is no direct element of a control input but rather of a derivative that was identified in the system identification process. It could be worthwhile to tune the lateral derivative in order to capture the lateral dynamics or modify the lateral acceleration equation. As in the previous time interval, the roll dynamics were captured satisfactorily (see Figure 5.22).

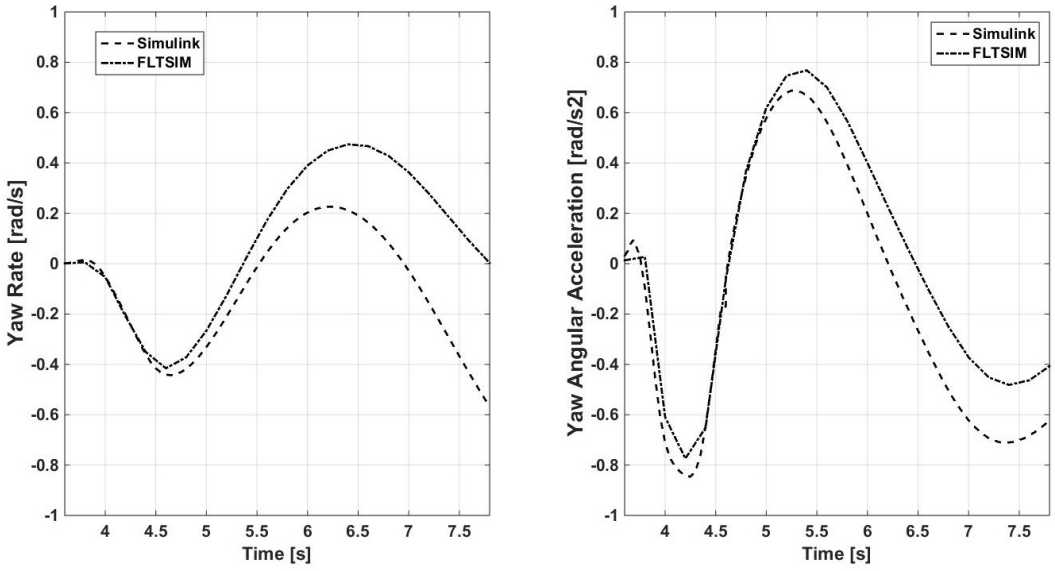


Figure 5.21: Yaw rate and yaw angular acceleration from 3.6 s to 7.8 s

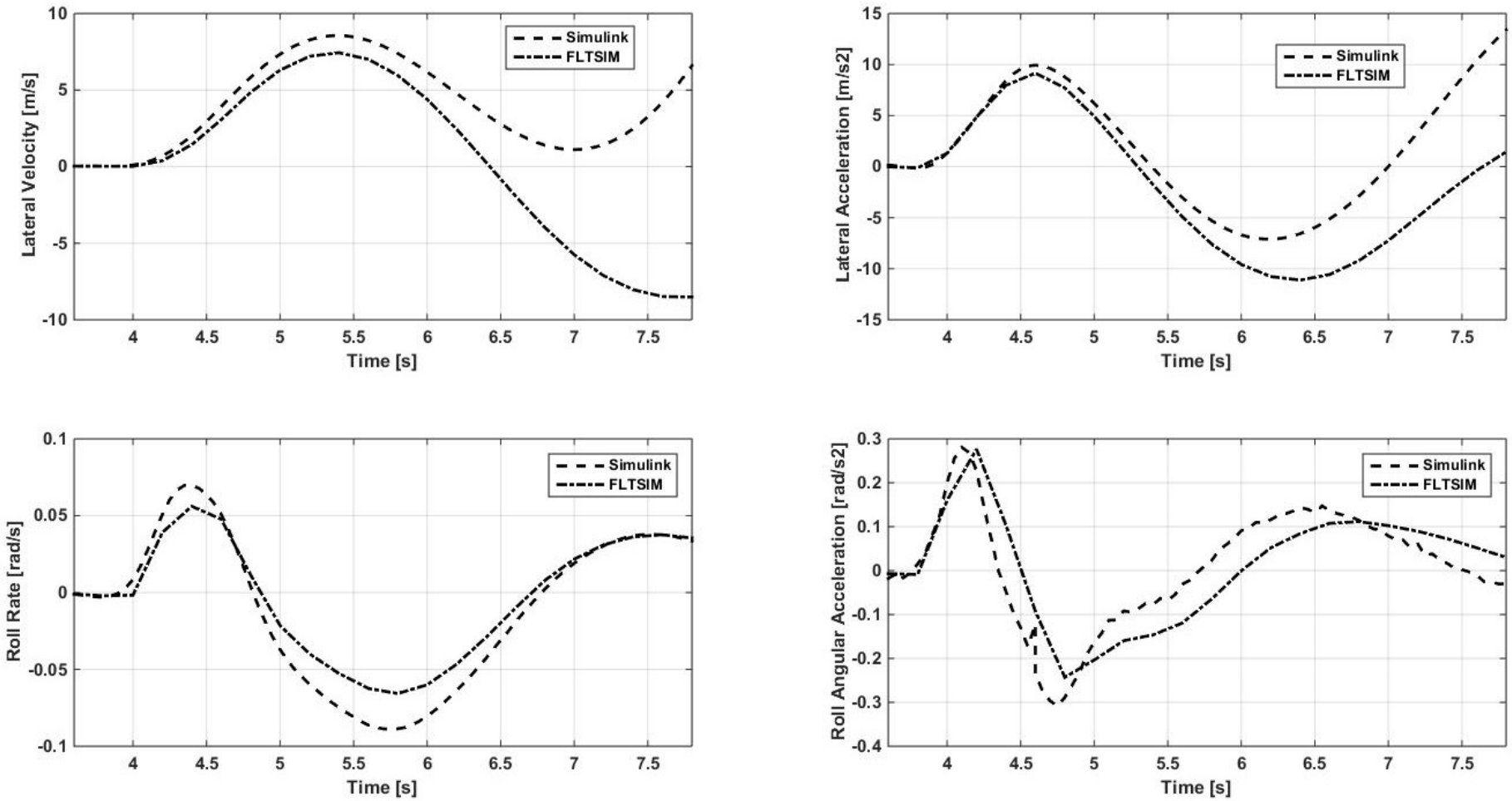


Figure 5.22: Time interval 3.6 s to 7.8 s

The roll dynamics in this time interval are not too concerning as it has been modelled to an adequate level. What seems to be of concern is the yaw rate. The model yaw rate is too high and there might be a number of reasons why it is so. The main reason could be due to that the tail rotor collective input is not increased to an acceptable level. There is an error in the lateral velocity but it converges towards 40s and is approximately zero at touch-down.

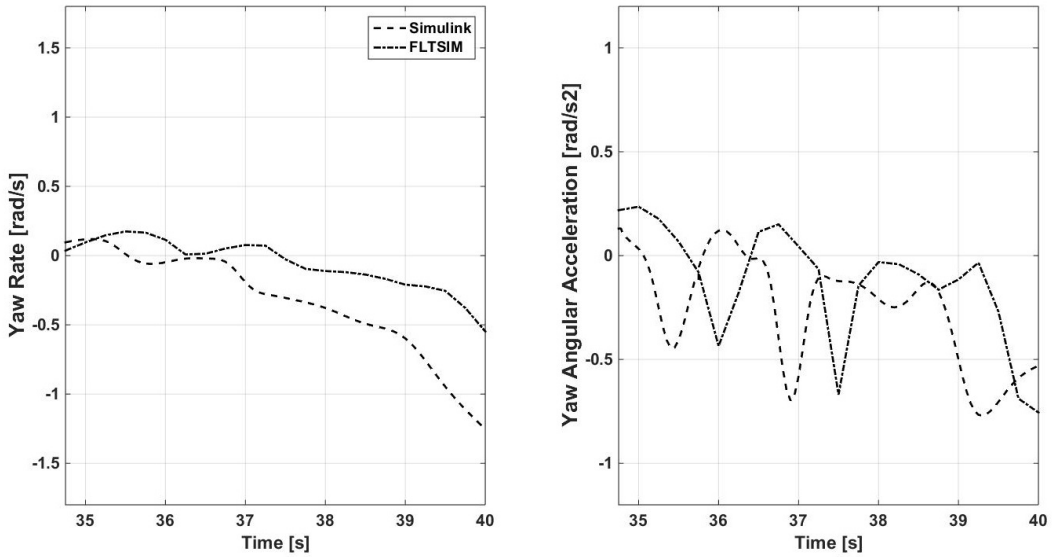


Figure 5.23: Yaw rate and yaw angular acceleration from 34.75 s to 40 s

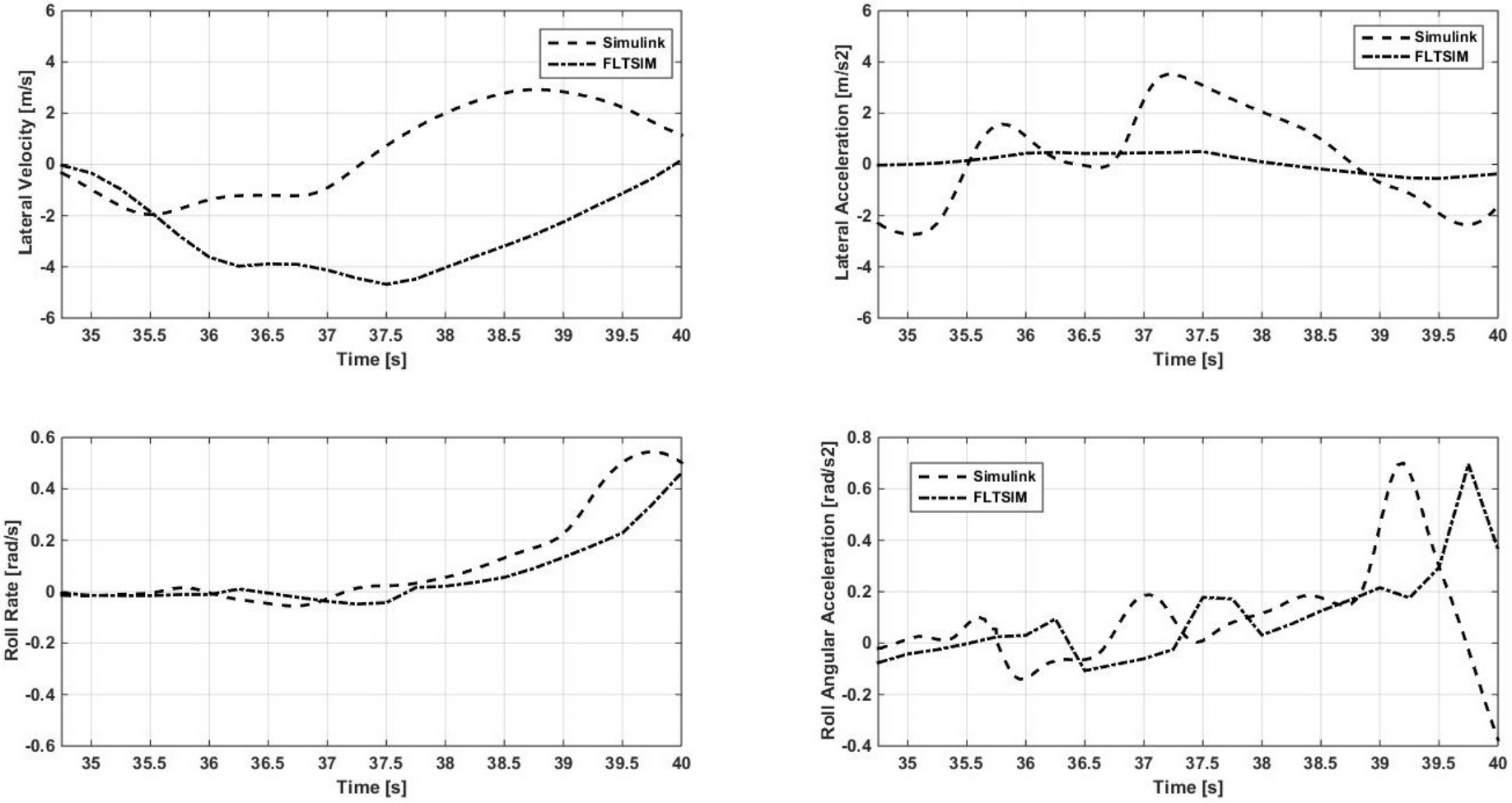


Figure 5.24: Time interval 34.75 s to 40 s

6 Conclusions and Recommendations

The aim of the study was to design a controller for autorotative landing of a RW UAV in the event of engine failure. Such a craft would represent a considerable capital outlay for any customer, as both the RW UAV as a platform and its payload comprising sophisticated sensors, would be costly items. Therefore, in the event of an engine failure, a means should be provided to get the craft safely back on the ground without incurring damage or causing danger. A study on autonomous autorotative landing was conducted by Stanford University but for RC helicopter, the XCell Tempest. This study sought to prove that a RW UAV weighing 560 kg would be able to autonomously execute an autorative landing following engine failure, controlled by a DDP controller.

An acceleration model was formulated and the unknown parameters were identified using MATLAB System Identification grey box modelling tool. This tool assumes some of the information about the dynamics or physical parameters are known meaning the physics of system or RW UAV could be included in the system identification process to a certain extent and therefore the tool makes certain assumptions in processing and identifying the unknown parameters. This led to some of the identified parameters not making logical and physical sense, which meant these parameters had to be tuned.

In order for the unknown parameters to be identified, FLTSIM was used to perform simulations for the regimes: entry into autorotation; glide; and flare and land. FLTSIM was populated with RW UAV geometric, kinematic and inertial parameters. The simulations yielded the necessary RW UAV response data for the control inputs to be used in the identification process.

The results showed that the mathematical model could not accurately model the pitch dynamics. The pitch rate is important, particularly in landing because if the RW UAV has a pitch down nose attitude in landing, depending on the size of the pitch angle, the nose may impact the ground and RW UAV may topple. The main

rotor dynamics were modelled satisfactorily, which are important for autorotation because without the power from the engine, the energy in main rotor would not be enough to facilitate successful execution of an autorotative landing. The longitudinal and vertical velocity were also satisfactory. More attention should be paid to the lateral velocity and the yaw dynamics in certain phases of the autorotation manoeuvre. The roll rate was modelled accurately.

The inaccuracies in the mathematical model could have been expected as Abbeel et al. 2005 stated that building an accurate helicopter model remains a challenge in autonomous flight. Inaccuracies are normally expected when simplified models are utilized.

The challenges faced in designing the DDP controller was in determining successive approximation magnitudes increments or decrements to obtain an optimum policy. As the choice of actions had to minimize future costs under the constraints of the state space dynamics. The DDP controller has the ability to control the RW UAV in an autorotation landing but the study should be taken further to improve certain aspects as detailed in the recommendations section. The major disadvantage of DDP is dimensionality and it is prevalent in a problem characterized by multiple states, which is the case in this study. This drove to a direction in which the control inputs were pre-calculated such that the simulation time could be drastically decreased.

A study on autonomous autorotative landing was conducted by Stanford University but for RC helicopter, the XCell Tempest. The research was deemed successful. Dalamagkidis (2009) derived a method to optimize the trajectory and control inputs but the method pre-calculated the control inputs and trajectory before entering into autorotation. This method is not robust with regards to modelling errors and outside interference, [Dalamagkidis, 2009]. This is illustrated in this research as the pre-calculated inputs are for certain conditions and not accounting for outside interference.

The major limitation of the study is that the DDP controller is designed for only ISA sea level conditions without turbulence and for a specific initial RW UAV flying attitude and speed. Opportunities for future research would be to broaden the scope by applying online system identification instead of offline system identification. This could eliminate the pre-calculated control inputs.

The section that follows details the recommendations to improve certain aspects of the study.

6.1 Recommendations

The following is recommended:

- To run the entire autorotation simulation manoeuvre from entry to touch down without dividing it into phases.
- The pitch dynamics should be investigated further to improve the current results. The pitch dynamics could be possibly tuned or the pitch model dynamics altered to capture the dynamics accurately.
- Yaw and lateral dynamics derivatives should probably be tuned to improve the results.
- In this study the control inputs are pre-calculated before running the autorotation simulation to decrease the processing time and the real simulation time, the control inputs should be calculated as the simulation is running for corrective actions in the form of a control input to be incorporated.
- To modify the offline system identification to online system identification such that the controller is robust and not limited to a particular flight conditions.
- Turbulence should be included, as this currently modelled for ideal flying conditions.

References

Abbeel, P., Coates, A., Hunter, T., and Ng, A. Y. (2009). Autonomous autorotation of an rc helicopter. In *Experimental Robotics*, pages 385–394. Springer.

Abbeel, P., Ganapathi, V., and Ng, A. Y. (2005). Learning vehicular dynamics with application to modeling helicopters. In *Advances in Neural Information Processing Systems*, pages 1–8.

Abraham, I., Delling, D., Goldberg, A. V., and Werneck, R. F. (2013). Alternative routes in road networks. *Journal of Experimental Algorithmics (JEA)*, 18:1–3.

Alvarenga, J., Vitzilaios, N. I., Valavanis, K. P., and Rutherford, M. J. (2015). Survey of unmanned helicopter model-based navigation and control techniques. *Journal of Intelligent & Robotic Systems*, 80(1):87–138.

Aponso, B., B. E. . L. D. (2005). Automated autorotation for unmanned rotorcraft recovery. Technical report, AHS International Specialists' meeting on unmanned rotorcraft.

Bagnell, J. A. and Schneider, J. G. S. (2001). Autonomous helicopter control using reinforcement learning policy search methods. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 2, pages 1615–1620. IEEE.

Bibik, P. and Narkiewicz, J. (2012). Helicopter optimal control after power failure using comprehensive dynamic model. *Journal of Guidance, Control, and Dynamics*, 35(4):1354–1362.

Cai, G., Chen, B. M., and Lee, T. H. (2010). An overview on development of miniature unmanned rotorcraft systems. *Frontiers of Electrical and Electronic Engineering in China*, 5(1):1–14.

Choudhury, S., Scherer, S., and Singh, S. (2012). Realtime alternate routes planning: the rrt*-ar algorithm.

Choudhury, S., Scherer, S., and Singh, S. (2013). Autonomous emergency landing of

a helicopter: Motion planning with hard time-constraints. *presented at the AHS 69th annual forum*.

Chow, V. (1971). Methodologies for water resources planning: Ddp and mlom (tlom). Technical report, Water Resource Center, University of Illinois.

Coates, A., Abbeel, P., and Ng, A. Y. (2008). Learning for control from multiple demonstrations. In *Proceedings of the 25th international conference on Machine learning*, pages 144–151. ACM.

Daily, D. I. (2012). A160 Hummingbird: Boeing's Variable-Rotor VTUAV kernel description.

Dalamagkidis, K. (2009). *Autonomous Vertical Autorotation for Unmanned Helicopters*. PhD thesis, Department of Computer Science and Engineering, College of Engineering, University of South Florida.

Dalamagkidis, K., Valavanis, K. P., and Piegl, L. A. (2010). Autonomous autorotation of unmanned rotorcraft using nonlinear model predictive control. *Journal of Intelligent and Robotic Systems*, 57(1-4):351–369.

Gavrilets, V., Martinos, I., Mettler, B., and Feron, E. (2002). Control logic for automated aerobatic flight of miniature helicopter. In *AIAA Guidance, Navigation and Control Conference*, number 2002-4834.

Godbolt, B. and Lynch, A. F. (2013). Physical input modelling and identification for a helicopter uav. In *Unmanned Aircraft Systems (ICUAS), 2013 International Conference on*, pages 890–896. IEEE.

Godbolt, B., Vitzilaios, N. I., and Lynch, A. F. (2013). Experimental validation of a helicopter autopilot design using model-based pid control. *Journal of Intelligent & Robotic Systems*, 70(1-4):385–399.

Jategaonkar, R. V. (2006). Flight vehicle system identification(a time domain methodology). *Progress in astronautics and aeronautics*.

Jategaonkar, R. V., Fischenberg, D., and Gruenhagen, W. (2004). Aerodynamic modeling and system identification from flight data-recent applications at dlr. *Journal of Aircraft*, 41(4):681–691.

Johnson, W. (1977). Helicopter optimal descent and landing after power loss. Technical report, National Aeronautics and Space Administration.

Karaman, S. and Frazzoli, E. (2010). Incremental sampling-based algorithms for optimal motion planning. *arXiv preprint arXiv:1005.0416*.

Kendoul, F. (2012). Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems. *Journal of Field Robotics*, 29(2):315–378.

Kensaku, H., spicy Fu-dama, Daigo, F., Igarashi, K., and Kenzo, N. (2004). Autonomous autorotation landing of small unmanned helicopter. *The Japan Society of Mechanical Engineers Part C*, 70(698):2862–2869.

Khan, R., Williams, P., Hill, R., Bil, C., et al. (2011). Fault tolerant flight control system design for uav's using nonlinear model predictive control.

Meng, W. and Chen, R. (2013). Study of helicopter autorotation landing following engine failure based on a six-degree-of-freedom rigid-body dynamic model. *Chinese Journal of Aeronautics*, 26(6):1380–1388.

Mettler, B. (2003). Autonomous uav guidance build-up: Flight-test demonstration and evaluation plan.

Mettler, B., Tischler, M. B., and Kanade, T. (1999). System identification of small-size unmanned helicopter dynamics. In *Annual Forum Proceedings- American Helicopter Society*, volume 2, pages 1706–1717.

Natarajan, A. and Balasubramani, P. (2006). *Operations Research*. Pearson Education India.

Ng, A. Y. (2003). *Shaping and policy search in reinforcement learning*. PhD thesis, Computer Science in the Graduate Division, University of California, Berkeley.

Padfield, G. D. (2008). *Helicopter flight dynamics*. John Wiley & Sons, Oxford.

Prouty, R. W. (1995). *Helicopter performance, stability, and control*.

Rahideh, A., Shaheed, H., and Bajodah, A. H. (2007). Adaptive non-linear model inversion control of a twin rotor multi-input multi-output system using artificial intelligence. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 221(3):343–351.

Raol, J. R., Girija, G., and Singh, J. (2004). *Modelling and Parameter Estimation of Dynamic Systems*, volume 65. Iet.

Santamaría, D., Viguria, A., Bejar, M., Kondak, K., and Ollero, A. (2013). Towards autonomous autorotation landing for small size unmanned helicopters. *Journal of Intelligent & Robotic Systems*, 69(1-4):171–180.

Scherer, S. and Singh, S. (2011). Multiple-objective motion planning for unmanned aerial vehicles. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 2207–2214. IEEE.

Taamallah, S. (2012). Optimal autorotation with obstacle avoidance for a small-scale flybarless helicopter uav. In *AIAA Guidance, Navigation and Control Conference*, pages 13–16.

Tassa, Y., Erez, T., and Smart, W. D. (2008). Receding rorizon differential dynamic programming. In *Advances in Neural Information Processing Systems*, pages 1465–1472.

Theodorou, E., Tassa, Y., and Todorov, E. (2010). Stochastic differential dynamic programming. In *American Control Conference (ACC), 2010*, pages 1125–1132. IEEE.

Tierney, S. (2010). *Autorotation Path Planning Using Backward Reachable Set and Optimal Control*. PhD thesis, The Pennsylvania State University.

Williams, K. W. (2004). A summary of unmanned aircraft accident/incident data: Human factors implications. Technical report, DTIC Document.

Yomchinda, T. (2013). *Real-time path planning and autonomous control for helicopter autorotation*. PhD thesis, The Pennsylvania State University.

Yomchinda, T., Horn, J., and Langelaan, J. (2011). Autonomous control and path planning for autorotation of unmanned helicopters. In *Proceedings of the American Helicopter Society 68th Annual Forum, Fort Worth, Texas*.

Appendix A Nonlinear Simulation File

——————— SIMULATION: VARIABLES ——————— TITLE .....

SERVICE H-L7.2tWKLFprofF-A

USERS INITIALS ..... F-Acont and Wdisp

GRAPHICS OUTPUT: 'ECRAN' OR 'AUTO ' ..... ECRAN

LISTINGS: 'ECRAN' OR 'PRINT' ..... PRINT

AUTO PILOT (OFF/ON): ALTITUDE ..... OFF

SPEED VH ..... OFF

SIDE SLIP CORRECTION ..... OFF

HEADING BY YAW CONTROL .... OFF

HEADING BY ROLL CONTROL .... ON

PITCH ATTITUDE ..... OFF

ROLL ATTITUDE ..... ON

PRECOMMANDS: COLLECTIVE-LATERAL ..... OFF

COLLECTIVE-LONGITUDINAL ..... OFF

COLLECTIVE-YAW ..... OFF

STABILATOR ..... OFF

C OF G (M): XG (FO:>0 N :=0 AF:<0) ..... -0.08

YG (L :<0 N :=0 R :>0) ..... 0.

ALTITUDE H (M) .... 0.

TEMPERATURE TC (DEG.C) .... 15.

ROTOR RPM NR (NOMINAL BY DEFAULT) .... 463.

SIMULATION TIME (S) 20.5

TABLES: NUMBER OF POINTS .... 98.

TIME T (S)

0.2 0.4 0.6 0.8 1.0 1.2 1.4 1.6 1.8 2.0 2.2 2.4 2.6 2.8 3.0 3.2 3.4 3.6 3.8 4.0 4.2 4.4 4.6 4.8 5.0 5.2 5.4 5.6 5.8 6. 6.2 6.4 6.6 6.8 7. 7.2 7.4 7.6 7.8 8. 8.2 8.4 8.6 8.8 9. 9.2 9.4 9.6 9.8 10. 10.4 10.6 10.8 11. 11.2 11.4 11.6 11.8 12. 12.2 12.4 12.6 12.8 13. 13.2 13.4 13.6 13.8 14. 14.2 14.4 14.6 14.8 15. 15.2 15.4 15.6 15.8 16. 16.2 16.4 16.6 16.8 17. 17.2 17.4 17.6 17.8 18. 18.2 18.4 18.6 18.8 19. 19.2 19.4 19.6 19.8

PILOT INPUTS: COLLECTIVE DT0 (DEG)

0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.40 0.77 1.12 1.43 1.63 1.63 1.63 0.40 0.77 1.12 1.43 1.63 1.63 1.63 0.26 0.52 0.78 1.04 1.3 1.56 1.82 1.82 2.5 4. 6. 8. 9. 9.5

LATERAL STICK DDL (

0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.

LONGITUDINAL STICK DDM (

0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. -0.2 -0.4 -0.6 -0.8 -1. -1.2 -1.4 -1.4 -1.2 -1.2 -0.6 0.2 0.4 0.6 0.8 1.0 1.5 2. 2.5 3.0 3.5 4.0 6. 7. 8.

YAW PEDALS DDN (

0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.

0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.

AVAILABLE POWER WDISP (KW)

-43.23 -43.23 -43.23 -43.23 -43.23 -43.23 -43.23 -43.23 -43.23 -43.23 -43.23 -43.23
-43.23 -43.23 -43.23 -43.23 -43.23 -43.23 -43.23 -43.23 -43.23 -43.23 -43.23 -43.23
-43.23 -43.23 -43.23 -43.23 -43.23 -43.23 -43.23 -43.23 -43.23 -43.23 -43.23 -43.23
-43.23 -43.23 -43.23 -43.23 -43.23 -43.23 -43.23 -43.23 -43.23 -43.23 -43.23 -43.23
-43.23 -43.23 -43.23 -43.23 -43.23 -43.23 -43.23 -43.23 -43.23 -43.23 -43.23 -43.23
-43.23 -43.23 -43.23 -43.23 -43.23 -43.23 -43.23 -43.23 -43.23 -43.23 -43.23 -43.23
-43.23 -43.23 -43.23 -43.23 -43.23 -43.23 -43.23 -43.23 -43.23 -43.23 -43.23 -43.23
-43.23 -43.23 -43.23 -43.23 -43.23 -43.23 -43.23 -43.23 -43.23 -43.23 -43.23 -43.23
-43.23 -43.23

A.P. COMMANDS: ALTITUDE H (M)

0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.

HORIZONTAL SPEED VH (KM/H)

0. 0. 0. -0.05 -0.15 -0.5 -0.75 -1.15 -1.85 -2.5 -3.2 -3.7 -4.2 -4.8 -5.5 -5.9 -6.3 -6.9
-7.2 -7.6 -8.1 -10.5 -12.5 -15. -20. -25. -30. -35. -40. -45. -50. -55. -60. -65. -70.
-75. -80. -85.15 -85.15 -85.15 -85.15 -85.15 -85.15 -85.15 -85.15 -85.15 -85.15 -85.15
-85.15 -85.15 -85.15 -85.15 -85.15 -85.15 -85.15 -85.15 -85.15 -85.15 -85.15 -85.15
-85.15 -85.15 -85.15 -85.15 -85.15 -85.15 -85.15 -85.15 -85.15 -85.15 -85.15 -85.15
-85.15 -85.15 -85.15 -85.15 -85.15 -85.15 -85.15 -85.15 -85.15 -85.15 -85.15 -85.15
-85.15 -85.15 -85.15 -85.15 -85.15 -85.15 -85.15 -85.15 -85.15 -85.15 -85.15 -85.15
-85.15 -85.15

ROLL ATTITUDE PHI (DEG)

0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.

PITCH ATTITUDE TETA (DEG)

0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.

HEADING PSI (DEG)

0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.

STABILATOR TETAB (DEG)

0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0.

——————— SIMULATION: RUN OPTIONS ———————

AUTOROTATION FROM TRIM M=560. VH=85.15 VZ=-7.75 11 END

Appendix B System Identification Model File

$function[dx, y1] = acc_model(t, x, u, Cu, Cv, Cw1, Cw2, Cw3, Cw4, Co1, Co2, Co3, Co4, Co5, Co6, Cp1, Cp2, Cp3, Cq1, Cq2, Cq3, Cr1, Cr2, Cr3, varargin)$

g = 9.81;

output equations

y1(1) = x(1);

y1(2) = x(2);

y1(3) = x(3);

y1(4) = x(4);

y1(5) = x(5);

y1(6) = x(6);

y1(7) = x(7);

y1(8) = x(8);

y1(9) = x(9);

y1(10) = x(10);

states update

$dx(1) = x(2) * u(7) - x(3) * u(6) - g * sin(u(9)) + Cu * x(1);$

$dx(2) = x(3) * u(5) - x(1) * u(7) + g * cos(u(9)) * sin(u(8)) + Cv * x(2);$

$dx(3) = x(1) * u(6) - x(2) * u(5) + g * cos(u(9)) * cos(u(8)) + Cw1 + Cw2 * x(3) + Cw3 * u(3) * x(6) + Cw4 * sqrt((x(1))^2 + (x(2))^2);$

$dx(4) = u(5) + (u(6) * sin(x(4)) + u(7) * cos(x(4))) * tan(x(5));$

$dx(5) = u(6) * cos(x(4)) - u(7) * sin(x(4));$

$dx(6) = Co1 + Co2 * x(6) + Co3 * u(3) + Co4 * x(3) + Co5 * sqrt((x(1))^2 + (x(2))^2) + Co6 * ((u(2))^2 + (u(1))^2);$

$dx(7) = Cp1 + Cp2 * u(5) + Cp3 * u(2) * x(6);$

$$dx(8) = Cq1 + Cq2 * u(6) + Cq3 * u(1) * x(6);$$

$$dx(9) = Cr1 + Cr2 * u(7) + Cr3 * u(4) * x(6);$$

$$dx(10) = (u(6) * sin(u(8)) + u(7) * cos(u(8)))/cos(u(9));$$

Appendix C System Identification Plots

Time Interval 4 – 8 seconds:

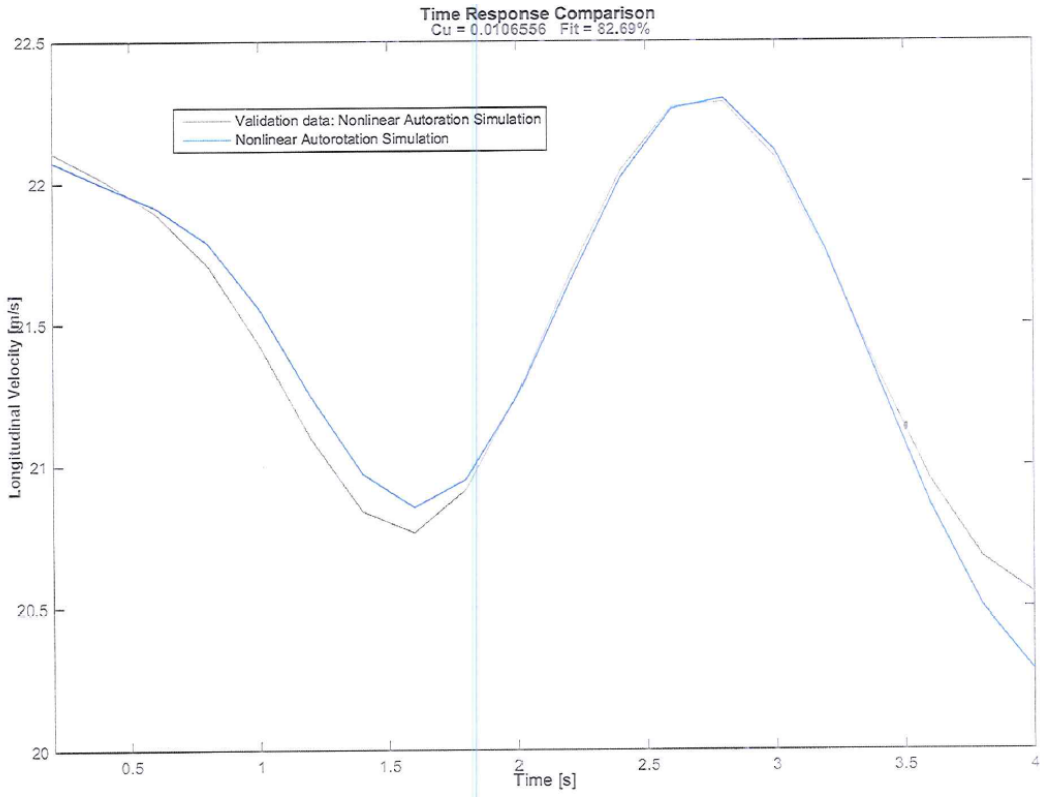


Figure C.1: Time response comparison: longitudinal velocity

Time Interval 8 – 12 seconds:

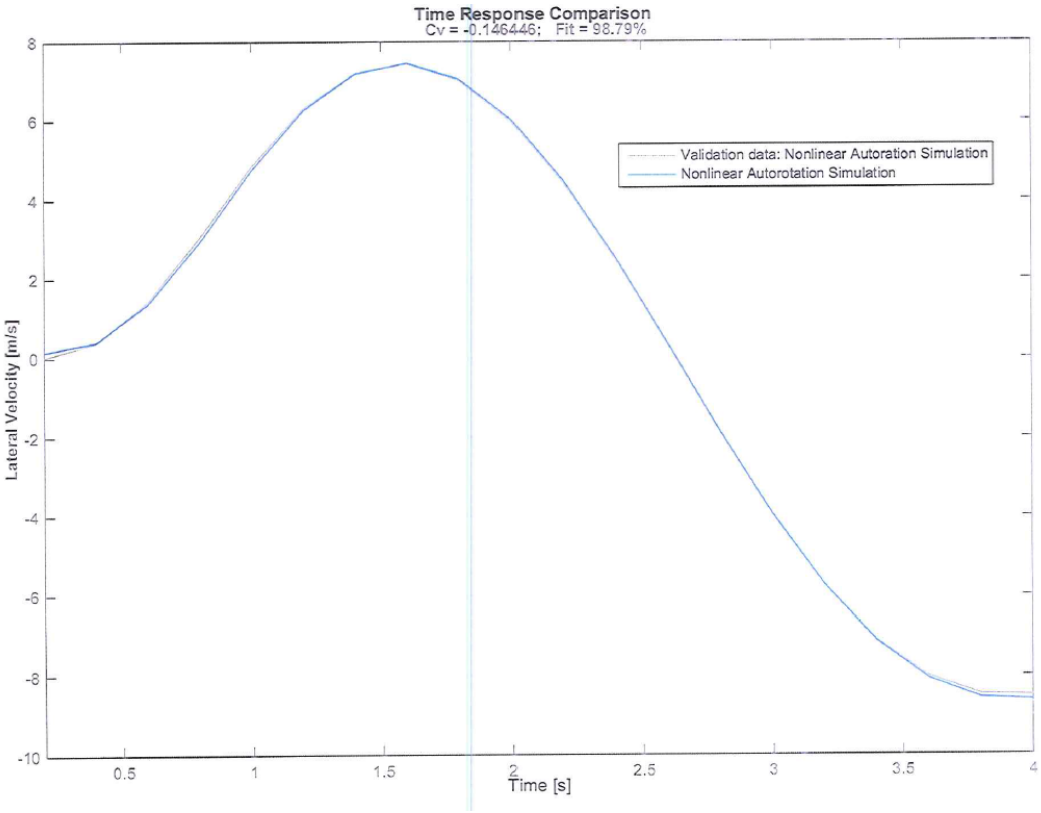


Figure C.2: Time response comparison: lateral velocity

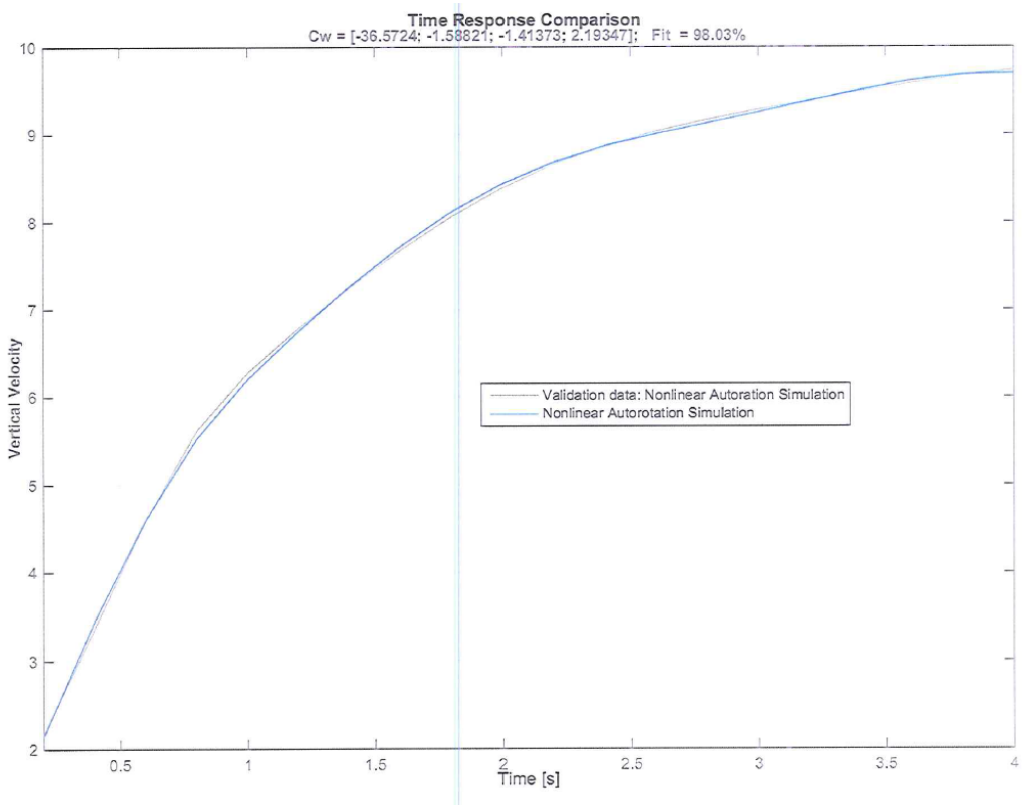


Figure C.3: Time response comparison: vertical velocity

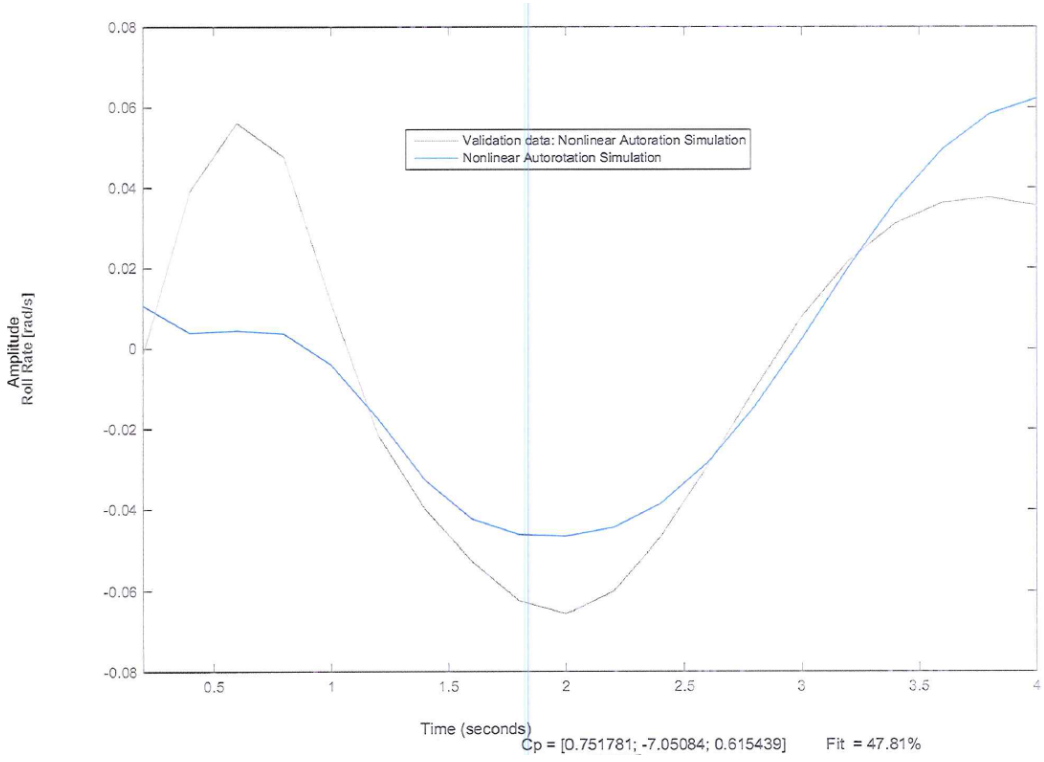


Figure C.4: Time response comparison: roll rate

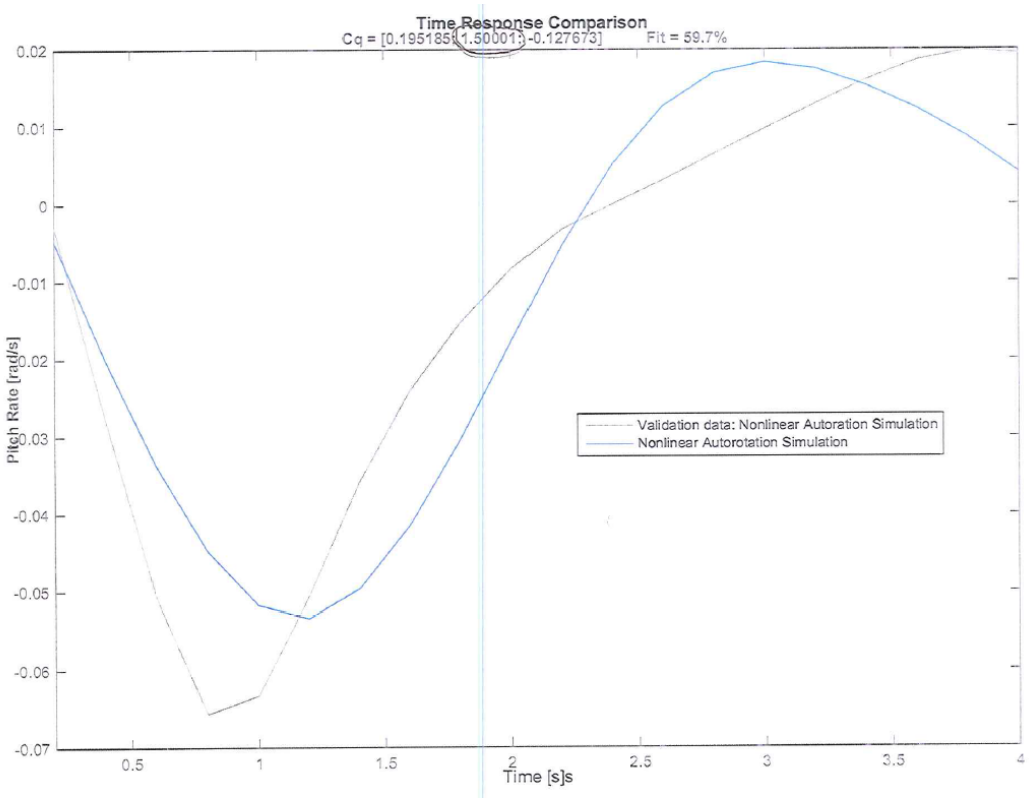


Figure C.5: Time response comparison: pitch rate

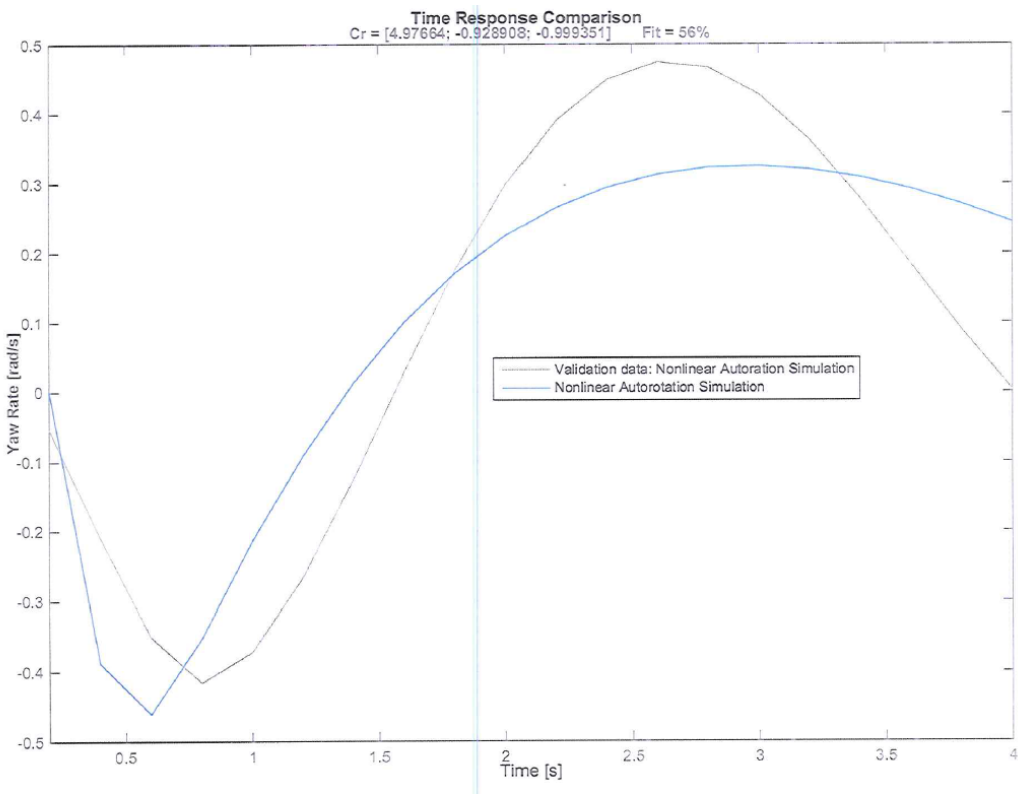


Figure C.6: Time response comparison: yaw rate

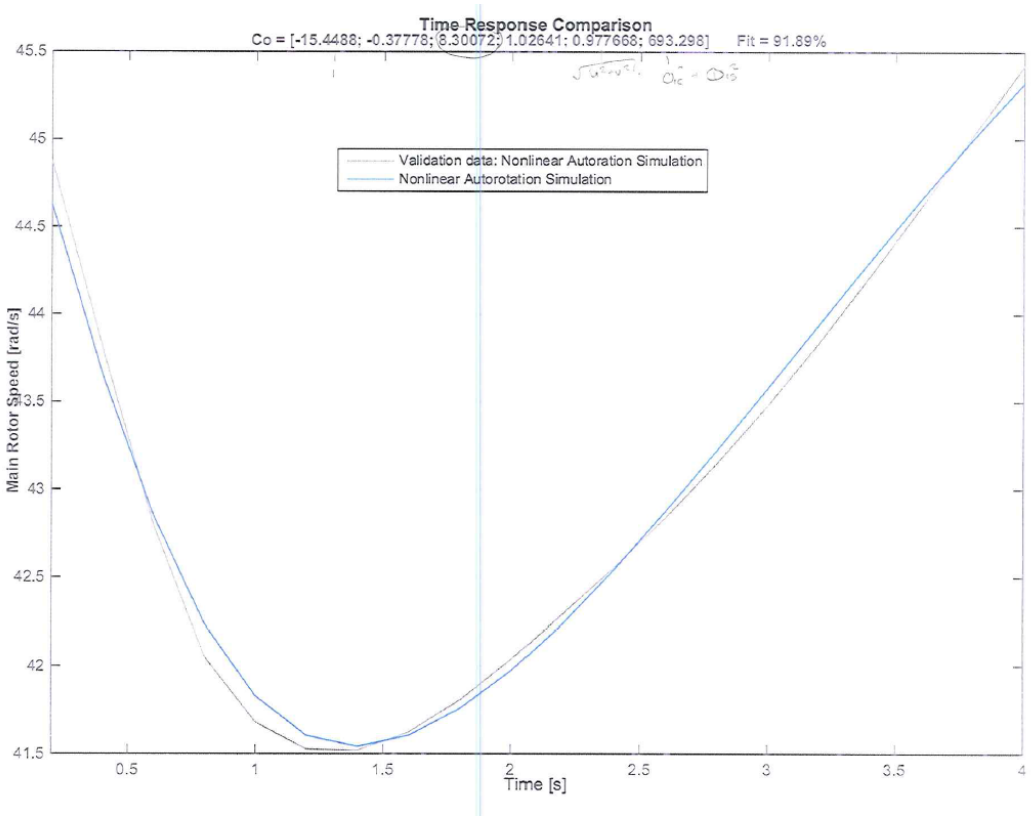


Figure C.7: Time response comparison: main rotor speed

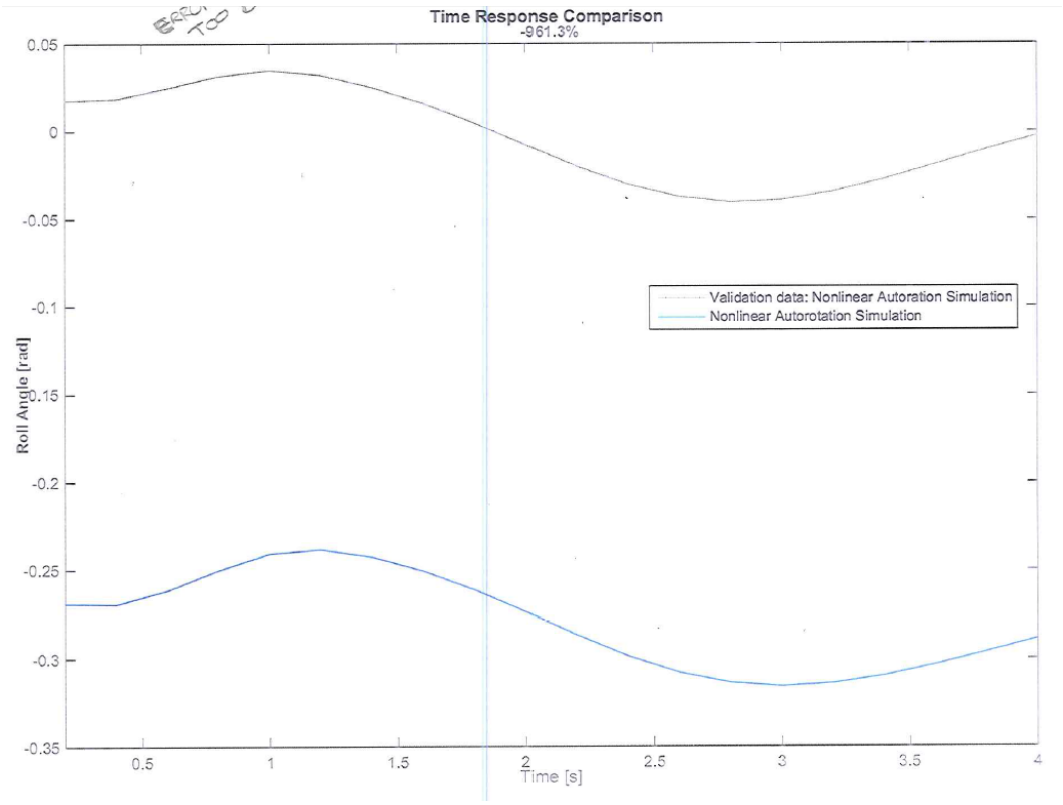


Figure C.8: Time response comparison: roll angle

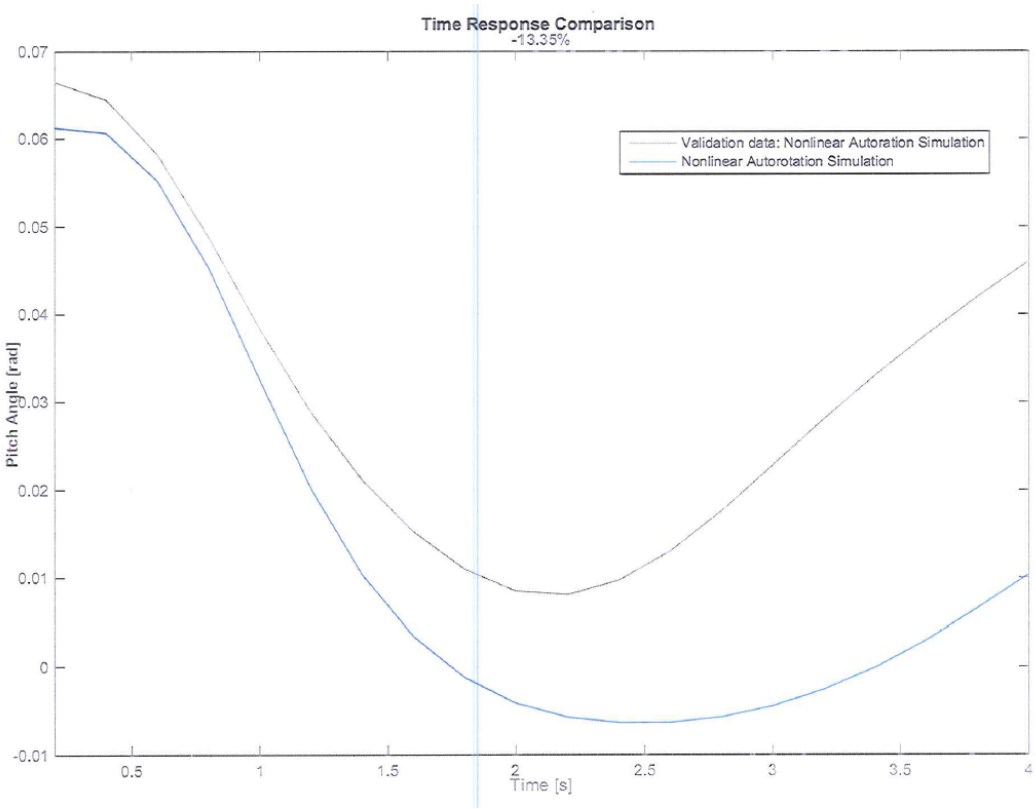


Figure C.9: Time response comparison: pitch angle

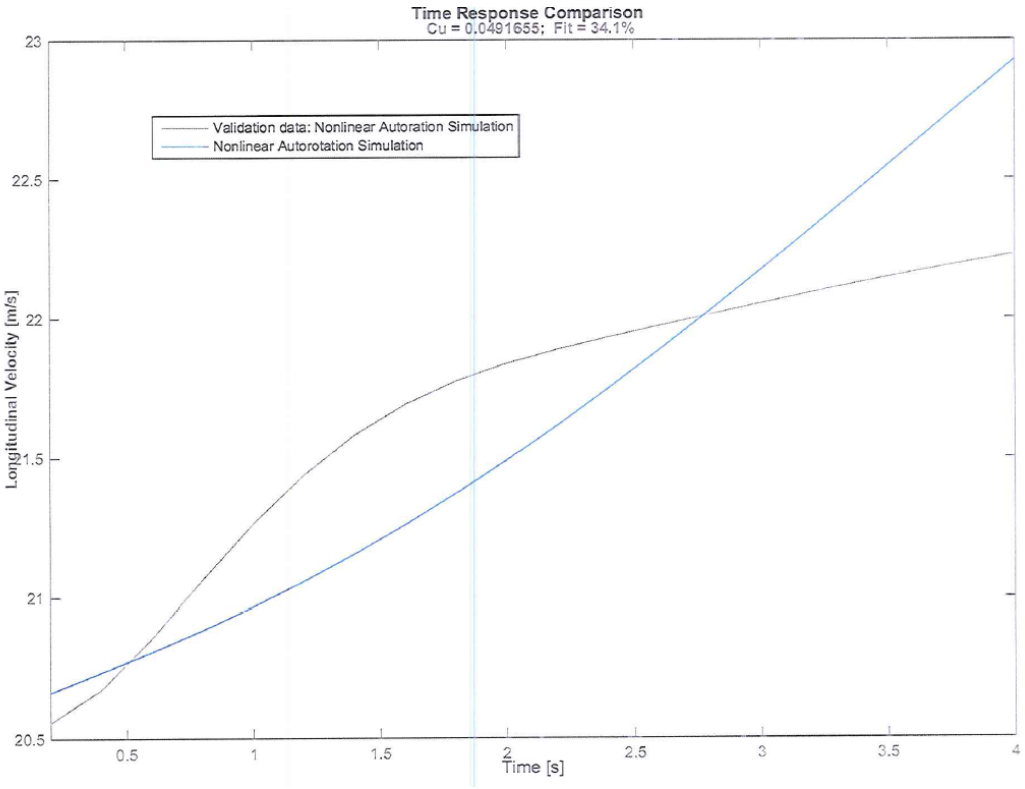


Figure C.10: Time response comparison: longitudinal velocity

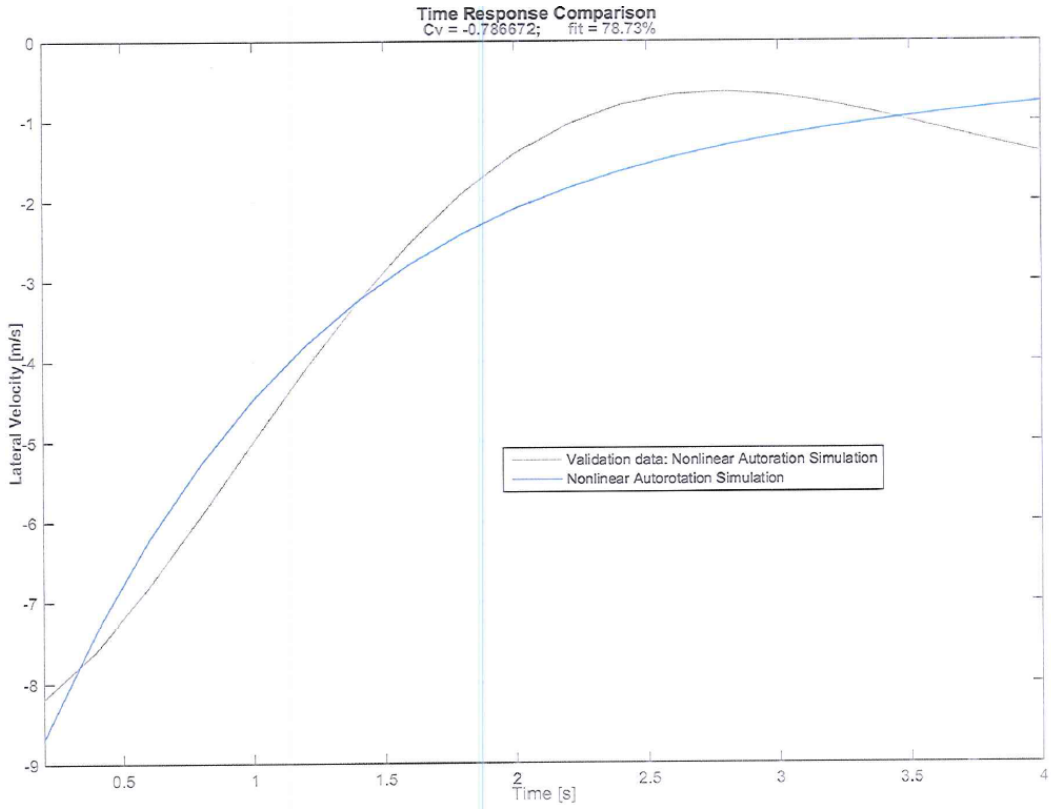


Figure C.11: Time response comparison: lateral velocity

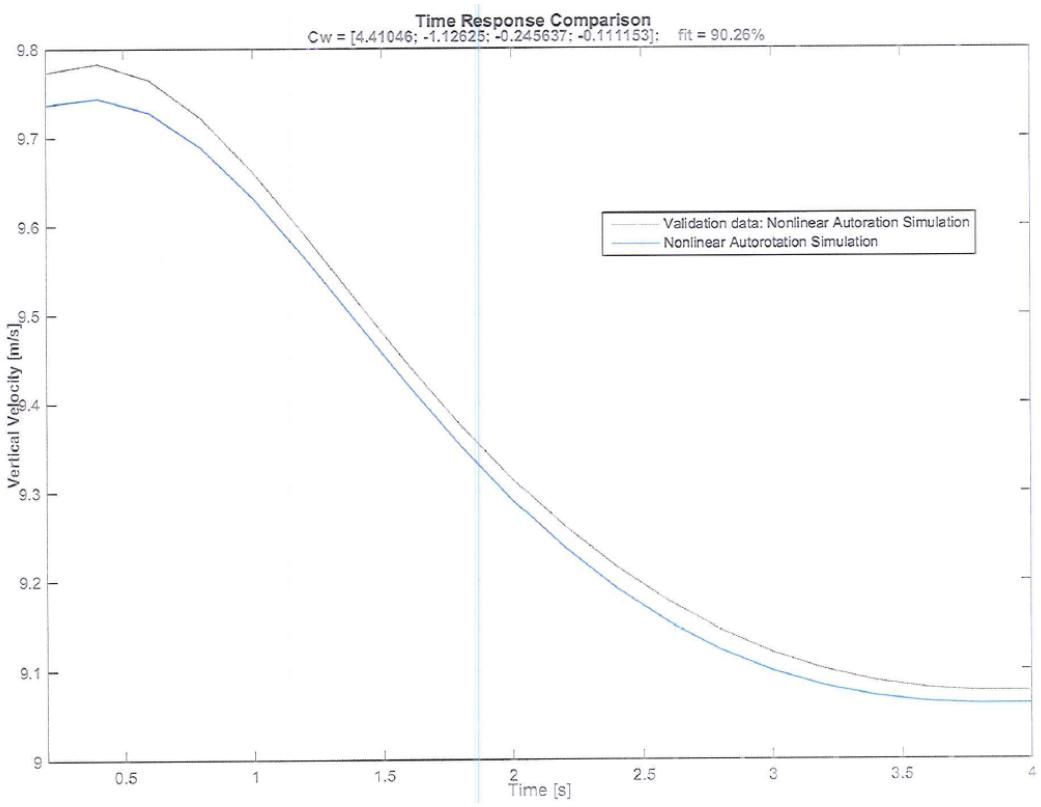


Figure C.12: Time response comparison: vertical velocity

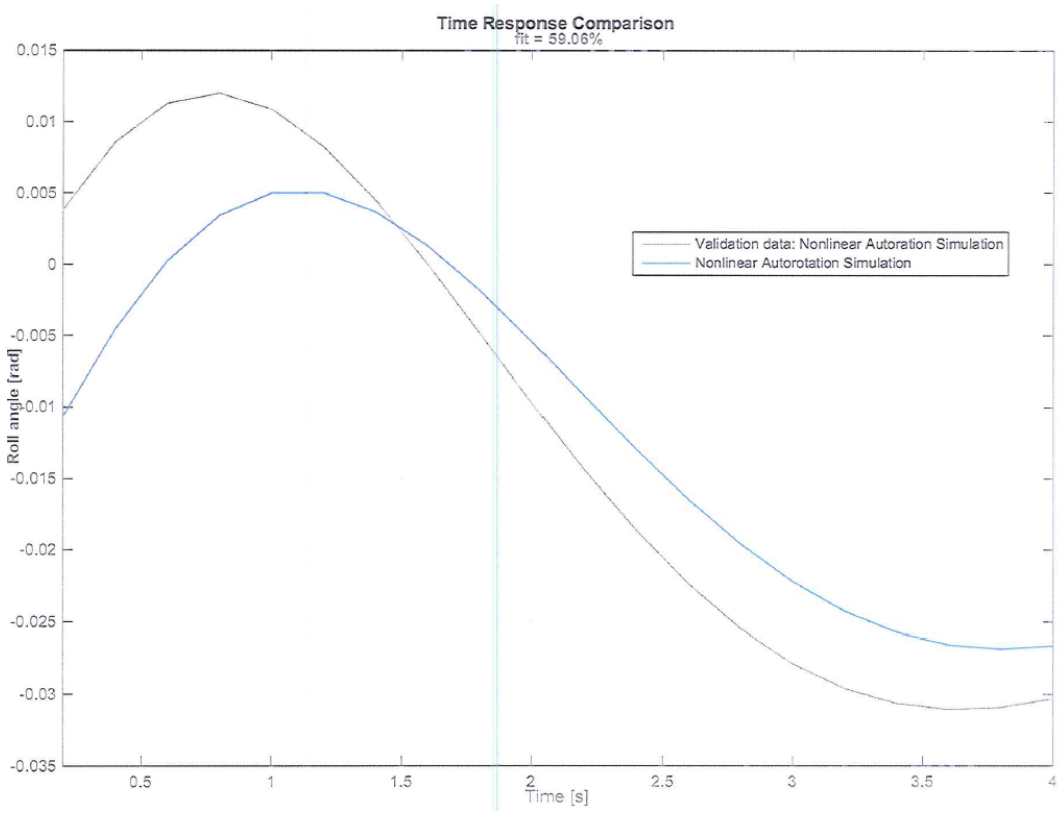


Figure C.13: Time response comparison: roll angle

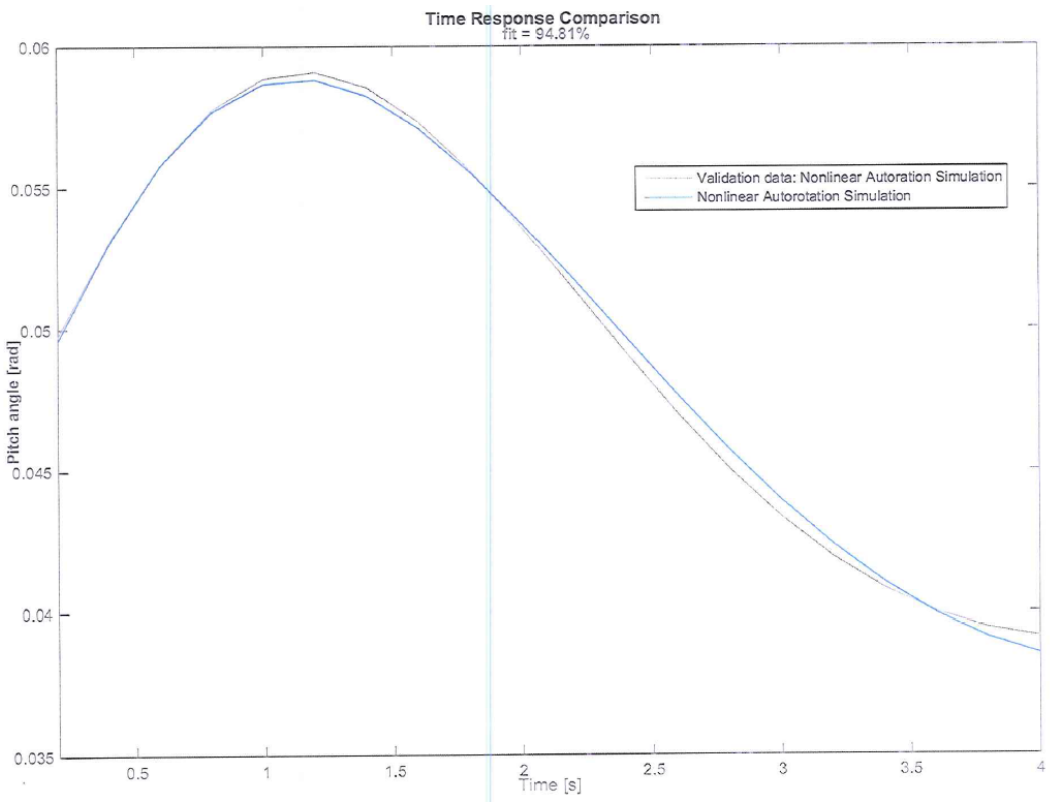


Figure C.14: Time response comparison: pitch angle

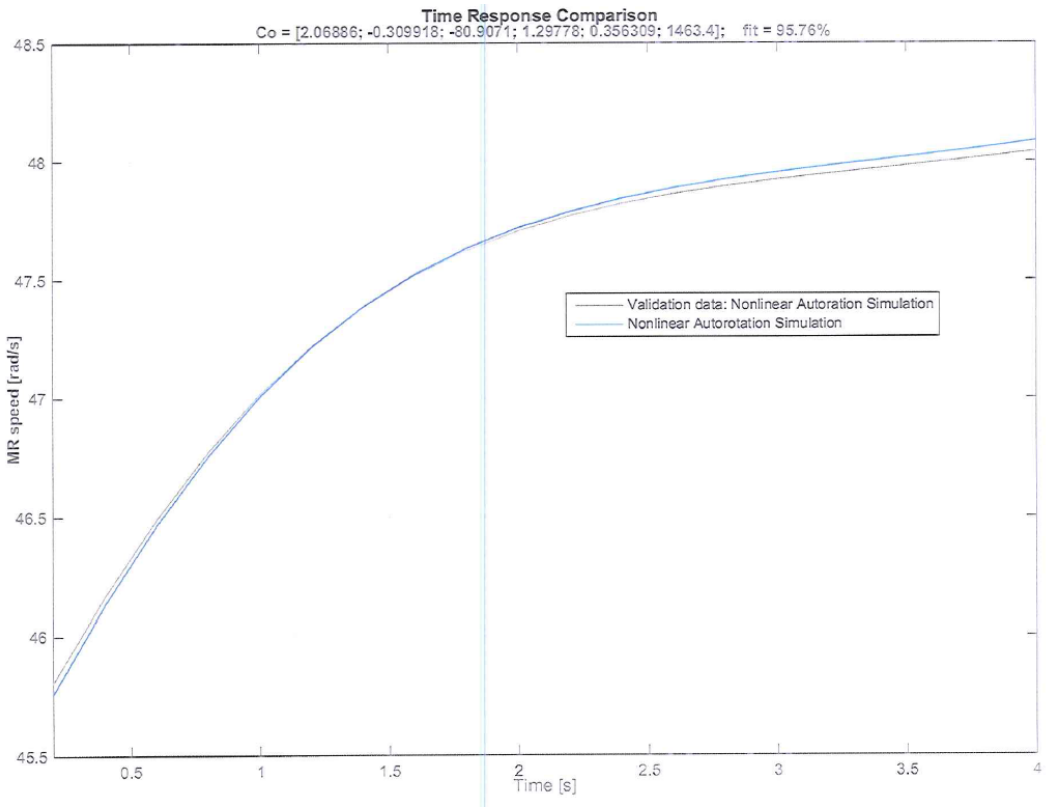


Figure C.15: Time response comparison: main rotor speed

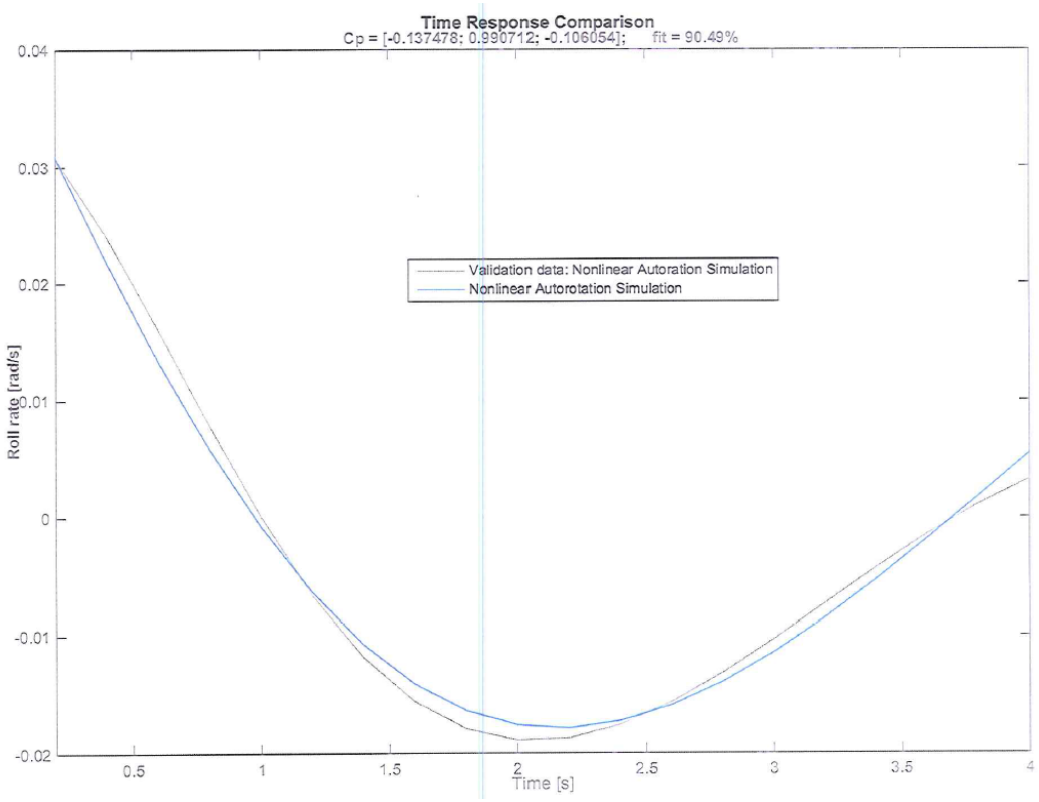


Figure C.16: Time response comparison: roll rate

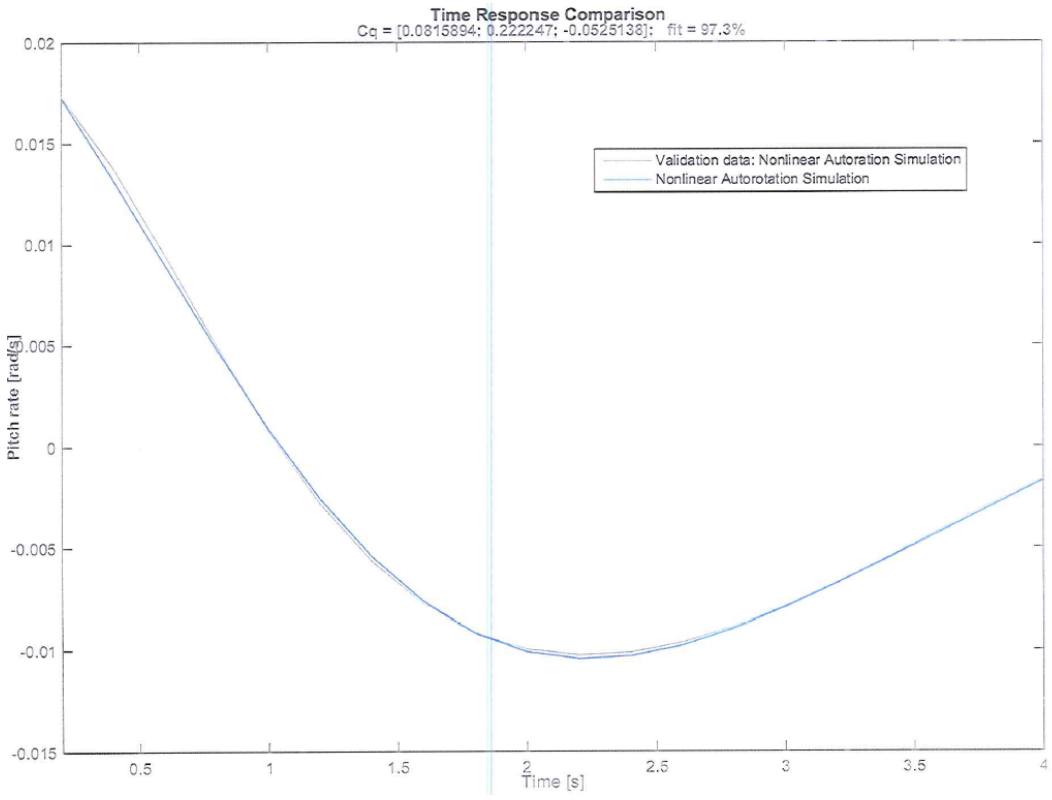


Figure C.17: Time response comparison: pitch rate

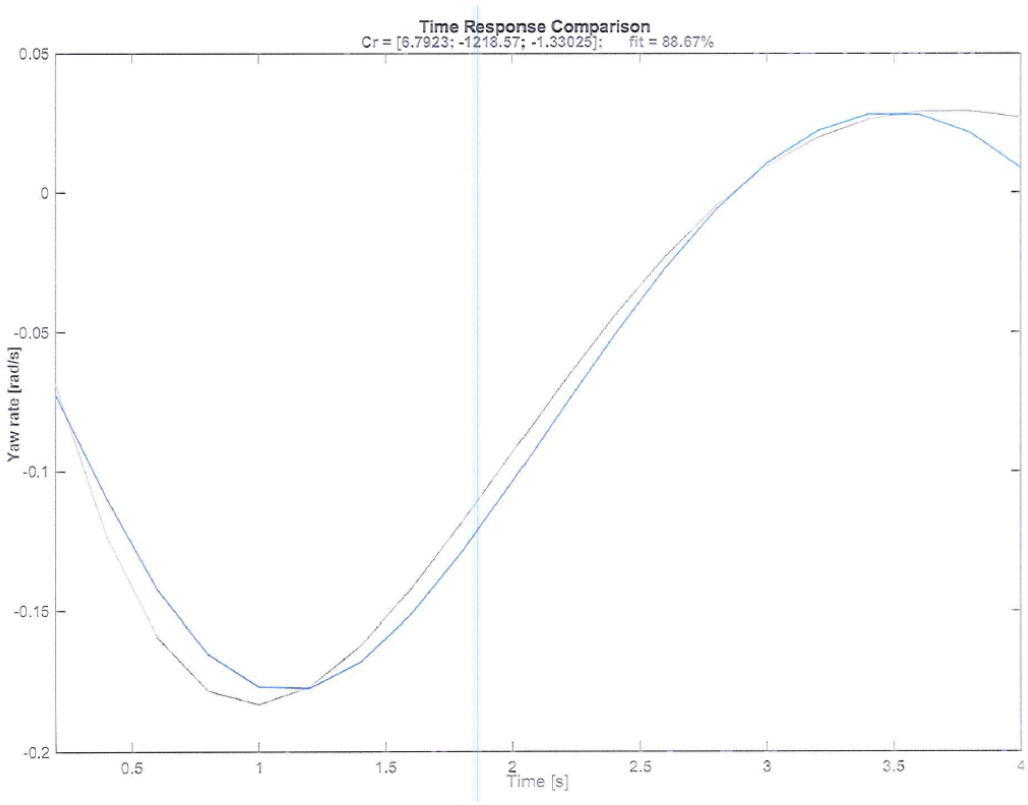


Figure C.18: Time response comparison: yaw rate

Appendix D DDP Control Algorithm

$function[Fmse, u_optimal, delta] = DDDPController(u_trial, elem1, delta_u, u_ideal, Fcrit)$

initialising the varibles, so that runtime is decreased $u_trial1 = zeros(elem1, 1);$

$u_trial2 = zeros(elem1, 1);$

$u_trial3 = zeros(elem1, 1);$

$F_trial1 = zeros(elem1, 1);$

$F_trial2 = zeros(elem1, 1);$

$F_trial3 = zeros(elem1, 1)$; $F_trial = zeros(elem1, 1);$

$F = zeros(elem1, 1);$

Puseletso = 'AWESOME';

Creating the corridor while strcmp(Puseletso,'AWESOME')

$fori = 1 : elem1$

$u_trial1(i) = u_trial(i) + delta_u(1);$

end

$fori = 1 : elem1$

$u_trial2(i) = u_trial(i) + delta_u(2);$

end

$fori = 1 : elem1$

$u_trial3(i) = u_trial(i) + delta_u(3);$

end

calculating the error $for i = 1 : elem1$

$F_trial1(i) = abs(u_ideal(i) - u_trial1(i));$

$F_trial2(i) = abs(u_ideal(i) - u_trial2(i));$

$F_trial3(i) = abs(u_ideal(i) - u_trial3(i));$

end

$for i = 1 : elem1$

$if(F_trial1(i) < F_trial2(i))\&\&(F_trial1(i) < F_trial3(i))$

$F_trial(i) = F_trial1(i);$

$u_trial(i) = u_trial1(i);$

$F(i) = (u_ideal(i) - u_trial1(i))^2;$

end

$if(F_trial2(i) < F_trial1(i))\&\&(F_trial2(i) < F_trial3(i))$

$F_trial(i) = F_trial2(i);$

$u_trial(i) = u_trial2(i);$

$F(i) = (u_ideal(i) - u_trial2(i))^2;$

end

$if(F_trial3(i) < F_trial2(i))\&\&(F_trial3(i) < F_trial1(i))$

$F_trial(i) = F_trial3(i);$

$u_trial(i) = u_trial3(i);$

$F(i) = (u_ideal(i) - u_trial3(i))^2;$

end end

Calculate MSE $Fmse = 0$;

$fori = 1 : elem1$

$Fmse = Fmse + F(i)$;

end

$Fmse = Fmse/elem1$;

$if(Fmse < Fcrit)$

$Puseletso =' EXTREMELY AWESOME'$;

end

$check = Fmse/delta_u(1)$;

$if(check < 1)$

$delta_u(1) = delta_u(1)/2$;

$delta_u(2) = delta_u(2)/2$;

$delta_u(3) = delta_u(3)/2$;

end

end

$u_optimal = u_trial$;

$delta = delta_u$;

end