LINEAR PARAMETER VARYING CONTROL FOR AUTONOMOUS SYSTEMS: METHODS AND APPLICATION EXAMPLES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

FATIH ÇALIŞ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

AUGUST 2022

Approval of the thesis:

LINEAR PARAMETER VARYING CONTROL FOR AUTONOMOUS SYSTEMS: METHODS AND APPLICATION EXAMPLES

submitted by **FATIH ÇALIŞ** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. İlkay Ulusoy
Head of Department, **Electrical and Electronics Engineering** _____

Prof. Dr. Klaus Werner Schmidt
Supervisor, **Electrical and Electronics Engineering, METU** _____

Examining Committee Members:

Prof. Dr. Kemal Leblebicioğlu
Electrical and Electronics Engineering, METU _____

Prof. Dr. Klaus Werner Schmidt
Electrical and Electronics Engineering, METU _____

Prof. Dr. Umut Orguner
Electrical and Electronics Engineering, METU _____

Assist. Prof. Dr. Halil Ersin Söken
Aerospace Engineering, METU _____

Prof. Dr. Çağlar Başlamışlı
Mechanical Engineering, Hacettepe University _____

Date: 24.08.2022

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Fatih Çalış

Signature :

ABSTRACT

LINEAR PARAMETER VARYING CONTROL FOR AUTONOMOUS SYSTEMS: METHODS AND APPLICATION EXAMPLES

Çalış, Fatih
M.S., Department of Electrical and Electronics Engineering
Supervisor: Prof. Dr. Klaus Werner Schmidt

August 2022, 110 pages

Linear parameter varying (LPV) systems are nonlinear systems which can be modelled as linear systems whose parameters change as a function of different "scheduling parameters". In other words, the dynamics of the LPV systems change during the operation hence they require a parameter dependent controller. Although classical gain-scheduling approaches satisfy some performance criteria for constant dynamics, they don't guarantee stability while the scheduling parameter is changing. On the other hand, H∞-norm based LPV control methods utilizing parameter dependent Lyapunov functions provide stability and performance guarantees for the closed-loop system throughout the whole operation. This controller synthesis problem is infinite-dimensional due to the dependency on the scheduling parameter, with the help of polytopic approach it turns into a finite-dimensional convex search with constraints in the form of linear matrix inequalities.

In this thesis, LPV control is applied for lane keeping and a launch vehicle system. LPV system models are derived for both systems based on respective nonlinear models of the lateral vehicle dynamics and a rocket by linearization and selection of a suitable scheduling parameter. LPV controllers are designed using a linear matrix

inequality (LMI) formulation of the stability conditions and performance constraints. The functionality of the designed controllers is validated by extensive high fidelity simulations.

Keywords: Linear parameter varying systems, H2 control, gain scheduling, LMI, autopilot design

ÖZ

OTONOM SİSTEMLERİN DOĞRUSAL PARAMETRE DEĞİŞİMLİ KONTROLÜ: METOTLAR VE UYGULAMALI ÖRNEKLER

Çalış, Fatih
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Klaus Werner Schmidt

Ağustos 2022 , 110 sayfa

LPV sistemler belirli "planlama parametrelerine" bağlı olarak değişen doğrusal sistemler olarak modellenebilen fakat doğrusal olmayan sistemlerdir. Başka bir deyişle, LPV sistemlerin dinamiği operasyon esnasında değişmektedir ve bu nedenden ötürü parametreye bağlı kontrolcülere ihtiyaç duyarlar. Her ne kadar klasik kazanç programlamalı metotlar sabit dinamikli sistemler için bazı performans isterlerini sağlasa da, bu yöntemler sistemin dinamiği değişirken kararlılığı garanti etmezler. Diğer taraftan, parametreye bağlı Lyapunov fonksiyonlar kullanan H∞ tabanlı LPV kontrol metotları, operasyon boyunca kapalı döngü sistem için performans ve kararlılık garantisi verirler. Bu kontrol problemi parametreye bağlı olarak değiştiği için sonsuz boyutludur, politop yaklaşımı ile bu problem doğrusal matris eşitsizliği şeklinde kısıtları olan sonlu boyutlu dışbükey optimizasyon problemine dönüşür.

Bu tezde, şerit takip sistemine ve bir fırlatma aracına LPV kontrol uygulanmıştır. LPV sistem modelleri, doğrusal olmayan roket ve yatay araç dinamiğinin doğrusallaştırılması ve uygun planlama parametreleri seçim işlemleri ile elde edilmiştir. LPV kontrolcüler ise kararlılık koşulları ve performans kısıtlarının doğrusal matris eşitsizlikleri

şeklinde formüle edilmiş halleri kullanılarak tasarlanmıştır. Tasarlanan kontrolcülerin işlevsellikleri yüksek hassasiyetli ve kapsamlı simülasyonlar ile doğrulanmıştır.

Anahtar Kelimeler: Doğrusal Parametre Değişimli sistemler, H2 kontrol, kazanç ayarlama, LMI, otopilot tasarımı

ACKNOWLEDGMENTS

First and foremost, I want to express my deepest gratitude to my supervisor, Prof. Dr. Klaus Werner Schmidt, for his countless hours spent on this thesis as well as his invaluable assistance, guidance, and encouragement throughout my graduate study under his supervision.

I would like to thank my current employer Roketsan A.Ş. for funding this work.

I would also like to express my gratitude to the examining committee for their suggestions and criticisms.

I would like to extend my sincere thanks to my colleagues, Sena Güler for her invaluable assistance in the gravity and 6DoF modelling, Tahir Yanık for his precious reviews and support, Kağan İpek and Korhan Dokumacı for their practical suggestions about aerodynamics modelling and Onur Altın for his contributions on mechanical calculations.

I would also like to give special thanks to Gamze Mert, and my friends from "Autonymous" team, "2010-B-2014" and "D^3" for their friendship and motivational support during this process.

Without the help and support of my sister Şeyda and my parents, I would not have been able to complete this thesis.

to my family...

TABLE OF CONTENTS

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

CG	Center of Gravity
DCM	Direction Cosine Matrix
DoF	Degree of Freedom
ECI	Earth-Centered Inertial
ECEF	Eearh-Centered Earth-Fixed
GNC	Guidance Navigation and Control
IMU	Inertial Measurement Unit
LFR	Latitude Longitude Altitude
LFR	Linear Fractional Representation
LFT	Linear Fractional Transformation
LKA	Lane Keeping Algorithm
LMI	Linear Matrix Inequality
LPV	Linear Parameter Varying
LTV	Linear Time Variant
LTI	Linear Time Invariant
NED	North East Down
SOF	Static Output Feedback
SW	Steering Wheel
TVC	Thrust Vector Control
VISTA	Variable Stability In-flight Simulator Test Aircraft

CHAPTER 1

INTRODUCTION

Systems can be categorized according to different properties, one of which is linearity. Linear systems are easier to analyze and control than their nonlinear counterparts. However, if not all, most of the systems are inherently nonlinear in nature, meaning that they cannot be expressed as a linear system. One way of getting around this issue is the process of linearization. Although it can successfully present the dynamics of the nonlinear system around some operating point, the accuracy of the linearized model drops if the states move away from this point. Another methodology is to describe a nonlinear system such that it is linear, but the dynamics depend on some parameters which can change over time. This system class is called a Linear-Parameter Varying (LPV) system [1].

LPV systems can be regarded as a bridge between linear and nonlinear systems and are useful when the dynamics of a system changes as a function of some parameters. There are various methodologies on the control of these systems. The most straightforward one is classical gain scheduling [2]. More advanced approaches are called LPV control methods including gridding-based [3] [4], polytopic [5] [6] and LFT-LPV synthesis [7] [8]. These methods can be applied to the control of LPV systems such as lane keeping for autonomous cars [9] or autopilot systems for launch vehicles [10].

Lane keeping algorithms are utilized in both completely autonomous [11] and advanced driver assistance systems [12]. The main objective of these algorithms is lateral vehicle control in the sense of keeping the vehicle at the center of the lane despite the existence of disturbances such as a changing road curvature or initial deviations from the lane centerline. There are different control methods used for these applica-

tions such as PID [11] or nested PID control [13], model predictive control [14] [15], H_2 control [16] etc. Hereby, it has to be noted that the lateral dynamics of a vehicle depends on different constant parameters such as tire coefficients as well as varying parameters like the longitudinal velocity [17].

Most of the airborne systems possess 6 degree-of-freedom (DoF) movement with changing dynamics as a function of different parameters. Launch vehicles or rockets can be given as an example for these systems. They are generally aerodynamically unstable, exhibit non-minimum phase characteristics and have relatively slow actuators [18]. Moreover, during the flight, different parameters such as mass, dynamic pressure, velocity, center of gravity (CG) distance varies. As a result, the controller, which is called autopilot, needs to keep the rocket stable against disturbances, ensure some reference tracking performance while taking these parameter changes into account. For this purpose, in some applications linear controllers with frozen time approach [19] are used where the trajectory is divided into multiple parts considering a stationary dynamic between these points. Although this approach provides some performance at the design points, it does not satisfy any stability guarantee in the transition between these points.

The lateral dynamics of a vehicle changes with the longitudinal velocity. Similarly, the launch vehicle has different parameters that the dynamics of it depends on. As a result, the controller for both systems needs to take these changes into account otherwise stability and performance problems will be inevitable. For that purpose, LPV control can be applied to these systems.

In this thesis, firstly, modelling of the lateral dynamics of a vehicle and a launch vehicle is done. Secondly, LPV control is applied to these systems, namely, to a lane keeping problem and autopilot design for a launch vehicle. Finally, to verify the designed controllers, nonlinear simulations are performed. The lane keeping algorithm is tested in diverse scenarios with road curvatures and nonzero initial heading or displacement errors at different velocities, realistic velocity and curvature profiles. The autopilot of the launch vehicle is verified by extensive Monte Carlo simulations under uncertainties in different parameters including aerodynamics, structural parameters, thrust, with disturbances such as wind or delay in the sensors. The novelty of this

work lies within the design procedure of the autopilot algorithm. In the literature, usually gain scheduling or LFT-LPV methodologies are used for LPV control where in this work, the polytopic approach is utilized.

The contribution to the literature can be summarized as follows.

- Development of lane keeping and autopilot models in a unified framework that is suitable for a polytopic LPV controller formulation,
- Application of this controller formulation to the two different system namely lane keeping algorithm and launch vehicle autopilot,
- Verification of the controller with extensive simulations under disturbances.

The remainder of the thesis comprises five chapters. In the second chapter, the preliminary information that will be useful in the following chapters is introduced. The LPV systems and control methods are explained in more detail. The mathematical background along with the reference frame definitions are provided in this chapter. The third chapter is about the modelling of the selected systems namely vehicle in lane keeping problem and launch vehicle dynamics. In this chapter, the nonlinear models for these two systems with the equations of motion, actuator dynamics and error dynamics are constructed. Then, these nonlinear models are transformed into linear models using jacobian linearization. Chapter 4 and 5 are devoted to the LPV controller design and simulations. In these chapters, the linear models obtained in the third chapters are transformed into LPV form and they are used to synthesize an LPV controller. After the controller design, the implementation of the simulations are explained. Then extensive closed loop simulations are executed. For the lane keeping algorithm different scenarios with various velocities, initial conditions and road curvature profiles are tested while the launch vehicle autopilot is verified using the Monte Carlo simulations. In the last chapter, the conclusion summarizing the work done in this thesis with the possible future work are given.

CHAPTER 2

BACKGROUND

In this chapter, linear parameter varying (LPV) systems will be introduced. Their properties along with the similarities and the differences between other system classes will be examined. Example systems belonging to this class will be given. Various methodologies used for the control of LPV systems will be mentioned and compared briefly. The mathematical preliminaries together with the definition of the reference frames will be given.

2.1 Linear Parameter Varying (LPV) Systems

The dynamics of an LPV system can be expressed as the following state-space representation

$$\begin{aligned} \dot{x} &= A(\rho)x + B(\rho)u \\ y &= C(\rho)x \end{aligned} \tag{2.1}$$

where u is the input, y is the output and ρ is the exogenous parameter called "scheduling parameter" that can be time dependent [20]. The time variation of the scheduling parameter ρ is unknown, but it is assumed to be measurable. Another assumption is that ρ is an exogenous parameter, meaning that it does not depend on the states. If this was the case, then this type of system is called quasi-LPV system. On the other hand, if ρ is a function of the time only, then this system becomes Linear Time-Variant (LTV) system. Similarly, if ρ is constant, then it turns into Linear Time-Invariant (LTI) system, which is the easiest class to control. This property makes LPV systems a useful bridge between nonlinear and LTI systems. This relation is depicted in Figure

2.1.

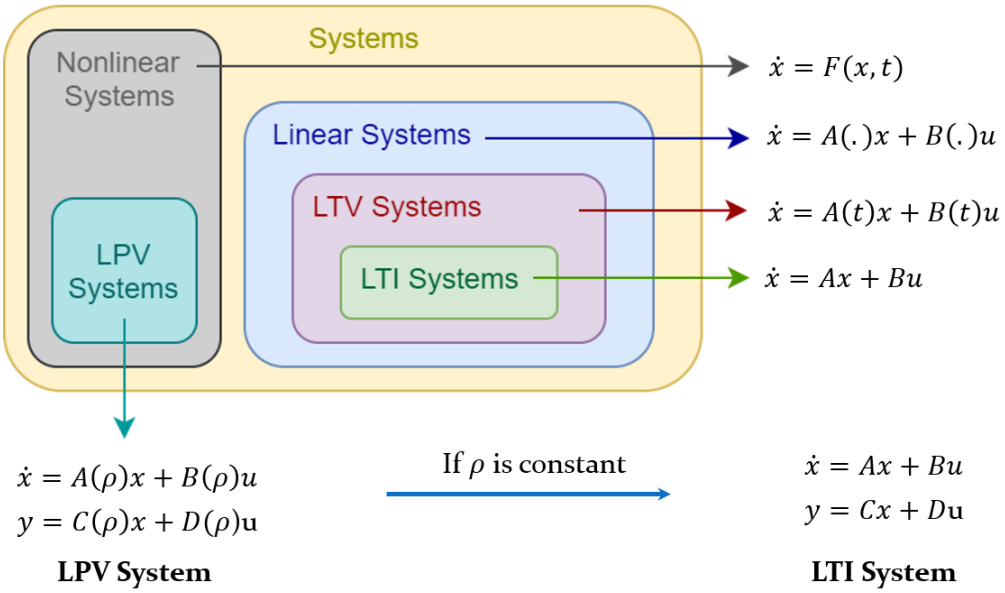

Figure 2.1: The LPV systems and their relations with other classes

The LPV paradigm is introduced for the first time in the PhD thesis of Shamma [1] for the systematic analysis and design of gain-scheduled controllers, which will be mentioned in the following section.

Although 6 DoF equations of motion possess high non-linearity, many aircraft systems can be modelled as an LPV system (with some assumptions) since their dynamics change as a function of various flight parameters such as dynamic pressure, mach number or angle of attack. There are various examples for the aircrafts modelled as an LPV system in the literature, some of them are airplanes such as B737-800 [21], B747-100/200 [22], fighter jets such as F14 [23], F16-VISTA [7] and F18 [24]. LPV modelling is also used in the rocket industry, some examples for the missiles are [25] [26] [27] [8] [28] [29] [30] and launch vehicles are [10] [31] [4] [32].

Another area where the LPV formalization is useful is the ground vehicle control. There are different controller design applications on the autonomous ground vehicles with the assistance of LPV modeling in the literature [33] [34] [35] [6]. Moreover, LPV systems are being used in the driver assistance systems, such as lane keeping systems [5] [9], lane change system [12], vertical controllers [36] [37].

One of the important factors for the modelling and control of an LPV system is to choose the scheduling parameters ρ. For simplification purposes usually the least possible number of parameters are chosen, but if the dynamics of the actual system cannot be captured using a small number of parameters, then the chosen scheduling parameters can be increased. One should note that with the addition of each scheduling parameters, the number of design points increases exponentially, which increases the complexity of the controller process. In the literature, usually one parameter is scheduled, however there are few applications where the number of scheduling parameter can be increased up to seven [38].

2.2 Control Methods for LPV Systems

One of the most straightforward methods is called classical gain scheduling [2]. In this method, the nonlinear system is linearized for different points along the system trajectories with the assumption of the system parameters being "frozen" and LTI controllers are designed for these linear systems. Then, these linear controllers are interpolated during the operation. The linearity property comes with powerful tools for stability and performance analysis such as bode plot, root locus etc. and assures the desired stability and performance criteria for these linear systems obtained at specific trajectory points. However, there is no guarantee for these criteria throughout the whole trajectory since the scheduled parameter can take any value while controllers are designed for some specific values of it. To overcome this weakness, LPV control methods can be utilized.

Most of the LPV controller synthesis techniques are based on a sufficient condition in terms of an infinite-dimensional parameter dependent matrix inequality to analyze stability and performance, e.g. the Bounded Real Lemma [38], which is given in Section 2.3.

The LPV controller synthesis procedure generally follows the steps below [39]

1. Derive a (in general, sufficient) analysis condition for a desired closed-loop property.

2. Evaluate this condition on the closed-loop LPV system
3. Transform the search for control parameters into a convex search.
4. If the convex search is successful, extract controller parameters.

After the first two steps, infinite set of Linear Matrix Inequalities (LMI) defined over the parameter set $\rho \in \{\rho_{min}, \rho_{max}\}$ are obtained. This infinte set of LMIs can be solved via semidefinite programing using the following approaches:

- Gridding based LPV synthesis
- Polytopic LPV synthesis
- LFT LPV synthesis

Gridding approach is the most straightforward of the three, the steps for this method are as follows [40]:

1. Define a grid G for the value set of scheduling parameter ρ
2. Minimize the defined performance level γ subject to the LMI constraints associated with G and the rate limits of parameter ρ
3. Check the constraints with a denser grid.
4. If Step-3 fails, increase the grid density and return Step-2.

One of the advantages of the gridding approach is that the implementation of the controller is computationally inexpensive. [38]. However, with the dimension of scheduling parameter n and grid point number M, the required controller number is M^n and hence it may require a large amount of memory to store the local controller. This method first appeared in [41], refined in [40]. There are different examples for the implementation of this approach on the literature [31] [4] and a MATLAB Toolbox "LPVTools" is created for this purpose [42].

Another approach for the LPV control is called "Polytopic LPV Synthesis", which is the most widely used approach among the three [38]. In this method, the infinite

dimension LMIs are converted to a finite set of LMIs obtained at the vertices of the polytope that ρ spans. The controller is synthesised only at the vertices, and a convex combination of the controllers are used during the operation. Since a controller is designed for each vertex, if the dimension of the scheduling parameter ρ increases by n, then the number of controllers are increased by 2^n, meaning that it is better than the gridding approach in terms of required memory. There are different applications for the polytopic approach on the literature [33] [35] [6] [9] [43]

The final LPV control method is "Linear Fractional Transform (LFT) LPV synthesis". This approach utilizes the LPV interpretation of Linear Fractional Representation (LFR), where the feedback gain is assumed to vary in time as it is a function of the scheduling parameter ρ. Note that LFRs of LPV systems can be seen as a generalization of LFRs of uncertain LTI systems where the feedback gains is assumed to be a constant or time varying uncertainty [44]. This method is based on S-procedure [45] and its variants and extensions, so called full-block S-procedure [46]. There are different examples of applicaions with LFT LPV approach in the literature [7] [8] [26]

2.3 Mathematical Preliminaries

In this section, useful mathematical formulations which will be used in the next chapters are given.

The positive definiteness or positive semi-definiteness of a matrix is an important property that is used in most of the matrix inequalities. The definiton of this property is as follows.

Definition 1 (Positive Definiteness of a Matrix)**.** *A square matrix M is called positive definite (positive semi-definite) if it is symmetric and all eigenvalues are positive (non-negative) and shown as*

$$M > 0 \quad (M \geq 0) \tag{2.2}$$

Linear Matrix Inequalities (LMI) are useful when it comes to formulate an optimization problem. Most of them can be solved using interior-point algorithms [47]. There are various applications of LMIs in control [45] such as Lyapunov stability.

Definition 2 (LMI). *A linear matrix inequality, $G : \mathbb{R}^m \to \mathbb{S}^n$, in the variable $x \in \mathbb{R}^m$ is an expression of the form*

$$F(x) = F_0 + \sum_{i=1}^{m} x_i F_i \leq 0 \tag{2.3}$$

where $x^T = [x_1\ x_2\ ...\ x_m]$ and $F_i \in \mathbb{S}^n, i = 0, ..., m$.

The norm of a signal is also an important property used in the control theory. There are different norm definitions one of the most widely used one is given in the following definition.

Definition 3 (L_2 Norm of a Signal). *The L_2 norm of a square integrable signal $u(t)$ is defined in time and frequency domain as*

$$\|u(t)\|_2 = \left(\int_0^{+\infty} u(t)^2\, dt \right)^{1/2} = \left(\frac{1}{2\pi} \int_{-\infty}^{+\infty} |U(j\omega)|^2\, d\omega \right)^{1/2} \tag{2.4}$$

where $U(j\omega)$ is the Fourier transform of the signal $u(t)$

H_∞ norm is an important parameter showing the maximum possible amplification of a system and defined as follows.

Definition 4 (H_∞ Norm of a System). *The H_∞ norm of a system $G(s)$ is as defined*

$$\|G(s)\|_\infty = \sup_\omega \|G(j\omega)\|_2 = \sup_\omega \overline{\sigma}(G(j\omega)) \tag{2.5}$$

where $\overline{\sigma}$ denotes maximum singular value.

The bounded real lemma which is also referred as Kalman–Popov–Yakubovich (KYP) lemma [48] is a useful expression to calculate H_∞ norm of a system and given below.

Lemma 1 (Bounded Real Lemma). *Consider the system G, state space matrices of which are as follows.*

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t) \\ x(0) &= x_0 \end{aligned} \tag{2.6}$$

Where A is Hurwitz, i.e., every eigenvalue of A has strictly negative real part, then the following are equivalent:

- $||G||_{H_\infty} \leq \gamma$
- *There exists a* $X > 0$ *such that*

$$\begin{bmatrix} A^T X + XA & XB \\ B^T X & -\gamma \end{bmatrix} + 1/\gamma \begin{bmatrix} C^T \\ D^T \end{bmatrix} \begin{bmatrix} C^T & D^T \end{bmatrix} < 0 \tag{2.7}$$

- *There exists a* $X > 0$ *such that*

$$\begin{bmatrix} A^T X + XA & XB & C^T \\ B^T X & -\gamma I & D^T \\ C & D & -\gamma I \end{bmatrix} < 0 \tag{2.8}$$

2.4 Reference Frames

In this section, different reference frames used in the modelling and simulation chapters will be given.

2.4.1 Body Frame

The definition of the body frame for both the car and the rocket is the same. It has its origin at the CG of the vehicle. Its x axis points forward towards the nose, while y and z axes points to right hand side and down side respectively. This frame is depicted for the rocket in the figure 2.2.

2.4.2 Earth-Centered Earth-Fixed Frame

Earth-centered earth-fixed (ECEF) coordinate frame is fixed to the Earth and moves with it. It is defined as follows:

1. The origin is located at the center of mass of the Earth.
2. z axis points towards the north pole of the Earth.
3. x axis points towards the point where the equator and the Greenwhich meridian intersect.

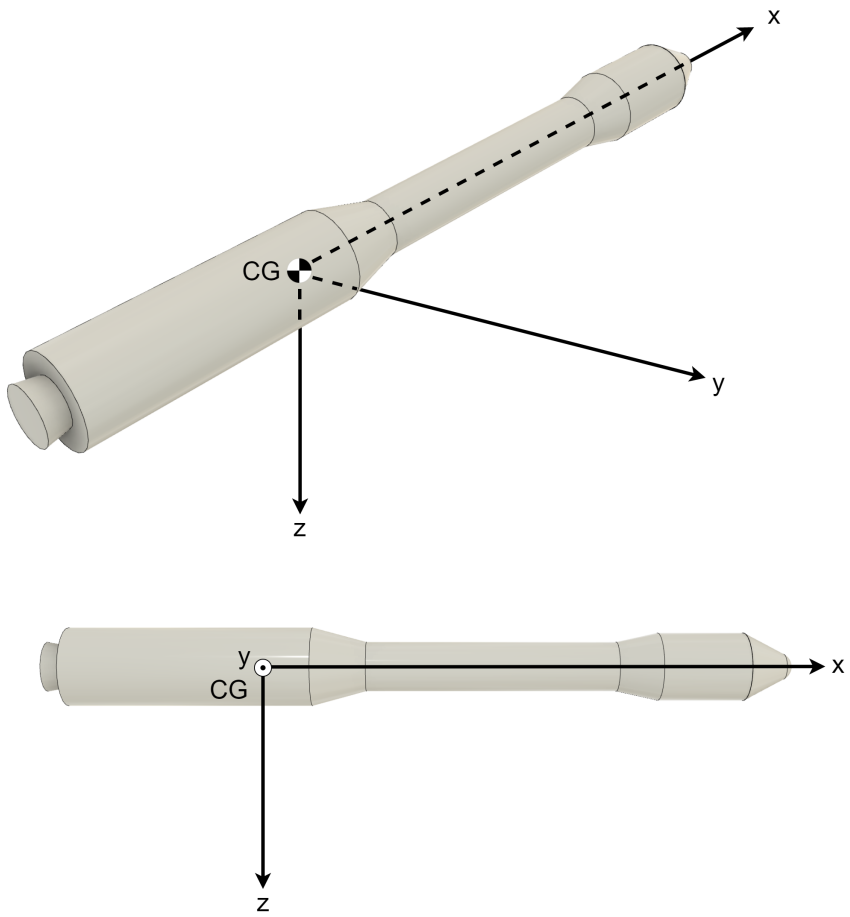

Figure 2.2: Body frame drawing on the isometric view (top) and on the right view of the rocket

4. y axis is chosen such that it completes the orthogonal coordinate system by complying the right hand rule.

The axes of the ECEF frame is depicted in the figure 2.3.The ECEF frame is useful to express the location of a point both inside or outside the Earth. It can be utilized to express the cartesian coordinates of a point in terms of x_{ECEF}, y_{ECEF} and z_{ECEF}. Similarly, the position of a point can be described by spherical coordinates using the latitude Φ, longitude λ and altitude (LLA). ECEF coordinates and LLA coordinates can be converted into one another.

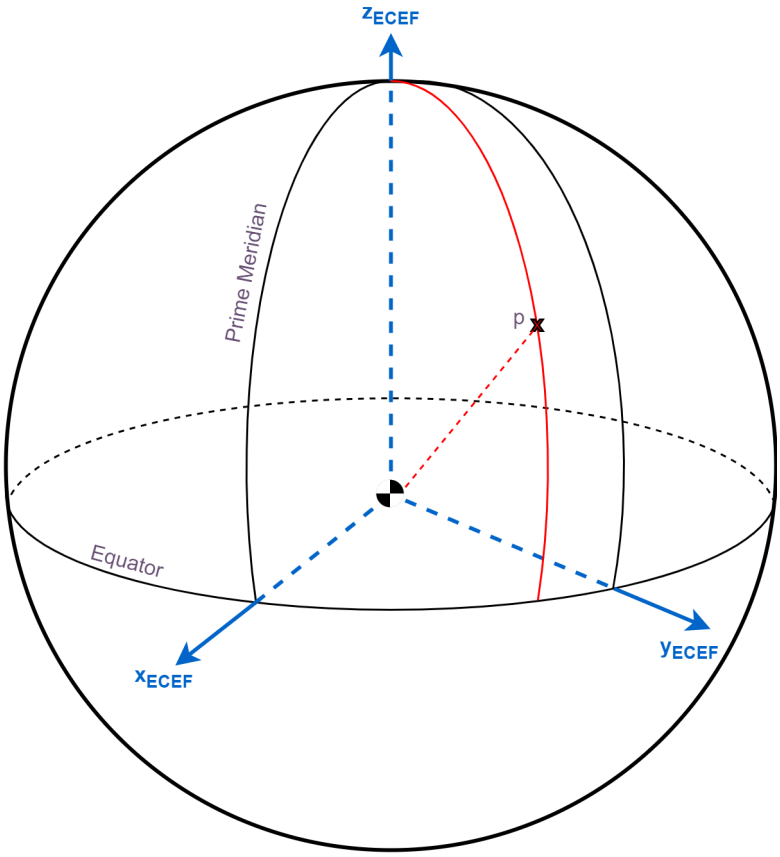

Figure 2.3: ECEF frame definition

2.4.3 North East Down Frame

North East Down (NED) frame is generally used to describe the attitude of an aircraft with respect to the Earth. It is a local frame whose origin is located at the CG of the aircraft. The axes of this frame is obtained as follows. First, the tangent plane to the earth whose normal vector pass through the CG point is drawn. The x axis lies in the tangent plane pointing towards north, y axis also lies in the tangent plane pointing towards east and z axis is normal to the tangent plane and points down.

Note that although z axis of the NED frame points down, it does not necessarily pass through the center of the Earth, i.e., it may not be coincident with the gravity vector due to the oblateness of the Earth. The axes of the NED frame together with the ECEF frame are given in the following figure.

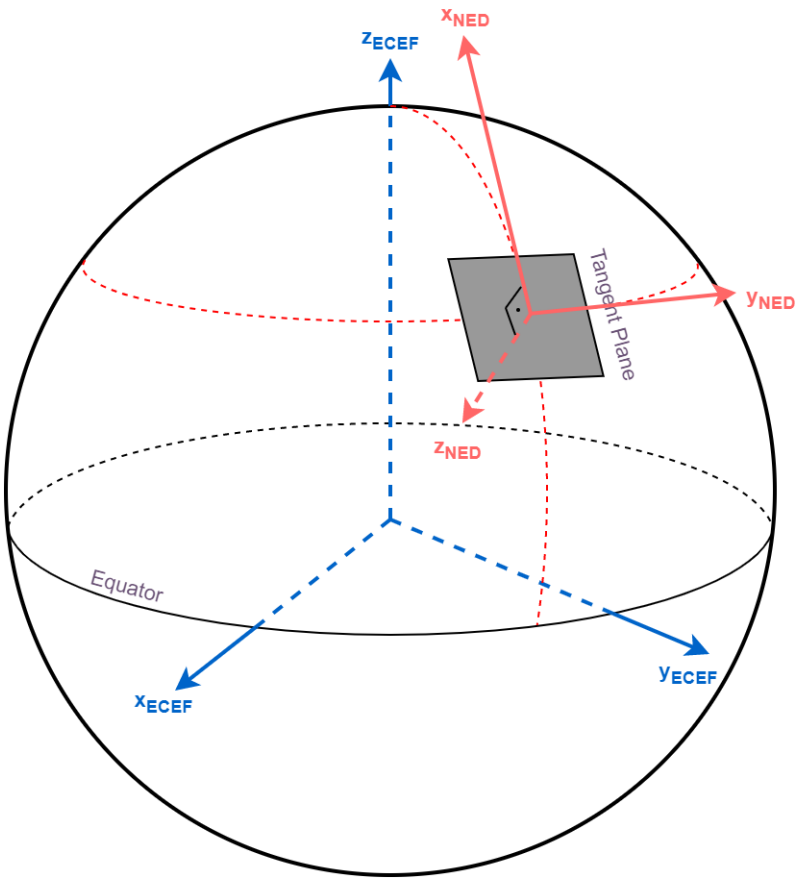

Figure 2.4: NED frame definition

2.4.4 Earth-Centered Inertial Frame

The origin of the Earth-Centered Inertial (ECI) frame is located at the center of mass of the Earth like ECEF frame. However, unlike the ECEF frame, ECI frame does not rotate with the Earth. Its axes are defined as follows.

1. x axis points towards the vernal equinox.
2. z axis points towards the north pole of the Earth.
3. y axis is chosen such that it completes the orthogonal coordinate system by complying the right hand rule.

The axes of the ECI frame along with the ECEF frame are given in the following figure.

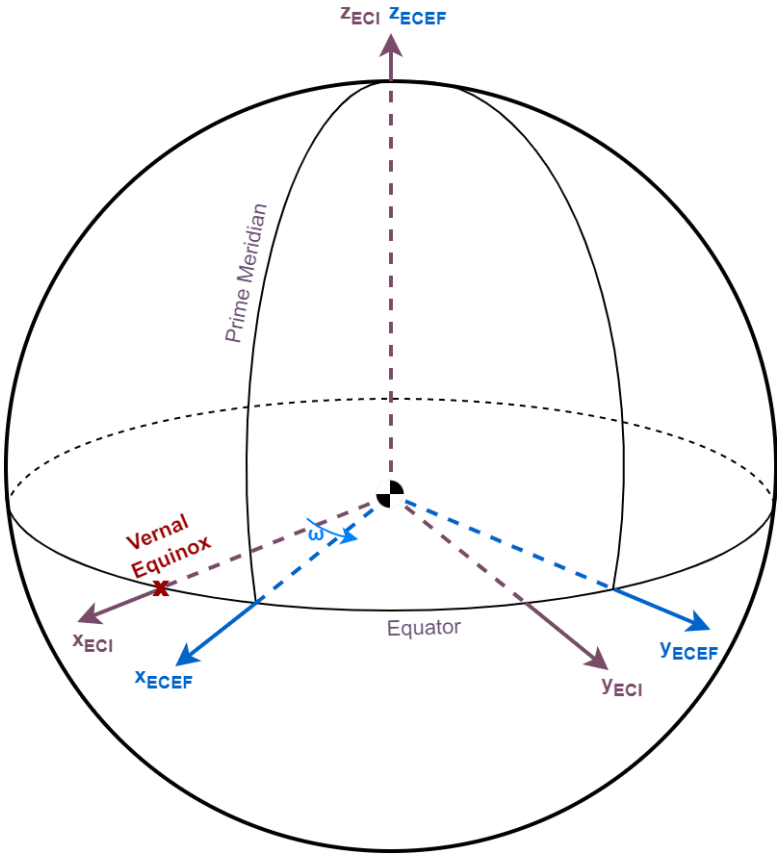

Figure 2.5: ECI frame definition

CHAPTER 3

MODELLING AND SIMULATION OF THE SELECTED SYSTEMS

In this chapter, nonlinear and linear models of the selected systems namely, a ground vehicle and a launch vehicle will be obtained. Although these systems seem very different, they have the following common property: their dynamics change as a function of their velocity. Which implies that these systems are a good candidate for LPV modelling and control. The linear models obtained in this chapter will be used in the controller design procedure and nonlinear models will be used for thorough simulations to validate the controller.

First, nonlinear model of the lateral dynamics of a vehicle will be obtained. By Jacobian linearization of this model, linear model will be acquired. After the vehicle, the launch vehicle will be modelled following the same procedure.

3.1 Vehicle Lateral Dynamic Model

In the lane keeping problem, the motion of the vehicle in the yaw axis is aimed to be controlled. For this reason, the lateral dynamics of the vehicle will be modelled. Since the longitudinal velocity also affects the behaviour in the lateral axis, it needs to be taken into account.

There are various vehicle models, considering motions in different numbers of degree-of-freedom in the literature [49]. The simplest is the two DoF model, which represent the lateral velocity and yaw motion. This model does not capture the dependency of the longitudinal velocity, hence it does not suit the LPV approach. A 3 DoF model adds the acceleration in the longitudinal axis to the 2 DoF model and can describe

the full vehicle motion in the X-Y plane. There are higher order models considering the slip angles of each tires, enabling an in-depth study of traction and braking forces on handling maneuvers [49]. In this study, the decoupled dynamic bicycle model obtained from [50] will be used to model the vehicle.

3.1.1 Nonlinear Vehicle Lateral Dynamic Model

To obtain the nonlinear lateral vehicle model, first the dynamics of the vehicle is to be analyzed. Then, the error dynamics and lastly actuator dynamics will be given. The parameters of the model is depicted in the figure 3.1 [50].

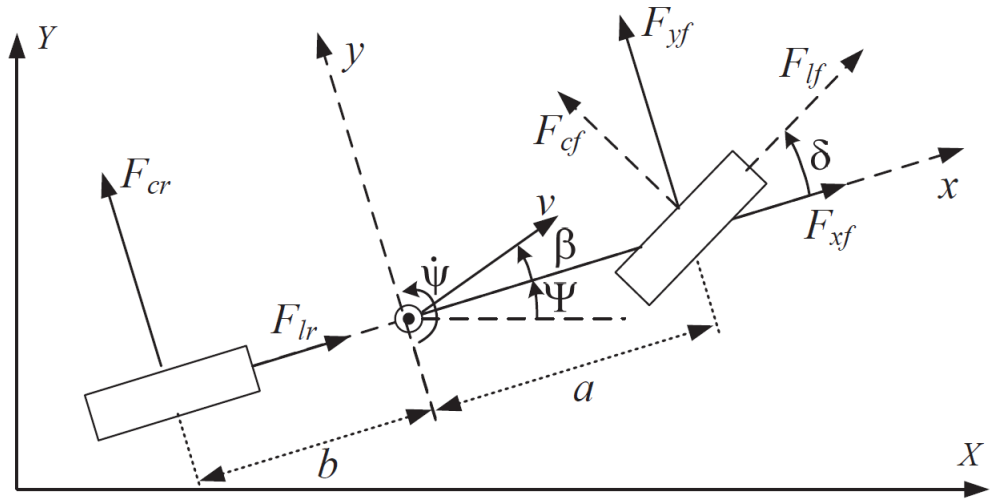

Figure 3.1: Dynamic bicycle model and its parameters

3.1.1.1 Vehicle Dynamics

Starting from the dynamic bicycle model given in the figure 3.1, where the lateral and longitudinal forces of the tires are taken into account, one can obtain the lateral motion dynamics of the decoupled dynamic bicycle model given below following the steps from [50]. Note that the decoupled model is obtained using the following assumptions: Longitudinal force acting on the rear tire is zero, $F_{cr} = 0$ and the engine traction force F_{lf} is chosen such that the desired acceleration is obtained.

Table 3.1: Parameters used in the the dynamic bicycle model

Parameter Name	**Definition**
m	Mass
V	Longitudinal velocity
a	Distance between center of gravity (CG) and front wheel
b	Distance between CG and rear wheel
I_{zz}	Inertia of the vehicle around body z axis
F_{cf}	Cornering force of the front wheel
F_{cr}	Cornering force of the rear wheel

$$\dot{\beta} = \frac{\cos(\beta)}{mV}\left(F_{cr} + F_{cf}\cos(\delta_t) + \frac{ma_r - F_{cf}\sin(\beta - \delta_t)}{\cos(\beta - \delta_t)}\right) - \dot{\Psi}\cos^2(\beta) \tag{3.1}$$

$$\ddot{\psi} = \frac{a(ma_r + F_{cf}\cos(\beta) - F_{cr}\sin(\beta)\sin(\delta_t)) - bF_{cr}\cos(\beta - \delta_t)}{I_{zz}\cos(\beta - \delta_t)} \tag{3.2}$$

In these equations, the β is the side-slip angle which is defined as the angle between the velocity vector and body x vector, where ψ stands for the heading angle in body frame. Moreover, δ_t is the steering angle of the front tires. The other parameters used in the above equations are defined in the table 3.1 and depicted in the figure 3.1.

The cornering forces at the tires are calculated using the Pacejka's magic formula as follows.

$$F_{cf} = D\sin(C\tan^{-1}(B\alpha_f - E(B\alpha_f - \tan^{-1}(B\alpha_f)))) \tag{3.3}$$

$$F_{cr} = D\sin(C\tan^{-1}(B\alpha_r - E(B\alpha_r - \tan^{-1}(B\alpha_r)))) \tag{3.4}$$

Where the parameters α_f and α_r are the slip angle of the front and rear tires respectively and B,C,D are the tire coefficients. The slip angles can be calculated using the following relations between the linear and angular velocities of the vehicle.

$$\alpha_f = tan^{-1}\left(\frac{\dot{y} + a\dot{\psi}}{\dot{x}}\right) - \delta_t \tag{3.5}$$

$$\alpha_r = tan^{-1}\left(\frac{\dot{y} - b\dot{\psi}}{\dot{x}}\right) \tag{3.6}$$

Note that velocities along the body axis can be rewritten using relation between the side slip angle and velocity as follows.

$$\dot{x} = Vcos(\beta) \tag{3.7}$$

$$\dot{y} = Vsin(\beta) \tag{3.8}$$

With all the equations given above combined, one can obtain the complete nonlinear vehicle model with the states β, $\dot{\Psi}$, Ψ, V and input δ.

3.1.1.2 Error Dynamics

With the model of the vehicle completed, the next objective is to obtain road curvature model. Assuming to have an access to the lateral distance from vehicle CG to the road centerline and road curvature information, one can integrate the road curvature into the model in the heading error $\Delta\psi$ and lateral displacement error y_R [51] form using the following relations,

$$\Delta\dot{\psi} = \dot{\psi} - V\rho \tag{3.9}$$

$$\ddot{y_R} = V(\dot{\beta} + \dot{\psi}) - V^2\rho \tag{3.10}$$

where ρ is the road curvature.

3.1.1.3 Actuator Dynamics

There is a dynamic relation between the steering wheel angle δ_{sw} and the actual tire steering angle δ_t. This dynamic is assumed to be of 2nd order [9] as follows:

$$\frac{\delta_t(s)}{\delta_{sw}(s)} = \frac{n_s\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2} \tag{3.11}$$

With this actuator, the states for tire steering actuators become:

$$\ddot{\delta}_t = -2\xi\omega_n\dot{\delta}_t - \omega_n^2\delta_t + n_s\omega_n^2\delta_{sw} \tag{3.12}$$

where ω_n is the bandwidth of the actuator, ξ is the damping ratio and n_s is the gear ratio. Combining equations from 3.1 to 3.12, one can obtain the complete vehicle-road model.

3.1.2 Linear Vehicle Lateral Dynamic Model

Since the side-slip angle β can be expressed in terms of other states, it can be substituted. After that, this model can be linearized using the states $\dot{\psi}$, ψ, $\dot{y_R}$, y_R, $\Delta\psi$, $\dot{\delta}_t$ and δ_t and control input δ_{sw} and disturbance ρ around the operating point

$$x = \begin{bmatrix} \dot{\psi} \\ \psi \\ \dot{y_R} \\ y_R \\ \Delta\psi \\ \dot{\delta}_t \\ \delta_t \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad u = \delta_{sw} = 0, \quad \rho = 0, \quad V = V_{eq}, \quad \dot{V} = 0 \tag{3.13}$$

After the linearization, and defining the tire coefficient as $c_t = BCD$, following state equations for the complete road-vehicle model are obtained.

$$\ddot{\psi} = \frac{c_t(a^2 + b^2)}{V_{eq}I_{zz}}\dot{\psi} + \frac{c_t(a-b)}{I_{zz}V_{eq}}\dot{y_R} - \frac{c_t(a-b)}{I_{zz}}\Delta\psi - \frac{c_t a}{I_{zz}}\delta_t \tag{3.14}$$

$$\ddot{y_R} = \frac{c_t(a-b)}{mV_{eq}}\dot{\psi} + \frac{2c_t}{mV_{eq}}\dot{y_R} - \frac{2c_t}{m}\Delta\psi - \frac{c_t}{m}\delta_t - V_{eq}^2\rho \tag{3.15}$$

$$\Delta\dot{\psi} = \dot{\psi} - V_{eq}\rho \tag{3.16}$$

$$\ddot{\delta}_t = -2\xi\omega_n\dot{\delta}_t - \omega_n^2\delta_t + n_s\omega_n^2\delta_{sw} \tag{3.17}$$

The linear state equations can be converted into the state space form $\dot{x} = A_p x + B_{pu}u + B_{pw}w$ with the following state space matrices.

$$A_p = \begin{bmatrix} \frac{c_t(a^2+b^2)}{V_{eq}I_{zz}} & 0 & \frac{c_t(a-b)}{I_{zz}V_{eq}} & 0 & -\frac{c_t(a-b)}{I_{zz}} & 0 & -\frac{c_t a}{I_{zz}} \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{c_t(a-b)}{mV_{eq}} & 0 & \frac{2c_t}{mV_{eq}} & 0 & -\frac{2c_t}{m} & 0 & -\frac{c_t}{m} \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -2\xi\omega_n & -\omega_n^2 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.18}$$

$$B_{pu} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ n_s\omega_n^2 \\ 0 \end{bmatrix}, \quad B_{pw} = \begin{bmatrix} 0 \\ 0 \\ V_{eq}^2 \\ 0 \\ -V_{eq} \\ 0 \\ 0 \end{bmatrix} \tag{3.19}$$

3.2 Launch Vehicle Model

The launch vehicles can move in and rotate around 3 directions, resulting in a 6 degree of freedom motion. In this work, a controller (autopilot) for pitch axis will be designed and due to the symmetry of the rocket around its body x axis, the same controller will be used in the yaw axis. The vehicle model is assumed to be controlled in roll axis by other means. Hence, in this chapter, the dynamics of a launch vehicle in the pitch plane will be introduced. For the formulation, although some steps can be skipped, mainly the steps from [52] will be followed.

For the model parameters, small-lift launch vehicle VEGA is chosen since its properties can be found in the literature. It is an expendable launch system jointly designed and manufactured by Italian Space Agency (ASI) and the European Space Agency (ESA). It is designed to launch small payloads, 300 to 2500 kg, satellites for scientific and Earth observation missions to polar and low Earth orbits. The first Vega mission is conducted in 2012 [53]. It is a four stage rocket with 30m height and 3m diameter and shown in 3.2 [54]. Most of the model parameters for Vega is acquired from its user manual [54].

3.2.1 Nonlinear Launch Vehicle Model

To obtain the dynamics of the launch vehicle in the pitch plane, first aerodynamic coefficients in this axis will be obtained using DATCOM. Using these coefficients, translational dynamics will be examined. Then, rotational dynamics will be acquired.

After 6 DoF equations are derived, then actuator and error dynamics will be given. The dynamic behaviours such as bending and sloshing will not be included in the models.

To clarify the difference between scalar and vector parameters, bold symbols are used to describe vector parameters. The unit vectors $\widehat{i}$, $\widehat{j}$ and $\widehat{k}$ are used to describe the direction of x, y and z axes of the related reference frame respectively. The linear velocities u, v and w with the angular velocities p, q and r are given in body frame, while the Euler angles are given in NED frame.

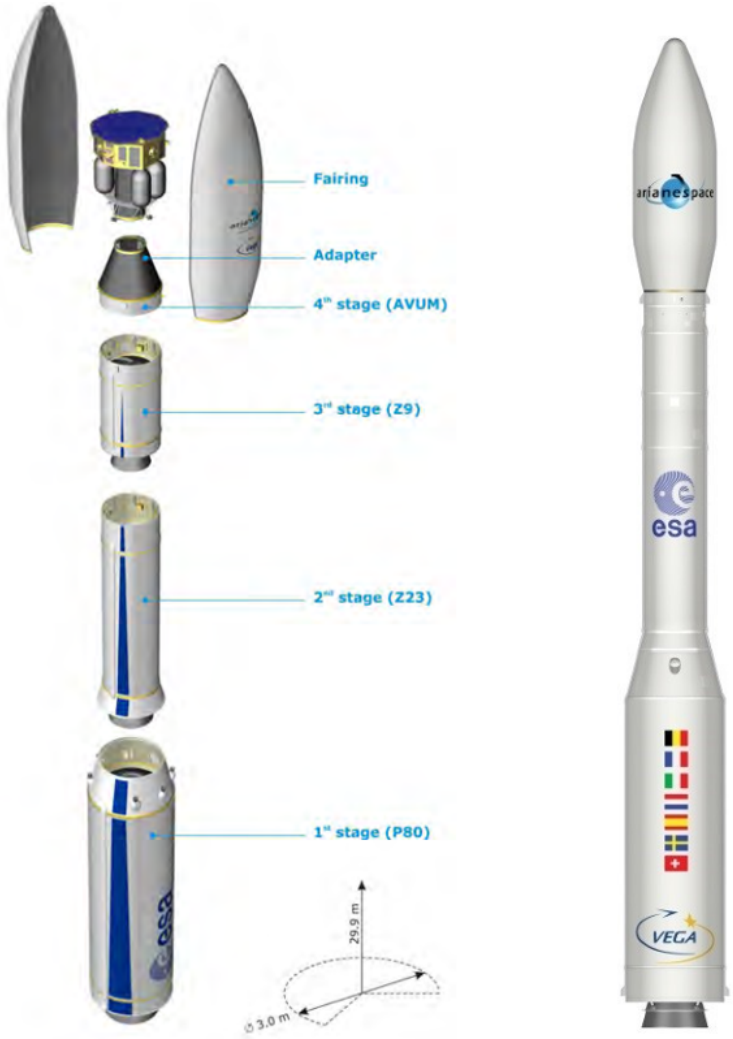

Figure 3.2: VEGA rocket and its stages

3.2.1.1 Aerodynamics

The aerodynamic properties of a rocket can be modelled in terms of force and moment coefficients around different axes. These coefficients are used to calculate the exerted moment and forces on the rockets using the following equations.

$$\begin{aligned} F_i &= q_\infty S_{ref} C_{F_i} \\ M_i &= q_\infty S_{ref} l_{ref} C_{M_i} \end{aligned} \tag{3.20}$$

Where q_∞ is dynamic pressure, S_{ref} and l_{ref} are reference area and length, C_{F_i} and C_{M_i} are the aerodynamics force and moment coefficients around the axis-i respec-

tively.

These coefficients depend on various parameters such as angle of attack, Mach number, side-slip angle, roll angle, and the derivative of these parameters [55]. However, in most of the applications these coefficients can be simplified such that they only depend on static parameters. In this work, they are assumed to depend on only Mach number and angle-of-attack. These coefficients are calculated using the software Missile DATCOM with arbitrary geometry input mode. In this mode, user can enter the rocket geometry in the form of a length-from-nose and radius [56]. For the Vega rocket, this data is obtained in MATLAB as shown in the following figure.

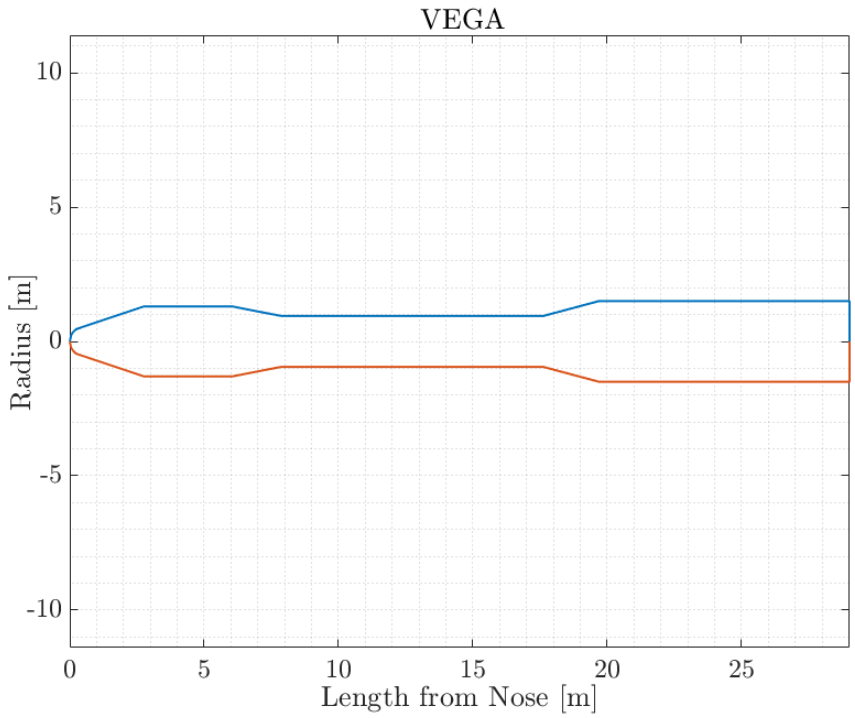

Figure 3.3: Body geometry of VEGA (with all 4 stages) launch vehicle for aerodynamic calculations

Using the geometry data given in 3.3, the moment and force coefficients for Mach values between 0.3 and 10 and AOA values between $-9°$ and $9°$ are obtained. These coefficients are depicted in Figures 3.4, 3.5, 3.6 and 3.7. The graphs are given in 2D to improve the visibility.

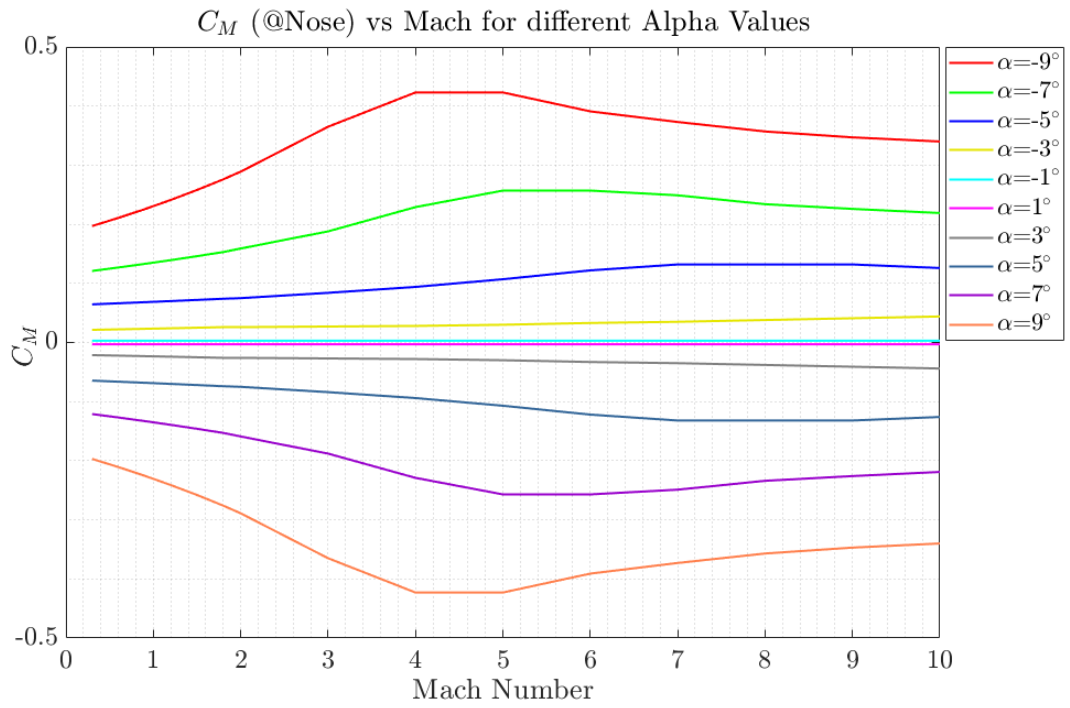

Figure 3.4: Moment coefficient C_M (at nose) vs Mach for different angle of attack values

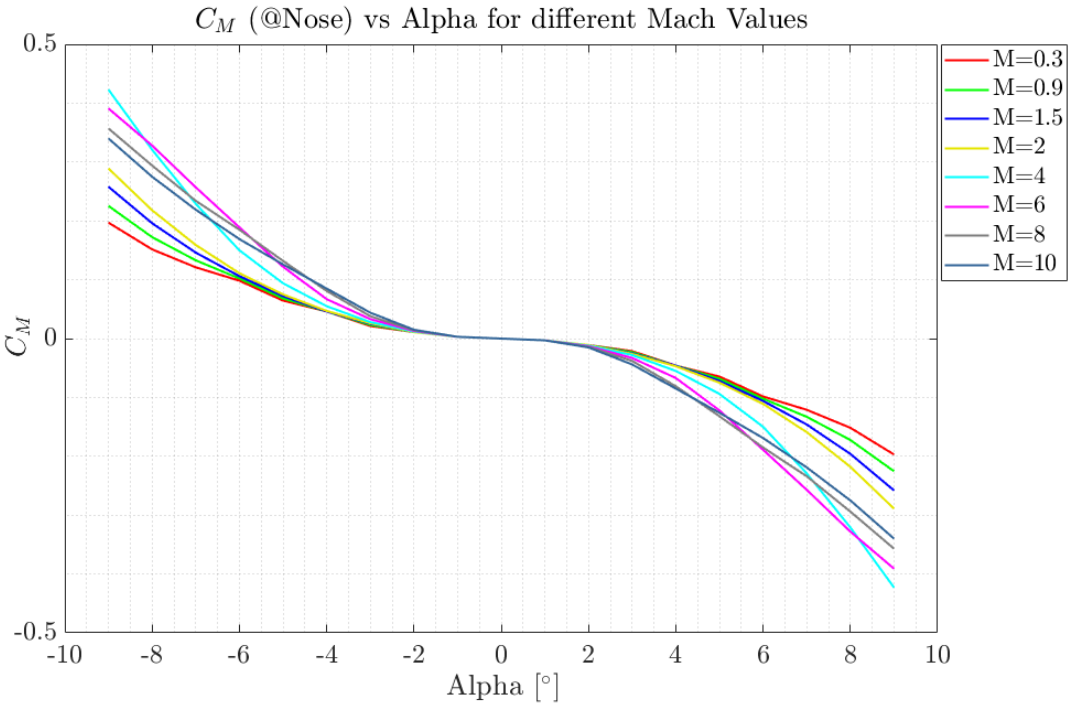

Figure 3.5: Moment coefficient C_M (at nose) vs Alpha graph of the rocket

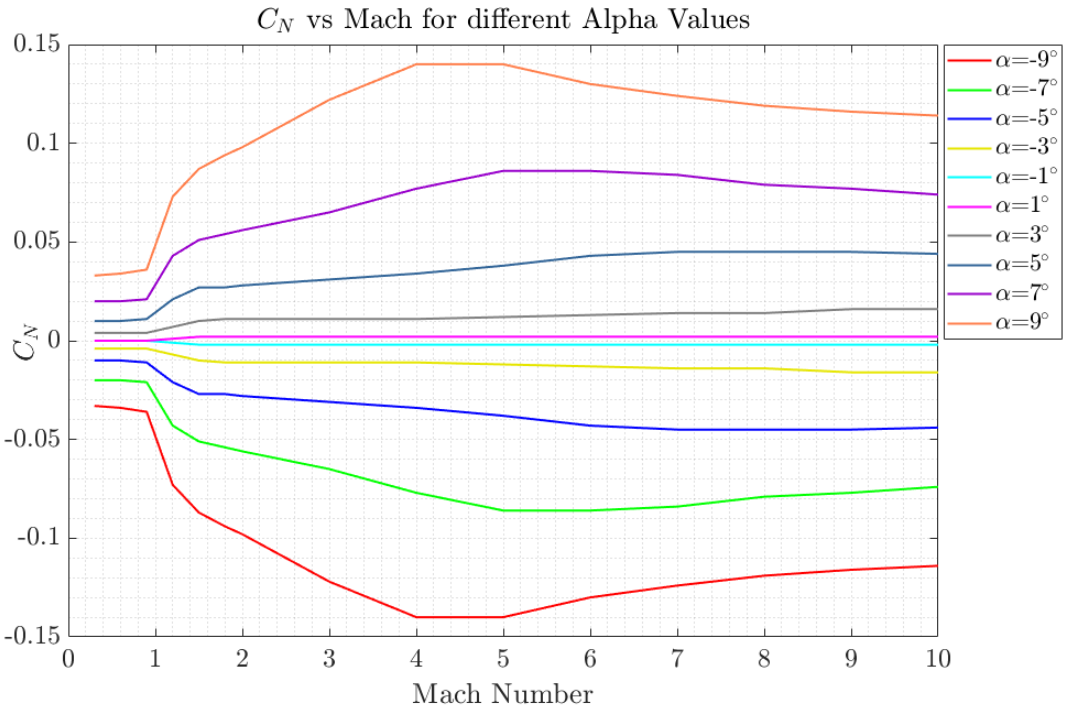

Figure 3.6: Force coefficient C_N vs Mach graph of the rocket

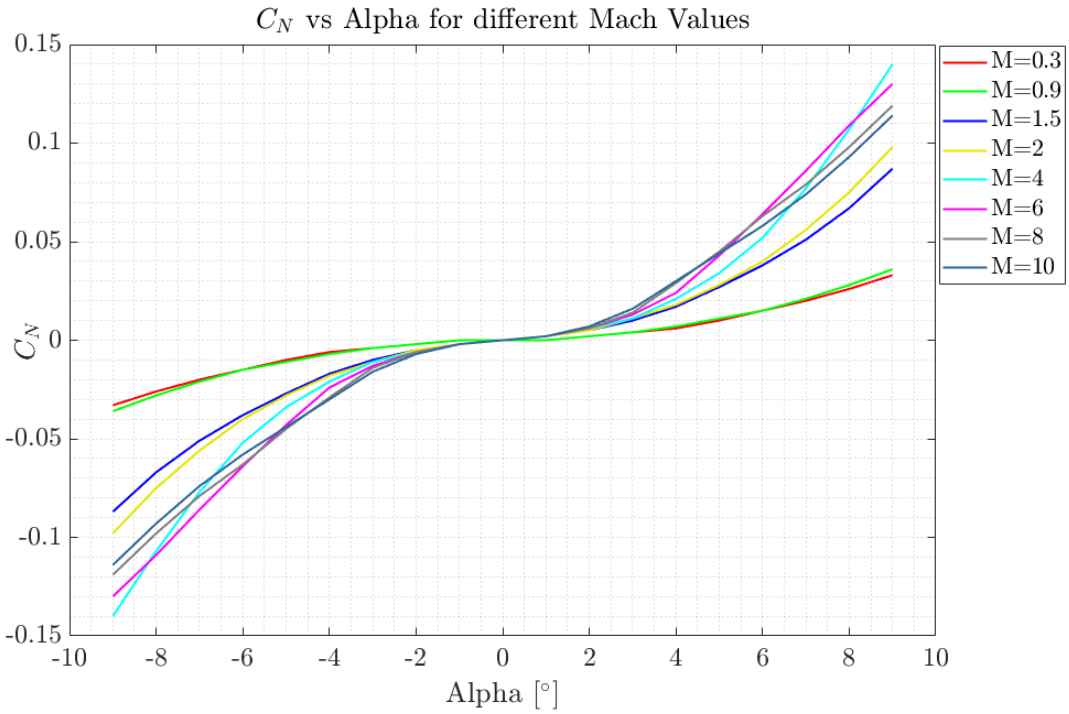

Figure 3.7: Force coefficient C_N vs Alpha graph of the rocket

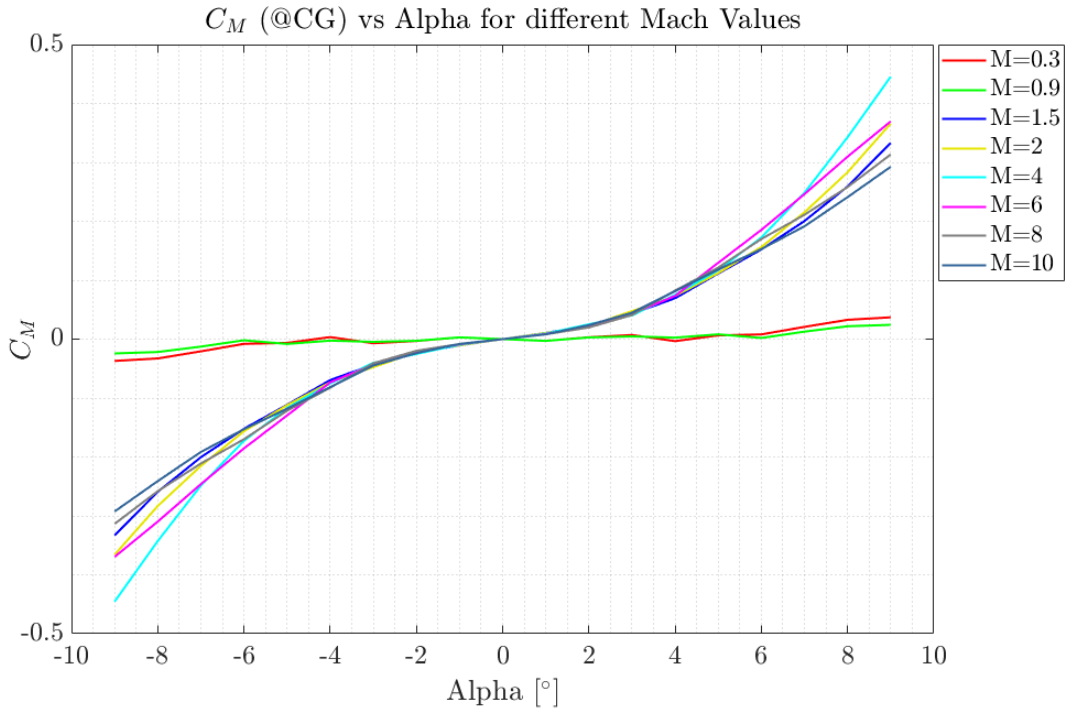

Figure 3.8: Moment coefficient C_M (at CG) vs angle of attack α graph of the rocket with all 4 stages

Using the moment and force coefficients obtained for the nose of the rocket, the moment coefficient at the center of the gravity can be calculated. This data can be seen in Figure 3.8. As seen in this graph, C_{M_α} which is the derivative of the moment coefficient C_M with respect to angle of attack α is positive, hence this rocket is statically unstable [55].

3.2.1.2 Translational Dynamics

From the Newton's 2nd law of the motion, the force equation for a body with infinitesmall mass dm, linear velocity $\boldsymbol{V} = u\hat{i} + v\hat{j} + w\hat{k}$ and angular velocity $\boldsymbol{\omega} = p\hat{i} + q\hat{j} + r\hat{k}$, can be written in the form:

$$\mathbf{F} = m \left.\frac{d\mathbf{V}}{dt}\right|_{body} + m(\boldsymbol{\omega} \times \mathbf{V}) \tag{3.21}$$

Which results in the following 3 equations:

$$\begin{aligned} \sum \Delta F_x &= m(\dot{u} + wq - vr) \\ \sum \Delta F_y &= m(\dot{v} + ur - wp) \\ \sum \Delta F_z &= m(\dot{w} + vp - uq) \end{aligned} \tag{3.22}$$

Writing the forces acting on the rocket in 3 different body axis:

$$\begin{aligned} \boldsymbol{F_x} &= \boldsymbol{T_x} + \boldsymbol{G_x} + \boldsymbol{F_{Ax}} \\ \boldsymbol{F_y} &= \boldsymbol{T_y} + \boldsymbol{G_y} + \boldsymbol{F_{Ay}} \\ \boldsymbol{F_z} &= \boldsymbol{T_z} + \boldsymbol{G_z} + \boldsymbol{F_{Az}} \end{aligned} \tag{3.23}$$

where F_i means the total force, T_i stands for thrust, G_i is gravitational force and F_{Ai} is the aerodynamic force along the direction "i". To visualize these forces, the force diagram on the pitch plane is depicted in Figure 3.9.

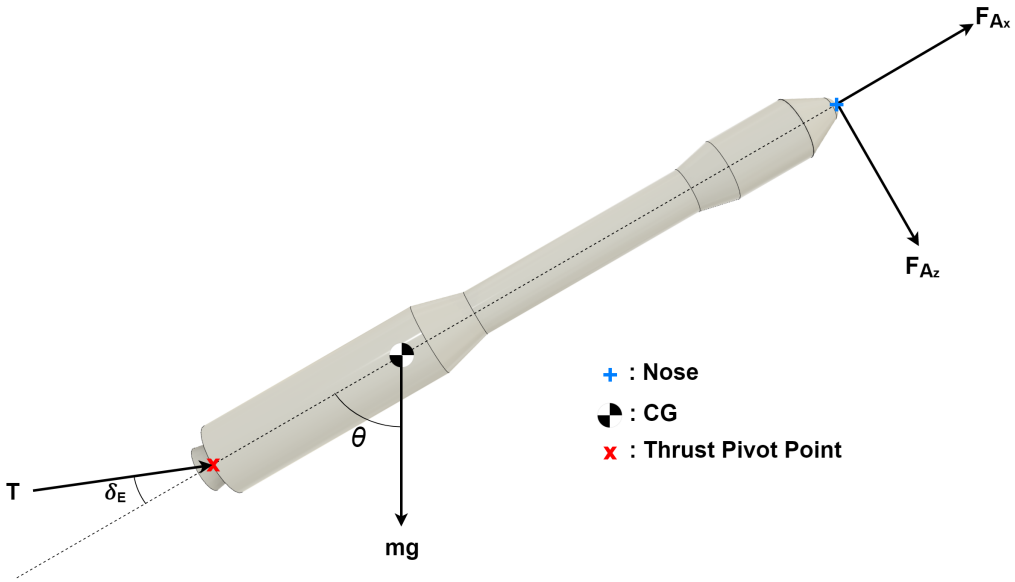

Figure 3.9: VEGA rocket and forces exerted on it in the pitch plane

The calculation of these forces are as follows.

$$
\begin{aligned}
T_x &= T\cos(\delta_e)\cos(\delta_r) \\
T_y &= -T\cos(\delta_e)\sin(\delta_r) \\
T_z &= T\sin(\delta_e)\cos(\delta_r)
\end{aligned} \tag{3.24}
$$

where T is the total thrust force magnitude, δ_e and δ_r is the thrust deflection angle in pitch and yaw plane,

$$
\begin{aligned}
G_x &= -mg\sin(\theta) \\
G_y &= -mg\sin(\psi)\cos(\theta) \\
G_z &= mg\cos(\psi)\cos(\theta)
\end{aligned} \tag{3.25}
$$

where m is mass, g is the gravitational acceleration, ψ and θ are attitude angles in NED Frame.

$$
\begin{aligned}
F_{Ax} &= q_\infty S_{ref} C_x \\
F_{Ay} &= q_\infty S_{ref} C_y \\
F_{Az} &= q_\infty S_{ref} C_z
\end{aligned} \tag{3.26}
$$

where q_∞ is the dynamic pressure, S_{ref} is the reference area and C_i is the aerodynamic force coefficient along the direction "i". Note that the rocket is symmetrical around body x axis, hence the aerodynamic coefficient around pitch and yaw plane are equal. Moreover, instead of using directional coefficients, generally axial and drag coefficients are preferred. Hence, the following substitutions will be made: $C_x = -C_A$, $C_z = -C_N$. Combining the equation sets 3.23, 3.24, 3.25 and 3.26:

$$
\begin{aligned}
m(\dot{u} + wq - vr) &= T\cos(\delta_e)\cos(\delta_r) - mg\sin(\psi) - q_\infty S C_A \\
m(\dot{v} + ur - wp) &= -T\cos(\delta_e)\sin(\delta_r) - mg\sin(\psi)\cos(\theta) - q_\infty S C_N \\
m(\dot{w} + vp - uq) &= T\sin(\delta_e)\cos(\delta_r) + mg\cos(\psi)\cos(\theta) - q_\infty S C_N
\end{aligned} \tag{3.27}
$$

Rearranging the equations given in 3.27, final translational dynamics equations can be obtained.

$$
\begin{aligned}
\dot{u} &= \frac{T\cos(\delta_e)\cos(\delta_r)}{m} - g\sin(\psi)\frac{-q_\infty S C_A}{m} - wq + vr \\
\dot{v} &= -\frac{T\cos(\delta_e)\sin(\delta_r)}{m} - g\sin(\psi)\cos(\theta) - \frac{q_\infty S C_N}{m} - ur + wp \\
\dot{w} &= \frac{T\sin(\delta_e)\cos(\delta_r)}{m} + g\cos(\psi)\cos(\theta) - \frac{q_\infty S C_N}{m} - vp + uq
\end{aligned} \tag{3.28}
$$

3.2.1.3 Rotational Dynamics

The angular momentum of the mass dm rotating around a point at a distance r with angular velocity $\boldsymbol{\omega} = p\widehat{i} + q\widehat{j} + r\widehat{k}$ is defined as:

$$d\boldsymbol{H} = \boldsymbol{r} \times dm\boldsymbol{V} = dm\boldsymbol{r} \times (\boldsymbol{\omega} \times \boldsymbol{r}) \tag{3.29}$$

Expanding the term on the most right hand side of the equation 3.29, using the inertia definition, neglecting the cross inertia terms ($I_{xy} = I_{xz} = I_{yz} = 0$), and finally with the integration over the entire mass, the elements of $\boldsymbol{H}$ can be found as follows.

$$\begin{aligned} \boldsymbol{H_x} &= pI_x \\ \boldsymbol{H_y} &= qI_y \\ \boldsymbol{H_z} &= rI_z \end{aligned} \tag{3.30}$$

From the Newton's 2nd law of the motion, the moment equation can be written in the form:

$$\sum \boldsymbol{M} = \left.\frac{d\boldsymbol{H}}{dt}\right|_I + \boldsymbol{\omega} \times \boldsymbol{H} \tag{3.31}$$

Combining equations 3.30 and 3.31, one can obtain the following 3 equations:

$$\begin{aligned} \sum \Delta M_x &= \dot{p}I_x + qr(I_z - I_y) \\ \sum \Delta M_y &= \dot{q}I_y + pr(I_x - I_z) \\ \sum \Delta M_z &= \dot{r}I_z + pq(I_y - I_x) \end{aligned} \tag{3.32}$$

Writing the aerodynamic moments acting on the rocket around 3 different body axis:

$$\begin{aligned} M_{A_x} &= C_{M_x} q_\infty S_{ref} l_{ref} \\ M_{A_y} &= C_{M_y} q_\infty S_{ref} l_{ref} \\ M_{A_z} &= C_{M_z} q_\infty S_{ref} l_{ref} \end{aligned} \tag{3.33}$$

Writing the and moments due to the deflection of the thrust vector:

$$\begin{aligned} M_{\delta_x} &= 0 \\ M_{\delta_y} &= T\sin(\delta_e)\cos(\delta_r)l_{arm} \\ M_{\delta_z} &= T\cos(\delta_e)\sin(\delta_r)l_{arm} \end{aligned} \tag{3.34}$$

Where C_{M_i} stands for the aerodynamic moment around the axis "i", l_{arm} is the lever arm between the thrust deflection pivot point and center of gravity, and l_{ref} is the reference length of the rocket. Combining the equations 3.32, 3.33 and 3.34:

$$\begin{aligned} \dot{p}I_x + qr(I_z - I_y) &= C_{M_x}q_\infty S_{ref}l_{ref} \\ \dot{q}I_y + pr(I_x - I_z) &= C_{M_y}q_\infty S_{ref}l_{ref} + T\sin(\delta_e)\cos(\delta_r)l_{arm} \\ \dot{r}I_z + pq(I_y - I_x) &= C_{M_z}q_\infty S_{ref}l_{ref} + T\cos(\delta_e)\sin(\delta_r)l_{arm} \end{aligned} \tag{3.35}$$

Re-arranging the equation 3.35, one can obtain the final rotational dynamics of the rocket.

$$\begin{aligned} \dot{p} &= \frac{C_{M_x}q_\infty S_{ref}l_{ref}}{I_x} - \frac{qr(I_z - I_y)}{I_x} \\ \dot{q} &= \frac{C_{M_y}q_\infty S_{ref}l_{ref}}{I_y} + \frac{T\sin(\delta_e)\cos(\delta_r)l_{arm}}{I_y} - \frac{pr(I_x - I_z)}{I_y} \\ \dot{r} &= \frac{C_{M_z}q_\infty S_{ref}l_{ref}}{I_z} + \frac{T\cos(\delta_e)\sin(\delta_r)l_{arm}}{I_z} - \frac{pq(I_y - I_x)}{I_z} \end{aligned} \tag{3.36}$$

3.2.1.4 Error Dynamics

The rocket is expected to move a predefined trajectory which has nonzero curvatures. The lateral distance to this predefined trajectory is not very important but the heading angle error needs to be minimized. This heading errors can be defined using the trajectory curvatures in pitch and yaw planes, ρ_{pitch} and ρ_{yaw} as follows:

$$\begin{aligned} \Delta\dot{\psi} &= r - V\rho_{yaw} \\ \Delta\dot{\theta} &= q - V\rho_{pitch} \end{aligned} \tag{3.37}$$

3.2.1.5 Actuator Dynamics

In the absence of an ideal actuator, there is a dynamic relation between the thrust deflection command and the actual thrust deflection. This dynamic is assumed to be

of 2nd order as follows:

$$\frac{\delta(s)}{\delta_c(s)} = \frac{\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2} \tag{3.38}$$

With this actuator, the states for elevator and rudder actuators become:

$$\begin{aligned} \ddot{\delta}_e &= -2\xi\omega_n\dot{\delta}_e - \omega_n^2\delta_e + \omega_n^2\delta_{e_c} \\ \ddot{\delta}_r &= -2\xi\omega_n\dot{\delta}_r - \omega_n^2\delta_r + \omega_n^2\delta_{r_c} \end{aligned} \tag{3.39}$$

where ω_n is the bandwidth of the actuator and ξ is the damping ratio.

3.2.2 Linear Launch Vehicle Model

Gathering the equations given in 3.28, 3.36, 3.37 and 3.39, the total nonlinear dynamics of the launch vehicle around pitch axis can be summarized as:

$$\dot{w} = \frac{T\sin(\delta_e)\cos(\delta_r)}{m} + g\cos(\psi)\cos(\theta) - \frac{q_\infty S C_N}{m} - vp + uq \tag{3.40}$$

$$\dot{q} = \frac{C_{M_y} q_\infty S_{ref} l_{ref}}{I_y} + \frac{T\sin(\delta_e)\cos(\delta_r) l_{arm}}{I_y} - \frac{pr(I_x - I_z)}{I_y} \tag{3.41}$$

$$\Delta\dot{\theta} = q - V\rho_{pitch} \tag{3.42}$$

$$\ddot{\delta}_e = -2\xi\omega_n\dot{\delta}_e - \omega_n^2\delta_e + \omega_n^2\delta_{e_c} \tag{3.43}$$

Instead of having two different states for lateral velocity w and angle of attack α, since they are related through the following equation, one can be expressed as in terms of the other using the small angle approximation:

$$\alpha = \tan^{-1}\left(\frac{w}{u}\right) \cong \frac{w}{u} \tag{3.44}$$

Then the final nonlinear state equations using α as one of the states becomes:

$$\dot{\alpha} = \frac{T\sin(\delta_e)\cos(\delta_r)}{mu} + \frac{g\cos(\psi)\cos(\theta)}{u} - \frac{q_\infty S C_N(\alpha, V)}{mu} - \frac{vp}{u} + q \tag{3.45}$$

$$\dot{q} = \frac{C_{M_y} q_\infty S_{ref} l_{ref}}{I_y} + \frac{T\sin(\delta_e)\cos(\delta_r) l_{arm}}{I_y} - \frac{pr(I_x - I_z)}{I_y} \tag{3.46}$$

$$\Delta\dot{\theta} = q - V\rho_{pitch} \tag{3.47}$$

$$\ddot{\delta}_e = -2\xi\omega_n\dot{\delta}_e - \omega_n^2\delta_e + \omega_n^2\delta_{e_c} \tag{3.48}$$

With the linearization of these equation around the equilibrium point

$$v = p = r = \alpha = \delta_r = 0, \;\; u = V \tag{3.49}$$

the linear state equations of the rocket around the pitch axis can be obtained as follows:

$$\dot{\alpha} = -\frac{q_\infty S C_{N\alpha}(V)}{mV}\alpha + q + \frac{T}{mV}\delta_e \tag{3.50}$$

$$\dot{q} = \frac{C_{M_{y\alpha}}(V) q_\infty S_{ref} l_{ref}}{I_y}\alpha + \frac{T l_{arm}}{I_y}\delta_e \tag{3.51}$$

$$\ddot{\delta}_e = -2\xi\omega_n\dot{\delta}_e - \omega_n^2\delta_e + \omega_n^2\delta_{e_c} \tag{3.52}$$

$$\Delta\dot{\theta} = q - V\rho_{pitch} \tag{3.53}$$

Using the following state vector, input and disturbance

$$x_p = \begin{bmatrix} \alpha \\ q \\ \dot{\delta}_e \\ \delta_e \\ \Delta\theta \end{bmatrix}, \quad u = \delta_{e_c}, \quad w = \rho_{pitch} \tag{3.54}$$

and rewriting the equations given in 3.50 through 3.53, the linear state equations of the launch vehicle in the pitch plane in the matrix form can be obtained as follows:

$$\dot{x_p} = A_p x_p + B_{pu} u + B_{pw} w \tag{3.55}$$

$$\begin{bmatrix} \dot{\alpha} \\ \dot{q} \\ \ddot{\delta_e} \\ \dot{\delta_e} \\ \dot{\Delta\theta} \end{bmatrix} = \begin{bmatrix} a_{11} & 1 & 0 & a_{14} & 0 \\ a_{21} & 0 & 0 & a_{24} & 0 \\ 0 & 0 & a_{33} & a_{34} & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ q \\ \dot{\delta_e} \\ \delta_e \\ \Delta\theta \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \omega_n^2 \\ 0 \\ 0 \end{bmatrix} \delta_{e_c} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ -V \end{bmatrix} \rho_{pitch} \tag{3.56}$$

where

$$a_{11} = -\frac{q_\infty S C_{N\alpha}(V)}{mV}, \quad a_{14} = \frac{T}{mV}, \quad a_{21} = \frac{C_{M_{y\alpha}}(V) q_\infty S_{ref} l_{ref}}{I_y}$$

$$a_{24} = \frac{T l_{arm}}{I_y}, \quad a_{33} = -2\xi\omega_n, \quad a_{34} = -\omega_n^2$$

CHAPTER 4

EXAMPLE APPLICATION-I: LANE KEEPING CONTROLLER

In the first part of this chapter, using the linear model of the vehicle obtained in the previous chapter, LPV model will be obtained. After that, using this LPV model, controller for this system will be designed. In the second part of this chapter, the nonlinear model of the vehicle will be implemented on MATLAB/Simulink and both open and closed loop simulations will be performed.

The controller for the vehicle will be used to make sure that the car stays on the lane. This controller algorithm can be utilized in a normal car to aid the driver [57] [58] or they can be used in a complete autonomous car [59]. In this work, the lateral dynamics of a vehicle will be modelled as a standalone system and it will be controlled against disturbances such as road curvature or change in the longitudinal velocity.

First, the main theory (from [9]) that will be followed in the control design process will be introduced. It consists of Lemmas and an optimization problem in an linear matrix inequality (LMI) form. Then, using the linear models derived in the previous chapter, LPV models of the systems will be obtained. Finally, the optimization problem given in the theorem will be constructed using the LPV models and will be solved to synthesise the controller.

There are different open source MATLAB toolboxes designed for LPV controller synthesis and LMI analysis [42] [60] [61]. In this study, the YALMIP toolbox will be used as it has flexible syntax and provide various solvers [61].

4.1 Theoretical Background

The theorem used in thesis is retrieved from [5] [43] [9] and is constructed by combining Lemma1 and Lemma2. In Lemma 1, a condition relating an LMI with H_2 norm of the LPV system with a relationship of the stability is given. In Lemma2, a property on the functions that aims to convert a parameter-dependent condition to a finite set of LMIs is given. These Lemmas are defined as follows:

Lemma1

Given an LPV system Σ

$$\Sigma_v(\theta) = \begin{cases} \dot{x} = \sum_{i=1}^{N} \eta_i(\theta)(A_i x + B_i^u u + B_i^\omega \omega) \\ z = \sum_{i=1}^{N} (C_i^z x + D_i^z u) \end{cases} \tag{4.1}$$

and a positive scalar α, assume there exists a symmetric positive definite matrix $Q(\theta) \in \mathbb{R}^{n_x \times n_x}$ and a matrix $Z(\theta) \in \mathbb{R}^{n_\omega \times n_\omega}$ and a positive scalar $\gamma > 0$ such that

$$\begin{bmatrix} \hat{A}(\theta)Q(\theta) + (\hat{A}(\theta)Q(\theta))^T + \alpha Q(\theta) - (\dot{\theta}) & (\hat{C}_Z(\theta)Q(\theta))^T \\ \hat{C}_Z(\theta)Q(\theta) & -I \end{bmatrix} < 0 \tag{4.2}$$

$$\begin{bmatrix} Z(\theta) & \hat{B}_w(\theta)^T \\ \hat{B}_w(\theta) & Q(\theta) \end{bmatrix} > 0 \tag{4.3}$$

$$trace(Z(\theta)) < \gamma^2 \tag{4.4}$$

Then it follows that $||\Sigma||_2 < \gamma$ and the associate Lyapunov function of LPV system is:

$$V(x) = x^T (\sum_{i=1}^{N} \eta_i(\theta) Q_i)^{-1} x = x^T Q(\theta)^{-1} x \tag{4.5}$$

Lemma2

Let Υ_{ij}, i,$j \in \Omega_N$, be symmetric matrices of appropriate dimensions and $\{\eta_i\}_{i\in\Omega_N}$, be any family of functions satisfying the following property:

$$\eta_i\theta \geq 0, \quad \sum_{i=1}^{N} \eta_i(\theta) = 1, \quad \sum_{i=1}^{N} \dot{\eta}_i(\theta) = 0 \tag{4.6}$$

$$\dot{\eta}_i(\theta) \in [\phi_{i1}, \phi_{i2}], \quad i \in \Omega_N \tag{4.7}$$

The condition

$$\sum_{i=1}^{N}\sum_{j=1}^{N} \eta_i\eta_j\Upsilon_{ij} < 0 \tag{4.8}$$

holds if

$$\begin{cases} \Upsilon_{ii} < 0, & i \in \Omega_N \\ \frac{2}{(N-1)}\Upsilon_{ii} + \Upsilon_{ij} + \Upsilon_{ji} < 0, & i,j \in \Omega_N, \text{ and } i < j \end{cases} \tag{4.9}$$

The following theorem that utilizes the Lemma1 and Lemma 2 will be used to synthesise the controller. In this theorem, an LMI related to the H_2 norm of the LPV system is given. By solving this LMI while minimizing the norm the controller will be synthesized.

Theorem

Given an LPV system (with N vertices) (4.1) and a positive scalar $\alpha > 0$, assume that there exist symmetric positive definite matrices $Q_i \in \mathbb{R}^{n_x\times n_x}$, matrices $M_i \in \mathbb{R}^{n_u\times n_y}$, $L_i^{\omega} \in \mathbb{R}^{n_u\times n_\omega}$, $X \in \mathbb{R}^{n_y\times n_y}$, $Z_i \in \mathbb{R}^{n_\omega\times n_\omega}$ for $i \in \Omega_N$, where Ω_N means the integer set $1, 2, ..., N$., and positive scalars $\gamma > 0$, $\epsilon > 0$ such that

$$
\begin{aligned}
\begin{bmatrix} Z_i & * \\ B_i^{\omega} + B_i^u L_j^{\omega} & Q_i \end{bmatrix} > 0, \quad i,j \in \Omega_N \\
trace(Z_i) < \gamma^2, i \in \Omega_N \\
\Xi_{ii}^{klm} < 0, \quad i,k,l \in \Omega_N, \quad m \in \Omega_2, \quad k \neq l \\
\frac{2}{N-1}\Xi_{ii}^{klm} + \Xi_{ij}^{klm} + \Xi_{ji}^{klm} < 0, i,j,k,l \in \Omega_N,\ m \in \Omega_2,\ i < j,\ k \neq l
\end{aligned}
\tag{4.10}
$$

where the quantity Ξ_{ii}^{klm} is defined as follows:

$$
\Xi_{ij}^{klm} = \begin{bmatrix} \Xi_{ij(1,1)}^{klm} & 0 & \epsilon B_i^u M_j \\ D_i^z M_j C_y + C_i^z Q_j & -I & \epsilon D_i^z M_j \\ C_y Q_j - X C_y & 0 & -\epsilon X \end{bmatrix} + \begin{bmatrix} \Xi_{ij(1,1)}^{klm} & 0 & \epsilon B_i^u M_j \\ D_i^z M_j C_y + C_i^z Q_j & -I & \epsilon D_i^z M_j \\ C_y Q_j - X C_y & 0 & -\epsilon X \end{bmatrix}^T
\tag{4.11}
$$

$$
\Xi_{ij(1,1)}^{klm} = A_i Q_j + B_i^u M_j C_y + \alpha Q_j - \phi_{km}(Q_k - Q_l)/2
$$

Then, the SOF controller stabilizes the LPV system and satisfies $\|\Sigma\|_2 < \gamma$. Moreover, the control feedback gains are given by

$$
K_i = M_i X^{-1}, \quad i \in \Omega_N \tag{4.12}
$$

In other words, the controller gains are obtained by solving the following optimization problem.

$$
\min_{Q_i, Z_i, M_i, L_i^{\omega}, X} \gamma^2 \quad \text{subject to}
$$

$$
\begin{aligned}
\begin{bmatrix} Z_i & * \\ B_i^{\omega} + B_i^u L_j^{\omega} & Q_i \end{bmatrix} > 0, \quad i,j \in \Omega_N \\
trace(Z_i) < \gamma^2, i \in \Omega_N \\
\Xi_{ii}^{klm} < 0, \quad i,k,l \in \Omega_N, \quad m \in \Omega_2, \quad k \neq l \\
\frac{2}{N-1}\Xi_{ii}^{klm} + \Xi_{ij}^{klm} + \Xi_{ji}^{klm} < 0, i,j,k,l \in \Omega_N,\ m \in \Omega_2,\ i < j,\ k \neq l
\end{aligned}
\tag{4.13}
$$

4.2 Lane Keeping Controller Synthesis

To synthesise the controller for the purpose of lane keeping, the LPV model of the lateral dynamics of the vehicle is needed. This model will be constructed using the linear model after choosing output and performance vector and the scheduling parameter since these parameters change state space matrices of the LPV system. In order to improve the behaviour of the controller, an input shaping filter in the form of a road curvature model will be integrated to the model. Although some steps can be minorly changed or skipped, mainly the formulation given in [9] will be followed in this section.

4.2.1 LPV Vehicle Lateral Dynamic Model

In order to incorporate some information about the disturbance dynamics into the controller design, generator model is considered, which can be regarded as input shaping filter. Hence, the controller will use this information to improve its overall behaviour [9]. The input of this filter is an irreducible signal and its output will be the road curvature. The impulse response of this curvature model will contain the information regarding the expected road curvature in the following way. The peak value of the impulse response corresponds to the maximum value of the expected road curvature and the time to peak corresponds to the expected peak time of the road curvature change. For this purpose, the following 3rd order model will be used.

$$\Sigma_w = \begin{cases} \dot{x_w} = A_c x_w + B_c x_w \\ y_w = C_c x_w \end{cases} \tag{4.14}$$

where $x_w = \begin{bmatrix} \rho & \dot{\rho} & \ddot{\rho} \end{bmatrix}^T$ and $y_w = \rho$ with the state space matrices

$$A_c = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -a_{c0} & -a_{c1} & -ac2 \end{bmatrix}, \quad B_c = \begin{bmatrix} 0 \\ 0 \\ K_c \end{bmatrix}, \quad C_c = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \tag{4.15}$$

The output vector used in the feedback and corresponding C_p matrix is shown in

the equation 4.16. Note that $\dot{y_R}$ and $\dot{\delta}$ are absent in the output vector, since they require more expensive sensors which is not an ideal situation for the mass production systems. It is also important to point out that the vector y_p is not the complete output vector which will be used in the feedback. The road curvature information will be also used in the feedback. This situation can be seen in the definitions made in the equation 4.21.

$$y_p = \begin{bmatrix} \dot{\psi} \\ y_R \\ \Delta\psi \\ \delta_T \end{bmatrix}, \quad C_p = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.16}$$

Another vector to be decided is the performance vector. One of the performance measures of the lane keeping algorithm is how fast the lateral error diminishes. After all, the main purpose of this algorithm is to keep this error as minimum as possible. Moreover, the heading angle error also needs to be eliminated. The response of the LKA to eliminate these two errors are important for the success of the algorithm. However, apart from keeping the error states at minimum, there is another property that the vehicle needs to satisfy: the driving comfort. To increase the comfort, the lane keeping algorithm should minimize the lateral acceleration, which is related with the steering angle of the tires. Hence, the controlled output vector of the algorithm $z = D_{zx_p}x_p + D_{zw}x_w$ is chosen as follows.

$$z = \begin{bmatrix} \Delta\psi \\ y_R \\ a_y \\ \delta_t \end{bmatrix} \tag{4.17}$$

Where the matrices D_{zx_p} and D_{zw} are defined as follows.

$$D_{zx_p} = W_z \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ \dfrac{c_t(a^2+b^2)}{I_{zz}V} & 0 & \dfrac{c_t(a-b)}{I_{zz}V} & 0 & -\dfrac{c_t(a-b)}{I_{zz}} & 0 & -\dfrac{c_t a}{I_{zz}} \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$D_{zw} = W_z \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ -V^2 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{4.18}$$

The term W_z used in the above definitions is the weight matrix. It is used to emphasize more importance on some states than others and has the following form.

$$W_z = \begin{bmatrix} W_1 & 0 & 0 & 0 \\ 0 & W_2 & 0 & 0 \\ 0 & 0 & W_3 & 0 \\ 0 & 0 & 0 & W_4 \end{bmatrix} \tag{4.19}$$

Combining the state space matrices of the vehicle model given in equation 3.18 and 3.19 the road curvature model given in equation 4.14, with the matrices for output and performance vectors given in equations 4.16 and 4.18, the complete vehicle-road model becomes as follows.

$$\Sigma(V) : \begin{cases} \dot{x} = Ax + B_u u + B_w w \\ z = C_z x, \quad y = C_y x \end{cases} \tag{4.20}$$

where

$$x = \begin{bmatrix} x_p \\ x_w \end{bmatrix}, \quad y = \begin{bmatrix} y_p \\ y_w \end{bmatrix}$$

$$A = \begin{bmatrix} A_p & B_{pw}C_c \\ 0 & A_c \end{bmatrix}, \quad B_u = \begin{bmatrix} B_{pu} \\ 0 \end{bmatrix}, \quad B_w = \begin{bmatrix} 0 \\ B_c \end{bmatrix} \tag{4.21}$$

$$C_y = \begin{bmatrix} C_p & 0 \\ 0 & C_c \end{bmatrix}, \quad C_z = \begin{bmatrix} D_{zx_p} & D_{zw} \end{bmatrix}$$

The complete model given in the equation 4.20 is still not in LPV form. To convert this system into LPV form, the scheduling parameter must be chosen. Note that the state space matrices of the road vehicle model depends on longitudinal velocity of the vehicle in different form such as V, V^2 and $1/V$. For scheduling parameter, $1/V$

is chosen through parameter θ and parameters V and V^2 are expressed in terms of θ using the 1st Taylor series as follows.

$$\frac{1}{V} = \frac{1}{V_0} + \frac{1}{V_1}\theta, \quad V \approx V_0\left(1 - \frac{V_0}{V_1}\theta\right), \quad V^2 \approx V_0^2\left(1 - \frac{2V_0}{V_1}\theta\right) \tag{4.22}$$

with

$$V_0 = \frac{2V_{min}V_{max}}{V_{min} + V_{max}} \quad \text{and} \quad V_1 = \frac{2V_{min}V_{max}}{V_{min} - V_{max}} \tag{4.23}$$

Hence, the variation in the velocity is taken into account by the parameter θ which changes between -1 and 1 corresponding to velocities V_{min} and V_{max}. Using the scheduling parameter θ and the complete road vehicle model given in equation 4.20, the polytopic LPV model of the system becomes:

$$\Sigma(\theta) : \begin{cases} \dot{x} = \sum_{i=1}^{2} \eta_i(\theta)\left(A_i x + B_i^u u + B_i^w w\right) \\ z = \sum_{i=1}^{2} \eta_i(\theta) C_i^z x, \quad y = C_y x \end{cases} \tag{4.24}$$

where the state space matrices are given by:

$$\begin{aligned} A_1 = A(\theta_{min}), \quad B_1^u = B_u, \quad B_1^w = B_w, \quad C_1^z = C_z(\theta_{min}) \\ A_2 = A(\theta_{max}), \quad B_2^u = B_u, \quad B_2^w = B_w, \quad C_1^z = C_z(\theta_{max}) \end{aligned} \tag{4.25}$$

The function η_i used in the polytopic representation given in 4.24 is called scalar membership function. These functions are utilized for expressing a model in terms of convex combination of the systems at the vertex of the polytope. They have the following definition and properties.

$$\eta_1(\theta) = \frac{1-\theta}{2}, \quad \eta_2(\theta) = 1 - \eta_1(\theta) = \frac{1+\theta}{2} \tag{4.26}$$

$$\eta_i(\theta) \geq 0, \quad \sum_{i=1}^{2} \eta_i(\theta) = 1, \quad \sum_{i=1}^{2} \dot{\eta}_i(\theta) = 0 \tag{4.27}$$

In this work, besides the limits on the velocity of the vehicle, the acceleration limits are also taken into account.

$$a_{min} \leq a_x \leq a_{max} \tag{4.28}$$

From the equations 4.22 and 4.28, it follows that

$$\frac{a_{min}}{a_0} \leq \dot{\theta} \leq \frac{a_{max}}{a_0}, \text{ with } a_0 = \frac{-V_0^2}{V_1} \tag{4.29}$$

These acceleration limits are integrated into the synthesis through the membership function satisfying the additional propery:

$$\dot{\eta}_i(\theta) \in [\phi_{i1}, \phi_{i2}] \tag{4.30}$$

where

$$\phi_{11} = \frac{-a_{max}}{2a_0}, \quad \phi_{12} = \frac{-a_{min}}{2a_0}, \quad \phi_{21} = \frac{a_{min}}{2a_0}, \quad \phi_{22} = \frac{a_{max}}{2a_0} \tag{4.31}$$

4.2.2 Controller Design

First, the parameter for the road curvature model will be decided. After that, the scheduling parameter and its derivative limits will be set. Finally, using the numerical values of all the required matrices, the optimization problem given in 4.13 will be solved using YALMIP toolbox and feedback gains will be obtained.

In this work, the minimum road curve radius of $250m$ will be considered, hence the maximum expected road curvature is $1/250 = 4 \times 10^{-3} m^{-1}$. The impulse response peak time is chosen as 8 second. Using these values, the parameters for the road curvature model is decided as given in the table 4.1.

The velocity and acceleration limits are chosen as follows.

Table 4.1: Numeric values of the road curvature model

Parameter Name	**Value**
a_{c0}	0.05
a_{c1}	5
a_{c2}	10
K_c	0.022

$$\begin{aligned} V_{min} &= 50km/h, \quad V_{max} = 120km/h \\ a_{min} &= -3m/s^2, \quad a_{max} = 3m/s^2 \end{aligned} \tag{4.32}$$

To incorporate a priority among the states, the coefficient in the weighting matrices can be set different than 1. In this work, the heading error $\Delta\psi$ is considered as slightly more important and hence the weight matrix is chosen as follows.

$$W_z = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.33}$$

Apart from these constant values, there are other constant parameters needed to be set in order to solve the LMI defined in the theorem given in the chapter 4.1. One of these parameters is α. It is related with the decay rate of the states and the closed loop poles of the system is forced on the left of the line $x = -\alpha$. This parameter is set considering the preferred decay rate. Another parameter is γ, which defines the upper limit of the L_2 norm of the system, as it is pointed out in the theorem 4.1. Finally, the last parameter is ϵ, it is chosen by doing a practical method where a logarithmic grid over $[10^{-5}, 10^5]$ is scanned. These parameters are chosen as follows.

$$\alpha = 0.001, \quad \gamma = 40, \quad \epsilon = 0.2783 \tag{4.34}$$

To solve the LMI given in the theorem 4.1 with these parameter values, MOSEK solver introduced in [62] with the YALMIP toolbox [61] is used and following gain values are obtained (K_1 corresponds to $V = V_{min}$ and vice versa).

Table 4.2: Numeric values of the vehicle model

Parameter Name	Definition	Value
m	Mass	1480 kg
I_{zz}	Inertia of the vehicle around body z axis	1950 kgm^2
a	Distance between cg and front wheel	1.421 m
b	Distance between cg and rear wheel	1.029 m
B	Tire force coefficient	8.22
C	Tire force coefficient	1.65
D	Tire force coefficient	-1.7×10^4
E	Tire force coefficient	-10
ω_n	Steering dynamics bandwidth	3 Hz
ξ	Steering dynamics damping ratio	0.707
n_s	Steering dynamics gear ratio	16.34

$$\begin{aligned} K_1 &= \begin{bmatrix} -0.0010 & -0.0021 & -0.0381 & -0.0147 & 0.1682 \end{bmatrix} \\ K_2 &= \begin{bmatrix} -0.0017 & -0.0005 & -0.0250 & -0.0169 & 0.1904 \end{bmatrix} \end{aligned} \tag{4.35}$$

4.3 Nonlinear Dynamic Bicycle Model Simulation

In this section, the vehicle model for the lane-keeping problem given in section 3 will be implemented. Firstly, the implementation on MATLAB-Simulink will be described. Secondly, to validate the model, some open loop simulations will be conducted and results will be given. And finally, the controller will be integrated to the model and closed loop simulations in different scenarios will be performed.

The values of the vehicle parameters are taken from [50] while actuator values are obtained from [9] and can be seen in the table 4.2.

4.3.1 Implementation of the Nonlinear Simulation

The nonlinear model is implemented on Simulink using the plant state equations given in 3.1 3.2 3.3 3.4 3.5 3.6, error state equations given in 3.9 and 3.10, actuator states

given in 3.12. These equations are realized with the use of MATLAB function block and the derivatives of the states are obtained. With the integration of these derivative vector with the proper initial conditions, the state vector is obtained. The state vector then is multiplied with the matrix C to obtain the output vector, which is multiplied with the gain. The value of the feedback gain K is calculated using the value of the scheduling parameter θ whose calculation is as follows.

$$\theta = \frac{V_{max} + V_{min} - (2V_{max}V_{min}/V_x)}{V_{max} - V_{min}} \tag{4.36}$$

The gain is calculated using the value of the θ with the convex combination of the gain values at the vertices of the polytope. The vertices of the polytope corresponds to V_{min} where $\theta = -1$ and V_{max} where $\theta = 1$. The value of the gain as a function of the scheduling parameter θ is calculated as follows.

$$K(\theta) = \frac{(K_1 - K_2)(1 - \theta)}{2} + K_2 \tag{4.37}$$

where K_1 corresponds to the gain value obtained for $V_x = V_{min}$ and K_2 corresponds to the gain value obtained for $V_x = V_{max}$. The longitudinal velocity is calculated by the integration of the longitudinal acceleration, which is an input to the model. The implementation of this model on the Simulink software is depicted in Figure 4.1

The validation of this model is realized with an open loop simulation, where a constant steering angle and velocity is used as an experiment. The vehicle is expected to draw a circle in this scenario. This case is called "Case0" and the simulation result can be seen in figure 4.2.

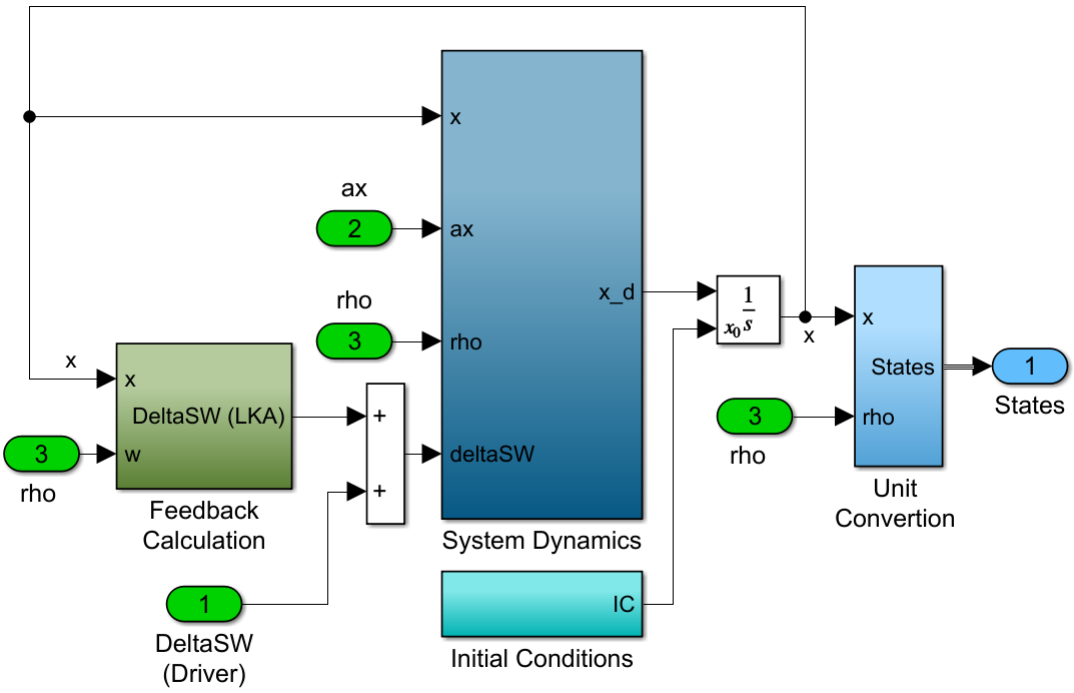

Figure 4.1: Implementation of the nonlinear vehicle model on Simulink

4.3.2 Open Loop Simulation of the Vehicle Lane Keeping

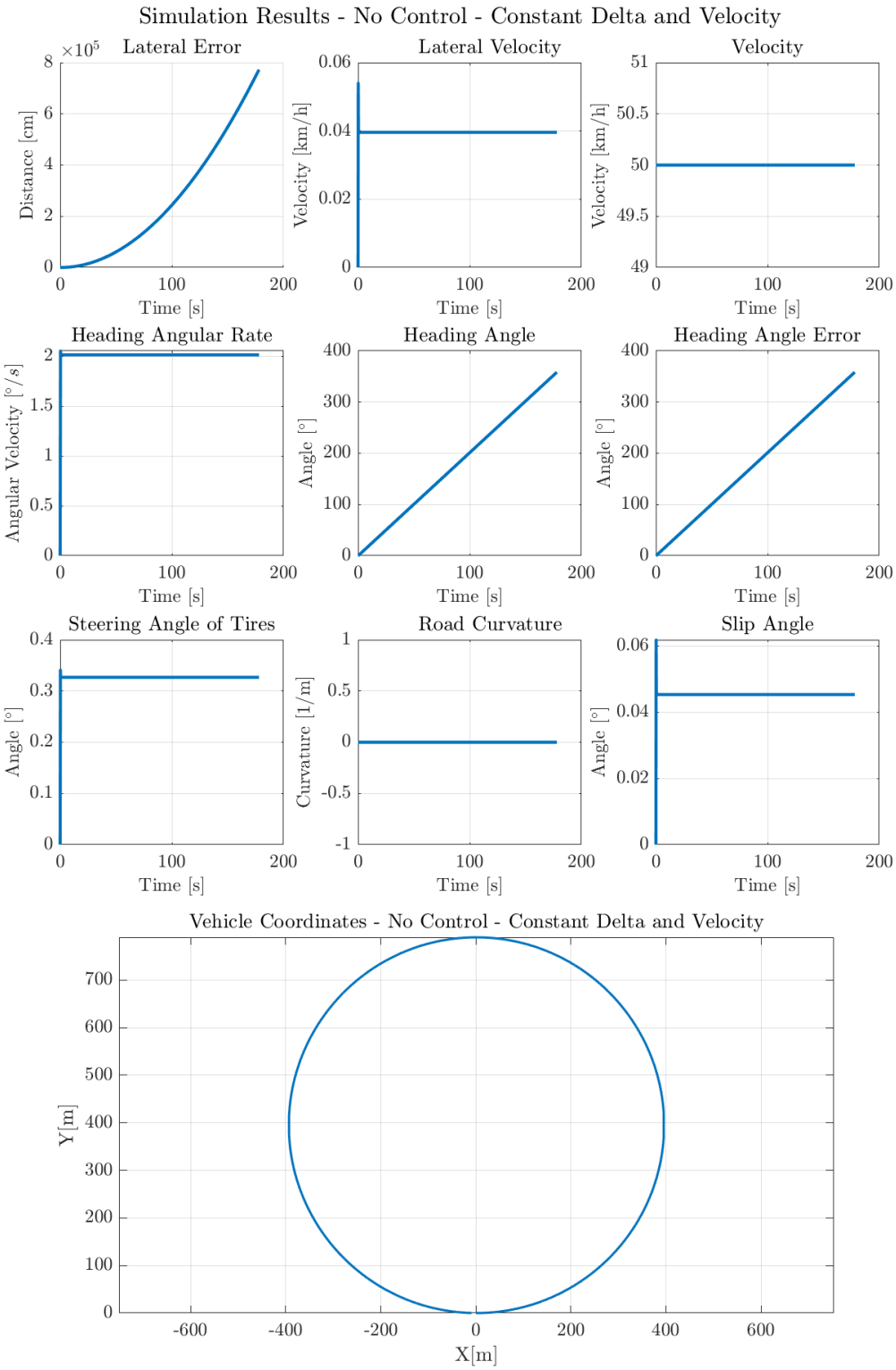

Figure 4.2: Simulation results for Case0

4.3.3 Closed Loop Simulation of the Vehicle Lane Keeping

In this section, the performance of the LPV controller will be tested against different initial conditions and disturbances. Firstly, the response of the controller against initial displacement error will be tested. This scenario is very likely to happen in real life since the lane-keeping algorithm can be activated when the vehicle is not in the middle of the road. The first two cases will show the performance of the controller against this initialization displacement error for different velocities, namely $V = V_{min}$ and $V = V_{max}$. Note that these velocities are used in the design, so another case where $V = (V_{min} + V_{max})/2$ will be tested in the third case.

Case-4, Case-5 and Case-6 are added to test cases to validate the controller under nonzero initial heading angle error condition. Again, in each case different velocities will be tested

The seventh, eighth and ninth scenarios consider a situation where the vehicle is in the center of the line with LKA is open and road starts turning. In a perfect scenario, the vehicle follows the center-line of the road with zero error, but this is not a practical situation. Usually there is some small steady state error while turning, and this error goes to zero when the road curvature drops to zero.

The last three cases utilizes varying road curvature and velocity. In the Case-10, the square wave and smoothed saw-tooth wave will be used for road curvature and acceleration values respectively. In the last two cases, some predefined profiles will be utilized. The profile used in the Case-11 is taken from [9]. The profile for the Case-12 is constructed by considering a road around METU. In these cases, the performance of the controller against both nonzero road curvature and change in the lateral dynamics will be tested. These simulation cases are shown in the table 4.3.

After testing the designed LPV controller in 12 different cases, its performance should be compared to the methods in literature. For this purpose, another controller from the literature [63] in the form of PIDD^2 will be implemented and tested in the scenario Case-12 to compare the controllers. The block diagram of this controller is as shown in the figure 4.3.

Table 4.3: Scenarios that will be used to validate the controller

Case Name	**Explanation**
Case-1	Initial lateral displacement error & $V = 50km/h$
Case-2	Initial lateral displacement error & $V = 85km/h$
Case-3	Initial lateral displacement error & $V = 120km/h$
Case-4	Initial heading error & $V = 50km/h$
Case-5	Initial heading error & $V = 85km/h$
Case-6	Initial heading error & $V = 120km/h$
Case-7	Nonzero road curvature & $V = 50km/h$
Case-8	Nonzero road curvature & $V = 85km/h$
Case-9	Nonzero road curvature & $V = 120km/h$
Case-10	Nonzero a_x & ρ, $V_0 = 100km/h$
Case-11	a_x & ρ Profile #1, $V_0 = 80km/h$
Case-12	a_x & ρ Profile #2, $V_0 = 100km/h$

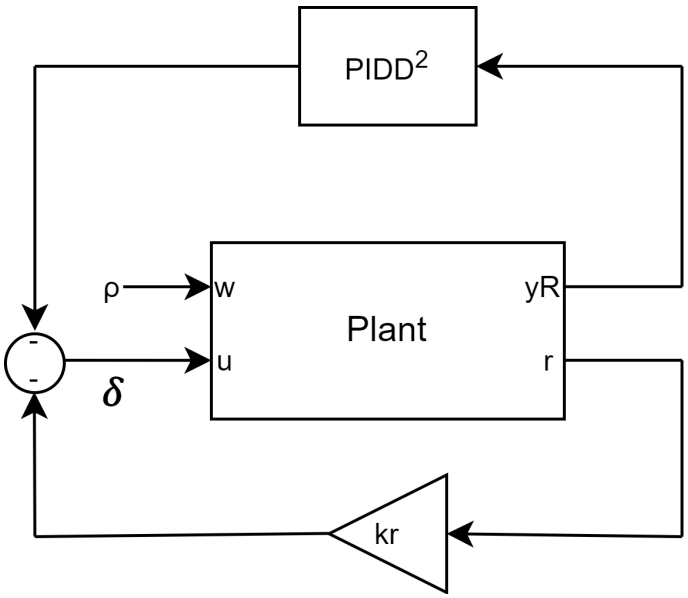

Figure 4.3: Controller used for comparison

As seen in the block diagram given in the figure 4.3, this controller utilizes a feedback consisting of the angular yaw velocity and lateral displacement error. This algorithm is obtained from [63]. Note that to implement this controller, the steering dynamics is removed from the model.

4.3.3.1 Case-I: Nonzero Initial Lateral Error, V = 50km/h

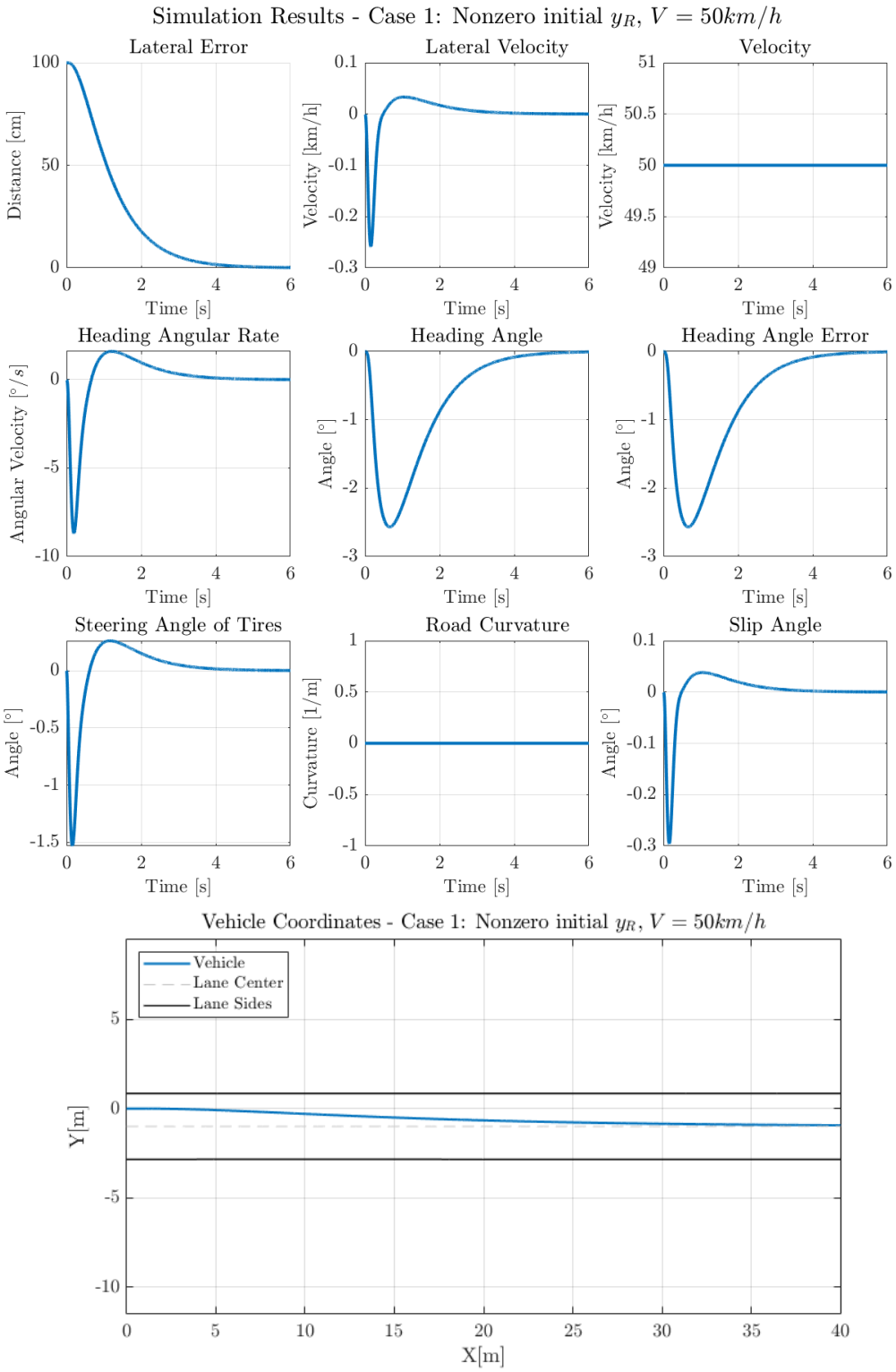

Figure 4.4: Simulation results for Case1

4.3.3.2 Case-II: Nonzero Initial Lateral Error, V = 85km/h

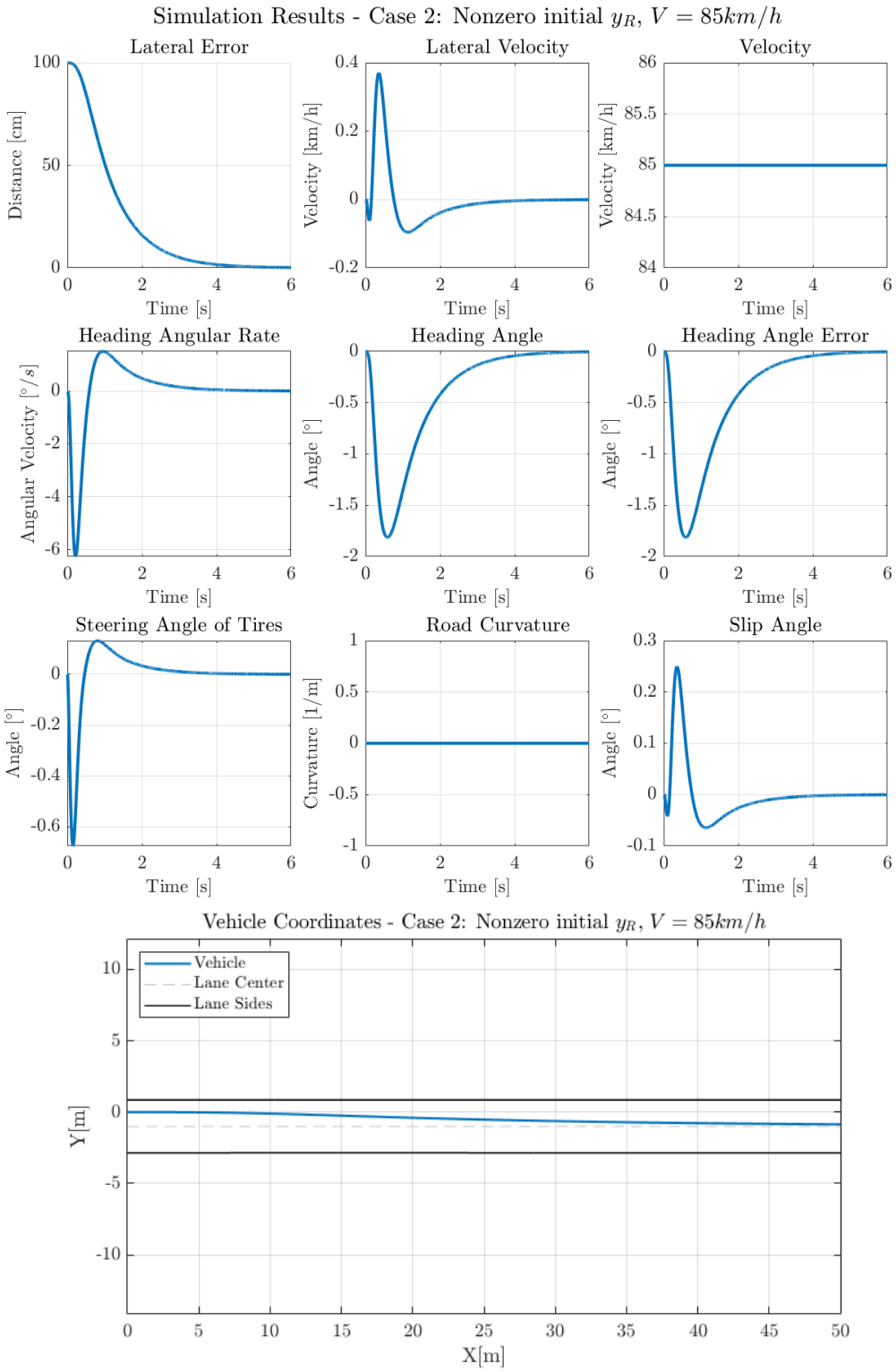

Figure 4.5: Simulation results for Case2

4.3.3.3 Case-III: Nonzero Initial Lateral Error, V = 120km/h

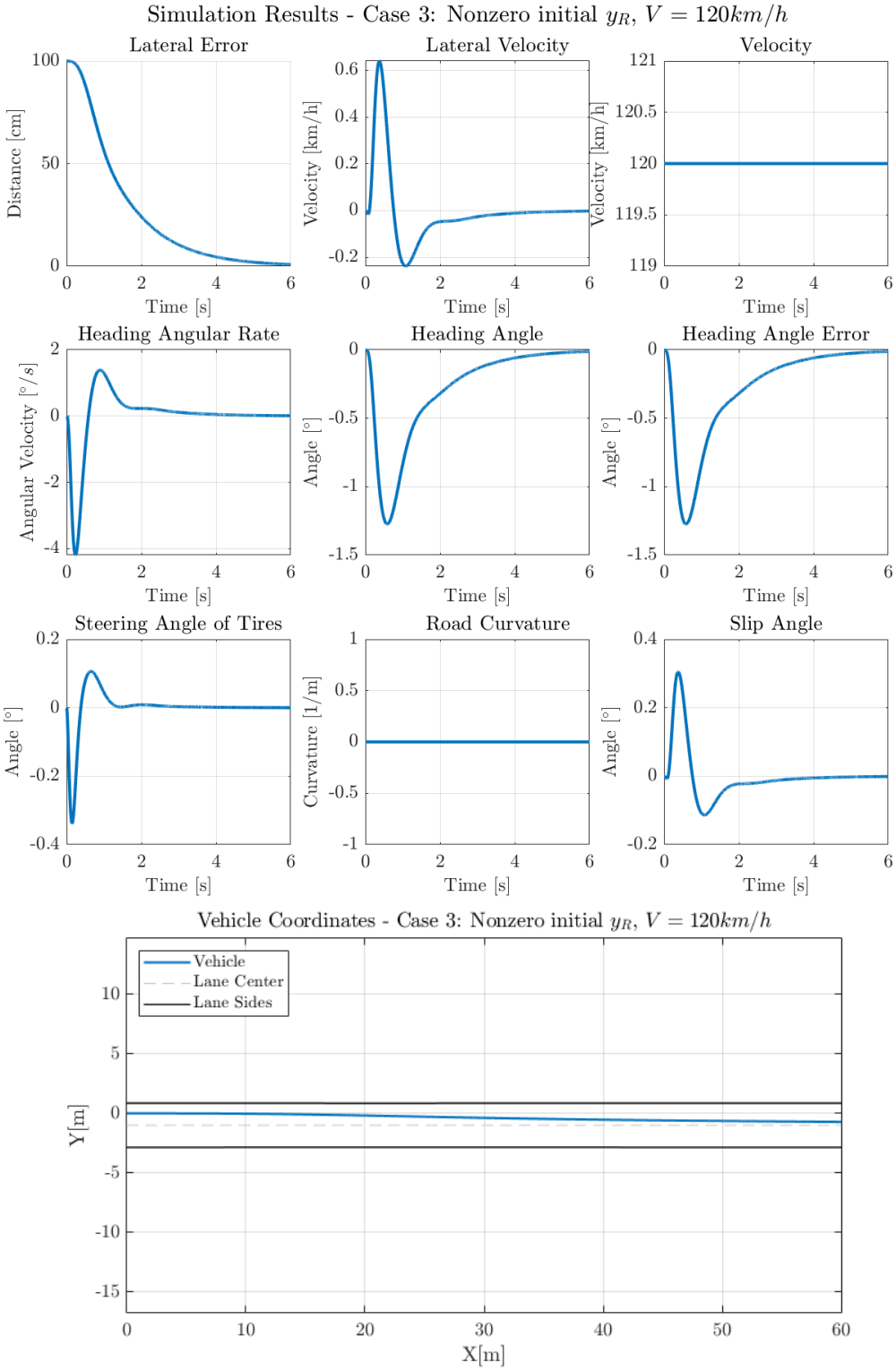

Figure 4.6: Simulation results for Case3

4.3.3.4 Case-IV: Nonzero Initial Heading Error, V = 50km/h

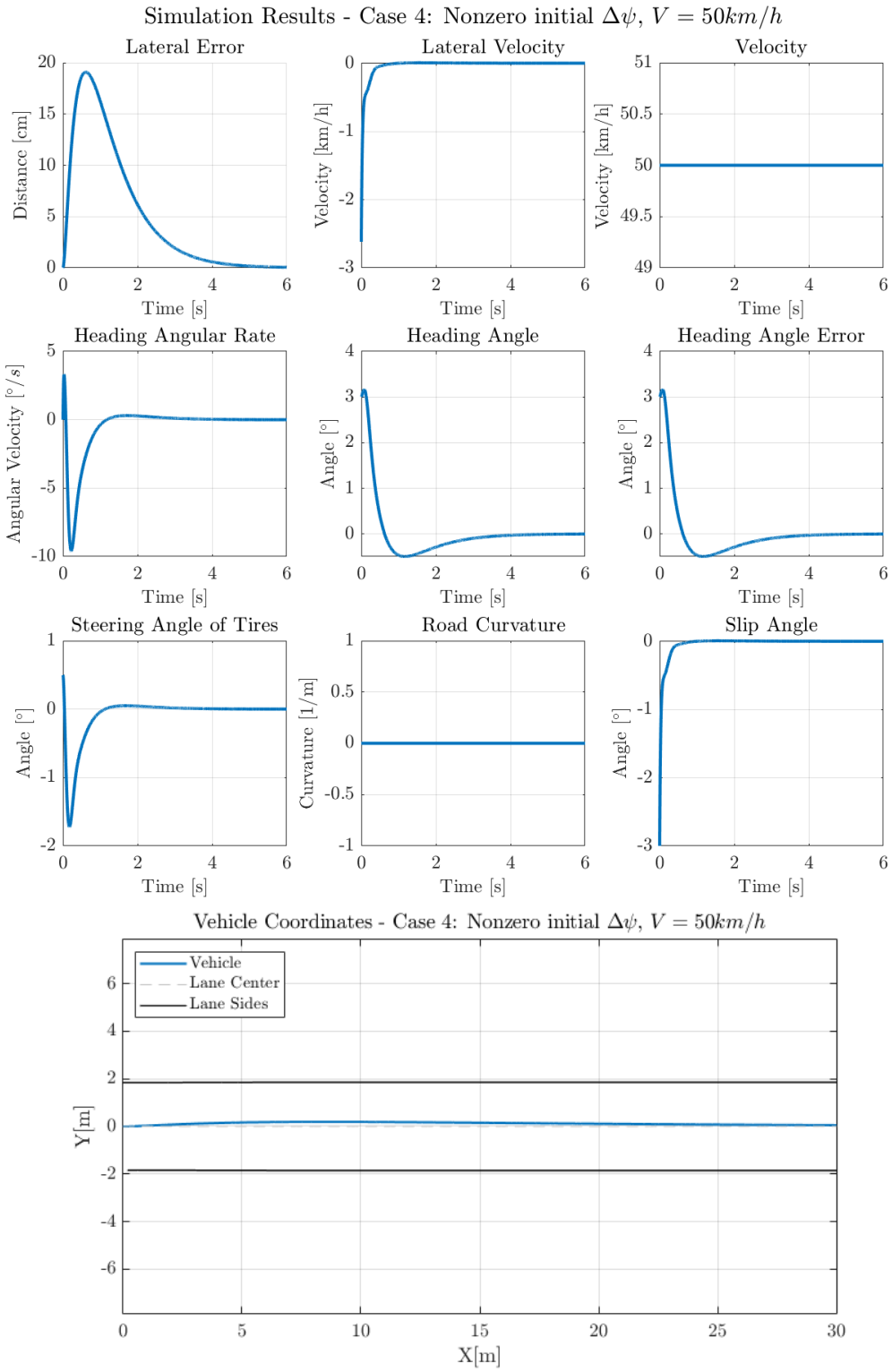

Figure 4.7: Simulation results for Case4

4.3.3.5 Case-V: Nonzero Initial Heading Error, V = 85km/h

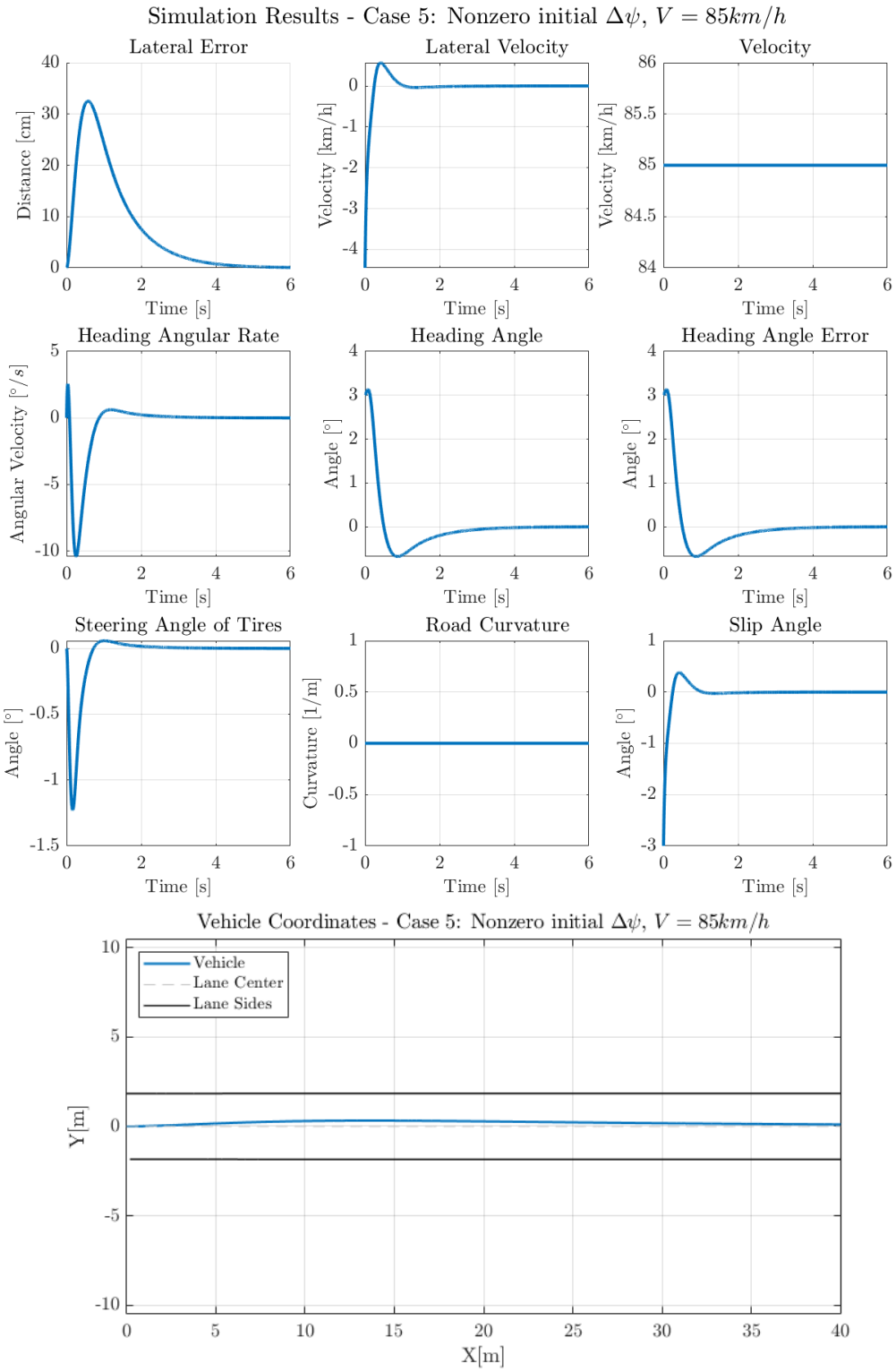

Figure 4.8: Simulation results for Case5

4.3.3.6 Case-VI: Nonzero Initial Heading Error, V = 120km/h

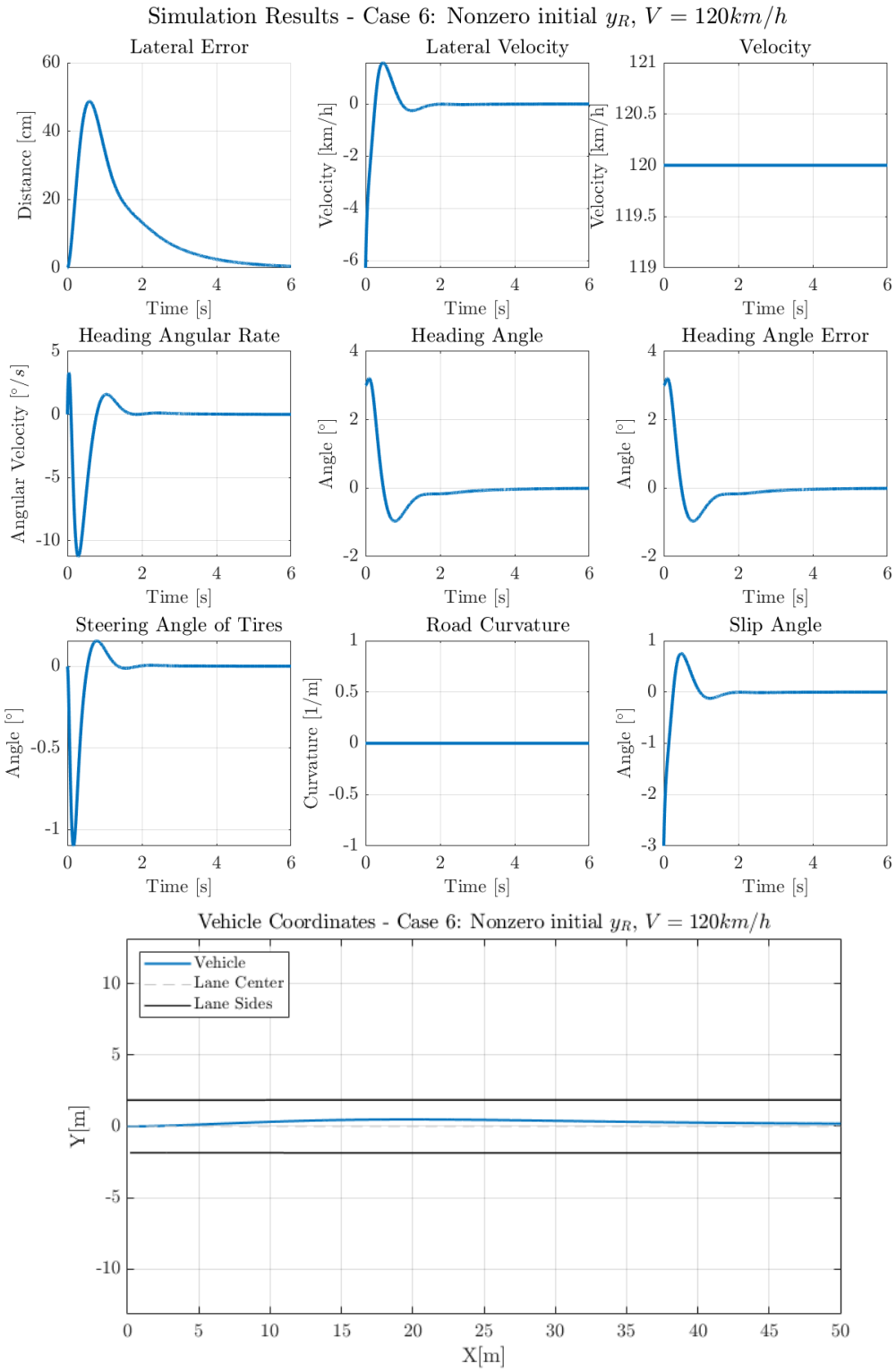

Figure 4.9: Simulation results for Case6

4.3.3.7 Case-VII: Nonzero Road Curvature, V = 50km/h

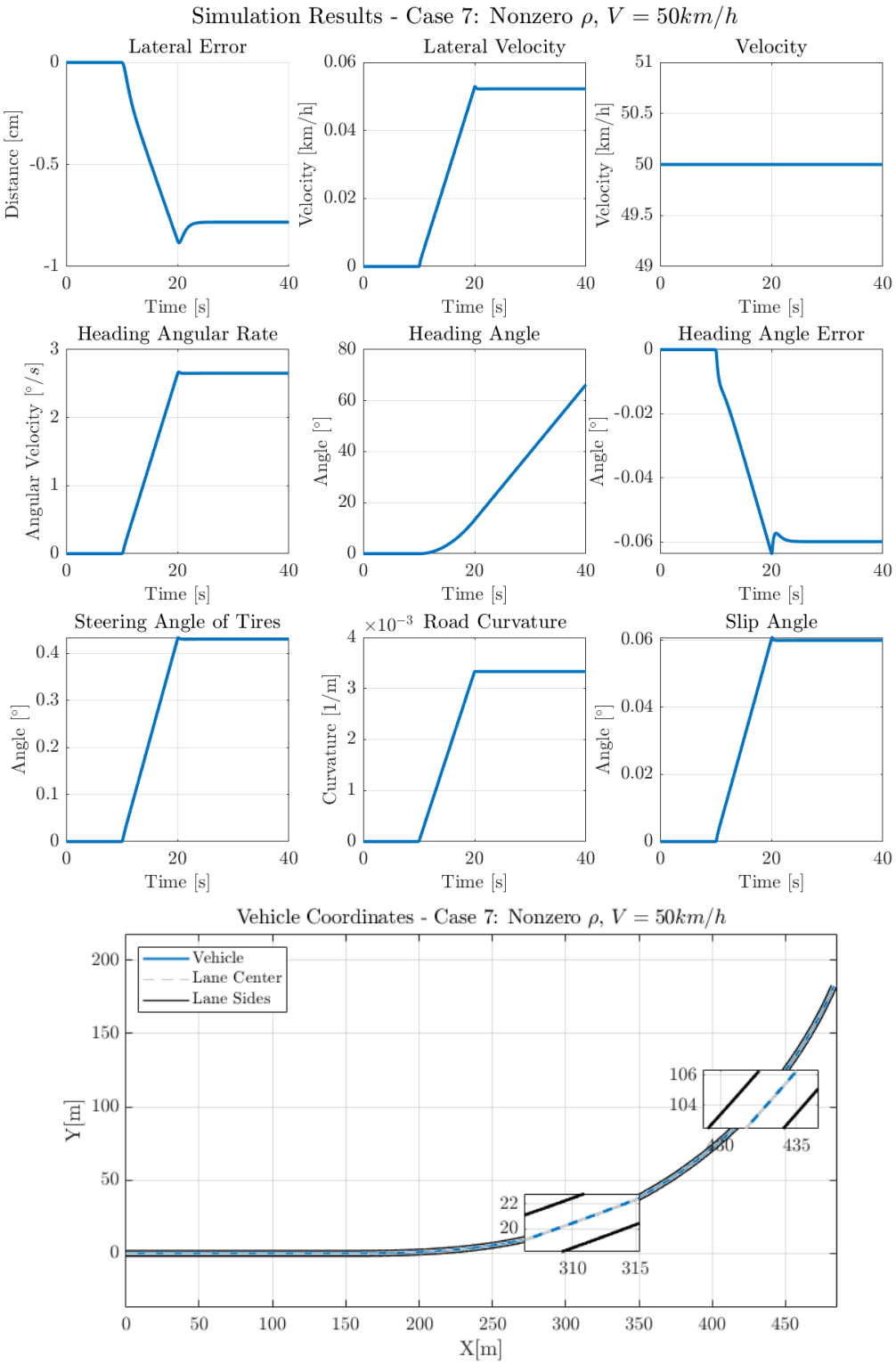

Figure 4.10: Simulation results for Case7

4.3.3.8 Case-VIII: Nonzero Road Curvature, V = 85km/h

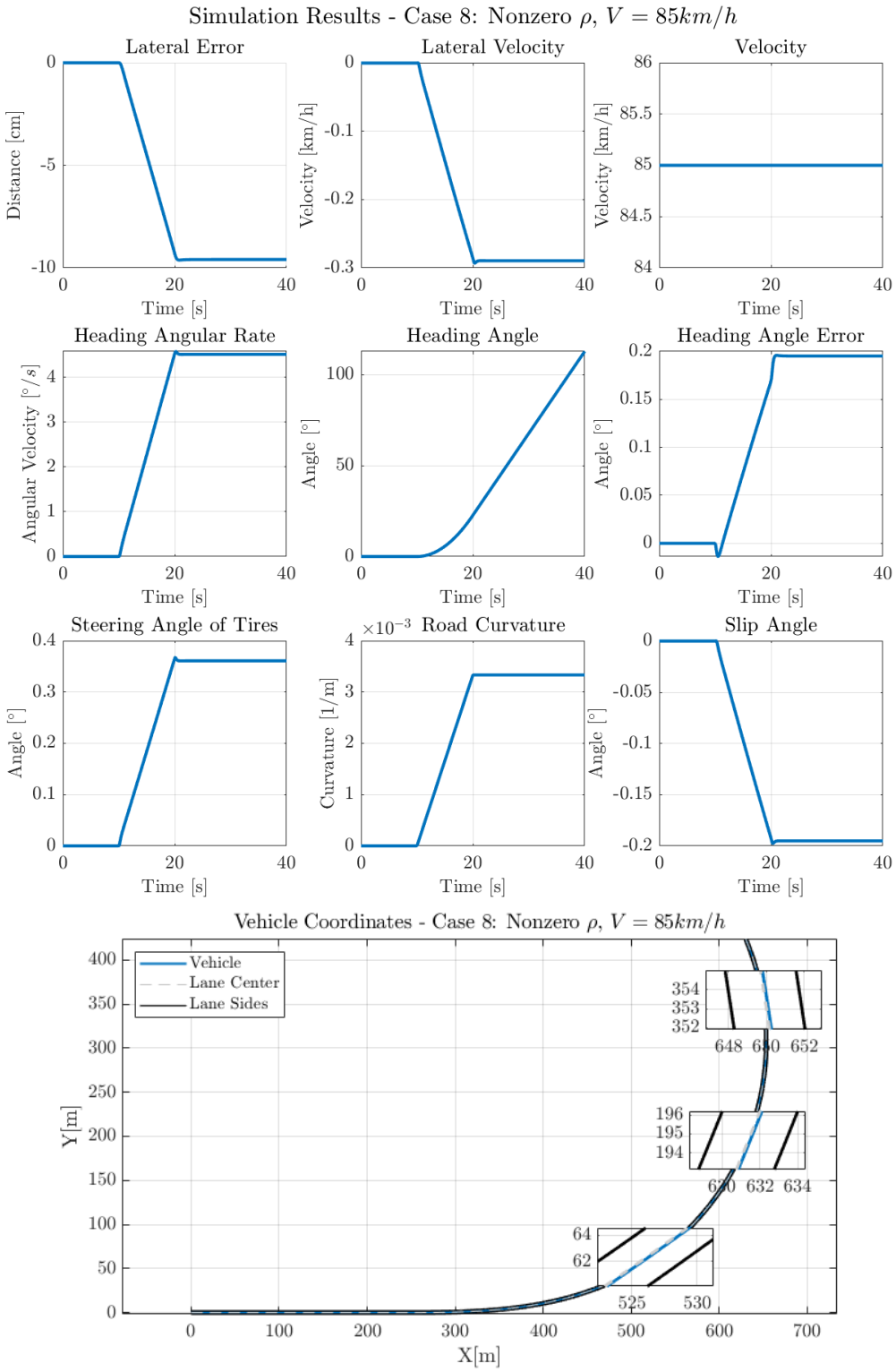

Figure 4.11: Simulation results for Case8

4.3.3.9 Case-IX: Nonzero Road Curvature, V = 120km/h

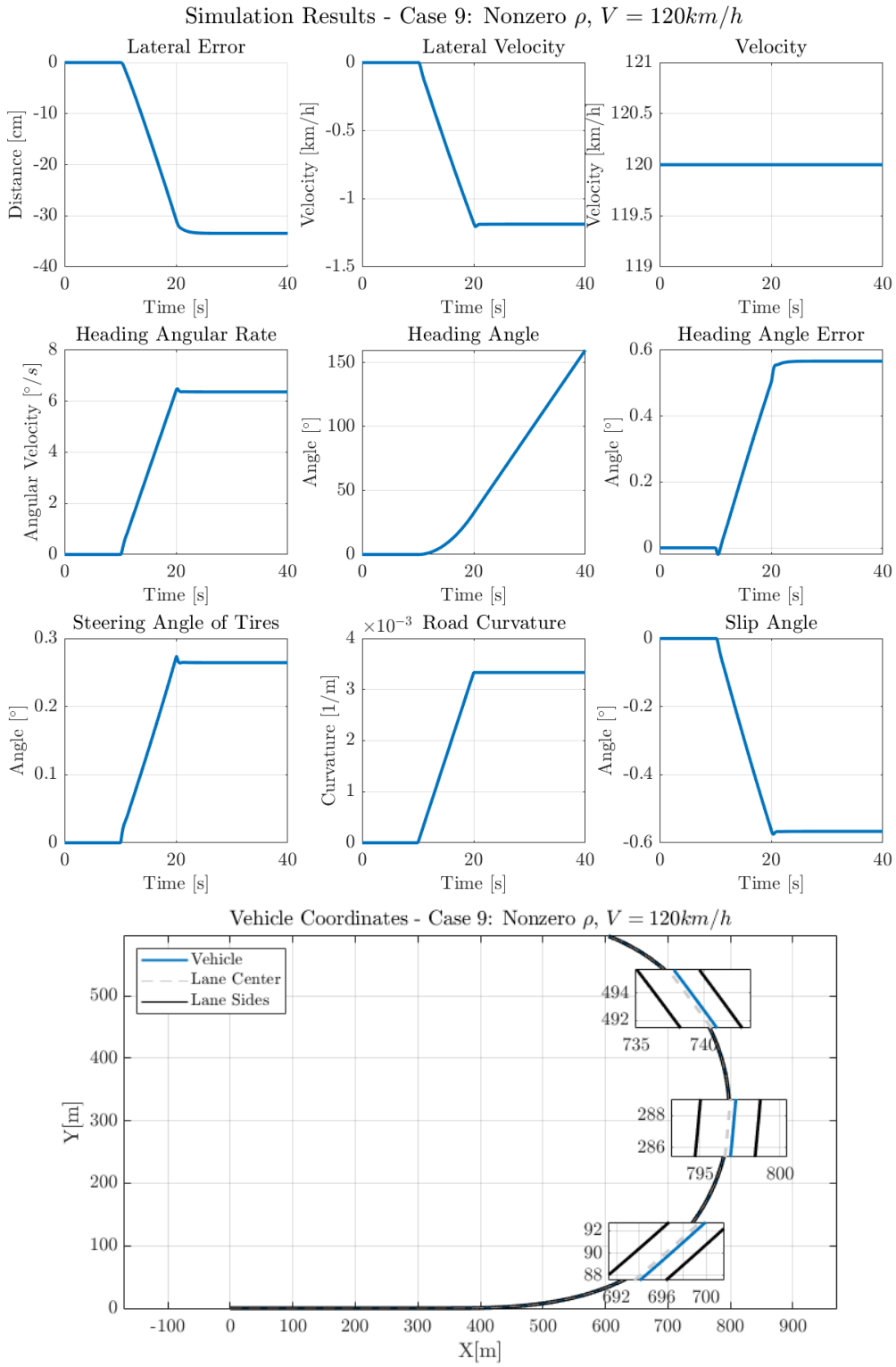

Figure 4.12: Simulation results for Case9

4.3.3.10 Case-X: Square Wave Acceleration and Curvature

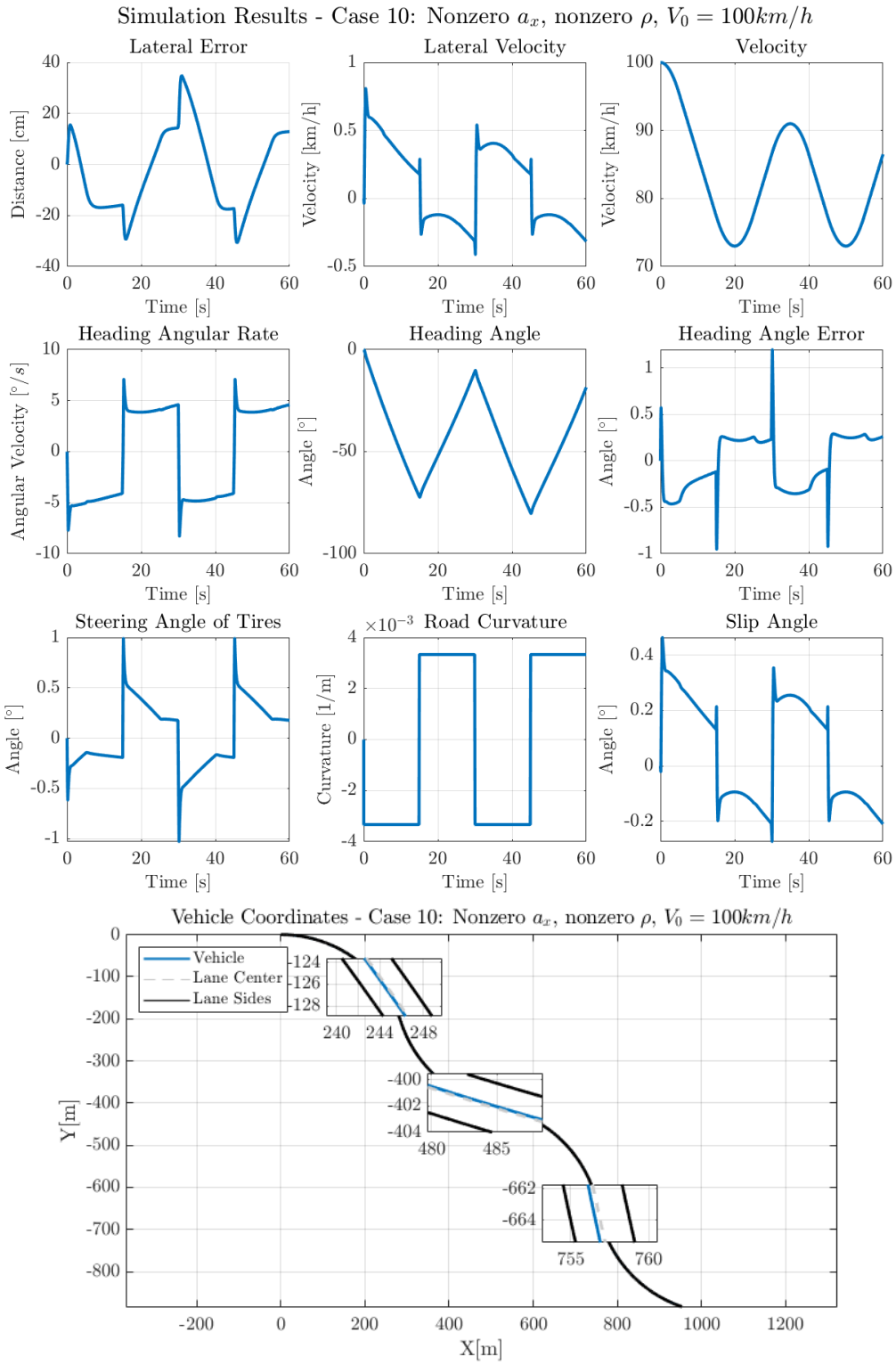

Figure 4.13: Simulation results for Case10

4.3.3.11 Case-XI: Acceleration and Curvature Profile #1

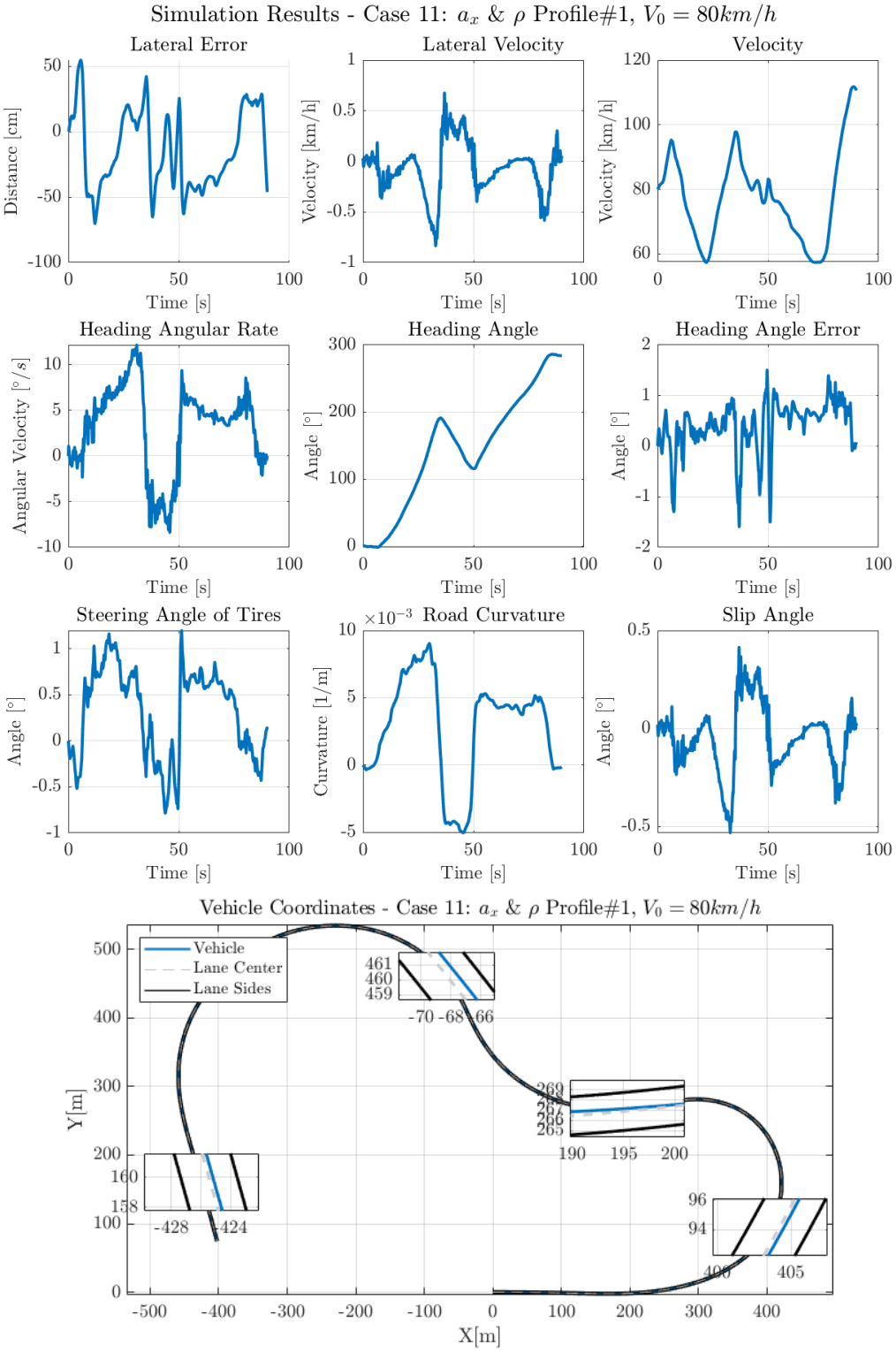

Figure 4.14: Simulation results for Case11

4.3.3.12 Case-XII: Acceleration and Curvature Profile #2

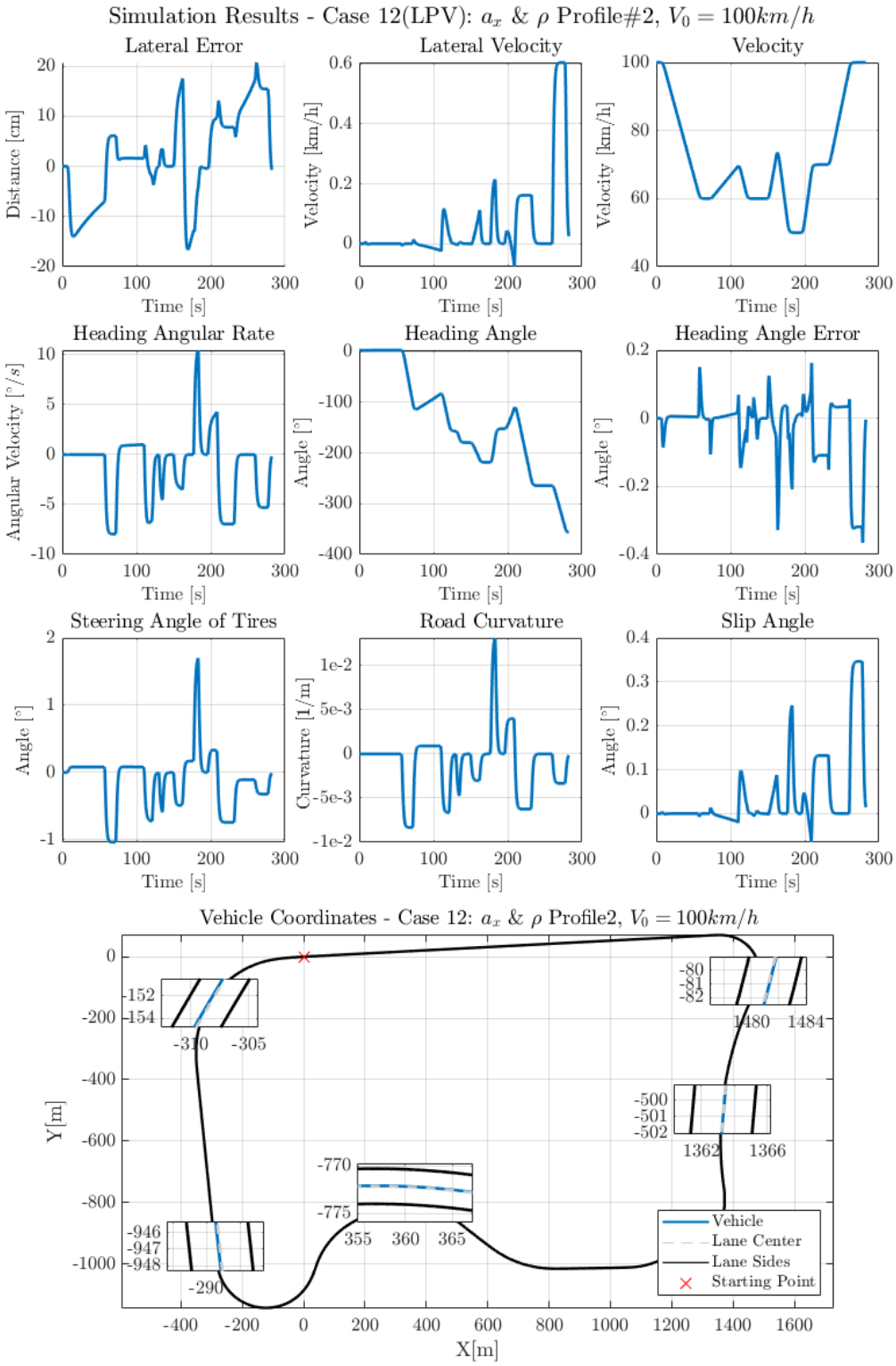

Figure 4.15: Simulation results for Case12

4.3.3.13 Controller Comparison

In this section, the designed LPV controller and the linear controller [63] given in the figure 4.3 will be compared. In this comparison, the same acceleration and road curvature profile utilized at the Case-12 will be used.

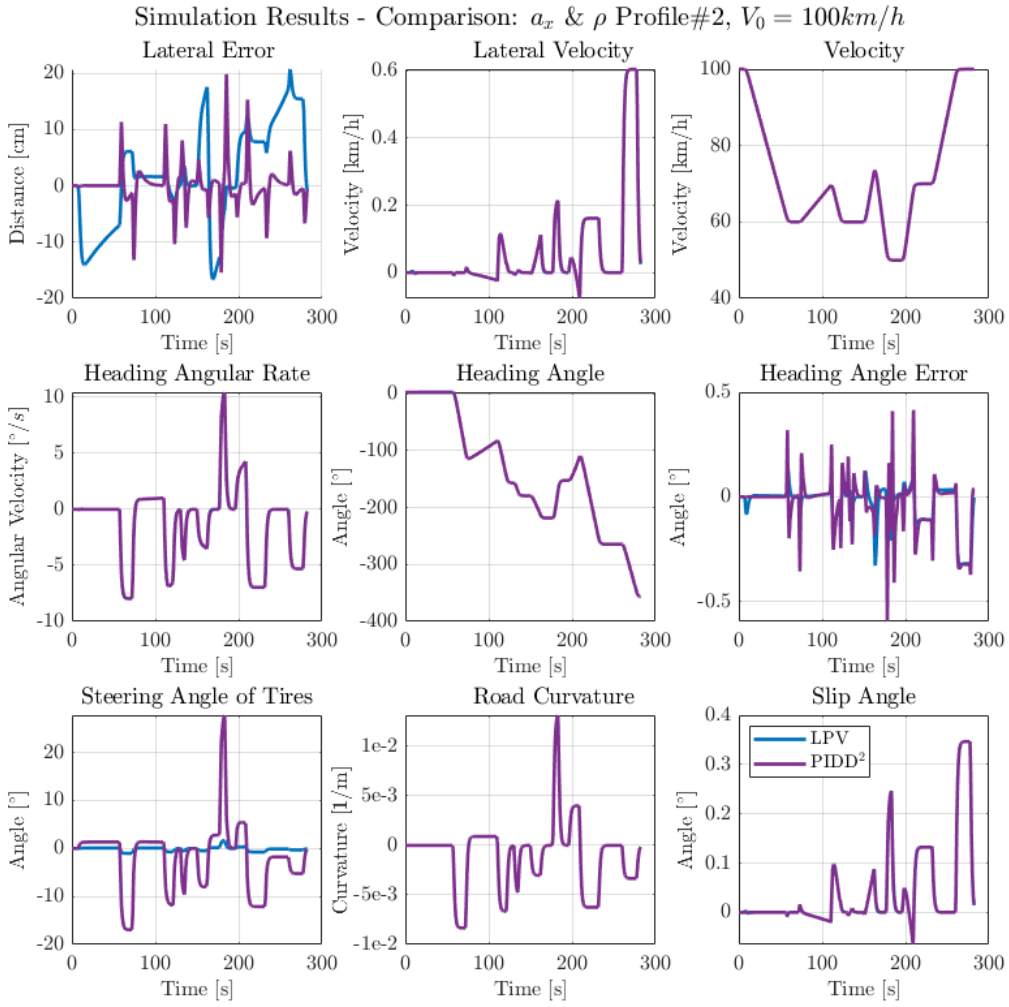

Figure 4.16: Simulation results for controller comparison with the same scenario used at Case12

As seen in the figures from Figure 4.4 to 4.6, the controller manages to eliminate the 1 meter initial lateral displacement error around 4 seconds while the vehicle is moving at the constant velocities (V_{min} and V_{max}) used in the controller design. To eliminate this error, the controller responds with a sudden change in the steering angle which causes a temporary heading error diminishing when the lateral displacement goes to zero. The similar situation occurs when $V = 85km/h$, which is important to check since the controller is designed only at vertex points (V_{min} and V_{max}).

From the Figures 4.7 to 4.9, the effect of 3 degrees initial heading angle error can be seen. The controller responds with a sharp turn resulting that the heading angle error makes an overshoot around 1 second and goes to zero around 3 seconds. The responses in 3 different velocities are roughly the same, except as the velocity increases, the maximum lateral error occurring in the manoeuvre increases as well, which is expected.

The effect of the road curvature on the performance of the controller is tested at different velocities with a step-wise curvature signal as seen in the Figures 4.10 to 4.12. When the road starts turning, some lateral displacement error begins to increase and controller responds to this error. Thus, the tire angle increases to some point such that the lateral distance stays at the same value. While the road is still turning, there is a steady state lateral displacement error which again increases with the velocity. The maximum value of this steady state error is around 35cm when the velocity is 120km/h and road curvature is 1/300m^{-1}. The value of the error will increase with a sharper turn but the chosen radius curvature for this speed is as safe as double since in the highway design handbook the suggested minimum curvature radius (for 0 elevation) is 600m in USA [64] and 565m in Turkey [65] for $V = 120km/h$.

In the Case-10 which is depicted in 4.13, there are some spikes in the response of the controller, which is due to the jumps in the road curvature signal. In this scenario, the velocity is changing with a profile similar to smoothed saw-tooth wave. Together these conditions result in a lateral error with a maximum value of 35cm and heading angle error reaches 1 degree at peaks. In the Case-11, the response of the controller in a more realistic velocity and curvature profile [9] is tested as shown in Figure 4.14. In this scenario the road turn radius drops to values as low as 100m and velocity

changes with relatively sharper peaks. As a result, the lateral error reaches values around 70cm at peak. The last profile used in the Case-12 has smoother transition between the velocity values but still the road curvature changes with sudden jumps as seen in Figure 4.15. As seen in the same figure, the lateral displacement error in this case has the maximum value of 20cm with a maximum heading error 0.35°.

In the last simulation, a PIDD^2 controller and the LPV controller is tested with the same acceleration and curvature profiles used in the Case12. As seen in the figure 4.16, the magnitude of the lateral displacement errors are similar except the PIDD^2 controller is faster to eliminate the displacement error. The LPV controller is better than the a PIDD^2 controller in terms of heading error elimination. As can be seen in the graphs, the maximum heading error for LPV controller is around $0.3°$ while in the PIDD^2 controller case it reaches a value up to $0.6°$. Another difference is the steering angles, although it seems that PIDD^2 controller leads to huge steering angles, this is due to the absence of the actuator model in the PIDD^2 controller simulation. For a better comparison, the gear ratio of 16.34 should be taken into account and with this scale the tire angles become similar. Although the responses are similar, the LPV controller achieves this performance with a second order actuator, while the PIDD^2 controller require an ideal actuator. If the actuator dynamics is added to the model, then the closed loop system becomes unstable. Another advantage is that the LPV controller guarantees stability in the existence of acceleration. Although in the simulation the PIDD^2 controller manages to keep the system stable, this does not necessarily state that it will in any acceleration.

CHAPTER 5

EXAMPLE APPLICATION-II: LAUNCH VEHICLE AUTOPILOT

This chapter is composed of two main sections. In the first section, using the linear model of the launch vehicle obtained in Chapter 3, LPV model will be obtained. Using this LPV model, autopilot for this system will be designed. In the second part of this chapter, the nonlinear model of the vehicle will be implemented on MATLAB/Simulink and both open and closed loop simulations with the Monte Carlo analysis will be performed.

Launch vehicles are generally aerodynamically unstable, have relatively slow servo actuators, and may exhibit non-minimum phase characteristics [18]. Hence, the controller for the launch vehicle, which is called "autopilot", is needed to make the rocket stable throughout the trajectory. The second most important mission of the autopilot is to follow the guidance commands.

Launch vehicles consists of different number of stages to increase the efficiency of the thrust. Generally, the first stage is the hardest to control due to the high dynamic pressure, aerodynamic instabilities, gust, turbulence etc. [66]. Hence, in this study, autopilot design and nonlinear simulations for 1^{st} stage will be conducted.

5.1 Autopilot Design for Launch Vehicle

5.1.1 LPV Model of the Launch Vehicle

To convert the linear system given in 3.56 into LPV form, the scheduling parameters must be decided. To simplify the controller design process, the least possible number of parameters should be chosen. For that reason, these parameters should be chosen

such that other varying parameters can be expressed as a function of this parameter. In the rocket case, many parameters are changing during the flight such as velocity, dynamic pressure, angle of attack, mass, thrust, inertia, center of gravity etc. In some applications [31], time is chosen as a scheduling parameter, however, due to the uncertainties in the thrust profiles, the profile that these parameters follow can severely change. To overcome this, velocity V is chosen as scheduling parameter. Hence, if the thrust is lower than nominal, the velocity would be less than nominal too. Same principle applies to the other parameters as well, resulting in a lower sensitivity on the uncertainties. This situation can be seen in Figure 5.2 and 5.3 that corresponds to the %10 uncertainty in the thrust duration given in Figure 5.1.

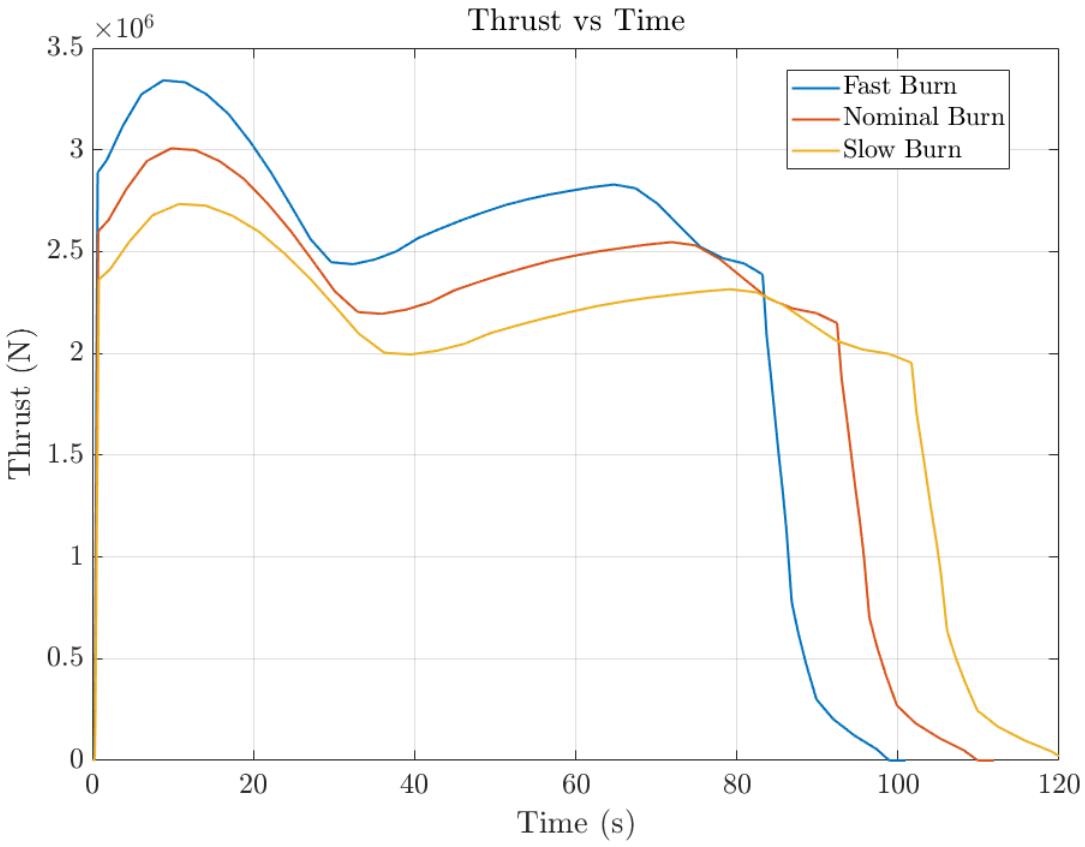

Figure 5.1: Thrust vs time graphs of the rocket (at 1st stage) with burn-time uncertainties

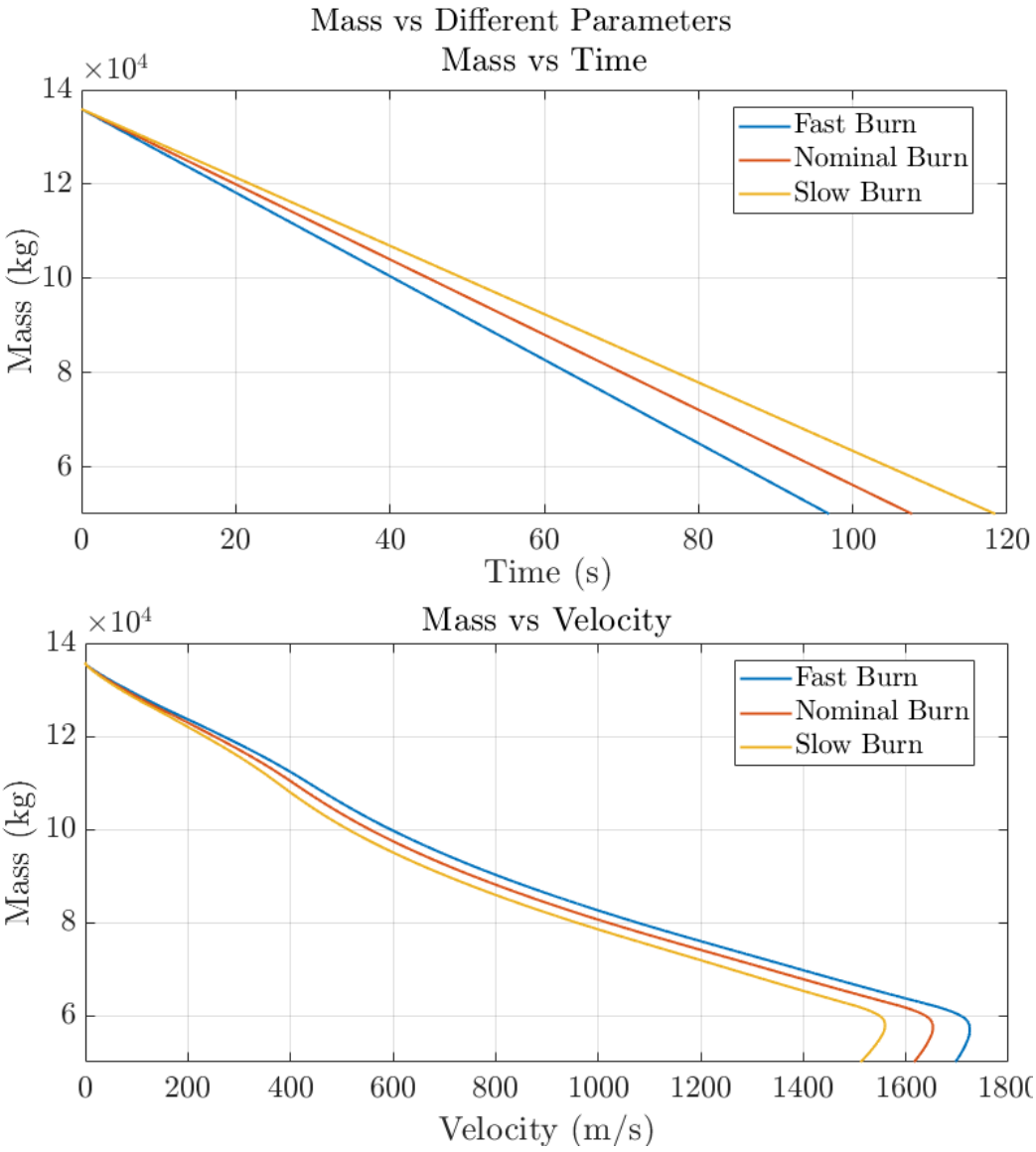

Figure 5.2: Mass vs time (up) and mass vs velocity graphs of the rocket (at 1st stage) with burn-time uncertainties

As seen in Figures 5.2 and 5.3, the mass and altitude changes in a smaller distribution for the same velocity than for the same time. Which proves the velocity being better than the time as a scheduling parameter in terms of being robust against thrust uncertainty. One point is that when the thrust ends, the velocity starts decreasing, which may lead to multiple parameter values for the same velocity. But since the thrust is ended, the controller would not be effective anymore meaning that it does not change anything in terms of control performance.

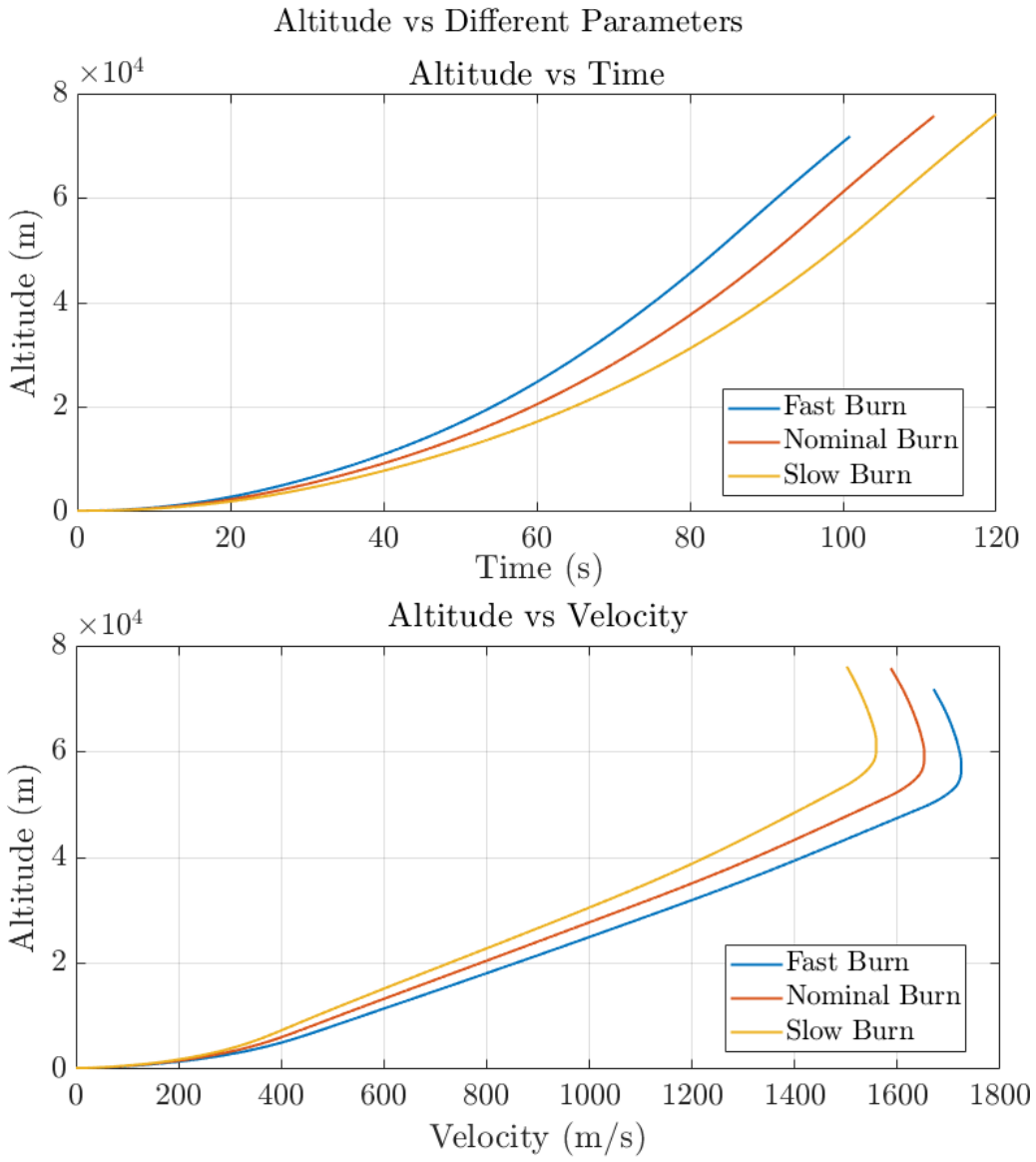

Figure 5.3: Altitude vs time (up) and altitude vs velocity graphs of the rocket (at 1st stage) with burn-time uncertainties

With the scheduling parameter selected, the next objective is to transform the linear model into LPV form. The linear model for the dynamics of the launch vehicle is obtained in the equation 3.56 in the following configuration.

$$\begin{aligned} \dot{x_p} &= A_p x_p + B_{pu} u + B_{pw} w \\ y_p &= C_p x_p \end{aligned} \tag{5.1}$$

The state dynamics of the rocket is expressed in the equation 3.56, however the output y_p is not defined. The output of the plant is chosen assuming that the angular velocity of the actuator angle $\dot{\delta_e}$ is not measurable as follows.

$$y_p = \begin{bmatrix} \alpha \\ q \\ \delta_e \\ \Delta\theta \end{bmatrix}, \quad C_p = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.2}$$

Similar to the process done in the Section 4.2.1, a trajectory curvature model whose dynamics is given in the equation 5.3 will be utilized in the design. The impulse response of this model contains the information regarding the expected curvature in the following way. The peak value of the impulse response corresponds to the maximum value of the expected trajectory curvature and the time to peak corresponds to the expected peak time of the curvature change.

$$\Sigma_w = \begin{cases} \dot{x_w} = A_c x_w + B_c x_w \\ y_w = C_c x_w \end{cases} \tag{5.3}$$

where $x_w = \begin{bmatrix} \rho_{pitch} & \dot{\rho_{pitch}} & \ddot{\rho_{pitch}} \end{bmatrix}^T$ and $y_w = \rho_{pitch}$ with the state space matrices.

$$A_c = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -a_{c0} & -a_{c1} & -ac2 \end{bmatrix}, \quad B_c = \begin{bmatrix} 0 \\ 0 \\ K_c \end{bmatrix}, \quad C_c = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \tag{5.4}$$

By completing the system dynamics with the curvature model and defining the output vector, the only vector to define before synthesising the controller is the performance vector. Throughout the flight, the aerodynamical loads are proportional with the angle of attack, whose dynamics depend on the angular rate q proportionally. Hence, these states are important in terms of the satisfying mechanical design requirements. The attitude error $\Delta\theta$ is important in terms of the trajectory following, hence it should be included in the performance vector as well. Finally, to minimize the actuator loads, the thrust deflection angle δ_e is added to performance vector whose final form is given in the equation below.

$$z = \begin{bmatrix} \alpha \\ q \\ \delta_e \\ \Delta\theta \end{bmatrix} \tag{5.5}$$

With this definition of the performance vector, relation between the performance and the state vector becomes

$$z = D_{zx_p} x_p, \quad D_{zx_p} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.6}$$

Combining the state space matrices of the launch vehicle model given in equation 3.56, the trajectory curvature model given in equation 5.3 and 5.4, with the matrices for output and performance vectors given in equations 5.2 and 5.6, the complete launch vehicle model becomes as follows.

$$\Sigma(V) : \begin{cases} \dot{x} = Ax + B_u u + B_w w \\ z = C_z x, \quad y = C_y x \end{cases} \tag{5.7}$$

where

$$x = \begin{bmatrix} x_p \\ x_w \end{bmatrix}, \quad y = \begin{bmatrix} y_p \\ y_w \end{bmatrix}$$

$$A = \begin{bmatrix} A_p & B_{pw} C_c \\ 0 & A_c \end{bmatrix}, \quad B_u = \begin{bmatrix} B_{pu} \\ 0 \end{bmatrix}, \quad B_w = \begin{bmatrix} 0 \\ B_c \end{bmatrix} \tag{5.8}$$

$$C_y = \begin{bmatrix} C_p & 0 \\ 0 & C_c \end{bmatrix}, \quad C_z = \begin{bmatrix} D_{zx_p} & 0 \end{bmatrix}$$

Note that during the flight, mass m, dynamic pressure q_∞, thrust T, inertia I_y, aerodynamic moment and force coefficients $C_{M_{y\alpha}}$ & C_{N_α} change. Some of them changes nearly linearly while others have nonlinear characteristics. These parameters can be

approximated as a function of the velocity. This process is done similar to the lateral LPV model of the vehicle as explained in the 4 through the parameter θ which changes between -1 and 1 corresponding to velocities V_{min} and V_{max}. Using the scheduling parameter θ and the launch vehicle model given in equation 5.7, the polytopic LPV model of the system becomes:

$$\Sigma(\theta) : \begin{cases} \dot{x} = \sum_{i=1}^{2} \eta_i(\theta) \left(A_i x + B_i^u u + B_i^w w\right) \\ z = \sum_{i=1}^{2} \eta_i(\theta) C_i^z x, \quad y = C_y x \end{cases} \tag{5.9}$$

where the state space matrices are given by:

$$\begin{aligned} A_1 = A(\theta_{min}), \quad B_1^u = B_u, \quad B_1^w = B_w, \quad C_1^z = C_z(\theta_{min}) \\ A_2 = A(\theta_{max}), \quad B_2^u = B_u, \quad B_2^w = B_w, \quad C_1^z = C_z(\theta_{max}) \end{aligned} \tag{5.10}$$

5.1.2 Autopilot Design

In the design process, similar steps given in the controller design Chapter 4.2.2 will be followed. Firstly, the parameters for the trajectory curvature model will be decided. After that, the scheduling parameter and its derivative limits will be set. And finally, using the numerical values of all the required matrices, the LMI given in the theorem will be solved using YALMIP toolbox and feedback gains will be obtained.

In this work, considering the first manoeuvre of the rocket, the minimum trajectory curve radius of $300m$ will be considered, hence the maximum expected curvature is $1/300 = 3.3 \times 10^{-3} m^{-1}$. The impulse response peak time is chosen as 12.8 second. Using these values, the parameters for the trajectory curvature model is decided as given in the table 5.1.

The velocity and acceleration limits that will be used in the design process are chosen as follows.

Table 5.1: Numeric values of the trajectory curvature model

Parameter Name	**Value**
a_{c0}	0.5
a_{c1}	15
a_{c2}	100
K_c	0.07

$$\begin{aligned} V_{min} &= 100km/h, \quad V_{max} = 1600km/h \\ a_{min} &= 0m/s^2, \quad a_{max} = 35m/s^2 \end{aligned} \tag{5.11}$$

To incorporate a priority among the states, the coefficient in the weighting matrices can be set different than 1. In this work, the attitude error $\Delta\theta$ is considered as slightly more important and hence the weight matrix is chosen as follows.

$$W_z = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 5 \end{bmatrix} \tag{5.12}$$

Apart from these constant values, there are other constant parameters needed to be set in order to solve the LMI defined in the theorem which are explained in Section 4.2.2. These parameters are chosen following the similar steps given in Section 4.2.2 as follows.

$$\alpha = 0.001, \quad \gamma = 20, \quad \epsilon = 0.087 \tag{5.13}$$

To solve the LMI given in Theorem 4.1 with these parameter values, MOSEK solver [62] with the YALMIP toolbox [61] is used and following gain values are obtained (K_1 corresponds to $V = V_{min}$ and vice versa).

$$\begin{aligned} K_1 &= \begin{bmatrix} 0.0767 & -0.7967 & 0.0530 & -0.3781 & 623.5157 \end{bmatrix} \\ K_2 &= \begin{bmatrix} 0.0635 & -1.3984 & 0.0806 & -0.5455 & 987.5879 \end{bmatrix} \end{aligned} \tag{5.14}$$

5.2 6 DoF Simulation of the Launch Vehicle

The complete nonlinear launch vehicle model derived in Chapter 3.2.1 will be implemented in the Simulink software to validate the controller. Apart from this 6 DoF model of the rocket, imperfections such as delays, uncertainties and disturbances will also be modelled.

5.2.1 Implementation of the Nonlinear Simulation

The simulink model of the launch vehicle will consist of 2 main blocks: rocket model and environment model as shown in Figure 5.4. While the former incorporates the algorithms and hardware on the rocket, the latter is constructed to model the dynamics of the rocket along with the environmental affects such as wind.

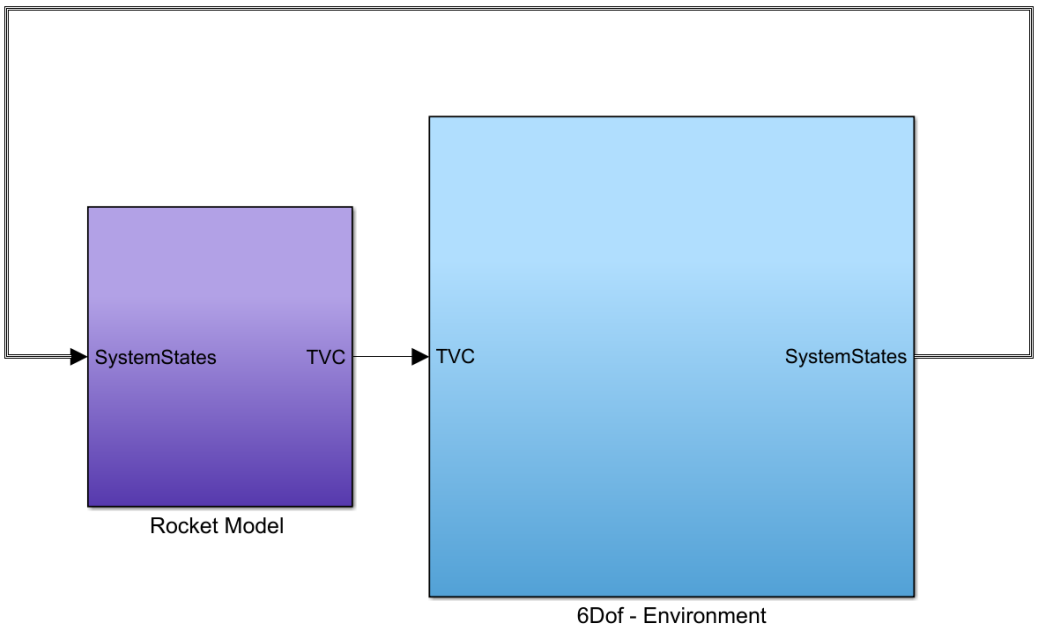

Figure 5.4: Implementation of the nonlinear launch vehicle model on Simulink, main blocks

The rocket model will be composed of 3 main parts namely flight computer, sensor models and actuator as shown in Figure 5.5. The flight computer runs the guidance, navigation and control (GNC) algorithms in the configuration depicted in Figure 5.6.

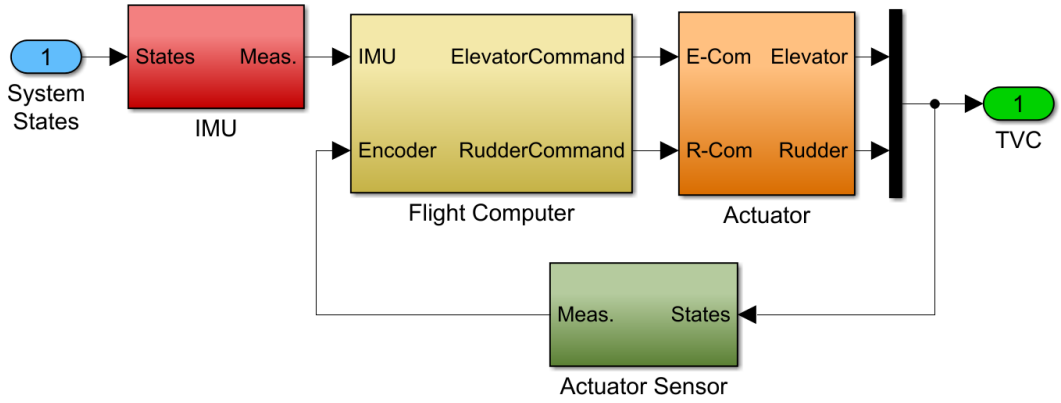

Figure 5.5: Inside of the "Rocket Model" block given in the figure 5.4

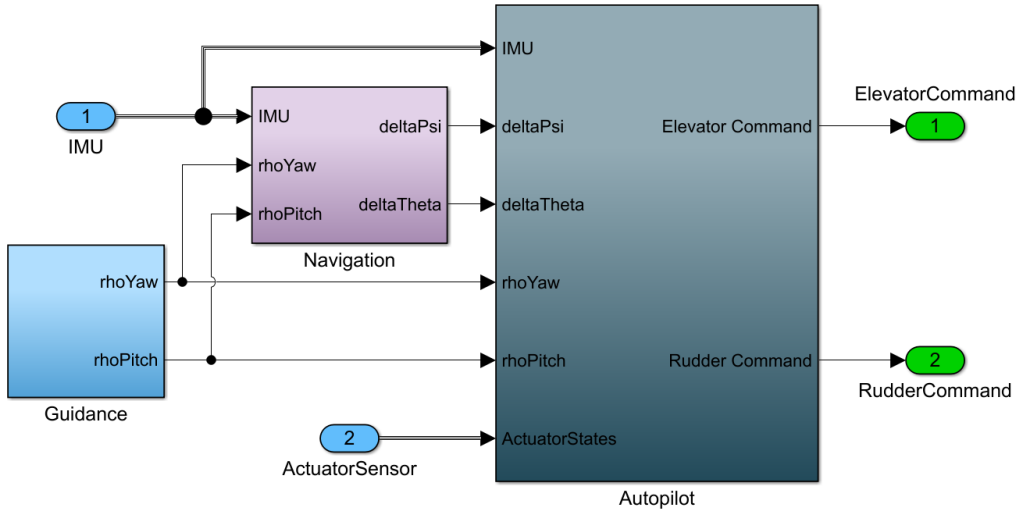

Figure 5.6: GNC algorithms inside of the "Flight Computer" block given in the figure 5.5

In a typical application, the guidance algorithm generates the reference commands for the controller. In this work, it is generating the trajectory curvature information. Note that there are various approaches for guidance algorithm design in the literature, however, in this study an open loop trajectory curvature in the shape of half sine wave will be used.

The second algorithm is the navigation. Normally it is responsible for detecting the current states of the rocket such as attitude, position, velocity, acceleration etc. In

this work, it is modelled such that it calculates the attitude error in the pitch and yaw planes using the trajectory curvature information coming from the guidance block and feeds this delta-angle information into the controller (autopilot) algorithm.

Finally, the autopilot uses these error state and disturbance information coming from navigation and guidance algorithms along with the sensor outputs to generate the output vector. By multiplying this output with the feedback gains, the actuator commands to be sent to the actuators are obtained.

Apart from the algorithms running inside the flight computer, there are sensor models which are added for the modelling of inertial measurement unit (IMU) and the actuator sensors dynamics. In this study the dynamics of these sensors are implemented as a pure delay.

The final block inside the rocket model is the actuator. It is utilized to model the dynamics of the thrust vector control (TVC) actuator. The linear actuator dynamics are given in the equation 3.39, however a practical actuator has non-ideal properties such as angle and rate limit. Hence, in this work the actuator is modelled to have $\pm 5^{\circ}$ angle limit and $20^{\circ}/s$ angular rate limit.

The second main block of the model is the "6 DoF - Environment" block as shown in Figure 5.4. This block is utilized to simulate the laws of physics. It consists of 4 sub-blocks: system parameters, force & moment calculations, 6 DoF calculations and atmosphere model as depicted in Figure 5.7.

System parameters block generates the necessary system parameters such as mass, inertia, thrust, center of gravity. These values are generated beforehand and saved in the workspace. Mass value is obtained such that it starts from full stack mass to dry mass and is assumed to change linearly as a function of burn-time. Inertia and center of gravity are also assumed to change linearly from the full configuration the 1st-stage-burned configuration as a function of time. The inertia values are obtained assuming a uniform density of the rocket. Note that, if the burn time is short, then the time change rate of these parameters are high and vice versa. The values of these parameters are obtained using different sources. For example, the mass and the dimension information are obtained from the user manual of the VEGA [54].

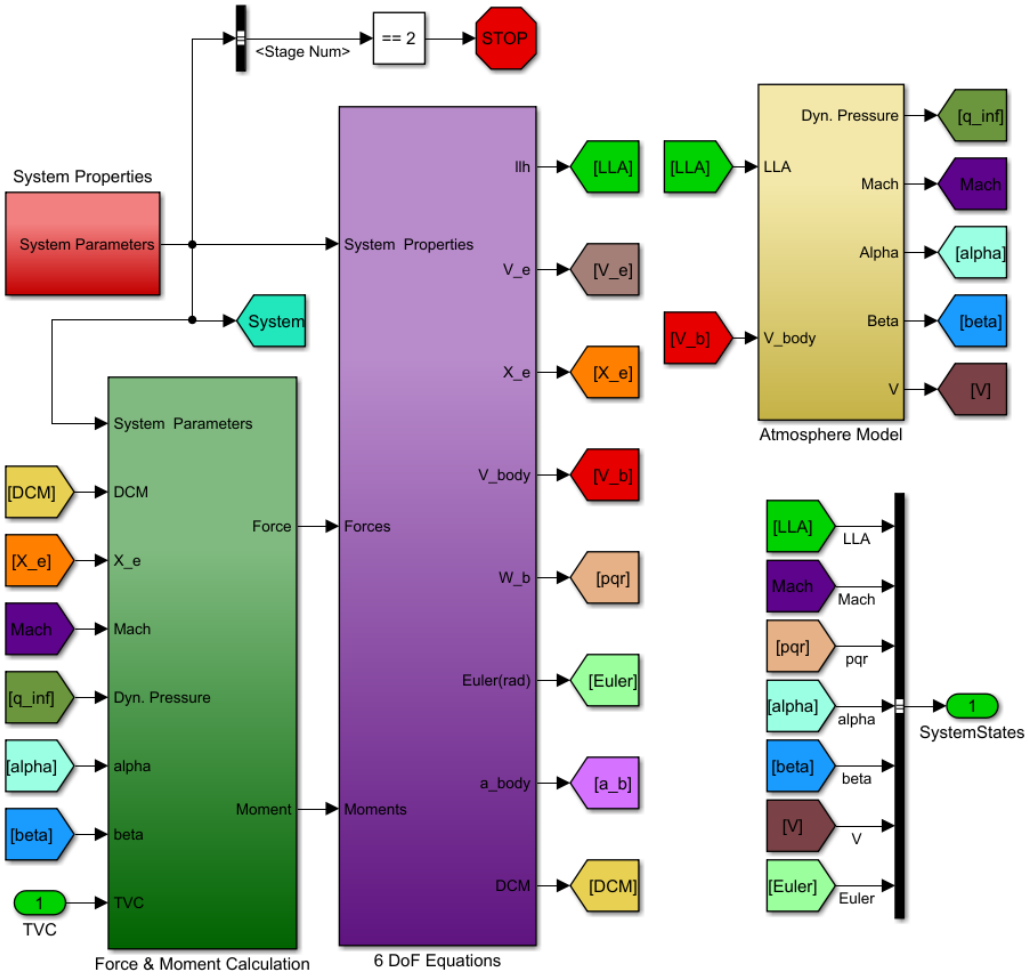

Figure 5.7: Inside of the "6Dof - Environment" block given in the figure 5.5

Using these dimension and mass data with the uniform density assumption, the launch vehicle created in the CATIA environment to calculate both the CG and inertia values for full and completely burned configurations. The thrust data is obtained using the values given in the manual [54] and the approach given in [67]. The values of the complete system parameters are given in table 5.2, while the obtained thrust data is given in 5.1 with $\pm 10\%$ burn-time uncertainties.

The second block is force & moment calculation block which produce the force and moments that launch vehicle experience. In this block, aerodynamics, gravity and thrust force and moments along with the tower dynamics are modelled as shown in Figure 5.8. Aerodynamic forces and moments are calculated using the equations given in 3.33 and 3.26 respectively. The aerodynamic coefficients C_M and C_N change

Table 5.2: Numeric values of the system parameters of VEGA launch vehicle used in the simulation

Name	**Value @ t=0**	**Value After 1st Stage is Burned**
Mass	135926kg	48216 kg
Inertia around x axis	214884 kgm^2	49490 kgm^2
Inertia around y axis	5243000 kgm^2	1521000 kgm^2
CG distance from nose	21.56 m	15.68 m

as a function of angle of attack and Mach number as shown in Figure 3.5, 3.4, 3.7 and 3.6. Interpolation block is used to calculate these aerodynamic coefficients depending on the current values of angle of attack and Mach number. Note that the moment coefficient C_M is generated at nose and since the rotations are defined around the CG, this moment coefficient is carried to the CG. The dynamic pressure q_∞ used in the calculation of aerodynamic forces and moments are obtained in the atmosphere block, which will be mentioned in the following paragraphs.

The thrust forces and moments are calculated using the equations given in 3.24 and 3.34 respectively. Note that the thrust moment component around the x axis is neglected with the assumption of thrust pivot point lying on the body-x vector.

The gravity force is calculated using the gravitational acceleration and mass. The gravitational acceleration between two masses is directly proportional to the product of their masses and inversely proportional to the square of the distance between their center of mass. To find the distance from the rocket to the center of the earth, the position vector $\boldsymbol{p} = p_x\hat{i} + p_y\hat{j} + p_z\hat{k}$ in the ECI coordinates can be utilized as proposed in [68] as follows.

$$g = -\frac{GM}{\|\boldsymbol{p}\|^3}\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \tag{5.15}$$

where $GM = 3.986004418 \times 10^{14}$m^3/s^2 is the Earth-mass gravitational constant. Note that the gravitational acceleration depends on the length of $\boldsymbol{p}$ because of the inverse square law and the latitude angle of $\boldsymbol{p}$ due to the non-spherical shape of the earth. The

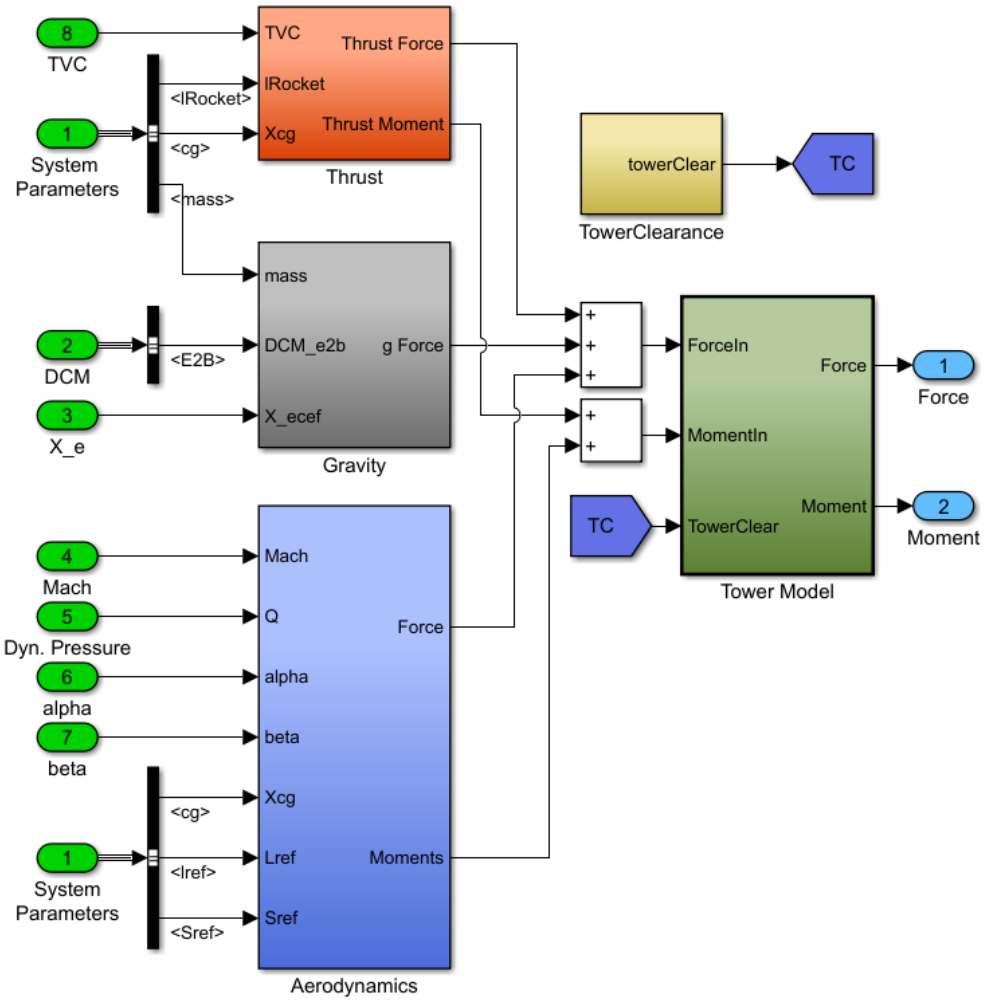

Figure 5.8: Inside of the "Force & Moment Calculation" block given in the figure 5.7

latter is negligible if the rocket moves near the surface of the Earth, however this is not the case in this work. Hence, to account for the effect of the Earth's oblateness on g, the equation 5.15 is modified by replacing p_x, p_y and p_z by $\overline{p}_x$, $\overline{p}_y$ and $\overline{p}_z$ [68] from

$$\overline{p}_x = p_x\left[1 + 1.5J_2\left(\frac{r_E}{\|\boldsymbol{p}\|}\right)^2(3 - 5\sin^2\lambda)\right] \tag{5.16}$$

$$\overline{p}_y = p_y\left[1 + 1.5J_2\left(\frac{r_E}{\|\boldsymbol{p}\|}\right)^2(1 - 5\sin^2\lambda)\right] \tag{5.17}$$

$$\overline{p}_z = p_z\left[1 + 1.5J_2\left(\frac{r_E}{\|\boldsymbol{p}\|}\right)^2(1 - 5\sin^2\lambda)\right] \tag{5.18}$$

Table 5.3: The numerical values of the parameters used to generate aerodynamic coefficients

Name	**Value**
Mach	$0.3, 0.6, 0.9, 1.2, 1.5, 1.8, 2, 3, 4, 5, 6, 7, 8, 9, 10$
AOA (°)	$-9, -8, -7, -6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9$
Altitude (km)	$0, 1.1, 2.2, 3.6, 5.4, 7.5, 8.9, 16.7, 33, 53, 64, 74, 88, 99, 110$
L_{Ref} (m)	3
S_{Ref} (m^2)	7.069

where $J_2 = 1.08263 \times 10^{-3}$ is the gravitational harmonic constant of the Earth, $r_E = 6378137$m is the Earth's equatorial radius and λ is latitude angle calculated from $\sin^{-1}(p_x/\|p\|)$.

With the gravity force explained, the last force and moment components to obtain are aerodynamically generated forces and moments. As explained in the previous sections, the aerodynamical force and moment coefficients are calculated using the Missile DATCOM software. These coefficients are plotted in Figure 3.4, 3.5, 3.6, 3.7 and calculated for discrete values of angle of attack and mach values which are shown in the table 5.3 along with other parameters used in this process.

While implementing the force and moments that the rocket experiences, the tower model needs to be considered. Towers are used to hold the rocket while thrust force is building up against the gravity. If there is not a tower at the start, the gravity force will be bigger than the thrust and the rocket would go down and crash. To model the effect of the tower, "Tower Model" block is added inside the "Force & Moment Calculation" block as shown in Figure 5.8. This block ensures that no force or moment is applied to the rocket until the release. The release action is decided inside the "Tower Clearance" block by time such that the thrust force gains some value.

With completing the force and moment calculations, the next block is the 6DoF dynamics block labelled as "6 DoF Equations" and shown in Figure 5.7. In this block, the 6DoF equation of motion block "Custom Variable Mass 6DOF ECEF (Quaternion)" of MATLAB is utilized and is set to custom variable mass mode since the mass and inertia of the launch vehicle changes with time. This block has 4 inputs,

force, moment, mass and inertia. The force and moments are in the body frame and they are obtained above. Mass and inertia values come from the system properties block. The outputs of this block includes velocity and position (both cartesian and LLA) in ECEF frame, Euler angles, Direction Cosine Matrices (DCM) between different frames, body translational and angular velocities, acceleration etc.

Apart from the 6DOF equations of motion, gusts are also implemented in this block. They are important disturbances and may severely effect the stability of a rocket. To model the gust, the widely-used half sine gust model of the NASA will be used [69]. In this approach, the gust profile is chosen in the form of a half sine wave. The magnitude and the length of the gust corridor is varying. In this work, the gusts are applied on the pitch axis, hence the velocity of the gust is added to w, which is the velocity of the rocket in the body-z axis.

The last block given in Figure 5.7 to examine is the "Atmosphere Model" block. In this block, "NRLMSISE-00 Atmosphere Model" of the MATLAB is used to implement the mathematical representation of the 2001 United States Naval Research Laboratory Mass Spectrometer and Incoherent Scatter Radar Exosphere atmosphere model [70]. This model is capable of calculating the temperature and the air density from surface to lower exosphere (0 to 10^6 meters) according to the MATLAB documentation. Then, these values are used to calculate the speed of velocity which is necessary to obtain the Mach values and air density ρ which is utilized to calculate the dynamic pressure q_∞ as follows.

$$q_\infty = \frac{1}{2}\rho V^2 \tag{5.19}$$

Also, the angle of attack α and side slip angles β are calculated in this block as follows.

$$\alpha = \tan^{-1}\Big(\frac{w}{u}\Big), \quad \beta = \sin^{-1}\Big(\frac{v}{|\boldsymbol{V}|}\Big) \tag{5.20}$$

5.2.2 Open Loop Simulation of the Launch Vehicle

As stated earlier, the aerodynamic moment coefficient at the CG has a negative derivative with respect to AOA as shown in 3.8, meaning that the dynamics of the launch vehicle is unstable. This can be shown by obtaining the transfer function from thrust deflection angle δ to angular velocity q in the pitch plane. For this purpose, the transfer function using the system properties at 1.2Mach is obtained as follows:

$$G(s) = \frac{q(s)}{\delta(s)} = \frac{5.3014(s + 0.006841)}{(s + 0.5962)(s - 0.5929)} \tag{5.21}$$

As can be seen in the equation 5.21, the system has positive poles and is hence unstable. Note that this unstability occurs due to the aerodynamics. Hence, to get the open loop simulation, the aerodynamic moments needs to be eliminated. For this purpose, the aerodynamic moments are multiplied with 0 and simulation is run. Note that since there is no controller, the rocket does not make any manoeuvre, hence it will go directly up since the initial theta angle is 90°. The simulation is ended when the 1st stage is separated, which is modelled to happen after 2s elapsed before the end of the thrust. The simulation is run with the gust properties given in the table 5.4. The results are depicted in the figures from 5.9 to 5.14.

Table 5.4: Values of the gust parameters used in the open loop simulation

Gust Number	**Gust Time**	**Gust Duration**	**Gust Magnitude**
Gust-1	10	3	7
Gust-2	40	3	5
Gust-3	70	3	3

From the graph given in Figure 5.9, it can be seen that the longitudinal acceleration has a similar shape to the thrust curve as expected, which can be seen in Figure 5.1. Towards the end, the acceleration goes to negative numbers indicating that the thrust is coming to end and gravity becomes stronger. From the velocity graph given in Figure 5.10, the rocket speeds until the thrust ends and starts do decline due to the gravity. The Mach number and the velocity has different envelope which is caused by the nonlinear change in the speed of sound.

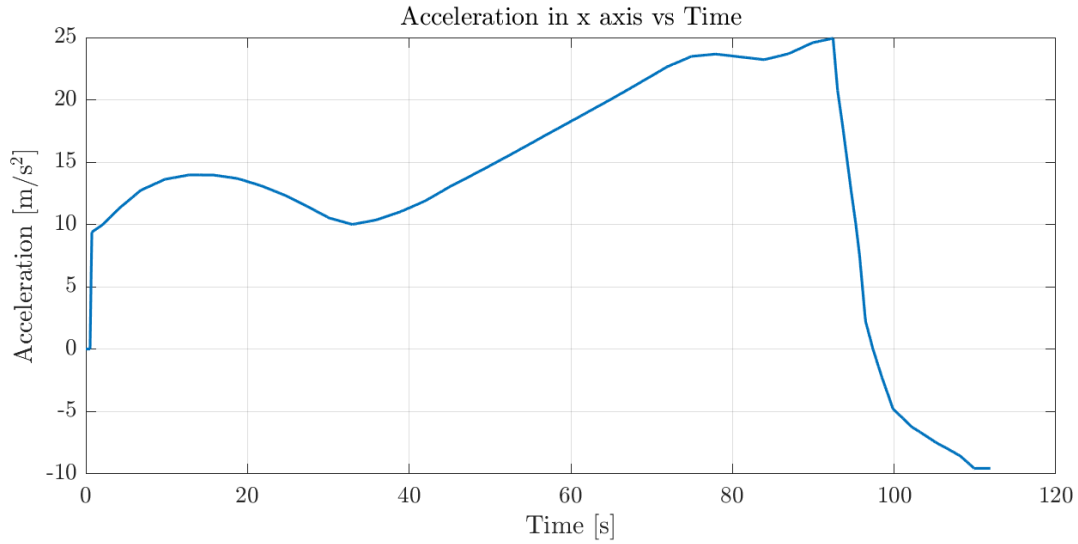

Figure 5.9: Open loop simulation results: Acceleration in x axis graph

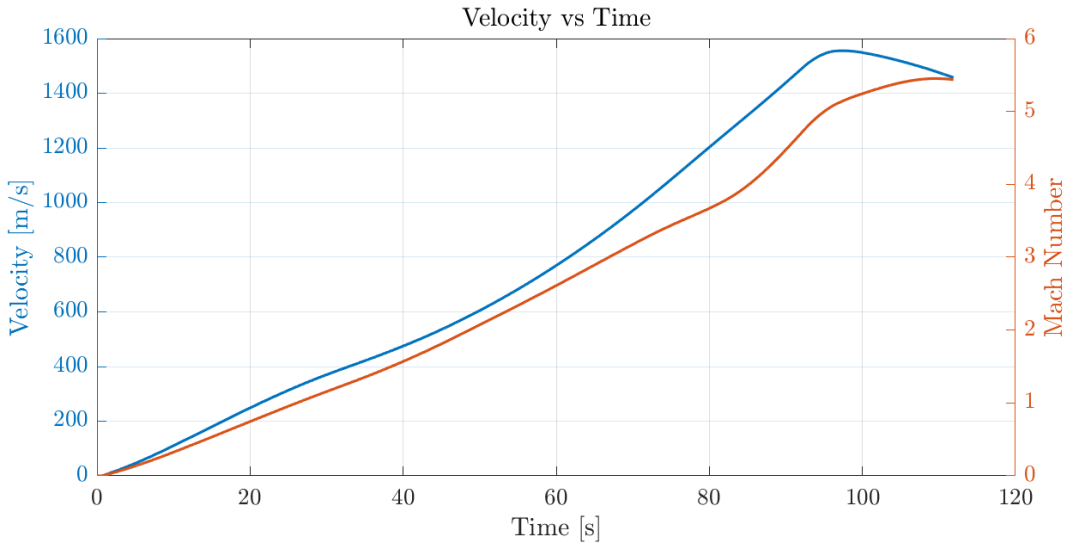

Figure 5.10: Open loop simulation results: Velocity magnitude graph

The rocket reaches the altitude of around 90km in this simulation as seen in Figure 5.11 and experiences the maximum dynamic pressure around 10km altitude. This is important since the rocket experiences the maximum aerodynamic load at this point.

The angle of attack and side slip angles can be seen in Figure 5.13. The half sine waves shapes are due to the gust. The AOA is 0 if there is no gust since the rocket is going directly up, meaning that it does not gain any velocity in the body-z axis. However it gains some velocity in the body-y axis, which is due to gravity. As a result

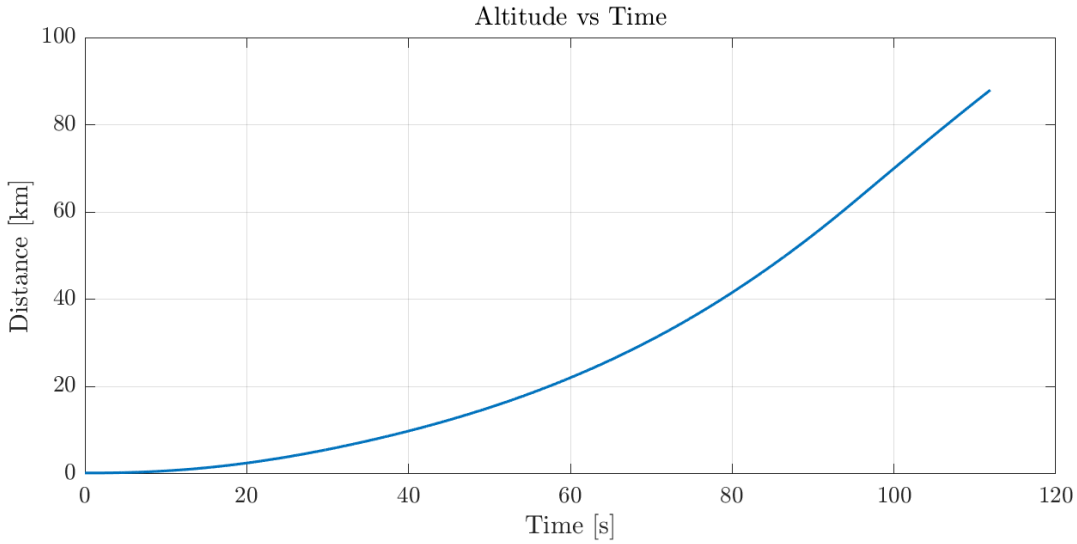

Figure 5.11: Open loop simulation results: Altitude graph

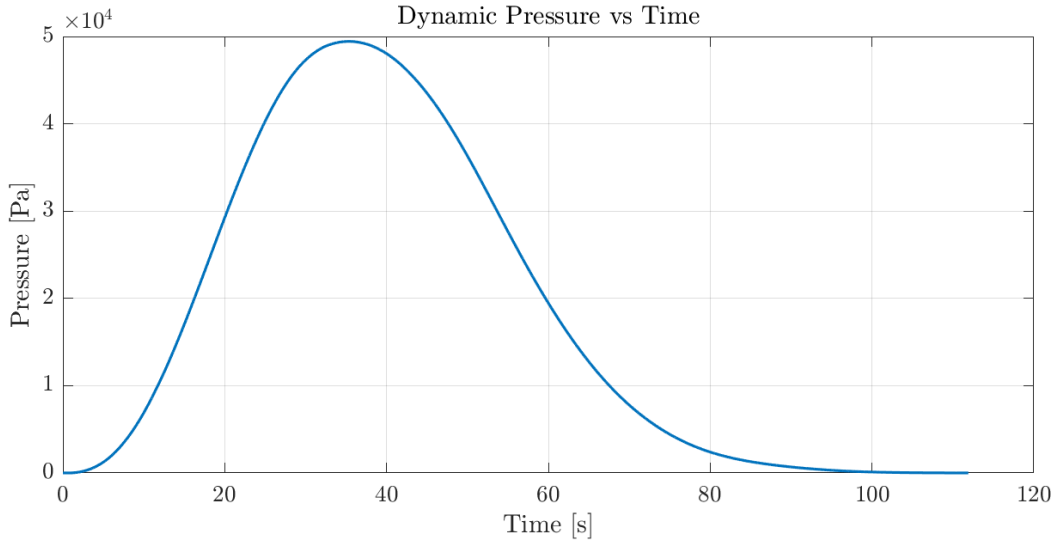

Figure 5.12: Open loop simulation results: Dynamic pressure graph

the side slip angle of the launch vehicle does not stay at zero. Note that although theta angle is 90° as seen in Figure 5.14, it does not mean the gravity is completely on x axis, since the z axis of the NED frame does not necessarily intersect the center of mass of the Earth.

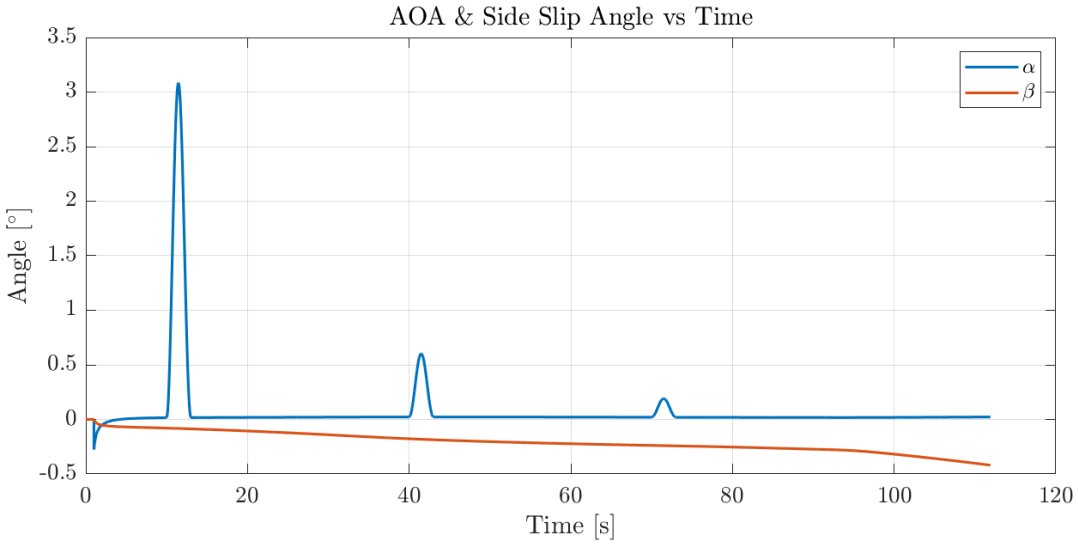

Figure 5.13: Open loop simulation results: AOA and side slip angle graph

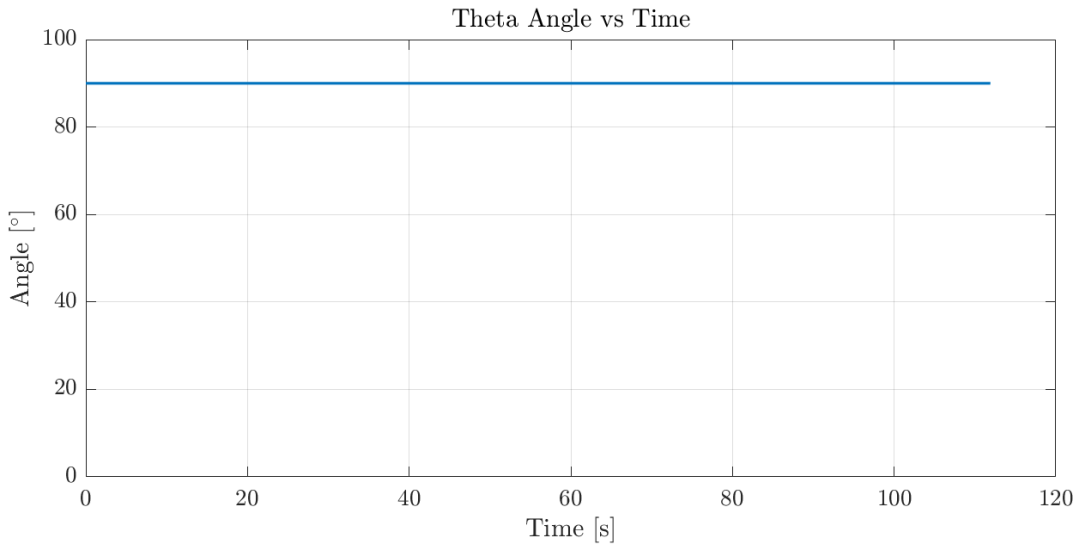

Figure 5.14: Open loop simulation results: Attitude angle in pitch plane graph

5.2.3 Closed Loop Simulation of the Launch Vehicle

In this section, the nominal trajectory will be explained and closed loop simulations will be performed. Firstly, the trajectory in the absence of any uncertainties which will be referred as "nominal trajectory" will be given. Secondly, to validate the controller against the uncertainties and disturbances, Monte Carlo simulations will be performed and results will be given.

In a typical implementation, the reference trajectory is calculated by the guidance algorithm. There are different approaches for the design of the guidance algorithms in the literature. Usually open loop commands are used for the lower altitudes and closed loop commands for the higher altitudes [71]. Since the guidance algorithm is out of the scope of this work, an open loop command will be used throughout the flight of the 1st stage. The formation of the curvature information is explained in the following paragraphs.

In this work, the reference trajectory is described by the trajectory curvature, whose effect on the attitude angle is given in the equation 3.37. Since the VEGA rocket is symmetric around the body-x axis, the rocket is assumed to manoeuvre only in the pitch axis. In other words, the body axes are defined such that the rocket is expected to rotate only around body-y axis. Hence, the trajectory curvature in the yaw plane ρ_{yaw} will be taken as zero. This means that the rocket will not make any yaw motion during the flight. As expressed earlier, the roll control of the launch vehicle is assumed to be realized by other means and hence in this simulations the rocket will not experience any moment around body-x axis.

The reference trajectory in the pitch axis is created with the classical approaches of minimizing the aerodynamics loads and maximizing the gained velocity [72]. The aerodynamic loads are proportional with the dynamic pressure q_∞ and angle of attack α. Hence, to minimize the aerodynamic loads, the term $q_\infty \alpha$ should be minimized. The dynamic pressure is proportional to the velocity squared and air density as given in the equation 5.19 and cannot be controlled. As a result, the term $q_\infty \alpha$ is minimized through the angle of attack. Especially when the dynamic pressure is high, around the time 37s in Figure 5.12, the angle of attack needs to be very small meaning that the rocket should not rotate near this point. Apart from the loads, there is another property that needs to be considered, the gained velocity. If the rocket does not rotate and go directly up, then most of the thrust is wasted since the gained velocity is much smaller than the potential. To prevent this, launch vehicles start maneuvering as soon as possible after the launch. Considering these facts, the trajectory curvature in the pitch axis is chosen as given in Figure 5.15. The two manoeuvres that corresponds to the half sine-wave shapes are initialized when the velocity is 10m/s and 420m/s.

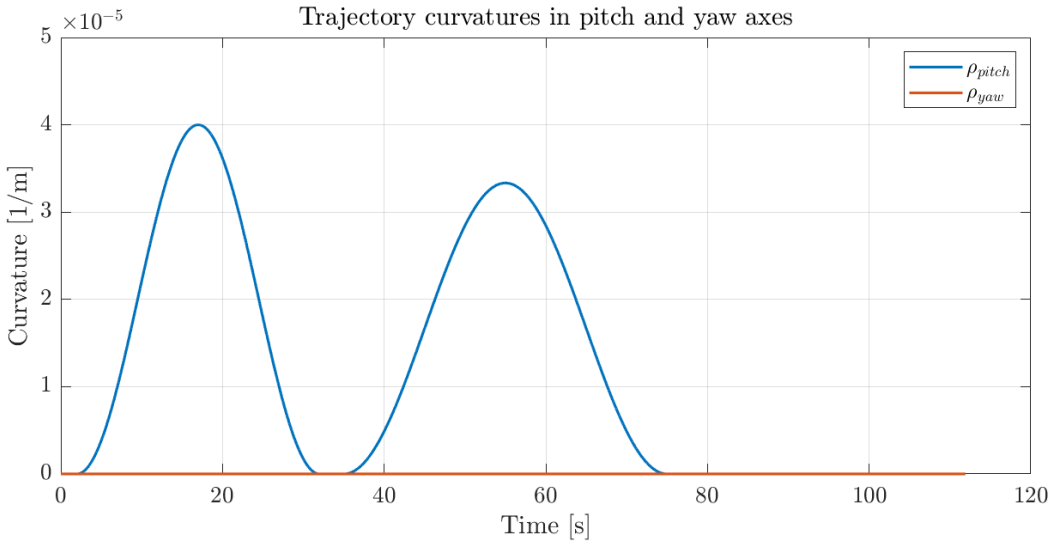

Figure 5.15: Nominal trajectory: Trajectory curvature in pitch and yaw axis

5.2.3.1 Nominal Trajectory

With the trajectory curvature information given in Figure 5.15, the other simulation outputs are as given in Figure 5.16 through 5.20.

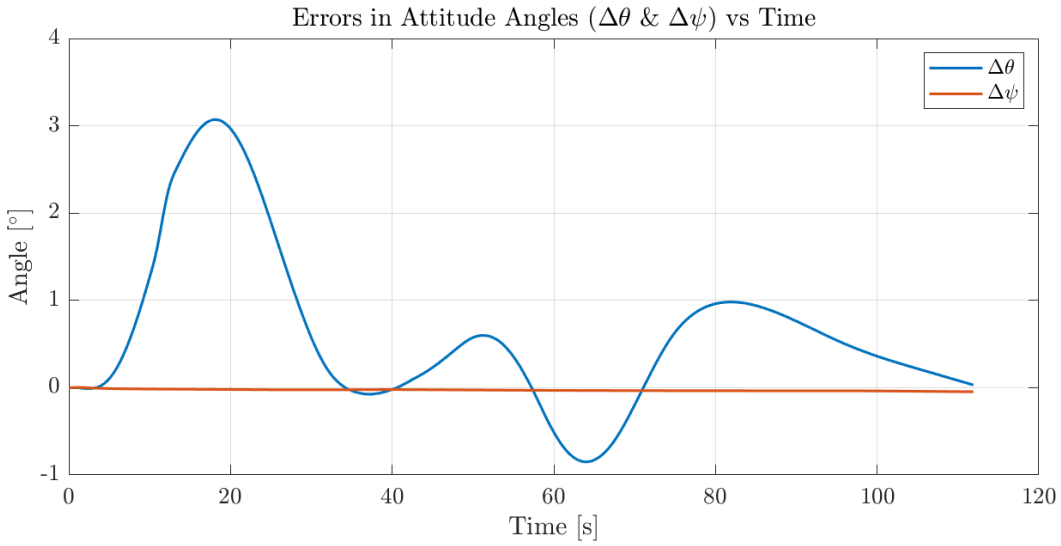

Figure 5.16: Nominal trajectory: Attitude angle errors

As seen in Figure 5.16, after the trajectory curvature rises, the error in the attitude angle is accumulated at first. As a result, the controller reacts to this by increasing the actuator command which is the thrust deflection angle resulting in a rotation of

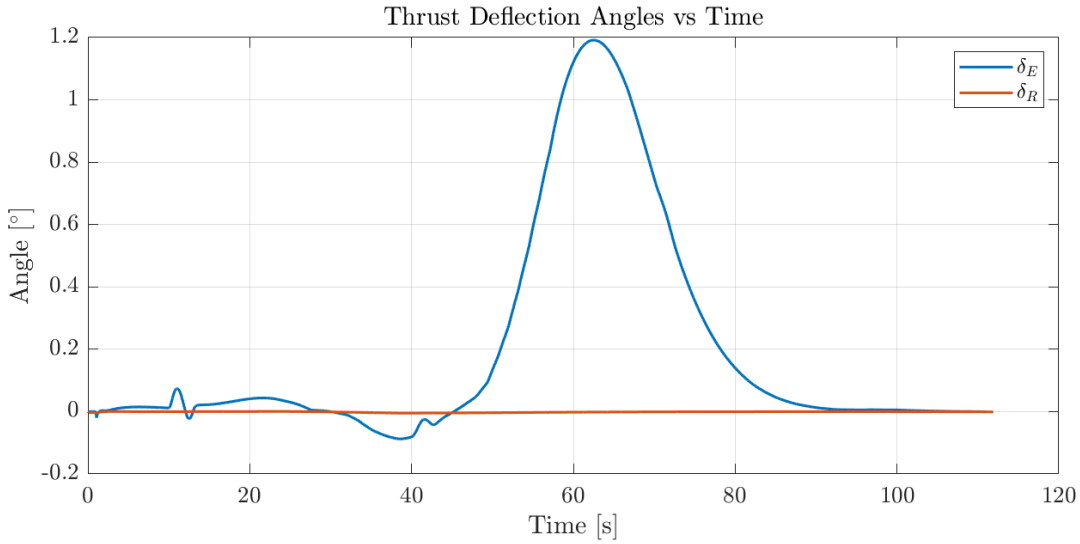

Figure 5.17: Nominal trajectory: Thrust deflection angles

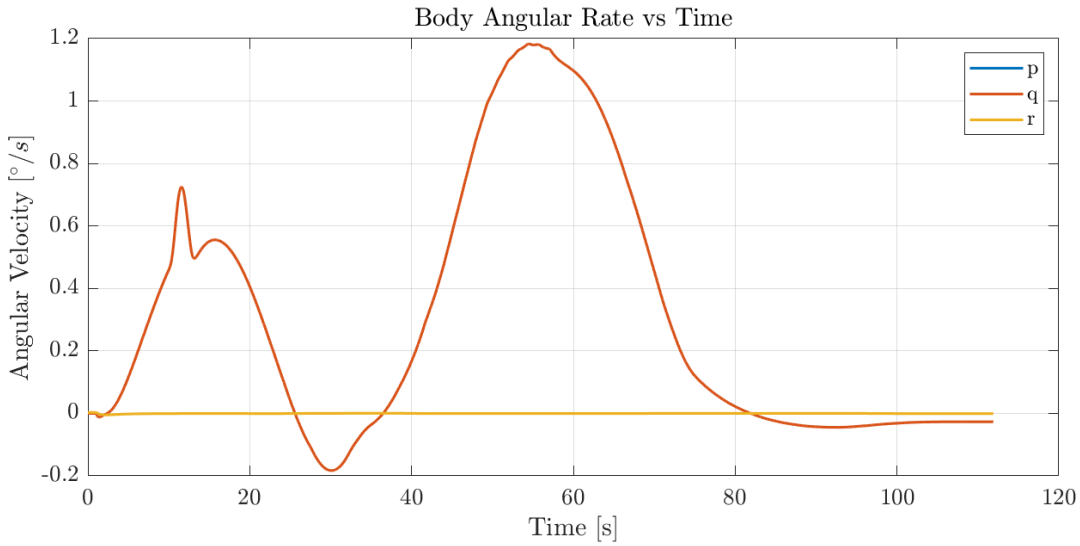

Figure 5.18: Nominal trajectory: Body angular rates

the rocket in the desired way to decrease the attitude error, as seen in Figure 5.17 and 5.18.

The spikes in the angle of attack graph given in Figure 5.19 is caused by the gust, whose properties are given in Table 5.4. The Euler angles θ and ψ are given in Figure 5.20. The attitude angle in the pitch plane is decreased to around $57°$ at the end of the 1^{st} stage of the flight. Note that since the Euler angles have a discontinuity around $\theta = 90°$, although the initial value for ψ is set to $0°$, it jumps to some value and then

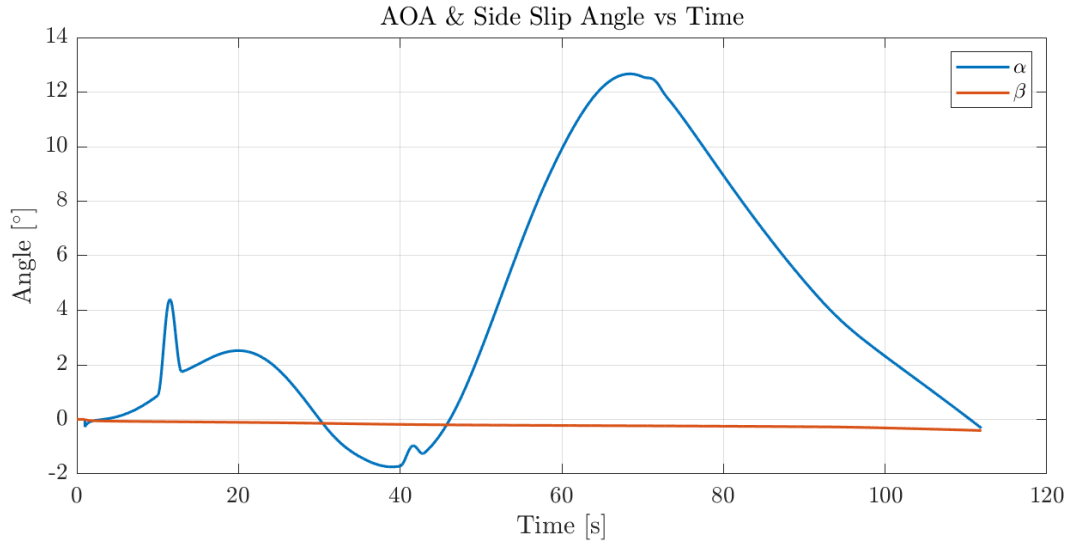

Figure 5.19: Nominal trajectory: AOA and side-slip angles

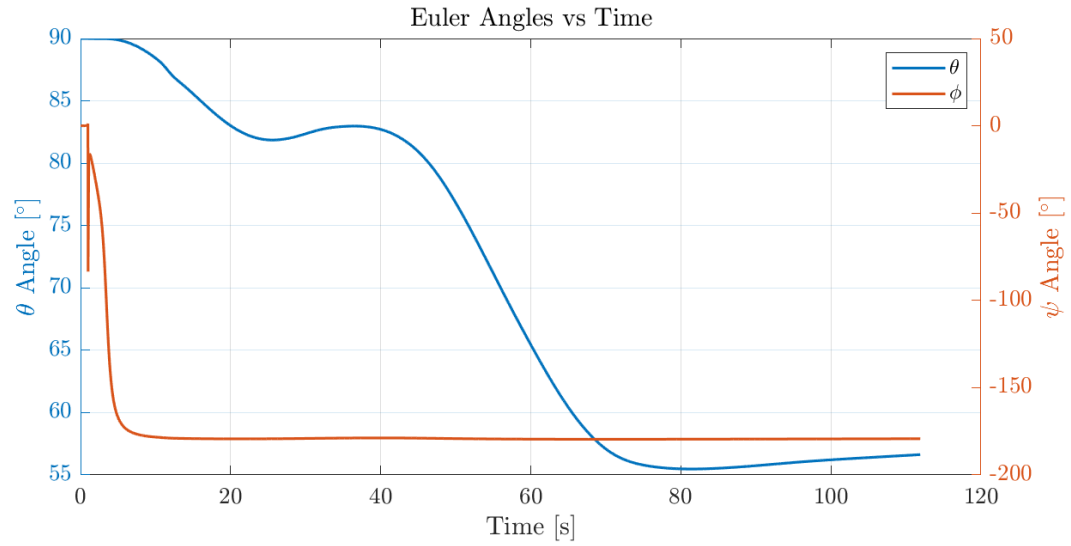

Figure 5.20: Nominal trajectory: Euler angles

drops to $-180°$. The same phenomena occurs on the roll angle ϕ as well, although the system have zero angular rate around body-x axis, it jumps to some value and then drops to $-180°$. This is due to the discontinuity of the Euler angles near the point $\theta = 90°$.

Table 5.5: Numeric values of the Monte Carlo parameters

Parameter Name	**Distribution**	**Uncertainty**	**Mean**
1st Stage Dry Mass	Normal	%5	8533kg
1st Stage Fuel Mass	Normal	%5	87710kg
Other Stages Total Mass	Normal	%5	39683kg
1st Stage Initial CG	Normal	%5	21.563m
1st Stage Burn Time	Normal	%12	109.9s
C_M	Uniform	%10	Figure 3.8
C_N	Uniform	%10	Figure 3.7
C_A	Uniform	%10	-
Gust Time	Uniform	[30 30 20]s	[0 30 60]s
Gust Magnitude	Uniform	[9 7 5]m/s	[3 2 1]m/s
Gust Duration	Uniform	[3 2 1]s	[3 2 1]s
IMU Delay	Uniform	5s	2s
Actuator Sensor Delay	Uniform	5s	2s

5.2.3.2 Monte Carlo Simulations

The response of the controller in the ideal scenario is given in the previous section. However, this is not enough by itself to validate the controller since this scenario is practically not realistic. To test the performance of the controller in the presence of the disturbances and uncertainties, Monte Carlo simulations will be performed. In this simulations, the system parameters such as mass, thrust, center of gravity distance from the nose and their rate of change, along with the aerodynamic moment and force coefficients, magnitude, time and duration of the gusts, delays of the sensors will be dispersed. These parameters are summarized in table 5.5.

In the table given in 5.5, if the distribution is uniform, then the uncertainty value means the difference between maximum and minimum possible uncertainty value. If it is normal, then its value means the 1σ value of the uncertainty. The gust parameters are chosen such that 3 gusts occur throughout the flight, whose magnitude and duration are decreasing with increasing the time. The gust magnitudes are chosen using the document [69].

With the values of the uncertain parameters given in the table 5.5, the Monte Carlo analysis is performed. In this analysis, 200 different runs are performed. The results of these simulations are presented in Figures 5.21 through 5.26.

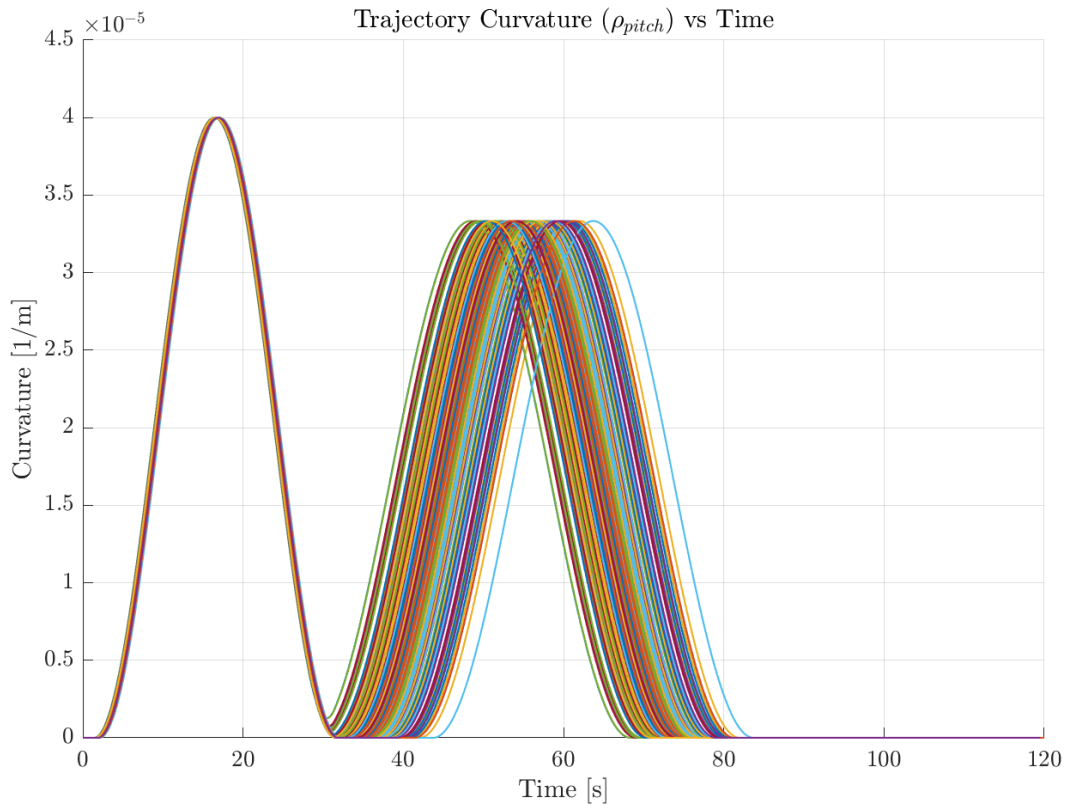

Figure 5.21: Monte Carlo analysis: Trajectory curvatures in different runs

As explained in the previous part, the first manoeuvre starts when the velocity is 10m/s and the second manoeuvre is initialized when the velocity is 420m/s. In the Monte Carlo analysis, due to the uncertainties in the thrust force and the mass, the rocket does not reach the same speed at the same time in different runs. As a result, the trajectory curvature may change from run to run. This situation can be seen from the graph given in Figure 5.21. The first peak is initiated when the velocity is very low so there is not much a difference which ensures that the rocket will experience almost the same curvature during the first maneuver.

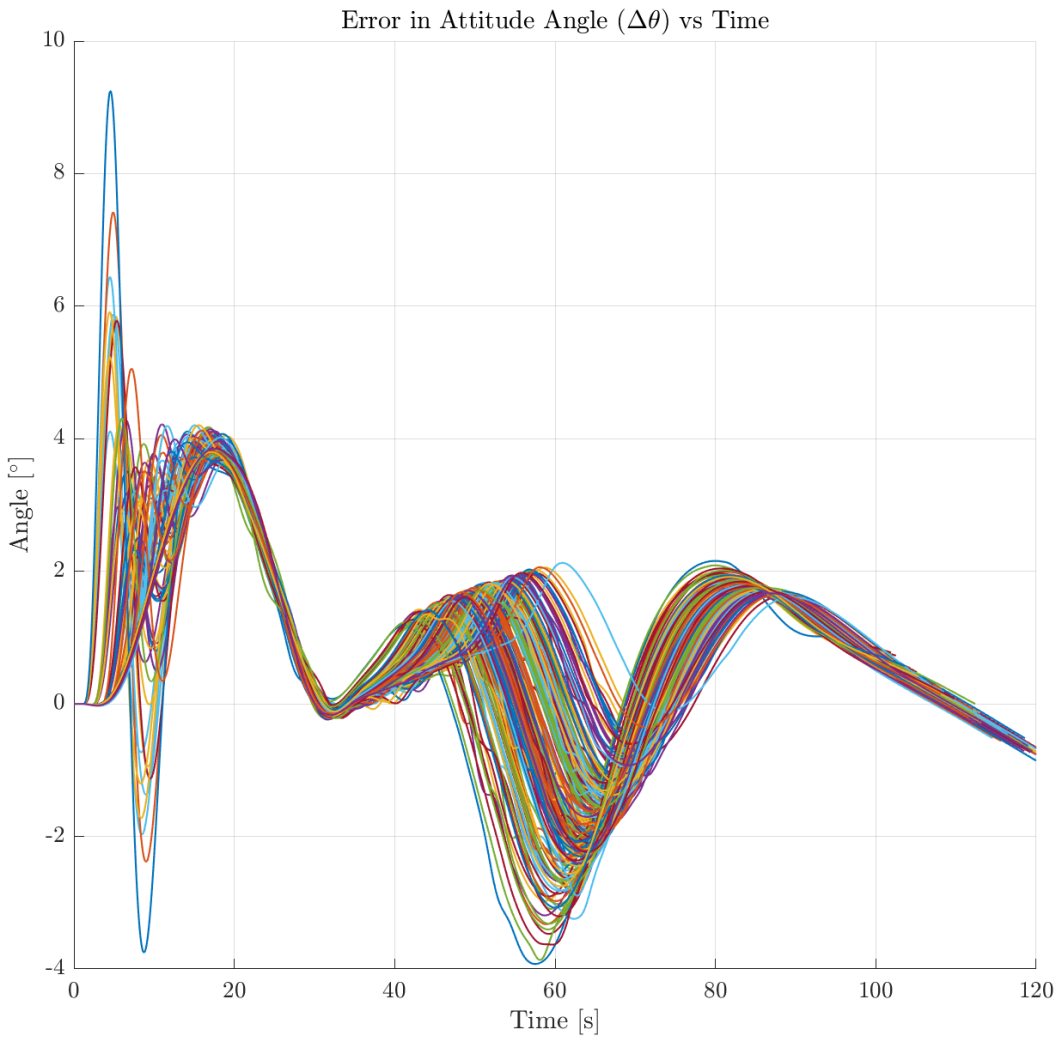

Figure 5.22: Monte Carlo analysis: Attitude angle error in different runs

From Figure 5.22, it can be seen that in the first manoeuvre the error in the pitch attitude angle can get as high as 8°. These peaks at the start are caused by the gust. In the second manoeuvre which happens in the higher altitudes, maximum attitude error is around 4°. At the end of the 1st stage flight, the error in the attitude angle is around -0.7°. As can be seen in Figure 5.26, the autopilot does not respond to these errors and even if it does, the error may not change since the thrust value is getting near zero at these moments.

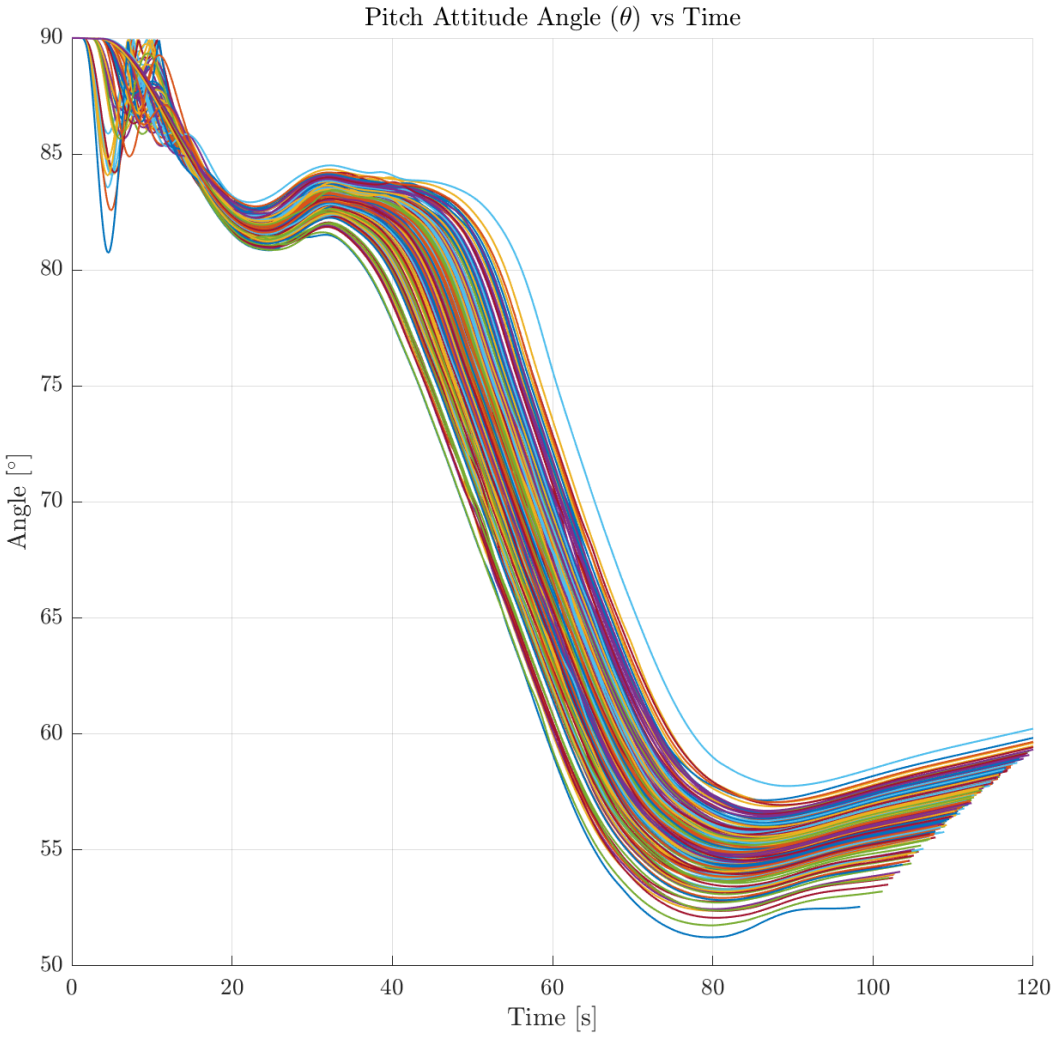

Figure 5.23: Monte Carlo analysis: Attitude angle in different runs

The attitude angle of the launch vehicle in the Monte Carlo runs is depicted in Figure 5.23. After the first rotation, the θ angle is dropped around 82°. After the second manoeuvre, the final θ angle becomes around 55°.

Note that although the curvature signals have the same duration and amplitudes, they do not imply the same change in the attitude angle. This is due to the differences in the velocity between runs. If a faster rocket experiences the same curvature, then the change in the attitude angle increases since the rocket will move through more distance in this curvature. As a result, although the attitude error angles are close to 0° at the end as seen in Figure 5.22, the final attitude angle changes between 53° and

60°.

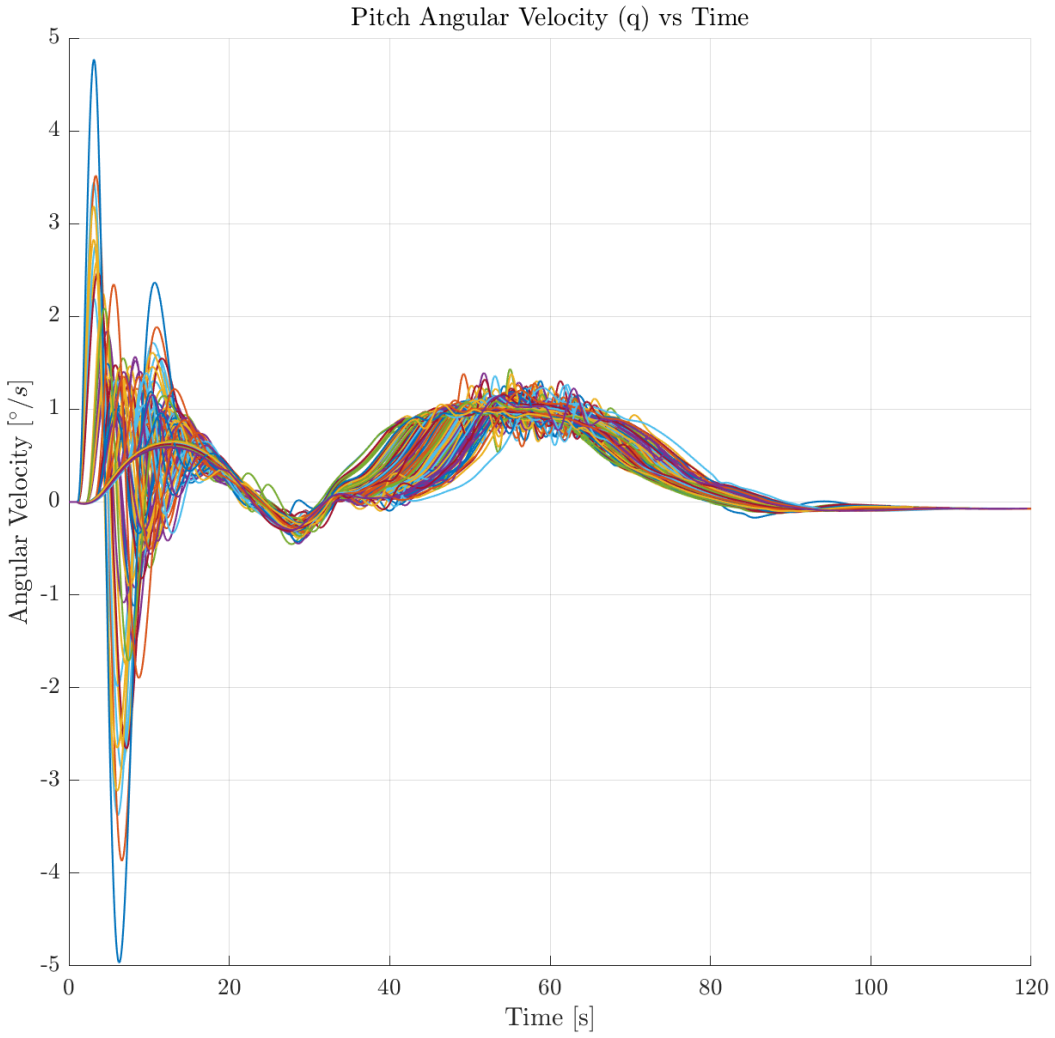

Figure 5.24: Monte Carlo analysis: Body angular rate around pitch axis in different runs

The effect of the gust on the body angular velocity can be seen in Figure 5.24. At the beginning, the gusts cause rocket to make bigger turns, this is due to the fact that the rocket is slow at these moments meaning that gust and rocket velocities are comparable. As the rocket is speeding up, the effect of the gust is still observable but their magnitude is decreasing. The same situation can be observed in the AOA, whose graphs are depicted in Figure 5.25. Note that in a realistic scenario the rocket might tear apart in the case of high AOA. In this simulation this is not taken into account and simulation is completed with the end of thrust.

Note that the rocket has positive q values and the θ is decreasing. This is due to the values of other attitude angles ψ and ϕ. At first, the rocket is initialized with the Euler angles $0°$, $90°$, $0°$ and due to the discontinuity around $\theta = 90°$, positive rotation around pitch axis lead to a jump in the ψ and ϕ to $-180°$. This situation can be seen in Figure 5.20.

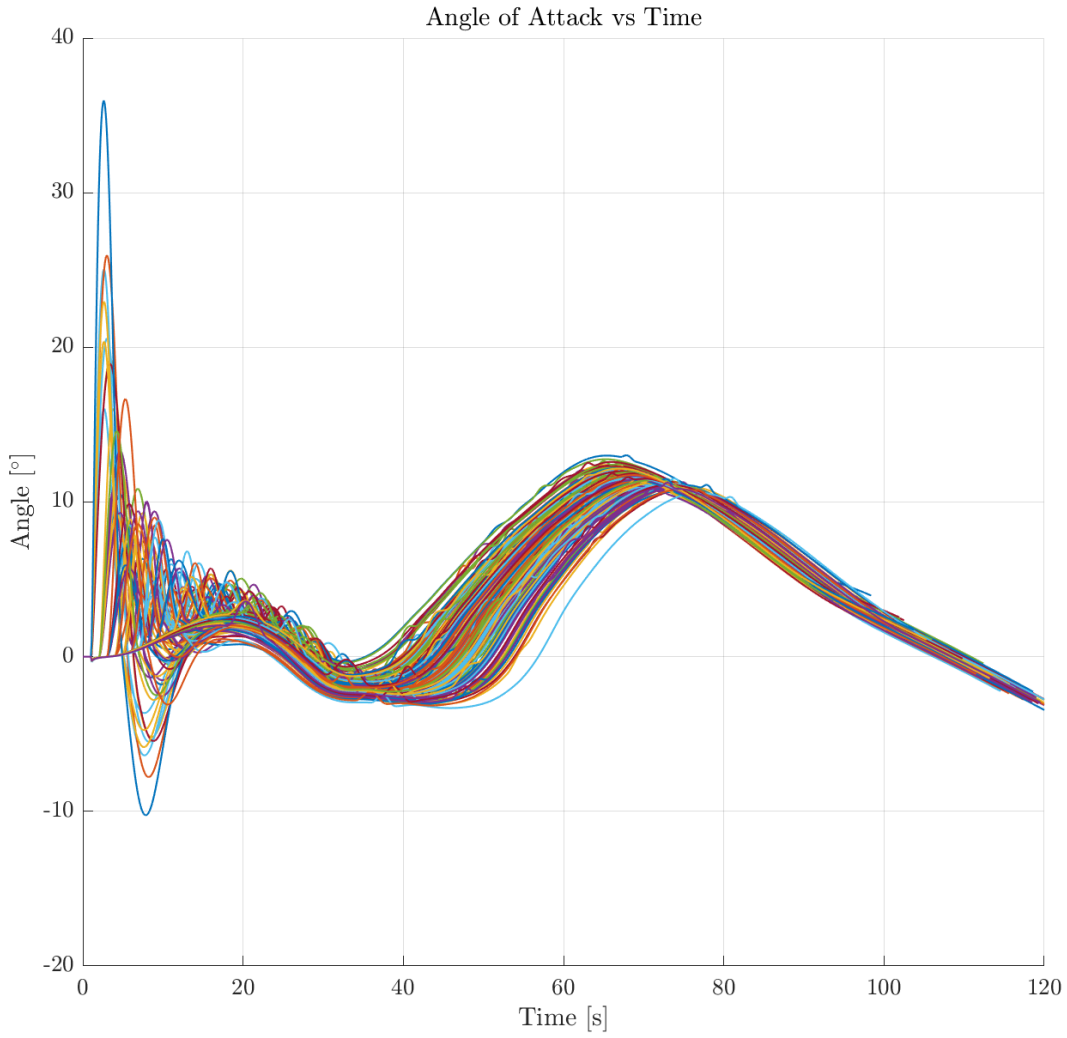

Figure 5.25: Monte Carlo analysis: Angle of attack in different runs

The actuator angle, i.e., thrust deflection angle can be seen in Figure 5.26. The rocket reacts to gust with higher actuator commands at first, this is due to the gust velocity being comparable with the rocket velocity. After the $t = 40s$, the actuator angle increases due to the excessive road curvature. Overall, the thrust deflection angle do

not cross $\pm 1.3°$.

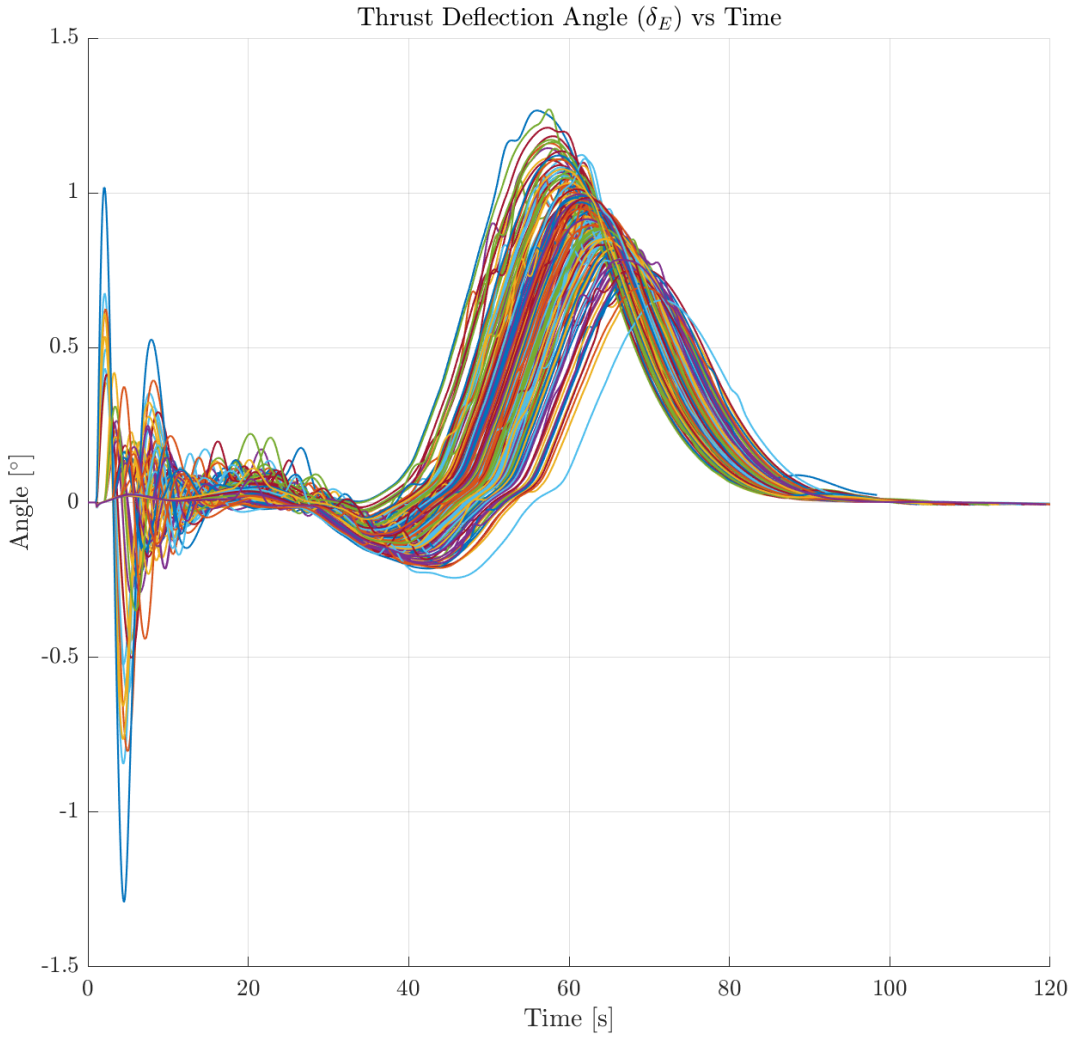

Figure 5.26: Monte Carlo analysis: Thrust deflection angle in different runs

The controller successfully maintains the stability of the rocket under the uncertainties and disturbances throughout the flight. This result can be deduced from Figure 5.24. Although the gust caused some oscillations at first, there is no sign of instability such as growing oscillations after some time. Note that the beginnings of the launch generally require different control schemes to maintain different requirements such as "minimum drift", or "minimum load" [73] [74]. Hence, these gust-related short oscillations can be regarded as "acceptable" for this work.

CHAPTER 6

CONCLUSION

This thesis is concerned with the application of control methods for linear parameter varying (LPV) systems to different types of vehicles.

First, linear and nonlinear models of lateral vehicle dynamics in the lane keeping configuration for ground vehicles are obtained. The linear model is transformed into LPV form and a controller is synthesized using the polytopic approach of LPV control. The nonlinear model is implemented on the MATLAB/Simulink software to test the controller. It is verified in the extensive nonlinear simulations under different initial conditions, disturbances and velocities. Some velocity and road curvature profiles are used in the process.

The simulation results prove that the applied method manages to keep the vehicle stable and errors below some limits even though the velocity is changing. To give numerical examples, the controller eliminates the 1m initial displacement error or 3° heading angle error within around 4 seconds. In a constant turn with a radius of 300m the maximum lateral displacement error is around 35cm while it can get as high as 65cm if the velocity and curvature change together. In these simulations sensor model is missing, hence in the future works the noise or delay in the lateral displacement signal might be considered.

The similar steps are executed for the launch vehicle modelling and controller design. For the system parameters, the launch vehicle VEGA in the first stage configuration is chosen. The aerodynamic coefficients are obtained using the DATCOM software. For the controller design, different from applications in the literature, the LPV controller is designed with the polytopic approach instead of gridding approach. To test the

designed autopilot in different scenarios, the nonlinear model considering realistic disturbances such as non-ideal actuator model, delays in sensors and environmental effects is implemented on Simulink. 6DoF high fidelity simulations are performed in the Monte Carlo analysis. To test the robustness of the controller, uncertainties in different parameters such as mass, thrust, CG location, aerodynamics and inertia.

The Monte Carlo simulation results show that the designed controller is able to maintain the stability in the presence of disturbances and uncertainties. The effect of sudden gusts becomes apparent at lower velocities in the form of high angle of attacks or turn rates. This problem is generally solved by different control schemes at the start of the launch such as "minimum drift" approaches, hence this behavior for the designed controller is expected. Apart from the stability, the reference tracking performance of the controller is also tested with nonzero trajectory curvature values in the pitch plane. The maximum attitude error throughout the trajectory is around 4° if the effect of the gust at the beginning is neglected. The final values of the attitude error is between 1° and -1°, since there are other stages, this error at the 1^{st} stage separation is not very critical and hence accepted.

In most applications, the rotations in these axes are tried to be decoupled by maneuvering the system around one axis at a time. However in practice this is rarely the case. In this work, the roll movement is assumed to be controlled by other means, hence the rocket rotates only around the pitch axis. As a result, the effect of the coupling between these axes are absent in these analysis. Also, the bending and sloshing dynamics are also neglected and with some addition the states, they can be included in the controller design in a possible future work.

REFERENCES

[1] J. S. Shamma, *Analysis and Design of Gain Scheduled Control Systems*. PhD thesis, Massachusetts Institute of Technology, 1988.

[2] D. J. Leith and W. E. Leithead, "Survey of gain-scheduling analysis and design," *International journal of control*, vol. 73, no. 11, pp. 1001–1025, 2000.

[3] R. D. Mocsányi, B. Takarics, A. Kotikalpudi, and B. Vanek, "Grid-based and polytopic linear parameter-varying modeling of aeroelastic aircraft with parametric control surface design," *Fluids*, vol. 5, no. 2, p. 47, 2020.

[4] D. Navarro-Tapia, A. Marcos, S. Bennani, and C. Roux, "Linear parameter varying control synthesis for the atmospheric phase VEGA launcher," vol. 51, pp. 68–73, Elsevier, 2018.

[5] A.-T. Nguyen, P. Chevrel, and F. Claveau, "On the effective use of vehicle sensors for automatic lane keeping via LPV static output feedback control," vol. 50, pp. 13808–13815, Elsevier, 2017.

[6] A. Abubakar, K. I. Dahiru, S. H. Sulaiman, and H. Mustapha, "Robust polytopic LPV based adaptive cruise control design for autonomous vehicle system," *International Journal of Scientific & Engineering Research*, vol. 10, 2019.

[7] M. S. Spillman, "Robust longitudinal flight control design using linear parameter-varying feedback," *Journal of Guidance, Control, and Dynamics*, vol. 23, no. 1, pp. 101–108, 2000.

[8] P. C. Pellanda, P. Apkarian, and H. D. Tuan, "Missile autopilot design via a multi-channel LFT/LPV control method," vol. 12, pp. 1–20, Wiley Online Library, 2002.

[9] S. Mustaki, A.-T. Nguyen, P. Chevrel, M. Yagoubi, and F. Fauvel, "Comparison of two robust static output feedback H_2 design approaches for car lateral con-

trol," in *2019 18th European Control Conference (ECC)*, pp. 716–723, IEEE, 2019.

[10] M. Ganet and M. Ducamp, "LPV control for flexible launcher," in *AIAA Guidance, Navigation, and Control Conference*, p. 8193, 2010.

[11] O. Törő, T. Becsi, and S. Aradi, "Design of lane keeping algorithm of autonomous vehicle," *Periodica Polytechnica Transportation Engineering*, vol. 44, no. 1, pp. 60–68, 2016.

[12] J. Baek, C. Kang, and W. Kim, "Practical approach for developing lateral motion control of autonomous lane change system," *Applied Sciences*, vol. 10, no. 9, p. 3143, 2020.

[13] R. Marino, S. Scalzi, and M. Netto, "Nested PID steering control for lane keeping in autonomous vehicles," *Control Engineering Practice*, vol. 19, no. 12, pp. 1459–1467, 2011.

[14] Z. Li, G. Cui, S. Li, N. Zhang, Y. Tian, and X. Shang, "Lane keeping control based on model predictive control under region of interest prediction considering vehicle motion states," *International journal of automotive technology*, vol. 21, no. 4, pp. 1001–1011, 2020.

[15] M. Samuel, M. Mohamad, M. Hussein, and S. M. Saad, "Lane keeping maneuvers using proportional integral derivative (PID) and model predictive control (MPC)," *Journal of Robotics and Control (JRC)*, vol. 2, no. 2, pp. 78–82, 2021.

[16] M. Shimakage, H. Kawazoe, O. Sadano, and T. Murakami, "Design of lane-keeping control with steering torque input for a lane-keeping support system," *SAE transactions*, pp. 448–455, 2001.

[17] P. Pfeffer and H.-H. Braess, *Basics of Lateral Vehicle Dynamics*, pp. 91–120. Cham: Springer International Publishing, 2017.

[18] J. Orr and T. Van Zwieten, "Robust, practical adaptive control for launch vehicles," in *AIAA Guidance, Navigation, and Control Conference*, p. 4549, 2012.

[19] B. Clement, G. Duc, S. Mauffrey, and A. Biard, "Aerospace launch vehicle control: A gain scheduling approach," *IFAC Proceedings Volumes*, vol. 35, no. 1, pp. 145–150, 2002.

[20] J. Mohammadpour and C. W. Scherer, *Control of linear parameter varying systems with applications*. Springer Science & Business Media, 2012.

[21] G. Zhenxing and F. Jun, "Robust LPV modeling and control of aircraft flying through wind disturbance," *Chinese Journal of Aeronautics*, vol. 32, no. 7, pp. 1588–1602, 2019.

[22] A. Marcos and G. J. Balas, "Development of linear-parameter-varying models for aircraft," *Journal of Guidance, Control, and Dynamics*, vol. 27, no. 2, pp. 218–228, 2004.

[23] G. J. Balas, I. Fialho, A. Packard, J. Renfrow, and C. Mullaney, "On the design of LPV controllers for the F-14 aircraft lateral-directional axis during powered approach," in *Proceedings of the 1997 American Control Conference (Cat. No. 97CH36041)*, vol. 1, pp. 123–127, IEEE, 1997.

[24] G. Balas, J. Mueller, and J. Barker, "Application of gain-scheduled, multivariable control techniques to the F/A-18 system research aircraft," in *Guidance, Navigation, and Control Conference and Exhibit*, p. 4206, 1999.

[25] F. Wu, A. Packard, and G. Balas, "LPV control design for pitch-axis missile autopilots," in *Proceedings of 1995 34th IEEE Conference on Decision and Control*, vol. 1, pp. 188–193, IEEE, 1995.

[26] J.-M. Biannic and P. Apkarian, "Missile autopilot design via a modified LPV synthesis technique," *Aerospace Science and Technology*, vol. 3, no. 3, pp. 153–160, 1999.

[27] W. Tan, A. K. Packard, and G. J. Balas, "Quasi-LPV modeling and LPV control of a generic missile," in *Proceedings of the 2000 American Control Conference. ACC (IEEE Cat. No. 00CH36334)*, vol. 5, pp. 3692–3696, IEEE, 2000.

[28] H. Pfifer and S. Hecker, "LPV controller synthesis for a generic missile model," in *2010 IEEE International Conference on Control Applications*, pp. 1838–1843, IEEE, 2010.

[29] H. Pfifer, "Quasi-LPV model of a NDI-controlled missile based on function substitution," in *AIAA Guidance, Navigation, and Control Conference*, p. 4970, 2012.

[30] R. Tekin and H. Pfifer, "Linear parameter varying control of an agile missile model based on the induced L2-norm framework," in *Advances in Aerospace Guidance, Navigation and Control*, pp. 3–14, Springer, 2013.

[31] D. Navarro-Tapia, A. Marcos, S. Bennani, C. Roux, and E. SpA, "Structured H-infinity and linear parameter varying control design for the VEGA launch vehicle," in *Proceedings of the 7th European Conference for Aeronautics and Aerospace Sciences (EUCASS)*, 2017.

[32] D. N. Tapia, *Robust and Adaptive TVC Control Design Approaches for the VEGA Launcher*. PhD thesis, University of Bristol, 2019.

[33] E. Alcalá, V. Puig, and J. Quevedo, "LPV-MPC control for autonomous vehicles," vol. 52, pp. 106–113, Elsevier, 2019.

[34] L. Jacobs, A. De Preter, J. Anthonis, J. Swevers, and G. Pipeleers, "Trajectory tracking of AGVs by linear parameter-varying control: a case study," vol. 51, pp. 43–48, Elsevier, 2018.

[35] P. Hang and X. Chen, "Path tracking control of 4-wheel-steering autonomous ground vehicles based on linear parameter-varying system with experimental verification," vol. 235, pp. 411–423, SAGE Publications Sage UK: London, England, 2021.

[36] M. Q. Nguyen, *LPV approaches for modelling and control of vehicle dynamics: application to a small car pilot plant with ER dampers*. PhD thesis, Université Grenoble Alpes, 2016.

[37] J. C. Tudon-Martinez, S. Varrier, O. Sename, R. Morales-Menendez, J.-J. Martinez, and L. Dugard, "Fault tolerant strategy for semi-active suspensions with LPV accommodation," in *2013 Conference on Control and Fault-Tolerant Systems (SysTol)*, pp. 631–636, IEEE, 2013.

[38] C. Hoffmann and H. Werner, "A survey of linear parameter-varying control applications validated by experiments or high-fidelity simulations," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 2, pp. 416–433, 2014.

[39] J. Mohammadpour and C. W. Scherer, *Control of linear parameter varying systems with applications*. Springer Science & Business Media, 2012.

[40] P. Apkarian and R. J. Adams, "Advanced gain-scheduling techniques for uncertain systems," in *Advances in linear matrix inequality methods in control*, pp. 209–228, SIAM, 2000.

[41] F. Wu, *Control of linear parameter varying systems*. PhD thesis, University of California, Berkeley, 1995.

[42] A. Hjartarson, P. Seiler, and A. Packard, "LPVTools: A toolbox for modeling, analysis, and synthesis of parameter varying control systems," *IFAC-PapersOnLine*, vol. 48, no. 26, pp. 139–145, 2015.

[43] A.-T. Nguyen, P. Chevrel, and F. Claveau, "LPV static output feedback for constrained direct tilt control of narrow tilting vehicles," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 2, pp. 661–670, 2018.

[44] R. Tóth, M. Lovera, P. S. Heuberger, M. Corno, and P. M. Van den Hof, "On the discretization of linear fractional representations of LPV systems," *IEEE Transactions on Control Systems Technology*, vol. 20, no. 6, pp. 1473–1489, 2011.

[45] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear matrix inequalities in system and control theory*. SIAM, 1994.

[46] C. W. Scherer, "LPV control and full block multipliers," *Automatica*, vol. 37, no. 3, pp. 361–375, 2001.

[47] Y. Nesterov, "Interior point polynomial methods in convex programming," *Theory and Applications*, 1994.

[48] S. V. Gusev and A. L. Likhtarnikov, "Kalman-Popov-Yakubovich lemma and the s-procedure: A historical essay," *Automation and Remote Control*, vol. 67, no. 11, pp. 1768–1810, 2006.

[49] P. P. Ramanata, *Optimal vehicle path generator using optimization methods*. PhD thesis, Virginia Tech, 1998.

[50] M. A. Ardm Haseeb, *Optimal Control Problems for Safe and Efficient Lane Changes of Self-Driving Vehicles*. PhD thesis, Çankaya University, 2017.

[51] K. Saraçoğlu, B. Üleş, and K. W. Schmidt, "A lane keeping system with a weighted preview measurement," in *2018 26th Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, IEEE, 2018.

[52] J. H. Blakelock, *Automatic control of aircraft and missiles*. John Wiley & Sons, 1991.

[53] S. Gallucci, F. Battie, M. Volpi, T. Fossati, and G. Curti, "Vega launch vehicle first flight mission analysis—VV01," in *2012 IEEE First AESS European Conference on Satellite Telecommunications (ESTEL)*, pp. 1–5, IEEE, 2012.

[54] Arianespace, *VEGA User's Manual*. Arianespace.

[55] B. Özkan, *Dynamic Modeling, Guidance and Control of Homing Missiles*. PhD thesis, Middle East Technical University, 2017.

[56] W. B. Blake, *Missile DATCOM User's Manual*. AFRL.

[57] S. Ishida and J. E. Gayko, "Development, evaluation and introduction of a lane keeping assistance system," in *IEEE Intelligent Vehicles Symposium, 2004*, pp. 943–944, IEEE, 2004.

[58] N. C. Basjaruddin, D. Saefudin, S. A. Aryani, *et al.*, "Lane keeping assist system based on fuzzy logic," in *2015 International Electronics Symposium (IES)*, pp. 110–113, IEEE, 2015.

[59] O. Törő, T. Becsi, and S. Aradi, "Design of lane keeping algorithm of autonomous vehicle," *Periodica Polytechnica Transportation Engineering*, vol. 44, no. 1, pp. 60–68, 2016.

[60] D. De Vito, A. Kron, J. de Lafontaine, and M. Lovera, "A Matlab toolbox for LMI-based analysis and synthesis of LPV/LFT self-scheduled H∞ control systems," in *2010 IEEE International Symposium on Computer-Aided Control System Design*, pp. 1397–1402, IEEE, 2010.

[61] J. Lofberg, "YALMIP: A toolbox for modeling and optimization in MATLAB," in *2004 IEEE international conference on robotics and automation (IEEE Cat. No. 04CH37508)*, pp. 284–289, IEEE, 2004.

[62] M. ApS, "Mosek optimization toolbox for matlab," *User's Guide and Reference Manual, Version*, vol. 4, 2019.

[63] J. Ackermann, J. Guldner, W. Sienel, R. Steinhauser, and V. I. Utkin, "Linear and nonlinear controller design for robust automatic steering," *IEEE Transactions on Control Systems Technology*, vol. 3, no. 1, pp. 132–143, 1995.

[64] DOT-NY, *Highway Design Manual - Chapter 5*. New York - Department of Transportation.

[65] KGM, *Karayolu Tasarım El Kitabı*. Karayolları Genel Müdürlüğü.

[66] J. Jang, A. Alaniz, R. Hall, N. Bedrossian, C. Hall, and M. Jackson, "Design of launch vehicle flight control systems using ascent vehicle stability analysis tool," in *AIAA guidance, navigation, and control conference*, p. 6652, 2011.

[67] E. Dumont, "Variations of solid rocket motor preliminary design for small TSTO launcher," 2012.

[68] B. L. Stevens, F. L. Lewis, and E. N. Johnson, *Aircraft control and simulation: dynamics, controls design, and autonomous systems*. John Wiley & Sons, 2015.

[69] D. Johnson, "Terrestrial environment (climatic) criteria guidelines for use in aerospace vehicle development, 2008 revision," tech. rep., 2008.

[70] J. Picone, A. Hedin, D. P. Drob, and A. Aikin, "NRLMSISE-00 empirical model of the atmosphere: Statistical comparisons and scientific issues," *Journal of Geophysical Research: Space Physics*, vol. 107, no. A12, pp. SIA–15, 2002.

[71] G. A. Dukeman, *Closed-loop nominal and abort atmospheric ascent guidance for rocket-powered launch vehicles*. Georgia Institute of Technology, 2005.

[72] G. Dukeman, "Atmospheric ascent guidance for rocket-powered launch vehicles," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, p. 4559, 2002.

[73] D. Garner, “Control theory handbook,” tech. rep., 1964.

[74] B. Wie, W. Du, and M. Whorton, “Analysis and design of launch vehicle flight control systems,” in *AIAA guidance, navigation and control conference and exhibit*, p. 6291, 2008.