



Christina Schwarz BSc

**Tool for schematic design and automated
parameterization for real-time generic drivetrain models**

MASTER'S THESIS

to achieve the university degree of

Diplom-Ingenieurin

Master's degree programme: Telematics

submitted to

Graz University of Technology

Supervisor

Univ.-Doz. Dipl.-Ing. Dr.techn. Daniel Watzenig

Institute of Electrical Measurement and Measurement Signal Processing

Head of the Institute: Univ.-Doz. Dipl.-Ing. Dr.techn. Georg Brasseur

Graz, March 2015

AFFIDAVIT

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis dissertation.

Date

Signature

Kurzfassung

Echtzeitfähige Antriebsstrang-Modellierungsansätze haben ein großes Potenzial zur Verringerung von Entwicklungskosten in der Automobilindustrie. Obwohl viele echtzeitfähige Antriebsstrang-Modelle existieren, sind diese oft spezifisch für eine Getriebetopologie realisiert. Im Rahmen dieser Masterarbeit wurde ein Tool für den Entwurf und eine anschließende automatische Parametrierung von Antriebsstrang-Modellen entwickelt. Die Basis für die Parametrierung stellt eine generische Methode dar, die auf alle Arten von Getrieben mit diskreten Gängen anwendbar ist. Dies ermöglicht tool-unterstützte Modellierung auch für unerfahrene Entwickler im Bereich des Maschinenbaus und der Regelungstechnik und reduziert daher den Entwicklungs- und Testaufwand. Die generische Methode wird für verschiedene exemplarische Antriebsstrang-Topologien, unter Verwendung des entwickelten Tools für die automatische Parametrierung, gezeigt. Im Anschluss wird ein so erstelltes Modell für ein 7-Gang-Automatikgetriebe mit Fahrzeug-Messdaten aus einem Fahrzyklus auf einer Teststrecke validiert.

Schlüsselwörter: Antriebsstrang-Design, automatisierte Parameterierung, Fahrzeuggetriebe, generische Modellierung, Reibungsmodellierung, Echtzeit-Simulation

Abstract

Real-time drivetrain modeling approaches have a significant potential for development cost reduction in the automotive industry. Even though real-time drivetrain models are available, these solutions are specific to single transmission topologies. In this Master's thesis a tool for drivetrain design and automated parameterization, which is applicable to all types of transmission topologies containing explicit gears is presented. This enables tool-guided modeling by non-experts in the fields of mechanic engineering and control theory leading to reduced development and testing efforts. The approach is demonstrated for different exemplary automatic transmissions using the environment for automated parameterization. Finally, an automated parameterization for a 7-gear automatic transmission is validated via vehicle measurement data of a real-life driving cycle.

Keywords: drivetrain design, automated parameterization, automotive transmission, generic modeling, friction modeling, real-time simulation

Danksagung

Ich möchte mich bei Univ.-Doz. Dipl.-Ing. Dr.techn. Daniel Watzenig für die Möglichkeit der Durchführung dieser Diplomarbeit am Institut für Elektrische Meßtechnik und Meßsignalverarbeitung an der Technischen Universität Graz in Zusammenarbeit mit dem Virtual Vehicle Research Center bedanken. Ein großes Dankeschön gilt auch Dipl.-Ing. Markus Bachinger, der mich während der gesamten Zeit der Masterarbeit betreut und unterstützt hat. Weiterer Dank gilt den Mitarbeitern des Virtual Vehicle Research Centers, im Besonderen Dr. techn. Michael Stolz und Dipl.-Ing. Wolfgang Ebner, die mir mit ihrer fachlichen Kompetenz zur Seite gestanden sind.

Bedanken möchte ich mich auch bei meiner Familie, vor allem bei meinen Eltern, ohne deren mentale und finanzielle Unterstützung ich nicht soweit gekommen wäre.

Abschließend möchte ich mich bei meinen Freunden bedanken, die mich in dieser intensiven Zeit auf andere Gedanken gebracht haben.

Graz, im März 2015

Christina Schwarz

Contents

| | | |
|----------|---|-----------|
| 1 | Introduction | 11 |
| 1.1 | Motivation | 11 |
| 1.2 | Ambition | 11 |
| 1.3 | Outline | 12 |
| 2 | Implementation strategy | 13 |
| 2.1 | Comparison of already existing simulation and modeling environments | 13 |
| 2.1.1 | Modelica based software (OpenModelica and Dymola) | 13 |
| 2.1.2 | SimScape | 14 |
| 2.2 | Comparison of different implementation strategies | 14 |
| 2.2.1 | Usage of OpenModelica as graphical user interface | 14 |
| 2.2.2 | Comparison of programming languages for (GUI) creation | 15 |
| 2.2.3 | Comparison of programming languages for mathematical computation | 15 |
| 2.2.4 | Final decision | 16 |
| 2.3 | Object-oriented programming | 17 |
| 2.3.1 | Agents and messages | 17 |
| 2.3.2 | Classes and instances | 17 |
| 2.3.3 | Inheritance and polymorphism | 17 |
| 3 | Automotive transmissions and modeling concept | 19 |
| 3.1 | Mechanical components in a drivetrain | 19 |
| 3.1.1 | Interfaces | 19 |
| 3.1.2 | Moment of inertia | 20 |
| 3.1.3 | Shaft | 20 |
| 3.1.4 | Friction elements | 21 |
| 3.1.5 | Planetary gear | 22 |
| 3.2 | Manual transmission | 24 |
| 3.3 | Automated manual transmission | 24 |
| 3.4 | Dual-clutch transmission | 24 |
| 3.5 | Automatic transmission | 25 |
| 3.6 | Continuously variable transmission | 26 |
| 3.7 | Modeling concept | 27 |
| 3.8 | Summary | 28 |

| | | |
|----------|---|-----------|
| 4 | Drivetrain design | 29 |
| 4.1 | Graphical user interface - drivetrain configuration | 29 |
| 4.2 | Components | 32 |
| 4.2.1 | External input | 32 |
| 4.2.2 | Ground | 33 |
| 4.2.3 | Inertia with output | 33 |
| 4.2.4 | Shaft | 34 |
| 4.2.5 | Gear | 35 |
| 4.2.6 | Clutch/Brake | 36 |
| 4.2.7 | Multiplexer (mux) | 36 |
| 4.2.8 | Planetary gear set | 36 |
| 4.2.9 | Extended Ravigneaux gear set | 37 |
| 4.2.10 | Specific gear set | 38 |
| 4.3 | Save and load configuration | 39 |
| 5 | Automated drivetrain parameterization | 41 |
| 5.1 | Input | 44 |
| 5.2 | Clutch/Brake | 44 |
| 5.3 | Shaft | 44 |
| 5.4 | Planetary gear set | 46 |
| 5.5 | Extended Ravigneaux gear set | 46 |
| 5.6 | Specific gear set | 46 |
| 5.7 | Short summary | 47 |
| 6 | Tool validation | 49 |
| 6.1 | Conventional automatic transmission | 49 |
| 6.2 | Exemplary dual clutch transmission | 53 |
| 6.3 | Complex hybrid-electric automatic transmission | 55 |
| 6.4 | Summary | 57 |
| 7 | Conclusion and Outlook | 59 |
| 7.1 | Conclusion | 59 |
| 7.2 | Outlook | 59 |
| A | MATLAB initialization files | 61 |
| A.1 | DCT configuration | 61 |
| A.2 | Future hybrid configuration | 63 |
| A.3 | Conventional drivetrain configuration | 66 |
| B | Definitions | 69 |
| B.1 | Abbreviations | 69 |
| B.2 | Used symbols | 69 |
| | Bibliography | 70 |

List of Figures

| | | |
|------|--|----|
| 2.1 | Example of a class hierarchy for various material objects taken from [11] | 18 |
| 3.1 | Torque explanation | 20 |
| 3.2 | Model of a moment of inertia taken from [14]. | 20 |
| 3.3 | Model of a elastic shaft taken from [14]. | 21 |
| 3.4 | Model of a clutch taken from [14]. | 21 |
| 3.5 | Simple planetary gear with four planets | 22 |
| 3.6 | Commonly used transmission types taken from [18]. | 24 |
| 3.7 | Mechanical diagram of a dual clutch transmission taken from [20] | 25 |
| 3.8 | AVL future hybrid transmission [21] | 26 |
| 3.9 | Exemplary CVT taken from [14] | 26 |
| 3.10 | Block diagram of the drivetrain model taken from [4] | 27 |
| 4.1 | Illustration of the developed GUI for drivetrain design | 29 |
| 4.2 | Class diagram of the solution | 30 |
| 4.3 | Inheritance diagram of the specific components including the parent class. | 31 |
| 4.4 | Parent class component with parameters. | 32 |
| 4.5 | External input icon and parameter configuration. | 33 |
| 4.6 | Icon of the ground component. | 33 |
| 4.7 | Inertia icon and parameter configuration. | 34 |
| 4.8 | Shaft icon and parameter configuration. | 35 |
| 4.9 | Input icon and parameter configuration. | 35 |
| 4.10 | Clutch icon and parameter configuration. | 36 |
| 4.11 | Multiplexer icon. | 36 |
| 4.12 | Planetary gear set icon and parameter configuration. | 37 |
| 4.13 | Extended Ravigneaux gear set icon and parameter configuration. | 38 |
| 4.14 | Extended Ravigneaux gear set icon and parameter configuration. | 39 |
| 4.15 | XML-file of an example configuration | 40 |
| 5.1 | Exemplary drivetrain configuration containing a shaft. | 45 |
| 6.1 | Diagram of a conventional automatic transmission. | 49 |
| 6.2 | Implementation of a conventional automatic transmission | 51 |
| 6.3 | Comparison of measurement data with simulation | 52 |
| 6.4 | Detailed comparison of measurement data and simulation | 53 |
| 6.5 | Diagram of a demo dual clutch transmission. | 54 |
| 6.6 | Implementation of a demo dual clutch transmission. | 54 |

| | | |
|-----|---|----|
| 6.7 | Diagram of a hybrid-electric transmission. | 55 |
| 6.8 | Implementation of a hybrid-electric transmission. | 56 |

List of Tables

| | | |
|-----|---|----|
| 2.1 | Main differences between Java and C++ | 16 |
| 3.1 | Overview over transmission ratios of a simple planetary gear | 23 |
| 4.1 | Torque relations between inputs of a simple planetary gear set depending on connected inertias | 37 |
| 4.2 | Torque relations of inputs of an extended Ravigneaux gear set. | 38 |
| 4.3 | Torque relations of inputs of an specific gear set. | 38 |
| 6.1 | List of all used components with their used icon in the diagram and the implementations. | 50 |
| 6.2 | Model parameters for simulation of driving cycle. | 53 |

Chapter 1

Introduction

1.1 Motivation

Automotive industry nowadays has to handle increasingly complex software systems in vehicles. According to [1] there are up to 80 different electronic control units (ECUs¹) which are connected via several different communication buses in one vehicle. These ECUs control the functionalities of the vehicle such as braking, shifting, etc. There are even more components such as electric machines and embedded powertrain controllers involved when developing electric, hybrid or fuel cell vehicles [2]. Model-based design simplifies the development of these mechatronic systems by providing an environment which includes both, design and communication across several engineering disciplines. Using a model-based approach, engineers are able to test their design at an early stage and can fix errors faster [3].

In this Master's thesis a tool has been developed, allowing the graphical build up of a geared transmission, by arranging, interconnecting and parameterizing its mechanical components. The tool covers the generation of simple conventional drivetrain layouts as well as complex hybrid transmissions comprising planetary gear sets, multiple clutches and propulsion sources. This enables tool-guided modeling by non-experts in the fields of mechanic engineering and control theory leading to reduced development and testing efforts.

1.2 Ambition

The aim of this thesis is twofolded, firstly, a graphical user interface (GUI) for schematic drivetrain model design should be created and, secondly, parameterization files of these designed drivetrains should be generated automatically. The basis of the development is the generic real-time drivetrain model presented in [4] which will be described in more detail later. As the model is generic, the drivetrain layout is defined by the model parameters only. Those parameters shall be found in a guided and automated way. As a result the drivetrain model is parameterized with a file generated by the tool.

¹For the readers convenience the abbreviations are summarized in appendix B.1

1.3 Outline

The contents of this thesis are structured as follows: chapter **2** covers a comparison of the already existing software for modeling and simulation. Furthermore, the implementation possibilities are discussed and a short overview over object-oriented programming is given. Chapter **3** describes the main topologies of automotive transmissions and the basic mechanical components needed for real-time modelling of a drivetrain. The main topologies of automotive transmissions are described in chapter **3**. Additionally, this chapter presents a short overview of the model, which was used as development base for the drivetrain design and parameterization.

Chapter **4** covers the realization of the GUI for drivetrain design. Therefore, the structure of the GUI is presented and the implemented components are described. Furthermore, this chapter covers the save and load strategy. The algorithm for the automated parameterization of these drivetrains, including a simplified pseudo code, is described in chapter **5**.

The tool has been tested on different drivetrain topologies, the topologies and the according results are presented in chapter **6**. The conclusion and a short outline on further implementation plans is presented in chapter **7**.

Chapter 2

Implementation strategy

In this chapter the already existing environments for simulation and modeling are compared. Furthermore, the different implementation possibilities concerning the usage of already existing environments and the different programming languages are listed.

2.1 Comparison of already existing simulation and modeling environments

In the automotive industry many different component based modeling languages for multi-body simulation exist. In this section the standard languages such as OpenModelica, SimScape, and Dymola will be discussed, details are published in [5].

2.1.1 Modelica based software (OpenModelica and Dymola)

Modelica is an object-oriented, multi-domain modeling language for component-oriented modeling of complex systems. These complex systems can contain subcomponents of various domains of physics such as mechanics, hydraulics, electronics, etc. Additionally these systems can be extended by mathematical and control components. The open source Modelica Standard Library is extended by developers of the organization Modelica Association. One essential characteristic of Modelica is that equations do not describe assignment but equality. As a result equations in Modelica have no pre-defined causality. There are many simulation environments based on Modelica, two of the most important will be presented now.

OpenModelica

OpenModelica is an open-source software and therefore can be used for free [6]. It provides libraries with components for modeling and simulation, but also new elements can be designed. Furthermore an interface to MATLAB is implemented. According to [5] the integration of new components is rather difficult and there is no possibility to create s-functions compatible to Simulink.

Dymola (DYnamic MOdeling LAboratory)

Dymola is a commercial product for modeling of physical objects of distinct types. Similar to OpenModelica many libraries containing objects of distinct section in physics are provided. Furthermore the source code of the components can be seen and modified. Hence, it is easier to understand the functionality of the components and modify them. The transformation of a Dymola model to an s-function is in general possible but the user needs a special license to do so. Furthermore the generated s-function needs Dymola running in background and therefore the s-function can not be used on computers without Dymola. Additionally there is a co-simulation interface to MATLAB.

2.1.2 SimScape

MATLAB provides opportunities for simulation, one of it is Simulink. Since Simulink is a signal based modeling language, the signal direction is unidirectional and therefore, interactions between components and systems are not considered. To model these interactions additional signal paths have to be modeled to provide a bidirectional signal path. The Simulink extension **SimScape** is, similar to Modelica, component based and contains predefined libraries and components, which are especially suitable for modeling of physical systems. SimScape also provides toolboxes which provide modified and application-specific components for different physical sections. A modification of the provided components is not possible.

2.2 Comparison of different implementation strategies

In this chapter the different possibilities for the implementation of the Graphical User Interface (GUI) are discussed. The focus of the discussion is put on on a customer-friendly handling. Therefore not only the different programming languages are compared but also the use of already existing software for the graphical development of drivetrains is taken into consideration.

2.2.1 Usage of OpenModelica as graphical user interface

OpenModelica is a open source software for modeling and simulation [6]. It is a environment for Modelica modeling, compilation, and simulation. The idea is to create the mechanical model of the drivetrain with OpenModelica and write another program to interpret and process this model. The advantage of using this software is that there is already an existing GUI for the drivetrain model design. Additionally, the mechanical models are saved in a format which can be interpreted and processed by a suitable programming language for further usage. However, only a few of the components provided by OpenModelica can be used for the design of drivetrain topology. Hence, it is very important for the user to recognize which elements can be used. Since the aim of this thesis is to enable drivetrain design for non-experts in the fields of mechanics, a program in the background has to check every component and connection of the user. Additionally OpenModelica is not installed on every computer by default. Due to the fact, that the user has to handle at least two interfaces, one for the graphical modeling and one for the interpretation of the model, the customer-friendly usage is questionable.

2.2.2 Comparison of programming languages for (GUI) creation

In the following subsections the different programming languages are compared with regard to their suitability for GUI creation. Owing to the demand that different components of the drivetrain should be connected via *Drag and Drop*, this property is discussed in particular.

Matlab

Matlab is a numerical computing environment developed by MathWorks [7]. However it is also possible to create simple GUIs. Due to the fact that Matlab is created for numerical computing, more complex GUIs with the realization of *Drag and Drop* are difficult to implement. Hence, Matlab is possible but not suitable for the development of the GUI of this project.

JavaScript

Another possibility is to create a GUI with JavaScript. It is a dynamic computer programming language and is mostly used as part of web browsers. The advantage of this language is that it is easy to learn and to use. Besides the realization of *Drag and Drop* is very simple. In contrary the interpretation of the data for further processing is difficult and the application runs on browsers only.

Java

Java is a general-purpose programming language which is class-based and object-oriented. One great advantage of Java is its platform independence which means it does not have to be recompiled if it is transferred to another platform. To develop a GUI Java provides the Java Swing library which allows an easy GUI creation and implementation of *Drag and Drop*. Furthermore the object-oriented structure of Java is well suited for the development of the further processing of the drivetrain configuration.

C++

The last discussed programming language is C++ and is another general-purpose programming language. Same as Java, it is object-oriented and can create applications which run on every Operating System (OS) although they have to be recompiled on each platform. The Qt-Library provided by C++ can be used for GUI development. There is also a well documented development environment Qt-Creator which simplifies the creation of GUIs.

2.2.3 Comparison of programming languages for mathematical computation

Now the different programming languages will be discussed regarding the further interpretation and automated parameterization of the designed drivetrain topology.

Matlab

Matlab may not be the best possibility for interpreting the drivetrain topology, but for processing the data. The aim of this thesis is to create an initialization file for a Matlab/Simulink Model. Therefore it would be an opportunity to create and interpret the configuration with another program and to create the initialization file with Matlab.

JavaScript

Due to the fact that JavaScript is commonly used in web browsers it is not well suited for the interpretation and further processing of the model.

C++ and Java

Owing to the object-oriented structure of C++ and Java it is easier to create complex programs in contrast to JavaScript. Additionally mathematical operations can be implemented easier with appropriate libraries. Nevertheless, it is more difficult to process the information of the drivetrain configuration than in Matlab.

2.2.4 Final decision

Even though Matlab is very useful when processing mathematical data the possibilities for the GUI creation are too poorly developed for the purpose of this thesis. In contrary the GUI creation is very simple with JavaScript but the further processing of the model is too difficult. An opportunity would be to develop the GUI in JavaScript and process the data in Matlab. Since the targeted customer-friendliness can not be maintained due to the usage of at least two interfaces this opportunity is eliminated. Furthermore, the usage of OpenModelica as GUI with external processing of data is ruled out as well as again two interfaces would be needed.

To make a final decision, the two remaining programming languages are discussed in more detail. Java is easier to handle for inexperienced programmers since the rudimentary tasks such as freeing memory are done by Java itself. According to [8] this is done cyclic in background and leads to a poorer run-time and memory efficiency. When using C++ the memory has to be released by the programmer. As a result the memory can be released immediately but the risk of memory leaks is much higher. The main differences between Java and C++ are shown in tab. 2.1.

| | C++ | Java |
|-----------------------|-------------------------------|--------------------------|
| platform independence | write once, compile anywhere | write once, run anywhere |
| programming paradigm | procedural or object-oriented | object-oriented |
| memory management | manually | garbage collector |
| GUI creation | Qt | Swing |
| focus | execution efficiency | developer productivity |
| preprocessor | yes | no |

Table 2.1: Main differences between Java and C++

C++ is normally compiled to machine code which is executed directly by the operating system whereas Java is first compiled to byte-code which is compiled to machine code and executed by either interpreters or „Just in Time“ (JIT) compilers of the Java virtual machine (JVM) [9]. Hence, Java may have a little longer execution time. However, Prechtelt [10] wrote:

‘Interpersonal variability, that is the capability and behavior differences between programmers using the same language, tends to account for more differences between programs than a change of the programming language.’ [10]

Summarizing the above, it can be stated that C++ with Qt for GUI development is more suitable when creating large projects concerning run-time and memory management. As Prechtelt [10] wrote, programming capabilities of a programmer contribute more to the program efficiency than the programming language. Due to the easier development and data management, Java was chosen as developing language. Within this program run-time considerations are irrelevant, as the program will be used offline and the overall project code line number should be held small.

2.3 Object-oriented programming

Object-oriented programming (OOP) is a programming paradigm based on objects that contain data fields, often known as attributes and code in form of procedures, often known as methods. In this chapter the major design patterns of object-oriented programming are discussed [11].

2.3.1 Agents and messages

To perform an action in object-oriented programming a message is transmitted to an agent (an object) responsible for the action. This message consists of the request for an action and additional information (arguments). If the receiver accepts the message, it also accepts the responsibility to carry out the action by performing a method to satisfy the request. This also implies the principle of information hiding which means the client sending the request does not need to know the details of how the request is satisfied [11].

2.3.2 Classes and instances

The second principle of object-oriented programming is distinction between classes and instances. A class describes the architecture of objects and is not used directly. Commonly all objects are instances of a class. If a method is invoked in response to a message, it was determined by the class of the receiver. Every instance of a certain class uses the same method in response to similar messages [11].

2.3.3 Inheritance and polymorphism

The principle that knowledge of a more general category is also applicable to a more specific category is called inheritance. Classes can be organized in a hierarchical structure where a child class will inherit attributes and methods from a parent class. Abstract

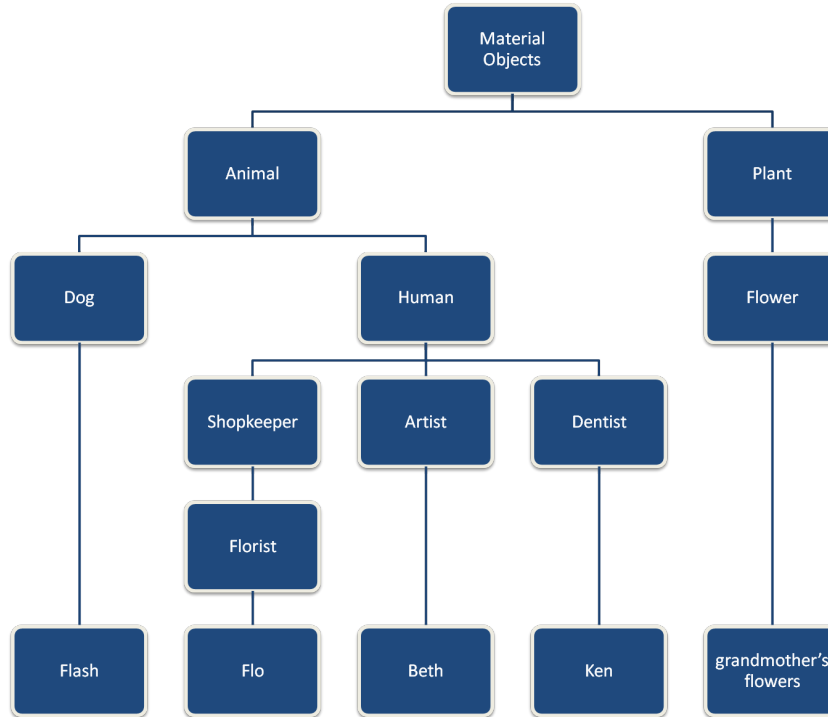


Figure 2.1: Example of a class hierarchy for various material objects taken from [11]

parent classes are classes which are used only to create subclasses. Methods can be either defined in a parent class or in the child class. Additionally there is the possibility to override a method. If methods with the same name exist in both, parent and child class, the method executed is said to override the inherited method.

The principle of polymorphism allows shared code to be tailored to fit the specific circumstances of individual data types. For example an instance of the class component can either be an input but also a much more complex planetary gear set [12]. An example for a class hierarchy is shown in fig. 2.1. It includes parent and child classes with according instances.

Chapter 3

Automotive transmissions and modeling concept

Various types of geared transmissions are available in the automotive industry. In this chapter commonly used transmission systems will be described. For a better understanding of the behavior of these transmissions some mechanical components are described. Within this work all units are SI, except if especially stated differently. The symbols used for variables in this thesis are listed in B.2.

3.1 Mechanical components in a drivetrain

3.1.1 Interfaces

The interfaces, rotational speeds and torques, between these components will be briefly introduced.

Rotational speed is the number of rotations per time unit and is usually measured in hertz. The used symbol within this thesis is ω . The relation between the rotational speed ω and the velocity v is given through $2\pi r$.

Torque is a physical dimension and describes the tendency of a force to rotate an object about an axis. The used symbol for torque is τ . The magnitude of torque depends on the force applied and the length of the lever arm.

Fig. 3.1 shows a force F action on an object with radius r . The radius is measured from the axis of rotation to the point of application of the force. An easy way to calculate the magnitude of the torque is, to determine the lever arm first which is $r \sin \Theta$. Then it has to be multiplied by the applied force. Torque has dimension of force times distance [13].

$$\tau = rF \sin \Theta \quad (3.1)$$

The product of torque and rotational speed is power.

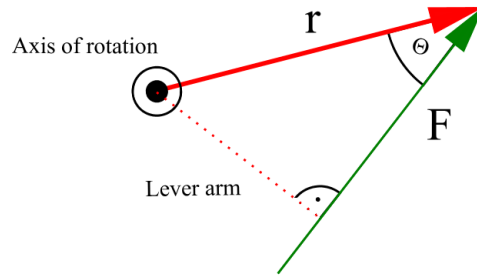


Figure 3.1: Force acting on an object creating a torque.

3.1.2 Moment of inertia

Moments of inertia are very fundamental in the terms of this work, since every mass is modeled with a moment of inertia. Fig. 3.2 illustrates the model of such a moment of inertia.

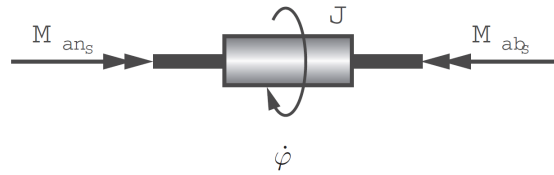


Figure 3.2: Model of a moment of inertia taken from [14]. In this figure the torques have the symbol M and the rotational speed has the symbol $\dot{\varphi}$.

Moments of inertia are the simplest possible elements of a machine including input and output torque and a rotational inertia. The equation of motion is simply

$$\underbrace{J}_{M_{RS}} \underbrace{\ddot{\varphi}}_{q_{RS}} = \underbrace{M_{ans} - M_{abs}}_{h_{RSs}} \quad (3.2)$$

3.1.3 Shaft

Shafts are fundamental parts in mechanical systems. Due to various design possibilities shafts possess various influence on the overall system.

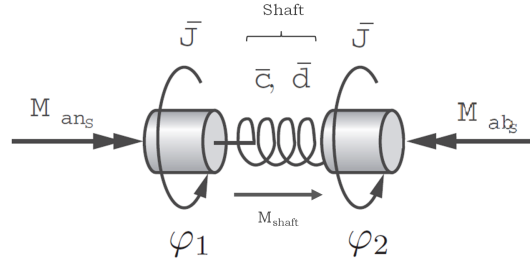


Figure 3.3: Model of an elastic shaft taken from [14]. \bar{J} represents the moment of inertia, the term M is used for torque and $\dot{\varphi}$ represents the rotational speed.

Elastic shafts represent the simplest possible case of an elastic multibody element when rotational linear elasticity is concerned. Since a shaft has to be connected to inertias to derive an equation of motion, the system in fig. 3.3 depicts two inertia elements interconnected by springs and dampers of a shaft. The torque of the shaft can be seen in eqn. (3.3).

$$M_{shaft} = (\varphi_1 - \varphi_2)c + (\dot{\varphi}_1 - \dot{\varphi}_2)d \quad (3.3)$$

The equations of motion for the whole system indicated in fig. 3.3 are shown in eqn. (3.4).

$$\underbrace{\begin{pmatrix} \bar{J} & 0 \\ 0 & \bar{J} \end{pmatrix}}_{M_{ES}} \underbrace{\begin{pmatrix} \ddot{\varphi}_1 \\ \ddot{\varphi}_2 \end{pmatrix}}_{\dot{\varphi}_{ES}} = \underbrace{\begin{pmatrix} (\varphi_2 - \varphi_1)c + (\dot{\varphi}_2 - \dot{\varphi}_1)d \\ (\varphi_1 - \varphi_2)c + (\dot{\varphi}_1 - \dot{\varphi}_2)d \end{pmatrix}}_{h_{ES}} + \underbrace{\begin{pmatrix} M_{an_s} \\ -M_{ab_s} \end{pmatrix}}_{h_{ES_s}} \quad (3.4)$$

3.1.4 Friction elements

Friction elements play an important role in geared transmissions. The term „clutches“ within this work, whenever appearing, refers to all types of friction elements which can be found in clutches, brakes and mechanical synchronizing elements. They cause a variable order and structure system as the number of mechanical degrees of freedom depends on the clutch state („slip“ or „stick“) as well as the clutch torque dependency changes with the clutch state.

A clutch is a device which is used to connect two shafts at their ends to transmit power. A part of the power is lost due to speed difference between the clutch plates.

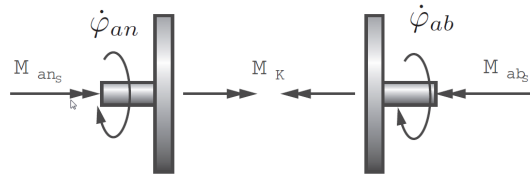


Figure 3.4: Model of a clutch taken from [14]. Here M is used for torque instead of τ and $\dot{\varphi}$ instead of ω for rotational speed.

Clutches are essential for transmissions and therefore are used in a wide range. They are controlled hydraulically and transmit torques between gear components. A simple model of a clutch is shown in fig. 3.4.

3.1.5 Planetary gear

As described in [15] and [16] planetary gears are gear systems which at least have three coaxial shafts. The designation *planetary gear* is derived from the arrangement of the gearwheels since the planet gears are arranged around the sun gear. Planet gears are mounted on a carrier and normally circle the sun gear. Planetary gears also incorporate a outer gear which meshes with the planet gears and is called ring gear. A great advantage of the planetary gear set is their little installation space compared to their high transmission ratios.

In fig. 3.5 the schematic representation of a planetary gear is shown. The most important advantage of planetary gears for transmission systems is the easy realization of different gear transmission ratios by holding sun, carrier or ring stationary. Each shaft can overtake the role of input or output. The gear ratios are determined by the gear teeth, the selection of the in- and output shaft.

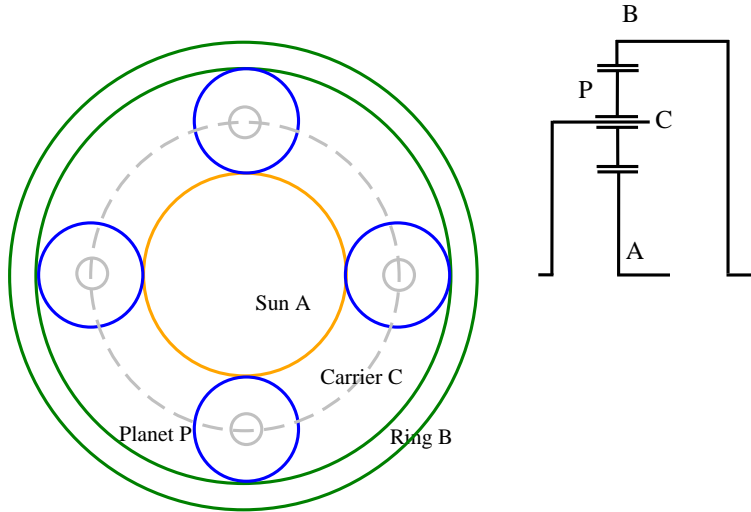


Figure 3.5: Simple planetary gear with four planets: schematic representation in view and side view.

With the WILLIS-Equations given in eqn. (3.8) all gear ratios can be calculated, the following listed ratios are special cases. There are three different shaft bearing possibilities at a simple planetary gear.

- **carrier held stationary:** First of all, the stationary gear which results from holding the carrier stationary has to be analysed. If the sun gear is used as input with an input speed of n_{in} and the ring is used as output, the direction of rotation is inverted. The overall gear ratio is

$$i = \frac{\omega_{in}}{\omega_{out}} = \frac{\omega_A}{\omega_B} = \frac{z_B}{z_A} \stackrel{\text{def}}{=} i_0 \quad (3.5)$$

whereat z is the number of teeth of each gear. This is defined as the stationary gear ratio i_0 , $i_0 < -1$.

- **sun held stationary:** If the sun is held stationary and the ring is used as input the carrier has to be the output with the same rotation direction as the ring. Due

to the same rotation direction the overall gear ratio from input to output shaft is speed reducing.

$$i_{ov} = \frac{\omega_{in}}{\omega_{out}} = \frac{\omega_B}{\omega_C} = \frac{z_B - z_A}{z_B} = 1 - \frac{1}{i_0} \quad (3.6)$$

- **ring held stationary:** If the ring is held stationary it is not only possible to generate a speed increase $0 < i < 1$ but also a speed reduction $i > 1$ by maintaining the rotation direction. If the sun is used as input the carrier has to rotate with half of the rotational speed of the planet gears. In this case the overall gear ratio is

$$i_{ov} = \frac{\omega_{in}}{\omega_{out}} = \frac{\omega_A}{\omega_C} = \frac{z_A - z_B}{z_A} = 1 - i_0 \quad (3.7)$$

If two of the three inputs are linked by a clutch the planetary gear is a block with $i = 1$.

| Input | Output | Stationary | ratio $i = \omega_{in}/\omega_{out}$ | note |
|-------------|-------------|-------------|---|---|
| sun (A) | ring (B) | carrier (C) | $i = i_0 = \frac{z_B}{z_A}$ | rotation direction inversion to reduction, $-\infty < i < -1$ |
| ring (B) | sun (A) | carrier (C) | $i = 1/i_0 = \frac{z_A}{z_B}$ | rotation direction inversion to step-up, $-1 < i < 0$ |
| ring (B) | carrier (C) | sun (A) | $i = 1 - 1/i_0$ | transmission reduction, $1 < i < 2$ |
| carrier (C) | ring (B) | sun (A) | $i = i_0/i_0 - 1$ | transmission increase, $0.5 < i < 1$ |
| sun (A) | carrier (C) | ring (B) | $i = 1 - i_0$ | transmission reduction, $2 < i < \infty$ |
| carrier (C) | sun (A) | ring (B) | $i = 1/1 - i_0$ | transmission increase, $0 < i < 0.5$ |

Table 3.1: Overview over transmission ratios of a simple planetary gear

If the planetary gear is deployed as summation gearbox, power is applied over two shafts and the total power is delivered at the third 'free' shaft. For the identification of the speed of all shafts it is sufficient to know the two input speeds. If it is deployed as a transmission gearbox, the applied power is split over two outputs. To identify the speed of all shafts the speed of the input shaft and the relation between the two output speeds are needed. To gain these relationships kinetic and kinematic constraints have to be considered. The kinematic constraint is defined by the WILLIS-Equation which is formulated by

$$\omega_A - \omega_B i_0 - (1 - i_0)\omega_C = 0. \quad (3.8)$$

The kinematic constraints are defined by the equilibriums of power eqn. (3.9) and external torques eqn. (3.10) for a planetary gear set.

$$\sum P = \omega_A \tau_A + \omega_B \tau_B + \omega_C \tau_C = 0 \quad (3.9)$$

$$\sum \tau = \tau_A + \tau_B + \tau_C = 0. \quad (3.10)$$

| Transmission topologies | | | | |
|-------------------------|---------------------------|------------------------|----|--|
| Geared transmission | | | | Continuous transmission |
| MT | AMT | DCT | AT | CVT |
| Torque interruption | | No torque interruption | | |
| Manually gear shift | Semi-automatic gear shift | Automatic gear shift | | Automatic torque and rotational speed converting |

Figure 3.6: Commonly used transmission types taken from [18].

All possible transmission ratios can be lead back to the stationary gear ratio in tab. 3.1 with the WILLIS-Equation eqn. (3.8) and the equations eqn. (3.9) and eqn. (3.10) [17]. Fig. 3.6 illustrates an overview over the commonly used transmission topologies which will be described in the following sections.

3.2 Manual transmission

Manual transmissions (MT) are widely used due to their low production costs and their high efficiency. The overall vehicle efficiency heavily depends on the driver. They consist of a driver-operated clutch and a movable gear stick. The gear pairs for each gear are arranged at at least two parallel running shafts, whereas one is coupled with a shaft while the other spins free on the other shaft. Only if a gear is selected, the free flywheel is connected to the shaft over a switchgear unit [19].

3.3 Automated manual transmission

An automated manual transmission (AMT) is an automotive transmission which has principally the same topology as a MT. Electronic sensors, pneumatics, processors and actuators are used to execute gear shifts. This removes the need of a clutch pedal, since the clutch itself is actuated by electronic equipment, which allows quick, smooth gear shifts. An automation of the gear selection can be achieved by using hydraulic or electromechanical components. The gear selection program and the switching point as well as the shifting process require software functions. These software functions are embedded in the transmission control unit [19]. AMTs have a high overall efficiency, since the efficiency is not depended on the driver. Optionally, many AMTs provide a manual gear selection for the driver.

3.4 Dual-clutch transmission

A dual-clutch transmission (DCT) is a transmission topology which allows a gear shift without torque interruption. It combines the comfort of an automatic transmission with

the efficiency of a manual transmission. Two separate clutches are used, one for even and one for odd gears. Generally it can be described as two separate manual transmissions in one housing, working as one unit. This allows the control strategy to select two gears at one time. The actual gear shift takes place through releasing the active clutch and simultaneously coupling the passive one, which allows a gear shift without an interruption of the traction. Usually DCTs are operated in a fully automatic mode, but many also have the ability to allow the driver to manually shift gears in a semi-automatic mode [19].

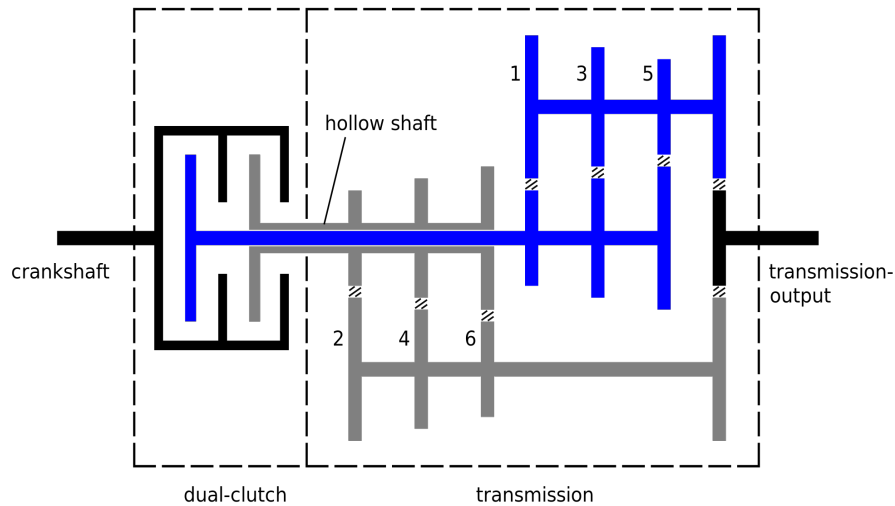


Figure 3.7: Mechanical diagram of a dual clutch transmission taken from [20]

3.5 Automatic transmission

Automatic transmissions generally consist of planetary gear sets. In standard applications torque converters (TC) are used to connect the combustion engine with the gear box. These TCs can be replaced by friction clutches and in hybrid-electric applications the electric motor can overtake the function. In fig. 3.8 an exemplary automatic transmission for a hybrid is illustrated taken from [21].

The sun gear of the standard planetary gear set (PSG 0) is fixed to the housing, the ring gear is connected to the internal combustion engine (ICE) via the separation clutch C0. The extended Ravigneaux gear set (PGS 1) consists of two sun gears, two ring gears and two planet gears which share one common carrier. Sun 1 of PGS 1 is connected to the carrier of PGS 0 over the clutch C2, sun 2 is connected to the housing via brake B1. Ring gear of PGS 0 and carrier are linked via clutch C1, ring 1 is connected to electric motor and ring 2 is the output of the transmission.

Hydrodynamic converters

Hydrodynamic converters are basic components of automatic gear boxes and consist of a pump, a turbine and an impeller. Due to their specific design even large speed differences (i.e. starting car) can be balanced. They also include a special clutch bridging the con-

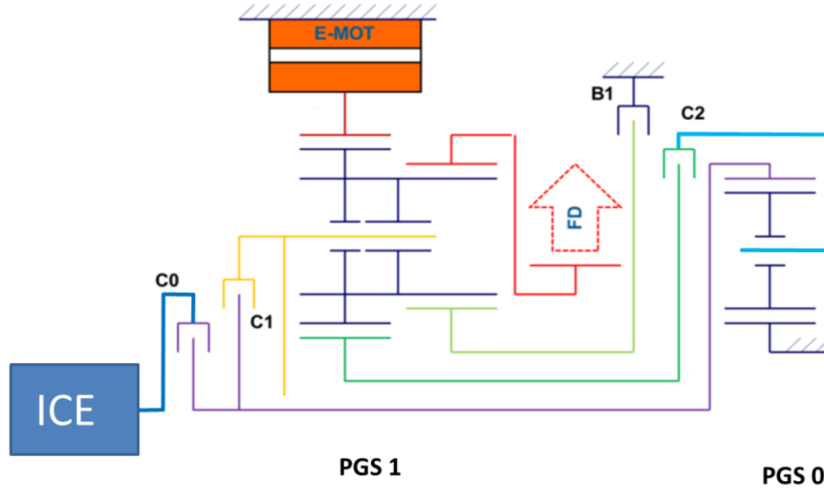


Figure 3.8: AVL future hybrid transmission with an extended Ravigneaux (PGS1) and a planetary (PGS 0) gear set taken from [21]. The four friction elements are one brake (B1) and three clutches (C0, C1, and C2) including a separation clutch. This allows the transmission to operate at 7 different operating modes.

verter and the operation in a closed, open and in a slip state. Hence, such torque converter can increase augment comfort and reduce fuel consumption with an appropriate control [14].

3.6 Continuously variable transmission

Another transmission type is the continuously variable transmission (CVT). The advantage of gear trains with gear wheels consists in a better component efficiency due to power transmission form closure, the disadvantage is an only step wise approximation of the drag-velocity hyperbola. The advantages of the CVT configuration is a perfect adaption to the drag-velocity hyperbola and the possibility of a very smooth change of the transmission ratio without any danger of generating a jerk. The disadvantages are a lower efficiency due

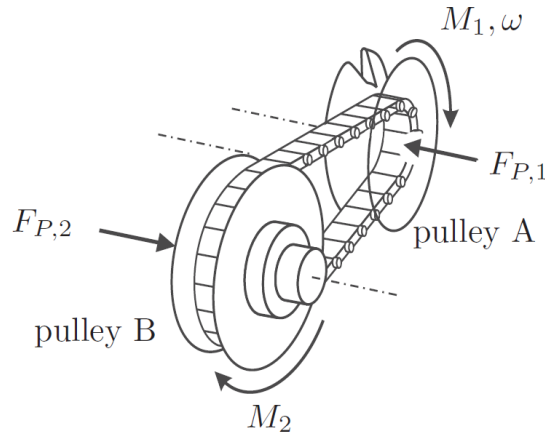


Figure 3.9: Exemplary CVT taken from [14]

to power transmission by friction and a somehow limited torque transmission [14]. The fig. 3.9 gives an impression of the system and its components. Many different configurations are used in the industry, but the operation of a chain- or belt-driven CVT is always the same. The chain or belt moves between two pulleys with conically shaped sheaves, whereas one side of these pulleys possesses a movable sheave controlled by a hydraulic system and the other side is fixed. Reducing or increasing the distance between the pulley sheaves forces the chain or belt to move radially upwards or downwards thus changing the transmission ratio of the CVT.

3.7 Modeling concept

Modeling drivetrain motion bases on the equations of motion of a multibody system. As already mentioned above geared transmissions contain clutches and, due to the clutch state change (stick vs. slip), are systems of variable structure and order. Hence, a mathematical description for each friction element state combination has to be set up. Therefore, a method with a single model and implicit consideration of clutch locking states is presented in [4].

This concept allows a unique model code for different drivetrain topologies only by adapting the parameterization. The clutch slip drivetrain model shown in fig. 3.10 only covers slipping and open clutches. Locked clutch torques are functions of internal states, external inputs and torques of slipping clutches. Locked clutch behavior is considered via nonlinear feedback, as indicated in fig. 3.10.

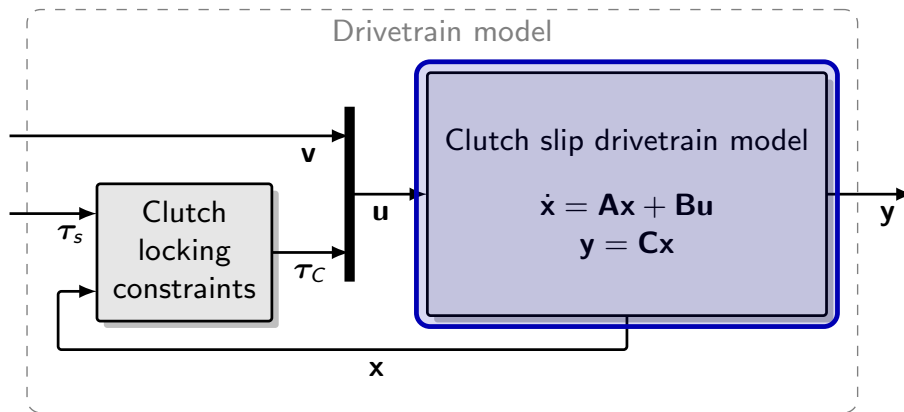


Figure 3.10: Block diagram of the drivetrain model taken from [4]

The model uses input torques \mathbf{v} , provided by propulsion and load elements, and clutch slip torques τ_s . An approximation, that possible stiffness (k) and damping (d) factors are constant, is justified for real-time execution limitations. Furthermore, inertias and shafts are considered as lumped inertias and shafts. This approximation allows representation of the clutch slip drivetrain model by a continuous-time state-space model eqn. (3.11) using the following definitions:

- \mathbf{x} : summarizes the minimum number of required states consisting of rotational speeds of inertia elements and angular position differences over spring elements.

- \mathbf{u} : represents the external inputs \mathbf{v} and clutch torques τ_c .
- \mathbf{y} : available sensor signals.

$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \\ \mathbf{y} &= \mathbf{C}\mathbf{x}\end{aligned}\tag{3.11}$$

Using the mass matrix \mathbf{M} the system matrices \mathbf{A} and \mathbf{B} can be defined as $\mathbf{A} = \mathbf{M}^{-1}\bar{\mathbf{A}}$ and $\mathbf{B} = \mathbf{M}^{-1}\bar{\mathbf{B}}$, slightly reformulating eqn. (3.11) to eqn. (3.12).

$$\begin{aligned}\mathbf{M}\dot{\mathbf{x}} &= \bar{\mathbf{A}}\mathbf{x} + \bar{\mathbf{B}}\mathbf{u} \\ \mathbf{y} &= \mathbf{C}\mathbf{x}\end{aligned}\tag{3.12}$$

The drivetrain topology is completely defined by the parameters of the matrices $\bar{\mathbf{A}}, \bar{\mathbf{B}}, \mathbf{C}$ and \mathbf{M} in the context of this approach. To obtain these parameters it is sufficient to formulate the equations of motion and define the states \mathbf{x} , inputs \mathbf{u} and outputs \mathbf{y} . With the developed tool it is convenient to draw the schematic of a drivetrain topology and let the algorithm derive the parameters of the system matrices automatically.

This model is valid for continuous-time simulation which in most cases uses variable step solvers. For real-time execution adaptive time-step methods are not feasible. Therefore, fixed-step execution is required. A method is proposed in [4] to transform this continuous-time model to discrete-time as well as to deal with large fixed time-steps, when model execution will not coincide with clutch state change instants.

3.8 Summary

The main transmission topologies used in passenger cars and their mechanical components have been presented in this chapter. Furthermore, the modeling concept, which was used as the development base for this Master's thesis, has been described. With this knowledge the design of the GUI and the algorithm development has begun.

Chapter 4

Drivetrain design

This chapter covers the implementation of the GUI for the drivetrain design was developed with the common known development tool *eclipse*¹. As mentioned before the used programming language is Java with Java Swing for the realization of the GUI. Additionally, the library Forms 1.0.3² was used for better GUI design.

4.1 Graphical user interface - drivetrain configuration

The GUI is called „Drivetrain Configurator“, it consists of four components and is shown in fig. 4.1.

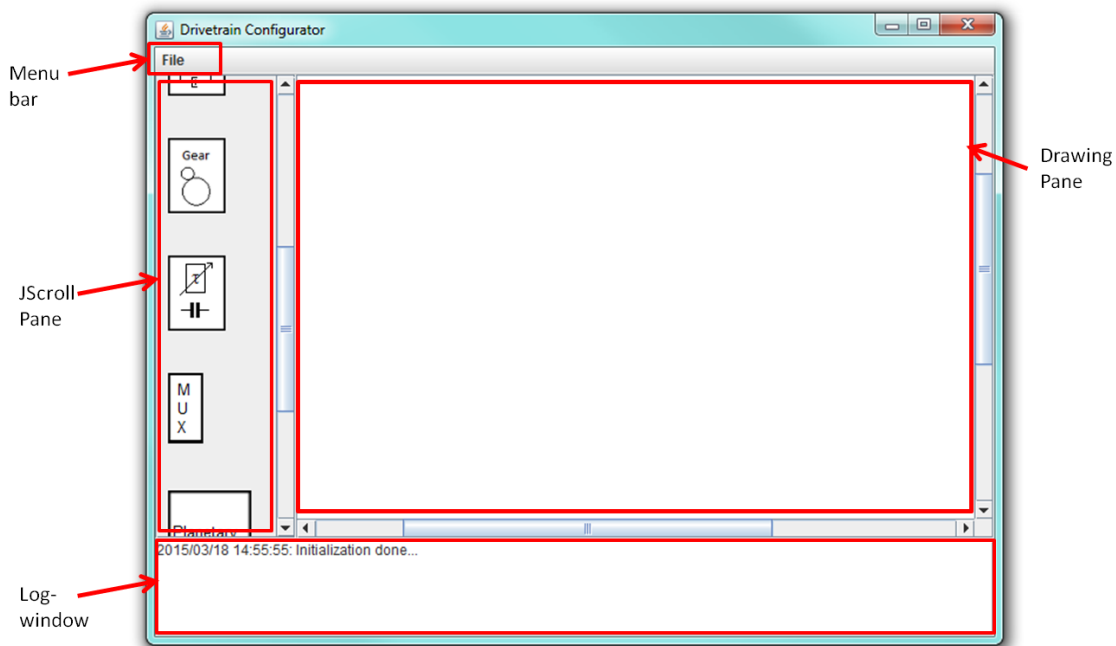


Figure 4.1: Illustration of the developed GUI for drivetrain design

¹ Eclipse Standard/SDK, Version: Luna Release (4.4.0), <https://eclipse.org/>

² JGoodies Java Libraries, <http://www.jgoodies.com/downloads/libraries/>

First there is the drawing pane, which extends the class JPanel. This is the main component for the drivetrain model design. On the left side is a JScrollPane with the available components which can be added to the drawing pane by double-click. The menu bar is at the top of the GUI and gives the user the possibility to save and load configurations, to delete all the components on the drawing pane and to create an initialization file for MATLAB. Lastly there is a log-window at the bottom of the GUI.

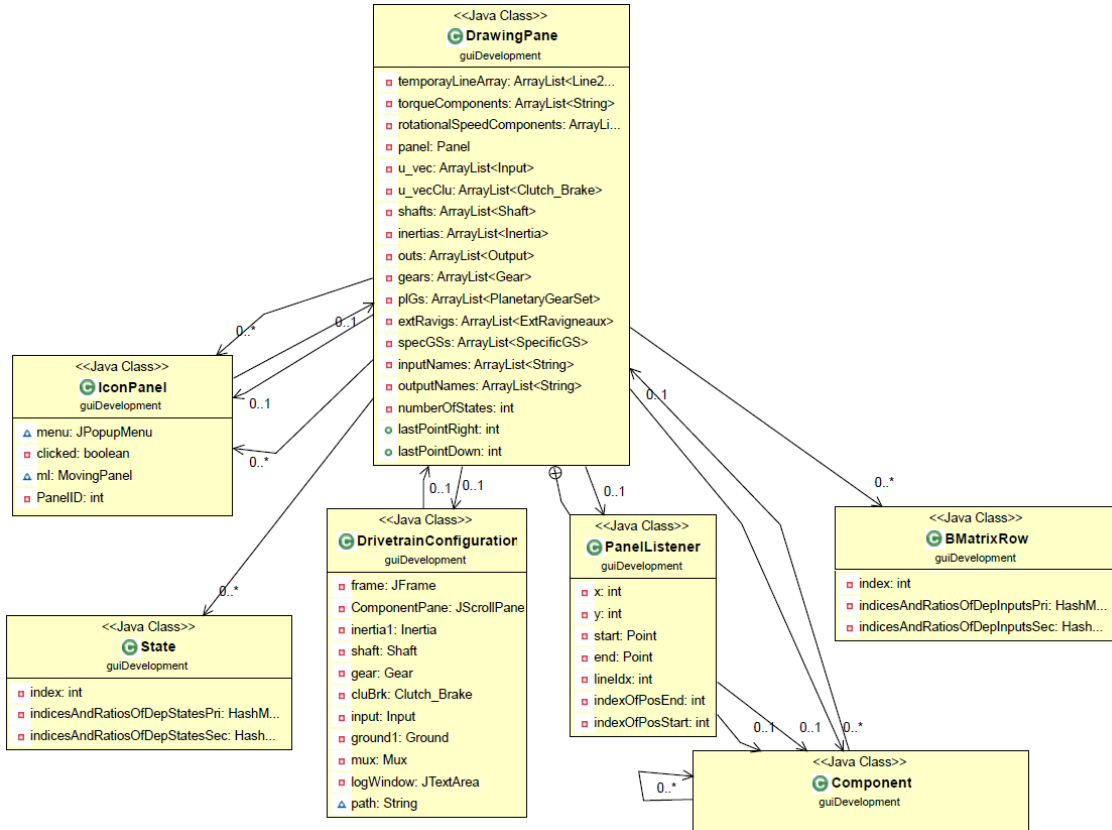


Figure 4.2: Class diagram of the solution without specified component and configuration classes.

In fig. 4.2 the class diagram of the main classes without the inherited classes for the specific components are shown. It can be seen that the most important class is the DrawingPane which is the main component for the schematic drivetrain design. To allow different number of inputs and input positions IconPanel was implemented. IconPanel combines the icons of the components with the correct position of the their inputs. This class holds the class MovingPanel which is responsible for *Drag and Drop* functionality of each component. To draw the connections between the components the class PanelListener was implemented. The check if the connection between two components is valid is also accomplished within this class. Fig. 4.3 shows the inheritance class diagram for the specific components. For the sake of clarity, the methods have been omitted in both class diagrams.

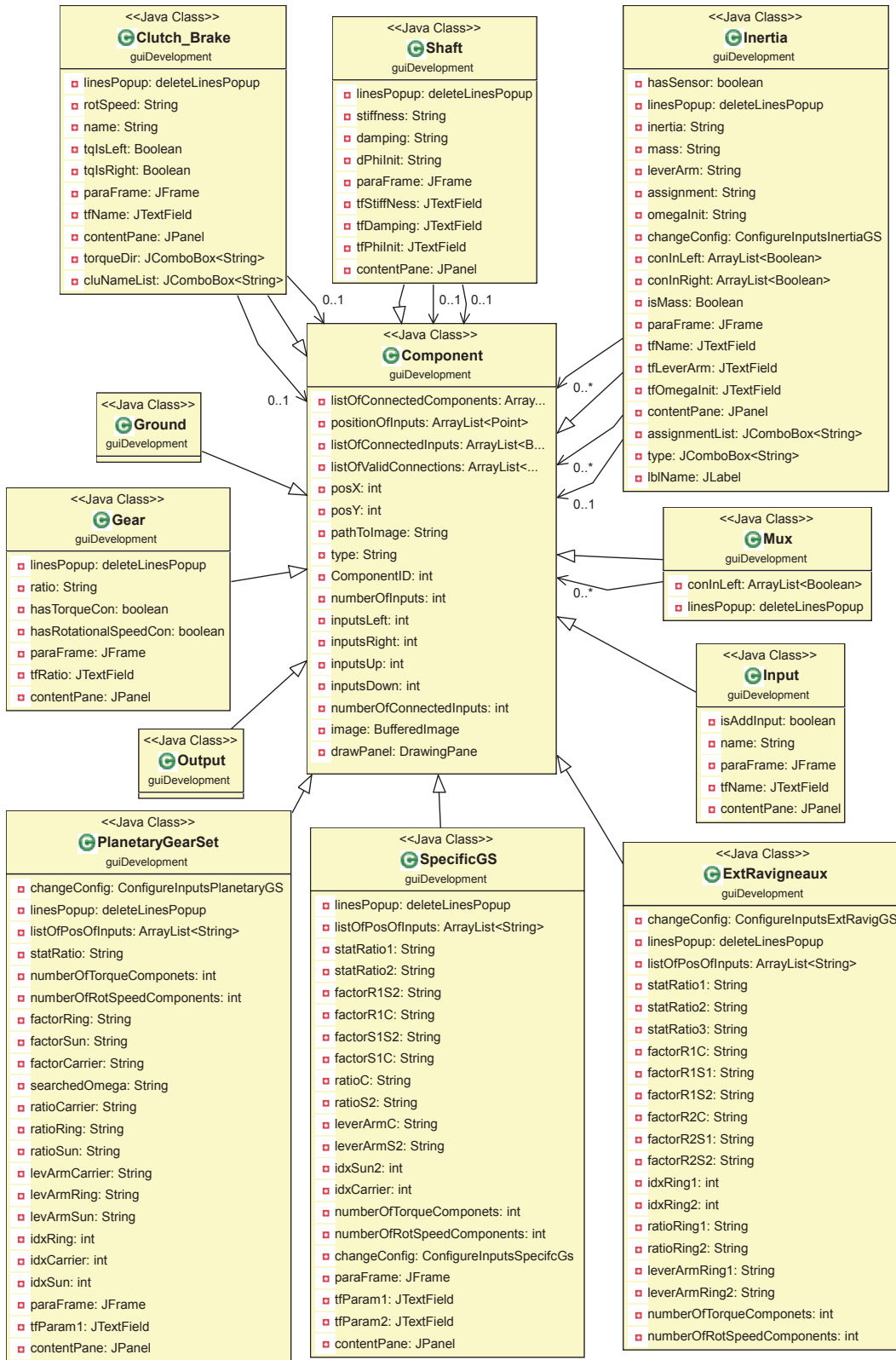


Figure 4.3: Inheritance diagram of the specific components including the parent class.

4.2 Components

The different components, which are described in detail below, all inherit some attributes and functions from the component class shown in fig. 4.4. Using variables for parameterization instead of explicit values is supported.

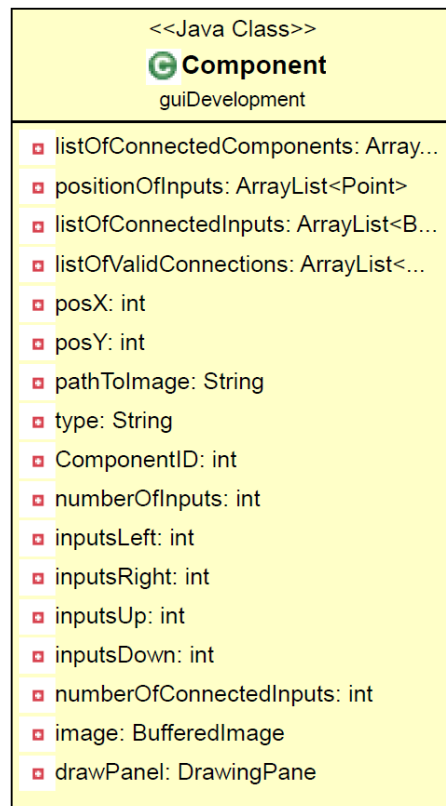


Figure 4.4: Parent class component with parameters.

4.2.1 External input

The external input component acts as interface and is only implemented to deliver a torque. It is possible to switch the direction of the input from right to left only for graphical representation. The external input is limited to be connected to a gear or an inertia. In fig. 4.5(a) and 4.5(b) the icon of the input component is shown as well as the configuration of the parameter name of the input.

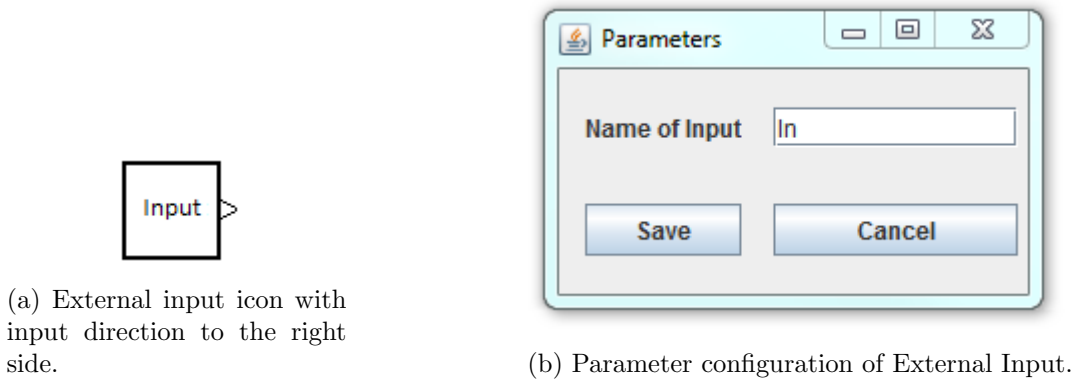


Figure 4.5: External input icon and parameter configuration.

4.2.2 Ground

There is also a ground component implemented which can be connected to a clutch or to one of the planetary gear sets. If it is connected to a clutch, the clutch will act as brake. If it is connected to a gear set, the connector of the gear set will be ignored in the generated formula.



Figure 4.6: Icon of the ground component.

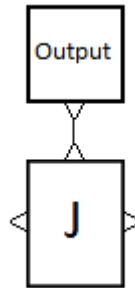
4.2.3 Inertia with output

For the realization of the vehicle, engine, etc. a component representing a lumped inertia of neighboring components is implemented. Fig. 4.7(a) depicts the icon for inertia and the configuration possibilities are shown in figs. 4.7(b) and 4.7(c).

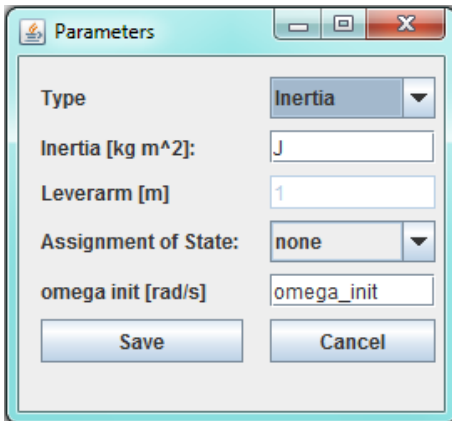
Additionally, the implementation allows to add an optional sensor to each inertia. Each inertia is represented by its motion state. As presented in fig. 4.7(a) the inertia has two inputs by default, though the user can choose how much inputs should be on each side of the inertia. There is the possibility to have 0 – 4 inputs on each side, however, there need to be at least two inputs in total. Each inertia may be assigned to one of the following standard locations within an automotive drivetrain:

- **none** is the default configuration and can be used for inertias which do not need to be specialized.
- **engine** represents the common combustion engine.
- **eMot** represents the electric engine.
- **TCimpeller** represents the impeller of the torque converter.

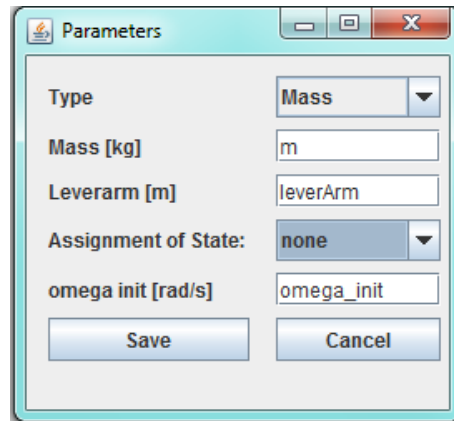
- **TCturbine** represents the turbine of the torque converter.
- **GearbOut** represents the position at the secondary side of the gearbox/transmission (i.e. between gearbox and final drive next to the gearbox).
- **WhlDrv** represents the driving wheels.
- **Vehicle**



(a) Inertia icon with an optional sensor on top.



(b) Parameter configuration of Inertia.



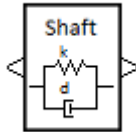
(c) Parameter configuration of mass.

Figure 4.7: Inertia icon and parameter configuration.

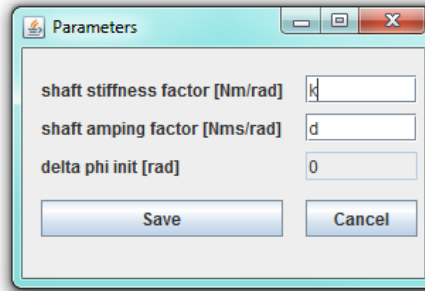
Additionally the user can configure the initial value of ω . Furthermore there is also the possibility to use the inertia as a mass. If it is a mass, the user has to configure the lever arm in meters and the mass in $[kg]$. If it is a inertia, the lever arm does not need to be configured. The inertia delivers rotational speed.

4.2.4 Shaft

This component serves to transmit torque and rotation to connect components of the drivetrain. It is possible to configure two parameters: stiffness and damping. As illustrated in fig. 4.8(a) the shaft has two inputs, one on each side.



(a) Shaft icon.

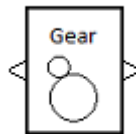


(b) Parameter configuration of shaft.

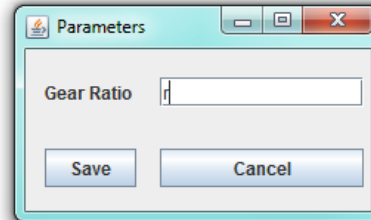
Figure 4.8: Shaft icon and parameter configuration.

4.2.5 Gear

The gear component represents torque speed ratios introduced by mechanical gears, multiplying torque by the specified ratio and speeds by its inverse. The user can configure the transmission ratio as shown in fig. 4.9(b). To support the user while creating a configuration and avoid possible errors, a gear needs to be connected to one torque giving and one rotational speed giving component.



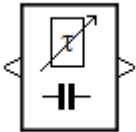
(a) Gear icon.



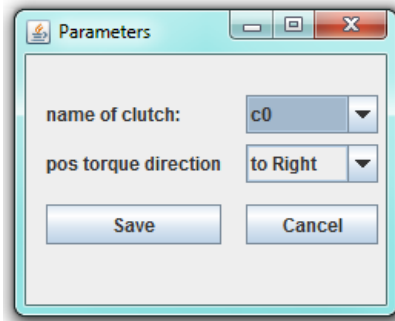
(b) Parameter configuration of gear.

Figure 4.9: Input icon and parameter configuration.

4.2.6 Clutch/Brake



(a) Clutch icon.



(b) Parameter configuration of clutch.

Figure 4.10: Clutch icon and parameter configuration.

This component represents either a clutch or a brake. If the clutch icon is connected to a ground it acts as a brake, otherwise it acts as clutch. Every clutch needs to be configured with a name which has to be chosen from $c_0 - c_7$. Furthermore the user can define the positive torque direction. The default configuration of the torque direction is from left to right.

4.2.7 Multiplexer (mux)

This component was implemented to connect an input of a planetary gear set to more than one clutch. This is needed in various transmission topologies, e.g conventional automatic transmission.

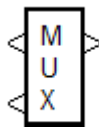


Figure 4.11: Multiplexer icon.

4.2.8 Planetary gear set

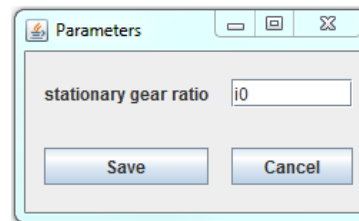
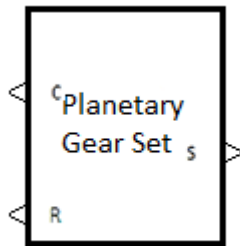
The first gear set which is explained now is the simple planetary gear set. The schematic representation is shown in fig. 3.5. Fig. 4.12(a) and fig. 4.12(b) illustrate the icon for the planetary gear set in the developed environment and the according parameter configurations. The three inputs are representing the three main components: ring, sun, and carrier. The user can configure the arrangement of the inputs and the stationary gear ratio. The user interface ensures, that the number of degrees of freedom and the number of connected inertias fit, which is two in this case. According to which connector is linked to a torque delivering component such as shaft or clutch, the relations between the gears are calculated. In tab. 4.1 the torque relations for each input are shown depending on

| inertia | inertia | ratio sun | ratio carrier | ratio ring |
|---------|---------|----------------------------|---------------------------------|-------------------------------|
| sun | carrier | $r_S^R = -\frac{1}{i_0}$ | $r_C^R = -\frac{(-1+i_0)}{i_0}$ | - |
| sun | ring | $r_S^C = \frac{1}{-1+i_0}$ | - | $r_R^C = -\frac{i_0}{-1+i_0}$ |
| ring | carrier | - | $r_C^S = -i_0$ | $r_C^R = (-1 + i_0)$ |

Table 4.1: Torque relations between inputs of a simple planetary gear set depending on connected inertias

which input is not connected to an inertia. The first two columns indicate which gears are connected to an inertia. These relations were calculated by a computational software by solving the kinematic constraint (WILLIS equation) and the following kinetic constraints.

- All outer torques have to be zero in total, $\Sigma\tau = \tau_1 + \tau_2 + \tau_3 = 0$.
- The sum of all power is zero, $\Sigma P = \omega_1\tau_1 + \omega_2\tau_2 + \omega_3\tau_3 = 0$



(a) Planetary gear set icon.

(b) Parameter configuration of planetary gear set.

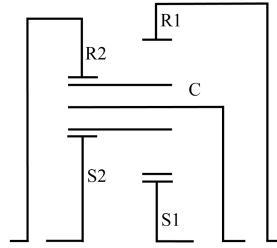
Figure 4.12: Planetary gear set icon and parameter configuration.

4.2.9 Extended Ravigneaux gear set

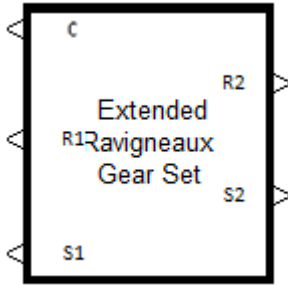
The second implemented gear set is the extended Ravigneaux gear set. Fig. 4.13(a) illustrates that this gear set consists of two suns, two rings, and one carrier. The user is able to configure three stationary gear ratios. Similarly to the simple planetary gear set, two inertias have to be connected to the gear set, since the number of degree of freedom has not changed. However, in this case the inertia cannot be connected to any input but only to a ring. This constraint was implemented because the relations between the inputs change, depending on which connectors are linked to inertias and would lead to an additional effort to solve symbolic equations for each newly implemented combined gear set. In tab. 4.2 the torque relations are shown. These relations have been calculated similar to the planetary gear set (see subsection 4.2.8).

| input | ring1 | ring2 |
|---------|--|---|
| carrier | $r_C^{R1} = -(i_{11} - i_{12})^{-1}i_{11}$ | $r_C^{R2} = (i_{11} - i_{12})^{-1}i_{12}$ |
| sun1 | $r_{S1}^{R1} = -(i_{11} - i_{12})^{-1}(i_{11} - i_{11}i_{12})$ | $r_{S1}^{R2} = (i_{11} - i_{12})^{-1}(i_{11} - i_{11}i_{12})$ |
| sun2 | $r_{S2}^{R1} = -(i_{11} - i_{12})^{-1}(i_{11} - i_{11}i_{22})$ | $r_{S2}^{R2} = (i_{11} - i_{12})^{-1}(i_{11} - i_{11}i_{22})$ |

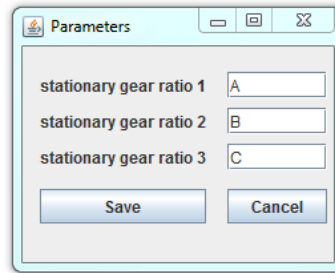
Table 4.2: Torque relations of inputs of an extended Ravigneaux gear set.



(a) Schematic representation of the extended Ravigneaux gear set.



(b) Extended Ravigneaux gear set icon.



(c) Parameter configuration of extended Ravigneaux gear set.

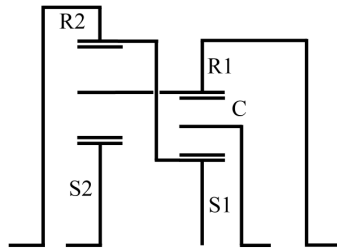
Figure 4.13: Extended Ravigneaux gear set icon and parameter configuration.

4.2.10 Specific gear set

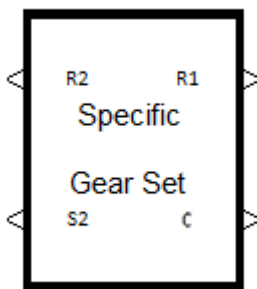
This specific gear set was implemented to configure the conventional automatic transmission shown in chapter 6. As illustrated in fig. 4.14(a) it has four shafts and consists of two planetary gear set. The ring of the first gear is coupled with the carrier of the second gear and the sun of the first gear is coupled with the ring of the second gear. The user can configure two stationary gear ratios for the creation of the initialization file. The torque-relations shown in tab. 4.3 are calculated similar to the planetary gear set (see subsection 4.2.8) as well.

| input | carrier | sun2 |
|-------|--|---|
| ring1 | $r_{R1}^C = -(1 - i_{22} + i_{11}i_{22})^{-1}(i_{11}i_{22} - i_{22})$ | $r_{R1}^{S2} = -(1 - i_{22} + i_{11}i_{22})^{-1}$ |
| ring2 | $r_{R2}^C = -(1 - i_{22} + i_{11}i_{22})^{-1}(i_{11}i_{22} - i_{22} - i_{11} + 1)$ | $r_{R2}^{S2} = -(1 - i_{22} + i_{11}i_{22})^{-1}i_{11}$ |

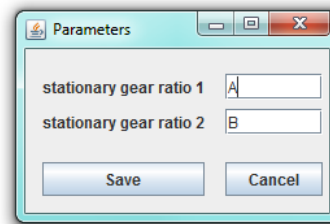
Table 4.3: Torque relations of inputs of an specific gear set.



(a) Schematic representation of the specific gear set.



(b) Extended Ravigneaux gear set icon.



(c) Parameter configuration of extended Ravigneaux gear set.

Figure 4.14: Extended Ravigneaux gear set icon and parameter configuration.

4.3 Save and load configuration

The tool allows to save the created configuration as xml-File. First of all, a list of the components used in the configuration is saved on top of the xml-File. This is necessary for loading the configuration. Afterwards the created function saves each component with its parameters and its connected components. In fig. 4.15 a xml-File of a simple example presented in fig. 5.1(a) in the following chapter with two inertias, two inputs and one shaft is depicted.

When loading a configuration a list of all components is created. Afterwards each component is parameterized according to the parameters saved in the xml-file and the connected components are saved.

Figure 4.15: XML-file of an example configuration

```

<?xml version="1.0" encoding="UTF-8"?>
-<Configuration>
  +<ComponentMap>
    -<Component id="0">
      <type>Input</type>
      +<ConnectedComponents>
      +<Parameters>
      +<Position>
    </Component>
    -<Component id="1">
      <type>Inertia</type>
      +<ConnectedComponents>
      +<ConnectedComponentsLeft>
      +<ConnectedComponentsRight>
      +<Parameters>
      +<Position>
    </Component>
    -<Component id="2">
      <type>Shaft</type>
      +<ConnectedComponents>
      +<Parameters>
      +<Position>
    </Component>
    -<Component id="3">
      <type>Inertia</type>
      +<ConnectedComponents>
      +<ConnectedComponentsLeft>
      +<ConnectedComponentsRight>
      +<Parameters>
      +<Position>
    </Component>
    -<Component id="4">
      <type>Input</type>
      +<ConnectedComponents>
      +<Parameters>
      +<Position>
    </Component>
  </ComponentMap>
</Configuration>

```


Chapter 5

Automated parameterization of the drivetrain design

The main task, besides providing a user-friendly front end, is to provide an automated parameterization of the modeling approach presented in [4] for the created drivetrain topology. For this purpose, the used components and their connections have to be analyzed. First of all, the components are sorted by type and saved in array lists. Afterwards the relations between the connectors of the gear sets are added according to the tabs. 4.1, 4.2 and 4.3. The main aim is to create a MATLAB initialization file which parameterizes the matrices of the following state-space model eqn. (5.1) (see also eqn. (3.12)).

$$\begin{aligned} \mathbf{M}\dot{\mathbf{x}} &= \bar{\mathbf{A}}\mathbf{x} + \bar{\mathbf{B}}\mathbf{u} \\ \mathbf{y} &= \mathbf{C}\mathbf{x} \end{aligned} \tag{5.1}$$

\mathbf{x} represents the states of the configuration which are defined by the states of motion of inertias and the position difference over the shafts of the configuration. The inputs and clutches represent the vector \mathbf{u} and \mathbf{M} is generated from the inertias respectively the masses. Furthermore, \mathbf{y} represents the outputs (sensors, observations) of the configuration. Matrix \mathbf{C} maps the sensors (outputs) to the states. This is realized by creating a matrix filled with zeros with the number of outputs as rows and the number of states as columns. For every state which has an output the complying entry is set to one.

The next step is to create the matrices $\bar{\mathbf{A}}$ and $\bar{\mathbf{B}}$. To generate these matrices two additional classes called *State* and *BMatrixRow* were implemented. The next step is to divide the matrices in primary and secondary matrices. $\bar{\mathbf{A}}_{\text{pri}}$ holds all torques and forces acting on the 'left' side of masses and $\bar{\mathbf{A}}_{\text{sec}}$ holds all torques and forces action on the 'right' side of masses. The same applies for $\bar{\mathbf{B}}_{\text{pri}}$ and $\bar{\mathbf{B}}_{\text{sec}}$. Every inertia is mapped to an instance of the *State* class and *BMatrixRow* class which hold lists with indices and factors for the primary and secondary matrices. The next step is to check which components are connected to the inertia on each side. Gears are skipped in the implementation but their gear ratio is saved for later purposes. Furthermore, the distinction between mass and inertia is not important for the strategy of determination of the dependencies. After the determination of the parameters of the system matrices the entries influenced by the mass are divided by the lever arm. Algorithm 1 shows the simplified procedure for the initialization file creation

although there is no distinction between primary and secondary side. This algorithm will be described in the following sections.

Algorithm 1 Algorithm for automated parameterization of system matrices

```

1: procedure CREATEINITFILE
2:   enumerate Inertias, Shafts
3:   initialize  $\bar{A}$ 
4:   enumerate Inputs, Clutches
5:   initialize  $\bar{B}$ 
6:   enumerate Outputs
7:   initialize C
8:   for 1 to NumberOfInertias do
9:     if Inertia.hasSensor() is true then
10:      C(idxOutput, idxInertia)=1
11:   for 1 to NumberOfConnectors do
12:     [Connection,Ratio]=calculateRatio(Connection)
13:     switch Connection do
14:       case Clutch
15:          $\bar{B}(\text{idxInertia}, \text{idxClutch}) = \pm \text{Ratio}$ 
16:       case Mux
17:          $\bar{B}(\text{idxInertia}, \text{idxClutch1}) = \pm \text{Ratio}$ 
18:          $\bar{B}(\text{idxInertia}, \text{idxClutch2}) = \pm \text{Ratio}$ 
19:       case Input
20:          $\bar{B}(\text{idxInertia}, \text{idxInput}) = \pm \text{Ratio}$ 
21:       case GearSet
22:         for 1 to NumberOfGearSetConnectors do
23:           [GearSetConnector,RatioGS]=calculateRatio(GearSetConnector)
24:           switch Connection do
25:             case Clutch
26:                $\bar{B}(\text{idxInertia}, \text{idxClutch}) = \pm \text{Ratio} \cdot \text{ratioGS} \cdot \text{relationGS}$ 
27:             case Mux
28:                $\bar{B}(\text{idxInertia}, \text{idxClutch1}) = \pm \text{Ratio} \cdot \text{ratioGS} \cdot \text{relationGS}$ 
29:                $\bar{B}(\text{idxInertia}, \text{idxClutch2}) = \pm \text{Ratio} \cdot \text{ratioGS} \cdot \text{relationGS}$ 
30:             case Shaft
31:               idxInertia2=findIdxOfSecondInertia()
32:                $\bar{A}(\text{idxInertia}, \text{idxInertia}) = -d \cdot \text{Ratio} \cdot \text{ratioGS} \cdot \text{relationGS}$ 
33:                $\bar{A}(\text{idxInertia}, \text{idxInertia2}) = d \cdot \text{Ratio} \cdot \text{ratioGS} \cdot \text{relationGS}$ 
34:                $\bar{A}(\text{idxInertia}, \text{idxSecInertiaGS}) = -d \cdot \text{Ratio} \cdot \text{ratioGS} \cdot \text{relationGS}$ 
35:                $\bar{A}(\text{idxInertia}, \text{idxShaft}) = -k \cdot \text{Ratio} \cdot \text{ratioGS} \cdot \text{relationGS}$ 
36:       case Shaft
37:         for 1 to NumberOfShaftConnectors do
38:           [ShaftConnection,RatioShaft] = calculateRatio( ShaftConnection)

```

```

39:     switch ShaftConnection do
40:         case Inertia
41:              $\bar{A}$ (idxInertia, idxInertia) = -d·Ratio·RatioShaft
42:              $\bar{A}$ (idxInertia, idxInertia2) = d·Ratio·RatioShaft
43:              $\bar{A}$ (idxInertia, idxShaft) = -k·Ratio·RatioShaft
44:         case GearSet
45:              $\bar{A}$ (idxInertia, idxInertia) = -d·Ratio·RatioShaft·relationGS
46:              $\bar{A}$ (idxInertia, idxInertia2) = d·Ratio·RatioShaft·relationGS
47:              $\bar{A}$ (idxInertia, idxSecInertiaGS) = -d·Ratio·RatioShaft·relationGS
48:              $\bar{A}$ (idxInertia, idxShaft) = k·Ratio·RatioShaft·relationGS
49:     for 1 to NumberOfShafts do
50:         [ShaftConnection,RatioShaft] = calculateRatio( ShaftConnection)
51:         switch ShaftConnection do
52:             case Inertia
53:                  $\bar{A}$ (idxShaft,idxInertia)=± RatioShaft
54:             case GearSet
55:                 for 1 to NumberOfGSConnectors do
56:                     [GearSetConnection,ratioGS] = calculateRatio( GearSetConnection)
57:                     if GearSetConnection is Inertia then
58:                          $\bar{A}$ (idxShaft, idxInertia)=± ratioShaft · ratioGS · relationGS
59:
60:
61:     procedure [CONNECTION, RATIO]= CALCULATERATIO(CONNECTION)
62:     if Connection is not Gear then
63:         ratio=1
64:     else if Connection is Gear And TorqueDirection is GearDirection then
65:         ratio=Gear.getRatio()
66:         Connection=Gear.getConnection()
67:     else
68:         ratio=Gear.getRatio()-1
69:         Connection=Gear.getConnection()
70:     return ratio,Connection

```

5.1 Input

If an external input is connected on the left side of the inertia an entry is added at the column according to the input and the row according to the inertia in $\bar{\mathbf{B}}_{\text{pri}}$. This entry has to be positive because the torques point in the same direction. If there is a gear in between the input and the inertia, the current factor is multiplied with the gear ratio.

If it is connected on the right side of the inertia an entry is added at the same index but in $\bar{\mathbf{B}}_{\text{sec}}$. Due to the fact, that the torques do not point in the same direction the sign of the entry is negative. Furthermore, if there is a gear in between, the current factor is divided by its gear ratio.

5.2 Clutch/Brake

Similarly to the input, the clutch component is also represented in the u-vector and, therefore, an entry in matrix $\bar{\mathbf{B}}$ has to be added. Since there is the possibility to choose the direction of the clutch torque the determination of the sign does not only depend on which side it is connected but also of the clutch torque direction. However, in general the sign of the entry is positive if the two torques point in the same direction and negative otherwise. Depending on whether the clutch is connected on the left or the right side of the inertia an entry is either added to $\bar{\mathbf{B}}_{\text{pri}}$ or $\bar{\mathbf{B}}_{\text{sec}}$.

5.3 Shaft

If the connected component is a shaft the further procedure adapts due to the influence of the shaft on the following component which can either be an inertia or one of the gear sets. For better understanding of the shaft a small example is presented in fig. 5.1(a). The configuration of the inertia on the left side of the shaft is shown in fig. 5.1(b) and the configuration of the other inertia is shown in fig. 5.1(c). The parameters of the shaft are defined with k for stiffness and d for damping.

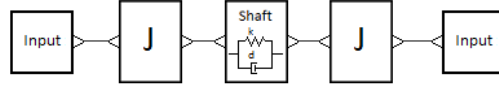
Granted that x_1 represents the motion state of J_1 , x_2 represents the motion state of J_2 and x_3 represents the position difference over the shaft, the following equations of motions can be expressed.

$$\begin{aligned} J_1 \dot{x}_1 &= -\tau_s + \tau_{in_1} \\ J_2 \dot{x}_2 &= \tau_s - \tau_{in_1} \\ \dot{x}_3 &= x_1 - x_2 \end{aligned} \tag{5.2}$$

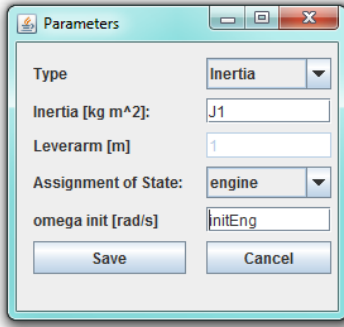
$$\tau_s = k\Delta\varphi_s + d\Delta\omega_s = kx_3 + d(x_1 - x_2) \tag{5.3}$$

Combining the equations in eqn. (5.2) and the equation eqn. (5.3) leads to eqn. (5.4).

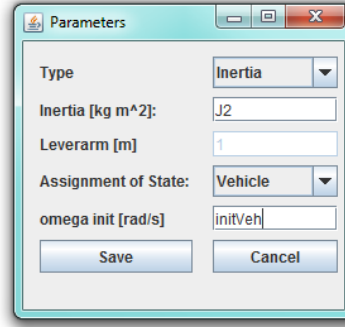
$$\begin{aligned} J_1 \dot{x}_1 &= -dx_1 + dx_2 - kx_3 + \tau_{in_1} \\ J_2 \dot{x}_2 &= dx_1 - dx_2 + kx_3 - \tau_{in_1} \\ \dot{x}_3 &= x_1 - x_2 \end{aligned} \tag{5.4}$$



(a) Block diagram of a simple example with shaft.



(b) Parameter configuration of the inertia left to the shaft.



(c) Parameter configuration of the inertia right to the shaft.

Figure 5.1: Exemplary drivetrain configuration containing a shaft.

Writing down these equations in matrix format gives the following matrices.

$$\begin{aligned}
 \mathbf{x} &= [\omega_1 \quad \omega_2 \quad \varphi_1 - \varphi_2] \\
 \mathbf{u} &= [\tau_{in_1} \quad \tau_{in_2}] \\
 \mathbf{M} &= \text{diag}([J_1 \quad J_2 \quad 1]) \\
 \bar{\mathbf{A}} &= \begin{bmatrix} -d & d & -k \\ d & -d & k \\ 1 & -1 & 0 \end{bmatrix} \\
 \bar{\mathbf{B}} &= \begin{bmatrix} 1 & 0 \\ 0 & -1 \\ 0 & 0 \end{bmatrix} \\
 \mathbf{C} &= 0
 \end{aligned} \tag{5.5}$$

For the creation of matrix $\bar{\mathbf{A}}$ the left inertia is considered first. The speed difference over the shaft (see eqn. (5.3)) is separated and is illustrated in the first two entries in the first row of $\bar{\mathbf{A}}$. The third entry in the first row is given through the torque over the shaft which is multiplied by the stiffness of it. Since x_1 represents the motion state of the left inertia and x_2 the motion state of the right inertia, the signs of the entries in the second row are inverse. According to the which side of the inertia the shaft is connected, these entries will be saved in $\bar{\mathbf{A}}_{Pri}$ or $\bar{\mathbf{A}}_{Sec}$. Now the first two rows of $\bar{\mathbf{A}}$ are complete. The last row represents the influences on the shaft which are 1 for the inertia on the left side and -1 for the inertia on the right side. The example in fig. 5.1a can be easily extended with gears. Depending on which side they are connected they will be divided or multiplied. In the algorithm 1 this is indicated with *ratio* and *ratioShaft*.

If the shaft is connected to a gear set the main configuration stays the same since the gear set can be seen as two additional inertias connected over gears. These gear ratios are

defined through the gear set and the arrangement of the inertias. In the algorithm 1 the ratio defined by the gear set is called *relationGS*. Furthermore, the shaft torque τ_s is now dependent on all three inertias in total. As a result, the two inertias connected to gear set are also dependent on each other.

The matrix \mathbf{C} is zero, since there are no sensors in the configuration. The determination of the entries of $\bar{\mathbf{B}}$ is already described in section 5.1.

5.4 Planetary gear set

If the inertia is directly connected to a planetary gear set the other components connected to the gear set have to be checked. In case that the next component is an inertia or a ground it can be skipped since it provides a rotational speed. There are three possible components providing torques which need to be treated differently.

1. **Shaft:** The determination of the dependencies is already described above. The only difference to the procedure above is that the considered inertia is not directly connected to the shaft but linked over the gear set.
2. **Clutch/Brake:** As mentioned above each clutch is represented in vector \mathbf{u} and therefore the dependencies have to be saved in $\bar{\mathbf{B}}$. A planetary gear set in between an inertia and a clutch can be seen as gear with a certain predefined gear ratio. It was assumed that all torques acting on the planetary gear sets point to the gear set. Furthermore, the clutch torque direction is configurable. In addition, the internal calculation bases on the assumption that all torques acting on a inertia point to it. As a result, all torque directions are defined and the determination of the sign of the dependency is simple. Depending on which side the inertia is linked to the planetary gear set the dependency is either saved in $\bar{\mathbf{B}}_{\text{pri}}$ or $\bar{\mathbf{B}}_{\text{sec}}$.
3. **Mux:** Considering that mux was implemented to combine two clutches the determination of the dependencies is similar to the procedure above. It can be treated as two independent clutches linked to one input of the gear set.

5.5 Extended Ravigneaux gear set

In case that the next component is a extended Ravigneaux gear set, the procedure of the determination of the dependencies is very similar to the planetary gear set described earlier. The only difference is that there are three inputs which are not connected to inertias. As a result, an inertia linked to an extended Ravigneaux gear set has influences of all other connected components. Depending on whether the considered connected component is a shaft, clutch or mux the dependencies are saved in $\bar{\mathbf{A}}$ or $\bar{\mathbf{B}}$.

5.6 Specific gear set

Similar to the extended Ravigneaux gear set, the specific gear set also uses the same procedure to determine the dependencies as the planetary gear set. Here, two components

are connected to the gear set which are not inertias and therefore have an influence on the connected inertias.

5.7 Short summary

With the developed algorithm it is possible parameterize different drivetrain configurations automatically. The algorithm 1 illustrates the main procedure for the systematic interpretation of the designed drivetrain. In the following chapter the automated parameterization will be validated with the model presented in [4].

Chapter 6

Tool validation

The automated parameterization is demonstrated for three different topologies. Therefore, the system matrices of eqn. (5.1) are shown for each topology. An overview over the used components is given in tab. 6.1.

6.1 Conventional automatic transmission

This drivetrain topology comprises a seven-clutch transmission. Fig. 6.1 illustrates the schema of this conventional automatic transmission, showing the different torque directions and configurations. In fig. 6.2 the drivetrain topology designed with the implemented development is presented. The inertia J_v in fig. 6.1 was configured as vehicle mass M_v with the lever arm r_w representing the wheel radius.

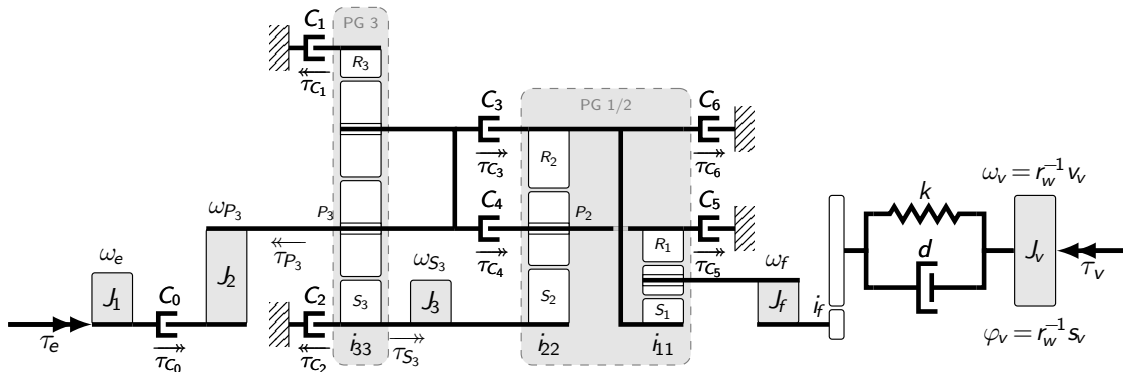


Figure 6.1: Diagram of a conventional automatic transmission which comprises a seven-clutch transmission with a specific combined gear set (PG 1/2) and a planetary gear set (PG 3). The seven friction elements are four brakes (c_1, c_2, c_5, c_6) and three clutches, including the separation clutch c_0 .




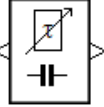
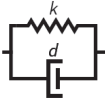
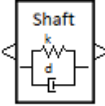



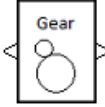

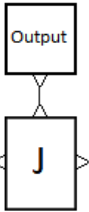
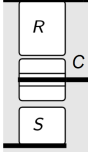
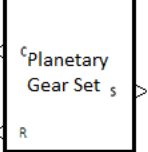
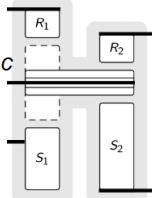
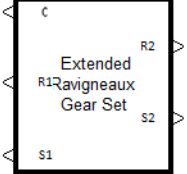
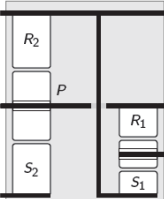
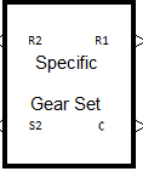
| Diagram | Implementation | Description |
|---|---|------------------------------|
|  |  | torque input |
|  |  | clutch |
|  |  | elastic shaft |
|  |  | ground |
|  |  | gear |
|  |  | motion of inertia |
|  |  | planetary gear set |
|  |  | extended Ravigneaux gear set |
|  |  | extended Ravigneaux gear set |

Table 6.1: List of all used components with their used icon in the diagram and the implementations.

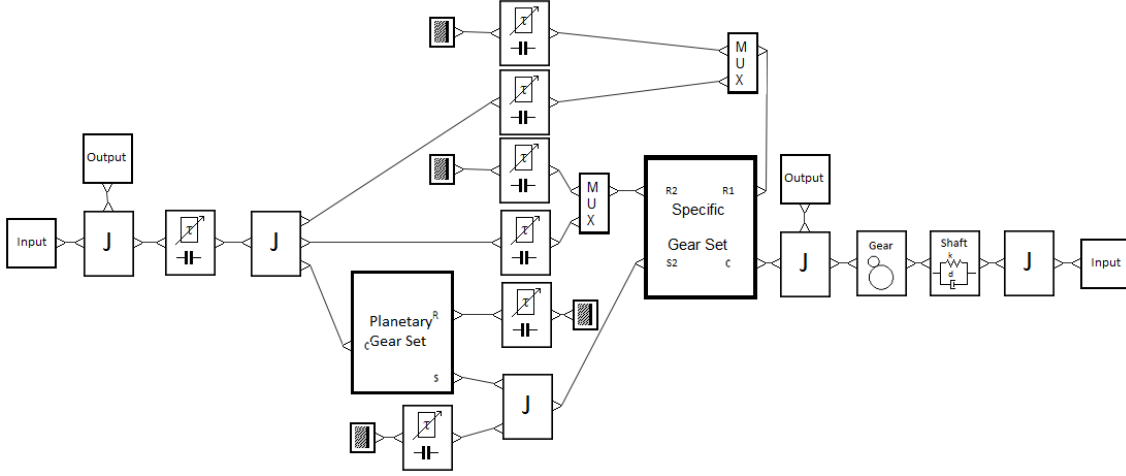


Figure 6.2: Configuration of the conventional automatic transmission given in fig. 6.1 in the implemented environment.

In this configuration a specific and a simple planetary gear set is used. The fifth state is a velocity and not an angular velocity and $\mathbf{M}(5, 5)$ is a mass and not an inertia. Therefore, the velocity and has to be divided by the lever arm r_w which is indicated in in the following matrices. The torque relations of the planetary gear sets have been determined with the kinematic and kinetic equations as mentioned in subsection 4.2.8. There are only inertias and clutches connected to the gear sets. Hence, their relations are only considered in $\bar{\mathbf{B}}$. The whole configuration file is shown in appendix A.3. The factors $r_{R_1}^C, r_{R_1}^{S_2}, r_{R_2}^{S_2}$ and $r_{R_2}^C$ can be found in tab. 4.3.

$$\begin{aligned}
 \mathbf{x} &= [\omega_e \quad \omega_{P_3} \quad \omega_{S_3} \quad \omega_f \quad v_v \quad i_f \varphi_f - \varphi_v]^T \\
 \mathbf{u} &= [\tau_e \quad \tau_v \quad \tau_{c_0} \quad \tau_{c_1} \quad \tau_{c_2} \quad \tau_{c_3} \quad \tau_{c_4} \quad \tau_{c_5} \quad \tau_{c_6}]^T \\
 \mathbf{M} &= \begin{bmatrix} J_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & J_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & J_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & J_f & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & M_v & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \\
 \bar{\mathbf{A}} &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -i_f^{-2}d - d_g & -i_f^{-1}d r_w^{-1} & i_f^{-1}k \\ 0 & 0 & 0 & i_f^{-1}d r_w^{-1} & -d r_w^{-2} & k r_w^{-1} \\ 0 & 0 & 0 & i_f^{-1} & -r_w^{-1} & 0 \end{bmatrix}
 \end{aligned}$$

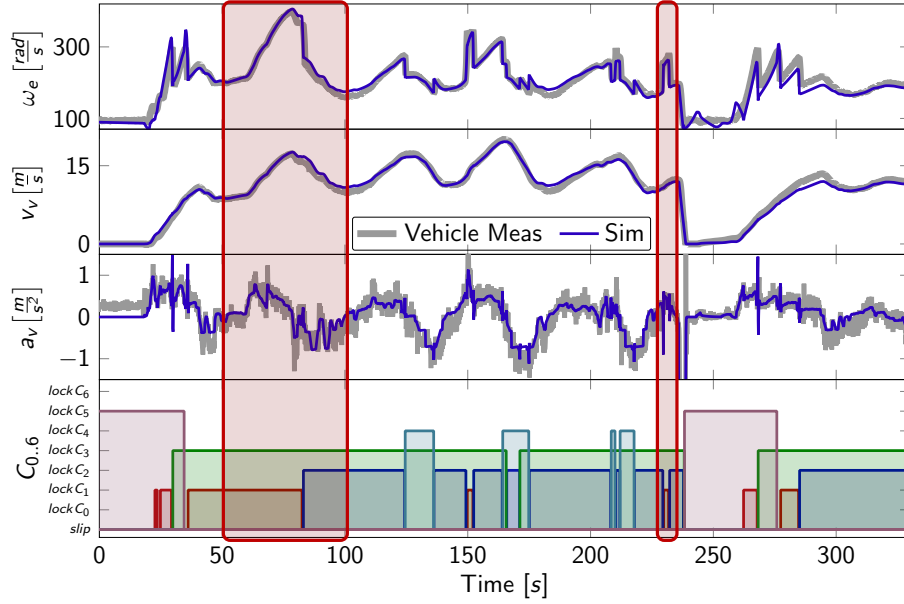


Figure 6.3: Vehicle measurement data of a driving cycle on a test track is used for validation of the modeling solution. The diagram illustrates the rotational speed of the engine ω_e , the vehicle velocity v_v and acceleration a_v as well as the clutch states $c_{0..6}$. For the sake of clarity, the different clutch states are shown in different colors. As can be seen, the clutch c_6 is never activated during the driving cycle, since it is responsible for reverse driving. For a better interpretation of the measured and simulated data the red sections in this figure are shown in more detail in fig. 6.4.

$$\mathbf{B} = \begin{bmatrix} 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -(1 + i_{33})i_{33}^{-1} & 0 & -1 & -1 & 0 & 0 \\ 0 & 0 & 0 & -i_{33}^{-1} & -1 & -r_{R_2}^{S_2} & -r_{R_1}^{S_2} & r_{R_1}^{S_2} & r_{R_2}^{S_2} \\ 0 & 0 & 0 & 0 & 0 & -r_{R_2}^{P_1} & -r_{R_1}^{P_1} & r_{R_1}^{P_1} & r_{R_2}^{P_1} \\ 0 & -r_w^{-1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

For this configuration vehicle measurement data of a driving cycle on a test track was available. Therefore, an offline simulation of this driving cycle was performed and compared to the vehicle measurement data. The comparison is illustrated in fig. 6.3 and reveals that the simulation model represents well the behavior of the overall vehicle including different maneuvers and gear shift sequences. The deviation of the simulation can be justified as model parameters were directly taken from mechanical design instead of fitted to measurement. Fig. 6.4 shows a more detailed few of to sections indicated in fig. 6.3. In the left section transient driving with only one shift in a 50s time period shows that simulation and measurement fit well. However, the shifting procedure in the right section reveals the limitations of the model. The modeled acceleration signal is not precise enough to allow a representative driveability evaluation. For a more precise model, the effort for parameterization of other model components, such as clutch hydraulics or the combustion engine, needs to be increased. The model parameters used for the simulation are summarized in tab. 6.2.

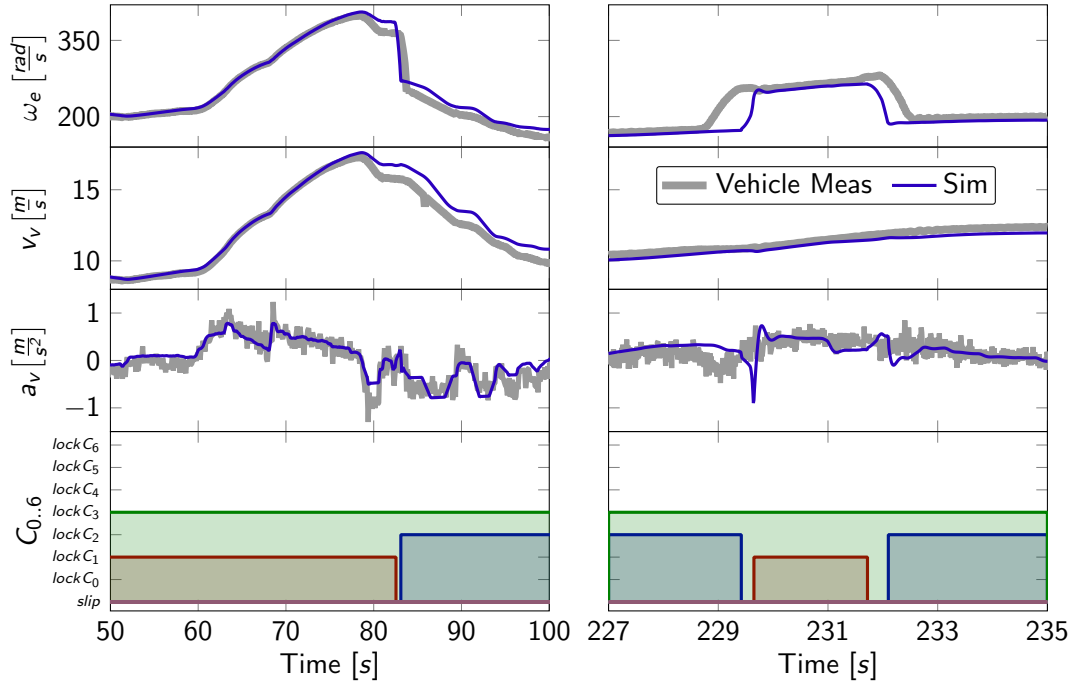


Figure 6.4: Detailed comparison of measurement data and simulation. A transient driving maneuver is shown in the left section with only one gear shift event at $\approx 82s$. The section on the right side illustrates a more dynamic shifting procedure, since there is an downwards and upwards shifting in a small time period of $\approx 5s$ representing a standard acceleration for an overtaking maneuver.

| Var | Parameter | Value | Unit |
|----------|---|--------|-------------|
| J_1 | Inertia engine, C_0 primary | 0.098 | $[kg\ m^2]$ |
| J_2 | Inertia C_0 secondary | 0.0124 | $[kg\ m^2]$ |
| J_3 | Inertia | 0.023 | $[kg\ m^2]$ |
| J_f | Lumped inertia gearbox secondary, side shafts | 1.32 | $[kg\ m^2]$ |
| M_v | Mass of vehicle | 1380 | $[kg\ m]$ |
| k | Shaft stiffness gearbox secondary to vehicle | 4000 | $[Nm/rad]$ |
| d | Shaft damping gearbox secondary to vehicle | 200 | $[Nms/rad]$ |
| i_{11} | Stationary gear ratio between S_1 and R_1 | -1.64 | |
| i_{22} | Stationary gear ratio between S_2 and R_2 | -2.08 | |
| i_{33} | Stationary gear ratio between S_3 and R_3 | 2.19 | |
| i_f | Gear ratio final drive | 3.61 | |
| r_w | Dynamic wheel radius | 0.28 | $[m]$ |

Table 6.2: Model parameters for simulation of driving cycle.

6.2 Exemplary dual clutch transmission

Another drivetrain topology tested is a DCT. The schematic of this drivetrain topology is depicted in fig. 6.5 again showing the torque directions and configurations. The dual-clutch transmission implemented with the developed environment is shown in fig. 6.6

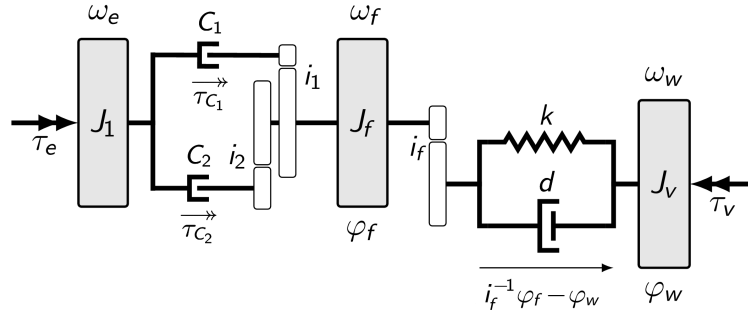


Figure 6.5: Diagram of a demo dual clutch transmission with one gear for each layshaft.

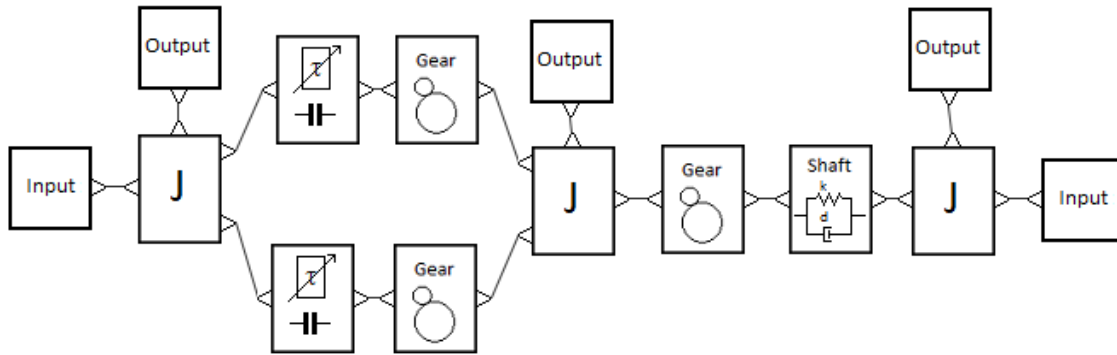


Figure 6.6: Design of the demo dual clutch transmission given in fig. 6.5 with the implemented environment.

The algorithm generated the following matrices. The shaft is connected to J_f over the gear i_f and to J_v . The relations between these components are in $\bar{\mathbf{A}}$. The entries in $\bar{\mathbf{B}}$ lead back to the clutches and inputs in fig. 6.6. The configuration file is shown in appendix A.1.

$$\begin{aligned}
 \mathbf{x} &= [\omega_e \quad \omega_f \quad \omega_w \quad i_f^{-1}\varphi_f - \varphi_w]^T \\
 \mathbf{u} &= [\tau_e \quad \tau_v \quad \tau_{c_1} \quad \tau_{c_2}]^T \\
 \mathbf{M} &= \begin{bmatrix} J_1 & 0 & 0 & 0 \\ 0 & J_f & 0 & 0 \\ 0 & 0 & J_v & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
 \bar{\mathbf{A}} &= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & -i_f^{-2}d & -i_f^{-1}d & i_f^{-1}k \\ 0 & i_f^{-1}d & -d & k \\ 0 & i_f^{-1} & -1 & 0 \end{bmatrix} \\
 \bar{\mathbf{B}} &= \begin{bmatrix} 1 & 0 & -1 & -1 \\ 0 & 0 & i_1 & i_2 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}
 \end{aligned}$$

6.3 Complex hybrid-electric automatic transmission

The last configuration is a automatic hybrid-electric drivetrain taken from [4]. Fig. 6.7 shows the schematic of this future hybrid transmission to gain clarity of names and torque directions. In fig. 6.8 the topology drawn with the developed tool is presented. The inertia J_v is configured as the vehicle mass M_v with the lever arm r_w representing the wheel radius. This example demonstrates the applicability of this solution to complex drivetrains.

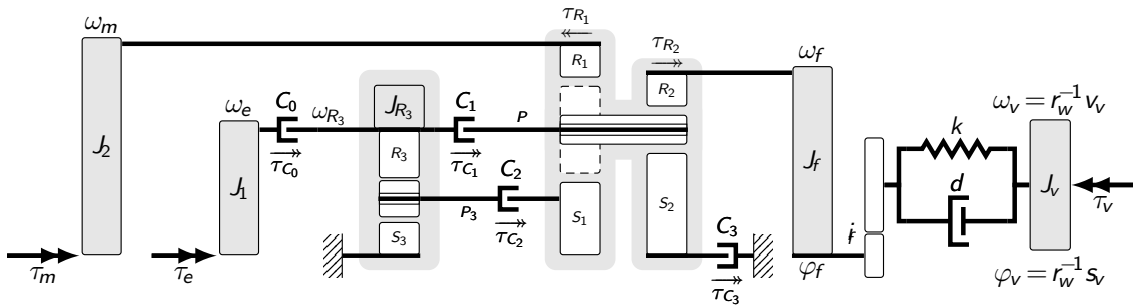


Figure 6.7: Diagram of a hybrid-electric transmission with an extended Ravigneaux and a planetary gear set. The four friction elements are one brake c_3 and three clutches, including one separation clutch c_0 . These allow a conventional, a power-split and a fully electric operation mode.

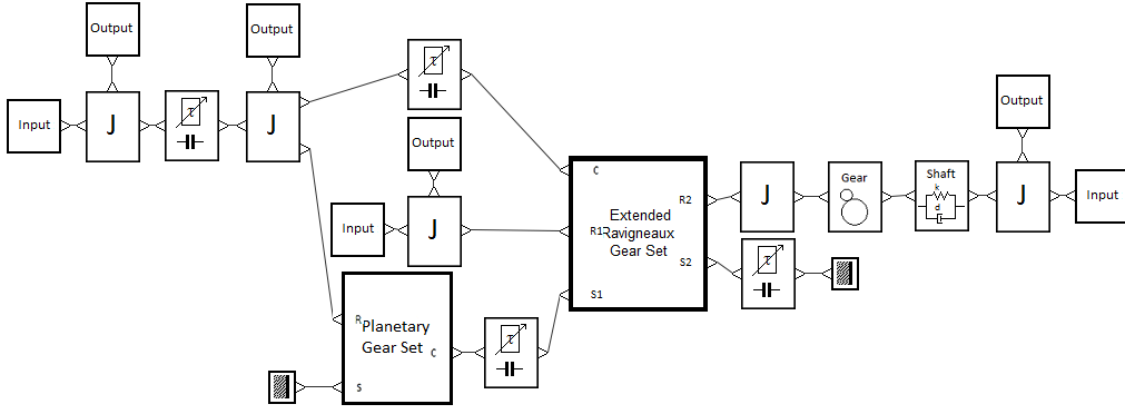


Figure 6.8: Design of the hybrid-electric transmission given in fig. 6.7 with the developed environment.

This configuration consists of an extended Ravigneaux and a simple planetary gear set. Similar to the conventional automatic transmission in section 6.1 the fifth state is a velocity and not an angular velocity and $\mathbf{M}(5, 5)$ is a mass and not an inertia. Therefore, the velocity and has to be divided by the lever arm r_w . The torque relations of the planetary gear sets have been determined with the kinematic and kinetic equations already mentioned above. These relations are used in $\bar{\mathbf{B}}$, since both gear sets are only connected to clutches, inertias and ground elements. The relations are presented in tab. 4.2. The initialization file containing these matrices is also shown in appendix A.2.

With the help of the automated parameterization of the tool, sign errors of the manual parameterization have been detected.

$$\mathbf{x} = [\omega_e \quad \omega_m \quad \omega_{R3} \quad \omega_f \quad v_w \quad i_f \varphi_f - \varphi_w]^T$$

$$\mathbf{u} = [\tau_e \quad \tau_m \quad \tau_v \quad \tau_{c0} \quad \tau_{c1} \quad \tau_{c2} \quad \tau_{c3}]^T$$

$$\mathbf{M} = \begin{bmatrix} J_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & J_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & J_{R3} & 0 & 0 & 0 \\ 0 & 0 & 0 & J_f & 0 & 0 \\ 0 & 0 & 0 & 0 & M_v & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\bar{\mathbf{A}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -i_f^{-2}d & -i_f^{-1}dr_w^{-1} & i_f^{-1}k \\ 0 & 0 & 0 & i_f^{-1}dr_w^{-1} & -dr_w^{-2} & kr_w^{-1} \\ 0 & 0 & 0 & i_f^{-1} & -r_w^{-1} & 0 \end{bmatrix}$$

$$\bar{\mathbf{B}} = \begin{bmatrix} 0 & 1 & 0 & -1 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & -r_C^{R1} & -r_{S1}^{R1} & r_{S2}^{R1} \\ 0 & 0 & 0 & 1 & -1 & -i_{33}(-1 + i_{33})^{-1} & 0 \\ 0 & 0 & 0 & 0 & -r_C^{R2} & -r_{S1}^{R2} & r_{S2}^{R2} \\ 0 & 0 & -r_w^{-1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

6.4 Summary

This chapter showed that the developed tool is useful for schematic design and parameterization of different commonly used drivetrain topologies such as dual-clutch transmission, transmissions for hybrid-electric vehicles and conventional automatic transmissions. The automated parameterization has matched the manual configuration for each given example. It has been verified with a real-life driving cycle.

Chapter 7

Conclusion and Outlook

7.1 Conclusion

In this Master's thesis a novel tool for schematic drivetrain design and its automated parameterization was developed. For this purpose, a graphical user interface was implemented, providing relevant mechanical components needed for the drivetrain configuration. These components can be arranged and connected via *Drag and Drop*. To simplify the drivetrain design for the user, some restrictions concerning the connections between components are treated automatically as well. The drivetrain design can be saved and loaded for further developments. For the automated parameterization an algorithm, which interprets the designed drivetrain, has been developed. The most challenging topics, while developing the algorithm, have been to find a way to generalize the sign determination of the clutch torques and the implementation of the different advanced gear sets.

The efficiency of the approach basing on the developed tool has been successfully demonstrated for several drivetrain topologies such as dual clutch transmission, conventional automatic transmission, and a transmission for a hybrid-electric vehicle. The parameterizations of these topologies have been validated with the results presented in [4]. Finally, an automated parameterization of a conventional automatic transmission has been validated, using measurement data of a real-life driving cycle.

Existing approaches for generic drivetrain modeling base on a model with full mechanical degree of freedom (all friction elements slipping) switch over to reduced models in the case of locked clutches. Building up model-internal generic algorithms requires expert know-how in mechanical engineering. The developed tool for automated parameterization enables non-experts to design drivetrain topologies, to derive the model parameterization for these and to reduce possible sources of errors usually occurring during manual modeling.

7.2 Outlook

Since drivetrain topologies can hold many different combined gear sets, a more general implementation of complex gear sets would be beneficial. Therefore, a coupling of the

tool with a computer algebra software to solve the kinetic and kinematic equations of any specific gear set would increase flexibility. Furthermore, the handling of depended parameters of the flexible shaft (damping and stiffness) and the integration of rotational speeds as interface are aims of future work. Moreover, the implementation of an improved torque converter will be subject to further investigations, possibly basing on the model description in [22].

Appendix A

MATLAB initialization files

The following table shows corresponding symbols used in the following configuration files and in the parameterization results in chapter 6.

| | |
|-------|--------------------|
| x_vec | \mathbf{x} |
| u_vec | \mathbf{u} |
| Aq | $\bar{\mathbf{A}}$ |
| Bq | $\bar{\mathbf{B}}$ |
| r_w | r_w |

In the following sections the three configuration functions for the drivetrain topologies presented in 6 are given. The function *dctConfig* is the configuration function of the dual-clutch transmission. *fhConfig* is the initialization function for the electric-hybrid transmission and *conventionalConfig* represents the initialization function for the conventional automatic transmission.

A.1 DCT configuration

Contents

- Create Mass-Matrix (M)
- Create A-Matrix (Aq)
- Create B-Matrix (Bq)
- Create C-Matrix (C)
- Create state initialization vector (x0)

```
function [ParNT, Pconfig] = dctConfig( P )

% x_vec = [Om_engine, Om_GearbOut, Om_Vehicle, PhiShaft [P.k, P.d]]
% u_vec=[tau_engine, tau_Vehicle, tau_c1, tau_c2]
% number of states
nrX = 4;
% Number of inputs (external, excluding clutches)
nrU = 2;
```

```
% number of clutches
nrC = 2;
% number of outputs
nrY = 3;
```

Create Mass-Matrix (M)

```
M = diag([P.J1, P.Jf, P.Jv, 1]);
```

Create A-Matrix (Aq)

```
% Torques/forces acting on primary ('left') side of masses
AqPri = zeros(nrX, nrX);
AqPri(3,2) = P.if^-1 * P.d;
AqPri(3,3) = -1^2 * P.d;
AqPri(3,4) = P.k;
% Torques/forces acting on secondary ('right') side of masses and
torques/forces acting on shafts
AqSec = zeros(nrX, nrX);
AqSec(2,2) = -P.if^-2 * P.d;
AqSec(2,3) = P.if^-1 * P.d;
AqSec(2,4) = -P.if^-1 * P.k;
AqSec(4,2) = P.if^-1;
AqSec(4,3) = -1;
Aq = AqPri + AqSec;
```

Create B-Matrix (Bq)

Torques/forces acting on primary ('left') side of masses

```
BqPri=zeros(nrX,nrU+nrC);
BqPri(1,1) = 1;
BqPri(2,3) = P.i1;
BqPri(2,4) = P.i2;
% Torques/forces acting on secondary ('right') side of masses
BqSec=zeros(nrX,nrU+nrC);
BqSec(1,3) = -1;
BqSec(1,4) = -1;
BqSec(3,2) = -1;
Bq = BqPri + BqSec;
```

Create C-Matrix (C)

```
y -> sensor values
```

```

C = zeros(nrY,nrX);
C(1,1) = 1; % engine
C(2,2) = 1; % GearbOut
C(3,3) = 1; % Vehicle

```

Create state initialization vector (x0)

```

x0 = zeros(nrX,1);
x0(1) = P.Dt_NEngInit * pi/30;
x0(2) = P.Dt_VVehInit;
x0(3) = P.Dt_VVehInit;
x0(4) = 0;

% Indices of components in states 'x' in order:
% (Engine, E-Motor, TC-impeller, TC-turbine, GearbOut, WhlDrv, Vehicle)
% if component is not available, use index -> 0
ParNT.idxXorig = [ 1, 0, 0, 0, 2, 0, 3]';

% Indices of components in external inputs 'u' in order:
% (Engine, E-Motor, TC-impeller, TC-turbine, WhlDrv, Vehicle)
% if component is not available, use index -> 0
ParNT.idxU = [1, 0, 0, 0, 0, 2]';

Pconfig.nrX = nrX;
Pconfig.nrY = nrY;
Pconfig.nrU = nrU;
Pconfig.nrC = nrC;
Pconfig.M = M;
Pconfig.AqPri = AqPri;
Pconfig.AqSec = AqSec;
Pconfig.Aq = Aq;
Pconfig.BqPri = BqPri;
Pconfig.BqSec = BqSec;
Pconfig.Bq = Bq;
Pconfig.C = C;
Pconfig.idxUN = idxUN;
Pconfig.Dv = Dv;
Pconfig.x0 = x0;

```

end

A.2 Future hybrid configuration

Contents

- Create Mass-Matrix (M)
- Create A-Matrix (Aq)

- Create B-Matrix (Bq)
- Create C-Matrix (C)
- Create state initialization vector (x0)

```
function [ParNT, Pconfig] = fhConifg( P )

% x_vec = [Om_engine, Om_eMot, Om_none, Om_GearbOut, V_Vehicle,
  PhiShaft [P.k, P.d]]
% u_vec=[tau_eMot, tau_engine, tau_Vehicle, tau_c0, tau_c1,
  tau_c2, tau_c3]
% number of states
nrX = 6;
% Number of inputs (external, excluding clutches)
nrU = 3;
% number of clutches
nrC = 4;
% number of outputs
nrY = 4;
```

Create Mass-Matrix (M)

```
M = diag([P.J_1, P.J_2, P.J_R3, P.J_f, P.M_v, 1]);
```

Create A-Matrix (Aq)

```
% Torques/forces acting on primary ('left') side of masses
AqPri = zeros(nrX, nrX);
AqPri(5,4) = P.i_f^-1 * P.d * P.r_w^-1;
AqPri(5,5) = -P.d * P.r_w^-2;
AqPri(5,6) = P.k * P.r_w^-1;
% Torques/forces acting on secondary ('right') side of masses and
torques/forces acting on shafts
AqSec = zeros(nrX, nrX);
AqSec(4,4) = -P.i_f^-2 * P.d;
AqSec(4,5) = P.i_f^-1 * P.d * P.r_w^-1;
AqSec(4,6) = -P.i_f^-1 * P.k;
AqSec(6,4) = P.i_f^-1;
AqSec(6,5) = - P.r_w^-1;
Aq = AqPri + AqSec;
```

Create B-Matrix (Bq)

Torques/forces acting on primary ('left') side of masses

```
BqPri=zeros(nrX,nrU+nrC);
BqPri(1,2) = 1;
BqPri(2,1) = 1;
```



```

BqPri(3,4) = 1;
BqPri(4,5) = - ((P.i_11 - P.i_12)^-1 * P.i_12);
BqPri(4,6) = - ((P.i_11 - P.i_12)^-1 * (P.i_12 - P.i_11*P.i_12));
BqPri(4,7) = ((P.i_11 - P.i_12)^-1 * (P.i_12 - P.i_11*P.i_22));
% Torques/forces acting on secondary ('right') side of masses
BqSec=zeros(nrX,nrU+nrC);
BqSec(1,4) = -1;
BqSec(2,5) = - (-(P.i_11 - P.i_12)^-1 * P.i_11);
BqSec(2,6) = - (-(P.i_11 - P.i_12)^-1 * (P.i_11 - P.i_11*P.i_12));
BqSec(2,7) = (-(P.i_11 - P.i_12)^-1 * (P.i_11 - P.i_11*P.i_22));
BqSec(3,5) = -1;
BqSec(3,6) = (-P.i33/(-1+P.i33));
BqSec(5,3) = - P.r_w^-1;
Bq = BqPri + BqSec;

```

Create C-Matrix (C)

y -> sensor values

```

C = zeros(nrY,nrX);
C(1,1) = 1; % engine
C(3,2) = 1; % eMot
C(2,3) = 1; % none
C(4,5) = 1; % Vehicle

```

Create state initialization vector (x0)

```

x0 = zeros(nrX,1);
x0(1) = P.Dt_NEngInit * pi/30;
x0(2) = P.Dt_NMotInit * pi/30;
x0(3) = 0;
x0(4) = P.Dt_VVehInit / 3.6;
x0(5) = P.Dt_VVehInit / 3.6;
x0(6) = 0;

```

```

% Indices of components in states 'x' in order:
% (Engine, E-Motor, TC-impeller, TC-turbine, GearbOut, WhlDrv, Vehicle)
% if component is not available, use index -> 0
ParNT.idxXorig = [ 1, 2, 0, 0, 4, 0, 5]';

```

```

% Indices of components in external inputs 'u' in order:
% (Engine, E-Motor, TC-impeller, TC-turbine, WhlDrv, Vehicle)
% if component is not available, use index -> 0
ParNT.idxU = [2, 1, 0, 0, 0, 3]';

```

```

Pconfig.nrX = nrX;

```

```

Pconfig.nrY = nrY;
Pconfig.nrU = nrU;
Pconfig.nrC = nrC;
Pconfig.M = M;
Pconfig.AqPri = AqPri;
Pconfig.AqSec = AqSec;
Pconfig.Aq = Aq;
Pconfig.BqPri = BqPri;
Pconfig.BqSec = BqSec;
Pconfig.Bq = Bq;
Pconfig.C = C;
Pconfig.idxUN = idxUN;
Pconfig.Dv = Dv;
Pconfig.x0 = x0;

```

```
end
```

A.3 Conventional drivetrain configuration

Contents

- Create Mass-Matrix (M)
- Create A-Matrix (Aq)
- Create B-Matrix (Bq)
- Create C-Matrix (C)
- Create state initialization vector (x0)

```

function [ParNT, Pconfig] = conventionalConfig( P )

% x_vec = [Om_engine, Om_none, Om_none, Om_GearbOut,
  V_Vehicle, PhiShaft [P.k, P.d]]
% u_vec=[tau_engine, tau_Vehicle, tau_c0, tau_c1,
tau_c2, tau_c3, tau_c4, tau_c5, tau_c6]
% number of states
nrX = 6;
% Number of inputs (external, excluding clutches)
nrU = 2;
% number of clutches
nrC = 7;
% number of outputs
nrY = 2;

```

Create Mass-Matrix (M)

```
M = diag([P.J_1, P.J_2, P.J_3, P.J_f, P.M_v, 1]);
```

Create A-Matrix (Aq)

```

% Torques/forces acting on primary ('left') side of masses
AqPri = zeros(nrX, nrX);
AqPri(5,4) = P.i_f^-1 * P.d * P.r_w^-1;
AqPri(5,5) = -P.d * P.r_w^-2;
AqPri(5,6) = P.k * P.r_w^-1;
% Torques/forces acting on secondary ('right') side of masses and
torques/forces acting on shafts
AqSec = zeros(nrX, nrX);
AqSec(4,4) = -P.i_f^-2 * P.d;
AqSec(4,5) = P.i_f^-1 * P.d * P.r_w^-1;
AqSec(4,6) = -P.i_f^-1 * P.k;
AqSec(6,4) = P.i_f^-1;
AqSec(6,5) = - P.r_w^-1;
Aq = AqPri + AqSec;

```

Create B-Matrix (Bq)

Torques/forces acting on primary ('left') side of masses

```

BqPri=zeros(nrX,nrU+nrC);
BqPri(1,1) = 1;
BqPri(2,3) = 1;
BqPri(3,4) = (-1/P.i_33);
BqPri(3,5) = -1;
BqPri(4,6) = - (1 -P.i_22 + P.i_11 * P.i_22)^-1 *
(P.i_11 * P.i_22 - P.i_22 - P.i_11 + 1);
BqPri(4,7) = - (1 -P.i_22 + P.i_11 * P.i_22)^-1 *
(P.i_11 * P.i_22 - P.i_22);
BqPri(4,8) = -(1 -P.i_22 + P.i_11 * P.i_22)^-1 *
(P.i_11 * P.i_22 - P.i_22);
BqPri(4,9) = -(1 -P.i_22 + P.i_11 * P.i_22)^-1 *
(P.i_11 * P.i_22 - P.i_22 - P.i_11 + 1);
% Torques/forces acting on secondary ('right') side of masses
BqSec=zeros(nrX,nrU+nrC);
BqSec(1,3) = -1;
BqSec(2,4) = (-(-1+P.i_33)/P.i_33);
BqSec(2,6) = -1;
BqSec(2,7) = -1;
BqSec(3,6) = -(1) * -(1 -P.i_22 + P.i_11 * P.i_22)^-1 * P.i_11;
BqSec(3,7) = -(1)^-1 * -(1 -P.i_22 + P.i_11 * P.i_22)^-1;
BqSec(3,8) = (1)^-1 * -(1 -P.i_22 + P.i_11 * P.i_22)^-1;
BqSec(3,9) = -(1 -P.i_22 + P.i_11 * P.i_22)^-1 * P.i_11;
BqSec(5,2) = - P.r_w^-1;
Bq = BqPri + BqSec;

```

Create C-Matrix (C)

y -> sensor values

```
C = zeros(nrY,nrX);
C(1,1) = 1;   % engine
C(2,4) = 1;   % GearbOut
```

Create state initialization vector (x0)

```
x0 = zeros(nrX,1);
x0(1) = P.Dt_NEngInit_P * pi/30;
x0(2) = 0;
x0(3) = 0;
x0(4) = P.Dt_VVehInit_P / 3.6;
x0(5) = P.Dt_VVehInit_P / 3.6;
x0(6) = 0;
```

```
% Indices of components in states 'x' in order:
% (Engine, E-Motor, TC-impeller, TC-turbine, GearbOut, WhlDrv, Vehicle)
% if component is not available, use index -> 0
ParNT.idxXorig = [ 1, 0, 0, 0, 4, 0, 5]';
```

```
% Indices of components in external inputs 'u' in order:
% (Engine, E-Motor, TC-impeller, TC-turbine, WhlDrv, Vehicle)
% if component is not available, use index -> 0
ParNT.idxU = [1, 0, 0, 0, 0, 0, 2]';
```

```
Pconfig.nrX   = nrX;
Pconfig.nrY   = nrY;
Pconfig.nrU   = nrU;
Pconfig.nrC   = nrC;
Pconfig.M     = M;
Pconfig.AqPri = AqPri;
Pconfig.AqSec = AqSec;
Pconfig.Aq    = Aq;
Pconfig.BqPri = BqPri;
Pconfig.BqSec = BqSec;
Pconfig.Bq    = Bq;
Pconfig.C     = C;
Pconfig.idxUN = idxUN;
Pconfig.Dv    = Dv;
Pconfig.x0    = x0;
```

end

Appendix B

Definitions

B.1 Abbreviations

| | |
|------------|-------------------------------|
| AMT | Automated manual transmission |
| AT | Automatic transmission |
| DCT | Dual-clutch transmission |
| ECU | Electronic control unit |
| GUI | Graphical user interface |
| ICE | Internal combustion engine |
| MT | Manual transmission |
| PGS | Planetary gear set |

B.2 Used symbols

| | | |
|----------------------------|-------------|-----------------------------|
| d | $[Nms/rad]$ | damping of spring element |
| i | - | gear ratio |
| J | $[kg\ m^2]$ | moment of inertia |
| k | $[Nm/rad]$ | stiffness of spring element |
| τ | $[Nm]$ | torque |
| ω | $[rad/s]$ | rotational speed |
| u | - | input |
| v | $[m/s]$ | velocity |
| x | - | state |
| y | - | output |

Bibliography

- [1] O. Niggemann, A. Geburzi, and J. Stroop, “Benefits of system simulation for automotive applications,” in *Model-Based Engineering of Embedded Real-Time Systems* (H. Giese, G. Karsai, E. Lee, B. Rumpe, and B. Schaetz, eds.), Springer, 2007.
- [2] D. W. Gao and A. Emadi, *Modeling and Simulation of Electric and Hybrid Vehicles*. IEEE, Proceedings of the IEEE, 2007.
- [3] T. Lennon, *Model-based design for mechatronics systems*. Natick, Mass.: The MATHWorks Inc., 2007. <http://machinedesign.com/archive/model-based-design-mechatronics-systems>.
- [4] M. Bachinger, M. Stolz, and M. Horn, “Fixed time-step drivetrain observer for embedded automotive applications,” in *Control Applications (CCA), 2014 IEEE Conference on*, pp. 47–52, Oct 2014.
- [5] J. Fuchs, “Dynamische modellierung eines elektrifizierten antriebsstranges,” Master’s thesis, Institut fuer Elektrische Messtechnik und Messsignalverarbeitung, Technische Universitaet Graz, September 2012.
- [6] O. M. 1.9.1, *Copyright Open Source Modelica Consortium (OSMC)*. <https://openmodelica.org/q>, Accessed on 01.12.2014, 2014.
- [7] MATLAB, *version 8.4.0.150421 (R2014b)*. Natick, Massachusetts: The MathWorks Inc., 2014.
- [8] L. Prechelt, “Comparing java vs c/c++ efficiency differences to inter-personal differences,” tech. rep., Fakultae fuer Informatik, Universitaet Karlsruhe, 1999.
- [9] M. K. Dalheimer, “Qt vs. java: A comparison of qt and java for large-scale, industrial-strength gui development,” *Klaraelvdalens Datakonsult AB*, -.
- [10] L. Prechelt, “An empirical comparison of c, c++, java, perl, python, rexx, and tcl,” *IEEE Computer*, 2000.
- [11] T. A. Budd, *An Introduction to Object-Oriented Programming*. Addison Wesley Longman Inc, 1998.
- [12] T. A. Budd, “Object-oriented programming,” in *Handbook of Programming Languages, Volume I* (P. H. Salus, ed.), Macmillan Technical Publishing, 1998.

- [13] P. M. Fishbane, S. Gasiorowicz, and S. Thornton, *Physics for Scientists and Engineers*. Prentice-Hall, 1996.
- [14] F. Pfeiffer, *Mechanical System Dynamics*. Lecture Notes in Applied and Computational Mechanics, Springer, 2008.
- [15] E. Kirchner, *Leistungsuebertragung in Fahrzeuggetrieben: Grundlagen der Auslegung, Entwicklung und Validierung von Fahrzeuggetrieben und deren Komponenten*. Springer: New York, Berlin, Heidelberg, 2007.
- [16] H. J. Foerster, *Automatische Fahrzeuggetriebe : Grundlagen, Bauformen, Eigenschaften, Besonderheiten*. Springer, 1991.
- [17] F. Kurth, *Efficiency Determination and Synthesis of Complex-Compound Planetary Gear Transmissions*. PhD thesis, Technische Universitaet Muenchen, 2012.
- [18] G. Lechner and H. Naunheimer, *Fahrzeuggetriebe: Grundlagen, Auswahl, Auslegung und Konstruktion*. Springer, 1994.
- [19] R. Fischer, F. Küçükay, and G. Jürgens, *Das Getriebebuch*. Der Fahrzeugantrieb, Springer Vienna, 2012.
- [20] “Dual clutch transmission.” http://en.wikipedia.org/wiki/Dual-clutch_transmission Accessed on: 16.03.2015.
- [21] M. Yolga, “Robust 3-element gear shift for electrified powertrains,” in *CTI Symposium Fahrzeuggetriebe, HEV- und EV-Antriebe*, 2014.
- [22] S. Z. et al., “Modeling and simulation of the automatic transmission assembly using matlab/simulink,” in *Applied Mechanics and Materials*, pp. 291–294, 2013.
- [23] K. R. (Hrsg.), *Konventioneller Antriebsstrang und Hybridantriebe*. Springer Fachmedien Wiesbaden, 2010.