
IPG Tutorial and Quickstart Guide

# FormulaCarMaker 12

SOLUTIONS FOR VIRTUAL TEST DRIVING

# Table of Contents

# Abstract

Where to start? A question many motorsport teams ask themselves before facing a new season - or their first. With IPG Automotive GmbH as a new team sponsor you have received two free licenses for IPGKinematics and CarMaker.

But having first installed the new programs, many don't know how to get started. Questions emerge like "How does this software work?", "What program am I supposed to start with?" or "What are the benefits of using IPGKinematics and CarMaker in the development process of my Formula Student car?".

Fortunately, IPG Automotive GmbH offers further technical support to all registered teams, such as: a User's Guide, Manuals, Formula CarMaker workshops, video tutorials and dedicated email-support at:

**FormulaCarMaker@ipg-automotive.com**

to answer any questions concerning CarMaker, the CarMaker Toolset and IPGKinematics.

This document will show you step by step how to integrate IPGKinematics and CarMaker efficiently into the design and optimization processes of your Formula Student car to help achieve an improved competition performance. But remember, this document is only supplementary to the handbooks. The CarMaker User's Guide and Reference Manual (available in the Help menu of the CarMaker main window) should always be used alongside this document.

The general approach outlined is based upon several theses and internship projects from different Formula SAE teams completed using IPG.

The current Formula CarMaker race car model ("FSC_RaceCar") is primarily based upon a Formula Student Combustion car from the 2015 season. The example car, as well as the featured test runs, are intended to provide a basis for your simulation. This document will guide you through adapting the existing model as to best represent your car - allowing you to start simulating as soon as possible. Furthermore, certain test procedures and their analysis are explained to help validate the model and aid the optimization of your Formula Student racing car.

Chapter 1
# Installation

## 1.1 Installation and Licensing

### 1.1.1 How to install CarMaker and other IPG software?

First, you must register for the current Formula CarMaker season via the following link:

https://ipg-automotive.com/company/research-teaching/formula-carmaker-program/

After successfully registering for a Formula CarMaker sponsorship you will receive a login for the Client Area. Download the latest version of CarMaker/Office and IPGKinematics on the same webpage you also downloaded the Formula CarMaker package that contains this tutorial. To be able to install CarMaker on your PC you must possess administrative rights. The .zip archive contains an Installation Guide which provides helpful information about the installation process.

- **1. Please read the enclosed license agreement carefully. By installing the software you accept the terms of the license agreement.**
- **2. Log in as a user with administrative privileges and extract the downloaded installation package (.zip archive). Open the executable file "ipg-install.exe" via double-click.**
- **3. Follow the instructions of IPGInstall. The destination directory you have chosen at the installation (recommended: C:\IPG) will be further referred to as "<Installation_Directory>".**
- **4. You will find a Start Menu icon under *Start* > *Programs* > *IPG* > *<Current_Version>*.**

You should choose C:\IPG as the installation library. It is not compulsory but recommended as otherwise changes must be made to the data sets of IPGKinematics. Further information can be found on the IPG website as well as within the Installation Guide downloaded amongst the installation files. If this information is not sufficient please feel free to contact the Formula CarMaker team using **FormulaCarMaker@ipg-automotive.com**

## The License File

If a valid license file is not detected, IPG products such as CarMaker will not be able to run. For simplicity this document uses *CarMaker* as an umbrella term for several IPG products.

As part of the sponsorship, each team receives a license file with two registered nodes. Each node is created using specific PC information and thus is bound to a sole terminal. Therefore, ensure that the workstation you choose is suitable for use with CarMaker as you will not be able to change it once you have received the license file.

Each license is valid until the end of the current season (October 31st). After expiry each team may register again to receive a new license file. Please contact the Formula CarMaker support team via **FormulaCarMaker@ipg-automotive.com** for questions regarding the license.

The license file must be placed within the folder *C:/ipg/etc* and renamed to “Licenses” or “Licenses.dat” otherwise CarMaker will display the error “License file not found” when it is started.

## The Cockpit Package

The FTP link you received also contains an installation package of our tool “Cockpit Package standard”. It is an interface to connect CarMaker with a game controller such as a Logitech (G27/G29) or Thrustmaster (RS500) steering wheel. Please find further information about this tool in the Cockpit Package Reference Manual which is part of the zip package. To request a license for the interface please contact the Formula CarMaker team via **FormulaCarMaker@ipg-automotive.com**

Chapter 2

# Formula CarMaker and the IPG Toolset

## 2.1 Introduction

Chapter 2 will provide a brief overview as to the features of each program, followed by a description of Formula CarMaker's structure and functionality in Chapter 3. This includes the creation of a Formula CarMaker project and an example which demonstrates how to build and interpret a Formula CarMaker TestRun.

Completion of these chapters is recommended and allows the user to become familiar with the software and reduce the likelihood of errors when defining their own models. This section is followed by Chapter 4 which gives an introduction to IPGKinematics. The interaction of both tools is demonstrated by importing kinematic and compliance characteristics generated using IPGKinematics into CarMaker.

Please remember that you always have access to the User's Guide and the Reference Manual which contain useful information about the two programs - this document is only intended to be a supplement to these manuals. The reader will be reminded continually through references and cross-references to utilize these resources. Many questions and mistakes can be avoided or solved through the use of these documents.

These can be found via *Help > User's Guide* or *Help > Reference Manual*. The Help menu contains further documentation regarding the main IPG tools e.g. IPGMovie or IPGControl.

### 2.1.1 CarMaker

Before starting to simulate with CarMaker, the general structure of CarMaker, the meaning of the virtual vehicle environment (VVE) and the CarMaker Interface Toolbox (CIT) should be explained.

CarMaker is a tool that is used to simulate the vehicle dynamics of four-wheeled cars. It is one of IPG's products and was initially designed for the simulation of passenger cars. By means of mathematical models a Virtual Vehicle Environment (VVE) is created and simulates vehicle, driver and road including wind, obstacles, traffic signs etc. These constitute all the parts necessary to evaluate a controller or to test the dynamics of a vehicle or a vehicle subsystem.

Using CarMaker/HIL, which has identical core functionality to that of CarMaker/Office, real hardware such as ECUs can be tested within a virtual vehicle environment. (HIL = Hardware in the Loop)

The second part of the CarMaker system features all the tools that are used to manage the VVE. These tools allow the user to: start and stop a simulation, select vehicle parameter data, define a vehicle maneuver, display results, show progress graphically or as an animation, send and receive messages from the VVE, etc. These tools are referred to as the CarMaker Interface ToolBox (CIT).

The tools can be classified as:

- *Control and Direct Access Tools* - control the actions performed by the simulation (e.g. start, stop, etc.) and allows certain parts of the simulation to be directly controlled by the user (e.g. direct variable access).
- *Parameterization Tools* - serve to specify the parameters that will be used in the VVE.
- *Analysis and Visualization Tools* - allow the data produced to be viewed and analyzed either during or after a simulation.
  - IPGMovie for visualization of the VVE
  - IPGControl to plot data for analysis
  - IPGInstruments to provide information regarding the state of the car

Control and parameterization is done via the GUI (= Graphical User Interface). The first step when using the GUI is to edit parameter fields in order to define your model. A simulation can then be performed as to predict the behavior of the car and produce results for examination.

Regularly refer to both the User's Guide and the Reference Manual to help prevent errors from occurring. Both are found via the Help menu in the Main GUI of CarMaker.

### 2.1.2 IPGKinematics

*IPGKinematics* is a 'calculator' that describes the spatial movement of the wheels in space through populating kinematics tables. It calculates the corresponding forces necessary to trigger a movement and thus can determine the kinematics, steering kinematics and compliance for a variety of suspension types. It can be likened to a virtual axle kinematics test bench.

Tools within IPGKinematics allow the user to generate diagrams and plot various curves for data processing and analysis. Another option allows editing of the datafile so that it can be exported to several other vehicle simulation tools, such as CarMaker.

Control and parameterization are also performed via the IPGKinematics GUI. Firstly, simulation settings must be defined followed by the data required to parametrize your axle ('Input Data'). A simulation is then performed to calculate relative wheel positions and their associated forces. Finally, results can then be analyzed and exported.

To address any issues you might encounter, please note that the manuals contain a lot of helpful information about the tool.

Chapter 3

# Getting Started with Formula CarMaker

## 3.0.1 Opening CarMaker

After installing CarMaker on your computer and having received the license file, you can start working with the software.

To start CarMaker using Windows, press the *Start* Button and select *Programs* > *IPG* > *CarMaker*.

To start CarMaker using Linux, open a shell like xterm. Type in the command to start CarMaker in the background:

- CM &

The CarMaker main GUI pops up and automatically loads the project folder you were last working in before the program was closed. You can change the current project folder by selecting *File* > *Project Folder* and selecting the one you would like to work in.

**Exercise 1 – Step 1**

- Start Formula CarMaker:
  Follow the description above for your operating system.

## 3.0.2 CarMaker Main GUI

Once CarMaker is started, the 'main window' opens (Figure 3.1). This is the central interface where you can define and edit all input quantities, set the simulation speed and change whether the results are to be saved or not. This window shall be referred to as the main Graphical User Interface (Main GUI).

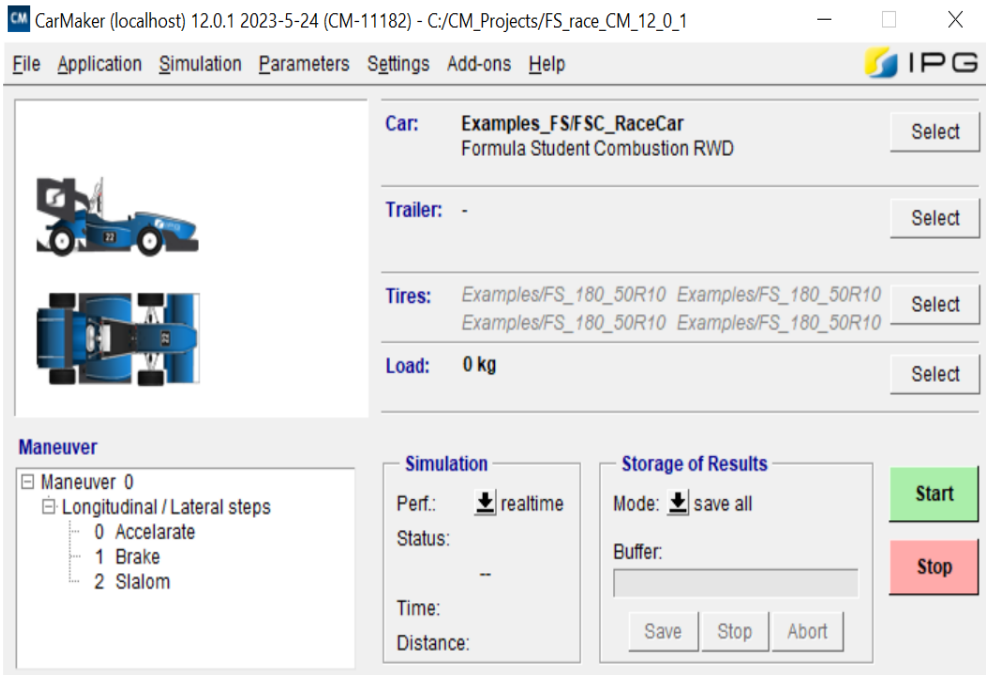



Figure 3.1: The CarMaker Main GUI with tutorial TestRun
Car: Chosen test car
Trailer: Chosen trailer model
Tires: Listed relative to position on vehicle. Gray indicates default of test car; black is assigned by the TestRun or user
Load: Assigned load in specified positions
Maneuver: List of maneuver steps in the TestRun

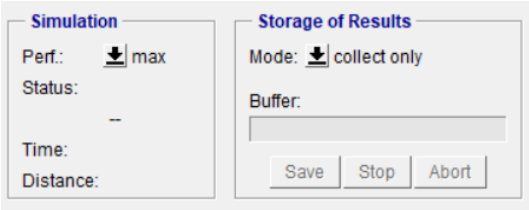



Figure 3.2: Setting the speed of simulation and storage behavior in the Main GUI

**Speed of Simulation**

There is lots of information and numerous buttons presented upon the Main GUI. One of which is the simulation speed (see the '*Simulation*' section of the interface). In general, any simulation speed may be chosen as it is a purely computational simulation (i.e. the computer can perform the calculations at any speed since there is no real component in the simulation loop). You can change this speed both before and during a simulation.

Note that this speed depends on the power of the processor and graphic card of your system (if IPGMovie is running, it may slow down calculations). This box also displays the duration of the simulation and the distance covered by the test vehicle.

**Storage of Results**

In most instances it should be sufficient to select 'collect only'. With this option results are only saved within temporary memory (you will have the option to save them to a file at the end of the simulation if you require the results again later). If one of the other storage options is selected, the results will be saved automatically - however, considering the numerous simulations that will be performed the consequent folder will quickly become enormous!

### File System

CarMaker operates within a project directory - all the folders used by CarMaker are stored according to a certain hierarchy that is identical for every project. Hence, CarMaker always knows where to find common files such as tire data, vehicle descriptions and so on. The data itself, of course, differs from project to project.

Thus, a *Project Folder* must be chosen after opening CarMaker as the software needs to be told which directory to work in. Of course, you can also create new project directories containing your own data, provided that the structure of the directory remains the same. Therefore, choose *File > Project Folder > Create Project*. In the popup window you can enter the path where the new folder should be created. This new project folder contains the folder hierarchy required by CarMaker to locate common files.

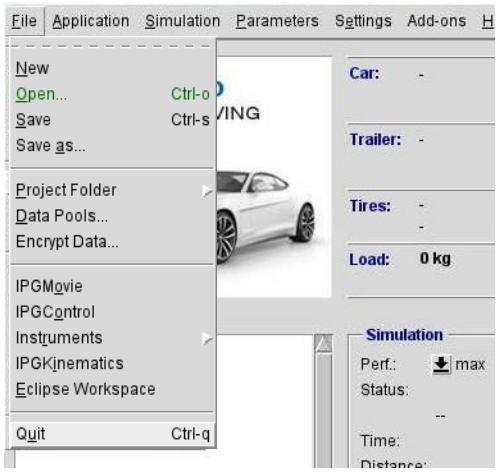



Figure 3.3: Selecting a project folder

We highly recommend **not** to create multiple project folders, but to create subfolders within a central project folder. For example, you could create one project folder per season and inside designate a subfolder to contain race circuits, and another to contain scenarios for braking and acceleration etc. Folders can be added via the *Select Project Folder* dialog or by using your File Explorer.

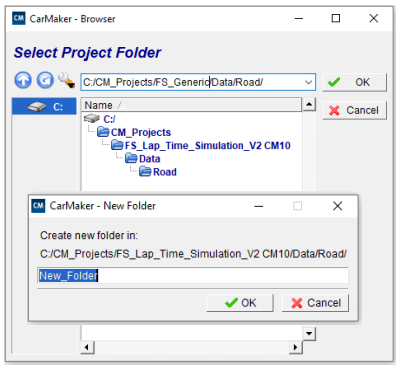



Figure 3.4: Creating subfolders inside the project folder

Please refer to Chapter 3.4, '*Directory Structure'* in the User's Guide (*Main GUI > Help > User's Guide*) for further information about the directory and shared data pools etc.

**Exercise 1 – Step 2**

- Create project folder:

  Download ‘FormulaCarMaker’ and extract the archive to your desired work directory.

  - In the main GUI choose *File* > *Project Folder* > *Select* and select the previously extracted project folder.

# 3.1 CarMaker TestRun

CarMaker is based on fixed models (vehicle, suspension, tires, etc.) whose properties (e.g. values for the mass of each body or spring stiffness) can be varied. This means that the number of bodies and the DOF between them are already defined and do not need to be constructed by the user. If the user wishes to extend a model, CM4SL is recommended. For further information on this topic, please refer to Quick Start Guide Chapter 7 ‘*CarMaker for Simulink’*.

Although the models are already defined, they still need to be parameterized according to their environment. For this, a ‘dataset’ is manually implemented or loaded for each of the models. Parameterizing includes choosing a vehicle, selecting or designing a road, defining a type of driver and specifying a maneuver. After all of these components are set, CarMaker has the information necessary to control the VVE and simulate.

All of these settings are stored in a file used by the VVE during simulation. Said file, which-can be saved, loaded or edited, is what we call the TestRun.

In summary: A TestRun represents a test scenario in which all parameters of the virtual environment (vehicle, driver, tires, etc.) are sufficiently defined. As a minimum requirement to be able to simulate, the following modules have to be parameterized within the TestRun:

- Vehicle and tires - definition of the vehicle data set used.
- Road - parameterization of the test track.
- Maneuver - mainly to specify the driver’s task.
- Driver - set driver behavior (defensive, normal, aggressive, or ‘racing driver’)

In CarMaker, the preparation for a vehicle test is very similar to how it would be conducted in a real world scenario.

1) Select a vehicle.
2) Choose or define a test track with obstacles and the desired conditions.
3) Specify the tires and loads.
4) Define the Maneuver(s).
5) Specify the type of driver or use a simple closed loop controller.
6) Save & Run the TestRun.
7) Analyze the TestRun results.

Next, we will create a TestRun to simulate and analyze the braking and slalom capabilities of our race car.

For further information regarding TestRuns in general, refer to the User’s Guide (*Main GUI* > *Help* > *User’s Guide*).

### 3.1.1 Vehicle Model

Instead of individually setting every parameter for each of the CarMaker sub-windows, we will load a predefined model. In this chapter the "FSC_RaceCar" vehicle data set will be used. This dataset represents a Formula Student racing car from the 2019 season.

To use a different vehicle, click on the Select button and choose another one out of the given product examples. Under *Project* > *Data* > *Vehicle* > *Examples_FS* you will find a variety of predefined race cars. In ‘*Preparing a Vehicle Dataset in CarMaker’*, pg. 75 it is explained how to create a bespoke vehicle model.

**Exercise 1 - Step 3**

- Open a new TestRun via *File* > *New*.
- Choose the test car:

  Press Select in the *Car* section of the Main GUI. Browse through the files and select: *Examples_FS* > “FSC_RaceCar”.

### 3.1.2 Tire Model

In this TestRun we will use a predefined tire dataset. In section 5.8, ‘*Creating a tire dataset using IPGTire’* it is explained how to create your own. The FSC_RaceCar contains a preconfigured dataset for tires which is a ‘global setting’. This means that unless you change it, the selected vehicle will always use the same tires. The gray font in the *Tire* section in Figure 3.5 indicates that a predefined data set from the vehicle is being used.

However, if you wish to use specific tires for a particular TestRun only, the best way to achieve this is via the Main GUI. Figure 3.5 shows the Main GUI and the Select button in the *Tire* section. By clicking this, you are able to choose a tire that will be used by all four wheels. To select a different tire for each wheel individually, click directly on the name of the tire on the Main GUI (in this case **"**FS_205_50_R13**"** for the rear wheels) and select accordingly.

By hovering the mouse over each the four tires shown in the *Tire* section of the Main GUI, a screen tip will appear describing which tire (front left, front right, rear left, rear right) will be modified.

In this example, the tire "FS_195_50R13" will be chosen for the front tires.

## Exercise 1 - Step 4

- Assign the Tire:
  - By using the *Select* button assign the **"**FS_195_50R13**"** tire from the available example models to the front  wheels.
  - Save your first TestRun as "myFirst_TestRun" by selecting *File* > *Save as* in the Main GUI.

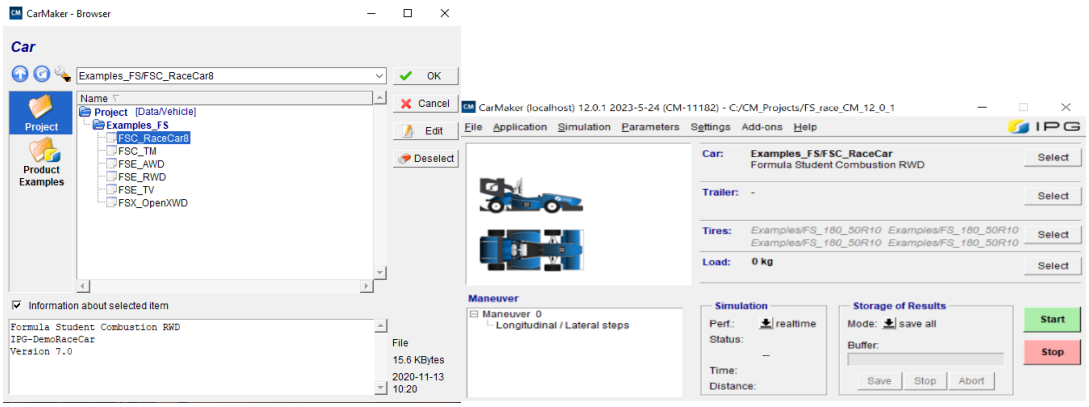



Figure 3.5: Left: Selecting a FSAE vehicle example
Right: Selecting a single tire in the Main GUI

### 3.1.3 Scenario Editor

In CarMaker, the *Scenario Editor* is the graphical user interface that enables the creation of highly complex road networks for vehicle and driving simulations. It is accessible via the CarMaker Main GUI by clicking the *Parameters* menu and selecting *Scenario/Road*.

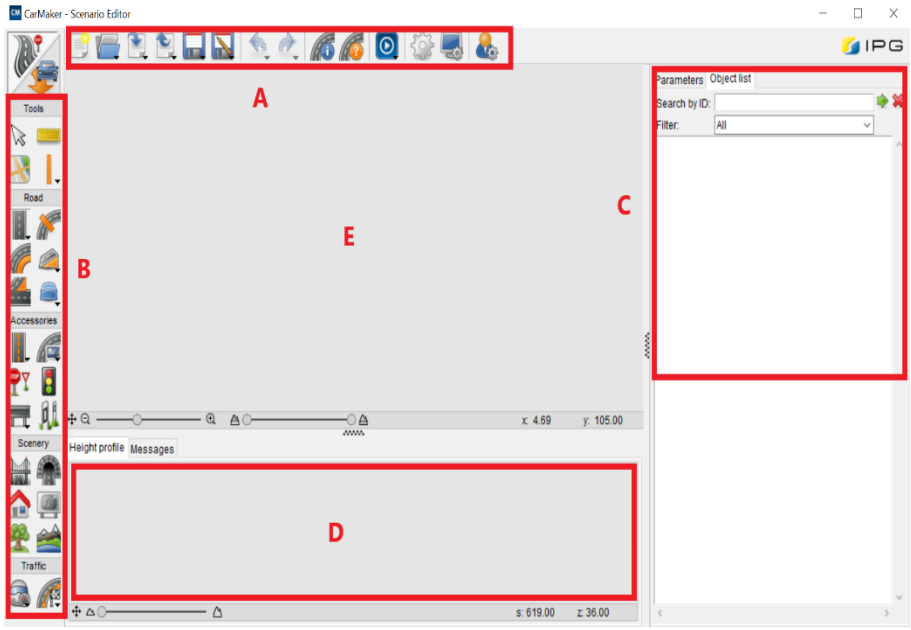



Figure 3.6: Scenario Editor GUI
A - Menu tab for accessing, viewing and saving information about the scenario
B – Road network and environment tool panel
C – The right sidebar lets you view and edit scenario data
D - General information about the state of Scenario Editor
E – Road network drawing area

**Exercise 2 – Step 1**

- Open the TestRun from the previous exercise "myFirst_TestRun"
- Open the Scenario Editor:
  In the Main GUI, click on *Parameters* > *Scenario / Road* or press Ctrl+r.

**Useful Scenario Editor Tips**

Before we start, please take note of the controls used in the Scenario Editor. The mouse buttons and *F9* key are used frequently, i.e. the *'Road Segement'* icon has a context menu which can be accessed by hovering over the icon and holding the left mouse button (LMB) or by pressing the right button (RMB) within the drawing field.

If you wish to place a straight or curved road segment, press *F9* on your keyboard to lock values to a constant increment size.

With the *Selection* tool active the LMB can also be used to select a drawn segment. Simply hover over the desired area and click the LMB. A context menu will appear where you can choose between different highlighted objects. These are sorted and tabbed according to their child-parent relationship.

A very useful tip is to become familiar with using the different manuals offered. The *Scenario Editor* has a comprehensive section (Chapter 5) in the User's Guide (*Main GUI > Help> User's Guide*). Please refer to this chapter for detailed Information about all the icons, tools, and procedures.

In the following chapters you will learn to build a course with two turns, a slalom section and braking test area. The course will give the opportunity to experience the *Scenario Editor, Test Manager* and use of *DVA* (direct variable access).

The decimal separator in all CarMaker and IPGKinematics applications is a dot (.) instead of a comma (,). If there is an error message related to inconsistent metrics, most of the time it is due to the wrong decimal separator.

Table 3.1: Basic terminology of IPGRoad

| Name | Description |
|---|---|
| Junction | Road element used to join different Links. |
| Link | A section of the road network, usually one road, and can be further specified using one or many Lane Segments. |
| Reference Line | The course of a Link is defined by its Reference Line. |
| Lane | Lateral extension of a Link. |
| Lane Section | Sections of a Link with a constant number of lanes. |
| Elevation Profile | Lateral, longitudinal and camber profile changes. |
| Route | Reference line defining the course of the vehicle travel on the road. |
| Path | Actual course that the vehicle travels on the road. |

Table 3.1 shows the terminology behind the theory of IPGRoad. For further information please look at Chapter 5.1, '*Theory of IPGRoad*' in the User's Guide (*Main GUI > Help > User's Guide*).

### Menu tab - Field A

The tab on the topmost portion of the Scenario Editor GUI offers general file operations such as loading and saving road files, and the ability to change various GUI settings. All icons are explained at Chapter 5.3, *'Building Roads with Scenario Editor'* in the User's Guide (*Main GUI > Help > User's Guide*).

- *Import road definition* - With this option, a Road network can be imported from either an external Road InfoFile or from a TestRun.
- *Undo/Redo* These two buttons allow you to undo/redo your last action.
- *Save TestRun* - This option has two functionalities. When a road network that is defined as part of a TestRun is being worked upon, the entire TestRun will be saved. However, if the road network is a Road InfoFile with the file format *.rd5 or *.road, then this button saves the changes to the Road InfoFile. Scenario Editor automatically identifies whether the road network is part of a TestRun or a separate Road InfoFile.
- *Save Road File As* - This option is a "Save as" function. It always saves the created road network as a CarMaker Road InfoFile.
- *3D Preview* - With this option, a 3D view of the road network can be seen in IPG-Movie.
- *Scenario Settings* - This option allows general settings for the scenario to be set. The parameters that can be defined are described in the User's Guide.

### Tool Panel - Field B

The tool panel contains a variety of tools which are arranged in the following sections:

- *Tools* - different attributes of the highlighted road section can be selected and measured.

- *Road* - create the basic road layout , modify lanes and add junctions or speed bumps .
- *Accessories* - road markings , guideposts , road paintings ,traffic signs , lights , barriers .
- *Scenery* - enables decoration of the simulation environment . All objects inserted here have no effect on the simulation itself, are ignored by IPGDriver and cannot be detected by Object Sensors.
- *Traffic* - The features of the *Traffic* panel affect simulation participants, such as the test car or traffic objects. A few markers affect solely the test car , while others also have an effect on traffic participants . The option *Route* is used to generate routes on a road network, defining the path on which the vehicle will drive.
- All icons are explained in the Chapter 5.3, *'Building Roads with Scenario Editor'* of the User's Guide (*Main GUI > Help > User's Guide*).

**Markers**

Another useful feature are *Markers*. Markers influence the behavior of IPGDriver, the driving maneuver and the vehicle.

**Driver Stop** This marker places a STOP sign on the road. IPGDriver will stop the vehicle at this sign.

**Pylon Alley** This type of marker inserts pylons on the road. These are always created in pairs. The Driver will always attempt to drive between them.

**DrivMan command** Allows a minimaneuver command to be defined using the road distance as a trigger. All available Minimaneuver Command Language commands can be used. A description of all commands can be found in Appendix D, '*Minimaneuver Command Language'* in the User's Guide (*Main GUI > Help > User's Guide*).

**DrivMan jump** This allows the user to create a specified maneuver with a start offset from the current road segment as the trigger point.

**Trigger point** Using the TriggerPoint marker, different actions can be executed at given positions upon the road.

**User - defined** User defined markers can be used to place a marker with a list of user defined parameters along the road.

## Building a Test track

The next seven exercise steps will create a closed circuit. The circuit will include a braking area, slalom course and a 75m acceleration track.

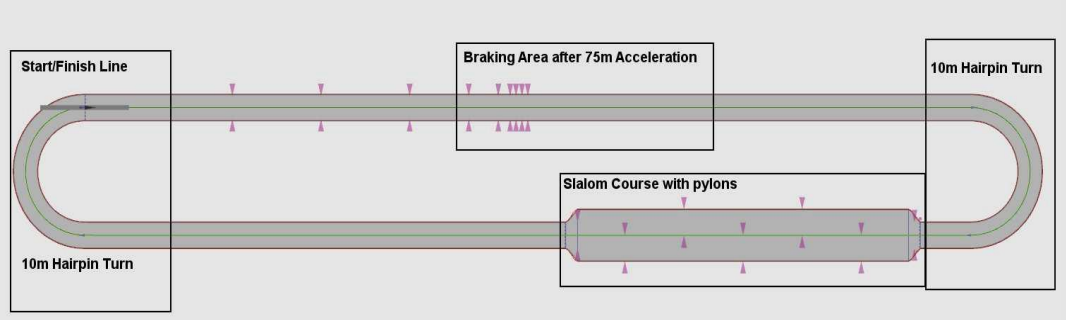



Figure 3.7: Closed test track with Braking Area and Slalom Course

### Lane Configuration

The Formula Student rules state that the circuit must be at least 3.5 m wide. We will edit the default lane configuration to fit this requirement, enabling us to become familiar with various icons in the process. This option permits different template lane configurations to be created and saved.

Before creating a new road segment, one of these lane configurations can be selected and the road segment will adopt this lane definition. However, selection of the lane type must be done prior to the generation of the road segment. If a lane configuration has not been selected, then the topmost lane definition is automatically nominated to create the road segment. The different lane configurations created by the user are saved as a part of the Project Folder. Hence, if a new Project Folder is created, the lane configurations will have to be transferred. Later, we'll learn how to change the width to single segments. In Figure 3.8 the lane configuration dialog is displayed.

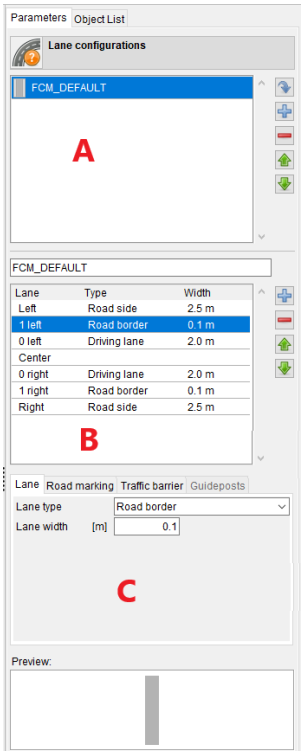



Figure 3.8: Road model to fit FSAE track rules
A: List of existing configurations
B: List of lanes in selected model
C: Preferences for each lane

### Exercise 2 - Step 2

- Add a new lane configuration:
  - Open the dialog via *Lane configuration* . Press the - button (Figure 3.8 - section A) on the right side to add a new configuration and rename it to "myFSA-Etrack".
  - Press the - button (Figure 3.8 - section B) to add seven lanes in total (three left, one center, three right).
  - Set the inner left and inner right driving lane to 2.0 m. Set both outer lanes as a road boarder with 0.1 m and set the both roadsides to 2.5 m.

- Remove the marking in the center of the road by selecting the '*Center*' lane and within the *Road marking* tab select "None" in the dropdown menu next to *Line Type* (Figure 3.8 - section C).

As mentioned, the new configuration will not overwrite preexisting segments, but will become the default configuration of every new segment within this project.

### Building Road Segments

The next step is to create the road itself. We'll learn how to create straights and corners and how to connect them. In Figure 3.9 shows the result of this step. When building roads with Scenario Editor there are a few details which you should remember:

1) The first point that is set is the origin of the global coordinate system.

2) If you did not draw the segment to the correct length, you can simply edit it afterwards by highlighting the drawn road segment with the *Selection* tool.

3) There are two possible methods of connecting road segments. The first one applies when you wish to create a new road segment connected to one drawn previously (e.g. a turn after a straight). Select the relevant icon (e.g. the *Turn* tool) from the tool panel and place the cursor over the middle line at the end of the first segment. An orange circle (see Figure 3.11) appears. Click using LMB and the start point of the new segment will be connected to the endpoint of the previous segment.

The second method is mainly used in creation of a closed loop circuit (i.e. a racetrack). The *Point List* tool can be used to connect two segments which are close together. This tool tries to find the shortest possible path between two segments.

### Exercise 2 - Step 3

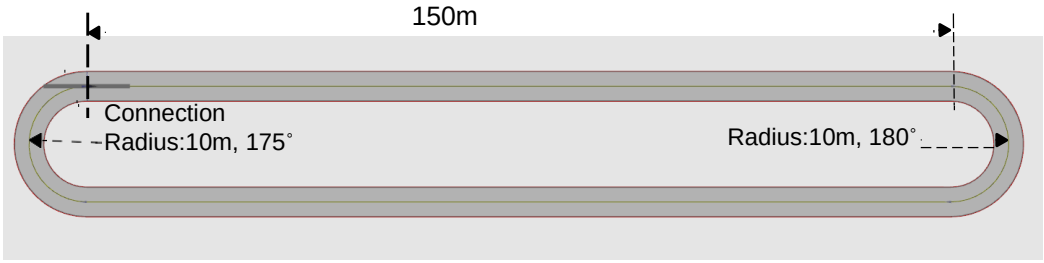



Figure 3.9: Simple closed loop circuit with 362m track length

- Place the Straight:
  - Left click and hold the *Road segment* icon (see Figure 3.10) until a drop down menu appears. Select the road segment tool: *Straight* .
  - Use the cursor to navigate into the design area of the editor (the middle window, see Figure 3.6 - section E) and click once to define the starting point of the straight segment.

- When drawing a line, press F9 to create a horizontal, straight road segment that is 150m long. Click again to place the line.


Figure 3.10: Defining a straight road

- Place a 10 m radius and 180° hairpin turn:
  - Select the *Turn* tool by right clicking in the design area and choosing it from the *Road Segment* submenu. Make sure to connect it to the endpoint of the first segment (orange dot inside the green marking as shown below).

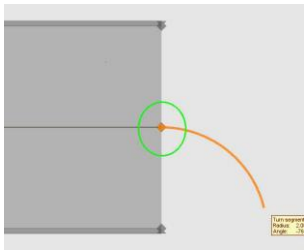


Figure 3.11: The orange circle indicates the correct connection of two road segments (The green marking is only for visualization in this tutorial)

- Place the second straight (150 m):
  - Pick the *Straight* Tool and connect (using the orange circle) to the previously drawn hairpin to place another horizontal straight (parallel to the first one).

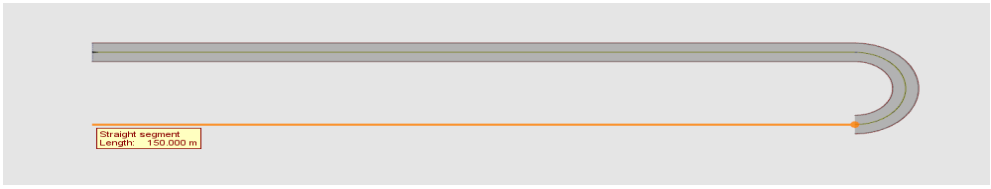



Figure 3.12: Placing the second straight

- Place the second and last hairpin turn (10 m radius) at 175° (not 180°).


Figure 3.13: Placing the second turn, connecting the end of the second straight

- Connect the last turn to the beginning of the very first straight:
  - Choose *Point list* via the *Road Segment* drop-down menu and connect the center of the last turn with the first straight. Wait for the two orange circles to appear.

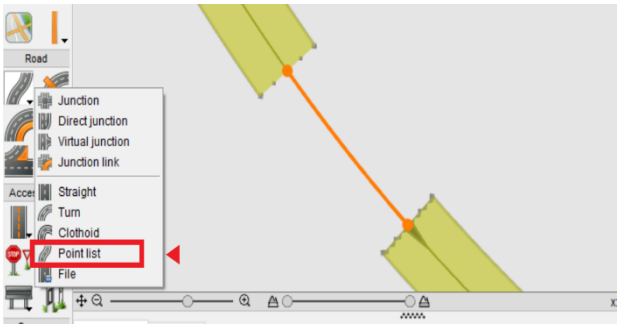



Figure 3.14: Connecting the road segments

### Single lane conversion

The road currently has two driving lanes: one in the clockwise direction and another in the counter-clockwise direction. In the following steps it is demonstrated how to convert this into a single road with cars driving in a clockwise direction in the center.

1) Use *Selection* mode (A) and click on the upper portion of the road. Then click on “Lane L0” (B). Delete the lane by clicking on the red ‘X’ in the upper right corner (C).

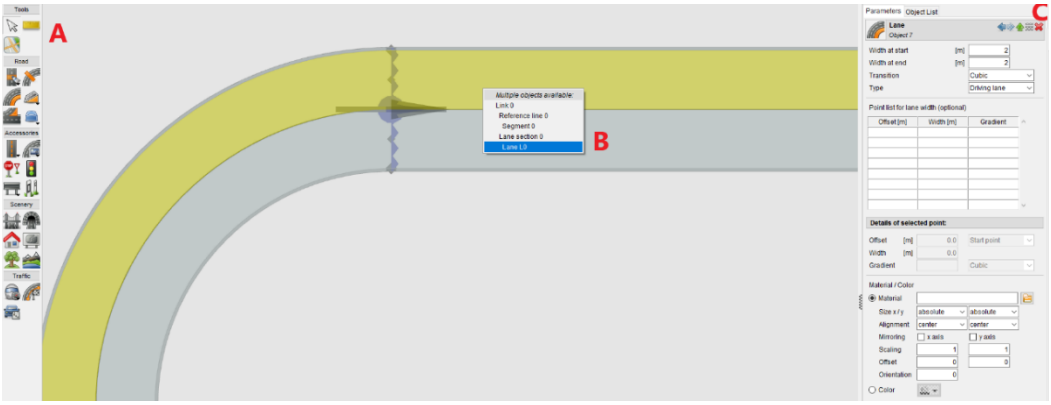



Figure 3.15: Delete Lane L0

2) Use the *Selection* mode again and click on the remaining road. Select “Lane R0” and using the panel on the right, set both the *Width at start/end* to 4 m.

3) On the *Road* panel, click on *Lateral offset (*A). Click on the road to select it. Hover with the mouse over the road until a small orange dot becomes visible. Then hold LMB and pull the dot down slightly. Enter the following values into the “List of nodes” table on the right panel (C): 0 2 0 (first row) and 364 2 0 (second row) and press enter. Now the reference line is centered in the middle of the road.

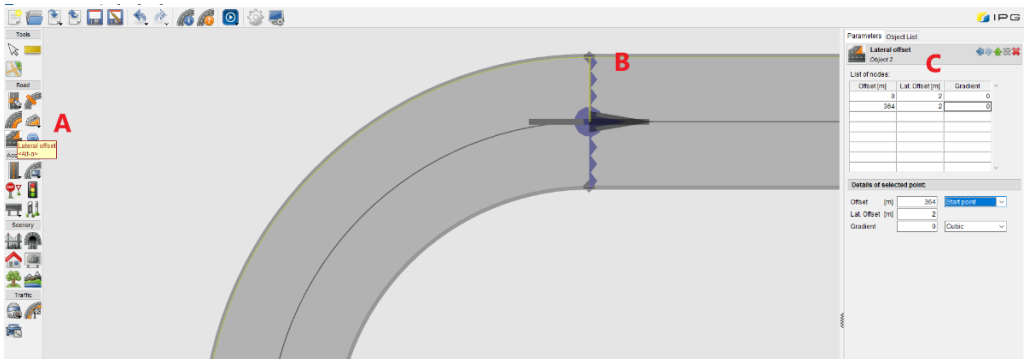



Figure 3.16: Change the reference line to be centered

### Path and Route

In order for the vehicle to drive on the road, a route and corresponding path are both required. The *Route* tool generates a path determined by the user and the road network.

The path has a variety of settings which can be edited and these are saved within the *TestRun* file. The next step will generate a path in the center of our circuit by commanding the route which the test car has to follow. This is achieved by selecting the corresponding *Links*. Please take note of the following hints for route generation:

1) Figure 3.17 shows the steps of route generation and the corresponding overlay color.

2) Renaming the route is possible, after selection, in the panel on the right side of the Scenario Editor.

3) A screen-tip will show you how many *Links* are added and how long the path is.

4) Available routes are marked using dashed orange lines and are selected using a left click. To complete the route definition, double click anywhere until the reference line becomes yellow.

5) You can check the results by clicking the *Preview* icon . If your route is defined correctly you will be able to move a yellow marker along the path using the slider within IPGMovie.

6) The road is saved inside the TestRun or it can be saved separately using .

### Exercise 2 - Step 4

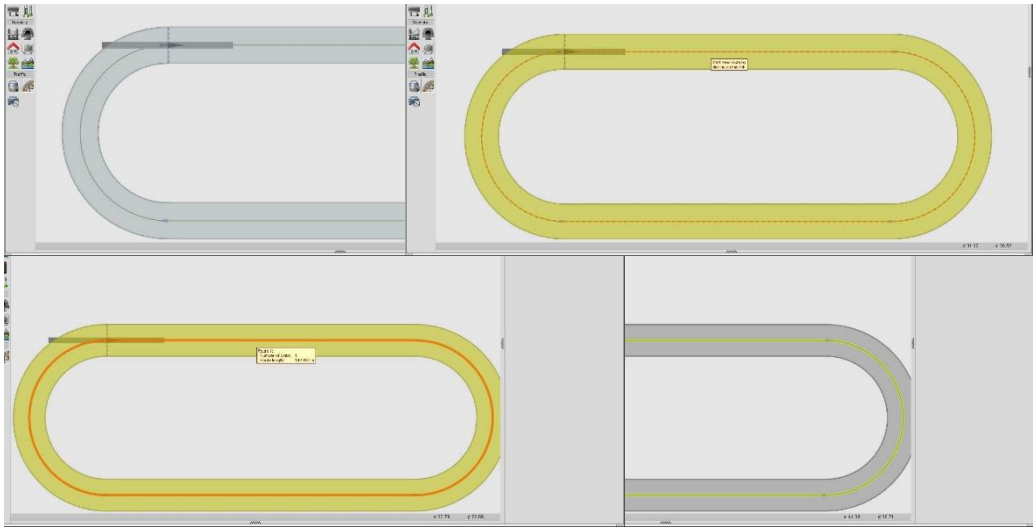


Figure 3.17: Route generation step by step:
Top left: Activation by hovering over road (grayish overlay)
Top right: Left click on the road to assign this link to the route (orange, dashed line)
Bottom left: Assigned route around the circuit (orange, full line)
Bottom right: Finished route generation by double clicking anywhere with LMB (yellow line)

- Add a route to the road:
  - In the *Traffic* tool panel select the *Route* icon and hover over the link which will be highlighted with a grayish layer.
  - Now the reference line needs to be chosen. Click LMB on the highlighted road and another click on the orange dashed line will assign the path to the route.
  - To finish the route generation simply double click LMB anywhere outside of the drawn segments.
- Save the road in Scenario Editor via under "myFSAE_Brakes_Slalom".

### Brake Test Area

The following is an extract from the 2021 Formula Student Rules:

> *"Brake Test Objective - The brake system will be dynamically tested and must demonstrate the capability of locking all four wheels and stopping the vehicle in a straight line*."

The braking area will consist of eight sets of pylons (cones). The vehicle will begin to brake as soon as it passes the 75 m mark. The corresponding maneuvers are configured in section 3.1.4 *'Maneuver'*, pg.31.

Some guidance for placing pylons:

1) If you wish to create a pylon alley of constant width, you should set the parameters when placing the first pair. This will ensure that the following sets are of the same spacing.

2) At 80-84 m the higher density of pylons helps ensure that the car is straight and perpendicular to the cones. The maneuver steps will be defined in the later exercises.

3) The pylons only work in one direction. The direction is shown as a small arrow at the beginning of the Link (see Figure 3.14).

### Exercise 2 - Step 5

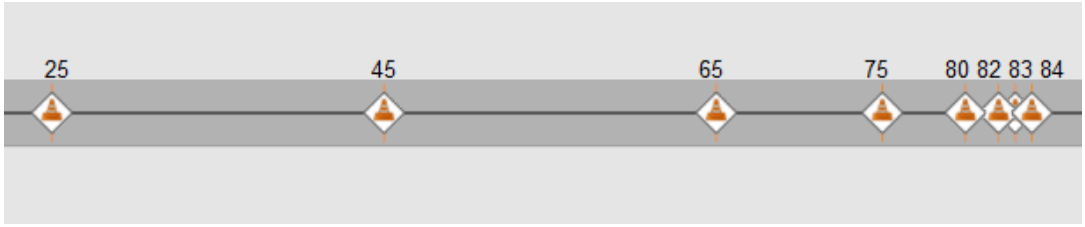



Figure 3.18: Brake test pylon placement

- Start with the *Pylon alley* tool :
  - Select via the *Markers* dropdown menu (hold LMB for context menu) in the *Traffic* section.
  - Place the first set of pylons in the center of the road, 25 m away from Link start. Define 3.5 m as the width and place seven more pylons as shown in Figure 3.18.

### Slalom Course

The slalom course is a must-have on any FSAE Autocross track. At Hockenheim, known as the Pendulum, a similar arrangement exists to test lateral acceleration. However, at the Pendulum, sets of straight pylons are driven around - not through.

In this example, we want to keep it simple and will therefore define the slalom as a pylon alley. The first step is to create the space needed to place the pylons. This widened area will be 60 m long and will have seven pairs of pylons positioned upon it.

1) The end of a lane section is defined by the start of the following lane section.

2) With every major change to the road network, CarMaker will need to regenerate the route.

## Exercise 2 - Step 6

The intermediate result of this exercise is a circuit with three individual sections:

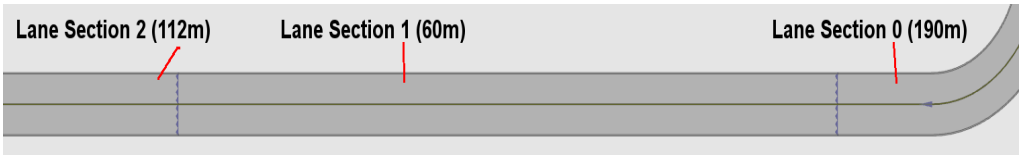



Figure 3.19: Lane Section 0: From Start 0 m to 190 m
Lane Section 1: From 190 m to 250 m
Lane Section 2: From 250 m to End (362 m)

- Create the lane section:
  - Select the *Lane section* tool , navigate to the second straight (after the first turn) and press the LMB to select the road segment. A secondary click on the road with the LMB defines the start of the lane section.
  - Choose the *Selection* tool to activate the new Lane Section and define the starting point. Set the beginning to 190 m (*Start offset*) and define the end of the section using an offset of 250 m from the start. The result should be a 60 m long section of road.

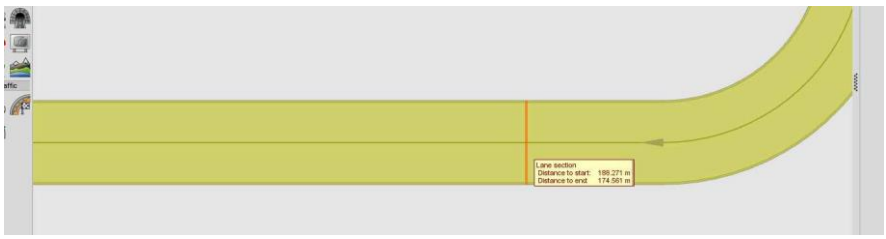


Figure 3.20: Orange line appears when intersecting a lane with *Lane Section* tool

### Lane Width

The next step is to increase the width of the newly configured 60 m long "Lane section 1". The goal is to have an 8 m broad track with a smooth transition from narrow to wide. This is done using the definition dialog on the right side of the Scenario Editor window.

1) The *Lane* tool can also be used to add new lanes to a road section. However, this is not necessary as we wish to edit the parameters of the already existing lanes.
2) To achieve a wider road with a smooth transition one has to edit the parameters on the right side of the Scenario Editor window. For further information, please refer to the User's Guide (*Main GUI* > *Help* > *User's Guide*) Chapter 5.2.5, '*Lane Section'*.

3) With every major change to the road network, CarMaker needs to regenerate the route.

## Exercise 2 - Step 7

This step will provide us a wider road segment within the test track:

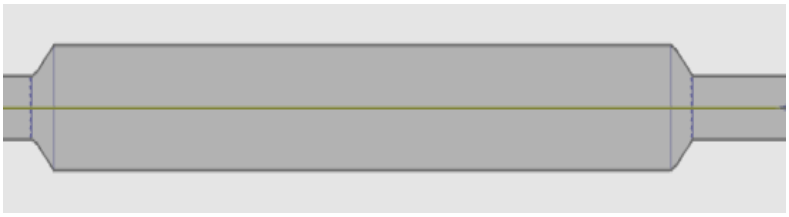


Figure 3.21: Wide section to accommodate pylons

- Create a wider road:
  - Select the *Lane* - tool from the *Road* section and click the existing lane from the previously created "Lane Section 1" (see Figure 3.19).
  - Then add a new lane by double-clicking the orange highlighted area

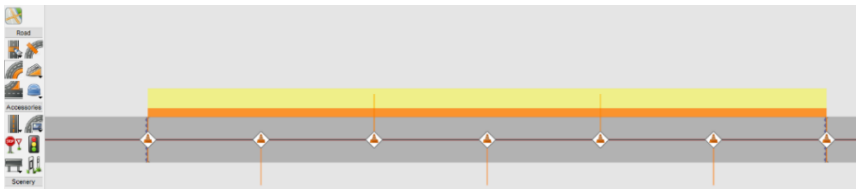


Figure 3.22: Adding a new lane

  - Afterwards, click on the new lane in the definition dialog (Figure 3.6 - section C) and edit the "Point list for lane width" and input the settings shown in Figure 3.23.
  - Repeat the steps for the other lane within the section.

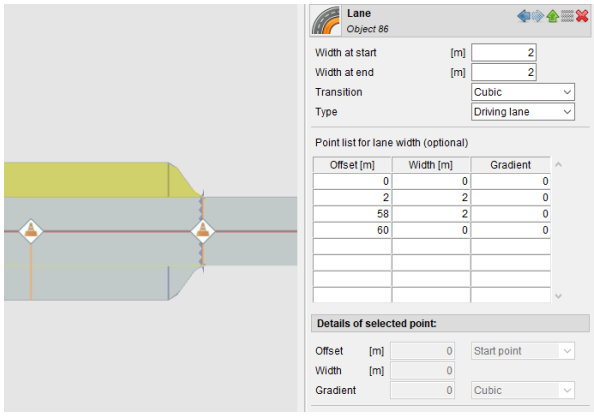



Figure 3.23: Configuring both lanes to fit the requirements

### Pylon Placement

A slalom course is a good way to test the cornering dynamics of a vehicle.

> "*The autocross track layout is a handling course built with the following guidelines: Slaloms: Cones in a straight line with 7.5 m to 12 m spacing*" - FSG Rules 2021

For simplicity, the pylon pairs will be placed with a width of 3.5 m for a distance of 10 m along "Lane section 1".

1) The placement method of the pylons can vary. You can either start from the center and add an offset to the desired position, or you can pick the left or right side lanes. For this step we will place two pairs (first and last) in the center (zero offset) and the rest will alternate between the left and right lanes.

2) If you wish to place multiple pylon pairs of constant width, you should set the parameters when positioning the first pair. This will ensure that the following sets are of the same spacing.

3) As mentioned in previous tips: the pylons only work in one direction. The link direction is shown as a small arrow at the beginning of the link. (see Figure 3.14).

4) Third time's a charm: with every major change to the road network, CarMaker needs to regenerate the route.

### Exercise 2 - Step 8

In step 7, the pylon pairs for the slalom course are positioned:

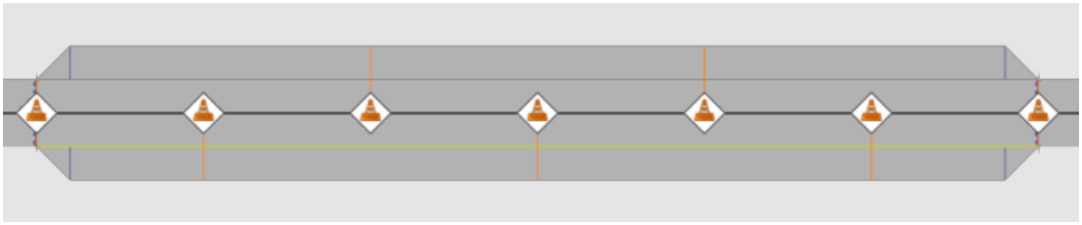


Figure 3.24: Finished Slalom course

- Place the Pylons:
  - Select the *Pylon alley* tool from *Markers* in the *Traffic* section of the tool panel.
  - Put the first pair of pylons in the center at the beginning of "Lane Section 1" (0 m from Link Start). Set the width to 4 m for the pylon pair and a lateral offset of 0.0 m from the Reference Line.
  - Add six additional pylon pairs with 10 m spacing between each. The lateral offset of the second pair is to be set to 2.0 m from the Reference Line. Therefore, the third pair has to be placed with a -2.0 m lateral offset. Alternate the lane for the following pylon pairs. Set the last pair to the center with zero lateral offset from the Reference Line.
- Regenerate the Route:
  - Select the *Route* tool and right-click upon the old route and select *Generate new path.* Save the roadfile and TestRun.
- Preview in 3D:
  In the menu tab press the - icon to load the new scenario in IPGMovie. If the route is correct, you will be able to move the camera along the route using the slider in IPGMovie.

More information about IPGMovie is found in the User's Guide (*Main GUI* > *Help* > *User's Guide*).

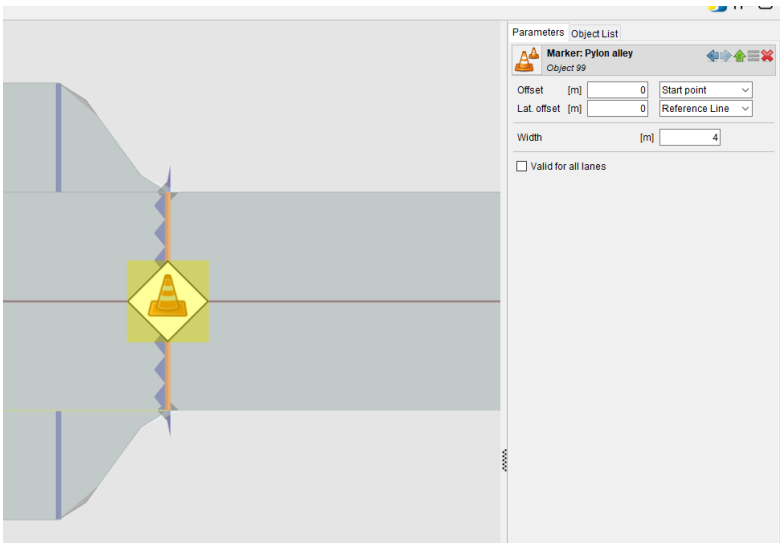



Figure 3.25: Placement of slalom course pylons in step 7

This was the last step in creating a test track. After saving, feel free to close Scenario Editor.

## 3.1.4 Maneuver

A maneuver describes the driver's actions. You can define several successive maneuver steps (mini maneuvers) along the track. The braking test requires a special maneuver which we will define in this chapter. The path through the slalom course will be commanded by the pylon alley. Therefore, no special maneuver is required as the driver will automatically know what to do. You can open the maneuver window in the Main GUI by clicking on *Parameters > Maneuver* or *Ctrl+m*.

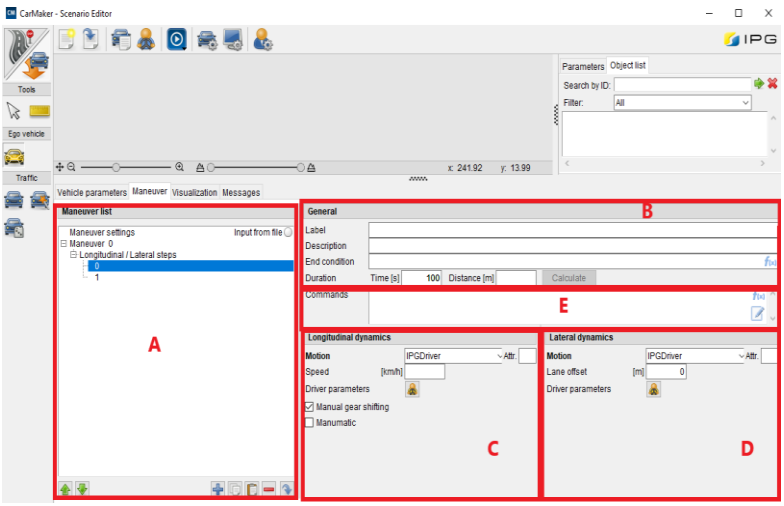



Figure 3.26: CarMaker Maneuver window
A: Overview of the maneuver's steps
B: Duration information of a step
C & D: Dynamics controller configurations
E: Minimaneuver commands and Real Time Expressions

### Maneuver Definition

A new maneuver step is added via the *New* button below the "Global Settings / Preparation" (section A) field.

Different values of the maneuver can be input in the "Specification of Maneuver step" portion of the window (B). A *Label* is an alias to the maneuver step.

Please bear in mind that a label must not begin with a number as it would lead to errors in the simulation. In the *Description* Field you can write a brief description of a maneuver step.

Two simple ways to define the duration are by either time or length. If both parameters are specified, the simulation stops when a condition is fulfilled. If only one field is initialized the second one will be disregarded.
A third way to stop a maneuver step is via user defined end conditions. To do so, click the RMB in the "End Condition" field. Now you can access all CarMaker variables (see section 3.3.1 *'Data Access'*, pg. 41).

Once one is selected, it can be paired with a logical operator to create a condition. If this condition becomes true, the maneuver step is terminated and the next in order is started.

At the bottom of the window (E), *Minimaneuver Commands* enables the user to specify several maneuver commands such as maneuver jumps, creation of new quantities and variables, calculations, etc (see appendix *'Real Time Expressions'* in the User's Guide).

To find out more about maneuver definition, please refer to the User's Guide (*Main GUI > Help > User's Guide).*

### Dynamic Controller

The maneuver parameterization is divided into two parts: *Longitudinal Dynamics* and *Lateral Dynamics*, which respectively control the (throttle, brake, clutch) pedals and the steering wheel. To learn more about this, please read the User's Guide (*Main GUI > Help > User's Guide)* Chapter 7, '*IPGDriver*' or the following section in this tutorial.

The goal of the next exercise is to define several maneuver steps. The first step #0 will tell the driver to accelerate the car for 75 m (see (EQ 1)). The last pylon pair of the braking area is set at this point on the track. Exactly at this position we want to start the brake test which is defined by the following maneuver step #1.

For step #1 we'll not use the IPGDriver, but the GBCP (Gas, Brake, Clutch and Gear-Number) - controller which offers the option to predefine the percentage of pedal actuation. We need to set the clutch (as to not kill the engine) and brake (standstill and block) both to 1.0, and 'gas' to 0.0. As the clutch pedal is fully pressed, we don't need to set the gears.
(EQ 2) describes the end condition, via Real Time Expressions, for the brake step and fullfills part of the rule:

> "*The brake system will be dynamically tested and must demonstrate the capability of locking all four wheels and stopping the vehicle in a straight line*" - FS Rules 2021

Later (Chapter 3.2), you will learn how to check the vehicle offset to the centerline and rotation around z-axis (yaw) to determine whether the vehicle remained in a straight line.

Step #2 will accelerate the car to a certain value (user defined driver) or to maximum possible velocity (racing driver). It will remain active until the end of the first lap (see (EQ 3)). Please note that this quantity (EQ 3) is an integer and starts with 0 (zero).

1) The racing driver attempts to stay in the center of the road but due to the car's performance it may veer off. Later, you will be able to check the steering wheel torque in IPGControl which is an interesting value to investigate.

2) Please keep in mind that the CarMaker sampling rate is locked at 1000hz. There is no guarantee that a variable can have an exact value (e.g. 75.0000000m). Therefore, we have to use "<=,>=" - logical operators, instead of "==".

## Exercise 3 - Step 1

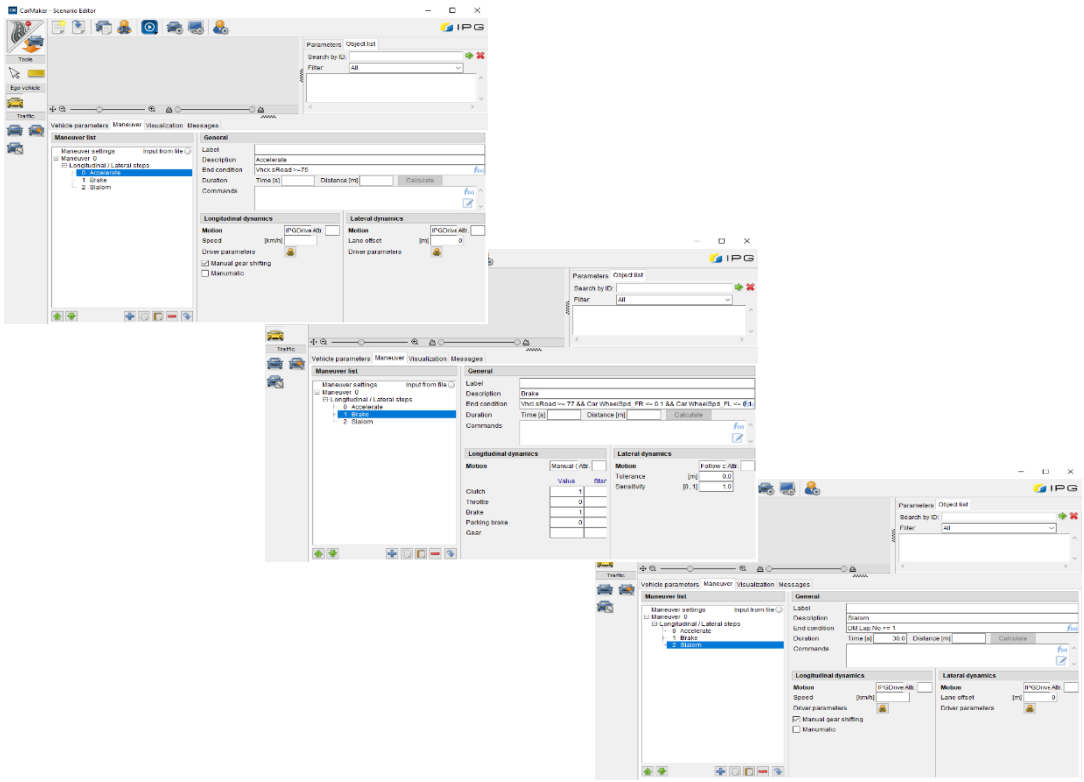


Figure 3.27: Maneuver dialog with successive maneuver steps

Left: Accelerate maneuver Middle: Braking maneuver Right: Slalom maneuver

- Open the TestRun from Exercise 1 "myFirst_TestRun"
- Load road in the Scenario Editor:
  - In the Main GUI, click on *Parameters* > *Scenario / Road* or press *Ctrl+r*.
  - Import road definition "myFSAE_Brakes_Slalom.rd5"
- Create Maneuver Step #0:
  - Open the maneuver window in the Main GUI by clicking on *Parameters* > *Maneuver*. Add a new maneuver with the *New* button and write "Acceleration" in the *Description* Field.
  - Select the *IPGDriver* for both dynamic fields from the dropdown menu. The end condition is:

$$Vhcl.sRoad >= 75 \quad \textbf{(EQ 1)}$$

- Create Maneuver Step #1:
  - Add a new maneuver with the *New* button. Write "Brake" in the *Description* Field.
  - Select "Manual" controller from the dropdown menu in the *Longitudinal Dynamics* field. The clutch and brake pedals must be fully pressed, set the values as shown in Figure 3.27: - (Middle).
  - The lateral controller is set to *IPGDriver*. The end condition is:

$$Vhcl.sRoad >= 77 \;\&\&\; Car.WheelSpd\_FR <= 0.1 \;\&\&\; Car.WheelSpd\_FL <= 0.1 \;\&\&\; Car.WheelSpd\_RF <= 0.1 \;\&\&\; Car.WheelSpd\_RL <= 0.1 \quad \textbf{(EQ 2)}$$

- Create Maneuver Step #2:
  - The third maneuver step is a simple acceleration maneuver with the lap number as the end condition. This quits the step and ends the simulation as the vehicle crosses the start/finish line:

$$DM.Lap.No >= 1 \quad \textbf{(EQ 3)}$$

## 3.1.5 Driver Model

*IPGDriver* is a controller used for following a path on a track with given speed parameters. It can be activated in the Maneuver window and enables you to add the control actions of a human driver to your vehicle simulation. These actions include steering, braking, throttle position, gear shifting and clutch operation. IPGDriver can be used to control only the steering and not the speed or vice versa.

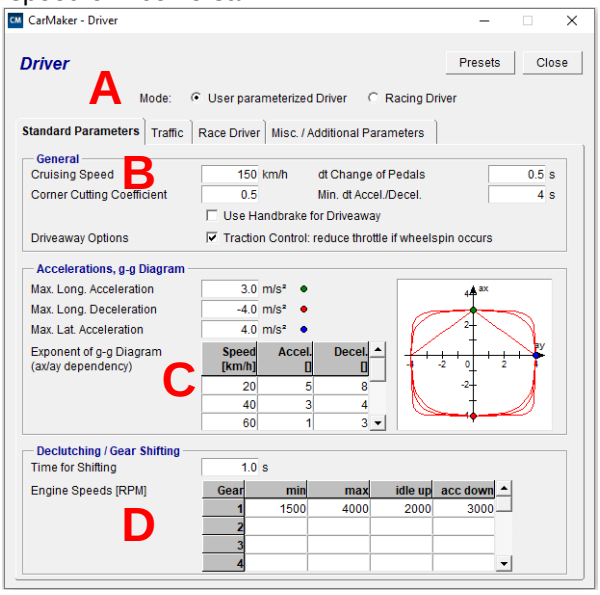



Figure 3.28: Driver dialog - Standard Parameters tab:
A: Driver mode selection in CarMaker
B: General settings such as target speed or corner cutting coefficient
C: Accelerations settings and g,g diagram
D: Shifting point definition

### User Parameterized Driver

There are two different driver configurations to choose from. The first option is the *User Parameterized Driver* which can be customized via an extensive list of parameters.

The *Cruising Speed* (see Figure 3.28 - section B) is the target speed which the driver will try to reach. In Figure 3.28 (section C) is the g-g diagram. It describes the maximum allowed (purely longitudinal and lateral, or combined) accelerations as plots.

The panel in section D is to set the gear shifting values. For example, the shift time can be varied (the shorter, the faster). A common error is to set values for idle up/acc down which are smaller/higher than the engine minimum/maximum speed. For further information please read section 7.2.3 '*User Parameterized Driver > Declutching/Gear Shifting'* in the User's Guide (*Main GUI > Help > User's Guide*).

This type of controller simulates the behavior of a real driver who can steer, accelerate, brake, slow down, shift gears, cut corners, and adapt their driving style according to the track.

### Racing Driver

The second type of driver is the *Racing Driver*. It drives the car up to its physical limits and thus attains maximum power and speed. However, it is essential that the driver knows the car's limits. The driver determines the limits through specific driving maneuvers. This procedure is called *Driver Adaption* in CarMaker and can be initiated via *Simulation > Driver Adaption > Basic Knowledge*.

In the *Basic Knowledge* tab four different adaptions are proposed:

- *Vehicle Limits* - This module calculates the maximum longitudinal acceleration, deceleration and lateral acceleration of the vehicle.
- *Controller Dynamics* - This module calculates the time preview used by the driver to predict his actions during a TestRun.
- *Engine speeds for shifting* - This module calculates the optimal engine speeds for shifting.
- *Friction* - The friction used by the dynamic area to train the driver.

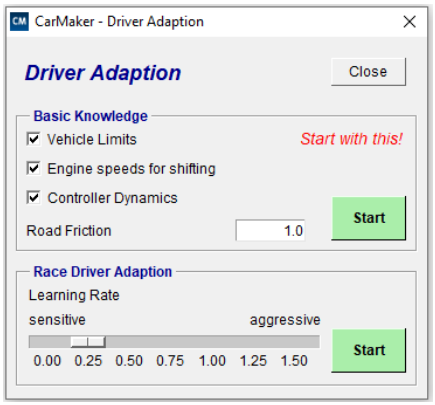


Figure 3.29: Driver adaption in CarMaker

A full guide as to Racer Driver Adaption can be found in section 6.4, pg. 140.

### Useful Tips

1) The Racing Driver controller should only be used on race circuits (closed loop). It is optimized to make the fastest lap time on a closed racetrack. Therefore, this is the right option for Autocross and Endurance events.

For other competitions (e.g. Acceleration and SkidPad), the User Parameterized Driver is the better choice. This driver can be tuned for a specific task, which means it can be focused on pure longitudinal dynamics (for the Acceleration event) or lateral dynamics (SkidPad). The focus is set by defining a very high max. long./lat. acceleration. It can be an unrealistically high value such as 50 $m/s^2$ to find the limits of the car.

2) Racing Driver must be adapted to both car and course.

3) A right-click in the Driver dialog gives access to predefined driver characteristics, such as a defensive, normal, or aggressive driver.

With the maneuver now defined we wish to observe the difference between each driver. As stated, the User Parametrized Driver is the optimal choice for maneuvers and specialized tasks. Conversely, the Racing Driver is perfect for closed loop circuits. In our case we have both, a closed circuit and maneuvers. But as we can tune the driver for this TestRun we will select the User parameterized Driver.

To show the difference between the driver characteristics, two TestRuns will be performed which only differ in the choice of the driver parameters. Later in the chapter we will learn how to check the lap time, maximum velocity, and lots of other interesting data.

**Exercise 3 - Step 2**

- Create TestRun with normal driver:
  - Open the Driver window in the Main GUI by clicking: *Parameters > Driver* or *Ctrl+d*. Activate the *User parameterized Driver* in the Driver's window and set the "normal" characteristics via right-click.
  - Close the window and save the TestRun as "Slalom_Brake".
- Create TestRun with aggressive driver:
  - Repeat the steps for an aggressive driver.
  - Save the TestRun to "Slalom_Brake_Aggressive".

## 3.1.6 Environment

The environment module under *Parameters > Environment* enables the definition of environmental conditions such as temperature, time of day or wind velocity of the simulation. If a model takes these parameters into account, they will influence the results of the simulation. Please read the User's Guide (*Main GUI > Help > User's Guide*) for further information.

## 3.1.7 Model Check

The last step before starting a simulation is to check the equilibrium state of the car using the "Model Parameter Check" (Model Check). This utility is quite useful for determining whether changes made have had the desired effect on your car. To do so open *Simulation > Model Check*. Select a field of interest and then click "Start". Please read the User's Guide (*Main GUI > Help > User's Guide)* for further information.

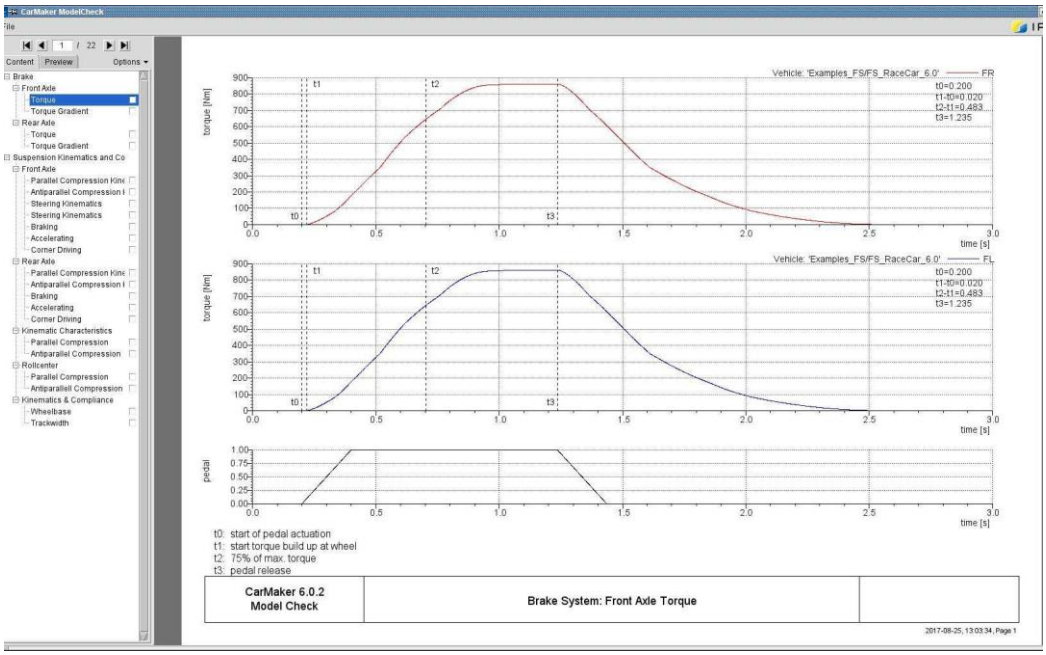



Figure 3.30: Results of simulated brake system actuation in Model Check

# 3.2 CarMaker Analysis and Manipulating Methods

### Introduction

Post-processing of simulation results can be used for several different reasons. For example, you may like to compare the results in a diagram, build simulation tables containing several quantities or create illustrative material for presentations. CarMaker offers the functionality to use generated data for all of these applications.

A precondition for post-processing is the generation of simulation results. Therefore, you first must successfully simulate a TestRun by:

- The parameterization of a vehicle including the tires,
- The definition of a virtual road,
- The selection of a driver and
- The declaration of maneuvers

## 3.2.1 Visualization

### IPGMovie

IPGMovie is a very powerful visualization tool that offers a lot of functionalities. To avoid lengthening this tutorial unnecessarily, please view IPGMovie's manual (*Main GUI > Help > IPGMovie)* for a full explanation of its features.

Either during a simulation or afterwards, you can alter the camera direction (displayed in the IPGMovie window). Click and hold the LMB, then move the mouse to the left or to the right. Do the same upwards and downwards. Click and hold the middle mouse button, then move the pointer upwards and downwards or use the mouse wheel to zoom in/out on the car.

In IPGMovie all the features of the road are visible: the differences in friction coefficient (different colors), road signs and, of course, the route itself. If your graphic card is powerful enough you can export the movie as an .avi, .mpg or .wmv file (in IPGMovie window select *File > Export Video/Images*). However, you'll need to install a video codec such as DivX or XVid first.

Furthermore, using the "Overlay" option in IPGMovie that can be found via *View > Overlay Left / Right* you can add additional displays to the IPGMovie window. In the top left/right corner you can add a route view, speedometer, gg-diagram, driver's steering wheel movements and much more.

Under *View > Quality Settings* you can change IPGMovie's resolution settings. For maximum perfomance we recommend you use the "Performance" mode.

Another useful feature of IPGMovie is the functionality to compare two cars in the same movie window. Once your primary simulation is finished you can select *Scene > Reference Vehicle > Copy current motion data* (upon IPGMovie menu panel). Then, activate "Primary & Reference Vehicle" in the "Scene" menu.

The goal of the next exercise is to view a primary and reference vehicle in IPGMovie. Therefore, the formerly created TestRuns with agressive and normal driver will be compared. While the second simulation is running you will see two cars in the IPGMovie window: the currently running aggressive TestRun and the former TestRun with the normal driver.

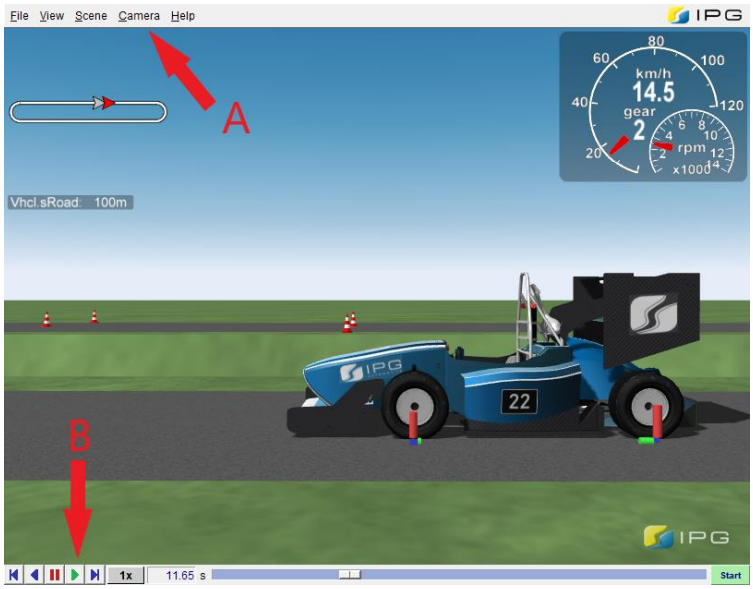



Figure 3.31: IPGMovie side view with overlay:
A: Menu panel to access settings
B: Control panel with buttons and slider

## Exercise 4

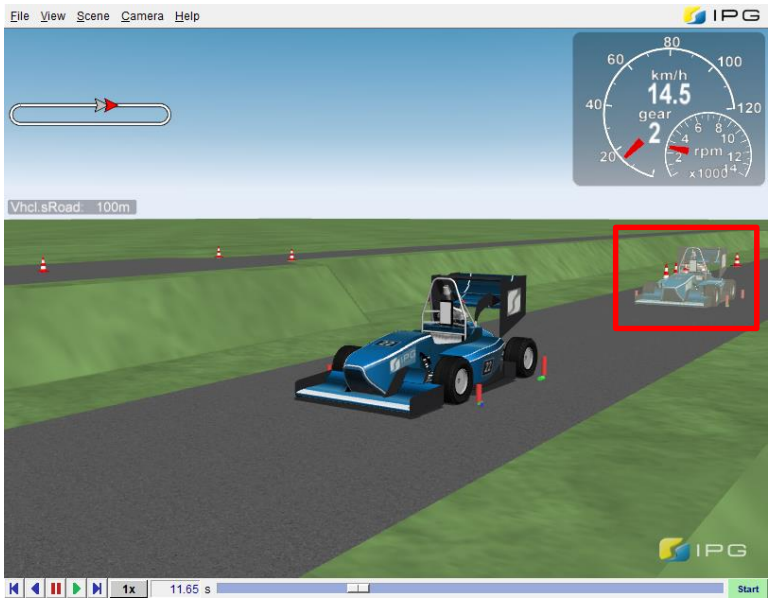



Figure 3.32: IPGMovie in "Quality" mode with reference vehicle (red frame), route view (left) and speedometer (right)

- Start TestRun with normal driver:
  - Open the first TestRun "Slalom_Brake". Set the simulation speed to *max* and the results to *collect only*. Open IPGMovie (*Main GUI > File > IPGMovie*)
  - Start the simulation and wait until it is complete. Save the results by hitting *Save* in *Storage of Results* (Main GUI).
  - In IPGMovie select *Scene > Reference Vehicle >Copy current Motion Data* and then *Scene > Primary & Reference Vehicle*.

- Start TestRun with aggressive driver:
  - Now open the second TestRun "Slalom_Brake_Aggressive" which was prepared previously and start another simulation with *collect only* settings as above.
  - Set the simulation speed to *real time* and start the simulation via the *Start* button in the Main GUI.

## Instruments

Another feature for observing the simulation is the *Instruments* window. You can choose between an ordinary and a Formula Student version. Both can be opened using the Main GUI via File > *Instruments*.
In both cases the "Instruments" window contains a speedometer, a rev counter, four histograms indicating the actual steering angle, clutch, brake and gas pedal position and another graph showing the active gear.

After opening the "Instruments_FS" version, at the bottom left of the window, sector times and lap times are recorded once you use the Racing Driver upon a closed circuit.

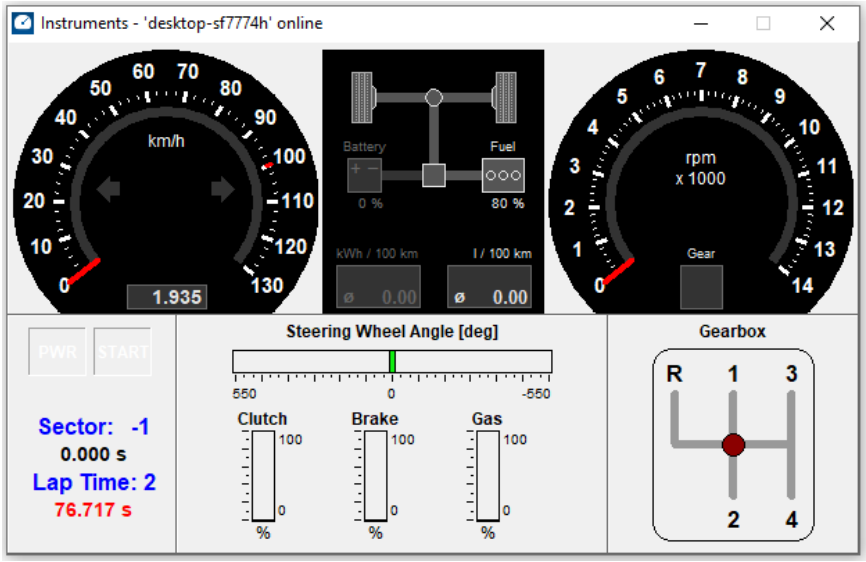



Figure 3.33: The "Instruments FS" window

Having finished both simulations and saved the results, we can now start to analyze both of the TestRuns.

# 3.3 IPGControl

IPGControl is CarMaker's analysis tool. Various simulation result files can be loaded (Figure 3.34 - section A) consecutively to plot and compare signals. Multiple diagrams can be generated and displayed within a single window - one below the other (Figure 3.35).

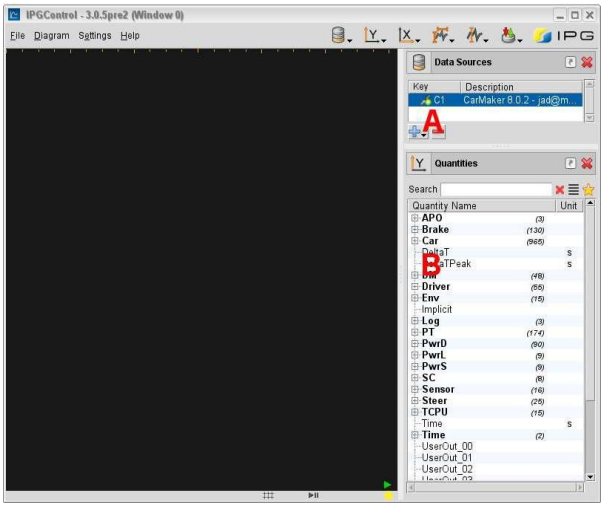



Figure 3.34: IPGControl GUI to select quantities to be plotted in Data Window
A: List of data sets
B: All available UAQ's

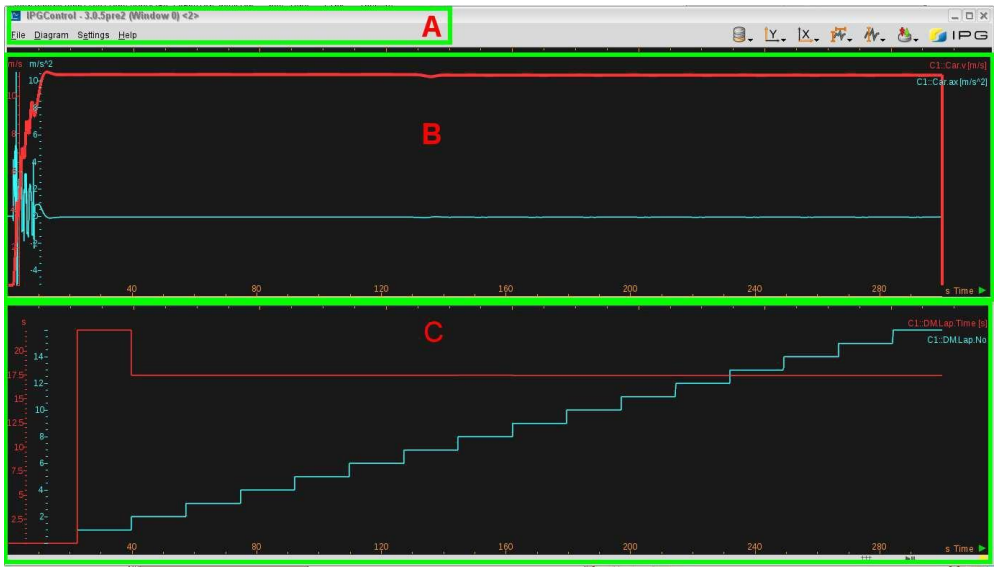



Figure 3.35: Data Window:
A: Menu panel
B: Diagram #0 (velocity and acceleration of the car is plotted)
C: Diagram #1 (lap number and time are plotted)

### 3.3.1 Data Access

To analyze the results, we must understand how the internal data within CarMaker is accessed. During a CarMaker simulation, a few hundred variables are calculated that are also available for analysis. Some of these are read-only, others can be edited using tools like Direct Variable Access (DVA) or MATLAB/Simulink.

The public variables, so called *User accessible quantities* (UAQ) are gathered in logical sub-systems. The naming convention is explained in detail in the CarMaker Reference Manual. The UAQ contains all public variables, both the read-only and the readable/writable ones. We have already utilized UAQs during the maneuver definition (see section 3.1.4). Readable data is plottable in IPGControl which is described later in the chapter.

To access a writable variable, open the DVA dialog. This can be found upon the Main GUI via *Application > Direct Variable Access*. The DVA enables the user to influence the simulation once it is already running by manipulating quantities and assigning new values (during the simulation). There are several ways to use the DVA functionallty.

- DVA dialog (this chapter)
- Minimaneuver commands in section 12.3, '*Direct Variable Access Commands'* in the User's Guide (*Main GUI > Help > User's Guide*)
- ScriptControl (*Main GUI > Help > Programmer's Guide*)

Using the Direct Variable Access dialog, a new value can be assigned to a quantity in a quick and easy manner. Furthermore, the DVA dialog can be used to display the current value of the selected quantity.

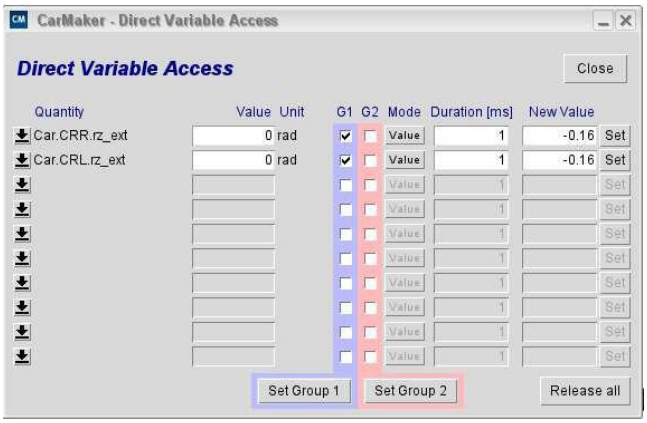



Figure 3.36: Direct Variable Access dialog

In Figure 3.36: Selected variables with their current and target values.

**Quantity** On the left, select the UAQ you wish to modify.

**Value** Current value of the UAQ.

**Unit** Reference unit for this UAQ.

**G1/G2** To assign UAQ to a group.

**Mode** You can either choose a certain value, an offset, or a factor for the UAQ.

**Duration** The time during which the quantity should be overwritten.

**New Value** An absolute value, offset or factor with which to manipulate the selected UAQ.

**Set Group** Pressing this button, the quantity is overwritten for the duration specified in #Cycles.

**Release all** This button resets all quantities that were overwritten.

Some of the quantities may be clones of others e.g. the *Car.Road* and *Sensor.Road* are exactly the same values, but for accessibility and due to backwards support these quantities are still available. The *Vhcl* subsystem is a subset of *Car* and contains frequently used quantities.

To become familiar with the use of UAQ's, IPGControl and real time expressions, we will generate a new TestRun. The goal is to simulate the effectiveness of rear axle steering. Since rear axle steering is best evaluated under steady-state conditions, we will create a simple left-handed turn of 30 m radius. As the simulation runs, we will overwrite two UAQ's which are used to change the rotation about the z-axis of the rear carrier. The rotation is calculated in radians. A good starting point will be -0,0523599 radians (~3 degrees) for each side. To manipulate the UAQ, we will use the DVA dialog.

### Exercise 5 - Step 1

- Create a new TestRun:
  - Open new TestRun and Save as "RearAxleSteering"
  - Pick "FSC_RaceCar" as your vehicle.
  - Choose the normal driver (Corner Cutting Coefficient = 1.0).
  - Create a 360° turn of 30 m radius in the Scenario Editor.

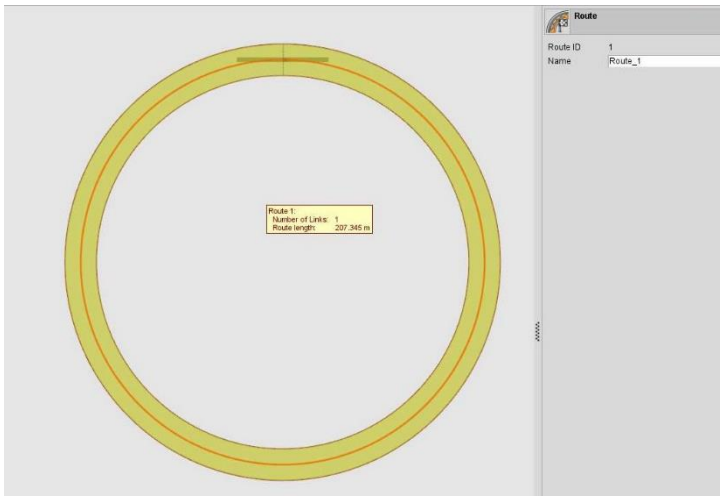



Figure 3.37: 360 Turn generation

- Move the Reference line to the center of the road:
  - Using the *Selection* mode (, A); click on the outer side of the road. Click on "Lane L0" (B) and delete the lane by pressing the red X in the upper right corner (C).
  - Click again on the remaining road, select "R0" and set width at start/end to 4 m

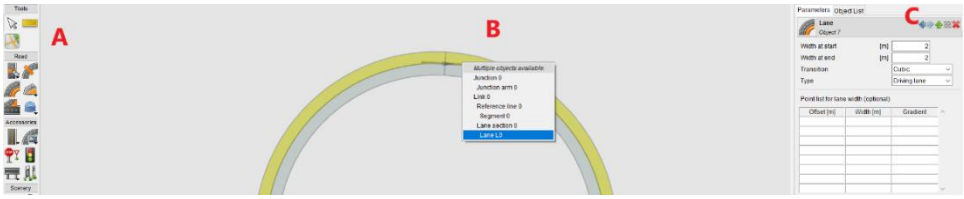



Figure 3.38: Deleting Lane L0

- Select *Lateral Offset* on the left *Road* panel, click on the Reference Line and edit the point list for lane width as shown in the following picture (like in Exercise 3):

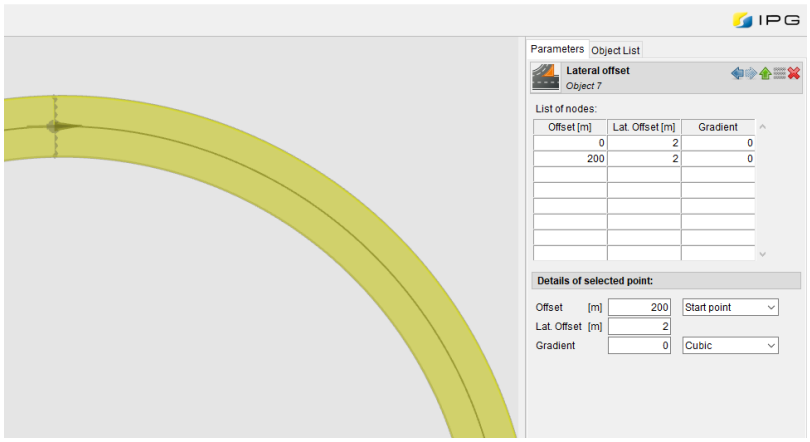



Figure 3.39: Lateral Offset for the Reference Line so that it is centered

- Create a route:
  - Generate the route by clicking on *Route* on the left panel and then use the LMB to select the desired Link. A dashed line will appear. Click the dashed line and end the definition with a double click anywhere else within the drawing area.
  - In the ⚙ - menu choose the active route (e.g. Route_1)

- Define the maneuver:
  - The maneuver should contain only one step with a duration of 300 s.
  - In Figure 3.40: there are two commented lines in the *Minimaneuver Commands* section. Those real time expressions (RTE) are not active (i.e. "#") right now. Before we start using RTE we should take a short look at the Direct Variable Access window.

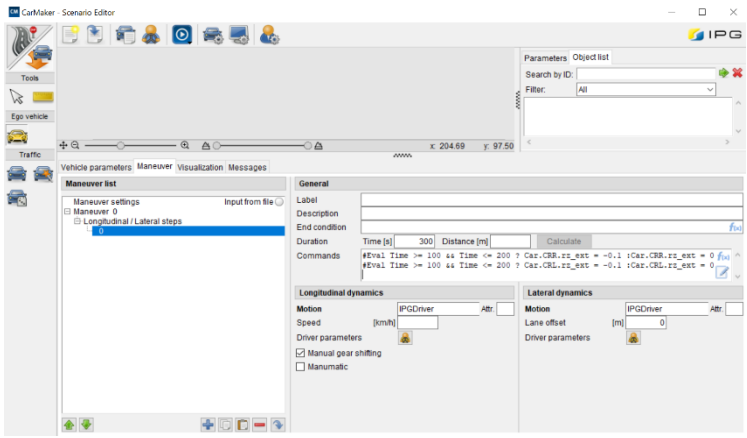



Figure 3.40: Maneuver dialog

- Select the two UAQ's:
  - In the DVA dialog choose "Car.CRR.rz_ext" and "Car.CRL.rz_ext" from the drop-down menu.
  - Set the UAQ to "Group 1" by marking the corresponding checkbox.

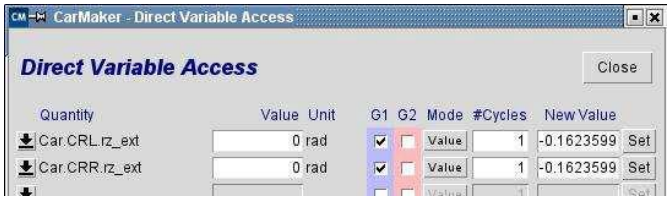



Figure 3.41 DVA dialog - manipulation tool for the writable UAQ's

- Create a second diagram in IPGControl Data Window #0:
  - Go to *Diagram* > *Add Diagram* in the Data Window menu panel. To set a white background as in Figure 3.42 go to the IPGControl GUI and configure via *Settings* > *White Diagram Background*.

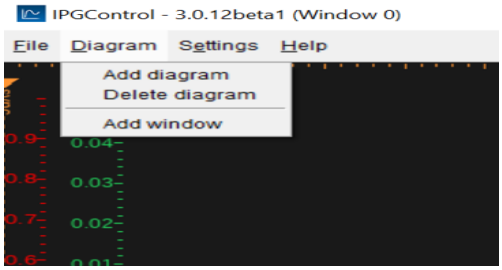



Figure 3.42: Add Diagrams to the active Data Window

- Add three UAQ to the upper diagram by using the quantity browser:
  - Clicking in the upper diagram activates it.
  - Go to *Data Window #0* > *Quantity* > *Browse* to open the browser dialog for UAQ.
  - Search for the following UAQ and add them to the upper diagram:
    "DM.Lap.Time" - The measured lap time
    "Car.v" – The velocity of the car
    "Car.ax" - Translational acceleration of vehicle connected body

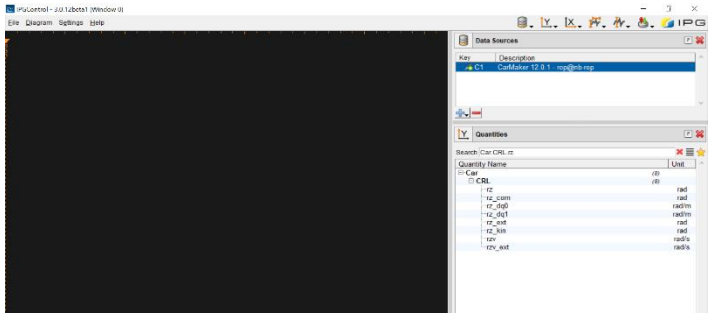


Figure 3.43: Quantity tab – where quantities can be browsed for plotting

- Add two UAQs to the lower diagram using the GUI:
  - Click in the lower diagram. You can either search in section B/C (see Figure 3.34) and select the UAQ with LMB or by entering it's name into the search bar.
  - Please add the following two UAQ to the lower diagram:
    "Car.CRR.rz_ext" - The rotation around rear right carrier z-axis
    "Car.CRL.rz_ext" - The rotation around rear left carrier z-axis
- Change the units to degrees:
  - As can be observed, the unit of this value is in radians. To change it to a different unit you must select the UAQ by left clicking it within the legend of the Data Window.
  - Then click RMB on the active UAQ (see Figure 3.44). Once clicked, the context menu appears and presents a variety of options. Choose *Unit of Selected Quantities* > *deg* for both UAQs and "fit the data totally" in the drawing field.

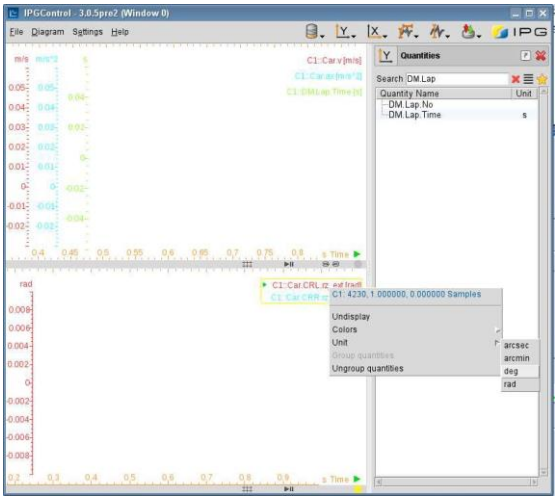



Figure 3.44: Context menu opens with RMB on plotted value

- Start the simulation:
  - Press the green *Start* button in Main GUI
  - After a few laps, click the *Set Group 1* button in the DVA dialog.
    During the following cycles the UAQ will be manipulated until the target value is hit. In IPGMovie and IPGControl you will see a shift in the rear carriers. The visible shift angle depends on the target value.

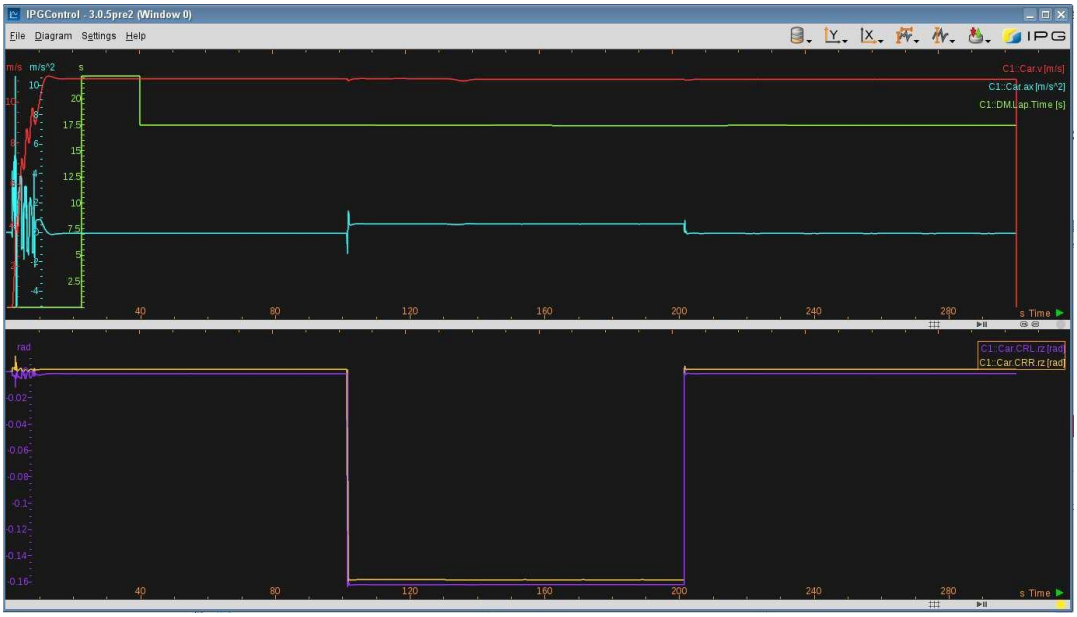



Figure 3.45: The simulation with a shift in rotation angle from 100 to 200 s

## Real Time Expressions

Since we want to compare the results of different steering angles, we need an exact manipulation of the UAQ's. The best way to do this is by implementing real time expressions (RTE) in the minimaneuver commands. There are several ways to use RTE as embedded commands in the minimaneuver command language. In that case, the RTE is indicated by the prefix "Eval".

In Figure 3.40 shows commented RTE. In minimaneuver syntax:

- "Eval" is to assign a line of code as an RTE.

- "?" - if (...) operator

- " : " - the column operator indicates the ‘else’ logic.

```
Eval Time >= 100 && Time <= 200 ? Car.CRR.rz_ext = -0.1 :Car.CRR.rz_ext = 0
```
**(EQ 4)**

```
Eval Time >= 100 && Time <= 200 ? Car.CRL.rz_ext = -0.1 :Car.CRL.rz_ext = 0
```
**(EQ 5)**

(EQ 4) and (EQ 5) contains a simple if-else logic which is active from 100s to 200s. The C-code snippet contains the same logic:

```
if (Time >= 100 && Time <= 200)
{Car.CRR.rz_ext = -0.1; Car.CRL.rz_ext = -0.1;}
else{ Car.CRR.rz_ext = 0 ; Car.CRL.rz_ext = 0;}
```
**(EQ 6)**

Please read the *Appendix - section C* of the User’s Guide to learn more about RTE.

**Exercise 5 - Step 2**

- Try a variety of values:
  - Observe the changes in lap time, acceleration and velocity.
  - Open *Instruments* to monitor the actuation and reaction of the driver as the UAQ changes.
  - You can change the accelerations to be more aggressive manually or via a right-click in the *Driver* menu.
- Create a result file:
  - Load *normal* driver and after completing a simulation, click the *Save* button upon the Main GUI.
- Repeat for the *aggressive* driver:
  - Save as “RearAxleSteering_Aggresive”

## Data Storage

If you choose to save the results of your simulation, all values are not stored due to memory constraints. Only the most common ones are saved, but this can be changed to include any value that the user desires. The current stored values can be checked and changed via *Application* > *OutputQuantities*.

To be able to compare the lap time, lateral acceleration and velocity of the car obtained by the two different drivers, please ensure that the lateral acceleration "Car.ay" and the velocity "Car.v" are activated in the *Storage of Results/OutputQuantities* dialog.

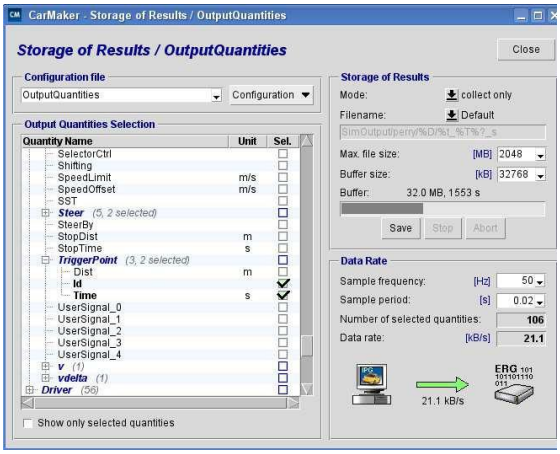



Figure 3.46: CarMaker OutputQuantities

Please also investigate "Vhcl.sRoad" for further analysis. To see the results (lap time) activate quantities "DM.Lap.Time" and "DM.Lap.No".

In this dialog you can also save or load custom settings for result storage behavior. Keep in mind that these settings will be applied to the whole project folder.

## 3.3.2 Compare Results in IPGControl

To compare simulation results you have to perform several TestRuns and save the results of each (making sure the *Storage of Results* is enabled). Pay attention to the *OutputQuantities* configuration (*Application* > *OutputQuantities*), to ensure that desired quantities will not be mistakenly omitted from the results file. You can find the simulation result files of CarMaker in the following file path of your project folder:

```
<yourprojectfolder>/SimOutput/<HostNameOfYourPC>/<Date>/NameOfYourTestRun-Time.erg
```

With IPGControl, you can compare the results using different diagrams. You can choose between two different options:

- compare the last simulated TestRun with a new one
- compare the last simulated TestRun with a saved simulation's results

IPGControl offers the possibility to add datasets from a file or an online simulation. To compare several TestRuns without saving the results, a 'snapshot' can be taken of the current plot. It is then possible to run and plot a new simulation and compare it to the 'snapshots' of the previous results. Follow the steps in the exercise below to do so.

**Exercise 6 – Step 1**

- Run the first Simulation:
  - Open *IPGControl* from the Main GUI.
    Choose a TestRun, run it and plot the vehicle velocity ("Car.v") to the Data Window #0.
  - Take a snapshot of the current plot:
  - Click and hold on the *Plus* icon beneath the Data Source Window and select: *Snapshot from current data*.

- Run the second Simulation:
  - Vary the TestRun (e.g. change the weight of your vehicle).
  - Reconnect the simulation using *File > Reconnect/Reload*.
  - Add "Car.v" from the second TestRun to same diagram.
  - Start the simulation.

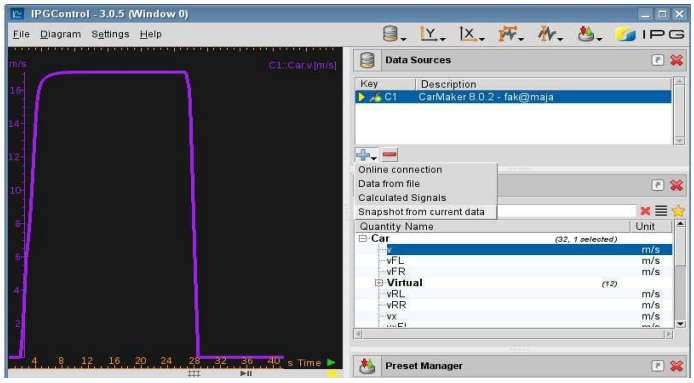



Figure 3.47: Left: Taking snapshot of results

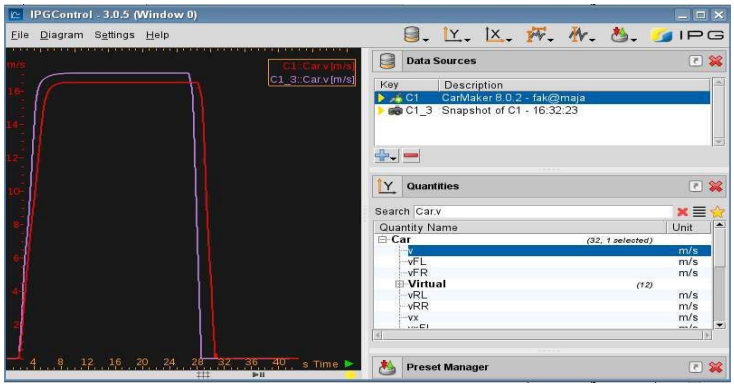



Figure 3.48: Compare the results

The second method to compare the simulation results is to save the result files first. The saved files are then loaded into IPGControl and are used in the same way as an online application (previous example).

## Exercise 6 – Step 2

- Gather Datasets:
  - Run a TestRun and save the results using the *Save* button on the Main GUI. Now, vary the TestRun, start the simulation and save the dataset (*Save* button on Main GUI).
- Add both Datasets:
  - In IPGControl, upload the last performed simulation. Choose *File > Add data from file* and load all simulation results you wish to compare. These results will be listed in the *Data Sources* box in IPGControl.

- Plot the Datasets:
  - Select the first simulation's results and add "Car.v". Do the same with the second dataset and change the color by right clicking on the legend in *Data Window* and clicking *Color of selected Quantity*.
  - Feel free to add further diagrams to the Data Window (*Diagram* > *Add Diagram*).

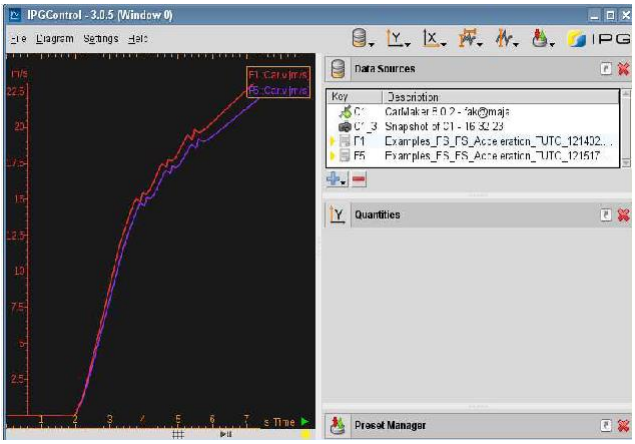



Figure 3.49: Load saved simulation results and compare them. The color of the graphs are changed to purple and red.

To become familiar with these features, complete the following exercise:

### Exercise 6 - Step 3

- Add the TestRuns to IPGControl:
  - Load the TestRuns result files via *File* > *Add data from file* (in IPGControl GUI) and browse to FormulaCarMaker/SimOutput/<hostname>/<date>/<TestRun-Results.erg>.
- Create two *Data Windows* with two diagrams:
  - Open a new Data Window by *Window* > *New Data Window* in IPGControl and add a second diagram (*Display* > *Add Diagram in Data Window*).
- Plot the normal driver velocity, acceleration and lap time to Data Window#1:
  - Click in the upper diagram and visualize the normal driver TestRun results by selecting them in *Data Sources* (IPGControl GUI) and plot "Car.v" for the car velocity and "Car.ay". Click in the lower diagram and add "DM.Clutch".
- Plot the aggressive driver velocity, acceleration and lap time to Data Window#2:
- Change the colors in both windows:
  - Activate the plotted UAQ by using LMB and open the context menu by right-clicking upon the variable name in the diagram. Assign *Color of selected Unit* > *Red* for velocity and *Color of selected Unit* > *Blue* to acceleration.

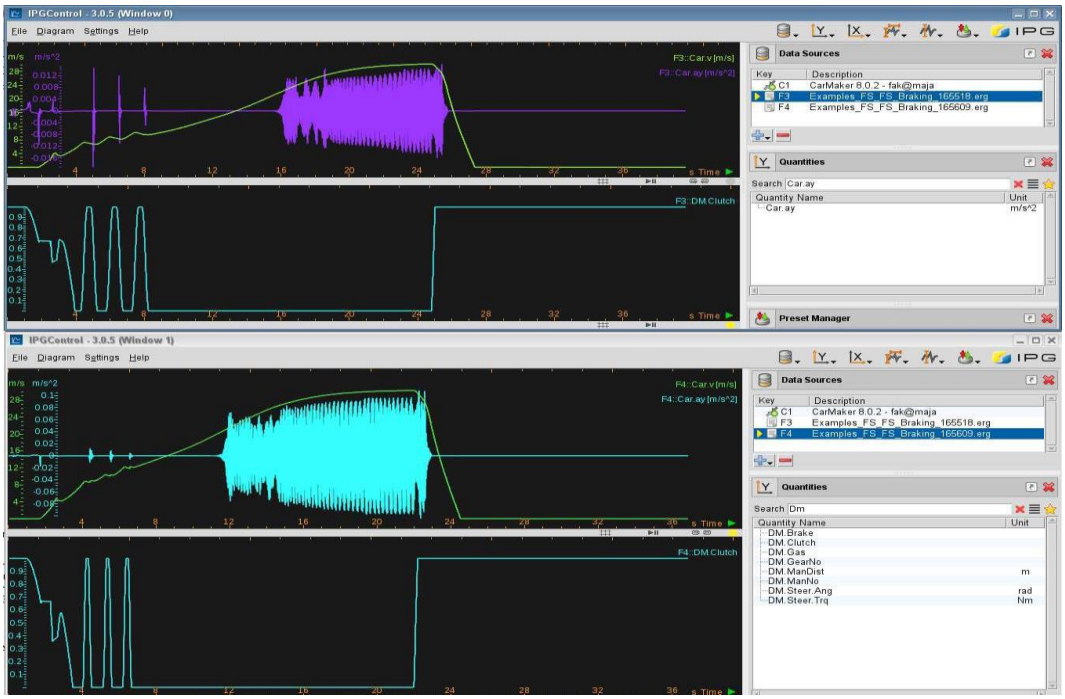


Figure 3.50: Comparison between *normal driver* (top) and *aggressive* driver (bottom)

### 3.3.3 Exporting Simulation Results

CarMaker also offers different file formats for the export of simulation results. To do so, wait until a TestRun is finished and follow the steps listed below.

- Right click anywhere in the *Data Window* within IPGControl and select *Export to File*.
- In the popup window select a file format, choose a file path and enter a feasible file name.
- Confirm with *Save.*

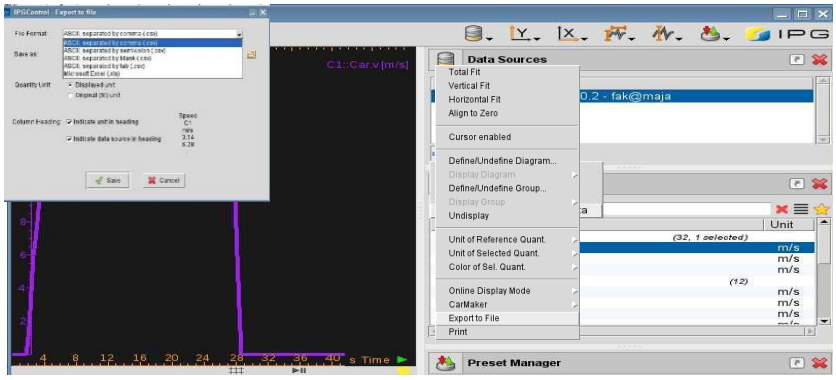


Figure 3.51: Export simulation results via right click in the *Data Window*

### 3.3.4 Print Diagrams

In IPGControl you can print plotted diagrams via the *Print* option on the RMB context menu. In the pop up window, enter a diagram title and confirm by clicking *Print.* IPGControl generates a preview of the diagram which can be sent to a local printer by *File* > *Print*.

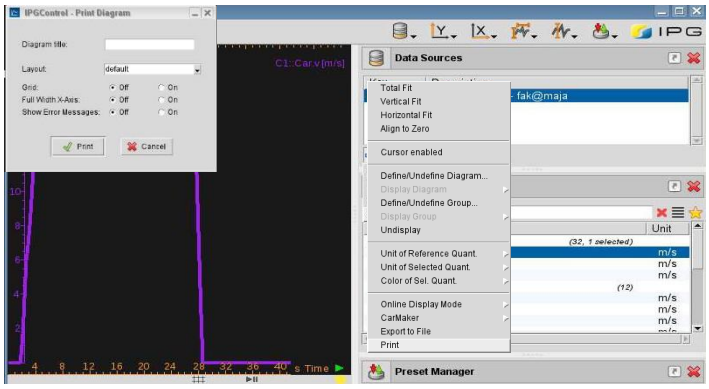



Figure 3.52: Print diagrams

## 3.3.5 Post-processing with MATLAB

To work with simulation results from CarMaker in MATLAB, a command is given to convert the .erg files to vectors used within MATLAB:

```
variabel1=cmread('.../SimOutput/<NameOfYourPC>/<Date>/<NameOfYourTestRun-Time.erg')
```

Instead of typing the path to your result file, simply hit *Enter* after "cmread". With this, an explorer pops up which prompts you to select the desired results file.

The following script demonstrates the procedure of how to load the .erg files in MATLAB, which should be located in the "src_cm4sl" folder of your Project Folder:

```
variabel1=cmread('...\SimOutput\<NameOfYourPC>\<Date>\<NameOfYourTestRun-Time.erg')
% Plot
figure
% subplot (2,1,1)
hold on;grid;
plot(variable.Time.data, variable1.Car_v.data, 'b');
legend('Vehicle Speed');
ylabel('Time');
title('Speed vs. Time')
```

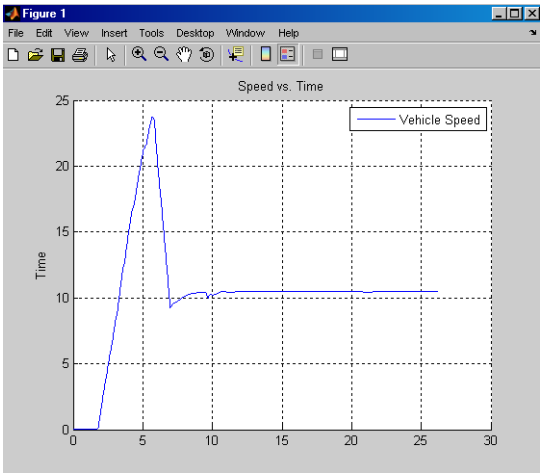



Figure 3.53: Plotting in MATLAB

If the command "cmread" cannot be found by MATLAB please ensure that the script "cmenv.m" from the *src_cm4sl* folder of your CarMaker project directory was ran successfully. If not, you may have to adapt the path to the CarMaker installation folder in the “cmenv.m” script.

Chapter 4

# Getting Started with IPGKinematics

IPGKinematics is a program designed to simulate a vehicle's axles upon a kinematics test bench and can be used to calculate the kinematics, steering kinematics and elastokinematics for a variety of suspension systems.

Each configuration is modeled as a multibody system using MESA-VERDE and is then converted to a library. Each library has its own Graphical User Interface (GUI) to suit the suspension's configuration. If desired, IPGKinematics can generate all the necessary parameter files required of a vehicle simulation software such as CarMaker.

For the simulation of high-performance competitive vehicles, such as Formula Student cars, the effects of compliance and elastokinematics become less prevalent due to generally stiffer suspension components and bushings. For this reason, it may be more relevant for a team to negate the effects of spring forces and torsion to produce a pure kinematics simulation, allowing suspension linkages to move freely throughout their range of motion, and therefore a 'Forces Off' simulation will be demonstrated. If you seek documentation to run a 'Forces On' simulation to consider the effects of external forces, please refer to Appendix B.

IPGKinematics includes a Reference Manual which contains further information. In the tool bar, click ? or *Help* > *Reference Manual*.

## 4.0.1 Opening IPGKinematics

As *IPGKinematics* is a stand-alone application it can be opened using the Windows Start menu. Alternatively, it can be started from the *File* menu of the CarMaker Main GUI.

It is advisable to create a new project directory as IPGKinematics simulations generate a large number of files. It is therefore recommended to use the FormulaCarMaker folder for CarMaker TestRuns and a standalone "Kinematics" folder for IPGKinematics simulations. The "Kinematics" folder should then also contain subfolders.

The vehicle and simulation settings are saved in editable *.kin files. Results can be saved in different file formats, some of which are used by CarMaker (e.g. *.skc - files). Therefore, for file management purposes we will create a new directory **outside** the FormulaCarMaker project folder called "Kinematics" in the following exercise.

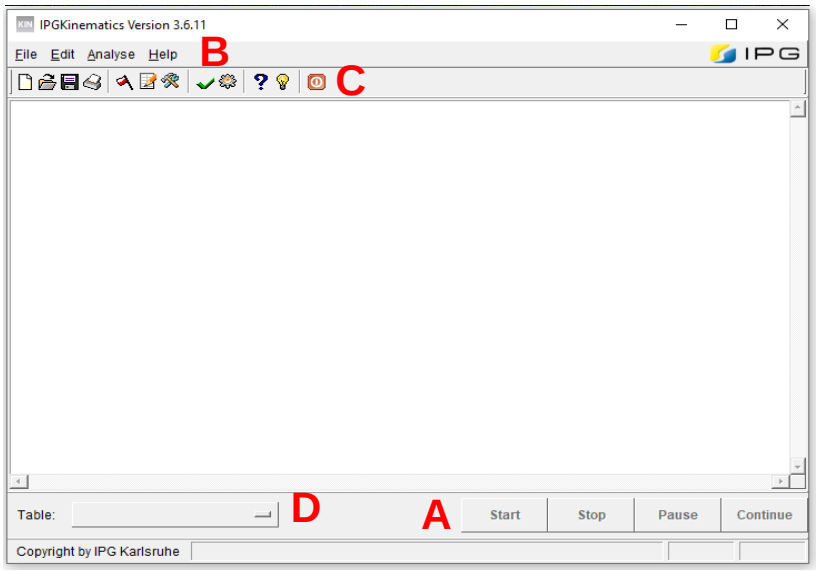



Figure 4.1: IPGKinematics GUI
A: Control buttons of simulation
B: Menu bar for editing parameters, accessing help
C: Tool bar for quick access to certain windows
D: Drop-down table to choose the presentation mode

**Exercise 7- Step 1**

- Create a new working directory for IPGKinematics:
  - Before starting IPGKinematics we want to create a new folder hierarchy.


Figure 4.2: Simulation models in IPGKinematics

- Open IPGKinematics via *Main GUI* > *File* > *IPGKinematics*.

## 4.0.2 Configuration of a Double Wishbone Axle

Before creating a new model, you must select the type of axle you wish to simulate. As Formula Student cars generally use a double wishbone system, this configuration will be chosen. The parametrization of the axle is done within the *Vehicle Data* submenu. This is where all general specifications for kinematics, buffers, masses and geometric are input.

The following example aims at providing a brief insight into the several submenus and analysis tool. The individual parameters will be discussed in more detail in section 4.2.

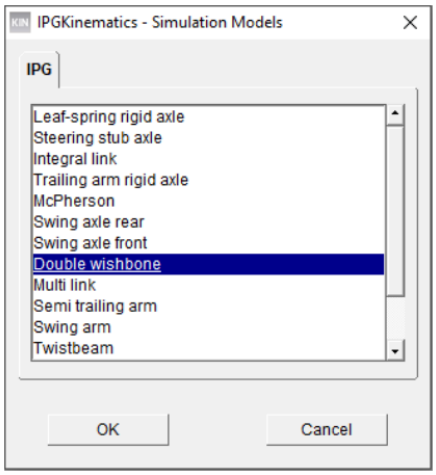



Figure 4.3: Simulation models in IPGKinematics

**Exercise 7 - Step 2**

- Create a new double wish bone model:
  - In the IPGKinematics GUI select: *File* > *New Model* > *Double Wishbone*.
- Save the model in the previously created "firstTutorial" folder as "doubleWishBone.kin"

# 4.1 Coordinate System

Traditionally, the optimization of a vehicle's dynamics and performance required many time consuming and expensive real world tests to be completed. Vehicle simulation enables the user to carry out the bulk of these tests within a virtual environment which saves a lot of cost and effort. However, this is only feasible if the results of the simulation replicate those produced by the real vehicle. For this reason, it is essential to achieve a high degree of affinity between the virtual model and reality which is achieved through accurate parametrization. After successfully producing a first simulation, the model must be validated by means of test runs and measurements which is both a complex and iterative process.

The following chapter demonstrates an approach of how to parametrize a model efficiently. The parameters used will be explained in detail and a few examples will be given as to how these values are determined. As the development of every vehicle model begins with the suspension, the front and rear axles will be modelled first.

At the beginning of a vehicle's development process the coordinate system must be defined. In IPGKinematics, two different coordinate system options are available:

- Coordinate system in accordance with DIN 70000:

  The x-axis is defined in the forward direction of travel with the y-axis to the left and the z-axis pointing upwards.

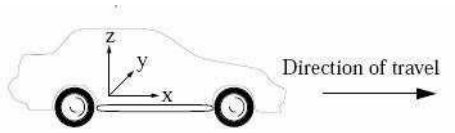



Figure 4.4: Coordinate system in accordance with DIN 70000 [KIN07]

- Coordinate system frequently used in the automobile industry:

  The x-axis is orientated opposite to the direction of travel, towards the back, with the y-axis to the right and the z-axis pointing upwards.

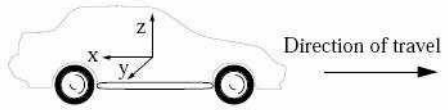



Figure 4.5: Coordinate system used in the automobile industry [KIN07]

Both of these coordinate systems are available in IPGKinematics but will only apply to data input. For calculations, the data will be translated into the coordinate system according to DIN 70000. The output of the results will also correspond to the DIN 70000 coordinate system. To avoid interpreting the results using the wrong coordinate system, it is recommended to work with DIN 70000 from the start. Another advantage is that this coordinate system can also be used in CarMaker as it is based exclusively on DIN 70000.

It is left to the user to decide where the origin is placed. However, it is crucial that all of the input data refers to the same origin. The center of gravity should not be chosen as its exact location is unknown and thus the results will be inaccurate. It is also helpful to use the same origin for both tools, IPGKinematics and CarMaker, to prevent the need to convert numerous coordinates. Furthermore, it is important to place the origin in the y-center of the vehicle as the geometrical input is only completed for the left half of the axle - the right side is mirrored automatically by the program.

As per the explanation above, the following convention applies to both IPGKinematics and CarMaker in the course of this document:
Coordinate system: DIN 70000

- Origin in x direction: 500 mm behind the rear axle
- Origin in y direction: in the center of the vehicle
- Origin in z direction: on the road surface

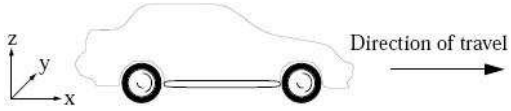



Figure 4.6: Coordinate system used in the following [KIN07]

# 4.2 Input Data

## 4.2.1 Simulation Control

This dialog controls the simulation parameters.

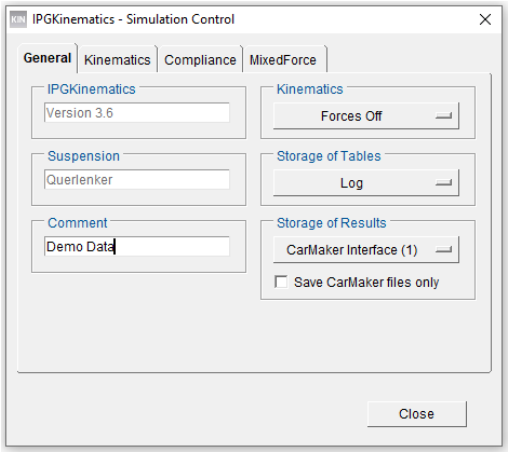



Figure 4.7: The Simulation Control window

### General

**Kinematics** Here you should select *Forces Off* from the *Kinematics* drop-down menu. *Forces Off* means that there are neither inertial nor spring forces - only pure kinematics are calculated.

**Storage of Tables** The options for the output of the calculated tables. *Off* - tables will not be output as a file. By selecting *On*, all tables will be output within a file and with *Log*, all tables will be displayed upon the screen as well as producing an output file.

**Storage of Results** Only the result files required for graphical output are stored when *On* is selected. This is the option which should be chosen when defining the axle. Otherwise, the entire compliance model for the use in CarMaker will be calculated with each simulation and needlessly increase the completion time.
To export the results to CarMaker you must choose one of the *CarMaker Interface* options:

- *CarMaker Interface (1)* provides linear compliance model;
- *CarMaker Interface (2)* provides a non-linear compliance model.

The second option will lead to increased simulation running times. However, the results do not become any more useful as Formula Student cars usually don't implement buffers. Therefore, the option *CarMaker Interface (1)* is preferred.

**Exercise 7 - Step 3**

- Edit the simulation settings:
  - Open the *Simulation Control* window using the *Edit* menu in the IPGKinematics GUI or by clicking on the corresponding icon.
  - *Kinematics = Forces Off.*
  - *Storage of Tables = Log.*
  - *Storage of Results = CarMaker Interface (1).*

## Kinematics

This tab defines the procedure according to which the movement of the wheels is calculated.

**Parallel Kinematics** Both wheels of an axle will move in the same direction upon compression if activated (*On*).

**Reciprocal Kinematics** This option will move the axle's wheels in opposite directions. To understand the processes of the different options, please take a look at the Reference Manual.

**Steering Kinematics** This is used only for the front axle to calculate steering parameters. It simulates rack-and-pinion steering which moves the front wheels accordingly.
*On (1)* should be selected as it does not consider the interaction between steering and reciprocal wheel travel. "Steering Forces" are used in certain situations such as a tire hitting a pavement.

**Compression and Distance Steering Rack** The maximum values for *Compression* and *Distance Steering Rack* should be kept small as to not unnecessarily increase computation time. This means that they should be only slightly out of the range of the achievable values of your car.

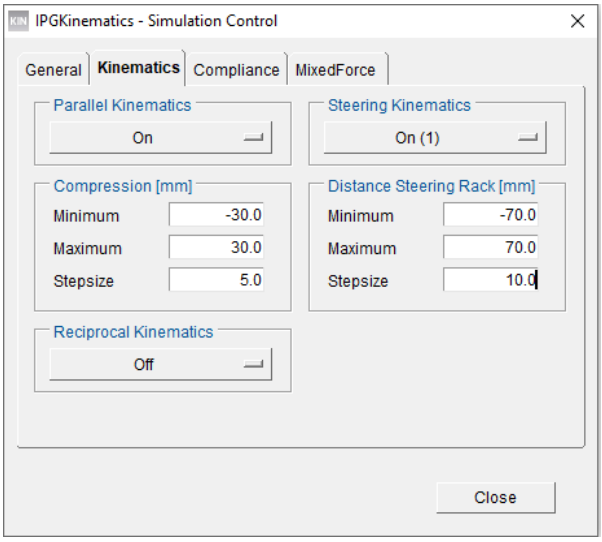



Figure 4.8: The Simulation Control Kinematics tab

### Exercise 7 - Step 4

- In the *Kinematics* tab set:
  - *Parallel Kinematics* = "On".
  - *Steering Kinematics* = "On (1)".
  - *Reciprocal Kinematics* = "Off".
  - *Compression* = "+/- 30 mm"; *Stepsize* = "5.0".
  - *Distance Steering Rack* = default values.

### Compliance and MixedForce

The last two tabs are used to apply external forces to the wheels in the longitudinal and lateral directions. If you are interested in the stresses of individual suspension components for FEM analysis, these options can be configured. For pure kinematics, external forces are deactivated which is why this option should be switched off. Further information can be found in the IPGKinematics Reference Manual (*Help > Reference Manual).*

**Exercise 7 - Step 5**

- In both the *Compliance* and *MixedForce* tabs, configure everything to *Off*.
- Save the model in the folder created earlier. Ensure that you choose *Input Data (.kin)* in the file type selection area.

## 4.2.2 Geometrical Control

Before starting the simulation you should review the specified inputs. For geometrical data, *Geometrical Control* gives an overview of all the positions of the defined points. You can access it via *Analyze > Geometrical Control* or by the corresponding icon.

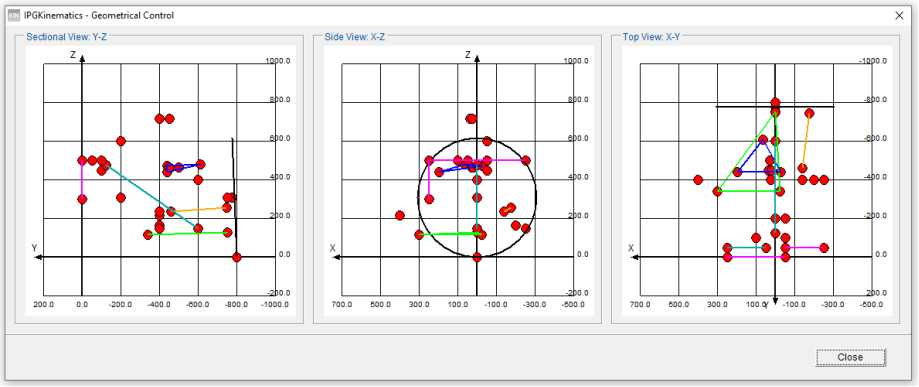



Figure 4.9: The Geometrical Control window

We will define the geometrical settings in the following chapter. However, as a trial you can check the pre-defined geometry by following the path outlined above. Next, click the *Start* button when ready to start the simulation. According to the *Simulation Control* parameters chosen you can view several arrays of calculations being performed. The calculation process can take quite a while depending on your input data. Once the simulation is finished (the *Start* button turns green) you should save the results immediately. In the later chapter '*Output Data'* you can learn how to analyze this data.

**Exercise 7 - Step 6**

- Perform and save the simulation:
  - Start the simulation.
  - Hit *File > Save* and overwrite the existing file to save your model with the output data of the simulation. Saving generates numerous files depending on the parameters set in *Simulation Control* window.

## 4.2.3 Vehicle Data

This subchapter describes the specific vehicle input parameters that are available for processing in the *Edit* > *Vehicle Data* menu. In the various tabs you must define general characteristics of the car and their positions. Although most of these inputs won't affect a 'Forces Off' simulation, it is suggested to input the reasonable values provided in case a 'Forces On' simulation is desired at a later stage.

### General

The *General* Tab contains some miscellaneous settings.

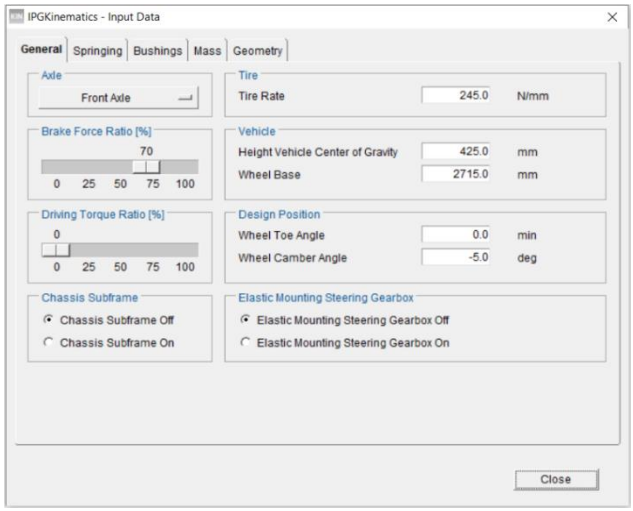



Figure 4.10: The Input Data General tab

**Axle** This option defines which axle will be modelled. The *Front Axle* option will include the calculation of various steering quantities. Note: If you choose *Rear Axle* no steering kinematics will be calculated even if they are defined in S*imulation Control*.

**Brake Force Ratio** In this field, the brake force ratio of the corresponding axle is defined. 0% means that the axle being modeling receives no braking power. This information is not required for a 'Forces off' simulation.

**Driving Torque Ratio** This defines the driving torque ratio of the axle. 0% means that the axle is not powered and 100% means that only this axle is powered. This value is necessary to calculate anti-squat and starting torque compensation angle values, but is not required for a 'Forces off' simulation.

**Model configuration (1)** For a Formula Student car you should select *Chassis Subframe off* as only the chassis supports the engine and lower suspension components (i.e. there is no subframe).

**Tire** This field is used to define the vertical stiffness of the tire (in N/mm). As the tire is modeled as a linear spring, a tire spring coefficient must also be entered. This spring coefficient is a function of:

- tire pressure
- tire tread
- tire height to width ratio
- driving speed
- rolling characteristics of the tire

To negate the effects of tire deflection, it is recommended to enter a high value such as 5000 N/mm.

**Vehicle** In this section the height of the vehicle's center of gravity and its wheelbase are entered. The wheelbase is used to calculate the turning track diameter and turning circle. The height of the center of gravity is crucial to compute the lever of the roll axis. To determine the center of gravity for your car, please refer to *section 5.1.1 Vehicle Body*, pg. 76.

**Design position** Here, the static wheel toe and camber angles are defined. The design position is the position of the car with only the axle load upon it. This corresponds to a wheel travel of 0 mm. Therefore, it must be established if the weight of the driver was included in the design position calculation or not. If not, attention must be paid to various settings such as the location of the center of gravity, camber and toe angles etc. All of these should be measured or calculated **without** the driver's weight.

**Elastic Mounting Steering Gearbox** This offers the option to model the rack-and-pinion steering elastically. However, as compliance will not be considered with 'Forces Off' this feature should not be enabled.

**Exercise 7 - Step 7**

- Select a rear axle configuration and set:
  - *Tire Rate* = "5000 N/mm"
  - *Height Vehicle Center of Gravity* = "300 mm"
  - *Wheelbase* = "1600 mm"
  - *Wheel Toe Angle* = "0 min"
  - *Wheel Camber Angle* = "-3 deg"
  - *Chassis Subframe* = "Off"
  - *Driving Torque Ratio* = "100%"
  - *Brake Force Ratio* = "30%"

Table 4.1: Settings for the exercise

| Variable | Value |
|---|---|
| Tire Rate | 200 N/mm |
| Height Vehicle Center of Gravity | 300 mm |
| Wheelbase | 1600 mm |
| Wheel Toe Angle | 0 min |
| Wheel Camber Angle | -3 deg |
| Chassis Subframe | off |
| Driving Torque Ratio | 100 % |
| Brake Force Ratio | 30 % |

## Springing

This includes the parameters for springs and a stabilizer bar, as well as the values required for wheel suspensions featuring pull rod/push rod actuation.

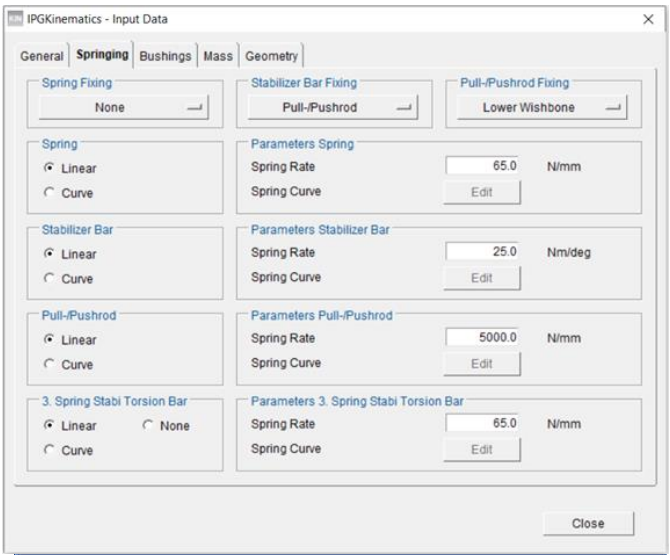



Figure 4.11: The Springing tab

**Spring Fixing** Defines which component the spring is fastened to. It is assumed that the "Spring Fixing" is on the suspension side. The option *None* is chosen in case of a pull-/pushrod.

As ‘Forces Off’ does not consider inertial nor spring forces, many of the settings upon this tab merely require a non-zero value to prevent a computational error.

**Stabilizer Fixing** In IPGKinematics, you have the option to mount the stabilizer to either the lower wishbone, wheel carrier or the pull-/pushrod. If you have anti-rollbars which are attached to the rocker, you should select the option pull-/pushrod. By choosing this, IPGKinematics will consider a ‘T-Bar Stabilizer’, otherwise a ‘U-Bar’ is used.

*This parameter does not affect ‘Forces Off’ kinematics,* however you should select pull-/pushrod.

**Parameter Spring/ Stabilizer/ /Pull-/Pushrod /Torsion Bar** *These parameters do not affect ‘Forces Off’ kinematics,* however non-zero values should be input to prevent a computational error. These stiffness values are used to consider compliance effects of various suspension compnents, which does not concern a pure kinematics simulation.

### Exercise 7 - Step 8

- Set the fixings as follows:

Table 4.2: Settings for the exercise

| Variable | Value |
|---|---|
| Spring Fixing | None |
| Stabilizer Bar Fixing | Pull-/Pushrod |
| Pull-/Pushrod Fixing | Lower wishbone |
| 3rd Spring Stabi Torsion Bar | None |
| Spring Rate | Linear, 23.9 N/mm |
| Torsional Spring Rate | Linear, 40 Nm/deg |
| Parameters Pull-/Pushrod | Linear, 5000 N/mm |

## Bushings

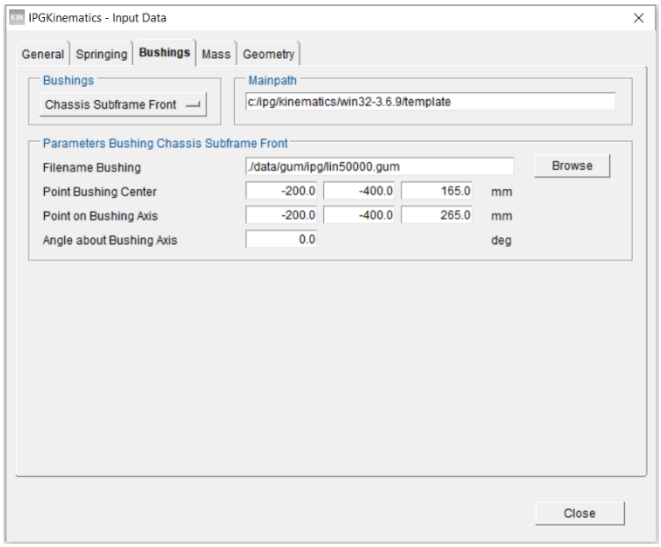



Figure 4.12: The Bushings tab

As compliance is not considering during a 'Forces Off' simulation, the specification of bushings has no effect upon the results and will therefore be modelled as rigid.

**Bushings** This drop-down menu lets you choose from various bushings according to the selected suspension model. When the *Bushings* field is selected, you can switch between the individual bushings within the suspension system. The contents of the tab are then updated accordingly and the parameters for the selected bushing are displayed. In the case of a double wishbone model there should be 9 different bushings present.

**Mainpath** Specification of the main path for bushing files. The purpose of the *Mainpath* is to eliminate the need to specify the complete path and file name for each bushing. The path must end with "/". This is then used by the program to establish the path to the input files for the bushings.

**Parameters Bushing .** Control parameters for selecting the bushings. All parameters for a bushing are entered into this field.

To simulate nonexistent bushings in IPGKinematics they are defined as extremely rigid (E = 5000 N/mm$^2$ = 5 GPa). This stiffens the bushings in all directions. Bushings only need to not be parallel to connected members for a 'Forces Off' simulation. I.e., the cross product of the connected member and bushing axis should not be zero. For a 'Forces On' simulation the Bushing axis should be aligned with the general axis of rotation of the connected member. I.e., the cross product of the connected member and bushing axis should be as large as possible. Further information can be found in the IPGKinematics Reference Manual.

### Exercise 7 - Step 9

- Set the path for each Bushing:
  - Since the suspension geometry should be defined first, we have to make sure the path (defined in *Mainpath*) is set to the corresponding directory on your workstation (e.g. C:\IPG\kinematics\version\template - This is not your "Kinematics" project folder).
- For a 'Forces Off' simulation, ensure bushing rotation axes are not parallel to the connected member (i.e., cross product does not equal 0).

## Mass

As no mass forces will be considered during the ‘forces off’ kinematics calculations, these values can be left as their defaults. (‘zero’ values will result in a computational error)

Note that all masses are in kilograms and the decimal point must not be a comma!

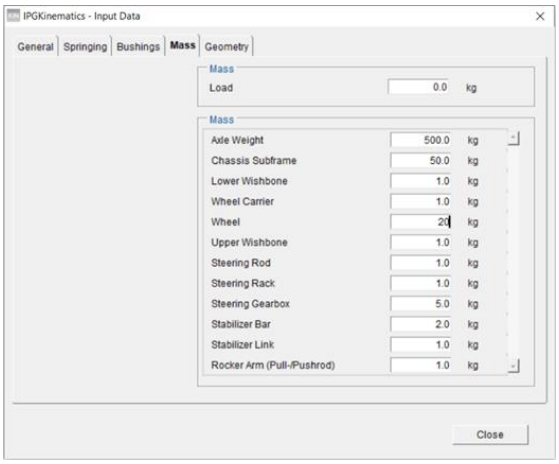



Figure 4.13: The Mass tab

**Load** This mass is only used if you want to add a specific load that does not belong to the vehicle itself. It is usually set to 0, as you do not wish to simulate additional load in racing cars!

**Axle Weight** This is the load on the front/rear axle minus the single masses listed below in Figure 4.15.

**Chassis Subframe** The chassis subframe mass can be set to any value except 0 (we have chosen the option *Chassis Subframe Off* in the *General* tab so the mass is not considered in the calculation).

**Other Masses** These depend on the chosen type of axle. More information can be found in the IPGKinematics Reference Manual (*Help > Reference Manual*).

The appearance of the GUI does not differ between front axle and rear axles. Please use the default values for components which are not relevant. Entering a value of 0 could cause arithmetic errors to occur.

### Exercise 7 - Step 10

- Set the following values for the masses:

Table 4.3: Settings for the exercise

| Part | Mass in kg |
|---|---|
| Load | 0 |
| Axle Weight | 140 |
| Chassis Subframe | 10 |
| Lower Wishbone | 0.5 |
| Wheel Carrier | 0.45 |
| Wheel | 10.5 |
| Upper Wishbone | 0.4 |
| Steering Rod | 0.25 |
| Steering Rack | 0.2 |
| Stabilizer Bar | 0.4 |

Table 4.3: Settings for the exercise

| Part | Mass in kg |
|---|---|
| Stabilizer Link | 0.08 |
| Rocker Arm | 0.08 |

## Geometry

The *Geometry* tab is used to define the hardpoints of various suspension components and its correct parametrization is critical to the accuracy of kinematics calculations.

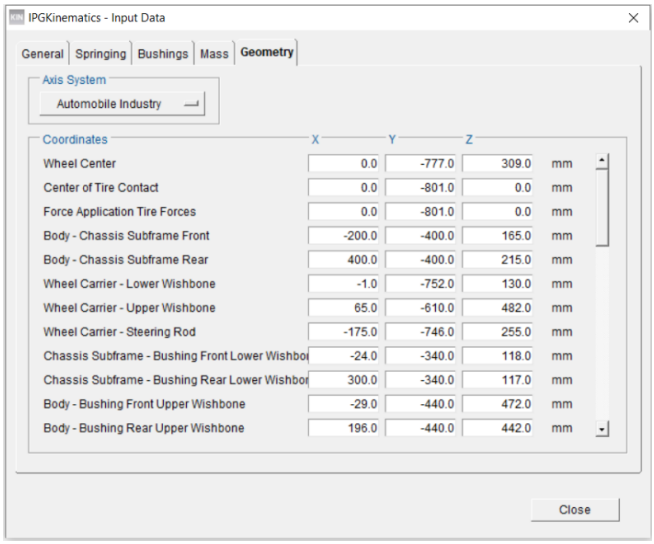



Figure 4.14: The Geometry tab

**Axis System** As explained in the section 4.1, ‘*Coordinate System*’, pg. 54, a coordinate system in accordance with DIN 70000 is used.

**Coordinates** The coordinates defined in this section are the most important of all. The calculations of the kinematics are based upon these values. For this reason, the values of these points should be determined carefully in order to generate an accurate as possible model.

Within the table you only define points for the left half of the axle. IPGKinematics automatically mirrors the values for the other side to create a complete axle.

Populate the Geometry tab according to the below table. All other unlisted inputs can be any non-zero value during the parametrization process. For a ‘Forces On’ simulation, parameterization guidelines can be found in Appendix B.

Table 4.4: Brief description of common values used for Formula Student cars

| Parameter | Comment |
|---|---|
| Wheel Centre | - |
| Centre of Tire Contact | - |
| Body - Chassis Subframe Front | Can be set same as Chassis Subframe - Bushing Front Lower Wishbone. |
| Body - Chassis Subframe Rear | Can be set same as Chassis Subframe - Bushing Rear Lower Wishbone. |
| Wheel Carrier - Lower Wishbone | - |
| Wheel Carrier - Upper Wishbone | - |

Table 4.4: Brief description of common values in Formula Student

| Parameter | Comment |
|---|---|
| Wheel Carrier - Steering Rod | Not used by rear axle - will still require a non-zero value e.g., coordinates of Wheel Centre. |
| Chassis Subframe - Bushing Front Lower Wishbone | As the subframe is deactivated this corresponds to the mounting points of the lower wishbones to the body. |
| Chassis Subframe - Bushing Rear Lower Wishbone | As the subframe is deactivated this corresponds to the mounting points of the lower wishbones to the body. |
| Body - Bushing Front Upper Wishbone | - |
| Body - Rear Upper Wishbone | - |
| Body - Rocker Arm | - |
| Rotation Axis - Rocker Arm | Can be any point upon the rotation axis of the rocker arm. |
| Pull-/Pushrod - Wheel Suspension | - |
| Pull-/Pushrod - Rocker Arm | - |
| Spring Element - Body | The point at which the spring assembly is connected to the vehicle body / chassis. |
| Spring Element - Rocker Arm | The point at which the spring assembly is connected to the rocker. |

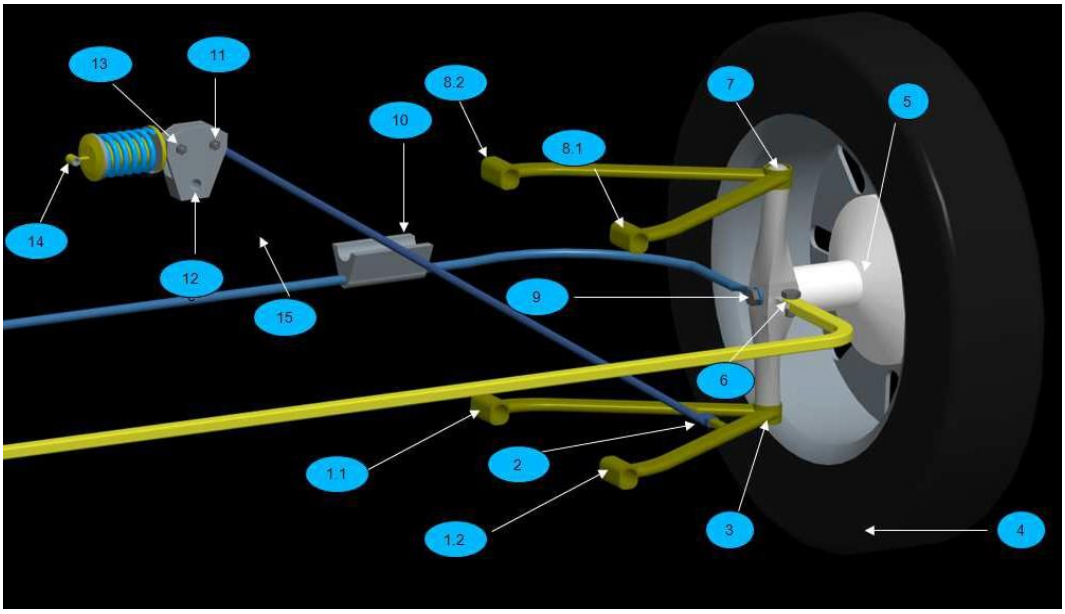



Figure 4.15: Overview of the kinematics points

Table 4.5: Listing of the geometry points in Figure 4.17

| Number | Item |
|---|---|
| 1.1/1.2 | Chassis Subframe - Bushing Front/Rear Lower Wishbone |
| 2 | Pull/Pushrod - Wheel Suspension |
| 3 | Wheel Carrier - Lower Wishbone |
| 4 | Center of Tire Contact |
| 5 | Wheel Center |
| 6 | Wheel Carrier - Steering Rod |
| 7 | Wheel Carrier - Upper Wishbone |
| 8.1/8.2 | Body - Bushing Front/Rear Upper Wishbone |
| 9 | Wheel Carrier - Stabilizer Link (Stabilizer Link - Stabilizer Bar) |
| 10 | Chassis Subframe - Stabilizer Bar |
| 11 | Pull-/Pushrod - Rocker Arm |
| 12 | Body - Rocker Arm |
| 13 | Spring Element - Rocker Arm |
| 14 | Spring Element - Body |
| 15 | Rotation Axis Rocker Arm |

# 4.3 Output Data

## 4.3.1 IPGGraph

The graphical analysis is based on the created .erg files and the tool *IPGGraph*. This allows you to plot multiple types of diagrams to display the calculated values. Open IPGGraph by clicking in the IPGKinematics GUI *Analyse* > *Graphical Analysis* or by using the corresponding icon.

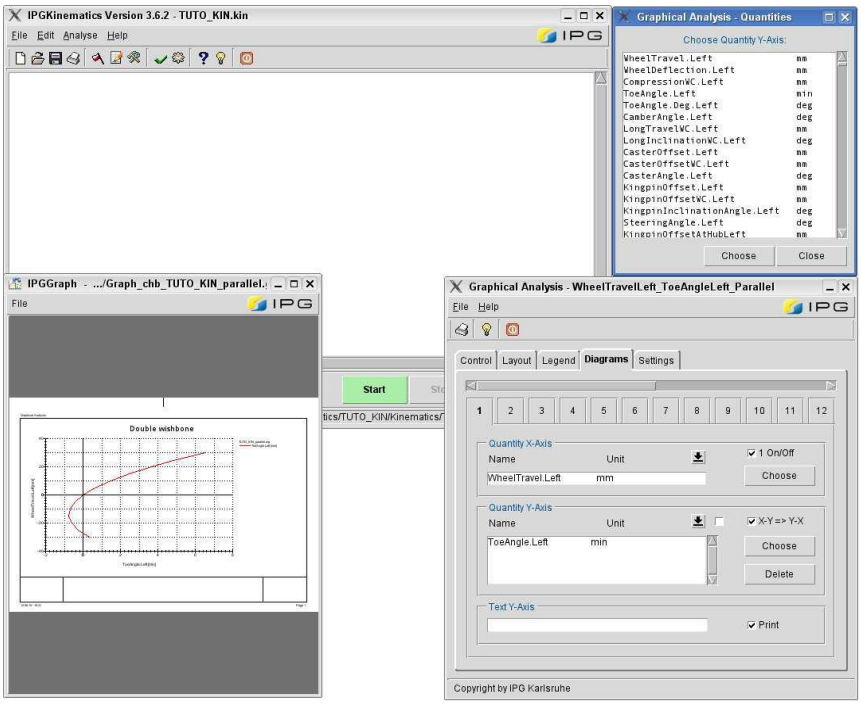



Figure 4.16: Graphical analysis in IPGKinematics

### Control

In this tab you can load the results you want to analyze.

**Choose Results** This field is for loading the file. It will usually be `NameOfYourKinFile_parallel.erg` for the parallel compression, etc.

**Choose Sequence** The *Choose Sequence* field lets you select a predefined diagram layout that will load pre-set diagram parameters for use with different sets of result files. If you are using IPGGraph for the first time, there will be no sequences available.

### Layout

This tab lets you set the orientation of the page, number of the created plots and other settings regarding appearance of the diagram.

### Legend

Here you can input information regarding the data within title blocks and document headers. You can also add a logo.

## Diagrams

In this tab you can define which parameters you'd like to plot. You can set up to 24 diagrams in one sequence. To switch to other diagrams, left click in the number panel (1,2,3,..).

**Quantity X-Axis** This is where the reference variable is chosen.

**Quantity Y-Axis** Here you choose one or more corresponding signals to plot against the reference variable.

**# On/Off** Lets you remove a particular diagram.

**X-Y =>Y-X** For inverting the X and Y axis, mark the checkbox.

## Settings

This tab contains general graph settings. Both the path to results and sequence should be set to their corresponding locations within your work folder.

IPGGraph will create the plots and open a viewer. If there are several pages, use the arrows on your keyboard to view them. Once the diagrams are displayed you can send them to a printer or to a file (.ps or .pdf format): click on *File* > *Print*.

To save the layout of your diagrams, generate a sequence by hitting the *Save* button in tin the *Choose Sequence* area of the *Control* tab and give it a suitable name. When you perform another simulation you can apply these layout settings by selecting this sequence.

The next exercise will give a short example on how to create a plot.

## Exercise 7 - Step 11

- Create a plot from a data set:
  - Open IPGGraph via *IPGKinematics > Analyse > Graphical Analysis*.
  - Make sure that *Path to Results* and *Path to Sequences* lead to your project files created earlier ("Kinematics/firstTutorial" and "Kinematics/Sequences"). If not, insert the correct path.
  - In the *Control* tab load the file "doubleWishBone_parallel.erg". Do not define any sequences yet.
  - Parametrize the options in *Layout* to have a single diagram in landscape orientation. Leave the other options to their defaults.
  - Feel free to add information witihin *Legend*.
  - In the *Diagrams* tab select quantities y(x) for:
    Tab 1) ToeAngle.Left (WheelTravel.Left)
    Tab 2) CamberAngle.Left (WheelTravel.Left)
- Print the plot and save the sequence:
  - Once all settings are configured you are ready to plot your diagram. To do so, click *File > Print* or the corresponding icon. Your result should look like this:

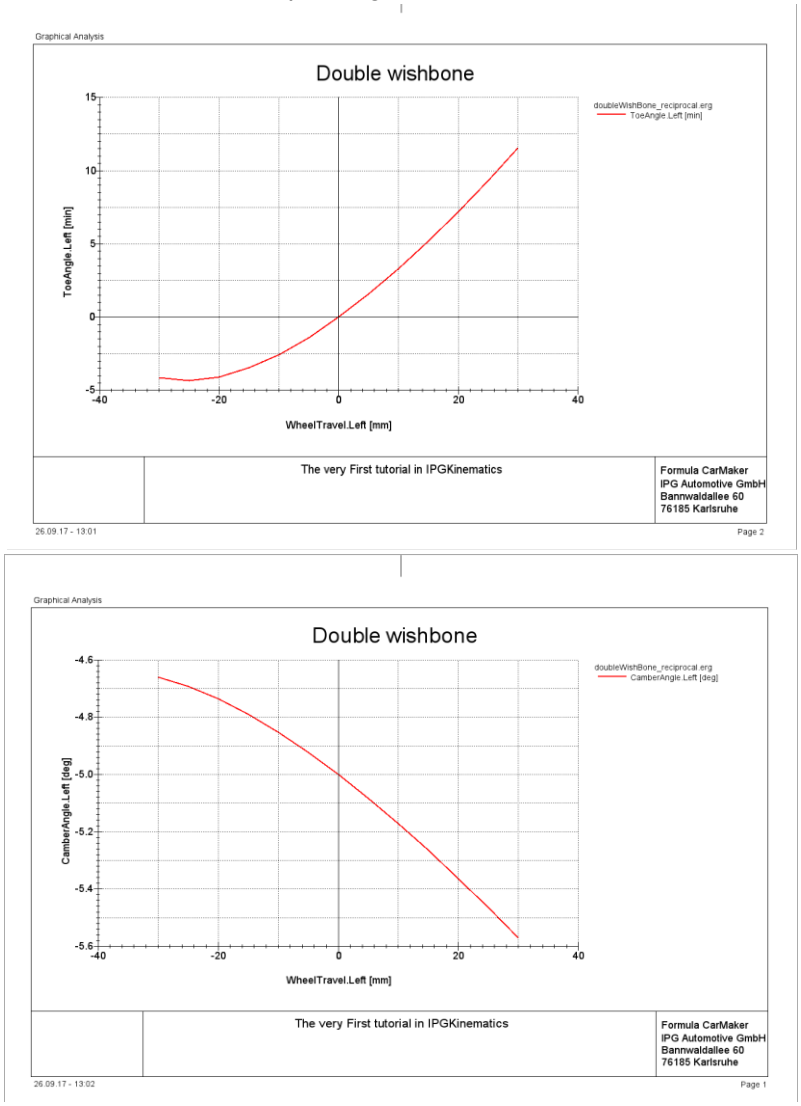



Figure 4.17: Graphical analysis using IPGGraph

- If your plot is satisfactory, save the sequence using *Save* in the *Choose Sequence* section of the *Control* tab.

## 4.3.2 Exporting Results to CarMaker

As explained previously, to export results to CarMaker you must enable a specific option: In *Simulation Control* > *Global* > *Storage of Results* select *CarMaker Interface (1).*

With this option, among the numerous data files generated by the simulation, one is called

- `NameOfResults_front.skc` (for a front axle)

  or

- `NameOfResults_rear.skc` (for a rear axle).

To use IPGKinematics results in CarMaker you have to copy those skc-files and paste them into the "<yourProjectDirectory>/Data/Chassis" folder of your CarMaker project directory, for instance:

“FormulaCarMaker /Data/Chassis”

We will learn how to use this file in the following chapter.

**Exercise 7 - Step 12**

- Copy the .skc files to "Data/Chassis":
  - Look into your "Kinematics/firstTutorial" folder for the *.skc files.
  - Copy and paste them to “FormulaCarMaker/Data/Chassis”

Chapter 5

# Preparing a Vehicle Dataset in CarMaker

## Introduction

There are two ways to prepare a completely new vehicle dataset.

The first one is to adapt a pre-existing, error free dataset step by step. The main advantage of this method is that the newly entered data can be verified easily. Knowing that the former TestRun worked without any problems, a driver adaption or a new simulation run can be used to indicate if there are any mistakes in the changes made.

The second method is to begin with an empty dataset. This method saves a lot of time during the parameterization process, however, the time gained can be easily lost during a complex debugging process as the user doesn't know where exactly the mistake is located. A few common error messages are described in section 6.3.1.

For electric cars there are essentially two different options to creating an electric drivetrain. Firstly, you can parameterize your electric powertrain directly in the CarMaker GUI. Again you have the choice to create your own powertrain or to modify one of the example vehicles (Rear Wheel Drive (RWD) or All Wheel Drive (AWD)) from the "FS_Generic_2021" Project Folder. The electric powertrain model in CarMaker provides the functionality to simulate one to four electric motors. Along with the motor and driveline characteristics, the operation strategy and the power supply can be parameterized directly within CarMaker.
To test your own controllers, it is possible to substitute or expand the predefined control units with MATLAB/Simulink models.

Secondly, you can use the CarMaker interface with MATLAB/Simulink. You can create your own electric drivetrain model in MATLAB/Simulink and deactivate the IPG powertrain model in the vehicle data set as shown in our example "FS_Generic_XWD_6.0". The basic idea of the FSE_RaceCar is that the generic powertrain is substituted with a modified electric powertrain modeled in Simulink. You are free to design a model with one, two or more electric motors.

Furthermore, the Simulink environment is a very convenient way to develop driver assistance systems such as a simple traction controller or an advanced Torque Vectoring model. Of course, this can be combined with the first approach of parameterizing the electric powertrain in the CarMaker vehicle data set, too.

# 5.1 General Vehicle Data Set

In the instance of preparing a vehicle dataset representative of a Formula Student race car it is recommended to follow the first method of parametrization as a predefined model (“FSC_RaceCar”) already exists. It was designed with the intent of providing characteristic values and therefore eradicate the need to start from scratch. Thus, you have an operating vehicle model with values within an illustrative range. All you need to do is to adjust the predefined values to fit your car. To open the *Vehicle Data Set* window click: *Parameters* > *Car*.

Note: Unless specified, all units are set to the International System of Units (SI). Accordingly, lengths are in meters and angles in radians (not degrees!).

**Exercise 8 - Step 1**

- Select the “FSC_RaceCar" in the Main GUI.

## 5.1.1 Vehicle Body

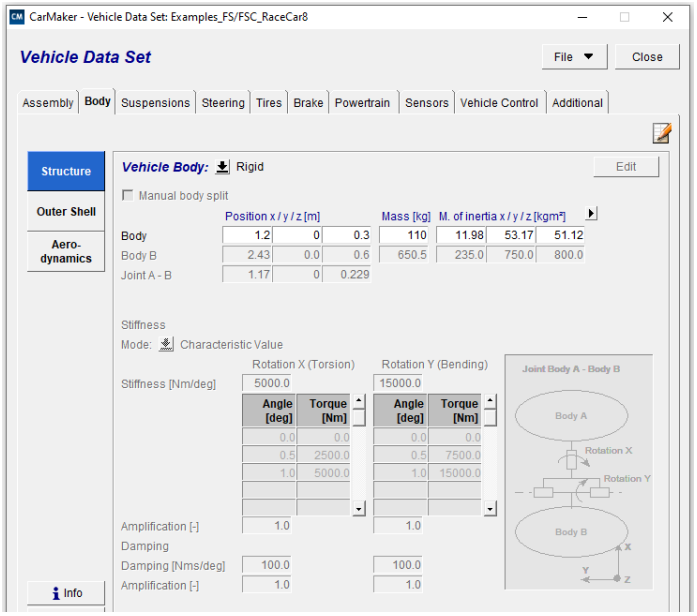



Figure 5.1: Vehicle Data Set - Vehicle Body

In this tab you can model the body as flexible if you select *Vehicle Body* > *Flexible*. This option allows to consider bending and torsion effects of the main body. Otherwise, the car is considered as infinitely stiff. By activating the option *Flexible*, the hidden input boxes turn active. In the upper portion you can separate the vehicle body into two sections: A and B. A refers to the front part and B to the rear. Both parts are defined by their own mass, center of mass and inertias.

Below, coefficients of torsional body stiffness in the X and Y directions can be defined. When choosing the mode *1D Look-Up Table* instead of *Characteristic value*, non-linear characteristics can be edited.

Please keep in mind, that only very exact (measured) values for a flexible body are useful, otherwise the use of the rigid body is recommended.

### Calculating Center of Gravity

The calculation of a vehicle's center of gravity is a very complex and challenging task. Even with the help of CAD tools, it is only possible when every body within a model is defined using the correct physical properties.

A much easier alternative is to measure the total center of gravity. To do this, the car must be brought to a completely horizontal position. To measure each axle load, one axle at time is placed upon a set of scales. Equating moments about the front axle, the horizontal position of the center of gravity can be determined as follows:

$$l_f = \frac{m_{V,r}}{m_{V,t}} \cdot l \qquad \textbf{(EQ 7)}$$

$$l_r = \frac{m_{V,f}}{m_{V,t}} \cdot l \qquad \textbf{(EQ 8)}$$

With

- l = wheelbase
- $l_f$ = distance between the center of gravity and the front axle
- $l_r$ = distance between the center of gravity and the rear axle
- $m_{V,f}$ = axle load at the front
- $m_{V,r}$ = axle load at the rear
- $m_{V,t}$ = total weight

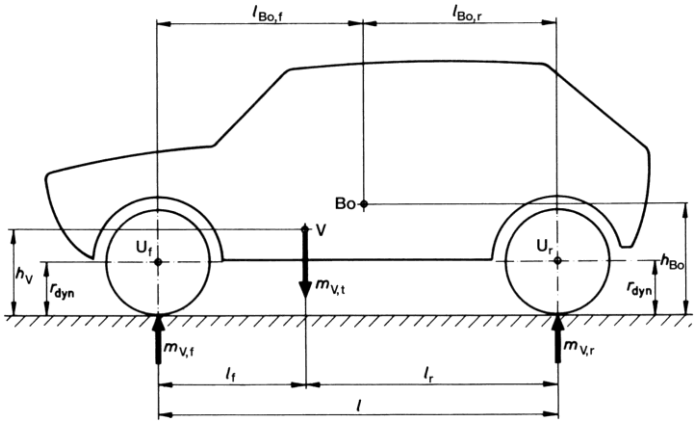



Figure 5.2: Locating of the vehicle center of gravity [RB00]

To evaluate the height of the center of gravity a second measuring setup is required. This involves one axle being raised by a height *h*. It is necessary to jack the vehicle as high as possible to avoid measuring faults. Additionally, the following issues should be considered:

- Rolling of the car should be prevented using wedges. Brakes should be released and the engine should be at idle. If this is not the case, bracing can occur and is likely to affect the correctness of the measurement.
- The wheels should be placed in the middle of the scales to prevent inexact measurements due to indifferent in force initiation points.

- The vehicle should be ‘race ready’ which means refueled, including the driver with their equipment.
- The suspension springs should be blocked to prevent deflection. Otherwise this could lead to measuring faults.
- To prevent tire deflection a high pressure should be attained.

The car’s slope angle is defined as:

$$sin\alpha = \frac{h}{l} \quad \textbf{(EQ 9)}$$

To determine the height $h_V$ of the center of gravity, $\Delta l_r$ is required (see Figure 5.3) because:

$$h_V = h'_V + r_{dyn} \quad \textbf{(EQ 10)}$$

$$h'_V = \frac{\Delta l_r}{tan\alpha} \quad \textbf{(EQ 11)}$$

$\Delta l_r$ can be calculated by equating moments about the front axle. Then, the height $h_V$ can be estimated with the following equation:

$$h'_V = \frac{l}{m_{V,t}} \cdot \frac{\Delta m}{h} \cdot \sqrt{l^2 - h^2} + r_{dyn} \quad \textbf{(EQ 12)}$$

The meaning of the parameters is shown in the figure below:

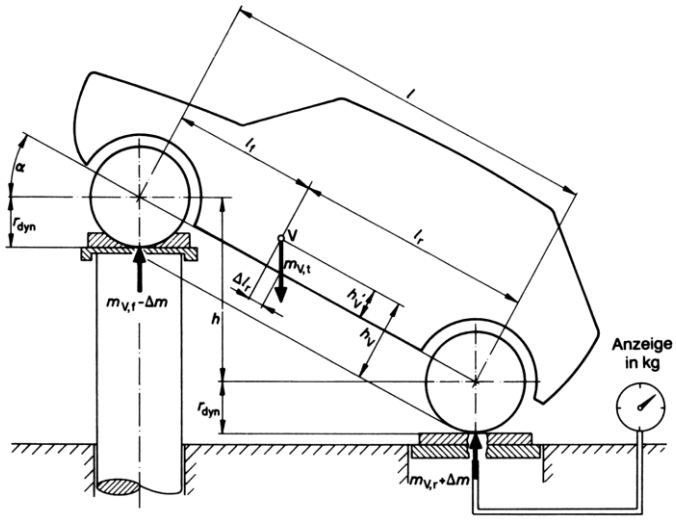



Figure 5.3: Determining the height of the center of gravity [RB00]

With this procedure, however, only the center of gravity of the entire vehicle can be measured. The many centers of gravity of single components required by CarMaker must be determined in a different manner. To locate the driver’s center of gravity the measurements explained above are carried out both with and without the driver. Afterwards, the center of gravity can be calculated through the equating of moments.

The same procedure enables the location of the engine's center of gravity to be evaluated. The determining of which for other required masses can be achieved by weighing each part individually. Beware, for half-sprung components such as wishbones, tie rods and drive shafts the weights are split evenly between the 'body' and the 'wheel carrier'.

**Moments of Inertia**

One method of estimating moments of inertia is to replace each assembly with a representative simple geometric body and calculate using known formulas. These approximations deliver acceptable results.

More exact values can only be gained with complicated and cost intensive experiments. Here, the moments of inertia are determined by the oscillation period of the respective part on a torsion pendulum in relation to the oscillation period of a reference body. For this reference body, a simple geometry is chosen so that its moments of inertia can be easily calculated for comparison. An even better possibility offers a test bench specially designed to determine the inertial tensor of four wheeled vehicles.

## 5.1.2 Bodies

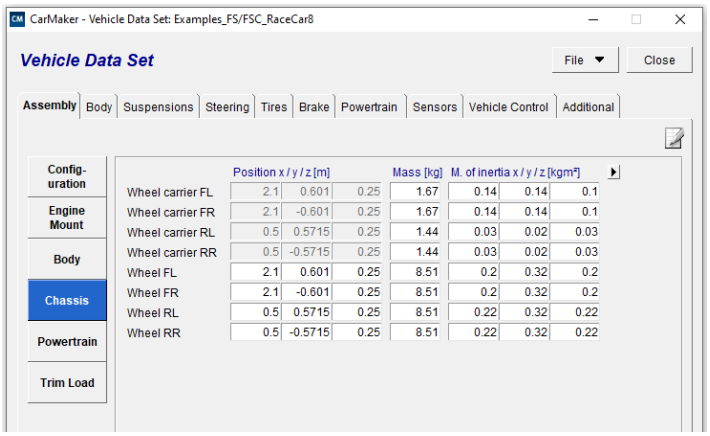



Figure 5.4: Vehicle Data Set - Bodies

Each simulation model is based upon the masses of individual bodies and their moments of inertia. Regarding the FS car, these masses are the suspension (unsprung masses) and the body (sprung masses). In CarMaker these two groups are divided into further subdivisions. The unsprung masses are classified into spinning (e.g. wheel, rim, wheel-hub) and non-spinning masses (e.g. wheel carrier, half of the wishbone and brake caliper). Do not forget to add the weight of the brake disk to the wheel mass! Additionally, note that the unsprung masses which are connected to the body should only be included using half of their weight as they are not entirely unsprung. The driver can be defined with the *Trim Load* entries.

**Positions** CarMaker calculates the car's movements in an axis system referred to as "Fr1". In the box *Origin Fr1* you can define a translation between Fr1 and your design axis system. You will then be able to describe positions directly using your own axis system.

## 5.1.3 Engine Mount

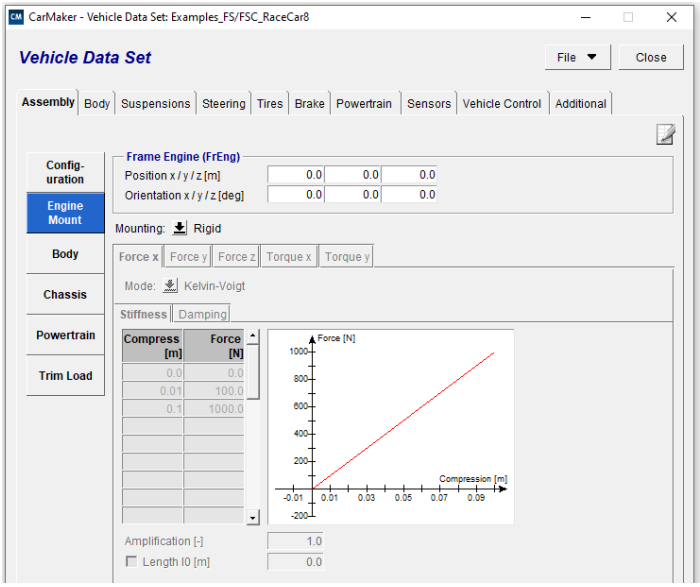



Figure 5.5: Vehicle Data Set - Engine

There are three methods to including the engine's mass in the vehicle data set. The easiest is to add the mass to the *Vehicle Body* masses, especially when you do not have any detailed mass information. If the position of the engine's center of gravity is known, and potentially the moments of inertia, you should define a *Trim Load* within the *Bodies* tab.

Lastly, you can enable an *Elastically mounted Engine* within the *Engine* tab and enter the required parameters. This helps to fine-tune the race car model to achieve a higher level of realness. More information about elastic mounting can be found in the Reference Manual.

### Exercise 8 - Step 2

- Edit the masses:
  - Increase the vehicle body weight to 220 kg and its center of gravity to 0.5 m above the ground (RigidBody, Vehicle Body A).
  - Then subtract 0.2 kg from the spinning brake disks (which are included in the wheels).
  - Keep the moments of inertia values the same.

## 5.1.4 Suspensions

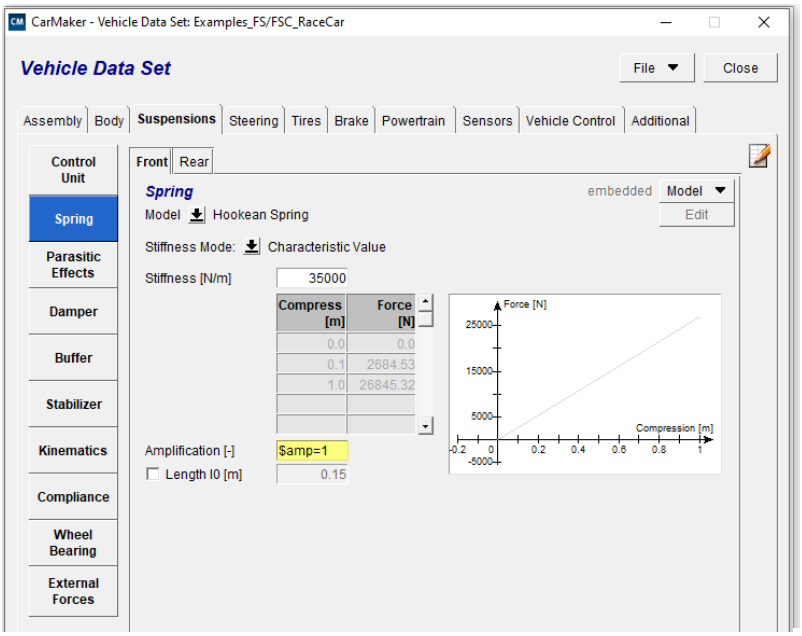



Figure 5.6: Vehicle Data Set - Suspensions

### Spring

The spring rates of the suspension system have already been defined in IPGKinematics. The same values should be entered here. The parameter $l_0$ is the actual free length of the spring in its unmounted state. You can let this value be calculated automatically by CarMaker which does some iterations during the determination of the car's static equilibrium position in the preparation phase ahead of every simulation. If the car's position on the road does not match with your real car, this value serves as an adjustment parameter.

**Exercise 8 - Step 3**

- Edit the spring:
  - Increase the front stiffness to 45 000 N/m and the rear stiffness to 52 300 N/m.
  - Leave the *Length* $l_0$ option unticked.

### Damper

Damper settings are mostly determined empirically using driver feedback and intuition. For this reason, accurate values for damper forces in the final setup are rarely known. Accordingly, a significant number of damper characteristics can only be determined via test runs on the real vehicle or through the use of specialized test benches. If both resources are available always opt for a test bench, as measurements on the real vehicle can be distorted by friction within the suspension system. Moreover, test benches enable a wide range of speeds (up to n = 200 $min^{-1}$) and travel (5 - 150 mm) to be tested. They can also directly determine the main value required by CarMaker: "damper force via speed".

**Exercise 8 - Step 4**

- Set the damper values to:

Table 5.1: Push values

| **Damper Front - Push:** | | **Damper Rear - Push:** | |
|---|---|---|---|
| velocity [m/s] | force [N] | velocity [m/s] | force [N] |
| 0.0 | 0.0 | 0.0 | 0.0 |
| 0.125 | 525.80 | 0.125 | 566.25 |
| 0.25 | 788.70 | 0.25 | 849.37 |

Table 5.2: Pull values

| **Damper Front - Pull:** | | **Damper Rear - Pull:** | |
|---|---|---|---|
| velocity [m/s] | force [N] | velocity [m/s] | force [N] |
| 0.0 | 0.0 | 0.0 | 0.0 |
| 0.125 | 1183.06 | 0.125 | 1274.06 |
| 0.25 | 1774.58 | 0.25 | 1911.09 |

## Buffer

The springs used in Formula Student cars are generally so stiff that only small spring displacements are possible. However, with push and pull-rod actuated systems much larger deflections are feasible. For this reason, and to prevent elasticity from making the car softer, buffers are not commonly used on these cars.

To implement this in CarMaker, two steps are required. One is to assign a large stiffness to the buffers which considerably reduces the effects of elasticity. The next is to set their positions very high so that they aren't activated during typical testing. The length *tz0* defines the limit above which the wheels are prevented from deflecting. For a clear description, refer to the Reference Manual (*Help > Reference Manual*) Chapter 10.9.5 '*Suspension Force Elements'.*

## Stabilizer

The stabilizer parameters have previously been explained in section 4.2.3 '*Parameter Stabilizer'*, pg. 62. In CarMaker, the same stiffnesses must be used as in IPGKinematics. Beware: CarMaker uses a linear spring rate [N/m] as opposed to IPGKinematics which uses a torsional spring rate [N/deg]. The following will show you how to convert the spring rate.

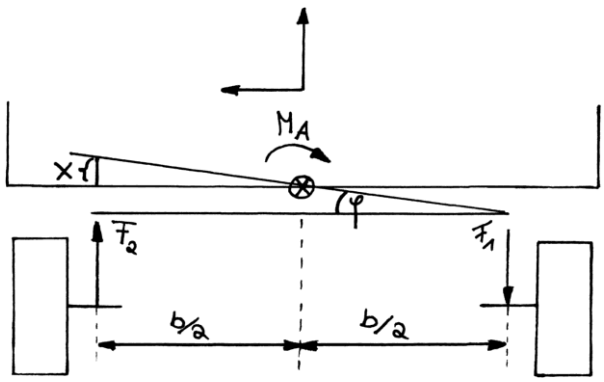



Figure 5.7: Converting the stabilizer bar spring rate

The translation is based on the equating of moments about the pivotal point and the balance of forces in z-direction:

$$F_1 \cdot \frac{b}{2} + F_2 \cdot \frac{b}{2} + M_A = 0 \quad \text{(EQ 13)}$$

$$F_1 = F_2 \quad \text{(EQ 14)}$$

Solving and substituting gives:

$$F_1 = \frac{M_A}{b} \quad \text{(EQ 15)}$$

The rolling moment $M_A$ is calculated using:

$$M_A = c_\varphi \cdot \varphi \quad \text{(EQ 16)}$$

With $M_A$ being inserted:

$$F_1 \cdot b = c_\varphi \cdot \varphi \quad \text{(EQ 17)}$$

Considering:

$$F = c_x \cdot x \quad \text{(EQ 18)}$$

and

$$tan\varphi = \frac{2x}{b} \cong \varphi \quad \text{(EQ 19)}$$

The stabilizer bar spring rate is therefore:

$$c_x = c_\varphi \cdot \frac{2}{b^2} \quad \text{(EQ 20)}$$

## Kinematics, Compliance and External Forces

Each of these fundamental parameters were calculated using IPGKinematics. To import this data into CarMaker, load both of the skc-files for the front and rear axle (see section 4.3.2).

**Exercise 8 - Step 5**

- Load the generated skc-files:
  - If you have generated an IPGKinematics dataset for the front axle as well as for the rear, load both via *Vehicle Data Set* > *Suspension* > *Kinematics* > *SKC- File*.

## 5.1.5 Steering

The steering system consists of a mechanical module which defines the ratio between the steering wheel angle (or steering wheel torque) and the steering rack displacement. Optionally, a power steering unit can be added by selecting the *Pfeffer Steering Model* which also considers frictional losses within the steering system.

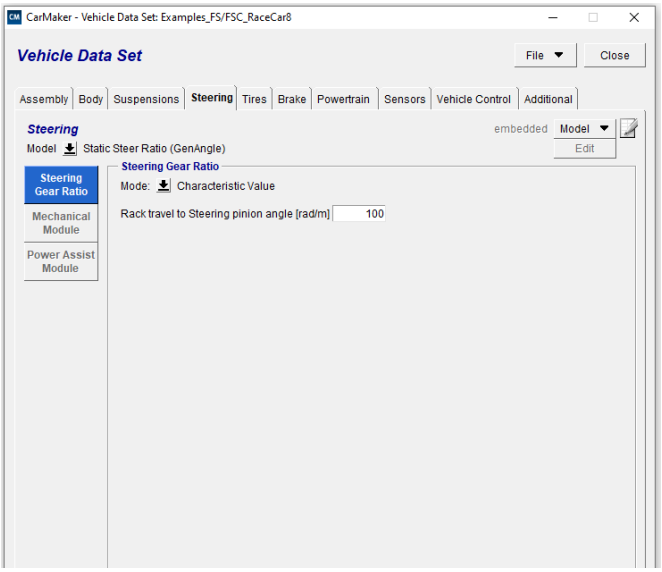



Figure 5.8: Vehicle Data Set - Steering

### Steering Model

#### Static and Dynamic Steer Ratio

To simulate a steered axle CarMaker requires only the ratio between steering wheel angle (in radians) and steering rack travel (in meters). The ratio does not have to be static: non-linear maps are available under the *Mode* selection menu.
*Note: The ratio between steering rack travel and wheel angle is a part of the kinematic data generated in IPGKinematics and will be transferred to CarMaker using the skc files.*

#### Pfeffer with Power Steering

You can also implement the *Pfeffer Power Steering* model which is a very detailed representation, including friction loss effects, elasticities and power steering.

**Mechanical Module**

This model consists of a mechanical module which includes all mechanical components (transferring torque from the steering wheel to the tie rods).

**Power Assist Module**

A power assistance module e.g. hydraulic (HPS), electrical (EPSc and EPSapa) can also be parameterized.

In the User's Guide (*Main GUI > Help > User's Guide*), Chapter 10.13 '*Parameterization: Steering System'* where you will find more information about the different models.

### 5.1.6 Tires

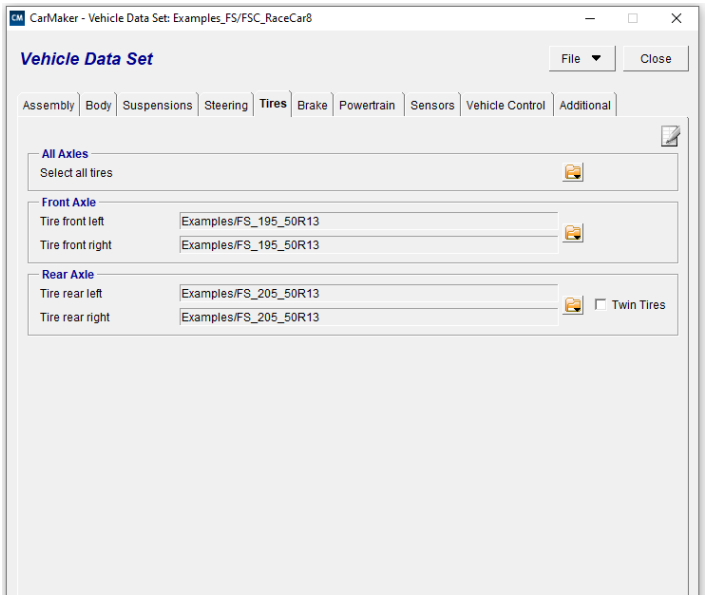



Figure 5.9: Vehicle Data Set - Tires

Here, the tires used by the car can be selected. The tire attributes belong to the vehicle parameters and will be loaded automatically by selecting a car. Optionally, you may overrule these tire files by selecting an alternative in the CarMaker Main GUI. Learn more about how to import your own tire data in ‘*Creating a Tire Dataset Using IPGTire’*, pg. 119.

## 5.1.7 Brake

The brake system in CarMaker is divided into two parts: The *Brake Control unit* and the *Brake System*. Whereas the first is an interface for controllers, the Brake System represents the physical brake unit. The pre-implemented brake unit is a hydraulic brake system with a control unit that can also support regenerative braking.

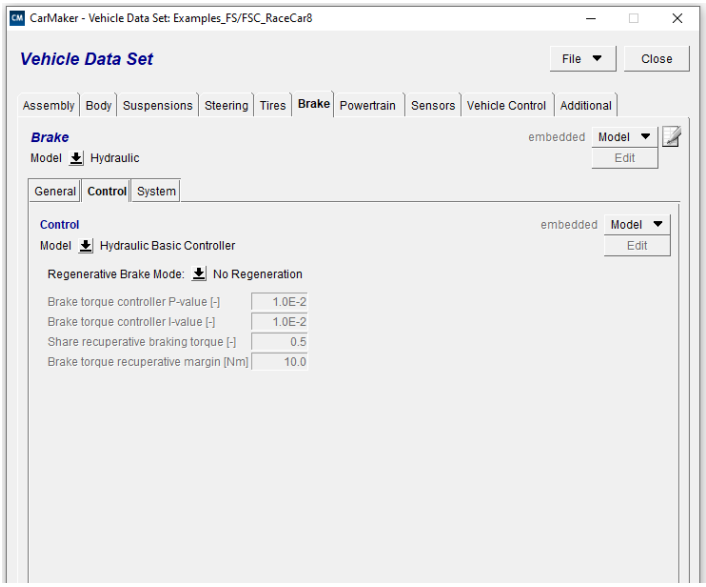



Figure 5.10: Vehicle Data Set - Brake Control

The Hydraulic Brake Control model "HydBasic" gives the option to use regenerative wheel braking torque in an electrical or hybrid powertrain. The regenerative braking torque can either be applied by a parallel or a serial strategy as show in Figure 5.11.

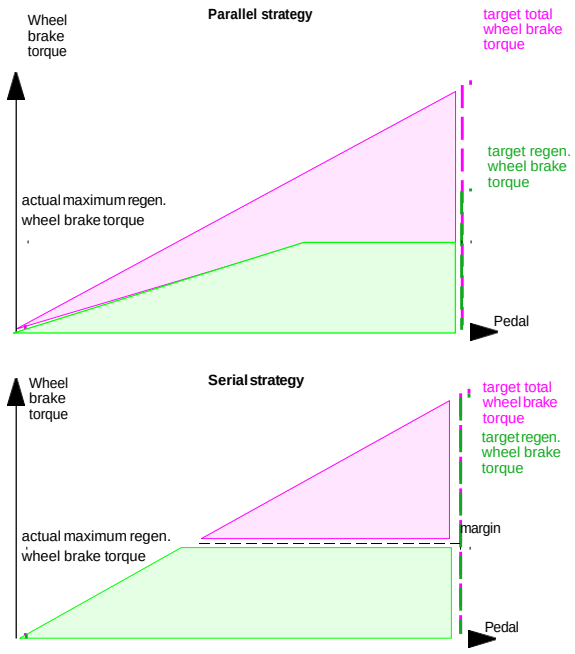



Figure 5.11: Brake torque repartition strategy

The actual hydraulic brake system is parameterized in the *System* tab, shown in Figure 5.12.

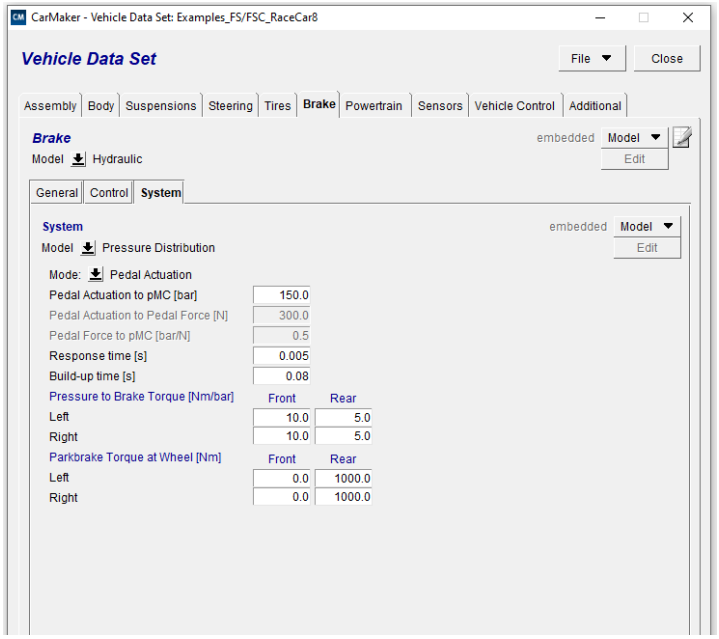



Figure 5.12: Vehicle Data Set - Brake System

It takes several steps to transform the brake pedal force applied by the driver into a braking torque at the wheels:

- Force applied to pedal by the driver.
- Transmission of this force through the pedal mechanism.
- Transformation of this force into a pressure in the main brake cylinder.
- Division of this pressure between the front and rear systems.
- Transformation of pressure into a braking torque.

These five steps are implemented in CarMaker by the definition of the maximum pedal force and two 'transmissions'. The meaning of the two transmissions *Pedal Force to* $p_{MC}$ and *Pressure to Brake Torque* will be explained in detail in the following.

Table 5.3: Required parameters for the calculation of the brake system

| Explanation | Variable Name |
|---|---|
| Pedal to Pedal Force | $F_P$ |
| Transmission Braking Pedal | $i_{bp}$ |
| Diameter Main Brake Cylinder | $d_{mbc}$ |
| Diameter Brake Piston Front | $d_{bp,f}$ |
| Diameter Brake Piston Rear | $d_{bp,r}$ |
| Adhesion Coefficient of the Brake Pads | $\mu_b$ |
| Effective Radius Braking Disk Front | $r_{bd,f}$ |
| Effective Radius Braking Disk Rear | $r_{bd,r}$ |
| Brake Portion Front | $b_f$ |
| Brake Portion Rear | $b_r$ |

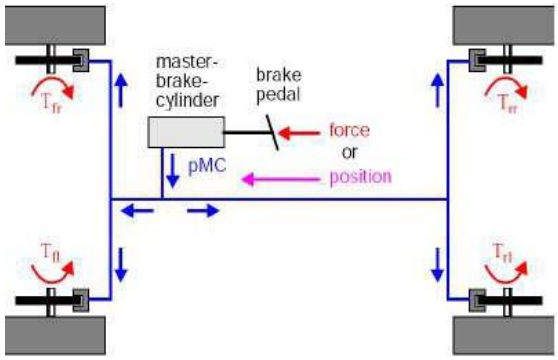



Figure 5.13: Schematic layout of the brake system parameters

The first transformation, "Pedal force to $p_{MC}$", is made up by the quotient of the pressure at the main brake cylinder $p_{MC}$ and the pedal force $F_P$. Therefore, the pressure $p_{MC}$ must be determined first:

$$p_{MC} = \frac{F_{mbc}}{\pi \cdot \frac{d_{mbc}^2}{4}} \quad \textbf{(EQ 21)}$$

The force $F_{mbc}$ at the main brake cylinder is a result of the pedal force multiplied by the ped- al transmission factor:

$$F_{mbc} = F_P \cdot i_{bp} \quad \textbf{(EQ 22)}$$

Thus:

$$\frac{p_{MC}}{F_p} = \frac{i_{bp}}{\pi \cdot \frac{d_{mbc}^2}{4}} \quad \textbf{(EQ 23)}$$

In general, the ratio "Pressure to Brake Torque" is different at the front and rear axles. This is due to an unequal brake balance and the brake type used which commonly differ from front to back. Thus, this ratio must be calculated for each axle separately.

The pressure at the front brake piston is:

$$p_f = b_f \cdot p_{MC} \quad \textbf{(EQ 24)}$$

The contact pressure on the front brake disc is equal to:

$$F_{ap,bd,f} = p_{VA} \cdot \pi \cdot \frac{d_{bp,f}^2}{4} \quad \textbf{(EQ 25)}$$

With the braking force at the front disks:

$$F_{bd,f} = F_{ap,bd,f} \cdot \mu_b \quad \textbf{(EQ 26)}$$

The braking torque at the front wheels can be estimated using:

$$T_f = 2 \cdot r_{bd,f} \cdot F_{bd,f} \quad \textbf{(EQ 27)}$$

Using equation (EQ 35) - (EQ 38) the ratio "Pressure to Brake Torque Front" can be determined:

$$\frac{T_f}{p_{MC}} = 2 \cdot r_{bd,f} \cdot \mu_b \cdot \pi \cdot \frac{d_{bp,f}^2}{4} \cdot b_f \quad \textbf{(EQ 28)}$$

The ratio "Pressure to Brake Torque Rear" is calculated in the same way:

$$\frac{T_r}{p_{MC}} = 2 \cdot r_{bd,r} \cdot \mu_b \cdot \pi \cdot \frac{d_{bp,r}^2}{4} \cdot b_r \quad \textbf{(EQ 29)}$$

The formulas guide you through the parameterization of the linear brake model "Pressure Distribution".

If you have detailed information about your brake system, you can activate the more precise "Hyd_ESP_FS_RaceCar_8.0" brake model by selecting "External File" (*Vehicle data set* > *Brake* > *System* > *Model*). This model considers characteristics of the brake lines and the hydraulic fluid. Moreover, it covers the brake booster, as well as the suction and pilot values as an interface for ABS and ESP controllers.

For more information about this detailed model please refer to the Reference Manual or open the template "HydESP_FS_RaceCar_8.0". Each line starting with a hash is a comment.

**Exercise 8 - Step 6**

- Increase the braking ratio to 75% without changing the available input braking force:
  - Set the *Pressure to Brake Torque* at the front to 12 at both wheels and to 4 at the rear.

## 5.1.8 Powertrain

Please refer to section 5.2 *'Powertrain: Combustion Race Car'*, pg. 92 or section 5.3 *'Powertrain: Electric Race Car'*, pg. 98.

## 5.1.9 Aerodynamics

As aerodynamics become ever more important in Formula Student, CarMaker offers the functionality to model and simulate their effects. Relative to the angle of air flow *tau*, the coefficients for the resistive forces and torques in all three axis must be specified. For this, wind tunnel test results or a very comprehensive CFD model is required.

Please find further information about the measurement procedure of the required coefficients in the Reference Manual (*Help* > *Reference Manual*).

**General** The *Reference Point* determines the point of attack for the combined wind forces. The *Reference Area* usually describes the frontal area of your car and the *Reference Length* corresponds to the wheelbase of your car.

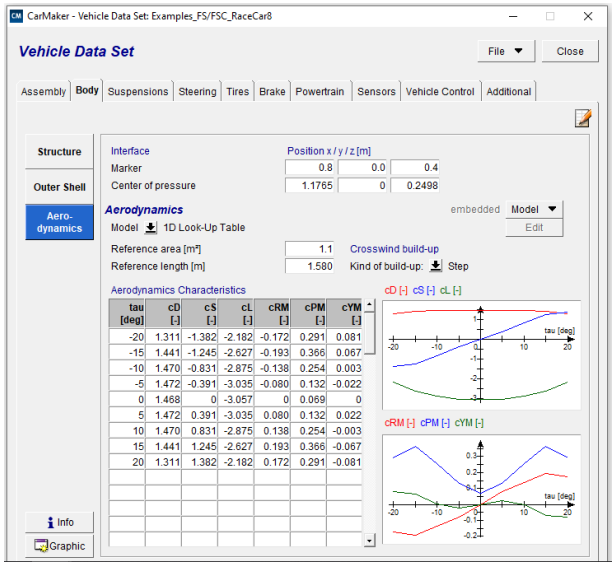



Figure 5.14: Vehicle Data Set - Aerodynamics

## Sensors

Within the *Sensor* tab you have the option to create different kinds of sensors.

**Slip Angle** This sensor is normally placed about the center of gravity. If you wish to place it elsewhere, change the parameters of the sensor's position.

**Inertial** A *Body Sensor* is an inertial sensor that can be placed anywhere on the vehicle to measure movements, velocity, rotational velocity, acceleration and rotational acceleration. All corresponding output quantities will be accessible in IPGControl. To learn more about sensors please refer to the Reference Manual.

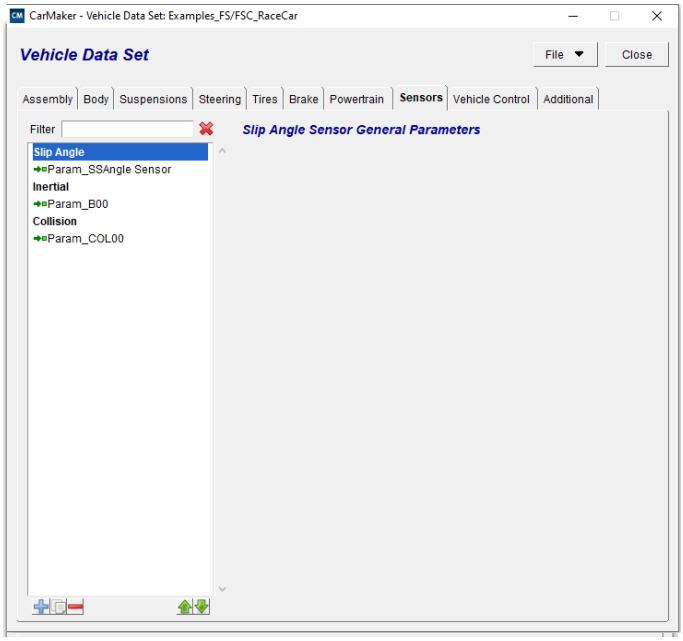



Figure 5.15: Vehicle Data Set - Sensors

## 5.1.10 Vehicle Control

This tab is only relevant for the integration of controller systems, mainly driver assistance systems.

The integration of controller models relevant to Formula Student is demonstrated by the *CarMaker for Simulink* example "TorqueVect.mdl".

## 5.1.11 Misc.

The object file used by IPGMovie can be changed under *Movie Geometry*. This file can be created by the user with the help of CAD software, or one can be bought online. The file must be in .obj, .kmz, .dae or .3ds format.

File definition guidelines:

- ONLY polygons and triangles, NO splines or free forms.
- No more than 50,000 nodes.
- If this option is available while generating the file: generate the normal for all points.
- Colors: Generate the corresponding material file.

Under *Vehicle Graphics*, a picture in the PNG format can be chosen to be displayed in the CarMaker Main GUI and in the Vehicle Editor as a preview of the 3D model. The user can also create a preview by taking snapshots of the 3D object by clicking the button next to the file browser. The field *Additional Parameters* allows user to define their own parameters.

Please find further information on the various options for obj files in the IPGMovie Reference Manual to be accessed via the Help menu of IPGMovie.

### Vehicle Outer Skin

If an .obj file as described above is not available, there is another method of displaying your car in IPGMovie: an Abraxas model. This model is a simple, rectangular box which represents the maximal dimensions of your car. To determine these dimensions, two diagonal outermost points are required. These points are then visualized by the model.

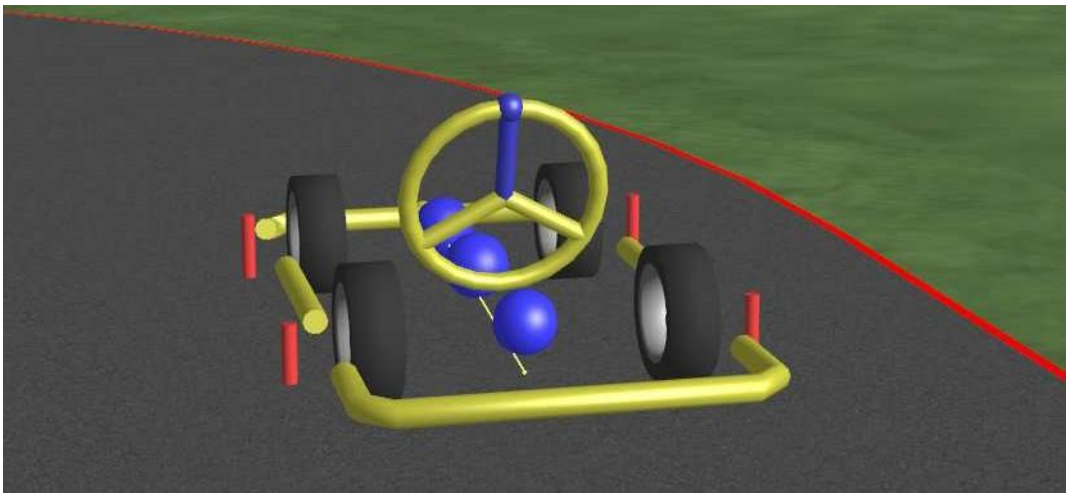


Figure 5.16: Outer dimensions of the FSC_RaceCar shown in the Abraxas mode

# 5.2 Powertrain: Combustion Race Car

## 5.2.1 General

CarMaker gives you the option to choose between several powertrain models such as pure combustion, hybrid and electric vehicles. You can start the parameterization of any powertrain model by editing predefined settings. To do so, simply right click in the powertrain window or select one from the *Pre-Configuration* drop down menu (see Figure 5.17)

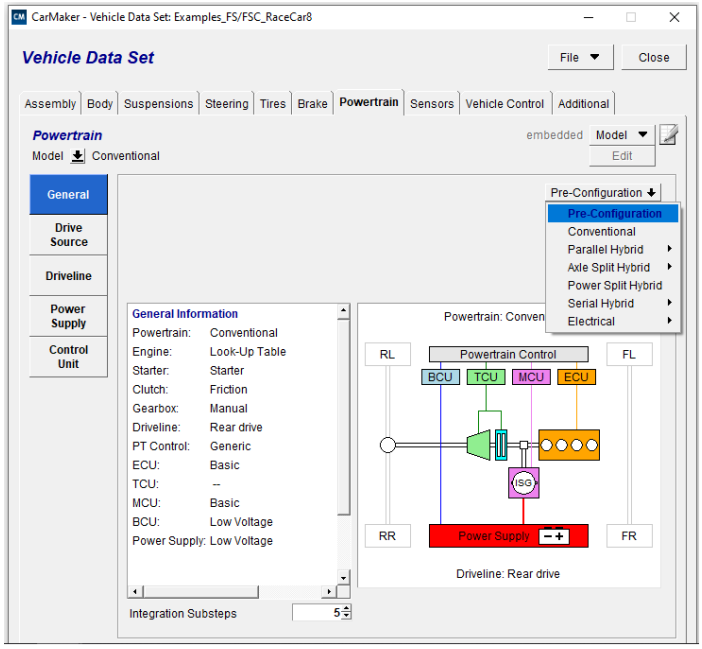



Figure 5.17: Vehicle Data Set - Combustion Engine, browsing predefined models

Depending on the selected powertrain model, one to four drive sources (numeration: 0-3) appears in the powertrain window. The drive source characteristics, as well as the gearboxes and clutches can be parameterized in the corresponding *Drive Source* tab.
The connection between the drive sources and the wheels is defined by the driveline model.
The required control units for hybrid or electrical vehicles can be parameterized in the corresponding tab.
Finally, the power supply is presented as a LV and, if necessary, a HV battery model.

The first step to parameterize a combustion powertrain is to select the “Coventional” powertrain model with a right click within the *Powertrain* tab. "Conventional" is a common combustion engine model including clutch, gearbox and differential (used by the FSC_RaceCar). All quantities and parameters can be changed using the GUI.

## 5.2.2 Drive Source

### Engine

**General** The engine used in your Formula Student car can be defined in several ways. One possibility is to use an engine model characteristic value that assumes a linear distribution between gas pedal actuation and engine torque. This can be represented using the "Look-Up Table".

**Torque** With the "Look-Up Table" option you can define a full load power curve (1D Look-Up Table) by inserting individual points. This option is a lot more precise, requires measurements to be taken using an engine test bench.
Another exponent defines the transition from the full load to the part load power curve. Along with the full load torque values, some other engine parameters like the drag torque values, the range of speeds and the idle speed are also required. Note that the full load power curve must start and end at zero.

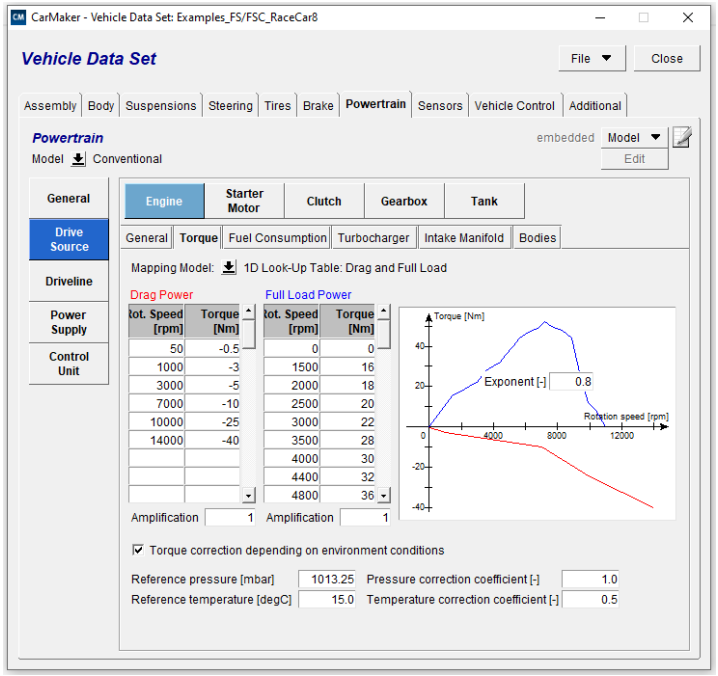



Figure 5.18: Vehicle Data Set - Combustion Engine

A further approach is to define a 2D Look-Up Table with corresponding values for gas pedal position, engine speed and output torque.

**Fuel Consumption** This feature enables the prediction of fuel consumption. Activate the tick box and you can enter values of specific fuel consumption. To do so, you must perform measurements which can be used to populate these points.

**Fuel Tank, Turbocharger, Intake Manifold** These values can be left at their default values. For further information, please look at the Reference Manual.

### Starter Motor

It is possible to define the power and torque of the starter motor as well as the power supply source. For Formula Student cars, the predefined values can be used.

### Clutch

The pre-defined values can normally be used for a Formula Student car. Choose the mode "Friction".

### Gearbox

Here, the ratio of each gear is specified. There is also an option to define the synchronization time. Although Formula Student cars usually don't have a reverse gear, it is necessary to define one as otherwise the program will generate an error message.

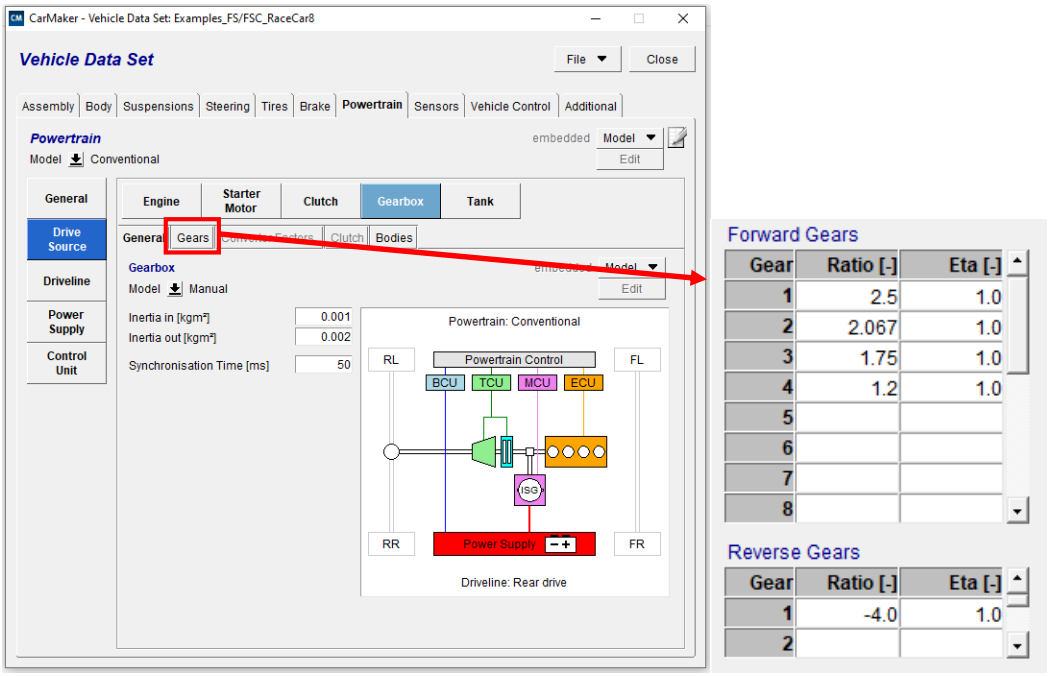



Figure 5.19: Powertrain - Gearbox

## 5.2.3 Driveline

### General

The *Driveline Model* determines the connection between the *Drive Source* and the wheels. For most Formula Student combustion cars, the "Rear Drive" model will be suitable.

### Rear Axle

For a Formula Student car, use the "Torque Sensing" model with the mounting mode "Left to Right". To parameterize the differential, the three values *Torque Bias Ratio Driven, Torque Bias Ratio Dragged,* and *Transmission* should be adjusted.

In the case of using a motorcycle powertrain, you will have a primary ratio between the engine and the gear box. You should consider this primary ratio together with the final ratio. Multiply the final transmission ratio by the primary transmission ratio to gain a value for the *Transmission* field.

The torque bias (TB) is used when one wheel is spinning (e.g. on ice) while the other has more grip. In this situation, some differentials are able to limit the torque transmitted to the spinning wheel ($T_{low}$) so as to favor the wheel that has more grip (it receives $T_{high}$). TB is the factor between $T_{low}$ and $T_{high}$. For instance: if TB = 2, the differential will limit the torque to the spinning wheel to 33.3% and give 66.6% to the other wheel (twice as much as at the spinning wheel).

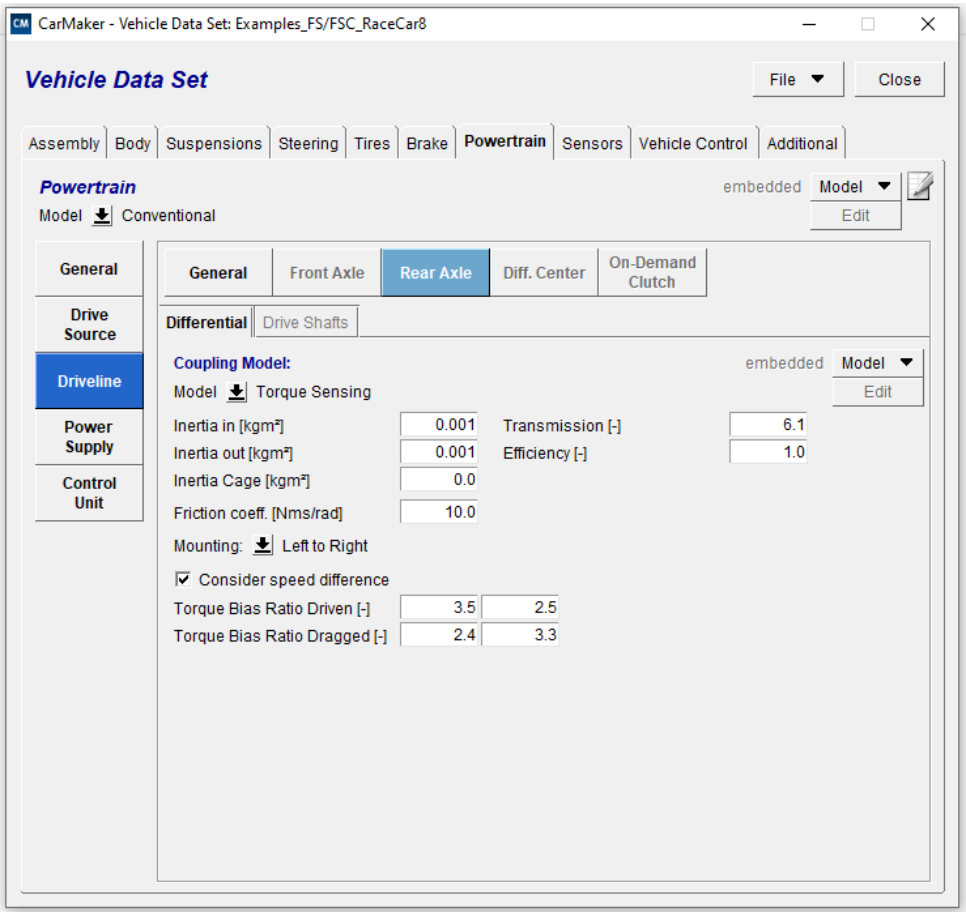



Figure 5.20: Powertrain - Differential Rear

## 5.2.4 Control Unit

The powertrain model contains four *Control Units*:

- ECU (Engine Control Unit) – combustion engine
- TCU (Transmission Control Unit) - gearboxes and clutches
- MCU (Motor Control Unit) - electric motors including integrated starter motor
- BCU (Battery Control Unit) - power supply including batteries

### PT Control

PT Control is a global control unit, the main task of which is to provide target values for the algorithms of the other control units.

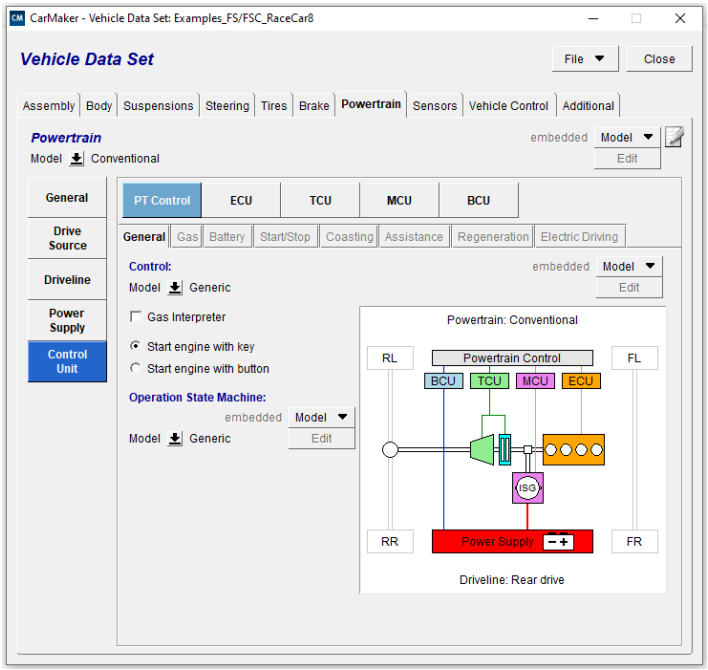



Figure 5.21: Powertrain - Control Units

For a FS combustion car, the control model "Generic" is most suitable. This model allows for driving using the combustion engine only, without use of the starter motor or other electric motors during driving.
The *Operation State Machine* can be set to "Generic" as well.

### ECU

The Basic ECU Model contains a PI controller for the idle speed control and the load controller. The predefined values can be used for most FS cars.
For more information on P/I values, please look at the Reference Manual.

### TCU

There are five different TCU Models in CarMaker. For an automatic gearbox, the logic for gear shifting and shifting limits need to be parameterized.
In the case of a manual gearbox that is shifted by the driver, select "--not specified--" as the TCU Model.

**MCU**

The “Basic” MCU Model consists of a PI controller to control the loads of all electric motors. For a purely combustion-powered vehicle, this will only concern the starter motor. The predefined settings should be suitable for most applications. For more information on the controller, please refer to the Reference Manual (*Main GUI* > *Help* > *Reference Manual*).

**BCU**

The BCU calculates characteristic battery values such as the SOC. If the battery is not of interest within your model, the BCU model can be defined as “--not specified--".

## 5.2.5 Power Supply

The *Power Supply* consists of the electric circuits, battery models, and the ‘auxiliary consumers’. There are four different *Power Supply* configurations available. A LV circuit can be expanded with up to two HV circuits. The batteries are modeled as Chen Battery Models with two RC-circuits and a single resistance that are connected in series.
If the power supply is not of interest, it can also be declared as “--not specified--".

**Exercise 8 - Step 7**

- Beat the clock:
  - Keep the rear driven transmission configuration (*Generic* > *Rear drive*).
  - Increase the engine maximal torque up to 65 Nm and smooth the curve.
  - Reduce the final transmission ratio to 3.5 (in *Driveline* > *Rear Axle* > *Differential* > *Transmission*) and increase the driven *Torque Bias* to 3 (*Driveline* > *Rear Axle* > *Differential* > *Torque Bias Ratio Driven).*
  - Start a TestRun (e.g. “Slalom_Brake”) and compare the pace of “FSC_RaceCar” to your parameterized model.

# 5.3 Powertrain: Electric Race Car

## 5.3.1 General

The *Powertrain* model in CarMaker gives the option to create an electric powertrain with one to four electric motors. The best way to start the parameterization of your electric powertrain in CarMaker is to adapt the “FSE_2021" example models as these come fully paramterized with all elements required for an FSE race car.

The project folder FormulaCarMaker contains two example vehicles with an electric powertrain model. The data set "FSE_RearWheelDrive" is rear wheel drive and "FSE_AllWheelDrive" comes with all-wheel drive. The selected settings for the electric powertrain are described in the following.

If you wish to create a bespoke electric powertrain from scratch, we highly recommend to carefully read the Chapter 15: ‘*Powertrain’* in the CarMaker Reference Manual (accessible via *CarMaker Main GUI > Help > Reference Manual).* You should then start by selecting a predefined template with the *Predefined* button in the *Powertrain* tab. For a purely electric powertrain, select "Electrical" and select the number of electric motors. The number of drive sources will correspond to the number of motors and the parameterization of the control units is adapted to suit an electric powertrain.

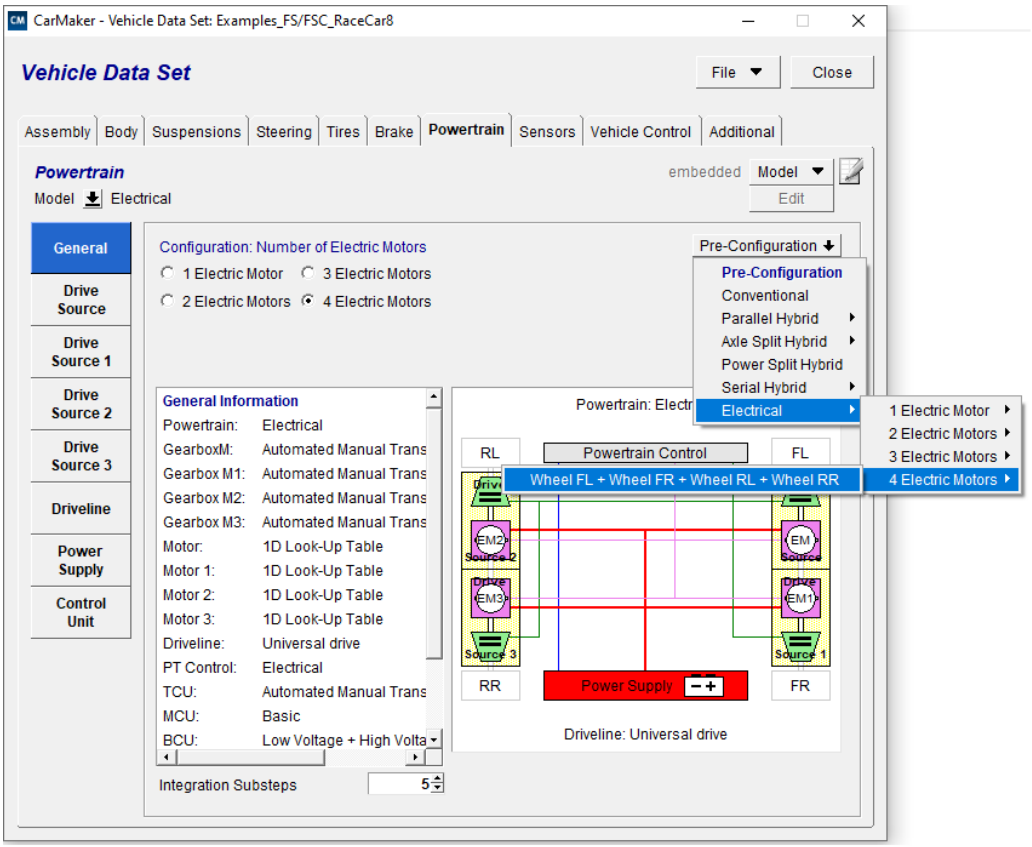



Figure 5.22: Selecting an AWD electric powertrain template with a right-click

## 5.3.2 Drive Sources

Each electric motor is parameterized as an individual *Drive Source*. Besides the characteristic values of the motor, each Drive Source contains a gearbox and a clutch model. The gearbox and clutch are modeled the same way as for a combustion engine described in section "Drive Source", pg. 93. If the powertrain does not include a gearbox and a clutch for each motor, but has only one fixed gear as is common in FS cars, it is possible to

define only one forward gear in the gearbox (without a reverse gear).

The specification of only one forward gear instigates the following procedure in CarMaker: The clutch is disabled internally, and CarMaker models a rigid connection between the motor's output shaft and the gearbox.

The electric motor requires some general information such as inertia and the voltage which describes the circuit used by the motor (high voltage or low voltage).

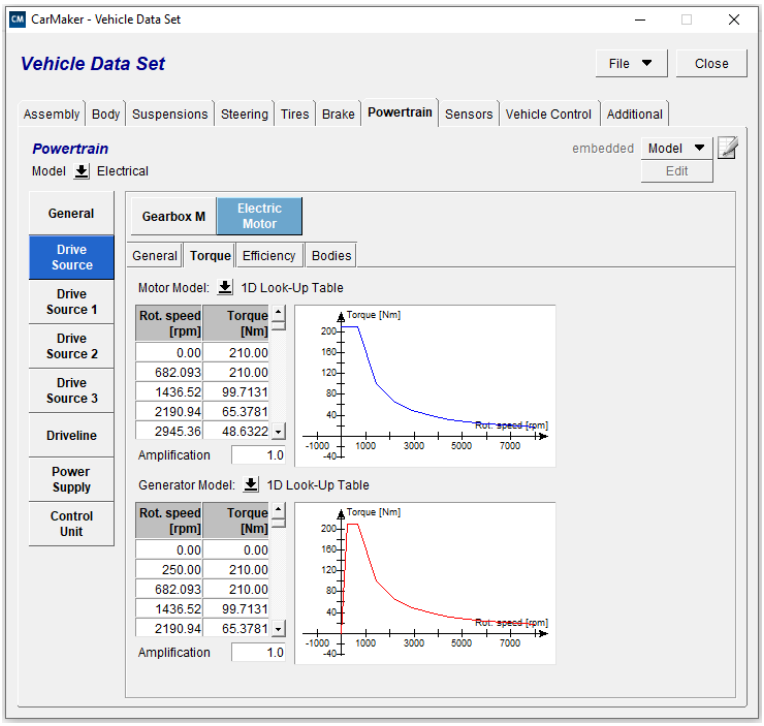



Figure 5.23: Powertrain - Electric Motor Mapping via look-up table

There are two ways to define the torque map. The maximum torque curve can be either be parameterized by characteristic values or using a look-up table as shown in Figure 5.23. The implication of these characteristic values is shown in Figure 5.24.

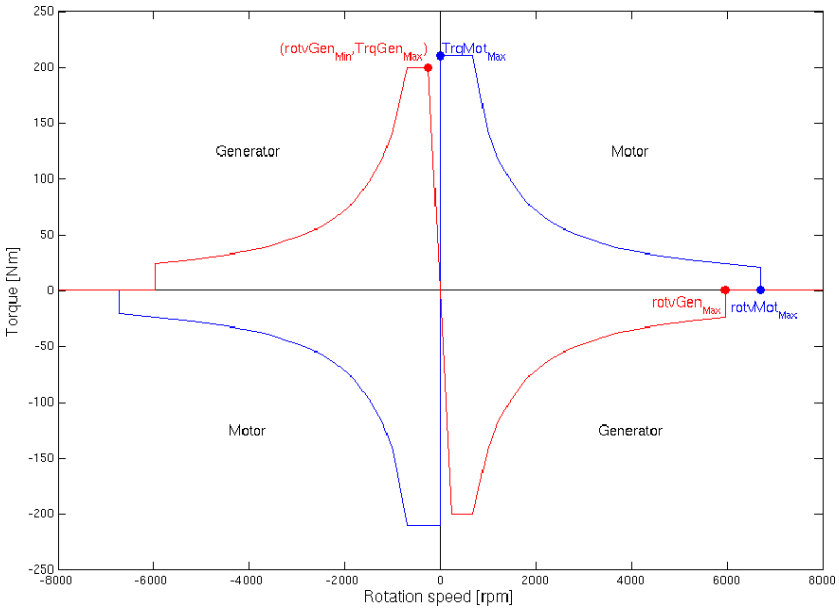



Figure 5.24: Powertrain - Electric Motor Mapping via characteristic values

To prevent an unstable scenario when switching from *motor* mode to *generator* mode, the generator torque curve should start at the point 0 rpm / 0 Nm.

Both the motor and generator characteristics should be specified using absolute values. CarMaker internally handles the sign convention.

Finally, each motor's efficiency can be defined as a single value or a look-up table in the *Efficiency* tab.

## 5.3.3 Driveline

There are several Driveline models available in CarMaker. For an electric powertrain with more than one drive source (motor), the Driveline model "Universal drive" should be selected.

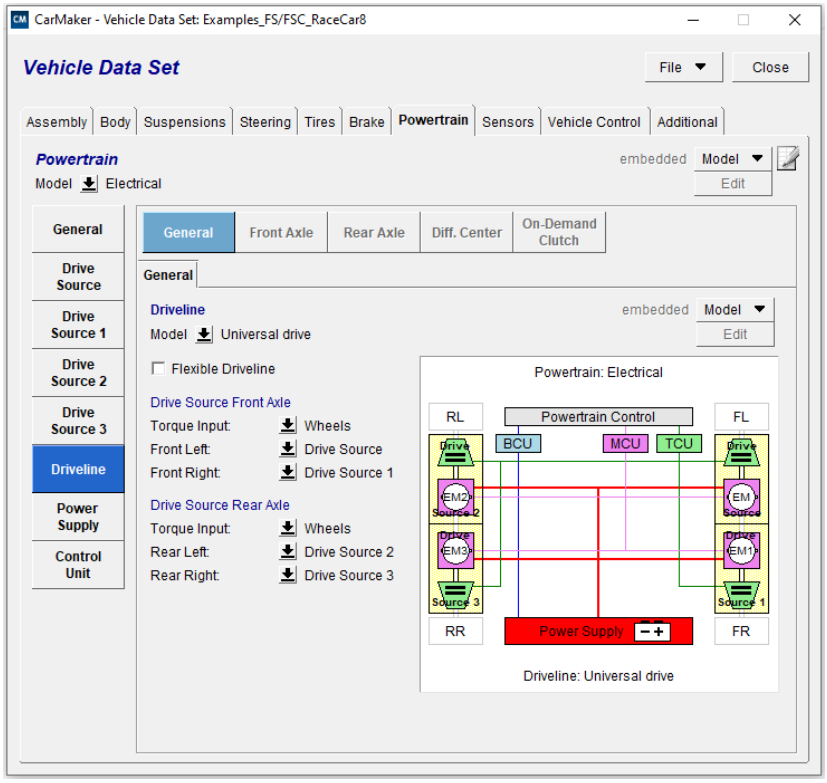



Figure 5.25: Powertrain - Universal drive

As shown in Figure 5.25, the "Universal drive" model provides several options to applying the drive source torque.

**Torque Input: Wheels** The Drive Source is directly connected to the wheels; the driving torque is supported by the wheel carrier. This is the recommended mode for Formula Student Electric cars.

**Torque Input: Driveshafts** The Drive Source is connected to the wheel by a shaft, the driving torque is supported by the vehicle body.

**Torque Input: Differential** The Drive Source is connected to the axle's differential.

Furthermore, the different Drive Sources (electric motors) can each be assigned to a wheel.

## 5.3.4 Control Units

*Control Units* are responsible for managing the *Drive Sources* and the *Power Supply.*: One Control Unit is responsible for all components of the same category:

- Motor Control Unit (MCU) --> electric motors
- Transmission Control Unit (TCU) --> gearboxes and clutches
- Battery Control Unit (BCU) --> batteries responsible for power supply

The *Control Units* have two main tasks: The evaluation of the current state of the component and the regulation of electro-mechanic components in order to reach specified values (e.g. target torque for motor) provided by PTControl.

The *PT Control* unit is the ‘highest ranking’ and provides targets for the other control units in order to implement the defined powertrain control strategy.

The powertrain model "Electrical" comes with a typical control strategy for an electric powertrain. This operation strategy is described in the following. If further functions are required, it is possible to replace the control strategy by a model extension via Matlab/ Simulink, an FMU or C-code.

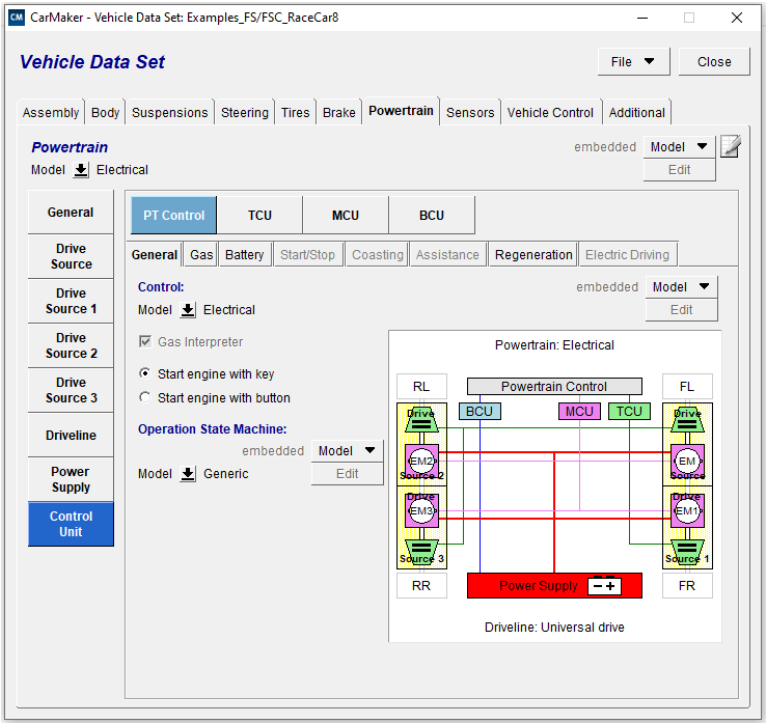



Figure 5.26: Control Units in the Powertrain dialog

### PT Control

The *PT Control* is the main control unit of the entire powertrain. It manages the vehicle’s operation states, the battery, the interpretation of the gas pedal and power regeneration.

In the case of an electric vehicle, you should always use the “Electrical” control model. It automatically deactivates the PT Control features, which are not required of an all-electric drivetrain.

**General**

*Battery Management* controls the batteries state of charge by assigning torques to the electric motors and manages the energy transfer between the electrical circuits. It can be deactivated if not required and is an optional feature. The *Operation State Machine* manages the vehicle’s operational state based on pedal position and start/stop

button activation. IPGDriver is adapted to the "Generic" Operation State Machine and thus this setting should be left unchanged. For detailed information on the controller parameters, take a look at Reference Manual Chapter 23.15.2 *'PTControl'*.

**Gas** The interpretation of the gas pedal position is specified here. It can either be a linear correlation defined by characteristic values or nonlinear by populating a look-up table.

**Regeneration** This tab provides the two most common operational strategy modes for electric vehicles: *regenerative braking* and *regenerative drag*. For both modes, the limits for initiating or deactivating the strategy must be defined. The meaning of these limits are shown in Figure 5.27 and Figure 5.28. The regenerative braking strategy is defined within the brake model (section '*Brake*', pg. 86).

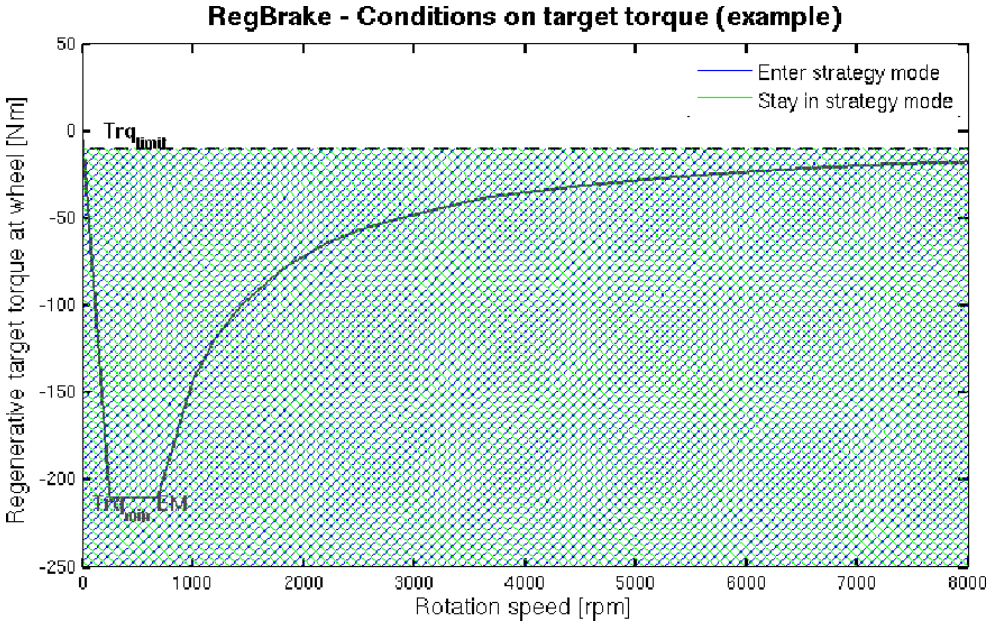



Figure 5.27: Powertrain - Conditions of target torque

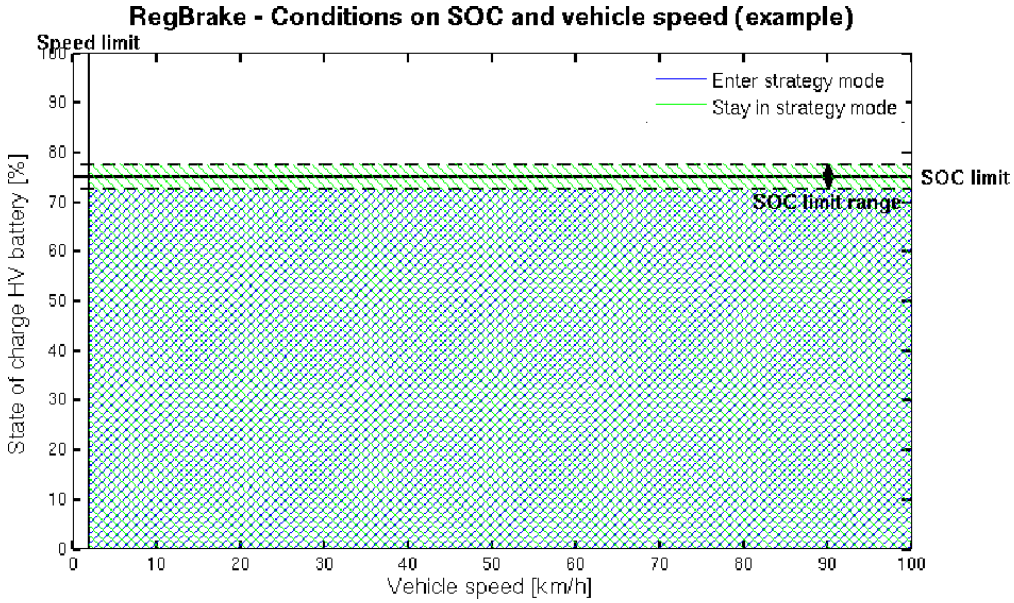



Figure 5.28: Powertrain - Conditions of SOC and vehicle speed

### Transmission Control Unit (TCU)

The TCU functionality is similar to that provided for the combustion engine as explained in section 5.2.4 *'Control Unit'*, pg. 96. If there is only one static gear defined in the gearbox, the TCU can be deactivated by selecting "--not specified--".

### Motor Control Unit (MCU)

The MCU model "Basic" sets each motor's load and determines its maximum motor and generator torque using the target rotation speed. For more information on the P/I values of the torque and speed controller, please take a look at the Reference Manual.

### Battery Control Unit (BCU)

The BCU calculates the State of Charge (SOC) and State of Health (SOH) of the batteries. It attempts to keep the batteries State of Charge at the target value by applying additional generator torques to the electric motors, or by controlling the energy transfer between the low voltage and high voltage 1 electrical circuits. The energy can be transferred between a maximum of three circuits: A low voltage circuit and up to two high voltage circuits.
This control unit is optional. Choose "BCU Model: Low Voltage + High Voltage 1" for your race car.

## 5.3.5 Power Supply

The main task of the *Power Supply* model is the accounting of the electrical power flow to and from the electrical circuits. It contains the 'auxiliary consumers' and can also include the converters between the low voltage (LV) and high voltage (HV) circuits.
Since most Formula Student cars do not use a converter, it can be deactivated by entering 0 kW as the maximum power.

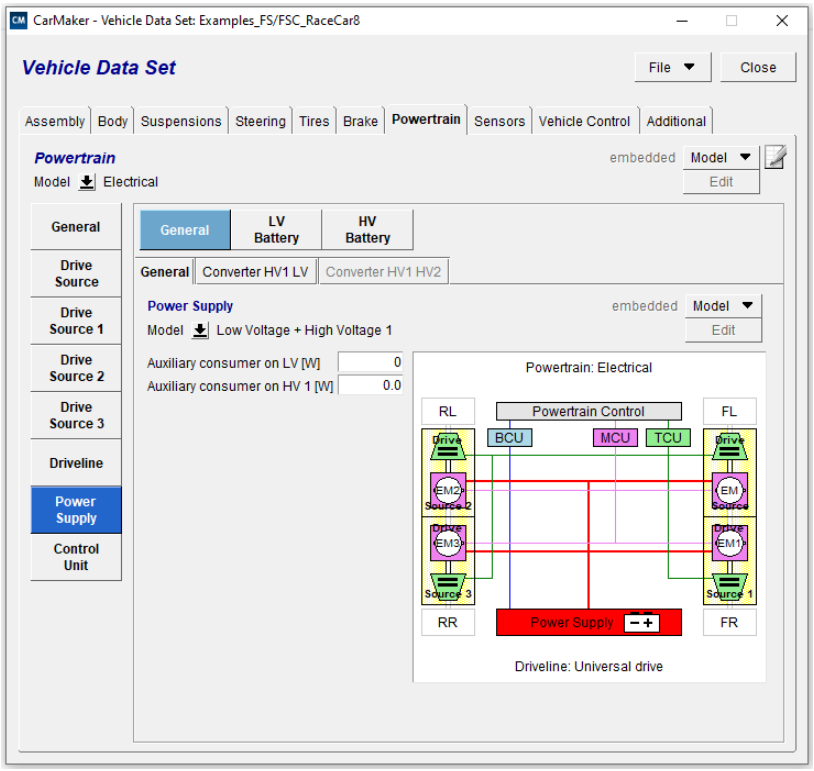



Figure 5.29: Settings of the Power Supply module

## Battery Model

The *Battery Model* is based on an electric battery model combined with a characteristic curve to introduce the influence of State of Charge (SOC) on the idle voltage. The HV and LV batteries are replicated using a Chen Model. It consists of two RC-circuits and a single resistance that are connected in series as shown in Figure 5.30.

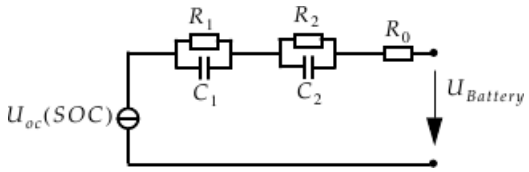



Figure 5.30: Powertrain - Chen Model

The dependency of an ideal voltage source on the batteries' SOC can be parameterized via a look-up table. The battery's SOC is not allowed to drop below the minimum state of charge or to exceed the maximum state of charge

You can find more information on the battery model in the Reference Manual Chapter 15.6.2: *'Battery'*.

# 5.4 Powertrain: "OpenXWD" Model

## Introduction

To get started with CarMaker and Simulink you should read the section '*CarMaker for Simulink'* in the Quick Start Guide (to be found in the CarMaker Main GUI under *Help*) beforehand. Do not forget to choose "src_cm4sl" from the project folder FormulaCarMaker as your Current Directory in MATLAB. There you find some files which are related to the FSE race car example:

**OPENXWD.mdl** The entire generic CarMaker powertrain model is replaced with the "OpenXWD Stand Alone" configuration. The driving torque is generated by the Simulink model which includes two electric motors and traction control.

**FSE_Parameters.m** The m-file initializes the necessary parameters and characteristics for all Simulink models.

**FSE_Define_Motor Characteristic.xls** This is an Excel sheet (version 2003 or later) used to generate motor characteristics.

**FSE_M85.txt** This text file includes motor characteristics with a maximum torque of 85 Nm.

**FSE_M100.txt** This text file includes motor characteristics with a maximum torque of 100 Nm.

To run the Simulink model, the OpenXWD powertrain model needs to be chosen in the vehicle data set. The example vehicle "FSE_OpenXWD" already includes the necessary settings.

To start CarMaker for Simulink please make sure that the MATLAB script "cmenv.m" was run successfully. It adds several paths to the current MATLAB session which give access to the CarMaker toolbox. If you did not install CarMaker in the default installation folder C:\IPG, the execution of the cmenv.m script will result in an error. Please adapt the path in line 6 of the cmenv.m script to your IPG installation folder.

cmenv.m

```
1  function cmenv (varargin)       % -*- Mode: Fundamental -*-
2  % CMENV - Add CarMaker directories to the MATLAB search path.
3  %
4
5      % CarMaker installation directory.
6      if isempty(which('cmlocaldir'))
7      cminstdir = 'C:/IPG/carmaker/win64-9.1';
8      else
9      cminstdir = cmlocaldir  % for mat: CM-9.0
10     end
11
12     disp(['CarMaker directory: ', cminstdir]);
13     if ~exist(cminstdir, 'dir')
14     error('Unable to find specified CarMaker installation directory.');
15     end
16
```

Figure 5.31: This path must be set to the correct installation folder

### Exercise 9 - Step 1

- Prepare yourself:
  - Read the section '*CarMaker for Simulink'* in the Quickstart Guide (Help > *Quickstart Guide*).
- Open MathWorks MATLAB
  - Change your current directory to "<pathto>/FormulaCarMaker/src_cm4sl".
  - Check "cmenv.m" path and execute the script.

## 5.4.1 OpenXWD Example

The FormulaCarMaker project folder includes a Simulink model for the Formula Student Electric competition: “OPENXWD.mdl”. It uses the OpenXWD interface and has two electric motors at the rear axle combined with traction control.

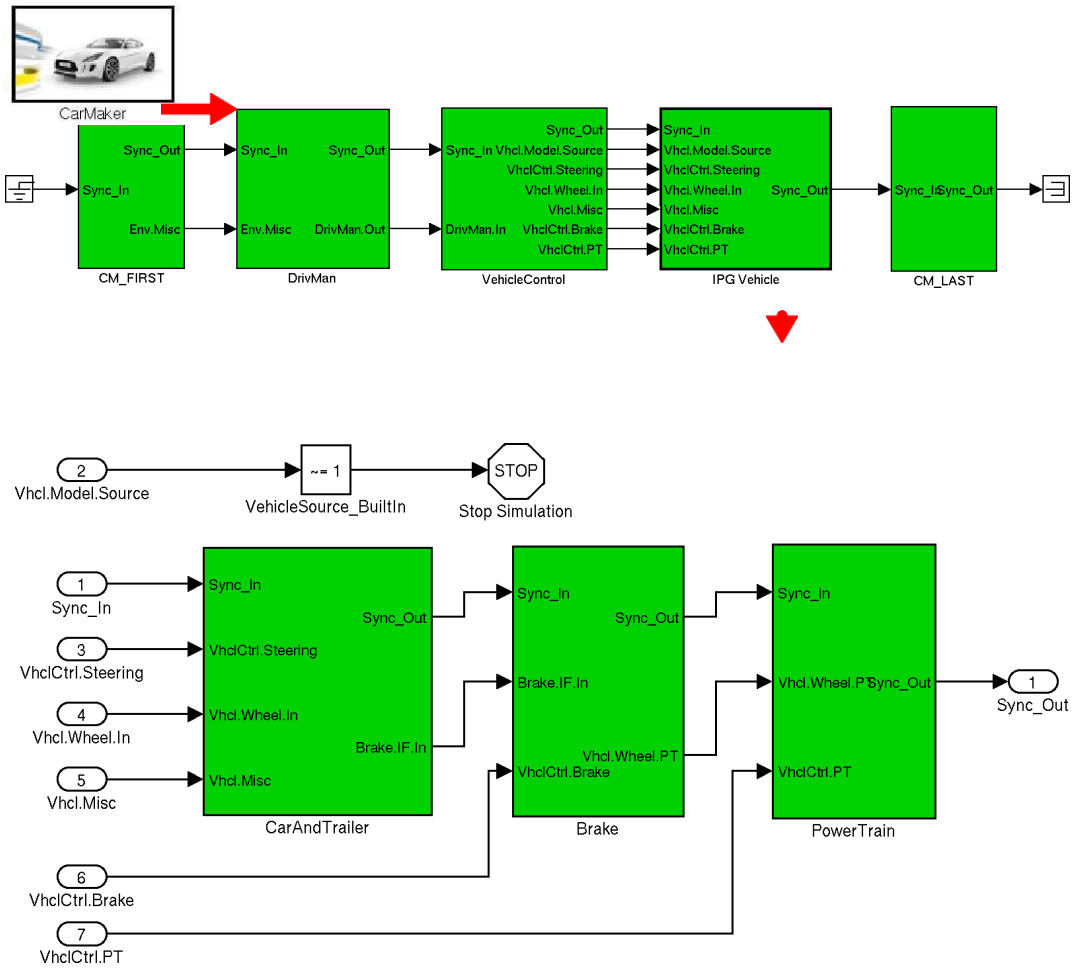



Figure 5.32: Simulink with the “OPENXWD.mdl” Model

With the OpenXWD interface there is a lot flexibility to designing your own powertrain model as every pre-defined component (e.g. clutch, gearbox, differential...) can be deactivated. The basic concept of this model is relatively simple: CarMaker expects a driving torque at the wheels. It is up to the user’s model extension to determine as to how this torque is generated. The wheel integration is processed by CarMaker and it returns the resulting wheel speed.

To activate the *OpenXWD* driveline model, select the Powertrain Model: "OpenXWD" in the vehicle data set editor (see Figure 5.33) or select the example vehicle data set "Examples_FS/FSE_OpenXWD". Without any extension, the vehicle will not drive in this configuration. The interface variables required to transmit a driving torque from your model extension to CarMaker are called "PT.W<position>.Trq_Drive". For this, the Simulink model extension is used.

**Exercise 9 - Step 2**

TestRun with MATLAB:

- Open the "OPENXWD.mdl" in MATLAB-Simulink.
- Double-click - icon to open *CarMaker for Simulink*.
- In the *CarMaker for Simulink* GUI, create a new TestRun, add the "Examples_FS/FSE_OpenXWD" race car to the TestRun.
- Save TestRun (e.g. “Acceleration_XWD”)

**Or**

- In the *CarMaker for Simulink* GUI, open the TestRun “Slalom_Brake_XWD”

## 5.4.2 Powertrain Data Set Based on "OpenXWD" Model

With the powertrain "OpenXWD", the options *With Engine* or *With Motor* are similar to using the “Generic" model, but the connection from the gearbox output to the wheels is removed and has to be modelled by the user (e.g. in Simulink). The option *Stand Alone* deactivates every component of the CarMaker powertrain. All required quantities and parameters (from generating torque right up to transmitting it to the wheels) need to be calculated by an external program (e.g. Simulink).

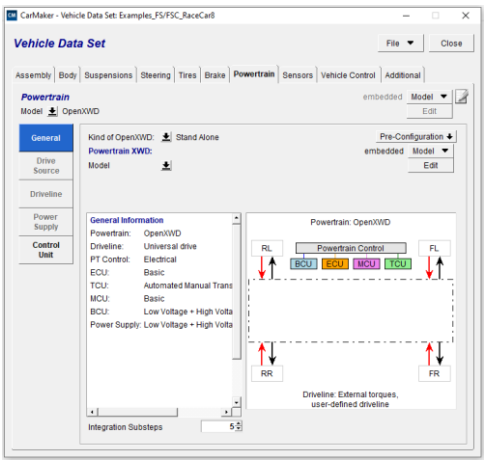



Figure 5.33: Vehicle Data Set > Powertrain > OpenXWD > Stand Alone

More detailed information about the powertrain models can be found in the Reference Manual (*Main GUI* > *Help* > *Reference Manual*), Chapter 15 *‘Powertrain’*.

## 5.4.3 General Remarks to the Simulink Models

Generally, you can manipulate or overwrite a signal in CarMaker using Simulink. Using the pre-defined CarMaker interface blockset (available in the "CarMaker4SL" library) you have easy access to modifiable/overwritable quantities. Use the "Read CM Dict" block to read the current value and the "Write CM Dict" block to overwrite the desired variable. Remember, when you overwrite a quantity calculated in CarMaker (also called UAQ, see section *‘Data Access’*, pg. 41) you influence the variable within the entire CarMaker and MATLAB/Simulink workspace. To get an overview of these quantities, view the Reference Manual Chapter 23.4 *‘User Accessible Quantities’*.

Additionally, you can also create your own variables by using the "Define CM Dict" block. For integrating your self-defined quantities into a Simulink model, please use the "Read/Write CM Dict" blocks. More information can be found in the Programmer’s Guide Chapter 6.2: *‘The CarMaker Interface Blockset’*.


Figure 5.34: Simulink Dictionary

## Additional Parameters in the Vehicle Data Set

Please note, that the Open_XWD powertrain model only replaces the electrical and mechanical components of the drivetrain. It still uses the pre-implemented powertrain controller models (see section 5.2.4).

To make an Open_XWD model run, you need to select a suitable PTControl unit. For an FS race car we recommend the PTControl unit "CM4SL". This requires some information about the powertrain model. These parameters can be provided in the vehicle data set under *Additional* > *Additional Parameters*. In our example, we want to avoid any interference with the PTControl unit which is why the information provided to the PTC is reduced. The minimum set of required parameters is defined in the example "FSE_OpenXWD:

Table 5.4: Additional Parameters for PTControl with Open XWD model

| Parameter | Value | Description |
|---|---|---|
| PowerTrain.PTKind | BEV | Definition of the powertrain kind (electric) |
| PowerTrain.DL.DriveSourcePos | RL | Drive source position 0 |
| PowerTrain.DL.DriveSourcePos1 | RR | Drive source position 1 |

For further information about the parameters please review the section 15.7.10 *'Powertrain model "OpenXWD"'* in the Reference Manual (*Main GUI* > *Help* > *Reference* Manual).

## 5.4.4 OpenXWD: Powertrain in Simulink

In comparison to "Generic.mdl" the "OPENXWD.mdl" has a modified powertrain with two terminators which output "PT.OperationState" and "PT.Engine.rotv". Using the blue sub-model, a way of generating these UAQs is provided. Note that there is a saturation block which only allows a range from 100 - 16000 rpm at the engine output shaft.

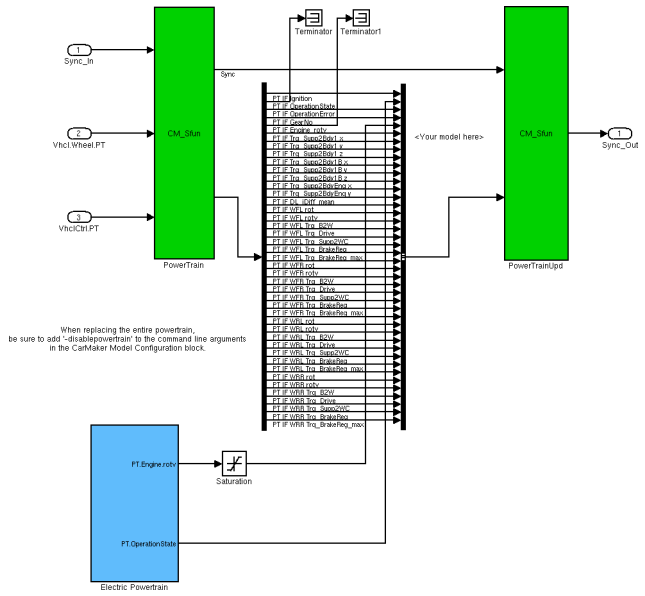



Figure 5.35: Overview of "IPGVehicle > Powertrain" of "OPENXWD.mdl"

A inside look in the blue sub model displays a predefined option to generate the required UAQ's.

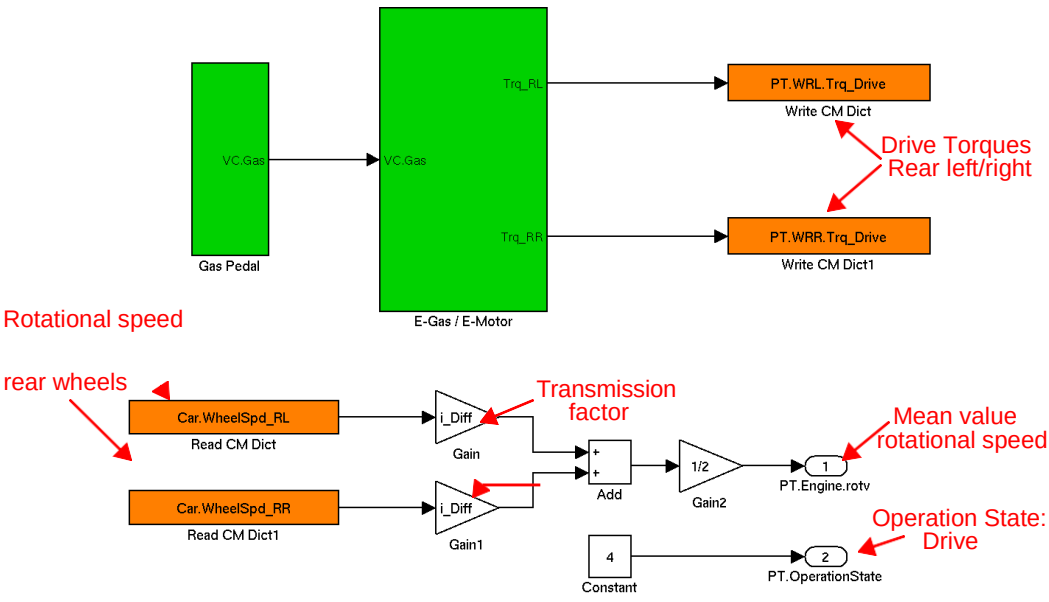



Figure 5.36: Simulink Powertrain of "OPENXWD.mdl"

## Gas Pedal

The function of this subsystem is both to initialize the signal "Vhcl.Ignition" (enables switching off of the ignition during simulation) and the evaluation of the gas pedal position.

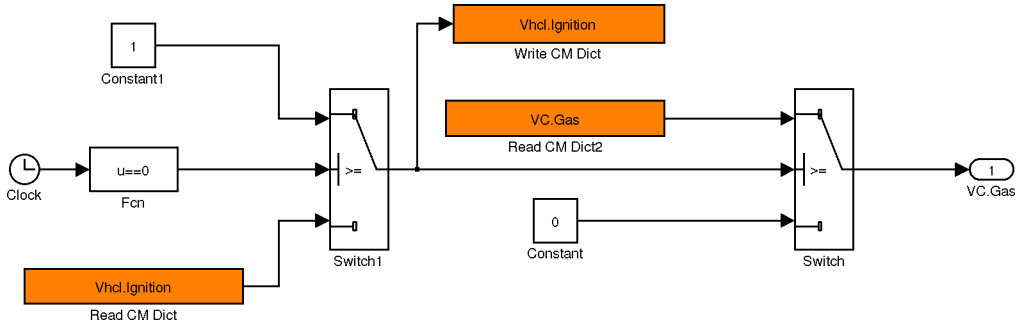



Figure 5.37: Electric Powertrain > Gas Pedal

## E-Gas / E-Motor

This subsystem includes the "E-Motor Mapping" block, the heart of the electric powertrain. Here you can upload your own motor characteristics (more details in section 5.5.2 *'Manipulating the OpenXWD Example Model'*, pg. 115) into the system. The PT1 transfer functions should give the system a plausible step response. You can also change the fixed transmission factor, but do not forget to adapt these factors for every corresponding block!

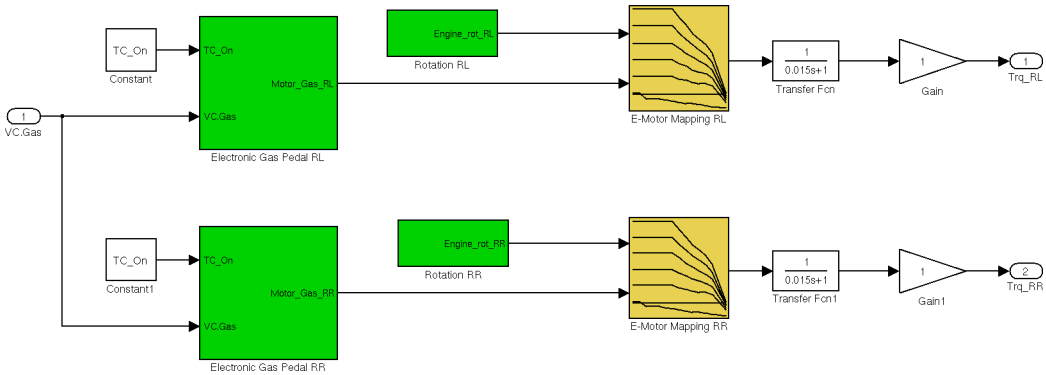



Figure 5.38: Electric Powertrain > E-Gas / E-Motor model

Next, the rotational speed (blocks: Rotation RL/RR) of each of the motor shafts is determined. The calculation includes a transmission factor and a unit change from revolutions per minute to revolutions per second.

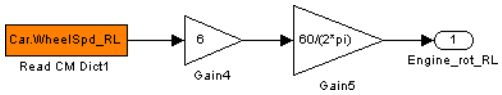



Figure 5.39: Electric Powertrain > E-Gas / E-Motor model > Rotation <Pos>: Calculation of engine speed

Lastly, you will find the "Electronic Gas Pedal" block. This subsystem determines whether the gas pedal value from the driver or from a control system (e.g. traction control) will be used. It can serve as an interface for your own control system by simply replacing the model.

The model includes a very simple traction control system for each of the driven wheels. When the variable "TC_On" equals 1 in the parameter file (“FSE_Parameters.m”) the traction control is activated. It can be deactivated by entering zero.
The traction control (short: TC) monitors the current slip in the longitudinal direction and the gas pedal position values. When the slip increases too quickly, the TC sets the electronic gas pedal into a waiting position (mode 1). If the slip oversteps a threshold, mode 2 will be activated and the gas pedal actuation will automatically be decreased (linearly).
In mode 0 the TC simply decides whether an increasing gas pedal value comes from the driver or from a controller command to accelerate the car.

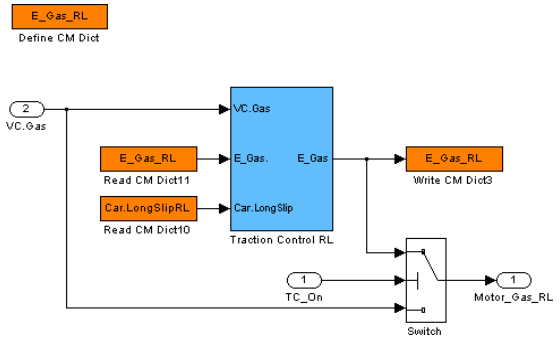



Figure 5.40: Simulink Electronic Gas Pedal model

## 5.4.5 Parameter File

The parameter file "FSE_Parameters.m" should be loaded into the MATLAB workspace before you start a simulation. This file initializes the necessary motor characteristics and defines the parameters for the traction controller (OpenXWD model). It also contains the parameters for regenerative braking and torque vectoring models.

Table 5.5: Parameters in FSE_Parameters.m

| Parameter | Unit | Description |
|---|---|---|
| i_Diff | [-] | Factor for a mechanical transmission |
| TC_On | [-] | Turn traction control on or off by setting this parameter to 1 or 0 |
| TC_Dericative | [-] | Threshold traction control mode 1 |
| TC_Relay_On | [-] | Maximum allowed longitudinal slip |
| TC_Relay_Off | [-] | Minimum allowed longitudinal slip |
| TC_Incr_Gas | [-] | Increasing electronic gas pedal (mode 0] |
| TC_Decr_Gas | [-] | Decreasing electronic gas pedal (mode 2) |
| TV_On | [-] | Turn torque vectoring on or off by setting this parameter to 1 or 0 |
| TV_Override | [-] | Increase the torque applied to outer wheels in corners |
| eta_Gen | [-] | Efficiency of the electric motor in generator mode. |
| Gen_Loss | [-] | Braking torque loss due to time needed to build up stator load |

Table 5.5: Parameters in FSE_Parameters.m

| Parameter | Unit | Description |
|---|---|---|
| Brake_Pedal_Travel | [-] | Defines how much of the brake pedal travel is allowed for purely regenerative braking |
| Brake_Mech_Rec_Ratio | [-] | Defines how much of the braking torque is regenerated after reaching "Brake_Pedal_Travel" |
| SOC_Init | [%] | Initial state of charge |
| C_Batt | [kWh] | Battery’s capacity |
| U0_Batt | [V] | No-load voltage of battery |
| R0_Batt | [Ohm] | Resistance of battery |
| C1_Batt | [F] | Capacity |
| R1_Batt | [Ohm] | Resistance |
| C2_Batt | [F] | Capacity |
| R2_Batt | [Ohm] | Resistance |

After every change a reload of the complete parameter file to the workspace is required.

# 5.5 Adaption of the Example Models

## 5.5.1 User Defined Powertrain Control Models

In case you are using the pre-implemented electric powertrain model in CarMaker, as explained in section 5.3.2 *'Drive Sources'*, pg. 98, the PT Control strategy can be replaced by a user defined control strategy. CarMaker offers different ways to implement user models such as the C-code interface, FMUs or MATLAB/Simulink. As the latter is the most popular interface among most Formula Student teams, we will explain this approach in the following.

The "TorqueVect.mdl" in the FormulaCarMaker project folder is an example of a self-developed controller containing both traction and yaw control.
In contrast to the OpenXWD example presented in section 5.4.1 '*OpenXWD Example'*, pg. 106, this model only replaces the PTControl unit. Therefore, it is not necessary to model the entire powertrain such as the motor or the battery model. As shown in Figure 5.41, the con- troller calculates a load for each motor which is passed to the MCU.

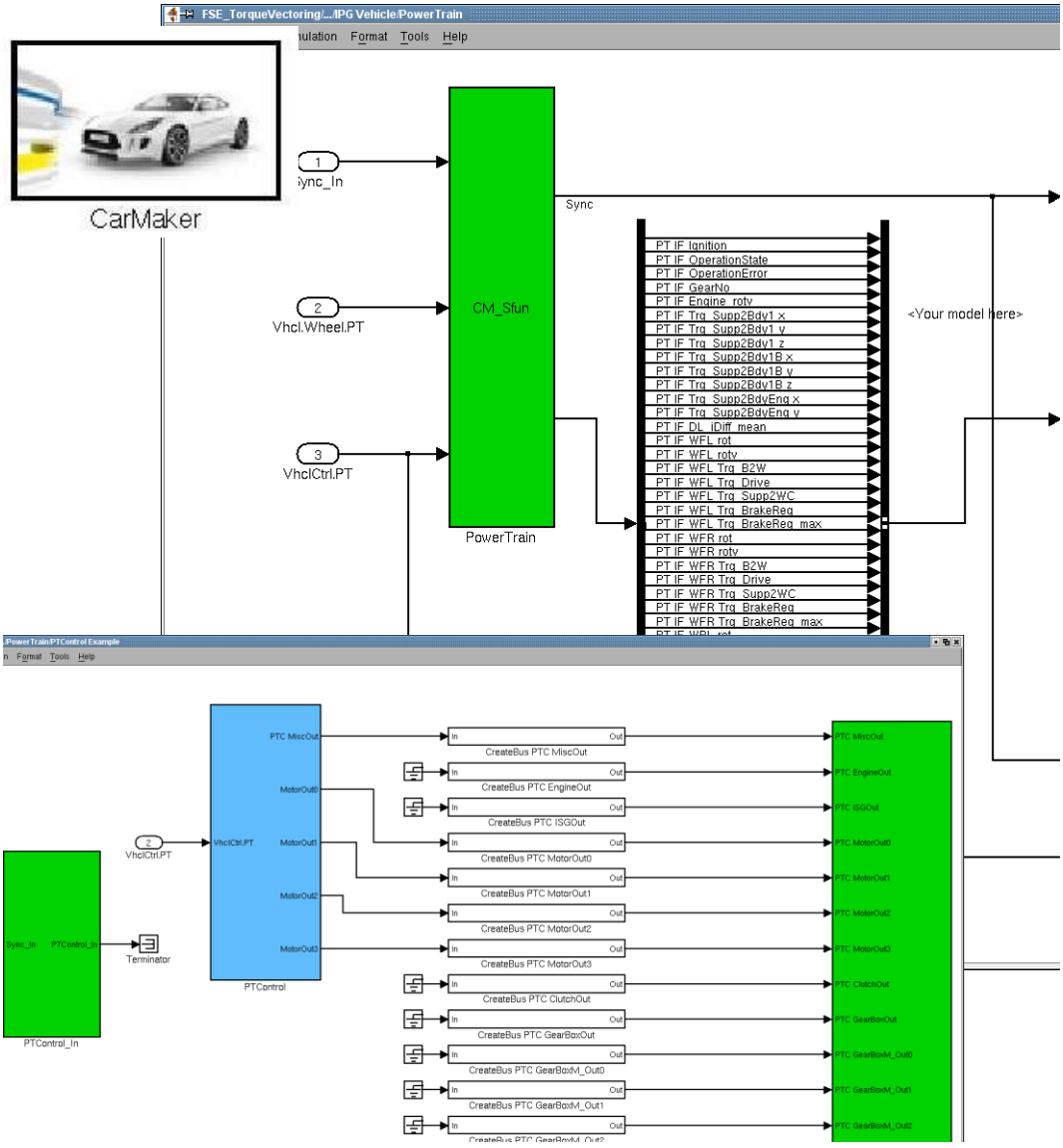



Figure 5.41: PTControl - Simulink interface

There are four torque vectoring controllers, one for each wheel. In this example model, a higher torque is applied to the outer wheel in a cornering scenario. The yaw momentum is increased and the vehicle becomes more agile. The maximum possible yaw rate is calculated by multiplying the friction coefficient with the gravity coefficient which is then divided by the current velocity of the car. The difference between actual and maximum yaw rates indicates the potential for a torque increase to the outer wheel. This parameter is then added to the power demanded from the motor. Using the TV_Override parameter (in FSE_Parameters.m) you can artificially increase the maximum yaw momentum in order to increase this value - however, this may lead to unstable cornering behavior.

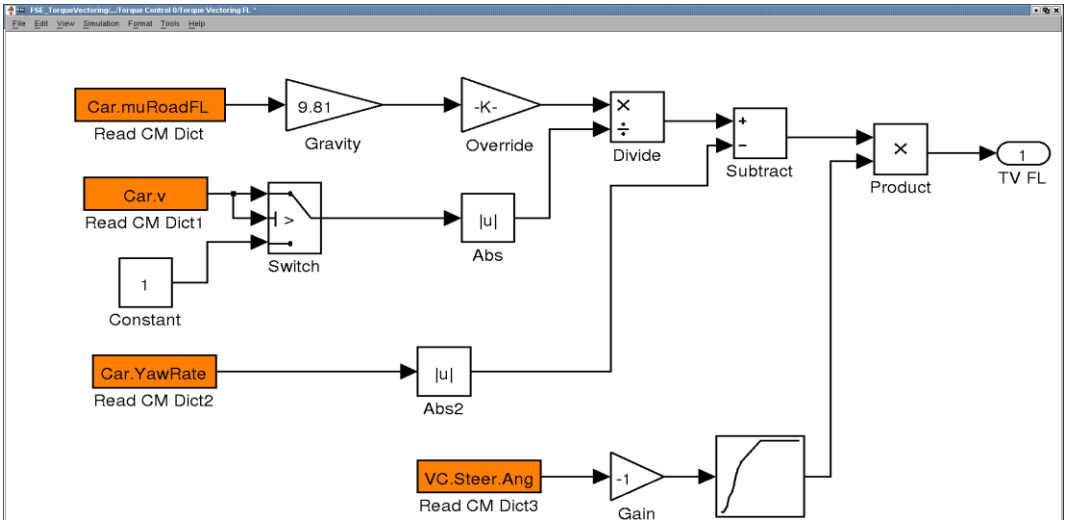



Figure 5.42: Torque Vectoring parameter calculation for the rear left wheel

To replace the PT Control model provided by CarMaker with a MATLAB/Simulink model, *CarMaker for Simulink* must be initialized as explained above and select the example vehicle "Examples_FS/FSE_TorqueVectoring". With any other vehicle data set, "CM4SL" must be selected as the control model in the PTControl tab.

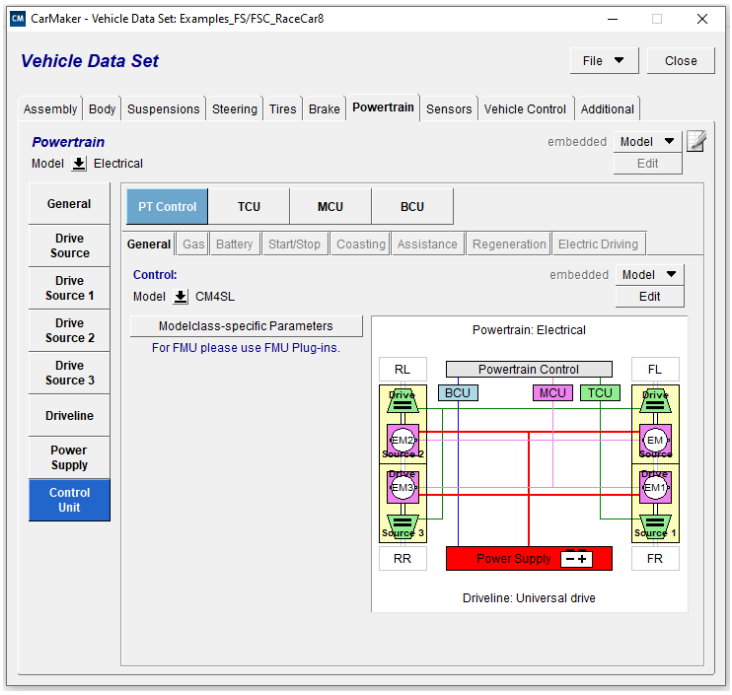



Figure 5.43: Activating a Simulink based PT Control strategy

## 5.5.2 Manipulating the OpenXWD Example Model

This chapter explains how to implement your own motor characteristics in the OpenXWD example model. The file "FSE_Define_MotorCharacteristic.xls" is an Excel sheet used to define motor characteristics. Only the green highlighted fields should be adjusted.

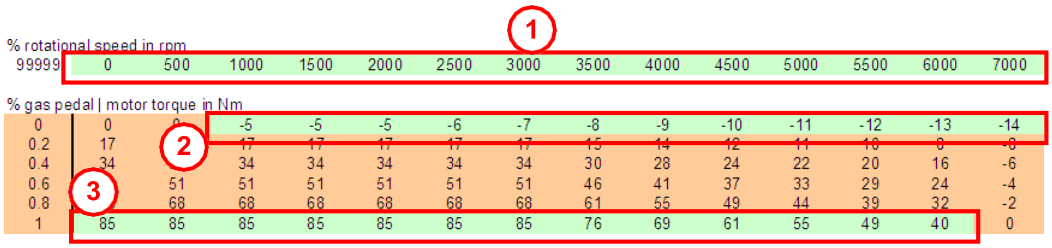



Figure 5.44: Table Motor Characteristic
Box 1: Rotational speed range
Box 2: Drag torque
Box 3: Full load

**Exercise 10**

- Manipulate the .xls file with your own data:
  - Enter the rotational speed range (in rpm) in box 1
  - Enter the drag torque curve (in Nm) in box 2
  - Enter the full load curve (in Nm) in box 3. Set at least one point to 120.
  - Check your entries using the torque-speed diagram
  - Save the file as a text file (tab separated) and ignore the warnings, e.g. FSE_M120.txt
- Load your motor characteristics:
  - To initialize the model with your motor characteristic please load the generated text file into the MATLAB workspace with the command

    *load('FSE_M120.txt')*

    (you also can extend the parameter file).
  - Do not forget to change the matrix names in the 2D-Look-up-Table for the "E-Motor Mapping" (e.g. to FSE_M120).

### 5.5.3 Driver Model

It is highly recommended to use the User Parameterized Driver when simulating with FS cars. The simulations are reproducible, and the driver can be adapted to needs of each discipline.

However, if there is the need to simulate with the Racing Driver there are several things which must be taken into account to successfully perform a Driver Adaption with a FSE Powertrain Model. The following steps have to be completed:

**Exercise 11 - Step 1**

- Teach the driver:
  - After opening one of the Simulink models using MATLAB and opening the CarMaker GUI, a TestRun can be created or loaded.
- Start a *Basic Knowledge* Driver Adaption.
- Start the Simulation
  - After completing the adaption and starting the simulation, you might observe that the vehicle drives very slowly along the defined course. In this case continue with Step 2.

**Exercise 11 - Step 2**

- If the vehicle appears to be slow, make necessary changes to the info file:
  - To fix this, open the ASCII-file of your TestRun with a text editor (e.g. Notepad++, Kate). Near the end of the file there is a line called "Driver.Learn.vIdle" which defines the vehicle's idle velocity.
  - **Set a value greater than 0 for this parameter**.
  - Save the text file.

```
Driver.Learn.nEngine.Standard:
     0 0
     0 0
     0 0
     0 0
     0 0
Driver.Learn.vIdle = 1.000
Driver.Learn.vMax = 103.956
Driver.Learn.vG2nEng025 = 25.989
```

Listing 5.1: Extract of an ASCII-file of a TestRun

## Exercise 11 - Step 3

- Driving with the User Parameterized Driver:
  - There is also the option to select the User Parameterized Driver instead of the Race Driver, which is recommended.
  - Load the *aggressive driver* parameter set with a right-click anywhere within the *Driver* window.

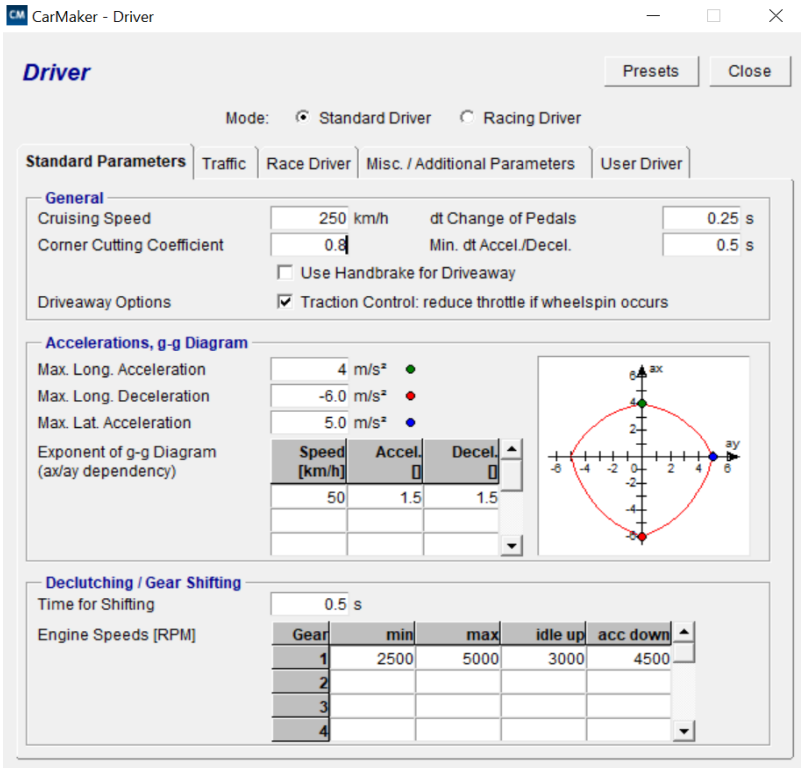



Figure 5.45: User parameterized Driver parameters

It is recommended to adjust the parameters in the *General* area:

- Adjust the *dt. Change of Pedals* to a lower value.
- *Min. dt Accel/Decel* = 0.
- Double the values within the *Accelerations, g-g-Diagram* section.

- Start your TestRun:
  - If the vehicle leaves the road or rolls over, decrease the acceleration parameters incrementally.

## Exercise 11 - Step 4

- Driving with the Race Driver:

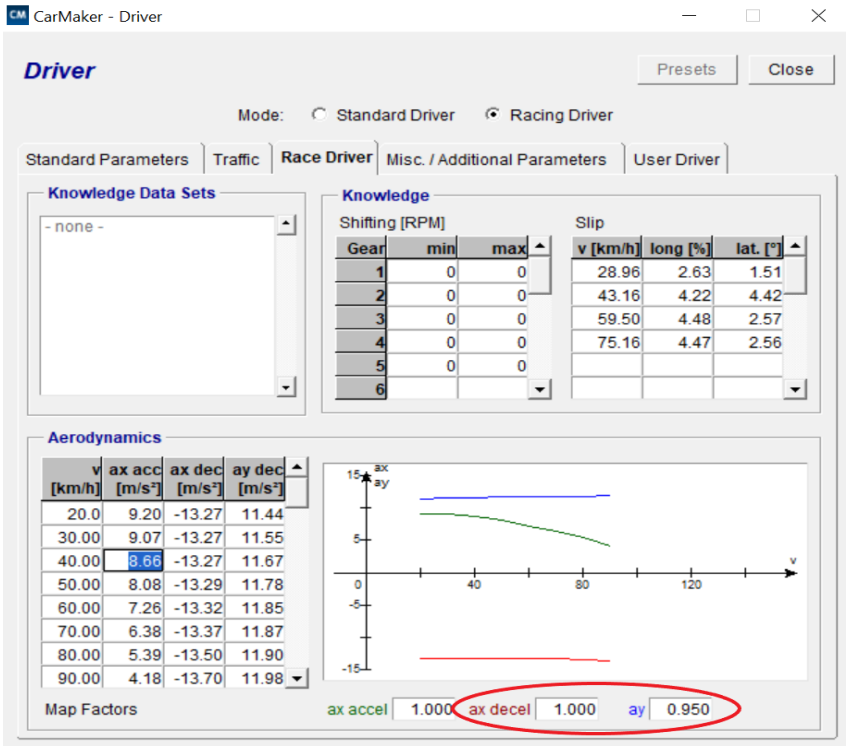



Figure 5.46: Racing Driver Parameters

- If you select the Race Driver following a *Race Driver Adaption* after completing step 1, in some cases you will encounter that the vehicle leaves the road or rolls over. But there is a way to avoid this: Decrease the acceleration factors *ax deccel* and *ay* incrementally and observe the results.
- Optimize these factors so that the vehicle stays on track, but still accomplishes an acceptable lap time.

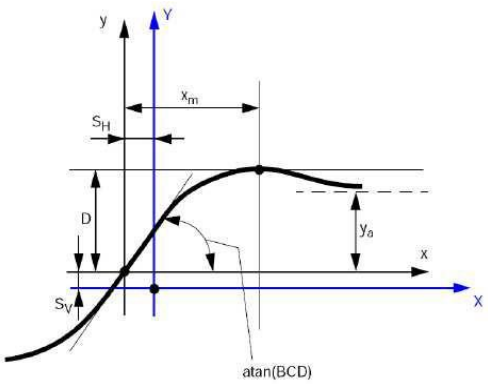
# 5.6 Creating a Tire Dataset Using IPGTire

The tire is one of the most important components of a vehicle. All forces and torques are transferred from the road to the car through the tire. Hence, the tires provide the basis for all lateral and longitudinal dynamics of a vehicle. Thus, the tire model used in a simulation must be a detailed one. CarMaker offers different options for this:

- Pacejka Magic Formula
- IPGTire
- TameTire
- Tire Data Set Generator

## 5.6.1 Pacejka Magic Formula

The Pacejka Model is based on a complex mathematical formula. Its parameters, called "Magic Parameters", do not have any physical basis. They are calculated with specialized programs using test readings. If you would like to learn more about the Magic Formula, see the CarMaker Reference Manual (*Main GUI > Help > Reference Manual*).




$$y(x) = D \cdot \sin[C \cdot \operatorname{atan}\{Bx - E(Bx - \operatorname{atan}(Bx))\}]$$

Figure 5.47: The Magic Formula

The following will give some advice as to generating a Magic Parameters dataset based on measured raw data from a flat belt tire test rig. The used test data is from the database of the FSAE TTC (Tire Test Consortium).

"The Formula SAE Tire Test Consortium (FSAE TTC) is a volunteer-managed organization of Formula SAE teams who pool their financial resources to obtain high quality tire force and moment data. The FSAE TTC's role is to gather funds from participating FSAE teams, organize and conduct tire force and moment tests and distribute the data to all participating teams." [2]

To be able to generate a set of parameters from the tire data, the Pacejka model needs to be fitted to the measured raw data. The fitting of the Magic Formula is an iterative process. It is feasible to either download/program a script in e.g. MATLAB to generate the parameter set or to use commercial software. In FSAE many teams use OptimumTire from OptimumG.

"OptimumTire is a convenient and intuitive software package that allows users to perform advanced tire data analysis, visualization, and model fitting. The model fitting procedure is very fast and efficient partially due to the data processing tools incorporated into the software." [3]

The software is available at optimumg.com. The preprocessing and fitting procedure to generate the Magic Formula parameters is described in a tutorial available from OptimumG's website.

In addition to the tutorial, a few hints regarding preprocessing and optimization:

- Pay attention to the coordinate system you use while importing the raw data into OptimumTire. The data was measured using the SAE coordinate system.
- The raw data's SA sweeps looks like the following (including warming phase):

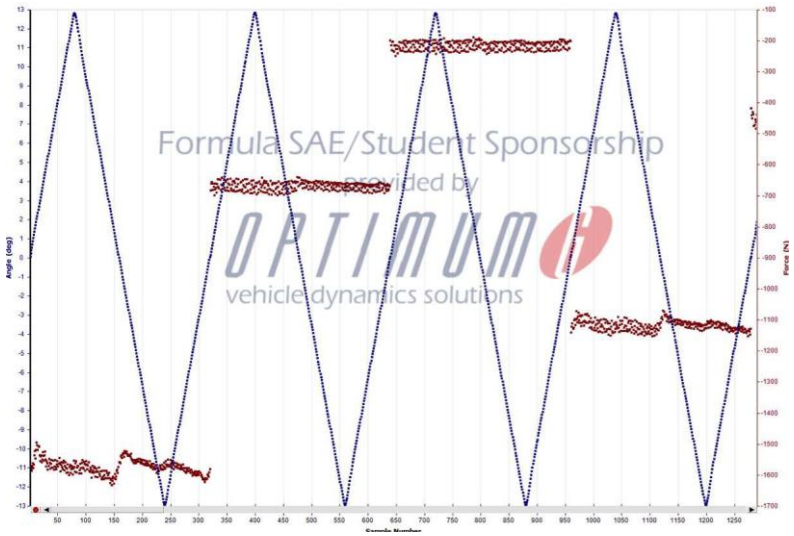



Figure 5.48: Raw tire data in OptimumG

- You should remove the warming phase and spring stiffness measurements, as well as the first and last 3˚ of SA sweeps.

  The filtered data looks a lot smoother and periodical. The fitting process also becomes more effective.

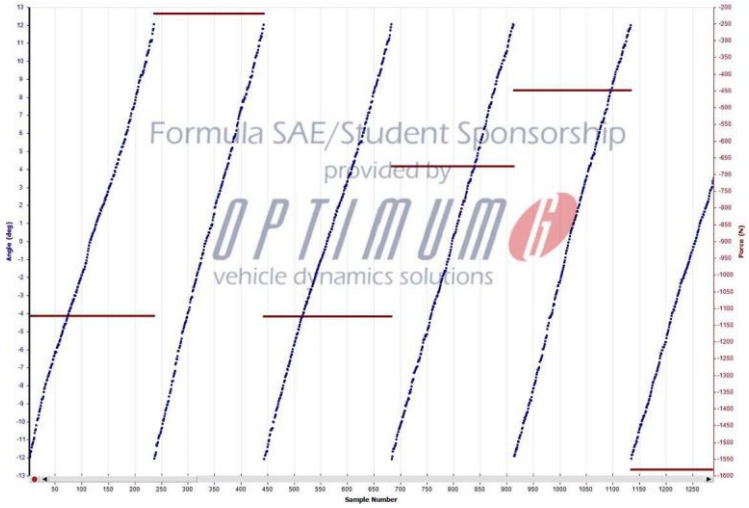



Figure 5.49: Cropped tire data in OptimumG

## Import Model into CarMaker

Before you export your data, select “ISO” as the coordinate system for your fitted Pacejka MF5.2 model. The question pops up whether you wish to "convert" or "interpret as new coordinate system". Choose convert!

Within OptimumTire there are several options to exporting the generated data. Use the “export to TIR” function.

### Exercise 12

- Copy the generated tire data and paste it in an ASCII File:
  Rename the file to something similar to: “DD.MM.JJJJ_TireType_TyreSize.tir” and put it into “/Data/Tire/Examples/Pacejka” within your CarMaker project directory.
- Use the CarMaker *Tire Data Set Editor* to convert the .tir file to a CarMaker Infofile via *Parameters* > *Tires* in the CarMaker Main GUI. Or in *Vehicle Data Set* > *Tires* hold the folder button and select "Edit" from the dropdown menu. The *Tire Data Set Editor* opens.

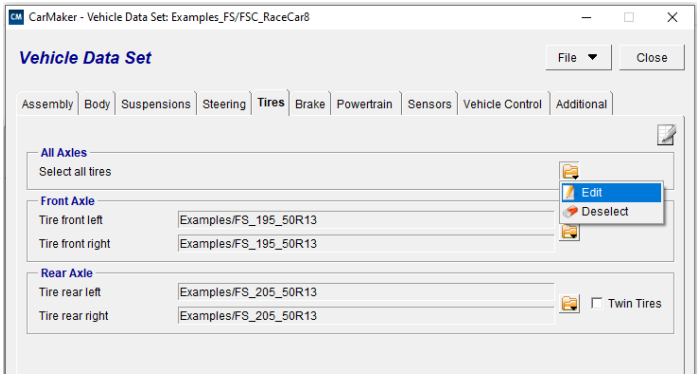



Figure 5.50: The Tire Data Set Editor in CarMaker

- In the *File* menu of the new window, select *New* > *Magic Formula 5.2*.

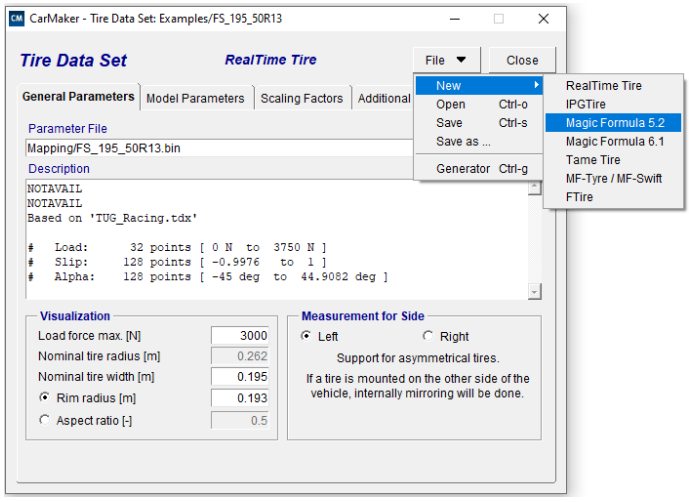



Figure 5.51: Generating a new Pacejka tire data set

- In the tab *General Parameters,* go to the *Visualization* section and adapt the tire size according to your data set. This information is used by IPGMovie for visualization only, and does not affect the physical tire characteristics. In the next tab, *Model Paramaters,* import the generated. tir file with all the Magic Parameters to define the

Pacejka model. Click the button labeled *Import Adams Property File* and choose your tire data from the *Examples>Pacejka* folder.

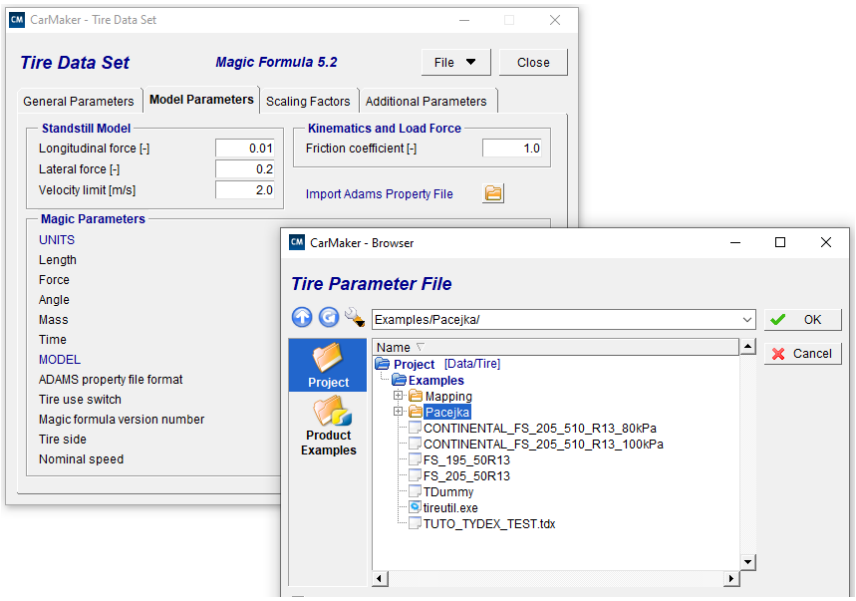



Figure 5.52: Import of an Adams property file

- Make sure that the following parameters are set correctly:
  - section "MODEL"

```
PROPERTY_FILE_FORMAT= MF_05  # ADAMS property file format
USE_Mode= 14         # Tire use switch
```

  - section "ALIGNING"

```
Align.SSZ2= -2.2371e-09 # variation of distance s/R0 with Fy/FzNom
Align.SSZ3=3.6029e-08   # variation of distance s/R0 with camber
Align.SSZ4=3.9826e-08   # variation of distance s/R0 with camber and load
Align.QTZ1=0.3      # Gyroscopic torque constant
Align.MBELT=7.5      # belt mass of wheel
```

  - section "LONGITUDINAL"

```
PTX1= 1   # Relaxation length SigKappa/Fz at FzNom
PTX2= 0.2 # Variation of SigKappa/Fz with load
PTX3= -0.15 # Variation of SigKappa/Fz with exponent of load
```

Attention: The decimal separator in all CarMaker and IPGKinematics applications is a dot (.) instead of a comma (,). If an error message appears related to inconsistent metrics, most of the time it is due to the wrong decimal separator.

- Save your tire data set via *File > Save As* in the *Tire Data Set* window.

## 5.6.2 IPGTire

The *IPGTire* model is also based upon test measurements. However, the results aren't used to implement any mathematical formulas but the discrete values can be used directly. The missing values are estimated via interpolation. The basis of the tire data provides the TYDEX format. Files required for CarMaker are generated out of a TYDEX file using the program "tireutil.exe" which is also a tool of IPG Automotive GmbH. Thus, all you have to do to create a tire dataset is prepare a TYDEX file and transform it.

### What is a TYDEX File?

TYDEX is an abbreviation of "Tyre Data Exchange". A TYDEX file is a special format widely used in the automotive industry to export tire measurement data. It is a ASCII-file with the extension .tdx that can be opened and read without specialist software. The TYDEX format aims to make it easier for companies and institutes to exchange tire data. It is usually directly generated by the tire testbed measurement software.

You can find an example TYDEX file in the FormulaCarMaker project folder under "Data/Tire/Examples/TUTO_TYDEX_TEST.tdx"

### Content of a TYDEX file

#### Coordinate System

The TYDEX format utilizes three different coordinate systems in which the values are measured:

**TYDEX-C** The origin lies in the center of the wheel. The x-axis is in the central plane of the wheel and is parallel to the ground, the y-axis is identical to the rotation axis of the wheel (thus it may not be parallel to the ground in case of non-zero camber angle) and the z-axis points upwards and is perpendicular to the x-y-plane. Therefore this coordinate system moves with slip and camber.

**TYDEX-H** The origin also lies within the center of the wheel. The x-axis is in the central plane of the wheel and is parallel to the ground, the y-axis is perpendicular to the x-axis and is also parallel to the ground, and the z-axis points upwards and is perpendicular to the track surface. This axis moves with slip but remains perpendicular to the road with camber change.

**TYDEX-W** This convention is similar to the H-coordinate system, but its origin lies at the center of the contact patch on the track surface. All axes are oriented like in the H-coordinate system. Therefore it moves with slip, but also remains perpendicular to the road with camber change.

#### TYDEX Structure

Each TYDEX file is separated into individual paragraphs. These paragraphs serve to structure the tire data. In total there are 14 paragraphs, but they are not all required. However, the more information is available the better the resulting tire model will be. Each paragraph is introduced by one of the following key words:

```
**HEADER
**COMMENTS
**CONSTANTS
**MEASURCHANNELS
**MEASURDATA
**MODELDEFINITION
**MODELPARAMETERS
**MODELCOEFFICIENTS
**MODELCHANNELS
```

```
**MODELOUTPUTS
**MODELEND
**END
```

The two most important blocks are "**MEASURCHANNELS" and "**MEASURDATA". The first defines the variables for which values are available. In the second block these values are then listed. While editing you must be aware of the order: Every line of the "**MEASUR- CHANNELS" paragraph belongs to a corresponding column in the "**MEASURDATA" block. The first row of the "**MEASURCHANNELS" paragraph provides, for instance, the lateral force $F_y$. The first column in the "**MEASURDATA" block must therefore contain the measured data of the lateral force. All acting forces (tangential force $F_x$, lateral force $F_y$, tire load $F_z$) and torques (pitching moment $M_x$, driving/braking torque $M_y$, self-aligning moment $M_z$) on the tire can be retrieved. Moreover, there are several other parameters such as slip and inclination angle. You can find a complete list of all possible parameters on the following website:

```
http://www.fast.kit.edu/download/DownloadsFahrzeugtechnik/
TY100531_TYDEX_V1_3.pdf
```

## Data with Camber Angle

The lateral force applied at the contact patch has three components (the same applies for the self-aligning torque):

- one resulting from the slip angle ($F_{slip}$),
- another (constant) caused by the camber angle ($F_{camb}$),
- and a third due to the distortion of the contact surface area ($F_{dis}$).

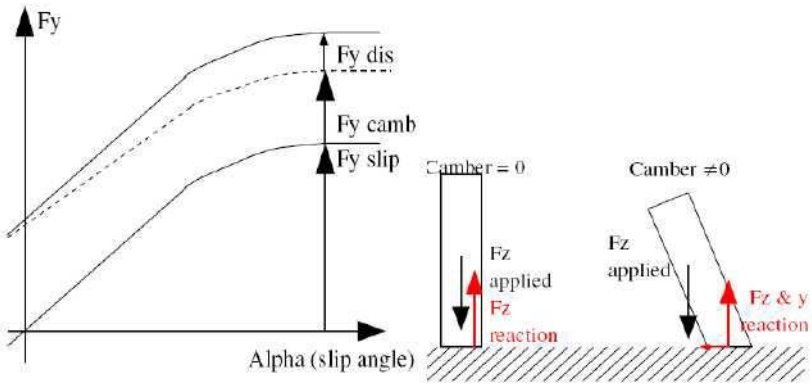



Figure 5.53: Components of the lateral force [TIR07]

In the case of an automobile tire, the last component can be neglected - as opposed to a motorcycle tire. The component due to the camber angle can be found easily using a state equilibrium calculation. This calculation is completed automatically by CarMaker.

CarMaker does not require two curves with the same term. For instance: if you have two measures with $F_y$ vs. slip angle with the same vertical load but with different camber angle values, it is unnecessary to store both of these in the TYDEX file as only one camber angle measurement is required by CarMaker.

## Offset coefficients

You can specify offset and scaling factors to be applied to the values. These factors are specified in the "**MEASURCHANNELS" block.

## Syntax of a TYDEX file

To enter your data correctly, you should use the following advice.

All characters placed after the symbol "!" are comments which will be ignored by the program.

**Measurchannel**

The syntax of the Measurchannel is:

```
**MEASURCHANNELS
PARAMETER_KEY1  comment1  unit1  correction factors value1
PARAMETER_KEY2  comment2  unit2  correction factors value2
PARAMETER_KEY3  comment3  unit3  correction factors value3
```

and so on...

Between each section of a line you should leave a certain number of blank spaces; pay attention to not use Tab-spaces:

- the first character of the parameter key starts at the first character of the line
- the first character of the comment is the 11th character of the line
- the first character of the units is the 42nd character of the line
- the first character of the first factor is the 52nd character of the line
- the first character of the second factor is the 62nd character of the line
- the first character of the third factor is the 72nd character of the line

The parameter key must be written according to the list available on the associated website. The parameter key also defines which axis system is used for the values: for example, "FZH" defines a vertical load in the TYDEX-H axis system and "FZY" defines the same vertical load in the TYDEX-C axis system.

**Measurdata**

Within this block, the values on each line describe a particular point of the curve. A new line defines a new point. Each value must be separated another by at least one blank space.

For instance, below 3 points are defined. In each line the first value is the vertical force, the second the slip angle, the third the camber angle, the fourth the lateral resulting force and the fifth the self-aligning torque. In the below example the "**MEASURCHANNELS" block is displayed again for recall:

```
**MEASURCHANNELS
FZH    Vertical force         N     1    0     0
SLIPANGL Slip angle         deg    1     0     0
INCLANGL Inclination angle        deg    1     0     0
FYH    Lateral force        N     1     0     0
MZH    Aligning moment       Nm    1     0     0
**MEASURDATA
1211.00 -0.50  0.00  211.21 -2.27
3021.70 -0.50  0.00  460.01 -8.08
4828.00 -0.50  0.00  641.08 -15.50
```

If, for example, you don’t have the values of the self-aligning torque, you should write:

```
**MEASURCHANNELS
FZH    Vertical force         N     1    0     0
SLIPANGL Slip angle         deg    1     0     0
INCLANGL Inclination angle        deg    1     0     0
FYH    Lateral force        N     1     0     0
**MEASURDATA
1211.00 -0.50  0.00  211.21
3021.70 -0.50  0.00  460.01
4828.00 -0.50  0.00  641.08
```

Furthermore, it is possible to define several pairs of "**MEASURCHANNELS" and "**MEASURDATA" blocks within the same file. Thus, it is a good idea to define each curve using a special pair of blocks.

However, the former examples are only extracts from a file. Copying and pasting the examples in their current form into a file will generate an error message as there is only one value in the entire column for the inclination angle. If you only have one value for a certain parameter you must leave it out.

## Converting a TYDEX file for CarMaker

Although a TYDEX file contains all the data required by CarMaker, it can't be simply embedded as the data must be transformed first. Therefore, IPG Automotive offers a tool called "tireutil.exe" that generates the two files needed by CarMaker to model a tire.

To implement this tool it must be located in the same folder as your TYDEX file. You can find it in the installation folder "IPG", usually under: "C:\IPG\hil\win32-versionNumber\bin". Copy it into your current project directory in the folder "FormulaCarMaker\Data\Tire" where the TUTO_TYDEX_TEST.tdx file is saved.

Now, hit *Windows Start button* > *Run*. To open the command line window type "cmd". To use "tireutil.exe" you have to browse to your project folder, for instance "C:\CM_Projects\FormulaCarMaker_Release202X.X\Data\Tire". In the command line window type "C:" and hit "enter". To open the respective folder type the following line and hit *enter*:

```
"C:\>cd CM_Projects\FormulaCarMaker_Release202X.X\Data\Tire".
```

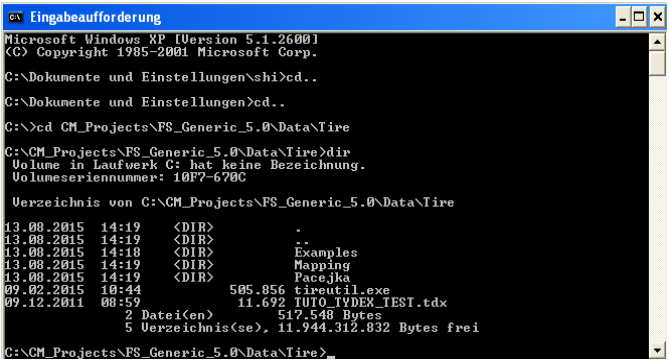



Figure 5.54: Windows command line

Once you are in the right directory you have to call the "tireutil.exe" command. You must specify the input file (your TYDEX file, e.g. "TUTO_TYDEX_TEST.tdx") and the output files (a file without extension and a file with the .tir extension). The name of the output files can be different from the input file, and must be written without extension:

```
C:\CM_Projects\FormulaCarMaker_Release202X.X\Data\Tire > tireutil -if
TUTO_TYDEX_TEST.tdx -of TUTO_TYDEX_TEST -ofbin Mapping/TUTO_TYDEX_TEST.bin
```

The file generation will then start (according to the tydex file weight, it can last several seconds). Once the generation is finished the command line displays:

```
C:\CM_Projects\FormulaCarMaker_Release202X.X\Data\Tire>_
```

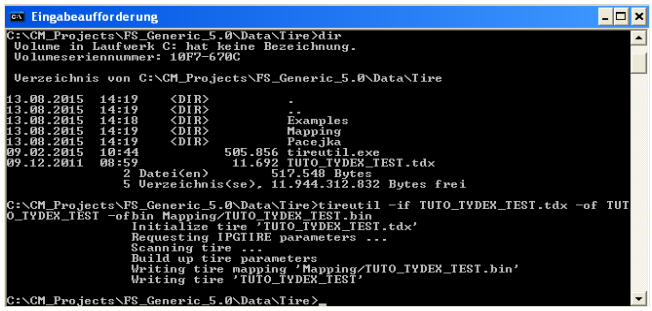



Figure 5.55: Initializing Tire

Open your Windows Explorer and browse to the folder “TIRE”. You should now see, in addition to `TUTO_TYDEX_TEST.tdx`, a new file named TUTO_TYDEX_TEST (control file). The generated TUTO_TYDEX_TEST.bin (data file) was directly attached to the “Mapping” folder. You can close the command line window.

**Exercise 13**

- Complete the changes described previously with your own files.
- You have now created your own tire file successfully! Open CarMaker and test the new tire: in the *Tires* selection area you will find your new tire under the name "TUTO_TYDEX_TEST". Select it and perform a simulation.

## 5.6.3 TameTire

*Tame Tire* is a physical tire model developed by the Michelin R&D Team. The model aims to accurately describe the transient mechanical and thermal behavior of the tire. It is a part of the standard CarMaker package, but also requires an additional license.

For detailed information about the TameTire model please refer to the TameTire documentation and the section 17.6 *‘Tame Tire’* of the Reference Manual.

## 5.6.4 Tire Data Set Generator

The *Tire Data Set Generator* is a tool used to create generic tire data sets for a specific vehicle classes and tire dimensions. The algorithm is based upon the Magic Formula approach and the tire characteristics defined by the European Standard Tire and Rim Technical Organization (Abbreviated as ETRTO).

Based on user inputs and the ETRTO references, the relevant parameters are determined analytically. Measurement curves which define the tire force and torque characteristics are generated with the help of Pacejka’s Magic formula. The results are saved as measurement curves in the TYDEX format. (See section 5.6.2 for more information about TYDEX). Once the TYDEX file has been created, the Tire Data Set Generator calls the *tireutil* application to convert the TYDEX file into an infofile and a binary file that can be used by CarMaker.

The Tire Data Set Generator is available in the *File* menu of the Tire Data Set Editor. Figure 5.56 describes the path for accessing the Tire Data Set Generator.

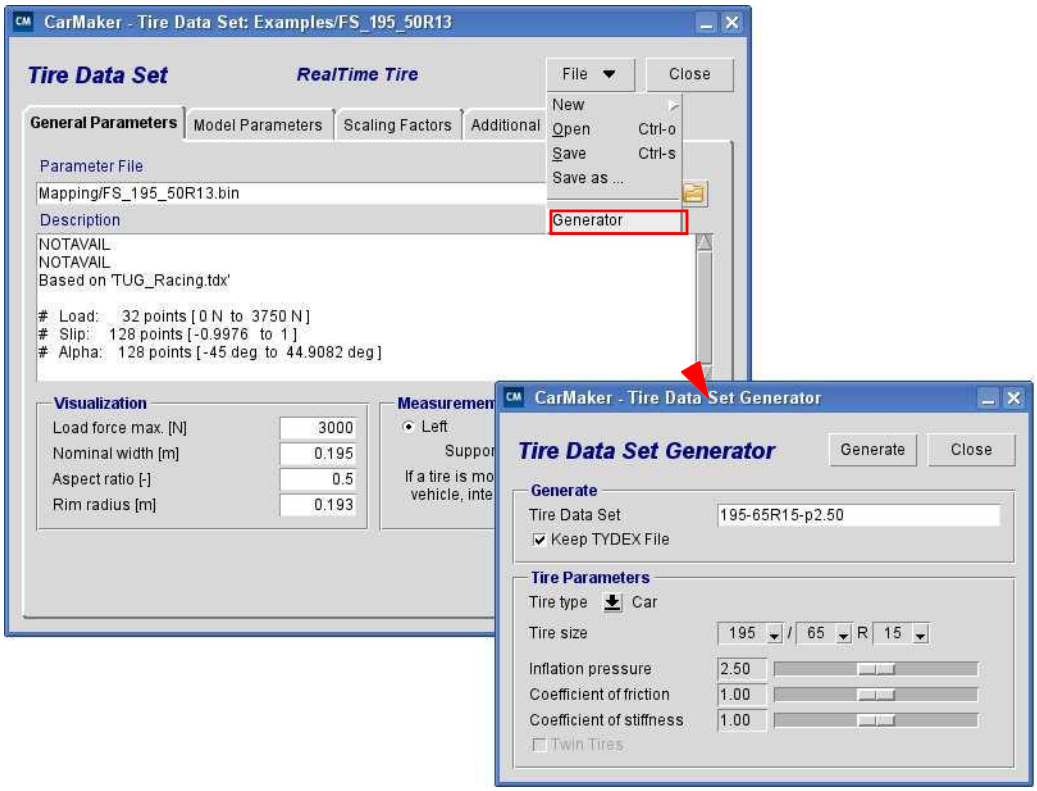



Figure 5.56: Opening the Tire Data Set Generator

For further information, please refer to Chapter 11.9 '*Tire Data Set Generator'* in the User's Guide.

Chapter 6

# Simulation and Model Validation

## Introduction

Any preexisting or self-generated vehicle cannot serve as a reference so long as it hasn't been validated. This is due to the many estimations, assumptions and potential mistakes made during the parametrization process. Before starting the model validation, a few plausibility checks should be performed to eliminate fundamental input errors. In this regard, the suspension models and kinematics results should be checked as well.

Once the plausibility checks are complete the validation process can be started. To which extent it is practiced mainly depends upon the available time, equipment and budget. If the relevant resources can't be provided to a sufficient degree, the accuracy of measurements should have a higher priority than the quantity of measurements. In that case, even more attention should be paid to the plausibility checks.

# 6.1 Plausibility Checks of Axle Models

## 6.1.1 Variation of Camber and Inclination vs. Wheel Travel

The variation in camber of an axle can easily be defined as a function of wheel travel vs. change in inclination angle ($\Delta\sigma$). The resistances that exist are neglected. Concerning a double wishbone axle, both control arms (lengths e and f) move on circular paths about the joints on the body (C and D). If the lower joint between the wishbone and the suspension is lifted or lowered by the distance $s_1$, the inclination change (and respectively, the actual angle of inclination change) can be determined. In the case of a double wishbone axle, the angle of inclination change is the same as the angle of camber change and thus the camber angle can be estimated using the same test.

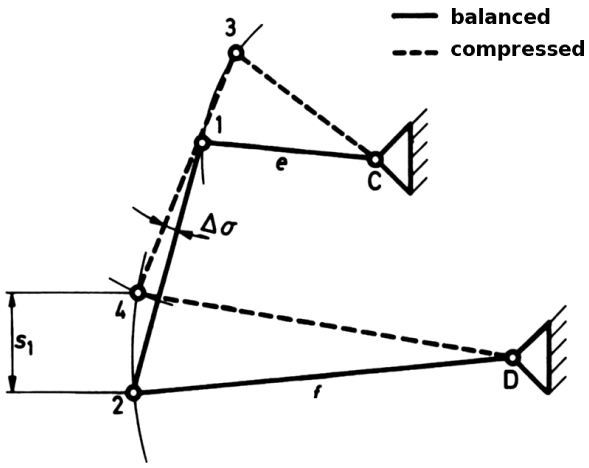



Figure 6.1: Determination of the angle of inclination and camber change [RB00]

The process explained above needn't be done with drawings but is a lot more precise with the use of CAD programs. All kinematic points are inserted into a sketch and the lengths of the connecting lines and the two joints between the wishbone and chassis are fixed. By adjusting the point "Lower Wishbone - Suspension", compression can be simulated.

Table 6.1: Camber and toe change

| wheel travel $s_1$ | camber measured | camber calculated | inclination measured | inclination calculated | relative error |
|---|---|---|---|---|---|
| 30 mm | -1.61 deg | -1.46 deg | 16.26 deg | 16.26 deg | 0.00% |
| 25 mm | -1.07 deg | -1.20 deg | 15.99 deg | 15.86 deg | 0.82% |
| 20 mm | -1.61 deg | -0.95 deg | 15.71 deg | 15.60 deg | 0.71% |
| 15 mm | -0.79 deg | -0.71 deg | 15.44 deg | 15.36 deg | 0.52% |
| 10 mm | -0.52 deg | -0.47 deg | 15.17 deg | 15.12 deg | 0.33% |
| 5 mm | -0.26 deg | -0.23 deg | 14.91 deg | 14.88 deg | 0.20% |
| 0 mm | 0 deg | 0 deg | 14.65 deg | 14.65 deg | 0.00% |
| -5 mm | 0.25 deg | 0.23 deg | 14.40 deg | 14.43 deg | 0.21% |
| -10 mm | 0.51 deg | 0.45 deg | 14.14 deg | 14.20 deg | 0.42% |
| -15 mm | 0.96 deg | 0.67 deg | 13.89 deg | 13.98 deg | 0.65% |
| -20 mm | 1.01 deg | 0.89 deg | 13.64 deg | 13.77 deg | 0.95% |
| -25 mm | 1.25 deg | 1.10 deg | 13.40 deg | 13.55 deg | 1.11% |
| -30 mm | 1.50 deg | 1.31 deg | 13.15 deg | 13.34 deg | 1.44% |

The relative error is always less than 2%. It results from resistances which exist within the bearing which are not considered in the measurements - unlike IPGKinematics.

### 6.1.2 Track Change

Compression of the wheels causes a change in track in almost every suspension configuration, including double wishbone axles. Thus, this track change should be checked as well. It can be done using the sketch below. Add the wheel contact point and fix the connecting line of the two joints between wheel carrier and body.

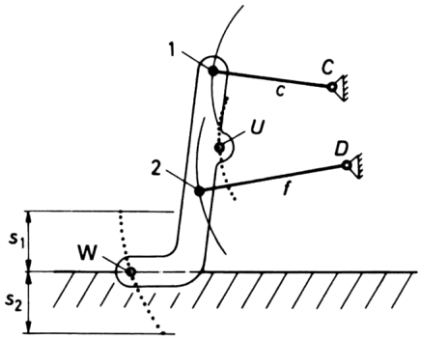



Figure 6.3: Determination of the track change [RB00]

### 6.1.3 Toe Change

A change in toe angle due to the compression of the wheels can be used to achieve certain driving characteristics. In general, the toe change should be in the range of minutes and never above as this can lead to unpredictable handling of the car.

If IPGKinematics produces a nearly constant toe change vs. wheel travel this result is likely accurate and a check-up isn't necessary. But if it varies, or exceeds the minutes-range, the model should be checked.

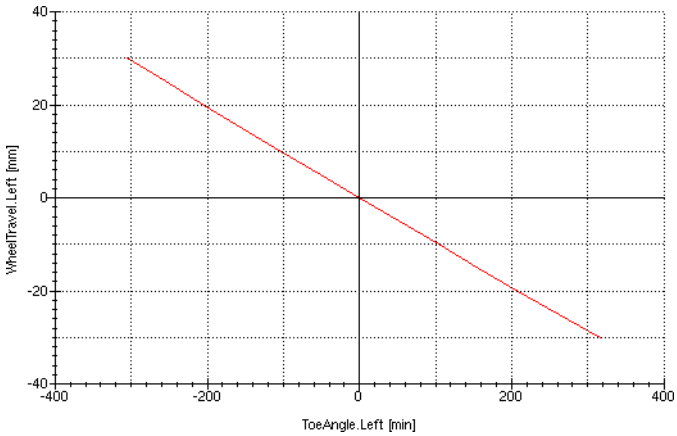



Figure 6.4: Toe change vs. wheel travel

If the toe change is as large as is shown in the above diagram, there is probably a mistake in the model's parametrization. Of course, this is not unusual and you should check your model carefully. Pay attention in particular to the coordinates of the wishbones, the axle joints and the tie rod. If you can't find a mistake with your model, you should also check if the calculations are correct. This explanation isn't ideal, but is still technically possible! To check this out, a measurement should be performed directly on the car.

To avoid any compression inducing forces, the car is jacked up and the wheels are disconnected from the spring/damper unit. Before doing so, the wheels should be put on a vertically adjustable platform so that the wishbone bearings don't support the entire weight of the system. With the use of an angle plate it is possible to measure the true toe angle.

As the tire deflects at the contact patch, it will become wider at this point. For this reason the angle plate should be attached to the rim and never to the tire. Therefore the angle plate must feature spacers to equate the distance between rim and tire. The following figure shows such an angle plate.

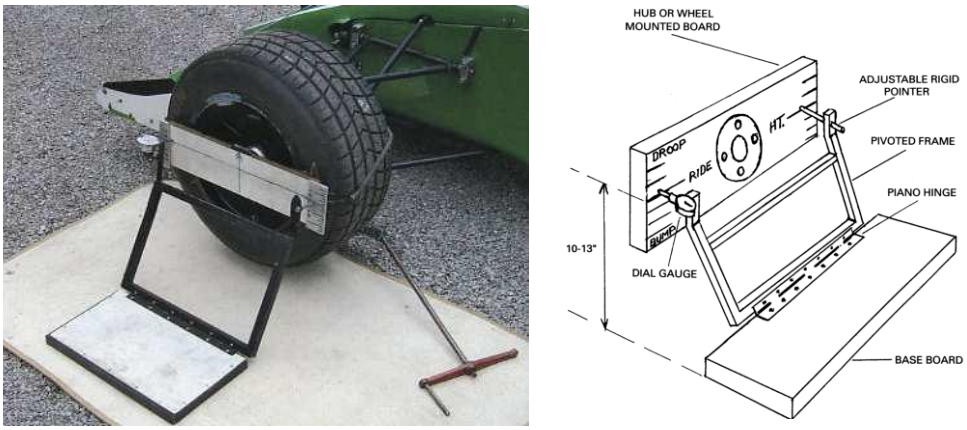



Figure 6.5: Angle plate used to measure the toe angle

By lifting and lowering the plate under the tire, the compression can be simulated and the corresponding toe angle can be measured. During the measurements it is important to brace the steering to exclude the influence of steering angle.

After measuring the toe angle directly the results can be compared with the software's calculations. If both values are the same then your model is correct. Otherwise, it should be modified. This can be done by adjusting the steering system, specifically the position of the tie rod. Further explanations on this can be found in the following paragraph.

### 6.1.4 Steering Angle/Steering Ratio

After checking the toe change the steering system should be reviewed. To measure the steering angle, the same configuration as described above can be used. Another option is to place the car on the ground to perform the required measurements . If the driver was included in the former calculations of the design position, this should also be the case when taking measurements to prevent errors.

Once the car is on the ground, firstly the steering ratio should be verified. Therefore, the best way is to induce a certain steering angle and measure the corresponding rack travel. To verify the result, the test can be performed in the opposite direction.

After checking the steering ratio, the steering angle is next. For this, you can use the angle plate again by attaching it to the rim. Once you have marked a reference line, the steering angle (or the rack travel) can be increased incrementally while recording the corresponding steering angles at the right and left side.

### 6.1.5 Turning Track Diameter

The turning track diameter is the circular arc followed by the outer wheel's contact patch using a fixed steering angle at slow speeds. It is common to refer to the smallest possible arc as the turning track diameter. It is attained by applying the maximum steering angle. Hence, measuring the turning track diameter requires measuring the steering angles. For this procedure, a driver isn't required. It is sufficient to fix a certain steering wheel angle or a steering rack travel and then push the car along a circular path. With this procedure it is a lot easier to maintain a given steering angle than it is with a driven car.

Table 6.2: Checking the turning track diameter

| steering rack travel | turning track diameter measured | turning track diameter calculated | relative error |
|---|---|---|---|
| 0 mm | infinite | Infinite | 0.00% |
| 5 mm | 69.00 m | 68.12 m | 1.29% |
| 10 mm | 34.50 m | 34.06 m | 1.29% |
| 15 mm | 23.00 m | 22.71 m | 1.28% |
| 20 mm | 17.20 m | 17.05 m | 0.88% |
| 25 mm | 13.80 m | 13.65 m | 1.10% |
| 30 mm | 11.60 m | 11.39 m | 1.84% |
| 35 mm | 10.00 m | 9.78 m | 2.25% |
| 40 mm | 8.80 m | 8.58 m | 2.56% |

### 6.1.6 Anti-Dive and Anti-Squat

While braking, the front axle is compressed by the distance $\Delta s_f$ and the rear axle by the distance $\Delta s_r$. The pitch angle $\varphi$ [rad] can be calculated using the two distances and the wheelbase $l$:

$$\varphi = \frac{\Delta s_f + \Delta s_r}{l} \quad \textbf{(EQ 30)}$$

The deforming of the rods due to compression is reduced by a using double wishbone axle. If the brake is located within the wheel, the wishbones must 'twist' in order to counteract the emerging forces. However, if the brake is located centrally at the differential, the wishbones are deformed in the same rotational direction. If the wishbones are orientated in a parallel configuration, there is no anti-dive or anti-squat.

# 6.2 Vehicle Data Set Plausibility Checks

## 6.2.1 Center of Gravity and Moments of Inertia

In section 5.1.1 '*Vehicle Body'*, pg. 76 the determination of the vehicle's center of gravity was explained in detail. If the process wasn't performed as described in that chapter, the results should be reviewed again. The generated tcl-chart can serve as a rough approximation. Here, the points entered for the individual centers of gravity should be placed at feasible positions. To inspect the dimensions of the moments of inertia, each component should be replaced by a similar geometric body whose moment of inertia can be determined exactly. Therefore, simple bodies such as rectangles, cylinders or spheres are sufficient.

## 6.2.2 Comparability of IPGKinematics and CarMaker

A few common parameters are required by both programs, IPGKinematics and CarMaker, including: front and rear stabilizer bar properties, spring rates and spring characteristics front/rear, masses and axle loads. Each of these parameters should have the same value in both programs. Most of the parameters are defined in the same way, the only exception being the stabilizer bar. It is defined in different terms, explained in section 5.1.4 '*Suspensions*', pg. 81.

## 6.2.3 Assignation of the Spring Length $l_0$ at Front and Rear Axle

In section 5.1.4 '*Suspensions'*, pg. 81 the spring rates for front and rear axles were entered but the *spring lengths l0* have been omiited for the following reason. If you chose a linear kinematic model the length l0 serves as adjustment parameter. Using non-linear models, the length l0 is the actual free length of the spring. The determination of this parameter is done via the static equilibrium configuration of the vehicle. To perform an equilibrium calculation, you can use the CarMaker *Model Check* (CarMaker Main GUI: *Simulation* > *Model Check*, see section 3.1.7 *'Model Check'*, pg. 36). With this tool the user can create and analyze specific diagrams relating to the entire car. One of the options is the analysis of the static equilibrium configuration.

To activate this tool, deselect all options except "Vehicle Characteristics". Activate both options in this section and the tool can then be started ny hitting the *Generate Diagrams* button. After completing the calculations, a text file opens containing the results. At the beginning of this file, all the important input parameters are listed. At the end you can find the results of the static equilibrium calculation. The following shows an extract of such a file which contains the results of the static equilibrium calculation.

```
### Geometry (equilibrium or start-off configuration)

                :              x              y              z
 VhclPoI        :          1.383 m        0.000 M        0.529 m  FrD
                :          1.383 m        0.000 M        0.529 m  Fr1
                :          2.386 m       -0.000 m        0.506 m  Fr0

 Fr1 Origin     :          1.000 m        0.000 m       -0.018 m  Fr0
 Fr1 Roll       :          0.000 deg  Fr0.X
 Fr1 Pitch      :          0.229 deg  Fr0.Y
 Fr1 Yaw        :          0.000 deg  Fr0.Z


 GenBdy1        :          1.314 m       -0.000 M        0.316 m  FrD
                :          1.314 m       -0.000 M        0.316 m  Fr1
                :          2.315 m       -0.000 m        0.293 m  Fr0

 ConBdy1        :          1.334 m        0.000 m        0.328 m  FrD
                :          1.334 m        0.000 M        0.328 m  Fr1
                :          2.335 m       -0.000 M        0.305 m  Fr0

                :      FL             FR             RL             RR
 carrier WC tx : 2.2667206 m     2.2667206 m     0.4999997 m    0.4999997 m  FrD
           ty : 0.6049116 m    -0.6049116 m     0.6310333 m   -0.6310333 m  FrD
           tz : 0.2678416 m     0.2678416 m     0.2616688 m    0.2616688 m  FrD

 carrier WC tx : 2.2667206 m     2.2667206 m     0.4999997 m    0.4999997 m  Fr1
           ty : 0.6049116 m    -0.6049116 m     0.6310333 m   -0.6310333 m  Fr1
           tz : 0.2678416 m     0.2678416 m     0.2616688 m    0.2616688 m  Fr1

 carrier WC tx : 3.2677741 m     3.2677741 m     1.5010427 m    1.5010427 m  Fr0
           ty : 0.6049116 m    -0.6049116 m     0.6310333 m   -0.6310333 m  Fr0
           tz : 0.2409331 m     0.2409331 m     0.2418293 m    0.2418293 m  Fr0

carrier Mnt tx: -0.0002794 m   -0.0002794 m   -0.0000003 m   -0.0000003 m  Fr1
           ty : -0.0000884 m    0.0000884 m    0.0000333 m   -0.0000333 m  Fr1
           tz : 0.0008458 m     0.0008458 m    -0.0003271 m   -0.0003271 m  Fr1


 compression q0  : 0.0009741 -    0.0009741 -   -0.0003483 -   -0.0003483 -
 coordinate q1   : -0.0000000 -   -0.0000000 -   -0.0003483 -   -0.0003483 -
 spring coord    : 0.1751192 m    0.1751192 m    0.1426429 m    0.1426429 m
 damper coord    : 0.0989997 m    0.0989997 m    0.1003696 m    0.1003696 m
 buffer coord    : -0.0008458 m   -0.0008458 m    0.0003271 m    0.0003271 m
 stabi coord     : 0.0000000 -    0.0000000 -    0.0000000 -    0.0000000 -

 camber          : -0.0140927 deg  -0.0140928 deg    0.0159658 deg    0.0159656 deg
                 : -0.8455619 min  -0.8455700 min    0.9579481 min    0.9579378 min

 toe             : -0.1664832 deg  -0.1664846 deg   -0.0000944 deg   -0.0000944 deg
                 : -9.9889940 min  -9.9890780 min   -0.0056652 min   -0.0056651 min

 caster          : -0.0390493 deg  -0.0390497 deg    0.0000424 deg    0.0000424 deg
                 : -2.3429591 min  -2.3429791 min    0.0025417 min    0.0025417 min

 wheel center Fx : -2.7001 N    -2.7001 N      -3.8067 N       -3.8067 N Fr1
              Fy : 0.2095 N      -0.2095 N      -0.2650 N       0.2651 N
              Fz : 813.3719 N    813.3721 N     951.2038 N      951.2054 N

 wheel road   Fx : 0.0000 N       0.0000 N       0.0000 N       0.0000 N FrH
              Fy : -0.0000 N      0.0000 N       0.0000 N       -0.0000 N
              Fz : 813.3764 N    813.3767 N     951.2114 N      951.2131 N

 ----------------------------------------------------------------------------------------------------------------
```

Listing 6.1: Result file of static equilibrium calculation

In the middle of the last paragraph of the *Model Check* result file, there is a point called "carrier Mnt". Using a non-linear model, the variable "tz" in the "carrier Mnt" section should be approximately zero for all four wheels. To achieve, this the front and rear spring lengths must be modified. After each change of l0 a new *Model Check* should be performed until “tz” is adequately small.

If this parameter is not adjusted, the whole static equilibrium configuration of the car is wrong. It expresses whether the car is running under the track surface or above it. It can be easily visualized with IPGMovie. The following picture shows a vehicle without adjustment of the spring length and a car with a correct value for l0.

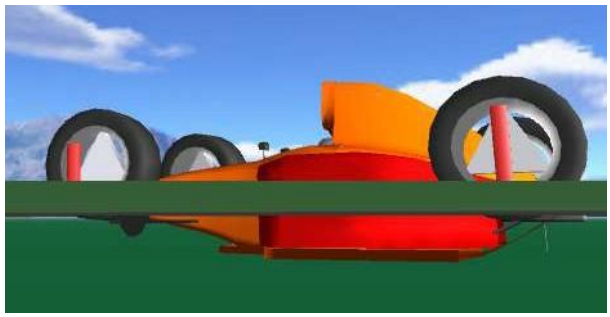


Figure 6.6: Vehicle in a wrong equilibrium state

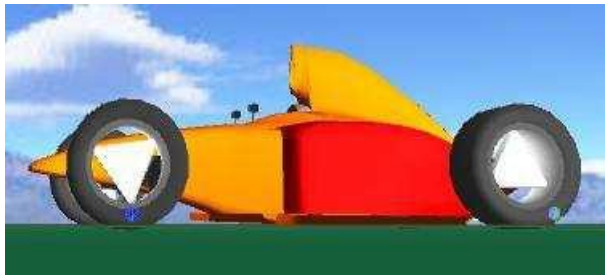


Figure 6.7: Vehicle with a correct spring length l0

# 6.3 Initial Start-up of the Model

## 6.3.1 Common Error Messages

Once you have finished the plausibility checks and the spring length configuration, the actual model validation can be started. The primary step should consist of a first general simulation. The performed TestRun should be as simple as possible to emphasize the car.

The following TestRun's aim is to make the car drive along a straight road. If this works on the first attempt, the remaining part of this chapter can be skipped. However, in the most cases an error message will occur. In the following topics, the most common error messages and their solutions are explained.

For the following example it is assumed you are already familiar with the previous exercises.

**Exercise 14**

- For the first TestRun the following settings should be specified:
  - Road: a long straight with a track width of 10 m.
  - Driver: standard driver set to "normal".
  - Maneuver: a maneuver duration of 100 s. Assign IPGDriver to both controllers.
  - Tire: Formula Student tire "FS_195_50R13" (front) and "FS_205_50R13" (rear).

**"'Idle speed' bigger than 'Minimal engine speed'"**

Either the "idle up/acc down" (Driver dialog) is too big/small or the engine idle speed (*Vehicle Data > Powertrain > Engine*) is too small/big. Adjust these to fit the settings.

**"Suspension front left: Wheel and carrier must have the same position"**

As explained in section 5.1.1 *'Vehicle Body'*, pg. 76 the center of gravity of the wheel must be the same as that of the wheel carrier. If all four centers of gravity are different, a simulation can't be executed. If the centers of gravity of three of the four wheels are different, the simulation starts but delivers three error messages. This can be reviewed in the session logbook (*Simulation > Session Log*). Here, all warnings and error messages of the current session are stored. Once CarMaker is restarted, a new session opens. After correcting the positions of centers of gravity of the wheels this error will not occur.

**"Wrong number of elements or syntax error in 'PowerTrain.GearBox.iBackwardGears'..."**

Again, this is a simple faulty insertion. As mentioned in section 5.2.2 *'Drive Source'*, pg. 93 "Powertrain: Gearbox" a reverse gear must be defined, although Formula Student cars usually don't have one.

**"Can't get parameters for body sensor 'Jack.fr'"**

This problem is similar to the non-existent reverse gear. The program needs one point for all four jack sensors. It is possible to place all four points on the origin.

**"Suspension KnC front left: No kinematics selected, model number 0"**

Here, the existing model is correct in most cases. The problem may be that the selected front axle kinematics is for a rear axle. This mistake can happen easily when selecting the skc-files. If you choose the right skc-file this problem will be solved.

**"Suspension KnC 'MapNL' rear left: damper length is not decreasing monotonously (0.071741 (0,6) .. 0.0758616 (1,6))"**

This error indicates that a wrong value was entered in the kinematics specifications. The indicator "rear" (respectively "front") shows the user which skc file is to be reviewed. As the damper length doesn't increase consistently, the mistake will associated with the coordinates of the damper. Besides this error, similar messages also indicate that the spring length doesn't increase uniformly, either. The problem will lie in the definition of the spring/damper elements. In Formula Student cars, push-/pullrods are mainly used. For this reason, the parameters "Damper - Body", "Damper - Lower Wishbone", "Spring - Body" and "Spring - Wheel Suspension" might not have been regarded or set to 0. In *'Geometry'*, pg. 67 it was specified that these parameters must have sensible values nonetheless. If this wasn't carried out an error emerges.

## Fault analysis and fault isolation

If you receive error messages different to those described above, a detailed fault analysis must be performed. If the error message does not tell you which parameter is affected nor in which section the problem lies, a review of each and every variable should be carried out to identify the missing point or wrong factor.

If you still don't find a mistake, another method should be applied. CarMaker offers the option to import data sets from other TestRuns. All subsystems - either fully defined to partial models - and even their subsystems, like e.g. dampers in the suspensions tab, can be imported from other vehicle data sets (*File* > *Import*).

Thus, there is the functionality to import the data of your car step by step into an existing, operating model until the problem is found. However, this reference model should be similar to your car (e.g. the Formula Student car "FS_RaceCar7" introduced in this document) to avoid new problems. In any case, different masses between the reference car and the newly generated skc-files must be removed.

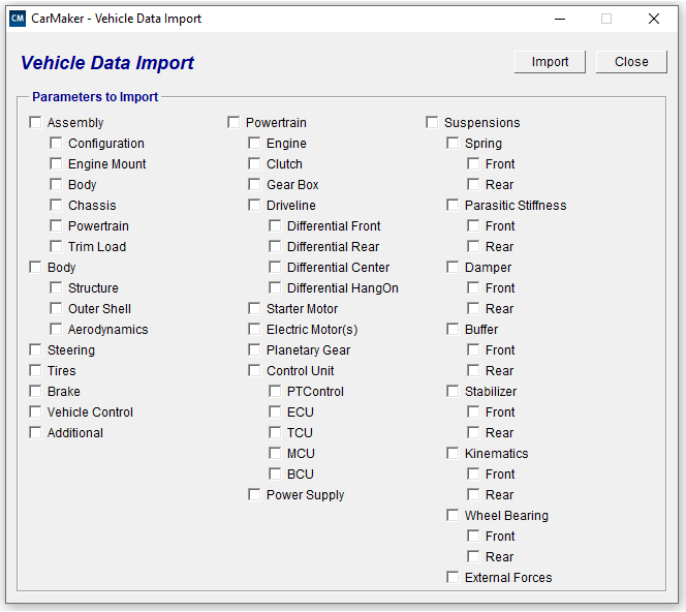



Figure 6.8: Submenu to import vehicle data sets

## 6.3.2 Driver Adaption

After the simulation runs without any errors or warnings, you can proceed to the next step: performing and analyzing initial TestRuns with your car. But before doing so, a driver adaption should be carried out (*Simulation* > *Driver Adaption*). This is not compulsory as the controller eventually learns its limits, but in the case of racing drivers such as used in FScars an adaption is advisable.

During the driver adaption the user can observe whether the car acts in a predictable way. If this is not the case or if the adaption can't even be finished, another parameter checkup should be performed. For this, see the previous chapter.

Once the driver adaption is completed successfully you can start to simulate TestRuns.

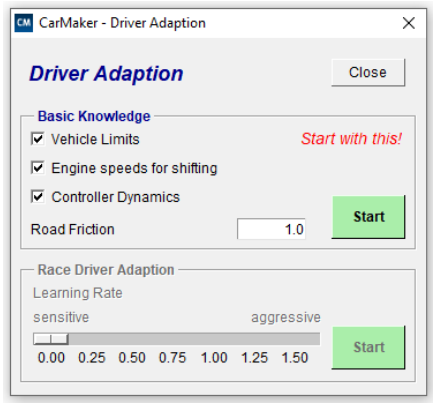



Figure 6.9: Driver Adaption menu

# 6.4 Race Driver Adaption / Lap Time Optimization

The *Race Driver Adaption* is an additional, optional procedure which is required to fully parameterize the Race Driver controller and permit the vehicle to be operated at the limits of its capabilities. This step should only be taken if a user is interested in finding an optimal lap time, etc.

Before this stage of the lap time optimization process can be started, it is important that a Basic Knowledge Driver Adaption has been successfully completed as a Driver Knowledge file is required as a starting point.

The Racing Driver adaption procedure requires driving multiple laps and therefore commands a closed-loop circuit. (i.e. vehicle starting from standstill conditions and not coincident start/finish lines). In order to complete a Racing Driver adaption (i.e. successfully running a certain number of full laps) you may need to use a slightly lower cutting corner coefficient or reduce g-g diagram exponents.

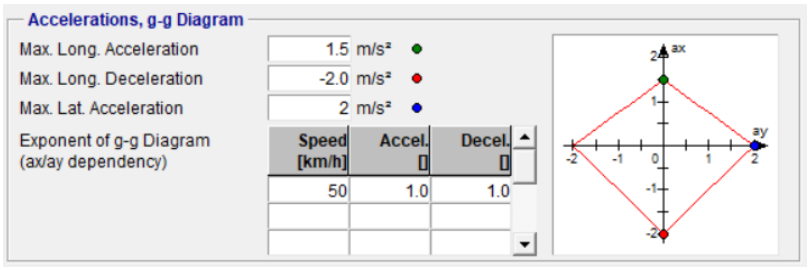



Figure 6.10 G-G Diagram as found upon the Driver GUI

A useful graphic which illustrates the objective of the Race Driver Adaption process is the g-g diagram located upon the Driver GUI (figure 6.10). Within motorsport, g-g diagrams are often used as a method of evaluating the overall performance envelope of a vehicle. This is also true for IPGDriver as the area bounded by the red lines of the diagram represent the area in which the controller will operate the vehicle

## 6.4.1 Max. Accelerations and Exponents

**Max. Accelerations (m/s²)**

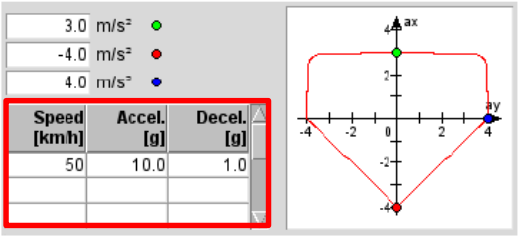
These three values specify the maximum absolute acceleration (purely longitudinal or lateral) and determine the outermost bounds of the diagram as shown by the green, red and blue dots along the axis.

When the Racing Driver Controller is enabled, the input boxes become grayed-out. This is due to these maximum accelerations being programmatically determined during the Race Driver Adaption process, a detailed description of which can be found in section 4.4 Race Driver Adaption pg.88 in the IPG Driver User Manual.

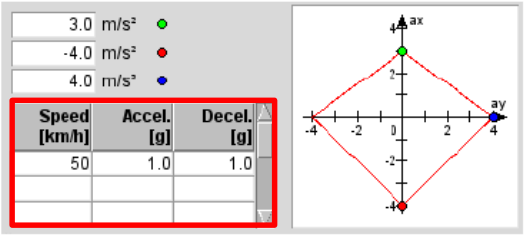
**Exponent of g-g Diagram (ax/ay dependency)**

The exponents define the dependency between the lateral and longitudinal accelerations (Accel. column) and the lateral and longitudinal decelerations (Decel. column). The exponents can be observed on the g-g diagram via the red curves that link the maximum accelerations specified above. In most cases 3 values are sufficient to define the shape.

The controller will drive so that the accelerations applied to the vehicle remain in the area defined by the exponents and the maximum limits (see next picture). A high value (10) will increase the area and a low value (1) will reduce it.




Exponent for the acceleration is set to 10.
Exponent for the deceleration is set to 1.




Exponent for the acceleration is set to 1.
Exponent for the deceleration is set to 1.

## 6.4.2 Race Driver Adaption Process

The Race Driver Adaption process can be summarized by the following flowchart:

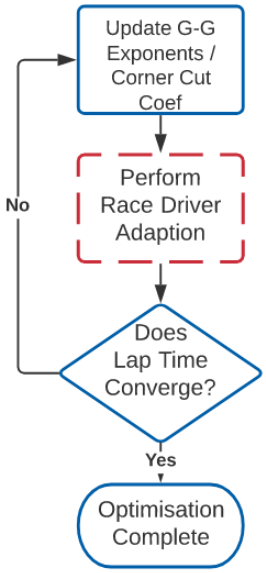



Figure 6.11 Race Driver Adaption Flowchart

The procedure begins with the user entering estimates of feasible exponent values to configure the driver's g-g diagram. The Race Driver Adaption will then iteratively attempt to lower the lap time by approximating the maximal tire slip lap by lap by updating the 'Aerodynamics' table within the *Race Driver* tab. The vehicle can be observed to be doing multiple circuits of the specified track via IPGMovie during this process.

The objective of this optimization stage is to match the controller's g-g diagram to the theoretical one of the vehicles as closely as possible, allowing the driver to extract maximum performance from the car without losing control. The process of manually increasing the exponents is repeated until either: The lap time converges, indictive of the maximum controllable tire forces being achieved or the vehicle leaves the track, in which case the exponents should be reduced.

When starting the optimization of exponents from scratch, it is useful to run an initial simulation using the default exponents and plot graphs in IPGControl of the vehicle's speed (*Car.v*), lateral acceleration (*Car.ay*) and longitudinal acceleration (*Car.ax*) to provide a reasonable starting point (3 speeds should be sufficient, e.g., 20, 40 and 60 kph).

The 'SnapShot' feature (*section 3.3.2, pg 47*) in IPGControl allows the user to easily compare lap times and determine whether each iteration is an improvement.

### 6.4.3 Learning Rate

The *Learning Rate* is a parameter which can be adjusted to control the magnitude of the changes made during the adaption process and provides another means of fine-tuning the driver controller. This is achieved by moving the slider at the bottom of the Driver Adaption GUI towards the left (*Sensitive*) or to the right (*Aggressive*). A Sensitive learning rate will make smaller changes, whilst the opposite is true for an Aggressive setting.

Although not always a necessary step, this parameter can be used to combat undesirable handling characteristics produced during the Race Driver Adaption, often indicated by inconsistent lap times. Depending on the telemetry produced, the Learning Rate should be adjusted accordingly - for example, for an oversteering vehicle it is recommended to opt for sensitive values and vice versa.

The Learning Rate should only be adjusted once a user is satisfied with the exponents of their g-g diagram. For an example on this scenario, please refer to IPGDriver User Guide section 5.5 '*Use Case: Lap Optimization*' pg. 104.

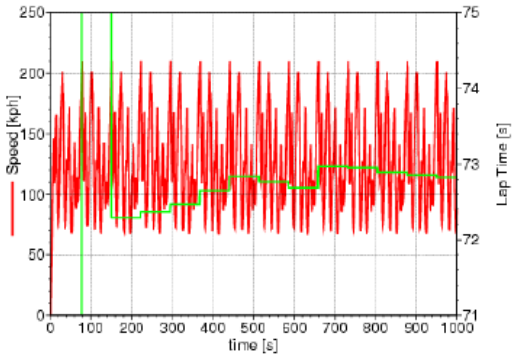



Figure 6.12 - Telemetry displaying inconsistent lap times (i.e., non-converging)

### 6.4.4 Summary

To conclude, the lap time optimization process can be summarized into the following steps:

1. Ensure vehicle parameterization is complete and produces no errors when simulated.
2. Set a benchmark using IPGControl (optional).
3. Complete a *Driver Adaption - Basic Knowledge.*
4. Perform a *Driver Adaption - Race Driver Adaption.*
5. Iteratively refine g-g diagram exponents.
6. Adjust Racing Driver *Learning Rate* if required.

The above steps should be recompleted whenever significant changes are made to the vehicle's configuration as these will affect performance characteristics.

Answers to common questions regarding IPGDriver's functionality and operation can be found in the *FAQ* section of the IPGDriver User Guide pg. 158.

# 6.4 TestManager

With this tool you can programmatically perform several TestRuns, one after another, or even simulate a TestRun loop which varies selected variables.

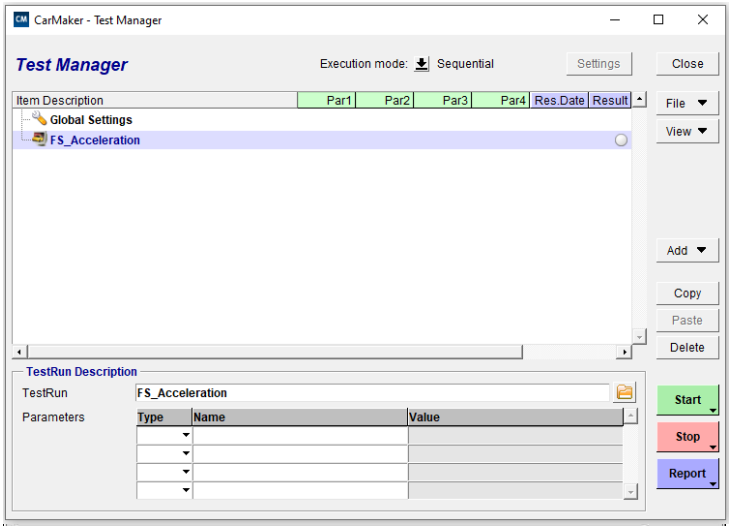



Figure 6.13: TestManager main window

One method to perform parameter variations is to define several TestRuns with different conditions, save them and perform one simulation after the other manually. The *TestManager* makes this process a lot easier. Simply select the TestRuns you have prepared by choosing *Add* > *TestRun* and select the relevant TestRuns from the library.

The loaded files are then displayed in the TestManager. Press the *Start* button and the simulations will be performed automatically. Furthermore, the TestManager provides flexibility in selecting individual TestRuns to simulate.

Don't forget to choose the option *save all* at the *Storage of Results* box for each TestRun. Otherwise each simulation will be performed, but the results will be only saved to a 'buffer' and then be overwritten during the next simulation!

## 6.4.2 Variations and Variable kinds

Instead of creating a new TestRun for each variation of a single quantity, you can also insert a loop using the TestManager. This can be done with the help of inserting *Variations* for the desired variables. Create the Variation by clicking *Add* > *Variation* for the selected TestRun. This means the same TestRun is ran several times but with different user defined Variations e.g. Longitudinal acceleration in each instance.

Variations can be defined using different variables e.g. NValue or KValue. NValues (abbr. Named Values) are used to change the parameters in the CarMaker GUI e.g. coordinates of CoG. On the other hand, KValues (abbr. Key Values) are used to change info-file parameters that do not have an editable parameter in the CarMaker GUI e.g. Gear Ratio.

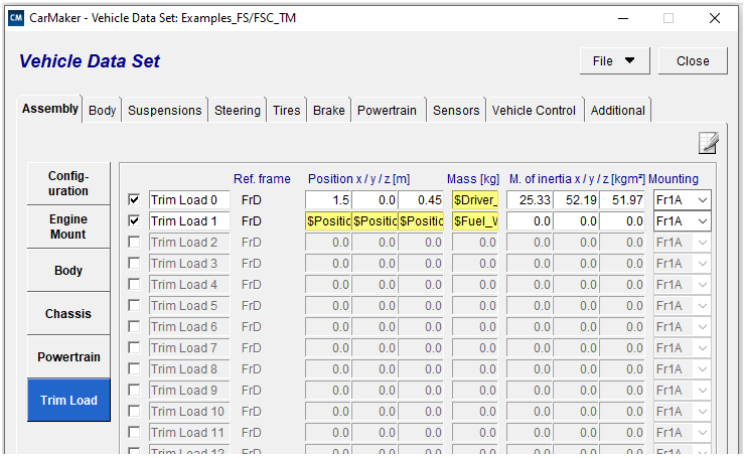



Figure 6.14: Setting a name value in the Vehicle Data Set dialog

Note: While defining NValues in the CarMaker GUI, it is important the user still provides a default value to the parameter to avoid errors when simulating TestRuns without the TestManager.

In the following example we will show how to prepare an automated TestRun using the Acceleration event with the driver weight as a variable. After having generated the *Test-Series*, you can start a simulation. If you want to save the TestSeries' configuration, click *File* > *Save As*. To compare the results, open IPGControl and load the different files as demonstrated in Exercise 5 on page 46. Feasible values could be "Car.Pitch" or "Car.ax".

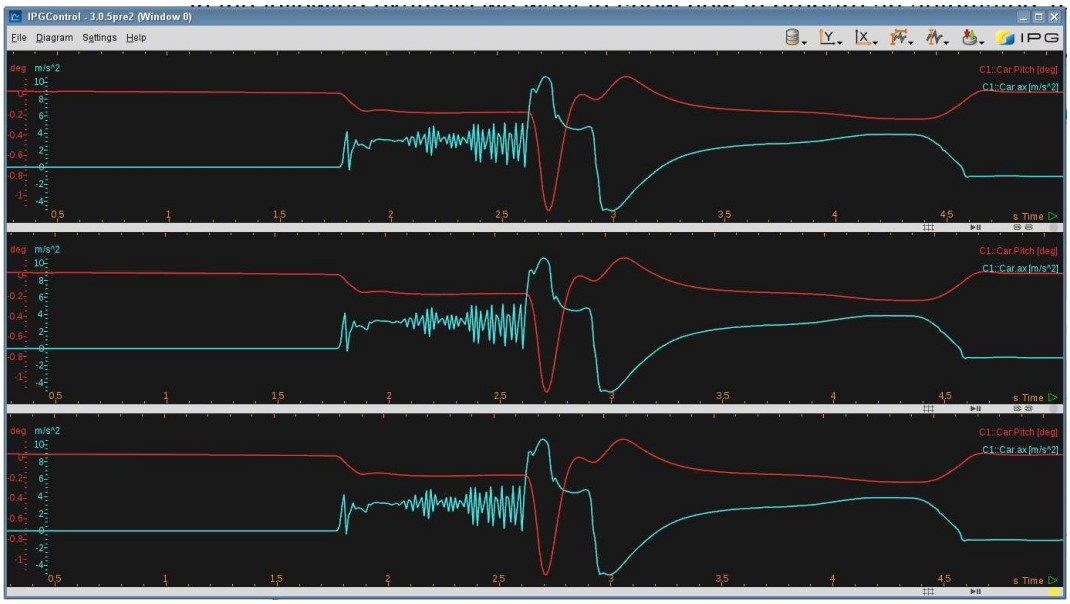



Figure 6.15: Comparison of the saved results

### Exercise 15 - Step 1

6.4.2.1 Please save your current work.

6.4.2.2 Add a TestRun to TestManager:

6.4.2.2.1 Open "Competition/FS_Acceleration" in the CarMaker Main GUI.

6.4.2.2.2 Change the car to “FSC_TestAutomation”.

6.4.2.2.3 Set the “Trim Load 0” in *Assembly* > *Trim Load* of the Vehicle Data Set to "$Driver_Wt=70" (notice that "=70" sets the variable to the default value 70). Select "save all" for the storage ofresults.

6.4.2.2.4 Save the new file as "Acceleration_TestManager".

6.4.2.2.5 Now start the TestManager (*Simulation* > *TestManager)* and add the previous creat- ed TestRun "Acceleration_TestManager". Select (Add > TestRun), then load your previously created TestRun in the field *TestRun Description*.

6.4.2.2.6 In *TestRun Description*: Type "NValue", Name "Driver_Wt"

6.4.2.3 Add variations:

6.4.2.3.1 Select (*Add* > *Variation*). Add 3 variations.

6.4.2.3.2 Type "NValue", Name "Driver_Wt" and a value for each variation. Use the values 70, 85 and 100.

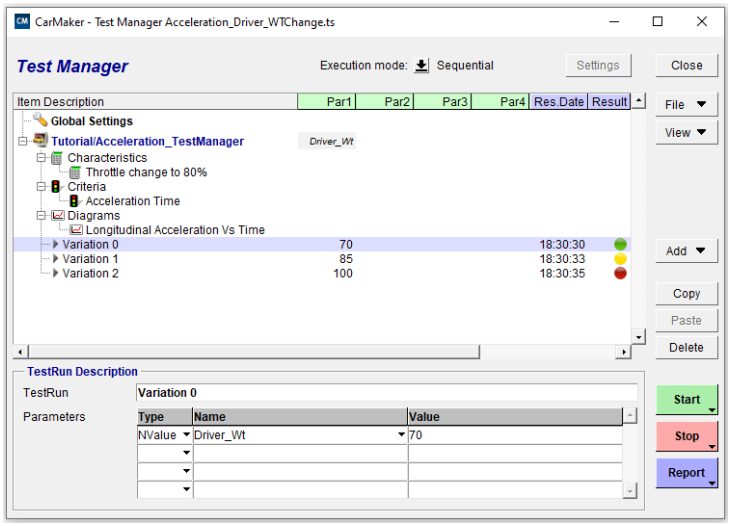



Figure 6.16: The declaration of variations

## 6.4.3 Criteria and Analysis

In addition to the execution of TestRun loops for parameter variation, the TestManager also provides an interface to evaluate the simulations directly with user defined criteria. These are basically used to describe the vehicle performance with respect to specific indicators. You can implement this via *Add* > *Criterion* for the selected TestRun (Figure 6.15). It is possible to add multiple criteria for the same TestRun.

Criteria are defined by using the syntax "[get Parameter]" followed by a condition. For example:

$$[\text{get Car.Yaw}] <= 6*\text{pi}/180 \qquad \textbf{(EQ 31)}$$

This criterion is considered to be "good" when the yaw angle is less than 6 degrees.

Note: The UAQs always use their default units but the TestManager uses the SI system. It is therefore important to convert the units to SI when required. If you are not sure about the default unit of a quantity, please have look at the Reference Manual section 23.4 '*User Accessible Quantities'* (*Main GUI > Help > Reference Manual*).

The parameter called within the criteria expression always returns the value which is taken at the end of simulation. Many parameters return to 0 at the end of the simulation and therefore can't be called by the criteria function. In such cases, new variable creation is advisable to take the value of such parameters as they will not return back to 0.

You can create a variable in the *Maneuver* tab of the TestRun using Mini-maneuver Commands in the CarMaker GUI or by defining *Characteristics* (*Add* > *Characteristic*, see Figure 6.17) for the respective TestRun in the TestManager.

Please note that the commands in the *Characteristics* tab of the TestManager override the Real time Expressions mentioned in the CarMaker GUI.

With the help of specified criteria, the TestManager evaluates whether a test was successful or not and provides the User with a visualization which comprises of 6 different signals, each with their own meaning. For eg. Green for satisfying the defined criteria or Red for indicating that the criteria has not been met. For more information please refer the User's Guide (*Main GUI* > *Help* > *User's Guide*), Chapter 14 '*Test Automation'*.

In addition to configuring the Criteria, the TestManager can produce diagrams resulting from a test series. These diagrams are plotted after the simulation has finished. Three different diagram modes are supported:

6.4.3.1 Quantity(s) vs. Time

6.4.3.2 Quantity(s) vs. Quantity

6.4.3.3 Characteristic Value vs. Variation

It is possible to plot one diagram for each variation or to summarize all variations in a single diagram for a better overview. You can create diagrams for a TestRun by choosing *Add* > *Diagram* (Figure 6.16). Here, the user will have the ability to name respective axis labels and parameters. To produce the correct diagram, it is necessary to insert a UAQ using the correct syntax.

The *TestManager* also includes report functionality which automatically generates an overview of the performed Test Series. This report can be called after a test series is either completed or aborted by pressing the *Report* button in the lower right corner of the TestManager window. It provides detailed results of every performed TestRun which are automatically prepared and arranged. The report starts with an overview of all the TestRuns and their respective results. It also provides details that consist of diagrams, variations and TestRun maneuvers (Figure 6.18). The Report can be found via "<your-ProjectDirectory>/SimOutput/>HostName>/Report".

For more information, refer to the User's Guide (*Main GUI > Help > User's Guide*), Chapter 14: '*Test Automation'.*

In the following exercises, we will define Criteria, Diagrams and Characteristics in the same TestRun using the *TestManager*, followed by a Report generation.

## Exercise 15 - Step 2

6.4.3.4 Create Criteria:

6.4.3.4.1 Select the " Acceleration_TestManager" TestRun in TestManager and *Add* > *Criterion*.

6.4.3.4.2 Set the *Title* as "Acceleration Time". Go to the *Evaluation* tab and set the Criteria for green, yellow and red as shown in figure below.

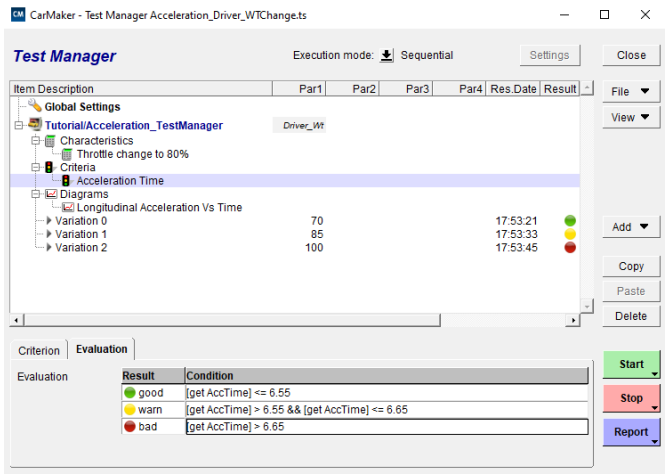



Figure 6.15: Defining Criteria

6.4.3.5 Create a Diagram:

6.4.3.5.1 *Add* > *Diagram*. Give diagram a title: "Longitudinal Acceleration Vs Time". Tick the checkbox: "*Show all variations in single graph"*.

6.4.3.5.2 Provide axis labels "Time" and "LongAcc". Select the User Accessible Quantity: "Car.ax" for the Y-Axis.

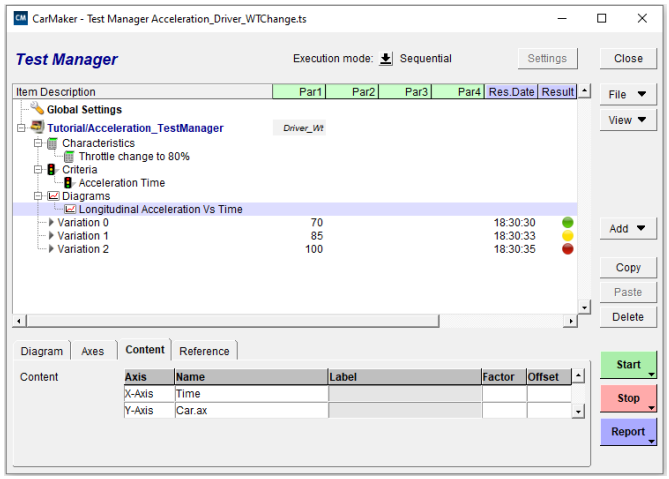



Figure 6.16: Creating Diagrams

6.4.3.6 Define Characteristics:

6.4.3.6.1 Using *Add* > *Characteristic*, set the title as “Throttle change to 80%”

6.4.3.6.2 Add the condition for 80% gas after 4 seconds as shown in Figure 6.17.

6.4.3.7 After assigning *Criteria*, *Diagrams* and *Characteristics*, simulate the TestRun from Test- Manager and generate the Report using the TestManager GUI.

6.4.3.8 Save the Test Series as "Acceleration_Driver_WTChange.ts"

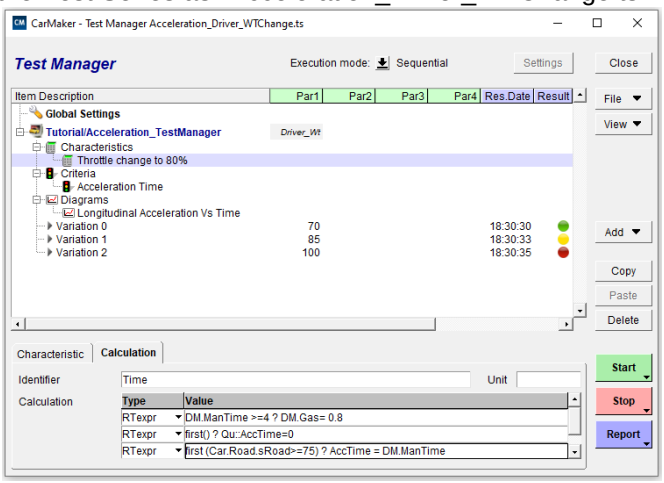



Figure 6.17: Defining Characteristics

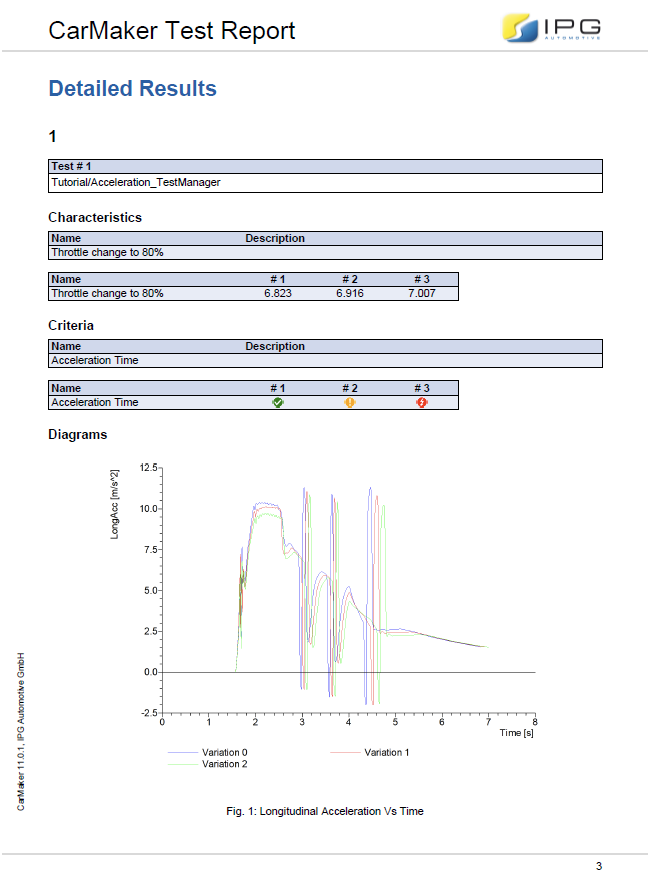



Figure 6.18: Sample Report page

## 6.4.4 Example TestSeries

To help Formula Student teams test their vehicle models, the IPG Team have created a special TestSeries named "FS_Automation.ts". This TestSeries helps students understand their car better by completing various static and dynamic TestRuns. Their car can then be optimized by analyzing the results of this TestSeries. It consists of various TestRuns that simulate FS cars by replicating a racing environment.

This TestSeries consists of 3 groups of events and their corresponding TestRuns. These groups are Equilibrium, Longitudinal Dynamics and Lateral Dynamics.

### Equilibrium

#### 1. Loading

**Objective:** To verify the static equilibrium of the vehicle with different loading conditions. This TestRun is a good way for students to start analyzing their vehicle behavior before going into dynamic trails. For analysis, students can optimize the "Toe Angle", "Camber Angle" and "Loading" on each wheel of the car.

Table 6.3: Standstill variables

| Type | Variable Name | Variable Description | Parameter Location |
|---|---|---|---|
| NV | Driver_Wt | Driver Weight | Parameters > Car > Bodies > Trim Load 1 > Mass |
| NV | Fuel_Wt | Fuel Weight | Parameters > Car > Bodies > Trim Load 2 > Mass |
| NV | Position_X | X Location of Fuel Weight | Parameters > Car > Bodies > Trim Load 2 > x |
| NV | Position_Y | Y Location of Fuel Weight | Parameters > Car > Bodies > Trim Load 2 > y |
| NV | Position_Z | Z Location of Fuel Weight | Parameters > Car > Bodies > Trim Load 2 > z |

Table 6.4: Quantities to be evaluated in Standstill Test

| Validation Output | User Accessible Quantities |
|---|---|
| Toe Angle | Car.ToeFL, Car.ToeRR |
| Camber Angle | Car.CamberFL, Car.CamberRR |
| Vertical Loads on all 4 Wheels | Car.FzFL, Car.FzRL, Car.FzFR, Car.FzRR |

## Longitudinal Dynamics

### 1. Pull Test

**Objective** To check the straight-line stability of the vehicle with no steering input. This TestRun is important for driver safety and vehicle performance. Desirable results for this test depend on CoG location, suspension, steering kinematics and overall symmetry. For analysis, students should look at the lateral deviation of the car and each wheel's toe and camber angles.

Table 6.5: Variables for Pull Test

| Type | Variable Name | Variable Description | Parameter Location |
|---|---|---|---|
| NV | Speed | IPG Driver Speed | Parameters > Maneuver > Longitudinal Dynamics > IPG Driver > Speed |
| NV | CoG_x | X Location of CoG | Parameters > Car > Vehicle Body > Rigid Vehicle Body > Vehicle Body > x |
| NV | CoG_y | Y Location of CoG | Parameters > Car > Vehicle Body > Rigid Vehicle Body > Vehicle Body > y |
| NV | CoG_z | Z Location of CoG | Parameters > Car > Vehicle Body > Rigid Vehicle Body > Vehicle Body > z |

Table 6.6: Quantities to be evaluated in Pull Test

| Validation Output | User Accessible Quantities |
|---|---|
| Lateral Deviation | Driver.Lat.dy |
| Toe Angle | Car.ToeFL, Car.ToeRR |
| Camber Angle | Car.CamberFL, Car.CamberRR |

### 2. Acceleration

**Objective** This TestRun evaluates the vehicle's ability to reach 75m in the shortest possible time when starting from standstill. To maximize the acceleration, the gear shifting RPM and gear ratios can be changed. For analysis, students should analyze the max. vehicle speed, max. longitudinal acceleration, max. engine RPM and pitch angle of the car.

Table 6.7: Variables for Acceleration Test

| Type | Variable Name | Variable Description | Parameter Location |
|---|---|---|---|
| NV | Gas | Throttle Value (0-1) | Parameters > Maneuver > Maneuver 0 > Minimaneuver Commands |
| NV | Shiftup | Shifting RPM | Parameters > Driver > User Parameterized Driver > Standard Parameters > Declutching / Gear Shifting > Engine Speeds [RPM] > max |
| KV | PowerTrain.GearBox.iForwardGears | Gear Ratio change for all gears | Path to File /Data/Vehicle/Examples_FS/FSC_TestAutomation |

Table 6.8: Quantities to be evaluated in Acceleration Test

| Validation Output | User Accessible Quantities |
|---|---|
| Vertical Loads for all 4 Wheels | Car.FzFL, Car.FzRL, Car.FzFR, Car.FzRR |
| Toe Angle | Car.ToeFL, Car.ToeRR |

Table 6.8: Quantities to be evaluated in Acceleration Test

| Validation Output | User Accessible Quantities |
|---|---|
| Camber Angle | Car.CamberFL, Car.CamberRR |
| Pitch Velocity | Car.PitchVel |
| Max Vehicle Speed | Car.v |
| Max Vehicle Acceleration | Car.ax |
| Max Engine RPM | PT.Engine.rotv |
| Max Engine Power | PT.Engine.PwrO |
| Pitch | Car.Pitch |
| Time | Time |

### 3. TopSpeed

**Objective**

This TestRun checks the vehicle's ability to reach its top speed in the shortest possible time when starting from standstill. To maximize the acceleration, the gear shifting RPM and gear ratios can be changed. For analysis, students should analyze the max. vehicle speed, max. longitudinal acceleration, max. engine RPM and pitch angle of the car.

Table 6.9: Variables for TopSpeed Test

| Type | Variable Name | Variable Description | Parameter Location |
|---|---|---|---|
| NV | Gas | Throttle Value (0-1) | Parameters > Maneuver > Maneuver 0 > Minimaneuver Commands |
| NV | Shiftup | Shifting RPM | Parameters > Driver > User Parameterized Driver > Standard Parameters > Declutching / Gear Shifting > Engine Speeds [RPM] > max |
| KV | PowerTrain.GearBox.iForwardGears | Gear Ratio change for all gears | Path to File /Data/Vehicle/Examples_FS/FSC_TestAutomation |

Table 6.10: Quantities to be evaluated in TopSpeed Test

| Validation Output | User Accessible Quantities |
|---|---|
| Vertical Loads on all 4 Wheels | Car.FzFL, Car.FzRL, Car.FzFR, Car.FzRR |
| Toe Angle | Car.ToeFL, Car.ToeRR |
| Camber Angle | Car.CamberFL, Car.CamberRR |
| Pitch Velocity | Car.PitchVel |
| Max Vehicle Speed | Car.v |
| Max Vehicle Acceleration | Car.ax |
| Max Engine RPM | PT.Engine.rotv |
| Max Engine Power | PT.Engine.PwrO |
| Pitch | Car.Pitch |
| Time | Time |

### 4. Braking_FS

**Objective** This TestRuns aim to check the competency of the braking system to lock all four wheels and stop the vehicle in a straight line. The vehicle accelerates from a standstill and travels a distance of 100 m, at which point the brakes are applied. Desirable results for this test depend on an appropriate brake pedal ratio, CoG location, suspension and steering kinematics. For analysis, students can evaluate the braking distance, car velocity and yaw angle.

Table 6.11: Variables for Braking_FS Tests

| Type | Variable Name | Variable Description | Parameter Location |
|---|---|---|---|
| NV | CoG_x | X Location of CoG | Parameters > Car > Vehicle Body > Rigid Vehicle Body > Vehicle Body > x |
| NV | CoG_y | Y Location of CoG | Parameters > Car > Vehicle Body > Rigid Vehicle Body > Vehicle Body > y |
| NV | CoG_z | Z Location of CoG | Parameters > Car > Vehicle Body > Rigid Vehicle Body > Vehicle Body > z |
| KV | Pedal.ratio | Brake Pedal Ratio | Path to File /Data/Misc/ HydESP_FS_RaceCar_4.0 |

Table 6.12: Quantities to be evaluated in Braking_FS Tests

| Validation Output | User Accessible Quantities |
|---|---|
| Vertical Loads on all 4 Wheels | Car.FzFL, Car.FzRL, Car.FzFR, Car.FzRR |
| Toe Angle | Car.ToeFL, Car.ToeRR |
| Camber Angle | Car.CamberFL, Car.CamberRR |
| Pitch Velocity | Car.PitchVel |
| Max Vehicle Deceleration | Car.ax |
| Pitch | Car.Pitch |
| Pedal force | Brake.Hyd.Sys.PedFrc |
| Pressure at master cylinder | Brake.Hyd.Sys.pMC |
| Pressure at each wheel | Brake.Hyd.Sys.pWB_FL, Brake.Hyd.Sys.pWB_FR, Brake.Hyd.Sys.pWB_RL, Brake.Hyd.Sys.pWB_RR |
| Yaw Angle | Car.Yaw |
| Max Vehicle Speed | Car.v |
| Braking Distance | To be calculated using Real Time Expressions |
| Braking Time | To be calculated using Real Time Expressions |

### 5. Braking

**Objective**

This TestRun aims to check the capability of brake system to lock all four wheels and stop the vehicle in a straight line. Vehicle accelerates from standstill to 100 km/h followed full application of the brakes Desirable results for this test depend on an appropriate brake pedal ratio, CoG location, suspension and steering kinematics. For analysis, students can evaluate the braking distance, car velocity and yaw angle.

Table 6.13: Variables for Braking Test

| Type | Variable Name | Variable Description | Parameter Location |
|---|---|---|---|
| NV | CoG_x | X Location of CoG | Parameters > Car > Vehicle Body > Rigid Vehicle Body > Vehicle Body > x |
| NV | CoG_y | Y Location of CoG | Parameters > Car > Vehicle Body > Rigid Vehicle Body > Vehicle Body > y |
| NV | CoG_z | Z Location of CoG | Parameters > Car > Vehicle Body > Rigid Vehicle Body > Vehicle Body > z |
| NV | Brake | Brake Value (0-1) | Parameters > Maneuver > Maneuver 1 > Longitudinal Dynamics > Manual (Pedals Gears) > Brake > Value |

Table 6.14: Quantities to be evaluated in Braking Test

| Validation Output | User Accessible Quantities |
|---|---|
| Vertical Loads on all 4 Wheels | Car.FzFL, Car.FzRL, Car.FzFR, Car.FzRR |
| Toe Angle | Car.ToeFL, Car.ToeRR |
| Camber Angle | Car.CamberFL, Car.CamberRR |
| Pitch Velocity | Car.PitchVel |
| Max Vehicle Deceleration | Car.ax |
| Pitch | Car.Pitch |
| Pedal force | Brake.Hyd.Sys.PedFrc |
| Pressure at master cylinder | Brake.Hyd.Sys.pMC |
| Pressure at all 4 wheels | Brake.Hyd.Sys.pWB_FL, Brake.Hyd.Sys.pWB_FR, Brake.Hyd.Sys.pWB_RL, Brake.Hyd.Sys.pWB_RR |
| Yaw Angle | Car.Yaw |
| Max Vehicle Speed | Car.v |
| Braking Distance | To be calculated using Real Time Expressions |
| Braking Time | To be calculated using Real Time Expressions |

## Lateral Dynamics

### 1. Zigzag

**Objective** This TestRun aims to evaluate the vehicle's response to steering input by following alternating left and right turns, spaced at distances of 15 m and 18 m. To increase the cornering capability, students should adjust suspension and steering parameters in order to optimize body roll. For analysis, students should assess body roll, lateral acceleration and steering wheel angle of the car.

Table 6.15: Variables for Zigzag Test

| Type | Variable Name | Variable Description | Parameter Location |
|---|---|---|---|
| NV | CSpeed | Cruising Speed | Parameters > Driver > User Parameterized Driver > Standard Parameters > General > Cruising Speed |

Table 6.16: Quantities to be evaluated in Zigzag Test

| Validation Output | User Accessible Quantities |
|---|---|
| Lateral Acceleration | Car.ay |
| Max Roll angle | Car.Roll |
| Roll angle gradient | Car.RollAcc |
| Steering wheel angle | DM.Steer.Ang |
| Understeer gradient | To be calculated using Real Time Expressions |
| Slip Angle (all wheels) | Car.SlipAngleFL, Car.SlipAngleFR, Car.SlipAngleRL, Car.SlipAngleRR, |
| Steering Turning Torque | DM.Steer.Trq |
| Fz (Vertical forces on all tyres) | Car.FzFL, Car.FzRL, Car.FzFR, Car.FzRR |
| Yaw | Car.Yaw |
| Yaw rate | Car.YawRate |
| Time | Time |

### 2. Steer Step

**Objective** This TestRun aims to assess the vehicle's response to sudden steering input at varying speeds and steering angles. To increase cornering capacity, students should alter suspension and steering parameters as to minimize body roll. For analysis, students should look at yaw rate, lateral acceleration and steering wheel angle.

Table 6.17: Variables for Steer Step Test

| Type | Variable Name | Variable Description | Parameter Location |
|---|---|---|---|
| NV | SteerAng | IPG Driver Steer Step Angle | Parameters > Maneuver > Maneuver 1,2 > Lateral Dynamics > Steer Step > Amplitude |
| NV | SpeedCtrl | Control Speed Long. Dynamics | Parameters > Maneuver > Maneuver 1 > Longitudinal Dynamics > Speed Control > Speed |

Table 6.18: Quantities to be evaluated in Steer Step Test

| Validation Output | User Accessible Quantities |
|---|---|
| Lateral Acceleration | Car.ay |
| Max Roll angle | Car.Roll |
| Roll angle gradient | Car.RollAcc |
| Steering wheel angle | DM.Steer.Ang |
| Understeer gradient | To be calculated using Real Time Expressions |
| Slip Angle (all wheels) | Car.SlipAngleFL, Car.SlipAngleFR, Car.SlipAngleRL, Car.SlipAngleRR, |
| Steering Turning Torque | DM.Steer.Trq |
| Fz (Vertical forces on all tyres) | Car.FzFL, Car.FzRL, Car.FzFR, Car.FzRR |
| Yaw | Car.Yaw |
| Yaw rate | Car.YawRate |
| Time | Time |

### 3. Steady State Circle

**Objective**

This TestRun aims to assess the lateral stability limit of the vehicle by following a 'steady-state' circle of 42 m radius at varying speeds and suspension parameters. To increase the cornering performance, students should adjust suspension and steering parameters to limit body roll. For analysis, students should look at lateral deviation, lateral acceleration and steering wheel angle.

Table 6.19: Variables for Steady State Circle Test

| Type | Variable Name | Variable Description | Parameter Location |
|---|---|---|---|
| NV | CSpeed | Cruising Speed | Parameters > Driver > User Parameterized Driver > Standard Parameters > General > Cruising Speed |
| NV | StabAmp | Stabilizer Stiffness Amplification | Parameters > Car > Suspension > Stabilizer > Front, Rear > Amplification |

Table 6.20: Quantities to be evaluated in Steady State Circle Test

| Validation Output | User Accessible Quantities |
|---|---|
| Lateral Acceleration | Car.ay |
| Max Roll angle | Car.Roll |
| Roll angle gradient | Car.RollAcc |
| Steering wheel angle | DM.Steer.Ang |
| Understeer gradient | To be calculated using Real Time Expressions |
| Slip Angle for all 4 wheels | Car.SlipAngleFL, Car.SlipAngleFR, Car.SlipAngleRL, Car.SlipAngleRR, |
| Steering Turning Torque | DM.Steer.Trq |
| Fz (Vertical forces on all tyres) | Car.FzFL, Car.FzRL, Car.FzFR, Car.FzRR |
| Yaw | Car.Yaw |
| Yaw rate | Car.YawRate |

Table 6.20: Quantities to be evaluated in Steady State Circle Test

| Validation Output | User Accessible Quantities |
|---|---|
| Lateral Deviation | Driver.Lat.dy |
| Time | Time |

For most of the TestRuns, relevant acceptance criteria were defined and their descriptions are expressed in the table below:

Table 6.21: List of Criteria and meanings

| TestRun Name | Criteria | Description |
|---|---|---|
| Pull Test | G - [get LatShift] < abs(0.35) | Lateral Deviation of Car on either side from center line is less than 0.35m |
| | Y - [get LatShift] >= abs(0.35) && [get LatShift] < abs(0.7) | Lateral Deviation of Car on either side from center line is between 0.35m and 0.7m |
| | R - [get LatShift] >= abs(0.7) | Lateral Deviation of Car on either side from center line is more than 0.7m |
| Acceleration | G - [get DM.ManTime] < 5 | Acceleration Maneuver Time is less than 5sec |
| | Y - [get DM.ManTime] >= 5 && [get DM.ManTime] <= 6 | Acceleration Maneuver Time is in between 5sec and 6 sec |
| | R - [get DM.ManTime] > 6 | Acceleration Maneuver Time is more than 6sec |
| TopSpeed | G - [get Time] < 16 | Time to reach Top Speed is less than 16sec |
| | Y - [get Time] >= 16 && [get Time] <= 20 | Time to reach Top Speed is in between 16sec and 20 sec |
| | R - [get Time] > 20 | Time to reach Top Speed is more than 20sec |
| TopSpeed | G - [get TopSpeed] > 44 | Top Speed is more than 44m/s |
| | Y - [get TopSpeed] >=41 && [get TopSpeed] <=44 | Top Speed is in between 41m/s and 44m/s |
| | R - [get TopSpeed] < 41 | Top Speed is less than 41m/s |
| Braking_FS & Braking | G - [get Car.Yaw] < 0.087 \|\| [get Car.Yaw] > -0.087 | Car Yaw angle is less than 0.087 rad (5 deg) in both the direction |
| | Y - [get Car.Yaw] >= 0.087 && [get Car.Yaw] <= 0.174 \|\| [get Car.Yaw] <= -0.087 && [get Car.Yaw] >= -0.174 | Car Yaw angle is between 0.087 rad (5 deg) and 0.174 rad (10 deg) in both the direction |
| | R - [get Car.Yaw] > 0.174 \|\| [get Car.Yaw] < -0.174 | Car Yaw angle is more than 0.174 rad (10 deg) in both the direction |

Table 6.21: List of Criteria and meanings

| TestRun Name | Criteria | Description |
|---|---|---|
| ZigZag | G - [get Rollmax] < 0.0104 && [get Rollmin] > -0.0104 | Car Roll Angle is less than 0.0104 rad (0.6 deg) on either side |
| | Y - [get Rollmax] >= 0.0104 && [get Rollmax] <= 0.0122 \|\| [get Rollmin] <= -0.0122 && [get Rollmin] >= -0.0104 | Car Roll Angle is between 0.0104 rad (0.6 deg) and 0.0122 rad (0.7 deg) on either side |
| | R - [get Rollmax] > 0.0122 && [get Rollmin] < -0.0122 | Car Roll Angle is more than 0.0122 rad (0.7 deg) on either side |
| Steady State Circle | G - [get LatDyMax] <=0.35 && [get LatDyMin] >=-0.35 | Lateral Deviation of Car on either side from centre line is less than 0.35m |
| | Y - [get LatDyMax] >0.35 && [get LatDyMax] <=0.7 \|\| [get LatDyMin] <-0.35 && [get LatDyMin] >=-0.7 | Lateral Deviation of Car on either side from centre line is between 0.35m and 0.7m |
| | R - [get LatDyMax] >0.7 \|\| [get LatDyMin] <-0.7 | Lateral Deviation of Car on either side from centre line is more than 0.7m |

## Exercise 16

6.4.4.1 Use your own car:

6.4.4.1.1 After running these TestRuns in the TestManager with the "FS_TestAutomation", you can now import your own car to perform these TestRuns.

6.4.4.1.2 Select "Global settings" in the TestManager and *Add* > *Vehicle*.

6.4.4.1.3 In the *Configuration* tab, select your own car from your project folder. After importing, save the TestSeries as "MyTestSeries".

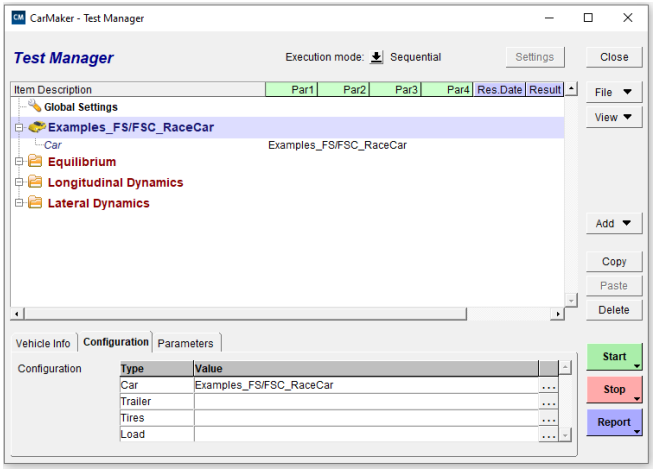



Figure 6.19: Importing the student car model

Chapter 7

# Helpful Suggestions

## Recommended Literature

If you are developing a Formula Student car for the first time, you may require some more detailed background information. Below is a list of recommended literature which we think is very valuable:

1. Gillespie, Thomas:
   Fundamentals of Vehicle Dynamics, SAE International, 1992
2. Heißing, Bernd; Ersoy, Metin:
   Fahrwerkhandbuch, Vierweg-Verlag, 2007
3. Matschinsky, Wolfgang:
   Die Radaufhängungen der Straßenfahrzeuge, Springer Verlag, 1998
4. Mitschke, Manfred:
   Dynamik der Kraftfahrzeuge, Band A-C, Springer Verlag, 1982
5. Milliken, William F.:
   Race Car Vehicle Dynamics, SAE International, 1995
6. Reimpell, Jörnsen; Betzler, Jürgen:
   Fahrwerktechnik: Grundlagen, Vogel-Verlag, 2000

IPG Automotive GmbH greatly admires how the committed members of each Formula Student team tackle such a huge project. It is from this admiration that the idea to support these projects first arose. This is done through the supply of software licenses as well as technical support. In the following paragraphs, some advice in relation to these services is given.

## Email Contact

Email support is available for every team at no additional cost. Please address every question to **FormulaCarMaker@ipg-automotive.com**. This address forwards the incoming emails to every member of the Formula CarMaker Support Team immediately. Through doing this, each email will reach the most relevant person to deal with your query.

IPG Automotive supports over 200 teams worldwide. With two registered members per team, this equates to over 400 students to help. Therefore, it is extremely difficult to remember which student belongs to each team!

Many Formula Student teams do not have an associated email address, leading to personal email accounts being used to address queries. The FCM Support Team is therefore very grateful for every inquiry that includes the name of your team and university.

**You make our life easier, when...**

- **You add the license number to the email inquiry.**
- **You add your name to the end of your email.**
- **You add both the university's and team's name.**

In addition to the contact email, the Formula CarMaker support features various tutorials and handbooks. It is recommended to browse through these documents as they provide many solutions to frequently asked questions. Furthermore, they are readily available and might solve your problem earlier than awaiting an email response.

## Further Development of the Programs

To enhance the development of CarMaker, IPGKinematics and other IPG products, IPG Automotive GmbH is grateful for any information regarding errors and potential improvements. The Formula CarMaker Support Team truly appreciates suggestions of any form!

## Remark to the Used Versions and Data Sets

This document refers to the following programs and versions: CarMaker 12, IPGKinematics 3.6.11 and Matlab R2020a.
All used datasets such as the TestRun "FS_CM6_BrakeSlalom" ran without problems in these versions and can be downloaded from the member area (FS_Generic_2022) on the IPG website.

Appendix A

# Bibliography

## Literature


[CMR07] IPG Automotive GmbH: CarMaker Reference Manual. IPG Automotive GmbH, 2014

[CMU07] IPG Automotive GmbH: CarMaker User's Guide. IPG Automotive GmbH, 2014

[HE07] Heißing, E.; Estoy, M.: Fahrwerkhandbuch. Wiesbaden: Vierweg-Verlag, 2007

[KIN07] IPG Automotive GmbH: IPGKinematics Handbuch. IPG Automotive GmbH, 2014

[RB00] Reimpell, J.; Betzler, J..: Fahrwerktechnik: Grundlagen. Würzburg: Vogel-Verlag, 2000

[Rei82] Reimpell, J.: Fahrwerktechnik 1. Würzburg: Vogel-Verlag, 1982

[Rei84] Reimpell, J.: Fahrwerktechnik: Lenkung. Würzburg: Vogel-Verlag, 1984

[RH84] Rompe, K.; Heißing, B.: Objektive Testfahrten für die Fahreigenschaften von Kraftfahrzeugen. Köln: Verlag TÜV Rheinland GmbH, 1984

[SAE07] Society of Automotive Engineers: Formula SAE Rules. Society of Automotive Engineers, 2007

[TIR07] IPG Automotive GmbH: IPGTire Tutorial. IPG Automotive GmbH, 2007


## Websites


[1] http://www.formulastudent.de

[2] http://www.millikenresearch.com/fsaettc.html

[3] www.optimumg.com

Appendix B

# Forces On Kinematics Simulation

If modeling the effects of compliance and calculating kinematics with the presence of forces is desired, the guide below should be used in conjunction with Chapter 4 '*IPGKinematics*' in order to configure a 'Forces On' simulation. Included are equations to help with the parameterization of stabilizers and associated stiffnesses.

Assuming the user has already completed Chapter 4, the simulation settings can be reconfigured to 'Forces On' by using the previously provided inputs with additional values given in this section to generate an example .kin file. Below details which parameters should be changed from Chapter 4.

'Forces On' simulations involve multiple additional calculation steps compared to 'Forces Off', therefore increasing computation time and also require the accurate parameterization of mass and stiffness properties of many components. For this reason, it is only recommended to proceed if such data is accurate and readily available, and external forces form a crucial part of your project.

## B.1 Simulation Control

### General

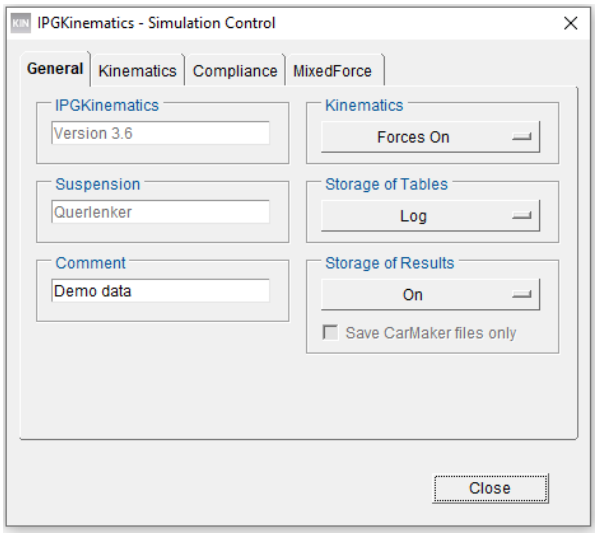



Figure B.1 - General tab

**Kinematics** *Forces On* should be selected from the *Kinematics* drop-down menu, meaning that inertial and spring forces are now considered in the kinematics calculations.

### Kinematics

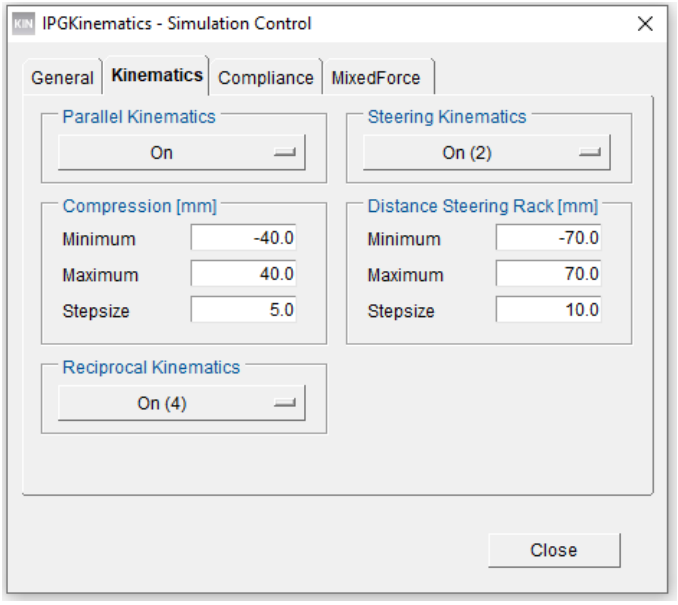



Figure B.2 - Kinematics tab

**Steering Kinematics**

This is used only for the front axle to calculate steering parameters. It simulates rack-and-pinion steering which moves the front wheels accordingly.

Select *On (2),* as *On (1)* does not consider the interaction between steering and reciprocal wheel travel. "Steering Forces" is used in certain situations such as a tire hitting a pavement.

**Reciprocal Kinematics**

This option will move the axle's wheels in opposite directions. To understand the processes of the different options, please take a look at the Reference Manual.

Select *On (2)* as this will simulate reciprocal jouncing and rebounding.

All other parameters within the Simulation Control GUI can be left as per Chapter 4.

## B.2 Vehicle Data

### General

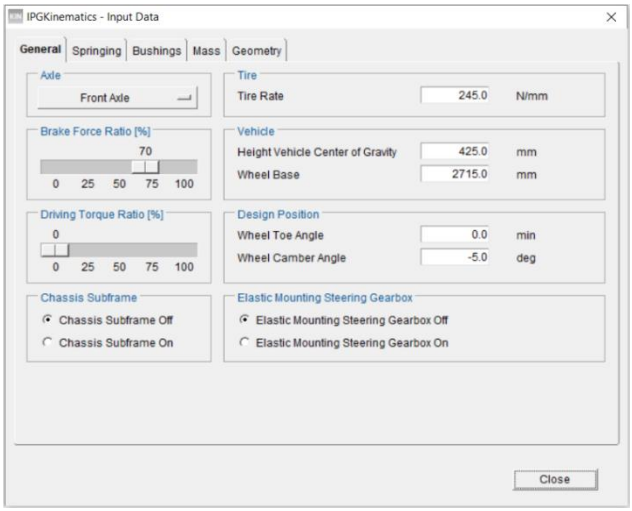



Figure B.3 - General tab

**Brake Force Ratio**

In this field, the brake force ratio of the corresponding axle is defined. 0% means that the axle being modeling receives no braking power. This information is required for calculations of the braking torque compensation angle and anti-dive properties.

**Driving Torque Ratio**

This defines the driving torque ratio of the axle. 0% means that the axle is not powered and 100% means that only this axle is powered. This value is necessary to calculate anti-squat and starting torque compensation angle values.

**Tire**

This field is used to define the vertical stiffness of the tire (in N/mm). As the tire is modeled as a linear spring, an accurate tire spring coefficient must also be entered.

The spring coefficient can be determined by a simple static test. All that is required is a spring scale and two flat plates. The tire is clamped between the two plates and placed on the spring balance. A known downward force, $F_R$ , is the applied. By measuring the tire deflection $\Delta s$ the spring coefficient can be determined as follows:

$$c_1 = \frac{F_R}{\Delta_s} \quad \textbf{(EQ 32)}$$

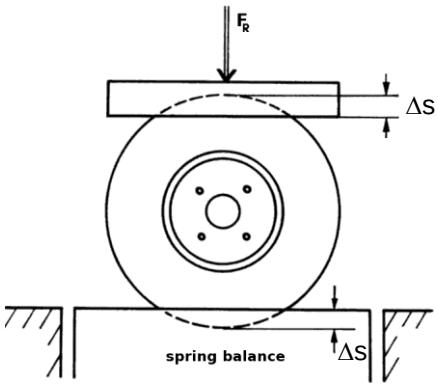



Figure B.4 - Measuring the tire deflection [Rei82]

If you are a member of the Tire Test Consortium you can access detailed spring rate data for your tires. For this information, view the “FromCalspan.zip” archive from the *RoundX* tire testing directory on the official TTC forum. The archive contains a spreadsheet titled “SummaryTables.xls”. Within it you will find measured cornering stiffness data as well as vertical stiffness as a function of the inclination angle, tire pressure and vertical load.

If you don’t have access to reasonable values, there is another method to gaining an approximation. For this you need the tire’s diameter D, static radius $r_{stat}$, load in accordance with the actual tire pressure $F_R$ and a correction factor $k_B$ for the tire design, tread and velocity. If these values are available, a good approximation for the spring coefficient can be found by using:

$$c_1 = k_B \frac{F_R}{\frac{D}{2} - r_{stat}} \quad \textbf{(EQ 33)}$$

## Springing

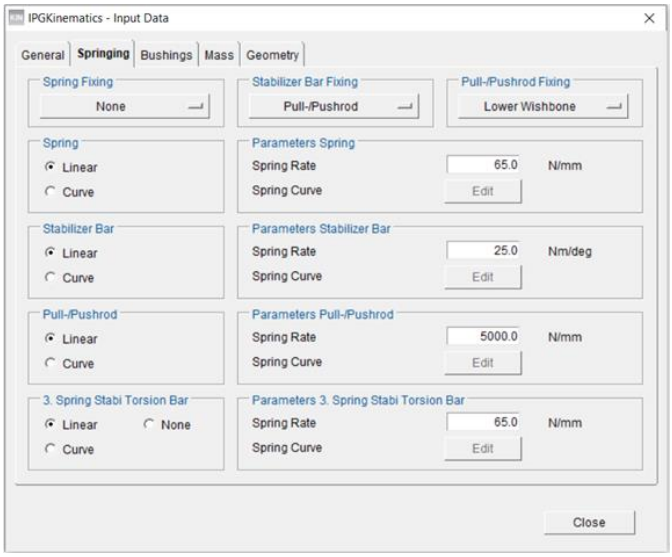



Figure B.5 - Springing tab

**Stabilizer Fixing**

In IPGKinematics you have the option to mount the stabilizer to either the lower wish- bone, wheel carrier or the pull-/pushrod. If you have anti-rollbars which are attached to the rocker, you should select the option *pull-/pushrod*. By choosing this, IPGKinematics will consider a 'T-Bar Stabilizer', otherwise a 'U-Bar' is used.

To properly parameterize the stabilizer in IPGKinematics you should select *pull-/pushrod*. if you do not have a T-Stabilizer in your car, calculate the equivalent stiffness of the stabilizer that matches the configuration.

Another option is to convert the stabilizer into a U-bar. You should define the wheel carrier as the stabilizer fixing due to the mechanism having a motion ratio of 1. This method should work especially well if you are using a constant motion ratio in reality as well.

There are a variety of possibilities to modelling a stabilizer mechanism, here are a few suggestions:

1. Choose the wheel center as connection point for the stabilizer link to the wheel.

2. The connection point of the stabilizer link and U-bar should be placed 100mm vertically from the former point.

3. The lever arm of the U-bar should be located 200mm horizontally in the x-direction from the stabilizer mounting point on the chassis.

This stabilizer parametrization will work correctly in IPGKinematics as long as you use an equivalent stiffness which allows the substituted component to have the same properties as the real one.

**Parameter Stabilizer**

To parameterize a model stabilizer you firstly need to calculate the total stiffness $c_S$ of the real stabilizer. In the case of a U-bar this is dependent upon the linearized torsional stiffness of the bar $c_{Ba}$ and the linearized bending stiffness $c_{Bl}$ of the lever arms (sometimes referred to as 'blades'). This linear total stiffness is transferred into the rolling stiffness of the front-/rear axle $c_{Ro,S}$. Recalculating with the motion ratio and track of the axle, you get the torsional stiffness of the stabilizer mechanism $c_{S,\theta}$, which is to be entered into IPGKinematics.

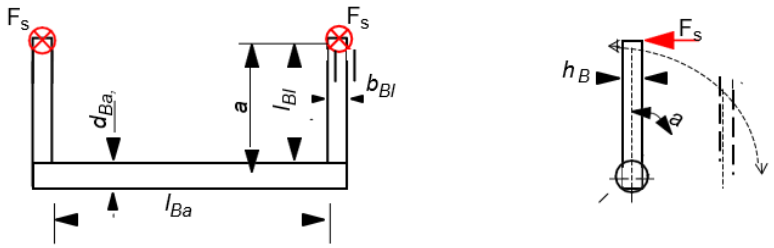



Figure B.6 - Geometrical notations for the calculation of the substituted stabilizer stiffness

In order to calculate the linearized torsional stiffness of the bar $c_{Ba}$ you need to calculate the torsional rigidity of the cross section $I_{p,Ba}$, with $d_{Ba,o}$ as outer diameter and $d_{Ba,i}$ as inner diameter.

$$I_{p,Ba} = \frac{\pi \cdot (d_{Ba,o}^4 - d_{Ba,i}^4)}{32} \quad \textbf{(EQ 34)}$$

With the stabilizer bar's shear modulus G, the torsional rigidity of the cross section $I_{p,Ba}$, the length of the lever arm *a* (with forces acting at *a*), and the length of the bar $l_{Ba,}$ you get $c_{Ba}$:

$$c_{Ba} = \frac{G \cdot I_{p,Ba}}{a^2 \cdot l_{Ba}} \quad \textbf{(EQ 35)}$$

To determine the blades' bending stiffness you first need to calculate the bending rigidity of their cross section. Many teams employ adjustable blades which means that the cross section on which the forces act can change. To account for this, you should calculate the rigidity of the blade in both the 0 deg and 90 deg positions. Using (EQ 38), the combined rigidity of the cross sections for any bending angle $\alpha_{Bl}$ are established:

$$l_{Bl,0°} = \frac{b_{Bl} \cdot h_{Bl}^3}{12} \quad \textbf{(EQ 36)}$$

$$l_{Bl,90°} = \frac{b_{Bl}^3 \cdot h_{Bl}}{12} \quad \textbf{(EQ 37)}$$

$$l_{Bl} = l_{Bl,0°} - \frac{\alpha_{Bl} \cdot l_{Bl,0°}}{90°} - \frac{\alpha_{Bl} \cdot l_{Bl,90°}}{90°} \quad \textbf{(EQ 38)}$$

To calculate the bending stiffness of the blade, you need the Young's modulus, E, of the blade material and the springing length of the blade $l_{Bl}$.

$$c_{Bl} = \frac{3 \cdot E \cdot I_{Bl}}{2 \cdot l_{Bl}^3} \quad \textbf{(EQ 39)}$$

By adding the linearized torsional stiffness of the bar to the stiffness of the blade, the total stiffness of the real stabilizer $c_S$ for a force acting at the end of a blade is gained.

$$c_s = \frac{c_{Ba} \cdot c_{Bl}}{c_{Ba} \cdot c_{Bl}} \quad \textbf{(EQ 40)}$$

To contribute the stiffness of the stabilizer into the overall stiffness of the U-bar, the roll stiffness of the axle (that the stabilizer is associated with) has to be calculated. $s/s_f$ is the motion ratio of the wheel (*s)* and stabilizer blade-end travel ($s_f$). *b* is the track width of the axle.

$$c_{Ro,s} = \frac{c_s}{\left(s/s_f\right)^2} \cdot b^2 \cdot \frac{\pi}{180°} \quad \textbf{(EQ 41)}$$

Using (EQ 42) it is possible to calculate the linearized stiffness of the U-bar $c_{S,x}$. This value is needed in the parametrization of the stabilizer in CarMaker. Currently, it is only needed to calculate the torsional stiffness in (EQ 43) using *a* = 200 mm as the blade length of the template U-bar.

The value of $c_{S,\theta}$ must be entered as the torsional spring rate in IPGKinematics within the *Parameters Stabilizer Bar* section.

$$c_{s,x} = \frac{c_{Ro,s}}{2 \cdot b^2} \cdot \frac{180°}{\pi} \quad \textbf{(EQ 42)}$$

$$c_{s,\theta} = c_{s,x} \cdot a'^2 \cdot \frac{\pi}{180°} \quad \textbf{(EQ 43)}$$

**Spring / Parameters Spring**

There are two options to parametrize the spring: linear and non-linear characteristics. For a linear progression the only required value is the spring rate. It can be determined by a simple tension test. With this test, the spring rate results from the quotient of applied force $\Delta F$ and deflection $\Delta s$:

$$c = \frac{\Delta F}{\Delta s} = \frac{m_1 \cdot g}{\Delta s} \quad \textbf{(EQ 44)}$$

A non-linear spring rate should only be selected if it is truly non-linear!

**Parameter Pull-/Pushrod**

With this parameter (also called spring rate), the elastic deflection of the pull-/pushrod is taken into consideration. Therefore, the pull-/pushrod is seen as a virtual spring. Accordingly, the unit is N/mm. By knowing the material and the geometry of the pull-/pushrod you can assume that the stresses lie within the elastic region and thus apply Hooke's law. The force triggering a deflection of 1mm of the pull-/pushrod is required:

$$F = \frac{\sigma}{A} = \frac{\sigma}{\pi \cdot r^2} = \frac{E \cdot \varepsilon}{\pi \cdot r^2} = \frac{E \cdot \Delta l}{(\pi \cdot r^2) \cdot l_0} = \frac{E \cdot 1mm}{(\pi \cdot r^2) \cdot l_0} \quad \textbf{(EQ 45)}$$

For a radius of *r* = 20 mm, a length of *l* = 430 mm and an E modulus of *E* = 70 GPa (Aluminum 7075) a force of *F* = 55 kN is required. Hence, this pull-/pushrod would have a spring rate of 55 kN/mm.

However, on a Formula Student car such high forces will never be applied to a pull-/ pushrod. You can therefore assume that a pull-/pushrod won't be deformed elastically during normal operation. To achieve this within IPGKinematics you needn't put actual values for the spring rate as this would also increase the simulation duration needlessly. Using an exact spring characteristic, as explained in case of the axle spring, will not produce better results. Additionally, there is a risk of damaging the pull-/pushrod during a live test!

**Spring Stabi Torsion Bar**

With this option you can add a 3rd spring to your axle, which is actuated in response to parallel bump of the left and right wheels.

Example values for each input are provided within Chapter 4 if properties of a particular component are not known.

## Bushings

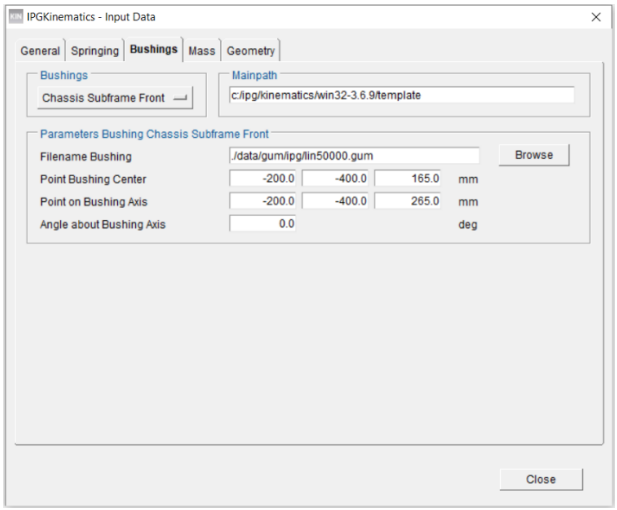



Figure B.7 - Bushings tab

**Parameters Bushing**

Control parameters for selecting the bushings. All parameters for a bushing are entered into this field.

To simulate nonexistent bushings in IPGKinematics they are defined as extremely rigid (E = 5000 N/mm$^2$ = 5 GPa). This stiffens the bushings in all directions. Therefore, the bushings can always be orientated parallel to the stationary coordinate system. That means you do not have to enter any value for *Angle about Bushing Axis.* Further information can be found in the IPGKinematics Reference Manual.

## Mass

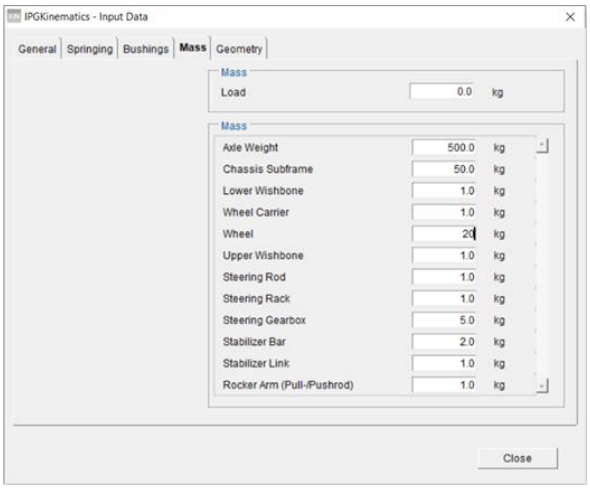



Figure B.8 - Mass tab

**Mass**

Accurate data for each of the listed components should be entered into the corresponding fields.

Example values for each input are provided within Chapter 4 if properties of a particular component are not known.

## Geometry

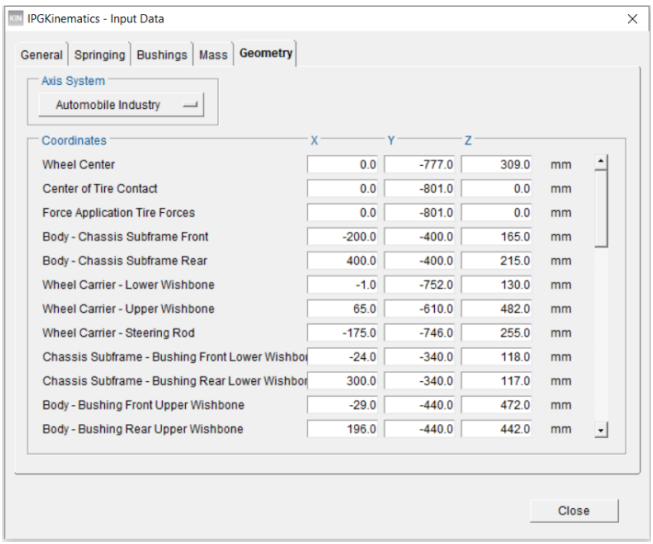



Figure B.9 - Geometry tab

**Coordinates**

The coordinates defined in this section are the most important of all. The calculations of the kinematics are based upon these values. For this reason, the values of these points should be determined carefully in order to generate an accurate as possible model.

Unlike a Forces Off simulation, the coordinates of components such as the Stabilizer bar must be included, as per the table below.

Table B.1 - Forces On coordinates guide

| Coordinates | Description |
|---|---|
| Force Application Tire Forces | Usually the same value as *Center of Tire Contact* |
| Body - Chassis Subframe Front/ Rear | As mentioned, Formula Student cars don't usually have subframes. To give the program feasible values feasible values, enter the mounting points of the lower wishbones to the chassis. |
| Chassis Subframe - Bushing Front/Rear Lower Wishbone | As the subframe is deactivated this corresponds to the mounting points of the lower wishbones to the body. |
| Spring/Damper Body | Although these points are not used for a pull-/pushrod actuated suspension, feasible values are required as IPGKinematics will generate an error if they are omitted. Thus, a good value to choose is the same as Wheel Carrier - Lower Wishbone + 500 mm in height. |
| Spring/Damper Wheel suspension | Although these points are not used for a pull-/pushrod actuated suspension, feasible values are required as IPGKinematics will generate an error if they are omitted. Thus, a good value to choose is the same as Wheel Carrier - Lower Wishbone |
| Axle Drive Shaft - Differential/Wheel | Even if you are simulating a non-driven shaft, plausible values are still necessary. A reasonable value to choose is the same as the Wheel Center and the middle point of the axle. |
| Rotation Axis - Rocker Arm | Can be any point upon the rotation axis of the rocker arm. |
| Stabilizer Link - Wheel Carrier | Equal to the Wheel Center |

| Coordinates | Description |
|---|---|
| Stabilizer Bar - Stabilizer Link | Add 100 mm in height to the former point |
| Stabilizer Bar - Chassis Subframe | Add 200 mm in x direction to the former point. |
| Stabilizer Bar - Chassis Subframe | Add 200 mm in x direction to the former point. |
| 3. Spring Stabi Torsion Bar - Torsion Bar | This defines the connection point of the third spring, the stabilizer and the rotation axis of the torsion bar. Therefore, it needs to be placed reasonably even if there is no third spring used. |
| "Marker Damper Rocker Arm": | This does not affect the simulation results and is not required for Formula Student cars. The only condition for this point is that the y-coordinate must not be 0 as this leads to an error during the simulation. |
| Car Body (Turning Circle Diameter) | Defines the space required by the car for turning. It describes the difference between turning track diameter and turning circle diameter. Input values are the x- and y-coordinates of the 'radius' of the outer skin. For Formula Student cars this value is of little importance as the minimum turning track diameter is more important (as shown in Figure 4.19). Hence, you can set any values. |

If a T-Stabilizer is being used by choosing push/pull rod as the stabilizer bar fixing, the stabilizer points shown in Figure 4.17 (Chapter 4) above will remain unused during the simulation. Never- theless, the coordinates for the U-Stabilizer need feasible values and should be parameterized as explained above.

To illustrate the meaning of the points of the double wishbone model with the U-Stabilizer explained above, compare Figure 4.17: with Table B.1.

The points shown in Figure 4.18 describe the T-Stabilizer. In case of using a U-Stabilizer, these coordinates should keep the default values.

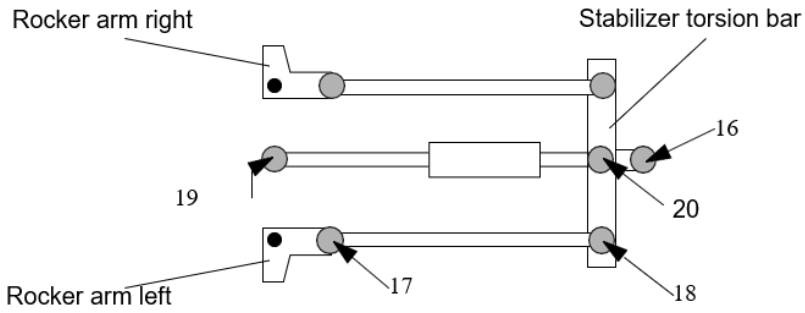
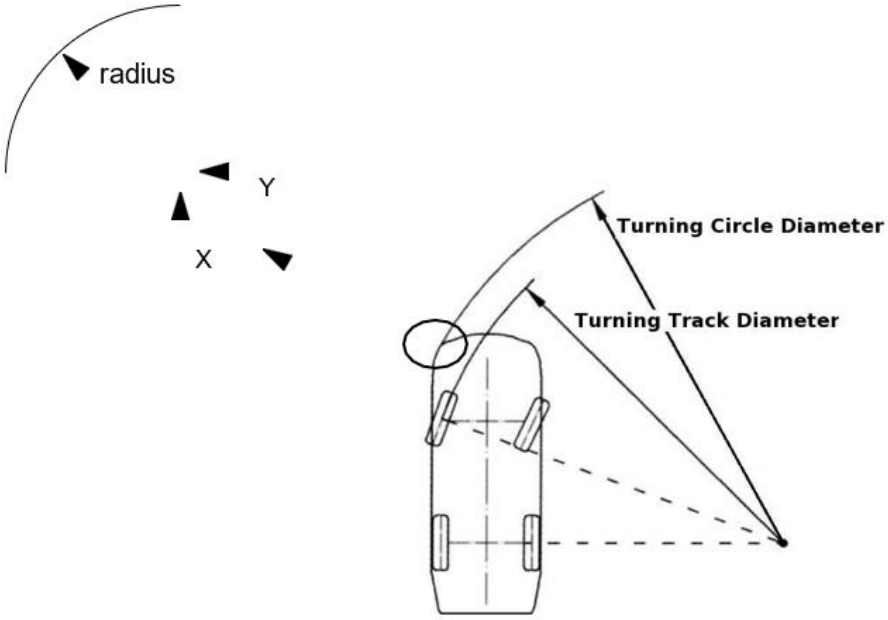



Figure B.10 - Geometry of Stabilizer via push/pull rod (top view)

Table B.2 - Listing of the geometry points in Figure B.10

| Number | Item |
|---|---|
| 16 | Joint Stabi Torsion Bar - Body |
| 17 | Rod Stabi Torsion Bar - Rocker Arm |
| 18 | Rod Stabi Torsion Bar - Torsion Bar |
| 19 | 3. Spring Stabi Torsion Bar - Body |
| 20 | 3. Spring Stabi Torsion Bar - Torsion Bar |


radius
Y
X
Turning Circle Diameter
Turning Track Diameter

Appendix C

# Data Files

## C.1 Example of a TYDEX Code

```
!# $Id: TUTO_TYDEX_TEST.tdx,v 1.2 2005/06/30 14:02:50 cr Exp $
**HEADER
!---------!-----------------------------!---------!---------!
! Key ! Comment ! Unit ! value !
!---------!-----------------------------!---------!---------!
RELEASE Release of TYDEX-Format 1.3
SUPPLIER Data supplier DEMO
DATE Date 07/98
CLCKTIME Clocktime 12:05
**COMMENTS
! Demo with TIME measurement procedure
! New tyre used
**CONSTANTS
!---------!-----------------------------!---------!---------!
! Key ! Comment ! Unit ! value !
!---------!-----------------------------!---------!---------!
TESTRIG Test rig DEMO
LOCATION Location DEMO
MANUFACT Manufacturer of the tyre DEMO
IDENTITY Identity of the tyre DEMO
```

NOMWIDTH Nominal section width of tyre mm 195

ASPRATIO Nominal aspect ratio % 65

RIMDIAME Nominal rim diameter inch 15

RIMWIDTH Rim width inch 6.5

RIMPROF Rim profile J

REFSIDEW Reference sidewall DOT

TYWHASSB Tyre-wheel assembly REFSWLEFT

TRDDEPB tread depth before mm 8

TRDDEPA tread depth after mm 5.4

TRCKSURF Surface of track SAFETYWALK

AMBITEMP Ambient temperature deg C 25

PATHRADC Path radius m INFINITY

INFLPRES Inflation pressure bar 2.5

TRCKCOND Track condition DRY

OVALLDIA Overall diameter mm 580

KROLRAD kinematic roll radius mm 280

**MODELPARAMETERS

!---------!------------------------------!---------!---------!----!

! Key ! Comment ! Unit ! value !factor!

!---------!------------------------------!---------!---------!----!

ITVS Vertikale Reifenfeder 350000.

ITVD Vertikaler Reifendaempfer 1000.

ITRLLO Relaxationslaenge laengs 0.05

ITRLLA Relaxationsl. quer 0.1

ITSSCLO StillStandskoeffizient long 0.01

ITSSCLA StillStandskoeffizient quer 0.01

ITRORETL roll resistance Trq/Load 0.01

**MEASURCHANNELS

!---------!-------------------------!---------!--------!-------!--

! Key ! Comment ! Unit ! factor ! offset!offset

!---------!-------------------------!---------!--------!-------!--

!

!physical value [Unit]= factor*(measured value + offset1) + offset2!

FZH Vertical force N 1 0 0

SLIPANGL Slip angle deg 1 0 0

INCLANGL Inclination angle deg 1 0 0

FYH Lateral force N 1 0 0

MZH Aligning moment Nm 1 0 0

**MEASURDATA

1211.00 -0.50 0.00 211.21 -2.27

3021.70 -0.50 0.00 460.01 -8.08

4828.00 -0.50 0.00 641.08 -15.50

6635.91 -0.50 0.00 748.59 -23.11

8444.54 -0.50 0.00 782.56 -29.70

8451.64 -1.00 0.00 1650.01 -74.74

6640.98 -1.00 0.00 1549.56 -55.52

4835.30 -1.00 0.00 1296.28 -34.95

3023.70 -1.00 0.00 883.88 -16.36

1217.11 -1.00 0.00 359.97 -3.51

1210.54 0.01 0.00 15.75 0.29

3022.89 0.00 0.00 -31.05 2.83

4828.88 0.00 0.00 -80.67 7.00

6641.35 0.00 0.00 -106.23 11.94

8447.86 0.01 0.00 -130.66 18.29

8447.03 0.51 0.00 -1014.16 64.33

6643.04 0.50 0.00 -927.98 46.09

4837.24 0.50 0.00 -763.21 28.49

3021.81 0.50 0.00 -478.76 13.10

1211.60 0.50 0.00 -143.84 2.69

1213.37 1.00 0.00 -305.18 4.09

3024.83 1.00 0.00 -890.32 20.23

4830.25 1.00 0.00 -1383.04 45.36

6639.96 1.00 0.00 -1677.46 75.28

8446.77 1.00 -0.01 -1816.43 105.83

8439.81 0.00 -3.04 323.53 28.80

4827.96 0.00 -3.03 216.27 17.45

1209.10 0.00 -3.02 99.88 6.40

1213.58 0.00 -1.00 72.70 2.71

4829.41 0.00 -1.00 45.71 10.10

8438.02 0.01 -1.00 39.30 21.39

8454.52 0.00 1.01 -213.57 13.88

4838.03 0.00 1.01 -108.97 3.08

1213.67 0.00 1.01 32.59 -2.27

1216.10 0.00 3.03 -7.26 -5.38

4843.10 0.00 3.02 -259.95 -4.09
8447.27 0.01 3.02 -467.54 6.22
6650.47 0.00 5.03 -561.46 -6.26
4839.59 -0.50 3.00 470.52 -28.39
3027.08 -0.50 3.00 355.43 -18.89
1210.09 -1.00 0.98 377.33 -6.01
1211.40 1.01 -1.04 -294.24 6.04
3013.27 0.51 -3.05 -314.68 23.15
4821.91 0.50 -3.04 -496.45 40.38
6632.49 0.00 -5.05 466.31 30.36
6204.67 -1.11 1.42 1596.27 -65.11
4671.47 1.11 -1.42 -1385.77 52.51
4676.80 -1.11 1.42 1345.09 -44.21
6192.29 1.11 -1.43 -1665.13 79.80
8480.00 -9.68 5.56 6653.04 -70.43
2398.14 9.69 -5.58 -2400.98 0.98
2387.44 -9.69 5.57 2326.81 -5.20
8475.57 9.70 -5.57 -6594.03 67.88
6950.80 -2.12 2.82 2646.00 -128.43
3882.44 2.12 -2.83 -1856.83 57.17
3895.17 -2.12 2.82 1863.24 -56.02
6938.60 2.12 -2.83 -2774.20 140.19
9285.47 -12.11 5.91 7232.49 -48.49
1640.53 12.12 -5.93 -1672.14 -0.80
1638.45 -12.12 5.92 1657.14 -0.70
9281.01 12.13 -5.92 -7192.70 49.90
7713.80 -5.05 4.22 5267.92 -193.88
3121.37 5.04 -4.23 -2620.77 38.67
3122.04 -5.04 4.22 2648.05 -41.97
7693.48 5.05 -4.23 -5236.87 186.65
4367.24 -2.32 0.61 2292.76 -57.70
1663.58 2.32 -0.62 -834.64 11.48
1662.51 -2.32 0.61 904.85 -11.56
4362.46 2.32 -0.62 -2380.05 61.50
6174.07 -9.34 1.39 5559.15 -26.00
773.02 9.35 -1.41 -915.13 0.99
798.32 -9.34 1.40 926.42 -2.22

6179.77 9.35 -1.40 -5566.71 16.28

4819.49 -4.12 0.81 3842.98 -81.60

1213.00 4.12 -0.82 -1007.44 7.96

1212.24 -4.12 0.81 1009.89 -7.99

4821.89 4.13 -0.82 -3861.91 74.69

5271.42 -5.83 1.00 4830.02 -72.66

770.04 5.83 -1.01 -804.12 4.11

763.44 -5.83 1.00 767.93 -3.31

5278.38 5.84 -1.01 -4780.28 58.53

8458.68 -12.04 -0.01 6912.70 -18.90

1199.51 12.05 0.00 -1330.46 -1.45

1198.00 -12.05 0.00 1343.88 1.13

8459.97 12.06 0.00 -6950.92 10.93

6649.84 -5.83 -0.01 5764.30 -102.48

1808.09 5.83 0.00 -1829.74 8.55

1806.19 -5.83 0.00 1823.33 -9.80

6649.88 5.84 0.01 -5683.13 86.77

7257.76 -8.34 -0.01 6394.02 -51.62

1198.32 8.34 0.00 -1353.07 1.28

1198.65 -8.34 0.00 1366.09 -1.91

7259.76 8.35 0.01 -6374.97 42.24

1207.40 -12.06 -1.98 1370.53 3.10

7237.09 12.06 1.98 -6209.88 -1.17

5444.26 -1.71 1.61 2181.08 -78.76

3012.59 1.72 -1.63 -1390.39 34.41

3011.54 -1.71 1.61 1422.13 -33.04

5438.58 1.72 -1.63 -2243.23 85.30

7262.97 -9.76 3.97 6109.24 -40.52

1209.37 9.78 -3.99 -1327.95 2.74

1198.86 -9.77 3.98 1268.81 -1.82

7269.73 9.77 -3.98 -6200.20 42.36

6035.69 -3.62 2.41 4085.05 -136.36

2396.46 3.63 -2.43 -1914.02 29.52

2396.28 -3.62 2.42 1937.77 -29.37

6025.61 3.63 -2.43 -4096.58 130.20

6655.31 -6.74 3.20 5750.83 -91.56

1809.13 6.75 -3.22 -1887.68 10.09

```
1801.47 -6.74 3.21 1851.49 -9.41
6646.61 6.75 -3.21 -5712.57 87.20
**END
```