Influence of performance degradation on vehicle dynamics

MATHIEU MASSOLO
4.8.1 Highest P gain .............................................. 22
4.8.2 Lowest P gain ............................................. 23
4.8.3 Mid P gains ................................................. 24
4.9 Slip ratio ...................................................... 24
4.10 Conclusion .................................................. 25

5 BTC Analysis .................................................. 27
  5.1 Testing protocol ............................................. 27
  5.2 Control system presentation .............................. 27
  5.3 Measurement analysis ..................................... 27
  5.4 Data reliability ............................................. 28
  5.5 Longitudinal acceleration ................................. 29
  5.6 Wheel speed ................................................. 31
      5.6.1 Very low gain setting ................................. 31
      5.6.2 Low gain setting ....................................... 32
      5.6.3 High gain setting ....................................... 33
  5.7 Brake pressure ............................................. 34
  5.8 Axle and target slip ....................................... 35
  5.9 Conclusion ................................................. 35

6 Conclusion .................................................... 37

7 Future work .................................................. 37

References ....................................................... 38

A Appendix ...................................................... i
  A.1 MATLAB implementation ................................. i
  A.2 mesurefctdivsel.m .......................................... iv
  A.3 premmesure.m ................................................. vi
  A.4 addmatfile.m .................................................... viii
  A.5 ParsePMSD.m ..................................................... ix
  A.6 buttonnom.m ..................................................... xi
  A.7 buttonpmsd.m ................................................... xii
  A.8 buttonmeas.m ................................................... xiv
  A.9 makeplot.m ..................................................... xv
  A.10 myslider.m ..................................................... xvi
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Split view of a Bosch ESP. Middle part: ECU. Right part: Pump and valves. Taken from [2].</td>
</tr>
<tr>
<td>2.2</td>
<td>Tread speed difference. Taken from [9].</td>
</tr>
<tr>
<td>2.3</td>
<td>Difference in traction and braking force. Measurements of an internal drum test of a 195/65 R15 V tire. Vertical load: 4 kN. Velocity: 100 km/h. Taken from [9].</td>
</tr>
<tr>
<td>2.4</td>
<td>Slip angle definition. Taken from [9].</td>
</tr>
<tr>
<td>2.5</td>
<td>Contact patch when cornering. Taken from [9].</td>
</tr>
<tr>
<td>2.6</td>
<td>Friction ellipse. Taken from [9].</td>
</tr>
<tr>
<td>4.1</td>
<td>Ford Tourneo Connect. Taken from [7].</td>
</tr>
<tr>
<td>4.2</td>
<td>PTC manoeuvre, longitudinal acceleration.</td>
</tr>
<tr>
<td>4.3</td>
<td>PTC manoeuvre, longitudinal acceleration study, extra curves removed.</td>
</tr>
<tr>
<td>4.4</td>
<td>PTC manoeuvre, target torque.</td>
</tr>
<tr>
<td>4.5</td>
<td>PTC manoeuvre, target torque, low P only.</td>
</tr>
<tr>
<td>4.6</td>
<td>PTC manoeuvre, delivered torque for high P gains.</td>
</tr>
<tr>
<td>4.7</td>
<td>PTC manoeuvre, delivered torque for low P gains.</td>
</tr>
<tr>
<td>4.8</td>
<td>Wheel speed for highest P gain.</td>
</tr>
<tr>
<td>4.9</td>
<td>Wheel speed for lowest P gain.</td>
</tr>
<tr>
<td>4.10</td>
<td>Wheel speed for mid P gains.</td>
</tr>
<tr>
<td>4.11</td>
<td>PTC manoeuvre, axle slip and target.</td>
</tr>
<tr>
<td>4.12</td>
<td>Wheel speed linear regression coefficient.</td>
</tr>
<tr>
<td>5.1</td>
<td>BTC manoeuvre, driver torque request.</td>
</tr>
<tr>
<td>5.2</td>
<td>BTC manoeuvre, longitudinal acceleration.</td>
</tr>
<tr>
<td>5.3</td>
<td>BTC manoeuvre, longitudinal acceleration.</td>
</tr>
<tr>
<td>5.4</td>
<td>BTC manoeuvre, wheel speeds from medium and very low gain settings.</td>
</tr>
<tr>
<td>5.5</td>
<td>BTC manoeuvre, wheel speeds from medium and low gain settings.</td>
</tr>
<tr>
<td>5.6</td>
<td>BTC manoeuvre, wheel speeds from medium and high gain settings.</td>
</tr>
<tr>
<td>5.7</td>
<td>BTC manoeuvre, brake pressure in the right wheel.</td>
</tr>
<tr>
<td>5.8</td>
<td>BTC manoeuvre, right wheel slip and target slip.</td>
</tr>
<tr>
<td>A.1</td>
<td>Example of main window.</td>
</tr>
<tr>
<td>A.2</td>
<td>Search box for parameters.</td>
</tr>
<tr>
<td>A.3</td>
<td>Search results.</td>
</tr>
<tr>
<td>A.4</td>
<td>Example of a plot with all types of parameters included.</td>
</tr>
</tbody>
</table>
Abstract

Technology&Strategy is a French-German consulting company based in Strasbourg, France. Most of its activities take place in the automotive sector with many of their consultants working for Bosch Chassis Control in Abstatt, Germany.

Bosch Chassis Control is responsible for the development of active safety control systems for many car manufacturers worldwide. The ESP (Electronic Stability Program) was created by Bosch and is now compulsory in every European car since 2014. The development of the ESP includes an important part of onboard testing before the production release. The application team is responsible for achieving those tests and analyzing the data recorded.

One of the features of the ESP is the TCS (Traction Control System) allowing the vehicle to remain at its maximum friction available depending on several parameters such as road friction, speed or wheel angle. The goal of this master thesis is to study the control logic of the TCS and highlight the influence of several parameters from the software to the vehicle dynamics.

The first part of this thesis was dedicated to the development of a MATLAB tool used to visualize measurements out of different test sessions on the same plot. The current tool used is limited to a single measurement display.

The second part focuses on the analysis of TCS data taken through several test sessions with different parameters. The two main controllers of the TCS, PTC and BTC (Powertrain Traction Control and Brake Traction Control) are the core of this analysis.
Acknowledgements

First and foremost, I would like to thank all of those who allowed me to finalize this thesis. My supervisor at Bosch Chassis System Control, Samir Bounabi, has been of precious help throughout those last months. I thank him deeply for all the knowledge that he shared with me, his patience and dedication. It has been an honor to work with someone which shares the same passion for cars. My colleagues, Robin Camus and Thomas Bertossi, also deserve their place here, as their kindness and their willingness to show me their work really motivated me to keep going.

Furthermore, I would also like to thank my supervisor at KTH, Mikael Nybacka, not only for having accepted to supervise this thesis, but also for his implication throughout the whole Vehicle Engineering master’s program and in the KTH Formula Student project in which I had the chance to be part of for one semester. Leaving France to enroll in a double degree program was one of the best decision I’ve ever made.

Finally, I have a thought for my home university, Centrale Nantes. I would like to thank the International Relations Office for having let me go abroad and seize the opportunity to get a double degree in Sweden. I realize today how lucky I was to be selected for this program and it deserves to be mentioned here.
1 Introduction

This master’s thesis was carried out on the Bosch Chassis System Control site in Abstatt, Germany for the account of Technology & Strategy, consulting company based in Strasbourg, France. The examiner of the master’s thesis is Associate Professor Mikael Nybacka at the division of vehicle dynamics at KTH Royal Institute of Technology in Stockholm, Sweden. Supervisor at Technology & Strategy is Samir Bounabi, Application engineer at Bosch Chassis System Control.

This thesis concludes a Master of Science in Vehicle Engineering carried out at KTH in Stockholm, Sweden, obtained through a double degree with the École Centrale de Nantes, France in the Ingénieur Centralien program.

1.1 Background

The background of this master’s thesis is the core of the Electronic Stability Program (ESP). The ESP invented by Bosch and released first in 1995 on a series car gathers data from different sensors and acts accordingly on several parameters to improve vehicle dynamics in case of grip loss. New functions have been added to the ESP throughout its development, including Anti Blocking System (ABS), Traction Control System (TCS) or Vehicle Dynamics Control (VDC).

Each of these systems have to be tuned accordingly to each vehicle during their development, as the chassis, weight and tires have a big influence over the vehicle dynamics. The real-world on-road testing is carried out by the application team in Bosch Chassis System Control. The goal of these tests is to check the launch conditions of each function along with good parameter settings in terms of vehicle dynamics and also driver feedback.

1.2 Aim

The aim of this master’s thesis is to understand the influence of different parameters inside the control system on the vehicle dynamics. This goal is reached through two different steps:

- Development of a MATLAB tool to visualize signals from different measurement sets on the same plot
- Analysis of real-world measurements taken from test sessions with different parameters

1.3 Limitations

As most of the control logic is Bosch property and is therefore confidential, this thesis will not focus on the control system itself. The influence of the parameters will only be studied through the results obtained by the measurement system installed inside the test vehicle.
2 Theory

This chapter is intended to give some theoretical information necessary to understand the goals and results of this thesis.

2.1 ESP presentation

The Electronic Stability Program developed by Bosch Chassis Systems Control is made of two different parts:

- An Electronic Computer Unit (ECU) responsible for collecting data from all sensors and processing them. It is linked to the whole electronic system of the vehicle through Controller Area Network (CAN) and more recently FlexRay. A more advanced description of both of them can be found in section 2.1.2.

- An hydraulic module with pump able to modulate brake pressure.

![Figure 2.1: Split view of a Bosch ESP. Middle part: ECU. Right part: Pump and valves. Taken from [2]](image)

2.1.1 ESP development history

One of the main functions now part of the ESP is the ABS (anti blocking system) first released in 1978 on the Mercedes S-Class and BMW 7-Series. Its goal is to reduce the hydraulic pressure in the brakes during a braking sequence where the wheels are supposed to get locked and thus slip.

Five years later, the development of a similar system that would keep the vehicle stable while braking in corner was launched by Bosch. Released in 1995 on the Mercedes S-Class, the VDC (Vehicle Dynamics Control) uses the hydraulic system to brake one wheel in case of under or oversteering to gain stability back.

An important part of this master’s thesis is based on the Traction Control System. The first appearance of a similar system is the Buick MaxTrac in 1971 [11]. In this system, a controller would modulate the engine torque when the speed difference between undriven front wheels and
rear wheels was too high. In 1986, Bosch released in series its Traction Control System.

All functions are explained more in detail in section 2.3.

2.1.2 Communication protocols

As mentioned previously, two types of communication protocols are widely used in automotive controls, CAN and FlexRay. This part will describe their main differences and their respective advantages leading to their use in vehicle dynamics control.

2.1.2.1 Controller Area Network (CAN)

Developed in the 1980s by Bosch, the CAN is still used today for all the electronic network of most vehicles. CAN allows multiplexing: several signals can be transmitted through the same cable. This is a big asset in the automotive industry in order to save weight.

The CAN bus is a multimaster asynchronous bus: without common memory, traffic controller nor global clock, any transmitter can emit a message as soon as the bus is free. The collisions between transmissions are solved through a priority system: if two messages are sent at the same time, the one with the highest priority will be sent first without any destruction of the other messages. If an error is detected, the error notice is sent to all stations. The message corrupted by the error is then emitted once again.

The advantages of the CAN are its liability (probability of undetected error < $4.7 \cdot 10^{-11}$), its price due to the use of twisted pair wires and the weight gain made possible by the multiplexing [4]. Several CAN networks are usually present in vehicles in order to reduce even more the risk of errors (parallel networks).

Yet, it is relatively slow (maximum 1Mbit/s) and still lacks reliability for new applications such as by-wire concepts (brake-by-wire, shift-by-wire, steer-by-wire) where digital transmission replaces all or part of mechanical transmissions. FlexRay was therefore developed to meet the requirements of technology development.

2.1.2.2 FlexRay

Compatible with other communication protocols such as CAN, the FlexRay is a deterministic protocol with transmission speeds up to 10Mbit/s. Its messages are divided into a static and a dynamic part allow real-time communication in a more robust way than CAN. Being dual-channel, the safety critical applications are ensured through redundancy, the message being transmitted even in case of damage on one channel. [5]

The static part is time-driven while the dynamic part is event-driven. The time-driven part allows each node to send messages only during a certain period of time, and the event-driven part sends messages with their priority set by the waiting time.

FlexRay is expected to become the next automotive standard in communication protocol in the upcoming years due to its speed, liability and compatibility with other protocols. Yet, its cost is still higher than CAN: a FlexRay node costs around 6$ to be implemented while a CAN node costs only 3$ [3].
2.2 Tire physics and need for ESP

The principle of most ESP functions is based on the tire-road interface. According to [8], pneumatic tires ensure several functions for the vehicle equipped:

1. Support the load of the vehicle
2. Minimize the necessary vehicle driving force by driving motion
3. Enable speed changes through traction and braking forces
4. Alter/maintain vehicle direction of travel through steering forces
5. Cushion the vehicle from road disturbances

The 3rd and 4th points are especially related to vehicle dynamics, respectively to longitudinal and lateral dynamics. Due to the visco-elasticity of the rubber used in tire manufacturing, grip can be generated thanks to the hysteresis of the material in both directions (acceleration and deceleration) but can also be exceeded and make the tire slip.

2.2.1 Longitudinal tire forces

When a tire is not free-rolling, a velocity difference appears between the tread layer inner and outer surface as seen in figure 2.2.

The rubber visco-elasticity allows for a deformation of the contact patch induced by the tire-road friction. A way of defining this speed difference is the longitudinal slip ratio. $\omega$ represents the angular wheel velocity, $R_e$ the wheel radius and $v_x$ the vehicle longitudinal speed.

$$s = \frac{R_e \omega - v_x}{v_x}$$  \hspace{1cm} (1)

The slip ratio as defined in equation (1) is usually arbitrarily bounded: [9]

- Wheel slipping while the vehicle is standing still:
  $$v_x = 0, \omega \neq 0 \implies s \to \infty$$
  Usually bounded with $\omega = \frac{2v_x}{R_e} \implies s = 1$

- Locked wheel:
  $$\omega = 0 \implies s = -1$$

\[\text{Figure 2.2: Tread speed difference. Taken from [9]}\]
The longitudinal force that can be generated by the tire can be plotted as a function of this longitudinal slip ratio for a given surface (with constant friction) as seen in figure 2.3.

Vertical load also has an influence on the generated longitudinal tire force. The optimal behavior is reached when the tire can generate the biggest longitudinal force available. The principle of remaining as close as possible to the optimal slip ratio is the idea behind two of the main ESP functions, the Traction Control System (TCS) and the Anti-lock Braking System (ABS).

2.2.2 Lateral tire forces

2.2.2.1 Slip angle definition

The slip angle \( \alpha \) is defined as the angle between the wheel plane and the wheel center velocity as shown in fig.2.4. When cornering, the wheels do not travel in a perfect parallel way to their plane but follow a smaller yaw angle. This is due to the tire structure where each contact patch is displaced from the other given the lateral distortion.
2.2.2.2 Tire deformation

While cornering, the elastic region of the tire deforms itself due to the lateral force created by steering. For small slip angles, the contact region between road and rubber does not slip. When the adhesion limit is reached, a part of the contact patch is unable to sustain a bigger deformation and starts sliding as shown in fig.2.5.

![Figure 2.5: Contact patch when cornering. Taken from [9]](image)

The contact patch is therefore divided between an adhesion and a sliding zone.

2.2.2.3 Friction ellipse

Each tire can only sustain a maximum force in any direction. To symbolize this force, a graphical representation such as fig.2.6 known as the friction ellipse can help understand some phenomenon.

![Figure 2.6: Friction ellipse. Taken from [9]](image)

The idea behind the friction ellipse is that a tire cannot sustain as much steering angle when braking or accelerating as it could while free rolling. The consequences are primordial in vehicle dynamics: when a wheel is saturated in one direction (either slipping, blocked during braking without ABS or oversteered) it will not be able to create any force in other directions. This leads to both oversteering or understeering: both can be avoided thanks to the Vehicle Dynamics Control (VDC), another function of the ESP.
2.2.2.4 Over- and understeer

Oversteering is a common behavior for rear-wheel driven vehicles. When entering a curve while accelerating, the driven wheels reach their maximum friction in the longitudinal direction and are therefore unable to sustain any lateral force. This induces slipping and makes the vehicle very unstable.

Understeering happens usually on front-wheel driven vehicles. While braking or accelerating too much, front wheels are saturated in the longitudinal direction and the driver is unable to steer anymore.

Brakes are usually set to prioritize an understeering behavior. Understeering is easier to handle for a driver, as oversteering will make the vehicle spin out. Better stability is thus ensured[6].

2.3 Main ESP functions

2.3.1 Anti-lock Braking System

2.3.1.1 Goals and principle

The goal of the ABS is to reduce the brake pressure when the tire is saturated in the longitudinal direction in order to limit the slip. The two main consequences on the vehicle dynamics are:

• Gaining back steerability
• Ensure vehicle stability
• Remaining closer to the optimal slip ratio in the longitudinal direction

As explained in section 2.2.2.3, a vehicle without ABS is not steerable when its wheels are blocked during an emergency braking. The ABS modulates the brake pressure in order to limit the longitudinal force and thus allowing more lateral displacement on the friction ellipse. Emergency manoeuvres are then possible (such as double lane change) even with the brake pedal fully pressed.

ABS must also be able to stabilize the car even when braking on uneven friction surface. On this type of ground called \(\mu\)-split (\(\mu\) is usually used to represent the friction coefficient), the wheels on the highest friction surface will have more braking force than the other side creating an extra yaw moment that the ABS should be able to compensate.

The braking distance on straight lane can also be reduced due to the longitudinal braking force remaining around its highest level, especially on low friction surfaces such as wet asphalt or ice.

2.3.1.2 Technical insight

The ABS uses the traditional brake system completed with wheel speed sensors and additional valves and pumps. The ESP maintains the brake pressure when the wheel deceleration reaches a negative threshold, and reduces it if the wheel speed also goes behind a threshold. According to the friction coefficient of the surface [10], the ABS will release the brake pressure more slowly if needed.
2.3.2 Traction Control System

2.3.2.1 Goals and principle

The TCS regulates the torque distributed to the wheels in order to let it remain as close as possible to the optimal slip ratio. As explained in section 2.3.1.1, this is also one of the ABS objectives.

The main difference is that where the ABS gets activated during braking to maximize the longitudinal force by reducing the brake force, the TCS acts on the brakes and the engine to reduce the torque when driving off in order to minimize the slip during acceleration.

The TCS is particularly useful on slippery surfaces as it allows the vehicle to keep its maximal traction without the driver having to control the throttle. In case of $\mu$-split, the TCS also counteracts the extra yaw angle created by the friction difference.

2.3.2.2 Technical insight

The internal combustion engine (ICE) powered vehicles are equipped with a TCS that can act both on brakes and engine to modulate the torque distribution. Three different controllers are used:

- Powertrain Traction Control (PTC)
- Symmetric Brake Traction Control (BTC)
- Asymmetric Brake Traction Control (BTC)

**Powertrain Traction Control** In order to control the wheel slip, the PTC modifies the engine torque according to the current driving situation. The engine torque demanded by the driver is unchanged in case of a stable vehicle. If some instability occurs, PTC is activated and the engine torque is computed through a non-linear PID controller. The target wheel slip and the controller gains are adjusted to the individual driving situation.

Control is permitted in general only if the driver accelerates and if the engine torque requested by the driver is larger than the controller output torque. Indeed, the TCS cannot request a higher engine torque than the driver input. The control signal is sent to the engine through the CAN network.

An antistall system is present inside the PTC in order to avoid a excessive reduction of the engine torque. If the torque reduction is not efficient enough, the brakes can also help stabilize the vehicle through the BTC.

**Symmetric Brake Traction Control** The symmetric BTC works together with the PTC. It acts as a support when the torque reduction delay from PTC is too high: waiting for the engine response, the brakes reduce themselves the torque to the wheels by braking both driven wheels at the same time.

Symmetric BTC and PTC act together on even friction surfaces to maintain stability when the driver overcomes the available friction. Making the wheels slip results in a loss of steerability and maneuverability.

**Asymmetric Brake Traction Control** During a $\mu$-split situation without any control system on, the differential will transfer most of the engine torque to the wheel on the slippery surface. The wheel located on the low friction side slips while the wheel on the high friction side could sustain a higher torque. To avoid this, the TCS brakes the wheel on low friction surface. Once the brake force is applied to this wheel, the differential will provide a raised torque on the
high friction surface wheel thus raising the motion force. The engine torque can be adjusted to
reach the maximum friction available on this wheel.

This situation is particularly critical on rear wheel driven cars. The yaw moment created
by the friction difference can destabilize the car and the driver steering may not be sufficient to
maintain a straight trajectory. The controller overshoot has to be reduced to avoid this even if
the correction is slower compared to a front wheel driven vehicle.

2.3.3 Vehicle Dynamics Control
2.3.3.1 Goals and principle
As explained in section 2.2.2.4, exceeding the friction ellipse can lead to either oversteering or
understeering. To counteract these behaviors, the ESP can brake one of the wheels that is not
saturated in order to create a yaw moment that can bring the car back to the direction given
by the driver.

2.3.3.2 Vehicle dynamics equations
To detect an unmatched trajectory with the steering angle, the ESP computes an ideal yaw rate
thanks to parameters either car-dependent or situation-dependent.

The following parameters are necessary to set out the optimal yaw rate:

- \( v_x \), longitudinal speed of the vehicle
- \( m \), vehicle mass
- \( C_{12}, C_{34} \) respectively front and rear cornering stiffness
- \( L \), distance between the two axles
- \( f \), distance between the front axle and the center of gravity
- \( b \), distance between the rear axle and the center of gravity
- \( J_z \), vehicle inertia around z-axis
- \( K_{us} = \frac{m(bC_{12} - fC_{34})}{L C_{12} C_{34}} \), understeer gradient

Through a development detailed in [12], it is possible to link the steering angle \( \delta \) to the yaw
rate \( \dot{\psi} \) through the following equation, referring to the bicycle model:

\[
C_{\psi}^\delta = \frac{v_x}{L + K_{us} \omega_0^2} \cdot \frac{1 + \frac{f m v_x}{L C_{12} C_{34}} s}{1 + \frac{2 \sigma}{\omega_0^2} s + \left( \frac{s}{\omega_0} \right)^2} 
\]

with the following parameters introduced:

- \( \sigma = \frac{m (C_{12} f^2 - C_{12} b^2) + J_z (C_{12} + C_{34})}{2 J_z m v_x^2} \)
- \( \omega_0^2 = \frac{C_{12} C_{34} L (K_{us} v_x^2 + L)}{J_z m v_x^2 + L} \)

The ESP ECU can thus compute in real time the optimal yaw rate, thanks to the sensors
and vehicle data. When a discrepancy between the current yaw rate and the model, the ESP
brakes one wheel on the unsaturated axle in order to create an extra yaw moment and reduce
the difference between the yaw rate and its target.
3 MATLAB tool development

3.1 Context

Application engineers are responsible for tuning the control system parameters in order to fit the automotive manufacturers’ requirements concerning the vehicle dynamics. An application engineer is typically responsible for an ESP function, such as ABS, TCS or value-added functions such as automatic parking brake for instance.

During on-board tests, measurement data is recorded and can be sent to software engineers in order to check the controllers behavior. Yet, this data can only be visualized one run at a time: when the same maneuver is repeated several times with different sets of parameters, it is not possible to plot them altogether.

To tackle this issue, a significant part of the master’s thesis consisted in the development of a MATLAB tool that would allow application engineers to compare several measures on the same graph.

3.2 Background

The current tool used is the UNIVIEW software that can directly read the files taken from the car after a test. It can display as many signals as wanted from a single measurement on the same graph.

During a measurement test, the application engineers can watch all the signals using this software and detect failures by checking the sensor signals. The user is free to select data among several thousands of different signals.

While convenient for a direct visualization of data acquisition, being able to plot only measurements for one test at a time impedes a clear verification of the parameters influence. As UNIVIEW is able to export all data in a .MAT file, and given my experience working with MATLAB throughout the master’s courses, it has been decided to develop a MATLAB tool that can help users plot different curves from various measures in order to compare them more easily and be more precisely aware of the consequences of a modification of a parameter.

3.3 Requirements

A list of requirements for this script has been set by the application engineers in order to give guidelines and needed features.

- Several signals from different measurements must be displayed on the same graph
- A maximum of 10 measurements is suitable
- No maximum is set on the number of signals coming from a given measurement
- As time scale is different for all measurements, the user must be able to move the graphs along the x axis to make them match at the beginning of TCS interventions for instance.
- A text file PMSD.par is created by the measurement software and contains all the parameters along with their values. They can be under the form of a number, a Boolean, a graph (with x and y values) or a double entry table. The user must be able to have access to this list of parameters and display them inside the graph window.
- Units of each signal must appear in the legend
• Colors must be different enough to allow good readability even when printed in black & white

The code of each script and function is available in appendix A.

3.4 Results

The program written fulfilled all requirements listed before. As the program uses the MATLAB plotting interface, it allows its users to directly modify the figure as they want to so they can be used in presentations. This is convenient for application engineers as they have to motivate their choice for some parameters. The export module of MATLAB is also a great asset given the great choice of formats available.

All figures used in sections 4 and 5 were generated with this program.

A guide to help application engineers to use this script has also been written. The code is commented to help further changes if needed. The complete implementation is available in appendix A.
4 PTC Analysis

4.1 Vehicle presentation

The vehicle used throughout this series of tests is a Ford Tourneo Custom powered by a 125HP 2.2 diesel engine with a 6-speed automatic gearbox weighting 2090 kg when empty. The Tourneo Custom is a 9-seater van based on the utility van Transit Custom, pictured in fig.4.1.

![Ford Tourneo Connect](image)

Figure 4.1: *Ford Tourneo Connect. Taken from [7]*

The vehicle is equipped with a measurement bench also used to flash software into the ESP ECU.

4.2 Testing protocol

The usual protocol for tuning the PTC consists in starting the vehicle on a low friction surface, such as polished ice. Both wheels would then spin when the driver floors the throttle if no control happens. The PTC must reduce the driver requested torque in order to reach the maximum slip ratio and make the vehicle move as fast as the surface friction allows it to. Furthermore, the driver feeling has also to be taken into account. Some sets of parameters can ensure a better physical response but the driver will feel less comfortable.

To replicate the behavior of polished ice, wet tiles are used in Bosch facilities.

4.3 Control system presentation

The control logic used for the PTC is a non-linear PID controller [13]. The input signal represents the speed difference between the driven and non-driven wheels. The gain values are not fixed but depend on a set of parameters. In this case, the P gain has been modified on 6 different levels to study its influence on vehicle dynamics. For confidentiality reasons, the gain value will remain hidden and the parameter sets will be described as extra low, very low, low, medium, high, very high, extra high.

The units along with the y-axis of the following graphs have been removed for confidentiality reasons.

The output of the PID controller is the target torque, measured at the cardan shaft.

This gain is a function of the gear step engaged. As the gearbox mounted on the Tourneo Custom is an automatic one, the 1st and 2nd gears can be engaged during the manoeuvre.
In a PID controller, the P gain is directly multiplied by the input signal then added to I and D components to form the output. A high P gain usually improves rise time and decreases steady-state error, while degrading stability and increasing the overshoot.

The risk of an improper tuning of this P gain is either to reduce too slowly the torque distributed to the wheels who would slip, or to apply a cut too harsh to the available torque thus maintaining the tire below its optimal slip ratio. In case of slipping, the driver loses all control over its vehicle as the friction ellipse is saturated. This behavior has to be avoided at all costs, hence the importance of P gain tuning.

During the test session, it is noticeable that the P gain does not remain at the values indicated in the parameters but gets decreased over time. This is due to an optional factor that reduces the gain when the corrector reduces too much the delivered torque.

4.4 Measurement analysis

The signals recorded during the test sessions can be split into several categories.

- Longitudinal acceleration
- Non-driven and driven wheels speed
- Torque signals
- Slip ratio (axle slip & target slip)

For all of these categories, the signals can be plotted with others taken from the same test session, or from other measurements with different parameter sets.

For the longitudinal acceleration, it is relevant to study the results of the 6 sets together. Indeed, the comparison allows to stress out which parameters give the best acceleration.

The wheel speeds will be studied first test by test in order to check if the speed of the driven wheels is reduced accurately to match the optimal slip ratio, then altogether to compare the speed of the vehicle with the different parameters.

4.5 Data reliability

In order to check if the data from the different test sessions are comparable, all the driver torque requests have been displayed together.

The torque is quite similar over the 6 sets of parameters. Some discrepancies before the manoeuvre are due to the time spend at standstill that is not exactly the same between each test. Those during the manoeuvre are a consequence of the different controller parameters and automatic gearbox shifting gears not at the same time for every measurement.
4.6 Longitudinal acceleration

The longitudinal acceleration is plotted for the 6 sets of parameters. To ensure both a correct behavior and accurate feeling for the driver, it should progress linearly over time.

![Figure 4.2: PTC manoeuvre, longitudinal acceleration](image)

It is noticeable on figure 4.2 that with the highest P gain, oscillations are happening during the whole test session and the mean acceleration is lower compared to other tests. With the lower P gain, the acceleration drops suddenly between $t = 13.5$ and $14s$ as shown in figure 4.3b. It can be assumed that the torque was reduced too slowly making the wheels slip at that time. This loss of grip could have slow down the car even though the driver is still flooring the throttle.

The overshoot at $t = 10.5s$ is counteracted for the 2 sets of parameters with the highest P gain as seen in fig.4.3a.
A similar discrepancy can be noticed for the medium P gain around $t = 13\text{s}$, while the small differences between the other curves do not show any relevant information. This highlights the importance of carrying further analyses with other signals, but also to collect the driver’s feeling after each test session.

### 4.7 Torque analysis

#### 4.7.1 Target torque

Another interesting plot to study is the one of the target torque for all the different sets of parameters. The control system is considered accurate when the target evolves slowly enough to avoid an over correction leading to overshoot and oscillations. The analysis of this figure proves that the P gain not only influences the correction itself, that is to say how fast and how accurately the controller manages to reach the target. It also has an influence over the target itself.
The goal of having a torque controller is to match the optimal slip ratio. This target slip ratio is defined according to different parameters such as speed, road friction, lateral acceleration or even ground material. Therefore, it is not directly computed by the controller. Yet, as the tire-driveline system is highly nonlinear, the controller evaluates the optimal torque at its refreshing rate. As this target torque depends on the current torque, the control has a direct influence on it.

For the test with the highest P gain, oscillations are very noticeable and do not disappear before the end of the wet tiles track. The controller is unable to give a steady target and even disables the PTC between $t = 12.6$ s and $t = 13.1$ s.

Although the red curve (very high P gain) was giving one of the best results concerning acceleration, here several oscillations are also happening. They vanish faster than the previous case and their amplitude is also smaller as seen in figure 4.5.

The lowest P gain has a very steady target compared to the other sets. Yet, the reduction above 0 between $t = 13s$ and $t = 13.5s$ show that the controller cannot reduce the torque as it should. Indeed, it requests an absence of torque during a long time meaning that wheels are still spinning too much for almost one second. A similar behavior is also noted for the green curve (low P gain). This is visible on figure 4.5.

The blue curve (high P gain) has an overshoot between $t = 13.5s$ and $t = 14s$. The consequences of this high torque request will be studied when plotting the wheel speed, but the hypothesis of it being the cause of the acceleration drop at $t = 12.3s$ on the previous curve has to be verified.

The closest curves are those whose P gain are intermediate: the medium gain one remains even steadier than the low gain one with as it does not drops below 0. The observations on the lowest P gain curve are also valid here to explain this behavior.
4.7.2 Delivered torque

The signal studied here represents the actual torque at the driven axle. It is not measured directly, but obtained by passing the engine torque through a low-pass filter representing the driveline inertia.

The figure 4.6 confirms the hypotheses made according to the target torque curves. When the P gain is too high, the torque evolves rapidly but oscillates a lot. No steady state is reached for the two highest gains.
On the other hand, as seen in fig.4.7 low P gains correct at first accurately the torque (all correctors give a similar output signal from \( t = 10s \) to \( t = 11.5s \) except for the lowest P gain) but they take a lot of time to give back some torque to let the car go. The three peaks at \( t = 12.3s, t = 12.9s \) and \( t = 13.5s \) up to negative torque show that the PTC takes too much time to let the engine provide some torque and therefore not letting the car reach its optimal slip ratio.
Figure 4.7: *PTC manoeuvre, delivered torque for low P gains*

With the lowest gain, the PTC is also too slow for reducing the initial torque when spinning is detected. No steady state is reached before the end of the track either.

The best compromise is observed for the medium gain curve. The overshoot is contained over the negative torque and the steady state is reached around $t = 15s$.

Over the last analysis, the medium gain curve seems then to be the best tuning available among the 6 sets of parameters.

### 4.8 Wheel speed

The Ford Tourneo Custom used during those tests is a front wheel driven van. Wheel speed sensors are mounted on the 4 wheels, thus the slip ratio can be computed considering the speed of the nondriven wheels as the speed of the vehicle. The recorded signals contain the speed of each wheel, the target slip ratio and the actual slip ratio.

To ensure a better readability, only two parameters set are plotted together on each figure. The supposedly most efficient parameter set, according to previous analysis, is always one of them. The rear left wheel speed also appears on figures to give an indication on the actual vehicle speed.

The vehicle speed can differ from the undriven wheel speed. The method used here is therefore only valid for this manoeuver.
4.8.1 Highest P gain

Figure 4.8: *Wheel speed for highest P gain*

Compared to the reference, the speed of the vehicle is reduced over the full run as shown in fig.4.8. An oscillatory behavior of the driven wheel is very noticeable: the correction puts the driven wheel speed at the speed of the vehicle (around $t = 14.5s$ for instance) while on the reference, their speed remain higher to keep a nonzero slip ratio. The brutal changes in driven wheel speed can destabilize the driver and can lead him to modify his driving in an inaccurate way. The PTC intervention is also disabled between $t = 13.3s$ and $t = 13.8s$ whereas it should be activated throughout the whole run to ensure stability.
4.8.2 Lowest P gain

As seen in 4.9, compared to the former controller, the correction only have consequences at $t = 15s$. Before this threshold, the corrector is too slow to act on the engine torque thus front wheels are slipping. No big difference between the reference speed and the actual speed happen before $t = 14s$. The reference starts then gaining grip with a steady difference between undriven and driven wheel speed, while with the low P controller, the speed is put down to the level of the undriven wheels before letting enough torque come closer to the slip ratio.

Here the vehicle is more stable than previously after $t = 15s$. Before this, the driven wheels slip more than on the previous controller: the consequences are similar even though the reason is the opposite.
4.8.3 Mid P gains

The three curves in fig.4.10 are close enough to be studied altogether. Unsurprisingly, the curve with low gain has a similar behavior to the lowest gain. Same observation for the two highest gains. The vehicle speed is slower for those two curves compared to the others.

It is interesting to note that the vehicle speed is slower at first with high gain compared to medium gain due to the driven wheel speed being slow down up to the undriven wheel speed at \( t = 12.2 \) s. Yet, it becomes steady afterwards and both controllers give similar results at the end.

4.9 Slip ratio

To check if the controller P gain has an influence over the optimal slip ratio, the behavior of the target slip for different sets of parameters can be studied. The controllers with lowest, highest and most efficient P gain are studied here as they can show the biggest discrepancies.

Albeit the current axle slip is very different due to the reasons explained previously, it is important to note that the slip target calculation is similar. The optimal slip ratio is negative at the beginning as the vehicle is standing still and this low value allows the driver to give a lot of torque when driving off.
Oscillations are present on the highest P curve after the initial drop in axle slip, starting at around $t = 13.5s$.

On the lowest P gain curve, although the target slip is generated without lag (compared to the highest P gain), the controller does not manage to reduce the axle slip fast enough (not before $t = 13.5s$ where a steep drop in axle slip occurs). The axle slip then overshoots and does not match the optimal value until the end of the measurement.

With the best set of parameters, the overshoot is contained between $t = 11.2s$ and $t = 12.6s$.

These observations confirm those previously made. As a conclusion, although the P gain has no influence over the calculation of the optimal axle slip, its value has a huge influence over the target torque. This target has to be followed by the engine, then reducing the performance as the maximum torque available (and requested by the driver) is not reached.

The parameter tuned does not have a direct influence over the torque reduction nor the target slip ratio. Yet, the target torque generated by the controller directly depends on the P gain, thus defining the amount of torque reduction.

Reducing the engine performance can thus improve the vehicle dynamics in the case of a launch on slippery ground. This reduction has to be steady with as few oscillations and overshoot as possible to be efficient.

### 4.10 Conclusion

All of the previous analyses have shown that a degradation in performance can have a big influence over the vehicle dynamics. Yet, the focus has only been set on qualitative analysis: as a conclusion, some quantitative results can be drawn from the measurements.

The vehicle speed is a good performance indicator: MATLAB can perform a linear regression over the recorded data, thus giving information over the controller allowing the vehicle to gain the most speed. This regression has been done using MATLAB operator `\`, where A is the time
data matrix and B the reference speed matrix [15].

As seen in figure 4.12, the controller with the medium gain give the best results. The hypotheses resulting from the qualitative part are therefore confirmed by the numbers. Yet the driver feeling has also to be taken into account. According to the requirements, driving feel can be prioritized over performances. Here, the driver felt better with the medium gain setting, correlating with the previous results.
5 BTC Analysis

In the upcoming part, the asymmetrical BTC will be written BTC. The full BTC also contains a symmetric controller used to assist the PTC if needed by braking both wheels at the same time. The asymmetric BTC only brakes one wheel at a time to transfer torque or to correct yaw. The VDC activates the BTC in case of over- or understeering.

5.1 Testing protocol

The protocol used for testing the asymmetrical BTC consists in a 10% slope $\mu$-slip road. The $\mu$-split is achieved by running over some wet tiles on the right side of a road which both right wheels rely on. The vehicle is stopped on the surface then launched with full throttle.

Without any BTC intervention, the open differential transfers the torque over the low $\mu$ driven wheel. This wheel would spin while the wheel on the high $\mu$ would not move, leading to the vehicle not being able to accelerate and even going down the slope.

The BTC brakes the low $\mu$ wheel in order to transfer the torque to the high $\mu$ side. The vehicle is then able to take advantage from the grip available and climb up the slope.

The driver has to correct the trajectory of the vehicle due to yawing when torque is distributed on the high $\mu$ side but these interventions must remain moderate.

5.2 Control system presentation

The control logic used for the BTC is a non-linear PI controller [14]. Its activation depends on the wheel speed difference between the 2 sides of the vehicle. The input signal represents the speed difference between the front left wheel and the front right wheel and is used for front wheel drive vehicles such as the Tourneo Custom used for those tests.

When the speed difference between the driven wheels exceeds a given threshold, the BTC is activated. The I gain is a variable and will evolve throughout the different situations encountered by the vehicle. The P gain remains steady. Its input is a function based on the wheel speed difference between right and left side.

The I gain calculation depends on the driving situation.

Throughout the test session, four different combinations of I and P gains have been tested. Their values are hidden for confidentiality reasons and the parameter sets are called very low, low, med and high.

5.3 Measurement analysis

The signals recorded during the test sessions can be split into several categories.

- Longitudinal acceleration
- Non-driven and driven wheels speed
- Brake pressure
- Slip ratio (axle slip & target slip)

Contrary to the PTC analysis, the engine torque will not give any relevant information as the BTC only activates the brakes. The PTC will intervene in order to reduce the total amount of torque to the wheels, yet the torque transfer between left and right is only done through the brakes, hence the analysis of the brake pressure.
5.4 Data reliability

In a related way to the PTC study protocol, the driver torque requests from the 4 sets have been plotted together to ensure that the manoeuvre has been repeated.

![BTC manoeuvre, driver torque request](image)

Compared to section 4.5, the curves are way less similar with each other in fig.5.1. Yet, the driver presses the throttle at its maximum each time. This is due to the PTC trying to compensate more when the BTC gain is not high enough. Indeed, PTC has to be active in order to let BTC on.

During the BTC low measurement, the throttle have been activated at $t = 10.5s$ as shown in the fig. 5.1 which shows a throttle position signal.

Despite this located discrepancy, the curves are close enough to allow a compared analysis. The differences, quite noticeable after $t = 15s$, are due to the different PTC activation levels and the impossibility to replicate a perfectly similar manoeuer at each run.
5.5 Longitudinal acceleration

![BTC manoeuver, longitudinal acceleration](image)

The longitudinal acceleration is harder to analyze than in PTC part. The throttle is pressed at $t = 12.9s$ as seen in fig.5.2, yet the curves are not stabilized at zero before the acceleration rises. The low gain corrector behavior is due to the throttle being pressed between $t = 10s$ and $t = 11.4s$. 

29
Figure 5.3: *BTC manoeuver, longitudinal acceleration*

When not taking into account the curves before pressing the throttle as in 5.3, the very low setting is noticeably less efficient than the other curves. The medium and high settings reach the highest acceleration in the elapsed time.

An important part of the BTC test is the non symmetric nature of the test, as right wheels are set on a slippery surface while the left rely on dry asphalt. To study this phenomenon more precisely, the individual wheel speeds can be plotted together.
5.6 Wheel speed

5.6.1 Very low gain setting

The medium gain setting lets the right wheel spin when the throttle is pushed as shown in fig.5.4. The speed peaks at $t = 10.5\text{s}$ before the BTC brakes it, thus reducing its speed while transferring torque on the left wheel. This left wheel speed grows up quite linearly, thus never exceeding its friction available. The right wheel brake is released at $t = 12\text{s}$ but is activated again at $t = 12.5\text{s}$ to counteract the slip.

On the other hand, the low gain setting never manages to transfer torque on the left wheel whose speed remains close to zero up to $t = 16\text{s}$. The right wheel gains velocity peaking at $t = 11\text{s}$ then the BTC brakes it, but not enough to recover grip. This shows that in a $\mu$-split situation, the intervention of the ESP is more than necessary to avoid a complete loss of control from the driver that can lead to the vehicle going down the slope.
5.6.2 Low gain setting

Figure 5.5: BTC manoeuver, wheel speeds from medium and low gain settings

In fig.5.5, the evolution is noticeable compared to the very low gain setting. The left wheel manages to gain speed at a steady pace, yet slower than the medium setting. The whole behavior is acceptable but the comparison with the medium gain shows that a better compromise exists.
5.6.3 High gain setting

The high gain setting leads to too much braking on the right wheel. Therefore, too much torque is transferred into the left wheel which starts slipping as well even though it is relying on dry asphalt. This is noticeable in fig. 5.6 at $t = 14.7s$, then from $t = 16s$ to $t = 17.2s$. The consequence is that the driver cannot steer anymore as both tires are friction saturated, which is an unwanted behavior. Although the vehicle climbs the slope at a similar pace, this controller is therefore not suitable compared to the medium gain.

Along with the wheel speed, the brake pressure induced by the BTC can also be plotted and studied in order to see the influence of the gain over it.

Figure 5.6: BTC manoeuver, wheel speeds from medium and high gain settings
5.7 Brake pressure

The signal used in fig.5.7 represents the brake fluid pressure in the right wheel. As the brake pedal is not pressed during the acceleration phase (after $t = 11s$), only the BTC is taken into account. The very low gain setting shows a much weaker brake pressure than the others. This correlates the result given by the wheel speed. The high gain setting is surprisingly quite similar to the medium one, yet sharper and with some oscillations. The low gain setting has an average behavior between the others.

The goal is then not only to find a parameter that ensures a suitable braking force, but also to build the force smoothly enough to transfer the torque progressively in order not to make the wheel on high $\mu$ surface slip. Hence the importance of playing not only with the P gain but also with the I in order to smoothen the corrector response.
5.8 Axle and target slip

As seen in fig.5.8, with the very low gain setting, the slip ratio remains at a high value which means that the brake force is too low to slow down efficiently the wheel on low $\mu$ surface. The slip target is very similar between high and medium gain settings, the only difference being the actual right wheel slip that is reduced too much in the high gain case. As a consequence, the slip goes way beyond the target at $t = 13.6s$ while the overshoot is contained in the medium gain setting.

The low gain setting is the only one to bring the wheel down to a zero slip ratio, meaning that the wheel does not slip at all. The overshoot is the biggest and the controller does not manage to reduce the oscillations around the reference efficiently.

5.9 Conclusion

The BTC pairs the PTC in allowing a torque redistribution between the two sides of the vehicle. The gain parameters play an important role in letting the vehicle steer throughout a $\mu$-split situation: a weak brake force will not allow enough torque on the wheel where friction is available, while excessive gains will make this wheel exceed its friction limit and thus slip even though it relies on a high $\mu$ material.

This part also highlights the importance of having real drivers executing tests before releasing a software for new models. The driver feeling is primordial not only as a buying motive, but also for security reasons as they must always be able to handle their vehicle, even in worst case scenario such as a $\mu$-split slope.

As a numerical reference, in order to compare all settings, the linear coefficient of the vehicle speed throughout the run has been computed. It confirms that the medium gain setting is the
most efficient, along with the driver feeling.
6 Conclusion

Throughout this whole thesis, the main features of the ESP have been explained, and the two components of the TCS were the focus of a parameter analysis.

Although it can seem contradictory, reducing the motor torque and braking a wheel can thus improve the vehicle dynamics when the ground does not allow the tires to create a high friction force. This is also a very important issue in motorsports, as powerful vehicles can receive too much torque at a race start thus slipping and losing grip, meaning precious time at the end of the lap.

The control systems used nowadays in the ESP can seem pretty basic at first, as it is only a P- or PI-controller in PTC and BTC respectively for instance. Yet, they have to be set with a great precision to adhere to the automotive manufacturer requirements. Furthermore, many supplementary gains can add up if the ESP detects a critical situation (such as sand or snow where the maximum friction is reached for the highest slip ratio possible due to the piling of the material).

This thesis also sets the highlight on the importance of having skilled drivers performing in-vehicle tests before releasing a new ESP software. Simulation can give hints on the range of suitable values, but the driving feeling and expected reactions can only be performed by a real driver inside the car.

7 Future work

The program developed during this thesis requires MATLAB to run. This can be inconvenient in case of license issues. Therefore, rewriting it in another language would turn it into a more universal tool. The final goal of this tool would be to allow applicators to compare their results more easily, thus making the parameter setting easier. As Python has evolved a lot these last years and is even catching up with MATLAB in the scientific field, it could be a good language choice to rewrite the program.
References


[12] *Vehicle dynamics in the asymmetry plane*, SD2225 Vehicle Dynamics course, KTH


A Appendix

A.1 MATLAB implementation

Once launched, the script opens a figure and starts by running the function \texttt{premessure}. This function invites the user to select a .MAT file that has been generated from UNIVIEW. The .MAT file created contains all data selected on the main UNIVIEW window, along with their names, a character string holding all of the units, and a time scale.

Using the \texttt{load} command creates a structure array where the name of each signal is the name of a field. To display the units in the final graph, a regular expression is run through the string of units in order to split it into a cell array where each cell contains a unit. After having selected a .MAT file, the user is asked to select the signals to plot. As the signals are not ordered in UNIVIEW, the command \texttt{orderfields} is used here to make the selection clearer. Furthermore, the user can type the first letters of the signal to plot, then MATLAB will select the closest value to his input.

If the user has already loaded a file with similar signals, the previous selection will remain on the signal selection box thanks to the parameter InitialValue of \texttt{listdlg} (function used to allow the user to select fields).

A consequence of using \texttt{orderfields} is that the units are not in the correct order anymore: the cell array containing them is still following the original order given by UNIVIEW during the export. Thus, as \texttt{orderfields} can also output a vector of the permutations proceeded, this vector is used to switch units back to their right position.

The software used to save the test measurements (MM6X) starts its time counter when the user starts to record data. Yet, only the last seconds (parameter entered by the user, typically 15 or 20) are then saved as a file. As a consequence, all measurements have a different time axis that can make them hard to compare. To put all measures in a comparable time set, the first value of the time vector is subtracted from the others in order to make all measurements start at $t = 0$. As described in the requirements, it is also possible to move each measure along the x axis as explained further.

The selected fields are then extracted and their data is merged into a matrix used for plotting. The legend contains the name of the signal, its source file and its unit. Once all data is computed and ready to be displayed, a plot is created and a function responsible for creating a slider is called. This slider allows the displacement along the x-axis for all signals coming from the same file. For each measurement, a new slider appears and allows the user to match the signals as described in the requirements.

Along with the sliders, three buttons allow the user to add a measurement, a PMSD parameter or a text string.
To add a measurement, the same function is called as before. Its inputs are the number of measurements already present on the graph and the previously plotted signals to preselect them.

The main problem was to keep the value of the number of measurements when the button is clicked several times as a GUI function cannot export or call variables from the workspace. To deal with these limitations, a persistent variable has been used. A persistent variable holds its value between each function call.

The Add text button calls the \texttt{gtext} inbuilt function. It allows the user to click on the graph and display a text string where he clicked.

For the PMSD file, the data is extracted from a \texttt{.txt} file where all parameters are defined. The name and value of each parameter are preceded by keywords. Therefore a regular expression is used to split the file into a cell array.

Two splits happen consecutively. The first one only separates the parameters from each other. The result of this first split is a cell array where each cell holds a string with all data (name of the parameter, description, value... ) mixed altogether.

The second split then looks for the keywords separating the different data types. The result is an imbricated cell array where each cell is another cell array, in which each cell holds the different data of a parameter.

A structure array is then created with 4 different fields: \texttt{name}, \texttt{value}, \texttt{sty} and \texttt{stx}. The \texttt{stx} field contains the x-axis values when the parameter is a curve or the column header if it is a table. The \texttt{sty} field is used for the table row header. The fields are created from the data of the previous cell array.

The user has a list box to select the parameter he wants to put on display. Before that, a text prompt allow him to search for a parameter which only includes the word entered as shown in figure A.2. At first, the command \texttt{strfind} was used to find the field titles containing the
user text input. Yet, `strfind` is case sensitive and would not return fields if the capital letters did not match.

To avoid this issue, the command `regexpi` is used. It is case insensitive and can find the user input inside the cell string array of field titles when the input is placed between two strings `\w*`. This command represents a word in general, therefore the regular expression will find the user input inside a field name even if it is located in the middle of that name.

![Figure A.2: Search box for parameters](image)

![Figure A.3: Search results](image)

The parameters have to be checked first to determine their nature (value, Boolean, curve or matrix). To proceed, the number of fields inside the parameter is matched with a switch. Single values or Boolean have 5 fields, while curves can have 7, 9 or more fields. As a matrix consists in minimum 11 fields, a further treatment must be done to separate curves and matrices.

At the end of each parameter name, a number (for a curve) or two (for a matrix) indicates
If the parameter is a Boolean or a numerical value, its name and its value are displayed on the main graph. If it is a curve, a graphical representation with axes is displayed inside the main graph. For tables, the command \texttt{uitable} is used and displays the table.

Some parameters need to be linked to a curve in order to insist on their influence on a highlighted signal. Several options were considered, such as drawing a line or an arrow to the curve wanted or modifying the parameter text color. The chosen feature is to frame the parameter name with a selected curve color. This choice was made to avoid the case where two curves are close to each other and therefore a graphic link (arrow or line) would be confusing. It also has the advantage of not being too intrusive nor affecting graph readability.

![Example of a plot with all types of parameters included](image)

The default colors of MATLAB under the Lines color map are limited to seven. Even when using the hold all parameter, it is impossible to avoid redundancy of colors when many signals are selected.

To fix this, a function called \texttt{linspecer} is used. It has been found on the Mathworks website [16] and is distributed with a license that allows its use and redistribution. The goal of this function is to set colors that are easily distinguishable from each other and readable on a white background. Although it can be still hard to distinguish the signals when around 20 of them are plotted on the same graph, the readability is improved a lot compared to the Lines default color map.

The \texttt{linspecer} function only selects colors dark enough to remain readable if the graph has to be printed in black & white.

**A.2 mesuresfctdivsel.m**

This function is the base of the program. It calls other functions when needed.
%% Main script of the signal viewer

%% Create the figure and add the first measure
hold all
grid on
[matforplot1,temps1,selec,nomsselec,plott]=premmesure(0,1);
xlabel('Time[s]')
autre=addmatfile;

%% Add signals
% adds up all signal names in order to create the graph legend from them
% afterwards
i=1;
while autre==1;
    nomsvieux=nomsselec;
    [matforplot1,temps1,selec,nomsselec,plot1]=premmesure(i,selec);
    i=i+1;
    nomsselec=[nomsvieux;nomsselec];
    autre=addmatfile;
end

%% Update colors
hline=findobj(gca,'type','line');
if length(hline)>7
    coul=linspecer(length(hline),'qualitative'); % linspecer chooses colors easily distinguishable and printable in b&w
    for j=1:length(hline)
        set(hline(j),'Color',coul(j,:));
    end
else
    coul=lines(length(hline)); % sinon on utilise la colormap lines sur l'ensemble des mesures
    for j=1:length(hline)
        set(hline(j),'Color',coul(j,:));
    end
end

%% Add legend and GUI
leg1=legend(nomsselec);
set(leg1,'Interpreter','none')
set(gcf,'Toolbar','figure');
buttonnom(coul);
buttonpmsd(coul);
buttonmeas(i,selec);
function [matforplot,temps,selec,nomsselec,ploplot]=premmesure(comp,SelectedMeas)
% premmesure lets the user select a .MAT file from UNIVIEW and extracts selected...
data to plot it afterwards in the main window.
% input: comp = number of measures used for cursor creation
% SelectedMeas = data previously selected to help the user select the same fields on the following measure
% output: matforplot = matrix holding all the data of the selected fields used for yaxis
% temps = time vector used for xaxis
% selec = vector of indices of selected fields
% nomsselec = cell array of signal names used for legend
% ploplot = plot (temps,matforplot) externally used to add GUI elements

[nomfil,cheminfil]=uigetfile('*.mat','Select the .MAT file from...
UNIVIEW','Multiselect','off');
mesure1=load([cheminfil nomfil]);

%% Unit display
%the q_* fields created by UNIVIEW are removed from the measure and the remaining fields are ordered. The units are taken from q_UNIT and reordered according to the same permutations.
mesure2=rmfield(mesure1,{’q_KEY’,’q_SCMAX’,’q_SCMIN’,’q_T0’,’q_TC’,’q_UNIT’});
[mesure2,permutfields]=orderfields(mesure2);
ListUnitsDer=regexp(mesure1.q_UNIT,’\s’,’split’);
ListUnits={};
if length(permutfields)==length(ListUnitsDer)
    for j=1:length(permutfields)
        NewOrder=permutfields(j);
        ListUnits{j}=ListUnitsDer{1,NewOrder};
    end
end
% the timeset is modified to begin at t=0
temps=mesure1.q_T0-mesure1.q_T0(1);

%% GUI field selection
noms1=fieldnames(mesure2);
if length(noms1)<max(SelectedMeas)
    selec=listdlg(’ListString’,fieldnames(mesure2));
else
    selec=listdlg(’ListString’,fieldnames(mesure2),’InitialValue’,SelectedMeas);
end
nomsselec=noms1(selec);
matforplot=zeros(length(selec),length(temps));

%% Data matrix for plotting
for i=1:length(selec)
    recup=nomsselec(i);
    pourplotres=mesure1.(char(recup));
    if length(pourplotres)==length(temps)
        matforplot(i,:)=pourplotres;
    else
    end
end

%% Legend creation
nombres=cellstr(repmat([blanks(1),nomfil,blanks(1)],length(selec),1));
CrochOuvr=repmat('][',length(selec),1);
CrochFerm=repmat('[',length(selec),1);

if ~isempty(ListUnits)
    nomsselec=cellstr([char(nomsselec) char(nommes) char(CrochOuvr) ...
                      char(ListUnits(selec)) char(CrochFerm)]); % nom de la mesure, espace, nom ...
                      du fichier d'ou elle est extraite
else
    nomsselec=cellstr([char(nomsselec) char(nommes)]); % nom de la mesure, ...
                      espace, nom du fichier d'ou elle est extraite
end

ploplot=plot(temps,matforplot);
myslider(temps,comp,ploplot);
end
A.4  addmatfile.m

function anotherone = addmatfile
%addmatfile outputs a boolean used to add other measures or stop
%input : user choice
%output : boolean used in main script
resquest = questdlg('Do you want to add another file?');
anotherone = strcmp(resquest,'Yes');
end
A.5 ParsePMSD.m

function [ StructPar, PMSDNames ] = ParsePMSD( pmsdid )
%ParsePMSD turns the pmsd.par.txt file into displayable data
%Given the pmsd file, it is read as a single string before being split
%into a cell array. Each cell contains the fields of the parameter that are
%fed into a structure array. The names are copied into another cell string
%array.
%
%% File acquisition and cell array generation
BigString = fscanf(pmsdid, '%c');
ParamCells = regexp(BigString, 'FESTWERT|KENNLINIE|KENNFELD', 'split');
ParamCells = regexp(ParamCells, ... 'EINHEIT_W|WERT|ST/X|EINHEIT_X|ST/Y|EINHEIT_Y|TEXT|KENNLINIE|\sEND|LANGNAME', 'split');

%% Struct Array & name list creation
for i = 2:length(ParamCells)
  StructPar(i-1).name = ParamCells{1,i}{1,1};
  PMSDNames{i-1} = ParamCells{1,i}{1,1};
  longparam = length(ParamCells{1,i});
  %According to the number of lines in the parameter data, the switch
  %gets the parameter kind.
  switch longparam
    case 5 %FESTWERT: text, value, boolean
      StructPar(i-1).value = ParamCells{1,i}{1,4};
    case 7 %KENNLINIE: 2D curve
      StructPar(i-1).value = ParamCells{1,i}{1,6};
      StructPar(i-1).stx = ParamCells{1,i}{1,5};
    case 9 %KENNLINIE: 2D curve with more points
      axex = horzcat(ParamCells{1,i}{1,5}, ParamCells{1,i}{1,6});
      StructPar(i-1).stx = axex;
      axey = horzcat(ParamCells{1,i}{1,7}, ParamCells{1,i}{1,8});
      StructPar(i-1).value = axey;
    otherwise
      % KENNFELD and KENNLINIE with more than 12 points have the same
      % number of fields. Each parameter name ends with its
      % dimensions, thus a regular expression detects if it is higher
      % than 12 (KENNLINIE) or not (KENNFELD)
      numberelem = regexp(PMSDNames{i-1}, '\s\d\d', 'match');
      if ~isempty(numberelem) %KENNLINIE
        axex = horzcat(ParamCells{1,i}{1,5}, ParamCells{1,i}{1,6}, ParamCells{1,i}{1,7});
        StructPar(i-1).stx = axex;
        axey = horzcat(ParamCells{1,i}{1,8}, ParamCells{1,i}{1,9}, ParamCells{1,i}{1,10});
        StructPar(i-1).value = axey;
      else
        valwert = cell(1);
        sty = cell(1);
        %KENNFELD: 2D matrix
        StructPar(i-1).stx = ParamCells{1,i}{1,6};
        k = 1;
        for j = 7:2:longparam %the ST/Y data is registered line by line
          sty{k} = ParamCells{1,i}{1,j};
          k = k + 1;
        end
        l = 1;
        for m = 8:2:longparam %same thing for values
valwert{l}=str2num(ParamCells{1,i}{1,m});
l=l+1;
end
StructPar(i-1).sty={[sty{;}]; %ST/Y data is saved as a vector
StructPar(i-1).value=vertcat(valwert{:}); %the values are saved as a matrix
end
fclose(pmsdid);
A.6  buttonnom.m

```matlab
function buttonnom(coul)
    % creates a button to add text on the main graph
    uicontrol('Style', 'pushbutton', 'String', 'Add text', ...
        'Position', [20 60 100 20],...
        'Callback', {@ajoutenom, coul});
end

function ajoutenom(hObject, event, coul)
    % User enters the text to display then is asked to select a curve to match
    % the text with. The text is then framed with this curve's color.
    nomcourbe=inputdlg('Curve name');
    tt=gtext(nomcourbe);
    [~,~,~,textleg]=legend;
    numbcolor=listdlg('ListString',textleg,'SelectionMode','single','Name','Curve ...
        linked to the text:');
    set(tt,'Interpreter','none');
    set(tt,'EdgeColor',coul(length(coul)-numbcolor+1,:));
end
```
Appendix

A.7 buttonpmsd.m

function buttonpmsd(coul)

uicontrol('Style', 'pushbutton', 'String', 'Add param. value', ...'
   'Position', [20 80 100 20], ...
   'Callback', {@ajoutepmsd,coul});
end

function ajoutepmsd(hObject,event,coul)
% ajoutepmsd allows the user to display the name of a parameter along
% with its values on the main graph.

%% Select & load txt file
[pmsdfile,pmsdpath]=uigetfile('*.txt','Select the pmsd.par.txt ...
   file','Multiselect','off');
pmsdid=fopen([pmsdpath pmsdfile]);
[structpar,pmsdlist]=ParsePMSD(pmsdid);

%% Parameter choice, mouse pointer for display
%User has access to a search box in order to reduce the parameter list to
%those containing the word entered
recherche=inputdlg('Filter parameters whose name contains: (leave empty for ...
   complete list)');
recherche=strcat('w*',recherche{1},'w*');
nomscherche=regexpi(pmsdlist,recherche); %regexpi is case insensitive compared ...
to strfind
resrecherche=cell(1);
correspindices=[];
j=1;
for i=1:length(nomscherche)
    if ~isempty(nomscherche{i})
        resrecherche{j}=pmsdlist{i}; %a new string cell array is created with ...
        the search results
        correspindices(j)=i; %it is necessary to save the original indices for ...
        fetching the parameter wanted
        j=j+1;
    end
end

%User selects its parameter, the corresponding data are extracted from the
%struct
selec=listdlg('ListString',resrecherche,'SelectionMode','single');
choix=correspindices(selec);

stg1=char(structpar(choix).name);
stg2=char(structpar(choix).value);
stg3=char(structpar(choix).stx);

%% Parameter display on the main graph
%If the parameter is a KENNLINIE, a small graph appears representing its value ...
%curve inside the main graph.
%ginput gets the mouse position, converted from graph units to relative
%position.
%To link a parameter to a curve, the user is asked to select one curve
%whose color will frame the title of the parameter.

if and(~strcmp(stg3,''),isempty(structpar(choix).sty)); %for KENNLINIE (1st ...
   condition excludes TESTWERT, 2nd excludes KENNFELD)
   [dispx,dispy]=ginput(1);
dispmsd=text(dispx,dispy-0.01,stg1);
[-,-,-,textleg]=legend;
numbcolor=listdlg('ListString',textleg,'SelectionMode','single','Name','Curve ... linked to the parameter:');
set(dispmsd,'Interpreter','none');
set(dispmsd,'EdgeColor',coul(length(coul)-numbcolor+1,:)); %color frame
xborder=xlim;
yborder=ylim;
xprop=(dispx-xborder(1))/(xborder(2)-xborder(1));
yprop=(dispy-yborder(1))/(yborder(2)-yborder(1)); % the mouse position ...
is converted from axis relative value to normalized
KennlinieAxes=axes('position',[xprop yprop 0.07 0.04]);
kennlinplot=plot(str2num(stg3),str2num(stg2));
set(kennlinplot,'Color','red','LineWidth',2);
set(KennlinieAxes,'color','none');
grid on
else
  if isempty(structpar(choix).stx) %for TESTWERT
    displaypmsd=gtext([stg1 stg2]); %gtext is used to display name and value
    [-,-,-,textleg]=legend;
numbcolor=listdlg('ListString',textleg,'SelectionMode','single','Name','Curve ... linked to the parameter:');
    set(displaypmsd,'Interpreter','none');
    set(displaypmsd,'EdgeColor',coul(length(coul)-numbcolor+1,:));
  else %for KENNFELD
    [dispx,dispy]=ginput(1);
    xborder=xlim;
yborder=ylim;
xprop=(dispx-xborder(1))/(xborder(2)-xborder(1));
yprop=(dispy-yborder(1))/(yborder(2)-yborder(1));
   uitable displays an interactive table inside the main graph
    tablepar=uitable('Data',structpar(choix).value,'ColumnName',str2num(structpar(choix).stx),
    dispmsd=text('String',stg1,'Units','normalized','Position',[xprop,yprop]);
    [-,-,-,textleg]=legend;
numbcolor=listdlg('ListString',textleg,'SelectionMode','single','Name','Curve ... linked to the parameter:');
    set(dispmsd,'Interpreter','none');
    set(dispmsd,'EdgeColor',coul(length(coul)-numbcolor+1,:));
  end
end
end
A.8 buttonmeas.m

function buttonmeas(i,selec)
% creates a button to add text on the main graph
uicontrol('Style', 'pushbutton', 'String', 'Add measurement',...
    'Position', [20 100 100 20],...
    'Callback', {@addmes,i,selec});
end

function addmes(hObj,event,i,selec)
persistent compteuradd
if isempty(compteuradd)
    compteuradd=i;
end
autre=1;
nomsselec=[];
while autre==1;
    nomsvieux=nomsselec;
    [matforplot1,temps1,selec,nomsselec,plot1]=premmesure(compteuradd,selec);
    compteuradd=compteuradd+1;
    nomsselec=[nomsvieux;nomsselec];
    autre=addmatfile;
end
[-,-,-,test]=legend;
nomsselec2=[test'; nomsselec];
leg2=legend(nomsselec2);

%% Redefine colors
hline2=findobj(gca,'type','line');
coul=linspecer(length(hline2),'qualitative'); % linspecer chooses colors easily distinguishable and printable in b&w
for j=1:length(hline2)
    set(hline2(j),'Color',coul(j,:));
end
set(leg2,'Interpreter','none')
set(gcf,'Toolbar','figure');
end
A.9  makeplot.m

```matlab
function makeplot(hObject,~,x,hplot)
%makeplot links the cursor value with the graph displacement along the x axis
n=get(hObject,'Value');
set(hplot,'xdata',x+n);
drawnow;
```
Appendix

A.10 myslider.m

function myslider(x,i,plott)
%myslider creates a cursor used to move plots over the xaxis
%inputs: x = time range of the plot
% i = number of measures plotted
% plott = current plot

hold on
grid on
h=uicontrol('style','slider','Min',-max(x),'Max',max(x),'units','pixel','position',[20+100*i ... 40 100 20],'SliderStep',[1e-4 1/max(x)]); %Affiche le curseur selon le ... numero de la mesure
uicontrol('Style','text','Position',[20+100*i 20 100 20],'String',strcat('Move ... measurem',num2str(i+1))); %Texte qui identifie le curseur
addlistener(h,'ContinuousValueChange',@(hObject,event) ... makeplot(hObject,event,x,plott)); %activation du mouvement du graphe
end