

MASTER'S THESIS

Graphical User Interface for Scania Truck And Road Simulation

Johan Thorén

Civilingenjörsprogrammet

Institutionen för Maskinteknik
Avdelningen för Fysik

Graphical user interface for Scania Truck And Road Simulation

by

Johan Thorén



A Matlab Master's Thesis
The Institution of Physics
Luleå University of Technology

Summary

This report is the result of a Master's Thesis to construct a graphical user interface for Scania truck and road simulations (STARS). The report can be divided into four major chapters.

The second chapter, Graphical User Interface, describes the general ideas for implementing the GUI. The design of the GUI is described and references to literature studies are made. Resulting options and finesses within the GUI are exemplified. The structure of the code for the program is also generally described.

The built in "Gradability" function in the GUI is described in the third chapter, Simulation of vehicle on road. The mechanical theory for rolling resistance for wheels, slip, aerodynamic forces on the truck and calculations for gradability are documented.

Numerical algorithms were used for calculating the Dymola simulation model. The fourth chapter, Numerical algorithms in Dymola, explains the background theory for Euler's and Runge-Kutta's method. A validation is made to find the most accurate method to use and the restrictions for the calculation are presented.

Results from the Dymola simulation model had to be evaluated. The fifth chapter, Simulation results calculations, explains calculations for finding find results about exhaust emissions, fuel consumption etc after the simulation. Statistical evaluations are also documented.

1 INTRODUCTION	4
1.1 PURPOSE	4
1.2 RESULTS	5
2 GRAPHICAL USER INTERFACE	6
2.1 BACKGROUND IDEAS FOR THE GUI.....	6
2.2 LAYOUT	7
2.2.1 <i>Two strategies of thinking</i>	8
2.2.1.1 Application of strategy thinking in the GUI.....	8
2.2.2 <i>Analogies</i>	9
2.2.2.1 Application of analogies.....	9
2.2.3 <i>Active thinking</i>	11
2.2.3.1 Applications of active thinking	11
2.2.4 <i>Colour coding</i>	12
2.2.4.1 Applications of colour coding.....	13
2.3 IMPLEMENTED CODE	14
3 SIMULATION OF VEHICLE ON FLAT ROAD	16
3.1 VEHICLE DYNAMICS.....	16
3.2 WHEEL THEORY	17
3.2.1 <i>Rolling resistance</i>	17
3.2.2 <i>Slip in relation to tractive force</i>	18
3.2.3 <i>Tractive effort and power train</i>	20
3.3 CHASSIS BODY THEORY	22
3.3.1 <i>Aerodynamic resistance</i>	22
3.3.2 <i>Aerodynamic lift</i>	23
3.4 SIMULATION OF VEHICLE IN TERRAIN.....	23
3.4.1 <i>Gradability</i>	24
3.4.1.1 Evaluating the force of rolling resistance.....	24
3.4.1.2 Evaluating the force of air resistance.....	27
3.4.2 <i>Gradability calculation</i>	28
3.5 DISCUSSION	28
4 NUMERICAL ALGORITHMS IN DYMOLA	30
4.1 TAYLOR SERIES	30
4.2 EXPLICIT EULER METHOD	31
4.3 IMPLICIT BACKWARD EULER METHOD.....	33
4.3.1 <i>Call Dymola using Eulers method</i>	34
4.3.2 <i>Global error and step length control</i>	36
4.4 RUNGE-KUTTA	34
4.4.1 <i>Call Dymola using Runge-Kutta</i>	36
4.5 DISCUSSION	37
5 CALCULATIONS ON RESULTS	38
5.1 EMISSIONS AND FUEL CONSUMPTION	38
5.2 ENGINE STATISTICAL TIME	40
5.3 ENGINE STATISTICAL FUEL	41
6 REFERENCES	42
7 APPENDIX	43

1 Introduction

Scania is building a new, modern simulation program called STARS, Scania Truck And Road Simulation. Simulations are an important part of the development process for trucks. The main goal is to virtually run trucks in a computer and receive results about emissions CO_x, NO_x, HC and fuel consumption etc. Running the tests in a computer is more economical as it saves time compared to real tests.

The simulation model was built with the software Dymola. With Dymola one can construct a model of a physical and mechanical process. The model is compiled to a file, which can be controlled. To make simulations easier to use and more efficient it is important to have a well functioning Graphical User Interface. Matlab has the power of handling large amounts of data and performs necessary calculations and is therefore a good platform for a GUI.

1.1 Purpose

This Master's Thesis is about constructing a Graphical User Interface for STARS in Matlab. The GUI is the easy-to-use platform from where the user runs simulations. During the simulation a separate simulation-file is used. The simulation-file was constructed with the program Dymola. The GUI communicates with this file and presents results for the user. Below the specifications for the most important contents of the GUI are listed.

- The GUI should handle all necessary preparations before the simulation. Different road profiles, engines and gearboxes are optional to the user. Changes to truck specific parameters, e.g. weight of the vehicle, air resistance coefficient etc should be possible.
- Save data and parameter values in specific files, which the simulation-file uses. The GUI should also be able to handle a list of separated data. In this way a number of different simulations can be prepared and the simulations can be run during the night.
- Present characteristic diagrams for the roads, engines and gearboxes within the GUI.
- The numerical calculation during the simulation is controlled. A number of different algorithms are available. Make an investigation of these and use the most accurate as the default algorithm for Stars GUI.
- Display the simulation results in proper diagrams, e.g. statistical truck performance, and print out information about emissions, fuel consumption.
- The GUI should have a Gradability function for simulating a truck climbing hills with different angles at a steady speed.
- If changes are to be made to the GUI, instructions should be documented.

1.2 Results

A list of the results of my Master's Thesis

- A final GUI was created for the purposes described above. It is ready to use and can handle a simulation and take care of the results.
- A Gradability function was derived and implemented in the preparation part (before one starts the simulation) of the GUI. This function studies what slopes a specific truck can handle at a steady speed. A Slopability function was also coded, which is the opposite of Gradability.
- Runge-Kutta was found to be the most reliable numerical algorithm that Dymola offers. It has also been chosen as the calculation algorithm for the STARS simulation.
- The GUI can be extended with more contents, e.g. new engines and visualising diagrams. Documentation on how extensions are made to the program is described in this report.

2 Graphical User Interface

“If the user can’t use a GUI, it doesn’t work” – Susan M Dray

In order to run truck simulations I was assigned to create a Graphical User Interface. The GUI is an illustrative platform and the co-ordinator between the data that the user chooses and the simulation. The user should, before starting a simulation, have the opportunity of choosing a complete truck and trailer and decide all the important parameters, e.g. objects like engine, drive train, tyres and weight of the vehicle and a number of other specific parameters. The GUI should also be able to start the simulation properly and deliver the results needed. A number of different road-profiles for the simulation can also be selected. One can see the GUI as a virtual workbench for handling truck simulations.

The GUI was created with Matlab. Benefits can be drawn from all the powers that Matlab supplies, which helps to handle the amount of data. The users are mainly concerned about the simulation results on performance of the vehicle, meaning that fuel consumption, exhaust emissions, simulations speed etc should be calculated and presented. This chapter is about a literature study of the topic of creating an efficient GUI and describes how I used these theories.

2.1 Background ideas for the GUI

There are several demands on a simulation GUI. An investigation of the problems facing the GUI showed the following ideas before starting working on the layout and implementing the code.

- A fast operating GUI is preferred since slow loading of data and long time of simulation are irritating to the user.
- A nice visual appearance of the GUI is important. A boring and dull interface will not make the user eager to prepare the simulation properly.
- Easy preparation and efficient simulations are essential for a well functioning GUI. A regular computer user and Scania employees should be able to handle the necessary preparation before starting the simulation.
- All demands the user asks for the GUI should be able to accomplish. E.g. print text information and plot informative data which the user needs.
- Object oriented code should be applied, meaning parts of the GUI are replaceable. This could be handy when the simulation is improved and other extensions to the program are needed. Object oriented code can also be reused, which is efficient for the programmer.
- The GUI must be universal. One must be able to download the program and run it on any computer. Assumed is that the workstation has a Matlab version of 5.3 or later
- A known computer environment is easier to work with. The user should recognise as much as possible of the graphical symbols and appearance the GUI from a regular window environment.

As one can see both ideas for making the looks of the GUI (layout) and structure of the program (coding part) has to be dealt with. The layout is what the user actually sees when running the program. The coding part on the other hand is the hidden part of the program. The regular user will never deal with this part but a more advanced user must be able to add new functions when needed and make changes (if necessary).

2.2 Layout

The layout is what the user can see with his eyes when running the program, the buttons and options one can choose. It also includes information that the user needs in order to be updated on the simulation status.

The program is divided into two parts. The select-part where the user sets all objects and parameters before the simulation. An after-treatment-part where the simulation results are presented. Dymola simulation, where all calculations take place, separates the two parts.

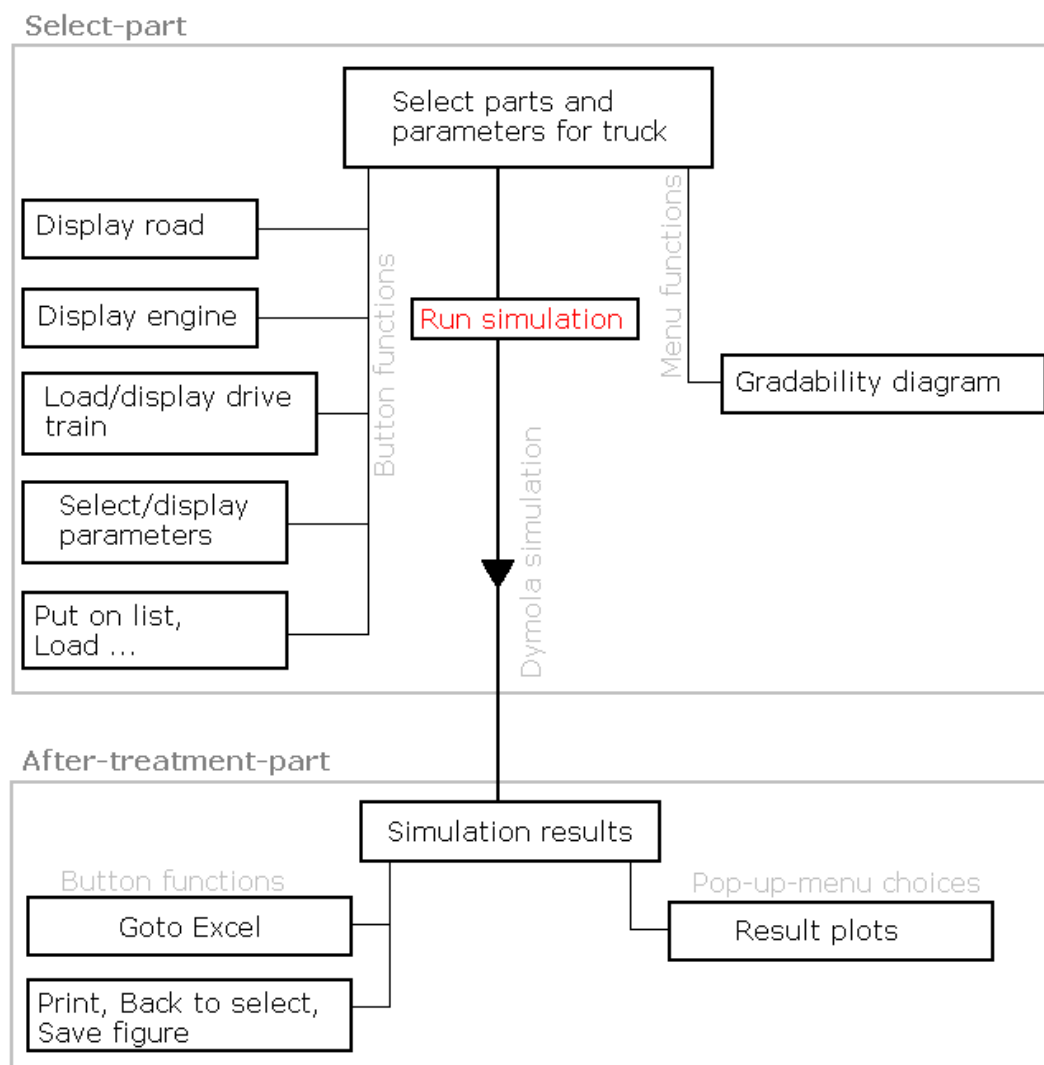


Figure 2.1 *Navigation map of the windows in the GUI. Notice that the GUI is occupied during the “Dymola simulation”.*

2.2.1 Two strategies of thinking

According to the literature that I have studied, (see reference list), there are some general characteristics about the human-computer performance one should keep in mind. The human thinking process is dependant on a number of factors, e.g. social, motivation and biological abilities. The interaction of these will result in the human-computer performance. A person turns on the GUI and smiles to the nice layout of the program. He then easily prepares and runs a simulation successfully, receives the results that he needs. He will also be motivated to run a new simulation knowing how efficiently it worked. Maybe he will also tell his friends how great the program is and recommend it.

Assume one can measure the social, motivation and biological abilities and sum them up. The result can be referred too as the pre-knowledge of all the background experiences the person has facing a new problem. This is generally speaking the most important factor affecting the thinking process of a person, therefore an important aspect when creating a GUI. This study was primarily done regarding what the GUI should be able to handle. The pre-knowledge of the users has been used in order to find what the GUI should consist of. Through discussions with staff at Scania and ideas from other simulation programs the content of the GUI was created. No questionnaire studies have been done on the actual users of the program. Literature studies have been done to produce a GUI that is user-friendly.

Pre-knowledge affects how a person works with a problem. If the individual has a poor knowledge in the actual area, he will have difficulties with preparations before actually taking on the assignment. E.g. a person who has not been developing engines will have to deal with fact-terms (that he probably never heard of) before fitting a suitable engine to a specific truck. This kind of user has no good experience for solving a simulation problem. A person with bad pre-knowledge will try to solve the assignment with what is referred to as *general strategies*. The biggest disadvantage with general strategy is the low efficiency of bringing the user closer to a result. If a person on the other hand has a wider experience in the topic of simulating trucks he will be using what is referred to as *specific strategies*. Specific strategies mean that the user uses his knowledge in order to get to a result. This will help the user to be more efficient.

2.2.1.1 Application of strategy thinking in the GUI

In the Stars GUI one has to be familiar with the problems of simulating trucks. All users of the program have seen simulations of trucks before and run simulations. The user knows about problems such as properties of the engine, drive train, rolling resistance of the wheel and air resistance to mention a few. Since the user is familiar with the parameters that can be changed he probably will be looking for them in the program.

Therefore it was decided that the GUI should be concentrated to as few windows as possible. As much information and options as possible should be fitted to each window without making a messy impression. The greatest benefit with this approach is that the user will find it easy to find information faster and make changes efficiently. Navigation in the GUI has also been made easy when all windows are reachable with only one click of the mouse. The most important parameters and information for the simulation are therefore showed directly.

Additional information and adjustments options can be displayed by clicking the mouse one time.

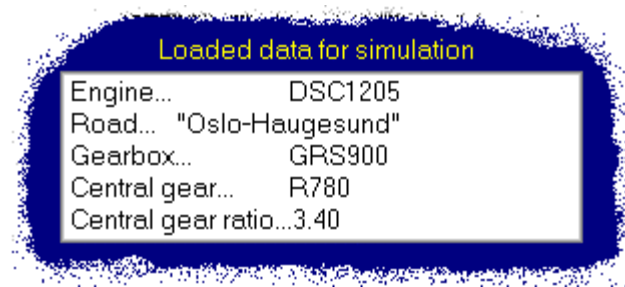


Figure 2.2 *Information textbox in the GUI. A common user is assumed to be familiar with these objects.*

2.2.2 Analogies

The ability of associating to analogies is powerful. One feels more at home in a known environment. In the literature it is referred to as analogy thinking. Analogy thinking is according to Ref. (3) an important characteristic of the mind that should be considered when constructing computer programs. An understanding of the pre-knowledge (see chapter 2.1.1) of the person is vital to apply the right analogies. The user mostly draws conclusions from what he already knows. The source from where he got his knowledge is called the *origin domain*, in this case old truck simulation programs and the Windows environment. The benefits of analogy thinking are taken in what is referred to as *application domain*, in the Stars GUI.

Analogies are often used when making computer programs. E.g. the most accustomed analogy is the word processor. These were built from the knowledge of the typewriter. The similarities are helpful to get the user started and errors appear less often. Research has also shown that many of the problems that occur to the user have their origin from pre-knowledge about the typewriter. E.g. the spacebar works differently. While the typewriter simply will shift one step when the spacebar is pressed the word processor pushes all the text to the right.

2.2.2.1 Application of analogies

Using analogies in a GUI will take advantage of a person's pre-knowledge of computer programs. The look of the program has therefore been adjusted so the user should recognise parts in the program. E.g. the look of loading procedures has been taken from Windows, a program that the user is familiar with (see figure 2.3).

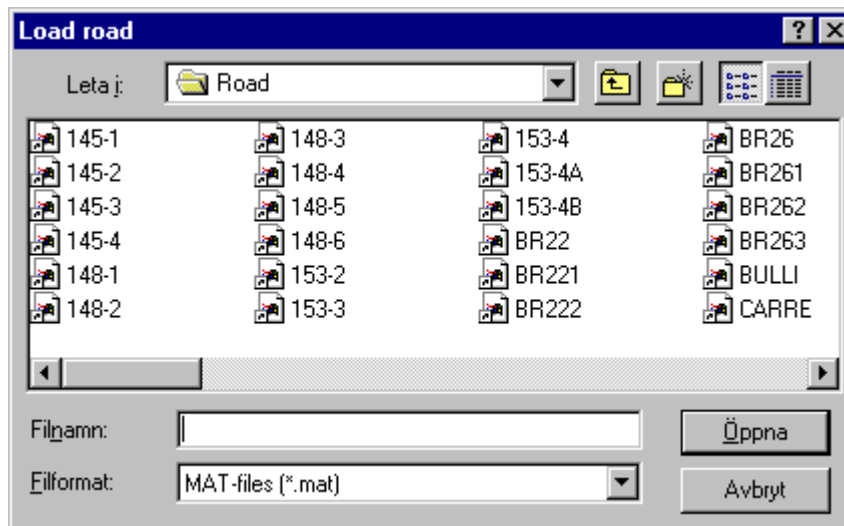


Figure 2.3 The same loading figure is used in the GUI as in other windows applications.

In accordance to other Matlab GUIs a button function also has a menu-bar-control in the top of the figure, which executes the same command. Commands can therefore be run both by button pressing and menu choosing. Exceptions have been made for seldom-used commands that only should be used by advanced users. An example of this is the function “Save as new standard”, which also is viewed in figure 2.4.

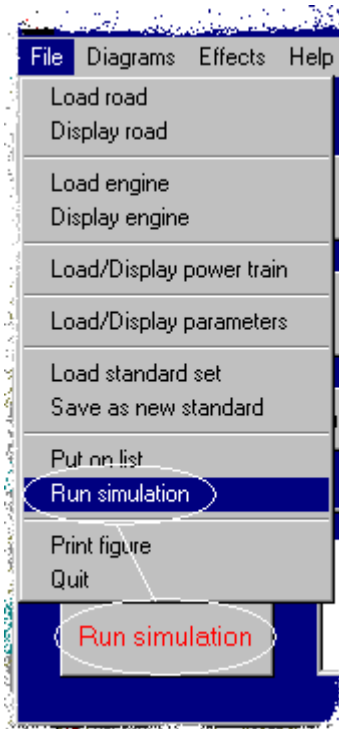


Figure 2.4 See how the button function “Run simulation” also has a menu bar function for the same purpose. The button has also been given a red colour since red is the common execution colour in analogy with e.g. CD-burner programs. Read more about this in chapter “Consequence in colours”.

Some restrictions have been made in the analogies as well. The menu figures look the same as a regular Matlab figure. The resize function in the right corner has though been deactivated (see figure 2.5). The reason for this is that the resize function makes the layout of the figure messy. The figure has also been adjusted so that they can be printed on a regular printer. The closing function is working as normal, but has also been moved to the menu. Other GUIs usually closes with a specific choice from a menu and accordingly the Stars GUI was chosen to work in the same way.



Figure 2.5 *The regular resize in the up right corner has been taken out of order. The minimisation button and closing button still work though.*

2.2.3 Active thinking

The ability of thinking active is an issue that is often discussed in the literature. Active thinking leads for instance to an individual creation of hypotheses about situations, which then are compared to the actual problems. The hypotheses are based on the pre-knowledge and former experience of the individual. This results in the individual creating a more or less true picture of himself and the surrounding world. The psychologist Bartlett expressed a similar thoughts in the thirties. A human has an active quest for finding meaning and connections.

One should keep in mind that humans mostly have a tendency of finding explanations from faulty information. Humans are often to large extent speculative. An answer, no matter how informative, is better then no answer at all.

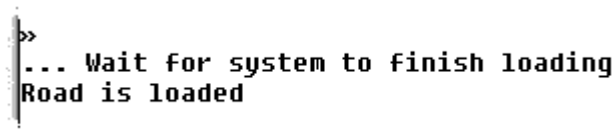
This relation shows in the Human-Computer-Interaction. Many studies have proven that beginners form explanations of how a program works from rather poor assumptions. These explanations are mostly incorrect. Beginners ability to create explanation from a weak background knowledge is important to take into account for a program-constructor.

Apart from drawing conclusions from poor information persons validate information differently. Some information will be more important to the user and other information the user will not pay that much attention to. E.g. a red-flashing warning text will surely be more valuable than a hand-written papernote.

2.2.3.1 Applications of active thinking

The active thinking is not an easy task to implement in a GUI. From the literature study above I came to some conclusions that I thought could be efficient. In the list below I explain the ideas that I personally implemented in the GUI.

- Every pause when running the program is always commented in the Matlab prompt. E.g. when loading a road file the user will have to wait for a couple of seconds. During this time the user can follow the system status when different text lines are written out. See figure 2.6. The user will in this way hopefully not draw conclusions that e.g. the system has crashed or try to run new options before the system is ready. In this way I thought that the user should be well informed and have a better probability of completing the options successfully. Since users can validate information differently, depending on where it is presented, I chose the Matlab prompt windows. The main reasons for this is that new figures during loading would require more loading time and the Matlab windows is the first place the user hopefully looks when his expectations are not fulfilled.



```
» ... Wait for system to finish loading
Road is loaded
```

Figure 2.6 Example of how information on the system status is displayed in the matlab prompt. In this example the button “Load road” was pressed.

- Every figure in the GUI has a headline, which explains the reason for the window. According to the “rule” that some information is better than none, see chapter 2.1.3, I tried shortly to tell the purpose of the figure to the user. The normal blue banner at the top of a Matlab figure was chosen as the appropriate place for this information. See figure 2.7 for how this looks in the GUI.



Figure 2.7 Example of how a figure in the GUI is always presented with a headline to give the user a hint of what to do.

2.2.4 Colour coding

In Matlab one has the ability of putting colours to the figures. From literature studies I came up with the following benefits and limitations of colour coding. From these ideas I then coded the colours in the GUI, which is further described in chapter 2.1.4.1.

Benefits:

Colour stands out from a monochrome background, which means that colour-coded targets are rapidly and easily noted. Colours can therefore be used as a means of highlighting an important item on a menu display.

Colour coding can act like a preattentive organising structure to tie together multiple elements that might be spatially separated.

It can also work as a redundant coding device.

Colours are also attractive and aesthetic.

Limitations:

Colour is subject to the limits of absolute judgement. Colours can be misidentified, especially if they are to be used in darkness or in poor illumination. Therefore no more than five to six colours should be used.

Colour does not naturally define an ordered continuum. It is impossible to for example define “least” and “most” with colour coding. Colours do not have a strong popular stereotype.

Despite what is mentioned above, there is an association of specific colours with specific meanings. Red, for example, means “danger”, “hostility” or “stop” for many people. This is both a strength and a weakness in colour coding.

Irrelevant colour coding can be distracting.

2.2.4.1 Applications of colour coding

The colours in the GUI is one important factor that either motivate the user or make him tired. Keeping in mind that a certain amount of continuity should be kept to the colour settings, as this helps the user to be navigated through the program, I decided to make a coding as listed below.

- The background is dark blue through all figures of the program. It looks nice and warm and does not make the eye tired. This is also one of few colours that Matlab can handle on different computers without adding some shade. - Clearly a bug in the Matlab program. Making hardcopies on a colour printer is not a problem since the background is printed as white. Black and white printers are adjusted to also deliver the same white background on hardcopies.
- Headlines to all options and information boxes in each figure are set to a yellow colour. Light colours on a dark background are generally accepted as the most easy-to-read colour. Yellow and dark blue together make a fine contrast and still a satisfying combination.
- Information about the variables and parameters are written with a black colour on a white background. This also includes the edit boxes where the user can alter parameters with the keyboard. The reason is that, edit boxes also work as information boxes with the current information displayed. White makes a sharp edge to the dark blue background colour. To diffuse the edge a little I made light grey frames around all white information boxes. The boxes will in this way look similar to islands on a world map scheme.
- Diagrams in the GUI are presented on a white background. Generally the most common way of displaying diagrams in Matlab. An alternative would have been light grey, but I decided that it look dirty, especially when they are printed as a hardcopy, which also would demand a large consumption of grey tone. Labels on the axes are written with a light grey colour. Light grey is easy to read both in the GUI on the dark blue background colour and on hardcopies.
- Buttons were made grey. Other GUIs often use this colour and importantly Windows does as well. All texts on the buttons are made black for the same reason. Exception has been made for the button “Run simulation”. This is an execution button, which is not returnable without losing information. The colour red was chosen in analogy with e.g. CD-burner programs, where the burning procedure starts when pressing the red-dot button. Red is also a colour that is associated with warning and precaution, e.g. Stoplights when driving a car. It was intended in a similar way that the user should stop and see that all preparation was ready and correct before pressing a button with red text.

The choice of colours was restricted by the limitations from Matlab. Since the GUI must work both on all computer screens and printed as hardcopies, I had to adjust the program according to the circumstances. Other combinations were tested but proved not to be viable. The resulting combination described above is a functional set of colours and still not simple.

2.3 Implemented code

All code to the program was written in the Matlab editor (edit.m) and saved as Matlab m-files. Without being too specific this chapter will explain the general structure of the GUI coding.

The code was written according to the basic layout with a main program connected to each figure. E.g. the button “Display road” calls on the function disproad.m. All options within the figure g41.fig (Display road figure) are then dealt with in the disproad-file. A scheme of the connections between different program files can be viewed as figure 2.8. Comments are also written in the code helping to get information on specific run syntax.

The figures in the GUI are saved with “callbacks” (see Matlab manual for more information). Generally one can say that a callback is a line that executes a certain command.

From this code layout, I have tried to use general routines as much as possible. These routines are called to with certain syntax from many of the m-files within the program. The gain is that the total number of code lines becomes less. Typical general commands are printz, cdz, zoom1, clabel1, load_mat and quitz1. All these are m-files and similar to their standard relatives in Matlab, but have been adjusted to accomplish the demands of the GUI. Comments to the code for these commands can be listed by typing help command-name, just as for a regular Matlab command.

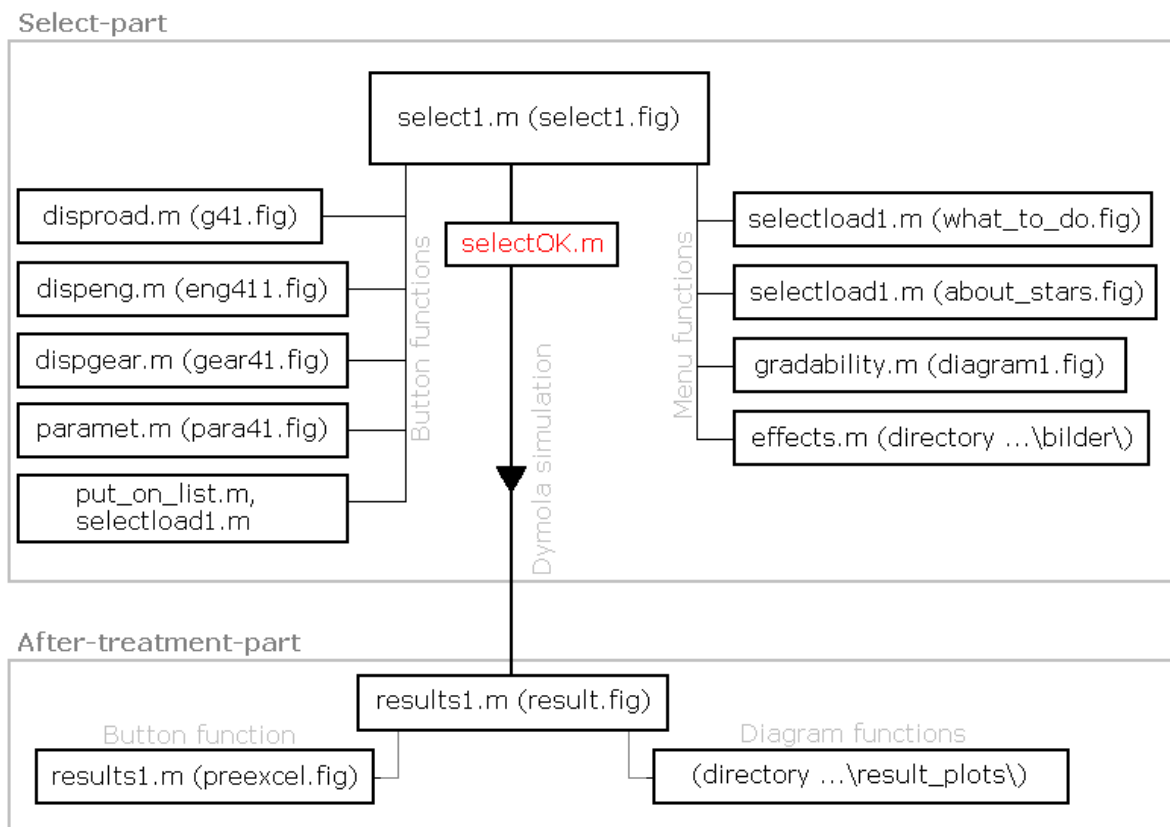


Figure 2.8 The implemented code and -.fig names scheme. Compare with figure 2.1 for more information. For more information about the diagram functions “result_plots” see user manual chapter XX

3 Simulation of vehicle on flat road

This chapter explains the different properties of forces acting on a vehicle. Fundamental studies of a driving vehicle are done on a flat road. In the longitudinal direction the vehicle body is affected by specific forces. Aerodynamic and gravity forces acting on the vehicle are put in relation to the tractive and braking forces applied by the driver. The movement of the vehicle can then be simulated. (A curved road model applies side forces to the vehicle. These will be neglected in this study.). Since part of my assignment was to create a simulation of a truck on a sloping road some investigation of the dynamics of vehicles will be described. How this is connected to what later is referred to, as gradability of a truck, will be described in chapter 3.

Knowledge of how a vehicle behaves on a road is commonly achieved by running tests. This is essential in order to find out and test the characteristics of a vehicle. Simulation on the other hand is a good help when a vehicle should be tested in a number of different versions. With the help of computers and simulation programs one can make accurate predictions of vehicle performance. This could be helpful to fine tune and improve an already produced vehicle. The major benefits can be drawn from lower costs in the development process. Less time is needed to run tests and adjust the vehicle.

3.1 Vehicle dynamics

The vehicle will be exposed to rolling and air resistance when driving on a flat road. Rolling resistance is caused by elastic deformation of the tyre. Air resistance occurs as aerodynamic phenomena effect the chassis body. The tractive effort is the result of engine power minus loss of force in the power train. These phenomena will be explained further in the following chapters.

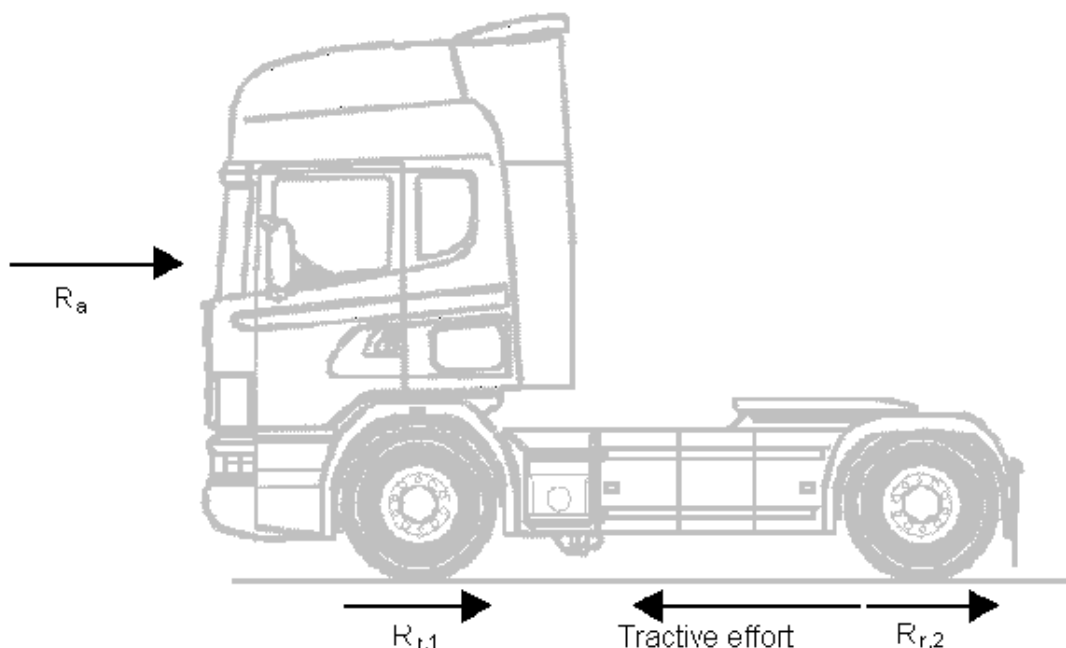


Figure 3.1 *Forces acting on the truck*

3.2 Wheel theory

Forces applied through the wheels control movement of the vehicle. Acceleration, deceleration and steering to mention the most essential. When choosing a proper set of tyres one would like to keep control over these factors. Still one must think of things as fuel consumption and lifetime of the wheel. Lower fuel consumption can be achieved by minimising force losses, but generally speaking at the expense of less lifetime and control. A common optimisation is choosing different wheels for the trailer and for the truck. The driving wheel needs certain characteristics since tractive force is only applied here. Braking forces works on all tyres. Both the tractive and braking forces are working where the wheel is in contact to the ground. Loss of force will occur at this area in form of heat. The rolling resistance will cause 60% of the total fuel consumption, for heavy loaded trucks it can be as high as 70% on a flat road. A proper set of wheels with the right pneumatic pressure is therefore an important ingredient to minimise fuel consumption. To be more specific, one has to understand the difference between rolling resistance and slip. These phenomena need to be described further before one can continue.

3.2.1 Rolling resistance

On a flat and hard road the flattening of the tyre primarily affects the rolling resistance. At the contact surface with the road the tyre will be deflected. See figure 3.2. A shift of the active normal force is achieved as tread-elements are compressed before entering the contact region. The pneumatic pressure inside the tyre is proportional to shift. Higher pressure gives smaller shift. Of secondary importance, the tyre will have a tendency to slide to different extents. This motion of tread over the contact surface leads to friction forces, which also

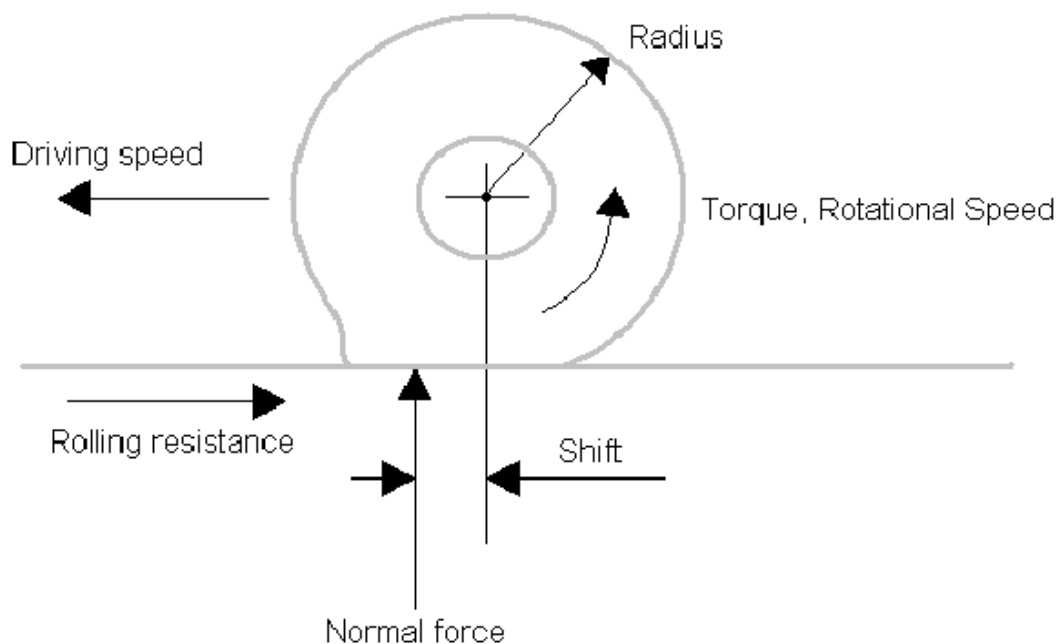


Figure 3.2 *Illustration of rolling resistance*

affect the resistance torque. Different grounds will give various friction forces. More can be

read about sliding in chapter 3.2.2. Of secondary importance the fan effect around the wheel and resistance caused by rotating air inside the wheel contribute to the rolling resistance. All these four phenomena are added in the resultant horizontal force more known as rolling resistance. The ratio of the rolling resistance to the normal load on the tyre is defined as the coefficient of rolling resistance.

The relation between elastic deformation of the tread elements and aerodynamic phenomena around and inside the tyre is complex. An analytic model for the rolling resistance is extremely difficult to compute. The most accurate determination of rolling resistance coefficient is therefore experimental analysis. The formula presented here has its origin from the tyre manufacturer Michelin. On a regular hard road surface the coefficient of rolling resistance for a truck tyre may be expressed by

$$C_r = C_{r_{iso}} + 9.45 \cdot E - 05 \cdot (v^2 - v_{iso}^2) - 1.54 \cdot (v - v_{iso}) \quad (3.1)$$

v is the speed of the vehicle in km/h. This rolling resistance coefficient is valid for speeds up to 100 km/h. $C_{r_{iso}}$ is the value from measurement according to ISO9948. v_{iso} is 80 km/h, speed during measurement according to ISO9948.

The resultant force of rolling resistance may then be expressed by

$$R_r = C_r \cdot N \quad (3.2)$$

N is the normal load on the wheel. The rolling resistance torque can finally be derived

$$N \cdot s = R_r \cdot r \quad (3.3)$$

where s is the shift of the normal force and r is the tyre radius..

3.2.2 Slip in relation to tractive force

A wheel exposed to a normal load and a driving torque will be compressed. Friction occurs at the contact to the ground as a result of slip. Slip is the sliding of tread over the contact area. Due to compression of the tread elements, the distance that the compressed wheel travels will be less than a free rolling wheel. This phenomenon is referred to as longitudinal slip. Most of the slip will be caused by elastic deformation of the tyre since tread elements are exposed to longitudinal stress. See figure 3.3. A corresponding shear deformation of the sidewall of the tyre is also developed. Free spinning slip can happen as a large driving torque is applied for fast acceleration or when driving on an ice surface. Larger losses result in the tractive force being lowered. One would like to find the optimum relation between slip and tractive force. A bit of slip is necessary since a zero slip would be the same as an infinitesimal small contact area and no tractive effort.

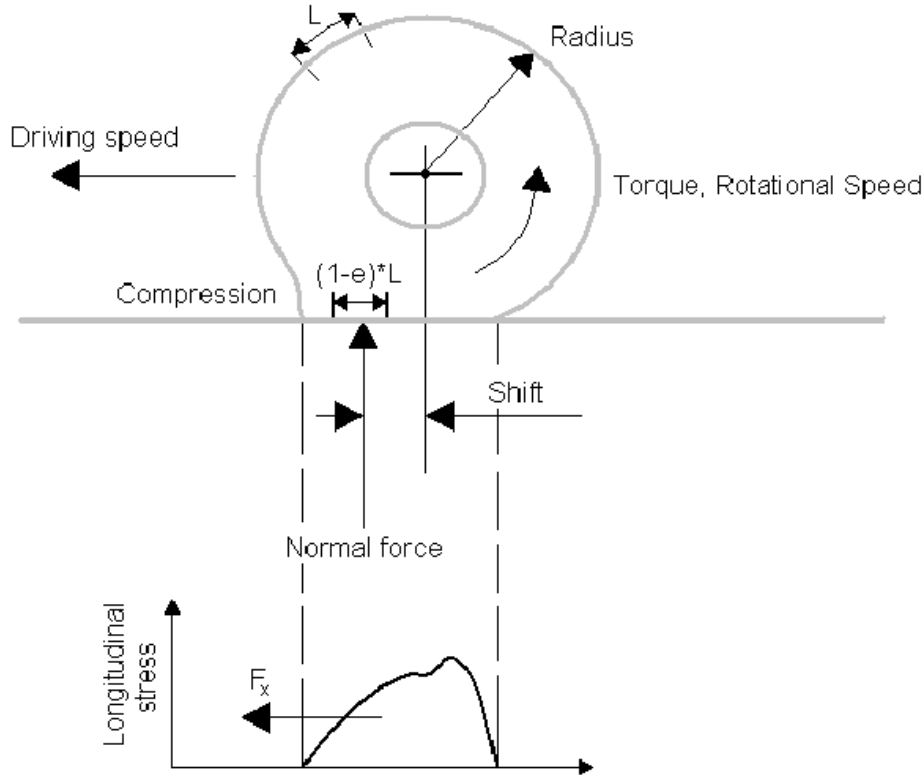


Figure 3.3 Behaviour of the tread element (L) as they are affected by longitudinal stress

The definition of slip is a percentage ratio between the actual velocity of the vehicle and rotational speed of the wheel. Accordingly the slip may be expressed by

$$s = \left(1 - \frac{v}{r\omega}\right) \times 100\% \quad (3.4)$$

where v is the velocity of the vehicle, r is the radius of the non compressed tyre and ω is the angular rotation speed of the wheel. If a tyre is rotating at a certain angular speed but the linear speed of the tyre centre is zero, then in accordance with Eq. 3.4 the longitudinal slip of the tyre will be 100%. This slip definition can not handle zero angular velocity, which is the same as hitting the brake. How braking effort is affected by slip will be neglected in this study. The reason for this is that slip is primarily discussed to optimise the tractive force in this report.

Tractive force is the total applied force for driving the vehicle forward. How the tractive force is connected to slip is of great interest. A general theory to accurately predict the relationship between longitudinal slip of pneumatic tyres and driving force on hard surfaces has yet to be evolved. Attempts have been done to do theoretical models, one of the earliest was made by Julien. His calculations will not be treated here. Instead one can look at experimental data to learn something about the properties of slip. Slip is a function of tractive force, which is proportional to applied wheel torque on the driving tyre. Generally speaking, at first the wheel slip increases linearly with increasing tractive force and torque. During this phase slip is mainly caused by elastic deformation of the tyre tread. A further increase of tractive force results in part of the tyre tread sliding on the ground. Under these conditions it can be found that the relation between slip and tractive force is non-linear. These statements are shown in

figure 3.4. Based on available experimental data, the maximum tractive force is usually reached somewhere between 15 and 20% slip. Any further increase in slip results in tractive force falling.

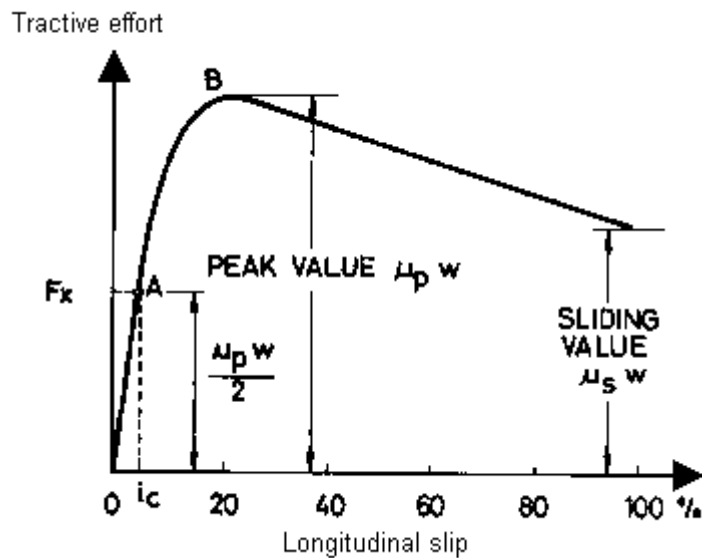


Figure 3.4 Relation between longitudinal slip and real tractive effort. μ is the peak and sliding values of the road adhesion. W is the normal load.

In the figure above the peak value, $\mu_p w$ is not the same as the tractive force delivered from the power train (see next chapter). Instead the peak indicates the maximum tractive force that can be transferred to the ground.

3.2.3 Tractive effort and power train

Vehicle power train consists mainly of engine, clutch, transmission, shaft and wheels. The tractive force is the result of what torque the power train generates. The power produced by the engine is transmitted through gearbox and central gear. Knowing the characteristics of these mechanical parts is essential to compute the total tractive force.

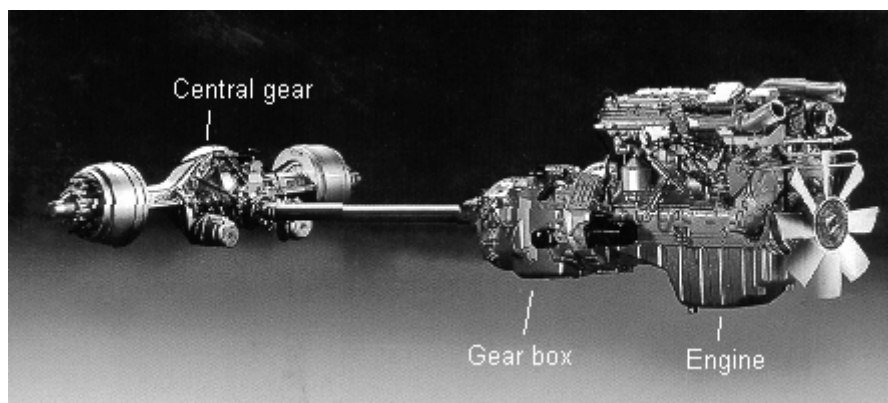


Figure 3.5 The power train of a truck

The engine produces different amounts of torque depending on the speed of the engine. Slow engine speed delivers less torque. A maximum of torque can be achieved between 1100-1400 rpm for trucks. Adjustment to the injection system determines the torque-rpm relationship. Operations at low engine speed and high torque are always more fuel economical than at higher engine speed and lower torque settings with the same power output. The relationship between engine speed, torque and rpm is shown in figure 3.6.

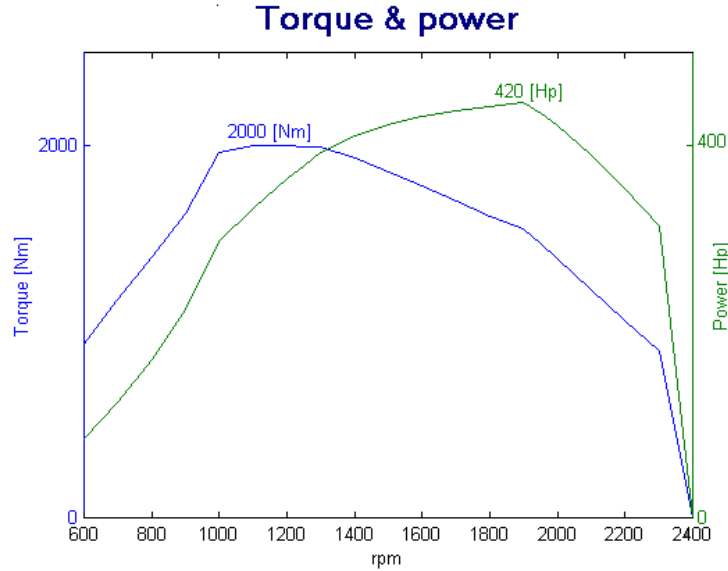


Figure 3.6 Torque and power for a diesel engine

Since the effect of the engine at a certain speed is known, the tractive effort can be derived. Engine torque is multiplied in the gearbox. The ratio of the gearbox depends on chosen gear. - the lower the gear the higher the ratio. In the central gear the torque is multiplied a second time. Accordingly the following formula for tractive effort can be written

$$T_{wheel} = T_{engine} \cdot i_{gearbox} \cdot i_{central_gear} \cdot \eta_{gearbox} \cdot \eta_{central_gear} \quad (3.5)$$

where $i_{gearbox}$ and $i_{central_gear}$ are the ratio of the respective components. Losses of energy occur in every component of the driving train. Most losses happen in the gearbox and the central gear. The magnitude depends primarily of how many cogs which are transmitting energy. Each cog contributes with approximately 1% of torque loss. Both η are the loss coefficients of gearbox and central gear. T_{engine} and T_{effort} are torque [Nm].

The tractive force can then be calculated as

$$F_{tractive} = \frac{T_{wheel}}{r} \quad (3.6)$$

r is the compressed tyre radius of the driving wheel.

3.3 Chassis body theory

The chassis body of a truck should be formed to get low fuel consumption. This is often hard since the appearance of the vehicle should look attractive to the customer. Since the early 20th century, car manufactures have had a good knowledge of the aerodynamic phenomena affecting the vehicle. New studies have not obtained any sufficient new results. To get an accurate prediction of driving force one must put up two expressions, resistance and lift, describing active forces of the air when a vehicle is running. On a flat road air resistance is said to cause 20 % of the total fuel consumption for a truck, according to Ref. (7). In comparison a normal passenger car, with lower weight, will be exposed too a much higher air resistance ratio. For the car, air resistance is the dominant factor for the fuel consumption. In the following text a deeper understanding of air resistance is presented, chapter 3.3.1. The effects of aerodynamic lift will also be described in 3.3.2.

3.3.1 Aerodynamic resistance

Air resistance is primarily the result of two factors. The first is the airflow across the exterior of the vehicle body. The second is the flow through the engine radiator system and the interior of the vehicle. Of the two the first is dominant. External air resistance generates normal pressure and shear stresses to the chassis body. Aerodynamic nature states that these can be divided into pressure drag and skin friction. The pressure drag is strongly related to the normal pressure on the chassis body. Normal pressure arises as the truck is driven with a speed and the frontal area is exposed to a wind flow. Skin friction is caused by shear stresses close to the chassis body external surface, called the boundary layer. Long trucks are subjected to higher amounts of skin friction. As a consequence the characteristic area of a truck is not the same as the frontal area. In practice, the active force of air resistance can be expressed as

$$R_a = \frac{\rho}{2} \cdot C_D \cdot A_f \cdot V_r^2 \quad (3.7)$$

where ρ is the density of air, C_D is the coefficient of aerodynamic air resistance, A_f is frontal area of the vehicle (characteristic area of the vehicle is C_D multiplied with A_f) and V_r is the speed relative to the wind [m/s]. The force needed to overcome air resistance increases with the square of the speed. As a result a rise in speed to the double increases the air resistance force four times.

The most accurate determination of air resistance is done on a flat road, commonly referred to as the coast-down test. Using this test the vehicle is running of a certain speed, then the engine power is disconnected. Deceleration caused by rolling resistance, power train resistance and air resistance can then be derived. To achieve good results knowledge about power train and rolling losses is essential. This method is an alternative to wind tunnel tests. Benefits from the test are primarily that it is less expensive. However, it is more sensitive to disturbances, e.g. wind shifts.

3.3.2 Aerodynamic lift

Aerodynamic lift, or induced drag, is the result of any lift forces that are generated by the moving vehicle. A chassis body produces an accelerated air flow and corresponding low pressure on its upper surface. The pressure difference between upper and under-body generates lift force. This results in a decrease of normal load on the tyres. For trucks lift forces should, theoretically speaking, be maximised in order to achieve lower weight on the wheels and thereby smaller rolling resistance, which is the most important factor to adjust fuel consumption. The problem is that the truck is heavy and lift forces must be very large in order to get any decrease of fuel consumption. The trucks of today do not even have a solid plate underneath the chassis, which would raise the magnitude of lift. Lift forces are therefore neglected in simulations of truck since the contribution is small. Racing cars on the other hand generate increased normal load from negative lift forces. Thus, the performance characteristics and directional control and stability of the vehicle may be affected positively. The resulting aerodynamic lift on a chassis body is usually expressed by

$$R_L = \frac{\rho}{2} \cdot C_L \cdot A_f \cdot V_r^2 \quad (3.8)$$

where ρ is the air density, C_L is the coefficient of lift usually obtained from wind tunnel testing, A_f is the characteristic frontal area and V_r is the velocity [m/s].

The magnitude of C_L depends on several factors. Characteristic ground clearance of the road affects the airflow underneath the vehicle body, which must be put in relation to the under-body surface of the vehicle. Aerodynamic phenomena as turbulence and boundary layer should be included. The angle of attack of the body to the air is also important.

3.4 Simulation of vehicle in terrain

The problem of understanding a truck's ability to handle a sloping road at a steady velocity was a part of the assignment. This is referred to as the gradability of the truck. How this was handled and implemented in the Stars GUI will be presented in this chapter.

Almost all roads have hills and the driver and vehicle must be able to negotiate these. Altitude shifts on the road expose the truck to a gravitational force gradient. Uphill driving needs more power supply than down hill driving. This has to do with the relation between dynamic- and potential -energy. Dynamic energy is the energy of the moving truck, which is proportional to the square of speed and the mass. Fast driving vehicles have higher dynamic energy. The potential energy is the amount of work the truck has achieved when climbing to a higher altitude. A vehicle on a higher level has more potential energy then compared to sea level. Potential energy is described as a function of gravitational force, mass and height difference. In an uphill slope the tractive effort is converted to potential energy according to the law of energy conservation. Driving down hill potential energy is converted to dynamic energy. As a result of this one will need more fuel driving up a hill and more braking force driving down hill. This chapter focuses on a vehicle driving uphill at a steady speed. Gradability, a

measurement of characteristic power of a vehicle is introduced. With the knowledge of the specific gradability one can get an idea of how strong the truck's power train is in relation to its weight, rolling resistance and air resistance.

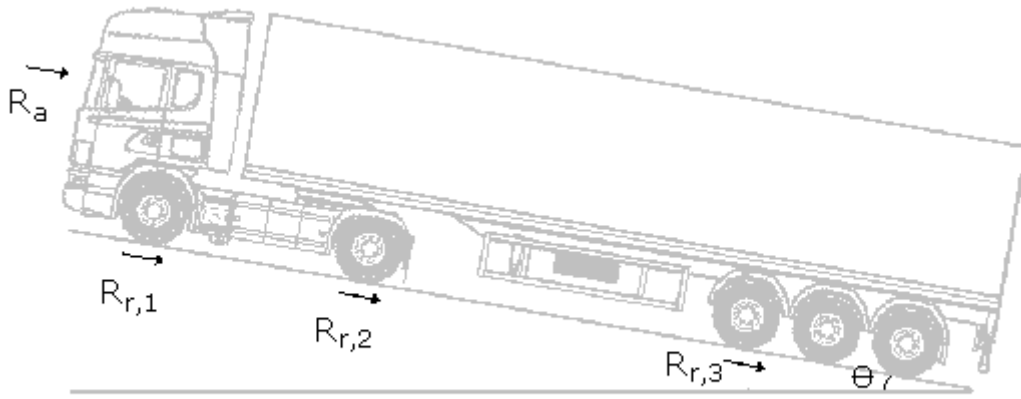


Figure 3.7 Truck climbing a hill. θ is the slope.

3.4.1 Gradability

Gradability is a popular measurement of the strength of the power train (engine, gearbox and central gear) in relation to the weight of the truck and trailer. It can for instance be used to set a proper power train to a vehicle or to classify different power trains. Gradability is usually defined as the maximum slope a vehicle can handle at a certain speed. On a slope at a constant speed, the tractive effort has to overcome grade resistance, rolling resistance and aerodynamic resistance. Assume the vehicle is running at a steady state speed. The following equilibrium can be stated

$$F_{tractive} = w \cdot \sin \theta + R_r + R_a \quad (3.9)$$

w is the weight of the vehicle in [N], R_r is the rolling resistance and R_a is the aerodynamic resistance. F is the tractive force. Rewriting the expression will give the percent grade G as

$$G = 100 \cdot \tan \theta = 100 \cdot \tan(\sin^{-1}(\frac{1}{w} \cdot (F - R_r - R_a))) \quad (3.10)$$

3.4.1.1 Evaluating the force of rolling resistance

To fully derive the force of rolling resistance on each tyre, which is needed to evaluate equation 4.3, one must lay the truck and trailer bare. Doing this first for the truck will result in the following equilibrium force and torque equations. See also figure 3.8 for information about the variables.

$$\sum F_x = 0 \Leftrightarrow R_a + R_{r,1} + R_{r,2} + Q \cdot \sin \theta - F_{tractive} + m_1 \cdot g \cdot \sin \theta = 0 \quad (3.11)$$

$$\sum F_y = 0 \Leftrightarrow \frac{R_{r,1}}{c_{r,1}} + \frac{R_{r,2}}{c_{r,2}} - Q \cdot \cos \theta - m_1 \cdot g \cdot \cos \theta = 0 \quad (3.12)$$

$$\sum M_A = 0 \Leftrightarrow H_1 \cdot (F_{tractive} - R_{r,1} - R_{r,2}) - H_2 \cdot Q \cdot \sin \theta + H_3 \cdot R_a + L_1 \cdot \frac{R_{r,1}}{c_{r,1}} + L_2 \cdot Q \cdot \cos \theta - L_3 \cdot \frac{R_{r,2}}{c_{r,2}} = 0 \quad (3.13)$$

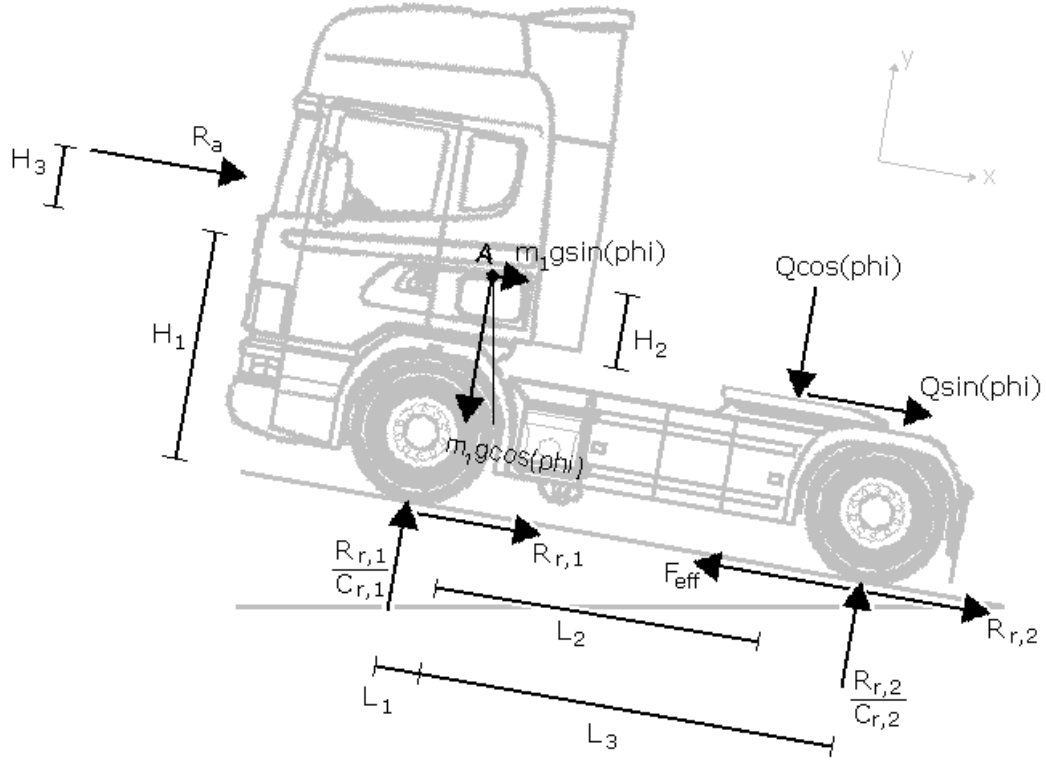


Figure 3.8 Forces and length variables on the truck. A is the centre of gravity on the truck

Writing the same equilibrium equations for the trailer leads to the following equations. See also figure 4.5 for information about variables. (Only two equations are needed for the trailer since one is searching for an expression for Q depending on θ)

$$\sum F_y = 0 \Leftrightarrow Q \cdot \cos \theta + \frac{R_{r,3}}{c_{r,3}} - m_2 \cdot g \cdot \cos \theta = 0 \quad (3.14)$$

$$\sum M_B = 0 \Leftrightarrow H_4 Q \sin \theta - H_5 R_{r,3} + L_4 Q \cos \theta - L_5 \frac{R_{r,3}}{c_{r,3}} = 0 \quad (3.15)$$

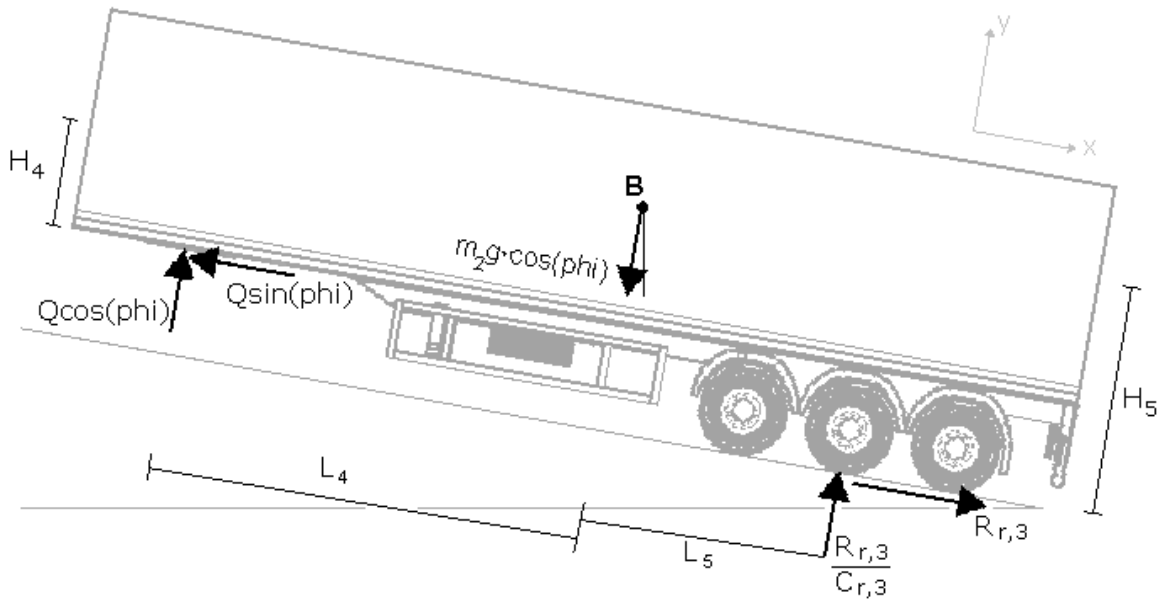


Figure 3.9 Forces and length variables on the trailer. B is the centre of mass

The system for equation 3.11-3.15 is now solvable. The resulting rolling resistance on the driving tyre can be written as a function of θ and air resistance

$$R_{r,2}(\theta, R_a) = \frac{c_{r,2}}{(1 + \frac{L_3}{L_1})} (m_1 g (\cos \theta + H_1 \sin \theta) + R_a (H_1 + H_3) - Q (H_2 \sin \theta - H_1 \sin \theta - L_2 \cos \theta - \cos \theta)) \quad (3.16)$$

Rolling resistance for the front tyre will also be a function of θ and air resistance

$$R_{r,1}(\theta, R_a) = \frac{1}{L_1} (\frac{R_{r,2}}{c_{r,2}} L_3 + Q (H_2 \sin \theta - H_1 \sin \theta - L_2 \cos \theta) - H_1 m_1 g \sin \theta - R_a (H_1 + H_3)) \quad (3.17)$$

The trailer wheel has a rolling resistance according to

$$R_{r,3}(\theta) = c_{r,3}(m_2 g \cos \theta - Q \cos \theta) \quad (3.18)$$

The force from the trailer Q (equation 3.14 and 3.15) evaluates to

$$Q = \frac{m_2 g \sin \theta (H_5 c_{r,3} + L_5)}{H_4 \sin \theta + L_4 \cos \theta + H_5 c_{r,3} \cos \theta + L_5 \cos \theta} \quad (3.19)$$

The force of rolling resistance depends of the weight on the tyre and velocity of the vehicle. Air resistance R_a is a function of velocity (see chapter 3.4.1.2). At a steady state speed rolling resistance will become a function of the grade according to

$$R_r(\theta, R_a) = \sum_{i=1}^3 R_{r,i} \quad (3.20)$$

this expression can then be put into equation 3.10

3.4.1.2 Evaluating the force of air resistance

The aerodynamic resistance is proportional to the frontal area, velocity and the resistance coefficient. A full description of the air resistance is done in chapter 3.3.1 and the resulting formula can be written

$$R_a = \frac{\rho}{2} \cdot C_D \cdot A_f \cdot v^2 \quad (3.21)$$

The force of air resistance is depending on the velocity at which the truck is driven. Velocity, v is the speed parallel to road line, is proportional to engine speed, power train variables, tyre radius and slip according to the following formula

$$v = \frac{v_{engine} \cdot r \cdot C}{P_{gearbox} \cdot P_{central_gear}} \quad (3.22)$$

v_{engine} is the velocity of the engine [rpm]. r is the wheel radius. C equals 0.06 for [km/h]. $P_{gearbox}$ and $P_{central_gear}$ are the ratios of the specific gear step and the central gear ratio.

3.4.2 Gradability calculation

As described above, the gradability equation is only a function of θ . Accordingly equation 3.10 can be rewritten as

$$G(\theta) = \frac{1}{w} \cdot (F_{tractive} - R_r(\theta, R_a) - R_a) \quad (3.23)$$

Knowing the velocity, tractive effort(maximum gas) and weight of the vehicle the above equation is solvable. Equality iteration loop is used. Guess an initial value for θ and put it into the right hand side of the expression. Calculate a new value for θ in the left hand side. Put it into the right hand side and recalculate. Repeat the procedure until the old and new value equal.

Running the iteration loop for all gears will deliver the gradability result over the whole velocity range. See figure 3.10.

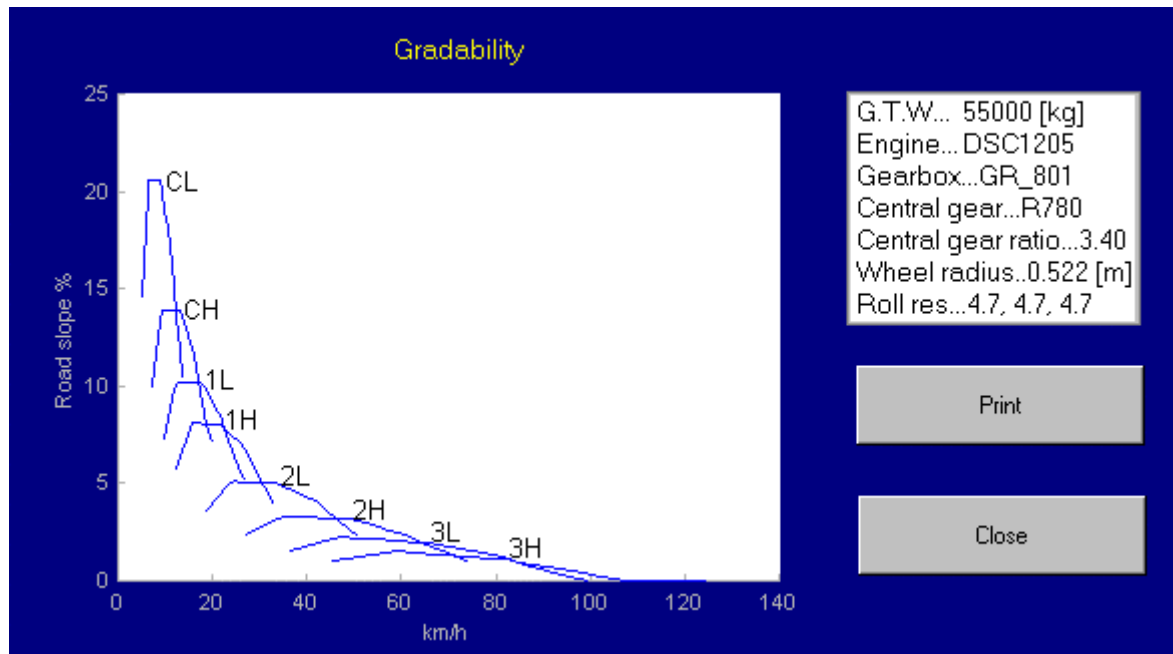


Figure 3.10 Gradability diagram for a truck. (CL,...,3H) are the gear step names. The two gears CL and CH are the creeping gears. “Roll res” is the rolling resistance coefficient C_r for each tyre. The same picture exists in the Stars GUI.

3.5 Discussion

At the time when this report was written, slip as introduced in chapter 3.2.2 has not been included in the calculations for gradability. Instead a default slip of the tyre of 15% was implemented. See figure 3.4. An extension of the gradability would be to build in the tractive effort to slip relation. The main problem is to find solutions that are illustrative in a diagram as above. Experimental calculations showed that free spin of the driving tyre occurred when

the slope is larger than 20% and the velocity less than 30 km/h. Leading to recalculation of gradability slope as the tractive effort is smaller.

4 Numerical algorithms in Dymola

The Dymola simulation is started by a call from Matlab. One has a number of numerical algorithms to choose from for solving the simulation. In this chapter the relevant algorithms for Stars are presented.

The simulation results are described by a DAE system. Numerical algorithms are powerful to use when one would like to solve DAE without knowing the analytical solution. A DAE is an expression where one puts the solution of a problem in relation to the first and/or second derivative, and so on, of the solution. Dymola defines a DAE-system, which is to be solved. This system, which is a number of differential equations, describes the whole simulation process and is as

$$f(x'(t), x(t), y, u) = 0 \quad (4.1)$$

In the GUI a built in code calls Dymola with a specific syntax when the simulation is started. How this is done is described in chapter 4.3.1. The simulation could be calculated with fourteen different numerical algorithms, but only four of them are of interest for simulation in Stars. In order to control and choose the right numerical method of these four one must understand the underlying theories. This is also important since one must have an idea of how accurate the calculated solution is. The system of DAEs in Dymola are hidden and never presented. But to illustrate how a differential equation is solved the following time dependant linear Ordinary Differential Equation (the vectors y and u are taken away) can be written

$$\begin{aligned} f(x'(t), x(t)) &= x'(t) - C \cdot x(t) = 0 \\ x(t=0) &= D \end{aligned} \quad (4.2)$$

C is a constant and $x(t=0)$ is the initial value. This chapter explains how one can solve such a problem numerically. The number of numeric methods that can be used is restricted since the model in Stars needs a fixed step length. The reason for this is that Stars uses c programmed routines for gear changes with a fixed time interval. Therefore only possible numeric methods for Stars be described in this chapter. Before one can move on there are some fundamentals about numeric calculations called Taylor series that need to be described.

4.1 Taylor series

Taylor series is a powerful way of predicting future values of an initial value problem. This is possible since the solution of an ODE problem is, not dealing with exceptions, only dependant on its initial value and the changes happening to the system applied at the initial time. According to the definition for the Taylor series the following expression can be written

$$x(t_{i+1}) = x(t_i) + x'(t_i) \cdot h + \frac{x''(t_i)}{2!} \cdot h^2 + \frac{x'''(t_i)}{3!} \cdot h^3 + \varepsilon(h^4)$$

$$\varepsilon = \text{numerical error} \quad (4.3)$$

$$h = \text{step length}$$

$$t_{i+1} = t_i + h$$

$$3! = 1 \cdot 2 \cdot 3$$

Note that this summation is always valid as a solution with the error of ε . This means that the precise future value can only be approximated with a Taylor series expansion. An alternative name for Taylor series is Maclaurin series, which has a slightly different definition. All numerical algorithms described in the following chapters can be derived from the Taylor series. Since the Taylor series is not precise, all numerical solutions are only a simulation of the behaviour of the system governed by the ODE.

4.2 Explicit Euler method

Explicit Euler method is a numeric method for calculating differential equations. One needs to know the initial value of the problem in order to start the computation. The numeric solution that results is only an approximation of the true solution. The accuracy of the solution depends on the step length. The method uses fixed step length, which is necessary in the Stars Dymola modell.

From the definition of the Taylor series one can derive the explicit Euler method. Rewriting the expression (4.2) gives

$$x(t_{i+1}) = x(t_i) + x'(t_i) \cdot h + \varepsilon(h^2) = x(t_i) + g(t_i, x(t_i)) \cdot h + \varepsilon(h^2) \quad (4.4)$$

t_n is the total simulation time and n is the number of solution points. Notice that the solution of the point at t_i is the initial value for the solution at t_{i+1} . This means that the error of the solution, local error is summed up in the solution points the further one chooses to calculate. A larger global error will occur at the end. The global error can be minimised by shrinking the step length. More can be read about this in chapter 4.2.2. The function f can according to differential equation (4.1) be written as

$$g(t_i, x(t_i)) = x'(t_i) = C \cdot x(t_i, x(t_i)) \quad (4.5)$$

Rewriting equation 4.4 gives

$$x(t_{i+1}) = x(t_i) + C \cdot x(t_i) \cdot h + \varepsilon(h^2) \quad (4.6)$$

Now one can use the Explicit Eulers method to calculate a numeric solution for the ODE introduced in equation (4.1). Assume the initial value to be $x(0)=1$ and $C=1$. The exact

solution of the problem was calculated with variable separation of equation 4.1. The following figure 4.1 can then be drawn.

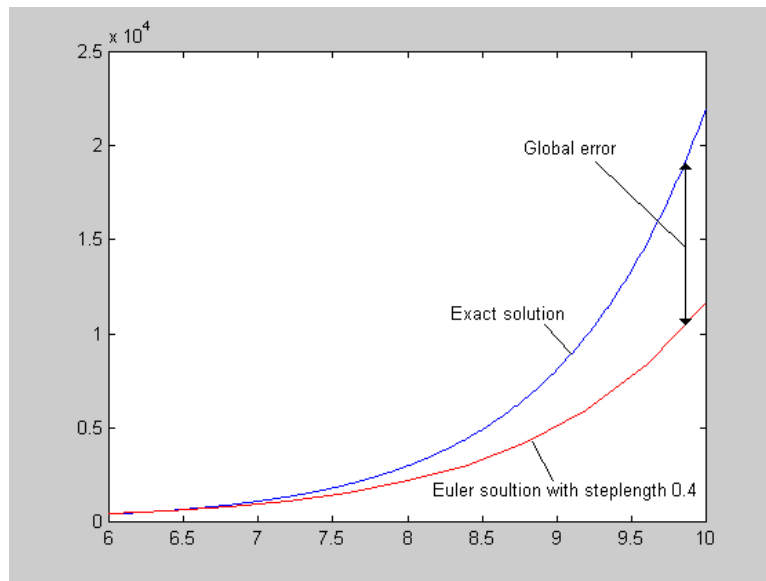


Figure 4.1 *Exact and Euler solutions to differential equation in (5.1). The steplength is 0.4. The exact solution was found through variable separation.*

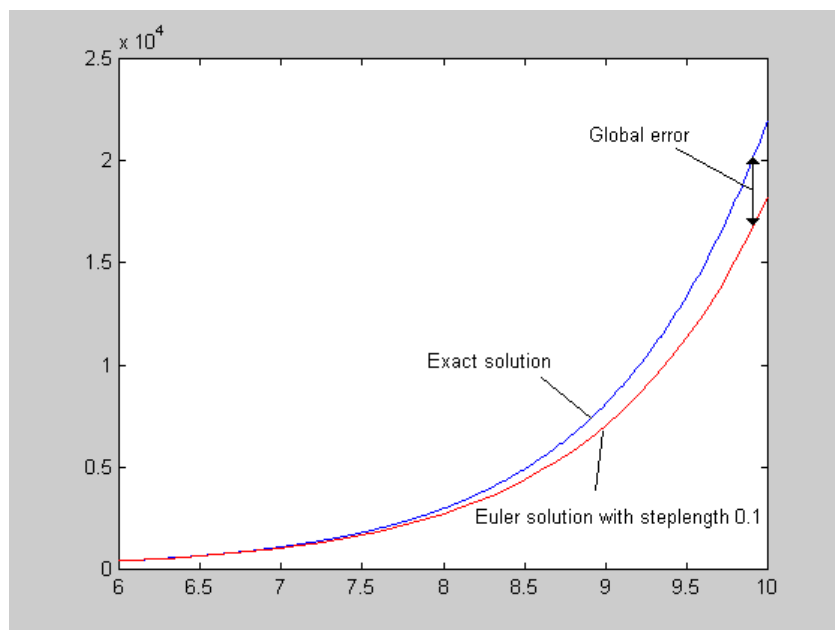


Figure 4.2 *Same as figure 4.1 but the step length is just 0.1. The global error is smaller because of shorter step length.*

In the figures above one sees how the local error in each evaluation is decreasing when the step length is shortened. The final global error is quite large in magnitude for both step lengths. By shrinking the step length to 0.1 the solution will be better.

4.3 Implicit backward Euler method

An improved numeric method is the implicit backward Euler method. It does not tell if Dymola uses the explicit or implicit Euler. But it could be of interest to know how the implicit Euler works. Explicit method uses information at time t_i to predict the value at time t_{i+1} . The disadvantage is that the stability region is limited. A larger stability region can be obtained by using information at time t_{i+1} , which makes the method implicit. According to the definition the simplest backward Euler method can be written

$$x(t_{i+1}) = x(t_i) + g(t_{i+1}, x(t_{i+1})) \cdot h + e(h^2) \quad (4.7)$$

According to equation 4.1

$$g(t_{i+1}, x(t_{i+1})) = x'(t_{i+1}) = C \cdot x(t_{i+1}) \quad (4.8)$$

Evaluating 4.7 and 4.8 gives

$$x(t_{i+1}) = \frac{x(t_i)}{1 - C \cdot h} + \varepsilon(h^2) \quad (4.9)$$

Resulting solution is plotted in figure 4.3. The same constant and initial value is used as in the explicit Euler method.

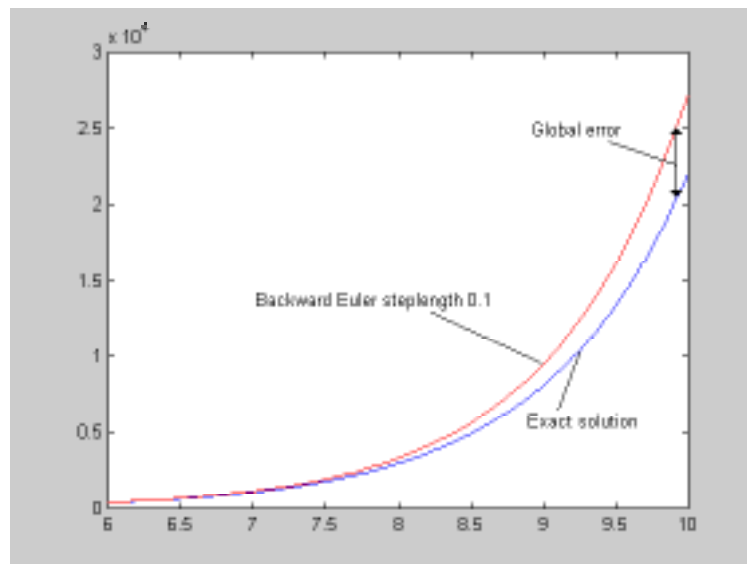


Figure 4.3 *Backward Euler solution. Notice how the backward Euler solution lies above the exact solution curve. This leads to increased stability.*

4.3.1 Call Dymola using Eulers method

When calling Stars dymola model from Matlab one uses the command, `dymosim(experiment)`. (This is executed in the file `selectOK.m`, case `dymola` and case `dymola2`.) Where `experiment` is a vector of length six containing a couple of parameters to control the numeric method. If one wants to use the Euler method one can set the `experiment` vector as follows

<code>experiment(1,1)=1</code>	(time for start of evaluation)
<code>... (1,2)=360000</code>	(stop of evalution. Infinity)
<code>... (1,3)=0.1</code>	(steplength)
<code>... (1,4)=0</code>	(number of communication intervals)
<code>... (1,5)=0.01</code>	(Tolerance. No importance for Euler)
<code>... (1,6)=11</code>	(11 is the number for Euler)

The most important parameters to set when one uses Euler is as described above, `experiment(1,3)` and `experiment(1,6)`. Notice that the tolerance, `experiment(1,5)` have no significance for the Euler method. The dymola manual does not state if it uses a explicit or implicit Euler method. The exact solver could be slightly different from those described above, but the general idea is the same. The `experiment(1,2)` should be set to a time value close to infinity.

4.4 Runge-Kutta

An alternative numeric solver is the Runge-Kutta method, which calculates a more precise numerical solution. It also has its origin, as Euler, from the Taylor series but is more intelligent. It makes slightly more work in each calculation and delivers more accurate results. Runge kutta method is often, in other applications, used as a numerical approximation solver.

The formula for Runge-Kutta of order four could be written as follows

$$x_{k+1}(t_{k+1}) = x_k(t_k) + \left(\frac{k_1 + 2 \cdot k_2 + 2 \cdot k_3 + k_4}{6} \right) \cdot h$$

$$k_1 = f(t_k, x_k)$$

$$k_2 = f\left(t_k + \frac{h}{2}, x_k + \frac{h}{2} \cdot k_1\right) \quad (4.10)$$

$$k_3 = f\left(t_k + \frac{h}{2}, x_k + \frac{h}{2} \cdot k_2\right)$$

$$k_4 = f\left(t_k + h, x_k + h \cdot k_3\right)$$

As one can see the order is the same as number of constants k . Higher order Runge-Kutta are developed, but Dymola only has implemented code for order four.

As done in previous chapters with Euler we will now apply the numerical method on equation 4.1 using Runge-Kutta order four. The constant $C=1$ and $x(0)=1$ delivers the following result, figure 4.4

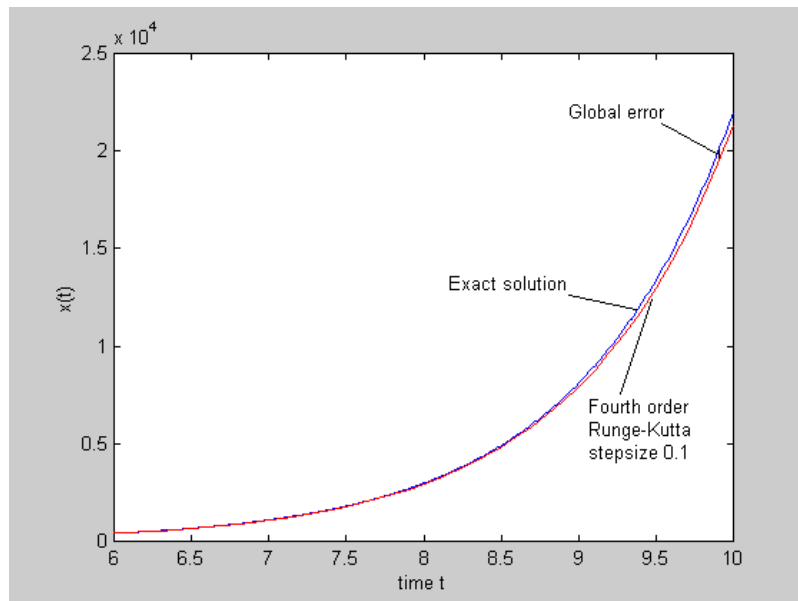


Figure 4.4 *Runge-Kutta numerical solution with step length 0.1 to equation 5.1*

The result is obviously more accurate using Runge-Kutta. Slightly longer calculating time is a disadvantage, but if one can negotiate with approximately 10% longer time one should always choose R-K. One could also rewrite the R-K method to be implicit in the same way as backward Euler. The solution will then become more stable since the numerical solution will “lie ahead” in the direction of the trend of the solution.

4.4.1 Call Dymola using Runge-Kutta

As described in 4.2.1 one must also change the experiment vector in the call to dymosim when one wants to use Runge-Kutta to solve the problem. Dymola has three various Runge-Kutta methods with different order. The order represents the number of constants $k_1, \dots, k_{\text{order}}$ that the method calculates. R-K of order two will by this way be slightly faster than the order four solution, but the accuracy somewhere in between of Euler and R-K of order four. Order four is always most precise. Set the experiment vector as follows

experiment(1,6)=14 (14 = R-K order four
 13 = ... three
 12 = ... two)

experiment(1,1:5) should be set as described in 4.2.1. Tolerance in experiment(1,5) has no importance for Runge-Kutta either. This parameter is only significant when using variable step length.

4.5 Global error and step length control

Since one knows that a local error can occur in each iteration calculation one can understand that the global error will be the summation of all the local errors. The magnitude of the local error is always dependant on the step length of the numerical solver. See figure 4.1 and 4.2. Shorter step length gives smaller local error and less global error. Still one would like to have as long a step length as possible in order to run efficient calculations. According to equation 4.2 and 4.3 one can put up the following equation for the local error

$$E_{\text{local}} = \varepsilon(h^2) = \frac{x''(t_i)}{2} \cdot h^2 \quad (4.11)$$

The step length should then accordingly be set to a value that obeys the following expression

$$h \leq \sqrt{\frac{2 \cdot E_{\text{local}}}{x''(t_i)}} \quad (4.12)$$

where the second differential of x could be approximated with

$$x''(t_i) = \frac{x'_i - x'_{i-1}}{t_i - t_{i-1}} \quad (4.13)$$

4.6 Discussion

Runge-Kutta has the disadvantage of being slightly slower than the Euler. Approximately 10% longer calculation time is needed, but the results should be more precise. (this has not been proven since no evaluation of the accuracy of the simulation has been done.). The difference in the solution using R-K instead of Euler has shown through test running to be 1% in the fuel consumption for the truck. Difference in solution for exhaust emissions, CO, NO_x and HC, are of the same magnitude.

Another advantage can be seen that the step length could be longer when using R-K and still get good results. But the Dymola model is restricted so the step length must be kept at 0.1, otherwise the gearbox routine in Dymola not be able to make correct gear shifts. Theoretically speaking though the simulation time would be much shortened if one implemented the code to the gearbox differently. If one choose to only make gear-shift-checks every 0.4 second the steplength could also be set to 0.4, which would make the simulation four times faster. This would also make a much smaller simulation result file, which means that simulation also could be made four times longer. The reason for using gear-shift-checks every 0.1 second is that the OptiCruse (Scania's automatic gearbox) has this time interval.

It should be noted that none of the numerical solvers introduced in these chapters can handle stiff differential equations. A stiff differential equation has a solution that converges rapidly with time. This means that numerical method could have a problem to find the desired solution. Since fixed-step-size-solvers as Euler and Runge-Kutta have a built in slow response (fast response is needed for stiff DE) unstable solutions can occur. A way to get around this phenomenon and still use fixed step length is to shorten the step length, which speeds up the response, but at the cost of time for the evaluation. Variable step length is good way of handling stiff equations. Unfortunately the Stars simulation model is unable to handle variable step length. For more information about stiff DE see reference (3).

5 Calculations on results

To receive and present the results from the calculation in the GUI is important. The user must find the results in a way he is used to see them. This chapter mainly presents how the values, presented in the GUI, were calculated. The most important results are the emissions of NO_x, CO_x, HC and the fuel consumption. These values are important in order to see if the engine has acceptably high emission values. Too high values would be a problem since the authorities would not approve the engine. This chapter also handles how the drive-time-scheme was calculated. In this scheme one can find out in what interval the engine has been working during the simulated driving regarding load (percentage of gas) and rpm (engine speed).

5.1 Emissions and fuel consumption

The main concerns about emissions and fuel consumption are to keep them on a low level. Too high values of emissions would not be environmental friendly and the authorities would not approve the engine. Fuel consumption should be kept low in order to keep the customer satisfied. Every engine is different and delivers certain emissions during driving. Depending on what road profile the truck is driven at the engine will work differently. Temperature and air pressure also affects the engine performance.

A correct combination of gearbox, central gear and wheels is essential in order to have an engine delivering low emissions and fuel consumption. Still one wants to keep the truck performance on a high level regarding acceleration and gradability etc. Tests are a good way of finding out information about fuel consumption for the truck. These tests must be done repeatedly with various combination of drive train (engine, gearbox and central gear). One combination is only optimal for a certain terrain and specific speeds. Compromising is the only way of finding an effective and proper drive train for the truck under a number of different conditions. It is therefore of great interest to receive information on fuel consumption and the exhaust emissions during the simulated driving.

5.2 Unity calculations

All emissions from the simulation are delivered with a unity of [g/h]. This result was calculated from the engine mussel of the respectively emission. See appendix B headline engine. The results should be presented in the following unites.

$$\left[\frac{g}{kWh}\right] \quad \left[\frac{g}{L}\right] \quad \left[\frac{g}{km}\right] \quad (5.1)$$

Integrating the emission [g/h] over the simulation period time will result in [g], the total weight of emissions over the simulation, according to

$$[g] = \int_{time[s]} \frac{1}{3600} \cdot \left[\frac{g}{h}\right] \cdot dt \quad (5.2)$$

One know needs the amount of [kW], fuel consumption [L] and distance that the truck has travelled to compute the right unites.

Starting with [kWh] one gets the results of effect from the engine in each time step of the simulation as a unity [kW]. Integrating and dividing with a constant gives

$$[kWh] = \int_{time[s]} \frac{1}{3600} \cdot [kW] \cdot dt \quad (5.3)$$

The fuel consumption is given as a fuel flow [g/h] one looks for [L]. The following expression was implemented

$$[L] = \frac{1}{\rho} \int_{time[s]} \frac{1}{3600} \cdot \left[\frac{g}{h}\right] \cdot dt \quad (5.4)$$

ρ is the density constant of the fuel which, according to Automotive Handbook (Ref 10), is approximated to 840 [g/L]. This value is valid for diesel at a temperature of 15 °C.

Fuel consumption is calculated in the same way.

5.3 Engine statistical time

From the Engine statistical fuel scheme one can find many interesting results. See figure 5.1. E.g. will an E-s-t from the South America look quite differently from a scheme from Europa. This has to do with the truck driver characteristics. A truck driver from South America will drive the truck at a much higher speed, about 100 [km/h] is common. While a European driver will be restricted by speed limitations of 80 [km/h]. Different set up of the driving train is necessary. The E-s-t gives a picture of where the engine is working most of the time during the driving.

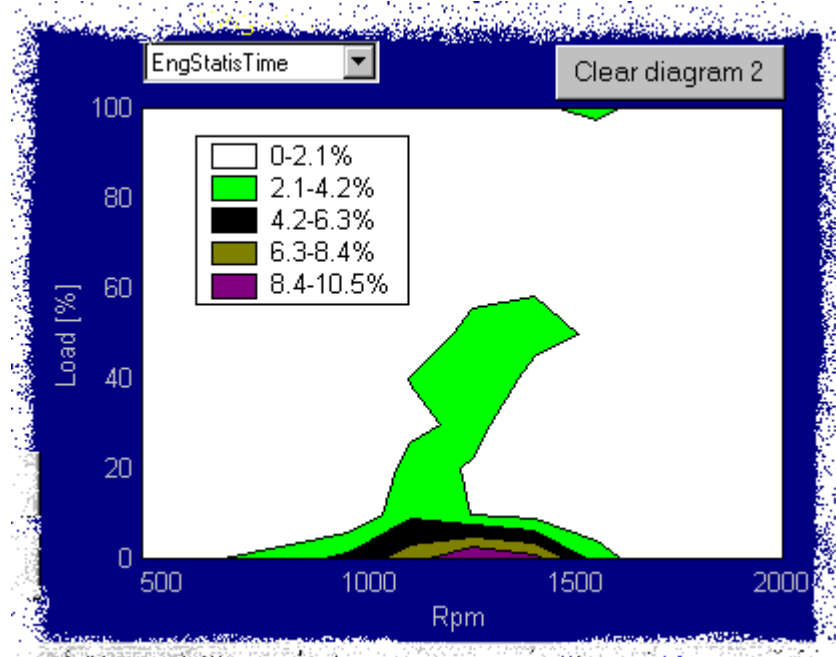


Figure 5.1 *Engine static time scheme for a simulation. Engine: DC1601, gearbox: GRS900, central gear: R780, central gear ratio: 3.4, total weight: 55000 kg and road: Jaddah->Khamis. Between 8.4-105% of the time the engine has been driving with zero gas at approximately 1250 rpm.*

The E-s-t is calculated statistically. A construction of a matrix is done where rpm are the columns and load the rows. A number is added in the column where the rpm for the engine is working for all sample points during the simulation. The same is done at each time step for the load. This results in a matrix, which describes where the engine has been working over the simulation time

5.4 Engine statistical fuel

Engine statistical fuel is a good view for information on where the engine has consumed fuel during the simulation. See figure 5.2. The calculation is to start with the same as for Engine statistical time, but every value is multiplied with the specific fuel injection at the sample time.

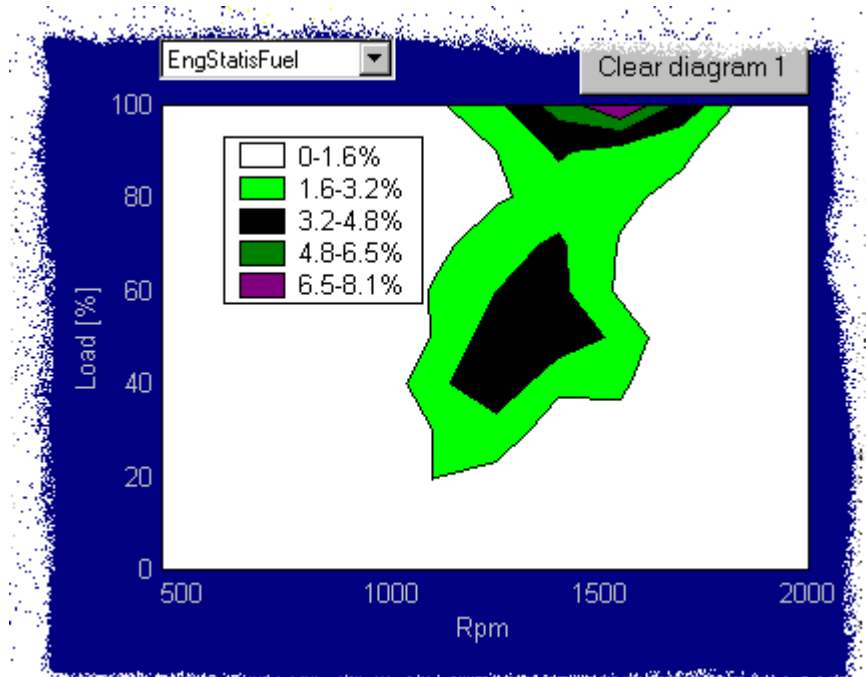


Figure 5.2 *Engine statistical fuel, the same simulation as in figure 5.1. Notice that zero load is the same as zero injection. It shows in the picture that between 6.5%-8.1% of the fuel has been consumed when maximum gas has been pressed and at engine speed around 1500 rpm.*

6 References

- (1) J.Y. Wong: Theory of ground vehicles, 2nd edition, New York, John Wiley & Sons, 1993. ISBN 0-471-52496-4
- (2) Ordinary differential equations, N. Finizio and G. Ladas, 1982. ISBN 0-534-00898-4
- (3) Scientific computing, Michael T. Heath, 1997. ISBN 0-07-027684-6
- (4) Modellbygge och simulering, Lennart Ljung and Torkel Glad, 1991. ISBN 91-44-31871
- (5) Combustion engineering, Gary L. Borman, 1998. ISBN 0-07-006567-5
- (6) Kompendium Maskinteknik Luth: Maskinelement, Utg 3 1990:05
- (7) Rapport C 112/95: Rullmotstånd hos däck. En litteraturstudie, Scania
- (8) Driveline modelling and control, Magnus Pettersson, Linköping 1997. ISBN 91-7871-937-2
- (9) Automotive aerodynamics, Progress in technology series volume 16, 1978
- (10) Automotive handbook, 3rd edition, Bosch
- (11) Carl Martin Allwood: Människa-datorinteraktion, Sweden, Studentlitteratur Lund, 1991, ISBN 91-44-32671-8
- (12) Engineering psychology and human performance, 2nd edition, Christopher D. Wickens, 1992. ISBN 0-673-46161-0

7 Acknowledgement

I want to thank the whole department at DMCA for a nice time. You are a happy crew which made working hours and after working hours more enjoyable. The ski trip to Sälen will be a good memory for me and I hope to see some pictures on a homepage soon. I would also like to thank my supervisor at Scania, Tony Sandberg, for help, when needed during the work. Always close, just a few steps or a quick email away.

In spite of the long distance my supervisor in Luleå, Hans Weber helped often. Thanks for your interest in my work and asking good questions helping me proceed with the work.

Finally I would also like thank Scania for the arrangements for me as a student. The house on Vårdsholmen is a nice environment to live in. Close distance to work and good communications to Stockholm. I reckon Scania will have no problem finding new students to work in Södertälje.

Appendix A

User manual

The main assignment of this Master's thesis was to develop a Graphical User Interface for the simulation program Dymola. This appendix is a user manual that should be used parallel to the GUI in order to run more efficient simulations.

The Stars Graphical User Interface was created with Matlab version 5.3. To run successful and not too slow simulations it is recommended that one uses a computer with at least 32 MB free internal memory and a clock frequency of minimum 150 MHz.

Start up Stars GUI

In the directory "u:\stars" there is an icon "Stars". Copy the icon and paste it on your local desktop. Simply click on the icon to start Stars.



Figure A1 *Short cut to Stars*

Load parts for the truck

When Stars has started a figure will show where one has a number of options to choose from. This is the preparation figure for the simulation. To load parts for the truck you have to press one of the three load-buttons.

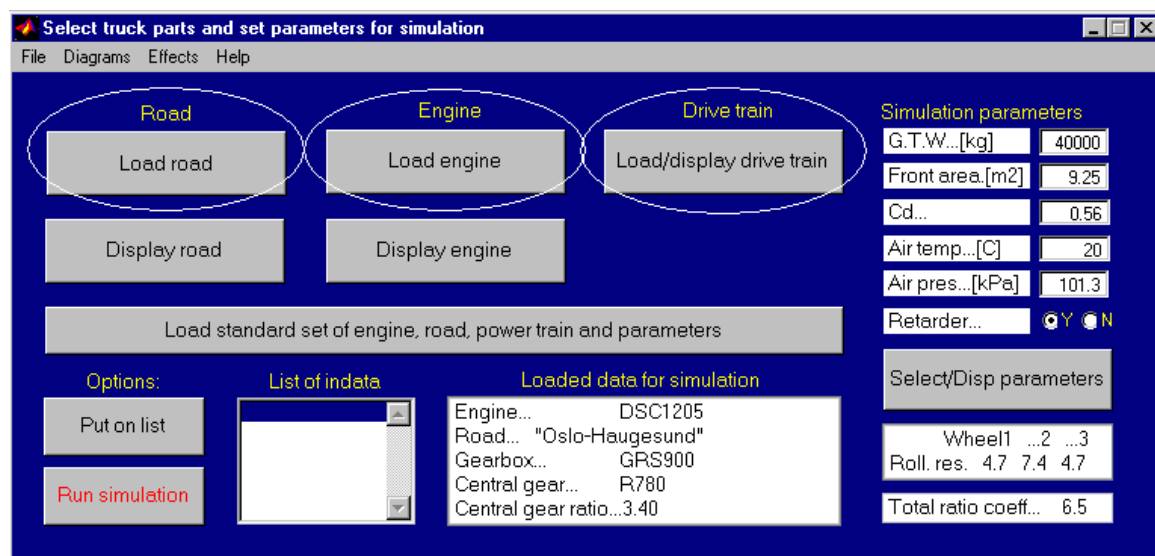


Figure A2 *The three load-buttons (marked with circles) can be pressed to respectively parts.*

The above picture is the first one showed in when Stars is started.

The parameters (weight, frontal area etc.) for the truck can be altered in the edit boxes to the right in the figure. G.T.W. is the total weight of the truck and trailer, Cd is the air resistance coefficient.

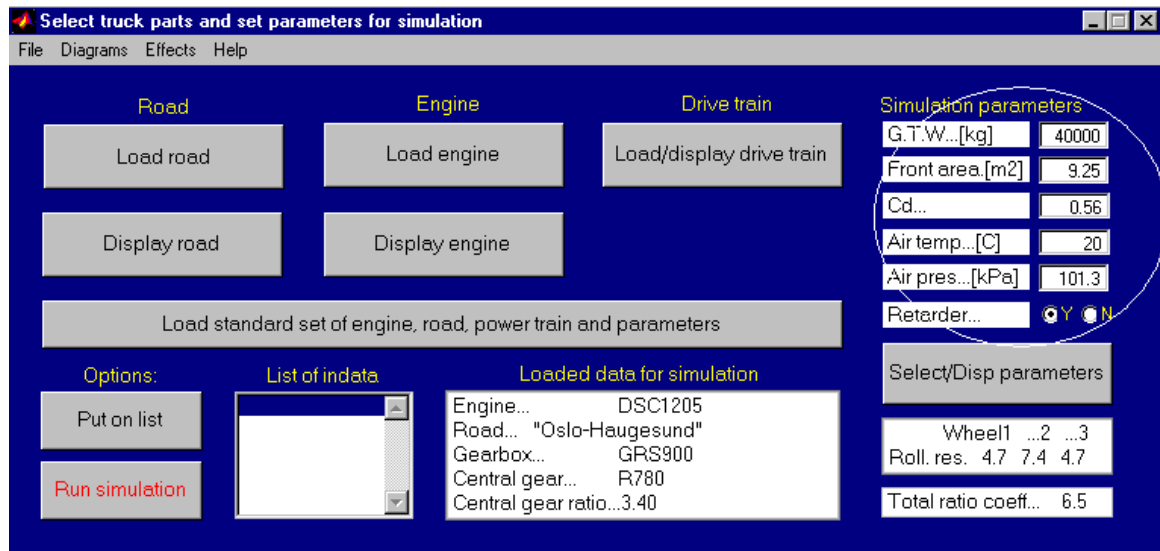


Figure A3 Changes to the parameters can be done to the right in the edit boxes. For further information and options press “Select/disp parameters”.

If one wants to load a standard set for the simulation simply press the long button “Load standard set of engine, road, power train and parameters”.

Display road

When the button “Display road” is pressed a figure with information about the road profile is displayed. The road profiles are numerical values on real roads all around the world. All available information about the profile is showed.

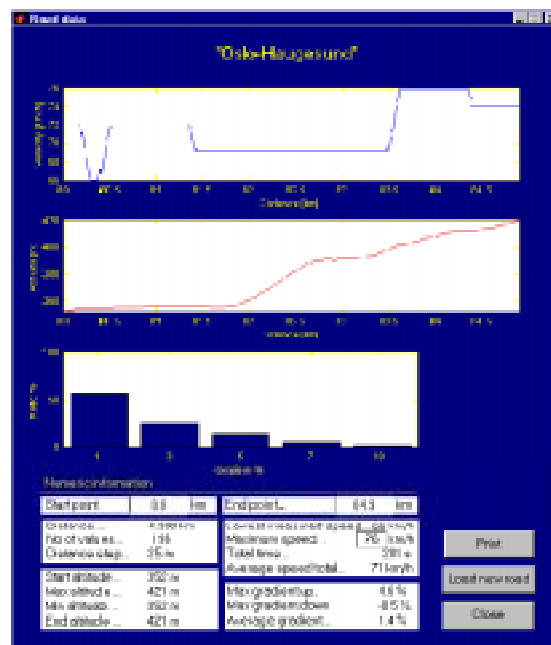


Figure A4 Display road picture

In the top diagram the velocity profile of the recording truck is shown. In the middle diagram the altitude changes are displayed over the road profile. In the bottom diagram the grade distribution is presented. One can see that most of the slope on the loaded profile is between 1-3 %.

To simulate a shorter distance on the road profile, click and drag with the mouse in the two top diagrams. Or write numerical values in the edit boxes “Start point” and “End point”.

Typing a value in the edit boxes “Maximum speed” can change the highest allowed velocity. Doing this will automatically update the velocity diagram.

Display engine

By pressing the button “Display engine” a figure with information about the engine will show. In this figure there are a number of different diagrams that can be plotted. The purpose of this picture is to help the user to get information about characteristics of the loaded engine.

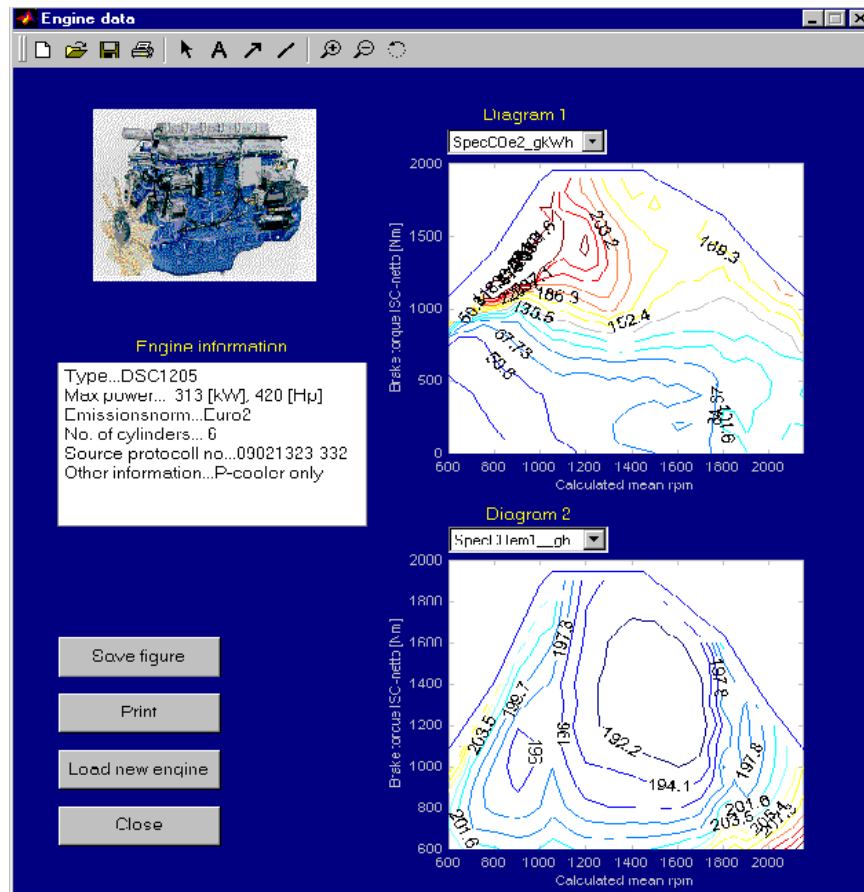


Figure A5 Picture with engine information, displayed when pressing “Display engine”-button

Both “pop-up-menus” above each diagram are viewed by one click with the mouse. A list of available diagrams will show. The names are a little complicated and need an explanation

Exhaust_Brake is the torque which the engine is braking with when no gas is supplied. This is mainly used in the downhill slopes and complemented with the braking force from the retarder.

PowerMap__kW is the Power the engine supplies at a certain engine speed [rpm] and output torque [Nm].

SpecCOe1_gkWh is the specific CO exhaust emission [g/kWh] at a certain engine speed [rpm] and output torque [Nm].

SpecCOem1__gh is same as above but in unit [g/h]

SpecFUE1_gkWh is the specific fueling to the engine [g/kWh] at a certain engine speed [rpm] and output torque [Nm].

SpecFUEL1__gh is same as above but in unit [g/h]

SpecHCe1_gkWh is the specific HC exhaust emission [g/kWh] at a certain engine speed [rpm] and output torque [Nm].

SpecHCem1__gh is same as above but in unit [g/h]

SpecNOX1_gkWh is the specific NOx exhaust emission [g/kWh] at a certain engine speed [rpm] and output torque [Nm].

SpecNOXem1_gh is same as above but in unit [g/h]

TORQUE_Map__1 is the torque [Nm] delivered by the engine at a certain engine speed [rpm] and fueling [mg/stroke].

TORQUE__POWER is the maximum torque and horse powers from the engine over all engine speeds [rpm].

Load/display drive train

The drive train includes gearbox and central gear. Both the gearbox and central gear are objects, which can be selected in the GUI. By pressing the button “Load/display drive train” information and options to load available drive train parts are displayed in a picture.

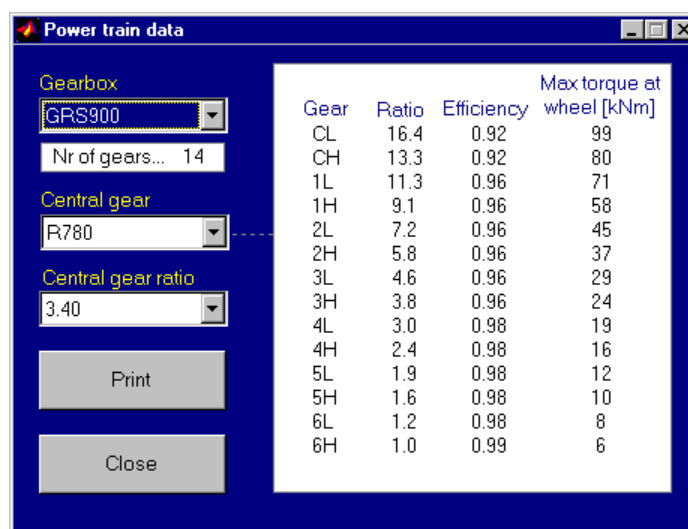


Figure A6 In the pop-up-menus to the right Gearbox, central gear and central gear ratio can be displayed. In the diagram to the right in the picture information about the gearbox is viewed. The “Max torque at wheel” is the total ratio, including central gear ratio multiplied with maximum torque from the engine.

Select/disp parameters

The button “Select/disp parameters” is an option for further information and choices regarding parameters on the truck. The parameters are mainly wheel dimensions and rolling resistance. Additional options to adjust the weight on the truck and trailer separately and the characteristic lengths computing the frontal area are also displayed when pressing this button.

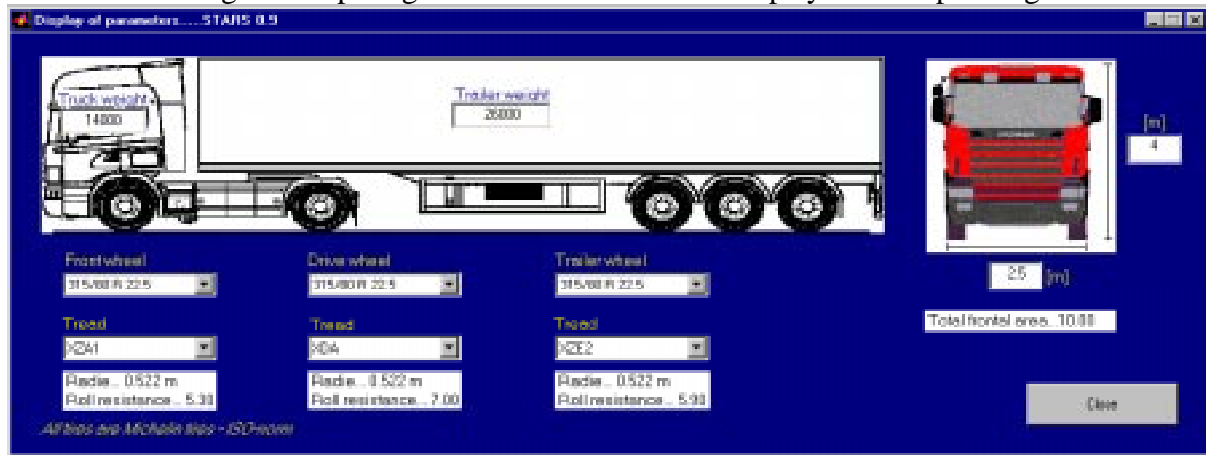


Figure A7 Picture when “Select/disp parameters” is pressed

Put on list and run simulation

“Put on list” is a button function in the preparation picture of the GUI. When pressed a new Data1 will show in listbox underneath “List of data”. A new data is saved for each push of the button. To view and be able to change the data click on the data number and the GUI will automatically load and display information on the selected data. When selected the data will also be saved every time changes are made to parameters and/or objects.



Figure A8 After two pushes of the button “Put on list” two different data files are saved 1 and 2. A data file in the “List of data” can be changed by first selecting it and then make ordinary changes in the GUI to parameters and objects. The data files are automatically updated.

The purpose of this function is mainly to prepare a number of different, limited by ninety-nine, simulations. Then starting the simulation and let the computer calculate during the night. When coming back to the office in the morning, desired results for all the simulation can be studied.

The red button “Run simulation” is the option that starts the simulation. When pressed the program uses the “List of data” (if activated) and repeats the simulation for all prepared data.

The system status on the calculation is viewed in the Matlab prompt as it progresses. The result is displayed in a new window that automatically starts when the calculations are done.

Simulation results

The “Simulation result” picture shows when the simulation is ready. Here all important information on the simulation is displayed. The look of the picture is similar to the “Display engine” picture. The picture looks as follows

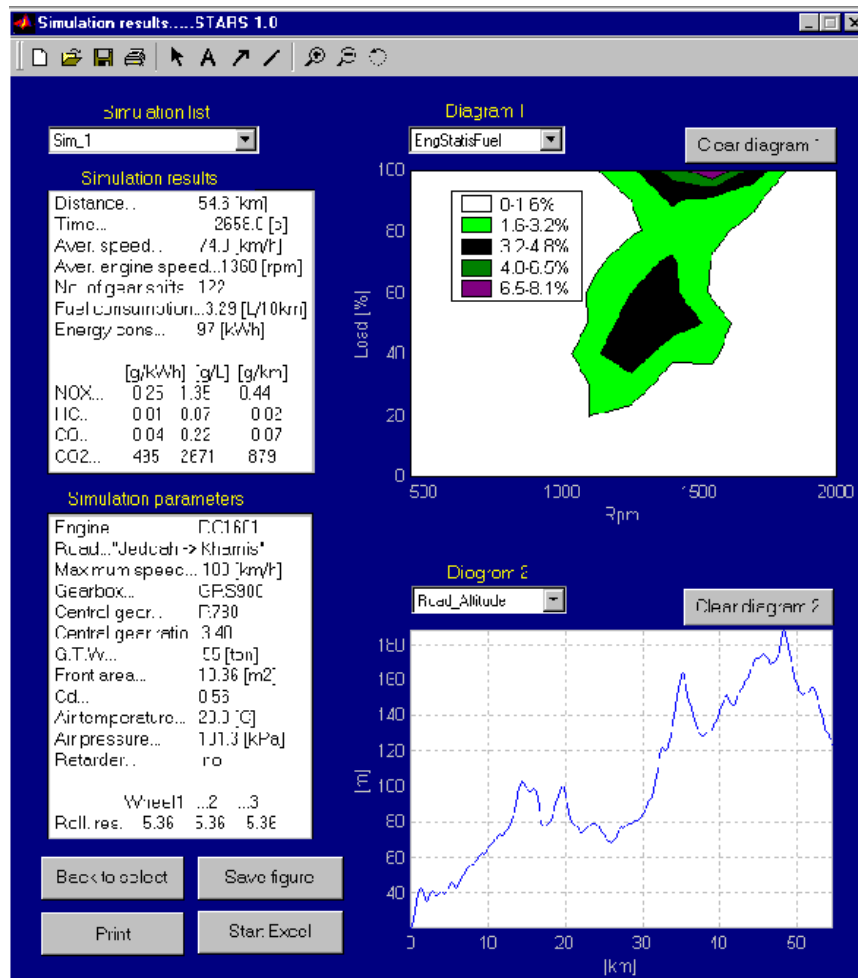


Figure Two optional diagrams can be viewed above each diagram. If a couple of simulation were run (See Put on list) the simulation list will have the same number of simulation results to choose from.

The diagrams that can be chosen in the pop-up-menus are the following

Air_resistanc is a plot of the air resistance force acting on the truck during the simulation

EnginePowerHP plots the engine effect in horse power [HP]

EnginePowerKW same as above but in [kW]

Engine_Fuelin plots the fueling to the engine

Engine_Torque plots the output torque from the engine

Engine____rpm plots the engine speed [rpm]

EngStatFuel plots the statistical fuel consumption of the simulation with Load [%] on the y-axis and engine speed [rpm] on the x-axis. This function is further described in chapter 5.3.

EngStatTime plots the statistical working time of the engine during the simulation. Load [%] on the y-axis and engine speed [rpm] on the x-axis.

ExhaustBrake plots the exhaust brake torque [Nm] the engine produces during the simulation

Gear___Shifts plots the gear number as a step plot

Road_Altitude plots the altitude profil of the simulation road

Truck_speed plots the simulation speed of the truck and the reference speed of the road profil.

Whe1_NorForce is the normal force on the front tyre

Whe2_NorForce is the normal force on the driving tyre

Whe3_NorForce is the normal force on the trailer tyre

Wheel1_RolRes is the force of rolling the resistance on the front tyre

Wheel2_RolRes same as above but on the driving tyre

Wheel3_RolRes same as above but on the trailer tyre

Start Excel

In the “Simulation results” picture one has the option of a button “Start Excel”. When pressing this button the user will be offered to choose what variable he wants to view in Excel.

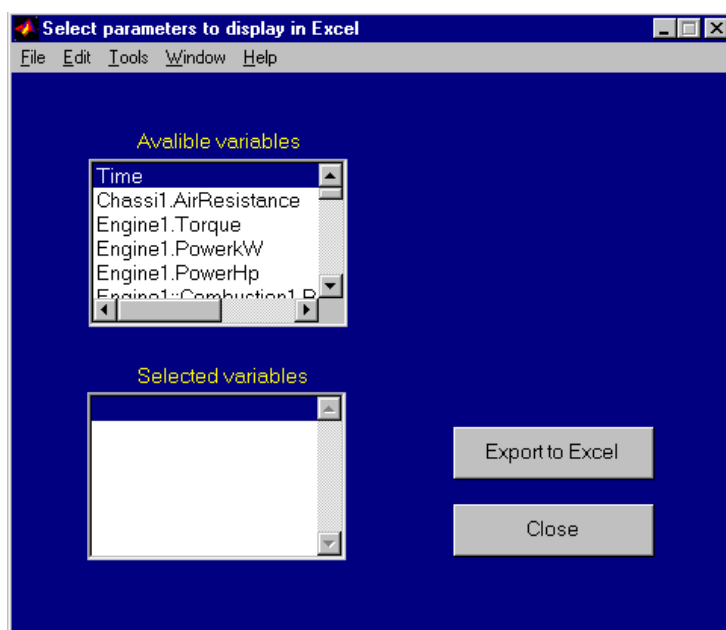


Figure When pressing the button “Start Excel” the above picture will appear. Click on the variable names in “Available variables” and they will be displayed in “Selected variables”. Then push the button “Export to Excel” and the values for each will appear in every time step of the simulation.

Appendix B

GUI objects

For the more advanced user of the Stars GUI it could be of interest to make changes to the available truck parts. This chapter explains how this could be achieved. A GUI object is a part that can be changed and modified. The Stars GUI was built to make a number of objects modifiable. In this way one can update or modify truck parts and still simulate them. The objects are Engine, Gearbox, Central gear, Road and Wheels and stored as mat-files. These objects are different from parameters since they have some special characteristics. A parameter on the other hand is just a number that is used in the calculations. There are also two plot objects engine plots and result plots, which could be extended. How you declare objects will be described in this chapter.

Objects in general can, in the Stars GUI, be described as free lying data parts. An object communicates with the GUI from the way they are named. Since the code knows what a certain variable name should contain the information is used in the right way. Every variable name is needed to run a successful simulation. The size of the variable vectors is free to vary since the code is universal. This means that the amount of data could vary as long as it is consistent through the whole object.

Engine

The engine object is built from data recorded while running the engine. Data is saved in a mat-file (.mat). See Matlab manual for further instructions. All different engine files must be stored in the directory "...\\engine\\". No other object and no other files can be stored here. The GUI reads all files from this directory as possible engines for the truck. The filename must be seven letters long and written with capital letters.

The whole engine object (mat-file) consists of the following vectors. Display the vectors by loading the mat-file into workspace. Use command load from the Matlab prompt. Display the workspace with command whos.

CO1	21x32	5376 double array
ExhaustBrake	1x11	88 double array
F2T1	33x24	6336 double array
Fueling1	23x1	184 double array
HC1	21x32	5376 double array
NOX1	21x32	5376 double array
No_Cyl	1x1	8 double array
Norm	1x5	10 char array
PowerMap	21x32	5376 double array
Protocol	1x12	24 char array
Rpm1	1x32	256 double array
RpmExh	1x11	88 double array
RpmTmax	1x20	160 double array
T2CO1	33x22	5808 double array
T2HC1	33x22	5808 double array

T2NOX1	33x22	5808 double array
Torque1	21x1	168 double array
TorqueMap1	23x32	5888 double array
TorqueMax	1x20	160 double array
bsfc1	21x32	5376 double array
enginename	1x7	14 char array
extra	1x13	26 char array
flim_n	1x16	128 double array
flim_q	1x16	128 double array

Where the vectors can be explained as follows

CO1 is the CO-matrix with Rpm1 on the x-axis and Torque1 on the y-axis

ExhaustBrake is the braking torque of the engine. Plot the vector with the vector RpmExh on the x-axis. The plotting command can be written plot(RpmExh,ExhaustBrake). [Nm]

F2T1 fuelling and torque matrix used for the calculations in Dymola

Fueling1 is a vector for defining the correct values of the rows of the matrix TorqueMap1. [mg/stroke]

HC1 is HC matrix with Rpm1 on the x-axis and Torque1 on the y-axis

NOX1 is the NOX matrix with Rpm1 on the x-axis and Torque1 on the y-axis

No_cyl is the number of cylinders for the engine. Parameter number

Norm is the definition of the norm for the engine. I.e “Euro 2”. Character string

PowerMap

Protocol a string array containing the information of what protocol number the engine data was detected from.

Rpm1 is the x-axis definition for CO-matrix etc.

RpmExh see ExhaustBrake

RpmTmax plotted with TorqueMax. Command used plot(RpmTmax,TorqueMax)

T2CO1 modification of CO1 for use in Dymola only

T2HC1 same as above

T2NOX1 same as above

Torque1 row definition of all matrixes CO1,...,NOX1

TorqueMap1. Rows defined by Fueling1 and columns by Rpm1

TorqueMax see RpmTmax. [Nm]

bsfc1 specific fueling matrix. Rpm1 is the x-axis definition and Torque1 is the y-axis

enginename is the name of the engine. I.e “DSC1205”. It is a character string. The name is not restricted to any number of characters, but seven letters is recommended and one should use capital letters.

extra is additional information about the engine. The text is viewed in button function “Display engine”

flim_n is the rpm vector to flim_q

flim_q maximum fuel injection at the rpm range. Plotted by plot(flim_n,flim_q).[mg/stroke]

These vectors must have these specified names otherwise calculations in the GUI will not work. Note: Capital letters must be defined in the vectors in the same way as above. The length and size of the vectors are free to vary as long as they still fit to their relatives.

Gearbox

The gearbox object is defined by the mechanical properties of the specific gearbox. Data to gearboxes can be found in Scania literature or from separate tests. Gearbox mat-files are stored in the directory ...\\gearbox. Each gearbox file must be six letters long. A gearbox file consists of the following vectors

GearNo	1x12	96 double array
geareff1	1x14	12 double array
Gearefficiency	1x12	96 double array
gearname	1x6	12 char array
gearstepnames	14x2	56 char array
Ratio	1x12	96 double array
ratio1	1x14	112 double array
type	1x1	8 double array

GearNo is a string of numbers. In this case [1 .. 12]. Note that this vector neglects the two creeping gear steps.

geareff1 is the efficiency coefficients for all gear steps including the two creeping gear steps.

Gearefficiency is the same as above except the two creeping gears. Both are necessary since the simulation does not use creeping gears in the calculations.

gearname is the name of the gearbox. Must have the same order of capital- or non-capital

letters as the mat-file.

gearstepnames are the real names of the gear steps. Define the vector by using command `sprintf`. I.e `gearstepnames=sprintf('CL\nCH\n1L...\n6H')`.

Ratio/ratio1 are the coefficient of ratio for every gear step. A separation in two vectors must be done. See `gearefficiency`.

type is set to one for a gearbox with fourteen steps and two for a gearbox with eight steps. Note that this is not the same as the type parameter in the dymola modell. This parameter in Dymola is also depended on what engine is to be simulated.

Central gear

The defined central gears in Stars are placed in the directory ".../centralgear". The complete central gear object contains the following vectors

cegearname	1x4	8 char array
centralgeareta	1x1	8 double array
centralgearratio	7x4	56 char array

cegearname is the name of the central gear. E.g. "R660". Use capital letters!

centralgeareta is the efficiency coefficient of the central gear. Generally one can assume that every cog in the central gear contributes with 1% power loss.

centralgearratio is ratios available for the central gear. The ratio numbers must be written as chars. Use the command `sprintf` to define the numbers. E.g. for R660 one would write the string as `centralgearratio=sprintf('5.57\n4.88\n...\n2.92')`.

Road

All saved road files are stored in the directory "\Road". They are mat-files and was converted and developed with the file `invej1.m` and `vejstr.m`. These command files could be found in the directory "\reading_routines". All road mat-files has its origin from the vej-files. These files are the same as used in Strass. Some of the data in the vej-files has been cut out since it is of no use in Stars.

All road mat-files consists of the following vectors

Altitude	1x1357	10856 double array
Krtid	1x1	8 double array
Lutn	1x7	56 double array
Position	1x1357	10856 double array
Roadname	1x6	12 char array
Slope	1x1357	10856 double array
Tst	1x1	8 double array
Velocity	1x1357	10856 double array
Vlim	1x1	8 double array
Vmin	1x1	8 double array
line1	1x38	76 char array
nstop	1x1	8 double array
steg	1x1	8 double array

Altitude is the height above sea level in each measured point of the road

Krtid is the time it took the truck to drive the while recording the road. The truck drove with the RecordedVelocity-vector

Lutn is the percentage ratio of slope of the road in intervals of 1-3%, 3-5%, 5-7% and 7-% road slope

Position is the distance vector of the road. Position(length(Position)) is the total distance of the road

Roadname is a vector containing the file name (.mat) of the road

Slope is the slope in each measured point on the road. The Slope-vector was calculated according to the following expression

$$Slope(i) = \frac{Altitude(i+1) - Altitude(i)}{steg}$$

Tst is the Total stop time. The truck that recorded the road did move during a total time of Tst. This vector is left for future extensions.

Velocity is the same as RecordedVelocity

Vlim is the maximum allowed speed on the whole road

Vmin is the minimum speed restriction on the whole road

line1 is the name of the road. I.e.

line1 = 3,"Singapore Route 153 - 4","12 - 04 - 1993"

Not. The GUI will look for (“) and thereby finding the right string to view

nstop is the number of stops on the road while recording

steg is step length interval in [m]. Recording was done of road data was done one time each steplength

Wheels

Relevant wheel data for the GUI and simulation are the wheel's name, number of different treads, radius, tread names and specific rolling resistance.

The characteristics of the wheels are implemented in the m-file called ('wheelstr.m'). This file declares the following variables

```
s.whstr=strcat('295/80 R 22.5','315/80 R 22.5','305/70 R 22.5','315/70 R  
22.5','385/65 R 22.5','385/65 R 22.5');  
  
s.whstrnum=[7,0.507;8,0.522;4,0.485;4,0.492;1,0.536;1,0.533];  
  
s.whstrtreadname=['Energy XZA ','XZA1 ','XZE2 ','Energy XDA  
,','XDA ','XDY ','XDE2 ','#Energy XZA ','XZA1  
,','XZY ','XZE2 ','Energy XDA ','XDA ','XDY  
,','XDE2 ','#Energy XZA ','XZA1 ','Energy XDA ','XDA  
,','#Energy XZA ','XZA1 ','Energy XDA ','XDA ','#Energy  
XTA2','XTE2 ','#'];  
  
s.whstrrollres=[4.8,5.3,6.1,5.45,7.1,6.25,7.62,4.7,5.3,5.3,5.9,5.36,7,  
6.13,7.4,5.13,5.74,5.35,7.71,5.1,5.61,5.36,7.37,4.5,5.3];
```

s.whstr is the name of the tyre. To add new tyres simply type the new name of the tyre at the end in the same structure. Add in the same way for all variables below.

s.whstrnum is the number of treads, tyre radius for the wheels declared in s.whstr. I.e. [7,0.507] means that tyre '295/80 R 22.5' has seven different treads (s.whstrtreadname) and radius 0.507 meter

s.whstrtreadname is the name of the treads. Each new tyre must have a (#) in the beginning of the first treadname. The treadname must be eleven chars long.

s.whstrrollres is the specific rolling resistance coefficient corresponding with s.whstrtreadname

Engine plots

The engine plots are plotted from the figure "Engine data" (eng411.fig), which is viewed by pressing the button "Display engine" in the "Select parts for the truck"-figure. All plots are regular m-files which are stored under the directory "...\\engine_plots\\". The GUI reads all files from this directory and displays the name of the m-files in the two pop-up-menus. Each engine plot file must have the following function call in the beginning

```
function SpecCOem_gkWh(e,ph)
```

where e is the global vector containing all vectors from the engine data. I.e. $e.Rpm1$. ph is the pointer to the correct axis. With the help of these any engine specific data can be plotted. All available engine plots are listed in Appendix A.

Result plots

The result plots are used in the figure “Simulation results” (result.fig). The same idea with the pop-up-menus are used as in the “Engine data”-figure. The result plot are save under the directory "...\\result_plots\\". The GUI reads this catalogue each time it starts result.fig. The available result plots are listed in Appendix A.

All result plots start with the following function call.

```
function Engine_Torque(s,n,ph)
```

s is a matrix containing all data from the simulation. n is an index matrix helping to find specific data is the s matrix. ph is the pointer to the current axis.