Formula student vehicle analysis by means of simulation

R. van der Aalst 0532289
2005.146
Supervisor: Dr. Ir. I. Besselink

December 6, 2005
Contents

1 Introduction 2

2 The vehicle model 3
   2.1 Introduction .................................................. 3
   2.2 The model ..................................................... 3
   2.3 Model parameters ............................................. 4
   2.4 A simple simulation ......................................... 5

3 The test program 8
   3.1 Introduction .................................................. 8
   3.2 Tests program built up .................................... 8
   3.3 Post processing ............................................... 9
   3.4 The tests ...................................................... 11
      3.4.1 Static equilibrium position test ................... 11
      3.4.2 Suspension kinematics simulation ................... 12
      3.4.3 Steady-state circular test ......................... 13
      3.4.4 Driving on a bumpy road ............................ 14
      3.4.5 Acceleration test ...................................... 15
      3.4.6 Brake test .............................................. 16
      3.4.7 J-turn manoeuvre (step steer) ...................... 17
      3.4.8 Fishhook manoeuvre ................................... 18
      3.4.9 Slalom test .............................................. 19

4 Comparison of two vehicle variants 20
   4.1 Introduction .................................................. 20
   4.2 Results ....................................................... 20

5 Conclusion 21
   5.1 Conclusion ................................................... 21
   5.2 Recommendations ............................................ 21

Bibliography 21

A List of symbols 23

B List of available model output 24

C Example forcereport.out 26

D Example testresults.out 27

E Force definition 28
Chapter 1

Introduction

Since 2003 the Technical University of Eindhoven (TUE) has his own Formula Student Racing Team (FSRTE). This is a competition wherein engineering students design and build their own race car. The main core of the competition is engineering instead of real racing. Analysis and good funded design choices are of great importance to win.

At this moment there isn’t an existing FS race car yet so real tests can’t be done on a car. But for an optimal design there exists the need of some data to get insight in vehicle behaviour. For that purpose a car simulation model is used.

This report first gives some insight in how the simulation model is built up [Chapter 2]. However most attention will be put on setting up a test program to perform standardized tests with different car designs. The total test program contains nine different vehicle tests. All the tests will be treated separately with a short explanation of test specific things and data processing [Chapter 3]. Finally an example will be given of a real design problem, which is solved with use of the simulation test program [Chapter 4].
Chapter 2

The vehicle model

2.1 Introduction

For the optimisation of the design of the Formula Student racing car a MATLAB (v.7.0.4)/SimMechanics (v.2.2.2) model has been developed. In this model adjustments on the design or vehicle setup can be made easily. The advantage of such a model is that the effects of these adjustments become clear after analyzing simulations results. In this chapter the model, and usage of it, is explained globally.

2.2 The model

Bram de jong, Bas de Waal and Dr. Ir. Igo Besselink are mainly responsible for making the current model.

First of all it should be mentioned that the model is stored in a Simulink library file, FS-car.mdl. The reason for this will become clear in chapter 3. The model is built up in four subsystems. These subsystems are the front and rear suspension, the engine and differential together and the chassis.

In case of the chassis it is important to realize that it is modeled rigid. Furthermore the car is built up with a correct use of various joints so that it is possible for the car to experience movements like body roll or pitch. Another element in the chassis subsystem is the aerodynamics block. This block implements an approximation of air resistance and lift or downforce. This is especially important if the vehicle drives with higher speeds. Without it, the car would reach impossible high speeds. On the other hand, for simulation it is not so very important because in the formula student competition the car will probably not reach speeds over 110 km/hour so that aerodynamics can almost be neglected in comparison to other factors.

The suspension systems represent the double-wishbone configuration and the parts responsible for providing bump and roll stiffness. These parts are modeled according to the design of Wouter Berkhout. Also a steering rack in combination with a steer rod upright connection can be found in the front suspension system. With steer input for this steering rack the steering ratio between steering wheel and wheels is taken into account. There is also a drive shaft placed in the rear suspension system, responsible for putting a drive moment on the wheels. Further also braking is included in these suspension systems.

Furthermore a system representing the engine and differential is present. In this block the torque that works on the left and right drive shaft is calculated. The principle used here is:

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

so that

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

Formula (2.2) 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 modeled like a cvt. In practice this is not the case but for simulation it is sufficient. Therefore a saturation is made to limit the maximal 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.
The last thing that is worth mentioning here are the Delft-Tyre blocks. They are a representation of the tyre on road behaviour as defined by the Magic Formula tyre model.

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. For the purpose of assigning these value’s there’s a m-file made, *FS-modeldata.m*. In the base model there’s made use of parameters that refer to the names, with added value, of the m-file.

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>c1</td>
<td>connection upper A-arm - chassis</td>
<td></td>
</tr>
<tr>
<td>c2</td>
<td>connection upper A-arm - upright</td>
<td>c8</td>
</tr>
<tr>
<td>c3</td>
<td>connection upper A-arm - chassis</td>
<td>c9</td>
</tr>
<tr>
<td>c4</td>
<td>connection lower A-arm - chassis</td>
<td>c10</td>
</tr>
<tr>
<td>c5</td>
<td>connection lower A-arm - upright</td>
<td>c11</td>
</tr>
<tr>
<td>c6</td>
<td>connection lower A-arm - chassis</td>
<td>c12</td>
</tr>
<tr>
<td>c7</td>
<td>connection upright - wheel center</td>
<td>c13</td>
</tr>
<tr>
<td>c14</td>
<td>connection rocker - rollbar rod</td>
<td></td>
</tr>
</tbody>
</table>

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. Only for the left side of the vehicle this has to be done because in the model it is mirrored for the right side. The numbering that is used in *FS-modeldata.m* for giving in the coordinates of specific suspension points corresponds with the numbering as defined in figure (2.1). The coordinate system is always defined as figure (2.2). In the rear suspension the steering rod is replaced with a tie rod. The numbering corresponding to the connection points of the steering rod in the front suspension apply for the tie rod connections in the rear suspension.
Furthermore the spring and damper coefficients have to be defined. Here must be noted that with the bump offset is meant the length of what the spring has to be compressed when connected between the rockers\(^1\). This holds that a positive number here increases ride height.

A last remark that has to be made is about the ground clearance definition that can be found in \textit{FS-modeldata.m}. Here the user gives in coordinates of two points as desired from which the distance to the ground is measured and given as output in the result of simulations.

Furthermore always an initial velocity and simulation time 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 \(V_{x, \text{init}}\) and the simulation time \(t_{\text{max}}\).

Beside the \textit{FSmodeldata.m} file and a m-file that starts the simulation there are two other files with parameters; \textit{FSfronttyre.tpf} and \textit{FSreartyre.tpf}. In these files properties like tyre dimensions, tyre mass, tyre moment of inertia etc. are defined.

### 2.4 A simple simulation

Now the vehicle model definition is complete with the library file \textit{FS-model.mdl} and the parameter definitions in \textit{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 minus \(\pi\) to \(\pi\).

As an example lets do 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 there’s made a simulink file like figure (2.3) where the library file is copied in as a link.

\begin{itemize}
  \item \textbf{Throttle}
  \item \textbf{Steering angle}
  \item \textbf{Brake}
  \item \textbf{s_time}
\end{itemize}

**Figure 2.3**: Simulink file from the simple example with the car library link placed in.

\(^1\)With rockers are meant parts that translate a direction of motion by toppling over around a defined point. In figure (2.1) rockers are visible. The coordinate numbers \(c_{11}, c_{12}, c_{13}\) and \(c_{14}\) define how the motion is translated specifically by the rocker.
Then a m-file is made wherein a few things need to be defined. One of these things are the values for the look-up tables that serve as inputs for the vehicle. The script then looks like this:

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

The look-up tables then become like figure (2.4):

![Look-up tables](image)

Figure 2.4: Look-up tables that serve as input

While running the simulation data is stored in an array `s`. For example the vehicle speed data can be used with `s.V`. A list of available output can be found in appendix B.

For the example presented here results are found in figure (2.5).
Figure 2.5: Results from example simulation
Chapter 3

The test program

3.1 Introduction

As mentioned before, the vehicle model is necessary to be able to get insight in the vehicle behaviour of a potential FS car and to optimise this vehicle behaviour. To make the comparison of different vehicle settings easy and objective, a standardised vehicle testing program is made containing nine different tests. The tests are:

1. static equilibrium position test
2. suspension kinematics simulation
3. steady state circular test
4. driving on a bumpy road
5. acceleration test
6. (emergency) brake test
7. J-turn manoeuvre
8. fishhook manoeuvre
9. slalom test

In this chapter we will discuss the different tests.

3.2 Tests program built up

Each test has the same file structure and gives the same output. One simulation test consists of six main file parts:

- the car library file, denoted by FS-car.mdl.
- the vehicle parameter file FS-modeldata.m which assigns specific values to model parameters in the car library file as described in paragraph (2.3).
- the tyre files FS-fronttyre.tpf and FS-reartyre.tpf.
- a test specific simulink model file where the library car model is placed in. It contains three blocks that define the throttle, brake and steer input respectively. Further controllers are added to this simulink model if necessary for that specific test.
- the m-file with specific test values. It defines values as $V_{xinit}$, $t_{max}$, the input blocks etc, runs FS-modeldata.m, sometimes redefines values of it and also runs the simulink file with the car library model inside.
- the file vehicle_drivetests.m

1 The bumpy road test and suspension kinematics test are an exception on this cause they slightly differ in program files and output. See paragraphs (3.4.4) and (3.4.2) for more details.
2 The bumpy road test has is own library file FS-car-bump.mdl
The first three files were already presented in chapter 2. It will be clear to the reader that these three files together define the total vehicle. So if different car designs need to be compared, then adjustments have to be made in these files. These files will be kept constant during the test program so that each test is performed with the same vehicle configuration. To be able to do this for each separate test, it has been decided to model the vehicle as a Simulink library file. Now it is possible to put a link to this library model file in each Simulink test model. There is no need to change each test separately when a different car setup needs to be tested. Only changing the car library and `FS-modeldata.m` will do for all the tests in the test program.

The fourth and fifth files together define the specific test. What lasts is the m-file called `vehicledrivetest.m`. When running this file the user needs to define which tests from the choice of nine he wants to perform. After this input is given the test program will run autonomously till all the prescribed tests are performed.

A schematical representation of how the files are related to each other can be found in figure (3.1).

![Schematical representation of underlying program file relations](image)

**Figure 3.1:** Schematical representation of underlying program file relations

### 3.3 Post processing

For processing simulation results some standarized methods are made available. This consists of two m-files named `post_processing1.m` and `post_processing2.m`. With running the first the user gets the question which data sets (from which tests) he wants to process. Dependent on what the user defines the file `post_processing1.m` uses parts of the function `post_processing2.m`. In this last function file the real postprocessing is done. It loads the specific test data and post processes it to useable results. One of these results are test specific graphs stored in `testfigures.ps`. Further some
characteristic vehicle values are calculated and stored in \textit{testresults.out}. Examples of such values are the time to reach 100 km/hour for the acceleration test or the time necessary to come to stand still in the emergency brake test. It must be noted that these values can differ from reality because of approximations done while creating the simulation model. Though, they give good indications for comparison between different vehicle set ups (an example of a \textit{testresults.out} file can be found in appendix D). Another important result of the post processing are the maximum forces on the suspensions occurred during the tests. These forces, for all the suspension connection points, are stored in the file \textit{forcereport.out} (an example of such a file can be found in appendix C). How these forces are defined can be found in appendix E.

These processing and result files are related as shown in figure 3.2.

Figure 3.2: Schematical representation of post processing program file relations
Figure 3.3: Static equilibrium position test
3.4 The tests

3.4.1 Static equilibrium position test

Results are shown in figure (3.3).

modeling

In this test the static vehicle position is calculated. In FS-model-statictest.mdl all the input to the car library is zero. This is defined in statictest.m.

Furthermore the option of giving the car an initial vertical displacement is available. For this option the initial condition of the model is used and change the starting height of the vehicle. This initial height can be adjusted by changing the value of d.initialstartingheight in statictest.m.

post processing

In static equilibrium of the vehicle, values as pitch angle, roll angle, spring compressions and vertical tyre forces are interesting. These values become available, after post processing, in the several output files.
Figure 3.4: Results suspension kinematics simulation
3.4.2 Suspension kinematics simulation

Results are shown in figure (3.4).

modeling

Another test that is available in the test program is for testing the front and rear suspension independently. The suspension models used for the front suspension and rear suspension are the same as the suspension systems in the previous used libraries. The Delft Tyre blocks for simulation the wheels are now replaced with bodies. These bodies are connected to the world with custom joints. On these joints a vertical displacement is placed with a joint actuator, simulating that the tires experience bump. Further joint sensors are linked to these custom joints which receive data.

post processing

With the data collected by the joint sensors the change of camber angle, toe angle, vertical stiffness and roll center height as a reaction to bump and rebound is available. These changes are presented in graphs to get a good overview of how the suspensions reacts to bump and rebound.
Figure 3.5: Results steady-state circular test
3.4.3 Steady-state circular test

Results are shown in figure (3.5).

modeling

First of all it must be mentioned here that the structure of this test differs from the previous ones. In this test the same simulation is run several times with a slight change in forward velocity. The idea behind this test is that the vehicle drives a circle with a fixed radius and fixed steering angle with a certain lateral acceleration. The vehicle performs the same manoeuvre multiple times only with a higher lateral acceleration. With the information gained, graphs can be made that visualize vehicle behaviour like under- or oversteer.

To be able to let the vehicle simulation drive a steady state circle two controllers are needed. The first controller is the controller that is responsible for driving a circle with a steady radius. It defines the steering wheel input. When the simulation begins the vehicle drives a straight line which would mean an infinite radius. This isn’t practical with calculations while simulating and because of that it has been decided to control on the inverse, the curvature. This curvature, $\frac{1}{R}$, equals zero when $R$ goes to infinite. The feedback controller is a proportional integrator with a curvature of 0.01 as reference signal.

The forward velocity also needs to be controlled. This cruise controller is also a simple PI feedback controller with as reference signal the desirable forward velocity and as feedback the forward velocity from the model output (figure (3.6)). The forward velocity for the reference signal used here is calculated as follows. First in `steadystatecirculartest.m` the desirable lateral accelerations are defined. Then with a corner radius $R$ of 100 meters the corresponding forward velocities can be calculated with:

$$V = \sqrt{a_y \cdot R}.$$  \hspace{1cm} (3.1)

For each different forward velocity $V$ the test has to be performed.

![Figure 3.6: Schematical representation of the cruise controller](image)

post processing

Like mentioned before, the steady state circular test consists of simulations with different lateral accelerations while driving the circle. With data processing, in the file `dataprocessing2.m`, from each individual simulation the last value from a specific parameter is taken and put in to one column. The last element is taken because on that moment the vehicle is driving under steady state conditions. With the data stored like this valuable figures can be made which give information on vehicle behaviour like understeer or oversteer.
Figure 3.7: Driving over a bumpy road
3.4.4 Driving on a bumpy road

Results are shown in figure (3.7).

modeling

Here the vehicle is driven over a bumpy road to see how it responds with respect to comfort and loss of road contact. To make this simulation possible it has, in contrary to the other tests, its own specific car library file: `FS-car-bump.mdl`. The necessity of another library file is because of the essential difference that has to be made in the file. Namely 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 in these blocks now refers to a file called `divine-roadx2.rdf`. This is a file where measurement data from a real road is stored in (figure 3.8). Now this data is used as the road where the vehicle drives on. The vehicle drives a simple straight end over this defined road.

![divine roadx2.rdf](image)

**Figure 3.8:** A graphical representation of the road data.

post processing

In this case there’s chosen to add figures with information over the vertical forces, vertical accelerations, vehicle angles (roll/pitch) and spring compressions to the file `testfigure.ps`. This should give a good view of vehicle response on driving over bumps. For clarity an algorithm is placed in `dataprocessing.m` that is responsible for adding a plot where comes visible over what part of the predefined road is driven exactly on which moment in time.

---

3 For the road data used during tests it doesn’t matter which manoeuvres the vehicle performs because there’s only looked at the traveled distance of the tires.

4 For this option it is necessary that the file `roaddata.mat` is stored in the same directory.
Figure 3.9: Acceleration test
### 3.4.5 Acceleration test

Results are shown in figure (3.9).

**modeling**

Here the car gains speed with maximal acceleration. Because giving in full throttle results in a car with wheel spin (what isn’t preferable for maximal acceleration) some controlling has to be put in `FS-model-accelerationtest.mdl`.

The parameter called $\kappa$ defines the longitudinal slip ratio of a tyre [1]:

$$\kappa = -\frac{V_{sx}}{V_x} = -\frac{V_x - \Omega \cdot r_e}{V_x}$$  \hspace{1cm} (3.2)

Definitions of used variables can be found in figure (3.10).

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

“S” is the pole of the free rolling tyre

**Figure 3.10: Definition of longitudinal slip ratio Kappa**

This parameter is used for controlling the throttle input with respect to a suitable slip.

In figure (3.11)[1] the longitudinal force with respect to a negative $\kappa$ (braking) is visible. For acceleration, so a positive $\kappa$, this figure can be mirrored in the origin. Then becomes clear that the largest force $F_x$ can be generated with a $\kappa$ around 0.18.

In the model the wheels are modeled with use of Delft-Tyre blocks. One of the outputs of such blocks is the $\kappa$ parameter. A disadvantage of this $\kappa$ is that when $V_x$ is zero there is divided by zero, what obviously isn’t possible.

To deal with this problem a new $\kappa$ is calculated with the suitable approximation:

$$\kappa_{approx} = -\frac{V_{sx}}{|V_x| + 1}$$  \hspace{1cm} (3.3)

This $\kappa_{approx}$ instead can be used to control with, even at low kappa’s close to zero. The car library file gives this $\kappa_{approx}$ as an direct feedback output. In `FS-model-acceleration.mdl` this $\kappa_{approx}$ output is connected to a lookup table that’s connected to the throttle input again. After four seconds, when the car stands totally still, the acceleration procedure begins. The look-up table is of such a shape that it gives in full throttle till $\kappa_{approx}$ is above 0.2. After that the throttle decreases to avoid to much spin and resulting $F_x$ decrease. With this an optimum for the throttle input (and following acceleration) isn’t reached but it gives a better approximation then full throttle at once.

**post processing**

The data post processing gives as a results figures and important values. The important output values are the maximal speed reached in the test, the time to reach 100 km/hour and the time to reach 75 meters. This last value is of importance because it is one of the tests that are a part of the Formula Student competition.
Figure 3.12: (Emergency) brake test
3.4.6 Brake test

Results are shown in figure (3.12).

modeling

This test simulates an emergency stop with maximal deceleration so speed is rapidly decreased to zero. It has the same problem as the acceleration test. That is, to reach a maximal deceleration it is not suitable to apply full brake, which would cause the wheels to lock. Instead there is a maximum $-F_x$ around a certain $\kappa$ again. A brake controller is made, similar to the one used in the acceleration test, in \textit{FS-model-braketest.mdl}. The only difference with the throttle controller of the acceleration test are the values used in the look-up table. In this case it applies full brake (\(\text{= value 1}\)) when $\kappa_{\text{approx}}$ is more than -0.01. Below this value the brake input decreases till it becomes zero at $\kappa_{\text{approx}} = -0.12$. Like the acceleration controller this is also an approximation of maximal braking but it gives better results than applying full brake though. Furthermore the cruise controller is implemented again which is responsible for maintaining constant speed until the moment of braking.

post processing

Here the values that give insight in the vehicle performance are the brake distance and the braking time. For clarification the speed when braking is applied is also stored. Further the model has the ground clearance as output. This output shows the space between the ground and two defined points from the car during tests. It can be used to check if the vehicle hits the ground as a result of an increased pitch angle during brake.

One of the figures stored in \textit{testfigures.ps} is the ground clearance of these points to the time. In the simulation it is possible to get a negative ground clearance output, in practice however this isn’t possible and the defined point of the car will touch the road surface ($s.ground\text{clearance} \leq 0$).
Figure 3.13: J-turn manoeuvre
3.4.7 J-turn manoeuvre (step steer)

Results are shown in figure (3.13).

**modeling**

In case of the J-turn manoeuvre the idea of the test is to see how the car reacts to an immediate (relatively big) steering angle applied (step steer). For this purpose a steer input of 50 degrees is given while driving with a speed of 50 km/hour. Also here is made use of the cruise controller like explained before in paragraph (3.4.3) to maintain speed.

**post processing**

One thing that is of interest in the case of hard steering is wheel lift. Are there wheels that loose contact with the ground? The answer to that question can be found in the figure $F_z$ against time which is one of the figures stored in *testfigures.ps*. In this figure the vertical force on all the tires is visible. When one of the wheels loses contact with the ground it won’t have any vertical force so $F_z$ will be zero. Another interesting figure is the roll against time figure where we can see how the orientation of the car is during hard cornering. Furthermore the figure of ground clearance is stored because this can be worth full in looking at the jacking effect while cornering. The ground clearance will enlarge when the jacking effect occurs.
Figure 3.14: Fishhook manoeuvre
3.4.8 Fishhook manoeuvre

Results are shown in figure (3.14).

modeling

With a fish-hook manoeuvre the car first makes a turn and on the moment that the car totally leans over to one side it gets a hard steering angle in the opposite direction [3]. The point of interest here is when the car makes the change in cornering from one direction to the opposite. In that case the car rolls over to the other side and is thereby helped with the energy stored in the springs. This results in a greater roll velocity and because of that extra energy that helps rolling, the car has more chance to lose contact with the ground on the inner side or even tip over.

The best moment to give in the great change in steering wheel angle would be when the roll velocity turns sign. This is the moment when the car has reached full roll and leans totally in his springs. A controller that uses this knowledge couldn’t be made because the models roll velocity turns sign continuously. To do a vehicle rollover whereby the springs help, another solution is chosen. It gives in a steering wheel input, very slow enlarging so that the car can roll over more and more in its springs and then suddenly an opposite steering wheel angle is applied. For comparison of different car setups it doesn’t matter if the car isn’t turned over on the right moment exactly. Same as with the brake test and J-turn manoeuvre also here the cruise controller is implemented.

post processing

This is almost similar to the J-turn manoeuvre. Likewise the $F_z$-time figure is of great value in detecting wheel lift. Also the roll-time figure gives good insight again of the chassis orientation.
3.4.9 Slalom test modeling

In this test the vehicle response to different frequencies of steering input is considered. Therefore in `FS-model-slalomtest.mdl` there’s placed a chirp signal on to the steering input port. This is a signal with a begin frequency that increases till a defined frequency is reached on a certain predefined moment in time. In `slalomtest.m` these specific variables get a value assigned. The begin frequency from where the chirp signal starts is named `c.initialfrequency`. The end frequency is defined as `c.endfrequency` and the moment in time where this frequency has to be reached is named as `c.targettime`. Now the chirp signal can be totally defined to the users’ wishes. There’s estimated that the FS vehicle has a yaw frequency of 8.8 Hz [4]. A high yaw frequency will mean that the switch between a left and a right bend can be made very quickly. For the simulation experiment there’s chosen for an reasonable frequency range from 0 to 11 Hz as a steer input so that the yaw frequency should become visible.

Post processing

With such a test it is of importance how the vehicle responds to the steer input. To get insight in that matter there’s the need of analyzing transfer functions. These transfer functions are made visible in bode plots like for example figure (3.15). To calculate these transfer functions there’s made use of the Matlab command ‘tfe’ in combination with ‘cohere’ for checking the coherence. The transfer functions that are calculated like this are: the lateral acceleration response to steering wheel angle, the yaw velocity response to steering wheel angle, the roll angle response to steering wheel angle and the roll angle response to the lateral acceleration.

![Bode plots](image)

**Figure 3.15: Example of slalom test result**

---

5Running this test takes a few hours so for quick results it is advised to leave this test out.
Suspension test: toe characteristics

<− rebound        vertical displacement [m]        bump−>
<− toe−in        toe angle [deg.]        toe−out−>

Figure 3.16: Toe change of design 1

Figure 3.17: Toe change of design 2

Figure 3.18: Forces experienced during driving over a bumpy road, design 1

Figure 3.19: Forces experienced during driving over a bumpy road, design 2
Chapter 4

Comparison of two vehicle variants

4.1 Introduction

Let’s give an example of how the test program can be used. For this a design issue from reality is compared. The designs compared differ in suspension geometry and spring/damper constants. This is defined in FS-modeldata.m. The spring/damper constants of design one are approximately twice the size of design two.

All the tests in the test program are performed with use of vehicletestprogram.m.

4.2 Results

To get an overview of the results from the test simulations done with vehicletestprogram.m dataprocessing.m is used. Here appear some interesting figures.

One of the conclusions that can be drawn with respect to the results is that design two reacts much smoother to movements than design one.

The most important aspect that results is that in design one almost three degrees of toe change appears on the rear wheels when they experience a vertical displacement, see figures (3.16) and (3.17). This effect is called bumpsteer. Three degrees of bumpsteer is very much. Design two for example has maximal 0.35 degrees of bumpsteer. This is preferable of course.

Another result is that when looked at the forces experienced during driving over a bumpy road, that design two has much less jumping than design one, see figures (3.18) and (3.19). Design one even loses contact with the ground now and then. Design two is obviously much better to handle while driving over a bumpy road.

So with use of the simulation test program a comparison can be made and decided that design two is better then design one.
Chapter 5

Conclusion

5.1 Conclusion

In this report the Formula Student vehicle model for performing test simulations is explained. The vehicle model setup can easily be adjusted to new designs with use of m-files and a Simulink library file. The model is a good approximation of reality and all important aspects for vehicle behaviour, like for example bump and roll stiffness, are included.

The test program contains nine common vehicle tests. They are usable to get good insight in both the static and dynamical vehicle behaviour. The tests that the vehicle has to do in the Formula Student competition are always kept in mind with the vehicle simulation tests and post processing characteristic results.

The test program is clearly a first version. Though it gives enough results as an output to compare different vehicle designs with, but for getting reliable characteristic values like the time to reach 100 km/h it isn’t good enough.

5.2 Recommendations

Like said above the test program is a first version wherefrom the model and tests can certainly be improved. For further development of the vehicle simulation test program the following recommendations apply:

- In the car library file the position of the wheels to the upright isn’t correctly modeled. Now the wheel orientation is initially always parallel to the road surface. Like this the static toe angle and static camber angle isn’t correct. That’s why the position of the wheels during driving isn’t always conform reality.

- The parameters of the car should be checked. The vehicle is the subject of rapid design changes so the parameters should be checked for sensible up to date output.

- Aerodynamics is included but it is a very rough estimation. Because of this it is possible that the car reaches speeds from over the 220 km/hour. This can be improved to get a more realistic view.

- In the emergency brake test and acceleration test the controllers on $\kappa$ must and can be perfected. More tyre information is desirable for this.

- For the fishhook manoeuvre it would be better if the change of turn was exactly on the moment when roll velocity changes of sign. Now it is only an estimation of the form from the manoeuvre.
Bibliography


## Appendix A

### List of symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Units</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>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>$V_{sx}$</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>$a_{ly}$</td>
<td>lateral acceleration</td>
<td>[m/s²]</td>
</tr>
<tr>
<td>$R$</td>
<td>radius</td>
<td>[m]</td>
</tr>
<tr>
<td>$t_{max}$</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>approximately 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>$F_x$</td>
<td>force in x-direction applied by wheels</td>
<td>[N]</td>
</tr>
</tbody>
</table>
## Appendix B

### List of available model output

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>s.FFLc2</td>
<td>globally defined forces on A-arm to upright connection</td>
<td>N</td>
</tr>
<tr>
<td>s.FFLc5</td>
<td>globally defined forces on A-arm to upright connection</td>
<td>N</td>
</tr>
<tr>
<td>s.FFRc2</td>
<td>globally defined forces on A-arm to upright connection</td>
<td>N</td>
</tr>
<tr>
<td>s.FFRc5</td>
<td>globally defined forces on A-arm to upright connection</td>
<td>N</td>
</tr>
<tr>
<td>s.FRLc2</td>
<td>globally defined forces on A-arm to upright connection</td>
<td>N</td>
</tr>
<tr>
<td>s.FRLc5</td>
<td>globally defined forces on A-arm to upright connection</td>
<td>N</td>
</tr>
<tr>
<td>s.FRRc2</td>
<td>globally defined forces on A-arm to upright connection</td>
<td>N</td>
</tr>
<tr>
<td>s.FRRc5</td>
<td>globally defined forces on A-arm to upright connection</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 angles</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>Variable</td>
<td>Description</td>
<td>Unit</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------------------------</td>
<td>--------</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>yaw rate 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>
</tbody>
</table>
# Appendix C

## Example forcereport.out

FORCE REPORT:
12-Nov-2005 11:10:29

**Suspension Front:**

<table>
<thead>
<tr>
<th></th>
<th>Left c5</th>
<th>Left c2</th>
<th>Right c5</th>
<th>Right c2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Force</td>
<td>2863</td>
<td>2064</td>
<td>2380</td>
<td>1664</td>
</tr>
<tr>
<td>Force_x</td>
<td>-533</td>
<td>299</td>
<td>-573</td>
<td>430</td>
</tr>
<tr>
<td>Force_y</td>
<td>-2684</td>
<td>1981</td>
<td>2192</td>
<td>-1551</td>
</tr>
<tr>
<td>Force_z</td>
<td>844</td>
<td>495</td>
<td>728</td>
<td>423</td>
</tr>
</tbody>
</table>

**Suspension Rear:**

<table>
<thead>
<tr>
<th></th>
<th>Left c5</th>
<th>Left c2</th>
<th>Right c5</th>
<th>Right c2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Force</td>
<td>3246</td>
<td>1889</td>
<td>3435</td>
<td>2191</td>
</tr>
<tr>
<td>Force_x</td>
<td>73</td>
<td>1702</td>
<td>30</td>
<td>39</td>
</tr>
<tr>
<td>Force_y</td>
<td>-3139</td>
<td>693</td>
<td>3351</td>
<td>-2114</td>
</tr>
<tr>
<td>Force_z</td>
<td>824</td>
<td>435</td>
<td>753</td>
<td>575</td>
</tr>
</tbody>
</table>
Appendix D

Example testresults.out

TESTRESULTS:
12-Nov-2005 11:09:23

STATIC EQUILIBRIUM POSITION TEST:
initial drop heigth: 0.00 [m]
minimum groundclearance front: 0.10 [m]
maximum groundclearance front: 0.11 [m]
minimum groundclearance rear: 0.10 [m]
maximum groundclearance rear: 0.11 [m]
standstill groundclearance front: 0.11 [m]
standstill groundclearance rear: 0.11 [m]

ACCELERATION TEST:
maximum speed reached in test: 210.34 [km/h] dis: 646.98 [m] time: 20.00 [s]
100 km/h reached in test: 100.00 [km/h] dis: 50.07 [m] time: 7.51 [s]
75 meters reached in test: 116.59 [km/h] dis: 75.00 [m] time: 8.34 [s]

BRAKE TEST:
Speed when braking applied: 54.00 [km/h]
Brake distance: 10.68 [m]
Brake time: 1.50 [s]
Appendix E

Force definition

To be able to find the highest force experienced during the tests, in `post_processing2.m` all the forces from each test\(^1\) are stored in the same matrix. These forces are collected with joint sensors on each of the suspension to upright connections during tests. Like mentioned, in `post_processing2.m` all the forces from the different tests are stored in one matrices. There are extra columns added so it is possible to determine later in which test the maximal forces occurred. After all tests are runned throw with `post_processing2.m`, and obvious all forces from the different connection points and tests are stored in the matrices, `post_processing1.m` filters out the maximum situations that have occurred on the different connection points during tests. The algorithm used here is as follows:

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

- Then 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.

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

- So 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. This rotation matrices are processed on the same way as the forces are in `post_processing2.m` so there’s also an matrix with all the different rotation matrices from the different tests stored in.

- Now there is a function file made; `fun-rotationmatrix.m`. In `post_processing1.m` the element number which is found is filled in this function. This function now searches the corresponding elements from the matrix where all rotation matrix elements are stored in. From this elements it assembles the corresponding rotation matrix and this is returned as an output to `post_processing1.m`.

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

Now 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 as `forcereport.out`.

\(^1\)forces not available for the suspension test