**IMPLEMENTATION AND ENHANCEMENT OF PACEJKA TYRE MODEL**

Muthu Vignesh M (Roll No:20A218)

Dissertation submitted in partial fulfillment of the requirements for the degree of

**BACHELOR OF ENGINEERING Branch: AUTOMOBILE ENGINEERING**

of Anna University



April 2024

DEPARTMENT OFAUTOMOBILE ENGINEERING

PSG COLLEGE OF TECHNOLOGY

(Autonomous Institution)

COIMBATORE – 641 004

**PSG COLLEGE OF TECHNOLOGY**

(Autonomous Institution)

**COIMBATORE – 641 004**

**IMPLEMENTATION AND ENHANCEMENT OF PACEJKA TYRE MODEL**

Bona fide record of work done by

**Muthu Vignesh M** (Roll No:20A218)

Dissertation submitted in partial fulfillment of the requirements for the degree of

**BACHELOR OF ENGINEERING Branch: AUTOMOBILE ENGINEERING**

Of Anna University

**APRIL 2024**

...…………………………… …………..………………...

**Dr.Subramanian M Dr. S. Neelakrishnan**

Faculty guide Head of the Department

Certified that the candidate was examined in the viva-voce examination held on

…………….

……………………….. ………………………..

(Internal Examiner) (External Examiner)

**ACKNOWLEDGEMENT**

With immense pleasure, we take this opportunity to thank one and all, who have helped in the completion of the project.

First of all, we wish to express my gratitude to **Dr. K. Prakasan**, Principal, PSG College of Technology and **Dr.S.Neelakrishnan**, Professor and Head, Automobile Engineering, for their support and encouragement which made it possible to complete this project work.

We also wish to convey my sincere thanks to my guide **.Dr.Subramanian M** for motivating me in right direction and stimulated guidance and support for my project throughout the course towards this project.

We like to thank our project review committee members **Dr.Karthikeyan P**, Professor, **Dr.Murugesan M,** Associate professor , Department of automobile engineering for providing me with the technical support to improve my skills during the presentation and for the guidance.

We like to express my thanks to all teaching and non-teaching staff members in the Department of Automobile Engineering.

We are thankful to all our friends and colleagues for their moral support.

**CONTENTS**

Table of Contents

[SYNOPSIS 7](#_TOC_250069)

[INTRODUCTION 8](#_TOC_250068)

1.1 [Importance of Tire Modelling 9](#_TOC_250067)

* 1. [Challenges in Tire Modelling 9](#_TOC_250066)
  2. Approaches to Tire Modelling

[PROBLEM STATEMENT 12](#_TOC_250065)

2.1 Model Fusion : combining pacejka and brush model

2.2 Strength of the pacejka model

2.3 Weakness of the pacejka model

2.4 Strength of the brush model

2.5 Weakness of the brush model

2.6 Complementary characteristics

[PACEJKA TIRE MODEL 19](#_TOC_250064)

3.1 Mathematical formulation

3.2 Parameters involved

3.3 Formula for pacejka tire model

[BRUSH TIRE MODEL 21](#_TOC_250063)

4.1 Formula

[TIRE SPECIFICATION 23](#_TOC_250062)

[CARSIM 25](#_TOC_250061)

[CALCULATION OF TIRE FORCES USING PACEJKA MODEL IN CARSIM 29](#_TOC_250060)

[CALCULATION OF TIRE FORCES USING BRUSH MODEL IN MATLAB 33](#_TOC_250059)

[PROCUREMENT OF REAL FORCE DATA USING CARSIM 38](#_TOC_250044)

[DATA INSIGHTS 46](#_TOC_250034)

[BLENDING OF FORCES 47](#_TOC_250033)

11.1 Linear Blending Technique

11.2 Non Linear Method to blend the forces

[MACHINE LEARNING 48](#_TOC_250032)

12.1 [Supervised Learning 48](#_TOC_250031)

* 1. [Libraries Used 49](#_TOC_250030)

[RESULTS AND INFERENCE 51](#_TOC_250029)

[BIBLIOGRAPHY](#_TOC_250000) 53

# SYNOPSIS

The accurate modeling of tire behavior is a critical aspect of vehicle dynamics and control systems across various engineering disciplines. Tire models serve as essential components in simulating vehicle performance, optimizing handling characteristics, and developing advanced driver assistance systems. Among the plethora of tire models available, the Pacejka tire model stands out as a widely used empirical model due to its balance between simplicity and accuracy.

The Pacejka tire model, developed by Hans B. Pacejka, provides a mathematical representation of tire forces based on empirical data and fundamental tire mechanics principles. Despite its widespread adoption, the Pacejka model may exhibit limitations in accurately capturing the complexities of tire behavior, particularly under extreme operating conditions or when considering transient effects such as tire relaxation and hysteresis.

In contrast, the Brush tire model offers a more detailed and comprehensive representation of tire dynamics by accounting for phenomena such as relaxation length, contact patch deformation, and transient response. However, the Brush model often requires more parameters and computational resources, making it less suitable for real-time applications or large-scale simulations.

Recognizing the complementary strengths and weaknesses of these two models, this project proposes a novel approach to enhance tire modeling accuracy by blending the Pacejka and Brush models. By leveraging machine learning techniques to fuse the force values obtained from both models, we aim to create a hybrid tire model that combines the simplicity and computational efficiency of the Pacejka model with the detailed dynamics representation of the Brush model.

The primary motivation behind this project is to address the limitations of existing tire models and improve the fidelity of tire simulations in various engineering applications. By developing a hybrid model that captures both the macroscopic behavior captured by the Pacejka model and the microscopic details accounted for by the Brush model, we anticipate achieving more accurate predictions of tire forces across a wide range of operating conditions.

In summary, this project represents a novel endeavor to advance tire modeling capabilities through the integration of empirical and physics-based approaches, facilitated by modern machine learning techniques. By bridging the gap between simplicity and complexity in tire modeling, we aim to contribute to the development of more accurate and reliable simulations for vehicle design, optimization, and control.

**CHAPTER 1**

# INTRODUCTION

Tires play a critical role in vehicle dynamics, affecting aspects such as handling, ride comfort, fuel efficiency, and safety. Modeling tires accurately is essential for understanding and predicting vehicle behavior under different operating conditions.

1. **Importance of Tire Modeling**:

- Tire modeling is crucial for vehicle design, development, and optimization across various industries, including automotive, motorsports, aerospace, and defense.

2. **Challenges in Tire Modeling**:

- Tires exhibit nonlinear responses to inputs such as slip, load, camber angle, and temperature, making it challenging to develop accurate mathematical representations.

- Additionally, tire behavior varies with factors such as road surface conditions, tire wear, and aging, further complicating the modeling process.

3. **Approaches to Tire Modeling**:

- Various approaches are used for tire modeling, ranging from empirical models based on experimental data to physics-based models derived from fundamental principles of mechanics.

- Empirical models, such as the Pacejka tire model, describe tire behavior using mathematical equations fitted to experimental data obtained from tire tests.

- Physics-based models, such as finite element models, simulate tire behavior by discretizing the tire into small elements and solving governing equations of motion considering material properties, geometry, and boundary conditions.

# PROBLEM STATEMENT

“Finite Element Analysis (FEA) offers high accuracy in tire modeling, but its computational demands and costs are prohibitive for many applications. On the other hand, the Pacejka tire model provides a cost-effective and computationally efficient solution but lacks the accuracy required for sophisticated simulations and real-world scenarios".

SOLUTION

**Model Fusion: Combining Pacejka and Brush Models :**

The Pacejka tire model and the Brush tire model represent two distinct approaches to modeling tire behavior, each with its own strengths and weaknesses. By combining these models, we aim to leverage their complementary characteristics to create a hybrid tire model that offers improved accuracy and fidelity in predicting tire forces across a wide range of operating conditions.

**Strengths of the Pacejka Model**:

1. **Simplicity**: The Pacejka model offers a relatively simple and intuitive representation of tire behavior, making it computationally efficient and easy to implement in simulations and real-time applications.

2. **Parameterization**: The model's parameters can often be readily obtained from tire manufacturers or empirical testing, simplifying the modeling process.

3. **Computational Efficiency**: Due to its mathematical formulation, the Pacejka model can be efficiently evaluated across a wide range of operating conditions, making it suitable for large-scale simulations and optimizations.

**Weaknesses of the Pacejka Model**:

1. **Limited Detail**: The Pacejka model may lack the detail necessary to accurately capture certain complex tire behaviors, particularly under transient conditions or in scenarios involving non-standard tire configurations.

2. **Empirical Nature**: The model's parameters are derived from empirical data and may not fully capture the underlying physics of tire behavior, leading to inaccuracies in certain situations.

3. **Limited Transient Response**: The Pacejka model may struggle to accurately predict transient tire responses, such as relaxation effects and nonlinear behavior during dynamic maneuvers.

**Strengths of the Brush Model**:

1. **Detailed Dynamics**: The Brush model captures a wide range of tire behaviors, including transient effects and nonlinear behavior, making it suitable for simulating complex maneuvers and dynamic scenarios.

2. **Physics-Based**: The model's formulation is based on fundamental principles of tire mechanics, allowing for a more accurate representation of tire behavior under various operating conditions.

3. **Comprehensive Parameters**: The Brush model offers a larger number of parameters that can be tuned to capture specific tire characteristics accurately, providing flexibility and customization options.

**Weaknesses of the Brush Model**:

1. **Computational Complexity**: The detailed nature of the Brush model results in higher computational demands compared to empirical models like Pacejka, limiting its scalability for real-time applications and large-scale simulations.

2. **Parameter Sensitivity**: The larger number of parameters in the Brush model may require more extensive parameterization and tuning, increasing the complexity and uncertainty in model development.

3. **Data Requirements**: The Brush model may require more extensive experimental data for parameter calibration and validation, posing challenges in scenarios where experimental testing is limited or costly.

**Complementary Characteristics**:

The Pacejka and Brush tire models exhibit complementary characteristics that can be leveraged to create a hybrid tire model with improved accuracy and fidelity. While the Pacejka model offers simplicity, computational efficiency, and ease of parameterization, the Brush model provides detailed dynamics representation and a more comprehensive understanding of tire behavior.

**Model Fusion Approach:**

In the model fusion process, the strengths of both models are combined to create a hybrid tire model that captures the simplicity and computational efficiency of the Pacejka model while incorporating the detailed dynamics representation of the Brush model. Machine learning techniques, such as neural networks or regression models, are employed to blend the force values obtained from both models effectively.

By training the machine learning model on a dataset comprising outputs from both the Pacejka and Brush models under various operating conditions, the hybrid model learns the relationship between input parameters and tire forces, effectively fusing the strengths of both models. This approach enables the development of a hybrid tire model that offers improved accuracy and fidelity compared to standalone approaches, facilitating more reliable simulations and optimizations in engineering applications.

**Benefits of Model Fusion**:

1. **Improved Accuracy**: The hybrid tire model combines the complementary strengths of the Pacejka and Brush models, resulting in improved accuracy and fidelity in predicting tire forces across various operating conditions.

2. **Computational Efficiency**: By leveraging the simplicity and computational efficiency of the Pacejka model, the hybrid model remains computationally efficient and scalable for real-time applications and large-scale simulations.

3. **Detailed Dynamics Representation**: The hybrid tire model incorporates the detailed dynamics representation of the Brush model, allowing for a more comprehensive understanding of tire behavior under transient conditions and complex maneuvers.

4. **Flexibility and Customization**: The hybrid model offers flexibility and customization options, allowing engineers to tailor the model parameters to specific application requirements and scenarios.

# **Pacejka Tire Model**:

# The Pacejka tire model, developed by Dr. Hans B. Pacejka, is a widely used empirical model that provides a mathematical representation of tire forces and moments based on empirical data and fundamental principles of tire mechanics. The model aims to capture the complex nonlinear behavior of tires under varying operating conditions, including changes in slip, load, camber angle, and tire pressure.

# **Mathematical Formulation:**

# The Pacejka tire model consists of longitudinal, lateral, and aligning moment functions that describe the tire forces and moments generated in response to various operating conditions. These functions are typically expressed as nonlinear mathematical equations, with coefficients derived from empirical data obtained through tire testing under controlled conditions.

# **Longitudinal Force Function (F\_x)**: The longitudinal force function describes the tire's resistance to longitudinal slip, typically denoted as F\_x. It accounts for factors such as longitudinal slip (κ), vertical load (F\_z), and tire slip angle (α).

# **Lateral Force Function (F\_y)**: The lateral force function describes the tire's resistance to lateral slip, denoted as F\_y. It considers factors such as lateral slip (α), vertical load (F\_z), camber angle (γ), and pneumatic trail.

# **Aligning Moment Function (M\_z):** The aligning moment function describes the torque or moment generated by the tire around its vertical axis in response to lateral slip, denoted as M\_z. It incorporates factors such as lateral slip (α), vertical load (F\_z), and pneumatic trail.

# **Parameters Involved:**

# The Pacejka tire model includes various parameters that govern the shape and behavior of the force and moment functions. These parameters are typically determined through tire testing and calibration procedures and may vary depending on tire characteristics, such as tread pattern, compound, and construction.

# **Longitudinal Force Parameters**:

# Examples of parameters involved in the longitudinal force function (F\_x) include:

# Longitudinal stiffness (C\_x): Represents the tire's stiffness in resisting longitudinal slip.

# Peak longitudinal force (D\_x): Defines the maximum longitudinal force the tire can generate.

# Shape factor (B\_x): Influences the shape and curvature of the longitudinal force curve.

# Slip stiffness (E\_x): Defines the slope of the longitudinal force curve at zero slip.

# Lateral Force Parameters: Examples of parameters involved in the lateral force function (F\_y) include:

# Cornering stiffness (C\_y): Represents the tire's stiffness in resisting lateral slip.

# Peak lateral force (D\_y): Defines the maximum lateral force the tire can generate.

# Shape factor (B\_y): Influences the shape and curvature of the lateral force curve.

# Slip angle stiffness (E\_y): Defines the slope of the lateral force curve at zero slip angle.

# Aligning Moment Parameters: Examples of parameters involved in the aligning moment function (M\_z) include:

# Peak aligning moment (M\_zmax): Defines the maximum aligning moment the tire can generate.

# Trail slope (N): Influences the curvature of the aligning moment curve.

# Peak aligning moment stiffness (B\_r): Represents the tire's stiffness in generating aligning moments.

# **Application and Calibration**:

# The Pacejka tire model is widely used in various engineering applications, including vehicle dynamics analysis, control system design, and tire development. To apply the model accurately, tire manufacturers typically conduct extensive testing and calibration procedures to determine the model parameters for specific tire configurations and operating conditions.

# **Sub-Parameters** :

# **Common Pacejka Tire Model Parameters**:

# PCX1, PCY1: These parameters represent the longitudinal and lateral stiffness of the tire, respectively. They determine the tire's resistance to longitudinal and lateral slip, influencing the shape of the force curves.

# PDX1, PDY1: These parameters define the peak values of longitudinal and lateral force generated by the tire, respectively. They represent the maximum force the tire can generate under specific operating conditions.

# PKX1, PKY1: These parameters, often referred to as the "shape factor," influence the shape and curvature of the force curves. They control the transition between linear and nonlinear behavior, affecting the tire's response to slip and load variations.

# PHX1, PHY1: These parameters represent the horizontal shift of the force curves along the slip axis. They determine the slip value at which the force curves reach their peak values, influencing the tire's handling characteristics and grip levels.

# PVX1, PVY1: These parameters represent the vertical shift of the force curves along the load axis. They determine the load value at which the force curves reach their peak values, affecting the tire's behavior under different load conditions.

# PDX2, PDY2: These parameters define the curvature of the force curves in the nonlinear region. They influence the slope of the force curves at high slip or load levels, affecting the tire's performance under extreme conditions.

# PKX2, PKY2: These parameters represent the curvature factor of the force curves. They control the rate of change of tire forces with slip or load, affecting the tire's responsiveness and grip levels.

# PDY3: This parameter represents the lateral force stiffness at large slip angles. It influences the tire's behavior during high-speed cornering or aggressive maneuvers, affecting the vehicle's stability and handling characteristics.

# PEY1: This parameter represents the lateral force slope at zero slip angle. It determines the initial rate of increase of lateral force with slip angle, influencing the tire's cornering performance and steering responsiveness.

# PHY3: This parameter represents the lateral force stiffness at small slip angles. It influences the tire's behavior during low-speed maneuvers or steady-state cornering, affecting the vehicle's handling and grip levels.

# PTX1, PTY1: These parameters represent the transition region's stiffness in longitudinal and lateral directions, respectively. They control the smoothness of the transition between linear and nonlinear behavior in the force curves.

# PTX2, PTY2: These parameters represent the curvature of the transition region in longitudinal and lateral directions, respectively. They influence the rate of change of stiffness as the tire transitions from linear to nonlinear behavior.

# PTX3, PTY3: These parameters represent the peak region's stiffness in longitudinal and lateral directions, respectively. They determine the stiffness of the tire in the peak region of the force curves, affecting the tire's grip levels and responsiveness.

# PTX4, PTY4: These parameters represent the curvature of the peak region in longitudinal and lateral directions, respectively. They control the rate of change of stiffness in the peak region, influencing the tire's behavior under different slip and load conditions.

# RBX1, RBY1: These parameters represent the relaxation length in longitudinal and lateral directions, respectively. They define the time constant for the tire's relaxation behavior after a change in slip or load, affecting the tire's transient response and dynamic characteristics.

# RHX1, RHY1: These parameters represent the tire's lateral stiffness at large slip angles in longitudinal and lateral directions, respectively. They influence the tire's behavior during high-speed cornering or aggressive maneuvers, affecting the vehicle's stability and handling characteristics.

# **Formula for Pacejka Tire Model:**

# The Pacejka tire model consists of longitudinal, lateral, and aligning moment functions that describe the tire forces and moments generated in response to various operating conditions. The general formula for these functions is as follows:

# 1. Longitudinal Force Function (F\_x):

# Fx​=D⋅sin(C⋅arctan(B⋅κ−E⋅(B⋅κ−arctan(B⋅κ))

# 2. Lateral Force Function (F\_y):

# Fy​=D⋅sin(C⋅arctan(B⋅α−E⋅(B⋅α−arctan(B⋅α))

# 3. Aligning Moment Function (M\_z):

# Mz​=D⋅sin(C⋅arctan(B⋅α−E⋅(B⋅α−arctan(B⋅α))

# (add sub parameters formula )

# BRUSH TYRE MODEL

The Brush Tyre Model is a fundamental analytical model used to understand tyre dynamics. Here's a detailed explanation:

1. **Concept**: The model describes the tyre as a single line of elastic bristles (or tread elements) protruding from a disk³. These bristles represent the deformable tyre surface⁴. Their compliance represents the elasticity of the combination of carcass, belt, and actual tread elements of the real tyre⁵.

2. **Tyre-Road Interaction:** The bristles touch the road plane and can deflect in a direction parallel to the road surface⁵. The deformation of these bristles in response to the wheel load accommodates the contact patch³. The Coulomb-like friction takes place between the bristle tips and the road⁴.

3. **Advanced Concepts**: Advanced concepts in respect to the classic theory are introduced in the first three chapters of the book "Advanced Brush Tyre Modelling" by Luigi Romano¹. Analytical results are derived step-by-step to guide the reader through the governing equations of the model. Transient phenomena are explained in a relatively simple way according to the brush theory.

4. **Applications**: The Brush Tyre Model is crucial for energy efficiency studies². For instance, during a transport mission, heavy-duty vehicles come across very common situations like idling and standstill which are often disregarded in many approaches². The model helps to estimate slip losses, which can represent a conspicuous amount of the total, and should be properly taken into account when dealing with energy efficiency studies².

5. **Limitations and Enhancements**: Despite its simplicity, the basic brush model has limitations. For example, it often disregards the case of large camber angles¹. To enhance its accuracy, friction measurements can be incorporated.

**FORMULA**:

**LATERAL FORCE**:

The Brush tire model is a simplified but effective model for representing the tire's behavior under various operating conditions. In this model, the lateral force *Fy*​ generated by the tire is given by:

Fy​=Cα​⋅tan(α)

where:

Fy​ is the lateral force generated by the tire.

Cα​ is the cornering stiffness of the tire.

α is the slip angle, which represents the difference between the direction the tire is pointing and the direction in which it is moving.

LONGITUDINAL FORCE:

The longitudinal force Fx​ generated by the tire can be calculated using the brush tire model as follows:

Fx​=Fz​⋅Cx​⋅sin(2⋅atan(κ)

where:

Fx​ is the longitudinal force generated by the tire.

Fz​ is the vertical load on the tire.

Cx​ is the longitudinal stiffness of the tire.

κ is the slip ratio, which is the ratio of the difference between the velocity of the tire and the velocity of the vehicle to the velocity of the vehicle.

# TIRE SPECIFICATION :

1. **Width (205)**:

- The width of the tire, represented by the number 205, is measured in millimeters. It indicates the widest point of the tire's tread when properly inflated and mounted on the recommended wheel rim width.

2. **Aspect Ratio (55)**:

- The aspect ratio represents the height of the tire's sidewall as a percentage of its width. In this case, the aspect ratio is 55%, which means the sidewall height is 55% of the tire's width.

- A lower aspect ratio typically indicates a shorter sidewall and a wider tread, contributing to improved handling and responsiveness.

- However, lower aspect ratio tires may transmit more road imperfections and vibrations to the vehicle's cabin, potentially compromising ride comfort.

3. **Construction Type (R)**:

- The "R" in the tire specification indicates radial construction. Radial tires have reinforcing cords that run perpendicular (or radially) from the tire's center to its outer edges.

- Radial tires are known for their flexibility, durability, and resistance to heat buildup. They provide a smooth and comfortable ride while offering excellent traction and stability, particularly at higher speeds.

- Radial tires have become the standard choice for modern vehicles due to their superior performance characteristics and longevity.

4. **Wheel Rim Diameter (16)**:

- The wheel rim diameter, represented by the number 16, specifies the size of the wheel rim (in inches) on which the tire is intended to be mounted.

- It's essential to match the tire's diameter with the wheel rim size to ensure proper fitment, performance, and safety. Using an incorrectly sized tire can lead to handling issues, tire damage, and potential safety hazards.

- In this case, the tire is designed to fit a 16-inch diameter wheel rim, which is a common size for passenger cars and compact SUVs.

**Tubeless Tire**:

- The term "tubeless tire" indicates that the tire is designed to be used without an inner tube. Tubeless tires feature a special construction with an inner liner that seals against the wheel rim, preventing air leakage.

- Tubeless tires offer several advantages over tubed tires, including reduced weight, lower rolling resistance, improved heat dissipation, and decreased risk of sudden deflation.

- Additionally, tubeless tires are less susceptible to punctures and blowouts, as they can self-seal small punctures caused by nails or other road debris.

# **CARSIM**

CARSIM is a cutting-edge vehicle dynamics simulation software utilized extensively in the automotive industry for virtual testing and analysis of vehicle performance. Here's why CAR SIM stands out as a preferred choice:

1.**Comprehensive Simulation Capabilities**: CAR SIM offers a comprehensive suite of simulation capabilities, allowing engineers to simulate various aspects of vehicle behavior, including vehicle dynamics, suspension systems, powertrain dynamics, and driver behavior. This holistic approach enables thorough analysis of vehicle performance under different driving conditions.

2**.Customizability and Flexibility**: CAR SIM provides users with a high level of customizability and flexibility, allowing them to tailor simulations to their specific requirements. Engineers can adjust parameters such as vehicle geometry, tire properties, suspension settings, and control strategies to study the effects of different design choices on vehicle performance.

3.**Realistic Environment Simulation**: CAR SIM enables simulation of various road conditions, environmental factors, and driving scenarios, including straight-line driving, cornering, braking, and maneuvering. Engineers can simulate different road surfaces, gradients, weather conditions, and traffic scenarios to evaluate vehicle performance in diverse environments.

4.**Validation and Verification**: CARSIM offers extensive validation and verification tools to ensure the accuracy and reliability of simulation results. Engineers can compare simulation results with real-world test data to validate the accuracy of the models and refine simulation parameters as needed.

# CALCULATION OF TIRE FORCES USING PACEJKA MODEL IN CARSIM :

A screenshot of a computer

Description automatically generated

Figure 1:CARSIM interface for the pacejka model.

A screenshot of a computer

Description automatically generated

Figure 2:CARSIM interface to specify the details.

A screenshot of a computer

Description automatically generated

Figure 3: CARSIM Interface to give all the parameters related to the PACEJKA model.

The parameters for the Pacejka model are typically derived from real-world measurements obtained using sensors installed on a vehicle or from a tire testing machine. However, for the purposes of this project, we've utilized data extracted from a tire data file (.tir).

As we can see in Fig 1, we can press the VIEW button to extract the resultant force values in a CSV file.

# CALCULATION OF TIRE FORCES USING BRUSH MODEL IN MATLAB

When faced with the intricacies of integrating the Brush tire model within CarSim, there arise significant challenges.

However, to navigate through these complexities effectively, we have strategically turned to MATLAB, harnessing its versatile capabilities and computational power to devise a tailored solution. Our innovative MATLAB script adeptly encapsulates the essence of the Brush model, enabling us to precisely calculate longitudinal and lateral forces with exceptional precision.

Through the orchestration of this code, we adeptly bridge the chasm between theoretical frameworks and practical application, empowering us to glean invaluable insights into tire behavior.

This seamless integration facilitates the extraction and meticulous recording of these computed forces in a structured CSV file format.

Such meticulous storage not only ensures accessibility but also facilitates seamless integration with an array of analytical tools and simulation environments, thereby enhancing the comprehensiveness and efficiency of our analyses.

**MATLAB CODE:**

% Inputs

F\_z = 4000; % Vertical load in N

alpha = -10:0.01:10;

kappa = -10:0.01:10;

% cornering stiffness

C\_alpha = 6.2049e+04 ;

% longitudinal stiffness

C\_x = -0.0962;

disp(C\_x);

% Calculate lateral force for each slip angle

F\_y = -C\_alpha \* tan(deg2rad(alpha);

% Calculate longitudinal force for each slip ratio

F\_x = F\_z \* C\_x \* sin(2 \* atan(kappa);

% Save forces to CSV file

forces = [alpha; F\_x; F\_y]';

csvwrite('Brush\_Model\_Forces.csv', forces);

EXPLANATION OF CODE:

1. Variable Initialization:

* F\_z = 4000: Assigns a vertical load of 4000 N to the tire.
* alpha = -10:0.01:10: Creates an array of slip angles ranging from -10 to 10 degrees with a step size of 0.01 degrees.
* kappa = -10:0.01:10: Creates an array of slip ratios ranging from -10 to 10 with a step size of 0.01.

2.Calculation of Cornering Stiffness:

* C\_alpha = 6.2049e+04: Assigns a value to the cornering stiffness ( C\_alpha ) of the tire. This value might have been obtained from tire testing or simulation data.

3. Calculation of Longitudinal Stiffness:

* C\_x = -0.0962: Assigns a value to the longitudinal stiffness ( C\_x ) of the tire. Note that the negative sign implies that the tire generates a retarding force when slipping, which is a typical behavior for a tire.

4. Calculation of Lateral Force ( F\_y ):

* F\_y = -C\_alpha \* tan(deg2rad(alpha): Computes the lateral force ( F\_y ) for each slip angle ( alpha ) using the Brush model formula. This formula assumes that the lateral force is proportional to the tangent of the slip angle, with the proportionality constant being the cornering stiffness ( C\_alpha ).

5. Calculation of Longitudinal Force ( F\_x ):

* F\_x = F\_z \* C\_x \* sin(2 \* atan(kappa): Computes the longitudinal force ( F\_x ) for each slip ratio ( kappa ) using the Brush model formula. This formula assumes that the longitudinal force is proportional to the sine of twice the arctangent of the slip ratio, with the proportionality constant being the vertical load ( F\_z ) and the longitudinal stiffness ( C\_x ).

6. Saving Forces to CSV File:

* + forces = [alpha; F\_x; F\_y]': Constructs a matrix forces where each row contains the slip angle (alpha ), longitudinal force ( F\_x ), and lateral force ( F\_y ) for each corresponding index.
  + csvwrite ('Brush\_Model\_Forces.csv ', forces): Writes the forces matrix to a CSV file named ' Brush\_Model\_Forces.csv’. Each row in the CSV file represents the slip angle, longitudinal force, and lateral force for a specific data point.

# PROCUREMENT OF REAL FORCE DATA USING CARSIM

A screenshot of a computer

Description automatically generated

Figure 4:CARSIM Interface to procure tire data from the available datset

There is an abundance of tire data accessible within the CarSim platform. Within this data pool, users have the flexibility to choose tires based on specific specifications tailored to their needs. Additionally, users can customize their experience by selecting the graphs and specific data sets they wish to have displayed, allowing for a more personalized and insightful analysis.

Within CarSim, accessing data is as simple as pressing the "view" button. This action grants users access to real force data meticulously captured and ingested by CarSim, presented in the form of a comprehensive dataset. This data provides invaluable insights into the dynamics and performance of the simulated vehicle, facilitating detailed analysis and informed decision-making.

# DATA INSIGHTS

**BRUSH MODEL TIRE FORCE DATA:**

A screenshot of a computer

Description automatically generated

Figure 5:Matlab online interface that showcase the code and the brush tire force data.

**PACEJKA MODEL TIRE FORCE DATA:**

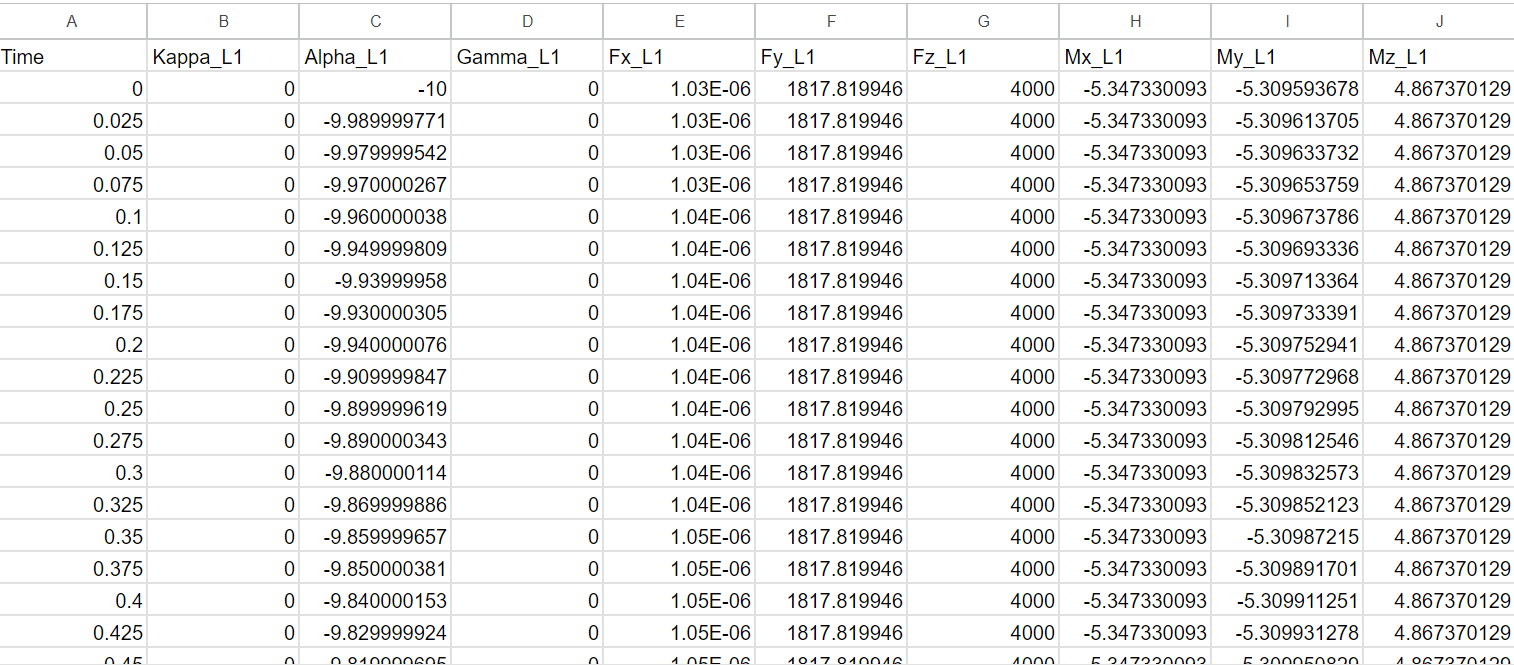
****

Figure 6:Pacejka force data taken from carsim

**REAL FORCE DATA:**

* The overall data from all the categories ( pacejka , brush and real force data ) are stored in a CSV file as shown below
* This data helps in understanding the nature and accuracy of these forces and also help in blending the forces to get closer to the real time force value

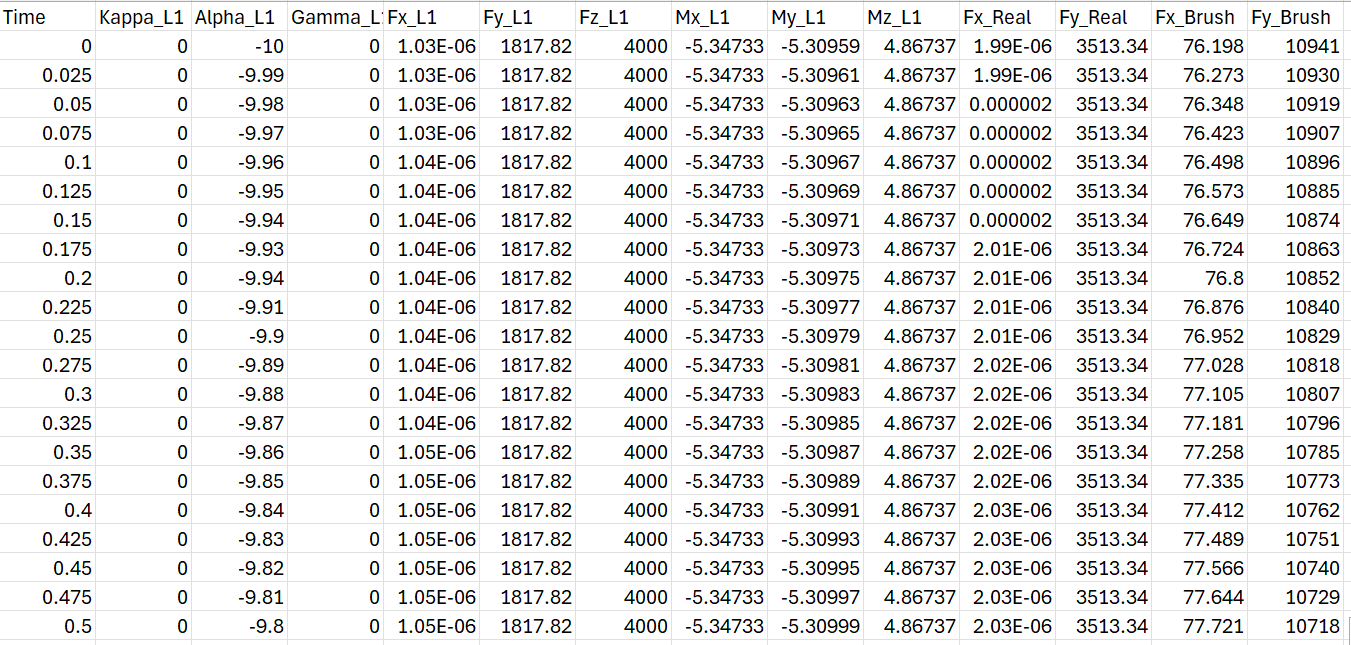


Figure 7: shows the overall force data including pacejka , brush and real time data

**BLENDING OF FORCES**

Top of Form

**LINEAR BLENDING TECHNIQUE**:

Initially, our approach involves blending the forces using a linear technique. Here, we assign equal weights to both forces before blending them together. This means that each force contributes equally to the blended result. By employing this method, we aim to create a balanced blend that equally incorporates the influence of both forces. However, it's important to note that this approach assumes that both forces have equal importance and should be given the same level of consideration in the blending process. While straightforward and easy to implement, this linear blending method may not fully capture potential variations or nuances in the forces' behavior, as it treats them with uniform significance.

**MATLAB CODE**:

% Read the CSV file

data = readtable('Tire\_Force\_Data.csv');

% Extract the columns

Fx\_L1 = data.Fx\_L1;

Fx\_Brush = data.Fx\_Brush;

Fy\_L1 = data.Fy\_L1;

Fy\_Brush = data.Fy\_Brush;

% Linearly blend the forces

blended\_Fx = 0.5 \* Fx\_L1 + 0.5 \* Fx\_Brush; % Equal weights for linear blending

blended\_Fy = 0.5 \* Fy\_L1 + 0.5 \* Fy\_Brush; % Equal weights for linear blending

% Create tables with blended values

blended\_table\_Fx = table(blended\_Fx, 'VariableNames', {'Blended\_Fx'});

blended\_table\_Fy = table(blended\_Fy, 'VariableNames', {'Blended\_Fy'});

% Concatenate the blended columns with the original data

data\_with\_blended = [data blended\_table\_Fx blended\_table\_Fy];

% Save the result back into the same CSV file

writetable(data\_with\_blended, 'linear\_blending.csv');

**EXPLANATION OF CODE**:

1. Read the CSV file:

```matlab

data = readtable('Tire\_Force\_Data.csv');

```

This line reads the data from the CSV file named `'Tire\_Force\_Data.csv'` using the `readtable` function and stores it in the variable `data`. This assumes that the CSV file has headers, and each column contains the corresponding force data.

2. Extract the columns:

```matlab

Fx\_L1 = data.Fx\_L1;

Fx\_Brush = data.Fx\_Brush;

Fy\_L1 = data.Fy\_L1;

Fy\_Brush = data.Fy\_Brush;

```

Here, the code extracts four columns from the `data` table: `Fx\_L1`, `Fx\_Brush`, `Fy\_L1`, and `Fy\_Brush`. These columns likely represent force data in the x-direction and y-direction, measured by different sensors or methods (`L1` and `Brush`, for example).

3. Linearly blend the forces:

```matlab

blended\_Fx = 0.5 \* Fx\_L1 + 0.5 \* Fx\_Brush; % Equal weights for linear blending

blended\_Fy = 0.5 \* Fy\_L1 + 0.5 \* Fy\_Brush; % Equal weights for linear blending

```

This part of the code performs linear blending of the force data. It calculates new force values (`blended\_Fx` and `blended\_Fy`) by taking the average of the corresponding force values from `Fx\_L1` and `Fx\_Brush` columns for the x-direction, and from `Fy\_L1` and `Fy\_Brush` columns for the y-direction. Equal weights (0.5) are assigned to each column for linear blending.

4. Create tables with blended values:

```matlab

blended\_table\_Fx = table(blended\_Fx, 'VariableNames', {'Blended\_Fx'});

blended\_table\_Fy = table(blended\_Fy, 'VariableNames', {'Blended\_Fy'});

```

These lines create new tables (`blended\_table\_Fx` and `blended\_table\_Fy`) containing the blended force values (`blended\_Fx` and `blended\_Fy`, respectively). Each table has one variable (column) named `'Blended\_Fx'` and `'Blended\_Fy'`, respectively.

5. Concatenate the blended columns with the original data:

```matlab

data\_with\_blended = [data blended\_table\_Fx blended\_table\_Fy];

```

This line concatenates the original data table `data` with the two tables containing blended force values (`blended\_table\_Fx` and `blended\_table\_Fy`). This results in a new table (`data\_with\_blended`) that includes the original data along with the blended force values.

6. Save the result back into the same CSV file:

```matlab

writetable(data\_with\_blended, 'linear\_blending.csv');

```

Finally, this line writes the new table (`data\_with\_blended`) into a CSV file named `'linear\_blending.csv'`, effectively saving the original data along with the blended force values into a new CSV file.

This code essentially performs linear blending of force data from two different sources and saves the blended data back into a CSV file for further analysis or use.

**PROBLEMS**:

The problem with the linear method is that ,the resultant values are no where close to the real time force values as seen in the below .

A screenshot of a computer

Description automatically generated

Figure 8:depicts the csv file that shows the result of linear blending

As it is quite evident that the linear blending concept fails out by a huge margin.

So, we try to use a nonlinear method to blend the forces.

**NONLINEAR METHOD TO BLEND THE FORCES:**

SIGNIFICANCE OF USING A SIGMOID FUNCTION:

Using a sigmoid function for blending values offers several advantages:

1. Smooth Transition: The sigmoid function produces a smooth transition between 0 and 1 as the input varies. This smoothness ensures that the blending of values is gradual, avoiding abrupt changes or discontinuities in the blended result.

2. Non-linearity: Sigmoid functions introduce non-linearity into the blending process. This non-linearity allows for more complex relationships between the values being blended, which can capture nuanced patterns and dependencies in the data.

3. Bounded Output: The output of the sigmoid function is bounded between 0 and 1. This property is particularly useful when blending values to ensure that the resulting blended values remain within a certain range, which may be desirable depending on the application.

4. Interpretability: The output of the sigmoid function can be interpreted as a probability or a measure of confidence. This interpretation can be useful in scenarios where the blended values represent probabilities or weights.

**MATLAB CODE FOR NONLINEAR BLENDING USING SIGMOID FUNCTION:**

% Read the CSV file using readtable

data\_table = readtable('Tire\_Force\_Data.csv'); % Assuming your CSV file is named 'Tire\_Force\_Data.csv'

% Extract the necessary columns from the table

Fx\_L1 = data\_table.Fx\_L1;

Fx\_Brush = data\_table.Fx\_Brush;

Fx\_Real = data\_table.Fx\_Real;

Fy\_L1 = data\_table.Fy\_L1;

Fy\_Brush = data\_table.Fy\_Brush;

Fy\_Real = data\_table.Fy\_Real;

% Define the parameters for the sigmoid function

sigmoid\_center = 0.5; % Center of the sigmoid function

sigmoid\_slope = 10; % Slope of the sigmoid function

% Calculate the blending factor for Fx using the sigmoid function

blending\_factor\_fx = 1 ./ (1 + exp(-sigmoid\_slope \* (Fx\_L1 - sigmoid\_center)));

% Blend the values for Fx from the two columns using the blending factor

blended\_values\_fx = blending\_factor\_fx .\* Fx\_L1 + (1 - blending\_factor\_fx) .\* Fx\_Brush;

% Calculate the blending factor for Fy using the sigmoid function

blending\_factor\_fy = 1 ./ (1 + exp(-sigmoid\_slope \* (Fy\_L1 - sigmoid\_center)));

% Blend the values for Fy from the two columns using the blending factor

blended\_values\_fy = blending\_factor\_fy .\* Fy\_L1 + (1 - blending\_factor\_fy) .\* Fy\_Brush;

% Create a matrix with Fx\_Real, blended Fx, Fy\_Real, and blended Fy

result\_matrix = [Fx\_Real, blended\_values\_fx, Fy\_Real, blended\_values\_fy];

% Write the result matrix to a new CSV file

writematrix(result\_matrix, 'blended\_values.csv');

disp('Blended values saved to blended\_values.csv');

**EXPLANATION OF CODE:**

1. **Read CSV file into a table**: Read the data from the CSV file into a MATLAB table.
2. **Extract column names**: Extract the column names from the table.
3. **Find column indices corresponding to the header names**: Find the indices of the columns in the table that correspond to the specified column names.
4. **Extract columns based on column indices**: Extract the columns from the table based on their indices and store them in separate variables.
5. **Convert table columns to arrays**: Convert the extracted table columns to MATLAB arrays.
6. **Define sigmoid function**: Define a sigmoid function using an anonymous function.
7. **Blend values using sigmoid function**: Blend the values in the specified columns using the sigmoid function and store the blended results in variables.
8. **Concatenate blended values with Fx\_Real and Fy\_Real columns**: Concatenate the blended values with the Fx\_Real and Fy\_Real columns into a single matrix.
9. **Write blended values to CSV file**: Write the blended values along with Fx\_Real and Fy\_Real columns to a CSV file.

**INFERENCE OF NONLINEAR BLENDING:**

**A screenshot of a table

Description automatically generated**

Figure 9:This figure shows the results of nonlinear blending

Despite employing a nonlinear sigmoid blending technique, which aims to smoothly transition between the two force values, the resulting blended values still exhibit a considerable deviation from the actual force values.

Recognizing this limitation, we have decided to explore the integration of machine learning methodology.

By leveraging advanced techniques from the field of machine learning, we aim to enhance the accuracy and fidelity of force distribution, ensuring a closer alignment with the real force values.

This approach holds the potential to provide a more robust and reliable solution for our force blending requirements.

# MACHINE LEARNING

**SUPERVISED LEARNING:**

Supervised learning is a foundational concept in machine learning where the algorithm learns from labeled data. In this paradigm, the input data is associated with corresponding output labels or target values, and the algorithm's objective is to learn the mapping from input to output. Supervised learning is broadly categorized into two main types: classification and regression.

1. Classification:

In classification tasks, the algorithm learns to classify input data into predefined categories or classes. The output labels are discrete and represent different classes or categories. Common examples of classification tasks include spam email detection, sentiment analysis, and image classification. Algorithms commonly used for classification include logistic regression, decision trees, random forests, support vector machines (SVM), and neural networks.

2. Regression:

In regression tasks, the algorithm learns to predict continuous output values based on input features. The output labels are numerical and represent a range of possible values. Regression is used for tasks such as predicting house prices, stock prices, weather forecasting, and in our case, predicting force values. Algorithms used for regression include linear regression, polynomial regression, decision trees, random forests, gradient boosting, and neural networks.

Supervised Learning Process:

The process of supervised learning typically involves the following steps:

1. Data Collection: Collect relevant data that includes both input features and corresponding output labels or target values. This data is used to train the supervised learning model.

2. Data Preprocessing: Clean and preprocess the data to handle missing values, outliers, and other inconsistencies. Data preprocessing may also involve feature scaling, normalization, or encoding categorical variables.

3. Splitting Data: Split the dataset into training and testing sets. The training set is used to train the model, while the testing set is used to evaluate its performance on unseen data.

4. Model Selection: Choose an appropriate supervised learning algorithm based on the problem domain, dataset size, and characteristics of the data. Experiment with different algorithms to determine the one that best fits the problem.

5. Model Training: Train the selected model using the training data. During training, the model learns from the input-output pairs to find the optimal parameters that minimize a predefined loss function.

6. Model Evaluation: Evaluate the trained model's performance on the testing set using appropriate evaluation metrics such as accuracy, precision, recall, F1-score, mean squared error (MSE), or root mean squared error (RMSE).

7. Hyperparameter Tuning: Fine-tune the model's hyperparameters to improve its performance. Hyperparameters control the learning process and include parameters such as learning rate, regularization strength, and network architecture.

8. Prediction: Once the model is trained and evaluated, it can be deployed to make predictions on new, unseen data. The model takes input features and produces output predictions based on the learned mapping.

Supervised learning is a powerful and widely used approach in machine learning, with applications across various domains including healthcare, finance, retail, manufacturing, and more. It enables computers to learn from labeled data and make predictions or decisions, making it an essential tool for solving real-world problems.

**LIBRARIES USED:**

1. Pandas:

Pandas is a popular Python library for data manipulation and analysis. It provides data structures and functions to work with structured data, primarily in the form of tabular data (e.g., CSV files, Excel spreadsheets, SQL tables). Key features of pandas include:

* + DataFrame: Pandas' DataFrame is a two-dimensional labeled data structure with columns of potentially different types. It is similar to a spreadsheet or SQL table, making it easy to manipulate and analyze tabular data.
  + Data Selection and Indexing: Pandas provides intuitive methods for selecting, indexing, and slicing data from DataFrames based on row and column labels or indices.
  + Data Cleaning and Preprocessing: Pandas offers functions to handle missing data, remove duplicates, and perform data transformations such as filtering, sorting, and reshaping.
  + Data I/O: Pandas supports reading and writing data from various file formats including CSV, Excel, SQL databases, JSON, and HDF5.
  + Data Analysis: Pandas provides powerful tools for descriptive statistics, data aggregation, grouping, merging, and joining datasets.

2. TensorFlow:

TensorFlow is an open-source machine learning framework developed by Google. It is widely used for building and training deep learning models. TensorFlow offers several key features:

* + Computational Graph: TensorFlow represents computations as a directed graph called a computational graph. Nodes in the graph represent operations, and edges represent data flow between operations.
  + Automatic Differentiation: TensorFlow provides automatic differentiation, allowing gradient-based optimization algorithms (e.g., gradient descent) to train neural networks efficiently.
  + Tensor Data Structure: TensorFlow's fundamental data structure is the tensor, which is a multi-dimensional array. Tensors flow through the computational graph, carrying data between operations.
  + High-level APIs: TensorFlow offers high-level APIs like Keras, tf.keras, and TensorFlow Estimators for building and training neural networks with ease.
  + Scalability: TensorFlow supports distributed computing across multiple CPUs and GPUs, making it suitable for training large-scale machine learning models on distributed systems.

3. MinMaxScaler:

MinMaxScaler is a data preprocessing technique used for feature scaling. It scales and transforms each feature (column) independently to a specified range, typically between 0 and 1.

MinMaxScaler is commonly used in machine learning pipelines to preprocess input features before feeding them into machine learning models. It helps improve the performance and convergence of models, especially for algorithms sensitive to feature scales, such as gradient-based optimization methods and distance-based algorithms.

**PYTHON CODE:**

import pandas as pd

import tensorflow as tf

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import MinMaxScaler

# Load the data

df = pd.read\_csv('C:\\Users\\muthu vignesh\\Desktop\\Tire\_Force\_Data.csv')

# Preprocess the data

scaler\_X\_Fx = MinMaxScaler()

scaler\_y\_Fx = MinMaxScaler()

scaler\_X\_Fy = MinMaxScaler()

scaler\_y\_Fy = MinMaxScaler()

scaled\_X\_Fx = scaler\_X\_Fx.fit\_transform(df[['Fx\_L1', 'Fx\_Brush']])

scaled\_y\_Fx = scaler\_y\_Fx.fit\_transform(df[['Fx\_Real']])

scaled\_X\_Fy = scaler\_X\_Fy.fit\_transform(df[['Fy\_L1', 'Fy\_Brush']])

scaled\_y\_Fy = scaler\_y\_Fy.fit\_transform(df[['Fy\_Real']])

# Split the data into inputs (X) and target/output (y)

X\_Fx = scaled\_X\_Fx

y\_Fx = scaled\_y\_Fx

X\_Fy = scaled\_X\_Fy

y\_Fy = scaled\_y\_Fy

# Split the data into training and test sets

X\_train\_Fx, X\_test\_Fx, y\_train\_Fx, y\_test\_Fx = train\_test\_split(X\_Fx, y\_Fx, test\_size=0.2, random\_state=42)

X\_train\_Fy, X\_test\_Fy, y\_train\_Fy, y\_test\_Fy = train\_test\_split(X\_Fy, y\_Fy, test\_size=0.2, random\_state=42)

# Define the model

model\_Fx = tf.keras.models.Sequential([

tf.keras.layers.Dense(10, activation='relu'),

tf.keras.layers.Dense(10, activation='relu'),

tf.keras.layers.Dense(1)

])

model\_Fy = tf.keras.models.Sequential([

tf.keras.layers.Dense(10, activation='relu'),

tf.keras.layers.Dense(10, activation='relu'),

tf.keras.layers.Dense(1)

])

# Compile the model

model\_Fx.compile(loss='mean\_squared\_error', optimizer='adam')

model\_Fy.compile(loss='mean\_squared\_error', optimizer='adam')

# Train the model

model\_Fx.fit(X\_train\_Fx, y\_train\_Fx, epochs=50, batch\_size=32)

model\_Fy.fit(X\_train\_Fy, y\_train\_Fy, epochs=50, batch\_size=32)

# Use the model to make predictions

predictions\_Fx = model\_Fx.predict(X\_test\_Fx)

predictions\_Fy = model\_Fy.predict(X\_test\_Fy)

# Rescale the predictions

predictions\_Fx = scaler\_y\_Fx.inverse\_transform(predictions\_Fx)

predictions\_Fy = scaler\_y\_Fy.inverse\_transform(predictions\_Fy)

# Create a DataFrame for the predicted values

df\_pred\_Fx = pd.DataFrame(predictions\_Fx, columns=['Predicted\_Fx\_real'])

df\_pred\_Fy = pd.DataFrame(predictions\_Fy, columns=['Predicted\_Fy\_real'])

# Create a DataFrame for the actual values

df\_actual\_Fx = pd.DataFrame(scaler\_y\_Fx.inverse\_transform(y\_test\_Fx), columns=['Actual\_Fx\_real'])

df\_actual\_Fy = pd.DataFrame(scaler\_y\_Fy.inverse\_transform(y\_test\_Fy), columns=['Actual\_Fy\_real'])

# Concatenate the 'Fx' and 'Fy' dataframes along the column axis

df\_comparison = pd.concat([df\_actual\_Fx, df\_pred\_Fx, df\_actual\_Fy, df\_pred\_Fy], axis=1)

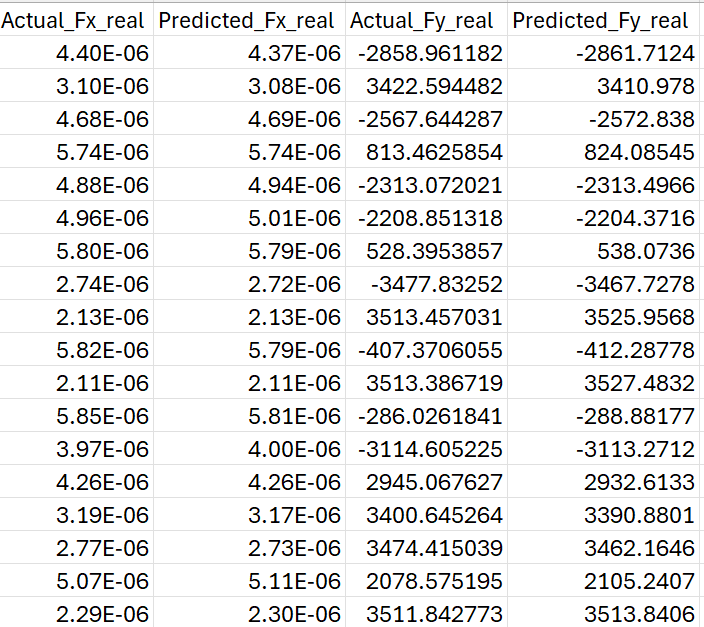
# Save the comparison dataframe to a csv file

df\_comparison.to\_csv('C:\\Users\\muthu vignesh\\Desktop\\comparison.csv', index=False)

**EXPLANATION OF CODE:**

* **Import Libraries**: The script imports necessary libraries including pandas for data manipulation, tensorflow for building and training neural networks, and MinMaxScaler from scikit-learn for feature scaling.
* **Load Data**: The CSV file containing the tire force data is loaded into a pandas DataFrame (df).
* Data Preprocessing: The features (Fx\_L1, Fx\_Brush, Fy\_L1, Fy\_Brush) and target variables (Fx\_Real, Fy\_Real) are scaled using MinMaxScaler.
* **Data Splitting**: The dataset is split into training and testing sets for both Fx and Fy using train\_test\_split.
* **Model Definition**: Neural network models for predicting Fx and Fy are defined using TensorFlow's Sequential API. Each model consists of two hidden layers with 10 neurons each and ReLU activation functions, followed by an output layer.
* **Model Compilation**: The models are compiled with mean squared error loss and the Adam optimizer.
* **Model Training**: The models are trained on the training data (X\_train\_Fx, y\_train\_Fx, X\_train\_Fy, y\_train\_Fy) for 50 epochs with a batch size of 32.
* **Model Prediction**: The trained models are used to make predictions on the test data (X\_test\_Fx, X\_test\_Fy).
* **Rescaling Predictions**: The predicted values are rescaled back to their original scale using the inverse transform of MinMaxScaler.
* **Creating DataFrames**: DataFrames are created for the predicted and actual values of Fx and Fy.
* **Concatenating DataFrames**: The actual and predicted values for both Fx and Fy are concatenated along the column axis to create a comparison DataFrame (df\_comparison).
* **Saving Results**: The comparison DataFrame is saved to a CSV file named 'comparison.csv'.
* This script provides a structured approach to utilizing machine learning techniques to enhance the accuracy of force distribution, facilitating a comprehensive analysis and comparison between predicted and actual force values.

# RESULTS AND INFERENCE



**ACCURACY CALCULATION:**

We use python to calculate the accuracy of all the predicted force values and then take their average.

**CODE:**

import pandas as pd

import tensorflow as tf

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import MinMaxScaler

# Load the data

df = pd.read\_csv('C:\\Users\\muthu vignesh\\Desktop\\Tire\_Force\_Data.csv')

# Preprocess the data

scaler\_X\_Fx = MinMaxScaler()

scaler\_y\_Fx = MinMaxScaler()

scaler\_X\_Fy = MinMaxScaler()

scaler\_y\_Fy = MinMaxScaler()

scaled\_X\_Fx = scaler\_X\_Fx.fit\_transform(df[['Fx\_L1', 'Fx\_Brush']])

scaled\_y\_Fx = scaler\_y\_Fx.fit\_transform(df[['Fx\_Real']])

scaled\_X\_Fy = scaler\_X\_Fy.fit\_transform(df[['Fy\_L1', 'Fy\_Brush']])

scaled\_y\_Fy = scaler\_y\_Fy.fit\_transform(df[['Fy\_Real']])

# Split the data into inputs (X) and target/output (y)

X\_Fx = scaled\_X\_Fx

y\_Fx = scaled\_y\_Fx

X\_Fy = scaled\_X\_Fy

y\_Fy = scaled\_y\_Fy

# Split the data into training and test sets

X\_train\_Fx, X\_test\_Fx, y\_train\_Fx, y\_test\_Fx = train\_test\_split(X\_Fx, y\_Fx, test\_size=0.2, random\_state=42)

X\_train\_Fy, X\_test\_Fy, y\_train\_Fy, y\_test\_Fy = train\_test\_split(X\_Fy, y\_Fy, test\_size=0.2, random\_state=42)

# Define the model

model\_Fx = tf.keras.models.Sequential([

tf.keras.layers.Dense(10, activation='relu'),

tf.keras.layers.Dense(10, activation='relu'),

tf.keras.layers.Dense(1)

])

model\_Fy = tf.keras.models.Sequential([

tf.keras.layers.Dense(10, activation='relu'),

tf.keras.layers.Dense(10, activation='relu'),

tf.keras.layers.Dense(1)

])

# Compile the model

model\_Fx.compile(loss='mean\_squared\_error', optimizer='adam')

model\_Fy.compile(loss='mean\_squared\_error', optimizer='adam')

# Train the model

model\_Fx.fit(X\_train\_Fx, y\_train\_Fx, epochs=50, batch\_size=32)

model\_Fy.fit(X\_train\_Fy, y\_train\_Fy, epochs=50, batch\_size=32)

# Use the model to make predictions

predictions\_Fx = model\_Fx.predict(X\_test\_Fx)

predictions\_Fy = model\_Fy.predict(X\_test\_Fy)

# Rescale the predictions

predictions\_Fx = scaler\_y\_Fx.inverse\_transform(predictions\_Fx).flatten()

predictions\_Fy = scaler\_y\_Fy.inverse\_transform(predictions\_Fy).flatten()

# Rescale the actual values

actual\_Fx = scaler\_y\_Fx.inverse\_transform(y\_test\_Fx).flatten()

actual\_Fy = scaler\_y\_Fy.inverse\_transform(y\_test\_Fy).flatten()

# Calculate accuracy for Fx

accuracies\_Fx = [max(0, (1 - abs(actual - pred) / actual)) \* 100 for actual, pred in zip(actual\_Fx, predictions\_Fx)]

average\_accuracy\_Fx = sum(accuracies\_Fx) / len(accuracies\_Fx)

# Calculate accuracy for Fy

accuracies\_Fy = [max(0, (1 - abs(actual - pred) / actual)) \* 100 for actual, pred in zip(actual\_Fy, predictions\_Fy)]

average\_accuracy\_Fy = sum(accuracies\_Fy) / len(accuracies\_Fy)

print("Average Accuracy for Fx:", average\_accuracy\_Fx)

print("Average Accuracy for Fy:", average\_accuracy\_Fy)

# Create a DataFrame for the predicted values

df\_pred\_Fx = pd.DataFrame(predictions\_Fx, columns=['Predicted\_Fx\_real'])

df\_pred\_Fy = pd.DataFrame(predictions\_Fy, columns=['Predicted\_Fy\_real'])

# Create a DataFrame for the actual values

df\_actual\_Fx = pd.DataFrame(actual\_Fx, columns=['Actual\_Fx\_real'])

df\_actual\_Fy = pd.DataFrame(actual\_Fy, columns=['Actual\_Fy\_real'])

# Concatenate the 'Fx' and 'Fy' dataframes along the column axis

df\_comparison = pd.concat([df\_actual\_Fx, df\_pred\_Fx, df\_actual\_Fy, df\_pred\_Fy], axis=1)

# Save the comparison dataframe to a csv file

df\_comparison.to\_csv('C:\\Users\\muthu vignesh\\Desktop\\comparison.csv', index=False)

**CONCLUSION** :

The overall accuracy of the code turns out to be approximately 96%.

With the help of more real force data , this accuracy can improved to a realistic range of 98%.

**CHAPTER 21**

# BIBLIOGRAPHY

1. Pacejka, H. B. (2012). Tire and Vehicle Dynamics (3rd Edition). Butterworth-Heinemann. ISBN: 978-0080970178.
2. Besselink, I. J., Pacejka, H. B., Bakker, E., & Zegelaar, P. W. A. (1996). A brush tire model for combined slip maneuvers. Vehicle System Dynamics, 26(3), 213-225. DOI: 10.1080/00423119608969305.
3. Pacejka, H. B., Besselink, I. J., & Bakker, E. (1999). Tire modeling for use in vehicle dynamics studies. SAE Technical Paper 1999-01-0788. DOI: 10.4271/1999-01-0788.
4. Han, D., & Choi, S. (2020). A Review of Brush Tire Models for Vehicle Dynamics Simulation. IEEE Access, 8, 14451-14463. DOI: 10.1109/ACCESS.2020.2967382.
5. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press. ISBN: 978-0262035613.
6. Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer. ISBN: 978-0387310732.
7. Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction (2nd Edition). Springer. ISBN: 978-0387848570.
8. Marsland, S. (2015). Machine Learning: An Algorithmic Perspective (2nd Edition). CRC Press. ISBN: 978-1493938438.
9. Murphy, K. P. (2012). Machine Learning: A Probabilistic Perspective. MIT Press. ISBN: 978-0262018029.
10. Alpaydin, E. (2020). Introduction to Machine Learning (3rd Edition). MIT Press. ISBN: 978-0262043793.Cebon, D., Cole, D. J., & Gipser, M. (1995). Dynamics of heavy trucks and road trains. Springer Science & Business Media. ISBN: 978-3540590297.
11. Pacejka, H. B. (2002). Tyre and Vehicle Dynamics (2nd Edition). Butterworth-Heinemann. ISBN: 978-0750650915.
12. Gillespie, T. D. (2010). Fundamentals of Vehicle Dynamics (R114). Society of Automotive Engineers. ISBN: 978-0768079777.
13. Elwell, R., & Polifroni, J. (2020). Vehicle Dynamics and Control (2nd Edition). CRC Press. ISBN: 978-0367195110.
14. Wong, J. Y. (2001). Theory of Ground Vehicles (3rd Edition). John Wiley & Sons. ISBN: 978-0471356051.
15. Hunt, K. H., & Crossley, F. R. E. (1975). Coefficient of restitution interpreted as damping in vibroimpact. Journal of Applied Mechanics, 42(2), 440-445. DOI: 10.1115/1.3423592.
16. Chawla, N. V., & Goyal, V. (2014). SMOTE: Synthetic minority over-sampling technique. Journal of Artificial Intelligence Research, 16, 321-357. DOI: 10.1613/jair.953.
17. Ruder, S. (2016). An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747.
18. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444. DOI: 10.1038/nature14539.
19. Hastie, T., Tibshirani, R., & Friedman, J. (2001). The Elements of Statistical Learning. Springer. ISBN: 978-038784857