TECHNISCHE
UNIVERSITÄT
WIEN

DIPLOMARBEIT

A Dynamic System Simulation Approach for Biomechanical Models

zur Erlangung des akademischen Grades

Diplom-Ingenieurin

im Rahmen des Studiums

Biomedical Engineering

eingereicht von

Ruth Leskovar

Matrikelnummer 0726512

ausgeführt am Institut für Analysis und Scientific Computing
der Fakultät für Mathematik und Geoinformation der TU Wien

Betreuer: Ao.Univ.Prof.i.R. Dipl.-Ing. Dr.techn. Felix Breitenecker
Mitwirkung: Dipl.-Ing Dipl.-Ing Dr.techn. Andreas Körner

Wien, 18. Dezember 2018

Unterschrift Studentin

Unterschrift Betreuer

Człowiek nigdy nie ogląda się na to, co zrobione, ale na to patrzy, co ma przed sobą do zrobienia.

One never notices what has been done, one can only see what remains to be done.

Marie Skłodowska Curie

Danksagungen

Das Verfassen einer wissenschaftlichen Arbeit erfordert viel Selbstständigkeit, aber ohne die Unterstützung von anderen, wären mir die folgenden Seiten um vieles schwerer gefallen. Daher möchte ich an dieser Stelle einigen Personen Dank aussprechen.

An erster Stelle möchte ich mich bei Andreas Körner bedanken, der mich inhaltlich begleitet hat. Er hat mit viel Geduld und Beharrlichkeit darauf geachtet, dass diese Arbeit einen roten Faden erhält und beibehält.
Ausgesprochen dankbar bin ich für die Förderung in der Arbeitsgruppe Modellbildung und Simulation an der TU Wien zu Beginn durch Felix Breitenecker und schließlich durch Andreas Körner. Die unterschiedlichen Facetten, an denen ich beteiligt war, von Lehre über Konferenzen bis hin zu sozialen Veranstaltungen, bereicherten meine vergangenen Jahre und erweiterten sowohl meine fachlichen als auch menschlichen Kompetenzen.
Allen meinen Kolleginnen und Kollegen, die sich mit mir über Erfolge gefreut haben und bei Rückschlägen mit Rat und Tat zur Seite standen, bin ich sehr dankbar. Besonders möchte ich Elisabeth, Franziska und Stefanie für die mühselige Arbeit des Korrekturlesens danken. Franziska gilt ein besonderer Dank für das Übernehmen meiner Pflichten an der TU in den letzten Wochen.
Meinen Eltern danke ich für Ihre Unterstützung während meines Studiums und für die wenigen unangenehmen Fragen nach dem Abschlussdatum. Auch meinen Geschwistern und ihren Familien bin ich sehr dankbar für die wunderbar chaotischen Treffen, die mich auf ganz andere Gedanken gebracht haben. Meinem Bruder Benedict möchte ich auch explizit für das Bekanntmachen mit LaTeXdanken.
Ich möchte mich auch bei meinen Freundinnen und Freunden bedanken, besonders bei Antonia, Lynette und Sebastian, die bis zuletzt auf meine zeitlichen Einschränkungen Rücksicht genommen haben und mich außerdem auch mental und inhaltlich unterstützt haben.
Schlussendlich gilt mein großer Dank Nikolaus. Nicht nur für die Sorge um das Notwendigste, Schlaf und Essen, sondern vor allem für das Teilen seiner Erfahrungen. Er hat mir vieles, was er im Laufe seiner wissenschaftlichen Arbeit gelernt hat, mitgegeben. Darüber hinaus hat auch er die Arbeit gewissenhaft Korrektur gelesen. Ihm danke ich ganz besonders, denn ist er mein wertvollster Gesprächspartner.

Muito Obrigada.

Zusammenfassung

In der Biomechanik werden mathematische Modelle unter anderem eingesetzt, um kinematische und kinetische Analysen durchzuführen. Daraus gewonnene Erkenntnisse werden beispielsweise verwendet, um die Form von Prothesen individuell anzupassen oder die Funktionalität von aktiven Prothesen zu verbessern. Der Entwurf einer modellbasierten Simulation erlaubt mehr Flexibilität in den Anwendungen der biomechanischen Modelle. Die unterschiedlichen Modellbildungsansätze, welche in der Biomechanik eingesetzt werden, gaben Anstoß diese als dynamische Systeme zu betrachten um anschließend in einen geschlossenen Simulationskreislauf einzubinden. Dies ist durch Einsatz der Systemtheorie möglich.
Diese Arbeit analysiert das dynamische Verhalten von biomechanischen Modellen für menschliche Gelenke. In der Biomechanik werden hauptsächlich zwei Modellierungsansätze verwendet. Einerseits basieren diese auf gewöhnlichen, andererseits auf partiellen Differentialgleichungen. Ausgehend von der Systemtheorie gilt es nun beide Modellbeschreibungen als dynamische Systeme darzustellen. Dies erfordert den Einsatz von unterschiedlichen Methoden.
Nach der Einführung in die beiden inhaltlichen Säulen der Arbeit, Modellbildung und Simulation sowie Systemtheorie, wird ein Modell vorgestellt, das die Flexion eines menschlichen Knies simuliert. Dieses Modell ist in drei Simulationsumgebungen implementiert, welche hinsichtlich ihrer Möglichkeiten, die sie für Simulationen bieten, beleuchtet werden. Zwei Simulationsmodelle basieren auf gewöhnlichen Differentialgleichungen, eines ist durch partielle Differentialgleichungen beschrieben. Ziel war es für beide Modellansätze eine ähnliche Beschreibungsform ihres dynamischen Verhaltens zu finden, was durch die Zustandsraumdarstellung möglich war. Somit konnten verschiedene geschlossene Simulationskreisläufe entworfen und das Verhalten der beiden Modellansätze in diesen untersucht werden.
Mehrkörpermodelle, welche auf gewöhnlichen Differentialgleichungen basieren, können direkt als dynamische Systeme angesehen werden. Bei Modellen, welche durch partielle Differentialgleichungen beschrieben werden, sind Einschränkungen notwendig, bevor sie als dynamische Systeme formuliert werden können.
Diese Arbeit zeigt Ansätze in der mathematischen Modellbildung und Simulation auf, welche es ermöglichen geschlossene Simulationskreisläufe für unterschiedliche mathematische Modellbeschreibungen zu entwerfen.

Abstract

Modelling and simulation is an important tool in the development and validation of new technologies in many research fields. In biomechanics mathematical models are used for example analysing kinematics and kinetics in the human body. These insights are used to improve prostheses in their usability and wearing comfort. The design of a feedback loop with biomechanical models as plant provides more flexibility in these applications. The system simulation approach allows to handle mathematical models as dynamic systems and to design feedback loops.

This thesis analyses biomechanical models for anatomic joints regarding their dynamics. Biomechanical models are based on two different modelling approaches mostly, ordinary and partial differential equations. Having these two different mathematical descriptions leads to the task of describing biomechanical models as dynamic systems in a simulation loop.

After an introduction on the basic principles of modelling and simulation as well as system theory, the structure of a mathematical model for the flexion of a human knee is presented. This model is implemented in three different simulation environments and after compared and benchmarked regarding simulation qualities. Two simulation models are multibody models, one is described by partial differential equations. Both mathematical descriptions are analysed with respect to their dynamics in order to describe their behaviour in similar forms, e.g. state space representation. Various control designs are investigated and compared regarding the different behaviour of the biomechanical models.

The usage of two different modelling approaches in the field of biomechanics was the incitement to investigate their different behaviour in a feedback loop. As multibody models are based on ordinary differential equations, they are dynamic systems and therefore it is easy to establish a simulation in a loop. Models based on partial differential equations what implies their dependence on time and space, respectively, require restrictions to get a description as dynamic system. In conclusion, it can be said that system simulation theory gives the possibility to examine different mathematical modelling descriptions on their behaviour. This allows to design closed simulation circuits which are suitable for both modelling approaches. Insights from this work can be extended to other research fields using modelling and simulation.

Contents

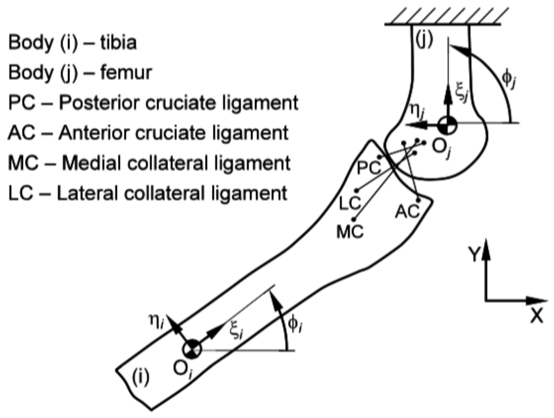
1. Introduction

Mathematical models are used in biomechanical research to describe and analyse the interactions in the human body. This comprises two main points, kinetics on the one hand and kinematics on the other hand. These two different research questions enable the application of different modelling approaches, multibody modelling and describing models by partial differential equations.
From a system theoretical point of view, the description as dynamic system for two different modelling approaches is an interesting challenge. This leads to the application of various modelling techniques and finally to the development of a simulation loop.
A short introduction to different applications of biomechanical models distinguished by their modelling approaches is given and followed by the description of the purpose of this thesis.

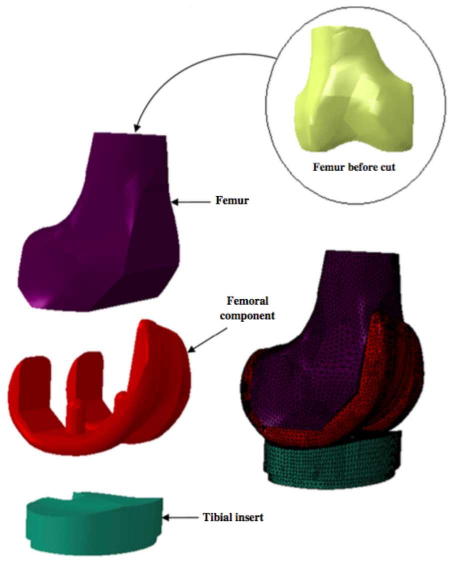
1.1. Overview of Biomechanical Models and their Application

The two modelling approaches used in biomechanics differ regarding their mathematical classification. Following that, as well their application fields differ, as it is stated more in detail in [21] and [20], two contributions which build the substantial base for the motivation of this work. This thesis focuses on the analysis of biomechanical models for anatomic joints. Due to their complex structure which is caused by the interaction of different components, as soft and rigid tissues, it is challenging to describe these systems mathematically.
Since multibody models describe relative motion between multiple bodies connected by joints, they analyse kinematics. This gives the possibility to use multibody models for the analysis of gross motion and interactions between a number of components. Mathematically, these models are represented by ordinary differential equations following the Lagrange formalism. Partial differential equations give the opportunity to analyse kinetics as they depend on time and as well on space. Furthermore, the analysis of local deformation can be investigated in detail. In biomechanics, they are applied for the investigation of adaptations in soft tissue under loads in the human body.
The development of multibody models for anatomic joints is still in focus of biomechanical research as it is done for the shoulder joint in [2] and the knee joint in [21]. Those joints show the most complex joint structures in the human body due to

their composition and high demands. Multibody models for anatomic joints help to analyse the interactions during motion between the ligaments and bones of the joint. Furthermore, the multibody modelling approach gives the possibility to analyse the entire human body during movement. This includes gait cycle analysis which gives insight to interactions in conventional conditions, but it can be extended to special situations. For example, the work of [31] analysed the interactions in the human body under various falling scenarios.
Although, the theory of multibody models can be extended to consider flexible bodies as it is explained in [13], adaptations in human structures which occur under applied loads, require a more detailed description. The adaptation of bone under load was investigated by [33]. The mathematical formulation for describing such problems can be accomplished by PDE models. A common application is the analysis of shape designs for prostheses in the human body as it is done in [17] for hip prostheses. Moreover, in recent times the development of materials used in prostheses are a challenging research question. The analysis of a functionally graded material for a femoral component in knee prostheses was established by [3] using finite element analysis.


Body (i) – tibia
Body (j) – femur
PC – Posterior cruciate ligament
AC – Anterior cruciate ligament
MC – Medial collateral ligament
LC – Lateral collateral ligament
(j)
ξ_j
ϕ_j
η_j
O_j
PC
LC
AC
MC
Y
X
η_i
ξ_i
ϕ_i
(i)
O_i

(a) Multibody model of the human knee developed in [21].



(b) Finite element analysis for knee replacement in [3].

Figure 1.1.: Different modelling approaches applied in biomechanics.

This leads to a common application of biomechanical models in the field of prosthetics. This research field shows multidisciplinary aspects, as medicine, biomechanics, biomaterials and electronics. Due to the fact that prostheses have to fulfil various tasks and the requirements are different, the development of new technologies represent a challenge. The requirements on lower limb prostheses can be summarised in three main points:

- Replacement of the static and dynamic function of the leg
- Adaptation to various activities
- Optical representation

The consideration of individual requirements is investigated in the work of [5] and [6]. This approach combines biomechanical analysis with psychological studies.

1.2. Scope of the Thesis

The incitement of this work is the application of two different modelling approaches in biomechanical research. This provides the challenge to describe both models as dynamic systems. The use of system theory gives the possibility to design simulation loops which gives flexibility in the application of biomechanical models. For example, the investigation of prostheses adapting to various activities, as walking on different grounds or running, can be accomplished using a feedback loop.
In the first step, it is necessary to deduce a biomechanical model for the human knee joint. The task is to analyse the behaviour of the simulation models regarding their different mathematical descriptions. Therefore, this conceptual model is implemented in two different simulation environments. The solution of the simulation model is calculated on the one hand using ODE solvers, on the other hand using FEM.
The focus of this work is the application of system theory to these simulation models. Various techniques should be applied in order to represent both simulation models as dynamic systems, e.g. using state space representation. The final aim is the combination of the simulation model and a feedback loop. This gives the possibility to simulate various purposes and scenarios, as for example the definition of a reference signal which determines the movement of the knee.

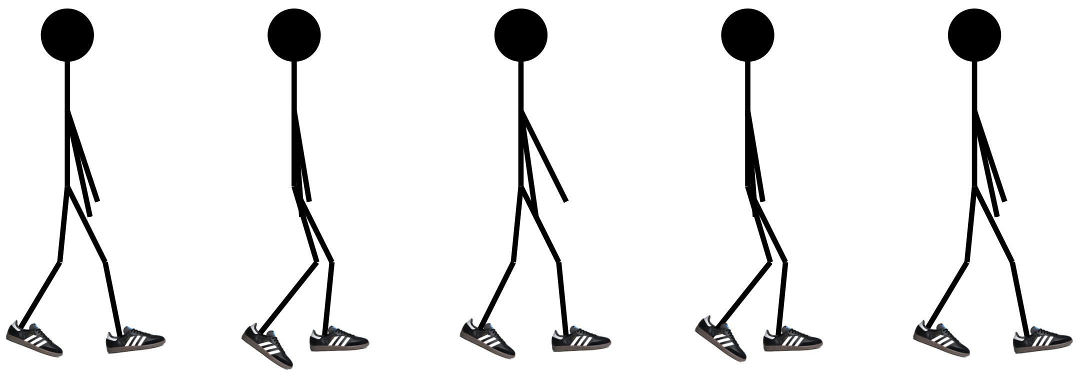


Figure 1.2.: Sketch of the human gait cycle.

2. Basics of Modelling and Simulation

This chapter gives an overview about basic methods in modelling and simulation. Various modelling approaches are discussed which are based on different mathematical theories. Furthermore, different possibilities of simulation are analysed. The field of modelling and simulations is wide due to the fact that it is used in many research fields. Hence, a selection of modelling approaches and corresponding simulation environments is given.

2.1. Modelling and Simulation Circle

As the application of modelling and simulation is widely used in many research fields, there exist various definitions, procedures and approaches which differ slightly. The presented methods are based on [36] and [10].
Before the modelling process is explained, some reasons applying simulation as problem-solving tool are given. The main reason to build a model, is to find solutions for a problem. This implies to define a system, where this problem is present. In engineering, simulation models are used instead of performing experiments which are either not feasible or too costly. Moreover, simulation models are applied in combination with experiments in order to extend their usage. For example, closed simulation loops give the possibility to suppress disturbances as measuring noise or to get access to all state variables which are not quantifiable in the real system.
The structure for the development of a new model has various focus in literature due to its different stages. The following description is mainly based on [26] and [4]. In the process of building a new model, three entities are considered. First, a system has to be defined, where the problem can be formulated, whose solution is the goal of the process. It is important to distinguish between two models which are built outgoing from the system. Different designations are used in literature. In this work, a distinction is made between the conceptual and the simulation model. The construction of a valid model requires interaction between the involved entities. Therefore, the modelling process is often compared to a circle as it is depicted in Figure 2.1.
Analysing the underlying system, performing experiments if possible, leads to a verbal description of the system which ideally can be extended to a mathematical representation of the system. The goal of this modelling phase is the conceptual or

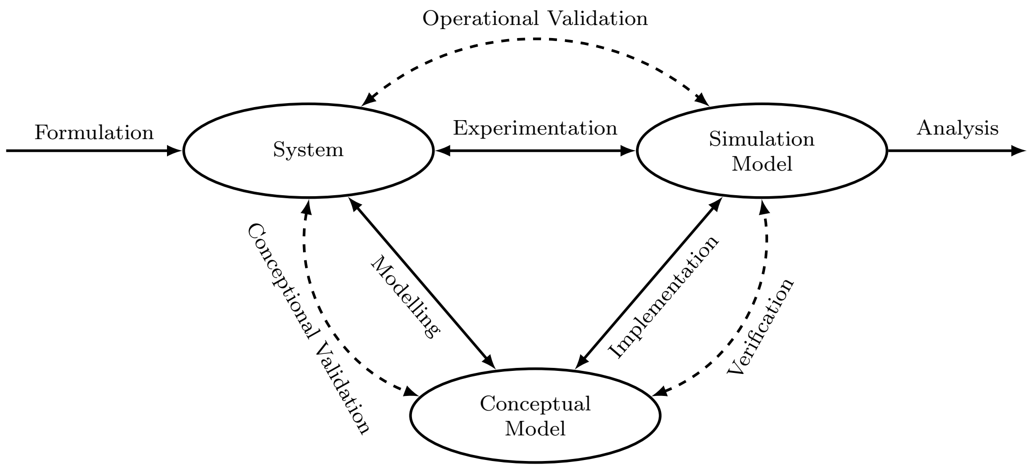


Figure 2.1.: Graphical illustration of the modelling and simulation circle.

mathematical model, respectively, which is the analytical base for further investigations.
Proceeding from the conceptual model, a simulation model is built. This comprises various possibilities of implementations which depend on the model structure but as well on the system and demands of applications.
The implementation process entails a verification between the mathematical model and its simulation model, a step dealing with models only. This ensures that the simulation model is implemented correctly and fulfils the structure of the conceptual model. This verifies, that the model is built right.
Obviously, the verification does not ensure that the conceptual model fulfils the dynamics of the system and undergoes true hypotheses. This is verified by a validation process which interacts between model and its underlying system. Since two models can be build, a distinction between two different validations has to be made. First, the conceptual model is validated against the system ensuring that the applied theories and assumptions are correct and reasonable. Second, the solution of the simulation model is analysed. This verifies if the simulation model is sufficient accurate and if the intended purpose of applications is reached. This stage describes the building process of the right model regarding the intended system's behaviour.
For validation, many techniques exist. Their application depends on the used simulation and model. For example, it is reasonable to use an animation of the simulator for validation. Some other models require more analytical tests, as statistical techniques. It is common to examine simulations under extreme conditions to verify the accuracy of the results.
The modelling process yields to a valid and verified model whos results can be used to find solutions of the underlying problem. Of course, this implies many experiments and analysis of the simulation results.

2.2. Modelling Methods

Since modelling is applied in various fields of research and engineering, there exist many methods to derive a conceptual model from an underlying system. A common distinction is made between the so-called white box and black box modelling approach. A graphical illustration is given in Figure 2.2, where two possible approaches are depicted to formulate a conceptual model.

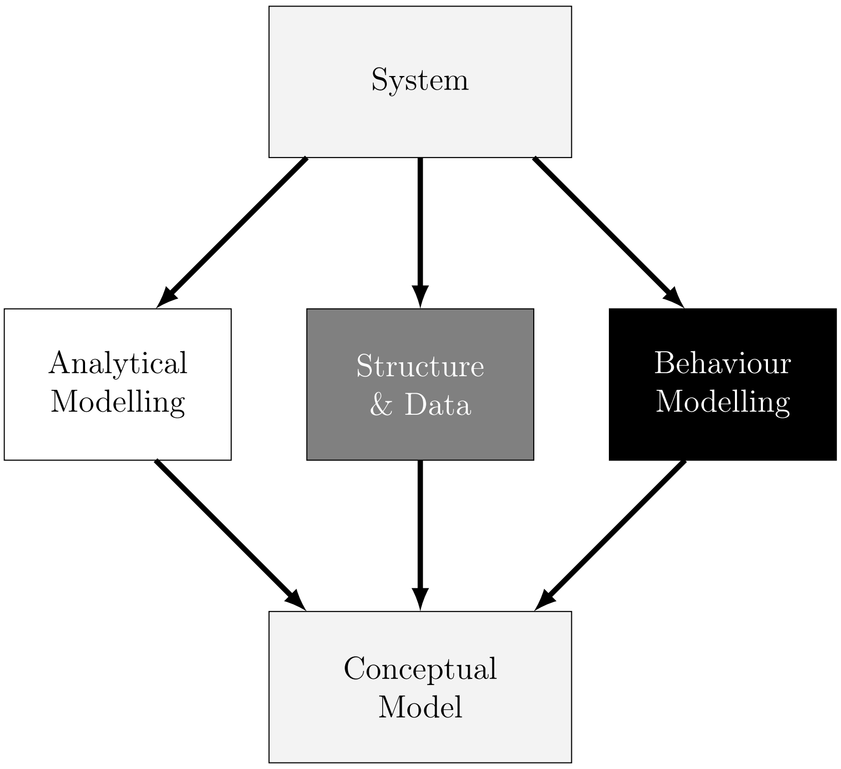


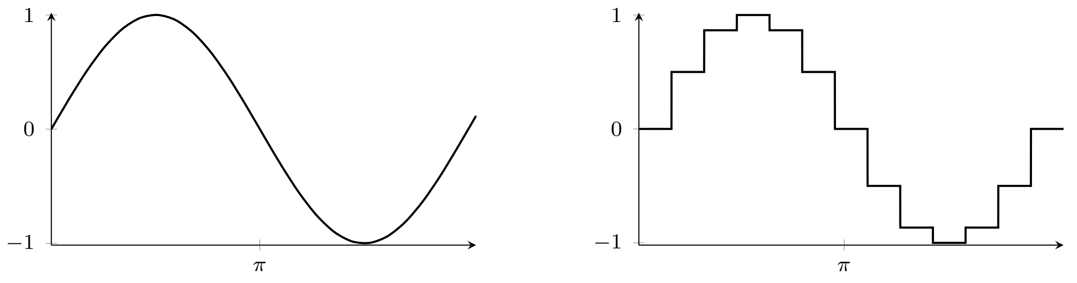
Figure 2.2.: Illustration of white, gray and black box modelling.

The white box modelling approach is based on mathematical analysis of the system. A mathematical model is formulated following already known laws and rules, such as physical, mechanical or electrical laws. This mathematical model is fitted to experimental measurement data, if available. In contrast to this, there is the black box modelling approach where a mathematical model description by observation and available data. The behaviour of the system is analysed and following this dynamic, an adequate model is formulated.
It depends on the underlying system which particular modelling approach fits best. Analysing a physical or engineering problem, allows to apply well known laws and rules. Observing systems where the behaviour can not be described by principles, the application of black box modelling is needed. Human behaviour is an example of processes which can not be described following some rules, so sociology, economy, etc. are examples for this.

A combination of both approaches lead to the grey box modelling approach, where a structure of the conceptual model is defined and the experimental data completes the formulation.

2.3. Types of Mathematical Models

A conceptual model can be described by various types of mathematical formulation. As it is written more in detail in [10], a distinction is made between continuous-time and discrete-time models. Trajectories from these two model types are illustrated in Figure 2.3.



(a) Continuous-time trajectory. (b) Discrete-time model trajectory.

Figure 2.3.: Trajectory behaviour of a dynamic model.

The behaviour of continuous-time models is described by a system of differential equations. Two classes are here again distinguished. One the one hand, systems, which are described by ordinary differential equations $\boldsymbol{f}\colon \mathbb{R}^n \times \mathbb{R}^l \times \mathbb{R}^p \times \mathbb{R} \to \mathbb{R}$,

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}, t),$$

where the change of the states $\boldsymbol{x} \in \mathbb{R}^n$ depends on the input $\boldsymbol{u} \in \mathbb{R}^l$, parameters $\boldsymbol{p} \in \mathbb{R}^p$ and time $t \in \mathbb{R}$. One the other hand, systems described by partial differential equations, as for example the heat equation

$$\frac{\partial u}{\partial t} = \alpha \left(\frac{\partial^2 u}{\partial x} + \frac{\partial^2 u}{\partial y} + \frac{\partial^2 u}{\partial z} \right),$$

for the change of temperatur $u = u(x, y, z)$ regarding the space coordinates x, y, z and time t.
For systems described by ODEs, a state space representation can be formulated as it is explained more in detail in chapter 3. This state space representation is finite dimensional and therefore, they are called lumped parameter models. The state space representation for systems described by PDEs in general is infinite dimensional due to the dependence on time and as well space. This is the reason why they are called

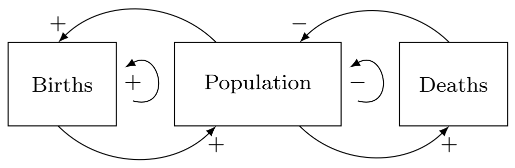
distributed parameter models.
Regarding discrete-time models, the solution of the system is given at fixed time steps. Their behaviour can be represented through difference equations for equidistant time steps,

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{p}_k, t_k),$$

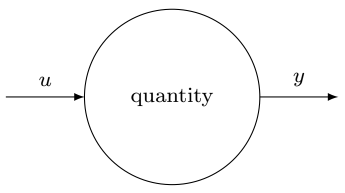
as counterpart to ODEs. Analogously, cellular automata allows a description distributed over space but discrete in time, the correspondence to PDEs. It is important to notice that ODE solvers and finite element analysis are calculating the solution for a continuous model by discretisation. This leads to a discrete representation of a continuous model.
For the sake of completeness, there exist two more classes of mathematical models, qualitative and discrete-event models.

2.4. Selected Overview of Modelling Approaches

The formulation of a mathematical description can be derived using different modelling approaches. Some systems allow the direct derivation of mathematical formulas by relying on more simple systems where the mathematical descriptions are already known or by combining already established mathematical model structures. More complex systems need further analysis before a mathematical description can be formulated.
Analysing causalities leads to the application of system dynamics, a method developed by [12]. This technique is based on the identification of feedback loops between entities which form a dynamical system. Analysing the positive and negative loops leads to a qualitative description of the system as it is depicted in Figure 2.4(a), in the example of a population model.



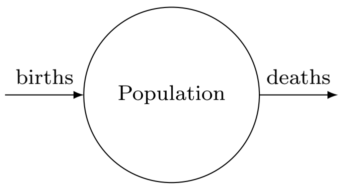
(a) Representation of system dynamics.



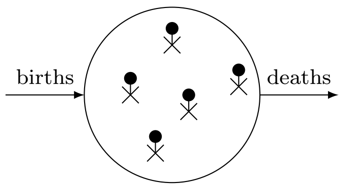
(b) Simple compartment model.

Figure 2.4.: Illustration of system dynamics and compartment modelling.

A similar description is given by compartment models which describes the flow of quantities from one entity to another as it is depicted in Figure 2.4(b). This modelling approach is often used in biomedicine, e.g. describing the blood flow in the human body divided in compartments as it is done in [23]. Both, compartment modelling and system dynamics approach, lead to a set of ordinary differential equations. Basically, they are based on the same hypotheses, as the system theory. This means, these modelling approaches are looking on a system as a whole, a so-called macroscopic modelling approach. Describing a system not only by one entity but as an entire system consisting of individual subsystem leads to the microscopic modelling approach as it is depicted in Figure 2.5. The agent based modelling approach is one example. Defining laws for individuals gives the opportunity to describe the entire system as combination of all single dynamics.



(a) Macroscopic modelling approach.



(b) Microscopic modelling appraoch.

Figure 2.5.: Representation of macroscopic and microscopic modelling appraoch.

In this work, a physical modelling approach is used. As it was mentioned above, this modelling approach is based on fundamental physical laws. Basically, this modelling approach builds the bridge to the simulation environments because the modelling process can be done in a simulation environment directly. The process in a simulator is done by the connection of pre-implemented components, which form a system model. Each component follows pre-defined functions and its dynamic can be influenced by parameters.
One derived method is the multibody modelling approach which is mainly used in this work. This approach gives the opportunity to analyse motion between multiple bodies. Given a global coordinate system allows to describe rotational and translational movements of bodies in respect to each other as it is depicted in Figure 2.6. Multibody models consists of bodies which are connected through joints. The bodies are described by their physical properties, e.g. mass, density and inertial rotation. The joints linking the bodies have various degrees of freedom, including rotational and translational degrees of freedom. Moreover, the bodies can be connected by flexible elements, as for example spring damper elements. Multibody modelling focuses on rigid bodies but it can be extended to flexible bodies as it is stated in [13]. The equations describing the dynamics in a multibody system can be derived following the Lagrange formalism $L = T - V$ for the kinetic energy T and potential energy V. Using the Lagrange's equations of the second kind

$$\frac{\mathrm{d}L}{\mathrm{d}t}\frac{\partial L}{\partial \dot{q}_k} - \frac{\partial L}{\partial q_k} = 0, \quad k = 1, 2, \ldots, n,$$

where q_k are the considered generalised coordinates, allows to describe the motion resulting under an applied force F in multibody systems by

$$M\ddot{q} + J_q^T \lambda = F,$$

with M, the mass matrix of the system, the Jacobian J in respect to the generalised coordinates q and the Lagrangian multipliers λ.

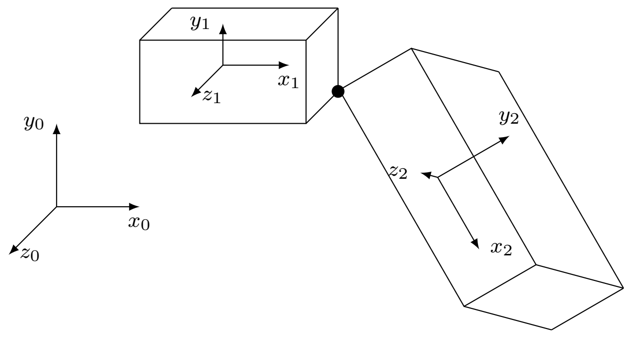


Figure 2.6.: Illustration of a multibody system and their local and the global coordinate system.

2.5. Selected Overview of Simulation Environments

The choice of the right simulation environment is an important question before the implementation of a conceptual model. This depends on various aspects which determine the entire implementation process. The structure of the conceptual model often requires special tools, e.g. ODE solver. Of course, this is influenced by the real system as well because the system limits the possibilities of the conceptual model structure. The application and usability of the simulation model for final experiments determine the choice as well. Furthermore, the usage of the simulation model by other people than the programmer itself or questions about infrastructure, as license e.g. can influence the software options.
In the following, three different simulation environments are introduced which are used in this work. The simulation environments are chosen due to their capabilities of multibody dynamics simulation. All of them contain modelling elements based on physical properties which are connected in the framework, forming the simulation model. This means, the implementation phase in these environments is supported by a visualisation of the simulation model. Furthermore, all of them offer animations of simulation results. These visualisations can be used for verification and validation. Obviously, the usage of visualisation is important in further work due to the importance of the underlying geometry of the bones in the knee in the models.

2.5.1. Simscape

Mathworks developed in the Simulink environment the Simscape library. It comprises various elements for physical modelling, including the Simscape multibody

library, formerly known as SimMechanics, which is used for modelling mechanical systems. In addition to the multibody blocks, elements for electrical circuits, thermal and fluid components exist.
The elements in the library are in their usage similar to the blocks in Simulink. Moreover, it is possible to use Simulink and Simscape elements in the same model. This combination requires blocks which convert the signal, `PS-Simulink Converter` from Simscape to Simulink and `Simulink-PS Converter` vice versa. The application of them is necessary, because the physical signals from Simscape differ in their format to the Simulink signals.
The embedment of Simscape in Simulink allows the usage of the powerful control tools offered in Simulink and MATLAB, one of the main reason to build the multibody model in Simscape in this work. Furthermore, the post processing of results in MATLAB and using numerical capabilities are important properties of Simscape. All Simscape models are based on ordinary differential equations which are solved by numerical ODE solvers. All seven solvers which are available in Simulink can be applied to Simscape models as well since the solving process is executed in the Simulink environment. Table 2.1 gives an overview about the available solvers and their used methods. Generally, the time steps of the solver can be chosen to be fixed or variable and the choice of the solver is done automatically.

ODE solver	Method
`ode45`	Dormand-Prince, Runge-Kutta, (4,5)
`ode23`	Bogacki and Shampine, Runge-Kutta, (2,3)
`ode113`	PECE Implementation of Adams-Bashforth-Moutlon
`ode15s`	Numerical Differentiation Formulas
`23s`	Second-order, modified Rosenbrock formula
`23t`	Trapezoidal rule
`23tb`	TR-BDF2

Table 2.1.: ODE solver in Simulink and Simscape.

The first three solvers `ode45`, `ode23` and `ode133` are explicit solvers, the following four are implicit solvers which are applicable for stiff systems.

2.5.2. MapleSim

Another simulation environment which provides a multibody library is MapleSim, developed by Maplesoft and based on Modelica. In contrast to Simscape, MapleSim is not directly embedded in Maple but has its own user interface. Similar to Simscape, MapleSim is intended for physical modelling. Therefore, the library contains

	ODE solver	Method
Fixed time steps	Euler	Forward Euler
	`Implicit Euler`	Backward Euler
	`RK2`	2nd order Runge-Kutta
	`RK3`	3rd order Runge-Kutta
	`RK4`	4th Runge-Kutta
Variable time steps	`RKF45`	Fehlberg, Runge-Kutta, (4,5)
	`CK45`	Cash-Karp, Runge-Kutta, (4,5)
	`Rosenbrock`	Implicit Rosenbrock, RK, (3,4)

Table 2.2.: ODE solver in MapleSim.

elements for thermal, hydraulic and electrical models as well.
The code of the models built in MapleSim is based on Modelica, an object-oriented modelling language for component-oriented modelling of complex systems. In contrast to Simscape, MapleSim offers Modelica code of the models. In the Modelica editor, the code describing the individual elements can be modified. Additionally, MapleSim offers the programming of custom defined components. This allows a lot more flexibility in model building than Simscape.
For all models built in MapleSim, the equations describing the model can be extracted. The equations can be calculated for the entire model or a subsystem only. If parameters are defined as symbolic variables using a parameter block, the equations are extracted with symbolic variables as well. The equations can be extracted as differential algebraic, ordinary differential or algebraic equations. The format depends on the model structure, not each form is supported for any model.
Furthermore, MapleSim offers control elements for designing feedback loops. The control tools are not that developed as in Simulink. Therefore, the design of the closed simulation loop is considered in Simulink henceforth.
As in Simscape, the models are solved using ODE solvers. They are distinguished by fixed and variable step time solvers. Table 2.2 gives an overview about the ODE solver in MapleSim.
The variable time-step solvers `CK45` and `Rosenbrock` are able to solve semi-stiff and stiff systems, respectively. The others are applicable to non-stiff systems only.

2.5.3. COMSOL Multiphysics

COMSOL Multiphysics is a software for modelling physical system which are solved by finite element analysis. The origin from COMSOL is FEMLAB, the former partial differential equation toolbox in MATLAB. The structure of COMSOL consists of modules, divided into various application fields, such as electrical, chemical, fluid and heat, structural and acoustic. The simulation models are built from pre-implemented components which vary in the modules, similar to the libraries in Simscape and MapleSim. The model structure is not visible as block diagram but as a model tree whose structure remains the same for each module. This means, the procedure of model building is the same, independent from the chosen module. After the development of the simulation model, studies can be performed representing the simulation experiments. COMSOL offers here a wide variety of studies which differ between the modules. For example, not only time-dependent, but as well eigenvalue studies for linear analysis can be performed. This expands the application fields of the simulation models due to its usability.

Since all models are solved by FEA, each model is defined on a geometry. An important step in the modelling process is the meshing of the geometry. Dependent on the quality of the mesh, the solution is more or less accurate. Various mesh designs which differ in size and shape can be chosen and evaluated.

The solution is calculated in discrete time steps on the defined mesh, the finite elements. Therefore, the solution depends for any study on time and space. A lot of post-processing tools for the solution are available, including plotting and animation.

COMSOL simplifies the model building process by LiveLinks to other software environments, e.g. MATLAB and Solidworks. This makes it possible to import geometries defined in Solidworks easily or to use simulation results for further investigations in MATLAB.

3. Basics of Control

This chapter presents aspects of basic control theory. After an introduction to the basic structure of a general control circuit and the applications of control, some mathematical tools for the analysis of the dynamics of systems are introduced. These techniques are needed to choose an appropriate design for the control circuit. Dependent on the structure of the underlying system, which will be controlled, different control designs are used. The main point in the choice of a control circuit is the difference between linear and nonlinear systems. Hence, various structures of control circuits, first for linear and afterwards for nonlinear systems, are introduced. Each method is illustrated by a basic example to outline different usage.
Unless stated otherwise, this chapter is based on [18], [19] and [11].

3.1. Dynamic Systems

In system theory and consequently in control theory, dynamic systems, as shown in Figure 3.1, are considered. A dynamic system is a set of dynamically interacting and independent components forming an integrated whole. Multiple influences coming from outside are acting on the dynamic system and consequently changing the behaviour of the system. Different input functions $\boldsymbol{u}\colon \mathbb{R}_0^+ \to \mathbb{R}^m$, initial values $\boldsymbol{x}_0$ and parameters $\boldsymbol{p} \in \mathbb{R}^p$ influence the internal state variables $\boldsymbol{x} \in \mathbb{R}^n$ and lead to a particular system behaviour and thus to output functions $\boldsymbol{y} \in \mathbb{R}^l$.

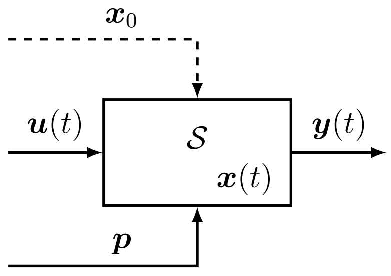


Figure 3.1.: Block diagram representing a dynamic system.

The behaviour of dynamic systems is not direct proportional to the input and disturbance adaption because it changes its behaviour dependent on its own dynamics. Nevertheless, it is possible to describe the behaviour of dynamic systems mathematically by functions in the form

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}, t), \quad \boldsymbol{x}(t_0) = \boldsymbol{x}_0 \tag{3.1a}$$

$$\boldsymbol{y} = \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{u}, t). \tag{3.1b}$$

The two functions $\boldsymbol{f}$ and $\boldsymbol{h}$ describe the dynamic of the system and the output, whereby the function $\boldsymbol{f}$ is a set of first order ordinary differential equations, and its solution is in the following indicated by $\boldsymbol{x}(t) = \boldsymbol{\varphi}(\boldsymbol{x}_0, \boldsymbol{u}, t)$.

3.2. The general Structure of Control Circuits and the Purpose of Control

The composition of a general control circuit can be seen in Figure 3.2. A control circuit is a closed loop system which consists of multiple parts. The aim of the closed loop is to influence the behaviour of a dynamic system, which is denominated as the plant. The output $\boldsymbol{y}$ is fed back to the origin of the circuit and therefore closes the simulation loop. The difference $\boldsymbol{e}$ between the output $\boldsymbol{y}$ and the reference input $\boldsymbol{r}$ is evaluated and defines the input of the controller. Depending on to the difference $\boldsymbol{e}$, the controller calculates an appropriate input for the plant.

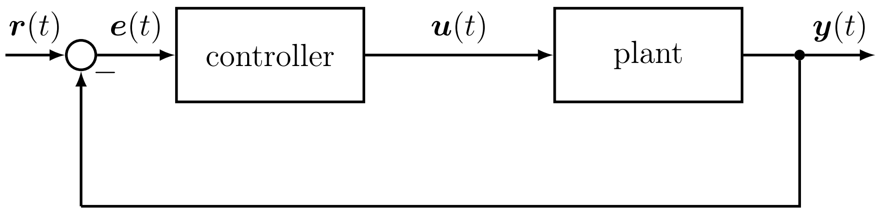


Figure 3.2.: A general control circuit consisting of a plant and one controller.

Control has to fulfil various tasks. The most obvious is to introduce a reference signal in order to influence the dynamics of the system in such way that the output matches the desired signal. Additionally, control can be applied for the stabilisation of an instable plant. Furthermore, in some cases it is possible to suppress the sensitivity to parameters with the help of control.
Thus, three points summarise the main tasks of control:

- Output $\boldsymbol{y}(t)$ should follow the reference signal $\boldsymbol{r}(t)$.
- Stabilisation of an instable system is possible.
- Suppress the sensitivity of a system to some parameters.

The last two points offer the possibility to include control theory in the context of modelling and simulation.

3.3. Control of linear Plants

This section introduces techniques for the control of linear systems. In the beginning, the definition of a linear system is given and its characteristics are investigated. This includes as well the properties of the solution of linear dynamic systems. Next, three important qualities of linear systems are introduced in order to investigate if a linear system fulfils all necessary conditions to include it in a closed feedback loop. To conclude, this section ends with three different designs of a control circuit for linear systems. These foundations give the opportunity to extend the theory of control for nonlinear systems in the next section.
Analysing a dynamic linear system includes determining the functions describing the behaviour and dynamics of the system. A dynamic system is linear with respect to the state variables, the input and time. Mathematically, this can be described by the following

Definition 3.3.1 (Linear dynamic system). A dynamic system given by the equations 3.1 is a linear dynamic system if for all input values $\boldsymbol{u}$ and each starting time $t_0 \geq 0$, the output $\boldsymbol{y}(\boldsymbol{x}_0, \boldsymbol{u}, t) = \boldsymbol{h}(\boldsymbol{\varphi}(\boldsymbol{x}_0, \boldsymbol{u}, t), \boldsymbol{u}, t) \in \mathbb{R}^m$ fulfils the following conditions

$$\boldsymbol{y}(\alpha_1 \boldsymbol{x}_{0,1} + \alpha_2 \boldsymbol{x}_{0,2}, \boldsymbol{0}, t) = \alpha_1 \boldsymbol{y}(\boldsymbol{x}_{0,1}, \boldsymbol{0}, t) + \alpha_2 \boldsymbol{y}(\boldsymbol{x}_{0,2}, \boldsymbol{0}, t) \tag{3.2a}$$

$$\boldsymbol{y}(\boldsymbol{0}, \beta_1 \boldsymbol{u}_1 + \beta_2 \boldsymbol{u}_2, t) = \beta_1 \boldsymbol{y}(\boldsymbol{0}, \boldsymbol{u}_1, t) + \beta_2 \boldsymbol{y}(\boldsymbol{0}, \boldsymbol{u}_2, t) \tag{3.2b}$$

$$\boldsymbol{y}(\boldsymbol{x}_0, \boldsymbol{u}, t) = \boldsymbol{y}(\boldsymbol{x}_0, \boldsymbol{0}, t) + \boldsymbol{y}(\boldsymbol{0}, \boldsymbol{u}, t) \tag{3.2c}$$

for $\alpha_i, \beta_i \in \mathbb{R}, i = 1, 2$.

Subsequently, also linear time-invariant systems are investigated in detail. The dynamic behaviour of these systems is independent on time delays. This property can be described as follows.

Definition 3.3.2 (Time-invariant System). A dynamic system is called time invariant if for all input values $\boldsymbol{u}$ and each starting time $t_0 \geq 0$ the following condition holds.
If $\boldsymbol{y}(t)$ indicates the output of the system at time t for the initial value $\boldsymbol{x}(t_0) = \boldsymbol{x}_0$ and the input value $\boldsymbol{u}(\tau), t_0 \leq \tau \leq t$, then $\boldsymbol{y}(t - T)$ is the output of the system with the initial value $\boldsymbol{x}(t_0 + T) = \boldsymbol{x}_0$ and the input value $\boldsymbol{u}(\tau - T), t_0 + T \leq \tau \leq t + T$.

Since the behaviour of linear dynamic systems is described by linear mappings, it is possible to use transform matrices. Consequently, four matrices are needed to represent a linear dynamic system. This description is called the state-space representation.

Theorem 3.3.3 (State-space representation). A system as denoted in 3.1 is linear if and only if it can be written as

$$\dot{\boldsymbol{x}} = A(t)\boldsymbol{x} + B(t)\boldsymbol{u}, \quad \boldsymbol{x}(t_0) = \boldsymbol{x}_0 \tag{3.3a}$$

$$\boldsymbol{y} = C(t)\boldsymbol{x} + D(t)\boldsymbol{u}. \tag{3.3b}$$

Hereby, $A(t) \in \mathbb{R}^{n\times n}$ designates the state or system matrix, $B(t) \in \mathbb{R}^{n\times m}$ is the input matrix, $C(t) \in \mathbb{R}^{l\times n}$ denotes the output matrix and $D(t) \in \mathbb{R}^{l\times m}$ the feedthrough matrix.

Furthermore, this description can be found as well for linear time-invariant systems. The only difference is that the four matrices are constant and not dependent on time anymore.

Theorem 3.3.4. A system is linear and time invariant if it can be convicted to

$$\begin{aligned} \dot{\boldsymbol{x}} &= A\boldsymbol{x} + B\boldsymbol{u}, \quad \boldsymbol{x}(0) = \boldsymbol{x}_0 && (3.4a)\\ \boldsymbol{y} &= C\boldsymbol{x} + D\boldsymbol{u}. && (3.4b) \end{aligned}$$

The four matrices are called as before in theorem 3.3.3.

3.3.1. Laplace Transform

In control engineering as well as in mathematical modelling and simulation, the Laplace Transform is a common tool used to describe LTI systems. The Laplace Transform is a transformation of functions given in time t to functions in the frequency domain s. Therefore, this mapping is a function from the real numbers to the complex numbers. The following is based on [16] and [24]. In order to simplify the area of existence for the Laplacian, a set containing functions is defined.

Definition 3.3.5. Let $I = [0, \infty)$. The set $\mathcal{P}(I, \mathbb{R})$ contains all functions $f\colon I \to \mathbb{R}$ which

(i) are piecewise continuous on any finite subinterval of I and

(ii) fulfils $|f(t)| \leq C\mathrm{e}^{\gamma t}$, for $C, \gamma \in \mathbb{R}$.

This gives the possibility to define the Laplacian.

Definition 3.3.6. Given a function $f\colon I \to \mathbb{R}$. For $f \in \mathcal{P}(I, \mathbb{R})$ the integral

$$F(s) = \mathcal{L}(f)(s) = \int_0^\infty \mathrm{e}^{-st} f(t)\,\mathrm{d}t, \quad s = \alpha + \mathrm{i}\,\omega \tag{3.5}$$

is convergent for all s with $\operatorname{Re} s = \alpha > \gamma$. This function $F(s) = \mathcal{L}(f)(s)$ is called the Laplace Transform of the function $f(t)$ and the domain $\Gamma = \{s \in \mathbb{C}\colon \operatorname{Re} s > \gamma\}$ the area of existence of $\mathcal{L}(f)(s)$.

Therefore, the Laplace Transform is a function with the following domains

$$\begin{aligned} \mathcal{L}\colon \mathcal{P}(I, \mathbb{R}) &\to \mathcal{M}(\mathbb{C}, \mathbb{C}),\\ f(t) &\mapsto F(s), \end{aligned}$$

whereby $\mathcal{M}(\mathbb{C},\mathbb{C})$ describes the space, which contains all functions from $\mathbb{C}$ to $\mathbb{C}$. The original function f in time-domain can be calculated by the Inverse Laplace Transform, given by

$$f(t) = \mathcal{L}^{-1}(F)(t) = \frac{1}{2\pi\,\mathrm{i}} \int\limits_{\Gamma} \mathrm{e}^{st} F(s)\,\mathrm{d}s, \tag{3.6}$$

with the convergence area $\Gamma := \{s \in \mathbb{C} : s = r + \mathrm{i}\,t, r \geq \gamma, t \in \mathbb{R}\}$. Following this calculation and the definition before shows that the inverse Laplacian is a function

$$\begin{aligned} \mathcal{L}^{-1}\colon\ \operatorname{img}(\mathcal{P}(I,R)) &\to \mathcal{P}(I,R), \\ F(s) &\mapsto f(t). \end{aligned}$$

The image of the Laplace Transform is denoted as $\operatorname{img}(\mathcal{P}(I,R))$.
Some of the most important properties of the Laplace Transform are summarised below.

Theorem 3.3.7 (Properties of the Laplace Transform). The following properties are valid for the Laplacians $\mathcal{L}(f)(s) = F(s), \mathcal{L}(f_1)(s) = F_1(s), \mathcal{L}(f_2)(s) = F_2(s)$.

(i) Linearity.

$$\mathcal{L}(c_1 f_1 + c_2 f_2) = c_1 F_1(s) + c_2 F_2(s), \quad c_1, c_2 \in \mathbb{R}$$

(ii) Differentiation.

$$\mathcal{L}\left(\frac{\mathrm{d}^n}{\mathrm{d}t^n} f\right)(s) = \mathcal{L}(f^{(n)})(s) = s^n F(s) - f(0)\, s^{n-1} - f^{(1)}(0) s^{n-2} - \ldots - f^{(n-1)}(0)$$

(iii) Integration.

$$\mathcal{L}\left(\int_0^t f(\tau)\,\mathrm{d}t\right) = \frac{1}{s} F(s)$$

(iv) Convolution theorem.

$$\begin{aligned} \mathcal{L}(f_1 * f_2)(s) =& \mathcal{L}\left(\int\limits_0^t f_1(\tau) f_2(t-\tau)\,\mathrm{d}\tau\right)(s) \\ =& \mathcal{L}\left(\int\limits_0^t f_1(t-\tau) f_2(\tau)\,\mathrm{d}\tau\right)(s) = F_1(s) F_2(s) \end{aligned}$$

(v) Displacement.

$$\mathcal{L}(f(t+a)) = \mathrm{e}^{as}\left(F(s) - \int\limits_0^a f(t)\mathrm{e}^{-st}\,\mathrm{d}t\right), \quad a > 0$$

$f(t)$	$F(s)$	$f(t)$	$F(s)$	$f(t)$	$F(s)$
t	$\frac{1}{s^2}$	$\sin(bt)$	$\frac{b}{s^2+b^2}$	$e^{at}\sin(bt)$	$\frac{b}{(s-a)^2+b^2}$
$t^n\, e^{at}$	$\frac{n!}{(s-a)^{n+1}}$	$\cos(bt)$	$\frac{s}{s^2+b^2}$	$e^{at}\cos(bt)$	$\frac{s-a}{(s-a)^2+b^2}$

Table 3.1.: Laplace Transforms of common used functions.

The Laplacians of common functions are listed in Table 3.1. Not only for univariate but also for multivariate functions $\boldsymbol{f}$ the Laplace Transform is defined.

Definition 3.3.8 (Laplace Transform for vectorial functions). Consider a vector field $\boldsymbol{f}\colon \mathbb{R}^n \to \mathbb{R}^n$. The Laplace Transform of this function is calculated by the Laplace Transform for each entry, as follows

$$\mathcal{L}(\boldsymbol{f})(s) = \begin{pmatrix} \mathcal{L}(f_1)(s) \\ \mathcal{L}(f_2)(s) \\ \vdots \\ \mathcal{L}(f_n)(s) \end{pmatrix}.$$

3.3.2. Transfer Functions

Given a LTI system as in (3.4), it is possible to apply the Laplace Transform in order to obtain the description by multiplications only. Hence, the solution of the system of differential equations is calculated easily. In order to reproduce this calculation, the state space representation of an LTI system is given with

$$\begin{aligned} \dot{\boldsymbol{x}} &= A\boldsymbol{x} + B\boldsymbol{u}, \quad \boldsymbol{x}(0) = \boldsymbol{x}_0, \\ \boldsymbol{y} &= C\boldsymbol{x} + D\boldsymbol{u}. \end{aligned}$$

The Laplace Transforms of the state vector, the input and the output are denoted in the following as $\mathcal{L}(\boldsymbol{x}) = \boldsymbol{X}, \mathcal{L}(\boldsymbol{x}_0) = \boldsymbol{X}_0$, $\mathcal{L}(\boldsymbol{u}) = \boldsymbol{U}$ and $\mathcal{L}(\boldsymbol{y}) = \boldsymbol{Y}$. Applying the Laplace Transform on the LTI system (3.4), it converts to

$$\begin{aligned} s\boldsymbol{X} - \boldsymbol{X}_0 &= A\boldsymbol{X} + B\boldsymbol{U} && (3.7) \\ \boldsymbol{Y} &= C\boldsymbol{X} + D\boldsymbol{U}. && (3.8) \end{aligned}$$

Equation (3.7) can now be transferred to

$$\begin{aligned} \boldsymbol{X}(sI_n - A) &= B\boldsymbol{U} + \boldsymbol{X}_0 \\ \boldsymbol{X} &= (sI_n - A)^{-1}(B\boldsymbol{U} + \boldsymbol{X}_0), && (3.9) \end{aligned}$$

with the identity matrix $I_n \in \mathbb{R}^{n \times n}$. Substitution of this result in equation (3.8) gives

$$\begin{aligned} \boldsymbol{Y} &= C(sI_n - A)^{-1}B\boldsymbol{U} + C(sI_n - A)^{-1}\boldsymbol{X}_0 + D\boldsymbol{U} \\ \boldsymbol{Y} &= C(sI_n - A)^{-1}\boldsymbol{X}_0 + (C(sI_n - A)^{-1}B + D)\boldsymbol{U}, \end{aligned} \tag{3.10}$$

which leads to the following

Definition 3.3.9 (Transfer function). Considering the LTI system (3.4) with the Laplace Transforms of the input $\mathcal{L}(\boldsymbol{u}) = \boldsymbol{U} \in \mathbb{R}^m$, output $\mathcal{L}(\boldsymbol{y}) = \boldsymbol{Y} \in \mathbb{R}^l$ and the initial value $\boldsymbol{x}(0) = \boldsymbol{x}_0 = 0$. The function G satisfying the equation

$$\boldsymbol{Y}(s) = G(s)\boldsymbol{U}(s), \tag{3.11}$$

is called the transfer function $G \in \mathbb{R}^{l \times m}$ of the LTI system.

In the case of a single input-single output (SISO) system, the transfer function is a scalar function describing the transmission behaviour of the output y with respect to the input u. For a multiple input-multiple output (MIMO) system, a matrix filled with transfer functions is considered. Each entry of this matrix describes the transmission behaviour for one entry of the output vector $\boldsymbol{y}$ and the corresponding entry in the input vector $\boldsymbol{u}$ as it is stated in more detail in [30]. The matrix G has the form

$$G(s) = \begin{pmatrix} G_{11}(s) & \cdots & G_{1m}(s) \\ \vdots & \ddots & \vdots \\ G_{l1}(s) & \cdots & G_{lm}(s) \end{pmatrix}.$$

The elements of the matrix G can be calculated following

Theorem 3.3.10. The transfer function G_{ij} of a LTI system defined in 3.4 is calculated by

$$G_{ij}(s) = (\boldsymbol{c}_i^T(sI_n - A)^{-1}\boldsymbol{b}_j + \boldsymbol{d}_i) \tag{3.12a}$$

$$= \frac{\boldsymbol{c}_i^T(sI_n - A)_{\text{adj}}\boldsymbol{b}_j}{\det(sI_n - A)} + \boldsymbol{d}_i = \frac{p_{ij}(s)}{q_{ij}(s)}, \tag{3.12b}$$

with the polynomials p_{ij}, q_{ij} and $i = 1 \ldots l, j = 1 \ldots m$. The vectors $\boldsymbol{c}_i, \boldsymbol{d}_i$ contain the values of the i-th row of the matrix C respectively D and the vector $\boldsymbol{b}_j$ the values of the j-th column of the matrix B. The zeros of the denominators q_{ij} are the poles of the transfer function G_{ij} and simultaneously eigenvalues of the state matrix A. The polynomial degree of the nominator p_{ij} defines the order of the transfer function G_{ij}.

It is important to notice that not all eigenvalues of the matrix A are poles of one transfer function G_{ij} due to cancellations with the nominator. Given a transfer function G and calculating the describing system leads to the realisability problem. If the transfer function fulfils some requirements, this realisation is unique. This is stated in detail in the following

Theorem 3.3.11. Consider the univariate transfer function

$$G = \frac{p}{q}.$$

This transfer function is realisable if

$$\deg p \leq \deg q.$$

The transfer function is called strictly proper if $\deg p < \deg q$, otherwise proper. Equivalent, the inequation

$$\lim_{s \to \infty} \|G(s)\| < \infty$$

is fulfilled.

If this condition is not satisfied, the polynomial division would lead to derivations of the input $\boldsymbol{u}$ and therefore to a non-unique structure of the system.
Considering two dynamic systems, described by transfer functions as

$$\begin{aligned} \boldsymbol{Y}_1(s) &= G_1(s)\boldsymbol{U}_1(s), \\ \boldsymbol{Y}_2(s) &= G_2(s)\boldsymbol{U}_2(s), \end{aligned}$$

leads to various options of combinations. The first possibility is to place them in series, therefore that $\boldsymbol{Y}_1 = \boldsymbol{U}_2$. This leads to a system with the input $\boldsymbol{u} = \boldsymbol{U}_1$ and the output $\boldsymbol{y} = \boldsymbol{Y}_2$, as depicted in Figure 3.3. Calculating the transfer function for

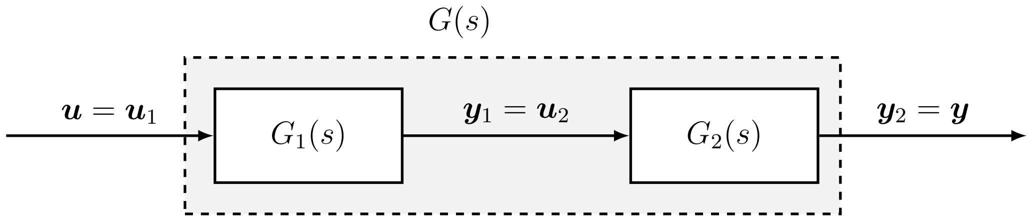

$G(s)$
$\boldsymbol{u} = \boldsymbol{u}_1$
$G_1(s)$
$\boldsymbol{y}_1 = \boldsymbol{u}_2$
$G_2(s)$
$\boldsymbol{y}_2 = \boldsymbol{y}$

Figure 3.3.: Block diagram of two transfer functions connected in series.

the entire system leads to the equations for the input $\boldsymbol{u}$ and the output $\boldsymbol{y}$,

$$\begin{aligned} \boldsymbol{Y}_1(s) &= G_1(s)\boldsymbol{U}_1(s) \\ \boldsymbol{Y}_2(s) &= G_2(s)\boldsymbol{U}_2(s) = G_2(s)G_1(s)\boldsymbol{U}_1(s) = G_2(s)G_1(s)\boldsymbol{U}(s). \end{aligned}$$

The resulting transfer function G can be calculated by component multiplication of the two transfer functions G_1 and G_2.

$$\frac{\boldsymbol{Y}(s)}{\boldsymbol{U}(s)} = G(s) = G_2(s)G_1(s) \tag{3.13}$$

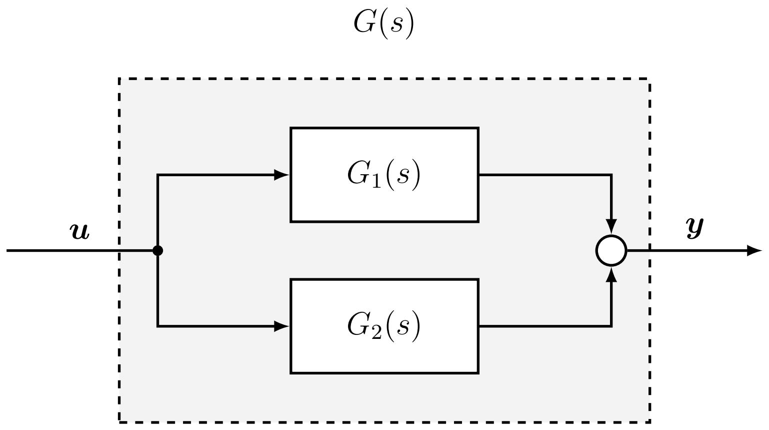


Figure 3.4.: Block diagram of two transfer functions connected in parallel.

Another possibility to combine transfer functions would be the connection in parallel as it is depicted in Figure 3.4 with two transfer functions G_1 and G_2. Again, it is possible to calculate the transfer function for the whole system, considering first the equation for the Laplacian of the output $\boldsymbol{y}$,

$$\boldsymbol{Y}_1(s) + \boldsymbol{Y}_2(s) = G_1(s)\boldsymbol{U}(s) + G_2(s)\boldsymbol{U}(s) = (G_1(s) + G_2(s))\boldsymbol{U}(s).$$

The resulting transfer matrix G is calculated by adding the two transfer matrices G_1, G_2.

$$\frac{\boldsymbol{Y}(s)}{\boldsymbol{U}(s)} = G(s) = G_1(s) + G_2(s) \tag{3.14}$$

Two transfer functions can be implemented in a closed feedback loop as it is depicted in Figure 3.5 and again, the resulting transfer matrix can be calculated easily. The

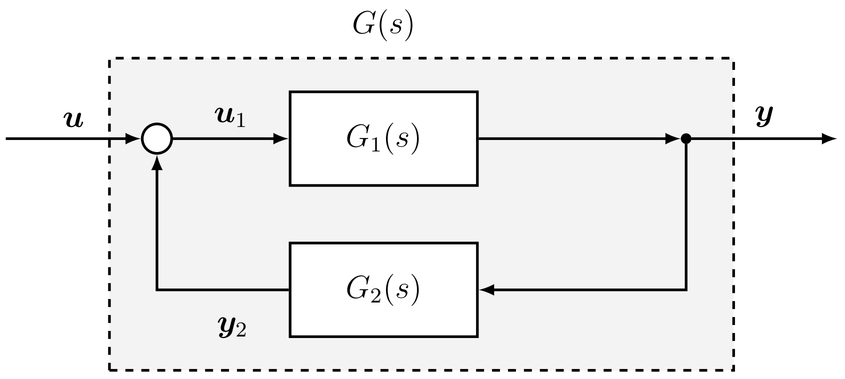


Figure 3.5.: Transfer functions switched in feedback.

input $\boldsymbol{u}_1$ for the transfer function G_1 is the difference between the input $\boldsymbol{u}$ and the output $\boldsymbol{y}_2$ of the transfer function G_2. In view of this relationship, the following calculations can be derived.

$$\begin{aligned}
\boldsymbol{U}_1(s) &= \boldsymbol{U}(s) + \boldsymbol{Y}_2(s) = \boldsymbol{U}(s) + \boldsymbol{Y}(s)G_2(s) \\
\boldsymbol{Y}(s) &= \boldsymbol{U}_1(s)G_1(s) = (\boldsymbol{U}(s) + \boldsymbol{Y}(s)G_2(s))G_1(s) \\
\boldsymbol{Y}(s)(1 - G_2(s)G_1(s)) &= \boldsymbol{U}(s)G_1(s)
\end{aligned}$$

For the transfer function G, the following equation holds

$$\frac{\boldsymbol{Y}(s)}{\boldsymbol{U}(s)} = G(s) = \frac{G_1(s)}{1 - G_2(s)G_1(s)}. \tag{3.15}$$

Concluding this, the transfer functions are forming an algebra, which means that for all transfer functions G_1, G_2, G_3 and $s \in \mathbb{C}$, the following equations are fulfilled.

$$\begin{aligned}
(G_1 + G_2)G_3 &= G_1G_2 + G_2G_3 && (3.16) \\
G_1(G_2 + G_3) &= G_1G_2 + G_1G_3 && (3.17) \\
s(G_1G_2) &= (sG_1)G_2 = G_1(sG_2) && (3.18)
\end{aligned}$$

These equations allow to combine transfer functions in such a way that the resulting transfer function still fulfils all required conditions.
Concluding the theory of transfer functions leads to the possibility to define some standard elements included in linear control, following [22]. Of course, it is possible to combine all these presented elements to create more complex control structures. These connections follow the rules introduced in section 3.3.2. In Tables 3.2 and 3.3, the mathematical formulas and the corresponding transfer function for dynamic systems with the input $\boldsymbol{u}$ and the output $\boldsymbol{y}$ are illustrated. Moreover, the corresponding step responses to the Heaviside function are illustrated to show their acting behaviour. In most of the cases, the reaction of the output related to the input is clear, e.g. implementing the proportional controller leads to a proportional reaction of the output to the constant C. Consider in the following, the constant $C \in \mathbb{R}$ and time constants $T, T_1, T_2 > 0$. It is important to notice that the transfer function for the differentiator

$$G(s) = Cs$$

is not realisable due to the fact that a transfer function requires

$$\lim_{s \to \infty} \|G(s)\| < \infty$$

in order to represent a unique structure of the state space representation. In other words, the degree of the nominator must be lower or equal than the degree of the denominator. This leads to an extension of the denominator with an additional term, resulting in the final transfer function of the differentiator of

$$G(s) = \frac{Cs}{Ts + 1}.$$

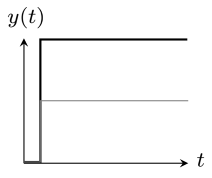
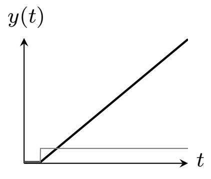
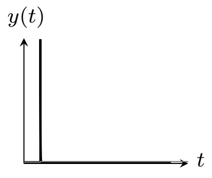
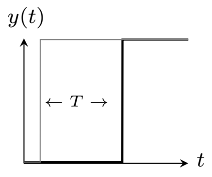
	$f(t)$	$G(s)$	Step response
Proportional	$y(t) = Cu(t)$	C	
Integrator	$y(t) = C\int\limits_0^t u(\tau)\,\mathrm{d}\tau$	$\frac{C}{s}$	
Differentiator	$y(t) = C\dot{u}(t)$	$\frac{Cs}{Ts+1}$	
Delay	$y(t) = u(t-T)$	e^{-sT}	

Table 3.2.: Basic control elements.

Of course, it is possible to combine these standard elements, ensuing control elements with more properties. An important and often used element is the well-known PID element which combines the proportional, the integrator and the differentiator element. These three elements are connected in parallel which implies for the entire transfer function the summation of all three elements. In the time domain, the output is calculated by

$$y(t) = C\left(\boldsymbol{u}(t) + \int\limits_0^t \boldsymbol{u}(\tau)\,\mathrm{d}\tau + \dot{\boldsymbol{u}}(t)\right).$$

Therefore, the transfer function in the s-domain is given as

$$G(s) = C\left(1 + \frac{1}{T_I s} + T_D s\right), \tag{3.19}$$

with two time constants T_I, T_D and two constants C_I, C_D depending on the proportional constant with $C_P = C_I T_I$ and $C_D = C_P T_D$. Simplifying the equation 3.19 and redefining two time constants T_1, T_2 depending on T_I and T_D leads to the equation

$$G(s) = C\frac{(T_1 s + 1)(T_2 s + 1)}{s}.$$

However, this transfer function is not realisable. Therefore, the denominator is substituted by the term $Ts + 1$, leading to the final transfer function of the PID element

$$G(s) = C\frac{(T_1 s + 1)(T_2 s + 1)}{Ts + 1}.$$

The step response of the PID element is depicted in Figure 3.6 and shows a re-

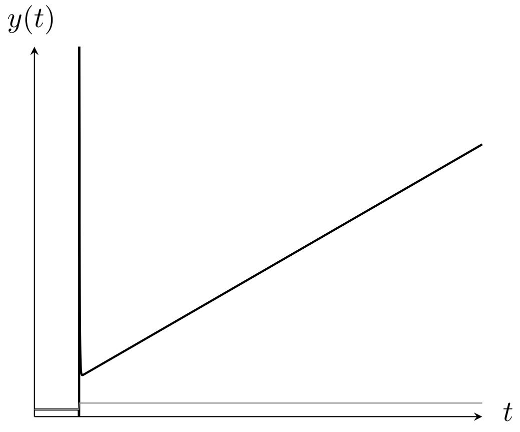


Figure 3.6.: Step response for the PID element.

action of all three included elements. In conclusion, two more standard elements are presented in Table 3.3, the P-T$_1$ and P-T$_2$ element. One additional note may be taken, that the element P-T$_2$ allows to implement an oscillating reaction of the output according to the input.

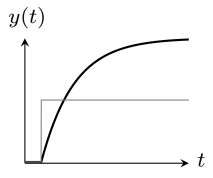
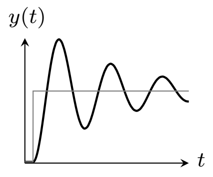
	$f(t)$	$G(s)$	Step response
P-T$_1$	$T\dot{y}(t) + y(t) = Cu(t)$	$\dfrac{C}{Ts+1}$	
P-T$_2$	$T^2\ddot{y}(t) + T\dot{y}(t) + y(t) = Cu(t)$	$\dfrac{C}{(sT)^2 + 2\xi sT + 1}$	

Table 3.3.: Standard control elements.

3.3.3. Properties of the Solution of linear Systems

Considering linear systems gives the opportunity to describe their analytical solutions and further to analyse them in respect to their properties. The solution can be calculated easily because the behaviour can be described by a first order differential equation.

Theorem 3.3.12 (Transition matrix for autonomous systems). Let $A \in \mathbb{R}^{n\times n}, \boldsymbol{x} \in \mathbb{R}^n$. Considering the linear autonomous system

$$\dot{\boldsymbol{x}} = A\boldsymbol{x}, \quad \boldsymbol{x}(t_0) = \boldsymbol{x}_0, \tag{3.20}$$

the solution can be calculated by

$$\boldsymbol{x}(t) = \Phi(t)\boldsymbol{x}_0, \tag{3.21}$$

where $\Phi(t) \in \mathbb{R}^{n\times n}$ is defined as

$$\Phi(t) = \sum_{k=0}^{\infty} A^k \frac{t^k}{k!} \tag{3.22}$$

which is called the transition matrix.

Following the definition for the exponential series, the transition matrix can be written as

$$\Phi(t) = e^{At}. \tag{3.23}$$

Theorem 3.3.13 (Characteristics of the transition matrix). As a consequence of the calculus for the exponential function, four characteristics can be formulated for the transition matrix.

(i) $\Phi(0) = I_n$

(ii) $\Phi(t+s) = \Phi(t)\Phi(s)$

(iii) $\Phi^{-1}(t) = \Phi(-t)$

(iv) $\frac{\mathrm{d}}{\mathrm{d}t}\Phi(t) = A\Phi(t)$

The solution can be found analogously for LTI systems, it is only extended by the dependence on the matrix B.

Theorem 3.3.14 (Transition matrix for LTI systems). For the linear time-invariant system of the form 3.4, the solution can be written as

$$\begin{aligned} \boldsymbol{x}(t) &= \Phi(t)\boldsymbol{x}_0 + \int_0^t \Phi(t-\tau)B\boldsymbol{u}(\tau)\mathrm{d}t && (3.24) \\ \boldsymbol{y}(t) &= C\boldsymbol{x}(t) + D\boldsymbol{u}(t) && (3.25) \end{aligned}$$

with the transition matrix $\Phi(t) \in \mathbb{R}^{n\times n}$ as in (3.23).

By applying the Laplace Transform on LTI systems, the transition matrix can be calculated with the state matrix A. Using formula 3.24 and applying the Laplace Transform leads to

$$\begin{aligned} \boldsymbol{x}(t) &= \Phi(t)\boldsymbol{x}_0 + \int_0^t \Phi(t-\tau)B\boldsymbol{u}(\tau)\mathrm{d}t \\ \boldsymbol{x}(t) &= \Phi(t)\boldsymbol{x}_0 + B\int_0^t \Phi(t-\tau)\boldsymbol{u}(\tau)\mathrm{d}t \\ \boldsymbol{X}(s) &= \mathcal{L}(\Phi)(s)\boldsymbol{x}_0 + B\mathcal{L}\left(\int_0^t \Phi(t-\tau)\boldsymbol{u}(\tau)\mathrm{d}t\right). && (3.26) \end{aligned}$$

The convolution theorem, introduced in 3.3.7, can be utilised and finally leads to

$$\boldsymbol{X}(s) = \mathcal{L}(\Phi)(s)\boldsymbol{x}_0 + B\mathcal{L}(\Phi)(s)\boldsymbol{U}(s). \tag{3.27}$$

Comparing this equation to

$$\boldsymbol{X}(s) = (sI_n - A)^{-1}\boldsymbol{x}_0 + (sI_n - A)^{-1}B\boldsymbol{U}(s), \tag{3.28}$$

which directly results from 3.9, leads to the conclusion, that the transition matrix Φ in the s-domain can be calculated by

$$\mathcal{L}(\Phi)(s) = (sI_n - A)^{-1}. \tag{3.29}$$

The knowledge of the eigenvalues of the matrix A allows to describe the solutions of a linear system as linear combinations which is stated in the following

Theorem 3.3.15 (Solution properties). Considering the linear autonomous system (3.20), each solution $x_j(t), j = 1, \dots, n$ is a linear combination of the functions

$$t^{k_1}\mathrm{e}^{\lambda t}, t^{k_2}\mathrm{e}^{\alpha t}\cos(\beta t), t^{k_3}\mathrm{e}^{\alpha t}\sin(\beta t)$$

for real eigenvalues λ of A, conjugated eigenvalues $\alpha \pm \mathrm{i}\beta$ of A and $k_m = 0, \dots, (r-1)$ with r as the multiplicity of the corresponding eigenvalues and $m = 1, 2, 3$.

With the help of this description for the solution of linear systems, it is now easy to determine if this system is stable or not. For further analysis, an equilibrium state of an autonomous system has to be defined.

Definition 3.3.16 (Equilibrium state). $\boldsymbol{x}_E \in R^n$ is an equilibrium state of the system $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$ if $\boldsymbol{f}(\boldsymbol{x}_R) = \boldsymbol{0}$ holds for all $t \geq 0$.

Theorem 3.3.17 (Global asymptotic Stability). Consider LTI autonomous systems as in (3.20). If all eigenvalues of A have a negative real part, then the equation

$$\lim_{t\to\infty} \boldsymbol{x}(t) = \lim_{t\to\infty} \Phi(t)\boldsymbol{x}_0 = \boldsymbol{0},$$

holds for all initial values $\boldsymbol{x}_0 \in \mathbb{R}^n$. In this case, the equilibrium state $\boldsymbol{x}_E = \boldsymbol{0}$ is called global asymptotic stable.

This theorem is not only valid for the equilibrium state $\boldsymbol{x}_E = 0$ but also for all equilibrium states. This can be proven by state space transformation in the form of $\boldsymbol{x}(t) = V\boldsymbol{z}(t)$ with a regular matrix $V \in \mathbb{R}^n$. Applying this transformation to a LTI system of the form (3.4) leads to the system

$$\begin{aligned} \dot{\boldsymbol{z}} &= V^{-1}AV\boldsymbol{x} + V^{-1}B\boldsymbol{u}, \quad \boldsymbol{z}(0) = \boldsymbol{z}_0 = V^{-1}\boldsymbol{x}_0 && (3.30a) \\ \boldsymbol{y} &= CV\boldsymbol{z} + D\boldsymbol{u}. && (3.30b) \end{aligned}$$

Both descriptions of an LTI system are equivalent and the matrices A and $V^{-1}AV$ are similar. The following theorems are formulated for the equilibrium state $\boldsymbol{x}_E = \boldsymbol{0}$. With the transformation (3.30) it is possible to extend them for all equilibrium states.

For LTI systems, an additional stability is introduced which is valid for the entire system and not only locally on the equilibrium states anymore. This stability is only valid for a scalar input or output u, y, respectively.

Definition 3.3.18 (BIBO Stability). Considering linear time-invariant systems with $A \in \mathbb{R}^{n\times n}, \boldsymbol{x}, \boldsymbol{b}, \boldsymbol{c} \in \mathbb{R}^n$, a scalar input $u \in \mathbb{R}$ and output function $y \in \mathbb{R}$ as well as $d \in \mathbb{R}$ of the form

$$\begin{aligned} \dot{\boldsymbol{x}} &= A\boldsymbol{x} + \boldsymbol{b}u, \quad \boldsymbol{x}(0) = \boldsymbol{x}_0, && (3.31a) \\ y &= \boldsymbol{c}^T\boldsymbol{x} + du. && (3.31b) \end{aligned}$$

A linear time-invariant single input-single output system is called bounded-input, bounded-output stable if for all bounded input functions $|u(t)| \leq r$ exists a bounded output function $|y(t)| \leq s$ for all $r, s > 0$.

Analogously to the global asymptotic stability, a theorem can be formulated to verify the BIBO stability just by knowing the eigenvalues of the matrix A.

Theorem 3.3.19. A system of the form (3.31) is BIBO stable if and only if all poles $s_j = \alpha_j + \mathrm{i}\omega_j$ of the transfer function $G(s)$ have $\operatorname{Re} s_j = \alpha_j < 0 \ \forall j$, where $Y(s) = G(s)U(s)$.

These definitions and theorems provide the basis for the design of a closed feedback loop so that an instable system can be stabilised.

3.3.4. Stability, Observability, Controllability

Three terms are introduced to ensure that the given system is suitable for an embedment to a closed feedback loop. In order to integrate a dynamic system in a closed feedback loop, the system has to fulfil some requirements.
Firstly, the system needs to be defined in such way that all states $\boldsymbol{x}$ can be reached, independent from the initial state $\boldsymbol{x}_0$. This property is called Reachability and only concerns the state variables of a LTI system.

Definition 3.3.20 (Reachability). A LTI-system of the form (3.4) is called fully reachable if each state $\boldsymbol{x}(T) \in \mathbb{R}^n$ can be reached for the initial state $\boldsymbol{x}_0 = 0$ and a piecewise steadily input function $\boldsymbol{u}(t)$, whereby $0 \leq t \leq T$.

This property can be verified by the matrices A and B which define the state of the system.

Theorem 3.3.21. A LTI-system of the form (3.4) is fully reachable if and only if the reachability matrix

$$\mathcal{R}(A, B) = \begin{pmatrix} B & AB & A^2B & \dots & A^{n-1}B \end{pmatrix}$$

has full rank.

Secondly, the ability to calculate the initial state $\boldsymbol{x}_0$ only with the knowledge of the input $\boldsymbol{u}$ and the output $\boldsymbol{y}$ is required to define the control principle. Therefore, the next characteristic, called observability, affects not only the state but also the output.

Definition 3.3.22 (Observability). A system of the form (3.4) is called fully observable if the initial state $\boldsymbol{x}_0$ can be calculated knowing the input $\boldsymbol{u}(t)$, the output $\boldsymbol{y}(t)$ in $0 \leq t \leq T$ and the matrices A, B, C and D.

Again, this property can be checked by matrices, in this case only the matrices A and C are needed.

Theorem 3.3.23. The system of the form (3.4) is fully observable if and only if the observability matrix

$$\mathcal{O}(C, A) = \begin{pmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{pmatrix}$$

has full rank n.

Both, the reachability and the observability matrix can be calculated using MATLAB with the command `ctrb` and `obsv`, respectively. Finally, it is possible to define the term controllability.

Definition 3.3.24 (Controllability). A system of the form (3.4) is called fully controllable if for all initial states $\boldsymbol{x}_0$ an input function $\boldsymbol{u}(t)$ exists so that $\boldsymbol{x}(T) = 0$, whereby $0 \leq t \leq T$.

In conclusion, the reachability is equivalent to the controllability for continuous systems which follows the definition. An example will show the calculation of the matrices.

Example 3.3.25. A given transfer function

$$G(s) = \frac{9}{s^2 + s}$$

describing a dynamic system leads to the question how to calculate the system matrix A. Rewriting and expanding the given term allows to express the output Y directly as done in

$$\begin{aligned} \frac{Y}{U} &= \frac{9}{s^2 + s} \\ s^2(\frac{1}{s}Y - Y) &= 9U \\ Y &= \tfrac{1}{s^2}9U - \tfrac{1}{s}Y. \end{aligned}$$

Introducing two new variables given by

$$\begin{aligned} X_1 &:= \tfrac{1}{s}9U, \\ X_2 &:= Y = \tfrac{1}{s}(\tfrac{1}{s}9U - Y) = \tfrac{1}{s}(X_1 - X_2), \end{aligned}$$

enables to rewrite the equation into

$$s\begin{pmatrix} X_1 \\ X_2 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} X_1 \\ X_2 \end{pmatrix} + \begin{pmatrix} 9 \\ 0 \end{pmatrix} U.$$

The application of the inverse Laplacian on this system leads to a first order differential equation system as

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 9 \\ 0 \end{pmatrix} u. \tag{3.32}$$

Calculating the state space representation out of a transfer function is the so-called realisability problem. The equation 3.32 gives the system matrix A and the input matrix B. The output matrix C is given by the transformation

$$\begin{aligned} Y &= \begin{pmatrix} 0 & 1 \end{pmatrix}\begin{pmatrix} X_1 \\ X_2 \end{pmatrix} \\ y &= \begin{pmatrix} 0 & 1 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}. \end{aligned} \tag{3.33}$$

Due to the scalar input and therefore scalar output, the matrices B and C are both vectors in $\mathbb{R}^2$. Equation (3.33) directly shows that the feedthrough matrix $D = 0$. Knowing these matrices gives the possibility to analyse the system for reachability and observability. The reachability matrix is calculated by

$$\mathcal{R}(A,B) = \begin{pmatrix} B & AB \end{pmatrix} = \begin{pmatrix} 0 & 9 \\ 9 & 0 \end{pmatrix},$$

which has full rank. Analogously the observability matrix is given by

$$\mathcal{O}(C,A) = \begin{pmatrix} C \\ AC \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & -1 \end{pmatrix},$$

which has again full rank. This proves the controllability of the system.

3.3.5. Control with one Degree of Freedom

Describing the closed loop system, more precisely the plant and the controller by transfer functions leads to a clear characterisation of designs for closed loop systems. This implies the consideration of linear systems only. In the following, three different designs are presented.
The most simple design of a control circuit consists of one controller and a plant. This concept was already introduced in the beginning of chapter 3.2 and the diagram can be seen in Figure 3.2. The transfer function describing the output $\boldsymbol{y}$ for the reference signal $\boldsymbol{r}$ can be calculated analgously to (3.15)

$$T_{\boldsymbol{r},\boldsymbol{y}}(s) = \frac{\boldsymbol{Y}(s)}{\boldsymbol{R}(s)} = \frac{C(s)G(s)}{1 + C(s)G(s)}, \tag{3.34}$$

whereby the transfer function for the controller is C and the transfer function describing the plant is G.
The implementation of a control circle with one degree of freedom is shown in the next

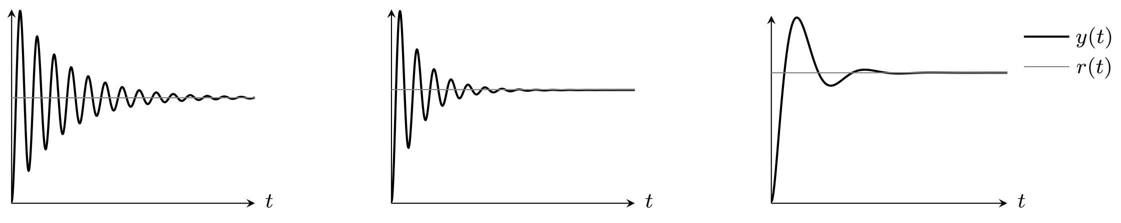
Example 3.3.26. Consider a continuous system described by

$$G(s) = \frac{9}{s^2 + s},$$

where $G(s)$ describes the dynamic behaviour of the plant. For this system, various controllers are implemented in Simulink, following the design of a controller with one degree of freedom. A constant signal was chosen as reference signal for this closed feedback loop. First, a proportional transfer function $C(s) = 9$ is investigated. In Figure 3.7(a) the results for this simulation are depicted. In the beginning of the simulation, an oscillation of the dynamic system can be observed which is a common reaction of a dynamic system in a closed feedback loop. After the excitation time, the system strives to the reference signal. Further, a continuous controller of the form

$$C(s) = \tfrac{9s+1}{s+1}$$

is considered. In Figure 3.7(b) the results for this more complex controller are shown and illustrate the same dynamic behaviour as for the simple controller. The only difference is the shorter excitation time, i.e. the controller is faster in regulating the system. Finally, a PID element as controller was investigated. The three corresponding constants are chosen by $C_P = 0.7, C_I = 0.5, C_D = 0.17$ and the time constant for the differentiator is $T = \frac{1}{10}$. As presented in Figure 3.7(c) the controller again reacts faster than before. It only needs two flashovers until the output of the dynamic system follows the reference signal.



(a) $C(s) = C$. (b) $C(s) = \frac{T_1 s+1}{T_2 s+1}$. (c) PID.

Figure 3.7.: Control with one degree of freedom.

3.3.6. Control with two Degrees of Freedom

In the next step, instead of one, two controllers will be introduced in the closed loop to create a more flexible and adaptable design. The output $\boldsymbol{y}$ of the plant is fed back multiplied by one transfer function C and is compared with the reference signal $\boldsymbol{r}$ multiplied by another transfer function V. The two transfer functions C and V represent the controller elements. This control design is illustrated in Figure 3.8. The transfer function for the output $\boldsymbol{y}$ with the input $\boldsymbol{r}$ is similar to the one with

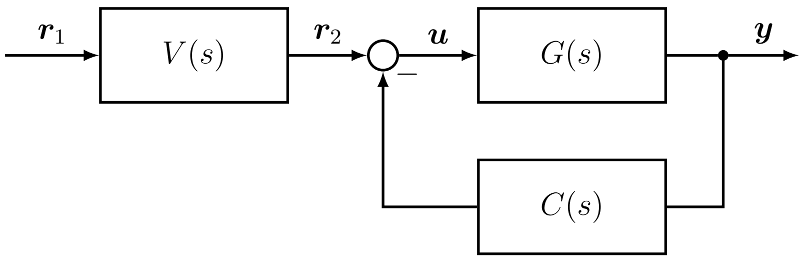


Figure 3.8.: Block diagram of a control circuit with two degrees of freedom.

one degree of freedom in equation (3.34). The only difference is the modification of the reference signal with the help of one transfer function.

$$T_{\boldsymbol{r},\boldsymbol{y}}(s) = \frac{\boldsymbol{Y}(s)}{\boldsymbol{R}(s)} = \frac{V(s)G(s)}{1 + C(s)G(s)} \tag{3.35}$$

Again, one example illustrates the implementation of this control design.

Example 3.3.27. The dynamic system in Example 3.3.26 is extended to a control with two degrees of freedom. The prefilter V is considered in all investigated examples as

$$V(s) = \tfrac{9s+1}{s+1}.$$

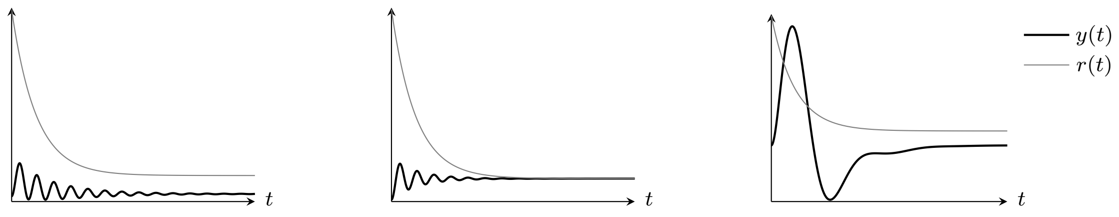
Different elements are analysed for the controller C. First, a proportional element is considered as controller. The results of the simulation can be seen in Figure 3.9(a). Since the constant reference signal r_1, which is a scalar function, is now a continuous decreasing function $R_2(s) = R_1(s)V(s)$ due to the influence of the prefilter V. The proportional controller adapts the scalar output y to follow the difference between the reference signal and the feedback $C(s)Y(s)$. Second, a continuous controller of the form

$$C(s) = \tfrac{9s+1}{s+1}$$

is considered. The results are plotted in Figure 3.9(b). The more complex structure of the controller leads to a faster adaptation to the desired signal. It is interesting to remark that the output follows the reference signal and not the difference between reference and output $e = y - r$. This results due to the fact that the sum $V(s) - C(s)Y(s)$ is set to 0 because of $V = C$. Thirdly, the PID element is investigated. Once again, this more complex structure of the controller leads to a fast adaptation of the system to the reference signal as it can be seen in Figure 3.9(c).

3.3.7. Cascade Control

Another way of designing a control loop would be the connection of two closed loop systems as shown in Figure 3.10. This combination of an inner closed loop system to an outer loop allows to calculate an appropriate controller C_2 after fitting the inner controller C_1.



(a) $C(s) = C$. (b) $C(s) = \frac{T_1 s+1}{T_2 s+1}$. (c) PID.

Figure 3.9.: Control with two degrees of freedom.

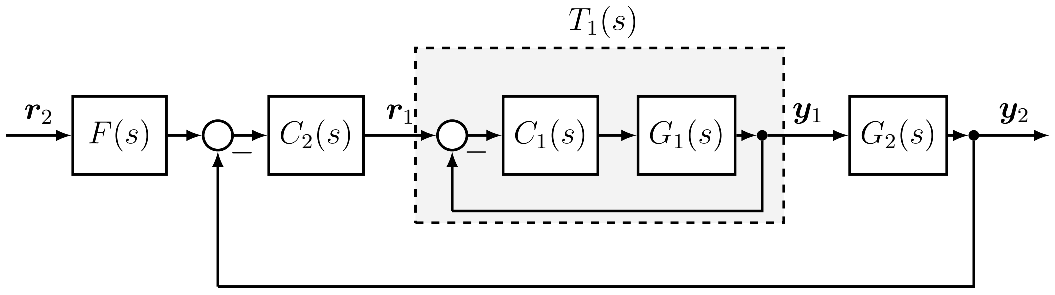


Figure 3.10.: Block diagram for cascade control.

The transfer function of the inner circuit, marked in Figure 3.10 as T_1, is calculated as in (3.34) and therefore given by

$$T_1(s) = T_{\boldsymbol{r}_1,\boldsymbol{y}_1}(s) = \frac{\boldsymbol{Y}_1}{\boldsymbol{R}_1} = \frac{C_1(s)G_1(s)}{1 + C_1(s)G_1(s)}.$$

The transfer function for the entire circle can now be calculated by expressing the output $\boldsymbol{y}_2$ as follows

$$\begin{aligned}\boldsymbol{R}_1(s) &= (\boldsymbol{R}_2(s)F(s) - \boldsymbol{Y}_2(s))C_2(s)\\ \boldsymbol{Y}_2(s) &= (\boldsymbol{R}_2(s)F(s) - \boldsymbol{Y}_2(s))C_2(s)T_1(s)G_2(s)\\ T_{\boldsymbol{r}_2,\boldsymbol{y}_2}(s) = \frac{\boldsymbol{Y}_2(s)}{\boldsymbol{R}_2(s)} &= \frac{1 + C_2(s)T_1(s)G_2(s)}{F(s)C_2(s)T_1(s)G_2(s)}.\end{aligned} \tag{3.36}$$

Example 3.3.28. For the design of the cascade control a dynamic system is considered using two transfer functions $G_1(s) = 9$ and $G_2(s) = \frac{1}{s^2+s}$. The prefilter and the controllers are considered as $F(s) = C_1(s) = C_2(s) = \frac{9s+1}{s+1}$. As it can be seen in Figure 3.11, the resulting controller of the whole circuit acts very fast and leads the output of the dynamic system to the reference signal again.

3.3.8. Full State Feedback Control

Another possibility for the design of a control circuit is the full state feedback which directly implies an influence on the state variables. This combines two aims for the

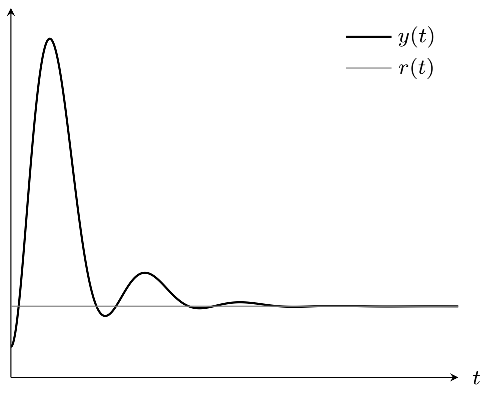


Figure 3.11.: Cascade control for transfer functions.

resulting closed loop. First, it is possible to stabilise an instable system by replacing the eigenvalues of the state matrix. Second, it is again possible to control the limit of the output to follow a reference signal. In the following, LTI systems of the form

$$\dot{\boldsymbol{x}} = A\boldsymbol{x} + B\boldsymbol{u}, \tag{3.37a}$$

$$\boldsymbol{y} = \boldsymbol{x}, \tag{3.37b}$$

are considered. This form guarantees the output of all state variables and therefore the possibility to feedback and influence all state variables. This implies an output matrix $C = I_n$. Introducing the input $\boldsymbol{u} = K\boldsymbol{x} + L\boldsymbol{r}$ with the reference signal $\boldsymbol{r} \in \mathbb{R}^p$ and the matrices $K \in \mathbb{R}^{m \times n}, L \in \mathbb{R}^{m \times p}$, system (3.37) can be written as

$$\dot{\boldsymbol{x}} = (A - BK)\boldsymbol{x} + BL\boldsymbol{r}, \tag{3.38a}$$

$$\boldsymbol{y} = \boldsymbol{x}. \tag{3.38b}$$

The corresponding block diagram is illustrated in Figure 3.12. Let the system

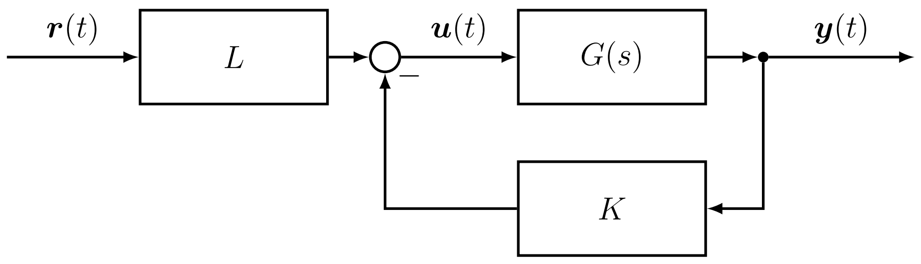


Figure 3.12.: Block diagram of full state feedback.

(3.37) be instable which implies a matrix A with eigenvalues having positive real part. In order to calculate the eigenvalues of the new system (3.38) the characteristic polynomial

$$\chi = \det(\lambda I_n - (A - BK)) = \prod_{j=1}^{n} (\lambda - \lambda_j)$$

with the desired eigenvalues λ_j has to be solved. This leads to n equations. In the case of a SISO system, K is a vector with n entries and therefore, a solution can be calculated, if the system is fully reachable. In the case of a MIMO system, the equation system has no unique solution because $K \in \mathbb{R}^{m \times n}$ and therefore the system is under-determined. For the calculation of the gain L, the equation

$$\lim_{t\to\infty} \boldsymbol{y}(t) = \lim_{t\to\infty} \boldsymbol{x}(t)$$

is considered. The gain L should be calculated in that way, that

$$\lim_{t\to\infty} \boldsymbol{y}(t) = \lim_{t\to\infty} \boldsymbol{r}(t) = \boldsymbol{r}_0$$

is fulfilled. This includes, that $\lim\limits_{t\to\infty} \boldsymbol{x}(t) = \boldsymbol{x}_E$ is a constant value and therefore $\lim\limits_{t\to\infty} \dot{\boldsymbol{x}}(t) = 0$. This allows to express $\lim\limits_{t\to\infty} \boldsymbol{x}(t)$ following the equation (3.38) as

$$\begin{aligned} 0 &= (A - BK)\boldsymbol{x}_E + BL\boldsymbol{r}_0 \\ BL\boldsymbol{r}_0 &= -(A - BK)\boldsymbol{x}_E. \end{aligned}$$

This shows that the calculation of L depends on the matrix K and is therefore not unique, especially for MIMO systems. An example demonstrates the implementation of the full state feedback. Of course, it is possible to design the full state feedback for systems where the output matrix is not the identity matrix but then it is required to introduce state observer which are not considered in this work.

Example 3.3.29. In Example 3.3.25 the state space representation for a given transfer function was calculated. For this state space, a full state feedback is implemented which implies an adaptation of the given LTI system in order to have a output vector $\boldsymbol{y}$ with all state space variables $\boldsymbol{x}$. This leads to the output matrix $C = I_2$ and further to the LTI system

$$\begin{aligned} \begin{pmatrix} \dot{x_1} \\ \dot{x_2} \end{pmatrix} &= \begin{pmatrix} 0 & 0 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 9 \\ 0 \end{pmatrix} u \\ \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \end{aligned}$$

with the eigenvalues $\lambda_{1,2} = \{-1, 0\}$ of the matrix A. Introducing the vector $K = \begin{pmatrix} 1, & 1 \end{pmatrix}$ leads to the eigenvalues $\lambda_{1,2} = \{-7.6, -2.4\}$ and therefore to a stable system.

3.4. Selected Aspects of nonlinear Control

The theory and applications of control are now extended to nonlinear systems of the form

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}), \quad \boldsymbol{x}(t_0) = \boldsymbol{x}_0 \tag{3.39a}$$

$$\boldsymbol{y} = \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{u}). \tag{3.39b}$$

This section is based on [1] and [19].

3.4.1. Linearisation

Controlling nonlinear systems has various approaches. Often, it is sufficient to consider a control circuit with a constrained model only for the plant. Hence, the nonlinear system can be linearised and afterwards the well-known approaches of controlling linear systems can be applied. There exist two different methods for the linearisation of a nonlinear system. First, the linearisation in the neighbourhood of an equilibrium state $\boldsymbol{x}_E, \boldsymbol{y}_E$ is introduced.

Theorem 3.4.1 (Linearisation at an equilibrium state). Considering a time-invariant nonlinear system of the form 3.39a with an equilibrium state at $\boldsymbol{x}_E, \boldsymbol{y}_E$. Small changes $\Delta\boldsymbol{x} = \boldsymbol{x} - \boldsymbol{x}_E$ of the solution of this system can be written as a linear time-invariant system of the form

$$\begin{aligned} \Delta\boldsymbol{x} &= A\Delta\boldsymbol{x} + B\Delta\boldsymbol{u}, \quad \Delta\boldsymbol{x}(t_0) = \Delta\boldsymbol{x}_0 = \boldsymbol{x}_0 - \boldsymbol{x}_E, \\ \Delta\boldsymbol{y} &= C\Delta\boldsymbol{x} + D\Delta\boldsymbol{u}, \end{aligned}$$

whereby the matrices are calculated by the corresponding Jacobi matrices

$$\begin{aligned} A &= \frac{\partial}{\partial\boldsymbol{x}}\boldsymbol{f}(\boldsymbol{x}_E, \boldsymbol{u}_R), \quad B = \frac{\partial}{\partial\boldsymbol{u}}\boldsymbol{f}(\boldsymbol{x}_E, \boldsymbol{u}_R), \\ C &= \frac{\partial}{\partial\boldsymbol{x}}\boldsymbol{h}(\boldsymbol{x}_E, \boldsymbol{u}_R), \quad D = \frac{\partial}{\partial\boldsymbol{u}}\boldsymbol{h}(\boldsymbol{x}_E, \boldsymbol{u}_R). \end{aligned}$$

The following example illustrates the linearisation at an equilibrium state.

Example 3.4.2.

(a) Consider the nonlinear system describing a double pendulum as it is depicted in Figure 3.13. Using Lagrangian mechanics allows us to derive a system of

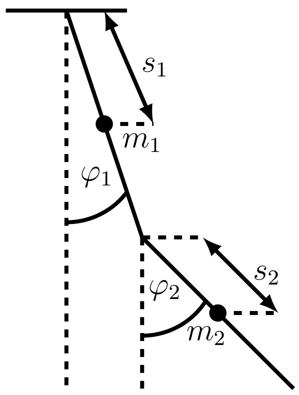


Figure 3.13.: Illustration for the double pendulum.

nonlinear differential equations for the two angles φ_1, φ_2 of the pendulum. This leads to a system

$$M\begin{pmatrix}\ddot{\varphi}_1\\ \ddot{\varphi}_2\end{pmatrix} = \boldsymbol{b}.$$

The mass matrix M and the right hand side $\boldsymbol{b}$ are given by

$$M(\varphi_1,\varphi_2) = \begin{pmatrix} \left(\frac{s_1}{l_1}\right)^2 + \frac{J_1}{m_1 l_1^2} + \frac{m_2}{m_1} & \frac{m_2 s_2}{m_1 l_1}\cos(\varphi_1-\varphi_2) \\ \frac{m_2 s_2}{m_1 l_1}\cos(\varphi_1-\varphi_2) & \frac{m_2}{m_1}\left(\frac{s_1}{l_1}\right)^2 + \frac{J_2}{m_1 l_1^2} \end{pmatrix},$$

$$\boldsymbol{b} = \begin{pmatrix} -\frac{m_2 s_2}{m_1 l_1}\dot\varphi_2^2 \sin(\varphi_1-\varphi_2) - \left(\frac{m_2}{m_1}+\frac{s_1}{l_1}\right)\frac{g}{l_1}\sin\varphi_1 \\ \frac{m_2 s_2}{m_1 l_1}\dot\varphi_1^2 \sin(\varphi_1-\varphi_2) - \frac{m_2 s_2 g}{m_1 l_1^2}\sin\varphi_2 \end{pmatrix}.$$

The parameters $m_1 = m_2 = 0.0295$ kg stand for the mass of the pendulums, $l_1 = l_2 = 0.2$ m represent their lengths, $s_1 = s_2 = 0.1$ m are the centroidal distances of each pendulum, $J_1 = J_2 = 9.83 \cdot 10^{-5}$ kg m^2 are the moments of inertia and g the gravitation constant with 9.81 ms^{-2}. This system has the equilibrium state $(\varphi_1,\varphi_2) = (0,0)$ because the right hand side equals 0. For the linearisation of the system one function, describing the dynamics, is required. Therefore, the matrix M is inverted and multiplied with the vector $\boldsymbol{b}$. This leads to the form

$$\begin{pmatrix} \ddot\varphi_1 \\ \ddot\varphi_2 \end{pmatrix} = \boldsymbol{f}(\varphi_1,\dot\varphi_1,\varphi_2,\dot\varphi_2).$$

In order to get a first order differential equation system the state variables are chosen by the angles and their derivatives. Hence, the state vector is given by

$$\boldsymbol{x} = \begin{pmatrix} \varphi_1 \\ \dot\varphi_1 \\ \varphi_2 \\ \dot\varphi_2 \end{pmatrix}.$$

Calculating the Jacobian of M^{-1} and substituting the parameters leads to a LTI system with the system and output matrix

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ -126.12 & 0 & 63.06 & 0 \\ 0 & 0 & 0 & 1 \\ 189.19 & 0 & -168.17 & 0 \end{pmatrix}, \quad C = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Hence, there is no input acting on this system, the matrices B and D do not exist.

(b) Since the state space representation is not unique, it is possible to find additional representations. Due to the fact that there is no acting input and no output in this system, the state space representation for the double pendulum in the previous example is not suitable for control. Considering the angle φ_1

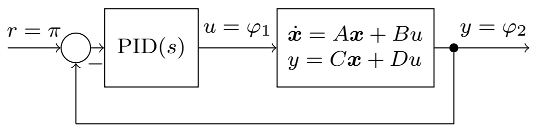
as input and φ_2 as output, allows to linearise the system once again around the equilibrium state $(\varphi_1, \varphi_2) = (0, 0)$. The state matrix A is calculated by the multiplication of the Jacobian of the inverted mass matrix M^{-1} in respect to $\dot{\varphi}_1, \varphi_2, \dot{\varphi}_2$ and the vector $\boldsymbol{b}$. The Jacobian of the right hand side vector $\boldsymbol{b}$ for the input φ_1 leads to the input matrix B.

$$A = \begin{pmatrix} 0 & 63.06 & 0 \\ 0 & 0 & 1 \\ 0 & -168.17 & 0 \end{pmatrix}, \quad B = \begin{pmatrix} -126.12 \\ 0 \\ 189.19 \end{pmatrix}.$$

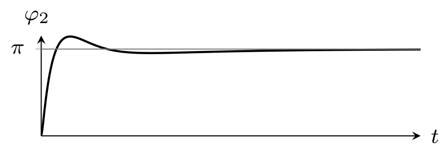
The given output function $y = \varphi_2$ permits the calculation of the output and the feed–through matrices by the Jacobians. This leads to

$$C = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \quad D = \begin{pmatrix} 0 \end{pmatrix}.$$

This state space representation allows the design of a closed control circle with a reference signal for φ_2 and the controller calculating the input φ_1. The constant value $\varphi_2 = \pi$ is considered as reference signal and a PID control design is chosen. The block diagram for the control design is illustrated in


$r = \pi$
$-$
PID(s)
$u = \varphi_1$
$\dot{\boldsymbol{x}} = A\boldsymbol{x} + Bu$
$y = C\boldsymbol{x} + Du$
$y = \varphi_2$

(a) PID control design for the double pendulum.



(b) Output φ_2 following the reference signal π.

Figure 3.14.: Control design and simulation results of the linearised double pendulum.

Figure 3.14(a) and the results of the simulation are depicted in Figure 3.14(b). Here, the plot shows the reference signal and the output of the angle φ_2. It can be seen that, after the excitation, the PID controller calculates the required input resulting in the equilibrium state for the angle φ_2.

Further, the linearisation of a nonlinear system at a trajectory $\boldsymbol{\varphi}$ is defined.

Theorem 3.4.3 (Linearisation at a trajectory). Consider a time-invariant nonlinear system of the form (3.39a) with a trajectory $\boldsymbol{\varphi}(t)$ for an input $\boldsymbol{u} = \tilde{\boldsymbol{u}}(t)$. Small changes $\Delta\boldsymbol{x}(t), \Delta\boldsymbol{y}(t)$ in the solution of this system can be written as a linear time-variant system of the form

$$\begin{aligned} \Delta\boldsymbol{x} &= A(t)\Delta\boldsymbol{x} + B(t)\Delta\boldsymbol{u}, \quad \Delta\boldsymbol{x}(t_0) = \Delta\boldsymbol{x}_0 = \boldsymbol{x}_0 - \boldsymbol{x}_E, \\ \Delta\boldsymbol{y} &= C(t)\Delta\boldsymbol{x} + D(t)\Delta\boldsymbol{u}, \end{aligned}$$

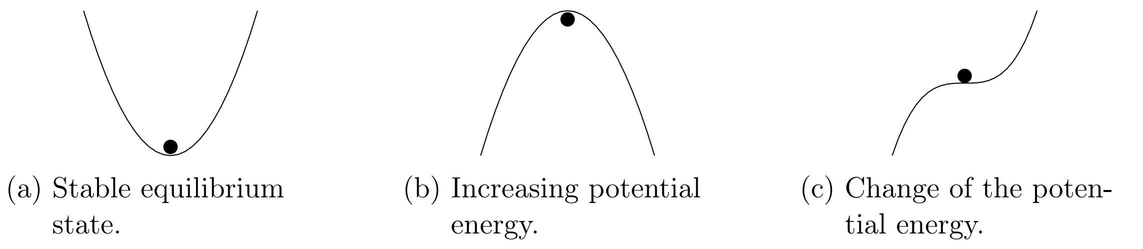
whereby the matrices are calculated by the corresponding Jacobi matrices

$$\begin{aligned} A(t) &= \frac{\partial}{\partial \boldsymbol{x}} \boldsymbol{f}(\boldsymbol{\varphi}, \tilde{\boldsymbol{u}}), \quad B(t) = \frac{\partial}{\partial \boldsymbol{u}} \boldsymbol{f}(\boldsymbol{\varphi}, \tilde{\boldsymbol{u}}) \\ C(t) &= \frac{\partial}{\partial \boldsymbol{x}} \boldsymbol{h}(\boldsymbol{\varphi}, \tilde{\boldsymbol{u}}), \quad D(t) = \frac{\partial}{\partial \boldsymbol{u}} \boldsymbol{h}(\boldsymbol{\varphi}, \tilde{\boldsymbol{u}}). \end{aligned}$$

The technique of linearisation enables the application of the linear control theory for nonlinear systems. Of course, this is only valid for the neighbourhood of an equilibrium state or a trajectory. Therefore, it is necessary to obtain the nonlinear control theory.

3.4.2. The Lyapunov Theory

Lyapunov theory is used commonly in control theory of nonlinear systems. This theory provides methods for analysing the stability of dynamic nonlinear systems.


(a) Stable equilibrium state.

(b) Increasing potential energy.

(c) Change of the potential energy.

Figure 3.15.: Sphere in a gravitational field for the motivation of Lyapunov functions.

Analogously to linear systems, an equilibrium state of a nonlinear system as in (3.2) is given, if $\boldsymbol{f}(\boldsymbol{x}_E) = 0, \quad \boldsymbol{x}_E \in \mathbb{R}^n$ is fulfilled. An illustration for the criteria needed for the stability of equilibrium states of nonlinear systems is given in Figure 3.15. Consider a sphere in a gravitational field, as it is illustrated in Figure 3.15. The equilibrium state of any system is only stable if the potential energy has a minimum in the equilibrium state and decreases or remains constant along the trajectories around the equilibrium state. This case is illustrated in Figure 3.15(a). The other two scenarios in Figures 3.15(b) and 3.15(c) show an increasing potential energy around the equilibrium state or a change of the behaviour of the energy, respectively. The analysis of nonlinear systems leads to different definitions for the stability of equilibrium states. In the following, the definition for the stability in the sense of Lyapunov and asymptotically stable are given.

Definition 3.4.4 (Stability for nonlinear systems). Consider a dynamic system of the form $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$ with an equilibrium state $\boldsymbol{x}_E$. The equilibrium state $\boldsymbol{x}_E$ is called **Lyapunov stable** if for all $\varepsilon > 0$ exists $\delta(\varepsilon) > 0$ such that

$$\|\boldsymbol{x}_0 - \boldsymbol{x}_E\| < \delta \quad \Longrightarrow \quad \|\boldsymbol{x}(t) - \boldsymbol{x}_E\| < \varepsilon \tag{3.40}$$

for all $t > 0$. Furthermore, the equilibrium state $\boldsymbol{x}_E$ is **attractive** if $\zeta > 0$ exists such that

$$\|\boldsymbol{x}_0 - \boldsymbol{x}_E\| < \zeta \quad \Longrightarrow \quad \lim_{t\to\infty} \|\boldsymbol{x}(t) - \boldsymbol{x}_E\| = 0. \tag{3.41}$$

Finally, the equilibrium state $\boldsymbol{x}_E$ is **asymptotically stable** if it is Lyapunov stable and attractive.

Since the verification of stability for an equilibrium state for a nonlinear system is often complex, it is easier to check it by the use of a Lyapunov function. This theory is motivated by energy dissipation in physical processes. Even if it is formulated for the equilibrium state $\boldsymbol{x}_E = \mathbf{0}$, it is valid for all equilibrium states $\boldsymbol{x}_E \neq 0$, which can be easily proven by the state transformation $\tilde{\boldsymbol{x}} = \boldsymbol{x} - \boldsymbol{x}_E \Rightarrow \tilde{\boldsymbol{x}}_E = 0$.

Theorem 3.4.5 (Lyapunov's direct method). Given the autonomous nonlinear system $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$ with the equilibrium state $\boldsymbol{x}_E = \mathbf{0}$ and an open neighbourhood $\mathcal{D} \subset \mathbb{R}^n$ of 0. If there exists a function $V(\boldsymbol{x})\colon \mathcal{D} \to \mathbb{R}$, fulfilling the conditions

(i) $V(\boldsymbol{x})$ is continuously differentiable,

(ii) $V(0) = 0$,

(iii) $V(\boldsymbol{x}) > 0$ for all $\boldsymbol{x} \neq 0$,

(iv) $\dot{V}(\boldsymbol{x}) \leq 0$ for all $\boldsymbol{x} \neq 0$,

then the equilibrium state $\boldsymbol{x}_E = \mathbf{0}$ is Lyapunov stable. If (iv) is replaced by

(v) $\dot{V}(\boldsymbol{x}) < 0$ for all $\boldsymbol{x} \neq 0$,

the equilibrium state $\boldsymbol{x}_E = \mathbf{0}$ is asymptotically stable. The function $V(\boldsymbol{x})$ is called Lyapunov function.

This theorem shows that Lyapunov functions are positive definite and their derivation $\dot{V}$ negative semi-definite or even negative definite in the case of asymptotic stability. The second and third points ensure that the function V has a minimum in $\boldsymbol{x} = \mathbf{0}$ and the fourth condition satisfies that the function V decreases or stays constant.
One consequence of these definitions is to specify the domain where the equilibrium state is stable.

Definition 3.4.6 (Positive invariant set). Let $\boldsymbol{x}_E = \mathbf{0}$ be an asymptotically stable equilibrium state of the autonomous system $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$. The set

$$\mathcal{P} = \{\boldsymbol{x}_0 \in \mathbb{R}^n\colon \lim_{t\to\infty} \boldsymbol{x}(t) = \boldsymbol{x}_E\}$$

is called the positive invariant set of $\boldsymbol{x}_E$. If $\mathcal{P} = \mathbb{R}^n$ holds, then the equilibrium state $\boldsymbol{x}_E$ is global asymptotically stable.

As before, this domain can be expressed with the help of Lyapunov functions.

Theorem 3.4.7 (Positive invariant set). Is $V(\boldsymbol{x})$ a Lyapunov function of the autonomous system $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$ with the equilibrium state $\boldsymbol{x}_E = \boldsymbol{0}$. Under the condition that the set

$$\mathcal{P} = \{\boldsymbol{x} \in \mathbb{R}^n \colon V(\boldsymbol{x}) < c\},$$

is bounded and for all $\boldsymbol{x} \in \mathcal{P}$ holds that

$$\dot{V}(\boldsymbol{x}) < 0,$$

then $\mathcal{P}$ is a positive invariant set of the equilibrium state $\boldsymbol{x}_E$.

To verify the global asymptotic stability, an additional definition is needed.

Definition 3.4.8 (Radially unbounded). A function $f(\boldsymbol{x})$ is called radially unbounded if

$$\lim_{\|x\| \to \infty} f(\boldsymbol{x}) = \infty. \tag{3.42}$$

In conclusion, this can be formulated to

Theorem 3.4.9 (Global asymptotic stability). Given the autonomous system $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$ with the equilibrium state $\boldsymbol{x}_E = \boldsymbol{0}$. If a function $V(\boldsymbol{x}) \colon \mathbb{R}^n \to \mathbb{R}$ exists therefore, that $V(\boldsymbol{x})$ is positive definite and radially unbounded and $\dot{V}(\boldsymbol{x})$ is negative definite, then the equilibrium state $\boldsymbol{x}_E = \boldsymbol{0}$ is global asymptotically stable.

This theoretical introduction allows to formulate control laws for nonlinear systems.

3.4.3. PD Control Law

Another control design is introduced which is applicable on systems describing motion of mechanical systems. Mostly, those models are derived following the Newton-Euler equations and Lagrange formalism. In the following, systems of the form

$$M(\boldsymbol{x})\ddot{\boldsymbol{x}} + C(\boldsymbol{x}, \dot{\boldsymbol{x}})\dot{\boldsymbol{x}} + \boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{\tau}, \tag{3.43}$$

are considered. The vector $\boldsymbol{x} \in \mathbb{R}^n$ contains the state variables and $M(\boldsymbol{x}) \in \mathbb{R}^{n \times n}$ is the mass matrix of the system. The matrix $C(\boldsymbol{x}, \dot{\boldsymbol{x}}) \in \mathbb{R}^{n \times n}$ consists of the coriolis and centrifugal terms, the vector field $\boldsymbol{f} \in \mathbb{R}^n$ holds the potential forces and $\boldsymbol{\tau} \in \mathbb{R}^n$ the generalised moments.
A control law should be defined so that a desired constant position of the coordinates $\boldsymbol{x}_d$ is stabilised asymptotically. Consider a control law

$$\boldsymbol{\tau} = K_P(\boldsymbol{x}_d - \boldsymbol{x}) - K_D\dot{\boldsymbol{x}} + \boldsymbol{f}(\boldsymbol{x}), \tag{3.44}$$

with positive definite matrices $K_P, K_D \in \mathbb{R}^n$ so that $\boldsymbol{x}_d$ is an equilibrium state of the closed circle. Due to the fact that the matrix K_P is acting on the state variables as proportional factor and the matrix K_D is acting on the derivatives of the state variables only, this definition resembles the design of a proportional-differentiator. Analysing the new system on stability leads back to the Lyapunov functions. The considered positive definite Lyapunov function is defined by

$$V(\boldsymbol{x}, \dot{\boldsymbol{x}}) = \tfrac{1}{2}\dot{\boldsymbol{x}}^T M(\boldsymbol{x})\dot{\boldsymbol{x}} + \tfrac{1}{2}\boldsymbol{e}_x^T K_P \boldsymbol{e}_x,$$

whereby $\boldsymbol{e}_x = \boldsymbol{x}_d - \boldsymbol{x}$. The derivative w.r.t. time can be calculated by

$$\frac{\mathrm{d}}{\mathrm{d}t}V(\boldsymbol{x}, \dot{\boldsymbol{x}}) = \dot{\boldsymbol{x}}^T M(\boldsymbol{x})\ddot{\boldsymbol{x}} + \tfrac{1}{2}\dot{\boldsymbol{x}}^T \dot{M}(\boldsymbol{x})\dot{\boldsymbol{x}} + \boldsymbol{e}_x^T K_P \dot{\boldsymbol{e}}_x.$$

Applying the control law in the system of differential equations and replacing the mass matrix with $M(\boldsymbol{x})\ddot{\boldsymbol{x}} = K_P(\boldsymbol{x}_d - \boldsymbol{x}) - K_D\dot{\boldsymbol{x}} - C(\boldsymbol{x}, \dot{\boldsymbol{x}})\dot{\boldsymbol{x}}$ leads to

$$\frac{\mathrm{d}}{\mathrm{d}t}V(\boldsymbol{x}, \dot{\boldsymbol{x}}) = \dot{\boldsymbol{x}}^T(K_P(\boldsymbol{x}_d - \boldsymbol{x}) - K_D\dot{\boldsymbol{x}} - C(\boldsymbol{x}, \dot{\boldsymbol{x}})\dot{\boldsymbol{x}}) + \tfrac{1}{2}\dot{\boldsymbol{x}}^T \dot{M}(\boldsymbol{x})\dot{\boldsymbol{x}} + \boldsymbol{e}_x^T K_P \dot{\boldsymbol{e}}_x.$$

Since the derivative of the desired coordinates $\boldsymbol{x}_d$ are constant, the derivative of the difference results in $\dot{\boldsymbol{e}}_x = -\dot{\boldsymbol{x}}$. The derivative of the Lyapunov function is calculated by

$$\frac{\mathrm{d}}{\mathrm{d}t}V(\boldsymbol{x}, \dot{\boldsymbol{x}}) = \underbrace{\dot{\boldsymbol{x}}^T(\tfrac{1}{2}\dot{M}(\boldsymbol{x}) - C(\boldsymbol{x}, \dot{\boldsymbol{x}}))\dot{\boldsymbol{x}}}_{=0} + \underbrace{\dot{\boldsymbol{x}}^T K_P \boldsymbol{e}_x - \boldsymbol{e}_x^T K_P \dot{\boldsymbol{x}}}_{=0} - \dot{\boldsymbol{x}}^T K_D \dot{\boldsymbol{x}},$$

which finally leads to $-\dot{\boldsymbol{x}}^T K_D \dot{\boldsymbol{x}} \leq 0$. For the last step, it is necessary to notice that $\frac{1}{2}\dot{M}(\boldsymbol{x}) - C(\boldsymbol{x}, \dot{\boldsymbol{x}})$ is skew symmetric. Concluding the Lyapunov theory, this control law assures asymptotic stability for the desired position of the coordinates $\boldsymbol{x}_d$. This control law is illustrated in

Example 3.4.10. Expanding Example 3.4.2 leads to the inverted double pendulum on a cart as it is illustrated in Figure 3.16. This system is already in the desired

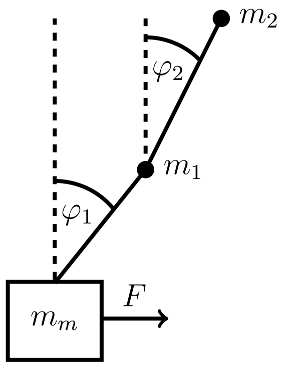


Figure 3.16.: Illustration for the inverted double pendulum.

formulation for the PD control law, given by (3.43). The vector $\boldsymbol{b}$ has to be splitted into the matrix C, containing the coriolis and centrifugal forces, and the vector f holding the potential forces. The vector $\boldsymbol{\tau}$ contains only the external force F acting on the cart. The mass matrix M is the same as in A.2.4. This leads to the final set of equations

$$\begin{aligned}
C &= \begin{pmatrix} 0 & -\left(\frac{m_1}{2}+m_2\right) l_1 \sin\varphi_1 \dot{\varphi}_1 & -\frac{m_2}{2} l_2 \sin\varphi_2 \\ 0 & 0 & -\frac{m_2}{2} l_1 l_2 \sin(\varphi_2-\varphi_1) \\ 0 & -\frac{m_2}{2} l_2 l_1 \sin(\varphi_1-\varphi_2)\dot{\varphi}_1 & 0 \end{pmatrix}, \\
\boldsymbol{f} &= \begin{pmatrix} 0 \\ -\left(\frac{m_1}{2}+m_2\right) g l_1 \sin\varphi_1 \\ -\frac{m_2}{2} l_2 g \sin\varphi_2 \end{pmatrix}, \\
\boldsymbol{\tau} &= \begin{pmatrix} F, & 0, & 0 \end{pmatrix}^T.
\end{aligned}$$

The desired positions of the state variables in the control design are

$$\begin{pmatrix} x, & \varphi_1, & \varphi_2 \end{pmatrix}^T = \begin{pmatrix} 0, & \pi, & \pi \end{pmatrix}^T.$$

Hence, the new input can be defined, following equation (3.44), as

$$\boldsymbol{\tau} = K_P \begin{pmatrix} x-0 \\ \varphi_1-\pi \\ \varphi_2-\pi \end{pmatrix} - K_D \begin{pmatrix} \dot{x} \\ \dot{\varphi}_1 \\ \dot{\varphi}_2 \end{pmatrix} + f \begin{pmatrix} x \\ \varphi_1 \\ \varphi_2 \end{pmatrix},$$

with two positive definite matrices K_P, K_D. Applying this input to the system of the differential equations leads to the form

$$\begin{pmatrix} \ddot{x} \\ \ddot{\varphi}_1 \\ \ddot{\varphi}_2 \end{pmatrix} = M^{-1} \left(K_P \begin{pmatrix} x-0 \\ \varphi_1-\pi \\ \varphi_2-\pi \end{pmatrix} - K_D \begin{pmatrix} \dot{x} \\ \dot{\varphi}_1 \\ \dot{\varphi}_2 \end{pmatrix} - C \begin{pmatrix} \dot{x} \\ \dot{\varphi}_1 \\ \dot{\varphi}_2 \end{pmatrix} \right).$$

The simulation results for the closed loop are plotted in Figure 3.17 including all three state variables x, φ_1, φ_2. The results show the stabilisation of the equilibrium state for all three state variables. After the excitation of the controller according to the system, the pendulums stay in the horizontal position, whereby the cart remains at $x = 0$. The simulation was executed for various matrices in order to find appropriate parameters for a good control. The final matrices are chosen as

$$K_P = \begin{pmatrix} 8 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix}, \quad K_D = \begin{pmatrix} 5 & 0 & 0 \\ 0 & 0.07 & 0 \\ 0 & 0 & 0.07 \end{pmatrix}.$$

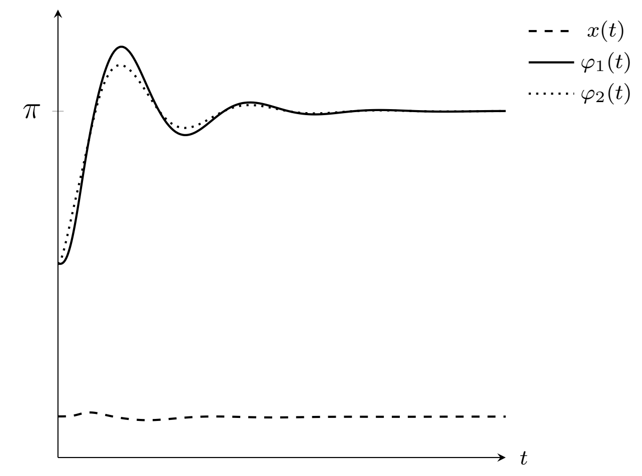


Figure 3.17.: PD control for the inverted double pendulum.

4. Modelling and Simulation of the Human Knee

In the following, four simulation models for the human knee joint are presented. In the last decades, the biomechanical research was focused on the development of valid models for the human knee joint. Many different models were developed using various modelling approaches and software tools. Furthermore, the models vary in complexity concerning their structure and depending on the aim of the model different components are included. The implemented models in this thesis are based on a validated multibody model for the knee joint which is established by biomechanical researchers in [8], [14] and [15] and will be summarised in the following. After that, three simulation models are presented in order to compare different software tools regarding their possibilities. The models use two different modelling approaches. Two multibody models, implemented in different simulation frameworks, are compared against one simulation model based on partial differential equations. In the beginning, some basic anatomical and biomechanical principles of the human knee joint are introduced.

4.1. Anatomical and Biomechanical Basics of the Human Knee Joint

To build a valid model of a system, basic knowledge of the underlying system is required. This section gives an overview about the anatomy of the human knee joint, the kinematics and some biomechanical principles of soft tissues.

4.1.1. Anatomy of the Human Knee Joint

The following description of the anatomy of the human knee joint is based on [27]. The human knee joint contains three bones, femur, tibia and patella. Figure 4.1 shows a sketch of the right human knee. The three bones are depicted and their connections by ligaments and the patellar tendon.
The human knee joint is, apart from the shoulder joint, the most complex joint in the human body. This is caused by various biological structures interacting together and forming a joint having multiple degrees of freedom. Due to the shape of the bones, the knee joint is divided in three sections.

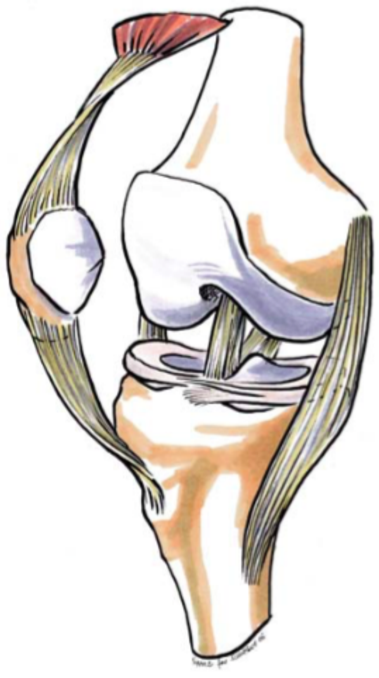


Figure 4.1.: View of a right knee with ligaments, tendon and bones [27].

Those are joint areas connecting

- Medial femur condyle and tibia plateau,
- Lateral femur condyle and tibia plateau,
- Femur and patella.

The contact between femur and tibia is divided due to two eminences at the lower end of the femur, which are both covered by cartilage. They are called condyles. One is situated at the inner side of the body, anatomically called medial and the other at the outer side, lateral. Between the condyles, the crucial ligaments have their attachment points. Analogously, the tibia has three condyles. Two on the sides, covered by cartilage and a third one is raising in between, not covered with cartilage. This eminence builds together with the crucial ligaments a lock. The patella is incorporated in the patellar tendon of the quadriceps, which connects femur and tibia. The quadriceps is located at the front of the femur and applies the force, which is responsible for the extension of the leg. On top of the tibia plateau, two moon shaped cartilages are situated. Those are called menisci and they divide the articular capsule. Synovial fluid in the joint cavity between the bones improves sliding. The menisci fulfil the task as shock absorbers, optimise the rolling-sliding motion and stabilise the joint.
The ligaments play an important role for the position and stabilisation of the joint bodies, as they prevent extreme motions in the knee. The collateral ligaments are

based on the lateral and medial sides of the bones. They have onion-like layers of fasciae and support the joint capsule with their thick structure. The medial collateral ligament arises from the medial femoral condyle and ends at the medial side of the tibia. It supports the joint structure preventing valgus stress, rotational motion in medial direction about the motion axis. The lateral collateral ligament runs from the lateral femoral condyle to the fibula which is situated laterally to the tibia. It prevents before varus stress, rotational motion in lateral direction perpendicular to the joint axis.
The intercondyloid notch is situated between the femoral condyles, where the crucial ligaments extend. The anterior crucial ligament is situated lateral and attaches to the anterior side of the intercondylar area of the tibia. Likewise, the posterior crucial ligament runs at the medial side to the posterior aspect of the intercondylar area of the tibia. Due to this course, the ligaments cross each other remaining distinct. They stabilise the knee in anterior and posterior directions or rotations.
The motion of the joint bodies is actuated by muscles. The quadriceps is a muscle group responsible for the extension of the knee. The muscles are situated at the front side of the femur. The antagonists, the muscles responsible for squatting, are situated on the back of femur and tibia.

4.1.2. Degrees of Freedom in the Human Knee Joint

A more detailed description of the kinematics in the human knee joint is found in [34]. The ligaments and constraints of the geometries determine the movements that take place in the human knee joint. Three principal axes in the knee specify where motion is applied, the tibial shaft axis, the epicondylar axis and the anteroposterior axis. They correspond to the longitudinal, transversal and sagittal axis in the human body. Each axis provides two types of movement, translation and rotation. This results in six degrees of freedom. The movements are referred to three rotations and three translations, which are

1. Flexion and Extension,
2. Valgus and Varus rotation,
3. External and Internal rotation,
4. Anterior and Posterior translation,
5. Medial and Lateral translation,
6. Proximal and Distal translation.

The axes and their corresponding movements are depicted in Figure 4.2. The knee enables flexion of up to $120° - 140°$ actively, but $150°$ passively. In this position, the collateral ligaments are relaxed while the crucial ligaments stay taught. The extension of the knee is possible until $0°$ actively and up to $5° - 10°$ passively. Passive motion is applied by external help, the action is not actuated by own muscles. The collateral ligaments remain taught with the anterior cruciate ligament. Internal and external rotation of the knee are both possible up to $30° - 40°$. In the following, only flexion and extension are considered which is accompanied by proximal and distal translation.

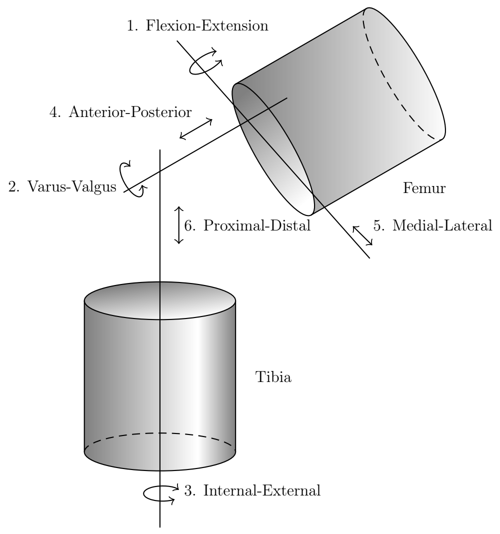


Figure 4.2.: Degrees of freedom and motions in the human knee joint.

4.1.3. Biomechanics of Ligaments and Tendons

Soft tissues, such as ligaments and tendons, possess a complex biomechanical behaviour. They determine the motion in joints and are important for carrying loads. Their characteristics are described in depth by [35] and [25]. In general, the biomechanics of ligaments and tendons are similar, they differ only in detail, as e.g. the percentage of elastin in their composition and the arrangement of collagen fibrils. This distinction is not considered in more detail, nevertheless the description of the biomechanics is valid for both tissues.
To describe mechanical properties of ligaments and tendons, following parameters are used

- Strain: elongation relative to length under no stress,
- Stress: force relative to cross-sectional area,
- Stiffness: relation between deformed length and applied force,
- Modulus: relation between stress and strain.

It is common to analyse the stress-strain behaviour of ligaments and tendons. This means the analysis of applied stress to soft tissue in respect to the resulting strain. A typical stress-strain curve can be seen in Figure 4.3. The slope of the curve represents the modulus. High modulus indicates stiffer tissue.
Three different regions in the stress-strain curve can be distinguished, which are typical for ligaments and tendons. They are called

1. Toe region,
2. Linear region,
3. Yield and failure region.

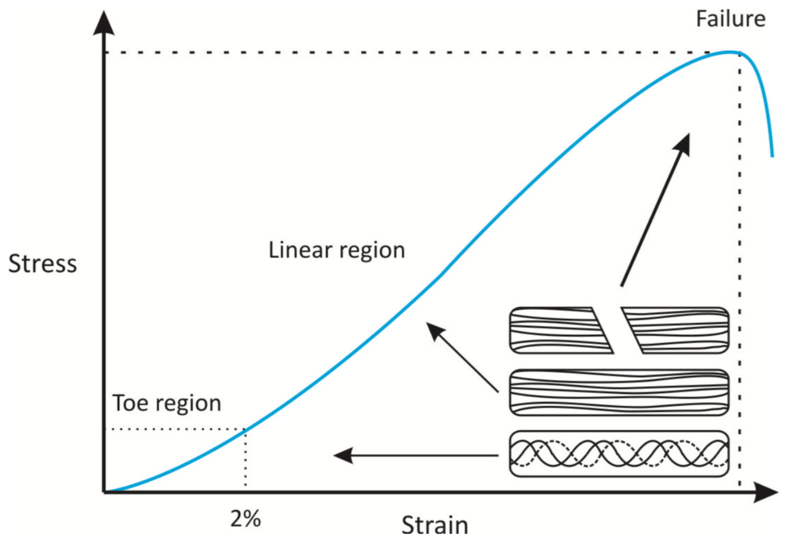


Figure 4.3.: Stress-strain curve for ligaments and tendons [25].

Before ligaments and tendons are strained, the collagen fibrils, forming their structure, are crimped. They start to straighten when force is applied. This leads to a non-linear slope at the beginning of the stress-strain curve, the toe region. When all crimped fibers are straightened, the toe region ends. Normally, this region ends after 2% of strain. Now, the collagen fibers stretch and the ligament or tendon acts as spring with a linear behaviour in the stress-strain curve. This implies a deformation of the soft tissue. Until 4% of strain, the deformation is elastic. This means, the ligament respectively the tendon remains to its normal length after the application of stress. Applying more force leads to micro failures in the structure of the fibrils. This determines the ultimate stress and the corresponding ultimate strain a ligament or tendon can sustain. The yield and failure region shows a reduced stiffness behaviour.
The nonlinear toe-region in the stress-strain curve of ligaments requires a non-linear modelling of the force generated by ligaments. The modelling process is explained in more detail in 4.2.

4.2. Multibody model for the Human Knee Joint implemented in Adams

The platform SimTK [29] offers the possibility for researchers to share their work, as e.g. collected data and developed models. The collection focuses on biomedical models. The website is hosted by the National Institutes of Health in the United States, is free of charge and open to public. Since the repository provides geometries supplementary to mathematical models, the development of new models is facilitated. The available models include various modelling approaches, thus, multibody models

and models described by partial differential equations can be found. Furthermore, the assembly contains models in various simulation environments.
Multibody models of the human knee joint evolved by Guess et al. are available on SimTK and one is used in this thesis for further analysis and as basis for knee flexion simulations in other simulation frameworks. The work of Guess et al. deals with the development of subject specific multibody models of the human knee joint, for details see [8], [14] and [15].
The knee joint model by Guess et al. is implemented in Adams, a multibody dynamics software developed by MSC Software Corporation. It simulates the flexion of the right knee of a 77 year old man. It includes the three main bones of the human knee, femur, tibia and patella. The geometries of the bones are measured by magnetic resonance images of a cadaver knee. Articular cartilage is included in the geometries of the bones. The bones are modelled as rigid bodies which means that they do not show elastic properties. The bodies are described by their geometries and specific parameters, as mass, density, center of mass and rotational inertia. These values are derived by experimental tests and summarised in Table 4.1.

		Coordinates of	
	Mass [kg]	Center of mass [mm]	Inertial rotation [kg mm^2]
Femur	0.327	$\begin{pmatrix}24.47, & 530.92, & 96.305\end{pmatrix}^T$	$\begin{pmatrix}311.74, & 254.45, & 187.89\end{pmatrix}^T$
Tibia	0.227	$\begin{pmatrix}25.52, & 468.32, & 87.06\end{pmatrix}^T$	$\begin{pmatrix}140.86, & 109.29, & 100.52\end{pmatrix}^T$
Patella	0.0352	$\begin{pmatrix}21.1, & 524.89, & 137.33\end{pmatrix}^T$	$\begin{pmatrix}7.12, & 4.85, & 4.02\end{pmatrix}^T$

Table 4.1.: Parameters for bones in the Adams model.

Since this model simulates the flexion without using joints, the movements between the bodies are described by external forces only. Between femur and tibia as well as between femur and patella contact forces act, more precisely between the respective cartilage on the bone surface. This force is modelled as the default Adams compliant contact model based on the Hertz contact law

$$F = k\delta^n + B(\delta)\dot{\delta}$$

with the contact force F. The parameters describe the spring constant k, the compliance exponent n and the damping coefficient $B(\delta)$. The interpenetration between the geometries is δ which qualifies the distance between the geometries, the corresponding velocity is $\dot{\delta}$. The contact parameters are derived after analysis of different methods. A simplified Hertzian contact law, a simplified elastic foundation contact theory and parameter optimisation from a model based on PDEs were used for parameter estimation. The resulting kinematics were compared with measured data

in-vitro kinematics, as is done in [14].
In addition to the bones, soft tissues as the crucial and collateral ligaments as well as the patellar tendon are integrated in the model. The tendon is modelled with four bundles of linear springs. The contact points of the tendon on femur and tibia are extracted from magnetic resonance images. The stiffness and damping parameter for the springs are obtained by testing the cadaver knee in a knee simulator and comparing the results to the model. The ligaments are implemented as bundles of one-dimensional non-linear spring damper elements. The fibers in ligaments change their reaction depending on the stress as it can be read in more detail in section 4.1.3. Unloaded, the fibrils in ligaments are crimped; under stress, the fibrils straighten first. Afterwards, all fibers start to stretch and ligaments behave as linear springs. This results in a typical stress-strain curve of ligaments with a non-linear behaviour. The force f from the ligaments is therefore calculated as

$$f(\varepsilon) = \begin{cases} \frac{1}{4\varepsilon_l} k\varepsilon^2, & 0 \leq \varepsilon \leq 2\varepsilon_l, \\ k(\varepsilon - \varepsilon_l), & \varepsilon > 2\varepsilon_l, \\ 0, & \varepsilon < 0, \end{cases}$$

with the stiffness parameter k and the spring parameter $\varepsilon_l = 0.03$. The strain ε is defined as

$$\varepsilon = \frac{l - l_0}{l_0},$$

with the length of the ligament l and the zero-load length l_0. The derivation of the formulas is done in [32] and [7]. The zero-load length is a sensitive parameter defining the length of a ligament when it becomes taut which differs for each type of ligament. The reference strain method is used to determine the zero-load length. It uses experimental data, obtained by cadaver knees loaded with forces which occur during walk as it is realised in [8]. The force of the crucial ligaments is realised in the model using a `C`-subroutine for the calculation of external forces.
The femur is fixed to the ground, tibia and patella are able to move. Their motion is constrained by the patellar tendon, the ligaments and the contact forces. Applying an external force to the tibia in posterior direction leads to flexion of the knee.

4.3. Reformulation for Multibody libraries

Proceeding from the multibody model for the human knee joint in the simulation environment Adams, a model of the knee flexion is formulated. This model is based on physical and biomechanical properties of the systems which allows to use three different simulation frameworks. In particular, the environments are Simscape and MapleSim which are based on the multibody modelling theory and COMSOL using partial differential equations for the mathematical model.
The requirement using different simulation frameworks which are further based on

various modelling approaches leads to a simplification of the model compared to the one presented above. Describing movements in simulation frameworks containing multibody model libraries or multibody dynamics modules requires the usage of joints. Therefore, it is not possible to use spring damper elements only as in the model presented for the simulation environment Adams.
First, the flexion of the tibia is discussed. As mentioned in section 4.1.2, the knee joint contains of six degrees of freedom. Considering only the motion responsible for flexion and extension simplifies the knee as a revolute joint. As it is stated in more detail in [9], this simplification does not describe the entire knee motion but it is sufficient for load estimations.
Joints are specified by their spring stiffness constant and damping coefficient which limit the movement and influence the velocity. In the revolute joint the spring stiffness signifies the torque which is required to rotate the joint primitive by a unit angle. In terms of biomechanics, the crucial ligaments stabilise the knee in rotation. Since the Adams model contains translational spring stiffness parameters only, these values can not be used. More information about elasticity coefficients in rotation is not available in the data. Therefore, these values are calibrated comparing the output angle between femur and tibia with the output angle of the Adams model. The damping coefficient is the parameter which determines the torque to maintain a constant angular velocity and is transferred directly from the Adams model.
Second, the movement of the patella with respect to the femur and tibia is introduced. In order to simulate the sliding of the patella between the condyles of the femur, a second revolute joint is implemented. The center of rotation for this joint is at the center of mass of the femur. Both joint centers are visualised in Figure 4.4. The point j_1 represents the center of rotation between femur and tibia, j_2 depicts the center of joint between femur and patella. The given geometries determine the global coordinate system. The spring stiffness value and the damping coefficient remain the same values as the parameters describing the patellar tendon in Adams. The connection between tibia and patella can be realised by one spring damper element. This interaction transmits the applied torque from tibia to patella and does not define additional degree of freedom for a movement. This element is added to the bones at the attachments points included in the data of the Adams model. Even the spring stiffness constant and damping coefficient could be carried over from the available data. Each spring damper element is defined additionally by its length which is calculated by the distance between the attachment points. The corresponding parameters are provided in Table 4.2. In Adams, the acting force vector $\boldsymbol{F}$ is applied at the tibia along the direction vector $\boldsymbol{v}_F$. The models, which are used for further investigation, consider a revolute joint with one rotational degree of freedom only. This requires a torque τ acting on this joint in the x-direction which represents the rotational axis. For this calculation, the direction vector $\boldsymbol{r}$ between the application point of the force p_F and the center of joint p_τ, where the torque is

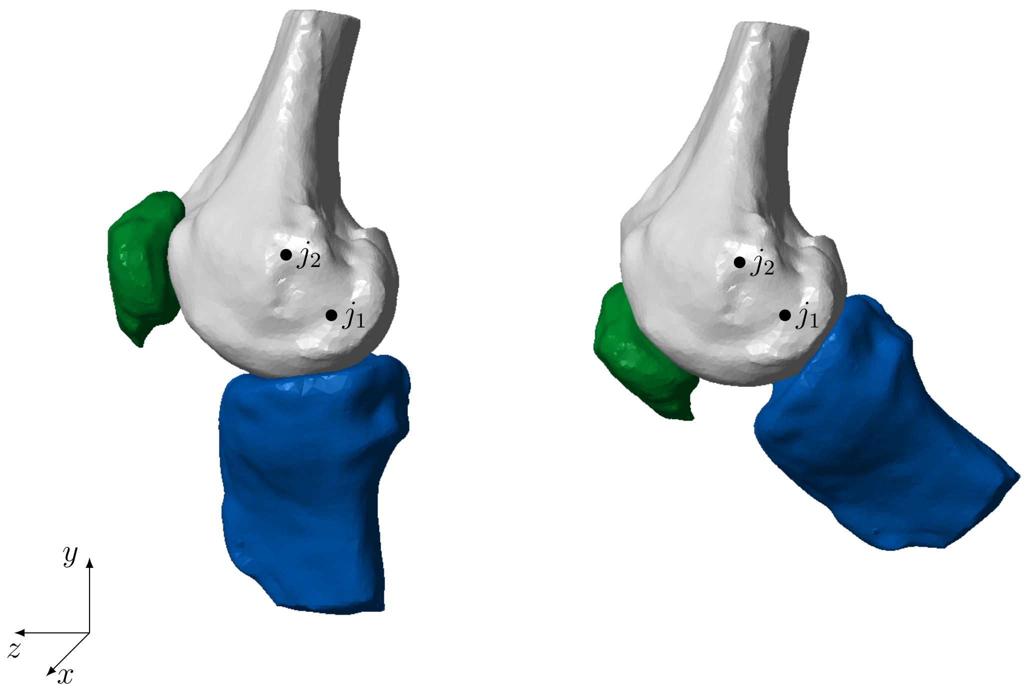


Figure 4.4.: Visualisation of the joint centers in the knee model.

acting, has to be known. It is calculated by

$$\boldsymbol{r} = p_\tau - p_F = \begin{pmatrix} 27.47 \\ 509.62 \\ 86.305 \end{pmatrix} - \begin{pmatrix} 35.58 \\ 368.35 \\ 57.32 \end{pmatrix} = \begin{pmatrix} -8.11 \\ 141.27 \\ 28.98 \end{pmatrix}.$$

In general, the torque is calculated by the crossproduct of the force and the direction vector which results in a torque vector and is therefore not applicable for this model. If the direction vector of the force $\boldsymbol{v}_F$ is orthogonal to $\boldsymbol{r}$, the length of $\boldsymbol{r}$ represents the lever arm. This allows to calculate the torque by multiplying the force and the length of the lever. The direction vector of the force is given by

$$\boldsymbol{v}_F = \begin{pmatrix} -171.12 \\ 7.04 \\ 38.11 \end{pmatrix}.$$

Therefore, the angle α between these vectors results in

$$\alpha = \arccos\left(\frac{\boldsymbol{r} \bullet \boldsymbol{v}_F}{\|\boldsymbol{r}\|_2 \cdot \|\boldsymbol{v}_F\|_2}\right) = 1.7 \text{ rad}.$$

This leads to the calculation of the length l for the lever arm as

$$l = \|\boldsymbol{r}\|_2 \cdot \sin\alpha \approx 144.44 \cdot 0.9 \text{ mm} \approx 143.06 \text{ mm} \approx 0.14 \text{ m}.$$

Parameter	Value
Attachment point Tibia	$\begin{pmatrix}19.4, & 438.2, & 112\end{pmatrix}^T$ mm
Attachment point Patella	$\begin{pmatrix}19.5, & 502.5, & 141.7\end{pmatrix}^T$ mm
Build Length	70.9 mm
Spring stiffness	158 $\frac{\mathrm{N}}{\mathrm{mm}}$
Damping coefficient	1 $\frac{\mathrm{N\,s}}{\mathrm{mm}}$

Table 4.2.: Parameters for the patellar tendon in the Adams model.

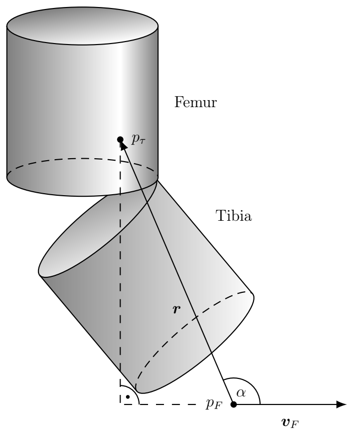


Figure 4.5.: Illustration for the force and torque acting on the human knee.

An illustration for this calculation is depicted in Figure 4.5. Using a sine wave as acting force, the torque is finally calculated by

$$\tau(t) = l \cdot F = 0.14 \cdot 200 \cdot \sin(2\pi \cdot 0.125(t-1)) \cdot \mathrm{H}(t-1) \text{ Nm},$$

dependent on time t. These assumptions allow to implement this model formulation in various simulation environments.

4.3.1. Simulation in Simscape

As the Simscape Multibody library is embedded in Simulink, the setup of the simulation model differs to the construction of a multibody model in the Adams environment. Each component in the model is represented by a block. Figure 4.6 shows the structure of the multibody model of a human knee in Simscape.
Every multibody model contains a solver configuration, a world frame and a mechanism configuration block. The solver block specifies information about solver parameters which are necessary to calculate the solution of the ODEs which describe the

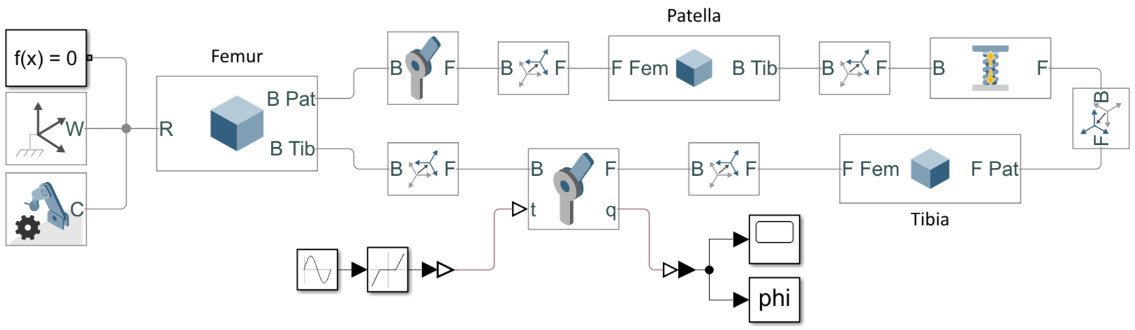


Figure 4.6.: Structure of the multibody model of a human knee in Simscape.

model. The solver type, initialisation options and sample time can be adjusted. More specific informations, such as tolerances of the solver and time steps are changed in the model configuration parameters, similarly to Simulink. The world frame defines the global reference frame of the model. The mechanism configuration block contains main parameters of the model, such as gravity and linearisation delta used during linearisation for computing partial derivatives.
The three bones, femur, tibia and patella are modelled as rigid bodies. Simscape provides the opportunity to import CAD files in the formats `stl` and `step`. The import of `.step`-files has the advantage that the properties of the geometry are calculated automatically by Simscape. The geometries of the bones are provided with the multibody model in Adams from the platform SimTK and are saved as `stl`-files. Therefore, all physical properties, as mass, center of mass and moments of inertia, are entered manually. The data were taken from the Adams multibody model as seen in Table 4.1.
Bodies have frames which act as connection points to different components, e.g. joints, constraints or other bodies. Per default, each body has one reference frame located at the origin of the body. Additional frames at the center of mass are defined for all bones to simplify the connection to each other. The definition of frames allows also to include transformations. Since the revolute joint is acting in Simscape at the z-axis only, the linking frames through the revolute joints contain rotational transformations that allow for a rotation in the desired direction. The location of the revolute joint defines its rotation center. Consequently, rigid transforms between bodies and joints are required as connection points.

	Spring stiffness	Damping coefficient
Revolute joint femur-tibia	$553.5\ \frac{\text{N mm}}{\text{deg}}$	$1\ \frac{\text{N mm s}}{\text{deg}}$
Revolute joint femur-patella	$33\ \frac{\text{N mm}}{\text{deg}}$	$1\ \frac{\text{N mm s}}{\text{deg}}$

Table 4.3.: Parameters for joints in Simscape.

In joint blocks, parameters for stiffness and damping coefficients can be set for each degree of freedom. The spring damper element is defined by the spring stiffness constant, the damping coefficient and the natural length. Simscape allows to adapt the unities in the blocks individually which facilitates the transfer of available parameters. The used parameters are summarised in Table 4.3.
The acting torque is applied to the joint directly as actuation using Simulink blocks, a sine wave and a death zone. The joint blocks allow directly to export the position which describes the angle between the connecting components. This feature enables to plot them in Simulink scopes and export them to the MATLAB workspace where they can be processed further. The Simscape blocks are 3D blocks which produce 3D signals. The Simulink blocks are described in 1D only. Hence, additional blocks are needed to convert the signals.
During simulation, Simscape starts the Mechanics Explorer using MATLAB for visualisation in 3D as it can be seen in Figure 4.7. In the Mechanics Explorer, all rigid bodies are visible with their generated shape. The center of mass and all frames defined in the model can be shown. Furthermore, it is possible to export the animation of a finished simulation.

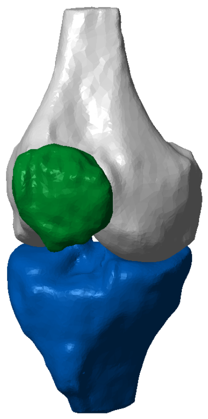


Figure 4.7.: View in the Mechanics Explorer in MATLAB.

4.3.2. Simulation in MapleSim

The simulation environment in MapleSim is similar to Simscape regarding the model setup. They differ in numerical and algebraic aspects as they rely on Maple and MATLAB. Again, the component are represented by blocks, which are nearly coincident with the library in Simscape. Therefore, the model structure in Figure 4.8 strongly resembles the one in Figure 4.6. The main parameters and the solver configuration of the model are defined in the model properties, so that no additional blocks are needed. Again, one reference frame defines the physical origin of the model and is therefore fixed in space. As in Simscape, rigid body frames act as connection points and define the distances between the components.

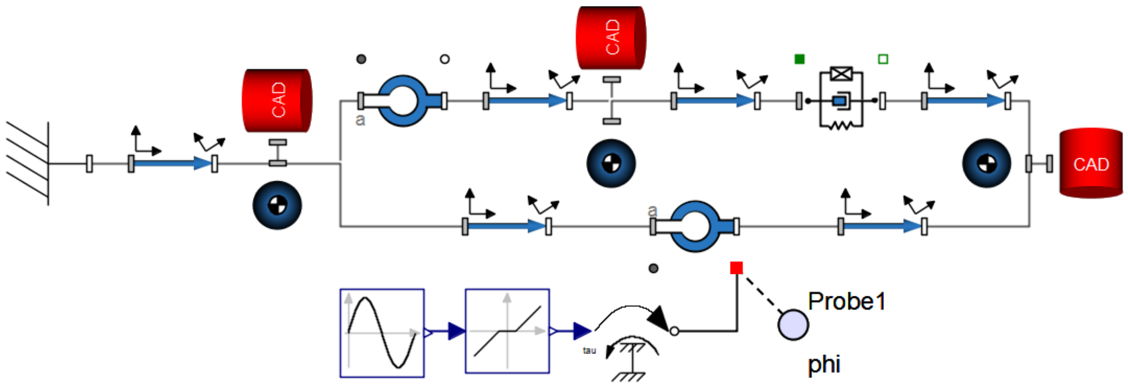


Figure 4.8.: Structure of the multibody model of a human knee in MapleSim.

	Spring stiffness	Damping coefficient
Revolute joint femur-tibia	$31.71\ \frac{\text{N m}}{\text{rad}}$	$0.05\ \frac{\text{N m s}}{\text{rad}}$
Revolute joint femur-patella	$1.89\ \frac{\text{N m}}{\text{rad}}$	$0.05\ \frac{\text{N m s}}{\text{rad}}$
Patellar tendon	$158 \cdot 10^3 \frac{\text{N}}{\text{m}}$	$10^3 \frac{\text{N s}}{\text{m}}$

Table 4.4.: Parameters for joints and patellar tendon in MapleSim.

Rigid bodies are defined by their mass and rotational inertia. Their center of mass is declared by their position. The rotational axis in the revolute joints can be chosen freely. The units for the mechanical parameters can not be changed freely and are therefore converted as it is listed in Table 4.4.
Analogously to Simscape, the torque is directly applied to the revolute joint between femur and tibia directly. In MapleSim, the joint blocks contain two flanges, from which one is used for applying a force and exporting the resulting angle. The torque is composed using Signal Blocks and the 1D-Mechanical torque block, available in the MapleSim Library.
The body shapes are used in MapleSim for visualisation only. Hence, additional blocks, which contain the geometry, are needed. MapleSim provides pre-built geometries, such as blocks, cylinders, spheres, torus, etc. and to import CAD geometries from `stl`-files. The geometry blocks are linked directly to rigid bodies which implies that they undergo the same translational transforms. The definition of a translational offset in the geometry block, allows a relocation of the geometry file. Similar to the Mechanics Explorer in Simscape, MapleSim offers two kinds of graphical 3D representation. First, the 3D workspace shows a rendering of the model which adapts to changes immediately. In addition to the model arrangement by blocks, it is possible to build the model setup in the 3D workspace directly. Second, the 3D animation of the simulation is accessible via the 3D playback window in the

simulation results tab in the analysis window where all results of the simulation are stored.

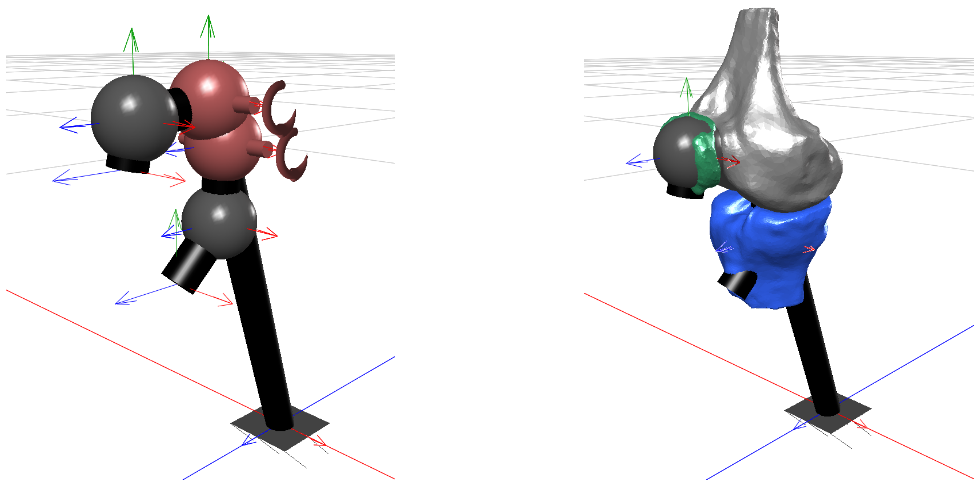


Figure 4.9.: View in the 3D Window and Playback Window in MapleSim.

The left side of Figure 4.9 shows the block components of the model setup in the 3D window, as the rigid bodies, the frames in black and the joints in red. In the playback window on the right side, the corresponding geometries of the rigid bodies are visible. An animation of the simulation is accessible via the playback window. The analysis window offers only the 3D representation of the model with the initial conditions.

4.3.3. Simulation in COMSOL

For the simulation in COMSOL, the Multibody Dynamics module and a time dependent study is chosen. This choice affects the included physics and its components in the model. The simulation environment of COMSOL differs to Simscape and MapleSim as the model elements are not represented by blocks. The model structure in COMSOL resembles a sequence of steps which have to be completed in order to supply all informations required for the simulation as it can be seen on the left side in Figure 4.10.
As COMSOL uses the finite element method to solve the partial differential equations describing the model numerically, the geometries defining the bodies constitute an important part of the model. Dependent on the geometries, a mesh is built to define the points for which the solution is calculated. Apart from building geometries by using shapes, COMSOL offers the possibility to import CAD geometries. Two different approaches of importing geometries are available. Either the geometry component directly from a part or a mesh can be imported and converted afterwards to a geometry part. The latter option was chosen to import the files describing the bones as meshes. Three geometries are built in directly from the meshes. The choice

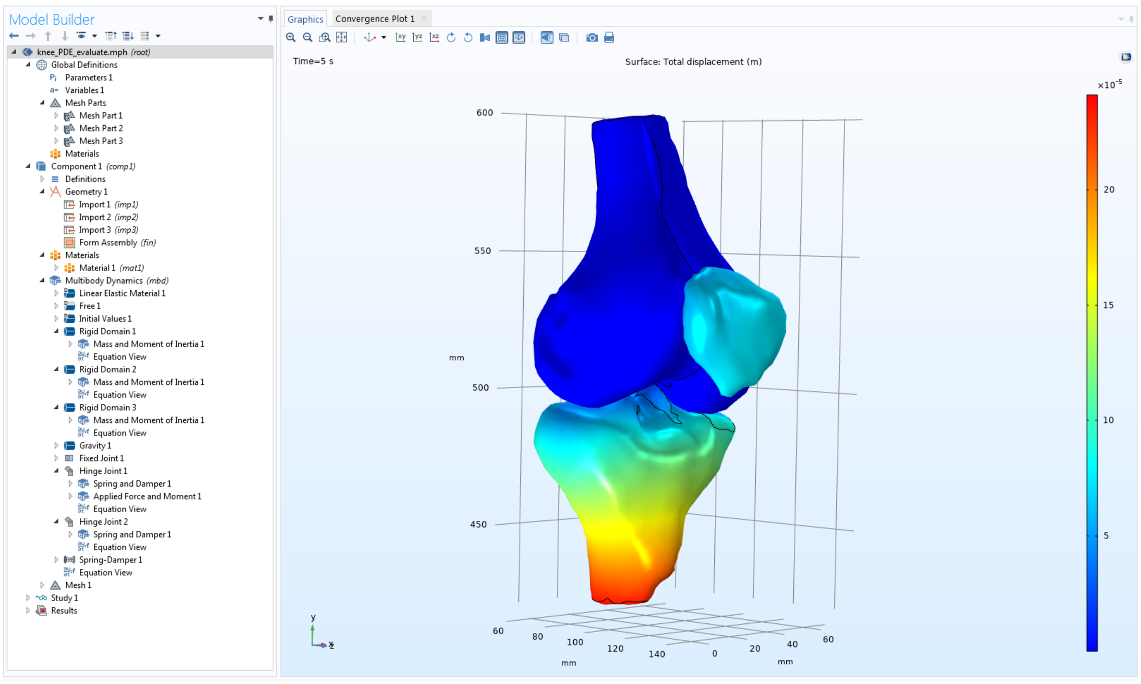


Figure 4.10.: Structure of the knee model in COMSOL.

of option mostly depends on the structure of the CAD files. After the import of the geometries, an assembly or a union can be built. Since the three bones should be able to move independently, they form an assembly.
All three bones are defined as rigid bodies. To each component attributes can be assigned, such as corresponding center of mass and rotational inertia. Furthermore, COMSOL gives the opportunity to transform units. Therefore, the values are the same as in Table 4.1 in the Adams model. The corresponding material properties, such as density, are added in the material subsection.
Again, two hinge joints, one between femur and tibia and the other between femur and patella, are defined. The center of rotation can be chosen freely. For both hinge joints, spring damper elements are included as attributes since the joints itself do not offer the possibility to define these parameters. Finally, the spring damper force describing the patellar tendon between patella and tibia is added. The corresponding parameter values are again the same as in the Simscape model, see Table 4.3. The acting torque is applied on the hinge joint between femur and tibia again using an attribute for the joint. The function defining the momentum can be entered manually. The Heaviside function is realised by a smoothed step function `flsmhs` dependent on the rise and a smoothing interval.
One additional joint is included in this model in order to prevent movement of the femur. Between femur and ground, a fixed joint is applied. No additional parameters are needed. Finally, gravity is applied to all three bones.

4.4. Discussion of Simulation Results

In the following, the results of the simulation models are presented and discussed. The three models describe the behaviour of the flexion between femur and tibia under the impact of the torque

$$\tau(t) = 28.8 \cdot \sin(2\pi \cdot 0.125 \cdot (t-1)) \cdot \mathrm{H}(t-1),$$

which is acting on the revolute joint. The resulting flexion angle is plotted in Figure 4.11 versus the simulation time span $[0, 5]$.

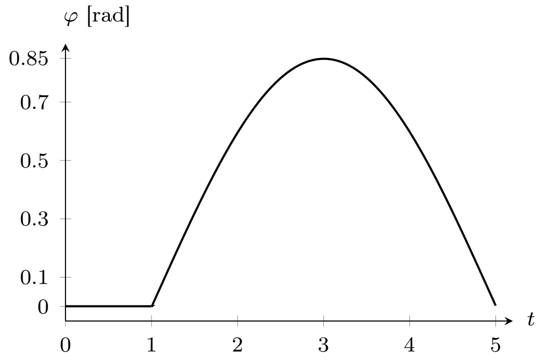


Figure 4.11.: Angle φ between femur and tibia in MapleSim.

Computations are done by MapleSim 2018.1. The results computed by the simulation models in Simscape and COMSOL show the same behaviour. The used versions are MATLAB 2018a and COMSOL 5.4. Table 4.5 gives an overview of the used differential equation solvers and tolerances in both multibody simulation frameworks. In COMSOL, the time step for calculating the solution is chosen by $\Delta t = 0.001$ to ensure a smooth behaviour of the results compared to the multibody models.

	Differential equation solver	Relative tolerance
MapleSim	`rkf45`	10^{-5}
Simscape	`ode45`	10^{-3}

Table 4.5.: ODE solver settings in the multibody model simulations.

In order to compare the simulation results of all three simulation tools, the absolute error of the angle for each time step is calculated. Hence, all three models use different time steps, the data are interpolated using MATLAB resulting in three vectors with the same length and time steps. Since MapleSim uses the smallest amount of time steps, these are used for the interpolation. Therefore, interpolating with these sample times leads to best results.

The results of MapleSim can be exported directly from the analysis window to a table. Since the solution of COMSOL is calculated using the finite element analysis, the results are available for all points defined by the meshes. For the angle, only the results for the coordinates of the hinge joint are exported.
The absolute error $\Delta\varphi$ is calculated by the absolute difference between the data points

$$\Delta\varphi = |\varphi_1 - \varphi_2|,$$

where $\varphi_i, i = 1, 2$, describe two simulation results for the angle. The absolute error is calculated by comparing all three simulations. In order to improve the illustration of the different model behaviour, the absolute error between the multibody models simulated in Simscape and MapleSim are considered separately to the absolute error between the multibody models and the PDE model simulated in COMSOL.

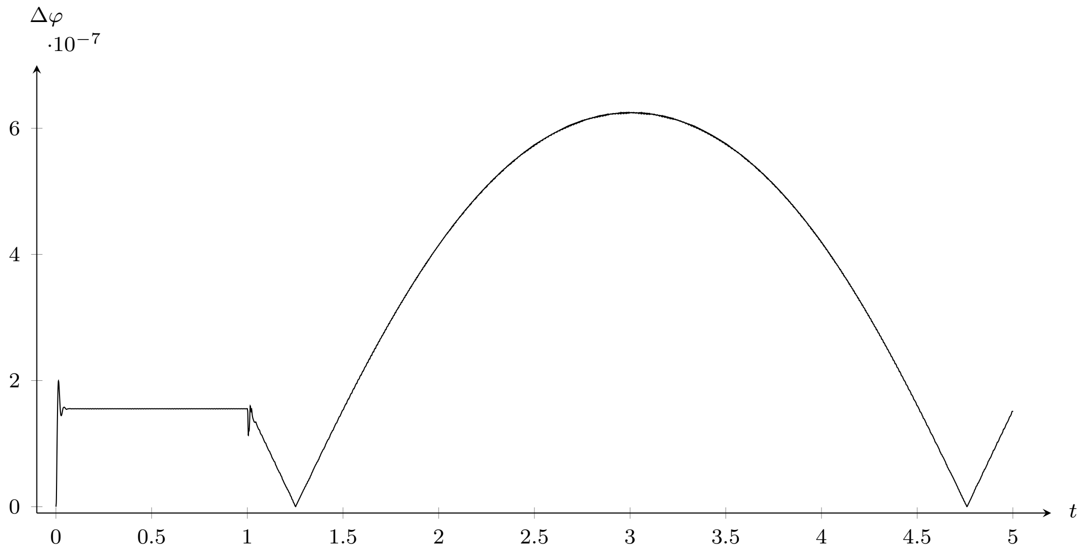


Figure 4.12.: Absolute error between the multibody models.

In Figure 4.12 the absolute error between the two multibody models is depicted over the simulation time t. The time span $t \in [0, 1]$ is plotted in Figure 4.13.The peak in the beginning of the simulation results from the adjustment of the linear spring connecting the tibia and patella. At this moment no other forces act on the system. At time step $t = 1$, the torque acting on the revolute joint increases which again results in an oscillating behaviour of the error. This result reveals how the calculation of the torque is treated differently. MapleSim is a computer algebra system, but Simscape, based on MATLAB, calculates numerically. Therefore, the movement of the tibia are slightly different. The norm of the vector describing the absolute error is given by $\|\Delta\varphi\|_2 = 1.73 \cdot 10^{-5}$. Since both simulation models are based on the multibody modelling theory, the simulation results are quite accurate.
As depicted in Figure 4.14, the errors of both multibody models compared to the COMSOL simulation show a similar behaviour. The peak at time $t = 1$ results

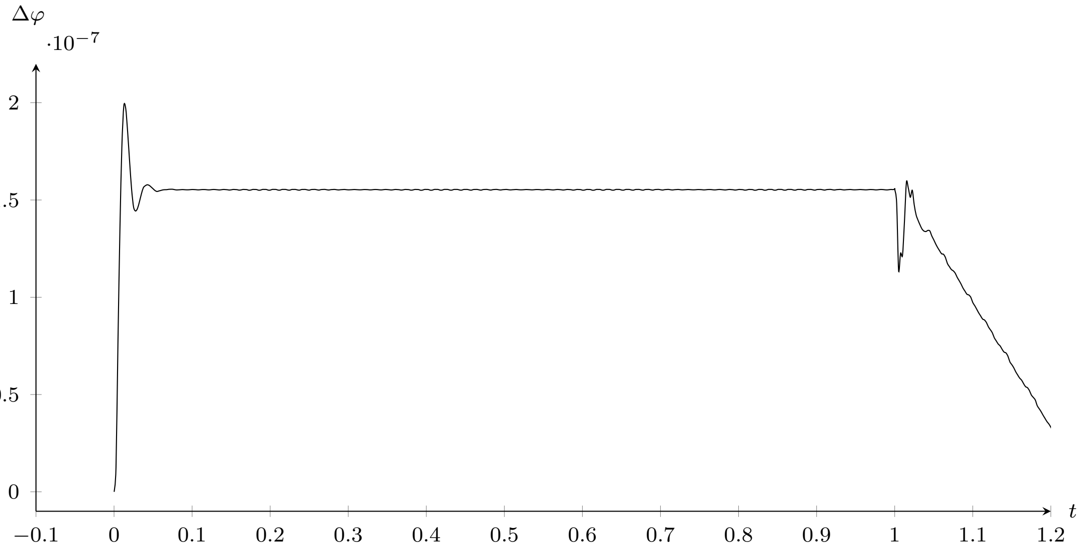


Figure 4.13.: Absolute error between the multibody models for $t \in [0, 1]$.

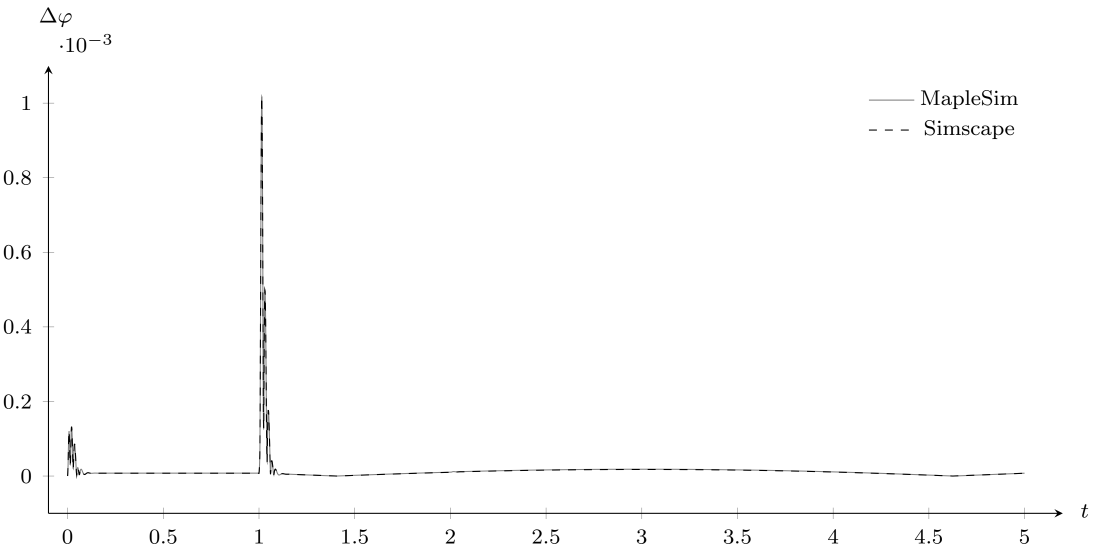


Figure 4.14.: Absolute error between multibody models and PDE model.

from using the heaviside function as input which is a sampled function in COMSOL within an interval, whose boundaries are chosen analogously to the timestep -0.001 and 0.001. During flexion of the tibia, an increase of $2 \cdot 10^{-5}$ in the error can be observed, as it is plotted in Figure 4.15. The norm of the absolute error between the COMSOL simulation and both multibody model simulations is $\|\Delta\varphi\|_2 = 0.0022$. In conclusion, the results show a satisfying behaviour regarding the numerical aspects.

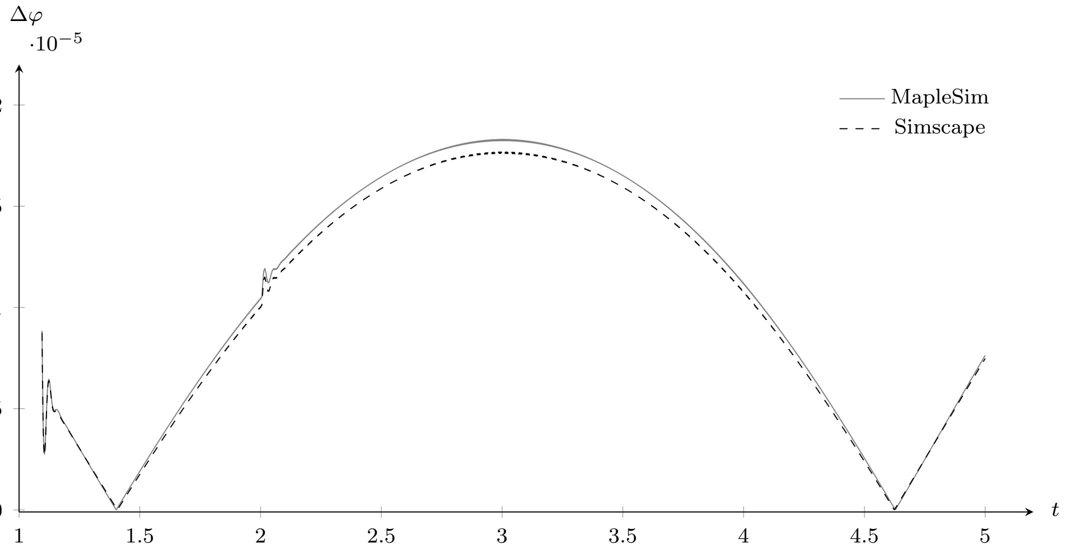


Figure 4.15.: Absolute error during flexion between multibody models and PDE model.

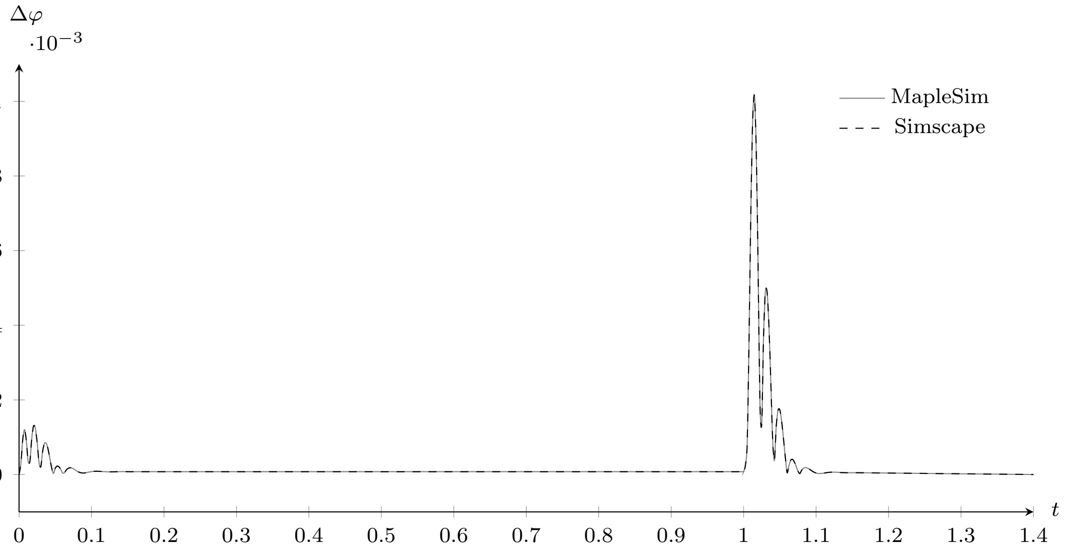


Figure 4.16.: Absolute error between multibody models and PDE model in time span $t \in [0, 1]$.

5. Simulation Loop for Knee Models

This chapter gives an overview about the possibilities of designing feedback loops for the knee simulation models introduced in chapter 4. Having two different modelling approaches, multibody modelling and models based on partial differential equations, requires various resolution methods in order to fulfil the necessary preconditions to embed the models to simulation loops. As presented in chapter 3, the plant in a control circuit is a dynamic system. Therefore, additional reformulation or some simplification, respectively, for the knee model described by partial differential equations is needed.
After a short introduction to the aim and motivation of the closed control loop, various control designs for the knee models are presented. This includes the discussion of the behaviour of the knee models in different simulation loops.
All three simulation models developed in chapter 4 are based on the same model description. This means, they fulfil the same biomechanical principles and calculate comparable output in respect to the same input. Nevertheless, this does not imply the same modelling approach but it gives the possibility to design one control circuit for all three simulation models.

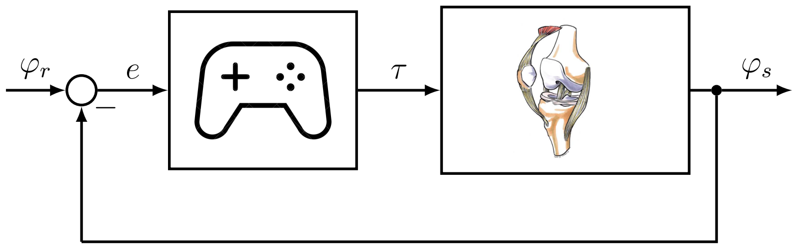


Figure 5.1.: Block diagram of the simulation loop principle for the knee model.

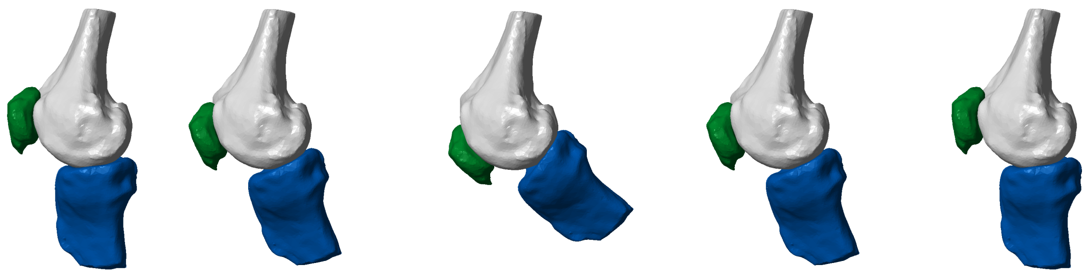
A block diagram of the considered control circuit design is depicted in Figure 5.1. In this case, the acting torque τ on the revolute joint between femur and tibia is input for the plant, and output is the angle φ_s between femur and tibia which is calculated by one simulation model. This choice facilitates the design for the feedback loop and enables an analysis of the behaviour for the different modelling approaches. While the plant and the controller are switched, the involved signals remain. The plant is changed in regard to the multibody model and the model based on partial differential equations. As controllers, various transfer functions and combinations of

multiple standard elements are considered as well as more complex control designs concerning nonlinear control. The behaviour regarding different model description is analysed and discussed.
The aim is to establish various controllers which are able to stabilise the knee in desired positions. The desired positions are summarised and visualised in Figure 5.2. The tibia starts in its initial position which corresponds to no flexion in respect to the femur. At time $t = 1$, the knee starts to move to position $\varphi = 0.3$ rad and

time t	position φ	Figure
0	0 rad	5.2(a)
1	0.3 rad	5.2(b)
2	0.8 rad	5.2(c)
3	0.3 rad	5.2(d)
4	0 rad	5.2(e)

Table 5.1.: Desired positions of the knee in the closed simulation loop.

goes until $\varphi = 0.8$ rad at time step $t = 2$. Afterwards, the knee returns to its initial position. This can be seen as a simulation for a gait cycle but it is possible to extend this reference signal for more complex applications as well. Concluding



(a) $t = 0$ $\varphi = 0$ (b) $t = 1$ $\varphi = 0.3$ (c) $t = 2$ $\varphi = 0.8$ (d) $t = 3$ $\varphi = 0.3$ (e) $t = 4$ $\varphi = 0$

Figure 5.2.: Visualisation of the desired positions of the knee.

control theory, this control design specifies a reference signal which should be the output from the plant after embedding the system to a closed loop. The following control designs stabilise the knee in any desired position. This allows to simulate various sequences of motion with a simple definition of the reference signal.

5.1. Control Designs for the Multibody Knee Model

The mathematical description of multibody models by ordinary differential equations allows to incorporate them directly to a closed simulation loop. This implies that multibody models are easily to describe as dynamic systems. Since the motion between bodies in multibody models is defined using a system of second-order differential equation

$$M\ddot{\boldsymbol{x}} + J_{\boldsymbol{x}}^T \lambda = F, \tag{5.1}$$

with the mass matrix M of the system, the vector $\boldsymbol{x}$ holding the state variables, $J_{\boldsymbol{x}}^T$ represents the Jacobian matrix of the state variables and λ the Lagrange multipliers. The external force F acts on the system. Reformulating the equation (5.1) for the second derivative of the state vector $\boldsymbol{x}$ on the left side only, leads to

$$\ddot{\boldsymbol{x}} = M^{-1}F - M^{-1}J_{\boldsymbol{x}}^T\lambda.$$

Introducing the state vector $\boldsymbol{z}$ with the state variables $\boldsymbol{x}$ and $\dot{\boldsymbol{x}}$,

$$\boldsymbol{z} = \begin{pmatrix} \boldsymbol{z_1} \\ \boldsymbol{z_2} \end{pmatrix} = \begin{pmatrix} \boldsymbol{x} \\ \dot{\boldsymbol{x}} \end{pmatrix}$$

allows to rewrite this system of second-order differential equations to system of a first-order differential equations only

$$\dot{\boldsymbol{z}} = \begin{pmatrix} \dot{\boldsymbol{z}}_{\boldsymbol{1}} \\ \dot{\boldsymbol{z}}_{\boldsymbol{2}} \end{pmatrix} = \begin{pmatrix} \dot{\boldsymbol{x}} \\ \ddot{\boldsymbol{x}} \end{pmatrix} = \begin{pmatrix} \dot{\boldsymbol{x}} \\ M^{-1}F - M^{-1}J_{\boldsymbol{x}}^T\lambda \end{pmatrix} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}, t).$$

This first-order differential equation is the function which describes the dynamics of the system, dependent on the input $\boldsymbol{u}$, the parameters $\boldsymbol{p}$ and time t. In combination with a function defining the output $\boldsymbol{y} = \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}, t)$, dependent on the state variables, all requirements for a dynamic system are fulfilled. Hence, multibody models do not require further reformulation because they already fulfil all necessary conditions. Indeed, this gives the possibility to create a subsystem containing the model in Simscape and use it directly as plant in Simulink in a feedback loop using linear control elements.

5.1.1. Linearisation of the Multibody Knee Model

Regarding the linear control design, the multibody model implemented in Simscape is considered as plant only. All control designs and theoretical aspects are valid for the multibody model in MapleSim as well, since the model description is the same. Using the control tools of Simulink, where the Simscape model can be embedded directly, allows more flexibility because no link between different software environments has to be established. First of all, the usage of linear control elements

requires a linearisation of the plant. Building a subsystem containing all blocks of the Simscape model for the human knee gives the possibility to embed this model as plant in a feedback loop in Simulink. Simulink offers a wide variety of analysis tools which are helpful in the design of control circuits. The linear analysis tool gives a linearisation of a model after defining input and output. The linearisation is shown in state space representation of the form (3.3.3) or as transfer function as in (3.11). For validation and understanding, the linearisation is realised manually as well.
As in the control circuit, only the relative motion between femur and tibia is relevant, these two bones are considered for further calculation in the first attempt. The connection of femur and tibia by one revolute joint recalls the equation of a mathematical pendulum. Since this joint contains a damping constant, an extended equation of the pendulum, is used, as in

$$\ddot{\varphi} = -\sin\varphi - \gamma\dot{\varphi} + \tau.$$

The acceleration of the angle φ is driven by the external torque τ and slowed down by the damping factor γ. Reformulating this to a first order differential equation by using $\dot{\varphi} = \omega$ leads to

$$\begin{pmatrix}\dot{\varphi}\\ \dot{\omega}\end{pmatrix} = \begin{pmatrix}\omega\\ -\sin\varphi - \gamma\omega + \tau\end{pmatrix}.$$

The state vector $\boldsymbol{x}$ is defined as

$$\boldsymbol{x} = \begin{pmatrix}\varphi\\ \omega\end{pmatrix},$$

the input is $u = \tau$ and the output function

$$y = h(\varphi, \omega, \tau, \gamma) = \varphi.$$

This allows to rewrite the equation for the pendulum as dynamic system

$$\dot{\boldsymbol{x}} = \begin{pmatrix}x_2\\ -\sin x_1 - \gamma x_2 + u\end{pmatrix} = \boldsymbol{f}(\boldsymbol{x}, u, \gamma), \tag{5.2a}$$

$$y = \varphi = h(\boldsymbol{x}, u, \gamma). \tag{5.2b}$$

For this nonlinear system a linearisation can be calculated at the stable equilibrium point

$$\boldsymbol{x}_E = \begin{pmatrix}\varphi_E\\ \chi_E\end{pmatrix} = \begin{pmatrix}0\\ 0\end{pmatrix}, \tag{5.3}$$

which corresponds to the lower vertical position of the pendulum. The following steps are accomplished as it was introduced in more detail in chapter 3. First, the Jacobians for $\boldsymbol{f}$ and h are calculated as

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} = \begin{pmatrix}0 & 1\\ -\cos\varphi & -\gamma\end{pmatrix}, \qquad \frac{\partial \boldsymbol{f}}{\partial u} = \begin{pmatrix}0\\ 1\end{pmatrix},$$

$$\frac{\partial h}{\partial \boldsymbol{x}} = \begin{pmatrix}1 & 0\end{pmatrix}, \qquad \frac{\partial h}{\partial u} = 0.$$

Evaluating the Jacobians at the equilibrium point $\boldsymbol{x}_E$ stated in (5.3) leads to the linear state space representation

$$\dot{\boldsymbol{x}} = \begin{pmatrix} 0 & 1 \\ -1 & -\gamma \end{pmatrix} \boldsymbol{x} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u,$$
$$y = \begin{pmatrix} 1 & 0 \end{pmatrix} \boldsymbol{x}.$$

The related transfer function is calculated by

$$\begin{aligned} G(s) &= \begin{pmatrix} 1 & 0 \end{pmatrix} \left(\begin{pmatrix} s & 0 \\ 0 & s \end{pmatrix} - \begin{pmatrix} 0 & 1 \\ -1 & -\gamma \end{pmatrix} \right)^{-1} \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} \frac{\gamma+s}{s^2+\gamma s+1} & \frac{1}{s^2+\gamma s+1} \\ -\frac{1}{s^2+\gamma s+1} & \frac{s}{s^2+\gamma s+1} \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ &= \frac{1}{s^2+\gamma s+1}, \end{aligned}$$

which corresponds to a P-T_2 element.
This linear system studies only the motion in the $x - y$ plane and simplifies the system regarding physical parameters as moment of inertia and mass of the rigid bodies. Simscape gives the possibility to extract the linear system of a model but the nonlinear equations describing the multibody model are not accessible. This is different compared to MapleSim which offers the possibility to extract the equations, which describe a simulation model. To reproduce the linearisation in Simscape, the equations for the knee model are extracted by using the equation extraction in MapleSim 2018.
The equations can be extracted for the entire model or for a certain part of the model only by creating a subsystem. Looking at the relative motion between femur and tibia, it is sufficient to analyse the equations for the subsystem with these two bones and the revolute joint connecting them. Rigid transforms between the bones and the joint are added to this subsystem as well. A parameter block can be introduced which contains equations with symbolic variables. Those include parameters concerning the bodies, the rigid transforms as well as global parameters like gravity. The ordinary differential equation calculated by MapleSim for the motion between femur and tibia is given by

$$\frac{\mathrm{d}^2\varphi}{\mathrm{d}t^2} \left(m_t(r_y^2 + r_z^2) + I_x \right) = g\, m_t \left(r_y \sin\varphi + r_z \cos\varphi \right) - \frac{\mathrm{d}\varphi}{\mathrm{d}t} K_d - \varphi K_s + \tau. \tag{5.6}$$

As before, this equation describes the acceleration of the angle φ between the two bones in respect to time t and dependent on the acting torque τ on the revolute joint. The explanations and values for the symbolic variables can be found in Table 5.2. The values are equal to the knee simulation model introduced in chapter 4.
With the state vector $\boldsymbol{x} = (\varphi, \omega)^T$, equation (5.6) can be rewritten to a system of

Variable	Value	Description
m_t	0.227 kg	Mass of tibia
r_y	$-41.3 \cdot 10^{-3}$ m	Transform between joint to tibia, y-direction
r_z	$0.755 \cdot 10^{-3}$ m	Transform between joint to tibia, z-direction
I_x	$140.86 \cdot 10^{-6}$ kg m^2	Inertial rotation of tibia, x-component
K_d	$\frac{180 \cdot 10^{-3}}{\pi} \frac{\text{N m s}}{\text{rad}}$	Damping coefficient in joint
K_s	$\frac{102.6}{\pi} \frac{\text{N m}}{\text{rad}}$	Spring stiffness constant in joint
g	$9.81 \frac{\text{m}}{\text{s}^2}$	Gravitation

Table 5.2.: Parameters $\boldsymbol{p}$ for the subsystem consisting of femur and tibia.

differential equations

$$\dot{\boldsymbol{x}} = \begin{pmatrix} \omega \\ \frac{g\, m_t(r_y \sin\varphi + r_z \cos\varphi) - \frac{\mathrm{d}\varphi}{\mathrm{d}t} K_d - \varphi K_s + \tau}{m_t(r_y^2 + r_z^2) + I_x} \end{pmatrix} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{p}, \tau, t), \tag{5.7a}$$

$$y = \varphi = h(\boldsymbol{x}, \boldsymbol{p}, \tau, t). \tag{5.7b}$$

The output function (5.7b) is defined additionally, fulfilling then all needed requirements for a dynamic system. Again, the Jacobian matrices of both functions $\boldsymbol{f}, h$ are calculated by

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} = \begin{pmatrix} 0 & 1 \\ \frac{g\, m_t(r_y \cos\varphi - r_z \sin\varphi) K_s}{m_t(r_y^2 + r_z^2) + I_x} & -\frac{K_d}{m_t(r_y^2 + r_z^2) + I_x} \end{pmatrix}, \qquad \frac{\partial \boldsymbol{f}}{\partial u} = \begin{pmatrix} 0 \\ \frac{1}{m_t(r_y^2 + r_z^2) + I_x} \end{pmatrix}, \tag{5.8a}$$

$$\frac{\partial h}{\partial \boldsymbol{x}} = \begin{pmatrix} 1 & 0 \end{pmatrix}, \qquad \frac{\partial h}{\partial u} = 0. \tag{5.8b}$$

Evaluating the Jacobians at the equilibrium point $\boldsymbol{x}_E = (0, 0)^T$ as before in (5.3) and substituting the variables in (5.8) with the values in Table 5.2 gives the linear state space representation

$$\dot{\boldsymbol{x}} = \begin{pmatrix} 0 & 1 \\ -62006.32 & -108.47 \end{pmatrix} \boldsymbol{x} + \begin{pmatrix} 0 \\ 1893.29 \end{pmatrix} u, \tag{5.9a}$$

$$y = \begin{pmatrix} 1 & 0 \end{pmatrix} \boldsymbol{x}. \tag{5.9b}$$

As before, the corresponding transfer function is calculated by

$$\begin{aligned} G(s) &= \begin{pmatrix} 1 & 0 \end{pmatrix} \left(\begin{pmatrix} s & 0 \\ 0 & s \end{pmatrix} - \begin{pmatrix} 0 & 1 \\ -62006.32 & -108.47 \end{pmatrix} \right) \begin{pmatrix} 0 \\ 1893.29 \end{pmatrix} \\ &= \frac{1893.29}{s^2 + 108.47 s + 62006.32}. \end{aligned}$$

As stated above, Simulink gives the possibility to linearise a model using the linear analysis tool. This makes it simple to derive a state space representation or transfer function of a model in Simulink or Simscape, respectively. Therefore, an additional model, which contains just femur and tibia, is created. The linearisation is calculated after the definition of input and output signal as well as an operating point, equivalent to the equilibrium point before. The initial position is chosen as operating point of the model which is equivalent to the vertical position (5.3). The state space representation of the linear analysis tool leads to almost the same result as in (5.9a) and is given by

$$\begin{aligned}\dot{\boldsymbol{x}} &= \begin{pmatrix} 0 & 1 \\ -6.201 \cdot 10^4 & -108.5 \end{pmatrix} \boldsymbol{x} + \begin{pmatrix} 0 \\ 1893 \end{pmatrix} u, \\ y &= \begin{pmatrix} 1 & 0 \end{pmatrix} \boldsymbol{x}.\end{aligned}$$

The state variables are again the angle φ and its corresponding derivative, the angular velocity. The corresponding transfer function is calculated using the MATLAB function `tf` and again, a P-T_2 element is the result

$$G(s) = \frac{1893}{s^2 + 108.5s + 6.201 \cdot 10^4}.$$

This shows that the simplification of the knee to a single pendulum represents a good approximation. From the pendulum's behaviour further deduction can be made fore more detailed analysis, even in the control design. The equivalence of the results, linearising the model using Simulink and achieving a state space representation manually after extracting the equations from MapleSim show that both models are based on the same mathematical description.
Defining the LTI system in (5.10) as state space model in MATLAB gives the opportunity to analyse it for observability and controllability. Both corresponding matrices have full rank which is verified with the MATLAB functions `obsv` and `ctrb`. This shows that this system is fully observable and controllable.
Considering not only femur and tibia, but patella as well, leads to a more complex state space. Again, using the linear analysis tool leads to a state space respresentation

$$\dot{\boldsymbol{x}} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ -6.62 \cdot 10^6 & -4.26 \cdot 10^4 & 6.61 \cdot 10^6 & 4.18 \cdot 10^4 \\ 0 & 0 & 0 & 1 \\ 8.46 \cdot 10^5 & 5.36 \cdot 10^3 & -9.11 \cdot 10^5 & -5.48 \cdot 10^3 \end{pmatrix} \boldsymbol{x} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1.89 \cdot 10^3 \end{pmatrix}, \tag{5.11a}$$

$$y = \begin{pmatrix} 0 & 0 & 1 & 0 \end{pmatrix} \boldsymbol{x}. \tag{5.11b}$$

Two more states are introduced which represent the angle and angle velocity in regard to the relative motion between femur and patella. This implies a higher order transfer function,

$$G(s) = \frac{1893s^2 + 8.06 \cdot 10^7 s + 1.25 \cdot 10^{10}}{s^4 + 4.81 \cdot 10^4 s^3 + 1.67 \cdot 10^7 s^2 + 4.24 \cdot 10^9 s + 4.36 \cdot 10^{11}}. \tag{5.12}$$

The extended linear system (5.11) can again be analysed regarding its observability and controllability. The rank of the observability matrix is 4 which corresponds to an observable system. This turns out as expected, because all state variables can be extracted from the model. The rank of the controllability matrix is only 3, which can be explained by the fact that there is just one input acting on the joint between femur and tibia. This means, no input is acting on the joint between femur and patella. Fortunately, this will not be obstructive for further consideration.
The representation by linear functions describing the multibody knee model allows to apply linear control designs.

5.1.2. PID Control for the Multibody Model

In the first approach, a PID block from the Simulink library is chosen as controller. The Simscape model for the human knee is embedded directly in the simulation loop as subsystem. This involves signal converter blocks, transferring the signal from Simulink to Simscape and vice versa. A block diagram of the feedback loop is depicted in Figure 5.3.

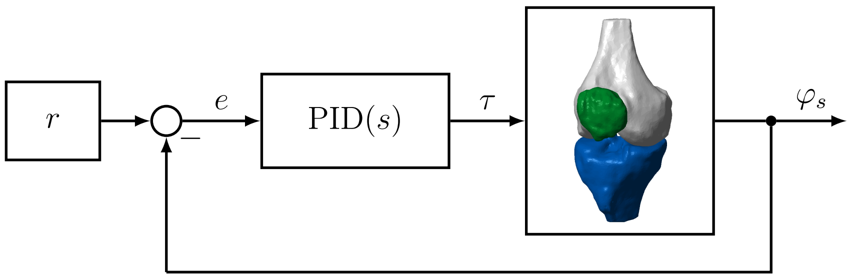


Figure 5.3.: Block diagram of a PID control circuit in Simulink.

As reference signal r, a repeating sequence stair is chosen. Using the Heaviside function H, this signal can be written as

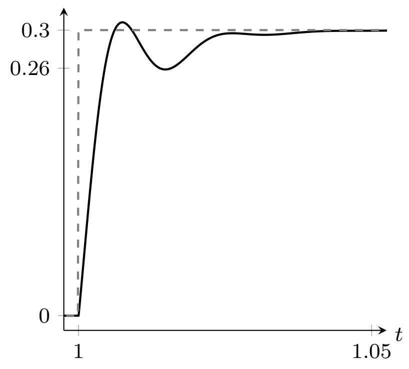
$$r(t) = 0.3 \cdot \mathrm{H}(t-1) + 0.5 \cdot \mathrm{H}(t-2) - 0.5 \cdot \mathrm{H}(t-3) - 0.3 \cdot \mathrm{H}(t-4), \tag{5.13}$$

what results in a motion of the tibia as illustrated in Figure 5.2.The control tools in Simulink include tuning for PID blocks. The parameters describing a PID tuner, the proportional, differentiator and integrator coefficient are calculated by Simulink. The PID block in Simulink contains one additional parameter, the filter coefficient N. This is the realisation term for the differentiator, equivalent to $\frac{1}{T}$. For the tuning, the plant is linearised first as it is shown in equation (5.11) and afterwards the step response of the closed loop is analysed. The step response is the output from the system having the Heaviside function H as input. The parameters are calculated to get the desired step response. For the step response, the default value for response time $t = 5 \cdot 10^{-3}$ and transient behaviour $b = 0.6$ are taken. The response time indicates how fast the response of the plant achieves the reference signal. A lower response time results in a faster adaptation to the reference signal. The transient behaviour indicates the oscillating behaviour of the step response. A higher value indicates a robust adaptation resulting in a lower oscillation. A low value shows an aggressive behaviour corresponding to high oscillation. The resulting parameters are summarised in Table 5.3. The results of the simulation are plotted in Figure 5.4

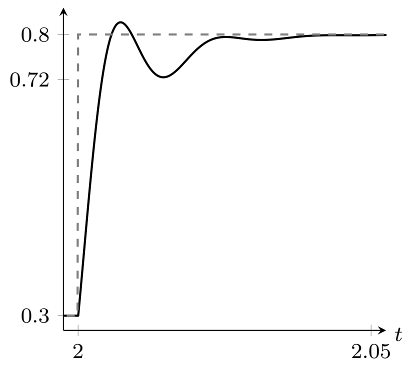
Parameter	Value
P	67.58
I	$89.32 \cdot 10^{2}$
D	$12.67 \cdot 10^{-2}$
N	$45.15 \cdot 10^{2}$

Table 5.3.: Parameters of PID controller calculated by Simulink.

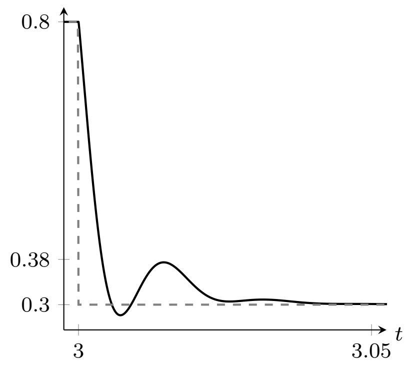
for the time steps, where the reference signal changes.
Since the mutibody knee model is represented in linear form as a P-T_2 element, an oscillating behaviour is observed in order to follow the desired reference signal. The steps representing the reference signal are in the first and last time step lower than in the second and third. This leads to a higher overshoot of the output signal at time $t = 2$ and $t = 3$ with a reference signal of 0.5 rad. This can be seen in Figure 5.4(b) and 5.4(c). At time $t = 1$ and $t = 4$, with a step of 0.3 rad, the step response shows an oscillating behaviour with an amplitude of 0.04 rad, what can be seen in Figure 5.4(a) and 5.4(d).



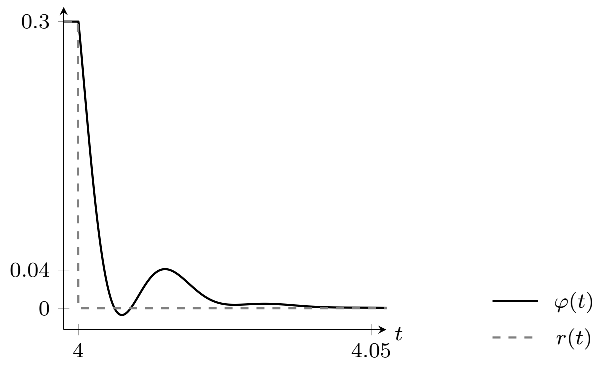
(a) PID control for $t \in [1, 2)$.



(b) PID control for $t \in [2, 3)$.



(c) PID control for $t \in [3, 4)$.



(d) PID control for $t \in [4, 5)$.

Figure 5.4.: PID control for the multibody knee model in four time spans.

5.1.3. Control with Transfer Functions for the Multibody Model

Additionally to the PID tuning, Simulink offers the Control System Designer, a tool, which allows to tune other blocks in order to establish a feedback loop. Blocks which can be tuned include LTI systems, gain blocks and transfer functions, in continuous and discrete time, respectively.
A control circuit is built containing the submodel for the Simscape knee simulation as plant and a transfer function

$$C(s) = \frac{p(s)}{q(s)}$$

acting as controller. A block diagram for this simulation loop is depicted in Figure 5.5. In the Control System Designer, the block C is chosen to be tuned and again input and output signals are defined. This allows to use automatic tuning methods. In this case, the internal model control method is chosen which calculates the coefficients for a transfer function in a way that the stability of the closed loop is

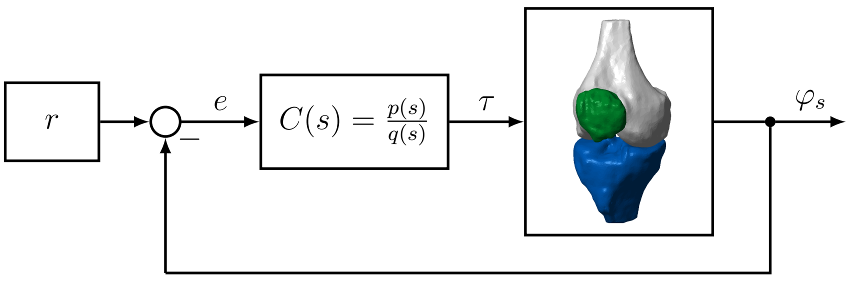


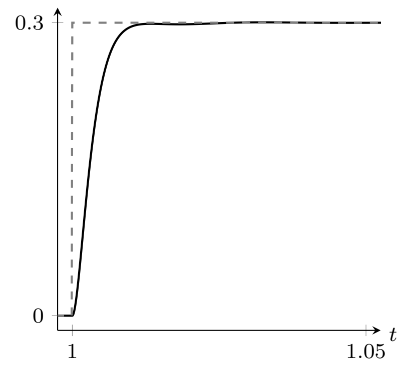
Figure 5.5.: Block diagram for control with transfer function for the multibody model.

guaranteed. Additional to the dominant closed-loop time constant, the controller order can be chosen which depends on the plants dynamics. For the analysis of a well tuned controller, transfer functions for all orders are calculated and then compared. The control system designer is not able to calculate a first-order and fifth-order transfer function. As dominant closed loop time constant the default value $t = 1.73 \cdot 10^{-3}$ is chosen which corresponds to 5% of the settling time of the plant. The calculated transfer functions are summarised in Table 5.4.

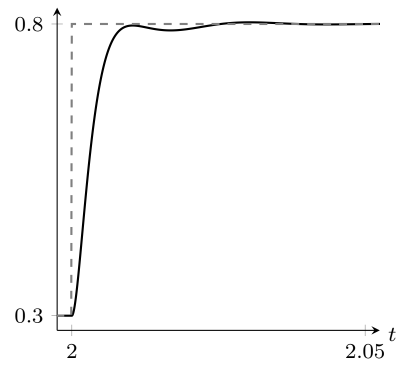
Order of degree	Transfer function
2	$C(s) = \frac{195.89(s^2+193s+5.86 \cdot 10^4)}{s(s+1143)}$
3	$C(s) = \frac{175.76(s+4.69 \cdot 10^4)(s^2+193.1s+5.87 \cdot 10^4)}{s(s+4.17 \cdot 10^4)(s+1156)}$
4	$C(s) = \frac{175.48(s+4.77 \cdot 10^4)(s+155.7)(s^2+192.5s+5.87 \cdot 10^4)}{s(s+4.24 \cdot 10^4)(s+1153)(s+156.1)}$

Table 5.4.: Transfer functions calculated with the Control System Designer in Simulink.

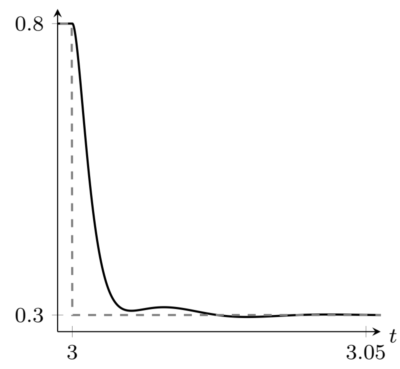
In the closed simulation loop, all transfer functions show the same behaviour. A higher or lower order of the transfer function does not have a remarkable effect on the output of the closed loop as it can be seen in Figure 5.6. In Figure 5.7 the control errors between output and reference signal for all three transfer functions are plotted. The index of the output signal represents the order of the transfer function. It can be seen that already the second order transfer function fits as good as the fourth order transfer function.
Looking at the poles and zeros of the transfer functions support these results. The corresponding positions of zeros and poles are summarised in Table 5.5.
This shows that the higher order transfer functions keep zeros and poles of the lower order transfer functions. Of course, poles and zeros are added. Since the knee simulation model is already stable, the poles of the control transfer function guarantee only that the model remains stable. The poles λ_i of the transfer function describing



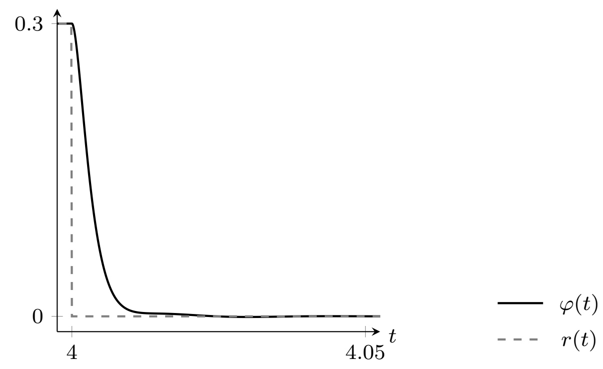
(a) Control with one transfer function for $t \in [1, 2)$.



(b) Control with one transfer function for $t \in [2, 3)$.



(c) Control with one transfer function for $t \in [3, 4)$.



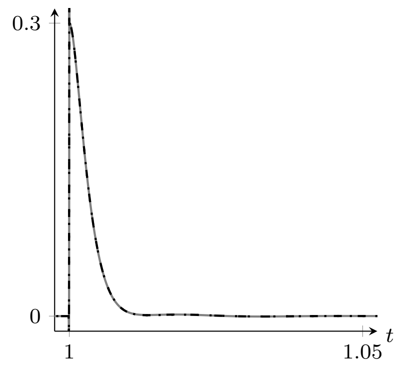
(d) Control with one transfer function for $t \in [4, 5)$.

Figure 5.6.: Control with one transfer function for the multibody knee model in four time spans.

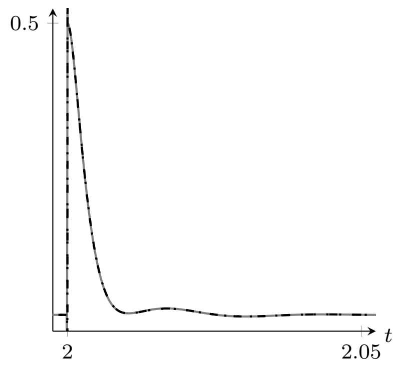
the knee simulation model as it is given in (5.12), are calculated to

$$\begin{aligned}\lambda_1 &= -4.77 \cdot 10^4,\\ \lambda_2 &= -1.55 \cdot 10^2,\\ \lambda_{3,4} &= -98.63 \pm 222.15\,\mathrm{i}\,.\end{aligned}$$

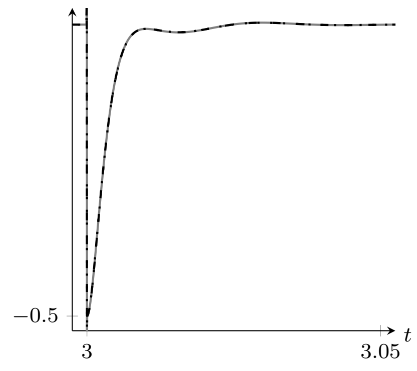
Theorem 3.3.19 shows that the system describing the multibody knee model is BIBO stable. The aim of the controller is to calculate an input for this system to simulate the desired output signal. This is achieved by proper coefficients of the transfer function.



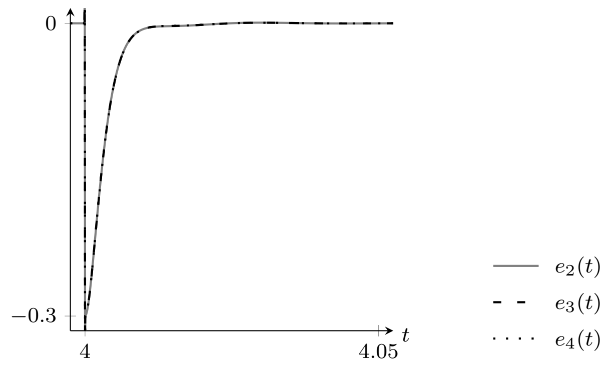
(a) Control errors $t \in [1, 2)$.



(b) Control errors $t \in [2, 3)$.



(c) Control errors $t \in [3, 4)$.



(d) Control errors $t \in [4, 5)$.

Figure 5.7.: Control error for various order of transfer functions for the multibody knee model.

Transfer function 2nd order	zeros	$-96.5 \pm 222\,\mathrm{i}$
	poles	$-1.14 \cdot 10^3$ $1.78 \cdot 10^{-15}$
Transfer function 3rd order	zeros	$-4.69 \cdot 10^4$ $-96.5 \pm 222\,\mathrm{i}$
	poles	$-4.17 \cdot 10^4$ $-1.16 \cdot 10^3$ $-2.76 \cdot 10^{-13}$
Transfer function 4th order	zeros	$-4.77 \cdot 10^4$ -156 $-96.2 \pm 222\,\mathrm{i}$
	poles	$-4.24 \cdot 10^4$ $-1.15 \cdot 10^3$ 156

Table 5.5.: Zeros and poles of the transfer functions.

5.1.4. Nonlinear PD Control Design for the Multibody Model

Concluding the control designs for the multibody model, a nonlinear control law is formulated. This is realised for the nonlinear equations extracted from MapleSim given in (5.6). Since the description of the multibody knee model follows the Lagrange equations as in (3.43), it is possible to apply the PD control law. Obviously, the following control law is valid for the restricted system, which considers femur and tibia, because the extracted equation describes the relative motion between these two bodies. Introducing the variables

$$\begin{aligned} M &= m(r_y^2 + r_z^2) + I_x, \\ C &= K_d, \\ f &= m \cdot g \cdot (-\sin\varphi) + r_y - r_z \cdot \cos\varphi + K_s\varphi, \end{aligned}$$

gives the desired form of the differential equation $M\ddot{\varphi} + c\dot{\varphi} + f(\varphi) = \tau$. Defining now a torque τ acting on the system of the form

$$\tau = K_P(\varphi_d - \varphi) - K_D\dot{\varphi} + f(\varphi),$$

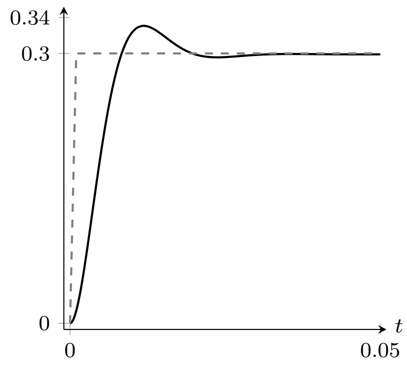
stabilises the knee in the desired position φ_d. K_P indicates the proportional factor of the controller. A higher value results in a better match of the output signal of the plant compared to the reference signal. The parameter K_D indicates the robustness of the controller. This constant influences the angle velocity and therefore the approximation from the output signal. A high value indicates a robust controller, a low value defines an aggressive control law. This results in an oscillating behaviour of the output signal before it reaches the desired value. The new defined system of differential equations

$$\ddot{\varphi} = -K_d - m \cdot g \cdot (-\sin\varphi) - r_y + r_z \cdot \cos\varphi - K_s\varphi + K_P(\varphi_d - \varphi) - K_D\dot{\varphi} + f(\varphi) \tag{5.14}$$

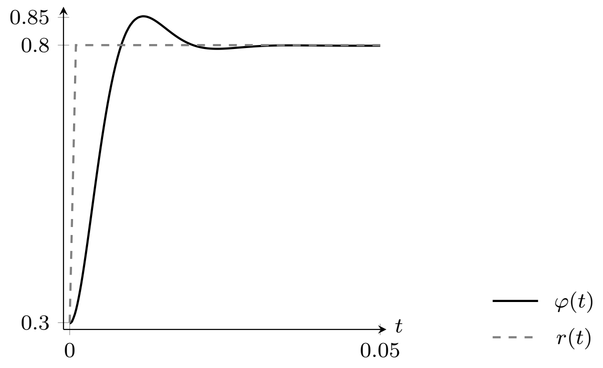
is solved using `ode45` in MATLAB. The parameters of the controller are calibrated to $K_P = 56$ and $K_D = 0.2$. This results in an overshoot of the output signal which matches the desired position with an error of 10^{-2}. The simulation is executed in four different time spans, one for each step of the reference signal. Since the desired position in the control law is a constant value, the simulation is performed when the desired position changes. As initial values for the simulations, the values of φ and $\dot{\varphi}$ from the last time step of the simulation before are applied. These initial and final values are summarised in Table 5.6. The results for all four simulations are visualised in Figure 5.8.

	Initial Values		Final Values	
	φ	$\dot{\varphi}$	φ	$\dot{\varphi}$
$t \in [1, 2)$	0	0	$29.89 \cdot 10^{-2}$	$5.82 \cdot 10^{-3}$
$t \in [2, 3)$	$29.89 \cdot 10^{-2}$	$5.82 \cdot 10^{-3}$	$79.89 \cdot 10^{-2}$	$9.74 \cdot 10^{-3}$
$t \in [3, 4)$	$79.89 \cdot 10^{-2}$	$9.74 \cdot 10^{-3}$	$29.9 \cdot 10^{-2}$	$-9.74 \cdot 10^{-3}$
$t \in [4, 5)$	$29.9 \cdot 10^{-2}$	$-9.74 \cdot 10^{-3}$	$-9.94 \cdot 10^{-4}$	$-5.84 \cdot 10^{-3}$

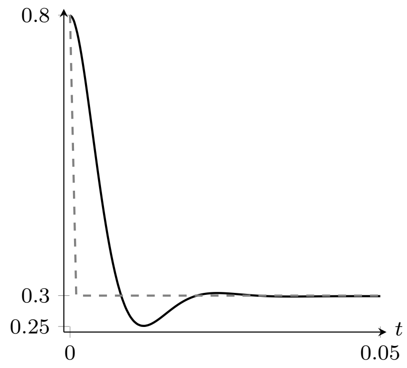
Table 5.6.: Initial and final values for the simulations with the PD control law.



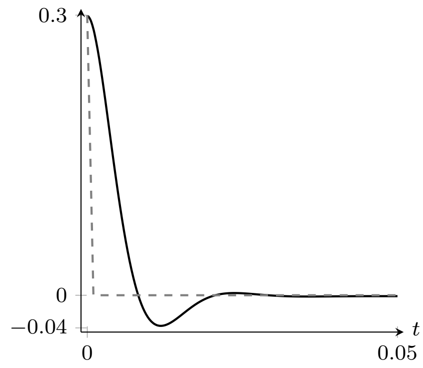
(a) PD control for $t \in [1, 2)$.



(b) PD control for $t \in [2, 3)$.



(c) PD control for $t \in [3, 4)$.



(d) PD control for $t \in [4, 5)$.

Figure 5.8.: PD control for the multibody knee model in four time spans.

5.2. Control Design for the PDE Knee Model

The design of a closed simulation loop is examined in Simulink, analogously to the procedure performed with the multibody model. This presumes that it is possible to establish an open loop simulation, where Simulink defines the input for the COMSOL model. After the simulation in COMSOL, it is necessary to access the output in Simulink. The solution for the simulation model in COMSOL is calculated using the finite element analysis because the model description is based on partial differential equations. This implies that this system does not depend on time only but as well on space. This is a so-called distributed parameter system with an infinite dimensional state space. Obviously, this complicates the formulation as dynamic system.

5.2.1. Possibilities of running a COMSOL simulation in MATLAB

COMSOL offers the possibility to create a LiveLink to MATLAB which allows to run simulations in COMSOL using the MATLAB interface. With the help of `S`-functions, a simulation loop in Simulink can be established. A detailed explanation of the structure for the `S`-function is found in [28]. This open loop structure is built for the knee simulation in COMSOL but it is not considered for feedback design. Since the COMSOL simulation model is based on nonlinear partial differential equations, the formulation of a control law is complex. Linear tools, as transfer functions for example, do not accomplish satisfying results with the direct usage of the COMSOL model as plant. This concludes to establish a linear representation of the COMSOL simulation model.
With the usage of COMSOL, it is possible to calculate a state space representation for the simulation model. The form of this state space representation is given by

$$\begin{aligned} M_c\dot{\boldsymbol{x}} &= M_c M_A \boldsymbol{x} + M_c M_B \boldsymbol{u}, \\ y &= C\boldsymbol{x} + D\boldsymbol{u}, \end{aligned}$$

where $M_A \in \mathbb{R}^{n\times n}, M_B \in \mathbb{R}^{n\times l}, C \in \mathbb{R}^{n\times m}, D \in \mathbb{R}^{m\times l}$ as well as the state vector $\boldsymbol{x} \in \mathbb{R}^n$, the output $y \in \mathbb{R}^m$ and the input $u \in \mathbb{R}^l$. If the mass matrix $M_c \in \mathbb{R}^{n\times n}$ is singular, the system is described by differential algebraic equations. For large systems, this state space representation is more suitable than (3.3.3) because the matrices M_c and M_A become more sparse. After defining as input again the acting torque on the hinge joint and as output the angle between femur and tibia, the state space matrices can be exported to

$$M_c = \begin{pmatrix} 0 & 0 & 0 & 10.21\cdot 10^{-4} & -3.39\cdot 10^{-4} & 0 \\ 0 & 0 & 0 & -3.39\cdot 10^{-4} & 3.03\cdot 10^{-4} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 10.21\cdot 10^{-4} & 0 & 0 & 0 & 0 & 0 \\ 0 & 10.21\cdot 10^{-4} & 0 & 0 & 0 & 0 \\ 0 & 0 & 10.21\cdot 10^{-4} & 0 & 0 & 0 \end{pmatrix}, \tag{5.15a}$$

$$M_A = \begin{pmatrix} -22.61\cdot 10^{-2} & 44.73\cdot 10^{-2} & 0 & -125.05\cdot 10^{-2} & 269.01\cdot 10^{-2} & 0 \\ 44.73\cdot 10^{-2} & -1 & 0 & 269.01\cdot 10^{-2} & -622.64\cdot 10^{-2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 83.86 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \tag{5.15b}$$

$$M_B^T = \begin{pmatrix} -5.12 & 2.04 & 0 & 0 & 0 & 0 \end{pmatrix}, \tag{5.15c}$$

$$C = \begin{pmatrix} -2.25\cdot 10^{-4} & 8.96\cdot 10^{-5} & 0 & 0 & 0 & 0 \end{pmatrix}, \tag{5.15d}$$

$$D = 0. \tag{5.15e}$$

Due to the singularity of the matrix M_c, no state space representation of the form (3.3.3) can be defined. With the usage of the descriptor state space block in Simulink, this state space representation can be directly imported in the simulation. Furthermore, this form gives the possibility to design a feedback loop in Simulink.

5.2.2. PID control for the COMSOL Model

In the first approach, a PID control design is evaluated, its block diagram is depicted in Figure 5.9. The COMSOL model is realised in Simulink with the descriptor state space block which allows to define a system with the structure introduced in (5.15).

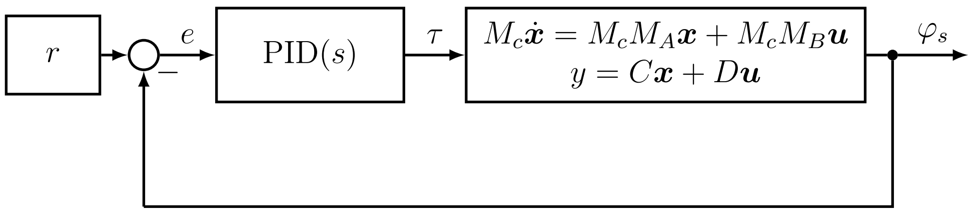

r
e
$-$
PID(s)
τ
$M_c\dot{\boldsymbol{x}} = M_c M_A \boldsymbol{x} + M_c M_B \boldsymbol{u}$
$y = C\boldsymbol{x} + D\boldsymbol{u}$
φ_s

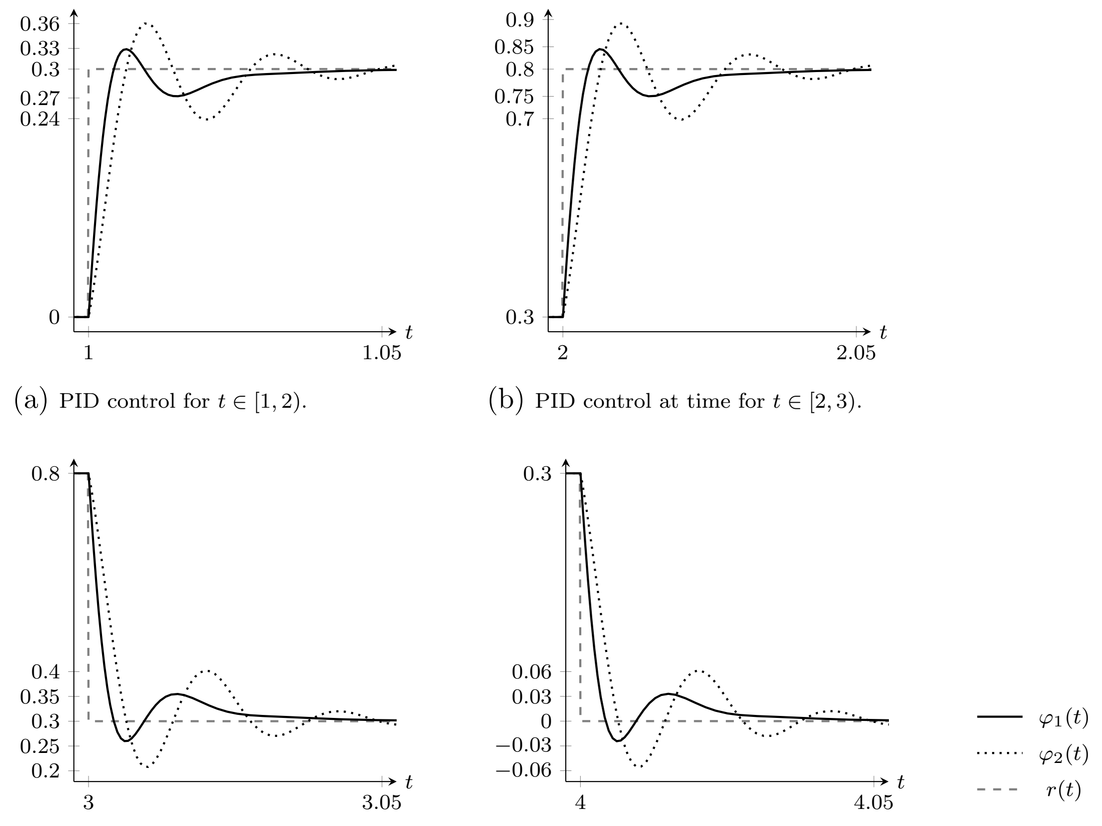
Figure 5.9.: Block diagram of a PID control circuit for the COMSOL model.

The values of the parameters from the multibody model for the PID control design do not show a satisfying behaviour for the COMSOL model. Therefore, the parameters of the PID controller are tuned again with the control design in Simulink. This leads to slightly different values for the parameters as it can be seen in Table 5.7. The values for the proportional and integrator parameter P and I are twice or three times as high, respectively, than the values which were tuned for the multibody model. The differentiator and realisator term D and N are in the same range as for the multibody model.

Parameter	Value
P	135.49
I	$11.27 \cdot 10^3$
D	$37.66 \cdot 10^{-2}$
N	$55.03 \cdot 10^3$

Table 5.7.: Parameters tuned by Simulink for the PID block for the COMSOL model.

In Figure 5.10 the PID control results for the COMSOL model are plotted with the parameter sets first tuned for the COMSOL model, second of the multibody model. The output signals use the index 1 for the parameter set for the PDE model, and 2 for the multibody model.
The results show that the PID control with the parameters of the multibody model needs more time to calculate the required input for the PDE model. The parameters tuned for the PDE model itself results in a more robust controller which is able to match the reference signal faster.



(a) PID control for $t \in [1, 2)$.

(b) PID control at time for $t \in [2, 3)$.

(c) PID control for $t \in [3, 4)$.

(d) PID control for $t \in [4, 5)$.

Figure 5.10.: PID control for the COMSOL model with two different parameter sets.

5.2.3. Control with Transfer functions for the COMSOL Model

The design of a controller with one degree of freedom, consisting of a transfer function is examined for the PDE model in COMSOL as well. The block diagram of this control design is depicted in Figure 5.11.

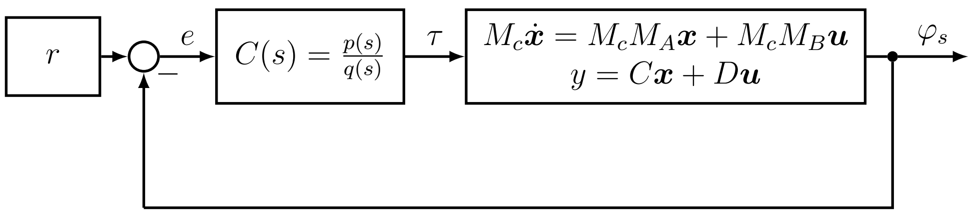

r
e
$-$
$C(s) = \frac{p(s)}{q(s)}$
τ
$M_c\dot{\boldsymbol{x}} = M_c M_A \boldsymbol{x} + M_c M_B \boldsymbol{u}$
$y = C\boldsymbol{x} + D\boldsymbol{u}$
φ_s

Figure 5.11.: Block diagram of a control with one transfer function for the COMSOL model.

Using the control system designer in Simulink, the transfer function is tuned again for the COMSOL model. Three different transfer functions are calculated, they are summarised in Table 5.9.

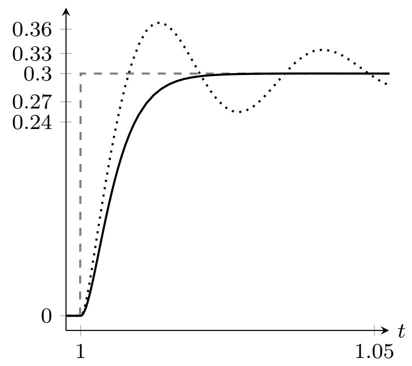
Order of degree	Transfer function
2	$C(s) = \frac{105.16(s^2+113.3s+3.27\cdot10^4)}{s(s+640)}$
3	$C(s) = \frac{92.74(s+2.48\cdot10^4)(s^2+113.2s+3.28\cdot10^4)}{s(s+2.17\cdot10^4)(s+648)}$
4	$C(s) = \frac{92.73(s+2.49\cdot10^4)(s+156.5)(s^2+113.1s+3.28\cdot10^4)}{s(s+2.18\cdot10^4)(s+647.3)(s+156.6)}$

Table 5.8.: Transfer functions calculated with the Control System Designer in Simulink for the COMSOL model.

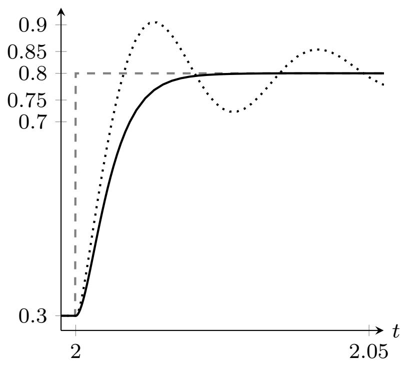
Analysing the poles and zeros in Table 5.9 of these transfer functions leads to similar observations as before regarding the transfer functions for the multibody model. Transfer functions with higher order have alomost the same poles and zeros as the ones with lower order. Again, using transfer function does not guarantee stability of the plant but they require proper coefficients in order to match the reference signal. Therefore, a higher order of the transfer function does not fulfil better conditions for the closed simulation circle. In Figure 5.12 the results of the control with transfer functions for the COMSOL model are plotted. The transfer function of second order is applied, once with the parameters tuned for the COMSOL model, secondly with the coefficients of the multibody model. This shows that the transfer function tuned for the COMSOL model results in a robust controller with no overshoot. The results with the transfer function calibrated for the multibody model leads to an oscillating behaviour of the COMSOL model. The final value of the reference signal is reached but the oscillation signifies that the parameters are not well tuned.

Transfer function 2nd order	zeros	$-56.7 \pm 172\,\mathrm{i}$
	poles	-640 $5.68 \cdot 10^{-14}$
Transfer function 3rd order	zeros	$-2.48 \cdot 10^{4}$ $-56.6 \pm 172\,\mathrm{i}$
	poles	$-2.17 \cdot 10^{4}$ -648 $2.2 \cdot 10^{-13}$
Transfer function 4th order	zeros	$-2.5 \cdot 10^{4}$ -156 $-56.5 \pm 172\,\mathrm{i}$
	poles	$-2.18 \cdot 10^{4}$ -647 -157 $5.68 \cdot 10^{-14}$

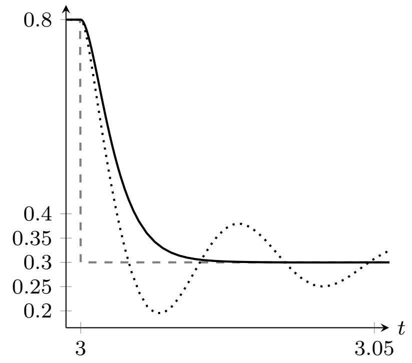
Table 5.9.: Zeros and poles of the transfer functions.



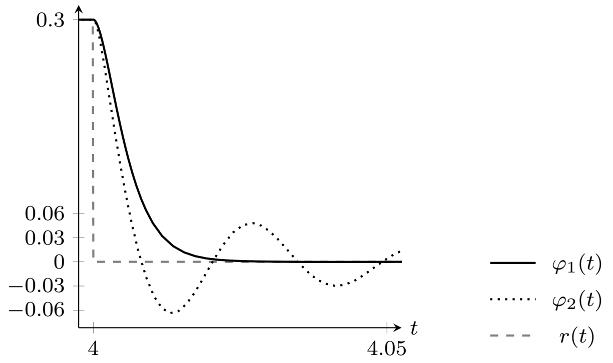
(a) Control with transfer function for $t \in [1, 2)$.



(b) Control with transfer function for $t \in [2, 3)$.



(c) Control with transfer function for $t \in [3, 4)$.

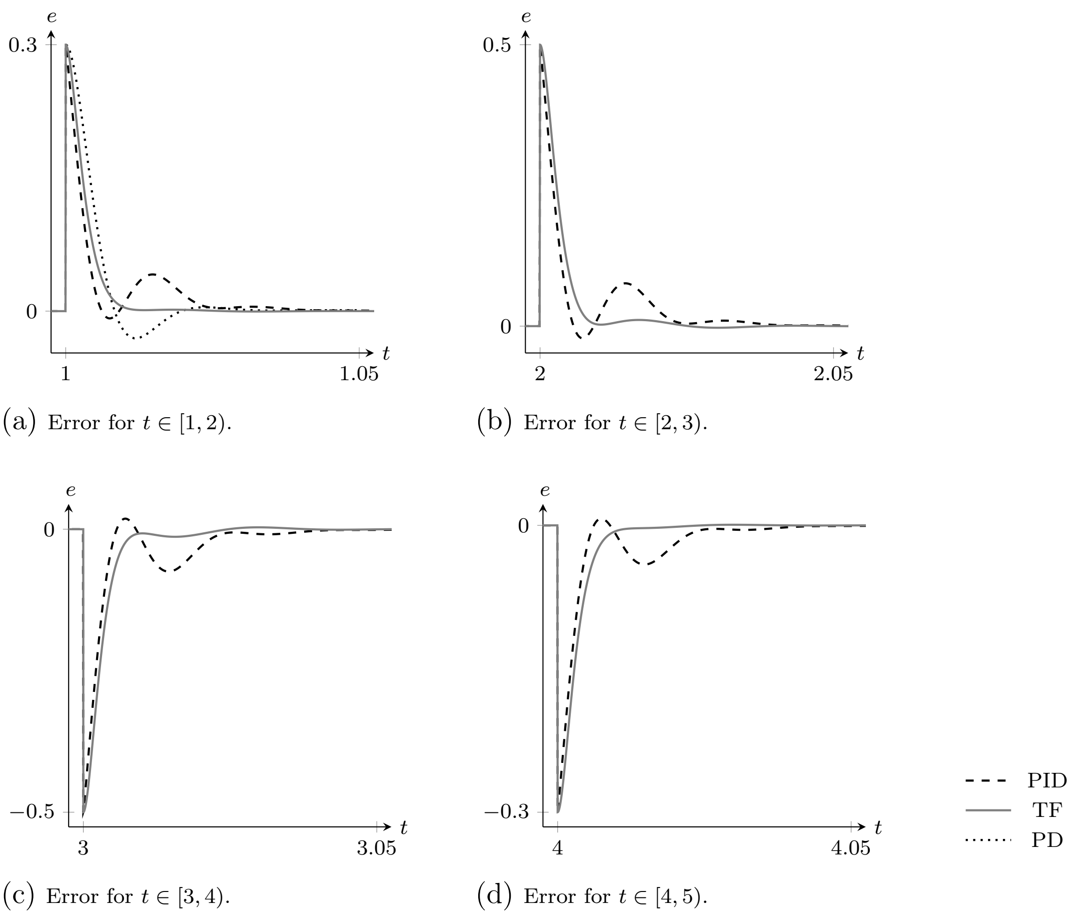


(d) Control with transfer function for $t \in [4, 5)$.

Figure 5.12.: Control with transfer function for the COMSOL model with two different parameter sets.

5.3. Discussion of Results

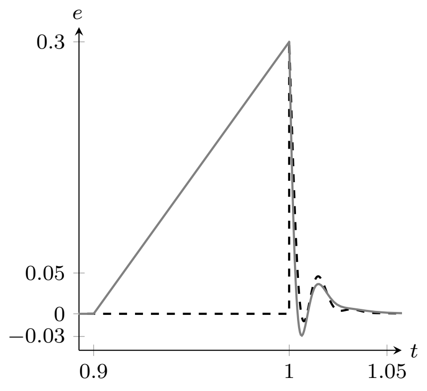
Looking at the difference between output signal φ of the plant and reference signal r indicates the controller performance. In Figure 5.13 the three different control designs for the multibody model are evaluated. The control error is plotted in the time ranges when the reference signal changes for the PID control, for the controller defined by the transfer function and the nonlinear PD control. Due to the step of the reference signal, defined by the Heaviside function, each controller has to react quickly. The PID control shows an oscillating behaviour of the output signal, the transfer function is calibrated in a robust way. The parameters of the PD control are chosen in that way that the output signal overshoots only once. All three control types match the reference signal. Since the calibration of the PD control is achieved manually, a difference in the dimension of 10^{-4} remains between output and reference signal.



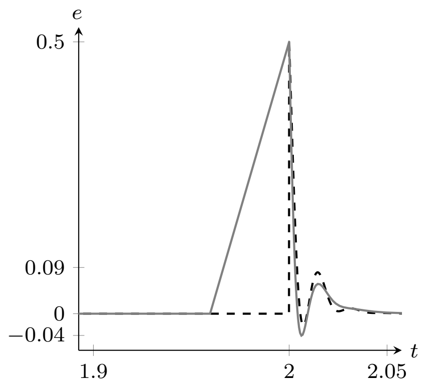
(a) Error for $t \in [1, 2)$.
(b) Error for $t \in [2, 3)$.
(c) Error for $t \in [3, 4)$.
(d) Error for $t \in [4, 5)$.

Figure 5.13.: Comparison of various control errors for the multibody model.

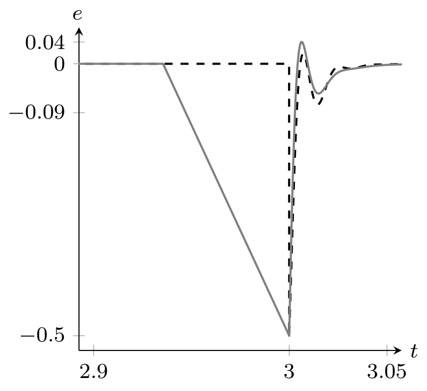
Figure 5.14 shows the difference between output and reference signal for the PID control first for the multibody model and second for the PDE model. The COMSOL simulation is based on fixed time steps $t = 0.02$, which explains the linear slope of the difference. Both models show an oscillating behaviour to the PID control. This is a result due to the same calibration method for the parameters.



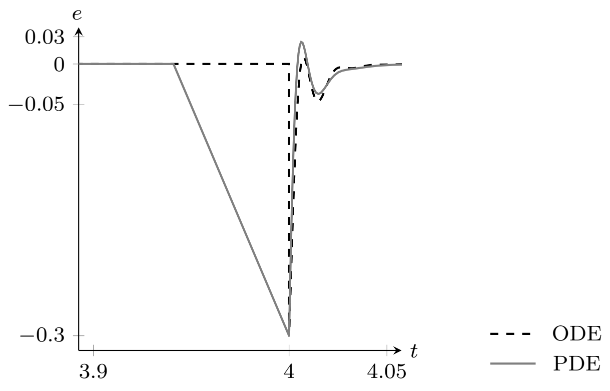
(a) Error for $t \in [1, 2)$.



(b) Error for $t \in [2, 3)$.



(c) Error for $t \in [3, 4)$.



(d) Error for $t \in [4, 5)$.

Figure 5.14.: PID control error for the ODE and PDE model.

Figure 5.15 shows the difference for both modelling approaches regarding a control design with a transfer function. The results are similar to the observations regarding the PID control. The multibody model and the PDE model show nearly no overshoot and match the reference signal quickly.
In conclusion, one can say that both modelling approaches, multibody models and models described by partial differential equations, can be formulated as linear state space representation. This allows to design for both modelling approaches linear control designs in the same software environment. This facilitates the control design process and the analysis of both models.
Furthermore, it is important to notice that even the linear control design shows satisfying or even better results than the nonlinear control design. This can be

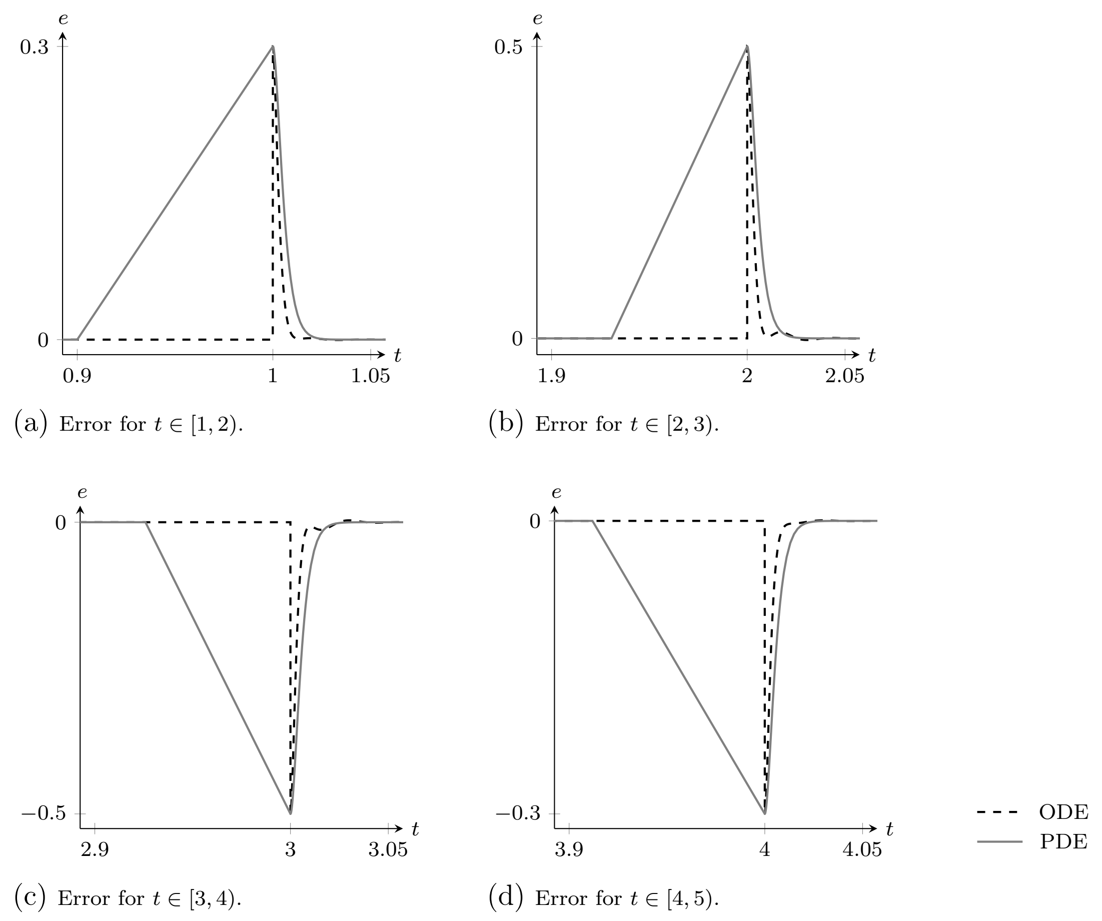


Figure 5.15.: Transfer function control error for the ODE and PDE model.

explained due to the aim of the controller. More complex output behaviour would have required nonlinear control design. The possibility of equation extraction in MapleSim is an useful tool which makes the usage of this software more attractive. The knowledge of the equations allows a better analysis of the underlying system.

6. Conclusion and Outlook

The conclusion of this thesis comprises two parts. First, the used simulation environments are discussed regarding their advantages and disadvantages. This is followed by the discussion of handling the different modelling approaches regarding their representation as dynamic systems. Last but not least, a short outlook gives an overview about possible future expansions of this work.

6.1. Benchmark of Simulation Environments

Mainly, three different simulation environments were used, Simulink including the multibody library in Simscape, MapleSim and COMSOL Multiphysics.
Although, in both multibody simulation environments, MapleSim and Simscape, the simulation models are described by ODEs, whereby the extraction of the equations is possible in MapleSim only. This facilitates further analysis and the usability of the simulation model for more investigation.
Another key point is the flexibility with respect to programming. The engine of MapleSim is based on Modelica, an object-oriented modelling language. This enables to edit the pre-implemented components by adapting the code. Furthermore, custom components can be defined with the use of Modelica programming.
Regarding to the modelling capabilities, MapleSim offers more possibilities than Simscape due to these two points. Focusing on the post processing of simulation results and flexibility of combination with the Simulink environment and its tools, Simscape offers more variety due to the interconnection to MATLAB.
The usability of COMSOL is different to the multibody libraries. This is caused not only by the modelling description by PDEs but as well regarding the modelling process. In contrast to the multibody simulation environments, COMSOL focuses on a solution oriented modelling process and not on the formulation of a mathematical model. Therefore, it offers a wide variety of visualisation for the results. LiveLinks to other software frameworks, as MATLAB and Solidworks, widen the application fields of simulation.

6.2. Biomechanical Models as Dynamic Systems

The aim of the thesis was the representation of biomechanical models, including different modelling approaches, as dynamic systems. The simulation models, established in chapter 4, involves one model described by ODEs and one by PDEs. In

chapter 5 a state space representation was derived or calculated numerically, respectively, for both simulation models. The application of system theory built the base for the used techniques. Due to the dependence of PDEs in time and as well space, a restriction of the output behaviour with respect to time was required for a state space representation.
Using the state space representation describing the dynamics of biomechanical models, allows the application of the same linear control designs for both simulation models. Nevertheless, both modelling approaches, the model in Simscape and the one in COMSOL, showed different sensitivity to control inputs. This inhibited to use one simulation model as calibration for the other. But this is a result due to their mathematical description and numerical differences in solving the simulation model. Furthermore, the multibody model gives a lot more flexibility of mathematical analysis than the PDE model due to its description by ODEs. This facilitates further investigation in a simulation loop.
The application of linear and nonlinear control design showed that for the considered control goal, following a defined reference signal, a linear controller results in a adequate behaviour of the plant.

6.3. Outlook

Future work could focuses on four main points. The established model for the flexion of the human knee in chapter 4 is simplified regarding the complex biomechanical structure of the human knee joint. The creation of custom components in MapleSim allows to implement nonlinear spring damper elements. The usage of nonlinear components in the simulation will improve the realistic behaviour of the ligaments. Furthermore, the analysis of the behaviour of nonlinear models in the simulation loop, in contrast to the linear models, is of interest. As well as the analysis of numerical aspects regarding the embedment of a linear or nonlinear model to a feedback loop, will be a task in the future.
The design of more sophisticated control designs, e.g. the definition of a trajectory as reference signal, would allow to simulate a human gait cycle. This could widen the application for biomechanical research questions and improve further biomechanical analysis of the interactions in the human body.
The simulation model in COMSOL was implemented following the hypotheses derived from the multibody model. In biomechanics PDEs are used for stress strain analysis due to their dependence on time and space. The development of a model described by PDEs analysing bone adaptation or the behaviour of soft tissue would give more application possibilities in biomechanical research.
Finally, the usage of `S`-functions would gives the opportunity to create an application of nonlinear control theory for the PDE model as well. Furthermore, this would allow the investigation of additional control designs, e.g. process control.

A. Nonlinear Control Laws

The nonlinear control theory gives a lot more possibilities than it was presented in this thesis. Two more nonlinear control laws are introduced for further investigation. Due to the fact, that they were not applicable to the established knee models, they are separated.

A.1. Control Laws based on Lyapunov Functions

With the help of Lyapunov stability, introduced above, it is possible to define control designs for nonlinear systems. For the first formulation of a control law, systems of the form

$$\dot{\boldsymbol{x}}_{\mathbf{1}} = \boldsymbol{f}_{\mathbf{1}}(\boldsymbol{x}_{\mathbf{1}}) + \boldsymbol{g}_{\mathbf{1}}(\boldsymbol{x}_{\mathbf{1}})x_2 \tag{A.1a}$$

$$\dot{x}_2 = u \tag{A.1b}$$

are considered, with the states $\boldsymbol{x}_{\mathbf{1}} \in \mathbb{R}^n, x_2 \in \mathbb{R}$, the input $u \in \mathbb{R}$ and the initial value $\boldsymbol{x}(0) = \boldsymbol{x}_0$. Systems of this kind are given in the so-called strict feedback form and are part of input linear systems. Therefore, a control law can be given by the following

Theorem A.1.1 (Integrator Backstepping). Given the nonlinear system (A.1) with the equilibrium state $\boldsymbol{x}_{\mathbf{1},\boldsymbol{R}} = \mathbf{0}$, i.e. $\boldsymbol{f}(\mathbf{0}) = \mathbf{0}$, a continuously differentiable function $\alpha(\boldsymbol{x}_{\mathbf{1}})$ with $\alpha(\mathbf{0}) = 0$ and a positive definite, radially unbounded Lyapunov function $V(\boldsymbol{x}_{\mathbf{1}})$. Additionally, the inequation

$$\frac{\partial V}{\partial \boldsymbol{x}_{\mathbf{1}}}\left(\boldsymbol{f}_{\mathbf{1}}(\boldsymbol{x}_{\mathbf{1}}) + \boldsymbol{g}_{\mathbf{1}}(\boldsymbol{x}_{\mathbf{1}})\alpha(\boldsymbol{x}_{\mathbf{1}})\right) \leq W(\boldsymbol{x}_{\mathbf{1}}) \leq 0$$

holds for a function $W(\boldsymbol{x}_{\mathbf{1}})$.

(i) If $W(\boldsymbol{x}_{\mathbf{1}})$ is negative definite, then there exists a control law $u = \alpha(\boldsymbol{x}_{\mathbf{1}}, x_2)$ so, that the equilibrium state $\boldsymbol{x}_{\mathbf{1},\boldsymbol{R}} = \mathbf{0}, x_2 = 0$ is global asymptotically stable. A possible control law is

$$u = \frac{\partial \alpha}{\partial \boldsymbol{x}_1}\left(\boldsymbol{f}_{\mathbf{1}}(\boldsymbol{x}_1) + \boldsymbol{g}_{\mathbf{1}}(\boldsymbol{x}_1)x_2\right) - \frac{1}{\gamma}\frac{\partial V}{\partial \boldsymbol{x}_1}\boldsymbol{g}_{\mathbf{1}}(\boldsymbol{x}_1) - c\left(x_2 - \alpha(\boldsymbol{x}_1)\right)$$

for $c > 0, \gamma > 0$. Additionally a Lyapunov function for the whole system is given by

$$V(\boldsymbol{x}_1, x_2) = V(\boldsymbol{x}_{\mathbf{1}}) + \tfrac{\gamma}{2}(x_2 - \alpha(\boldsymbol{x}_{\mathbf{1}}))^2.$$

(ii) If $W(\boldsymbol{x_1})$ is negative semi-definite, then there exists a control law $u = \alpha(\boldsymbol{x_1}, x_2)$ so that the states $\boldsymbol{x_1}(t), x_2(t)$ are bounded for all $t > 0$. Furthermore, the solution of the system converges for $t \to \infty$ to the greatest positive invariant set of

$$\mathcal{X} = \left\{ \begin{pmatrix} \boldsymbol{x_1} \\ x_2 \end{pmatrix} \in \mathbb{R}^{n+1} \colon W(\boldsymbol{x_1}) = 0 \text{ and } x_2 = \alpha(\boldsymbol{x_1}) \right\}.$$

It is now possible to expand this theorem to nonlinear systems of the form

$$\dot{\boldsymbol{x}}_{\boldsymbol{1}} = \boldsymbol{f_1}(\boldsymbol{x_1}, \boldsymbol{x_2}), \tag{A.2a}$$
$$\dot{\boldsymbol{x}}_{\boldsymbol{2}} = \boldsymbol{f_2}(\boldsymbol{x_1}, \boldsymbol{x_2}) + u, \tag{A.2b}$$

with the states $\boldsymbol{x_1} \in \mathbb{R}^n, \boldsymbol{x_2} \in \mathbb{R}^p$ and the input $u \in \mathbb{R}^p$. The control law u is modified to

$$u = \frac{\partial}{\partial \boldsymbol{x_1}} \alpha(\boldsymbol{x_1}) \boldsymbol{f_1}(\boldsymbol{x_1}, \boldsymbol{x_2}) - \left(\frac{\partial}{\partial \boldsymbol{x_1}} V(\boldsymbol{x_1}) G(\boldsymbol{x_1}, \boldsymbol{x_2} - \alpha(\boldsymbol{x_1})) \right)^T - c(\boldsymbol{x_2} - \alpha(\boldsymbol{x_1})) - \boldsymbol{f_2}(\boldsymbol{x_1}, \boldsymbol{x_2})$$

with the auxiliary function

$$G = \int_0^1 \frac{\partial}{\partial \alpha(\boldsymbol{x_1} + \lambda \boldsymbol{x_2})} \boldsymbol{f_1}(\boldsymbol{x_1}, \alpha(\boldsymbol{x_1} + \lambda \boldsymbol{x_2})) \mathrm{d}\lambda.$$

The next example illustrates the implementation of a control law by Lyapunov functions using the integrator backstepping theorem.

Example A.1.2. A model of a rotary flexible joint is considered. The system consists of a body holding one arm as it is depicted in Figure A.1. The state

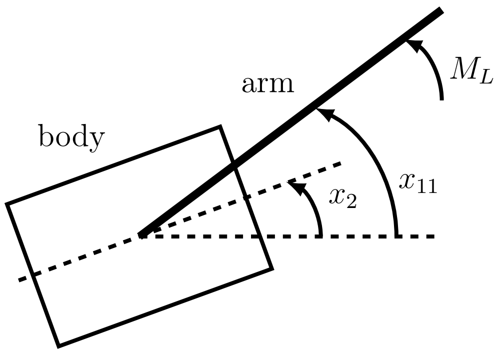


Figure A.1.: Illustration for rotary flexible joint.

variables x_{11}, x_{12} describe the angle and the angle velocity of the arm. They are merged in the vector $\boldsymbol{x_1} = (x_{11}, x_{12})^T$. The angle velocity of the body is the state variable x_3. A constant load torque M_L and the moment of inertia I_a are applied on the arm. Springs are connected between the arm and the body applying an aligning torque $c_0(x_{11} - x_2)$, $c_0 > 0$. The constant $d_a > 0$ is the friction coefficient defining the friction between arm and body. Hence, the differential equations of the system are given by

$$\dot{\boldsymbol{x}}_{\boldsymbol{1}} = \begin{pmatrix} x_{12} \\ \frac{1}{I_a}(M_L - c_0(x_{11} - x_2) - d_a x_{12}) \end{pmatrix}, \tag{A.3a}$$

$$\dot{x}_2 = x_3 = u. \tag{A.3b}$$

This description is given in strict feedback form which makes it simple to define a control law by the integrator backstepping following the system description as in (A.1). With the system description (A.3), the functions are given by

$$\begin{aligned} \boldsymbol{f}_{\boldsymbol{1}}(\boldsymbol{x}_{\boldsymbol{1}}) &= \begin{pmatrix} x_{12} \\ \frac{1}{I_a}(M_L - c_0 x_{11} - d_a x_{12}) \end{pmatrix}, & \boldsymbol{g}_{\boldsymbol{1}}(\boldsymbol{x}_{\boldsymbol{1}}) &= \begin{pmatrix} 0 \\ \frac{1}{I_a} c_0 \end{pmatrix}, \\ f_2(\boldsymbol{x}_{\boldsymbol{1}}, x_2) &= 0, & g_2(\boldsymbol{x}_{\boldsymbol{1}}, x_2) &= 1. \end{aligned}$$

Using the integrator backstepping designs a control law fulfilling the dynamics

$$\dot{\boldsymbol{x}}_{\boldsymbol{1}} = \begin{pmatrix} x_{12} \\ \frac{1}{I_a}(M_L - M(x_{11}) - d x_{12}) \end{pmatrix}. \tag{A.4}$$

Let the aligning torque $M(x_{11})$ be a smooth and increasing function with the property $M(x_{11})x_{11} > 0$ for $x_{11} \neq 0$.
In the first approach for a control design, only the arm is considered. Hence, it is possible to calculate the input $u = \alpha(\boldsymbol{x}_{\boldsymbol{1}})$ by comparing the ideal dynamics given by (A.4) to the real system in (A.3a). This leads to

$$\alpha(\boldsymbol{x}_{\boldsymbol{1}}) = x_{11} + \frac{(d_a - d)x_{12} - M(x_{11})}{c_0}.$$

Second, the stability of Lyapunov for this control law is proven. The transformation $\boldsymbol{z}_{\boldsymbol{1}} = \boldsymbol{x}_{\boldsymbol{1}} - \boldsymbol{x}_{\boldsymbol{1R}}$ for the equilibrium state $\boldsymbol{x}_{\boldsymbol{1R}}$ and the Lyapunov function

$$V_1 = \int\limits_0^{z_{11}} (M(\xi + x_{11R}) - M_L)\, \mathrm{d}\xi + \frac{I_a}{2} z_{12}^2.$$

are introduced. The derivative of the Lyapunov function is calculated by

$$\begin{aligned} \dot{V}_1(\boldsymbol{z}_{\boldsymbol{1}}) &= \nabla V_1^T \dot{\boldsymbol{z}}_{\boldsymbol{1}} = \begin{pmatrix} M(x_{11}) - M_L & I_a x_{12} \end{pmatrix} (\boldsymbol{f}_{\boldsymbol{1}}(\boldsymbol{x}_{\boldsymbol{1}}) + \boldsymbol{g}_{\boldsymbol{1}}(\boldsymbol{x}_{\boldsymbol{1}})\alpha) \\ &= \begin{pmatrix} M(x_{11}) - M_L & I_a x_{12} \end{pmatrix} \begin{pmatrix} x_{12} \\ \frac{M_L - M(x_{11}) - d x_{12}}{I_a} \end{pmatrix} = -d x_{12}^2 \leq 0. \end{aligned}$$

This shows, that the Lyapunov function is positive definite with a negative semidefinite derivative. This implies the stability of Lyapunov for the given control law α

for the system $\boldsymbol{x_1}$.
It is now possible to expand the control law for all three state variables by expanding the Lyapunov function to

$$V_2(\boldsymbol{z_1}, z_2) = V_1(\boldsymbol{z_1}) + \frac{\gamma}{2} z_2^2,$$

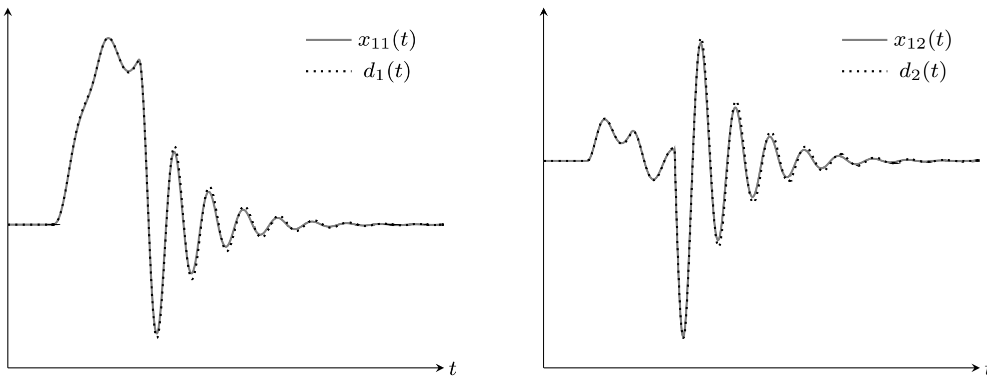
with the error $z_2 = (x_2 - \alpha(\boldsymbol{x_1}))$ of the real state x_2 and the desired state $\alpha(\boldsymbol{x_1})$. The Lyapunov function for the whole system is positive definite due to V_1. Calculating the derivative leads to

$$\begin{aligned}
\dot{V}_2 &= \dot{V}_1 + \gamma z_2 \dot{z}_2 = \dot{V}_1 + \gamma(x_2 - \alpha(\boldsymbol{x_1}))(\dot{x_2} - \dot{\alpha}) \\
&= \frac{\partial V_1}{\partial \boldsymbol{x_1}} (\boldsymbol{f_1} + \boldsymbol{g_1} x_2) + \gamma(x_2 - \alpha(\boldsymbol{x_1})) \left(g_2 u - \frac{\partial \alpha_1}{\partial \boldsymbol{x_1}} (\boldsymbol{f_1} + \boldsymbol{g_1} x_2) \right) \\
&= \frac{\partial V_1}{\partial \boldsymbol{x_1}} (\boldsymbol{f_1} + \boldsymbol{g_1} \alpha(\boldsymbol{x_1})) + \gamma(x_2 - \alpha(\boldsymbol{x_1})) \left(g_2 u - \frac{\partial \alpha}{\partial \boldsymbol{x_1}} (\boldsymbol{f_1} + \boldsymbol{g_1} x_2) + \frac{1}{\gamma} \frac{\partial V_1}{\partial \boldsymbol{x_1}} g_1 \right).
\end{aligned}$$

A possible control law with the parameters $k > 0, \gamma > 0$ is given by

$$\begin{aligned}
u &= \frac{\partial}{\partial \boldsymbol{x_1}} \alpha(\boldsymbol{x_1}) \left(\boldsymbol{f_1}(\boldsymbol{x_1}) + \boldsymbol{g_1}(\boldsymbol{x_1}) x_2 \right) - \frac{1}{\gamma} \frac{\partial}{\partial \boldsymbol{x_1}} V(\boldsymbol{x_1}) \boldsymbol{g_1}(\boldsymbol{x_1}) - k(x_2 - \alpha(\boldsymbol{x_1})) \\
&= \left(1 - \tfrac{\dot{M}(x_{11})}{c_0} \quad \tfrac{d - d_a}{c_0} \right) \left(\boldsymbol{f_1}(\boldsymbol{x_1}) + \boldsymbol{g_1}(\boldsymbol{x_1}) x_2 \right) - \frac{1}{\gamma} \left(M(x_{11}) - M_L \quad I_a x_{12} \right) \boldsymbol{g_1} \\
&\quad - k(x_2 - \alpha(\boldsymbol{x_1})) \\
&= x_{12} - k \left(x_2 - x_{11} + \frac{(d - d_a) x_{12} + M(x_{11})}{c_0} \right) \\
&\quad - \frac{d - d_a}{c_0} \frac{x_2 c_0 - c_0 x_{11} - d_a x_{12} + M_L}{I_a} - \frac{x_{12} c_0}{\gamma}.
\end{aligned}$$

This input leads to a negative semi-definite derivative $\dot{V}_2 \leq 0$. This proves that the closed circle is stable. The parameter γ regulates the influence of the Lyapunov function V_1.
The results of the simulation with the calculated input u are depicted in Figure A.2. The plots show the real coordinates x_{11}, x_{12} and the desired values of the coordinates d_1, d_2 where a good behaviour of the controller can be observed.



(a) Angle of the arm $x_{11}(t)$ plotted against its reference signal $d_1(t)$.

(b) Velocity of the arm $x_{12}(t)$ plotted against its reference signal $d_2(t)$.

Figure A.2.: Integrator Backstepping design for the rotable flexible joint.

A.2. Exact Linearisation

In the following, systems of the form

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})u \tag{A.5a}$$

$$y = h(\boldsymbol{x}) \tag{A.5b}$$

with the state $\boldsymbol{x} \in \mathbb{R}^n$, the input $u \in \mathbb{R}$ and the output $y \in \mathbb{R}$ are considered, which are part of the affine input systems. The functions $\boldsymbol{f}(\boldsymbol{x}), \boldsymbol{g}(\boldsymbol{x})$ describe smooth vector fields and $h(\boldsymbol{x})$ a smooth function. For the method of the exact linearisation, the Lie derivative is needed, which is introduced in

Definition A.2.1 (Lie derivative). The Lie derivative L_f of a differentiable function h with respect to a vector field $\boldsymbol{f}$ is defined by

$$L_{\boldsymbol{f}}h(\boldsymbol{x}) = \frac{\partial h}{\partial \boldsymbol{x}}\boldsymbol{f}(\boldsymbol{x}) = \nabla h^T(\boldsymbol{x})\boldsymbol{f}(\boldsymbol{x}).$$

The Lie derivative $L_{\boldsymbol{f}}^k h(\boldsymbol{x})$ for $k \in \mathbb{N}$ is defined by the recursion

$$L_{\boldsymbol{f}}^k h(\boldsymbol{x}) = L_{\boldsymbol{f}}^{k-1} h(\boldsymbol{x}), \quad L_{\boldsymbol{f}}^0 h(\boldsymbol{x}) = h(\boldsymbol{x}).$$

Furthermore, the relative degree is introduced to determine the recursive Lie derivative which is not 0 anymore.

Definition A.2.2 (Relative degree). The system (A.5) has relative degree $\delta \in \mathbb{N}$ at $\hat{\boldsymbol{x}}$ if

$$L_{\boldsymbol{g}} L_{\boldsymbol{f}}^k h(\boldsymbol{x}) = 0, \quad k = 0 \dots \delta - 2 \text{ for all } \boldsymbol{x} \text{ fulfilling } \|\boldsymbol{x} - \hat{\boldsymbol{x}}\| < \varepsilon$$

and

$$L_{\boldsymbol{g}} L_{\boldsymbol{f}}^{\delta-1} h(\hat{\boldsymbol{x}}) \neq 0.$$

Investigating in (A.5) the change in time of the output y leads to

$$\dot{y} = \frac{\partial h(\boldsymbol{x})}{\boldsymbol{x}} \dot{\boldsymbol{x}} = L_{\boldsymbol{f}} h(\boldsymbol{x}) + L_{\boldsymbol{g}} h(\boldsymbol{x}) u. \tag{A.6}$$

Assuming that $L_{\boldsymbol{g}} h(\boldsymbol{x}) \neq 0$, allows to introduce a state feedback

$$u = \frac{1}{L_{\boldsymbol{g}} h(\boldsymbol{x})} (v - L_{\boldsymbol{f}} h(\boldsymbol{x}))$$

with a new input v. This transforms the system (A.5) in a linear system of first order of the form

$$\dot{y} = v.$$

Assuming that $L_{\boldsymbol{g}} h(\boldsymbol{x}) = 0$ in (A.6), leads to

$$\ddot{y} = \frac{\partial L_{\boldsymbol{f}} h(\boldsymbol{x})}{\boldsymbol{x}} \dot{\boldsymbol{x}} = \frac{\partial L_{\boldsymbol{f}} h(\boldsymbol{x})}{\boldsymbol{x}} (\boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x}) u) = L_{\boldsymbol{f}}^2 h(\boldsymbol{x}) + L_{\boldsymbol{g}} L_{\boldsymbol{f}} h(\boldsymbol{x}) u.$$

Introducing the relative degree δ of the system allows to analyse the derivative of y where the input u is explicitly given, as shown in

$$\begin{array}{rcl} y &=& h(\boldsymbol{x}), \\ \dot{y} &=& L_{\boldsymbol{f}} h(\boldsymbol{x}) + \underbrace{L_{\boldsymbol{g}} h(\boldsymbol{x})}_{=0} u, \\ \ddot{y} &=& L_{\boldsymbol{f}}^2 h(\boldsymbol{x}) + \underbrace{L_{\boldsymbol{g}} L_{\boldsymbol{f}} h(\boldsymbol{x})}_{=0} u, \\ & \vdots & \\ y^{(\delta-1)} &=& L_{\boldsymbol{f}}^{\delta-1} h(\boldsymbol{x}) + \underbrace{L_{\boldsymbol{g}} L_{\boldsymbol{f}}^{\delta-2} h(\boldsymbol{x})}_{=0} u, \\ y^{(\delta)} &=& L_{\boldsymbol{f}}^{\delta} h(\boldsymbol{x}) + L_{\boldsymbol{g}} L_{\boldsymbol{f}}^{\delta-1} h(\boldsymbol{x}) u. \end{array}$$

The state feedback control law

$$u = \frac{1}{L_{\boldsymbol{g}} L_{\boldsymbol{f}}^{\delta-1} h(\boldsymbol{x})} (v - L_{\boldsymbol{f}}^{\delta} h(\boldsymbol{x}))$$

leads to a linear input-output behaviour for the δ-th derivation of the output

$$y^{(\delta)} = v.$$

Solving this equation for the output y yields in a n times integration. These insights can be summarised in

Theorem A.2.3 (Exact linearisation). Consider the system given in (A.5). Suppose the relative degree $\delta \leq n$ at $\hat{\boldsymbol{x}}$. The control law for the state feedback

$$u = \frac{1}{L_{\boldsymbol{g}} L_{\boldsymbol{f}}^{\delta-1} h(\boldsymbol{x})} (v - L_{\boldsymbol{f}}^{\delta} h(\boldsymbol{x}))$$

transforms the system with a linear input-output behaviour and the new input $v \in \mathbb{R}$. The transfer function

$$G(s) = \frac{1}{s^{\delta}}$$

calculates the output y.

Again, this control design is illustrated with an

Example A.2.4. The inverted double pendulum, considered in Example 3.4.10 allows to study more complex control designs. The state variables of the double pendulum are extended with the acceleration of the position x of the cart. Again, the system of second order differential equations for the state variables are described by a mass matrix M and a right hand side b containing external forces acting on the derivatives of the state variables. This concludes to an equation of the form

$$M \begin{pmatrix} \ddot{x} \\ \ddot{\varphi}_1 \\ \ddot{\varphi}_2 \end{pmatrix} = b + \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} u.$$

The input u is considered as an external force $\boldsymbol{F}$ acting on the cart. The mass matrix and the vector field b are given by

$$\begin{aligned} M &= \begin{pmatrix} m_m + m_1 + m_2 & \left(\frac{m_1}{2} + m_2\right) l_1 \cos\varphi_1 & \frac{m_2}{2} l_2 \cos\varphi_2 \\ \left(\frac{m_1}{2} + m_2\right) l_1 \cos\varphi_1 & \left(\frac{m_1}{3} + m_2\right) l_1^2 & \frac{m_2}{2} l_1 l_2 \cos(\varphi_1 - \varphi_2) \\ \frac{m_2}{2} l_2 \cos\varphi_2 & \frac{m_2}{2} l_1 l_2 \cos(\varphi_1 - \varphi_2) & \frac{m_2}{3} l_2^2 \end{pmatrix}, \\ b &= \begin{pmatrix} \left(\frac{m_1}{2} + m_2\right) l_1 \sin\varphi_1 \dot{\varphi}_1^2 + \frac{m_2}{2} l_2 \sin\varphi_2 \dot{\varphi}_2 \\ \frac{m_2}{2} l_1 l_2 \sin(\varphi_2 - \varphi_1) \dot{\varphi}_2 + \left(\frac{m_1}{2} + m_2\right) g l_1 \sin\varphi_1 \\ \frac{m_2}{2} l_2 (g \sin\varphi_2 + l_1 \sin(\varphi_1 - \varphi_2) \dot{\varphi}_1^2) \end{pmatrix}. \end{aligned}$$

The mass of the cart and the pendulums are considered as $m_m = 0.02$ kg, $m_1 = m_2 = 0.01$ kg, the length of the pendulums as $l_1 = 0.5$ m, $l_2 = 0.7$ m and the gravitation constant as $g = 9.81$ ms^{-2}. The differential equations of second order for the system are given by $M^{-1}b$. Considering as output of the system $h(\boldsymbol{x}) = x$ and calculating the derivatives of the output leads to the equations

$$\begin{aligned} y &= x, \\ \dot{y} &= \dot{x}, \\ \ddot{y} &= \ddot{x} = M^{-1}b(1) + u = v. \end{aligned}$$

Expressing the input as

$$u = v - M^{-1}b(1)$$

leads to the equation of the output for the second derivative

$$\ddot{y} = v.$$

Following the Theorem A.2.3 leads to the formulation of the equations for the derivatives of the output by the Lie Derivatives as

$$\begin{aligned} y &= h(\boldsymbol{x}), \\ \dot{y} &= \frac{\partial h}{\partial \boldsymbol{x}}(f(\boldsymbol{x}) + g(\boldsymbol{x})u) = L_f h(\boldsymbol{x}) + \underbrace{L_g h(\boldsymbol{x})}_{=0} u, \\ \ddot{y} &= \frac{\partial L_f h(\boldsymbol{x})}{\partial \boldsymbol{x}} \dot{x} = \frac{\partial L_f h(\boldsymbol{x})}{\partial \boldsymbol{x}}(f(\boldsymbol{x}) + g(\boldsymbol{x})u) = L_f^2 h(\boldsymbol{x}) + \underbrace{L_g L_f h(\boldsymbol{x})}_{=1} u. \end{aligned}$$

This implies a relative degree of 2 for the system. Reformulating the differential equations with the new input leads to

$$\begin{aligned} \ddot{x} &= v, \\ \ddot{\varphi}_1 &= M^{-1}b(2), \\ \ddot{\varphi}_2 &= M^{-1}b(3). \end{aligned}$$

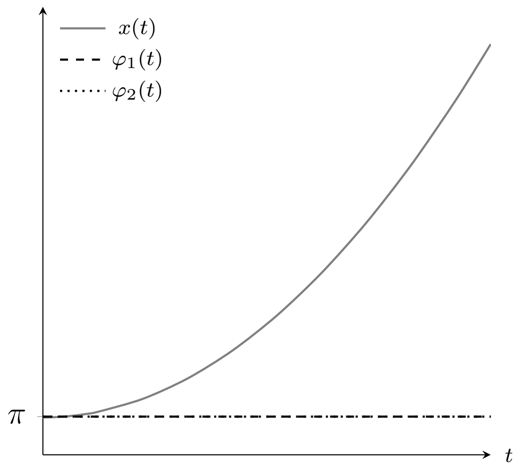


Figure A.3.: Exact linearisation for the inverted double pendulum.

In Figure A.3 the simulation results of the solution of the position of the cart $x(t)$ and the angles $\varphi_1(t), \varphi_2(t)$ are depicted. The results show that the equilibrium state at $(\varphi_1, \varphi_2) = (\pi, \pi)$ is stabilised with the exact linearisation. The position of the cart is increasing which is caused by the reformulation, due to the integration of the constant value $v = 1$.

Nomenclature

Abbreviations

Symbol	Description
BIBO	Bounded-Input Bounded-Output
FEA	Finite Element Analysis
FEM	Finite Element Method
MIMO	Multiple-Input Multiple-Output
LTI	Linear Time Invariant
ODE	Ordinary Differential Equation
PDE	Partial Differential Equation
SISO	Single-Input Single-Output

Mathematical Symbols and Functions

Symbol	Description
A	Capital letters for matrices
$\boldsymbol{x}$	Boldface for vectors
x	Normal font scalars
$()^T$	Matrix or Vector Transpose
H	Heaviside function

The Heavside H function is defined by

$$\mathrm{H}(t) = \begin{cases} 1, & t \geq 0, \\ 0, & t < 0. \end{cases}$$

List of Figures

List of Tables

Bibliography

[1] Jürgen Adamy. *Nichtlineare Regelungen.* Springer-Verlag Berlin Heidelberg, 2009.

[2] Jorge Ambrósio et al. "Multibody biomechanical models of the upper limb". In: *Procedia IUTAM* 2 (2011), pp. 4–17.

[3] Marjan Bahraminasab et al. "Multi-objective design optimization of functionally graded material for the femoral component of total knee replacement". In: *Materials and Design* 53 (2014), pp. 159–173.

[4] Osman Balci. "Verification, Validation, and Testing". In: *Handbook of simulation: principles, methodology, advances, applications, and practice.* Ed. by Inc. John Wiley & Sons. 1998, pp. 335–393.

[5] Philipp Beckerle. "Human-machine-centered design and actuation of lower limb prosthetic systems". PhD thesis. Technische Universität Darmstadt, 2014.

[6] Philipp Beckerle et al. "User-centered Prosthetic Design: A methodological approach to transfer psychological factor to technical development". In: *Technically Assisted Rehabilitation - TAR 2013, 4th European Conference, March 14 - 15, Berlin.* 2013.

[7] L. Blankevoort and R. Huiskes. "Ligament-Bone Interaction in a Three-Dimensional Model of the Knee". In: *Journal of Biomechanical Engineering* 113.3 (1991), p. 263.

[8] Katherine H. Bloemker et al. "Computational Knee Ligament Modeling Using Experimentally Determined Zero-Load Lengths". In: *The Open Biomedical Engineering Journal* 6 (2012), pp. 33–41.

[9] Paul Brinckmann et al. *Orthopädische Biomechanik.* Wissenschaftliche Schriften der WWU Münster, 2012, p. 485.

[10] François E. Cellier. *Continuous System Modelling.* Springer-Verlag New York, Inc., 1991.

[11] Anthony J. Pritchard Diederich J. Hinrichsen. *Mathematical Systems Theory I: Modelling, State Space Analysis, Stability and Robustness.* Ed. by L. Sirovich J.E. Marsden S.S. Antman. Springer-Verlag Berlin Heidelberg, 2005.

[12] Jay W. Forrester. *Industrial Dynamics.* MIT Press, 1969.

[13] Thomas Schaeffer Georg Rill. *Grundlagen und Methoden der Mehrkörpersimulation.* Vol. 3. Springer Vieweg, 2017.

[14] Trent M. Guess et al. “A multibody knee model with discrete cartilage prediction of tibio-femoral contact mechanics”. In: *Computer Methods in Biomechanics and Biomedical Engineering* 16.3 (2013), pp. 256–270.

[15] Trent M. Guess et al. “A subject specific multibody model of the knee with menisci”. In: *Medical Engineering and Physics* 32.5 (2010), pp. 505–515.

[16] Wilhelm Haager. *Regelungstechnik*. 2nd ed. Hölder-Pichler-Tempsky, 2005.

[17] R. Huiskes and R. Boeklagen. “Mathematical Shape Optimization of Hip Prosthesis Design”. In: *Journal of Biomechanics* 22.8/9 (1989), pp. 793–804.

[18] Andreas Kugi. *Automatisierung*. Lecture Notes. TU Wien, 2016.

[19] Andreas Kugi. *Regelungssysteme*. Lecture Notes. TU Wien, 2017.

[20] Ruth Leskovar, Andreas Körner, and Felix Breitenecker. “System Based Modelling Approach for Biomechanical Models in the Field of Prosthetics”. In: *2017 UKSim-AMSS 19th International Conference on Computer Modelling Simulation (UKSim)*. 2017, pp. 13–18.

[21] Margarida Machado et al. “Development of a planar multibody model of the human knee joint”. In: *Nonlinear Dynamics* 60.3 (2010), pp. 459–478.

[22] Heinz Mann, Horst Schiffelgen, and Rainer Froriep. *Einführung in die Regelungstechnik*. 10th. Carl Hanser Verlag München, 2005.

[23] Dietmar Möller. *Ein geschlossenes nichtlineares Modell zur Simulation des Kurzzeitverhaltens des Kreislaufs und seine Anwendung zur Identifikation*. Springer-Verlag Berlin Heidelberg, 1981.

[24] Wolfgang Preuß. *Funktionaltransformationen Fourier-, Laplace- und Z-Transformation*. 2nd ed. München: Hanser, 2009.

[25] Kelc Robi et al. “The Physiology of Sports Injuries and Repair Processes”. In: *Current Issues in Sports and Exercise Medicine*. June. 2013.

[26] Robert G. Sargent. “Verification and Validation of Simulation Models”. In: *Proceedings of the 2007 Winter Simulation Conference*. 2007, pp. 124–137.

[27] Rudolf Schabus and Elisabeth Bosina. *Das Knie*. Springer-Verlag Wien New York, 2007.

[28] Jos A.W.M van Schijndel. “Implementation of COMSOL Multiphysics®in Simulink®”. In: *Proceedings of the 2014 COMSOL Conference in Cambridge*. 2014.

[29] *SimTK*. 2018. URL: `https://simtk.org`.

[30] Ferdinand Svaricek. *Zuverlässige numerische Analyse linearer Regelungssysteme*. B.G. Teubner Stuttgart, 1995.

[31] Bastian Welke et al. “Multi-Body Simulation of Various Falling Scenarios for Determining Resulting Loads at the Prosthesis Interface of Transfemoral Amputees with Osseointegrated Fixation”. In: *Journal of Orthopaedic Research* 31.7 (2013), pp. 1123–1129.

[32] J. Wismans et al. “A three-dimensional mathematical model of the knee-joint”. In: *Journal of Biomechanics* 13.8 (1980).

[33] Julius Wolff. *Das Gesetz der Transformation der Knochen.* Ed. by Georg Bergmann, Georg Duda, and Charité Berlin Julius Wolff Institut. Hirschwald, 2010.

[34] Savio L Woo et al. “Injury and Repair of Ligaments and Tendons”. In: *Annual Review of Biomedical Engineering* 2.1 (2000), pp. 83–118.

[35] Savio L.Y. Woo et al. “Biomechanics of knee ligaments”. In: *American Journal of Sports Medicine* 27.4 (1999), pp. 533–543.

[36] Bernard P. Zeigler. *Theory of Modelling and Simulation.* John Wiley & Sons, Inc., 1976.

Index