Formula Student vehicle analysis by means of simulation

Razenberg, J.A.

Published: 01/01/2006

Document Version
Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
• A submitted manuscript is the author's version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

Citation for published version (APA):

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 11. Dec. 2018
Formula Student vehicle analysis by means of simulation

J.A. Razenberg

DCT 2006.074

Bachelor End Project report

Supervisor: Dr. Ir. I.J.M. Besselink

Technische Universiteit Eindhoven
Department Mechanical Engineering
Dynamics and Control Technology Group

Eindhoven, April, 2006
# Contents

1. Introduction .......................................................................................................................... 2
2. The Car Simulation Model .................................................................................................... 3
   2.1 Introduction ..................................................................................................................... 3
   2.2 The model ...................................................................................................................... 3
   2.3 Model Parameters ......................................................................................................... 4
   2.4 An example ................................................................................................................... 6
3. The test program ................................................................................................................... 7
   3.1 Introduction ................................................................................................................... 7
   3.2 Test program built up .................................................................................................... 7
   3.3 Post processing ............................................................................................................. 8
   3.4 The tests ....................................................................................................................... 11
      3.4.1 Static equilibrium test ......................................................................................... 11
      3.4.2 Acceleration test ................................................................................................. 13
      3.4.3 Brake test ............................................................................................................. 15
      3.4.4 J-turn test ............................................................................................................. 17
      3.4.5 Fishhook test ....................................................................................................... 19
      3.4.6 Slalom test .......................................................................................................... 21
      3.4.7 Steady-state circular test ..................................................................................... 23
      3.4.8 Bumpy road test ................................................................................................. 25
      3.4.9 Separate suspension test ...................................................................................... 27
      3.4.10 Brake turn test ................................................................................................. 29
4. Virtual Reality Animation .................................................................................................... 30
   4.1 Introduction ................................................................................................................... 30
   4.2 The VR animation program ............................................................................................ 30
   4.3 Example ......................................................................................................................... 31
   4.4 Unfinished work ............................................................................................................ 32
5. Discussion ............................................................................................................................. 34
   5.1 Conclusion ..................................................................................................................... 34
   5.2 Recommendations ....................................................................................................... 34
References ................................................................................................................................... 35
List of symbols ....................................................................................................................... 36
APPENDICES .......................................................................................................................... 37
A: List of available model output .......................................................................................... 37
B: Example forcereport.out ................................................................................................. 38
C: Description of data ........................................................................................................... 39
D: Example testresults.out ................................................................................................. 40
E: Force definition ............................................................................................................... 41
F: Explanation of variables for \( \kappa \) .................................................................................. 42
1. Introduction

Since 2003 the Formula Student Racing Team Eindhoven (FSRTE) has been participating in the Formula Student Racing competition. This competition isn’t just about winning on the tarmac, the main core is formed by the technical aspects off the tarmac: analysis and good funded design choices are of great importance to win this race among students from all over the world.

This is the final probationary year for FSRTE; if they don’t come up with a car that is capable to compete with the other cars in a race, the Formula Student project has to be stopped. The number of students has therefore been increased from less than twenty in previous years, up to about fifty in this year. The previous teams of FSRTE were not able to set a working vehicle on the tarmac.

Because there is no existing car yet, the car has to be tested by means of simulation. This makes it possible to get insight in the vehicle behaviour before the actual car is built. A car simulation model, based on the current model of Bram de Jong, Bas de Waal and Ruud van der Aalst, has been created for that purpose.

This report first gives an insight in the built-up of the simulation model [Chapter 2], and then the different simulation tests will be discussed in two steps: the model of the test itself and the results [Chapter 3]. Finally the Virtual Animation will be discussed [Chapter 4].
2. The Car Simulation Model

2.1 Introduction

The current model from Bram de Jong, Bas de Waal and Ruud van der Aalst is built with MATLAB (v. 7.0.4) / SimMechanics (v. 2.2.2). The advantage of the model is that it is easy to make adjustments in the setup of the car and that the results follow directly afterwards. This way there is no need for building a new car to see what an adjustment in the setup of the car would mean for the vehicle’s behaviour and what would be the optimum setup.

2.2 The model

The model is stored in a Simulink Library File, defined as FS_car.mdl, since the model has to perform several tests. The same file can be used for all different tests. The library file is built up in four subsystems. These subsystems are the front suspension, the rear suspension, the engine with the gearbox and the chassis. This is visible as the four blocks under the mask of the library file.

Both the suspension systems represent the double-wishbone configuration and the parts responsible for providing bump and roll stiffness. These parts are modelled according to the design in which the roll and the bump is controlled separately. A steering rack in combination with a steer rod upright connection can be found in the front suspension system. A drive shaft is placed in the rear suspension system, responsible for putting a drive moment on the wheels. The brakes are included in the suspension systems.

The chassis is modelled rigid and has six degrees of freedom. The use of joints in the SimMechanics model makes it possible for the car to experience degrees of freedom like roll or pitch. An aerodynamics block in the chassis subsystem implements an approximation of air resistance and lift or down force. Without this block, the car would reach impossible high speeds. On the other hand, for simulation it is not that important because in the Formula Student Competition the car will probably not reach speeds over 110 km/h due to the nature of the circuit which consists of many curves and bends.

The last subsystem is the engine and gearbox block. In this block the torque working on the left and right drive shaft is calculated. The principle used here is:

\[
P = T_{\text{engine}} \cdot \omega_{\text{engine}}
\]

Formula (2.1) is used for calculating the torque delivered by the engine. The power P is constant in the model. The gearbox reduction is taken into account here while calculating \(\omega_{\text{engine}}\) from the wheel speed. The whole is modelled like a CVT (Continuous Variable Transmission). In practice this is not the case but for simulation it is sufficient. A saturation-block is added in the model to limit the maximum torque that the engine can provide. Then this calculated torque is multiplied with the throttle input, ranging zero to one, before it is equally split over the right and left drive shafts. In practice this does not represent the real car in this respect, because a Torsen differential is used.

Finally we have the Delft-Tyre blocks, which are used to model the tyre on road contact as defined by the Magic Formula tyre model from the Technical University in Delft.
2.3 Model Parameters

To use the model for simulations of the formula student 2006 design, a lot of values need to be assigned to parameters as masses, inertia's, stiffnesses, coordinates etc. These values are defined in an m-file: *FS_modeldata.m*. In the base model these parameters are used.

<table>
<thead>
<tr>
<th>C1</th>
<th>connection upper A-arm - chassis</th>
<th>C8</th>
<th>connection upright - steering rod</th>
</tr>
</thead>
<tbody>
<tr>
<td>C2</td>
<td>connection upper A-arm - upright</td>
<td>C9</td>
<td>connection steering rod - steering rack</td>
</tr>
<tr>
<td>C3</td>
<td>connection upper A-arm - chassis</td>
<td>C10</td>
<td>connection lower A-arm - pushrod</td>
</tr>
<tr>
<td>C4</td>
<td>connection lower A-arm - chassis</td>
<td>C11</td>
<td>connection pushrod - rocker</td>
</tr>
<tr>
<td>C5</td>
<td>connection lower A-arm - upright</td>
<td>C12</td>
<td>connection rocker - chassis</td>
</tr>
<tr>
<td>C6</td>
<td>connection lower A-arm - chassis</td>
<td>C13</td>
<td>connection rocker - spring</td>
</tr>
<tr>
<td>C7</td>
<td>connection upright - wheel center</td>
<td>C14</td>
<td>connection rocker - rollbar rod</td>
</tr>
</tbody>
</table>

Table 2.1: Suspension geometry naming and numbering

![Suspension geometry numbering](image)

Figure 2.1: Suspension geometry numbering
One of the important things that the user has to define in `FS_modeldata.m` is the geometry of both the front and rear suspensions. This has only been done for the left side of the vehicle because in the model this is mirrored for the right side. The numbering that is used in `FS_modeldata.m` for specifying the coordinates of suspension points as shown in figure (2.1). In the rear suspension the steering rod is replaced with a tie rod.

![Figure 2.2: Definition coordinate system used in FS_modeldata.m](image)

Furthermore the spring stiffness and damping coefficient have to be defined. The spring must be compressed before connecting between the rockers. This compression is called the bump offset. This holds that a positive number increases ride height.

As a last remark the initial velocity and simulation time always have to be given in the m-file from where the simulation, which uses the library model, is started. This initial velocity should be named `Vx_init` and the simulation time `tmax`, as you can see in the example in the next paragraph.

Beside the `FS_modeldata.m` file and a test specific m-file that starts the simulation there are two other files with parameters: `FS_fronttyre.tpf` and `FS_reartyre.tpf`. In these files properties like tyre dimensions, tyre mass and tyre moment of inertia for the Delft-Tyre blocks are defined.
2.4 An example

Now the vehicle model definition is complete with the library file `FS_car.mdl` and the parameter definitions in `FS_modeldata.m`. The next step is to make a simulation with the vehicle model. The model has three inputs: steer input, throttle input and brake input. The last two have a range from zero to one which corresponds with no throttle or brake input to full throttle or brake respectively. The steer input is defined in radians and has a range from \(-\pi\) to \(\pi\). A function in the front suspension system block changes this in an input for the wheels ranging from -30 degrees to 30 degrees.

As an example, we perform a simple simulation where the car drives with an initial speed of 50 km/h. It has half throttle and after 5 seconds it's going to make a right turn with a steering wheel angle of 20 degrees. After 12 seconds throttle becomes zero and full braking is applied. To do this simulation first a simulink file (`FS_example.mdl`) is made, as shown in figure (2.3). Here the full vehicle is copied from the library. The input data as shown in the listing below, is stored in a m-file; `examptest.m`. The only thing the user has to do is to run this m file, which will automatically start the file `FS_example.mdl` and perform the simulation.

![Figure 2.3: FS_example.mdl](image)

```matlab
% Runs FS_modeldata.m to be sure that % the latest parameters are available.
% Defines the end time of the simulation
tmax=20;
% Reference moments in time for steer input
 steer_input = [0  5  7  tmax ;
                0  0  20*pi/180  20*pi/180];
% Defines curve of steer input
% Reference moments in time for throttle input
 throttle_input = [0 12 12 tmax ;
                  0.5 0.5 0  0 ];
% Defines curve of throttle input
% Reference moments in time for brake input
 brake_input = [0 12 12 tmax ;
                0 0 1 1 ];
% Defines curve of brake input
% Initial velocity [m/s]
Vx_init = 50/3.6;
sim('FS_example') % Runs the simulation
```

Listing 2.1: examptest.m
3. The test program

3.1 Introduction
In this chapter all tests performed by the simulation model of the car are discussed. To get a good insight in the vehicle behaviour 10 tests are performed. The tests are:

1. static equilibrium test
2. acceleration test
3. brake test
4. J-turn test
5. fishhook test
6. slalom test
7. steady state circular test
8. bumpy road test
9. separate suspension test
10. brake-turn test

Most of the tests are based on real situations the car will experience during the Formula Student race.

3.2 Test program built up
Each test consists of six main components:

- the library file, denoted $FS\_car.mdl$
- the vehicle parameter file, denoted $FS\_modeldata.m$
- the tyre parameter files $FS\_fronttyre.tpf$ and $FS\_reartyre.tpf$
- a test specific simulink model like $FS\_example.mdl$, where the library file is linked in. It contains three inputs: steer input, throttle input and brake input respectively. Controllers for these inputs are also added to this model if necessary
- the m-file with the specific test values, like $FS\_example.m$
- the m-file $vehicledrivetests.m$

The structure of the test program is kept as simple as possible. The file the user has to call up first is $vehicledrivetests.m$. In the m-file $vehicledrivetests.m$ the user must choose which test he wants to simulate.

The file $vehicledrivetests.m$ refers to the m-file that performs the test the user has chosen. The built up of that file is exactly like the built up of $FS\_example.m$. This file starts running the parameters of $FS\_modeldata.m$, then it gives the different inputs for steering, throttle and braking for the specific test and finally this file performs the test. After performing the test with the test specific simulink file, $vehicledrivetests.m$ lets the user know that the test is finished.

The test specific simulink file uses the data as defined in $FS\_modeldata.m$ (as described in paragraph 2.3). The controllers in this file have to take care that the preferred velocity, steering and braking are maintained. Because the library file is linked in this test specific simulink file, there is no need to build a model of the car for each test. The biggest disadvantage of that would be that the user has to make a new car model for each test after a little adjustment in the settings. Now it is possible to adjust the library file once for all the tests. After the test is performed, the data is stored in a mat-file. This file can later be used for post processing, e.g. analysis of time histories.
Practically each file is built up this way, but there are two exceptions. First one of them is the bumpy road test. This test has its own library file: `FS_car_bump.mdl` wherein the road is defined as in the road data file: `Divineroads2.rdf`. In all the other tests the road is assumed to be flat. The rest of the built up of this test is exactly the same as for the other tests.

The second and last exception is the test defined as separate suspension test. In this test, not the whole vehicle is tested, but only the front suspension and rear suspension are tested separately. The purpose of this test is to analyse the suspension kinematics of the vehicle only.

![Diagram](image)

**Figure 3.1: Schematically representation of the test program built up**

### 3.3 Post processing

The post processing module of the test program is also standardized. The user only has to open the file `post_processing.m`. This file asks the user whether he wants to see data graphs or an animation.

If the user chooses for the data graphs, the file `post_processing.m` links to another file, defined as `post_processing_figures.m`. This file prompts the user to make a choice for which test he wants to see the results of in graphics and will make a report of forces on parts of the suspension, which can be helpfull by dimensioning forces on different components of the vehicle. When the test is chosen, the file refers to the last file `post_processing_figures2.m`. `Post_processing_figures2.m` shows all the figures for the specific test.

If the user chooses for the Virtual Reality Animation, the file `post_processing.m` refers to `post_processing_VRanimations.m`. This file prompts the user for which test the user wants to perform the virtual reality animation. The user must beware that `VR_sim.mdl` has to be closed; otherwise MATLAB is not able to perform the new animation.

When the test is chosen, its specific `mat-data-file` is loaded and the file uses `VR_sim.mdl` for an animation. After performing the animation the program `VR_sim.mdl` is opened automatically and the user can easily open the `VR sink block` in `VR_sim.mdl`. 
As a final note it must be said that the separate suspension simulation test does not have a virtual reality animation. The file `post_processing_VRanimations.m` will show this to the user by giving a warning in the Command Window of MATLAB.

Figure 3.2: Schematic representation of the post processor built up
Figure 3.3: Results static equilibrium test
3.4 The tests

3.4.1 Static equilibrium test

Test description
This test, which is also called ‘standstill test’ or ‘drop test’, calculates the static position of the vehicle. The input for FS_model_standstilltest.mdl is zero for steering angle, throttle and braking, as shown in figure (3.3) and defined in standstilltest.m.

The last thing to mention is the option of giving the car an initial vertical displacement. For this option the initial condition of the model is used and changes the starting height of the vehicle. This initial height can be adjusted by changing the value of d.initialstartingheight in standstilltest.m. The vibrations in the graphs come from the vertical displacement of the vehicle when it starts hanging in its springs by its own weight after release from a height higher than its static height.

Post processing
In figure (3.3) the pitch angle, roll angle, spring compressions, ground clearances and vertical forces caused by its own mass are shown. These values become available after using the file post_processing.m. The most important data is also shown in table (3.1).

<table>
<thead>
<tr>
<th>initial drop height</th>
<th>0.00 [m]</th>
</tr>
</thead>
<tbody>
<tr>
<td>minimum groundclearance front</td>
<td>0.10 [m]</td>
</tr>
<tr>
<td>maximum groundclearance front</td>
<td>0.14 [m]</td>
</tr>
<tr>
<td>minimum groundclearance rear</td>
<td>0.08 [m]</td>
</tr>
<tr>
<td>maximum groundclearance rear</td>
<td>0.10 [m]</td>
</tr>
<tr>
<td>standstill groundclearance front</td>
<td>0.11 [m]</td>
</tr>
<tr>
<td>standstill groundclearance rear</td>
<td>0.09 [m]</td>
</tr>
</tbody>
</table>

Table 3.1: Static Equilibrium Test data

In appendix B the forces on several parts of the suspension are given for this test.
Figure 3.4: Results acceleration test
3.4.2 Acceleration test

Test description
In this test the car has to accelerate as fast as possible. One problem is that full throttle will result in excessive wheel spin. For an optimal acceleration a lot of wheel spin is not demanded, because then the car will accelerate slower than with optimal grip.

The wheel spin or also called slip of a car is parameter sized by the following formula:

$$\kappa = -\frac{V_x - \Omega \cdot r_c}{V_x}$$

(3.1)

The definitions of the used variables can be found in appendix F.

In the model the wheels are modelled with use of Delft-Tyre blocks from TNO Delft. These blocks already have $\kappa$ as an output, so it would be easy to use this for the test results. A disadvantage of this $\kappa$ is that when $V_x$ is zero there will be divided by zero, what obviously is not possible. To deal with this problem a new $\kappa$ is calculated with the suitable approximation:

$$\kappa_{\text{approx}} = -\frac{V_{sx}}{|V_x| + 1}$$

(3.2)

This $\kappa_{\text{approx}}$ can be used instead to make a feedback controller, even at low $\kappa$’s close to zero. The car library file gives this $\kappa_{\text{approx}}$ as a direct feedback output. In FS_model_acceleration.mdl this output is connected to a lookup table. In the file accelerationtest.m the lookup table is defined; the throttle is coupled with the $\kappa_{\text{approx}}$. In the simulink file this results in a feedback controller with $\kappa_{\text{approx}}$ as feedback signal for the throttle. With a controller on this signal the throttle is controlled and the acceleration will be as high as possible because $\kappa_{\text{approx}}$ is kept at its optimum.

Post processing
All the results are shown in figure (3.4). As you can see very clearly is that the throttle is not at the maximal value and that the car accelerates very fast. Within 3.6 seconds the car reaches the speed of 100 km/h. One of the tests in the Formula Student competition is to accelerate for 75 meters. The team with the fastest time wins this match. Our car would be able to run these 75 meters in 4.40 seconds. The maximum speed reached is 116.5 km/h. This is also visible in the testresults.out file after running post_processing.m. Of course we deal with some other influences here which will disadvantage the acceleration. A CVT is used in the model, while in reality the driver has to switch gears which will take time and engine power (the engine can’t be kept at the same optimum speed).
Figure 3.5: Results brake test
3.4.3 Brake test

**Test description**

One of the things that should be tested is the vehicles behaviour during braking. In order to brake as hard as possible, it is important to have maximum grip on the road surface. That means that the slip ratio has to be the same as in the acceleration test, noted that the sign must be opposite.

Actually this test is the opposite of the acceleration test. The input for braking is coupled with the feedback signal for $\kappa_{\text{approx}}$. The velocity of the car is prescribed. At $t=0$ the initial speed $V_x_{\text{init}} = 15$ m/s. After 3 seconds the braking is applied, so after 3.005 seconds the demanded speed is 0 m/s. It is up to the brake controller to apply full braking; keeping in mind that full braking can only be achieved by using the optimal value for the slip ratio.

To keep the car at its initial speed during the first 3 seconds, a feedback controller is built in the simulink model for this test: $F5\_model\_braketest.mdl$. The controllers are a normal PI-controller with gain 10 for the P control and gain 50 for the I control (figure 3.6). Without this controller the velocity of the car would decrease before it starts braking due to air resistance and tyre resistance. Switches take care that the signal doesn’t work when it is not supposed to or starts working at the right time; throttle stops at $d$\_brakemoment and braking starts at the same time.

![Figure 3.6: FS_model_braketest.mdl](image)

**Post processing**

In the results (figure 3.5) it is easy to see that full braking results in a negative value for $\kappa_{\text{approx}}$. The exact value for $\kappa_{\text{approx}}$ is about -0.05. The car brakes from an initial speed of 15 m/s. It stops in 1.65 seconds and the braking distance is 11.27 meters. These exact values can be found in testresults.out after running the post processor for this test.
Figure 3.7: Results J-turn test
3.4.4 J-turn test

Test description
This test illustrates the dynamic vehicle behaviour when entering a curve. The car has an initial velocity $V_{x\_init}$. In the test shown, this initial velocity is set at 100 km/h. Two seconds after the test is started the steering angle is set to 25 degrees to the right and 5.8 seconds later the steering angle is set back to zero. The curve that the car will make with this steering input has the shape of the letter J, that’s why this test is called the J-turn test. It’s important to keep the car at its original velocity and for this a feedback controller for the throttle is built in the model.

Interesting things to know of the vehicles behaviour in a curve are: angular velocity of each wheel, the lateral forces and accelerations, the forces lateral to the road surface for each wheel, the behaviour of the springs and dampers and the dynamics of the vehicle behaviour like overshoot.

Post processing
Looking at the results shown in figure (3.7), it appears that the throttle position has to change in a curve to maintain a constant velocity. Looking at the graph for the vertical tyre force $F_z$ it is easy to see that all wheels stay on the ground. Lifting a rear wheel from the ground has a problem: the lifted wheel goes spinning excessively around when it is driven by the engine while the other rear wheel does not get enough weight and loses grip, as shown in the graph for $\omega$. As soon as the forces lateral to the driving direction are at their maximum, the right wheel spins around faster than the other wheels.

In this simulation the magnitude of the lateral acceleration is 10 m/s$^2$. This is about one G. The magnitude of the roll angle is 1.5 degrees which is very acceptable for a car driving at 100 km/h and making a curve. The suspension of the Formula Student car is a lot stiffer than that of a normal passenger car.

The thing that is most concerning is the overshoot. When the car has entered the curve, the rear of the car brakes out and slides back again as shown in the graph for the yawrate. This overshoot to and fro can also be found in the graph for the lateral acceleration. The overshoot decreases and the vehicle stays on track.
Figure 3.8: Results fishhook test
3.4.5 Fishhook test

Test description
To see what the roll-over stability of the vehicle is, a test in which the car has to enter two curves in opposite direction is designed. The path of the car is shown in the fourth graph of figure (3.8). This path has the shape of a fishhook; the reason for the name of this test. First the car makes a turn to the right with a steering angle of $\frac{\pi}{2}$ radians and then the same angle of $\frac{\pi}{2}$ radians to the left after 6.7 seconds. The initial velocity is set to 20 m/s and is maintained during the test by a controller on the throttle input. The steering signal turns sign at the moment the roll angle to the left is steady and at its maximum.

Post processing
The graphs in figure (3.8) show that the car maintains its velocity. The car doesn’t lift any of its wheels from the road surface. Like in the J-turn test, the lateral acceleration is about 10 m/s$^2$. This means that the lateral acceleration is about one G. This happens as well in the curve to the right as in the curve to the left. The roll angle also is the same in both curves; about 1.8 degrees. It should be mentioned that one wheel spins extremely. The rear wheel at the left, driven by the engine, starts spinning around when the car has started his curve to the left. After a second the wheel finds its grip back on the road surface and the vehicle continues its way.

A possible problem of the cruise controller (the feedback controller for the velocity) is that once a wheel is spinning, all throttle is put in the spinning wheel and the velocity of the vehicle would decrease. Because the velocity decreases the cruise controller wants to raise the throttle even further to compensate the decrease; so the lifted wheel spins even faster. This could become a vicious circle and the system might become unstable.

It seems from the results that the wheel on the ground also responds on the extra torque from the engine. This way the vehicle maintains its velocity and the wheel does not spin extremely after a short while. This is due to ideal differential with the 50 – 50 split of the torque. As said before, in reality a Torsen differential is used in stead of the ideal differential and the torque is not split equally. This influence should still be tested.
Lateral acceleration response to steering wheel angle

Roll angle response to steering wheel angle

Yaw velocity response to steering wheel angle

Roll angle response to lateral acceleration

Figure 3.9: Results slalom test
3.4.6 Slalom test

Test description
As said before, the circuit of the competition consists of many curves. It’s important to know how the vehicle responds to the steering input. To maintain its initial velocity, the simulink model has a cruise control. It is exactly the same feedback controller for velocity as in the previous tests.

The steering input consists of a chirp signal; this is a signal with a starting frequency that increases until a defined frequency is reached on a predefined moment in time. These values are assigned in the m-file for this test: *slalomtest.m*. The begin frequency is defined as `c.initialfrequency`, the end frequency is defined as `c.endfrequency` and the moment in time where this frequency has to be reached is defined as `c.targettime`. The estimation for the yaw frequency for these kinds of vehicles for example is 8.8 Hz. \(^1\) A high yaw frequency will mean that the switch between a left and right curve can be made very quickly. In this test there is chosen for a reasonable frequency range from 0 to 13 Hz.

Post processing
The first graphs in figure (3.9) speak for themselves, but the last graphs are Bode plots found with transfer functions. These transfer functions are calculated by use of the MATLAB command ‘*tfe*’ in combination with ‘*cohere*’ for checking the coherence. The transfer functions calculated are: the lateral acceleration response to steering wheel angle, the yaw velocity response to the steering wheel angle, the roll angle response to the steering angle and the roll angle response to the lateral acceleration.

\(^1\) W.J. Berkhout, *Design for a formula student race car*, 2004
Figure 3.10: Results steady-state circular test
3.4.7 Steady-state circular test

Test description
The idea behind this test is that the vehicle drives a circle with a fixed radius and a fixed steering angle. Each time the velocity increases, the lateral acceleration does too. This test is necessary for visualizing vehicle steady state cornering behaviour like under- and oversteer. Unlike other test, this test is simulated for several times with a slight change in forward velocity.

Two controllers are needed; one for the velocity, which is the same as in previous tests, and one for driving a circle with a constant radius. This controller defines the steering wheel input. When the simulation begins, the vehicle drives a straight line. This means that the vehicle is driving a circle with an infinite radius. For the program, infinite numbers are not controllable so the inverse of the corner radius is controlled: the curvature. This curvature, \( 1/R \), equals zero when R goes to infinite. The feedback controller is a proportional integrator with a curvature of 0.01 as reference signal, which means that the reference radius is 100 m.

Post processing
This test has very valuable information about understeer and oversteer. After each run, the last results are taken and filled in a column. These final results are taken from the runs, because it is for sure that the vehicle is driving a steady state circle at the end of the run. With these columns, containing all data of the runs, the important figures can be visualized with the post processor (opened by the command post_processing).

If you take a look at the results, you can see that all data after 110 km/h is cut off from the graphs. This has been done since the vehicle loses control at any higher speed. The vehicle suffers understeer (as you can see in figure 3.10) and when the vehicle reaches speeds above 110 km/h, the vehicle drives straight through and does not bend like it was supposed to do.

The understeer gradient \( k_u \) is given by the gradient shown in the graph for delta and the lateral acceleration. Only the first data is necessary for this gradient, so the data for the lateral acceleration less than 5 m/s\(^2\) is used. In the graph in figure (3.10) and given by MATLAB, \( k_u \) is 0.1016 deg\(^\circ\) s\(^2\)/m. This means the vehicle is strongly understeered.
Figure 3.11: Results bumpy road test
3.4.8 Bumpy road test

Test description
So far the road was assumed to be completely flat. Another test therefore is designed to see how the vehicle responds on a bumpy road. Is the vehicle comfortable or is it too sportive so the driver is tired after a few laps from absorbing the shocks that he has to endure?

In this test the geometry of the road surface is stored in a file and shown in figure (3.12). The road file contains data for 1200 meters and varies in height by approximately 12 centimetres. To simulate a bumpy road the options defined in the Delft-tyre blocks have to be adapted. Main point here is that the road defined is no longer suggested flat, but refers to a file called divine_roadx2.rdf.

Because of the adaptation in the Delft-tyre blocks, a new library file for in the simulink model has to be made. Otherwise the user has to change these settings each time he wants to run this test. This new library file is called FS_car_bump.mdl and the simulink file is called FS_model_bump.mdl.

With this new data the vehicle drives in a straight line over this defined bumpy road surface with the same settings as the original vehicle defined in FS_car.mdl. Finally it has to be mentioned that again a controller is used for maintaining velocity. It’s the same controller as defined in all previous tests.

Post processing
As you can see in the graphs of figure (3.11) wheels sometimes loses contact with the ground. This is visible in the graphic for $F_z$. Also it is clear that the car has a spring compression of about 2 cm. On this bumpy road the car doesn’t hit the ground; the ground clearance is always positive and about 5 – 10 cm. Also important to know are the roll angle and the pitch angle. They both are very reasonable: between -0.5 and 0.5 degrees for the roll angle and between -1.0 and 0.3 degrees for the pitch angle. The last graph of figure (3.11) is the same as shown in figure (3.12), but for the first meters only. The graphs for the ground clearance look a little smoother than the graph in figure (3.12), but are still very capricious. This means that the springs and dampers do give the driver some kind of comfort, but this comfort is not one of a high standard as by a passenger car.
Figure 3.13: Results separate suspension test
3.4.9 Separate suspension test

Test description
One of the tests in the testing program of vehicle_drivestests.m is the separate suspension test. This test separates the rear suspension and front suspension and simulates the suspension separately. The suspension models used are the same as modelled in the suspension systems of the car library files. The Delft-Tyre blocks for simulating the wheels are replaced by bodies, connected to the world with custom joints.

For testing the suspension, vertical displacements are placed on these joints with a joint actuator. Joint sensors are linked to the joints to record the desired data.

This test is for insight in the suspension behaviour only; it is not something you’ll see at the circuit to be tested. However these tests are of great importance to check for example the toe angle and camber angle. Further this test is the ideal test to see what adjustments in the suspension system would mean to the roll centre height, the lateral displacement and spring compressions.

Post processing
In figure (3.13) the results for this test are shown in different graphs. The graphs show the influences of the vertical displacement of the wheels to the camber angle, the toe angle, the road centre height, the spring compression and the lateral displacement. These are the results for the optimum setup which are finally user for the model and the simulations.
Figure 3.14: Results brake turn test
3.4.10 Brake turn test

Test description
To see what would happen if the car would get in a slip, a final test is invented. The vehicle is steered into a slip on purpose by starting the J turn manoeuvre and brake halfway the manoeuvre. The vehicle must stay controllable which means the vehicle is not allowed to get lots of oversteer so it would spin around. The worst case would be that the car will land on its side or turn upside down.

The vehicle starts driving in a straight line at 70 km/h. After 2 seconds the vehicle begins its J-turn, it bends to the right with a steering angle of 25 degrees. After 5 seconds from start the vehicle brakes, maximum locking the wheels. During these first 5 seconds of the test, the velocity is maintained by the already known speed controller. After those 5 seconds the throttle input becomes zero and the braking input becomes 1. After 7.9 seconds the steering angle becomes zero again.

Post processing
As shown in the graphs of figure (3.14) the lateral acceleration is a little less than one G and the velocity of the vehicle from the braking moment is zero again in 2.7 seconds. In the graph for $F_z$ you can see that the rear right wheel is lifted from the ground, because $F_z$ becomes negative after about 5 seconds. Three of the four wheels block after 5 seconds as shown in the graph for $\omega$. As a consequence of these wheel blocks the car slides in a straight line (shown in the graph for the centre of gravity).

The spring compression ranges from 0 cm to 1.3 cm for the rear suspension. It must be mentioned that 0 cm means that the spring is totally decompressed and the wheels are as far beneath the vehicle as possible. The roll velocity is high, about 10 degrees per second, but the roll angle is good; only 1.5 degrees.

The vehicle stays on the road, even at 70 km/h. One of the wheels did lift however, but that is not enough to make the vehicle uncontrollable. The vehicle didn’t turn upside down or fall at one of its sides, but stays on the ground so braking still can be applied. The vehicle is stopped in a short notice.

These results show that the vehicle does the same like every other car would do in this situation. Because three wheels of the car block, the car slides sideways. Therefore the yawrate decreases to zero again.
4. Virtual Reality Animation

4.1 Introduction

Now all results are known for the tests and all these data can be visualized with a couple of commands, it is interesting to see how the vehicle behaves in the simulation. To see this MATLAB 7.0.4 / Virtual Reality Toolbox and VREALM builder are used. In this chapter the built up of the animation programming will be discussed. After a while it will become clear that this part is built up practically the same as the simulations part.

After running the programs the user can see an animation of the test with the ‘vehicle’ as the main character. The virtual model is a simple representation of the vehicle, but at least it gives an impression of the movements of the car.

4.2 The VR animation program

Each test uses the same programs for showing the animations, except for test 9, the separate suspension test, because there is no VR simulation data available for this test. The user starts again with the already known command post_processing. By doing so, the program asks the user to make a choice between showing the test figures or showing the virtual reality animation. In chapter 3 the first choice is explained, now it is the turn for the second option: the virtual reality animation. One should not forget to close VR_sim.mdl before starting the postprocessor.

![Diagram of post processing built up]

When the user chooses the second option, post_processing.m refers to an m-file named post_processing_VRAnimations.m. The first thing this program wants to know is for which test the program has to make a virtual animation. The user choses between the different tests and after having chosen a test, the program loads the mat-data-file for that test.

In this model the virtual reality representation of the vehicle is made with help of the software VREALM builder, which is stored in a file with the .wrl extension. The vehicle is modelled as a block for the chassis and four cylinders for the wheels.
After running `VR_sim.mdl`, the program opens this simulink file automatically. The only thing the user has to do is to open the VR-sink block in `VR_sim.mdl` and click on play. The user than is able to see an animation of the simulation.

### 4.3 Example

If the data is available, the user has to check whether `VR_sim.mdl` is closed and types the command `post_processing` in the MATLAB Command Window. A menu is visible and the user has to choose the test he wants to perform. In this case choice 5 is made and the program runs the animation of the simulations. In the data file of the specific test, its data for the translation and rotation of the components which are shown in the animation are saved.

When the user opens the VR Sink Block in `VR_sim.mdl`, he sees the window as shown in figure (4.3). The user has to press the play button in the toolbar and the animation starts running. It seems that the car is standing still, but this is only because the cameras are coupled with the chassis of the car.

The car in the animation first bends to the right and after a couple of seconds the car bends to the left. After $t = t_{max}$ (15 seconds in this case) the animation is finished and the car stops.
4.4 Unfinished work

To give the user a better insight in the motions of the vehicle, a virtual reality animation is made by components from Unigraphics NX3.0 The car is built in Unigraphics and the different parts for the wheels and the chassis are first set as ‘Work part’ from the Assembly model and then exported as wrl-files. These wrl-files are used in the VREALM Builder Editor.

In the previous virtual animation the different parts were set as ‘Transform’ – ‘Children’ – ‘Shape’, these wrl-files are set as ‘Transform’ – ‘Children’ – ‘Inline’ – ‘url’. For the url the correct files are inserted in the program. By doing so, the user gets a more detailed and more realistic impression of the vehicle.

With this virtual reality animation the user will no longer see the simple block for the chassis and the simple cylinders for the wheels. Now the user has a visual of what the car looks like and how the wheels are really positioned regarding the chassis.

This work is still unfinished because the wheels do not turn about the right axis. They rotate over a big radius as shown in figure (4.4). However the chassis is built up in the same way and the chassis has a correct displacement and a correct rotation.
It should be mentioned that the rotation matrices and translation matrices used in this second virtual reality animation are exactly the same as the rotation matrices and translation matrices as the previous animation. The file VR_sim.mdl is therefore exactly the same as the previous one. Only the file FS_car.wrl is changed in the editor of VREALM Builder. Both the library files: FS_car.mdl and FS_car_bump.mdl haven’t changed, nor has any of the test specific simulink files.

To make this animation a correct view of reality, one should check what went wrong with the wheels in the modified virtual animation. When this problem is solved, it would be easy for the user to add some more components in the same way as already done.
5. Discussion

5.1 Conclusion

This report describes the analysis of the vehicle behaviour of the car of Formula Student Racing Team Eindhoven (FSRTE) for the competition of 2006. Since this car still does not exist, the behaviour has to be researched by means of simulation. Simulations further have the advantage that for making adjustments to the car, the effects of these adjustments can be examined without building a new car for each change in the setup.

The vehicles behaviour is examined by ten tests. Each test works the same; for each test a specific simulink model is built. In this simulink model a library file is fitted as a link. This library file is used for each simulink model, except for the bump test, which has an own library file and the separate suspension test, which does not use any library file. This library file gets its data from FS_modeldata.m. The test specific simulink file gets its data from the m-files for the tests, wherein the lookup tables are defined.

The tests have to be started by the command vehicledrivetest. When the program is run the results can be shown by the command post_processing. The post processor can make the test figures of the specific test or the post processor can make a virtual reality animation; a very simple animation showing the path of the car during the test.

The tests are designed to get a total overview of the vehicle behaviour. In the competition the car will have to perform several tests too. These tests are kept in mind by designing the tests for the simulation program. With these results it is possible to predict how the real car will respond in the competition.

Of course these simulations are not perfectly representing the reality, but they are a good approximation. The simulations for example are designed with a perfect CVT and an ideal differential. In reality the vehicle has a manual gearbox and a Torsen differential. Also the aerodynamics deserves a little more investigation. The main goal now was to make a good working simulation test program for the vehicle of 2006, which has been succesfull.

5.2 Recommendations

As said above, the simulations made by MATLAB 7.0.4 / SimMechanics are approximations of the reality and can certainly be improved. For further development of the vehicle simulation test program the following recommendations apply:

- The drive train of the vehicle is not represented as in reality. The drive train consists of a Continue Variable Transmission and an ideal differential.
- The exact design of the suspension box for the roll and bump stiffness should be checked for the vehicle of 2006. The real vehicle has a couple of pre-setup options that should be investigated to see the difference between them and to choose the optimal setup for the car.
- It might be a good idea to see what would happen if the gas tank is half empty and the gravity point of the car moves within the vehicle.
- The animations are kept very simple. It is possible to export wrl-files from Unigraphics and import them in VRealm Builder. This way it is possible to get a very detailed animation of the car.
References

3. Matlab version 7.0.4, Simulink version, SimMechanics version 2.2.2
4. Matlab version 7.0.4, Simulink version, Virtual Reality Toolbox
5. VREALM Builder version 2.0
# List of symbols

These symbols are used in the report:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P$</td>
<td>engine power</td>
<td>[W]</td>
</tr>
<tr>
<td>$T_{\text{engine}}$</td>
<td>engine torque</td>
<td>[Nm]</td>
</tr>
<tr>
<td>$\omega_{\text{engine}}$</td>
<td>engine angular velocity</td>
<td>[rad/s]</td>
</tr>
<tr>
<td>$V_x$</td>
<td>speed in local x-direction</td>
<td>[m/s]</td>
</tr>
<tr>
<td>$V_{\text{slip}}$</td>
<td>longitudinal slip speed</td>
<td>[m/s]</td>
</tr>
<tr>
<td>$V$</td>
<td>total velocity</td>
<td>[m/s]</td>
</tr>
<tr>
<td>$a_y$</td>
<td>lateral acceleration</td>
<td>[m/s²]</td>
</tr>
<tr>
<td>$R$</td>
<td>corner radius</td>
<td>[m]</td>
</tr>
<tr>
<td>$t_{\text{max}}$</td>
<td>end time of simulation</td>
<td>[s]</td>
</tr>
<tr>
<td>$\kappa$</td>
<td>longitudinal slip ratio</td>
<td>[-]</td>
</tr>
<tr>
<td>$\kappa_{\text{approx}}$</td>
<td>Approximation for longitudinal slip ratio</td>
<td>[-]</td>
</tr>
<tr>
<td>$\omega$</td>
<td>angular velocity of wheel</td>
<td>[rad/s]</td>
</tr>
<tr>
<td>$r_e$</td>
<td>effective rolling radius</td>
<td>[m]</td>
</tr>
<tr>
<td>$x$</td>
<td>force in x-direction applied by wheels</td>
<td>[N]</td>
</tr>
</tbody>
</table>
# APPENDICES

## A: List of available model output

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>s.FFLc2</td>
<td>globally defined forces on A-arm to upright connection [Front Left c2 position]</td>
<td>[N]</td>
</tr>
<tr>
<td>s.FFLc5</td>
<td>globally defined forces on A-arm to upright connection [Front Left c5 position]</td>
<td>[N]</td>
</tr>
<tr>
<td>s.FFRc2</td>
<td>globally defined forces on A-arm to upright connection [Front Right c2 position]</td>
<td>[N]</td>
</tr>
<tr>
<td>s.FFRc5</td>
<td>globally defined forces on A-arm to upright connection [Front Right c5 position]</td>
<td>[N]</td>
</tr>
<tr>
<td>s.FRLc2</td>
<td>globally defined forces on A-arm to upright connection [Rear Left c2 position]</td>
<td>[N]</td>
</tr>
<tr>
<td>s.FRLc5</td>
<td>globally defined forces on A-arm to upright connection [Rear Left c5 position]</td>
<td>[N]</td>
</tr>
<tr>
<td>s.FRRc2</td>
<td>globally defined forces on A-arm to upright connection [Rear Right c2 position]</td>
<td>[N]</td>
</tr>
<tr>
<td>s.FRRc5</td>
<td>globally defined forces on A-arm to upright connection [Rear Right c5 position]</td>
<td>[N]</td>
</tr>
<tr>
<td>s.Fdrag</td>
<td>drag force</td>
<td>[N]</td>
</tr>
<tr>
<td>s.Flift</td>
<td>lift force</td>
<td>[N]</td>
</tr>
<tr>
<td>s.Frotation</td>
<td>elements of the rotation matrix</td>
<td>[N]</td>
</tr>
<tr>
<td>s.Fsteerrackleft</td>
<td>force in steerrack left rod</td>
<td>[N]</td>
</tr>
<tr>
<td>s.Fsteerrackright</td>
<td>force in steerrack right rod</td>
<td>[N]</td>
</tr>
<tr>
<td>s.Fx</td>
<td>longitudinal tyre forces</td>
<td>[N]</td>
</tr>
<tr>
<td>s.Fy</td>
<td>lateral tyre forces</td>
<td>[N]</td>
</tr>
<tr>
<td>s.Fz</td>
<td>vertical tyre forces</td>
<td>[N]</td>
</tr>
<tr>
<td>s.Mb</td>
<td>brake moments applied on to the wheels [front left, front right, rear left, rear right]</td>
<td>[Nm]</td>
</tr>
<tr>
<td>s.Msteer</td>
<td>moment applied on to the steering wheel</td>
<td>[Nm]</td>
</tr>
<tr>
<td>s.Mx</td>
<td>overturning moment</td>
<td>[Nm]</td>
</tr>
<tr>
<td>s.My</td>
<td>rolling resistance moment</td>
<td>[Nm]</td>
</tr>
<tr>
<td>s.Mz</td>
<td>self aligning moment</td>
<td>[Nm]</td>
</tr>
<tr>
<td>s.V</td>
<td>total velocity of the vehicle</td>
<td>[m/s]</td>
</tr>
<tr>
<td>s.Vx</td>
<td>longitudinal velocity of the vehicle</td>
<td>[m/s]</td>
</tr>
<tr>
<td>s.Vy</td>
<td>lateral velocity of the vehicle</td>
<td>[m/s]</td>
</tr>
<tr>
<td>s.ax</td>
<td>longitudinal acceleration of the vehicle</td>
<td>[m/s²]</td>
</tr>
<tr>
<td>s.ay</td>
<td>lateral acceleration of the vehicle</td>
<td>[m/s²]</td>
</tr>
<tr>
<td>s.alpha</td>
<td>tire side slip angle</td>
<td>[rad]</td>
</tr>
<tr>
<td>s.beta</td>
<td>vehicle side slip angle</td>
<td>[rad]</td>
</tr>
<tr>
<td>s.gamma</td>
<td>inclination angle</td>
<td>[rad]</td>
</tr>
<tr>
<td>s.omega</td>
<td>rolling speed of tires</td>
<td>[rad/s]</td>
</tr>
<tr>
<td>s.ds</td>
<td>steering wheel angle</td>
<td>[rad]</td>
</tr>
<tr>
<td>s.brake</td>
<td>brake applied [0,1]</td>
<td>[-]</td>
</tr>
<tr>
<td>s.throttle</td>
<td>throttle input [0,1]</td>
<td>[-]</td>
</tr>
<tr>
<td>s.curvature</td>
<td>vehicle curvature</td>
<td>[1/m]</td>
</tr>
<tr>
<td>s.frontbump</td>
<td>front spring compression</td>
<td>[m]</td>
</tr>
<tr>
<td>s.frontroll</td>
<td>front roll angle</td>
<td>[rad]</td>
</tr>
<tr>
<td>s.rearbump</td>
<td>rear spring compression</td>
<td>[m]</td>
</tr>
<tr>
<td>s.rearroll</td>
<td>rear roll angle</td>
<td>[rad]</td>
</tr>
<tr>
<td>s.roll</td>
<td>vehicle roll angle</td>
<td>[rad]</td>
</tr>
<tr>
<td>s.rollvel</td>
<td>vehicle roll velocity</td>
<td>[rad/s]</td>
</tr>
<tr>
<td>s.pitch</td>
<td>pitch angle of the vehicle</td>
<td>[rad]</td>
</tr>
<tr>
<td>s.yawrate</td>
<td>yawrate of the vehicle</td>
<td>[rad/s]</td>
</tr>
<tr>
<td>s.groundclearancefront</td>
<td>global coordinates of point in front of car</td>
<td>[m]</td>
</tr>
<tr>
<td>s.groundclearancerear</td>
<td>global coordinates of point in rear of car</td>
<td>[m]</td>
</tr>
<tr>
<td>s.kappa</td>
<td>longitudinal slip ratio</td>
<td>[-]</td>
</tr>
<tr>
<td>s.kappa1</td>
<td>approximately longitudinal slip ratio</td>
<td>[-]</td>
</tr>
<tr>
<td>s.rackdispl</td>
<td>the steering rack displacement</td>
<td>[m]</td>
</tr>
<tr>
<td>s.time</td>
<td>simulation time</td>
<td>[s]</td>
</tr>
<tr>
<td>s.xcg</td>
<td>global x position center of gravity</td>
<td>[m]</td>
</tr>
<tr>
<td>s.ycg</td>
<td>global y position center of gravity</td>
<td>[m]</td>
</tr>
<tr>
<td>VR_output</td>
<td>array consisting all data for Virtual Reality Animation</td>
<td>[-]</td>
</tr>
</tbody>
</table>
**B: Example forcereport.out**

**Force report of standstill test:**

<table>
<thead>
<tr>
<th></th>
<th>Left c2</th>
<th>Left c5</th>
<th>Right c2</th>
<th>Right c5</th>
</tr>
</thead>
</table>

**Suspension Front:**

Highest positive force: [N]

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>659</td>
<td>23</td>
<td>689</td>
<td>378</td>
</tr>
<tr>
<td>y</td>
<td>659</td>
<td>23</td>
<td>689</td>
<td>105</td>
</tr>
<tr>
<td>z</td>
<td>-733</td>
<td>-710</td>
<td>2202</td>
<td>178</td>
</tr>
</tbody>
</table>

Highest negative force: [N]

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>-733</td>
<td>-710</td>
<td>2202</td>
<td>-402</td>
</tr>
<tr>
<td>y</td>
<td>-610</td>
<td>-1081</td>
<td>1035</td>
<td>129</td>
</tr>
<tr>
<td>z</td>
<td>-733</td>
<td>-710</td>
<td>2202</td>
<td>178</td>
</tr>
</tbody>
</table>

**Suspension Rear:**

Highest positive force: [N]

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>64</td>
<td>-648</td>
<td>289</td>
<td>64</td>
</tr>
<tr>
<td>y</td>
<td>5</td>
<td>-26</td>
<td>1</td>
<td>30</td>
</tr>
<tr>
<td>z</td>
<td>2</td>
<td>-1691</td>
<td>833</td>
<td>13</td>
</tr>
</tbody>
</table>

Highest negative force: [N]

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>-45</td>
<td>-497</td>
<td>398</td>
<td>-43</td>
</tr>
<tr>
<td>y</td>
<td>11</td>
<td>-1705</td>
<td>781</td>
<td>-10</td>
</tr>
<tr>
<td>z</td>
<td>2</td>
<td>-1691</td>
<td>833</td>
<td>13</td>
</tr>
</tbody>
</table>
## C: Description of data

<table>
<thead>
<tr>
<th>Data</th>
<th>Description</th>
<th>Value</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>simulation time</td>
<td>[s]</td>
<td></td>
</tr>
<tr>
<td>Tmax</td>
<td>maximum simulation time</td>
<td>[s]</td>
<td></td>
</tr>
<tr>
<td>steer_input</td>
<td>lookup table input for steering wheel</td>
<td>[rad]</td>
<td>positive means steering to the right, negative means steering to the left</td>
</tr>
<tr>
<td>throttle_input</td>
<td>lookup table input for throttle</td>
<td>[-]</td>
<td>ranged between -0.2 (engine braking) and 1 (full throttle)</td>
</tr>
<tr>
<td>brake_input</td>
<td>lookup table input for braking</td>
<td>[-]</td>
<td>ranged between 0 (no braking) and 1 (full braking)</td>
</tr>
<tr>
<td>Vx_init</td>
<td>initial velocity</td>
<td>[m/s]</td>
<td></td>
</tr>
<tr>
<td>V_input</td>
<td>lookup table input for velocity</td>
<td>[m/s]</td>
<td>mostly used for feedback control for the throttle input</td>
</tr>
<tr>
<td>Fx</td>
<td>longitudinal tyre forces</td>
<td>[N]</td>
<td>positive means driving forwards, negative means driving backwards</td>
</tr>
<tr>
<td>Fy</td>
<td>lateral tyre forces</td>
<td>[N]</td>
<td>positive means forces to the right, negative means forces to the left</td>
</tr>
<tr>
<td>Fz</td>
<td>vertical tyre forces</td>
<td>[N]</td>
<td>positive means the wheel is pressed on the ground, negative means lifting of the wheel</td>
</tr>
<tr>
<td>Ax</td>
<td>longitudinal acceleration of the vehicle</td>
<td>[m/s²]</td>
<td>positive means acceleration forwards, negative means acceleration backwards</td>
</tr>
<tr>
<td>Ay</td>
<td>lateral acceleration of the vehicle</td>
<td>[m/s²]</td>
<td>positive means acceleration to the right, negative means acceleration to the left</td>
</tr>
<tr>
<td>roll angle</td>
<td>body roll angle</td>
<td>[deg]</td>
<td>positive means body roll to the right = bending to the left, negative means body roll to the left = bending to the right</td>
</tr>
<tr>
<td>pitch angle</td>
<td>body pitch angle</td>
<td>[deg]</td>
<td>&gt; -0.7 degrees means leaning to the back, &lt;-0.7 degrees means leaning to the front</td>
</tr>
<tr>
<td>yawrate</td>
<td>Yawrate of the vehicle</td>
<td>[deg/s]</td>
<td>positive means yaw to the right, negative means yaw to the left</td>
</tr>
<tr>
<td>K</td>
<td>wheel spin ratio</td>
<td>[-]</td>
<td>&gt; -1 means wheel is slower than road surface, &lt; -1 means wheel is faster than road surface, =1 means no slip</td>
</tr>
</tbody>
</table>
DROPTEST:
initial drop height: 0.00 [m]
minimum groundclearance front: 0.10 [m]
maximum groundclearance front: 0.14 [m]
minimum groundclearance rear: 0.08 [m]
maximum groundclearance rear: 0.10 [m]
standstill groundclearance front: 0.11 [m]
standstill groundclearance rear: 0.09 [m]

ACCELERATION TEST:
Max speed reached in test: 219.75[km/h] dis: 882.50[m] time: 20.00[s]
100 km/h reached in test: 100.00[km/h] dis: 50.13[m] time: 3.57[s]
75 meters reached in test: 116.50[km/h] dis: 75.00[m] time: 4.40[s]

BRAKE TEST:
Speed when braking applied: 54.00 [km/h]
Brake distance: 11.27 [m]
Brake time: 1.65 [s]
E: Force definition

To be able to find the highest force experienced during the tests, in post_processing_figures2.m all the forces from each test are stored in the same matrix. These forces are collected with joint sensors on part of the suspension to upright connections during tests. Like mentioned, in post_processing_figures2.m all the forces from the different tests are stored in one matrix.

Extra columns are added, so it is possible to determine later in which test the maximal forces occurred. After all tests are run with post_processing_figures2.m, and obvious all forces from the different connection points and tests are stored in the matrices, post_processing_figures.m filters out the maximum situations occurred.

The procedure used here is as follows:

• The collected vector data from the forces stored in matrices is made to one value with
  \[ F_{tot} = \sqrt{x^2 + y^2 + z^2} \]  \hspace{1cm} (E.1)

• The highest vector is searched with the MATLAB commands 'find' and 'max' which returns the corresponding element number.

• The elements of this force vector, \(F_x\), \(F_y\) and \(F_z\) are taken.

• These forces acting on the connection points are defined in the global coordinate system of the simulation. The \(x\), \(y\) and \(z\) direction are meaningless if isn't defined how the vehicle is positioned.

• To give sense to these values they have to be transformed to the corresponding forces defined in the local coordinate system of the vehicle itself. For this purpose there is a body sensor connected to the vehicle body in the car library. This stores the rotation matrices for each moment in time. These rotation matrices are processed on the same way as the forces are in post_processing_figures2.m so there's also an matrix with all the different rotation matrices from the different tests stored in.

• A function file is made: fun_rotationmatrix.m. In post_processing_figures.m the element number found is filled in this function. This function searches the corresponding elements from the matrix where all rotation matrix elements are stored in. With these elements it assembles the corresponding rotation matrix and returns this as an output to post_processing_figures.m

• With use of this assembled rotation matrix the original global force vector can be transformed with formula (E.2):
  \[ F_{local} = M_{rot}^T \cdot F_{global} \]  \hspace{1cm} (E.2)

• The force vector elements are defined in the local coordinate system of the vehicle (see figure (2.2)) and useful for design considerations. The maximal force vectors for the different connection points of the suspensions are stored in a report named: forcereport.out.

\(^1\) Forces not available for the separate suspension test
F: Explanation of variables for $\kappa$

nomenclature:
- free tyre radius $r_f$
- effective rolling radius $r_e$
- loaded radius $r$, tyre deflection $\rho$
- forward velocity $V_x$
- wheel angular velocity $\Omega$
- longitudinal slip speed $V_{sl}$
- longitudinal force $F_x$
- vertical force $F_z$
- rolling resistance moment $M_f$

"S" is the pole of the free rolling tyre

Figure H.1: Definitions of longitudinal slip ratio $\kappa$

Of course the values for $\kappa$ are different for each kind of car and tyre. In the figure below the longitudinal force with respect to a negative $\kappa$ (braking) is visible for a normal passenger car. For acceleration the figure has to be mirrored in the point $(0,0)$, but remains the same. The curve has an optimum. Even though each tyre has it own characteristic $\kappa$ they all have such an optimum where the largest force forward $F_x$ can be found for a specified $\kappa$.

Figure H.2: Force against the longitudinal slip ratio $k$