# Simulating the G-forces of a rallycross track

**Master's thesis**
performed in **Vehicular Systems**

by
**Ville Grandin**

Reg nr: LiTH-ISY-EX -- 07/3995 -- SE

14th December 2007

# Simulating the G-forces of a rallycross track

**Master's thesis**

performed in **Vehicular Systems**,
**Dept. of Electrical Engineering**
at **Linköpings universitet**

by **Ville Grandin**

Reg nr: LiTH-ISY-EX -- 07/3995 -- SE

Supervisor: **Anders Fröberg**
Linköping University, division Vehicular Systems

Examiner: **Professor Lars Nielsen**
Linköpings Universitet

Linköping, 14th December 2007

# Abstract

The purpose of this thesis is to design a motion simulator for a rallycross racing environment. The focus on the design is how to mechanically create the G-forces and to model them. After that is done the visually seen motion has to be electronically implemented into the motion simulator, creating as realistic as possible an experience for the driver. A program called Aprot is written in National Instruments Labview to handle the communication between the software simulator and hardware signals. Alot of focus is paid on how to represent the much larger G-forces that are experienced on a real track in the limited capacity that a motion simulator allows. For this purpose several formulas are proposed, all of which have their benefits. The simulation environment used is Racer, a well documented racing simulation that is still in development by the creator Ruud van Gaal. Aprot continuously reads specific data from a file in Racer and uses the formulas to form them into reference values for mechanics. Aprot also has a PID-controller, so that the piston positioning can be optimized.
The original plan of this master thesis was to use Aprot on a full-scale pneumatic or hydraulic prototype. However, due to time and money constraints, this was not done, leaving this work as a theoretical base on which to build upon.

**Keywords:** motion, simulator, Labview, Racer, design, pneumatic, hydraulic, G-forces

# Contents

# Chapter 1

# Introduction

## 1.1 Background and Goal

What exactly is a motion simulator? Within the boundaries of this thesis, it is described as an object that more or less encapsulates a person and creates a false sense of motion on a somewhat stationary object through visual display combined with a device that tilts and/or moves a platfrom, creating gravitational forces. They are common in amusement parks and simple versions of them can often be seen in arcade halls. More advanced versions are made for the automotive and aircraft industry, especially for military use, to train pilots on the ground. Motion simulators as a joy ride have become more and more common since their introduction in the late 1980's. The increased computer capacity and development of electromechanical steering mechanisms have allowed much smoother and wilder rides. Nowadays most software simulators for racing are designed with motion simulators in mind, often allowing movement information to be sent out on the TCP/UDP port for another computer to access. How exactly the information translates into movements, is one of the main business secrets.

Torpa racing, a small company that provides various outdoor activities, mainly focused around rallycross, wants a motion simulator as a part of expanding their activities for companies that visit them.

The main purpose of this thesis is to find a working concept for a motion simulator prototype explaining the solution to every mechanical and electrical part of it. Due to the lack of time involved compared to a company some elements are focused on more than the other. Therefore factors like nauseousness and the influence of visual display are only briefly mentioned although they have a significant importance in the motion experience.

The work in this thesis is largely based on a modular approach, where existing modules that can be used, will be used. As a result no new simulation software needs to be created, as there are already a number of candidates, some of them free of charge.

There are already many different motion simulators in existence. So what's the motivation for not just buying one? Firstly there is the price issue. There is not only the price for the motion simulator itself, but also the customization of a body, a rallycross car and a rallycross track that needs to be implemented. Also there are some additional experiences that can be built in with a custom simulator. For example creating a shaky and perhaps even slightly painful ride, which simulates the uneven gravel track. There is also the control of the equations. With complete control of how to create forces some new concepts, which perhaps don't work well on a general track, can be tried on this specific track.

## 1.2 Thesis outline

The main parts of the master thesis were to:

- Take a brief look at software simulation environments that could be used.
- Determine how the software to hardware interface would be solved.
- Extract data from Racer which could be used to calculate reference values.
- Create a control program with the data from Racer and the input from piston positions to control the analog outputs for the pistons.
- Design a motion simulator.

The last two points are the focus of the thesis. Creating the simulator, based on the design, is Torpa Racing's job.

## 1.3 Simulator modularity

To furthermore pinpoint what exactly needs to be done the motion simulator itself can be divided into several parts.

- Driver input
- Simulation software
- Software to hardware interface
- A way to create forces
- A module or platform on which forces act

In most specialized simulators the driver input, simulation software and software to hardware interface are handled by one program. But with modularity these parts can be divided into three separate, where the driver input and simulation software are bought, while the software to hardware interface is a background program that works in between the simulation software and the card that produces the outputs. The most important part of how to create forces is to have the right interface between hardware output from computer and hardware input on the actuators. The design of the motion platform is important for the equations creating the forces. If it is simple enough the formulas could just translate forces linearly. All this given the main part of the thesis became making the software to hardware interface and figuring out how to translate the G-forces from the simulation software to the motion simulator.

*Note: The lines of code that exist in this paper are all based on Matlab code structure.*

# Chapter 2

# Why use Racer?

There are a few simulation environments out there that could have been used as software instead of Racer (Racer website, see references). Here's a list of the main competitors that were considered:

- rFactor (www.rfactor.net)
- Live For Speed (www.liveforspeed.net)
- netKar (www.netkar-pro.com)

All of these simulators use physics engines that model real driving. They are all also finished products, with new versions being developed, as opposed to Racer, which is still not released in a final version.

There are some key facts that differ Racer from these commercial simulators:

- Racer is free
- You can model your own track, car and wheel setup in Racer. The other programs do not support anything but modifications to existing graphics. This is the main reason why Racer is still popular among sim gamers.
- The developer is positive toward his product being used for various non-commercial use. Such as education.
- The source code has been released. Albeit it's for an old version from 2003.

All of these factors, along with the fact that Racer is already being used in a course given by Vehicular Systems, made Racer the best choice. Even though the source code wasn't actually used in the final product, it was considered as a viable option for some time. Also, the limited time that a master thesis allows did not give the choice of examining the other software thoroughly.

The Aprot application has the possibility of, after being modified, being used on other simulation environments. One example of such is the simple and user friendly Outsim interface that Live For Speed has. Live For Speed includes an application, which when used will send the necessary data to a port, using the UDP interface, allowing it to be received on another computer. That computer can in turn control a motion simulator. The main reasons why Live For Speed wasn't chosen for this project was it's lack of physics documentation, and the inability to make your own tracks and cars.

# Chapter 3

# Extracting data from Racer

## 3.1 Different methods

When looking into different ways of getting the necessary data, of which acceleration of the car is the most important, three different approaches were considered.

1. Using the source code version. Modify source code.
2. Using the chair property of the 0.5.0 version.
3. Using the log from the latest beta version.

### 3.1.1 Source Code

The developer of Racer, Ruud van Gaal, has released the source code for the 0.5.0 Linux version of Racer. It is quite straightforward and easy to understand which classes and member functions to use to get acceleration and other data from the code. The approach had this method been used would be to write a class that compiled with the game, extracting the necessary data to the software to hardware interface. The main drawback of this approach is that it only works for Linux. This means that there is no guaranteed support for force feedback. Making the code compile on Windows is possible. However, just getting a compiled version to run on Windows without crashes would require several weeks of work, including tests. The insecurity of not knowing if the result would be good negated this option.

| Advantages | Drawbacks |
|---|---|
| Manipulate source code | Only works with Linux |
| Access to every variable in the game | Old version (0.5.0 is from 2003) |
| | No (guaranteed) force feedback |

### 3.1.2 Chair Property

Racer version 0.5.0 for Windows has a property that enables sending data out of the game through UDP on a specified port. It is easy to enable by just manipulating the configuration file ”racer.ini”. The data that is sent is acceleration (3 values) and position/orientation (9 values) of the car. The data could be received by another computer that would control the motion simulator. Unfortunately, this property has been excluded from the new beta releases.

| Advantages | Drawbacks |
|---|---|
| Windows version | Old version (0.5.0 is from 2003) |
| Access to necessary (albeit limited) variables | The latest racing wheels (e.g. G25) aren’t supported |

### 3.1.3 Using the log

In the most recent versions of Racer, the user has the choice of creating a log for later use. This is easily enabled in the main configure file, ”racer.ini”. The log file has a few key positive features. It can be read while the game is writing to it. This means that an application can poll the log file and every time new info is detected it can be used right away. The log file has a minimum sample time of 1 ms, and can store all the position and acceleration data, as well as data on throttle, brake, steering, suspension lengths and tire forces. There are also a few drawbacks. First, Racer writes data to the logfile in batches of 4 kilobytes each, meaning if you have a sample time of 1 ms, it still updates at a periodic time of 5-30 ms depending on how many parameters are written. This means that all data except the very latest is useless and takes up space (in the example of an update period of 10 ms, nine of ten samples are not needed). The second drawback is a consequence of the first. If you make the system fast, which is necessary in order to avoid motion sickness and a funky feeling, you have to make the sample time very low. This means the logfile will grow large with time. A simple calculation of 15 data values of 8 bytes (64 bits) each with 1 ms log frequency makes for a file of about 15x8x1000x60 = 7,200,000 bytes = 7.2 Megabyte of data per minute.

| Advantages | Drawbacks |
|---|---|
| Use of the latest beta version | Lots of omitted data |
| Support for advanced racing wheel (G25) | Log file grows very large with time |
| Windows version | Background application will steal computer capacity |
| | Windows is not a real-time system |

## 3.2 Decision

Racer is still in development, and the latest version, which at this time is four years newer (0.5.3 beta 5) than the 0.5.0 version, has improved features such as improved graphics, physics, and audio. Also, it supports the latest racing wheels with a stick-shifter and a clutch. Therefore, the choice ultimately fell on the method of extracting data from the log.

### 3.2.1 Tackle Drawbacks

When coding the application it was possible to optimize the time from the game writing to the log file until it was read to only a matter of a few milliseconds. This was done by logging every millisecond, and log a few more parameters than necessary. The average period it logged the specified parameters was approximately 8 ms. This means that 7/8 lines in the log are omitted. The application was also made to delete the log file after the driving mode was left, thus there will be no problem with big files. Running the application at this stage had very little effect on computer performance. A visual test of the amount of frames per second in Racer (using Fraps) showed a decline (on medium graphics settings) from about 30 to about 27. It should also be noted that the computer used for these tests was a AMD Athlon 2800+ (1.80 GHz), 1.0 GB of RAM. More importantly, a modern dual-core processor should be much more suitable to run both Racer and the software to hardware application on the same computer due to it's increased ability to run simultaneous processes.

# Chapter 4

# Simulator design

This part of the thesis was partly research about how existing motion simulators are built. The main part was to design a simulator focusing on minimizing the cost with a decent performance. Also important was that Torpa wanted the simulator to look and feel like a real rallycross car from the outside and the driver seat. As a result a pneumatic model was investigated at first (see section 7) but quickly discarded.

## 4.1 Some existing simulators

*Force dynamics* (see references for website) use a electromechanical triangular piston setup, scaling down on innecessities. Their simulator is made for racing and they use Live For Speed as one of their main computer simulators. Advanced motion simulators such as the one at *VTI in Linköping* (again, see references for website) use both rails and hydraulics to create realistic lateral G-forces. However they are mainly based on analyzing driver behaviour instead of racing. This simulator is based on simplicity and cost reduction in mind, similar to *Force dynamics*'.

## 4.2 Piston setup

Two piston configurations were considered. A setup with four pistons, simulating one at each wheel (again, see 7) and a triangular piston setup. The four piston setup has the advantage of initially having slightly easier equations, with opposing pistons being the same formula but with inverted signs on each factor. The major disadvantages are that this construction requires more signals and if the system is not to be stressed, dependability between the piston positions has to be introduced, so that the individual pistons can't assume any position. This problem does not exist in a triangular setup where all pistons can move freely without any problem. It was therefore decided that a triangular setup would be used.

### 4.2.1 Producing G-forces

The only factor that is used to theoretically describe the G-force experienced by the person in the motion simulator, is the angle relative to earth gravity of the motion platform. No attention is paid to the impact that change in G-force has on the sense of balance. Therefore the G-forces are only produced by the lean created by the pistons.

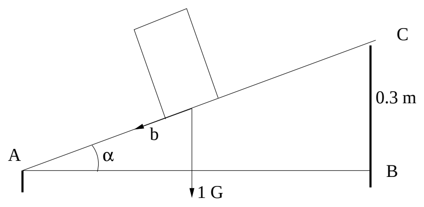

Figure 4.1: The G-force is produced only by leaning the simulator.

### 4.2.2 Piston placement

The pistons are locked into a vertical position in the ground, but not to the motion platform. This because the equations will be much simpler and create an almost perfect linear relationship between angle and piston position. To achieve this, the platform is able to slide a few centimeters as shown in figure 4.2.

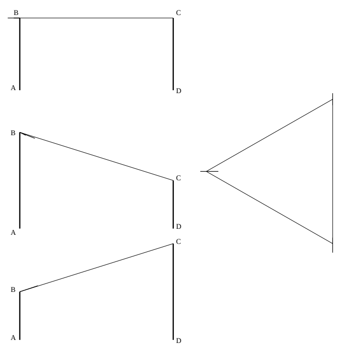

Figure 4.2: Rails on the junction between piston and simulator platform are used not to stress the fixed pistons. To the right, how the rails are placed relative to the pistons, as seen from above, with one piston in each corner of the triangle.

To scale down on the required length of the pistons, thereby cutting costs, they are placed relatively close to eachother. Thus shorter pistons can create the same amount of lean. Discussions lead up to a stroke length of 300 mm being chosen for each piston. The actual maximum lean is only a minor factor because the nautiousness also has to be considered (less lean leads to less nautiousness). Based on these facts and the data collected from the real track at Torpa (figure 6.3), the lateral G was decided to not be more than 0.3 G and the longitudinal G not to be more than 0.25 G.

The function of the pistons is such that in the ”default” state all pistons are extracted at half their length, having a total length of approximately 450 mm from the floor junction. With one piston fully retracted and one fully pulled out the situation can now be seen as a right triangle as in figure 4.1 with the important angle $\alpha$ as a measurement of the G-force. From the figure the experienced G-force, $G_{exp}$, is calculated as $G_{exp} = 1 \cdot \sin\alpha$. If $G_{exp}$ is to be no more than 0.3 in the lateral (sideways, caused by turning) case, then the minimum distance A-C is easily obtained trigonometrically as:

$$\alpha = \arcsin(G_{exp}) = \arcsin(0.3) = 17.5°$$
$$\tan\alpha = BC/AB \Leftrightarrow AB = BC/\tan\alpha = 0.3/\tan 17.5 = 0.96m$$

Same calculation for the maximum longitudinal (front-back, caused by accelerator and brake) G-force 0.25 gives:

$$\alpha = \arcsin(G_{exp}) = \arcsin(0.25) = 14.5°$$
$$\tan\alpha = BC/AC \Leftrightarrow AC = BC/\tan\alpha = 0.3/\tan 14.5 = 1.16m$$

Rounded up the values land at 1.0 meters and 1.2 meters.

### 4.2.3 Weight calculations

The request for a realistic rallycross car body as the platform creates some problems of it's own. The whole construction becomes bigger than necessary, which adds weight. The driver position in the platform is also important. This is because a person not placed in the center of the triangular setup will also experience undesired vertical acceleration as the platform moves. Additionally, the center of gravity is affected by the driver position. Another factor that is important but uncalculable is the nausiousness that people often experience from simulators, especially if they have great maximum lean.

The weight of the whole simulator on top of the pistons has been estimated by Torpa to be 400 kg (including driver). With the pistons positioned on the left side of the vehicle this opens up for some problems to balance the center of gravity of the simulator on the center of gravity between the triangular point. It's hard to say exactly where the center of gravity will be on a stripped down chassis, with everything under the bonnet, rear seats and right seat etc. removed. The chassis will also be cut off just behind the back seat,

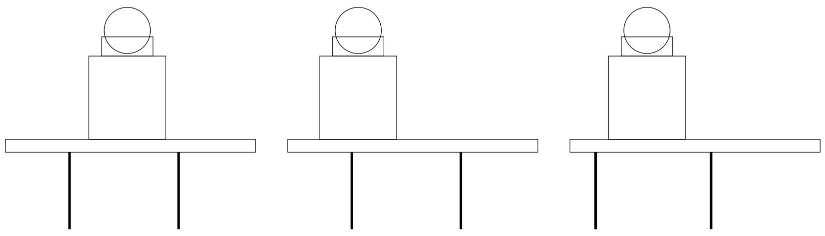

Figure 4.3: Three different ways of placing pistons in relation to the driver.

shortening the simulator by about one meter. This calculation is based on a chassis weight, m1, of 300 kg with center of gravity exactly at the center of the chassis, and the driver and seat, m2, being a 100 kg unit mass placed as shown in figure 4.4.

$$x = \frac{0 \cdot m2 + 0.4 \cdot m1}{m2 + m1} = \frac{0 \cdot 100 + 0.4 \cdot 300}{100 + 300} = 0.3$$

For the above reasons, the placement of the pistons is decided to be a compromise between having the driver in the middle, and placing the platform center of gravity as close to cm as possible. Additionally the triangular setup is rotated 90 degrees from a natural and common setup, which has one piston in the front and two in the back, to having one piston to the left and two to the right. This balances the weight better between the three pistons. As discussed in section 4.2.2 the distance laterally between the pistons is one meter. The optimal position for the driver is in between, at 0.5 meters, while the optimal point for weight balance is at the center of gravity of a triangle, which is 2/3 of the length or 67 cm from the left side piston. Now the driver is sitting 0.5 meters from the left side of the chassis, while the center of gravity is 0.8 meters from the left side. The difference between the two optimal places being $(0.8 - 0.5) - (0.67 - 0.5) = 0.13$ m = 13 cm. With this in mind it is decided that the piston setup is moved 10 cm to the right as seen in figure 4.7, which means the driver is 10 cm off his optimal position, and the center of gravity is 3 cm off it's optimal position.

### 4.2.4 Hydraulic specifications

Designing the hydraulic system and piston size/force is not part of the master's thesis. However some important specifications on their performance are needed for the company making it. Figure 4.6 shows how fast the G-forces change at the most extreme on the track. The pistons should have rise and falltime to match these. Looking at the figure a rise and falltime of at most 15 samples (600 ms) preferably less than 10 samples (400 ms) is desired. Additionally although the total weight is estimated at 400 kg, and the pistons

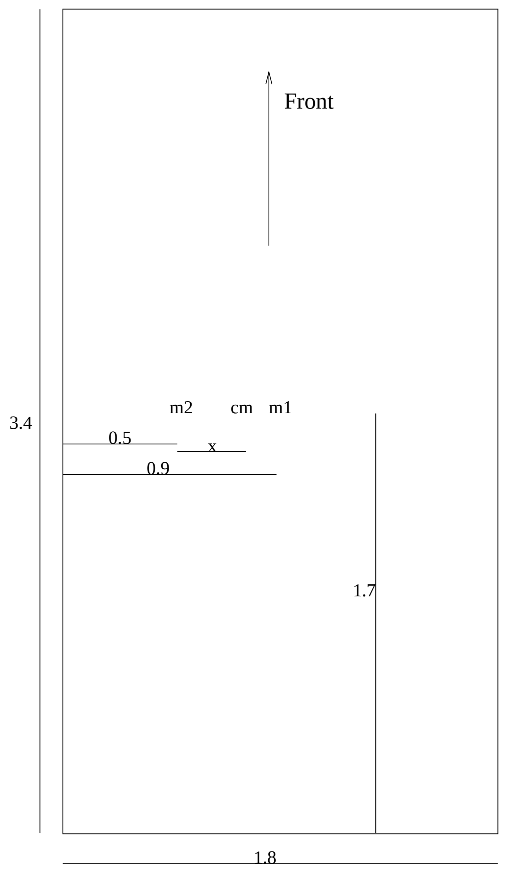

Figure 4.4: The common center of mass, cm, for the simulator design, is a function of the two point masses m1 (chassis), which is in the middle and m2 (driver), which is located to the left.

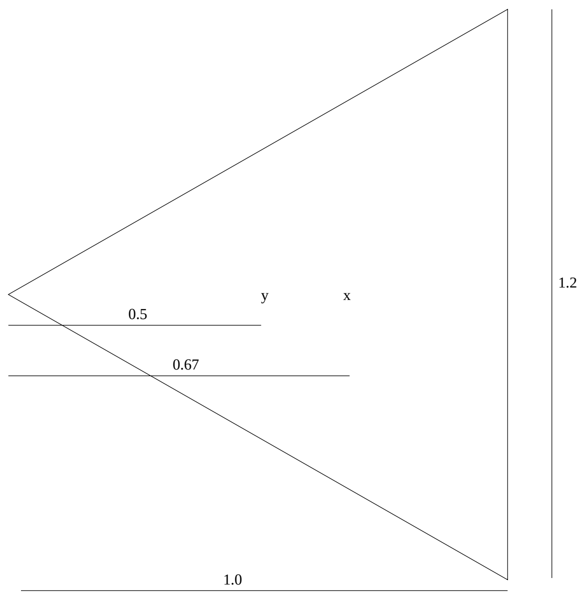

Figure 4.5: The hydraulic setup with one piston in each corner means that the ideal position to place the center of mass of the chassis is x, while the ideal position to place the driver in is y.

are taking about one third of the weight each, some extra weight has to be considered not to make them too weak. In this case 200 kg each is chosen.

- Rise and falltime of at most 0.6 seconds.
- Each piston capable of holding 200 kg of weight stationary.
- Proportional valves that have an interface which allows for analog input control, preferably in the range 0-5 Volt.

## 4.3 Final design

The final design is shown in figure 4.7. Some final notes of interest:
The non-centralized piston placement will mean that for an onlooker from the outside the simulator will move more vertically on the right side than the left side (about 15 cm more between extreme values). It is a good idea to add some kind of helping springs on the right side to lighten the force on the two pistons on the right-middle.

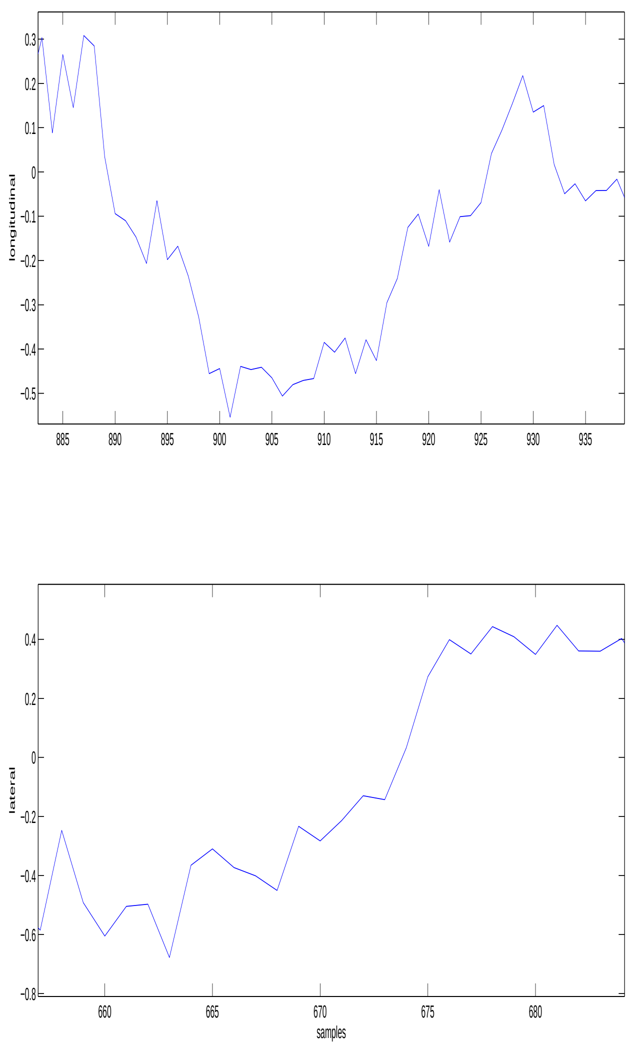

Figure 4.6: Zoom of extreme G derivative from figure 6.3. Sample rate 25 Hz. These extreme G derivatives are important as a specification of the capacity of the pistons, since they describe how fast they have to move.

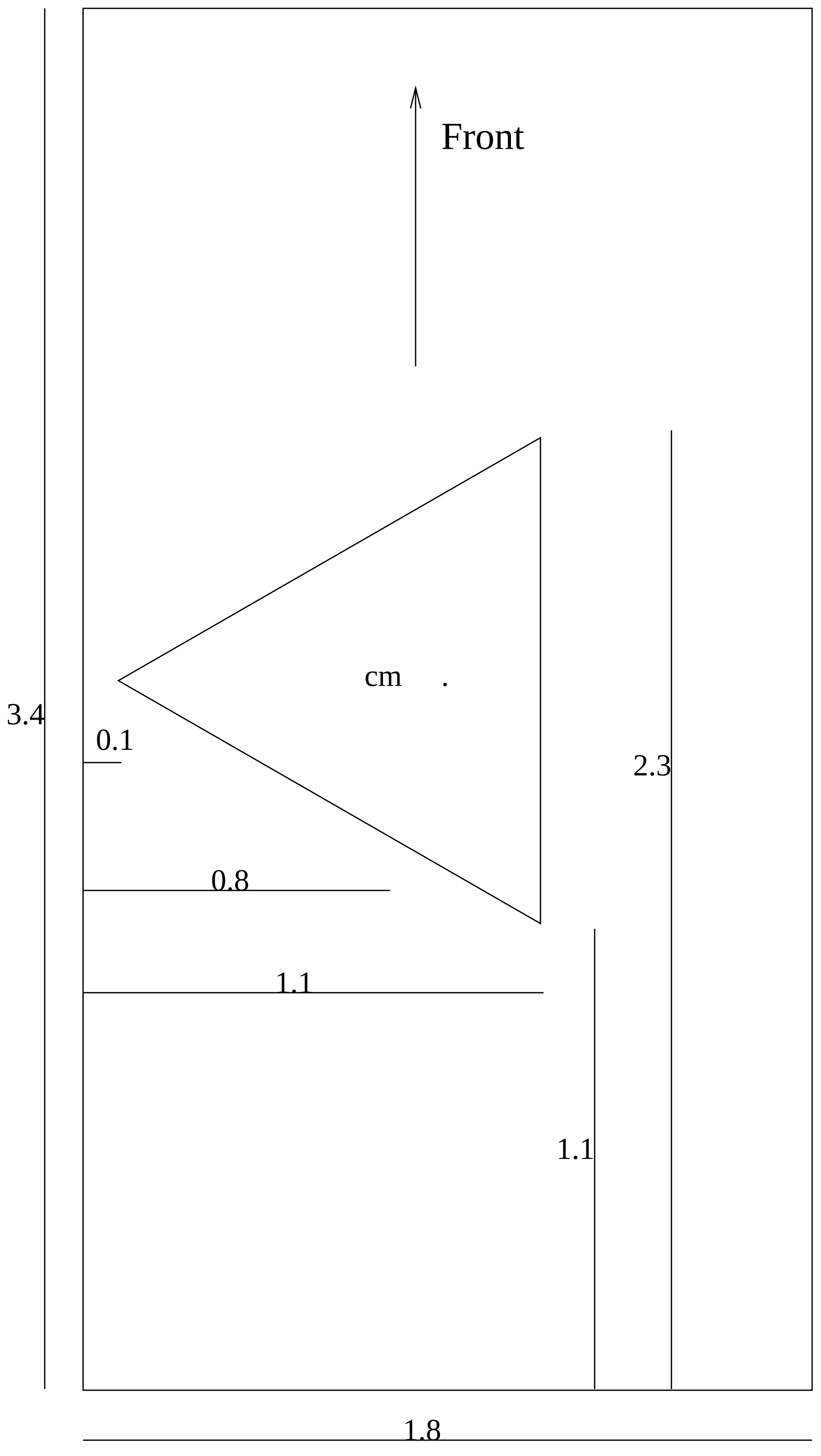

Figure 4.7: The final design. cm = center of mass. The pistons are placed in the corners of the triangle.

# Chapter 5

# Computer to hardware interface

Early on it was decided that pistons would be used as the actuators creating the G-forces. For further reference see chapter 4. With this in mind, and after the failed attempt with the pneumatic setup, the specifications for the hydraulic motion simulator that Torpa were building, in regards to signal input and output from the computer, were agreed to be:

- 3 or 4 analog or pulse width modulated inputs from potentiometers giving the position of each piston.
- Analog to digital resolution of at least 32 values (5 bits). This means that each piston can have at least 32 positions, which should be enough to be able to satisfy the need for enough diversity in the G-forces.
- Sample rate of at least 100 Hz. Since the values need to be updated at at least 10 Hz (probably 20 Hz) the sample rate needs to be a lot faster especially if there is a PID-controller in the software. A common rule (Glad, Ljung et al 2003) is that the sample rate should be 10 times faster than the system for good control.
- 3 or 4 analog or pulse width modulated outputs to control proportional valves. The valves control the pressure in each piston, esentially controlling their position.

Two ways to make this interface were considered. Either to program a PIC processor that meets the demands, or to buy a hardware module with A/D converters and digital outputs built into it. The latter one was chosen since all the tools, including a software development program, were already available.

## 5.1 Interface card

The National Instruments USB-6008 hardware interface board meets most of the above requirements. It has 8 A/D converters, 12 digital I/O ports, a sample rate of 2.5 kHz (the sample rate is esentially limited by the software to be lower), and 12 bits or $2^{12} = 4096$ values of resolution. It only has 2 D/A converters, giving only 2 analog outputs. It is possible to use 8 of the digital ports as an analog output with an external D/A converter. This is much cheaper than buying an overdimensioned better board with 4 analog outputs. This was also one of the reasons the three piston setup (see section 4) was chosen in the design phase. NI USB-6008 also has a USB interface, and NI-DAQmx API that Labview (see section 6) supports. In conclusion, all the necessary tools and an easy to learn interface.

# Chapter 6

# Simulator software control theory

The NI USB-6008 supports National Instruments' own software development tool, Labview (Labview website; see references). Labview was used to create the simulator software interface between Racer and the motion simulator.

## 6.1 Labview

Labview is a graphical programming language, similar to e.g. Simulink in MATLAB. It has functions for most things you can do in C++ code, including if and case structures, while and for loops, extensive file reading and writing, feedback loops, and user defined classes with member functions. It also has the possibility of displaying a graphical user window, similar to Visual Basic.

## 6.2 The application

### 6.2.1 Reading log file

Each line of data in Racer's logfile is a complete set of data, sampled at a specific time. Aprot manipulates the ini file for Racer, so that the data in the log is in the right order. Theoretically, the more parameters that are set to be logged, the faster the application will be. This because when data is written to a file it is stored in a buffer, usually 4 kb large, and written to file when this buffer gets full. Therefore there will be a lot of old data each time the logfile is written to.

1. When starting Aprot, Racer will also start. Aprot will run in the background.

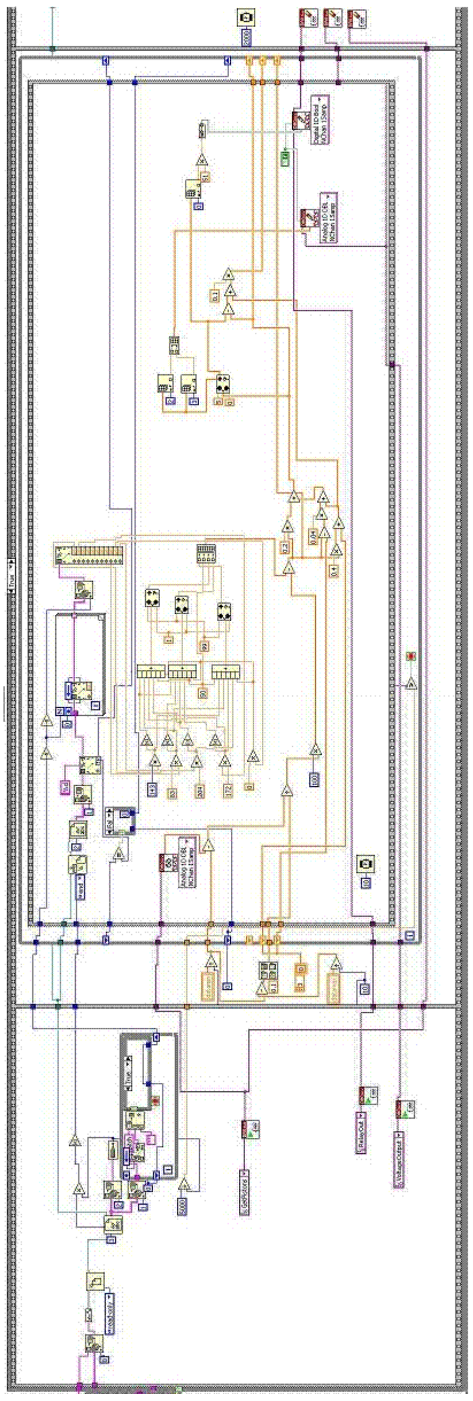

Figure 6.1: A view of the main component of the program Aprot created in Labview.

2. Aprot is scanning for the log file until it is found.

3. The log file is created, this means that the user is ”racing”.

4. Aprot reads the full data line above the line that detected end of file (most recent full data available).

5. The appropriate data from this line is used to update the control loop. After which the program jumps back to step 4. This is a major step for which possible ways to use the data is explained throughout the rest of this chapter.

6. A timeout means that Racer has stopped updating the logfile, thus quit ”racing”. This will occur when Aprot has read the exact same data a number (decided by logfile settings) of times. Aprot removes the log file, then jumps to step 2. The timeout can be set in the program, a timeout of 0.5 seconds, barely noticeable to the driver, held up well during several test runs.

## 6.3 The control loop

The control loop uses the data from Racer as reference values, compares them to the piston position values, and determines the voltage output from that. There are of course many different ways to control the system, but Labview doesn’t support advanced mathematical formula in an easy way. The choice fell on three PID (Proportional, Integral, Derivative) controllers (Glad, Ljung et al 2003) because there is only one value, piston position, that needs to be controlled. Details on the PID-controller are explained in section 6.3.7.

The data that is usable from racer is:

- x velocity (horizontal, $v_x m/s$)
- y velocity (vertical, $v_y m/s$)
- z velocity (horizontal, $v_z m/s$)
- longitudinal acceleration ($a_x m/s^2$)
- lateral acceleration ($a_z m/s^2$)
- vertical acceleration ($a_y m/s^2$)
- pitch angle ($\theta$radians)
- yaw angle ($\psi$ radians)
- roll angle ($\beta$radians)

As explained earlier, there are many more factors that can be used, such as tire slip, throttle, steering wheel position etc., Racer already uses these values to determine the data in the list above, therefore it would be unwise to recalculate use them again. A factor that has been left out is angular velocity (and angular acceleration). But they too are included in the acceleration data. Take the simple example of a car going through a ”perfect” curve at a constant speed and without tire slip. The force equation for the car is:

$$F = \frac{m \cdot v^2}{r} \tag{6.1}$$

where
r = radius of curve
v = velocity of car
m = mass of car
F = Force
v/r = angular velocity

Translated into mass independent G-forces:

$$\text{G-force} = \frac{F}{m} = \frac{v^2}{r} \tag{6.2}$$

Thus validating the point that the angular velocity is a function of the force, which determines the G-force, which is the same as acceleration. Therefore it would be unwise to use this factor in any way to generate the reference values for the pistons unless the acceleration factor is removed.

Acceleration, obviously is a data that must be used, since acceleration is almost equal to G-forces and this is what the driver actually feels.

The leaning angles of the car can be used e.g. if you want to have the car lean the same amount in both the physical and software simulators. This could be factored in only at low speeds, or at all speeds, both of which will decrease and increase the sensation of G-forces through acceleration in their own way.

Velocity can be used to determine if different multiplying factors on acceleration are to be used for different speeds, as well as for the leaning angle. It can be a scaling factor only. The sensation of speed is impossible to achieve just by leaning a car body.

It is of course very hard to know what the equation should be for the best driving experience in the motion simulator without doing a series of tests and evaluate. Since a real simulator was never built the options can only discussed, but it can’t be determined which one is the best.

Before proceeding it is important to look at how Racer defines positive accelerations, velocities and angles, so that the formulas are correct. Racer has it’s own definition of these units, as seen in figure 6.2.

Because of the velocity vectors fixed to the ground, to get a measurement of the speed (not necessarily the same as the speed on the speedometer, but

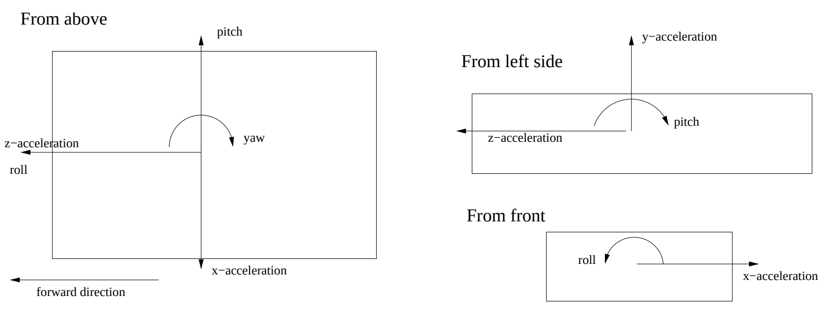

Figure 6.2: Definition of positive accelerations and angles in Racer. The velocity of the car is measured relative to a coordinate system fixed on the ground. These definitions were found through tests in Racer, which can display the data in real time.

close) of the vehicle the formula is:

$$speed = \sqrt{v_x^2 + v_y^2 + v_z^2} \tag{6.3}$$

### 6.3.1 Producing reference values

To aid in the process of retrieveing the best equations for reference values, some reference data collected from the track at Torpa was used, see figure 6.3. Some interesting facts if the plot is looked at carefully are:

1. The lateral acceleration is rarely more than 0.6 G, but spikes to 1.0 G at a few points.

2. The longitudinal acceleration seems to have an offset of 0.05 G, as seen at the beginning and end of the graph.

3. The longitudinal acceleration is rarely more than 0.35 G (0.4 - offset), but spikes to 0.5 G at a few points.

4. The longitudinal retardation is rarely less than -0.25 G (-0.2 - offset), but spikes to -0.5 G at a few points.

5. The lateral acceleration is generally greater than the longitudinal acceleration.

6. The vertical acceleration has a lot of noise.

Point 5 presents a problem for which an example of a solution is presented in 6.3.5.

The following equations are all based on the triangular piston setup with one piston to the left, and two to the right, see section 4. The two right

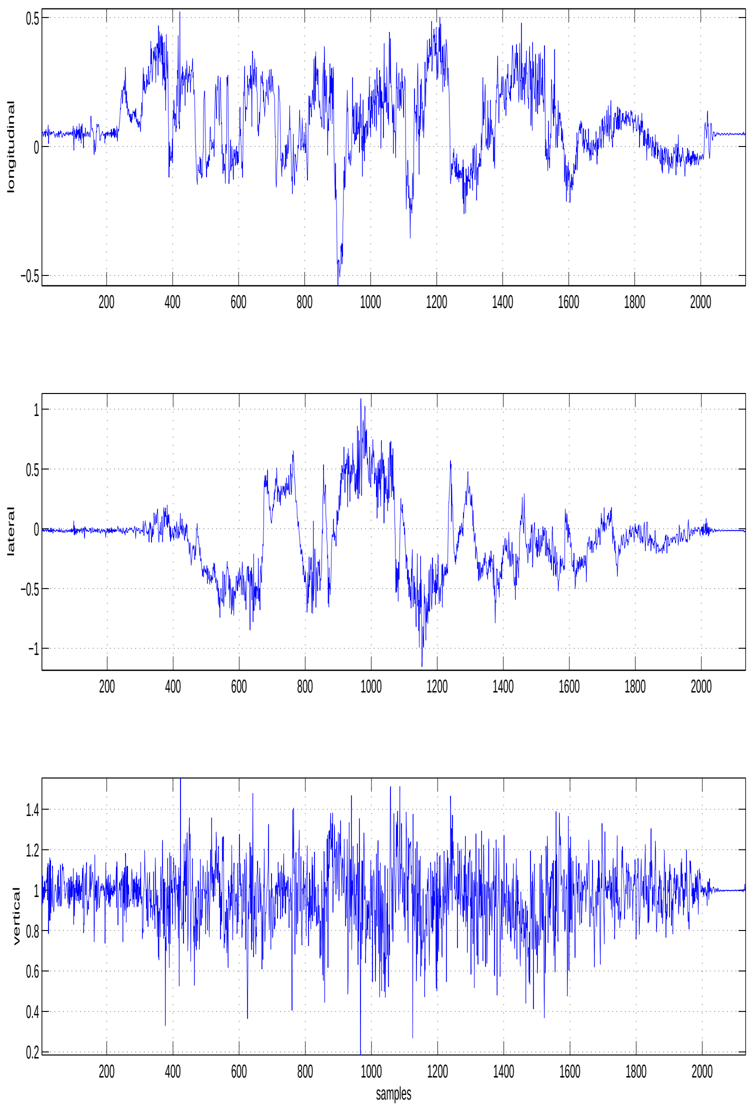

Figure 6.3: Measurement of G-forces at the real Torpa racing track at 25 samples per second (2000 samples = 80 seconds). Note: Due to unexpected problems when measuring the forces, this figure has samples of unread data removed, Therefore this is not a perfect measure of the track (see .1 for more information). It is however a decent measure of the max G-forces that are experienced on the rallycross track and with a rallycross car. The measurement was done with an accelerometer.

pistons will act identically on lateral forces and angles, while the left piston acts inverted to them. The longitudinal forces and angles have no impact on the left piston since it is in the middle seen longitudinally. The two right side pistons will be identically affected by the longitudinal forces, but in opposite directions.

Here it is assumed that if a copy of the real track and car is made in Racer, the data produced from Racer will be the same as the measured data. In reality this will not be the case. Consider for example the method of mesuring the G-forces on the real track. An accelerometer (a gyro) was used, which is affected by the car orientation. The accelerations from Racer is based on force formula at the center of gravity and would show 0 G acceleration on an immobile car no matter the orientation of the car. But since there is no data from a rallycross track in Racer available here, one can for the sake of discussion assume that real world and computer world data are equal. The only things that needs to be adjusted for a modeled track are the coefficients for each parameter.

### Piston position representation

Each piston has a sliding potentiometer that linearily represents it's position as a voltage value between 0 and 5 Volts. The potentiometer voltage value from each piston is represented as a value between 0 and 100, where 0 is minimum position and 100 is maximum position of piston. The process of getting the voltage value that represents a certain numeric value between 0 and 100 is by configuring the simulator. The pistons are sent to the minimum position, from which a mean minimum voltage value (min_cyl) is sampled from the potentiometers. Then they are brought to the maximum position, from which a mean maximum voltage value (max_cyl) is stored for each of the three potentiometers. The simple equation to create the piston position between 0 and 100 is then:

$$y = \frac{cur\_cyl - min\_cyl}{max\_cyl - min\_cyl} \bullet 100 \tag{6.4}$$

where cur_cyl is the current voltage readout. With this formula the importance of the calibration of potentiometers is minimal. All potentiometers can be individually different, and the difference between maximum and minimum reference values does not affect the performance, as long as the potentiometers act linearly between the values.

The simplification introduced in section 4.2.2 coupled with the small angles (no more than $18^\circ \Rightarrow \sin\alpha \approx \alpha$) that can be produced leads to all formulas being based on a linear relationship between piston position and experienced G-forces.

### 6.3.2 Simple acceleration

The easiest set of equations are to base the maximum reference values on the maximums from the data. Notice here that the dead-zone and non-linearity between control signal and valve flow that usually exists in proportional valves is not compensated at all. Dead-zone is a property which means that it takes a little extra power to get things moving, esentially meaning that the proportional valve is highly non-linear around it's center value. Because some proportional valves have the dead-zone compensated with built-in electronics, the amount of dead-zone varies alot between different price ranges and manufacturers. Therefore it is in this case not incorporated into the equations. In a real system one solution could be to increase the size of small control signals, so that the entire system becomes somewhat linear.

If no attention is paid to the lean of the car, the reference values, one for each piston, can be calculated according to the following formula:

Front-right:

$$r_{fr} = 50 + k_{lon} \cdot a_z - k_{lat} \cdot a_x \tag{6.5}$$

Left:

$$r_l = 50 + k_{lat} \cdot a_x \tag{6.6}$$

Rear-right:

$$r_{rr} = 50 - k_{lon} \cdot a_z - k_{lat} \cdot a_x \tag{6.7}$$

The question now is how to determine the coefficients. This is done based on the facts in section 6.3.1. Based on acceleration longitudinally from -0.25 to 0.35, because of the spikes 0.35 is used as the maximum representation to determine $k_{lon}$ in equation 6.5 (assume no lateral acceleration involved):

$$k_{lon} \approx abs\frac{r_{fr,max} - 50}{a_{z,min}} = abs\frac{100 - 50}{0.35} = 143 \tag{6.8}$$

This means that longitudinal acceleration is mapped linearily as 0.35 G on the track becomes 0.25 G in the motion simulator.

Based on acceleration laterally from -0.6 G to 0.6 G laterally $k_{lat}$ can be determined using equation 6.6 as (assume no longitudinal acceleration involved):

$$k_{lat} \approx \frac{r_{l,max} - 50}{a_{x,max}} = \frac{100 - 50}{0.60} \approx 83 \tag{6.9}$$

This means that lateral acceleration is mapped linearily as 0.6 G on the track becomes 0.3 G in the motion simulator.

This choice of coefficients is straightforward. It is also important to realize that with one coefficient much larger than the other (in this case $k_{lon} >> k_{lat}$), that force can dominate the feel in some way. For example let's say a driver is exiting a corner, in reality experiencing 0.3 G both longitudinally and laterally. If he is to experience something realistic the motion simulator should lean equally in both angles, but the equations would make each piston have the following position:

Front-right:

$$r_{fr} = 50 + 143 \cdot 0.3 - 83 \cdot 0.3 = 68 \tag{6.10}$$

Left:

$$r_l = 50 + 83 \cdot 0.3 = 75 \tag{6.11}$$

Rear-right:

$$r_{rr} = 50 - 143 \cdot 0.3 - 83 \cdot 0.3 = -18 = 0 \tag{6.12}$$

In this case the longitudinal acceleration is 0.3 G, but because of the shape of the equations, the longitudinal G-force is only 68% (difference between front and rear piston is $68 - 0$) of the maximum, which is 68% of 0.25 G = 0.17 G . The lateral acceleration of 0.3 G is scaled down to 41% (the difference between left and right pistons is $75 - \frac{68+0}{2} = 41$) of the maximum, which is 41% of 0.3 G = 0.12 G. This 0.05 G difference is probably not something a driver would notice severely, but is still worth mentioning. A slight modification in the formulas to reduce the difference is presented in section 6.3.3.

The actual G-forces that can be produced by lean (0.3 G) are much less than the 0.6 (up to 1.0) G that exist in the real system. The above equation maps 0.0 to 0.6 G linearly to the scale of the real system which is linear from 0 to 0.3 G. To aquire values closer to reality the $k_l at$ coefficient would need to be increased substantially to be twice as large. The downside to this is that the larger range of G-forces experienced laterally will be cut to max out at 0.3 G. In other words, the platform will lean at maximum in just about every corner. The same problem but not as great exists in the longitudinal direction where in the above equation acceleration from -0.35 to +0.35 G will be scaled down linearly to -0.25 to +0.25 G. Figure 6.5 shows what the largest possible coefficients, which map 0.1 G as 0.1 G, 0.2 G as 0.2 G etc., would result in. How the coefficients are actually chosen has to be based on tests on a real platform because the most important factor, how the G-forces are experienced in a real situation, can't be discussed here.

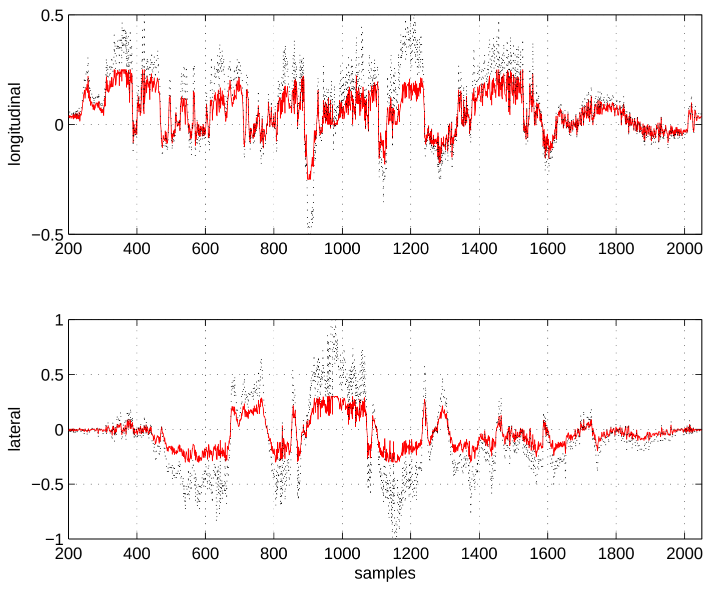

Figure 6.4: How the actual measured G forces (dotted black) are scaled down with equations 6.5 to 6.7 (solid red) if the coefficients are $k_{lon} = 143, k_{lat} = 83$. The individual pistons are at max or min level at an average of 12% of the time. The pistons are almost constantly moving.

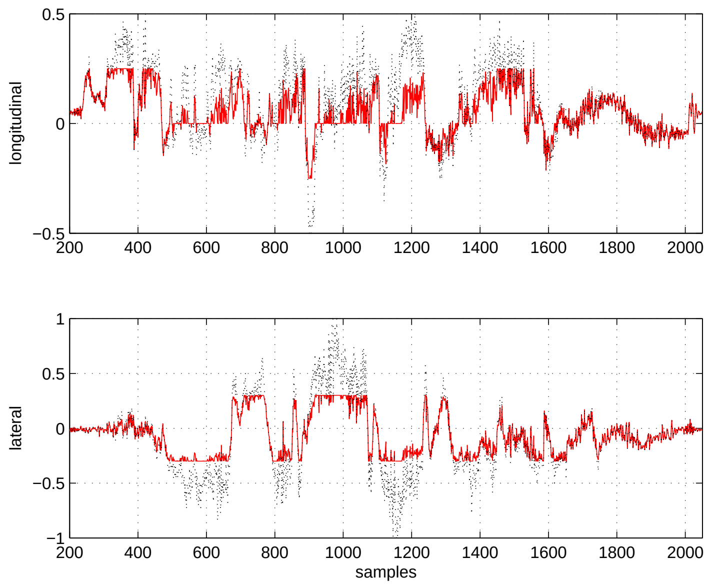

Figure 6.5: How the actual measured G forces (dotted black) are scaled down with equations 6.5 to 6.7 (solid red) if the coefficients are $k_{lon} = 200, k_{lat} = 160$. Especially for the lateral forces it can now be seen that they hit max level a lot more. The individual pistons are at max or min level at an average of 34% of the time. The real forces are followed more closely but there is less room for flexibility. If the coefficients are any larger than this the experienced forces in the simulator may be bigger than the real forces from the track.

### 6.3.3 Dependable formulas

To make the platform lean more closely to the desired G-force some dependability can be introduced into the formulas.

Front-right:

$$r_{fr} = 50 + k_{lon} \cdot a_z - k_{lat} \cdot a_x \tag{6.13}$$

Rear-right:

$$r_{rr} = 50 - k_{lon} \cdot a_z - k_{lat} \cdot a_x \tag{6.14}$$

Left:

$$r_l = abs\left(\frac{r_{fr} + r_{rr}}{2}\right) + 2 \cdot k_{lat} \cdot a_x \tag{6.15}$$

In the above equations, 6.13 and 6.14 are exactly the same as before, while 6.15 has been changed to depend on the result of the first two equations. This means the longitudinal G-force will be exactly the same as before, but the lateral G-force can be bigger. If these equations are adopted to the theoretical case discussed in section 6.3.2, with longitudinal and lateral G-forces both being 0.3 G and the coefficients being $k_{lat} = 83$ and $k_{lon} = 143$. Now the piston reference values become:

Front-right:

$$r_{fr} = 50 + 143 \cdot 0.3 - 83 \cdot 0.3 = 68 \tag{6.16}$$

Rear-right:

$$r_{rr} = 68 - 2 \cdot 143 \cdot 0.3 = -18 \Rightarrow r_{rr} = 0 \tag{6.17}$$

Left:

$$r_l = abs\left(\frac{68 + \mathbf{0}}{2}\right) + 2 \cdot 83 \cdot 0.3 = 34 + 50 = 84 \tag{6.18}$$

Comparing to the former example, the longitudinal G-force is the same $(68 - 0) = 68\%$ of the maximum (0.17 G). While the lateral G-force now is $84 - 34 = 50\%$ of the maximum (0.30 G), which is 0.15 G. So now the actual difference in G-force is only 0.02 G, as opposed to the 0.05 G in the earlier example. These formulas are exactly the same as those in 6.3.2 as long as all pistons have reference values between 0 and 100. But as soon as one or both the right side pistons hits the limit, the left piston can compensate for it until it too hits the limit. The result of these equations applied to the data from figure 6.3 are in figure 6.6.

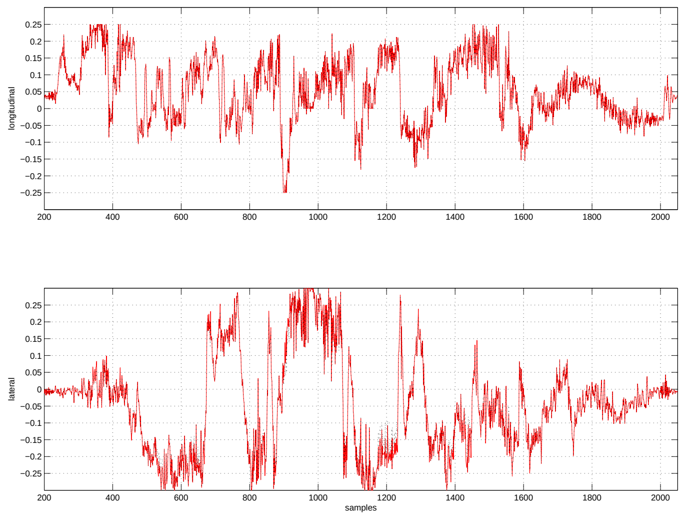

Figure 6.6: The difference between the G force representation using either equations 6.5 to 6.7 (dotted black) or 6.13 to 6.15 (solid red) if the coefficients are $k_{lon} = 143, k_{lat} = 83$. The dotted black is almost impossible to see because the difference is very small. There is no difference in the longitudinal G-force, but a general small increase of the lateral G-force (at most the increase is 0.03 G). This results in somewhat better representation of lateral G-forces.

### 6.3.4 Including orientation

The main purpose of the orientation data is to lean the motion simulator when the acceleration is very small. If you stop in a steep uphill, the motion simulator will also stop at a similar angle. In this model, the orientation input has the same effect on the reference values at all speeds. A $10°$ uphill slope will lean the motion simulator about $10°$. The equations are then:

Front-right:

$$r_{fr} = 50 + k_{lon} \cdot a_z - k_{lat} \cdot a_x + k_{pitch} \cdot \theta - k_{roll} \cdot \beta \tag{6.19}$$

Left:

$$r_l = 50 + k_{lat} \cdot a_x + k_{roll} \cdot \beta \tag{6.20}$$

Rear-right:

$$r_{rr} = 50 - k_{lon} \cdot a_z - k_{lat} \cdot a_x - k_{pitch} \cdot \theta - k_{roll} \cdot \beta \tag{6.21}$$

Here $k_{lon}$ and $k_{lat}$ can be the same as in equations 6.8 and 6.9. The $k_{pitch}$ and $k_{roll}$ have to be based on a immobile car (accelerations = 0) and the lengths from the motion simulator design.

Longitudinally the biggest producable angle is $\theta = \arctan(0.3/1.2) = 0.2450$ rad ($14.0°$) and it is produced when the piston value is either 0 or 100. Put into equation 6.19:

$$\begin{aligned} r_{fr,max} &= 50 + k_{lon} \cdot a_z - k_{lat} \cdot a_x + k_{pitch} \cdot \theta - k_{roll} \cdot \beta \Leftrightarrow \\ 100 &= 50 + k_{lon} \cdot 0 - k_{lat} \cdot 0 + k_{pitch} \cdot 0.245 - k_{roll} \cdot 0 \Leftrightarrow \\ k_{pitch} &= 50/0.245 = 204 \end{aligned}$$

Laterally the biggest producable angle is $\beta = \arctan(0.3/1.0) = 0.2915$ rad ($16.7°$). Put into equation 6.20:

$$\begin{aligned} r_l &= 50 + k_{lat} \cdot a_x + k_{roll} \cdot \beta \Leftrightarrow \\ 100 &= 50 + k_{lat} \cdot 0 + k_{roll} \cdot 0.2915 \Leftrightarrow \\ k_{roll} &= 50/0.2915 = 172 \end{aligned}$$

The upside of these added parameters is that the platform will lean much more closely with the track, especially when driving carefully (because of the less G-forces present).

### 6.3.5 A more advanced approach

An attempt at a solution of the problem of the real lateral forces being much larger than what can be experienced in the simulator is presented here.

From a close look at the lateral G-force graph 6.3 it is obvious that at a few points along the track the lateral G-force increases quickly from a high value to an even higher value (e.g. at sample 950 from 0.4 to 0.8 G). If the motion simulator is already at max sideway lean, nothing will happen here. But if there is a function that slowly straightens up the car a little at constant G, then these short G spikes could be partially felt too.

A theoretical way to do this is to constantly change the factor $k_{lat}$. The idea is that originally all lateral values up to 0.3 G are exactly mapped to the lean, which means $k_{lat} = k_{lat,max} = 165$, but as long as the lateral G is less than 0.6 G $k_{lat}$ will constantly go down to the value where 0.6 G is mapped as 0.3 G, and 0.3 G is mapped as 0.15 G. This value is $k_{lat} = 83$. This means that all values from 0.6 G (in this example) and up will be represented as maximum lean, but 0.3 G will also be possibly represented as maximum lean if the lateral G has been low for a short time.

```
if $(abs(a_x) < 0.6)$ & $(abs(a_x) > 0.15)$
    $k_{lat} = k_{lat} \cdot K_{lat,minus};$
else
    $k_{lat} = k_{lat} + K_{lat,plus};$
end
if $(k_{lat} < 83)$
    $k_{lat} = 83;$
elseif $(k_{lat} > 165)$
    $k_{lat} = 165;$
end
(continued with e.g. equations 6.5 to 6.7.)
```

The oddest sensation a driver can experience with these equations is if he is taking a fast corner at constant 0.3 G he will feel as if the lateral G falls from 0.3 G to 0.15 G in 1.9 seconds (with $K_{lat,minus} = 0.985$) in the simulator. The driver may then think that he can actually turn more because of the lesser side-force and therefore increase the force too much and spin out. The same situation could occur at the 0.15 G limit. Holding a somewhat constant lateral force of 0.14 G may in some situations make the simulator increase lean from 0.07 G to 0.14 G. The driver might attempt to compensate for this and then re-compensate for the compensation which could result in some odd behaviour.

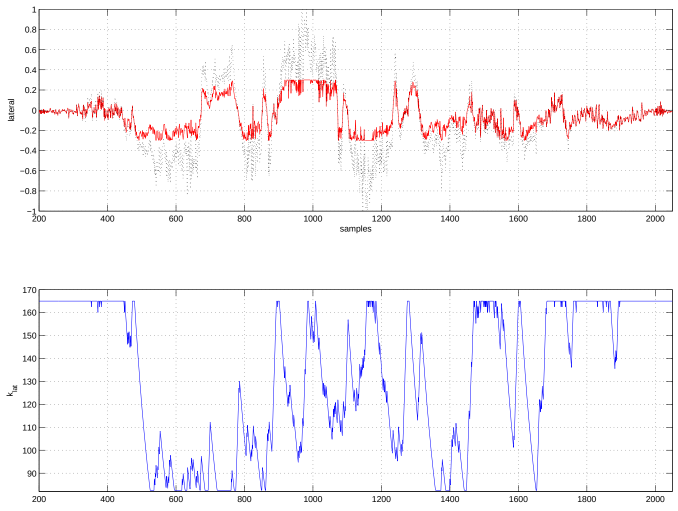

Figure 6.7: How the actual measured lateral G force (dotted black) is scaled down (solid red) with equations 6.5 to 6.7 and the if structure introduced in section 6.3.5 with $K_{lat,minus} = 0.985$ and $K_{lat,plus} = 0.03$. The second plot shows the change of the factor $k_{lat}$ with these equations. $k_{lat}$ falls from 165 to 83 in 1.9 seconds and rises from 83 to 165 in 0.7 seconds. Scale 25 samples per second.

### 6.3.6 Vertical acceleration

On the track at Torpa there is one considerably large downhill slope followed by an uphill slope. When the car accelerates along this straight Racer will produce vertical G-forces that may be desirable to experience. The vertical acceleration in Racer is much less noisy than the measured vertical acceleration data in figure 6.3 so it is possible to include it. The problem is that it will be hard to add vertical forces for the whole length of the downhill. For this reason it can be determined that the vertical acceleration only plays a part if it rises above a certain threshold which is only broken at these few points on the track. For example, a coefficient $k_{vert}$ could be added to each piston equation, and just like in section 6.3.5 it will rapidly increase above a threshold and slowly decrease below it.

```
if $(a_y > 1.2)$
    $k_{vert} = k_{vert} - K_{vert,add} \cdot (a_y - 0.2);$
elseif $(a_y < 0.8)$
    $k_{vert} = k_{vert} + K_{vert,add} \cdot (1.8 - a_y);$
else
    $k_{vert} = k_{vert} \cdot K_{vert,mult};$
end
if $(k_{vert} < k_{vert,min})$
    $k_{vert} = k_{vert,min};$
elseif $(k_{vert} > k_{vert,max})$
    $k_{vert} = k_{vert,max};$
end
(continued with e.g. equations 6.5 to 6.7 with the factor $k_{vert}$ added.)
```

Where $K_{vert,add}$ is chosen large enough to generate enough vertical G-forces, $K_{vert,mult}$ is chosen below 1 to slowly make $k_{vert}$ smaller and the limits $k_{vert,min}$ (around -50 is a good figure) and $k_{vert,max}$ (around 50) should not be reached.

### 6.3.7 PID-controller

The PID-controller takes the reference value produced by one of the equations and compares it to the actual piston position calculated with equation 6.4. The whole algorithm, which includes methodology to take care of reset windup, is for the front-right piston:

```
$e_{fr,old} = e_{fr}$
$e_{fr} = r_{fr} - y_{fr}$
if $u_{min} < v_{fr} < u_{max}$ then
   $I_{fr} = I_{fr} + K_I \cdot e_{fr}$
else
   $I_{fr} = I_{fr}$
end
$v_{fr} = K_P \cdot e_{fr} + K_D \cdot (e_{fr} - e_{fr,old}) + I_{fr}$
```

$$u_{fr} = \begin{cases} u_{max}, & \text{if } v_{fr} > u_{max} \\ v_{fr}, & \text{if } u_{min} < v_{fr} < u_{max} \\ u_{min}, & \text{if } v_{fr} < u_{min} \end{cases}$$

$u_{fr,out} = u_{fr}/20$
where $y_{fr}$ = measured position (0 to 100)
$r_{fr}$ = reference position (0 to 100)
$u_{fr}$ = output position (0 to 100)
$u_{fr,out}$ = output value (0 to 5 Volts)

Estimates for the $K_P, K_I and K_D$ values can e.g. be obtained by using the Ziegler-Nichols method (described in Glad, Ljung et al, 2003) on the pistons.

These equations rely on the Aprot program to execute the loop at a constant rate. Due to windows not being a real-time system this won't be the case. This will result in the derivative and integral parts being the wrong size. The PID-controller relies on the loop being sufficiently constantly executed.

If it would turn out that the program is too slow for the PID-controller or that it is too unreliable in terms of inconsistent timing as explained above, external controllers would have to be used. The Aprot program would then either A) output a voltage representation of the error or B) output a voltage representation of the reference value (this option would mean that specific knowledge of the min and max values of each individual potentiometer need to be known).

# Chapter 7

# Pneumatic motion board

The motion board was used to test the interface with a physical system before constructing a full scale simulator. It was also used to see if it is possible to use pneumatics instead of hydraulics.

## 7.1 Motion board setup

In this case the four piston setup was used, which has one piston in each corner, each piston representing an imagined wheel/suspension. This setup was used because it is the intuitive one. And may give a more stable experience. The air flow into the pistons is controlled by four valves which can be shut on and off by four relays controlled from the interface card, meaning no D/A converters are needed. Since the valves are driven by a 240 Volt, 50 Hz AC voltage, the relays are triggered on the zero-passages in the signal (a common AC voltage relay property). This means that the relays have a maximum on/off performance of 100 Hz. Each piston is connected to a potentiometer which gives a 0-5 V signal. A plate mounted on the top of the piston and the top of the linear potentiometer connects them. The software operates at approximately 100 Hz (this cannot be guaranteed since Windows isn't a realtime system) and includes a PID controller.

## 7.2 Test results

There were a couple of problems that arose during the test. The pistons reacted much faster going up than going down. The approximate risetime from 0 to 100 (full) was 135 ms without external weight. The corresponding falltime was 430 ms. The only factor that caused the pistons to fall was gravity. An attempt to balance this out by adding weight on the pistons reduced the risetime to 260 ms. The falltime was still 430 ms. Adding more weight

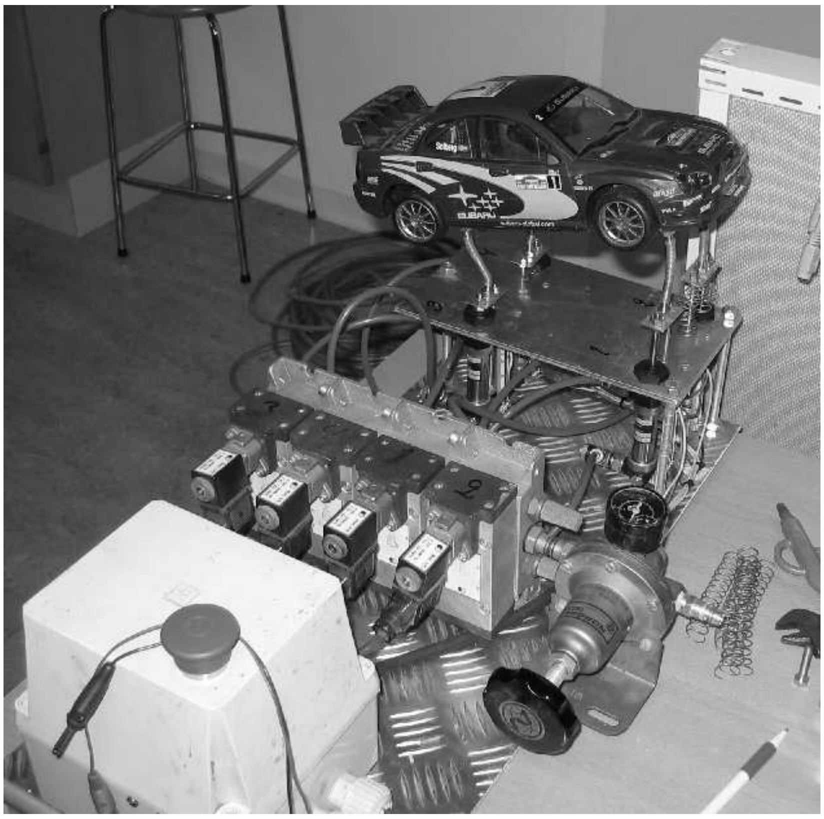

Figure 7.1: Motion board.

caused more trouble, namely the inconsistency of air caused by compression. The weight was too much for the current air pressure (about 1.4 bar) to be able to push up the weight. A slight increase in air pressure (0.1 bar) again caused the piston to rise very fast. Add to this the pistons being individually not identical in their behaviour, and the conclusion is that using air will be hard.

Letting the relays be controlled by a PID controller and measuring position as well as updating relays at 100 Hz frequency, there was some success. The control equations were:

$$e_{fr}(t) = r_{fr} - y_{fr} \tag{7.1}$$

$$I = I + K_I * e_{fr}(t) \tag{7.2}$$

$$ur_(fr) = -sign(K_P * e_{fr}(t) + K_D * (e_{fr}(t) - e_{fr}(t-T)) + I) \tag{7.3}$$

$$e_{fr}(t-T) = e_{fr}(t) \tag{7.4}$$

where fr = front-right piston The negative sign on the relaysignal depends on the card having open collector digital output pins. This means that sending out a low signal (ur $<$ 0) corresponds to the relay being on, and driving the output high (+5 V, ur $>$ 0) turns the relay off.

After much tweaking of the parameters the best result when desiring a position of 50 was that the piston would oscillate between 40 and 60, 20% of the whole piston length, at a frequency of about 20 Hz (see figure 7.2. The coefficient values were $K_P = 0.3, K_D = 5, K_I = 0.02$.

Theoretically you would expect the pistons to be able to oscillate at an amplitude of at most ”relay period / full piston movement time”, where full piston movement time is 260 ms and a fair relay period is 20 ms (10 ms, corresponding to 100 Hz, is not possible to guarantee). This means that the pistons should be able to have an amplitude of 8% when oscillating. Even 8% is a big number, and if you consider four pistons moving individually they would create a shaky experience, even when it's standing still at the finish line.

Another interesting effect was the amount of samples the relay was on compared to the amount of samples that the relay was off. Since the pistons have a faster risetime than falltime one would think that they would be off more than they are on. But, actually, counting over 640 samples it turned out that the relay was off for 262 samples and on for 378 samples. A significant difference that could depend on many factors, such as the air compressing slower when the piston isn't in position 0 due to it having more space to compress in.

### 7.2.1 Evaluation

The conclusion from this experiment was that pneumatics is too hard to control and too inaccurate to be used in a full-scale model. Because of it's high

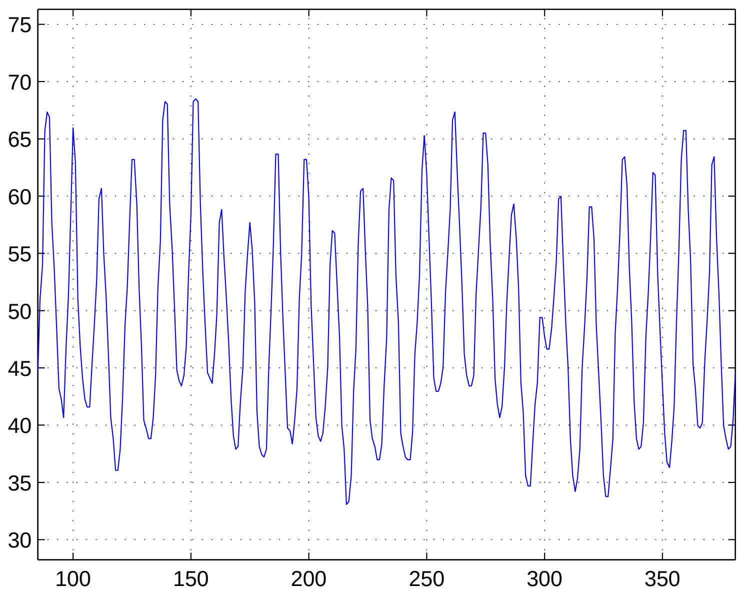

Figure 7.2: Measurement of one piston when trying to center it at 50 with a PID controller.

compression rate, it is almost impossible to steer an air piston into position by adjusting the pressure. Relay control is possible, but will make for a *very* bumpy ride. There are no professional (to the author's knowledge) industrial applications where pneumatics has been used to control piston positions between two values with decent precision.

# Chapter 8

# Future work

The most obvious future step based on this work is to use the formulas and evaluate the experience in a real simulator. There are also some other things that can and should be tested before implementing on a motion simulator:

- Test how well Racer and Aprot run together on a modern dual-core processor PC. Focusing on the frames per second in Racer to be reasonably high while the performance of Aprot (especially how fast the loop runs) is the best possible.

- Create a copy of the track and a car used at Torpa in Racer. Compare the G-forces from the simulation to those measured on the track. Optimize the equations (coefficients) based on the new data.

- Simulate the behaviour of the system. For this part only one piston could be necessary. Create a model in Matlab for the non-linearity of the valve-flow and for difference between risetime and falltime (falltime is probably shorter) and see how well a PID controller performs with these limitations.

Once this is done some things that can be done on the motion simulator itself are:

- Evaluate whether or not the pistons can be fixed vertically. If not, fix the pistons in such a way that the angle to piston extraction ratio is as linear as possible.

- If necessary modify the formulas to eliminate the non-linearity of a new design, valve-flow dead-zone characteristics and general valve-flow characteristics. One idea is to use the formulas as they are, but to have a lookup table to convert the calculated position to something appropriate as output signal.

- Evaluate the PID-controller. Use for example a Ziegler-Nichols test to get decent coefficients for the the proportional, derivative and integral parts.
- Try out some different sets of equations. Evaluate the experience and choose the most satisfying set. Perhaps include an option to have some coefficients change between runs for smoother or rougher rides.

# References

Lennart Ljung, Torkel Glad, Svante Gunnarsson, Tomas McKelvey, Anders Stenman, Johan Löfberg (February 2003), *Digital Styrning Kurskompendium* (in swedish)

Racer website, *Overview of features in Racer* [www]
http://www.racer.nl/overview.htm, accessed december 2007.

Racer website, *Links to documentation about the physics used in Racer* [www]
http://www.racer.nl/links.htm, accessed december 2007.

*Force dynamics simulator* [www]
www.force-dynamics.com/content, accessed december 2007

VTI Linköping, *driving simulator* [www]
http://www.vti.se/templates/Page____3258.aspx, accessed december 2007

Labview website. [www]
www.ni.com/labview

# Notation

Symbols used in the report.

## Vocabulary

Explanation of words and phrases in this document.

| | |
|---|---|
| API | Application Programming Interface (high level programming language) |
| Motion board | Model of motion simulator, used to test piston setup and control algorithms |
| G-force | $Accelerationasitisexperiencedbyaperson(independentofmass).1G \approx 9.8m/s^2$ |
| Fraps | A program that measures the amount of frames per second that games and other software are run at. Good fo |

## Variables and parameters

| | |
|---|---|
| $g$ | Gravitational constant (9.82 $m/s^2$) |
| G | G-force (acceleration/$g$) |
| $a_x$ | lateral acceleration |
| $a_y$ | vertical acceleration |
| $a_z$ | longitudinal acceleration |
| $\theta$ | pitch angle |
| $\psi$ | yaw angle |
| $\beta$ | roll angle |
| $r_{index}$ | reference values |
| $y_{index}$ | measured position (0 to 100) |
| $ur_{index}$ | signal to relay (1 = off, 0 = on) |

Note that the indexes are fl for front-left, fr for front-right, rl for rear-left and rr for rear-right.

## operators

| | |
|---|---|
| $\succ$ | Succeeds. |

## .1 Full data series

Here is the full data series collected from the Torpa racing track, including the areas of uncollected data. The 6000 samples represent three laps of driving.

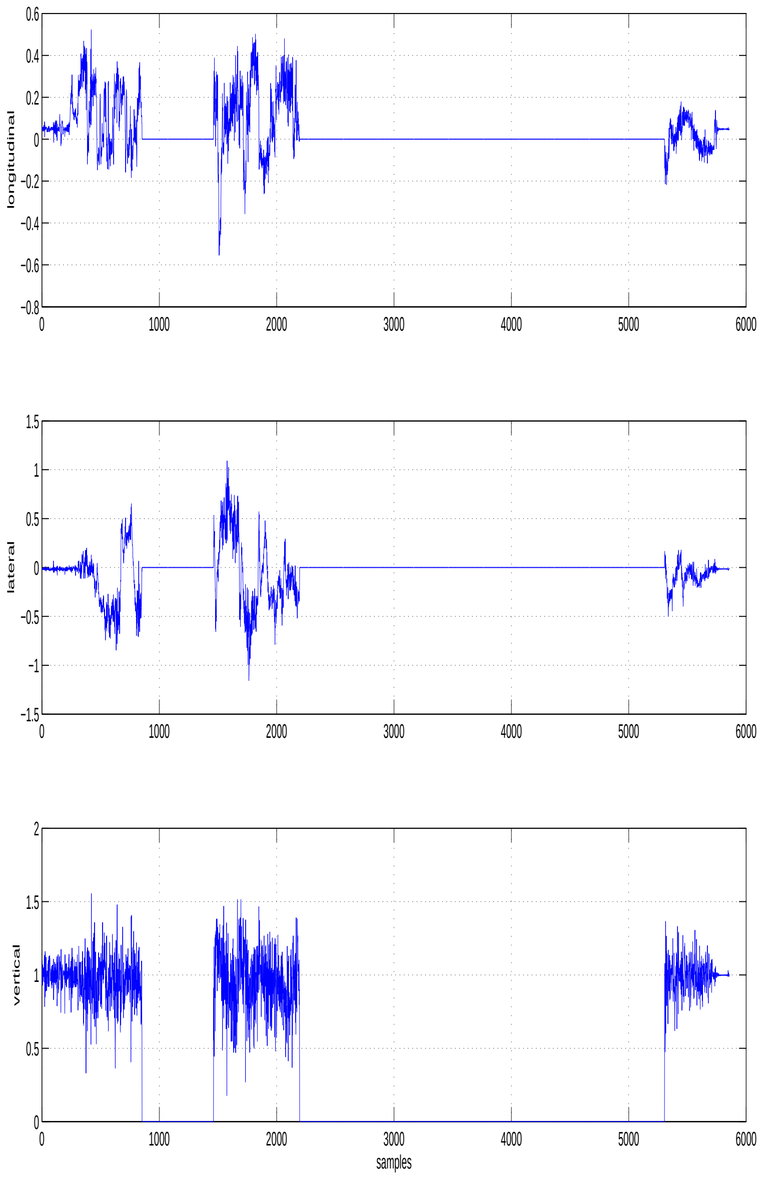

Figure 1: Full data series from torpa racing track

# Copyright

## Svenska

Detta dokument hålls tillgängligt på Internet - eller dess framtida ersättare - under en längre tid från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida: `http://www.ep.liu.se/`

## English

The publishers will keep this document online on the Internet - or its possible replacement - for a considerable time from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for your own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its WWW home page: `http://www.ep.liu.se/`

© Ville Grandin
Linköping, 14th December 2007