

DEVELOPMENT OF A FAULT-TOLERANT MODEL PREDICTIVE CONTROLLER FOR VEHICLE LATERAL STABILITY

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
MASTER OF SCIENCE
IN
MECHANICAL ENGINEERING

By
Muhammed Kemal Köysüren
August 2023

DEVELOPMENT OF A FAULT-TOLERANT MODEL PREDICTIVE CONTROLLER FOR VEHICLE LATERAL STABILITY

By Muhammed Kemal Köysüren

August 2023

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Melih Çakmakcı(Advisor)

Yiğit Karpaz

İsmail Uyanık

Approved for the Graduate School of Engineering and Science:

Orhan Arıkan
Director of the Graduate School

ABSTRACT

DEVELOPMENT OF A FAULT-TOLERANT MODEL PREDICTIVE CONTROLLER FOR VEHICLE LATERAL STABILITY

Muhammed Kemal Köysüren
M.S. in Mechanical Engineering
Advisor: Melih Çakmakcı
August 2023

Recently, there has been an increased interest in the automotive industry using scaled test vehicles to test the real-time performance of modeling and control algorithms. A scaled prototype, specifically developed and instrumented for enhancing vehicle lateral stability, offers distinct advantages in terms of cost reduction and the ability to repeat tests under various vehicle maneuvering scenarios rapidly. First, the mechatronic design of a 1:8 scaled electric vehicle with 4-wheel-drive and 4-wheel-independent-steering was done and the prototype vehicle was built. Plant model parameters such as the cornering coefficients of the tires are estimated using various methods such as traditional neural network training, a Physics Informed Deep Learning (PIDL) algorithm, and Pacejka's tire modeling procedure. Secondly, a fault-tolerant reconfigurable model predictive controller (MPC) is proposed to enhance reference tracking for four-wheel-drive and four-wheel-steering vehicles under concurrent steering actuator faults. The method detects, isolates, and estimates fault magnitudes, which inform adjustments to the MPC formula. Performance validation is conducted through obstacle avoidance maneuvers with a control-oriented vehicle model and real-time applicability tests with a Processor-in-the-Loop system using a high-fidelity vehicle model. The test results confirm the proposed algorithm's superior performance over the conventional MPC. Lastly, a computationally efficient two-path optimal control allocation method is proposed to reduce controller block execution time in vehicle ECU. High-fidelity results prove the computational cost reduction of the proposed algorithm over the conventional allocation method.

Keywords: Vehicle Modeling, Parameter Identification, Fault-Tolerant Model Predictive Control, Scaled Test Vehicle, Control Allocation.

ÖZET

ARAÇ YANAL STABİLİTESİ İÇİN ARIZAYA DAYANIKLI MODEL ÖNGÖRÜLÜ KONTROLÇÜ GELİŞTİRİLMESİ

Muhammed Kemal Köysüren
Makine Mühendisliği, Yüksek Lisans
Tez Danışmanı: Melih Çakmakcı
Ağustos 2023

Modelleme ve kontrol algoritmalarının gerçek zamanlı performansını test etmek için ölçekli test araçlarının kullanılmasına otomotiv endüstrisinde artan bir ilgi vardır. Aracın yanal stabilitesini artırmak için özel olarak geliştirilmiş ve donatılmış ölçekli bir prototip, maliyetin düşürülmesi ve çeşitli araç manevra senaryoları altında testleri hızla tekrarlayabilme açısından belirgin avantajlar sunar. İlk olarak 1:8 ölçekli 4 tekerlekten çekişli ve 4 tekerlekten bağımsız yönlendirmeye sahip elektrikli aracın mekatronik tasarımı yapılmış ve prototip araç üretimi gerçekleştirilmiştir. Bu elektrikli aracın viraj alma katsayıları, geleneksel sinir ağı, fizik bilgili derin öğrenme ve Pacejka teker modelleme methodlarıyla tahmin edilmiştir. İkinci olarak, yönlendirme aktüatörlerinin eşzamanlı arızaları durumunda dört tekerden çekişli ve dört tekerden yönlendirmeye sahip araçların referans takibini geliştirmek için arızaya dayanıklı yeniden ayarlanabilir model öngörülü kontrolcü sunulmuştur. Bu yöntem, hata tespiti, izolasyonu ve büyüklük tahminini gerçekleştirir ve bu bilgileri kullanarak kontrolcü parametrelerini düzenler. Performans kıyaslaması, yüksek doğruluğa sahip araç modelini kullanan gerçek zamanlı uygulanabilirlik testleriyle yapılmıştır. Test sonuçları, önerilen algoritmanın normal MPC'ye göre üstün performans sergilediğini doğrulamıştır. Son olarak, araç üzerinde kontrol bloğu çalışma süresini azaltmak için hesaplama açısından verimli iki yollu bir optimal kontrol tahsis yöntemi sunulmuştur. Önerilen algoritmanın hesaplama yükünün geleneksel tahsis yöntemine kıyasla azaldığını kanıtlar.

Anahtar sözcükler: Araç Modelleme, Parametre Tanımlama, Arızaya Dayanıklı Model Öngörülü Kontrol, Ölçeklendirilmiş Test Aracı, Kontrol Tahsisi.

Acknowledgement

First of all, I would like to express my gratitude to my academic advisor Melih Çakmakcı for his unflagging guidance throughout my graduate study. Thanks to his personal and professional support, he improved my knowledge and vision in both automotive and control engineering and enlightened my path in both personal and academic life.

I want to thank Yiğit Karpata and İsmail Uyanık for being on my thesis committee, and for their valuable suggestions and time. I also thank my lab friends Doğa Dağ and Ahmet Faruk Keleş for their invaluable support, suggestions, and considerable efforts in building and conducting the experiments with my test vehicle.

I have special thanks to my close friends Ahmet Enes Tan, Ozan Erel, and Berkan Taşar to make this long and uncertain journey fun.

I would like to thank my parents, Suna Köysüren and Erhan Köysüren, for all the opportunities and unflagging support they provided me throughout my entire life. Nothing would be possible without them on my back. Also, I thank to Begüm for being the greatest sister ever.

Last but not least, I want to glorify Matlab, for its great documentation of toolboxes and 'help' commands.

Contents

1	Introduction	1
1.1	Vehicle Modeling Techniques	1
1.2	Vehicle Lateral Stability Optimal Control and Fault-Tolerant Control Studies	2
1.3	Control Allocation for Vehicle Lateral Stability Studies	5
1.4	Motivation and Contribution	6
1.5	Organization of the Thesis	8
2	4-Wheel-Drive and 4-Wheel-Independent-Steering Scaled Electric Test Vehicle	9
2.1	Literature Review	10
2.2	Vehicle Modeling	11
2.2.1	Vehicle Linear Bicycle Model	11
2.2.2	Two Track Vehicle Model	13
2.3	Test Vehicle	17

2.3.1	Mechanical Design	17
2.3.2	Electronic System Architecture	18
3	Tire Cornering Parameter Estimation for Vehicle Lateral Stability Algorithms	25
3.1	Test Vehicle and Experiments	26
3.2	Data Collection and Processing	26
3.3	Pacejka's Tire Modelling	28
3.4	Physics-Informed Parameter Learning	29
3.5	Controller Implementation with Online Parameter Estimation and Experiments	31
3.5.1	Reference Generator	31
3.5.2	Optimal MIMO Controller	32
3.5.3	Comparison of PIDL and TNN Parameter Estimation Performance	33
3.5.4	PIDL Experiments with Varying Drive Conditions	34
4	Fault-Tolerant Model Predictive Controller with Reconfiguration for Optimal Lateral Stability in Over-actuated Electric Vehicles	36
4.1	Control Development	37
4.1.1	MIMO Constrained Model Predictive Control Formulation	37
4.1.2	Fault Detection, Isolation, and Magnitude Estimation	45

4.1.3	Fault Tolerant Reconfigurable MPC	49
4.2	Validation of the Controller	52
4.2.1	Simulation Results in the Non-Linear Model	52
4.2.2	Simulations with the High Fidelity Vehicle Model with PIL capabilities	59
5	Computationally Efficient Nonlinear Control Allocation	66
5.1	Nomenclature	67
5.2	Control Structure	67
5.2.1	Virtual Control Input Generator	68
5.2.2	Single-Path SQP Allocation Problem	70
5.2.3	Two-Path SQP Allocation Problem	72
5.3	Simulation Results	76
5.3.1	Obstacle Avoidance Test at $V = 16[m/s]$	77
5.4	Discussion and Future Works	79
6	Conclusion	84
A	Simulink Models for Arduino Programming	92
B	Simulink Model for Model Predictive Controller	96
C	Simulink Models for Control Allocation Methods	98

D Merit Function ψ for Feasibility Check

101



List of Figures

2.1	Vehicle Lateral Model Parameters with Single Track (Bicycle) Assumption.	11
2.2	Typical Simulation Output	13
2.3	Vehicle Two-Track Model	14
2.4	Test Vehicle	21
2.5	Test Vehicle Mechanical Components 1	22
2.6	Test Vehicle Mechanical Components 2	23
2.7	Test Vehicle Mechanical Components 3	24
2.8	Test Vehicle Communication System	24
3.1	Results of Experiments	27
3.2	Scaled Test Vehicle with 4WD-4WS equipped with inertial measurement sensing and wireless communication capabilities for modeling experiments.	28
3.3	Tire Model Regression	29

3.4	Feedback Control with Updated References Block Diagram	32
3.5	Controller Performances (PIDL: physics-informed deep learning, DL: regression deep learning). In bottom figure dotted lines: rear wheel steering angle and continuous lines: front wheel steering angle	33
3.6	Yaw Rate Comparison Experiment (not updated: constant K_{us} system, updated: updated K_{us} parameter with online estimation). In bottom figure dotted lines: rear wheel steering angle and con- tinuous lines: front wheel steering angle	35
4.1	Block Diagram for Model Predictive Control Implementation . . .	38
4.2	Fault Detection, Isolation, and Magnitude Estimation Scheme . .	46
4.3	Fault Tolerant Control Scheme	49
4.4	Scenario 1	54
4.5	Scenario 2	57
4.6	Scenario 3 Vehicle Trajectory	59
4.7	The independent 4WD and independent 4WS Scaled Test Vehicle [43]	60
4.8	PIL Test Scenario 1	61
4.9	PIL Test Scenario 2	63
4.10	PIL Scenario 3 Vehicle Trajectory	64
4.11	PIL Test Controller Block Execution Time	64
5.1	Single-Path Control Structure	70
5.2	Two-Path Control Structure	73

5.3	Vehicle States	78
5.4	Single-Path System Torque and Steering Outputs at 0.01 Execution Rate	79
5.5	Single-Path System Torque and Steering Outputs at 0.001 Execution Rate	80
5.6	Two-Path System Torque and Steering Outputs at 0.01 Execution Rate	81
5.7	Obstacle Avoidance at $V = 16[m/s]$	81
5.8	Lateral Displacement Drift	81
5.9	Integrated Value of Objective Function	82
5.10	Summary of Results	82
5.11	Single-Path vs Two-Path on Vehicle ECU	83
A.1	Arduino Mega Sensor Data Collection	93
A.2	Arduino Uno Motor Drive	94
A.3	Sliding Mode Observer for Tire Lateral Force Estimation	94
A.4	MIMO Controller Arduino Due Implementation	95
B.1	Fault Tolerant Model Predictive Controller Simulink Model	97
C.1	Single-Path Control Allocation Simulink Model	99
C.2	Two-Path Control Allocation Simulink Model	100

List of Tables

2.1	List of Mechanical Components	17
4.1	Test Vehicle Parameters	52
4.2	MPC Properties	53
4.3	Explanation of the Plot Legends for Figs. 4.4-4.10.	55
4.4	Total Absolute Error Values of (no failure) MPC	58
4.5	Total Absolute Error Values (Simulations)	58
4.6	Total Absolute Error Values (PIL Simulations)	62
5.1	Nomenclature	68
5.2	Hardware and Software Settings of the Simulations	76
5.3	SQP Optimization Problem Parameters	76
5.4	Average Execution Time	79

Chapter 1

Introduction

1.1 Vehicle Modeling Techniques

Many modern automatic control applications require parameter identification of the plant model. These identified parameters can be used for the initial controller design, online adjustments, and system health monitoring. Recent developments in data gathering and processing in embedded control systems enable combining modeling and control applications with learning techniques. However, due to the computational complexity, control-oriented neural networks are still difficult to implement in many applications [1]. Additionally, noisy sensor data also cause complications in processing collected data in a practicable way. To overcome this problem, researchers gravitate to the physics-informed deep learning method [2]. There are numerous examples of modeling the plant with learning procedures to increase the fidelity of the control systems. In [3], in order to push the model predictive control's (MPC) performance to the limits, the modeling learning procedure is improved so that the prediction model ensures the best closed-loop performance. In [4], the performance of data-driven and physical modeling approaches for vehicle lateral-longitudinal dynamics are compared. Results show that the data-driven model bests both linear and non-linear physical models.

In [5], an online model-based reinforcement learning method is proposed to identify vehicle linear tire parameters to maximize maneuverability under variable road conditions such as unknown terrain.

Automotive control applications are one of those fields where estimating the vehicle parameters can improve the vehicle's performance immensely. Automotive systems contain many complex dynamic mechanisms, considered lumped parameter models during the control system design phase. For example, the LTI state-space model is widely used in vehicle lateral stability control. In this model, the cornering coefficient is the hardest parameter to be estimated. It requires experimental data, such as tire lateral forces, which requires an estimator (addition to the IMU sensor data) or additional expensive force sensors. Pacejka's method [6] can estimate the nonlinear behavior of tire models but the method requires many data points and an estimator or additional sensors.

In deep learning, many data points and data sets with labeled data are required to train a network with high accuracy. To train such a network, extensive experiments with many sensors are required. Another option is to use a so-called physics-informed parameter learning scheme. This type of deep network contains the system physics in the loss function, enabling them to be utilized with very few data sets and fewer data points. This allows them to be used and trained for a single system with high accuracy.

1.2 Vehicle Lateral Stability Optimal Control and Fault-Tolerant Control Studies

In recent years, with the increased electrification in the automotive industry, there has been a tendency to use mechatronic components for both steering and traction actuators on each wheel since they can be run with relatively more straightforward controller modules that are also easier integrated into the vehicle control networks [7, 10]. This increases actuator redundancy in the overall control system. On the

other hand, compared to their mechanical correspondence, drive-by-wire, and steer-by-wire systems, i.e., mechatronic steering and traction actuators, are not fail-safe by their nature according to the extensive research in [11]. As examples of x-by-wire systems, electric hub motor directly mounted on the vehicle in wheel to provide traction force without addition of any powertrain element and servo motor that can control the wheel angle with electrical command can be shown. Hub motor and steering actuator on each wheel increase the unsprung mass of quarter car significantly. Consequently, these actuators and their hubs wear out faster due to the variable road conditions with more weight exposure, and insufficient passive suspension compensation [12]. Therefore, maintenance of the mechatronic actuators must be done more frequently than the mechanical ones to keep them reliable and to avoid actuator failures [13].

Fault-tolerant systems minimize the effect of the fault on the system against actuator failures and ensure safe operation [14, 15]. Additionally, increased processing power of the control hardware allows for running more complicated algorithms on vehicle ECU, such as model predictive controllers (i.e., MPCs), that perform real-time optimization on vehicle operation [16]. Combining the benefits of fault tolerance and optimality in performance will extend the scheduled maintenance intervals of mechatronic actuators without affecting the safety and performance of the vehicle.

There are examples in the literature about fault-tolerant control design and their performance evaluations in both simulation and real-life implementations for over-actuated electric vehicles to improve safety and stability. In [17], and [18], an integrated vehicle control method is proposed comprising a fault-tolerant adaptive control allocation for over-actuated electric vehicles to increase yaw stability by allocating desired longitudinal forces lateral forces and yaw moment with front steering angle correction, rear steering initiation, individual tire traction force and suspension force distribution on each wheel. In [19], adaptive fault-tolerant control for a four-wheel-independent-drive electric vehicle with an active fault diagnosis and passive fault-tolerant control method is proposed. The fault or system health detection procedure is done by comparing electrical signals driving wheel motors and the output of motor control gain which is expected to be

reduced in fault situations. After the detection procedure, control efforts of each motor is readjusted by adaptive control-based passive fault-tolerant controller (FTC) to avoid further damages to the faulty components. In [20], model-based fault detection and isolation and fault-tolerant control approach for over-actuated electric vehicles is proposed for traction control in nonlinear disturbances in order to improve the path following performance of the autonomous vehicle in faulty conditions. In [21], an active fault tolerant control system with a fault detection isolation algorithm is proposed for a four-wheel traction electric vehicle. The algorithm was then verified with simulations under the steering actuator fault. In [22], a non-linear reconfigurable model predictive controller is proposed to make autonomous systems fault-tolerant at the actuator level in order to maintain the vehicle on the reference trajectory generated by motion planner. The algorithm introduces equality constraints for steering and traction actuators depend on the location and the class of the fault. In this research, in the reconfiguration process of the MPC optimization, only constraints of the problems are intervened. There is no alternation inside the control input matrix of non-linear vehicle model used in MPC formulation. Additionally, in this study there is no information about fault detection and magnitude estimation process.

Researches about implementation of model predictive control (MPC) techniques and prove real-time applicability on vehicles with limited computational sources studies for scaled vehicles become widespread in automotive industry. A survey paper in [23] compiles the control algorithms used in autonomous vehicle races. A significant amount of research mentioned in this survey implements different types of MPCs for autonomous vehicles to reduce lateral and heading errors. In this research, the use of these algorithms in racing environment provides beneficial examples of real-time applicability and control at the limits of handling of scaled vehicles. In [24], the model-based fault diagnosis technique is combined with linear time-varying MPC technique to compensate for front steering failure. In the paper, MPC steering output and fault-tolerant steering output are calculated separately, then combined externally before forwarding to overall control action to the vehicle model. In [25], the fault-tolerant MPC method is proposed for quad-rotor helicopters to detect and isolate the faulty propeller with moving

horizon estimation and unscented Kalman filter methods in the case of loss of effectiveness of the actuator. Then MPC tries to recover the system by updating its control input matrix on each iteration step. In [26], a nonlinear MPC for vehicle lateral stabilization with rear axle torque vectoring of an electric vehicle is presented. The real-time applicability of the proposed method is verified in the Carmaker simulation environment.

1.3 Control Allocation for Vehicle Lateral Stability Studies

With increasing government restrictions about carbon emission, automobile manufacturers are forced to use green energy in their products. As a result, EV culture becomes more dominant against the traditional IC engines [27]- [29]. The introduction of new EV powertrain alternatives resulting new opportunities to improve the overall driving performance of attracting more researchers to focus more on vehicle control using mechatronic components such as electric actuators and brakes [7]. Research about stability control for four-wheel steering and four-wheel traction systems is more attractive because of the flexibility of the driveline architecture.

There are several examples of novel control methodologies for four-wheel-drive vehicle systems. Temiz et al. [17] propose an adaptive control allocation method for 4WD-4WS vehicles to improve the yaw control. The algorithm allocates the driver reference force to each wheel, corrects front steering and rear-wheel steering angles. In [30], the torque allocation problem of a 4WD vehicle is formulated as an optimization problem using allocation errors. Then, an improved version of SQP is proposed to solve this allocation for direct yaw-moment control of the vehicle. In [28], a control strategy that creates cooperation between active front-wheel-steering and direct yaw-moment control is proposed to improve the lateral stability of a four-wheel drive and four-wheel independent steering (4WD-4WIS) vehicle. DMAW is used to adjust the front wheel steering angle, and SQP

controls the tire forces. In [31], a control allocation technique based on quadratic programming around the linear vehicle model is examined for desired yaw rate trajectory tracking and minimizing vehicle sideslip angle. In [18], an adaptive control allocation method is developed in the presence of sensor and actuator communication delay for a 4WD-4WS electric vehicle. In [32], an SQP algorithm is used to solve the power management problem of shipboard under hardware limitations and operational constraints. This paper is an excellent example of how an SQP optimizer can be implemented as a real-time problem solver. In [33], the utilization of computing power at different portions of the automotive controller network and improving controller performance was discussed.

1.4 Motivation and Contribution

There are three contributions of this thesis regarding the modeling and control of a four-wheel-drive and four-wheel-independent-steering electric vehicle:

First, mechanical and mechatronic built of 1:8 scaled and four-wheel-drive and four-wheel-independent-steering electric test vehicle is presented. Then, identification of cornering coefficients of the proposed test vehicle with three different estimation methods are presented: physics-informed parameter identification, traditional neural network and Pacejka’s model-based estimations from the literature. The estimation performance of algorithms are measured with set of control implementation experiments with design of H_∞ regulators.

Secondly, mechanical components of the steering actuators wear out over time due to the continuous motion and multi-axial loads and forces caused by both the frictional force of tire-road contact and the vehicle’s weight. This wear may restrict the motion transmission and result in jamminf of steering actuators or it may reduce contact surface area between mechanical components, which causes a loss of effectiveness of the steering control input angle command. A fault-tolerant reconfigurable model predictive control (RMPC) approach is proposed for a four-wheel-drive four-wheel-steering electric vehicle to combine model predictive

control and fault estimation methods for real-time vehicle lateral stability control under multiple steering actuator faults. First, the magnitude of the faults on each actuator is estimated with input and output data of the plant using a modified vehicle state-space model. Then, extracted fault information is forwarded to the RMPC algorithm for fault compensation. Stability, convergence, and feasibility conditions are studied for every step and concurrently for the proposed method.

Conventionally, robust control and adaptive model predictive control methods are implemented in systems to compensate for the effects of failure. Robust control is classified as a passive fault-tolerant control method as it can only guarantee stability and tracking performance with predefined bounded parametric uncertainties of the control input matrix of the system using pre-ordered controller gain. On the other hand, Adaptive MPC is classified as an active fault-tolerant control method since it can adjust control gains online with a predefined adaptation law [34], [35]. Unlike these conventional methods, the RMPC method also considers the actuator jamming situation, it can interfere not only with the loss of effectiveness but also with controller parameters and MPC optimization constraints in case of jamming.

There are two main contributions of the proposed multiple fault estimation system and reconfigurable model predictive controller on vehicle safety and lateral stability improvements: (1) By extending the current model-based fault estimation techniques, a novel multiple and concurrent fault estimation observer formulation is introduced. (2) The proposed reconfigurable model predictive controller (RPMC) can compensate for concurrent multiple actuator failure events and actuator failure modes (loss of effectiveness and actuator jam) by updating the control input matrix and constraints of MPC in every iteration of the control loop. To the best of our knowledge, this study is the first of its kind in its comprehensive attempt to outline a method to develop a fault-tolerant multi-input multi-output (MIMO) reconfigurable model predictive controller for four-wheel-drive four-wheel-independent-steering vehicles that overcomes combination of multiple loss of effectiveness and jam types of failure simultaneously. The proposed algorithm with flexible MPC formulation has proven to work in real time with step sizes representative of the current vehicle controllers.

Lastly, a computationally efficient two-path SQP control allocation method is proposed to improve the yaw stability of a vehicle equipped with a 4WD-4WIS system. The proposed method is derived from a single, larger, and more complex non-linear optimization problem called the single-path approach, containing eight allocation variables. The current complexity of this optimization problem and its constraints can be problematic for real-time implementations. Therefore, this complex non-linear minimization problem is separated into two smaller, simpler optimization problems solved in parallel: steering and torque allocation. The steering allocation problem is non-linear (due to sinusoidal functions) but involves half of the allocation variables, therefore simpler. The torque allocation problem has linear constants once the steering is fixed and it also has four allocation parameters. The proposed method is analyzed in terms of reference tracking performance and real-time problem-solving speed to demonstrate the effectiveness of the optimal controller over the single-step system.

1.5 Organization of the Thesis

The outline for the remaining chapters of this thesis is as follows: In Chapter 2, the built of mechanical and mechatronic architecture of the 1:8 scaled test vehicle used throughout the modeling and control experiments are explained. Additionally, bicycle and two-track vehicle models used to develop control algorithms are presented. In Chapter 3, cornering coefficients of the bicycle model of the test vehicle are estimated with physics-informed deep learning method, Pacejka's tire model regression method and traditional neural network estimation method. In Chapter 4, the proposed fault-tolerant model predictive optimal control algorithm is explained in detail. The controller's performance is validated in both simulations and processor-in-the-loop tests with high fidelity vehicle model representing the actual test vehicle. In Chapter 5, a computationally efficient nonlinear optimal steering and traction control allocation method for four-wheel-drive and four-wheel-independent-steering vehicles is proposed to improve vehicle lateral stability.

Chapter 2

4-Wheel-Drive and 4-Wheel-Independent-Steering Scaled Electric Test Vehicle

This chapter introduces four-wheel-drive and four-wheel-independent-steering electric 1:8 scaled test vehicle that is used as experimental rig for our proposed parameter identification methods (See Chapter 3) for data collection and fault-tolerant model predictive controller optimal vehicle lateral stability control regulator (See Chapter 4) as controller code generation and real-life implementation, processor-in-the-loop tests and real-time applicability check.

This chapter is structured as follows: In section 2.1 literature review about existing test rigs used to implement vehicle control algorithms are discussed. Section 2.3 presents single-track and two-track vehicle models being used to develop modeling and control algorithms formulated in next chapters. Lastly, in Section 2.3 the test vehicle developed in this study is explained in detail.

2.1 Literature Review

Automotive is one of the rapidly emerging sector among the global industry. Frequent progress and fast pace of the innovations and researches regarding the development of the performance of the vehicles with novel control algorithms require a huge demand of rapid testing and prototyping in this field. Although there are numerous large-scale test systems available for the giants of the industry, there are limited number of non-standardised small-scale solutions available for university research groups. These test rigs are usually used to implement autonomous system features such as perception, path planning, path following.

There are some examples of these scaled test vehicles being used to implement vehicle control algorithms. In [36], the author introduces an autonomous car called "JetRacer-4WS" having four-wheel-steering capability that is still uncommon in the industry despite its great contribution to improve vehicle lateral stability. Then, an event-triggered model predictive controller is implemented on this vehicle to prove that the proposed vehicle can be used as a test platform. In [37]- [39], autonomous scaled test vehicles equipped with high-end computers and numerous onboard sensors are presented to the literature. However, the main focus in these vehicles is to improve the perception and planning algorithms, excluding the vehicle lateral stability motion control algorithms. Also, they all have front wheel ackermann steering mechanism and a single BLDC motor for traction. Hence, there is a need for a test vehicle in the industry that is solely focusing on the motion control part featured with four-wheel-independent-drive and four-wheel-independent-steering mechanism.



2.2.1 Vehicle Linear Bicycle Model

11

δ_r represent front and rear wheel steering angles. v_x and v_y are the vehicle's center of mass velocities in x and y directions in local coordinates. V is the vector summation of them and β is their inverse tangent. R is the turning radius and O is the center of rotation. r is the yaw rate of the center of mass of the vehicle and F_{ij} are forces acting on the front and rear wheels where $i = x, y$ and $j = 1, 2$. Linearization of this model is required to design such controllers. Hence, by assuming small tire slip angles we obtain a state-space form from (2.1)-(2.5). Cornering coefficient (C_{af} and C_{ar}) estimation is the biggest challenge throughout the linearization process since it requires experiments with vehicle and post-processing (for example curve fitting and finding the slope of the linear region in Pacejka) the collected data to identify those parameters.

$$\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bu} \quad (2.1)$$

$$\mathbf{x}^T = \begin{bmatrix} v_y & r \end{bmatrix} \quad (2.2)$$

$$\mathbf{u}^T = \begin{bmatrix} \delta_1 & \delta_2 \end{bmatrix} \quad (2.3)$$

$$\mathbf{A} = \begin{bmatrix} -\frac{C_{af} + C_{ar}}{mV_0} & \frac{bC_{ar} - aC_{af}}{mV_0} - V_0 \\ \frac{bC_{ar} - aC_{af}}{I_z V_0} & -\frac{a^2 C_{af} + b^2 C_{ar}}{I_z V_0} \end{bmatrix} \quad (2.4)$$

$$\mathbf{B} = \begin{bmatrix} \frac{C_{af}}{m} & \frac{C_{ar}}{m} \\ \frac{aC_{af}}{I_z} & -\frac{bC_{ar}}{I_z} \end{bmatrix} \quad (2.5)$$

Figure 2.2 is an example of the response of our linear vehicle model to sinusoidal steering input. The accuracy of the data collected from the sensors (v_y and r) will be compared with this graph.

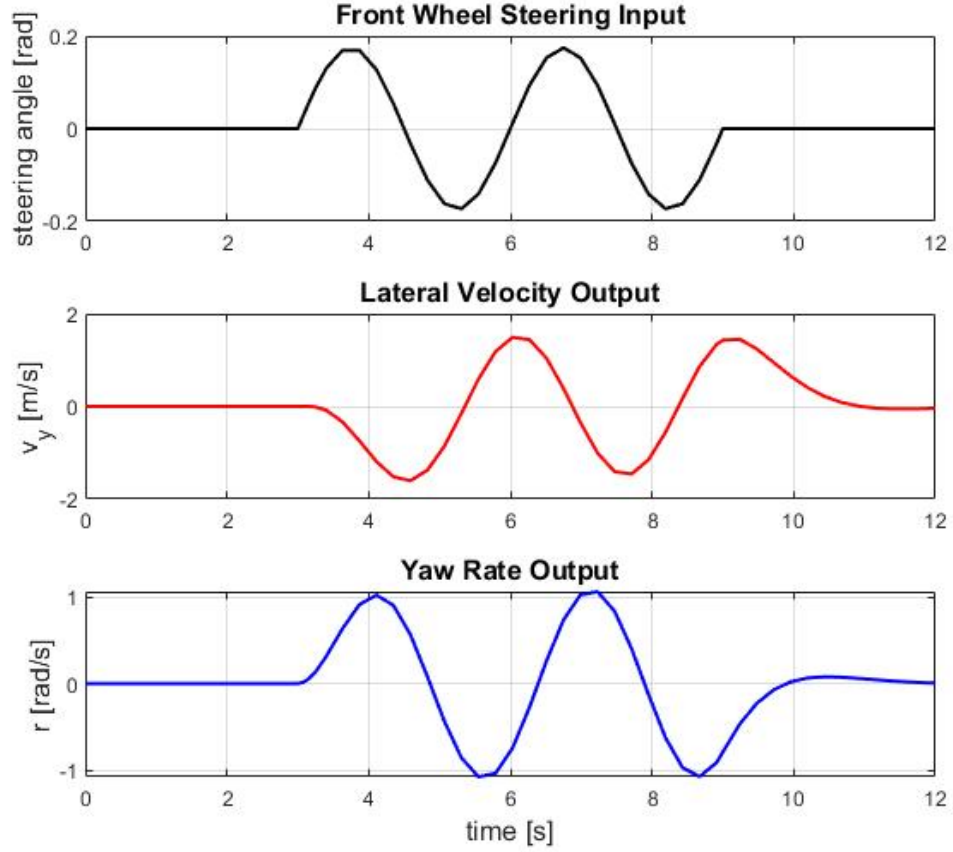


Figure 2.2: Typical Simulation Output

2.2.2 Two Track Vehicle Model

This section discusses the mathematical formulation of the four-wheel-drive and four-wheel-steering electric vehicle shown in Fig. 2.3.

Using the notations from Fig. 2.3, a non-linear four-wheel-steering and four-wheel-traction vehicle model can be obtained by using the Newton-Euler equations in translational (x -axis), lateral (y -axis), and the yaw motion (ψ -axis) as presented in (2.6)-(2.8).

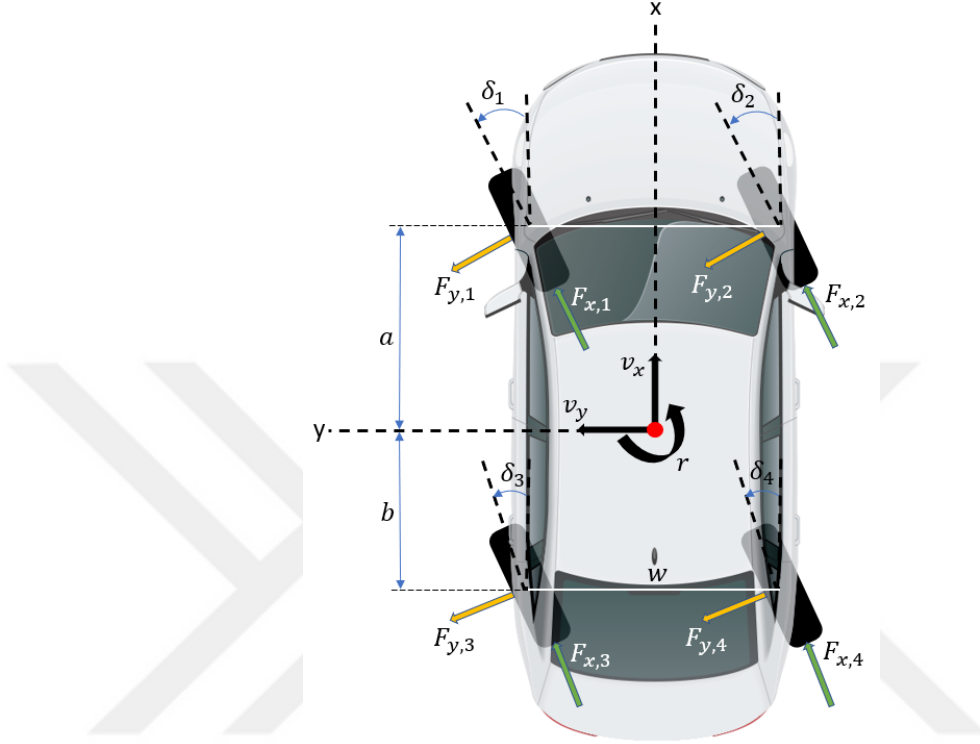


Figure 2.3: Vehicle Two-Track Model

$$\begin{aligned}
 \dot{v}_x = v_y r + \frac{1}{m} \{ & F_{x,1} \cos(\delta_1) + F_{x,2} \cos(\delta_2) \\
 & - F_{y,1} \sin(\delta_1) - F_{y,2} \sin(\delta_2) + F_{x,3} \cos(\delta_3) \\
 & + F_{x,4} \cos(\delta_4) - F_{y,3} \sin(\delta_3) - F_{y,4} \sin(\delta_4) \\
 & + \frac{1}{2} C_d A v_x^2 \}
 \end{aligned} \tag{2.6}$$

$$\begin{aligned}
 \dot{v}_y = -v_x r + \frac{1}{m} \{ & F_{x,1} \sin(\delta_1) + F_{x,2} \sin(\delta_2) \\
 & + F_{y,1} \cos(\delta_1) + F_{y,2} \cos(\delta_2) + F_{x,3} \sin(\delta_3) \\
 & + F_{x,4} \sin(\delta_4) + F_{y,3} \cos(\delta_3) + F_{y,4} \cos(\delta_4) \}
 \end{aligned} \tag{2.7}$$

$$\begin{aligned}
\dot{r} = \frac{1}{I_z} \{ & a[F_{x,1} \sin(\delta_1) + F_{x,2} \sin(\delta_2) \\
& + F_{y,1} \cos(\delta_1) + F_{y,2} \cos(\delta_2)] - b[F_{x,3} \sin(\delta_3) \\
& + F_{x,4} \sin(\delta_4) + F_{y,3} \cos(\delta_3) + F_{y,4} \cos(\delta_4)] \\
& + \frac{w}{2} [-F_{x,1} \cos(\delta_1) + F_{x,2} \cos(\delta_2) - F_{x,3} \cos(\delta_3) \\
& + F_{x,4} \cos(\delta_4) + F_{y,1} \sin(\delta_1) - F_{y,2} \sin(\delta_2) \\
& + F_{y,3} \sin(\delta_3) - F_{y,4} \sin(\delta_4)] \}
\end{aligned} \tag{2.8}$$

In (2.6)-(2.8), tire forces, $F_{x,i}$ and $F_{y,i}$, are assumed to be linear, i.e.:

$$\begin{aligned}
F_{x,i} &= C_{x,i} \lambda_i \\
F_{y,i} &= C_{\alpha,i} \alpha_i
\end{aligned} \tag{2.9}$$

for $i = \{1, 2, 3, 4\}$, where λ_i is the longitudinal slip coefficient of each tire. The front and rear lateral tire slip coefficients (α_j and α_k respectively) can be defined as:

$$\begin{aligned}
\alpha_j &= \delta_j - \frac{v_y + ar}{v_x} \\
\alpha_k &= \delta_k - \frac{v_y - br}{v_x}
\end{aligned} \tag{2.10}$$

for $j = \{1, 2\}$ and $k = \{3, 4\}$.

In order to develop MPC and estimate faults (See Chapter 4), a control-oriented state-space model is derived by linearization of (2.6)-(2.8) around zero steering angle for all tires, i.e., $\delta_i \approx 0$ for $i \in \{1, 2, 3, 4\}$. Constant longitudinal velocity, i.e., v_0 , is assumed to decouple the longitudinal and lateral dynamics of the vehicle. The state vector of the model is determined as:

$$\mathbf{x}^T = \begin{bmatrix} v_x & v_y & r \end{bmatrix} \tag{2.11}$$

Control input vector becomes:

$$\mathbf{u}^T = \begin{bmatrix} \delta_1 & \delta_2 & \delta_3 & \delta_4 & F_{x,1} & F_{x,2} & F_{x,3} & F_{x,4} \end{bmatrix} \quad (2.12)$$

The continuous-time state-space equations of this two-track linear vehicle model can be given as:

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}_c \mathbf{x} + \mathbf{B}_c \mathbf{u} \\ \mathbf{y} &= \mathbf{C}_c \mathbf{x} + \mathbf{D}_c \mathbf{u} \end{aligned} \quad (2.13)$$

$$\mathbf{A}_c = \begin{bmatrix} 0 & 0 & 0 \\ 0 & \mathbf{A}'_c & \\ 0 & & \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -\frac{C_{af}+C_{ar}}{mv_0} & \frac{bC_{ar}-aC_{af}}{mv_0} - v_0 \\ 0 & \frac{bC_{ar}-aC_{af}}{I_z v_0} & -\frac{a^2 C_{af} + b^2 C_{ar}}{I_z v_0} \end{bmatrix} \quad (2.14)$$

$$\mathbf{B}_c^T = \begin{bmatrix} \mathbf{B}_{c,1}^T & \mathbf{B}_{c,2}^T \\ \mathbf{B}_{c,3}^T & \mathbf{B}_{c,4}^T \end{bmatrix} = \begin{bmatrix} 0 & \frac{C_{af}}{m} & \frac{aC_{af}}{I_z} \\ 0 & \frac{C_{af}}{m} & \frac{aC_{af}}{I_z} \\ 0 & \frac{C_{ar}}{m} & -\frac{bC_{ar}}{I_z} \\ 0 & \frac{C_{ar}}{m} & -\frac{bC_{ar}}{I_z} \\ \frac{1}{m} & 0 & -\frac{w}{2I_z} \\ \frac{1}{m} & 0 & \frac{w}{2I_z} \\ \frac{1}{m} & 0 & -\frac{w}{2I_z} \\ \frac{1}{m} & 0 & \frac{w}{2I_z} \end{bmatrix} \quad (2.15)$$

In Equation (2.15), $\mathbf{B}_{c,1}$ and $\mathbf{B}_{c,3}$ have dimension of (1×4) , and $\mathbf{B}_{c,2}$ and $\mathbf{B}_{c,4}$ have dimension of (2×4) . The output ($\mathbf{y}$) of the model is equal to states ($\mathbf{x}$) of the system, i.e., $\mathbf{C}_c = \mathbf{I}_3$ and $\mathbf{D}_c = \mathbf{0}_{3 \times 8}$. For controller development, a discrete version of this model is also obtained, as discussed in Section 4.1. A complete derivation of linear and nonlinear models can be accessed from [18].

2.3 Test Vehicle

This section contains detailed information about both electronic and mechanical design of the 1:8 scaled test vehicle that is built to develop and implement modeling and lateral stability control algorithms. Figure 2.4 demonstrates the scaled test vehicle with different perspectives. The proposed vehicle is used with different electronic component combinations in Chapter 3 and 4 that will be explained in detail later in related chapters.

2.3.1 Mechanical Design

Mechanical components of the vehicle is produced mostly with 3D printer. It has a rigid aluminum main chassis. Rest of the parts such as upper chassis, steering mechanism, suspension wishbones, wheel hub, connection rods and servo motor hubs are PLA material. Design of all components are drawn with Solidworks. Table 2.1 lists the mechanical components of the vehicle.

Component	Material	Quantity
Main chassis	Aluminum	1
Upper chassis	PLA	2
BLDC in wheel motor hub	PLA	4
Steering connection rod	PLA	4
Upper wishbone	PLA	4
Lower wishbone	PLA	4
Wheel-BLDC motor joint	PLA	4
Suspension damper and spring	Purchased	4
Wheel	Purchased	4
Ball joint	Purchased	16

Table 2.1: List of Mechanical Components

Figure 2.5 presents the main chassis and upper chassis components. The upper chassis provides connection between the rigid body of the vehicle and the moving parts.

Figure 2.6 and 2.7 shows components of quarter car and half car respectively. Ball joints combined with steering connection rod links individual servo motors and the wheel hub that completes steering mechanism of quarter car. Lower wishbone, upper wishbone and passive suspension connects the wheel hub to the upper chassis and provides $\pm 5[cm]$ vertical motion that compensates road noise and obstacles at certain level. BLDC motors are mounted on each wheel hub that directly drives individual wheels independent from others.

2.3.2 Electronic System Architecture

2.3.2.1 Hardware Components

The custom built 1:8 scaled has 4 servo motors to steer each wheel individually and 4 BLDC motors mounted on each wheel hub in order to make it four wheel independent drivable. Other than the servo and BLDC motors, the vehicle contains three Arduino microcontrollers (Due, Mega, Uno) , a Bosch BNO055 9-axis IMU sensor, one Ublox NEO M8N GPS sensor, encoder sensors mounted on each wheel hub, an SD Card logger, a Bluetooth module and a WiFi module.

In this system Arduino microcontrollers are used to process and transmit collected sensor data, execute feedback and feedforward control algorithms, drive BLDC and servo motors. These microcontrollers are used with different combinations depend on the computational load of the control algorithms. If there is no mathematically heavy operations in the controller such as H_∞ state-space controller gain used in Chapter 3, then Mega and Uno combination is used. On the other hand, if the processing load is high, i.e. solving MPC optimization problem mentioned in Chapter 4 is required, then Due is used standalone which has larger size of flash memory and SRAM and much faster processor on it compared to the other boards. IMU sensor mounted on the center of the gravity (CoG) of the vehicle is used in order to measure yaw rate (r), yaw angle (Ψ) and lateral acceleration (a_y) of the system. GPS sensor mounted on the CoG measures ground speed (V), i.e. vehicle CoM velocity, and course angle (Ψ_{course}),

i.e. direction of the measured velocity. Encoder sensors mounted on each BLDC motor measure each wheels longitudinal velocity. These data will eventually be combined with GPS measurements in order to increase the velocity estimation accuracy. SD card logger connected to Arduino Due or Mega records test dataset (both raw and processed sensor data, motor control input commands, time) in order to evaluate and/or post-process the information. Bluetooth shield and dongle set mounted on Arduino control card adds wireless communication feature to the vehicle in order to drive it with PS3 controller for sensor data collection to post-process and estimate model parameters of the test vehicle and/or to overwrite control algorithms' motor drive commands during the tests with human interference to reduce severity of possible problems. WiFi module adds WiFi capability to the vehicle to communicate with Matlab TCP server real-time (in order to send sensor data to PIDL parameter estimation algorithm running on external computer and receive estimated vehicle parameters to update reference generator block for feedforward control mentioned in Chapter 3.

2.3.2.2 Programming Software and Languages

Throughout this study, vehicle modelling and control algorithms, mentioned in Chapter 3, 4 and 5 are developed in Matlab/Simulink environment. Therefore, Arduino control cards on this test vehicle must be programmed on there also. "Simulink Support Package for Arduino Hardware" toolbox from [40] is used to create and run Simulink models and Matlab scripts on these boards. This toolbox contains libraries of various Arduino sensors, servo motor control and communication protocols as Simulink blocks. With the help of these blocks, the components are programmed according to the desired test scenarios.

Appendix A shows Simulink models that are used throughout this work such as vehicle sensor data collection for parameter identification (Figure A.1 and A.2), sliding mode observer to estimate front and rear tire lateral forces (Figure A.3) and multi-input multi-output feedback controllers implemented on the test vehicle (Figure A.4). These models are then converted to C++ language, which is the native programming language of Arduino boards, via Simulink Coder from [41].

2.3.2.3 Communication Protocols

Figure 2.8 represents the overall communication protocols of the electronic components of the vehicle. The IMU sensor is connected to the Arduino control card with I^2C protocol. The GPS sensor communicates with control cards via *UART* serial protocol. Encoder sensors use interrupt pins of Arduino board to count interrupt per one revolution of wheel hub. It is then converted to wheel rotational speed. SD card logger is connected to the board via SPI protocol. Wifi and bluetooth capabilities are added to the control cards with Arduino shields. The vehicle communicates with Matlab server via TCP communication and the PS3 joystick sends driver commands to the vehicle via bluetooth connection.

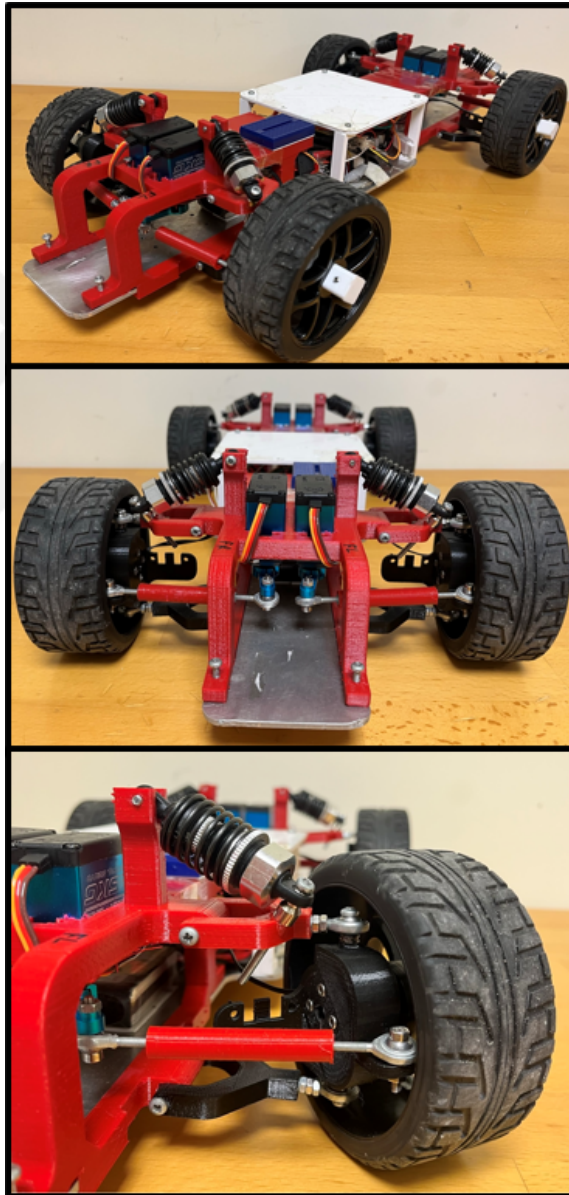


Figure 2.4: Test Vehicle

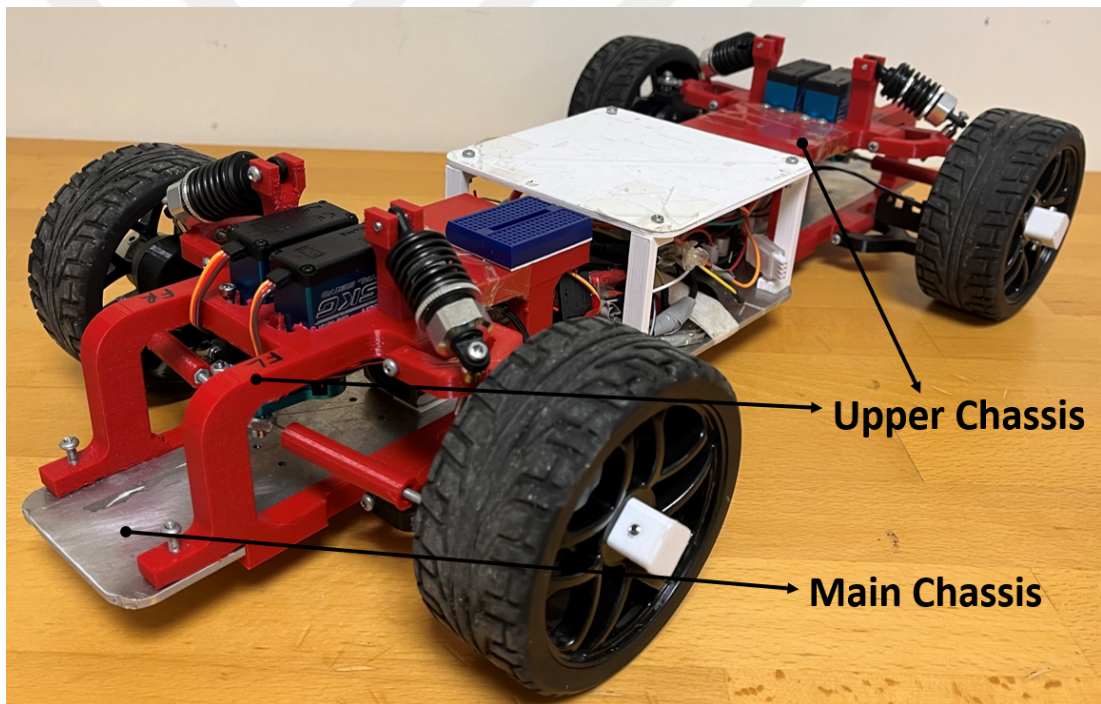


Figure 2.5: Test Vehicle Mechanical Components 1

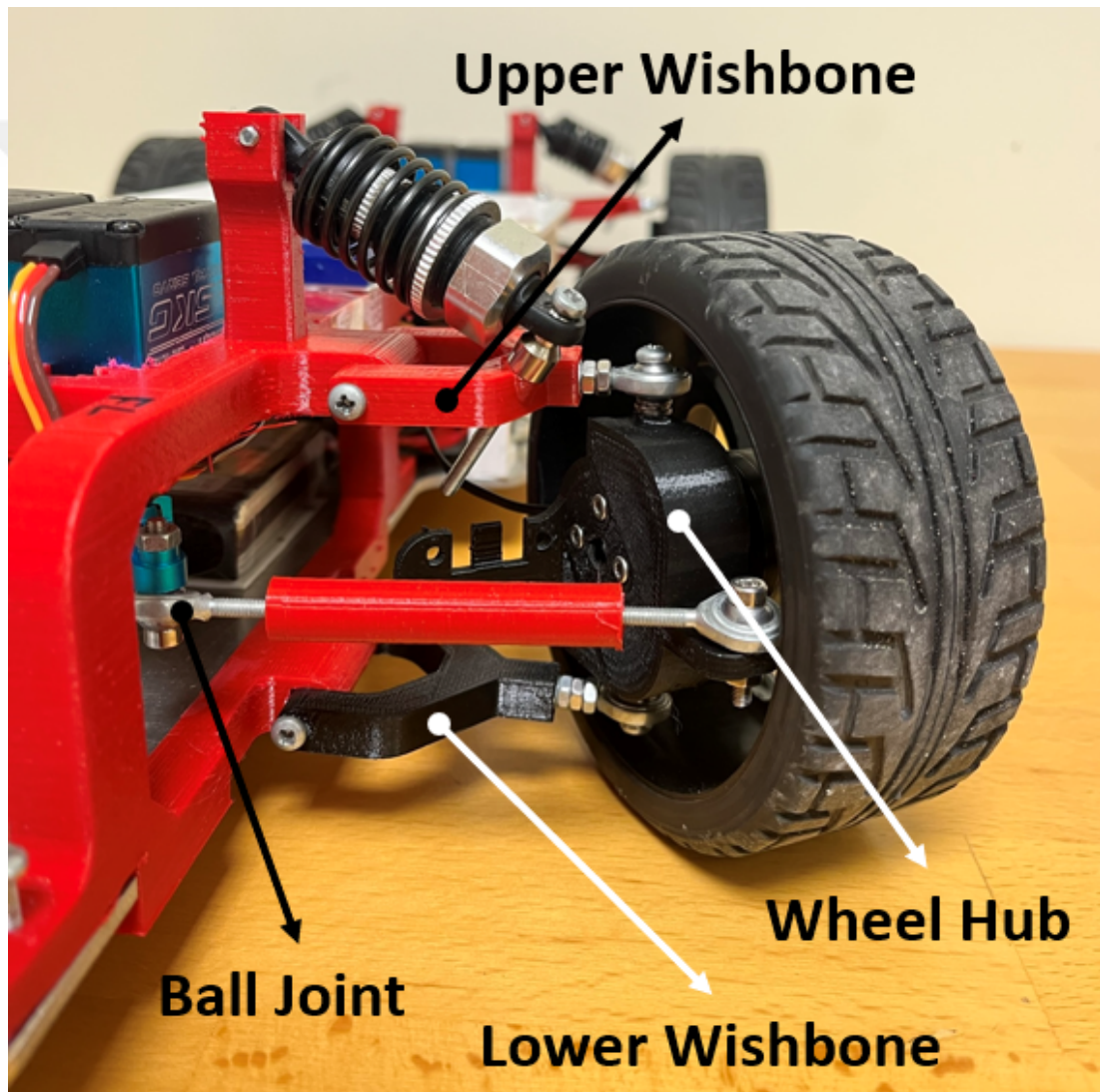


Figure 2.6: Test Vehicle Mechanical Components 2

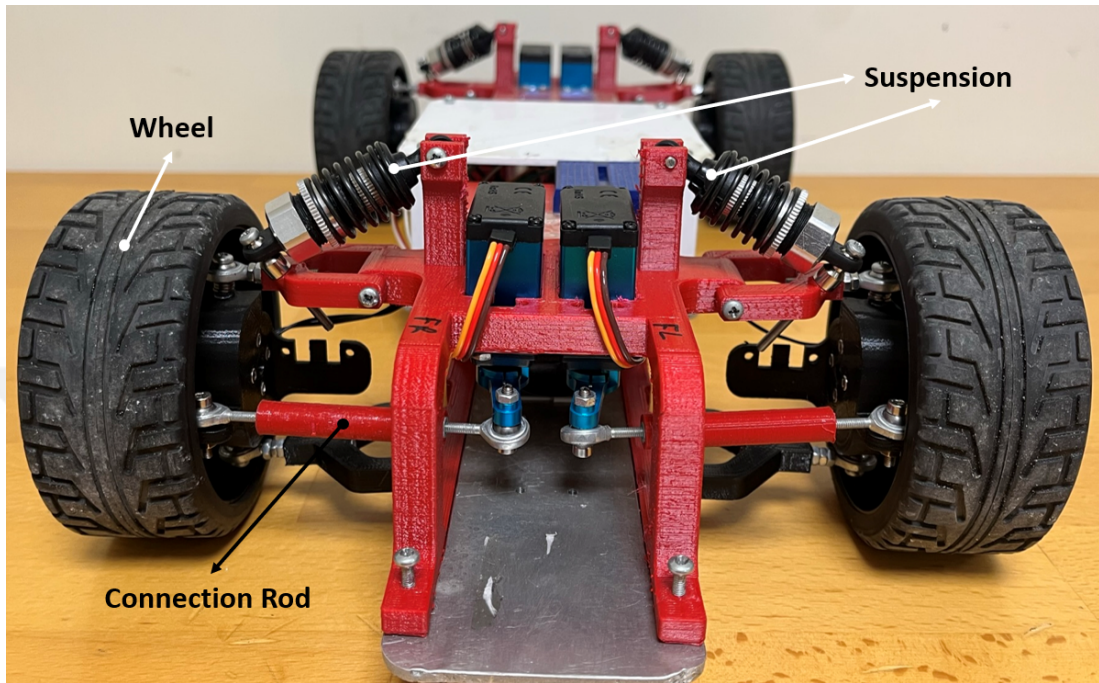


Figure 2.7: Test Vehicle Mechanical Components 3

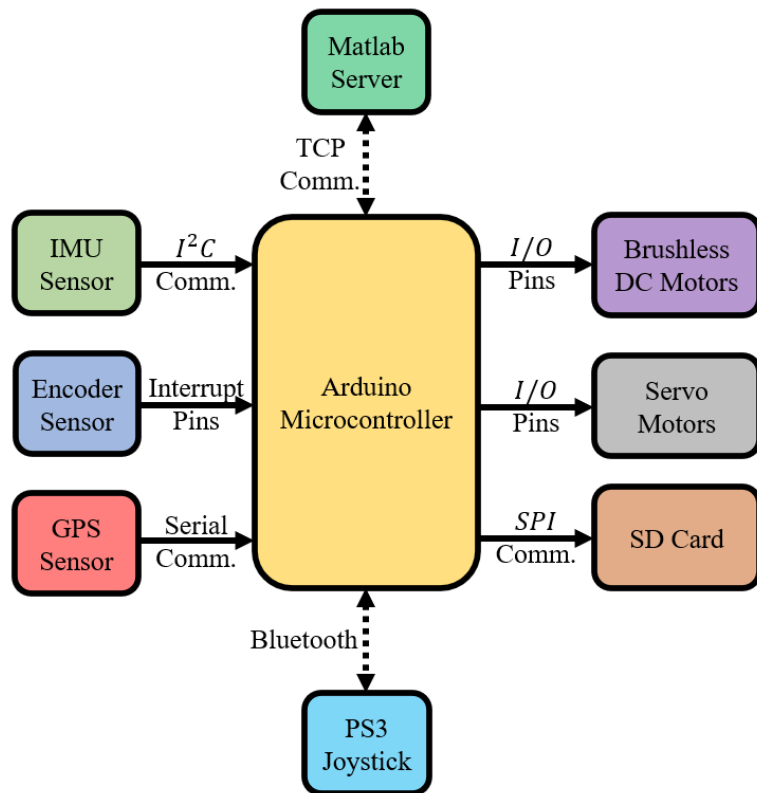


Figure 2.8: Test Vehicle Communication System

Chapter 3

Tire Cornering Parameter Estimation for Vehicle Lateral Stability Algorithms

This chapter presents three separate methods (Pacejka’s tire modeling, traditional neural network identification and physics-informed parameter identification) for identification of vehicle cornering stiffness coefficient, a crucial parameter in vehicle stability control models and control algorithms. The physics-informed parameter identification method enables designers to rapidly identify the vehicle front and rear cornering stiffness parameters so that the controller reference gains can be re-adjusted under varying road and vehicle conditions to improve the reference tracking performance of the control system during operation. In order to conduct experiments, collect sensor data, and real-time communicate with Matlab server running the estimation algorithms wirelessly, custom-built 1:8 scaled test vehicle platform proposed in Chapter 2 is used. The proposed method is applicable to predict derived vehicle parameters such as the understeering coefficient so it can be used in parallel with conventional MIMO controllers. Our H_∞ yaw rate regulation controller test results show that the reference gains updated with online estimation method improve the tracking performance in both simulations and vehicle experiments.

The rest of Chapter 3 is outlined as follows: In Section 3.1 our scaled vehicle prototype and the experimental setup are introduced. Section 3.2 presents the data collection procedures and post-processing of these measurements from sensors. Section 3.3 explains the data collection procedure and prediction of tire properties with a more conventional method given by Pacejka. In Section 3.4 estimation results of physics-informed parameter learning method and traditional neural network are presented. In Section 3.5, first the model estimation accuracy of the physics-informed learning method and conventional deep learning method are compared, then the physics-informed learning method is implemented on H_∞ vehicle yaw-rate regulation algorithm to show its effectiveness with experiments.

3.1 Test Vehicle and Experiments

In order to conduct experiments and collect sensor data to feed the physics-informed learning algorithm, custom-built 1:8 scaled test vehicle (Figure 3.2) mentioned in Chapter 2 is used with specific combination of the electronic components. In this combination, there are two Arduino-based control cards (Mega & Uno) on the vehicle. While the first board collects the sensor data, runs feedback control algorithms and communicates with the Matlab server, with a WiFi module, where learning algorithms are run, the second board provides the motor actuation commands. Each wheel has an optical encoder sensor. Thus, we can measure the wheel rotation speed of the vehicle and the speed of the center of gravity with high accuracy. There is also a 9-axis IMU sensor (BNO055) mounted on the center of the gravity of the vehicle. Parameters of test vehicle in this setup are measured as: $m = 2.15[kg]$, $I_z = 0.085[kgm^2]$, $a = b = 0.17[m]$.

3.2 Data Collection and Processing

This section presents the processing steps of experimental data. For parameter estimation and vehicle modeling purposes, 4 sets of timeseries data of lateral

acceleration and yaw rate from the test vehicle is exported by applying double sinusoidal front steering angle input (δ_f) to the system manually while the vehicle is cruising with $v_x = 1.2[m/s]$ constant velocity (See Figure 3.1).

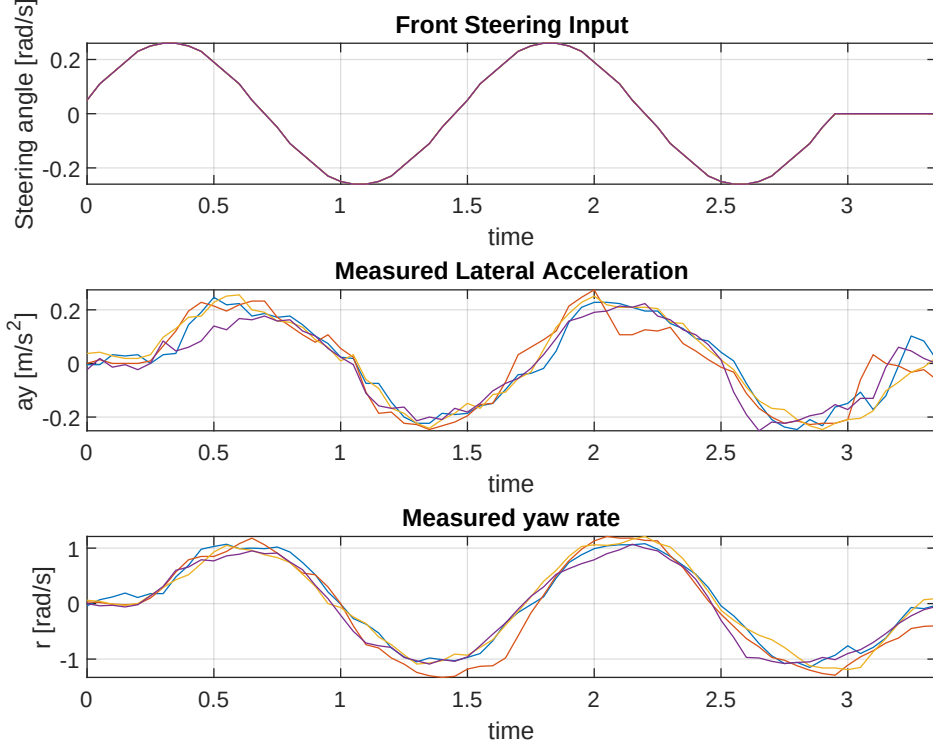


Figure 3.1: Results of Experiments

The collected timeseries data needs to be processed in order to feed the estimators. Hence, first, from the yaw rate and lateral acceleration data, $\dot{v}_y$ is calculated as:

$$\dot{v}_y = a_y - v_x r \quad (3.1)$$

For Pacejka's tire modelling method, tire slip angles (α_f and α_r) and tire lateral forces (F_{yf} and F_{yr}) are required. Therefore, first, Sliding Mode Observer with vehicle bicycle model from [42] is applied to estimate tire lateral forces (See Figure A.3 in Appendix D). Secondly, for front and rear tire slip angle calculation, equation (3.2)-(3.3) are used:

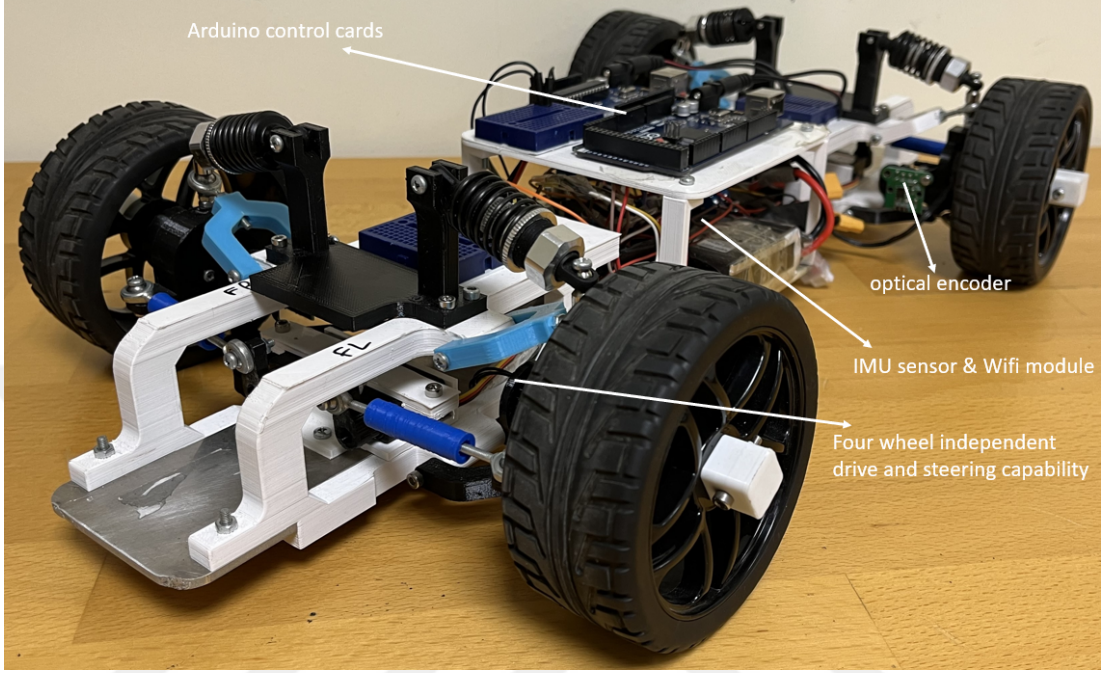


Figure 3.2: Scaled Test Vehicle with 4WD-4WS equipped with inertial measurement sensing and wireless communication capabilities for modeling experiments.

$$\alpha_f = \delta_f - \frac{v_y - ar}{v_x} \quad (3.2)$$

$$\alpha_r = -\frac{v_y - br}{v_x} \quad (3.3)$$

3.3 Pacejka's Tire Modelling

First, results of the most common method called Pacejka tire model (equation 3.4 and 3.5), or Magic tire formula, from [6] that is used for cornering coefficient estimation will be analyzed.

$$F_y = D \sin(C \arctan(B\Phi)) \quad (3.4)$$

$$\Phi = (1 - E)\alpha + (E/B)\arctan(B\alpha) \quad (3.5)$$

In these equations, $D = \max(F_y)$ and $C = 1.30$, B and E are unknown. Using the combination of sets of data collected in Section 3.2, 2 unknown parameters in equation (3.4)-(3.5) are calculated with the least square curve fitting.

Best fitting exponential curves for front and rear tires are shown on Figure 3.3.

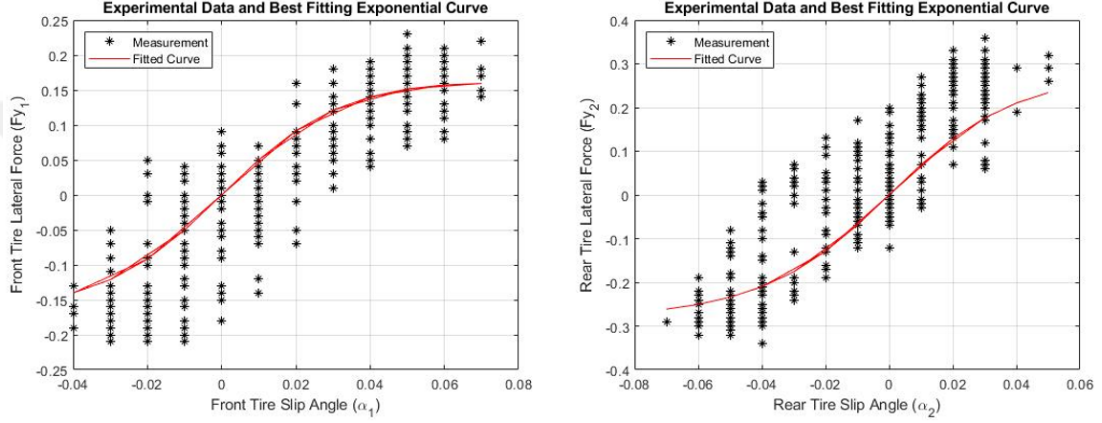


Figure 3.3: Tire Model Regression

Slope of the linear region of these curves, i.e. Pacejka's estimation results, are $C_{af} = 4.59$ and $C_{ar} = 6.45$ respectively. The estimation result obtained using this method is not reliable enough for the control-oriented model. This is because mid-range IMU sensor is mounted on the vehicle to collect data, and because the car was scaled, it is more affected by the roughness of the ground and the magnitude of road noise is high compared to the data collected, i.e. signal-to-noise ratio is high (SNR). Therefore, two learning-based more advanced methods are also used to estimate the tire cornering coefficients.

3.4 Physics-Informed Parameter Learning

The goal of the deep learning algorithms in this system is to estimate the C_{af} and C_{ar} values from the input data, which are r , $\dot{r}$, v_y , $\dot{v}_y$, δ_1 , δ_2 , v_x . Details of both physics-informed deep learning (PIDL) and traditional neural network (TNN) estimation methods are explained in our conference paper [43]. Both networks are fed by the set of collected data in Section 3.2.

From the estimation results of PIDL, C_{af} and C_{ar} are:

$$\begin{bmatrix} C_{af} \\ C_{ar} \end{bmatrix} = \begin{bmatrix} 7.26 & 8.67 & 8.59 & 8.05 \\ 9.83 & 9.45 & 10.35 & 9.21 \end{bmatrix}$$

Similarly, from the estimation results of TNN, C_{af} and C_{ar} are:

$$\begin{bmatrix} C_{af} \\ C_{ar} \end{bmatrix} = \begin{bmatrix} 1.95 & 1.91 & 1.95 & 1.97 \\ 5.41 & 4.89 & 5.23 & 5.38 \end{bmatrix}$$

Again, similar to the Pacejka's method, in TNN, since the noise value is high and there is insufficient amount of data collected by the vehicle to train the neural network, accuracy of the estimations are expected to be poorer. This explains the significant difference between the results of three different methods.

In order to compare the estimation results of three methods, we define an integrated error value for each estimation result formulated as:

$$e = \int (|v - v_{simulated}| + |r - r_{simulated}|)dt \quad (3.6)$$

In this formula, v and r are the actual data collected by the vehicle and $v_{simulated}$ and $r_{simulated}$ are state outputs of vehicle bicycle model with double sinusoidal steering maneuver (same input applied on test vehicle) created by each estimation result of C_{af} and C_{ar} . From this formula, the error values for PIDL estimation results are 2.09,2.81,2.18,3.06 respectively. Similarly, the error values for TNN estimation results are 3.25,4.12,3.77,4.03. Lastly, the error value for Pacejka's tire model regression method is calculated as 3.62. The results show that PIDL yields lower estimation errors compared to other two.

3.5 Controller Implementation with Online Parameter Estimation and Experiments

In this section, we investigate how online estimation can be beneficial to control applications using a MIMO H_∞ controller design example with variable reference signal generation. Figure 3.4 shows the block diagram of the closed system. Our proposed controller has two primary parts: a reference generator, and a MIMO controller. The MIMO controller is designed such that the system is stable to changes in the cornering stiffness tracking performance. The reference generator uses cornering stiffness values to generate the current MIMO controller reference. The parameter values it uses to calculate these commands are updated online estimation from the model-based deep learning algorithm as discussed earlier.

3.5.1 Reference Generator

The driver initiates the reference steering and velocity commands. In reference generator, with driver commands, the desired lateral acceleration gain and yaw rate gain are generated using (3.7) and (3.8) as mentioned in [44].

$$r_{ref} = \delta_{1,in} \frac{v_x}{(a+b) + K_{us}v_x^2} \quad (3.7)$$

$$a_{y,ref} = \delta_{1,in} \frac{v_x^2}{(a+b) + K_{us}v_x^2} \quad (3.8)$$

where understeer coefficient, K_{us} , is given as

$$K_{us} = \frac{mb}{(a+b)C_{af}} - \frac{ma}{(a+b)C_{ar}} \quad (3.9)$$

The learning algorithm updates C_{af} and C_{ar} values according to the data received from the vehicle and keeps the understeering coefficient updated in (3.9)

so that the reference generator block calculates its outputs based on the latest driving conditions.

3.5.2 Optimal MIMO Controller

The optimal MIMO controller shown in Figure 3.4 regulates front and rear steering angles using state error signals, to improve reference tracking. Both states from the vehicle model are used as feedback for the controller. The MIMO H_∞ controllers are designed using mixed-sensitivity minimization with the methods available from sources such as [45] and applicable for a range of cornering stiffness parameters. Input of the controller is set as yaw rate(r) and control outputs are set as front(δ_1) and rear(δ_2) steering.

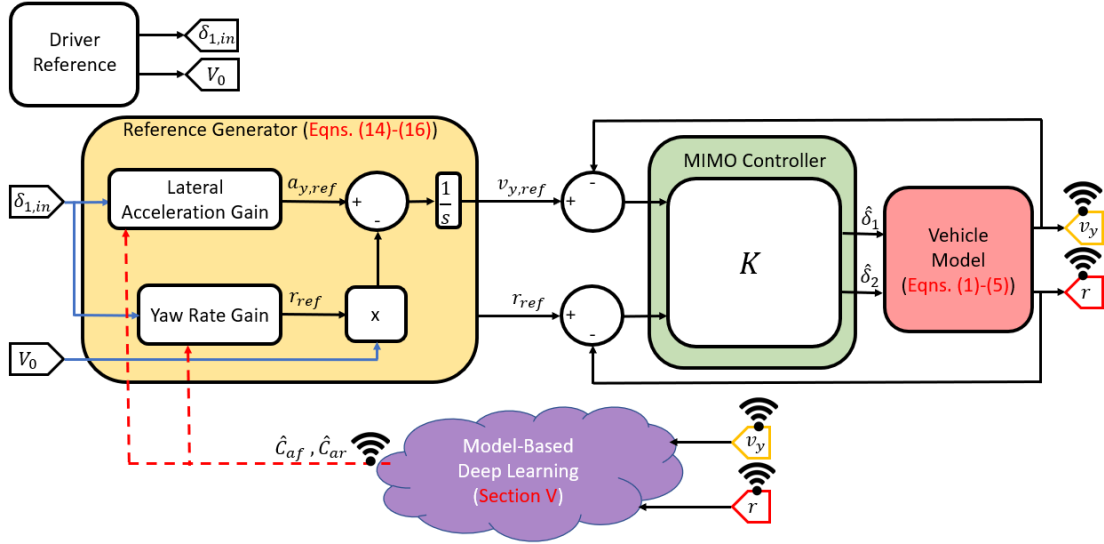


Figure 3.4: Feedback Control with Updated References Block Diagram

3.5.3 Comparison of PIDL and TNN Parameter Estimation Performance

In this section, using the state-space model mentioned in Section 2.2, two MIMO H_∞ controllers are designed using the cornering stiffness parameters estimated with physics-informed deep learning (PIDL) and traditional neural network (TNN) methods. These controllers are then implemented on the test vehicle (Section 3.1) and a typical lane change maneuver test given Figure 2.2 is performed to obtain and compare the error values of estimation results of both learning methods from Section 3.4. Cornering coefficients estimation of PIDL are taken as $C_{af} = 8.14$ and $C_{ar} = 9.71$, and coefficients from RDL are taken as $C_{af} = 1.95$ and $C_{ar} = 5.23$ (i.e. mean of the estimation set).

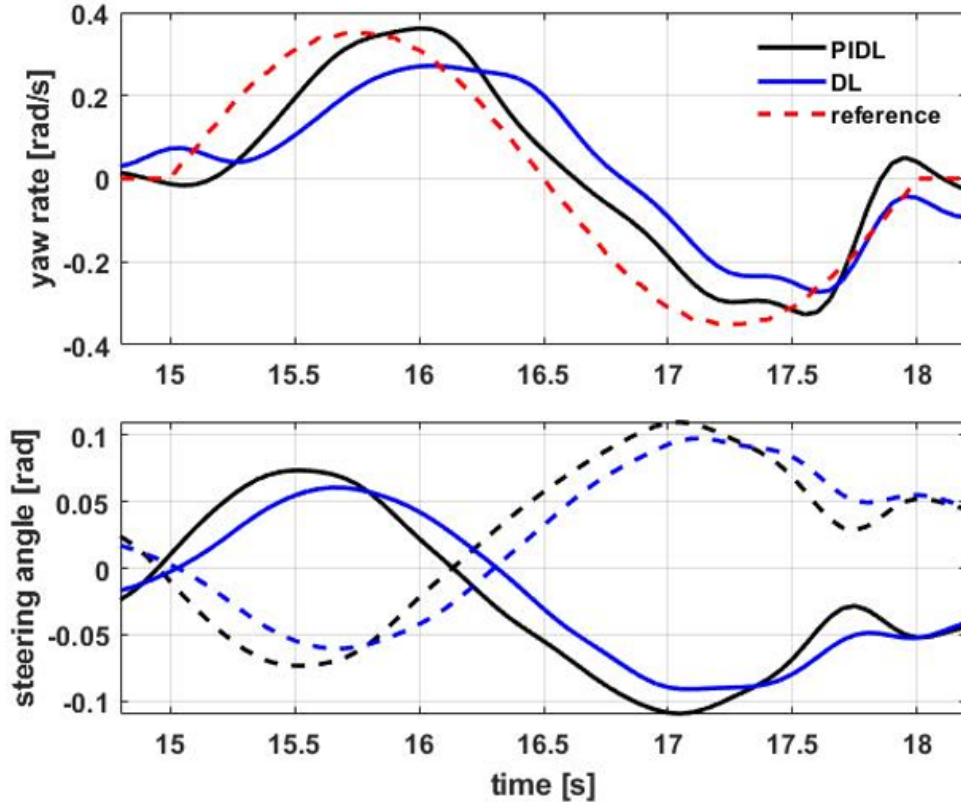


Figure 3.5: Controller Performances (PIDL: physics-informed deep learning, DL: regression deep learning). In bottom figure dotted lines: rear wheel steering angle and continuous lines: front wheel steering angle

Figure 3.5 presents the result of the feedback control experiments on the test vehicle. It shows that the yaw rate reference tracking performances of both controllers. The regulator designed with PIDL cornering parameters has better yaw reference tracking than the TNN. This verifies that PIDL estimation has better modeling accuracy since the controllers designed using this model generate better performance.

3.5.4 PIDL Experiments with Varying Drive Conditions

The discussion in the previous section show that prediction generated by the PIDL algorithm is more accurate even for short burst of data that can be obtained during a lane change maneuver. Therefore, it is also possible to use this method to update vehicle controller parameters online to improve performance.

As an example scenario we start with the conditions such that the test vehicle cruising with constant longitudinal velocity $v_x = 1.2[m/s]$ and the vehicle controller operating with the parameters $C_{af} = 8.14$ and $C_{ar} = 9.71$ as calculated earlier from the experiment data. When the vehicle moves to another surface where road-tire physical interaction is much higher, the cornering stiffness coefficients will change. The optimal MIMO controller can be designed such changes in mind but using the older coefficients in the reference generator will generate a sub-optimal overall controller since the understeering coefficient of the vehicle is changed in the new conditions.

Figure 3.6 shows the results of a such experimental scenario where from the from proposed physics informed deep learning algorithm, the new coefficient values are calculated as $C_{af} = 2.36$ and $C_{ar} = 4.38$. It shows that the control structure without K_{us} reference update deviates from the yaw rate reference and becomes less responsive to the inputs because of higher road-tire interaction that was not compensated for in the outdated reference generator. On the other hand, the control structure with an updated reference value, i.e. online parameter estimation system, is not affected by the road condition change much as its yaw rate tracking capability is better.

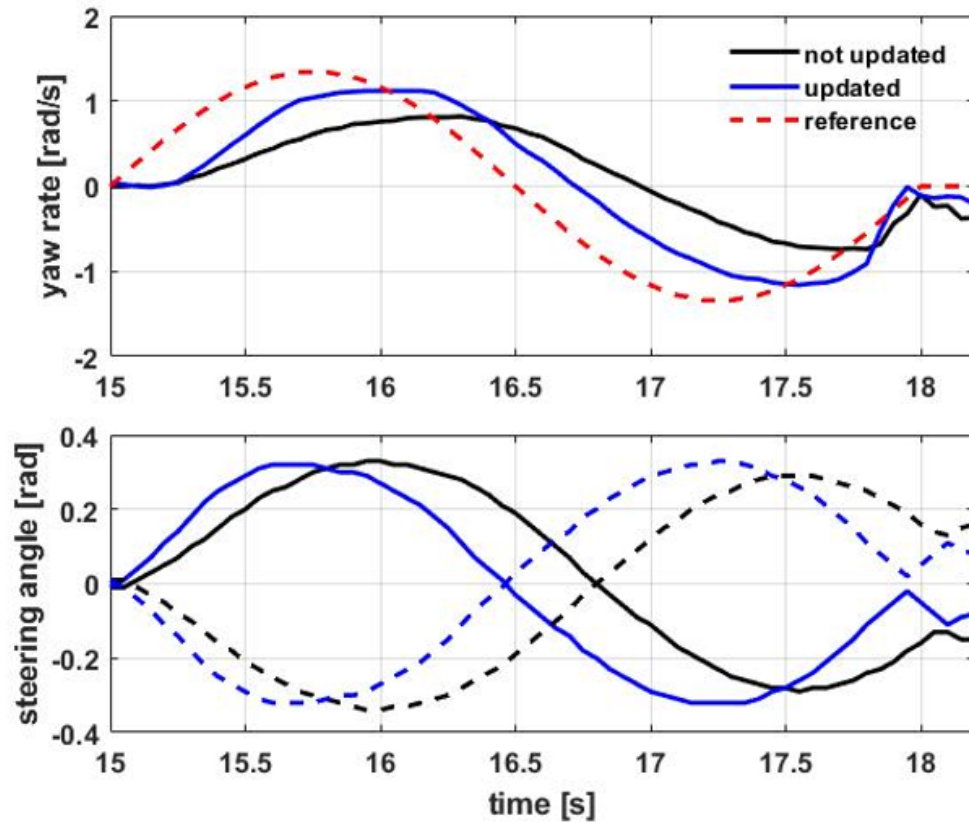


Figure 3.6: Yaw Rate Comparison Experiment (not updated: constant K_{us} system, updated: updated K_{us} parameter with online estimation). In bottom figure dotted lines: rear wheel steering angle and continuous lines: front wheel steering angle

Chapter 4

Fault-Tolerant Model Predictive Controller with Reconfiguration for Optimal Lateral Stability in Over-actuated Electric Vehicles

This chapter presents a fault-tolerant reconfigurable model predictive control method to improve the longitudinal and lateral reference tracking of four-wheel-steering and four-wheel-drive vehicles under concurrent steering actuator faults and fault modes. The proposed method detects, isolates then estimates the magnitude of individual faults. Fault magnitude information is then forwarded to the model-predictive controller, which configures its state-space input matrix and constraints based on the information provided to overcome the adverse effects of these faults that may manifest themselves in the form of both actuator loss of effectiveness and jam. It is shown that the integration of the fault estimation and the reconfiguration procedure does not affect the stability of the control system. The performance of the proposed method is first verified using obstacle avoidance maneuvering in a control-oriented simpler vehicle model. Tests with a Processor-in-the-Loop system (using an experimentally validated high-fidelity

vehicle model) are also conducted to prove the real-time applicability of the proposed control algorithm. Our analysis with different scenarios shows that the proposed reconfigurable fault-tolerant MPC algorithm can improve the performance of the system minimum of 27% (and as much as 86%) compared to the conventional model predictive controller designed for vehicle stability.

This chapter is structured as follows: Section 4.1.1 proposes a multi-input multi-output MPC optimization problem. Section 4.1.2 provides the details of fault detection, isolation, and magnitude estimation under steering actuator failure. In Section 4.1.3, details of the proposed fault-tolerant reconfigurable MPC algorithm are explained. In Sections 4.2.1 and 4.2.2, performance evaluation of the proposed algorithm in a simulation environment and processor-in-the-loop (PIL) test are demonstrated, respectively.

4.1 Control Development

This section presents the proposed reconfigurable fault-tolerant model predictive controller. First, the model predictive control optimization problem for the multi-input-multi-output plant is formulated. Then, simultaneous multiple fault magnitude estimation architecture using the modified state-space model is introduced. Then, a fault mitigation scheme capable of handling multiple simultaneous steering actuator failures is proposed based on the online estimation of the faults and reconfiguration of the predictive controller optimization problem. Lastly, the stability and performance of the proposed controller are discussed.

4.1.1 MIMO Constrained Model Predictive Control Formulation

Fig. 4.1 shows the feedback control diagram of the vehicle using a model predictive controller. MPC block optimizes the rate of change of steering and traction control inputs based on longitudinal velocity v_x , lateral velocity v_y , and yaw rate

r state feedback information by comparing them with reference signals $v_{x,ref}$, $v_{y,ref}$ and r_{ref} . In driver reference generator block, these reference state values are calculated based on the driver's steering angle input and throttle response to complete the intended maneuver. In order to calculate reference states $v_{y,ref}$ and r_{ref} , yaw rate gain and lateral acceleration gain formulas in [7] are used. For longitudinal velocity $v_{x,ref}$ reference, Newton-Euler equation in translational x -axis is used. Details of the reference generation procedure can be accessed in [44]. The MPC block then, forwards optimal steering angles ($\mathbf{u}_{steering}$) and traction forces ($\mathbf{u}_{traction}$) to the vehicle model where

$$\mathbf{u} = \begin{bmatrix} \mathbf{u}_{steering} \\ \mathbf{u}_{traction} \end{bmatrix} \quad (4.1)$$

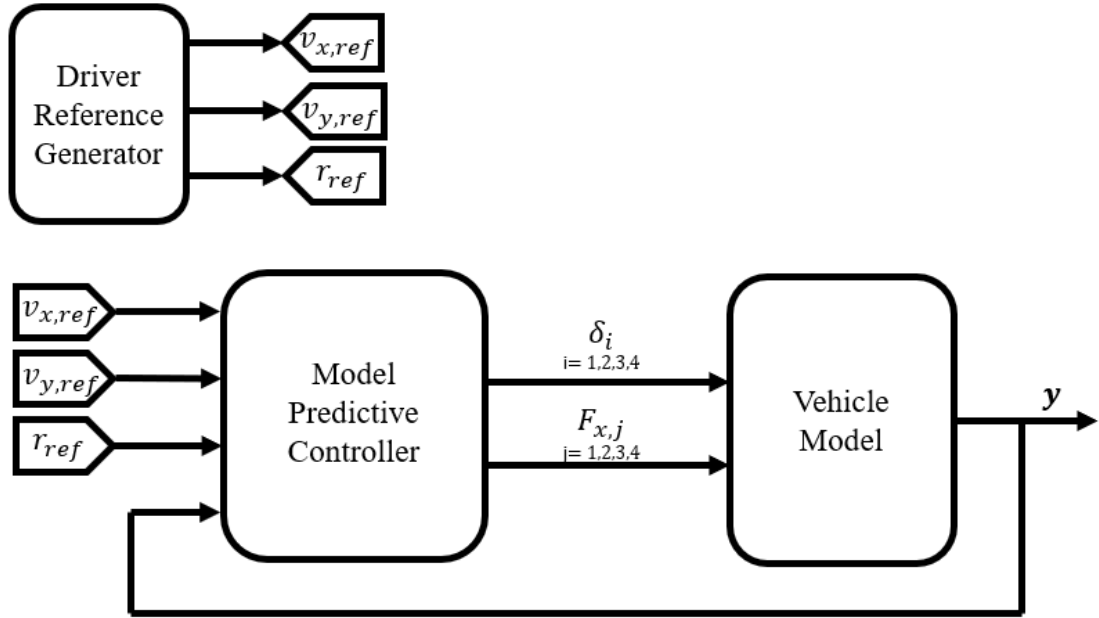


Figure 4.1: Block Diagram for Model Predictive Control Implementation

The model predictive control formulation presented in this section is an extended version of single-input single-output (SISO) formulation from [46]. In order to generate the optimization problem for predictive control, we first discretize our continuous state-space vehicle model given in (2.13). The discrete-time version of this system (Equation (4.2)) can be presented using the matrix notations ($\mathbf{A}_m$, $\mathbf{B}_m$, $\mathbf{C}_m$, $\mathbf{D}_m$).

$$\begin{aligned}\mathbf{x}_m(k+1) &= \mathbf{A}_m \mathbf{x}_m(k) + \mathbf{B}_m \mathbf{u}(k) \\ \mathbf{y}(k) &= \mathbf{C}_m \mathbf{x}_m(k) + \mathbf{D}_m \mathbf{u}(k)\end{aligned}\tag{4.2}$$

In Equation (4.2), k represents time step variable. For plant discretization, the zero-order-hold method in Equation (4.3) is used. In this method, the sample time is defined as T_s . $\mathbf{C}_m = \mathbf{C}_c$ and $\mathbf{D}_m = \mathbf{D}_c$, i.e., they remain the same in both continuous and discrete-time models.

$$\begin{aligned}\mathbf{A}_m &= e^{\mathbf{A}_c T_s} \\ \mathbf{B}_m &= \int_0^{T_s} e^{\mathbf{A}_c \mathbf{x}} d\mathbf{x} \mathbf{B}_c\end{aligned}\tag{4.3}$$

Here $\mathbf{A}_m \in \mathbb{R}^{n_1 \times n_1}$, $\mathbf{B}_m \in \mathbb{R}^{n_1 \times m}$, $\mathbf{C}_m \in \mathbb{R}^{q \times n_1}$ and (n_1, m, q) are the number of states, inputs, and outputs respectively.

Secondly, the MPC method optimizes the input difference $\Delta \mathbf{u}(k)$ instead of input $\mathbf{u}(k)$. Hence, plant augmentation is required to change the plant input from the magnitude of input to the rate of change of input signal. After the augmentation step (Equation (4.4)-(4.6)), the notation is $(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D})$ for the discrete state space system with updated state feedback vector $\mathbf{x}(k)$.

$$\begin{aligned}\mathbf{x}(k+1) &= \mathbf{A} \mathbf{x}(k) + \mathbf{B} \Delta \mathbf{u}(k) \\ \mathbf{y}(k) &= \mathbf{C} \mathbf{x}(k) + \mathbf{D} \Delta \mathbf{u}(k)\end{aligned}\tag{4.4}$$

An extended form of the augmented plant equation is (intermediate calculations for this step are in [46]):

$$\begin{aligned}\begin{bmatrix} \Delta \mathbf{x}_m(k+1) \\ \mathbf{y}(k+1) \end{bmatrix} &= \begin{bmatrix} \mathbf{A}_m & \mathbf{0}_m^T \\ \mathbf{C}_m \mathbf{A}_m & \mathbf{I}_{q \times q} \end{bmatrix} \begin{bmatrix} \Delta \mathbf{x}_m(k) \\ \mathbf{y}(k) \end{bmatrix} \\ &\quad + \begin{bmatrix} \mathbf{B}_m \\ \mathbf{C}_m \mathbf{B}_m \end{bmatrix} \Delta \mathbf{u}(k)\end{aligned}\tag{4.5}$$

$$\mathbf{y}(k) = \begin{bmatrix} \mathbf{0}_m & \mathbf{I}_{q \times q} \end{bmatrix} \begin{bmatrix} \Delta \mathbf{x}_m(k) \\ \mathbf{y}(k) \end{bmatrix} \quad (4.6)$$

After plant discretization and augmentation, a predictive control within the optimization window is described with given state feedback information $\mathbf{x}(k_i)$ at time $k_i > 0$, prediction horizon (N_p) and control horizon (N_c), where $N_c \leq N_p$:

Future control trajectory within control horizon:

$$\Delta \mathbf{u}(k_i), \Delta \mathbf{u}(k_i + 1), \dots, \Delta \mathbf{u}(k_i + N_c - 1)$$

Future state variables within the prediction horizon (i.e. length of optimization window):

$$\mathbf{x}(k_i + 1|k_i), \mathbf{x}(k_i + 2|k_i), \dots, \mathbf{x}(k_i + N_p|k_i)$$

Similarly, from predicted states with predicted control outputs, future output variables become:

$$\mathbf{y}(k_i + 1|k_i), \mathbf{y}(k_i + 2|k_i), \dots, \mathbf{y}(k_i + N_p|k_i)$$

From these outcomes, define the following column vectors:

$$\Delta \mathbf{U} = [\Delta \mathbf{u}(k_i)^T \Delta \mathbf{u}(k_i + 1)^T \cdots \Delta \mathbf{u}(k_i + N_c - 1)^T]^T$$

$$\mathbf{Y} = [\mathbf{y}(k_i + 1|k_i)^T \mathbf{y}(k_i + 2|k_i)^T \cdots \mathbf{y}(k_i + N_p|k_i)^T]^T$$

Note that further explanations and calculations can be found in [46].

As a result of optimization window calculations, we obtain output vector $\mathbf{Y}$, state gain matrix $\mathbf{F}$, input gain matrix Φ , and control input vector $\Delta \mathbf{U}$ within the borders of prediction and control horizons.

Based on the augmented state-space model ($\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, $\mathbf{D}$) developed in (4.4)-(4.6), these vectors and matrices can be combined as:

$$\mathbf{Y} = \mathbf{F}\mathbf{x}(k) + \Phi\Delta \mathbf{U} \quad (4.7)$$

where,

$$\mathbf{F} = \begin{bmatrix} \mathbf{CA} \\ \mathbf{CA}^2 \\ \vdots \\ \mathbf{CA}^{N_p} \end{bmatrix} \quad (4.8)$$

$$\Phi = \begin{bmatrix} \mathbf{CB} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{CAB} & \mathbf{CB} & \dots & \mathbf{0} \\ \mathbf{CA}^2\mathbf{B} & \mathbf{CAB} & \dots & \mathbf{0} \\ \vdots & & \ddots & \\ \mathbf{CA}^{N_p-1}\mathbf{B} & \mathbf{CA}^{N_p-2}\mathbf{B} & \dots & \mathbf{CA}^{N_p-N_c}\mathbf{B} \end{bmatrix} \quad (4.9)$$

It is important to note that, in the MIMO plant, $\mathbf{Y} \in \mathbb{R}^{qN_p \times 1}$, $\mathbf{F} \in \mathbb{R}^{qN_p \times n}$, $\mathbf{x}(k) \in \mathbb{R}^{n \times 1}$, $\Phi \in \mathbb{R}^{qN_p \times mN_c}$, $\Delta\mathbf{U} \in \mathbb{R}^{mN_c \times 1}$, and $n = n_1 + q$ where (n_1, m, q) are the number of states, the inputs, and the outputs respectively.

Next, an optimization problem is defined using (4.7) generated in the optimization window. To calculate the optimal control rate values, $\Delta\mathbf{U}$, a cost function, $\mathbf{J}$, consisting of the reference tracking error and the control input magnitude terms is used:

$$\mathbf{J} = (\bar{\mathbf{R}}_s \mathbf{r}(k) - \mathbf{Y})^T (\bar{\mathbf{R}}_s \mathbf{r}(k) - \mathbf{Y}) + \Delta\mathbf{U}^T \bar{\mathbf{R}} \Delta\mathbf{U} \quad (4.10)$$

where $\bar{\mathbf{R}}_s$ and $\bar{\mathbf{R}}$ are the reference and the control input-related weight matrices, respectively and $\mathbf{r}(k)$ is the reference signal. Combining (4.7) with (4.10) results in:

$$\begin{aligned} \mathbf{J} = & (\bar{\mathbf{R}}_s \mathbf{r}(k) - \mathbf{F}\mathbf{x}(k))^T (\bar{\mathbf{R}}_s \mathbf{r}(k) - \mathbf{F}\mathbf{x}(k)) \\ & - 2\Delta\mathbf{U}^T \Phi^T (\bar{\mathbf{R}}_s \mathbf{r}(k) - \mathbf{F}\mathbf{x}(k)) \\ & + \Delta\mathbf{U}^T (\Phi^T \Phi + \bar{\mathbf{R}}) \Delta\mathbf{U} \end{aligned} \quad (4.11)$$

The reference signal is $\mathbf{r}(k) \in \mathbb{R}^{q \times 1}$. $\Phi^T \bar{\mathbf{R}}_s$ equals to the last q columns of $\Phi^T \mathbf{F}$. Here $\bar{\mathbf{R}}_s$ can be extracted from this equality with the left matrix division operation. $\bar{\mathbf{R}} = \text{diag}(\mathbf{W}_1, \mathbf{W}_2, \dots, \mathbf{W}_{N_c})$ is block diagonal weight matrix with

dimension $\bar{\mathbf{R}} \in \mathbb{R}^{mN_c \times mN_c}$. Each block contains the weight of each actuator at i^{th} control horizon as shown in (4.12).

$$\mathbf{W}_i = \begin{bmatrix} w_1 & & & \\ & w_2 & & \\ & & \ddots & \\ & & & w_m \end{bmatrix}_{m \times m} \quad (4.12)$$

In model predictive control, only the first m inputs, i.e., $\Delta \mathbf{u}$, of solution vector $\Delta \mathbf{U}$ will be applied to the system at the current step:

$$\Delta \mathbf{u} = \begin{bmatrix} \mathbf{I}_m & \mathbf{0}_m & \dots & \mathbf{0}_m \end{bmatrix}_{m \times mN_c} \Delta \mathbf{U} \quad (4.13)$$

Then, the control input forwarded to the plant becomes $\mathbf{u}(k) = \mathbf{u}(k-1) + \Delta \mathbf{u}$ where $\mathbf{u}(k-1)$ previous control input applied to the discrete-time system $(\mathbf{A}_m, \mathbf{B}_m, \mathbf{C}_m, \mathbf{D}_m)$.

Additional constraints that limit optimal control input magnitude and rate of change of the optimal control input can be added to the proposed optimization problem as follows ($\mathbf{A}\mathbf{x} \leq \mathbf{b}$ and $\mathbf{A}_{eq}\mathbf{x} = \mathbf{b}_{eq}$):

For Control Input Rate Limit:

$$\begin{bmatrix} -\mathbf{I}_{mN_c \times mN_c} \\ \mathbf{I}_{mN_c \times mN_c} \end{bmatrix} \Delta \mathbf{U} \leq \begin{bmatrix} -\Delta \mathbf{U}_{min} \\ \Delta \mathbf{U}_{max} \end{bmatrix} \quad (4.14)$$

with $\Delta \mathbf{U}_{min} = \mathbf{C}_1 \Delta \mathbf{u}_{min}$ and $\Delta \mathbf{U}_{max} = \mathbf{C}_1 \Delta \mathbf{u}_{max}$ where $\Delta \mathbf{u}_{min} \in \mathbb{R}^{m \times 1}$ and $\Delta \mathbf{u}_{max} \in \mathbb{R}^{m \times 1}$ are column vectors setting upper and lower bound control rate of each actuator individually and:

$$\mathbf{C}_1 = \begin{bmatrix} \mathbf{I}_m \\ \mathbf{I}_m \\ \vdots \\ \mathbf{I}_m \end{bmatrix}_{mN_c \times m} \quad (4.15)$$

For Control Input Magnitude Limit:

$$\begin{bmatrix} -\mathbf{C}_2 \\ \mathbf{C}_2 \end{bmatrix} \Delta \mathbf{U} \leq \begin{bmatrix} -\mathbf{U}_{min} + \mathbf{C}_1 \mathbf{u}(k-1) \\ \mathbf{U}_{max} - \mathbf{C}_1 \mathbf{u}(k-1) \end{bmatrix} \quad (4.16)$$

with $\mathbf{U}_{min} = \mathbf{C}_1 \mathbf{u}_{min}$ and $\mathbf{U}_{max} = \mathbf{C}_1 \mathbf{u}_{max}$. $\mathbf{u}_{min} \in \mathbb{R}^{m \times 1}$ and $\mathbf{u}_{max} \in \mathbb{R}^{m \times 1}$ are column vectors set upper and lower bounds control input of each actuator individually and:

$$\mathbf{C}_2 = \begin{bmatrix} \mathbf{I}_m & \mathbf{0}_m & \mathbf{0}_m & \dots & \mathbf{0}_m \\ \mathbf{I}_m & \mathbf{I}_m & \mathbf{0}_m & \dots & \mathbf{0}_m \\ \vdots & & & \ddots & \vdots \\ \mathbf{I}_m & \mathbf{I}_m & \mathbf{I}_m & \dots & \mathbf{I}_m \end{bmatrix}_{mN_c \times mN_c} \quad (4.17)$$

Lastly, to solve the optimization problem within the rules of predictive control, (4.11)-(4.17) are converted into the following constrained quadratic programming problem:

$$\begin{aligned} \min_{\Delta \mathbf{U}} \quad & \frac{1}{2} \Delta \mathbf{U}^T \mathbf{H} \Delta \mathbf{U} + \mathbf{f}^T \Delta \mathbf{U} \\ \text{s.t.} \quad & \mathbf{A}_{ineq} \Delta \mathbf{U} \leq \mathbf{b}_{ineq} \end{aligned} \quad (4.18)$$

where $\mathbf{H} = \Phi^T \Phi + \bar{\mathbf{R}}$ is Hessian matrix and assumed to be positive definite, $\mathbf{f} = -\frac{1}{2} \Phi^T (\bar{\mathbf{R}}_s \mathbf{r}(k) - \mathbf{F} \mathbf{x}(k))$ and,

$$\begin{aligned}
\mathbf{A}_{ineq} &= \begin{bmatrix} -\mathbf{I}_{mN_c \times mN_c} \\ \mathbf{I}_{mN_c \times mN_c} \\ -\mathbf{C}_2 \\ \mathbf{C}_2 \end{bmatrix} \\
\mathbf{b}_{ineq} &= \begin{bmatrix} -\Delta \mathbf{U}_{min} \\ \Delta \mathbf{U}_{max} \\ -\mathbf{U}_{min} + \mathbf{C}_1 \mathbf{u}(k-1) \\ \mathbf{U}_{max} - \mathbf{C}_1 \mathbf{u}(k-1) \end{bmatrix}
\end{aligned} \tag{4.19}$$

Many popular quadratic programming solvers can solve this constrained optimization problem, and real-time implementation on hardware is possible.

Stability, Convergence and Feasibility

Closed-loop stability and convergence of MPC law primarily depend on two conditions [47]:

1. If the optimization problem (given in (4.18)) is feasible at time instant $k = 0$, then the state error ($e(k) = x_{ref}(k) - x(k)$) and input trajectories of the discrete state-space model Equation (4.2) under MPC algorithm satisfy $\lim_{k \rightarrow \infty} (\mathbf{e}(k), \mathbf{u}(k)) = (\mathbf{0}, \mathbf{0})$.
2. Under the MPC law, the origin ($\mathbf{x} = \mathbf{0}$) of the discrete state-space model (in Equation (4.2)) is asymptotically stable. If Hessian matrix in Equation (4.18) is positive definite, i.e. $\mathbf{H} > \mathbf{0}$, then $\mathbf{x} = \mathbf{0}$ is exponentially stable.

Tuning parameters of the MPC such as discretization sample time (T_s), prediction (N_p) and control horizon (N_c), control input weight matrices ($\bar{\mathbf{R}}_s$, and $\bar{\mathbf{R}}$) are adjusted by ensuring the stability and feasibility conditions given above.

In every control loop, the quadratic programming (QP) solver calculates the merit function ψ (See Appendix D) created by the matrices of MPC problem

(Equation (4.18)) to check the feasibility of the optimization problem [48]. If the merit function becomes large, then the problem becomes infeasible, i.e., no solution exists with given constraints. In the formulation presented in (4.18)-(4.19) actuator magnitude and rate constraints are chosen such that the merit function does not become large. Therefore, the problem remains inside the feasible region.

4.1.2 Fault Detection, Isolation, and Magnitude Estimation

This section presents an improved model-based fault detection, isolation, and magnitude estimation (FDIME) method adapted to the linear vehicle model (Section 2.2.2) to detect and isolate faulty steering actuators and then estimate the magnitude of multiple steering actuator failures. Similar methods proposed in references such as [34, 35], multiple fault isolation and magnitude estimation are not covered, i.e., only single wheel fault estimation is possible with the methods in these references. The FDIME formulation is improved by introducing new states, representing electrical signals to drive steering actuators, to the linearized vehicle model developed in the previous sections.

In the presence of a single actuator failure, our discrete-time state-space model (4.2), affected by a faulty actuator with unknown input $f_{d,i}$, can be written as:

$$\hat{\mathbf{x}}_m(k) = \mathbf{A}_m \hat{\mathbf{x}}_m(k-1) + \mathbf{B}_m \mathbf{u}(k-1) + \mathbf{F}_{d,i} f_{d,i}(k-1) \quad (4.20)$$

In his equation, $\mathbf{F}_{d,i}$ is a vector equal to the i^{th} column of $\mathbf{B}_m$ matrix where i is the number of the steering actuator with failure, and $f_{d,i}$ is the magnitude of the failure to be estimated if it exists.

Fig. 4.2 shows the block diagram of the proposed diagnosing methodology that monitors the system behavior online by processing steering angle input commands $\mathbf{u}_{steering}$ and outputs $\mathbf{y}$.

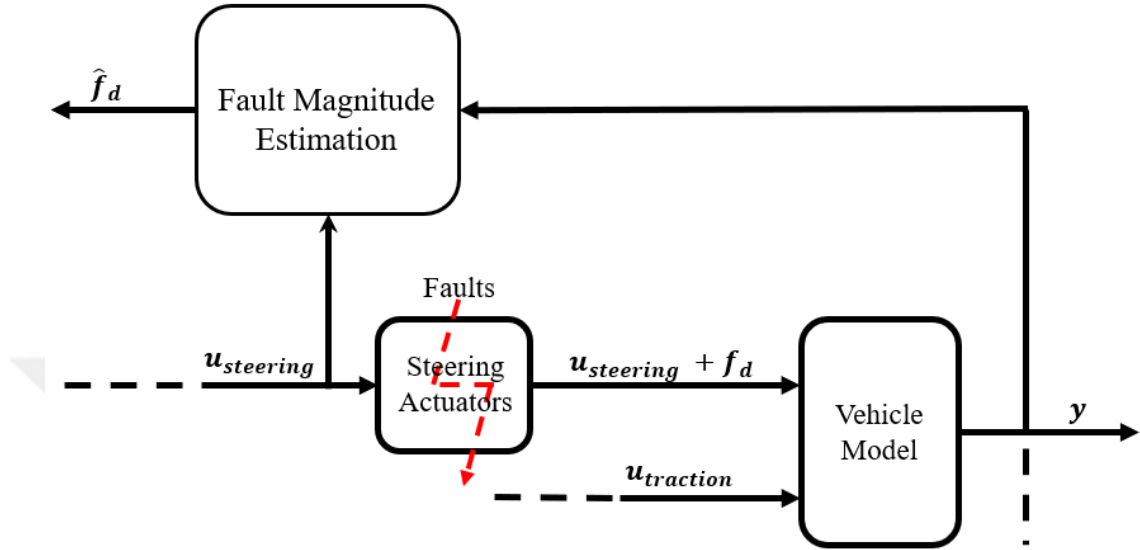


Figure 4.2: Fault Detection, Isolation, and Magnitude Estimation Scheme

In automotive-related fault-initiating events (such as rough road conditions, minor collisions, etc.), usually multiple actuators are affected. In many cases, the deterioration of an actuator can be detected by following the electrical signal information, such as the voltage and the current. Model-based fault magnitude estimation under multiple actuator failure situations can be convoluted compared to the single fault case. Since the effect of various failures on vehicle outputs can be similar, estimating multiple failure magnitudes can be an under-determined problem, and the single fault FDIME from (4.20) must be modified. In multi-failure case, faulty components of the input matrix, i.e., $\mathbf{F}_{d,i}$ column vector representing each wheel, must be linearly independent for proper separation of the total contribution of faults, i.e., $f_{d,i}$ for $i \in \{1, 2, 3, 4\}$, to the system. However, in the vehicle state-space model, left and right half steering gain elements in the control input matrix $\mathbf{B}_c$ are the same, i.e., $\mathbf{B}_c(:, 1) = \mathbf{B}_c(:, 2)$ and $\mathbf{B}_c(:, 3) = \mathbf{B}_c(:, 4)$. This means 1st and 2nd columns, and 3rd and 4th columns are linearly dependent. In order to overcome the problem, the state-space model proposed in (2.13)-(2.15) is modified as follows:

$$\dot{\mathbf{x}}_f = \mathbf{A}_f \mathbf{x}_f + \mathbf{B}_f \mathbf{u}_f \quad (4.21)$$

$$\mathbf{x}_f^T = \begin{bmatrix} v_y & r & x_1 & x_2 & x_3 & x_4 \end{bmatrix} \quad (4.22)$$

$$\mathbf{u}_f^T = \begin{bmatrix} \delta_1 & \delta_2 & \delta_3 & \delta_4 \end{bmatrix} \quad (4.23)$$

$$\mathbf{A}_f = \begin{bmatrix} \mathbf{A}'_c & \mathbf{0}_{2 \times 4} \\ \mathbf{0}_{4 \times 2} & \mathbf{0}_{4 \times 4} \end{bmatrix} \quad (4.24)$$

$$\mathbf{B}_f = \begin{bmatrix} \mathbf{B}_{c,2} \\ \mathbf{Z} \end{bmatrix} \quad (4.25)$$

$\mathbf{C}_f = \mathbf{I}_6$ and $\mathbf{D}_f = \mathbf{0}_{6 \times 4}$. In this estimation-oriented state-space model, first, traction force inputs are removed for simplicity since we only want to determine steering faults in the system. Second, four independent virtual states $x_i = z_i \delta_i$ for $i \in \{1, 2, 3, 4\}$ representing the expected current required to steer the wheels to required commands are added by introducing $\mathbf{Z} = \text{diag}(z_1, z_2, z_3, z_4)$ diagonal matrix into our control input matrix (Equation (4.25)) to make columns of the control input matrix, $\mathbf{B}_f$ linearly independent. These physical signals, related to the power supply of electromechanical systems in electric vehicles, are generally monitored by the controller hardware to increase safety and to check the health status of the system. Here z_i is a gain parameter to make the correlation between the actual wheel angle and the electrical current signal. Thus, the actual steering angles of the vehicle can be obtained via a model-based method without the need to add wheel angle sensors to each wheel.

The steering fault magnitude vector can be given as:

$$\mathbf{f}_d = \begin{bmatrix} f_{d,1} & f_{d,2} & f_{d,3} & f_{d,4} \end{bmatrix} \quad (4.26)$$

Once the faulty actuators are isolated, i.e., $\mathbf{F}_{d,i}$ column vectors are determined, from the mismatched correlation in electrical signals, the discrete-time state-space model with additional fault inputs can be written as:

$$\begin{aligned}\mathbf{x}_f(k) &= \mathbf{A}_f \mathbf{x}_f(k-1) + \mathbf{B}_f \mathbf{u}_f(k-1) \\ &\quad + \sum_{j=1}^4 \mathbf{F}_{d,j} f_{d,j}(k-1)\end{aligned}\tag{4.27}$$

Then the equation for fault magnitude estimation at i^{th} wheel can be obtained with the following formula derived with transposition in Equation 4.27:

$$\begin{aligned}\hat{f}_{d,i}(k-1) &= \mathbf{F}_{d,i}^{-1} [\hat{\mathbf{x}}_f(k) - \mathbf{A}_f \hat{\mathbf{x}}_f(k-1) \\ &\quad - \mathbf{B}_f \mathbf{u}_f(k-1) - \sum_{\substack{j=1 \\ j \neq i}}^4 \mathbf{F}_{d,j} f_{d,j}(k-1)]\end{aligned}\tag{4.28}$$

In order to separate magnitude estimations, (4.28) is calculated for each wheel, i.e., different i values from 1 to 4 simultaneously. Since $\mathbf{F}_{d,i}$ vectors are linearly independent, estimations will eventually converge to actual values.

Stability and Existence

The stability and existence of the multiple-failure estimation scheme given in (4.20)-(4.28) can be studied in three steps outlined below:

1. Since the overall system is designed to be stable, the states of the system will be finite a value for (4.27) exists for all k .
2. The pair $(\mathbf{A}_f, \mathbf{C}_f)$ is observable since observability matrix $\mathcal{O} = \begin{bmatrix} \mathbf{C}_f & \mathbf{C}_f \mathbf{A}_f & \mathbf{C}_f \mathbf{A}_f^2 & \dots & \mathbf{C}_f \mathbf{A}_f^5 \end{bmatrix}^T$ is full column rank. This ensures that (4.28) can be calculated for each step k .
3. Equation (4.28) contains the solution of matrix-vector equation $\mathbf{A}x = \mathbf{b}$ where $\mathbf{A} = \mathbf{F}_{d,i}$, $x = f_{d,i}$ and $\mathbf{b} = \hat{\mathbf{x}}_f(k) - \mathbf{A}_f \hat{\mathbf{x}}_f(k-1) - \mathbf{B}_f \mathbf{u}_f(k-1) - \sum_{\substack{j=1 \\ j \neq i}}^4 \mathbf{F}_{d,j} f_{d,j}(k-1)$. Since $\mathbf{A} = \mathbf{F}_{d,i}$ coefficient matrix has empty null space for each i value (from 1 to 4), (i.e. invertible) the equation has unique solution $x = f_{d,i}$ on each time instant and valid for each wheel.

4.1.3 Fault Tolerant Reconfigurable MPC

In this section, online reconfiguration of the MPC formulation from Section 4.1.1 under two different failure situations (actuator loss of effectiveness and actuator jam) is proposed to improve the control performance.

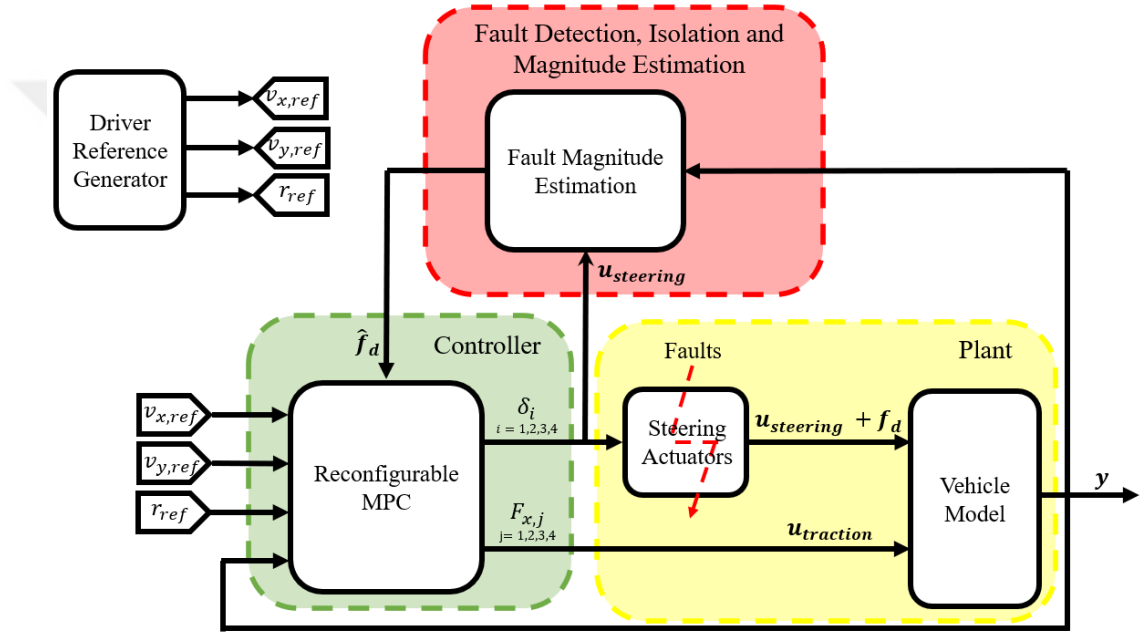


Figure 4.3: Fault Tolerant Control Scheme

Fig. 4.3 represents the overall control scheme of the system. It combines the information given in Fig. 4.1 and Fig. 4.2. Reconfigurable MPC block receives estimated fault magnitude $\hat{f}_{d,i}$ information from Fault Detection, Isolation and Magnitude Estimation block (formulated and explained in Section 4.1.2). Then, the algorithm updates its control input matrix, $\mathbf{B}_c$, or adds constraints to the optimization problem depending on the type of actuator failure.

4.1.3.1 Actuator Loss of Effectiveness

In this failure case, we need to modify the control input matrix $\mathbf{B}_c$ on each loop before plant discretization in (4.2). In order to be able to reconfigure the MPC based on which the actuator(s) has loss of effectiveness, a control effectiveness

matrix $\mathbf{B}_e$ is defined, and $\mathbf{B}_c$ is decomposed as $\mathbf{B}_c = \mathbf{B}_n \mathbf{B}_e$, where $\mathbf{B}_n = \mathbf{B}_c$ is the nominal control input matrix and $\mathbf{B}_e = \text{diag}(g_1, g_2, g_3, g_4, 1, 1, 1, 1)$, is the diagonal matrix that shows the effectiveness of each actuator. Only the first four parameters are set to variable coefficients as they represent the steering section of the input matrix. Coefficients g_i ($i \in \{1, 2, 3, 4\}$) get nominal value 1 if there is no steering actuator failure and get $0 < g_i < 1$ under failure situation. If $g_i = 0$ then the failure case is not classified as loss of effectiveness, instead it becomes actuator jam at zero steering angle.

Effectiveness gain parameter g_i is determined with the following formula:

$$g_i = \frac{\delta_i - \hat{f}_{d,i}}{\delta_i} \quad (4.29)$$

Here δ_i is the steering input applied by the MPC at fault estimation time instant without the knowledge of the failure. After determining the effectiveness gain parameter g_i , the matrix $\mathbf{B}_c$ will be updated. Then, the MPC optimization problem readjusts the control inputs with the knowledge of the failure condition. On each control loop, the MPC problem is formulated and solved starting from plant discretization with a new $\mathbf{B}_c$ matrix.

In this reconfiguration procedure, elements of $\mathbf{B}_c$ matrix are multiplied with some bounded positive gain g_i , so, $(\mathbf{A}_c, \mathbf{B}_c)$ pair remains controllable since row rank of the controllability matrix $\mathcal{C} = [\mathbf{B}_c \quad \mathbf{A}_c \mathbf{B}_c \quad \mathbf{A}_c^2 \mathbf{B}_c \quad \dots \quad \mathbf{A}_c^5 \mathbf{B}_c]$ unchanged. Additionally, after the reconfiguration, the Hessian matrix remains positive and definite, so stability and feasibility conditions in Section 4.1.1 are unaffected.

4.1.3.2 Actuator Jam

In this case, to represent the actuator jam condition at i^{th} wheel to MPC formulation, we introduce:

$$u_i(k) = \hat{f}_{d,i} \quad (4.30)$$

Equation (4.30) implies that the MPC problem must be reconfigured such that the previous steering input applied to the wheel i , i.e. $u_i(k-1)$, will no longer be valid through MPC optimization until Fault Detection & Magnitude Estimation block reports otherwise. Instead constant value $\hat{f}_{d,i}$ will be applied to the system. Hence, $\Delta \mathbf{u}$ at i^{th} actuator is set to 0 within the control horizon by introducing the following equality constraint to the MPC. Therefore an equality constraint can be added to the problem formulation given in (4.18) as follows

$$\mathbf{A}_{eq}\Delta \mathbf{U} = \mathbf{b}_{eq} \quad (4.31)$$

where $\mathbf{A}_{eq} \in \mathbb{R}^{m \times mN_c}$ with value one for the elements representing faulty actuator horizon values and zero otherwise, and the zero matrix $\mathbf{b}_{eq} \in \mathbb{R}^{N_c \times 1}$ is formed for the right side of the equality to make $\mathbf{A}_{eq}\Delta \mathbf{U} = \Delta u_i = 0$.

The resulting equality constraint in (4.31) ensures that the MPC algorithm does not change the value of i^{th} actuator during finding optimal control inputs, which makes it fault-informed. Again, on each control loop, the MPC problem is formulated and solved starting from plant discretization with new constraints. In the case of multiple jam situations, the same equality constraints with different i values are added. When there is no failure in the system, both $\mathbf{A}_{eq}$ and $\mathbf{b}_{eq}$ are zero matrices which nullify the effect of the constraint on the overall MPC problem.

Stability, Convergence, and Feasibility

Stability analysis of the model predictive controller mentioned in Sections 4.1.1 and 4.1.2 stay concurrently valid after the reconfiguration of the constraints since the MPC optimization problem Hessian matrix in (4.18) does not change. Additionally, the problem remains feasible after adding the fault information constraint. Since the addition of the equality constraint given in (4.31) to the existing MPC QP optimization routine will further limit the merit function to become large, the updated problem remains inside feasible region boundaries.

Parameter	Symbol	Value	Unit
front wheel lateral cornering stiffness	C_{af}	3.81	N/rad
rear wheel lateral cornering stiffness	C_{ar}	4.14	N/rad
mass	m	2.28	kg
yaw moment of inertia	I_z	0.11	kgm^2
distance from front axle to CoG	a	0.164	m
distance from rear axle to CoG	b	0.17	m
width of wheelbase	w	0.2	m

Table 4.1: Test Vehicle Parameters

4.2 Validation of the Controller

In this section, the performance of the proposed fault-tolerant reconfigurable MPC method is analyzed using three different steering actuator failure scenarios against the optimal MPC performance of the vehicle with no fault and the performance of the MPC in the presence of failures without reconfiguration. The simulations are first done using a desktop computer (Section 4.2.1) and then using our Processor-In-the-Loop setup (Section 4.2.2), which includes the real-time controller hardware and experimentally validated high-fidelity vehicle model, which was presented in [17, 18], and [43].

4.2.1 Simulation Results in the Non-Linear Model

In the first section, the nonlinear vehicle model presented in Chapter 2 is used as the mathematical vehicle representation during the tests using the Matlab/Simulink environment. This nonlinear model is also the starting point for our linear model presented in Section 2.2.2. For vehicle model parameters, the values in Table 4.1 are used. These values were experimentally obtained from our scaled test vehicle in Chapter 3.

MPC parameters used throughout the tests are listed in Table 4.2. The actuator magnitude and rate limits are selected to represent physical limits of the test

Parameter	Value	Unit
N_p	10	-
N_c	4	-
T_s	0.05	s
$\Delta \mathbf{u}_{min,steering}$	$-0.1 \cdot \mathbf{P}$	rad
$\Delta \mathbf{u}_{max,steering}$	$0.1 \cdot \mathbf{P}$	rad
$\mathbf{u}_{min,steering}$	$-0.5236 \cdot \mathbf{P}$	rad
$\mathbf{u}_{max,steering}$	$0.5236 \cdot \mathbf{P}$	rad
$\Delta \mathbf{u}_{min,traction}$	$-0.05 \cdot \mathbf{Q}$	N
$\Delta \mathbf{u}_{max,traction}$	$0.05 \cdot \mathbf{Q}$	N
$\mathbf{u}_{min,traction}$	$-0.5 \cdot \mathbf{Q}$	N
$\mathbf{u}_{max,traction}$	$0.5 \cdot \mathbf{Q}$	N
$\mathbf{W}_i$	$\mathbf{I}_8$	-

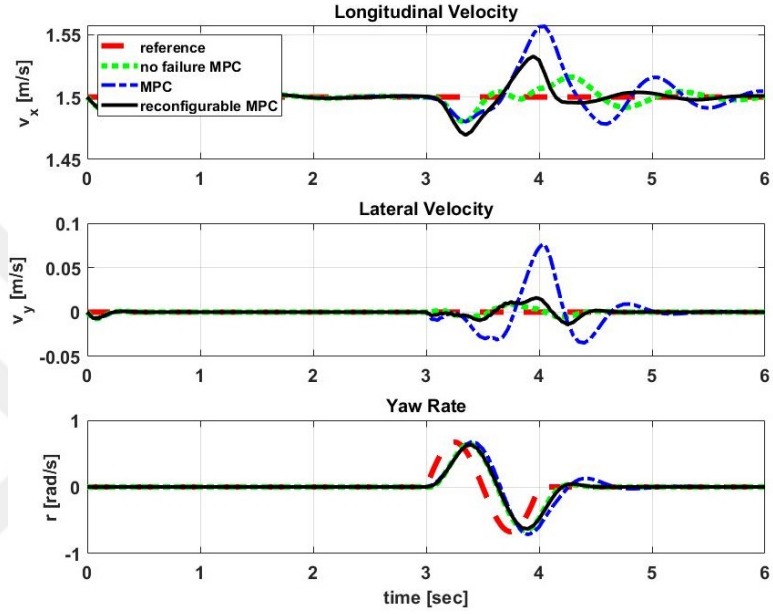
Table 4.2: MPC Properties

vehicle proposed in Chapter 2. Similarly, sample time T_s is selected to maintain the computational load of the controller under the limits of Arduino hardware. In Table 4.2, $\mathbf{P}^T = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$ and $\mathbf{Q}^T = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$. Quadratic Programming Solver chosen for the MPC optimization problem is *mpcActiveSetSolver*, readily available in Matlab Optimization Toolbox.

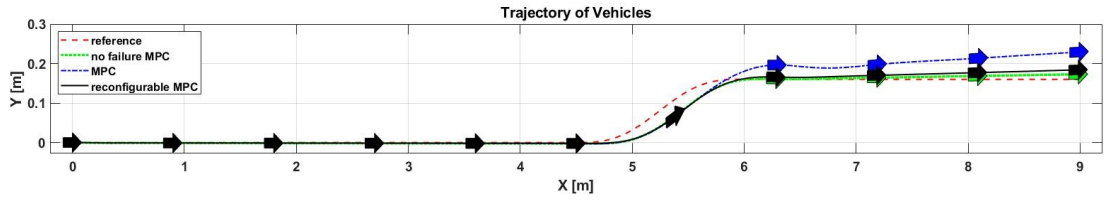
4.2.1.1 Scenario 1 (loss of effectiveness and jam)

In the first scenario, rear-left ($i = 3$) wheel is stucked at $-0.05[rad]$ at $t = 3$ and front-left ($i = 1$) and rear-right ($i = 4$) wheels effectiveness are 80% and 90% throughout the test. In other words, $f_{d,3} = -0.05$, $g_1 = 0.8$ and $g_4 = 0.9$.

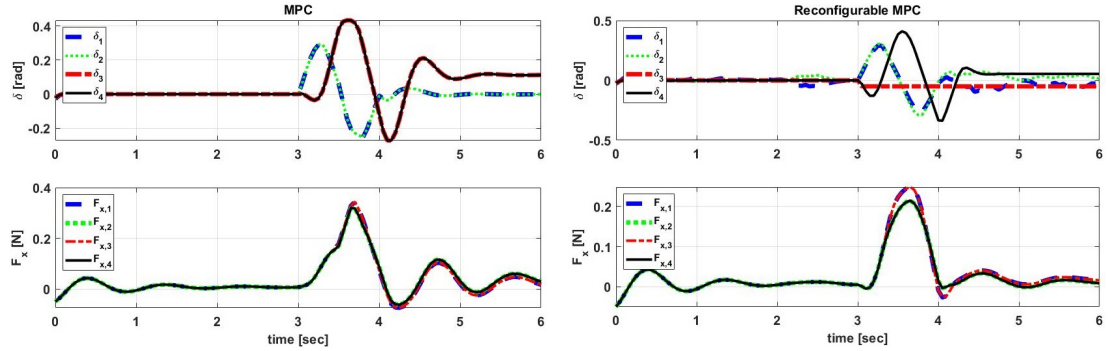
Fig. 4.4a presents the response of three controlled systems exposed to multiple steering actuator failure scenarios. In these plots, the red line represents the reference values of each state generated by the driver's intention. Table 4.3 details the labels of the legends in the figures. The "no failure MPC" trace (dashed green) shows the response of the original model predictive controller when there is no steering failure. The main objective of the proposed RMPC is to drive the states of the system as close as to this nominal response in the presence of



(a) Vehicle States



(b) Vehicle Trajectory



(c) MPC Inputs

(d) Reconfigurable MPC Inputs

Figure 4.4: Scenario 1

Legend	Description
no failure MPC	No actuator failure, and the vehicle controller uses standard MPC formulation given in (4.1)-(4.19)
MPC	Actuator failure described in Scenario 1 or 2 is introduced, and the vehicle controller uses standard MPC formulation given in (4.1)-(4.19)
RMPC	Actuator failure described in Scenario 1 or 2 is introduced, and the vehicle controller uses the reconfigurable MPC formulation given in (4.29)-(4.31).

Table 4.3: Explanation of the Plot Legends for Figs. 4.4-4.10.

steering actuator fault/faults. Blue and black lines show the response of the MPC and the RMPC algorithms experiencing the same failures described in Scenario 1 above. A comparison of the responses shows that the proposed RMPC algorithm significantly improves the reference tracking capability of the system since it can find optimal control inputs considering the faults.

Fig. 4.4c and Fig. 4.4d present the control outputs of the MPC and RMPC method under Scenario 1. Comparing the results in Fig. 4.4c and Fig. 4.4d shows that the fault information is successfully estimated and included in the model predictive control problem as RMPC can adjust individual steering actuators accordingly. Fig. 4.4d shows that the RMPC algorithm fixes the third steering actuator to $-0.05[rad]$ where jamming occurs and re-adjusts the control efforts on the first and fourth steering actuators where the loss of effectiveness happens. Fig. 4.4b shows vehicle trajectory in each scenario in longitudinal and lateral directions. This figure also shows that the reconfigurable MPC method follows the optimal performance (i.e., the no-failure MPC control case) much closer than the regular MPC with failure case.

4.2.1.2 Scenario 2 (loss of effectiveness)

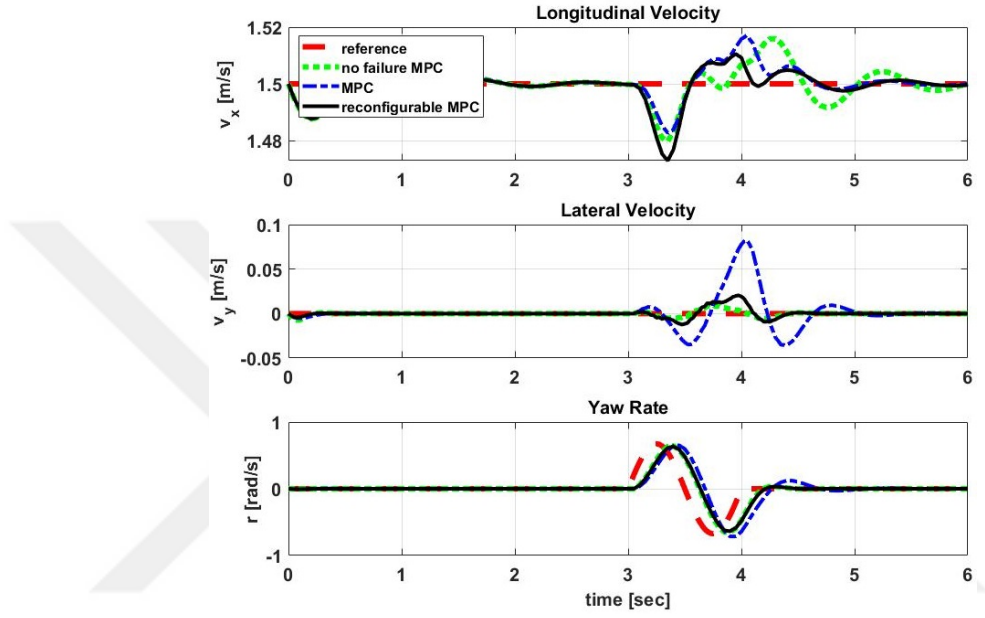
In the second scenario, an asymmetric loss of effectiveness is experienced on all wheels. In front-left ($i = 1$), front-right ($i = 2$), rear-left ($i = 3$) and rear-right ($i = 4$) wheels effectiveness are set to drop to 70%, 80%, 30% and 40% respectively. In other words, $g_1 = 0.7$, $g_2 = 0.8$, $g_3 = 0.3$ and $g_4 = 0.4$.

Fig. 4.5a again presents the response of two controller cases exposed to multiple steering actuator failures for Scenario 2 with the optimal response and the reference response. Legends of the plots are the same as in Scenario 1. A comparison of the responses shows that the proposed RMPC algorithm significantly improves the reference tracking capability of the system and drives the states closer to the optimal no-failure MPC situation.

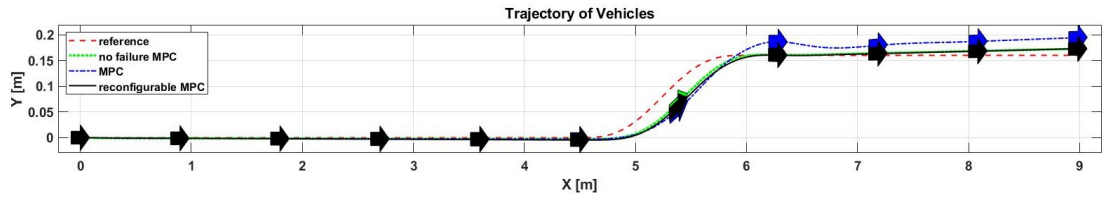
Fig. 4.5c and 4.5d present the control outputs of the MPC and the RMPC methods experiencing the failures described as Scenario 2. Compared to Fig. 4.5c, Fig. 4.5d shows that the fault information is successfully estimated and included in the predictive controller, as RMPC can adjust individual steering actuators accordingly. The RMPC algorithm readjusts the control efforts on each steering actuator according to their magnitude loss of effectiveness. Fig. 4.5b shows vehicle trajectories in each case. This figure also shows the superior performance of the reconfigurable MPC method to keep the performance much closer to no failure MPC control case than the MPC without reconfiguration.

In addition to the input/output plots of the failure scenarios, tables representing the accumulated absolute error of each state are generated to quantify the improvement percentages of the proposed method numerically. Table 4.4 shows the performance of the model predictive controller with actuator constraints given on Table 4.2 under nominal conditions. MPC formulated in (4.1)-(4.19), is run with the vehicle model with no failure, and the total absolute error of each state is calculated.

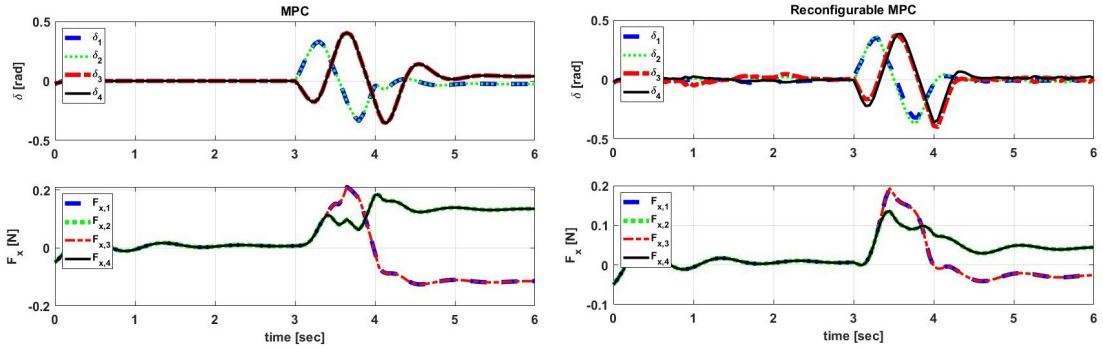
Results from Table 4.4 show that nominal MPC performs considerably well as



(a) Vehicle States



(b) Vehicle Trajectory



(c) MPC Inputs

(d) Reconfigurable MPC Inputs

Figure 4.5: Scenario 2

States	Total Absolute Error
$e_{v_x, total}$	0.0024
$e_{v_y, total}$	0.0066
$e_{r, total}$	0.34

Table 4.4: Total Absolute Error Values of (no failure) MPC

States	Total Absolute Error					
	Scenario 1			Scenario 2		
	MPC	RMPC	Imp.(%)	MPC	RMPC	Imp.(%)
$e_{v_x, total}$	0.036	0.025	30.56	0.0124	0.008	35.48
$e_{v_y, total}$	0.042	0.006	85.71	0.044	0.0066	85
$e_{r, total}$	0.127	0.0226	82.2	0.175	0.024	86.29

Table 4.5: Total Absolute Error Values (Simulations)

a controller since the state error values are insignificant. Therefore, it is reasonable to evaluate the performance of MPC under failure and RMPC under failure scenarios by comparing their state outputs with nominal MPC state outputs because nominal MPC has the ultimate performance that the proposed method can achieve.

Table 4.5 presents total absolute error values comparison of MPC and RMPC methods for both failure scenarios in simulations. Rows under the MPC title show total absolute error values of each state in the presence of specified fault scenario while the MPC algorithm formulated in (4.1)-(4.19) is used as controller. Similarly, rows under the RMPC title show total absolute error values of each state in the presence of specified fault scenario while the proposed RMPC algorithm formulated in (4.29)-(4.31) is used as controller. The error values are calculated by taking the nominal MPC response mentioned in Table 4.4 as a reference since it provides the ultimate performance level for this controller. The improvement column in Table 4.5 shows the percentage of error reduction contributed by RMPC. The proposed method significantly reduces the total absolute error values in both scenarios.

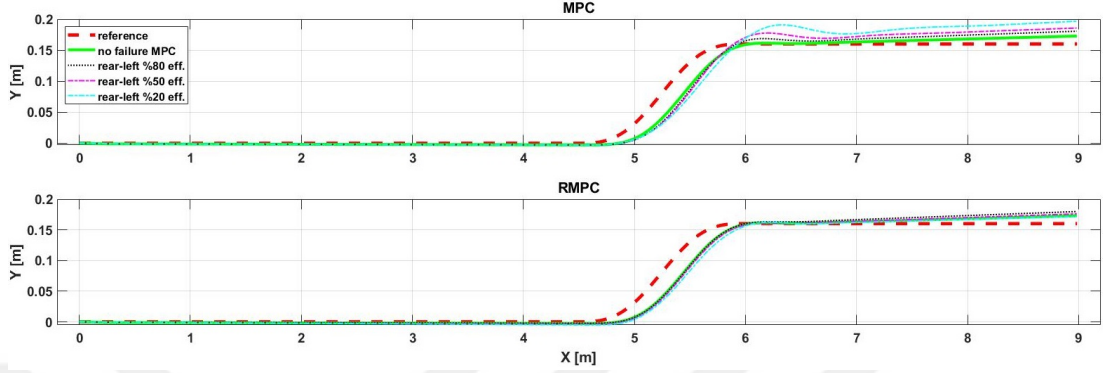


Figure 4.6: Scenario 3 Vehicle Trajectory

4.2.1.3 Scenario 3 (robustness to deteriorating failure)

In this scenario, the robustness of the proposed method against actuator loss of effectiveness is evaluated. It is possible to think that if the vehicle performance stays robust to deteriorating actuator failure, the regular service time can be extended. Front-left ($i = 1$), front-right ($i = 2$) and rear-right ($i = 4$) wheels effectiveness are set to drop to 70%, 80% and 40% respectively. In this scenario, rear-left ($i = 3$) effectiveness gradually decreases from 80% to 20%. Figure 4.6 demonstrates the responses of MPC (top) and RMPC (bottom) against the gradual drop, respectively. MPC algorithm drifts off course as the effectiveness percentage decreases. On the other hand, the RMPC algorithm shows similar tracking performance even in the presence of increasing loss of effectiveness. Therefore, this scenario implies an overall increase in vehicle robustness with the newly proposed RMPC method.

4.2.2 Simulations with the High Fidelity Vehicle Model with PIL capabilities

In [17], and [18], a high-fidelity vehicle model was proposed, formulated in detail, and validated for independent four-wheel drive and four-wheel steering vehicles with independent active suspensions. This model was then utilized in [43] for

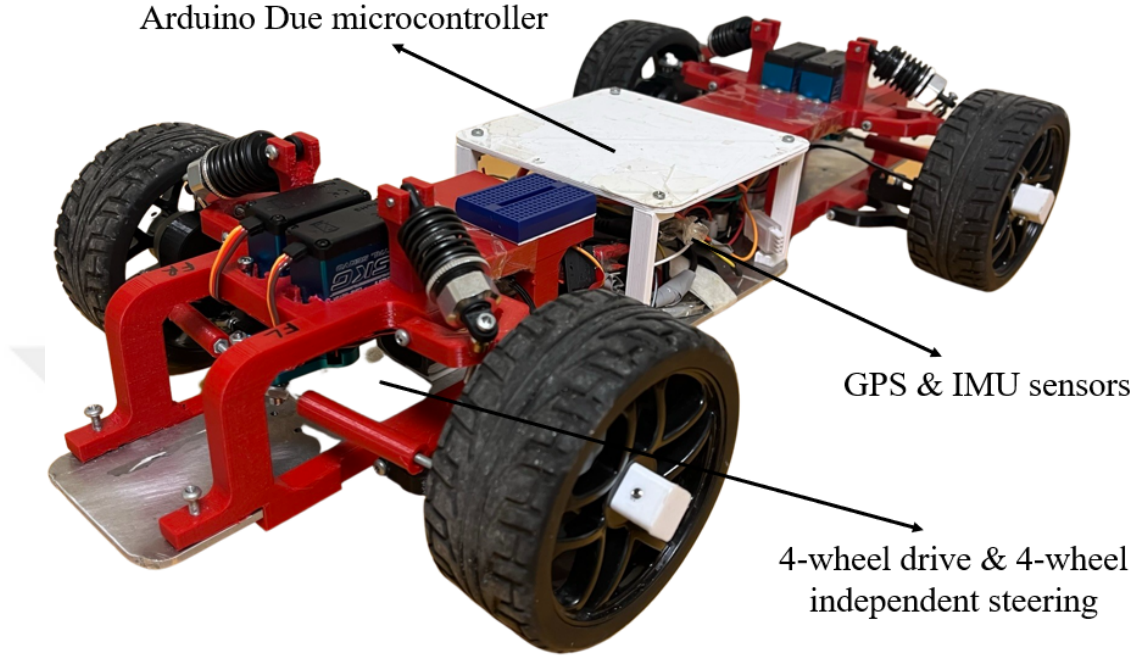
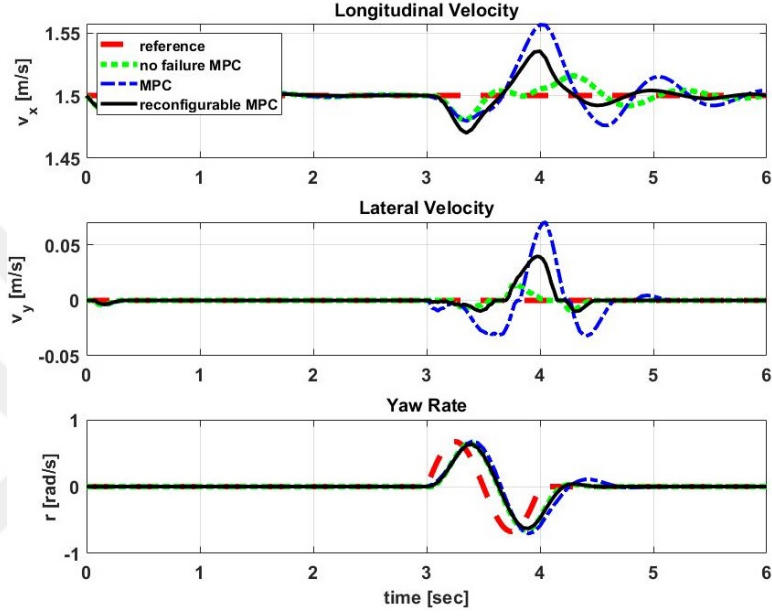


Figure 4.7: The independent 4WD and independent 4WS Scaled Test Vehicle [43]

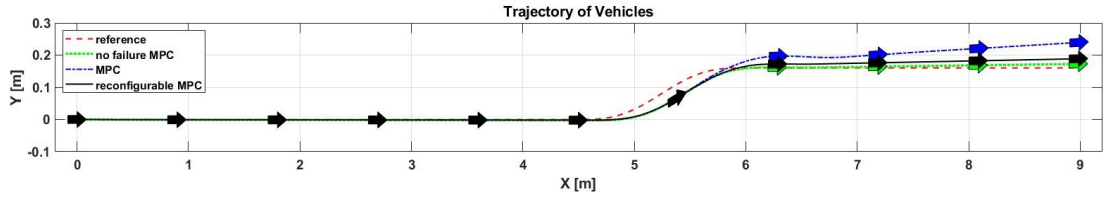
the parameter identification of a scaled test vehicle (shown in Fig. 4.7) for controller development. This vehicle and the high-fidelity model can also be used as part of a Processor-In-the-Loop (PIL) testing setup when detailed and realistic implementation-ready data is needed.

During the PIL testing, the validated model emulates our actual scaled test vehicle (from Chapter 2) presented in Fig. 4.7 and has online controller implementation capability. The test vehicle uses an Arduino Due microcontroller running the controller block with $T_s = 0.05[s]$ sampling time. Table 4.1 presents the parameters of the test vehicle that are used to build the high-fidelity model. The front and rear wheel cornering coefficients (C_{af} & C_{ar}) of the test vehicle identified in this table are obtained through the parameter identification method mentioned in [43]. Overall, in both simulation and hardware tests MPC controller parameter settings are adjusted according to Table 4.2.

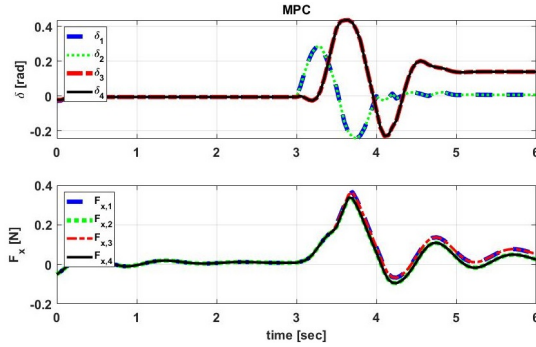
In these PIL simulations, for a fair comparison of the outputs between the simulation and implementation, failure conditions described as Scenario 1, Scenario 2, and Scenario 3 from Section 4.2.1 are applied.



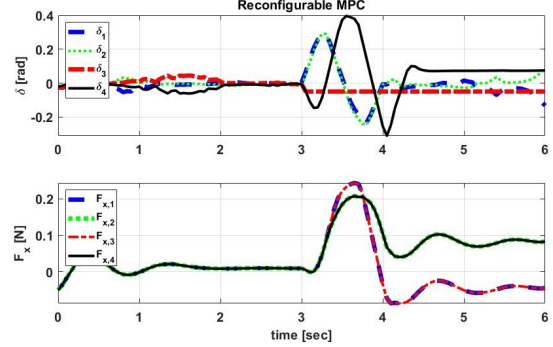
(a) Vehicle States



(b) Vehicle Trajectory



(c) MPC Inputs



(d) Reconfigurable MPC Inputs

Figure 4.8: PIL Test Scenario 1

The results shown in Fig. 4.8a support the Scenario 1 simulation results that the proposed reconfigurable MPC (RMPC) method significantly reduces the effect of the steering wheel failure. Fig. 4.8b shows vehicle trajectory on the PIL test. The Reconfigurable MPC reduces the error margin of the trajectory compared to the MPC with no configuration and fault estimation.

Similarly, Fig. 4.9a verifies the Scenario 2 simulation results that the proposed reconfigurable MPC (RMPC) method significantly reduces the effect of the steering wheel failure. Fig 4.9b shows vehicle trajectory on the PIL test. Reconfigurable MPC reduces the error margin of the trajectory compared to the regular MPC.

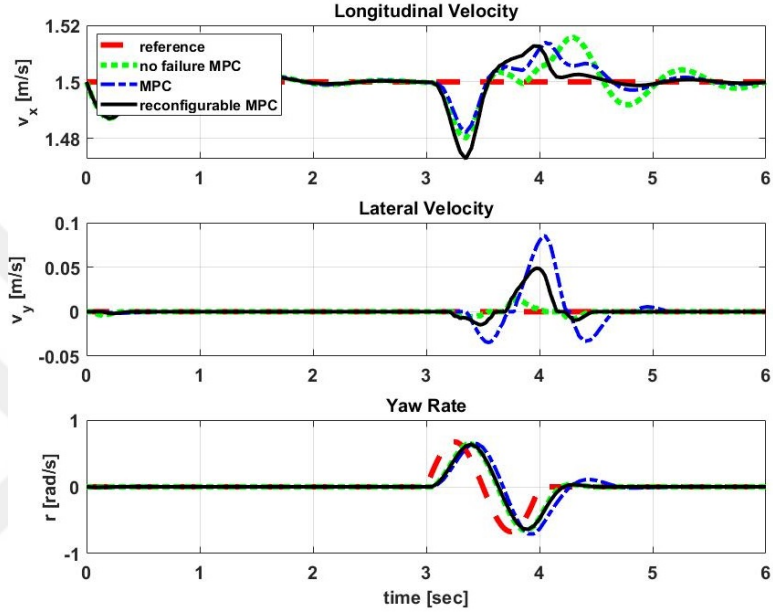
Fig. 4.10 shows PIL high-fidelity model outputs of robustness testing, i.e., Scenario 3. Supporting the simulation outputs presented in Fig. 4.6, the proposed RMPC algorithm shows less deterioration against a gradual drop of loss of effectiveness of the real-left ($i = 3$) wheel.

Additionally, the PIL test shows that the proposed RMPC method can be implemented on online hardware, such as on a low-cost professional-grade microcontroller. Fig. 4.11 demonstrates stable subroutine execution duration at each time step in the processor run test. The average controller execution time of Arduino Due microcontroller is 0.043[s], which is less than the controller step-size, 0.05[s]. This means the processor has sufficient time margin to compute optimal solutions on each step with given MPC settings.

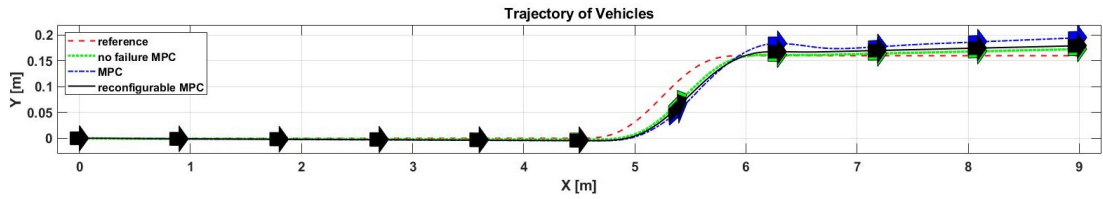
States	Total Absolute Error					
	Scenario 1			Scenario 2		
	MPC	RMPC	Imp.(%)	MPC	RMPC	Imp.(%)
$e_{v_x, total}$	0.033	0.024	27.27	0.013	0.0085	34.62
$e_{v_y, total}$	0.036	0.01	72.22	0.038	0.027	28.95
$e_{r, total}$	0.117	0.021	82.05	0.166	0.036	78.31

Table 4.6: Total Absolute Error Values (PIL Simulations)

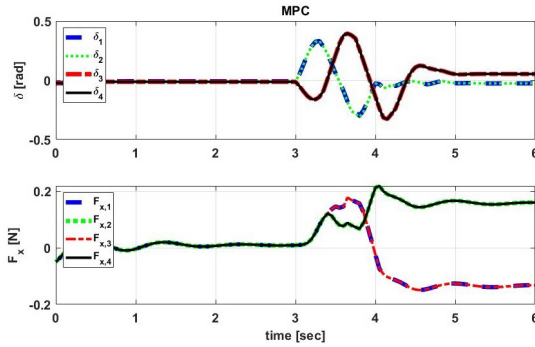
Table 4.6 shows the total absolute error calculations of each PIL test scenario.



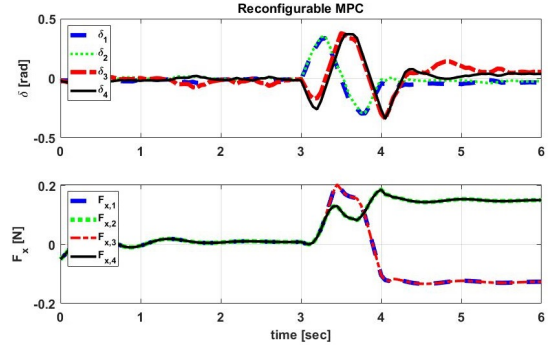
(a) Vehicle States



(b) Vehicle Trajectory



(c) MPC Inputs



(d) Reconfigurable MPC Inputs

Figure 4.9: PIL Test Scenario 2

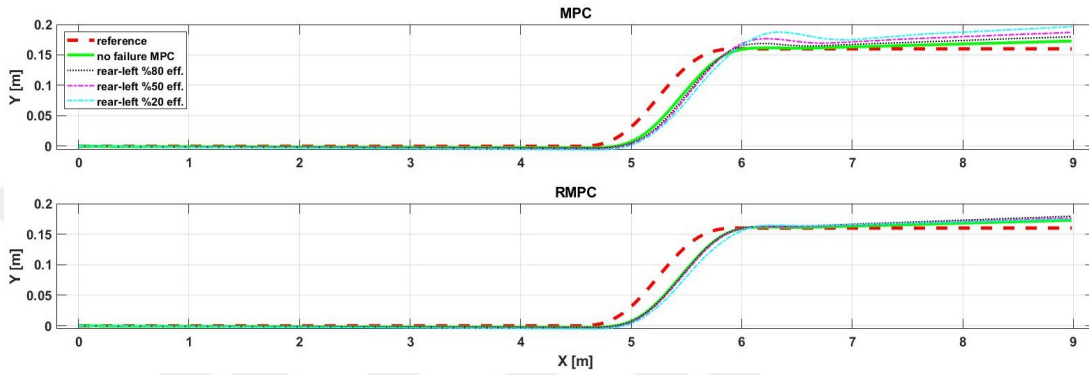


Figure 4.10: PIL Scenario 3 Vehicle Trajectory



Figure 4.11: PIL Test Controller Block Execution Time

Numerical results in this table also prove the contribution of the reconfigurable MPC on output error reduction under the steering actuator failure case. Compared to the results from Table 4.5, the improvement percentages are reduced. However, this is expected since the PIL test includes a much more realistic setup with a high-fidelity vehicle model with an online controller.



Chapter 5

Computationally Efficient Nonlinear Control Allocation

In this chapter, a computationally efficient two-step nonlinear optimal control allocation method is proposed to improve the yaw stability of four-wheel-independent-steering, four-wheel drive vehicles. The virtual controller output is allocated using an optimization problem to compute the steering and traction commands to each wheel at every controller time step. The optimization problem is solved by running a sequential quadratic programming (SQP) procedure, which may consume some time to obtain satisfactory results. The proposed two-path control structure is derived from a more complex single-path allocation problem where torque allocation and steering correction optimal solutions are calculated concurrently. Two-path optimization is obtained by dividing this large nonlinear problem into two smaller problems: a nonlinear problem for steering correction and one linear problem for torque allocation. In this separated two-path control structure, computational load due to the complexity of the single block problem is reduced. In real applications, each problem can be run in parallel on different controllers of the vehicle controller network, which decreases the execution time with near-optimal results. The shorter execution time is critical for real-time systems with lower execution rates. If the single-path control system cannot successfully find the optimal solution at its allocated time, it generates inferior

results in reference trajectory tracking than the two-path control system. The performance and speed comparisons of both approaches are studied using detailed vehicle simulations.

Rest of this chapter is structured as follows: Section 5.1 presents the vehicle model notations used in this chapter. In Section 5.2, formulation of the single-path and two-path optimal control problems are explained. Section 5.3 presents the simulation results and evaluates the reference tracking and computational cost performances of both control methods. Lastly, Section 5.4 future works regarding the proposed method is discussed.

5.1 Nomenclature

Before moving to the development of the control allocation algorithm, the nomenclature of two-track nonlinear vehicle model specifically adapted for this chapter, having slightly different notations than the one used in Chapter 2 which makes reasonable to follow the generation of the optimal control problems, is presented on table 5.1.

5.2 Control Structure

Here, two optimal control allocation approaches derived using the nonlinear two-track vehicle model from Chapter 2 are presented: the single-path SQP and the two-path SQP approaches. In both methods, assumed reference input from the driver manipulates the vehicle's motion by providing front wheel steering, $\delta_{1,in}$, and traction force, F_{in} , inputs. Figure 5.1 presents a typical layout for the control allocation. In Virtual Control Input Generator block, required traction force, F_c , moment, M_c , and lateral force, F_{yc} , are generated by comparing current and reference values from the driver input. In the single-path SQP method, the whole control allocation is done using a single non-linear optimization problem

Notation	Description
F_{xij}	Net longitudinal force at ij tire, $i \in [f, r]$, $j \in [l, r]$
F_{yij}	Net lateral force at ij tire, $i \in [f, r]$, $j \in [l, r]$
F_{zij}	Net normal force at ij tire, $i \in [f, r]$, $j \in [l, r]$
$F_{x,i}$	Traction force on tire i , $i \in [1, 2, 3, 4]$
$F_{y,i}$	Lateral force on tire i , $i \in [1, 2, 3, 4]$
$\delta_{f,in}$	Driver front steering angle input
δ_{fl}	Total front left steering angle
δ_{rl}	Rear left steering angle
δ_{fr}	Total front right steering angle
δ_{rr}	Rear right steering angle
$\Delta\delta_{fl}$	Front left steering angle correction
$\Delta\delta_{fr}$	Front right steering angle correction
δ'_{fl}	Corrected front left steering angle
δ'_{fr}	Corrected front right steering angle
M_z	Total moment around z axis
d	Width of wheelbase
l_f	Distance between vehicle CoG and front axle
l_r	Distance between vehicle CoG and rear axle
V	Velocity
m	Vehicle mass
β	Vehicle sideslip angle
r	Yaw rate
J_z	Moment of inertia of vehicle about z axis

Table 5.1: Nomenclature

with eight allocation variables. The two-path SQP method is derived from the single-path SQP. It contains one small non-linear steering optimization problem with four variables and one linear torque allocation optimization problem with the remaining four variables from the single path problem.

5.2.1 Virtual Control Input Generator

Virtual control input block takes driver reference inputs and compares them with current states of the vehicle to provide corrected longitudinal and lateral forces

and yaw moment for the allocator algorithms. To calculate the amount of the yaw moment needed for the steering action, first, the reference yaw rate, r_{ref} , is found with the yaw rate gain formula in Equation 5.1.

$$r_{ref} = \delta_{1,in} \frac{V}{(l_f + l_r) + K_{us} V^2} \quad (5.1)$$

where K_{us} is the understeering coefficient:

$$K_{us} = \frac{ml_r}{(l_f + l_r)C_\alpha N_f} - \frac{ml_f}{(l_f + l_r)C_\alpha N_r} \quad (5.2)$$

Then, the yaw moment correction, M_c , is calculated by finding the difference between the desired and the current angular accelerations using Equation 5.3.

$$M_c = I_z \frac{d}{dt}(r_{ref} - r) \quad (5.3)$$

To determine the correction in traction force, F_c , the difference between driver input, F_{in} , and current total traction force is used as shown in Equation 5.4.

$$F_c = F_{in} - \sum_{i=\{f,r\}} \sum_{j=\{l,r\}} F_{xij} \quad (5.4)$$

The lateral force correction, F_{yc} , is found by using the inertial force calculated using the lateral acceleration gain, $a_{y,ref}$.

$$\begin{aligned} a_{y,ref} &= \delta_{1,in} \frac{V^2}{(l_f + l_r) + K_{us} V^2} \\ F_{y,ref} &= m a_{y,ref} \\ F_{yc} &= F_{y,ref} - \sum_{i=\{f,r\}} \sum_{j=\{l,r\}} F_{yij} \end{aligned} \quad (5.5)$$

Overall, the virtual control input block outputs aim to map the driver commands to the SQP optimization block.

5.2.2 Single-Path SQP Allocation Problem

In this method, a nonlinearly constrained optimization problem is formulated to improve vehicle stability. The problem formulation is derived from vehicle dynamics and performance equations in literature with additional constraints representing the actuator limits to ensure the real-time implementation of the proposed control system. Figure 5.1 presents the single-path SQP control architecture where complete allocation is done on a single optimization on a single execution path.

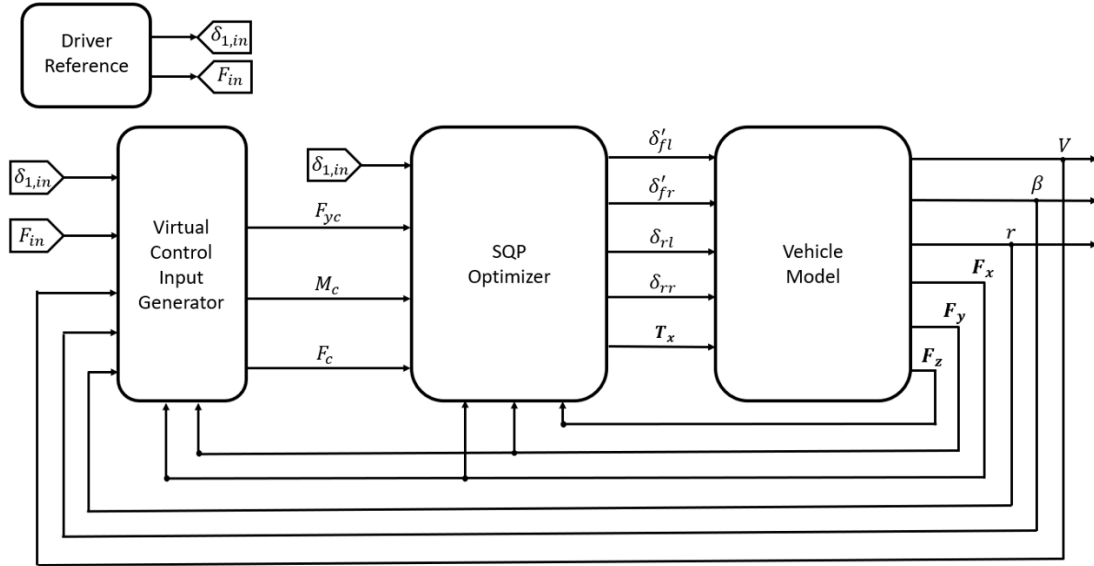


Figure 5.1: Single-Path Control Structure

The optimization problem solved in the SQP optimizer shown in Figure 5.1 can be given as:

$$\mathbf{f} = \operatorname{argmin}\{J\}$$

$$J = \sum_{i=\{f,r\}} \sum_{j=\{l,r\}} \frac{F_{xij}^2 + F_{yij}^2}{F_{zij}^2} \quad (5.6)$$

$\mathbf{f} = [F_{x,1} \ F_{x,2} \ F_{x,3} \ F_{x,4} \ \Delta\delta_1 \ \Delta\delta_2 \ \delta_3 \ \delta_4]$ is the solution array in the optimization problem given in Equation 5.6. The objective function J targets to minimize tire usage during torque allocation and steering correction.

$$\sum_{i=\{1,2,3,4\}} \mathbf{f}(i) - F_c = 0 \quad (5.7)$$

$$\begin{aligned} \frac{d}{2}(F_{xfr} + F_{xrr} - F_{xfl} - F_{xrl}) \\ + l_f(F_{yfl} + F_{yfr}) \\ - l_r(F_{yrl} + F_{yrr}) - M_c = 0 \end{aligned} \quad (5.8)$$

$$\sum_{i=\{f,r\}} \sum_{j=\{l,r\}} F_{yij} - F_{yc} = 0 \quad (5.9)$$

For torque allocation, two equality constraints about required traction force, F_c , and moment, M_c , are used (Equations 5.7 and 5.8). For steering optimization, an equality constraint about lateral force correction term, F_{yc} , is added (Equation 5.9). The expanded form of the terms in 5.8 and 5.9 are listed in 5.10 where the equations are derived from the non-linear vehicle model formulation in Chapter 2.

$$\begin{aligned} F_{xfl} &= \mathbf{f}(1)\cos(\delta_{1,in} + \mathbf{f}(5)) - F_{y,1}\sin(\delta_{1,in} + \mathbf{f}(5)) \\ F_{xfr} &= \mathbf{f}(2)\cos(\delta_{1,in} + \mathbf{f}(6)) - F_{y,2}\sin(\delta_{1,in} + \mathbf{f}(6)) \\ F_{xrl} &= \mathbf{f}(3)\cos(\mathbf{f}(7)) - F_{y,3}\sin(\mathbf{f}(7)) \\ F_{xrr} &= \mathbf{f}(4)\cos(\mathbf{f}(8)) - F_{y,4}\sin(\mathbf{f}(8)) \\ F_{yfl} &= F_{y,1}\cos(\delta_{1,in} + \mathbf{f}(5)) + \mathbf{f}(1)\sin(\delta_{1,in} + \mathbf{f}(5)) \\ F_{yfr} &= F_{y,2}\cos(\delta_{1,in} + \mathbf{f}(6)) + \mathbf{f}(2)\sin(\delta_{1,in} + \mathbf{f}(6)) \\ F_{yrl} &= F_{y,3}\cos(\mathbf{f}(7)) + \mathbf{f}(3)\sin(\mathbf{f}(7)) \\ F_{yrr} &= F_{y,4}\cos(\mathbf{f}(8)) + \mathbf{f}(4)\sin(\mathbf{f}(8)) \end{aligned} \quad (5.10)$$

and

$$\frac{|\mathbf{f}(i) - F_{x,i,prev}|}{\Delta t} \leq a \quad (5.11)$$

for $i = \{1, 2, 3, 4\}$.

$$\begin{aligned} |\mathbf{f}(5) - \mathbf{f}(6)| &\leq b \\ |\mathbf{f}(7) - \mathbf{f}(8)| &\leq c \end{aligned} \quad (5.12)$$

$F_{x,i,prev}$ is the force value at i^{th} tire from the previous step. a, b, c are constants, and Δt is the simulation step-size. The rate of change of traction force on each wheel is constrained to prevent large deviations between each solution step of the simulation with constrain given in Equation 5.11. Thus, achievable outputs are provided to torque actuators. To increase the fidelity of the solutions, front wheels, and rear wheels, which can all steer independently, contain bounding constraints since unrestricted independence may cause undesirable consequences due to kinematic contradictions/faults (Equation 5.12).

$$\underline{\mathbf{f}} \leq \mathbf{f} \leq \bar{\mathbf{f}} \quad (5.13)$$

Lastly, upper and lower bounds are added to steering correction and torque allocation to represent the actuator limits (Equation 5.13). $\delta'_{fl} = \delta_{1,in} + \delta_{fl}$ and $\delta'_{fr} = \delta_{1,in} + \delta_{fr}$ are the updated front steering inputs. $\mathbf{T}_x = \begin{bmatrix} F_{xfl} & F_{xfr} & F_{xrl} & F_{xrr} \end{bmatrix}$ on Figure 5.1. Since the formulation has non-linear components, the SQP method is chosen as a solver that can also be applied in real-time applications.

5.2.3 Two-Path SQP Allocation Problem

In this method, the non-linear optimization problem used in the single-path system (i.e. Equations 5.6-5.13) is divided into two smaller optimization problems:

constrained non-linear steering correction and constrained linear torque allocation problem. There are some examples in literature about dividing the allocation problem into two parts like direct yaw moment control (DYC) and active four-wheel steering (AFWS) as presented in [2]. The problems are divided such that DYC and AFWS has different priorities decided based characteristics. On the other hand, in two path SQP allocation problem, we formulate our sub-problems as the steering and torque allocation to solve a competitive problem with computational simplicity. By separating the original optimization problem, the torque allocation part reduces to a linear problem, making it easier to solve or iterate in the presence of real-time implementation deadlines. Figure 5.2 outlines the two-path SQP control architecture solving the smaller optimization problems in parallel.

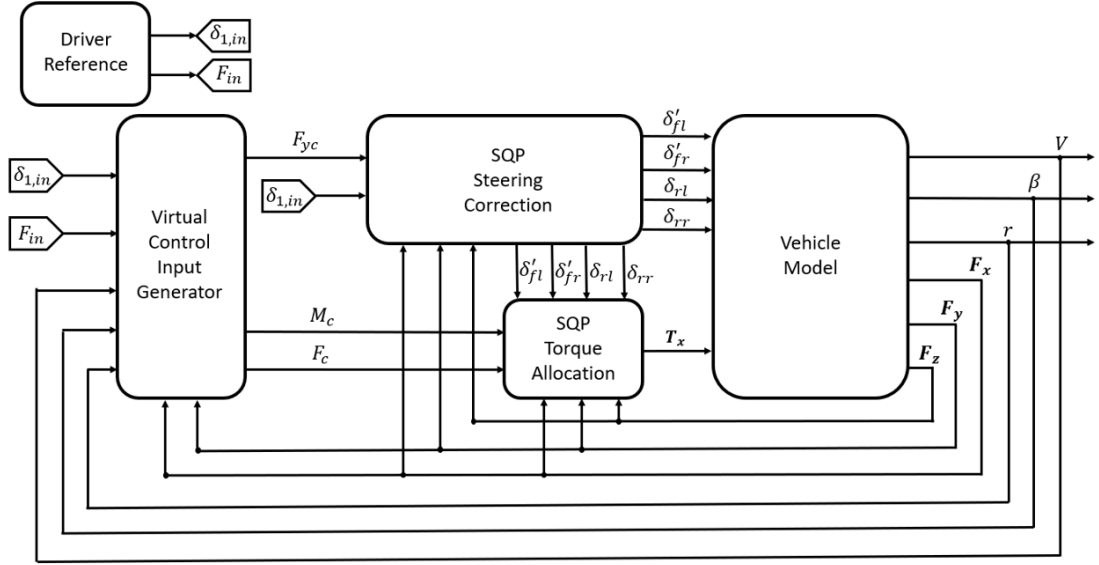


Figure 5.2: Two-Path Control Structure

The steering correction optimization problem to be solved in the first path can be formulated as shown in Equations 5.14-5.18. $\delta = [\Delta\delta_{fl} \ \Delta\delta_{fr} \ \delta_{rl} \ \delta_{rr}]$ is the solution array.

$$\delta = \operatorname{argmin}\{J_1\}$$

$$J_1 = \sum_{i=\{f,r\}} \sum_{j=\{l,r\}} \frac{F_{xij}^2 + F_{yij}^2}{F_{zij}^2} \quad (5.14)$$

such that:

$$\sum_{i=\{f,r\}} \sum_{j=\{l,r\}} F_{yij} - F_{yc} = 0 \quad (5.15)$$

where,

$$\begin{aligned} F_{yfl} &= F_{y,1} \cos(\delta_{1,in} + \delta(1)) + F_{x,1} \sin(\delta_{1,in} + \delta(1)) \\ F_{yfr} &= F_{y,2} \cos(\delta_{1,in} + \delta(2)) + F_{x,2} \sin(\delta_{1,in} + \delta(2)) \\ F_{yrl} &= F_{y,3} \cos(\delta(3)) + F_{x,3} \sin(\delta(3)) \\ F_{yrr} &= F_{y,4} \cos(\delta(4)) + F_{x,4} \sin(\delta(4)) \end{aligned} \quad (5.16)$$

subject to

$$\underline{\delta} \leq \delta \leq \bar{\delta} \quad (5.17)$$

$$\begin{aligned} |\delta(1) - \delta(2)| &\leq b \\ |\delta(3) - \delta(4)| &\leq c \end{aligned} \quad (5.18)$$

$\delta'_{fl} = \delta_{1,in} + \delta_{fl}$ and $\delta'_{fr} = \delta_{1,in} + \delta_{fr}$ are the updated front steering inputs.

In parallel, the torque allocation optimization problem can be formulated as the second path (Equations 5.19-5.24). $\mathbf{F}_x = \begin{bmatrix} F_{xfl} & F_{xfr} & F_{xrl} & F_{xrr} \end{bmatrix}$ is the solution array.

$$\begin{aligned} \mathbf{F}_x &= \operatorname{argmin}\{J_2\} \\ J_2 &= \sum_{i=\{f,r\}} \sum_{j=\{l,r\}} \frac{F_{xij}^2 + F_{yij}^2}{F_{zij}^2} \end{aligned} \quad (5.19)$$

such that:

$$\sum_{i=\{1,2,3,4\}} \mathbf{f}(i) - F_c = 0 \quad (5.20)$$

$$\begin{aligned} & \frac{d}{2}(F_{xfr} + F_{xrr} - F_{xfl} - F_{xrl}) \\ & + l_f(F_{yfl} + F_{yfr}) \\ & - l_r(F_{yrl} + F_{yrr}) - M_c = 0 \end{aligned} \quad (5.21)$$

where,

$$\begin{aligned} F_{xfl} &= \mathbf{F}_x(1)\cos(\delta'_{fl}) - F_{y,1}\sin(\delta'_{fl}) \\ F_{xfr} &= \mathbf{F}_x(2)\cos(\delta'_{fr}) - F_{y,2}\sin(\delta'_{fr}) \\ F_{xrl} &= \mathbf{F}_x(3)\cos(\delta'_{rl}) - F_{y,3}\sin(\delta'_{rl}) \\ F_{xrr} &= \mathbf{F}_x(4)\cos(\delta'_{rr}) - F_{y,4}\sin(\delta'_{rr}) \\ F_{yfl} &= F_{y,1}\cos(\delta'_{fl}) + \mathbf{F}_x(1)\sin(\delta'_{fl}) \\ F_{yfr} &= F_{y,2}\cos(\delta'_{fr}) + \mathbf{F}_x(2)\sin(\delta'_{fr}) \\ F_{yrl} &= F_{y,3}\cos(\delta'_{rl}) + \mathbf{F}_x(3)\sin(\delta'_{rl}) \\ F_{yrr} &= F_{y,4}\cos(\delta'_{rr}) + \mathbf{F}_x(4)\sin(\delta'_{rr}) \end{aligned} \quad (5.22)$$

$$\frac{|\mathbf{F}_x(i) - F_{x,i,prev}|}{\Delta t} \leq a \quad (5.23)$$

$F_{x,i,prev}$ is the force value at i^{th} tire from the previous step, a is constant, and Δt is the simulation step-size.

$$\underline{\mathbf{F}_x} \leq \mathbf{F}_x \leq \overline{\mathbf{F}_x} \quad (5.24)$$

Equation 5.13 is the superset of Equations 5.17 and 5.24. The same constraints and objective functions explained in the single-path problem are used for both steering and torque allocation problems since they will be solved for the same objective.

Hardware	CPU	RAM
	i7 6700k	16GB 2133MHz
Software	Simulation Time	Step size
	10s	0.001s

Table 5.2: Hardware and Software Settings of the Simulations

It is important to note that the union of the formulations for both problems can be given as Equations 5.6-5.13, i.e., the single-path problem allocation problem. However, these two approaches may have different results (two-path being suboptimal) since the solution set of the first problem is assumed as parameters in the second problem, as well as the solution of the second problem is assumed as parameters in the first problem.

5.3 Simulation Results

This section presents Simulink simulation results of proposed allocation approach by evaluating its reference tracking performance and real-life pertinence. The performance of the algorithms is evaluated at various execution speeds: two-path system with only 0.01s execution rate, the single-path system with 0.001s, and 0.01s execution rate separately. Hardware and software settings listed in Table 5.2 are used during these tests. The same constraints and objective functions, on Table 5.3, are used in the simulations for both methods to obtain a fair comparison.

Parameter	Value	Parameter	Value
$\underline{\mathbf{F}}_{\mathbf{x}}$	$-300[N]$	a	$400[N]$
$\overline{\mathbf{F}}_{\mathbf{x}}$	$700[N]$	b	$1.2[deg]$
$\underline{\delta}$	$-2.5[deg]$	c	$1.2[deg]$
$\overline{\delta}$	$2.5[deg]$		

Table 5.3: SQP Optimization Problem Parameters

5.3.1 Obstacle Avoidance Test at $V = 16[m/s]$

In this test, the vehicle is cruising with an initial velocity of $V = 16[m/s]$. The driver applies constant $800[N]$ traction force throughout the test, and sinusoidal steering input is performed in time interval from $t = 3$ to $t = 6s$. Figure 5.3 demonstrates the behavior of the vehicle with single-path and two-path controller allocation structures. The two-path system and single-path with 0.001s execution rate system have better yaw rate reference tracking performance than the single-step system with a 0.01s execution rate. Figures 5.4, 5.5 and 5.6 show the optimal steering correction angles and the traction forces found with single-path with 0.01s execution rate, single-path with 0.001s execution rate, and two-path approaches, respectively. Although the single-path system with a 0.001s execution rate has similar tracking performance with the two-path system, real-time applicability of the two-path system is far better because single-path system violates the actuator rate of change constraint (Equation 5.11) to satisfy equality constraints (Equations 5.7-5.9) for reference tracking. This means, even in 0.001s execution rate, single-path system fails to reach a feasible optimal solution. On the other hand, two-path system satisfies all the constraints while achieving the optimal solution since it has smoother torque and steering outputs (Figure 5.6). In Figure 5.6 baseline steering distribution mentioned in [5] is represented to show the allocation difference between a four-wheel independently steerable system and a restricted conventional system. The two-path system has more freedom in steering allocation, and it facilitates controlling lateral forces, thus improving reference tracking ability.

From Figure 5.7, it can be concluded that the two-path SQP controller and single-path SQP controller with 0.001s execution rate has better reference trajectory tracking performance than the single-path system with a 0.01s execution rate. The main reason for the poorer performance of the single-path controller is the higher computational cost. The system cannot provide optimal solution calculations at an execution rate of 0.01s, which causes deviation from the desired reference.

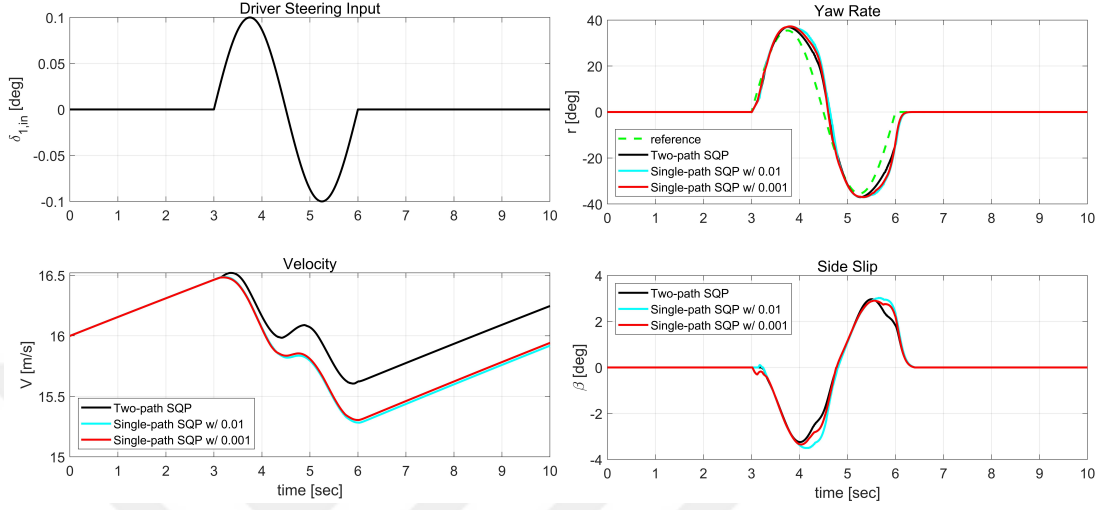


Figure 5.3: Vehicle States

Figure 5.8 and 5.9 verifies the performance comparison of the two systems numerically. Figure 5.8 shows that lateral (y) direction displacement error in single-path at 0.001s execution rate is almost equal to the two-path system, and lower than single-path at 0.01s, which explains the deviation from the reference mentioned in Figure 5.6. On the other hand, Figure 5.9 shows that the integrated value of the objective function over time for the two-path SQP system is slightly less. Although there is more freedom in utilizing variables in the single-path system where steering angles and torque values are calculated concurrently with a single algorithm, the single-path system cannot reach the optimal solutions at these execution rates in real-time. This results in poorer minimization performances of objective function values.

Table 5.4 shows that two-path SQP has a significant advantage in on average execution time per call metric because, in that control system, each SQP block can be solved in different threads simultaneously, reducing the computation time and load significantly. This table is populated by using the model profiler feature in Simulink and averaging the total time spent on the allocation blocks with the number of total calls during the simulation. Results show that, the single-path system can only reach the same tracking performance as the two-path system at slower optimization speed. In conclusion, the two-path system is a more appropriate selection to use in real-time implementation scenarios.

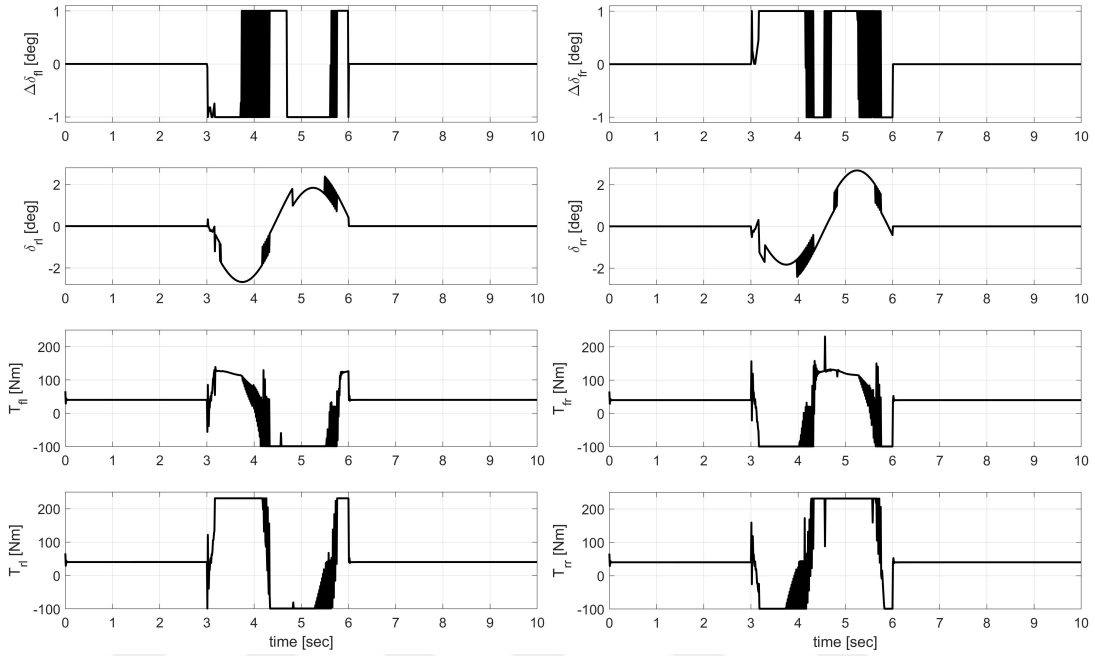


Figure 5.4: Single-Path System Torque and Steering Outputs at 0.01 Execution Rate

	Two-path SQP	Single-path SQP
Execution time[ms]	0.61	0.90

Table 5.4: Average Execution Time

5.4 Discussion and Future Works

In this chapter, a computationally efficient non-linear SQP control allocation procedure is proposed for automotive control systems. The non-linear two-track vehicle model is used to formulate the problems and constraints. Simulation results show that the two-path SQP method, which can run at different controllers in the vehicle controller network, is preferable to the single-path system because it significantly increases the speed of the algorithm execution, has better reference tracking performance under equal constraints, and same step-size condition.

Additionally, the two-path system has smoother outputs for actuators that make it real-time applicable. For future work, performance will be improved by

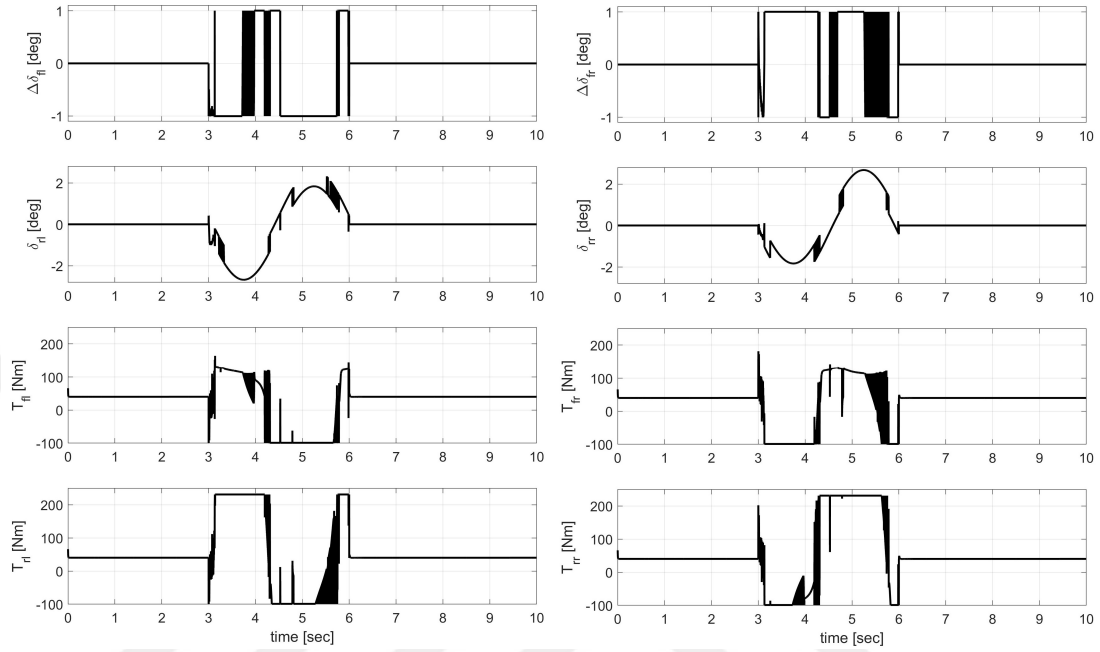


Figure 5.5: Single-Path System Torque and Steering Outputs at 0.001 Execution Rate

increasing the fidelity of the constraints such as actuation saturation and communication delay between controllers. Figure 5.10 represents a summary of our simulation results from previous section, i.e., two-path system is the preferred choice for real-time use case for being both acceptable in performance and computation time in a typical application.

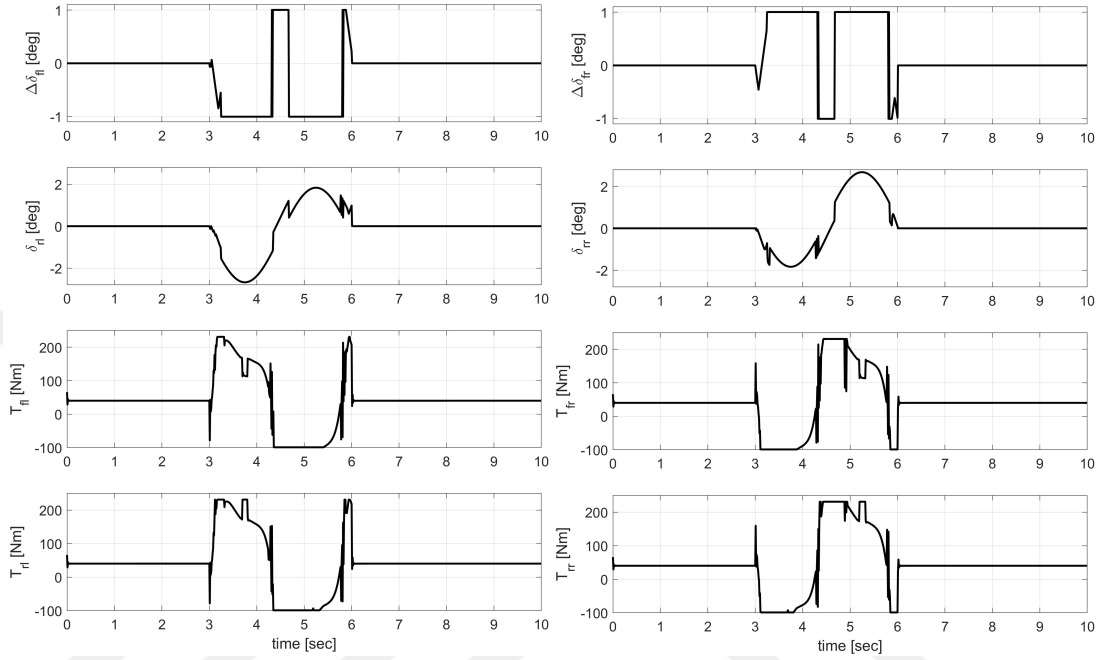


Figure 5.6: Two-Path System Torque and Steering Outputs at 0.01 Execution Rate

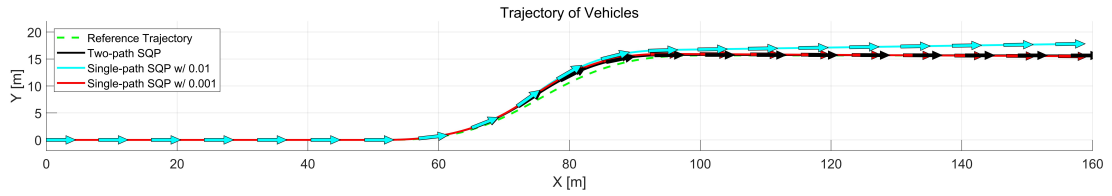


Figure 5.7: Obstacle Avoidance at $V = 16[m/s]$

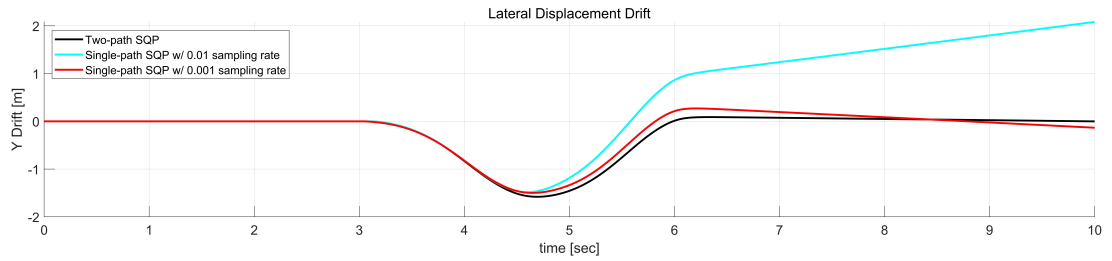


Figure 5.8: Lateral Displacement Drift

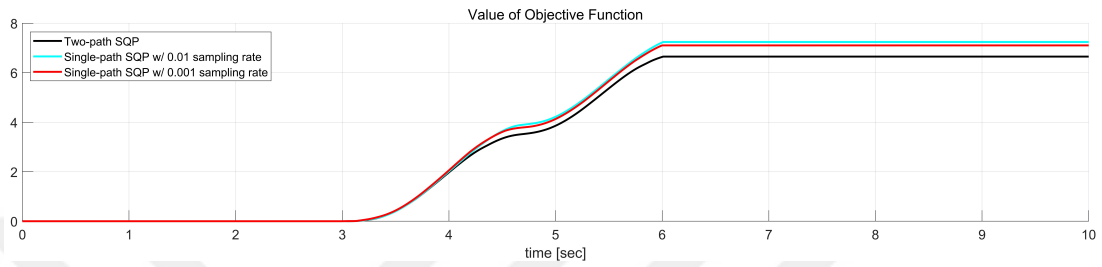


Figure 5.9: Integrated Value of Objective Function

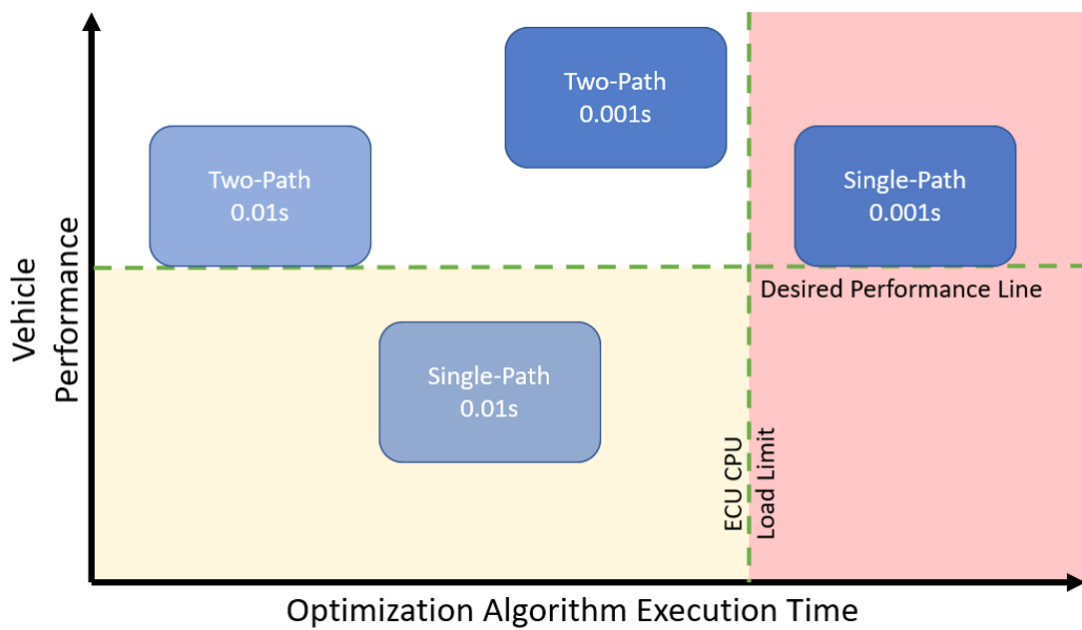


Figure 5.10: Summary of Results

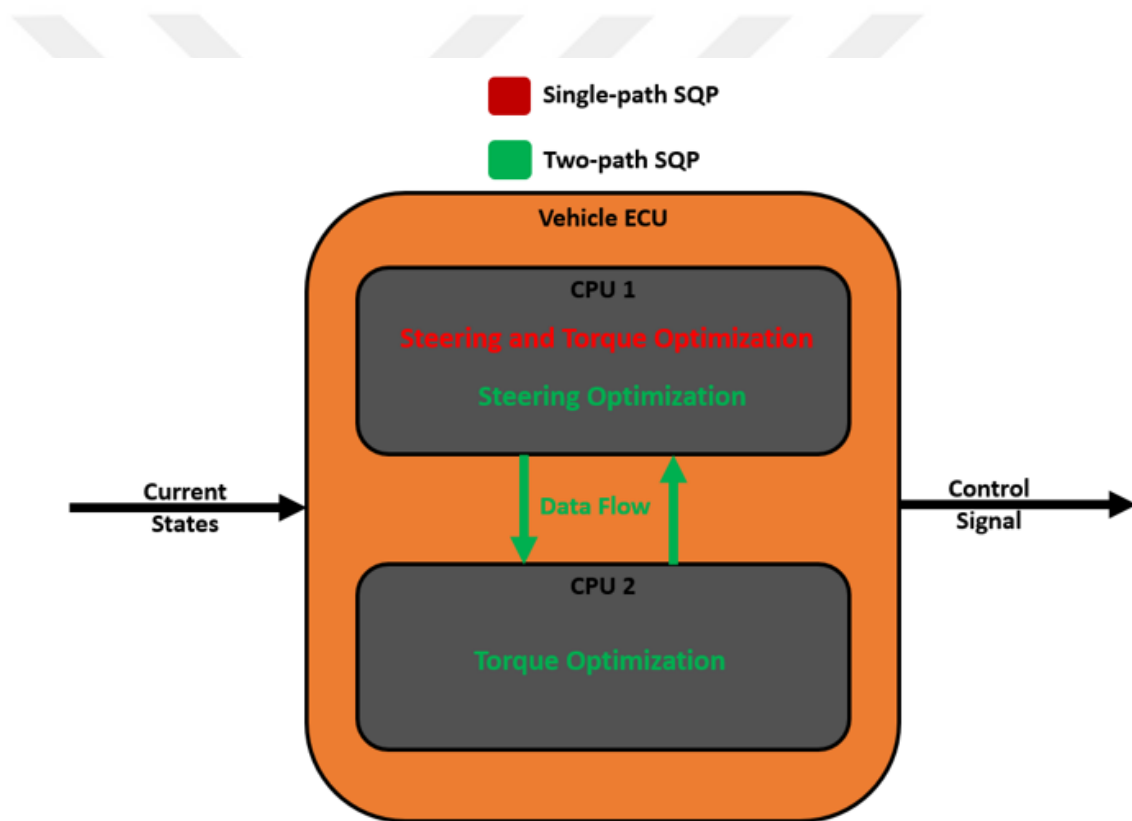


Figure 5.11: Single-Path vs Two-Path on Vehicle ECU

Chapter 6

Conclusion

This study introduces a novel physics-informed learning algorithm to estimate rapid vehicle tire cornering coefficients suitable to be used in conventional lateral stability control algorithms. Our simulation and experimental results show that, compared to the conventional regression-based learning algorithm and widespread Pacejka's method, the proposed method requires less data-set to fit collected data points. Hence, it allows real-time online parameter identification. The predicted coefficients straight forward and accurate enough to be used reference gain updates to improve the performance of control algorithms under varying road surfaces. The performance of online estimation, with experiments on the test vehicle, is verified by implementing it on an H_∞ yaw-rate regulator.

Additionally, a fault-tolerant model predictive control method for any over-actuated system to minimize the impact of actuator failures is presented. Model-based fault estimation technique combined with the online reconfiguration of the model predictive control constraints and control input matrix ensures that the deviation from the optimal performance is minimized during actuator failure. The proposed method can handle multiple actuator failures at the same time. Simulation results obtained with various scenarios verify that the proposed method successfully minimizes the effect of the failure by maintaining the vehicle lateral stability and reference tracking performance. Moreover, a processor-in-the-loop

test using an experimentally validated high-fidelity vehicle model proves that the implemented controller can be run in real time.

Lastly, a computational efficient nonlinear quadratic programming control allocation method for four-wheel-drive four-wheel-steering vehicle is presented. Simulation results show that the separation of steering and torque control allocation problems running parallel on different nodes of the ECU of the vehicle significantly reduces the computational load so that it maintains the lateral stability and reference tracking performance with lower control sample rates.

Bibliography

- [1] M. Sznaier, "Control Oriented Learning in the Era of Big Data," in IEEE Control Systems Letters, vol. 5, no. 6, pp. 1855-1867, Dec. 2021, doi: 10.1109/LC-SYS.2020.3045664.
- [2] A. J. Huang and S. Agarwal, "Physics Informed Deep Learning for Traffic State Estimation," 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC), 2020, pp. 1-6, doi: 10.1109/ITSC45102.2020.9294236.
- [3] D. Piga, M. Forgione, S. Formentin and A. Bemporad, "Performance-Oriented Model Learning for Data-Driven MPC Design," in IEEE Control Systems Letters, vol. 3, no. 3, pp. 577-582, July 2019, doi: 10.1109/LCSYS.2019.2913347.
- [4] B. A. H. Vicente, S. S. James and S. R. Anderson, "Linear System Identification Versus Physical Modeling of Lateral-Longitudinal Vehicle Dynamics," in IEEE Transactions on Control Systems Technology, vol. 29, no. 3, pp. 1380-1387, May 2021, doi: 10.1109/TCST.2020.2994120.
- [5] T. Kim and H. J. Kim, "Path tracking control and identification of tire parameters using on-line model-based reinforcement learning," 2016 16th International Conference on Control, Automation and Systems (ICCAS), 2016, pp. 215-219, doi: 10.1109/ICCAS.2016.7832324.
- [6] Bakker, E., Nyborg, L., and Pacejka, H., "Tyre Modelling for Use in Vehicle Dynamics Studies," SAE Technical Paper 870421, 1987, <https://doi.org/10.4271/870421>.

- [7] A. G. Ulsoy, H. Peng, and M. Çakmakci, "Automotive Control Systems". Cambridge: Cambridge University Press, 2012.
- [8] D. M. Nguyen, M. A. Kishk and M. -S. Alouini, "Toward Sustainable Transportation: Accelerating Vehicle Electrification With Dynamic Charging Deployment," in IEEE Transactions on Vehicular Technology, vol. 71, no. 9, pp. 9283-9296, Sept. 2022, doi: 10.1109/TVT.2022.3180495.
- [9] I. Husain et al., "Electric Drive Technology Trends, Challenges, and Opportunities for Future Electric Vehicles," in Proceedings of the IEEE, vol. 109, no. 6, pp. 1039-1059, June 2021, doi: 10.1109/JPROC.2020.3046112.
- [10] X. Ding, Z. Wang, L. Zhang and C. Wang, "Longitudinal Vehicle Speed Estimation for Four-Wheel-Independently-Actuated Electric Vehicles Based on Multi-Sensor Fusion," in IEEE Transactions on Vehicular Technology, vol. 69, no. 11, pp. 12797-12806, Nov. 2020, doi: 10.1109/TVT.2020.3026106.
- [11] R. Isermann, R. Schwarz and S. Stolz, "Fault-tolerant drive-by-wire systems," in IEEE Control Systems Magazine, vol. 22, no. 5, pp. 64-81, Oct. 2002, doi: 10.1109/MCS.2002.1035218.
- [12] C. Huang, F. Naghdy and H. Du, "Fault Tolerant Sliding Mode Predictive Control for Uncertain Steer-by-Wire System," in IEEE Transactions on Cybernetics, vol. 49, no. 1, pp. 261-272, Jan. 2019, doi: 10.1109/TCYB.2017.2771497.
- [13] W. Xiang, P. C. Richardson, C. Zhao and S. Mohammad, "Automobile Brake-by-Wire Control System Design and Analysis," in IEEE Transactions on Vehicular Technology, vol. 57, no. 1, pp. 138-145, Jan. 2008, doi: 10.1109/TVT.2007.901895.
- [14] C. Lin, S. Liang, J. Chen and X. Gao, "A Multi-Objective Optimal Torque Distribution Strategy for Four In-Wheel-Motor Drive Electric Vehicles," in IEEE Access, vol. 7, pp. 64627-64640, 2019, doi: 10.1109/ACCESS.2019.2917313.

- [15] B. Lu, J. Ma and Z. Zheng, "Adaptive Closed-Loop Control Allocation-Based Fault Tolerant Flight Control for an Overactuated Aircraft," in *IEEE Access*, vol. 7, pp. 179505-179516, 2019, doi: 10.1109/ACCESS.2019.2956818.
- [16] N. Guo, B. Lenzo, X. Zhang, Y. Zou, R. Zhai and T. Zhang, "A Real-Time Nonlinear Model Predictive Controller for Yaw Motion Optimization of Distributed Drive Electric Vehicles," in *IEEE Transactions on Vehicular Technology*, vol. 69, no. 5, pp. 4935-4946, May 2020, doi: 10.1109/TVT.2020.2980169.
- [17] O. Temiz, M. Cakmakci, and Y. Yildiz, "A Fault Tolerant Vehicle Stability Control Using Adaptive Control Allocation," *Dynamic Systems and Control Conference*, 2018, doi: 10.1115/DSCC2018-8976.
- [18] O. Temiz, M. Cakmakci, and Y. Yildiz, "A Fault-Tolerant Integrated Vehicle Stability Control Using Adaptive Control Allocation", in *International Journal of Adaptive Control and Signal Processing*, (in press), doi: 10.48550/arXiv.2008.05697.
- [19] R. Wang and J. Wang, "Fault-Tolerant Control With Active Fault Diagnosis for Four-Wheel Independently Driven Electric Ground Vehicles," in *IEEE Transactions on Vehicular Technology*, vol. 60, no. 9, pp. 4276-4287, Nov. 2011, doi: 10.1109/TVT.2011.2172822.
- [20] M. A. Djeziri, R. Merzouki, B. O. Bouamama and M. Ouladsine, "Fault Diagnosis and Fault-Tolerant Control of an Electric Vehicle Overactuated," in *IEEE Transactions on Vehicular Technology*, vol. 62, no. 3, pp. 986-994, March 2013, doi: 10.1109/TVT.2012.2231950.
- [21] P. E. Dumont, A. Aitouche, R. Merzouki and M. Bayart, "Fault Tolerant Control on an Electric Vehicle," 2006 IEEE International Conference on Industrial Technology, Mumbai, India, 2006, pp. 2450-2455, doi: 10.1109/ICIT.2006.372706.
- [22] T. Stolte, M. Loba, M. Nee, L. Wu and M. Maurer, "Toward Fault-Tolerant Vehicle Motion Control for Over-Actuated Automated Vehicles: A Non-Linear Model Predictive Approach," in *IEEE Access*, vol. 11, pp. 10499-10519, 2023, doi: 10.1109/ACCESS.2023.3239518.

- [23] J. Betz et al., "Autonomous Vehicles on the Edge: A Survey on Autonomous Vehicle Racing," in *IEEE Open Journal of Intelligent Transportation Systems*, vol. 3, pp. 458-488, 2022, doi: 10.1109/OJITS.2022.3181510.
- [24] L. Zhang, Z. Wang, X. Ding, S. Li and Z. Wang, "Fault-Tolerant Control for Intelligent Electrified Vehicles Against Front Wheel Steering Angle Sensor Faults During Trajectory Tracking," in *IEEE Access*, vol. 9, pp. 65174-65186, 2021, doi: 10.1109/ACCESS.2021.3075325.
- [25] H. A. Izadi, Y. Zhang, and B. W. Gordon, "Fault-tolerant model predictive control of quad-rotor helicopters with Actuator Fault Estimation," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 6343–6348, 2011.
- [26] E. Siampis, E. Velenis, S. Gariuolo and S. Longo, "A Real-Time Non-linear Model Predictive Control Strategy for Stabilization of an Electric Vehicle at the Limits of Handling," in *IEEE Transactions on Control Systems Technology*, vol. 26, no. 6, pp. 1982-1994, Nov. 2018, doi: 10.1109/TCST.2017.2753169.
- [27] Casper, Robert, and Erik Sundin. "Electrification in the Automotive Industry: Effects in Remanufacturing." *Journal of Remanufacturing* 11, no. 2 (2020): 121–36.
- [28] Zhao, Lin, Shaobo Lu, and Bohan Zhang. "Game-Based Hierarchical Cooperative Control for Electric Vehicle Lateral Stability via Active Four-Wheel Steering and Direct Yaw-Moment Control." *Energies* 12, no. 17 (2019): 3339.
- [29] Nicoletti, Lorenzo, Matthias Bronner, Benedikt Danquah, Alexander Koch, Adrian Konig, Sebastian Krapf, Aditya Pathak, et al. "Review of Trends and Potentials in the Vehicle Concept Development Process." 2020 Fifteenth International Conference on Ecological Vehicles and Renewable Energies (EVER), 2020.
- [30] Lu, Xiong, Chen Yuanlong, and Jin Chi. "A Nonlinear Control Allocation Algorithm for Dyc in Distributed-Motor Drive Electric Vehicles Using s-Sqp." *The 26th Chinese Control and Decision Conference (2014 CCDC)*, 2014.

- [31] Plumlee, J.H., D.M. Bevly, and A.S. Hodel. "Control of a Ground Vehicle Using Quadratic Programming Based Control Allocation Techniques." Proceedings of the 2004 American Control Conference, 2004.
- [32] Park, H., J. Sun, S. Pekarek, P. Stone, D. Opila, R. Meyer, Ilya Kolmanovsky, and Raymond DeCarlo. "Real-Time Model Predictive Control for Shipboard Power Management Using the IPA-SQP Approach." IEEE Transactions on Control Systems Technology 23, no. 6 (2015): 2129–43.
- [33] M. Cakmakci and A. G. Ulsoy, "Swappable Distributed MIMO Controller for a VCT Engine," IEEE Trans. Control Syst. Technol., vol. 19, no. 5, pp. 1168–1177, Sep. 2011
- [34] H. Noura, D. Theilliol, J. Ponsart, and A. Chamseddine, "Fault-tolerant Control Systems: Design and Practical Applications," 1st ed. London: Springer London, 2009.
- [35] Z. Wang, Y. Shen, "Model-Based Fault Diagnosis: Methods for State-Space Systems," 1st ed. Singapore: Springer Singapore, 2023.
- [36] C. Rother, Z. Zhou and J. Chen, "Development of a Four-Wheel Steering Scale Vehicle for Research and Education on Autonomous Vehicle Motion Control," in IEEE Robotics and Automation Letters, vol. 8, no. 8, pp. 5015–5022, Aug. 2023, doi: 10.1109/LRA.2023.3291916.
- [37] S. Karaman et al., "Project-based, collaborative, algorithmic robotics for high school students: Programming self-driving race cars at MIT," in Proc. IEEE Integr. STEM Educ. Conf., 2017, pp. 195–203.
- [38] X. Wu and A. Eskandarian, "An improved small-scale connected autonomous vehicle platform," in Proc. Dyn. Syst. Control Conf., 2019, Art. no. V001T04A003.
- [39] M. O'Kelly et al., "F1/10: An open-source autonomous cyber-physical platform," 2019, doi: 10.48550/arXiv.1901.08567.
- [40] The Mathworks Inc., "Simulink Support Package for Arduino Hardware," <https://www.mathworks.com/help/supportpkg/arduino/>.

- [41] The Mathworks Inc., "Generate C Code from Simulink Model," <https://www.mathworks.com/help/dsp/ug/generate-c-code-from-simulink-model.html>.
- [42] W. Zhang, N. Ding, G. Yu and W. Zhou, "Virtual sensors design in vehicle sideslip angle and velocity of the centre of gravity estimation," 2009 9th International Conference on Electronic Measurement & Instruments, 2009, pp. 3-652-3-656, doi: 10.1109/ICEMI.2009.5274228.
- [43] K. Koysuren, A. F. Keles and M. Cakmakci, "Online Parameter Estimation using Physics-Informed Deep Learning for Vehicle Stability Algorithms," 2023 American Control Conference (ACC), San Diego, CA, USA, 2023, pp. 466-471, doi: 10.23919/ACC55779.2023.10156092.
- [44] M. Cakmakci and M. Kemal Koysuren, "A Computationally Efficient Control Allocation Method for Four-Wheel-Drive and Four-Wheel-Independent-Steering Electric Vehicles," 2022 American Control Conference (ACC), Atlanta, GA, USA, 2022, pp. 3044-3049, doi: 10.23919/ACC53348.2022.9867549.
- [45] K. Zhou, J. C. Doyle, K. Glover, Robust and optimal control (1996).
- [46] L. Wang, "Model Predictive Control System Design and Implementation Using MATLAB®," 1st ed. London: Springer London, 2009.
- [47] B. Kouvaritakis and M. Cannon, "Model Predictive Control: Classical, Robust and Stochastic," 1st ed. Springer Cham, 2015.
- [48] Gill, P. E., W. Murray, M. A. Saunders, and M. H. Wright. "A practical anti-cycling procedure for linearly constrained optimization." Math. Programming 45 (1), August 1989, pp. 437-474.

Appendix A

Simulink Models for Arduino Programming

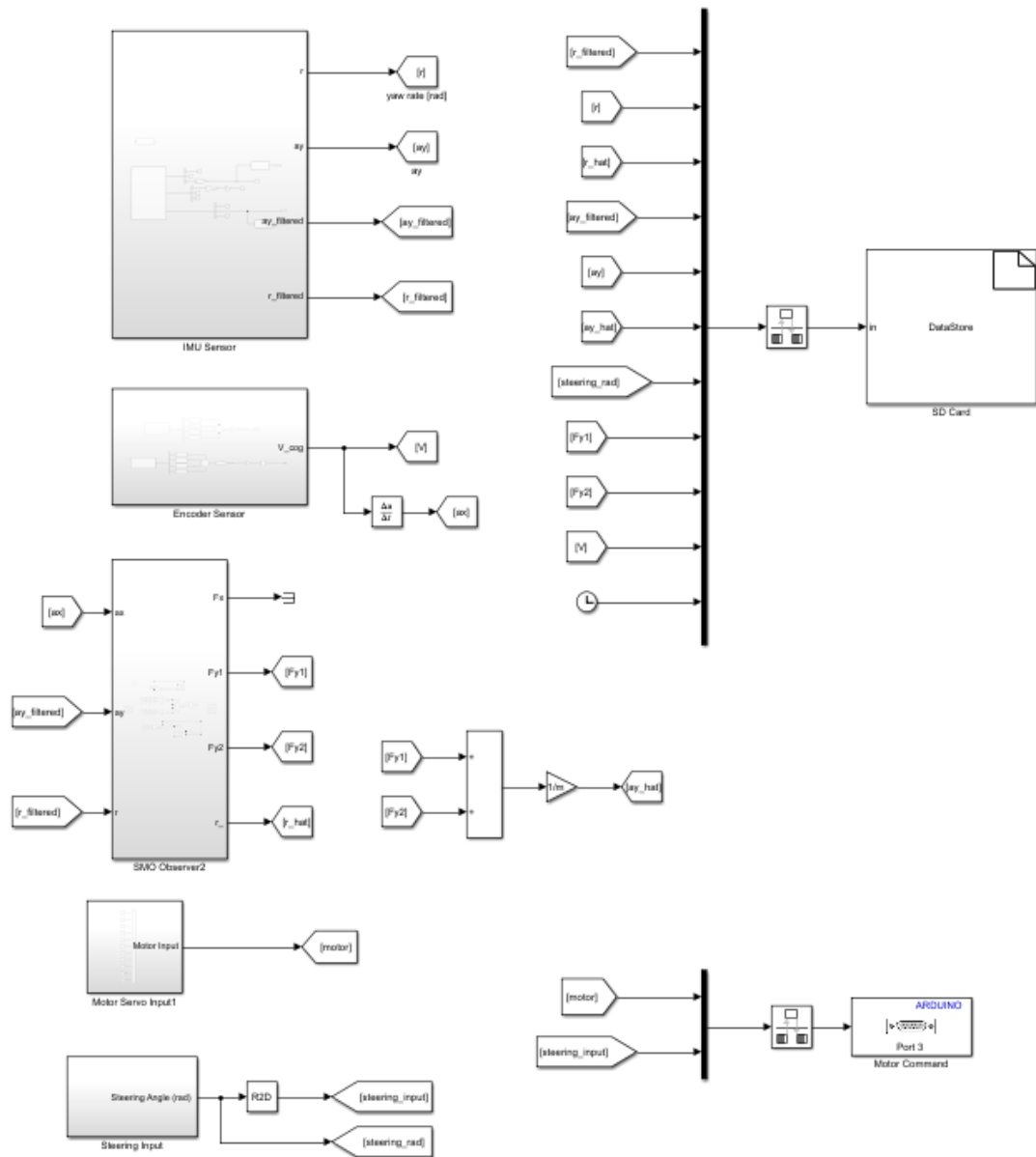


Figure A.1: Arduino Mega Sensor Data Collection

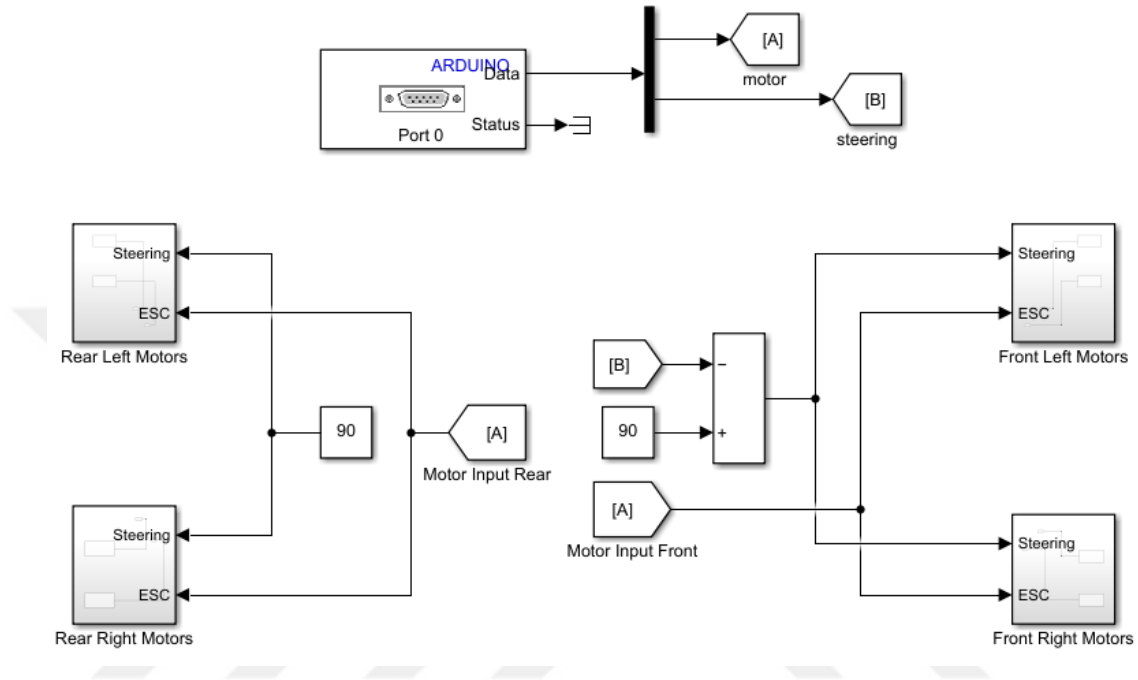


Figure A.2: Arduino Uno Motor Drive

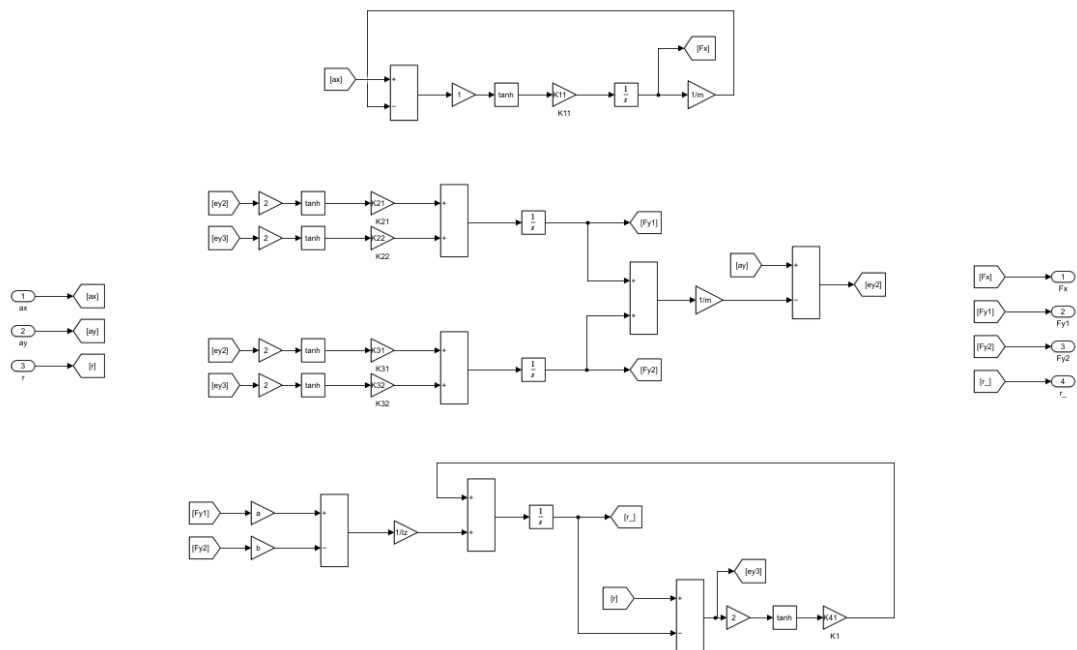


Figure A.3: Sliding Mode Observer for Tire Lateral Force Estimation

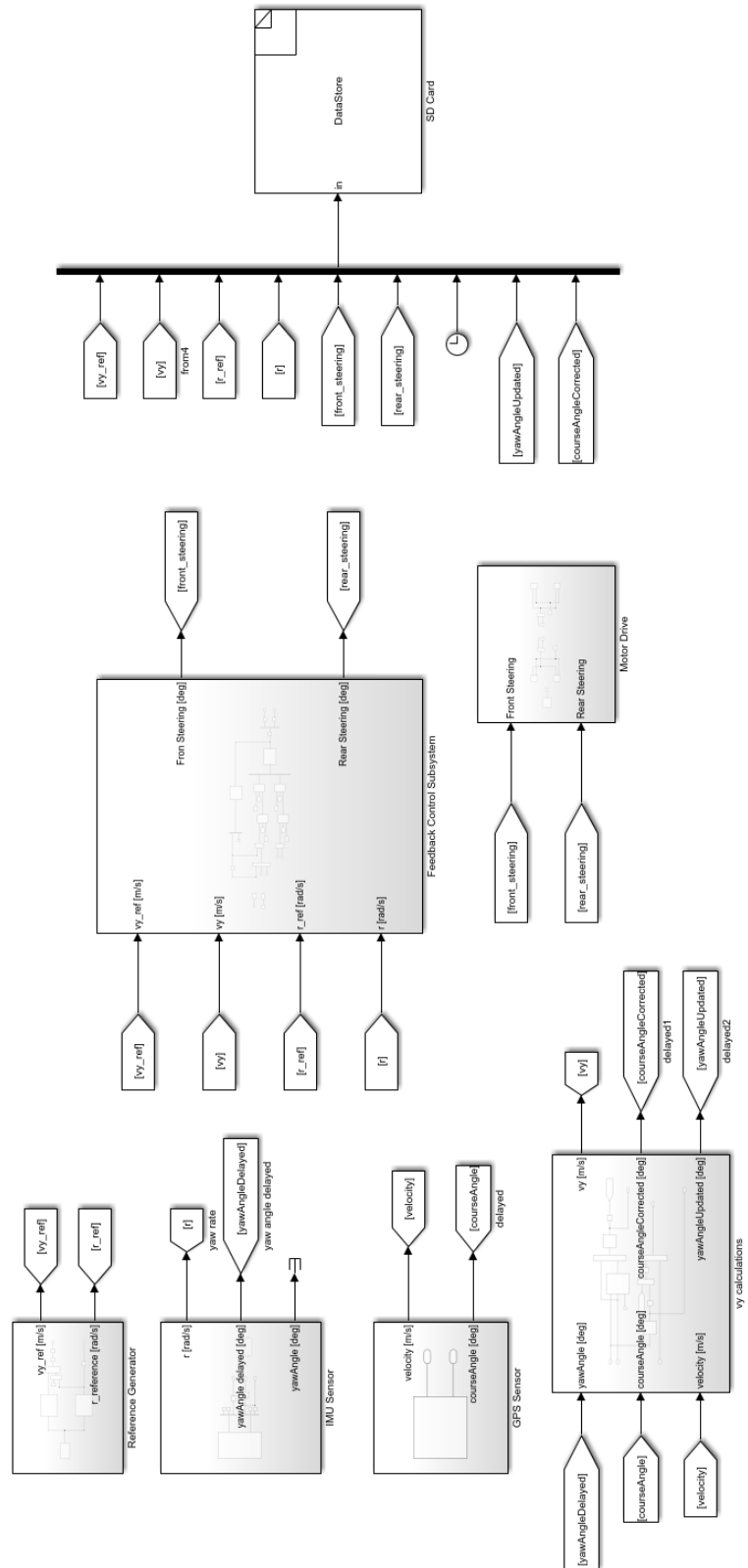


Figure A.4: MIMO Controller Arduino Due Implementation

Appendix B

Simulink Model for Model Predictive Controller

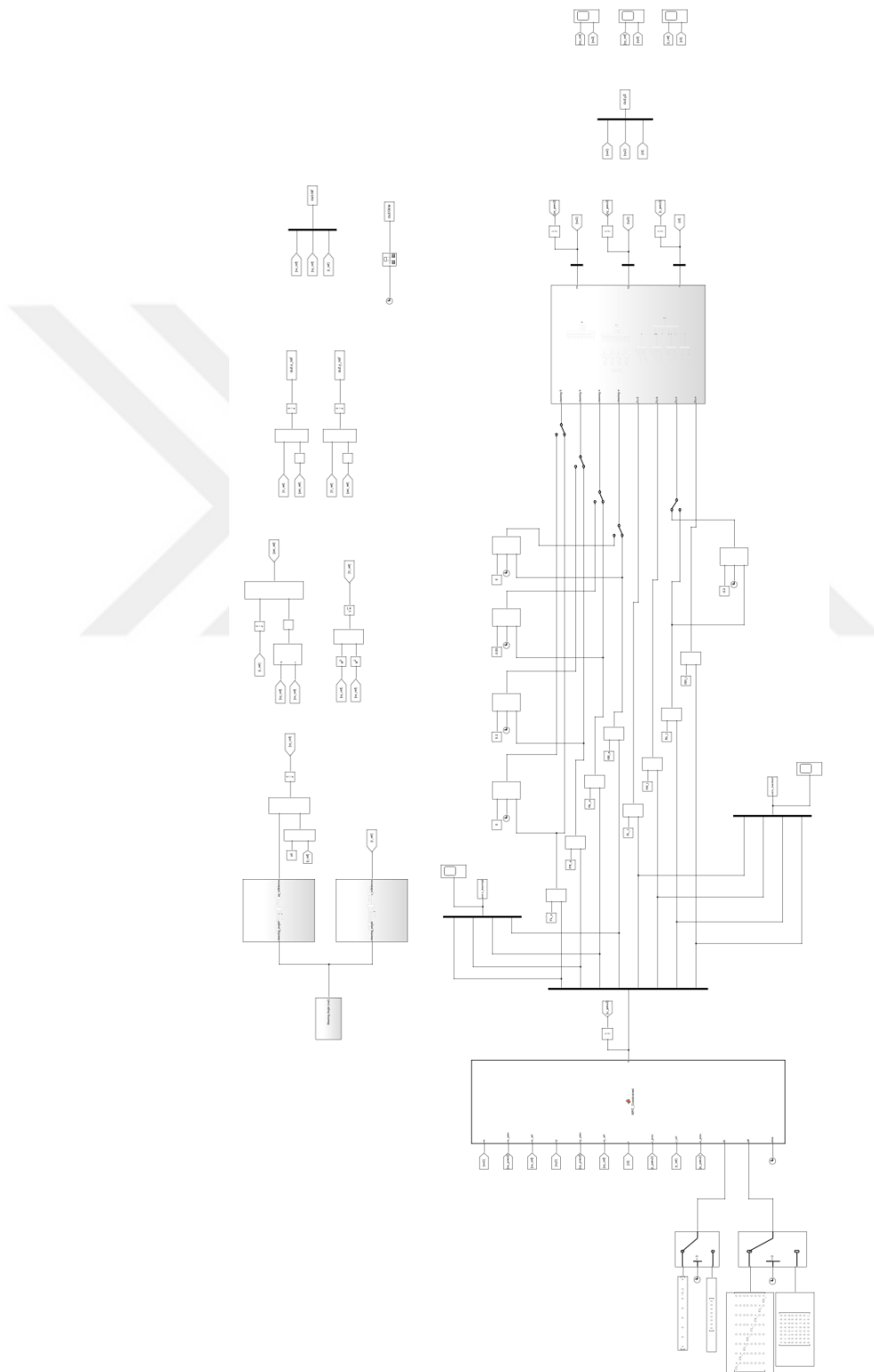


Figure B.1: Fault Tolerant Model Predictive Controller Simulink Model

Appendix C

Simulink Models for Control Allocation Methods

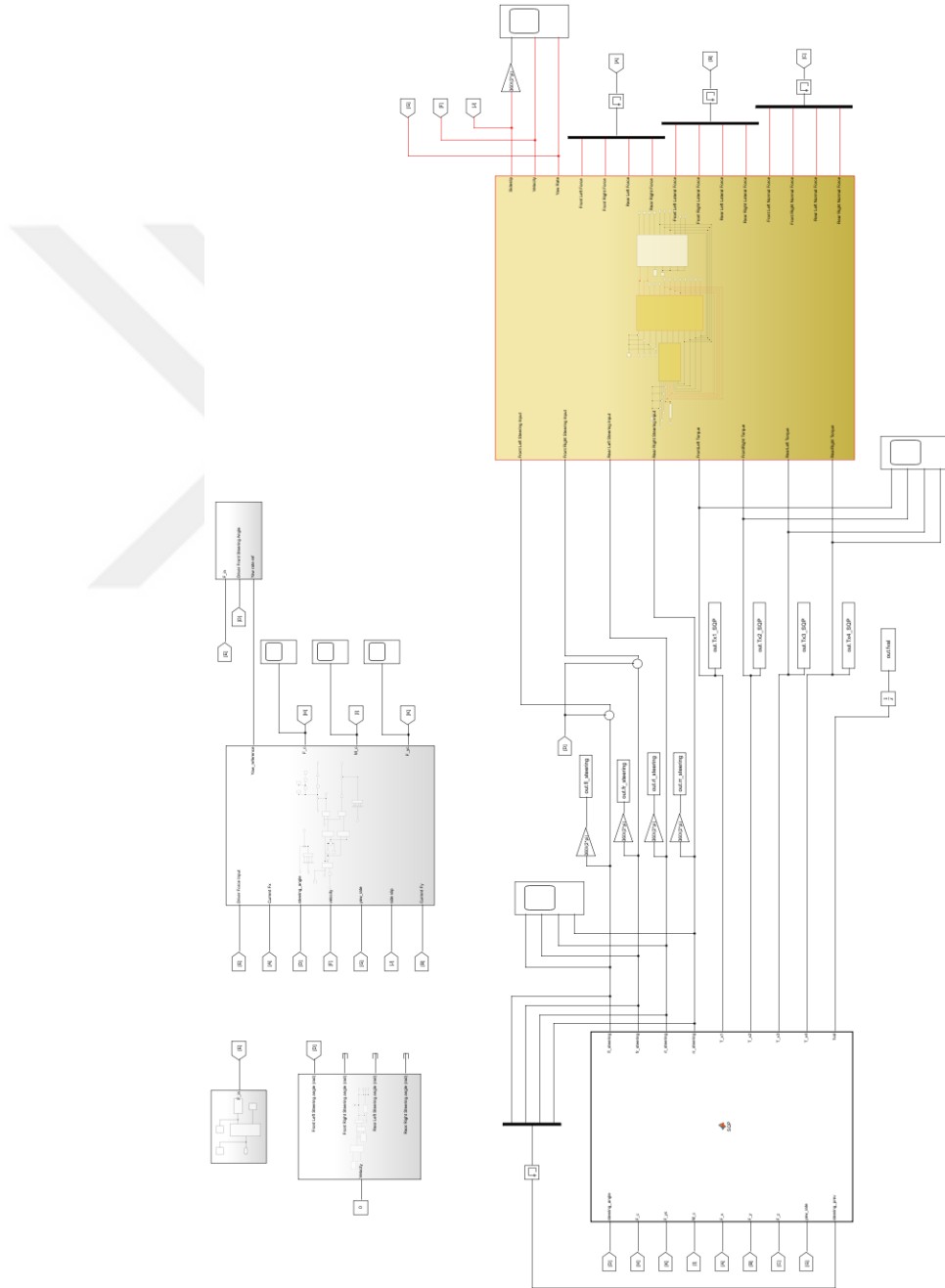


Figure C.1: Single-Path Control Allocation Simulink Model

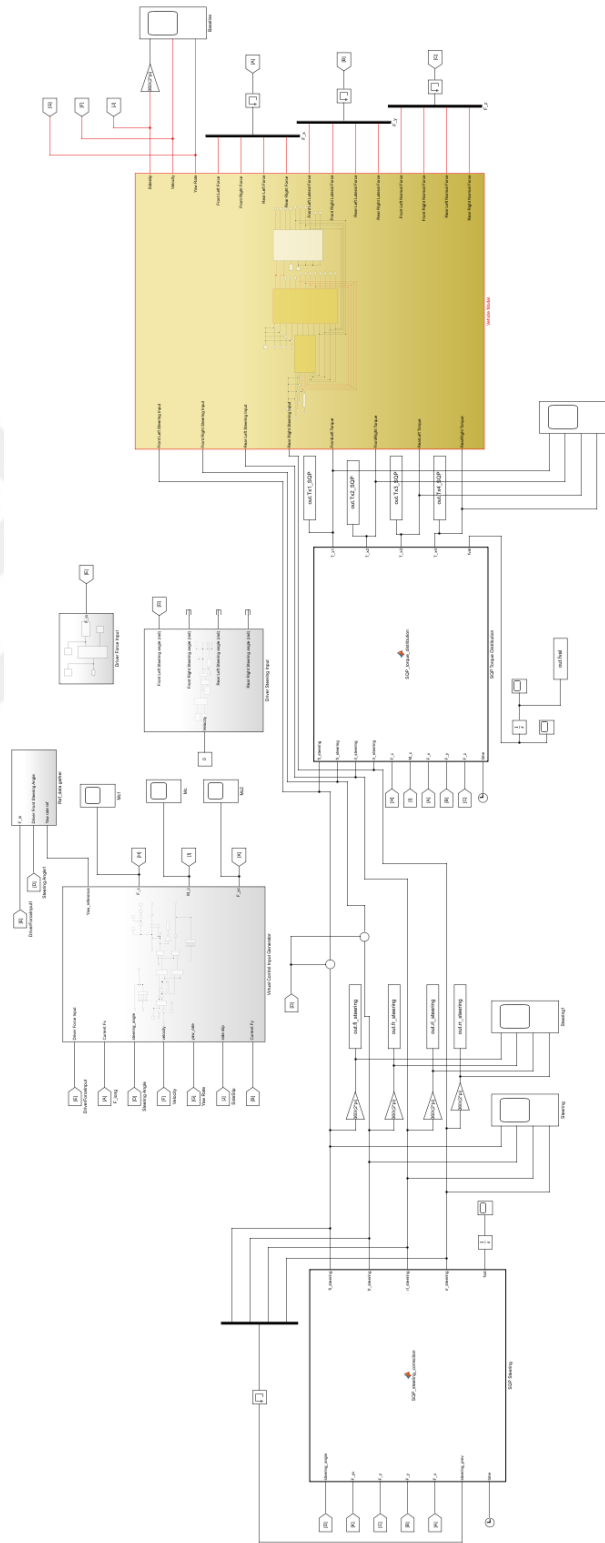


Figure C.2: Two-Path Control Allocation Simulink Model

Appendix D

Merit Function ψ for Feasibility Check

For the MPC QP optimization in Equation (4.18), the merit function for feasibility analysis of the constructed problem is calculated as:

$$\psi = \frac{1}{\rho}(\max(\|\mathbf{r}_{eq}\|_{\infty}, \|\mathbf{r}_{ineq}\|_{\infty}, \|\mathbf{r}_d\|_{\infty}) + \mathbf{g}) \quad (\text{D.1})$$

Coefficients in Equation (D.1) is expanded in Equation (D.2).

$$\begin{aligned} \rho &= \max(1, \|\mathbf{H}\|, \|\mathbf{A}_{ineq}\|, \|\mathbf{A}_{eq}\|, \|\mathbf{f}\|, \|\mathbf{b}_{ineq}\|, \|\mathbf{b}_{eq}\|) \\ \mathbf{r}_{eq} &= \mathbf{A}_{eq}\Delta\mathbf{U} - \mathbf{b}_{eq} \\ \mathbf{r}_{ineq} &= \mathbf{A}_{ineq}\Delta\mathbf{U} - \mathbf{b}_{ineq} - \mathbf{s} \\ \mathbf{r}_d &= \mathbf{H}\Delta\mathbf{U} + \mathbf{f} + \mathbf{A}_{eq}^T\lambda_{eq} + \mathbf{A}_{ineq}^T\lambda_{ineq} \\ \mathbf{g} &= \frac{1}{2}\Delta\mathbf{U}^T\mathbf{H}\Delta\mathbf{U} + \mathbf{f}^T\Delta\mathbf{U} - \mathbf{b}_{ineq}^T\lambda_{ineq} - \mathbf{b}_{eq}^T\lambda_{eq} \end{aligned} \quad (\text{D.2})$$

In Equation (D.2), λ_{eq} is Lagrange multipliers for the linear equality constraints, λ_{ineq} is Lagrange multipliers for the linear inequality constraints and $\mathbf{s}$ represents the vector of slacks that convert inequality constraints to equalities.

Other matrix notations are from the MPC optimization problem in Equation (4.18). Further details of merit function and feasibility analysis of quadratic programming are in [48].

