

Studienarbeit

Implementation of the Pacejka Tire Model for a vehicle simulation in MATLAB

Name: Vishwanath Annegowda

Matr.-Nr.: 1230228

Supervisor: Prof. Dr.-Ing. Robert Mayr

Supervising Assistant: Dr.-Ing. Peter Will

Submission: 26.03.2018

Contents

1	Introduction to Pacejka Tire Model	4
1.1	Pacejka Model Description	5
1.2	Propagation of Tire Forces in Pacejka Tire Model	8
1.2.1	Equations used in Modeling of Pacejka Tire Model	8
1.2.2	Equations for Longitudinal Force	9
1.2.3	Equations for Lateral Force	10
2	Tire Dynamics	11
3	Implementation of Pacejka Tire model using Simulink and Matlab	14
3.1	Introduction to Simulink and Matlab	14
3.1.1	User-Defined Blocks in Simulink Environment	14
3.2	Implementation	15
4	Simulation Results	19
4.1	Comparing The Results of Burckhardt And Pacejka Model	19
4.2	Plots of generated forces	22

Implementation of the Pacejka Tire Model for a vehicle simulation in
MATLAB

4.2.1	Longitudinal Force	22
4.2.2	Lateral Force	23
5	Conclusion	25
Appendix A	Simulink Model and Matlab Code	28

List of Figures

1	Curve produced by magic formula	6
2	Sign convention and reference frame used for pacejka tire model	12
3	S-function execution structure from mathworks	15
4	Flowchart of pacejka model through S-function	18
5	Adopted track	22
6	Longitudinal force of the tires	23
7	Lateral force of the tires	24
8	Pacejka model implemented in a S-function	28
9	Simulation file of the vehicle model	29
10	Simulation file of tire model	29

List of Tables

1	Simulink report generated for pacejka tire model	20
2	Simulink report generated for burckhardt tire model	21

1 Introduction to Pacejka Tire Model

The first version of magic formula tire model was published by Bakker [1], is comprised of coefficients which defined some of the crucial quantities of a tire for instance, the slip stiffness when the slip is zero and the force and torque being at their maximum values [2]. The formula was efficient to provide all the characteristics of side force, self-aligning torque and brake force with excellent accuracy, however, the mathematical modelling was applicable to steady-state force analysis in a tire but was a building block for developing a tire model when braking and cornering are associated with each other [2].

The broadly used semi-observational tire model to measure force and also the moment developed by a tire for utilization in studies of vehicle dynamics is based on the so-called magic formula [3]. In the present adaptation of the model the initial depiction of the aligning torque has been altered to accommodate a relatively simple physically placed combined slip extension [3], which in my work has not been concentrated as my whole point of work was to generate the steady state forces generated from the tire model while used in a vehicle model.

A thorough description of pacejka model is described below in section 1.1 and in 1.2.1 the model is farther drawn out by enlisting all the formulae used for depicting the scenario at large camber angle γ and turn slip ϕ [3].

1.1 Pacejka Model Description

The familiar form of the formula, that holds for given values of vertical load and camber angle, reads

$$y = D \sin[C \arctan(Bx - E(Bx - \arctan Bx))] \quad (1)$$

with

$$Y(X) = y(x) + S_v \quad (2)$$

$$x = X + S_h \quad (3)$$

Where Y being output variable F_x , F_y or probably M_z and X representing the input variable which can be either $\tan \alpha$ or κ , and B is the stiffness factor, C the shape factor, D the peak value, E the curvature factor these above-mentioned factors play a definitive part in defining the nature of the curve shape, S_h the horizontal shift, and the S_v the vertical shift [3], which are considered to be the offset in the axis of interest as shown in figure 1.

For example, Y might represent a force generated from the tire or torque for instance and x being the slip quantity the output Y(i.e. F_x or F_y slip values) depends on. Every single factor that determines how the curve shapes out to be is found by approximating the data obtained from experiments for the respective tire and environment.

The pacejka tire model generally provides a curve that passes through the origin $x = y = 0$, reaches a typifying maximum value and afterward gravitates to a horizontal asymptote. For a given value of coefficients responsible for the

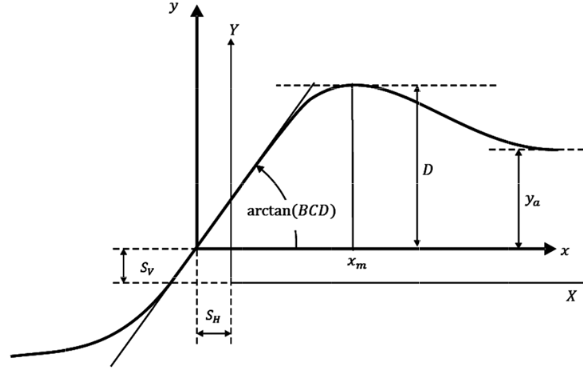


Figure 1: Curve produced by magic formula

[3]

curve, shapes show an anti-symmetric shape with respect to the origin [3]. To allow the curve to have an offset with regard to the origin as described above the two offsets S_h and S_v have been imported [3].

The pacjeka tire model is proficient in producing the attributes that firmly match measured curves for the side fore F_y and for the fore-and-aft force F_x as functions of their respective slip quantities, the slip angle α and the longitudinal slip κ with effect to the vertical load that vehicle bears F_z and the camber angle γ included in the parameters [3].

Figure 1 illustrates the context of some of the factors byways of a typical side force characteristic. Apparently, factor D represents the highest possible value with respect to the central x -axis and for $C > 1$ and the product BCD refers to the slope at origin ($x = y = 0$) [3]. The factor C which happens to be the factor which controls the shape of the curve has a control over limits of the range of the sine function which is modeled in the magic formula [4] and hence determines the characteristic of the curve, factor B is left to determine

the slope at the origin and is termed to be the stiffness factor. The term E is introduced to control the curvature at the maximum value and simultaneously the horizontal position at the maximum value [3].

A compelling property of the pacejka model is the case at the origin a local 'S'-shape may be detected. This leads to values being negative for E which emerges to be depending on the value of C. The critical value that E can have is found by considering the derivative of the third order of the curve at the origin [5].

In rather cases of extreme probabilities, the sharpness that can be attained by means of a function given in equation 1 may not be sufficient [3]. As it pans out that considerable increase in the sharpness of the curves by applying an additional term in the argument of arctan function, The modified formula reads,

$$y = D \sin[C \arctan(Bx - E(Bx - \arctan(Bx)) + H \arctan(Bx))] \quad (4)$$

However the arctan function are most of the times replaced by faster and almost identical pseudo-arctan function

$$psatan(x) = x(1 + a|x|)/(1 + 2(b|x| + ax^2\pi)) \quad (5)$$

with

$$a = 1.1, b = 1.6 \text{ [3].}$$

1.2 Propagation of Tire Forces in Pacejka Tire Model

Having described the basics of the magic formula created by H.Pacejka [3], in this section it would be utilised to form a basis for calculation of forces generated in the tire of an automotive or a vehicle.

The pivotal factors that play a decisive role in dynamics of tire and hence the vehicle dynamics are the longitudinal force, lateral force, slip, turn spin and tire forces have a functional dependency with longitudinal slip, vertical load, and side slip angle [6].

Below figure describes the sign conventions adopted from SAE(Society of Automotive Engineers), adapted version of SAE, ISO and adapted version of ISO(International Organization for Standards). For the description of forces in the pacejka tire model adapted version of SAE is used to define the force, moments and wheel slip [7].

1.2.1 Equations used in Modeling of Pacejka Tire Model

The essential format of the magic formula where the conditions are analogous to pure slip is depicted in this section in detail. For the detailed formulation of the pacejka model refer to chapter 4.3 of [3].

The properties of tire forces can be changed by the use of scaling factors defined by the user, pacejka model has several user scaling factors which helps in reducing the computational time for calculating the tire forces, which makes this kind of tire models to be used in different conditions pertaining to tire interactions [8].

1.2.2 Equations for Longitudinal Force

$$F_{xo} = D_x \sin[C_x \arctan(B_x \kappa_x - E_x (B_x \kappa_x - \arctan(B_x \kappa_x)))] + S_{Vx}$$

$$\kappa_x = s_x + S_{Hx}$$

$$C_x = p_{Cx1} \lambda_{Cx} (> 0)$$

$$D_x = \mu_x F_z \zeta_1 (> 0)$$

$$\mu_x = (p_{Dx1} + p_{Dx2} d_{fz})(1 + p_{px3} dp_i + p_{px4} dp_i^2) \lambda_{\mu x}$$

$$E_x = (p_{Ex1} + p_{Ex2} d_{fz} + p_{Ex3} d_{fz}^2) (1 - p_{Ex4} \text{sgn}(\kappa_x)) \lambda_{Ex} (< 1)$$

$$K_{x\kappa} = F_z (p_{Kx1} + p_{Kx} d_{fz}) \exp(p_{Kx3} d_{fz}) \lambda_{Kx sx}$$

$$B_x = K_{x sx} / (C_x D_x + \epsilon_x)$$

$$S_{Hx} = (p_{Hx1} + p_{Hx2} d_{fz}) \lambda_{Hx}$$

$$S_{Vx} = F_z (p_{Vx1} + p_{Vx2} d_{fz}) \lambda_{Vx} \lambda_{\mu x} \zeta_1$$

1.2.3 Equations for Lateral Force

$$F_{yo} = D_y \sin[C_y(\arctan(B_y \kappa_y) - E_y(B_y \kappa_y - \arctan(B_y \kappa_y)))] + S_{Vy}$$

$$\kappa_y = s_y + S_{Hy}$$

$$C_y = p_{Cy1} \lambda_{Cy} (> 0)$$

$$D_y = \mu_y F_z \zeta_1 (> 0)$$

$$\mu_y = (p_{Dy1} + p_{Dy2} d_{fz})(1 + p_{py3} dp_i + p_{py4} dp_i^2) \lambda_{\mu y}$$

$$E_y = (p_{Ey1} + p_{Ey2} d_{fz} + p_{Ey3} d_{fz}^2)(1 - p_{Ey4} \text{sgn}(\kappa_y)) \lambda_{Ey} (< 1)$$

$$K_{y\kappa} = F_z(p_{Ky1} + p_{Ky} d_{fz}) \exp(p_{Ky3} d_{fz}) \lambda_{Ky sy}$$

$$B_y = K_{y sy} / (C_y D_y + \epsilon_y)$$

$$S_{Hy} = (p_{Hy1} + p_{Hy2} d_{fz}) \lambda_{Hy}$$

$$S_{Vy} = F_z(p_{Vy1} + p_{Vy2} d_{fz}) \lambda_{Vy} \lambda_{\mu y} \zeta_1$$

For the computation of longitudinal force and lateral force the magic formula coefficients, i.e, B_x , C_x , D_x , E_x are defined user scaling factors [3].

2 Tire Dynamics

Before the implementation of the tire model, this section defines few of the basic parameters of a tire which play a crucial part in defining the tire dynamics and tire modelling.

Definitions

The coordinate frame used in this work is adapted from pacejka [3]. This is for practical reasons since the tire models proposed in [3] also use this as a reference.

From the below figure 2 following terms can be defined as,

V is the speed of travel of the wheel center.

γ is the camber angle.

α is the side slip angle.

ψ is referred to yaw speed.

F_x happens to be the force generated along x-axis.

F_y is termed as the side force that tire generates.

F_z Is the normal force at the contact surface.

F_x , F_y and M_z are depending on longitudinal slip κ , load F_z , side slip angle α and camber angle γ .

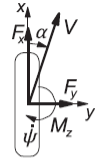
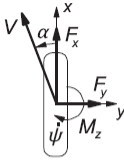
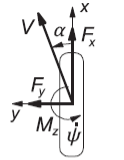
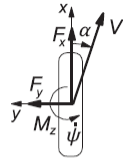
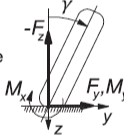
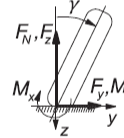
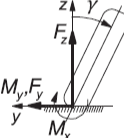
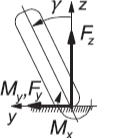
($V_x > 0$)	SAE	adapted SAE (Pacejka, this book)	ISO	adapted ISO (Besselink 2000)
side angle (top view)				
inclination/ camber angle (rear view)				
side slip	$\tan \alpha = \frac{V_{sy}}{V_x}$	$\tan \alpha = -\frac{V_{sy}}{V_x}$	$\tan \alpha = \frac{V_{sy}}{V_x}$	$\tan \alpha = -\frac{V_{sy}}{V_x}$
longitudinal slip	$\kappa = -\frac{V_{sx}}{V_x}$	$\kappa = -\frac{V_{sx}}{V_x}$	$\kappa = -\frac{V_{sx}}{V_x}$	$\kappa = -\frac{V_{sx}}{V_x}$
turn slip	not defined	$\varphi = -\frac{\dot{\psi}}{V}$	not defined	$\varphi = -\frac{\dot{\psi}}{V}$

Figure 2: Sign convention and reference frame used for pacejka tire model

[3]

Longitudinal and Lateral Slip

When a torque is applied to the tires, an analysis should be made between the actual leading speed of the tire over the road surface (speed along x-axis: V_x) and the velocity V_{xo} which happens to be the velocity of the tires [7].

$$V_{xo} = \omega r_e \quad (6)$$

Lateral or side slip is computed from the ratio of the lateral velocity and the forward velocity of the wheel [7].

$$\tan(\alpha) = -V_x/V_y \quad (7)$$

Again, positive side slip means positive side force. The slip angle α stands for the angle between the direction in which the tire is moving and the direction in which the tire is rolling at the same time [7].

Slip Ratio(Longitudinal slip)

Slip ratio κ , is the rate at which slip velocity varies with respect to the vehicle velocity.

$$\kappa = (V_x - r_e\omega)/(r_e\omega) \quad (8)$$

where V_x is defined to be the velocity of the wheel center in the forward direction with r_e as the effective radius and ω denoting the speed of revolution of the wheel body to be defined [3].

3 Implementation of Pacejka Tire model using Simulink and Matlab

3.1 Introduction to Simulink and Matlab

Simulink is a tool for multi-sphere simulation and model based design oriented dynamic systems. The environments matlab-simulink are unified into one entity, which will enable us to simulate, analyse our models in both of the environments at any given point [9]. Hence the platform of simulink was an efficient tool to implement the pacejka tire model and study its effects on a vehicle environment.

While executing the tire model two approaches were made,

1. Using the blocks continuous, discrete, math operation and ports and subsystem library blocks along with the most generalized blocks as well [9].
2. Since the pacejka model involves factors that are defined by the user to attain the results, the user-defined blocks of simulink were used to implement the functions, formulae and user-defined constants as described in figure 3 [3].

3.1.1 User-Defined Blocks in Simulink Environment

S-functions (system-functions) enable a powerful structure to simulink tool to increase its efficiency as a tool. S-function is a description language of a simulink block written in many programming languages which are compiled for desired results in simulink environment [9].

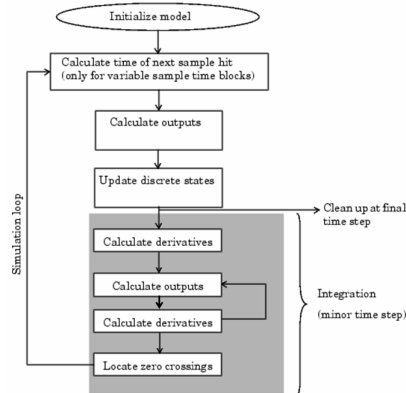


Figure 3: S-function execution structure from mathworks

[9]

S-functions enable us to include our own piece of code to simulink models, we can define the differential equations of our need and also discrete-time. We write our S-function which in this case would be the equations described in the first chapter and a detailed execution is shown in the appendix worth noting that the structure is not altered due to my supervisors instructions, hence only the part of burckhardt tire model was replaced by pacejka tire model which was my core part of work [10].

3.2 Implementation

All the equations in section 1.2 are converted into matlab-simulink environment where a user defined block which was defined in the latter section where the computer language description was done using the matlab and C language to incorporate the factors, constants and various parameters that govern the force generation in the tire using the pacejka tire model.

As described above the S-function in matlab consists a set of S-function call back methods that enables the required tasks at each simulation stage[9]. During the simulation of a model, in case of this project the model under consideration is pacejka tire model, at each simulation stage the simulink engine calls the relevant approaches for every S-function block in the model.

The first step in the implementation were the initialization of the user scaling factors of the pacejka model which are directly related to the factors B, C, D and E. Hence the user scaling factors were extracted from [11] where the data were given different vehicle configurations, after careful inspection user scaling factors were acquired from the following vehicle specifications.

- 1.Tire relaxation length = 0.1m.
- 2.Maximum torque generated = 255 Nm @ 2500 rpm.
- 3.Tire specification = 195 R 14.

Which can be interpreted as a tire with 195mm section width, a radially constructed tire(R) and having a diameter of 14 inches.

After the scaling factors were scripted in a matlab file the S-function of pacejka tire model was continued by deciding the number and dimensions of input and output ports the S-function can have among this allocation the input ports comprised of following parameters,

1. Velocity of the wheel at the contact point.
2. Slip angle.
3. Peripheral speed .
4. Vertical force at the contact point.
5. Velocity of the vehicle.
6. Moment values from the vehicle model.

The output ports from the S-function had the force generated from each tire which was the core essence of this project to enumerate the forces that were developed in pacejka tire model whose flow chart of computation is described in figure 4 and compare it with the already existing burckhardt tire model on the same vehicle dynamics that the burckhardt was running previously.

Below diagram describes the summary of above description on how the pacejka tire model was implemented on simulink platform.

Overall the S-function refers to the pacejka tire model equations which uses the parameters from the vehicle simulation model such as the velocity and the user scaling factors from the selected tire as the input. The magic formula algorithm itself consists of subroutines for computing generated forces and moments which depend on the parameters that govern characteristics

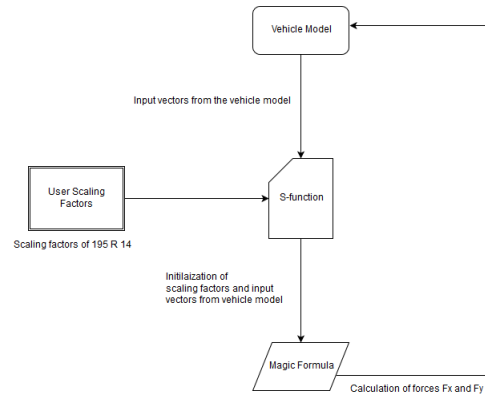


Figure 4: Flowchart of pacejka model through S-function

of the curves [3]. At first, lateral force, longitudinal force are computed for pure longitudinal and side slip conditions, respectively. These quantities are then utilized for calculating the actual forces and moments due to combined longitudinal and side slip.

After the computation of longitudinal and lateral forces by adopting the theory proposed by pacejka the simulations were executed, after which a run of parallel simulations were done using the old tire model that is burckhardt tire model and the new computed tire model pacejka and the force generation were observed both the plots matched and in the further sections both will be compared to scenarios like time for computation and in case of ABS scenario. In appendix A the S-function, system and subsystem model codes and models are published. It's worth noting that the vehicle and burckhardt model were given by my supervisor which was previous work in the department and upon his request, the language of the code was not altered however only the tire model part were written by my work.

4 Simulation Results

4.1 Comparing The Results of Burckhardt And Pacejka Model

In this study project we have proposed a simulation study of pacejka tire model whose nature depends on the coefficients B, C, D, E which in turn mainly depends on the F_z and camber angle, in this simulation camber angle equated to zero and the normal force on the tire would be F_z compared with the burckhardt tire model which has a relatively simpler model when compared with pacejka is given by the equation,

$$F(\lambda) = c_1(1 - \exp(c_2\lambda)) - (c_3\lambda) \quad (9)$$

where the burckhardt model solely depends on factor c which is a parameter of tire obtained by curve fitting data by adapting to the different values of c, different profiles of track profiles can be modeled [12].

Computational time

The time taken for completing a distance of 943m in the simulation frame and it was found that pacejka is faster when compared to burckhardt tire which is proven by the details given in the simulink report generator of both the tire models as illustrated in table 1 and table 2. Since the pacejka model doesn't rely on a physical structure but it involves several parameters which can be tuned in such a way that we obtain the required traction forces with regard to different driving profiles. However this model has a tremendous analogy

with experimental data, obtained under particular steady-state conditions, where the velocity values are assumed to be constant pacejka could be used in different driving conditions by tuning or altering the parameters that are described in the above sections.

Table 1: Simulink report generated for pacejka tire model

Parameters	Value
Total recorded time	1383 s
Number of block methods	922
Number of internal methods	8
Number of model methods	11
Number of nonvirtual subsystem methods	18
Clock Precision	0.00000006
Clock speed	1701 MHz

Table 2: Simulink report generated for burckhardt tire model

Parameters	Value
Total recorded time	1776 s
Number of block methods	922
Number of internal methods	8
Number of model methods	11
Number of nonvirtual subsystem methods	18
Clock Precision	0.00000006
Clock speed	1701 MHz

ABS Scenario

With regard to the ABS control scheme the pacejka has a better solution as the tire model requires F_z for its computation and mainly the ABS problem addressing occur in control of the wheel slip where the pacejka outperforms as the burckhardt doesn't allow a change in the rolling conditions it which might make it a bit difficult to address the ABS control with the burckhardt tire model, whereas pacejka assures the accuracy over the complete range of slip values from 0 to -1 which controls the angular velocity of the wheel and thus the vehicle speed as the vehicle speed cannot be controlled directly. Hence pacejka offers a better solution when compared to burckhardt in terms of addressing the control schemes of ABS.

4.2 Plots of generated forces

In this section, the simulation plots are shown, however before enlisting the force plots a short view of the track is shown in figure 5 for which forces are plotted is described below sections.

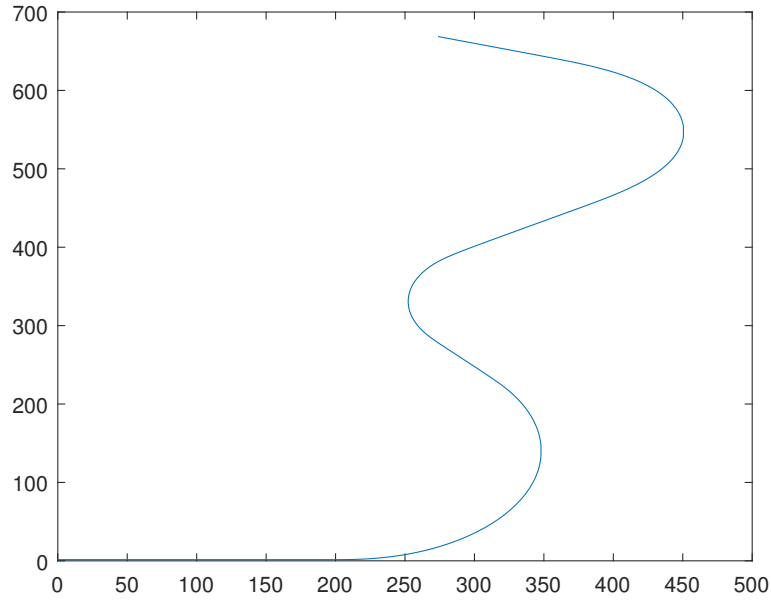
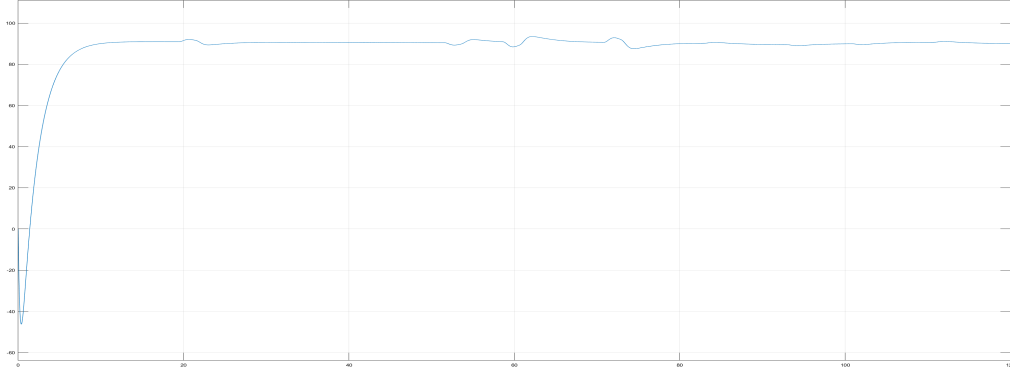


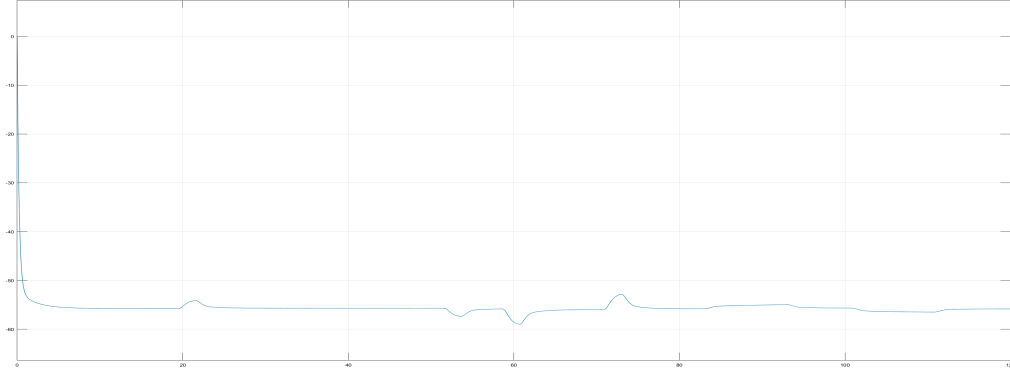
Figure 5: Adopted track

4.2.1 Longitudinal Force

Longitudinal force depends on the shape factors B, C, D and slip values σ . The longitudinal force is of prime importance as it will be used to compute the lateral force in the tire model. The force plots developed from this model is as shown in figure 6.



(a) Front tire F_y plot



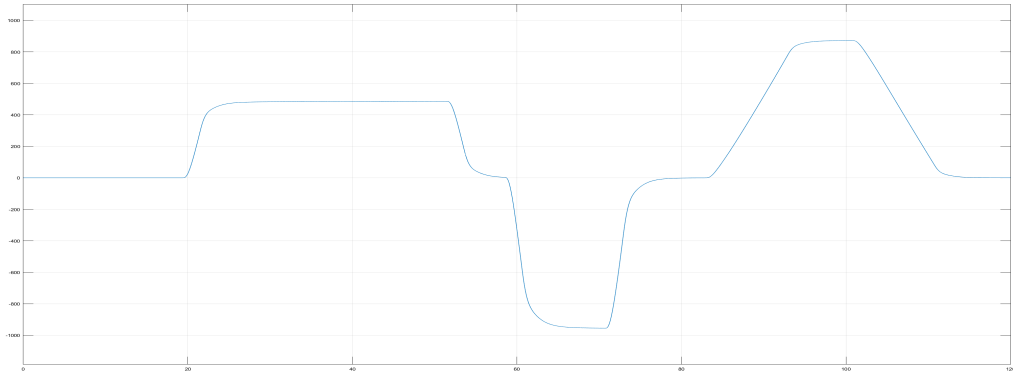
(b) Back tire F_y plot

Figure 6: Longitudinal force of the tires

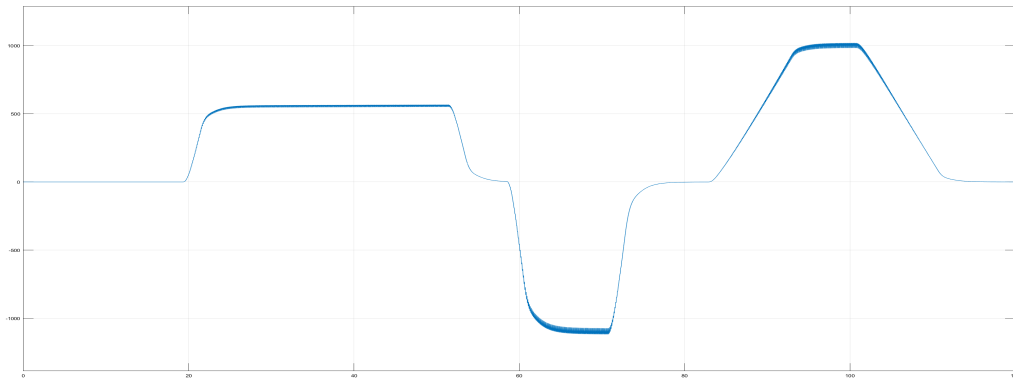
4.2.2 Lateral Force

The given tire model determines the lateral force and related torque which is dependent slip angle, longitudinal slip and force along the x-axis, the tire model properties bank on F_z of the tire usually mentioned in kN units. The lateral force which is generated due to lateral slip which usually exists during cornering movement where the contact of the tire slips laterally which causes rolling motion which would be no more in the plane where the tire would be rotating. The lateral force generated by the tire model is as shown in figure 7.

Implementation of the Pacejka Tire Model for a vehicle simulation in MATLAB



(a) Front tire F_y plot



(b) Back tire F_y plot

Figure 7: Lateral force of the tires

5 Conclusion

The primary target of this study project was to implement the pacejka tire model [3] to a vehicle simulation which already was existing with burckhardt tire model and try to compare the results of the simulation based on few scenarios, the particular reason which interested the adoption of pacejka model into my study was the computational efficiency that pacejka possesses even though its a complicated but with right adaption of scaling factors this study has proven that it can be implemented to a vehicle model to support this fact, [13] conducted a study on different types of tire model which were utilized to study the primary effects of lateral force in dependency with slip ratio and slip angle, they examined different models such as pacejka, segel, dugoff and proposed polynomial, whose computing time and accuracy were investigated. For the comparison, they computed longitudinal, lateral and aligned moment of each model. The result had its own merits and demerits, whilst pacejka scored the highest in the accuracy division which was my sole intention of mentioning this study.

Having said the advantages pacejka tire model does possess some of the disadvantages which include its capability in the narration of temperature effects, non-flat road conditions and off-road usage of the tire.

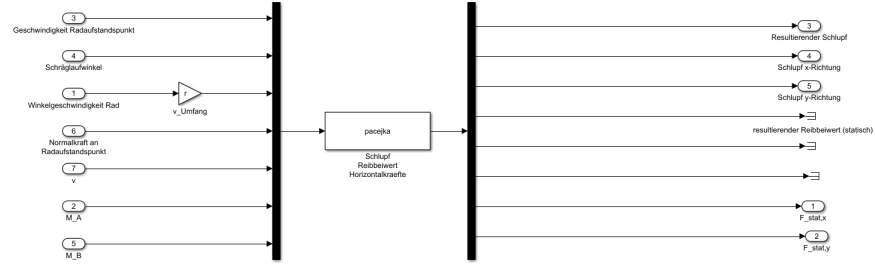
Further on my own interest post this study I have been investigating on synergy of the pacejka tire model with an electric vehicle whose torque generation can be monitored with suitable application of thoughts and ideas is an ongoing research but this study has enabled me to look into the possibilities of implementing a tire model and its emphasis towards vehicle dynamics.

References

- [1] Hans B Pacejka and Egbert Bakker. The magic formula tyre model. *Vehicle system dynamics*, 21(S1):1–18, 1992.
- [2] Andrius Ružinskas and Henrikas Sivilevičius. Magic formula tyre model application for a tyre-ice interaction. *Procedia Engineering*, 187:335–341, 2017.
- [3] Hans B. Pacejka. Chapter 4 - semi-empirical tire models. In Hans B. Pacejka, editor, *Tire and Vehicle Dynamics (Third Edition)*, pages 149 – 209. Butterworth-Heinemann, Oxford, third edition edition, 2012.
- [4] H. B. PACEJKA and I. J. M. BESSELINK. Magic formula tyre model with transient properties. *Vehicle System Dynamics*, 27(sup001):234–249, 1997.
- [5] Hans B. Pacejka and Egbert Bakker. The magic formula tyre model. *Vehicle System Dynamics*, 21(sup001):1–18, 1992.
- [6] Wolfgang Hirschberg, Georg Rill, and Heinz Weinfurter. User-appropriate tyre-modelling for vehicle dynamics in standard and limit situations. *Vehicle System Dynamics*, 38(2):103–125, 2002.
- [7] Markus Schmid. Tire modeling for multibody dynamics applications. *University of Wisconsin-Madison*, 2011.
- [8] F. Bucchi and F. Frendo. A new formulation of the understeer coefficient to relate yaw torque and vehicle handling. *Vehicle System Dynamics*, 54(6):831–847, 2016.

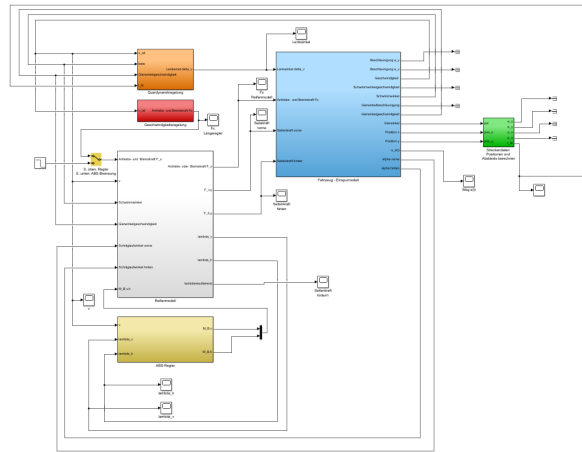
- [9] MATLAB MathWorks. the mathworks. *Inc., Natick, MA*, 1992.
- [10] Steven T Karris. *Introduction to Simulink with engineering applications*. Orchard Publications, 2006.
- [11] Giancarlo Genta and Lorenzo Morello. *The automotive chassis*, volume 1. Springer, 2009.
- [12] M Burckhardt. Fahrwerktechnik: Radschlupf-regelsysteme, vogel-fachbuch, 1, 1993.
- [13] AY Maalej, DA Guenther, and JR Ellis. Experimental development of tyre force and moment models. *International Journal of Vehicle Design*, 10(1):34–51, 1989.

A Simulink Model and Matlab Code

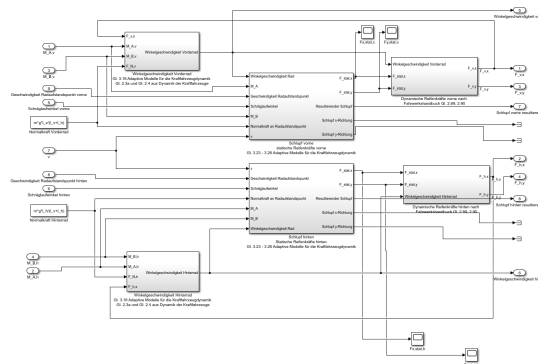


Pacejka model implemented in a S-function

Implementation of the Pacejka Tire Model for a vehicle simulation in MATLAB



Simulation file of the vehicle model



Simulation file of tire model

```

function [sys,x0,str,ts] = pacejka(t,x,u,flag)
%SFUNTMPL General M-file S-function template
%   With M-file S-functions, you can define you own ordinary differential
%   equations (ODEs), discrete system equations, and/or just about
%   any type of algorithm to be used within a Simulink block diagram.
%
%   The general form of an M-File S-function syntax is:
%       [SYS,X0,STR,TS] = SFUNC(T,X,U,FLAG,P1,...,Pn)
%
%   What is returned by SFUNC at a given point in time, T, depends on the
%   value of the FLAG, the current state vector, X, and the current
%   input vector, U.
%
%   FLAG      RESULT          DESCRIPTION
%   -----
%   0          [SIZES,X0,STR,TS] Initialization, return system sizes in SYS,
%                                     initial state in X0, state ordering strings
%                                     in STR, and sample times in TS.
%   1          DX              Return continuous state derivatives in SYS.
%   2          DS              Update discrete states SYS = X(n+1)
%   3          Y               Return outputs in SYS.
%   4          TNEXT           Return next time hit for variable step sample
%                                     time in SYS.
%   5                                     Reserved for future (root finding).
%   9          []              Termination, perform any cleanup SYS=[].
%
%
%   The state vectors, X and X0 consists of continuous states followed
%   by discrete states.
%
%   Optional parameters, P1,...,Pn can be provided to the S-function and
%   used during any FLAG operation.
%
%   When SFUNC is called with FLAG = 0, the following information
%   should be returned:
%
%       SYS(1) = Number of continuous states.
%       SYS(2) = Number of discrete states.
%       SYS(3) = Number of outputs.
%       SYS(4) = Number of inputs.
%
%       Any of the first four elements in SYS can be specified
%       as -1 indicating that they are dynamically sized. The
%       actual length for all other flags will be equal to the
%       length of the input, U.
%
%       SYS(5) = Reserved for root finding. Must be zero.
%       SYS(6) = Direct feedthrough flag (1=yes, 0=no). The s-function
%       has direct feedthrough if U is used during the FLAG=3
%       call. Setting this to 0 is akin to making a promise that
%       U will not be used during FLAG=3. If you break the promise
%       then unpredictable results will occur.
%
%       SYS(7) = Number of sample times. This is the number of rows in TS.
%
%
%

```

```

%      X0      = Initial state conditions or [] if no states.
%
%      STR      = State ordering strings which is generally specified as [].
%
%      TS       = An m-by-2 matrix containing the sample time
%                 (period, offset) information. Where m = number of sample
%                 times. The ordering of the sample times must be:
%
%                 TS = [0      0,      : Continuous sample time.
%                       0      1,      : Continuous, but fixed in minor step
%                                   sample time.
%                       PERIOD OFFSET, : Discrete sample time where
%                                   PERIOD > 0 & OFFSET < PERIOD.
%                       -2      0];    : Variable step discrete sample time
%                                   where FLAG=4 is used to get time of
%                                   next hit.
%
%
%      There can be more than one sample time providing
%      they are ordered such that they are monotonically
%      increasing. Only the needed sample times should be
%      specified in TS. When specifying than one
%      sample time, you must check for sample hits explicitly by
%      seeing if
%
%          abs(round((T-OFFSET)/PERIOD) - (T-OFFSET)/PERIOD)
%      is within a specified tolerance, generally 1e-8. This
%      tolerance is dependent upon your model's sampling times
%      and simulation time.
%
%
%      You can also specify that the sample time of the S-function
%      is inherited from the driving block. For functions which
%      change during minor steps, this is done by
%      specifying SYS(7) = 1 and TS = [-1 0]. For functions which
%      are held during minor steps, this is done by specifying
%      SYS(7) = 1 and TS = [-1 1].
%
%      Copyright 1990-2002 The MathWorks, Inc.
%      $Revision: 1.18 $
%
%
% The following outlines the general structure of an S-function.
%
switch flag,

    %%%%%%%%%%%%%%%%%%%%%%%%%%
    % Initialization %
    %%%%%%%%%%%%%%%%%%%%%%%%%%
    case 0,
        [sys,x0,str,ts]=mdlInitializeSizes;

    %%%%%%%%%%%%%%%%%%%%%%%%%%
    % Derivatives %
    %%%%%%%%%%%%%%%%%%%%%%%%%%
    case 1,
        sys=mdlDerivatives(t,x,u);

```



```

%%%%%%%%%%
% Update %
%%%%%%%%%%
case 2,
    sys=mdlUpdate(t,x,u);

%%%%%%%%%%
% Outputs %
%%%%%%%%%%
case 3,
    sys=mdlOutputs(t,x,u);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% GetTimeOfNextVarHit %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
case 4,
    sys=mdlGetTimeOfNextVarHit(t,x,u);

%%%%%%%%%%
% Terminate %
%%%%%%%%%%
case 9,
    sys=mdlTerminate(t,x,u);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Unexpected flags %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
otherwise
    error(['Unhandled flag = ',num2str(flag)]);

end

% end sfuntmpl

%
%=====
% mdlInitializeSizes
% Return the sizes, initial conditions, and sample times for the S-function.
%=====
%
function [sys,x0,str,ts]=mdlInitializeSizes

%
% call simsizes for a sizes structure, fill it in and convert it to a
% sizes array.
%
% Note that in this example, the values are hard coded. This is not a
% recommended practice as the characteristics of the block are typically
% defined by the S-function parameters.
%
sizes = simsizes;

sizes.NumContStates = 0;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 8; % Anzahl der Ausgaenge

```

```
sizes.NumInputs      = 7;    % Anzahl der Eingänge
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;    % at least one sample time is needed

sys = simsizes(sizes);

%
% initialize the initial conditions
%
x0 = [];

%
% str is always an empty matrix
%
str = [];

%
% initialize the array of sample times
%
ts = [0 0];

% end mdlInitializeSizes

%
%=====
% mdlDerivatives
% Return the derivatives for the continuous states.
%=====
%
function sys=mdlDerivatives(~,~,~)

sys = [];

% end mdlDerivatives

%
%=====
% mdlUpdate
% Handle discrete state updates, sample time hits, and major time step
% requirements.
%=====
%
function sys=mdlUpdate(~,~,~)

sys = [];

% end mdlUpdate

%
%=====
% mdlOutputs
% Return the block outputs.
%=====
%
function sys=mdlOutputs(t,x,u)
```

```

paramter;
global parameters;

% Zuweisungen der Eingaenge
v_Ri = u(1);      % Geschwindigkeit des Radaufstandspunktes
alpha_R = u(2);   % Schri%glaufwinkel
v_umfang = u(3);  % Umfangsgeschwindigkeit v_umfang = omega_Ri * r_R
F_NR = u(4);      % Normalkraft an der Kontaktflaeche
v_F = u(5);       % Fahrzeuggeschwindigkeit
M_A = u(6);
M_B = u(7);

if ((M_A + M_B) > 0) % Antreiben
    Indikator = 1;
else % Bremsen
    Indikator = 0;
end

% if(Indikator == 0) % Bremsfall
if ((abs(v_Ri) > abs(v_umfang*cos(alpha_R))) || Indikator == 0) % Bremsfall
    if (v_Ri ~= 0) % Radgeschwindigkeit ungleich Null
        lambda_L = (v_umfang*cos(alpha_R)-v_Ri)/(v_Ri);
        lambda_S = (v_umfang*sin(alpha_R))/(v_Ri);
    else % if (v_Ri == 0)
        % Radgeschwindigkeit gleich Null (v_Ri == 0)
        % Division durch Null abfangen

        if (v_umfang ~= 0)
            % v_Ri == 0 und v_R ~= 0
            lambda_L = 1 * cos(alpha_R);
            lambda_S = 1 * sin(alpha_R);
        else % if (v_R == 0)
            % v_Ri == 0 und v_R == 0
            lambda_L = 0;
            lambda_S = 0;
        end % if (v_R == 0)
    end % if (v_Ri == 0)

else % if (abs(v_Ri) < abs(v_R*cos(alpha_R)))
    % Antriebsfall und nicht angetriebenes oder
    % gebremstes Rad (abs(v_Ri) = abs(v_R*cos(alpha_R)))
    if (v_umfang*cos(alpha_R) ~= 0)
        % Umfangsgeschwindigkeit ungleich Null
        lambda_L = (v_umfang*cos(alpha_R)-v_Ri)/(v_umfang*cos(alpha_R));
        lambda_S = sin(alpha_R);
    else % if (v_R*cos(alpha_R) == 0)
        % Umfangsgeschwindigkeit gleich Null
        % Division durch Null abfangen

        if (v_Ri == 0)
            % v_Ri == 0 und v_R*cos(alpha_R) == 0
            lambda_L = 1 * cos(alpha_R);
            lambda_S = 1 * sin(alpha_R);
        else % if (v_Ri ~= 0)
            % v_Ri == 0 und v_R*cos(alpha_R) == 0
            lambda_L = 0;
            lambda_S = 0;
        end
    end
end

```

```

        end % if (v_Ri ~= 0)

    end % if (v_R*cos(alpha_R) ~= 0)

end % if (abs(v_Ri) > abs(v_R*cos(alpha_R)))

lambda_RES = sqrt((lambda_L)^2+(lambda_S)^2);

C = parameters.b_mtm(1);

mu_L = parameters.b_mtm(2)*F_NR/1000+parameters.b_mtm(3);

D = mu_L*F_NR/1000;

B = (parameters.b_mtm(4)*(F_NR/1000)^2+parameters.b_mtm(5)*F_NR/1000)*exp(-parameters.b_mtm(6)*F_NR/1000)/C/D;

E = parameters.b_mtm(7)*(F_NR/1000)^2+parameters.b_mtm(8)*F_NR/1000+parameters.b_mtm(9);

Sh = parameters.b_mtm(10)*F_NR/1000+parameters.b_mtm(11);

Gx_alpha = cos(C*atan((B*parameters.b_mtm(12)+alpha_R-E*(B*alpha_R-atan(B*alpha_R)))));

Gx = cos(C*atan(B*alpha_R- E*(B*alpha_R-atan(B*alpha_R))))/(Gx_alpha);

F_XR = D*sin(C*atan(B*(1-E)*(lambda_L)+E*atan(B*(lambda_L))));

F_XR = F_X/Gx;

if F_NR==0

    F_XR=0;
end

sideslip=tan(alpha_R*180/pi); % tire sideslip angle input in radians

camber = 0;

mu_S = parameters.a_mtm(2)*F_NR/1000+parameters.a_mtm(3);

D = mu_S*F_NR/1000;

E = parameters.a_mtm(7)*F_NR/1000+parameters.a_mtm(8);

B = parameters.a_mtm(4)*sin(2*atan(F_NR/1000/parameters.a_mtm(5)))*(1-parameters.a_mtm(6)*abs(camber))/C/D;

Sh = parameters.a_mtm(9)*camber+parameters.a_mtm(10)*F_NR/1000+parameters.a_mtm(11);

Sv =(parameters.a_mtm(12)*F_NR/1000+parameters.a_mtm(13))

```

```

*camber*F_NR/1000+parameters.a_mtm(14)*F_NR/1000+parameters.a_mtm(15);

F_YR=D*sin(C*atan(B*(1-E)*(sideslip)+E*atan(B*(sideslip))));

if F_NR==0
    F_YR=0;
end

mu_RES = sqrt (mu_L^2 + mu_S^2); % final formula for mu_RES

sys = [ lambda_RES lambda_L lambda_S mu_RES mu_L mu_S F_XR F_YR ];
% end mdlOutputs

%
%=====
% mdlGetTimeOfNextVarHit
% Return the time of the next hit for this block. Note that the result is
% absolute time. Note that this function is only used when you specify a
% variable discrete-time sample time [-2 0] in the sample time array in
% mdlInitializeSizes.
%=====
%
function sys=mdlGetTimeOfNextVarHit(t,~,~)

sampleTime = 1; % Example, set the next hit to be one second later.
sys = t + sampleTime;

% end mdlGetTimeOfNextVarHit

%
%=====
% mdlTerminate
% Perform any end of simulation tasks.
%=====
%
function sys = mdlTerminate(~,~,~)

sys = [];

% end mdlTerminate

```