Rudolf Reiter, BSc

Modelling of specific nonlinear drivetrain dynamics

MASTER'S THESIS

to achieve the university degree of

Diplom-Ingenieur

Master's Degree Program

Electrical Engineering

submitted to

Graz University of Technology

Supervisors:

Ao.Univ.-Prof. Dipl.-Ing. Dr.techn. tit.Univ.-Prof. Anton Hofer

Institute of Automation and Control

Dipl.-Ing. Dr.techn. Michael Stolz

Dipl.-Ing. Markus Bachinger

VIRTUAL VEHICLE Research Center

Graz, Dezember 2015

AFFIDAVIT

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

....

Date Signature

ACKNOWLEDGEMENT

This work was accomplished at the VIRTUAL VEHICLE Research Center in Graz, Austria. I would like to acknowledge the financial support of the COMET K2 - Competence Centers for Excellent Technologies Programme of the Austrian Federal Ministry for Transport, Innovation and Technology (bmvit), the Austrian Federal Ministry of Science, Research and Economy (bmwfw), the Austrian Research Promotion Agency (FFG), the Province of Styria and the Styrian Business Promotion Agency (SFG).

Moreover, I would like to express my thanks to Prof. Anton Hofer for the university support of this master thesis.

Particularly I would also like to show my gratitude to Dr. Michael Stolz and Dipl.-Ing. Markus Bachinger for accompanying the entire development process of this work and Dr. Georg Stettinger for his recommendations and review of this thesis.

Graz,

Reiter Rudolf

ABSTRACT

Contemporary drivetrain simulation is facing new challenges, due to the rising complexity of automotive functionality. Engineers rely on highly accurate simulation results, which moreover have to meet strict frame conditions, as real time performance and efficiency factors.

Automotive drivetrain simulation is an intense field of research all over the world, consequently several approaches were developed to handle outstanding simulation problems such as friction elements. Little research has been done in the fields of integrating further nonlinear elements into the total simulation. A state of the art linear drivetrain simulation method with an additional approach that handles friction elements is taken as a basis for investigating essential nonlinear elements, which improve accuracy substantially. Three major elements are chosen to show impacts on the simulation results and study its integration potential into the total drivetrain simulation. Since nonlinear spring/damper elements are in some kind of forming always part of drivetrains, they are subject of the first investigation. Several algorithms are figured out for handling particularly piecewise affine spring elements. A promising approach regarding the computation time and accuracy was found with the method of switching affine systems for the nonlinear spring and handling damper elements as differently parameterized clutches. Secondly, the highly nonlinear phenomenon of a collision at some point in the drivetrain, particularly in the gearbox, was of interest. By means of a nonlinear algorithm, which pre-calculates system states after a collision and appends torque inputs, collisions can be considered appropriately. Both of the first nonlinear elements require an approach of a "step size adjustment" which leads to negligible errors. Owing to the fact, that friction elements are already integrated into the simulation excellently, the attempt is made to treat the road/tire interaction as clutch model and turned out to show feasible results.

All in all, it is possible to integrate nonlinear elements to achieve higher accuracy, but for a final integration into an embedded real time system, computation time has to be weighed against accuracy improvement.

TABLE OF CONTENTS

1 INTRODUCTION

As the simulation of automotive systems, in particular drivetrain simulation, takes an increasingly significant part in automotive engineering, effective and accurate simulation approaches are required [1]. There are either methods for online simulation, where computation time plays an important role and offline simulation, where the focus lies on the accuracy of the results [2]. Since state-of-the-art drivetrain models are subject of increasing complexity, the approaches should be kept as simple as possible to keep transparency.

Effective operative solutions exist for drivetrain modelling [3], but they so far rely on the linearization of several system parts. As linearization always comes with a loss of accuracy, methods are appreciated, which embed fundamental nonlinear behavior. As a matter of fact, nonlinear approaches include several challenges, especially when nonlinear elements are merged into an existing modelling approach which already handles some nonlinear phenomena with nonlinear methods [3]. The main challenges to conquer with new algorithms are a large step size of time discretization, the real time necessity and the static friction problem of clutches.

Main points of interest in nonlinear drivetrain modelling are nonlinear spring-damper-mass-systems, which occur on several places in modern drivetrain topologies and represent fundamental mechanical behavior. Several nonlinearities of both, springs and dampers, are investigated and furthermore inspected for time discretization methods to be merged into existing drivetrain models.

The thesis is structured into an essential preliminary part (chapter 2), where a simulation model was created, following an approach of [3], which is used as a basis for the following chapters that discuss possible nonlinear extensions, particularly a nonlinear spring and damper force function (chapter 3), a gearwheel clearance (chapter 4) and nonlinear tire models (chapter 5).

2 Reduced simulation model for a certain automotive transmission system

2.1 Introduction

For the purpose of creating a basic environment for the development of new features, an automatic automotive transmission model of a hybrid engine was chosen, which for a model was created, with respect to existing automotive gearbox modelling methods [3].

The reference model of the automotive transmission system in [3] depends on the development environment of an encompassing large project, thus it involves a vast number of optional and extended components. For the scope of this work the model was implemented paying special attention to a lean model structure. As a first step the reference model was simplified and shaped according to literature [3], which describes an approach for real-time drivetrain modelling. The main functionality of the original model was reproduced with no divergence.

The description of the real-time drivetrain modelling approach is structured well into blocks [3]. This led to the usage of MATLAB®/Simulink® “MATLAB function blocks”, which embed MATLAB®-code into the MATLAB®/Simulink® model. These blocks make it easy for developers to associate the description of the paper with the programmed MATLAB®/Simulink® model and furthermore these blocks have the advantage of being treated as “atomic units” in Simulink. “Atomic units” are units, which perform their calculations not before they have loaded every input value and which deliver outputs simultaneously. In this way algebraic loops are revealed more likely and the signal flows are easy to follow.

By means of the designed model and the paper, future developers are easily able to understand and manipulate the functionality of the drivetrain model in Simulink.

In a modern drivetrain simulation, it is not possible to consider all nonlinear parts, due to the restricted computation power. In practice it is sufficient to use linear spring-damper elements and introduce only friction elements as nonlinear components.

Nevertheless to increase accuracy and analyze nonlinear behavior, it is subject to investigate several nonlinear extensions of the model and compare the approaches regarding accuracy, implementation afford, computation time and consistency with existing models.

Following nonlinear behavior is of major interest, since in some cases they might lead to fundamental differences as for example spring-damper resonance frequencies or amplitudes.

2.2 MODEL DESCRIPTION

2.2.1 FUNCTIONALITY

Usually drivetrain models consist of simple spring-damper-mass elements with gears (e.g. planetary gear sets) connected to each other [4]. This configuration is well known [4]. The additional usage of clutches and coulomb friction and the requirement of real-time operability (constant step size and relatively large time increments) lead to a non-linear and complex simulation problem.

The modeling of clutches leads to the problem of switching between locking and slipping with friction in the first place. In the case of a slipping clutches a torque on the two connected shafts is transmitted, which solely depends linearly on the contact pressure (kinetic friction) and the sign of the speed difference at the clutch. If the differential speed of the two shafts is small enough and the accelerating moments do not tear the clutch apart, the clutch is set to the locking state. The clutch is set in this state as long as the impressed moments of the outer system do not tear the clutch apart and set it into the kinetic friction mode again.

A clutch in the locking mode is defined as a rigid link and transmits precisely the torque, which is transmitted on any imaginary cross-section of the shaft. This means there is an equality of moments on each side of the clutch. (According to the conservation of the angular momentum, the sum of the moments must be zero). Rigidly connected shafts result in a system with fewer degrees of freedom and therefore a system with lower order. An intuitive approach of modeling these nonlinear switching would change the system matrix and order with any coupling operation, which leads to complex implementation.

The solution presented by [3] avoids the problem of the changing order and system matrix by using a constant system matrix for the system, where the torques of the clutches are used as input variables and their non-linear calculation is performed outside the system, in a nonlinear feedback control loop.

2.2.2 DESCRIPTION OF THE MODEL PARTS

Preliminary it should be mentioned, that several input variables act on the system. External moments are considered outside of the transmission like the electric engine torque τ_m, the combustion engine torque τ_e and the torque of the wheels of the vehicle τ_w, which is transmitted through the vehicle environment. In addition, the pressure of the clutches $\tau_{C0}, \tau_{C1}, \tau_{C2}$, and τ_{C3} is referred to as internal moments. The pressure is calculated in advance from the current, acting on a hydraulic system on the clutches. The pressure is passed as an absolute value. Consequently the sign of the transmitted torque acting on the couplings must be calculated by means of the sign of the angular speed differences, which is derived from the system state variables. The examined model is a sixth order model with state variables for the electric engine angular speed ω_m, the combustion engine angular speed ω_e, the translated wheel angular speed ω_w , the translating planetary gear set ring angular speed ω_{R3}, the final drive angular speed ω_f and the output shaft torsion $\Delta\varphi$.

The shaft torsion $\Delta\varphi$ is calculated by means of the final drive gear ratio i_f, the final drive shaft angle φ_f and the wheel shaft angle φ_w, as shown in equation (1):

$$\Delta\varphi = i_f^{-1}\varphi_f - \varphi_w. \tag{1}$$

Automatic transmissions of hybrid electric vehicles usually consist of at least one planetary gear set, where a ring R, a sun S and planets P are connected.

Inertias for the combustion engine J_e, the electric engine J_3, the final shaft J_f, the translating planetary gear set J_2 and the wheel shaft J_v are parts of the mass matrix M, as seen in [3].

Figure 1 shows the drive train model used in this thesis as a basis. In the center three planetary gear sets are shown in a Ravigneaux configuration [1]. The planets of the gear set one and two are rigidly connected to each other, as well as the electric engine and the ring of the gear set one. Four clutches enable different gears. The clutch C_0 is used to uncouple the combustion engine. A spring damper system is used to model the torsion of the final shaft.

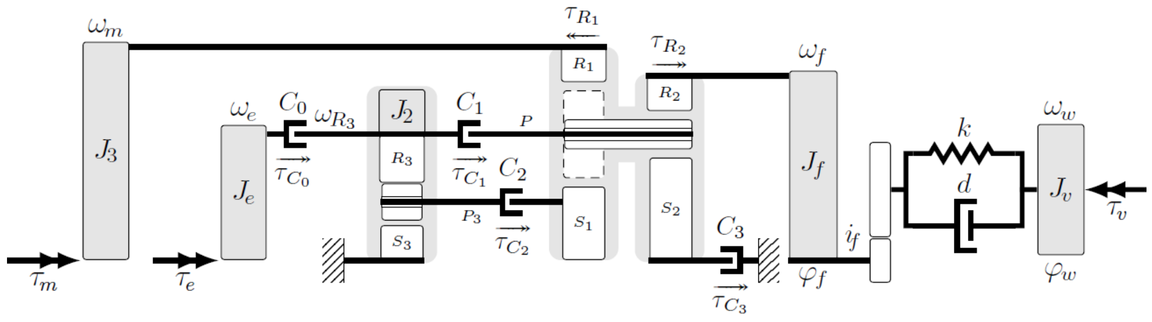

Figure 1: Future hybrid drivetrain model

The calculation of the clutch torques is now carried out in the following steps:

First, **the state of each clutch** is determined (slipping or locking). Therefore following input signals are necessary:

a. The state of the system (state vector)
b. All external and internal input torques.
c. The slip speeds of each clutch depending on the state vector.

The simulation model now operates in three sequential parts. First, each coupling with a slip speed close to zero is prone to get locked, but only if the torque, which would act on the clutch in the case of locking stays beyond a threshold value. Thus all clutches with a slip speed close to zero are set into a temporary locking mode and the reacting torques of the temporary locked system are calculated in the second sequential part. In the final part all torques which stay beyond the unlocking threshold value of the static friction and which are already locking candidates because of their slip speed are finally set into locking mode or otherwise unlocking mode for further calculations.

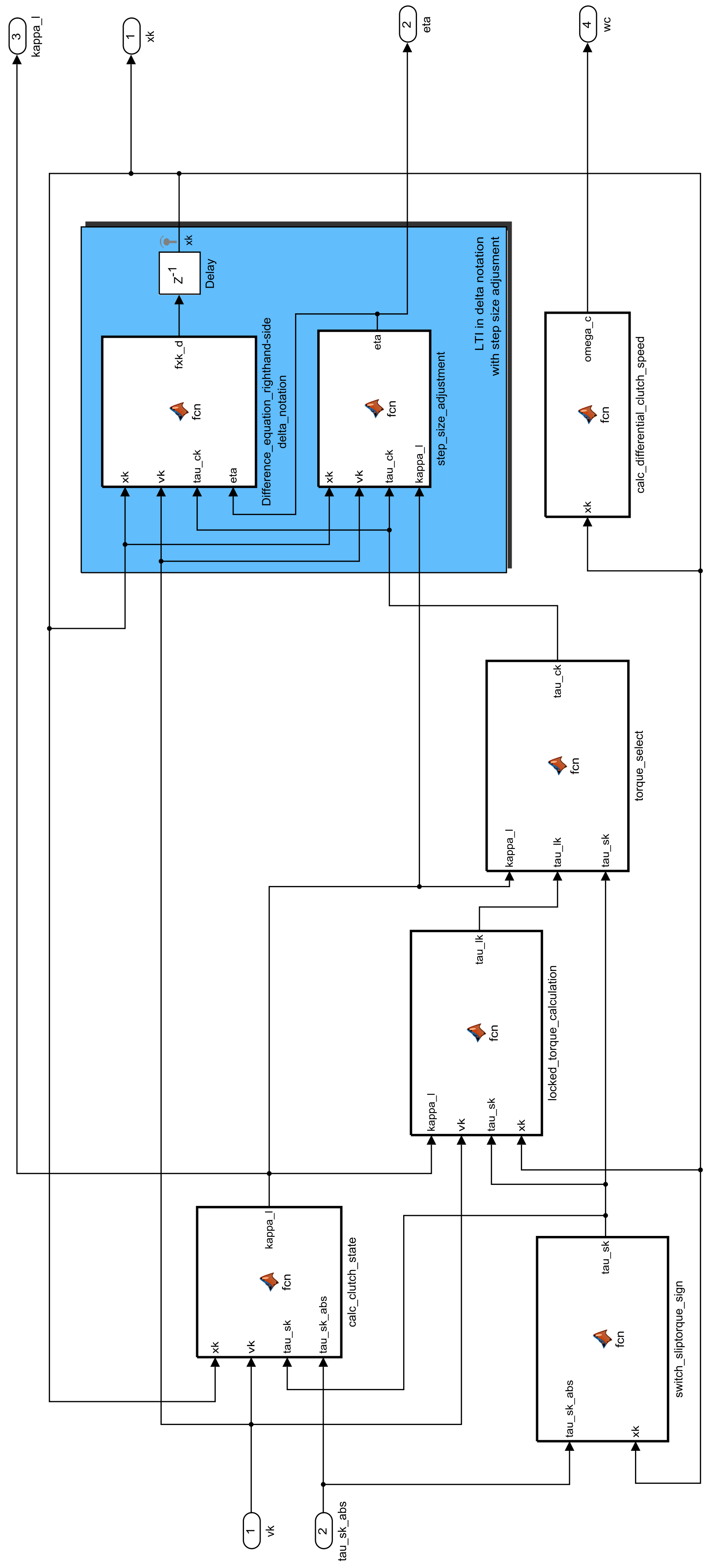

Figure 2: Top layer of the MATLAB® /Simulink® model

At a second step the **calculation of the acting torques on the locked clutches** is performed for the second time, but now not for the sake of determining locking mode. Here the torques are calculated for the input of the linear system, which does not embed clutches. Respectively calculated torques will lead to a slip-speed close to zero in the next step, which is exactly what the locked clutch causes.

Finally a **switching function** chooses by means of the clutch states, if either the **slip torque or the locking torque acts** on the linear system input.

2.2.3 BASIC MATHEMATICAL DESCRIPTION

In this chapter an overview of the fundamental mathematical equations should be given, which are derived from [3].

First of all, the linear part of the system can be written as

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}_\mathrm{v}\mathbf{v} + \mathbf{B}_\mathrm{c}\boldsymbol{\tau}_\mathrm{c} \text{ and } \mathbf{y} = \mathbf{C}\mathbf{x}. \tag{2}$$

The external nonlinear feedback and switching of $\boldsymbol{\tau}_\mathbf{c}$ realizes the sliding and locking of the clutches. The system is defined by the system matrix **A,** the input matrix for external torques (engines, vehicle resistance) $\mathbf{B}_\mathrm{v}$, the input matrix for the clutch moments $\mathbf{B}_\mathbf{c}$ and the output matrix **C**, which is not important here.

The discretization of the system with the step size $\mathrm{T_d}$ leads to the discrete time system, with the discrete time system matrix $\boldsymbol{\Phi}$ and the input matrices $\mathbf{H}_\mathbf{v}$ and $\mathbf{H}_\mathbf{c}$:

$$\mathbf{x}_{\mathrm{k}+1} = \boldsymbol{\Phi}\mathbf{x}_\mathrm{k} + \mathbf{H}_\mathrm{v}\mathbf{v}_\mathrm{k} + \mathbf{H}_\mathrm{c}\boldsymbol{\tau}_{\mathrm{c,k}} \text{ and } \mathbf{y}_\mathrm{k} = \mathbf{C}\mathbf{x}_\mathrm{k}. \tag{3}$$

The switching between sliding or locking of the clutches, and therefore the used torque is realized by a binary state vector for the clutches $\boldsymbol{\kappa}$, which saves the current state. The vector holds the state of each clutch, where a logic zero means locked and a logic one refers to the state unlocked. Whether the torque of the clutches derived from the hydraulic pressure in the slipping mode $\boldsymbol{\tau}_{\mathbf{c,s}}$ or the torque of the locked system $\boldsymbol{\tau}_{\mathbf{c,l}}$ is set as an input for the linear system, is defined as follows:

$$\boldsymbol{\tau}_{\mathrm{C,k}} = \mathbf{K}_{\mathrm{s,k}}\boldsymbol{\tau}_{\mathrm{s,k}} + \mathbf{K}_{\mathrm{l,k}}\boldsymbol{\tau}_{\mathrm{l,k}} \quad \text{with} \quad \mathbf{K}_{\mathrm{s,k}} = \mathrm{diag}(\neg\boldsymbol{\kappa}),\ \mathbf{K}_{\mathrm{l,k}} = \mathrm{diag}(\boldsymbol{\kappa}) \tag{4}$$

The speed difference of each clutch can be computed with the kinematic relation between the model states $\mathbf{x}_\mathrm{k}$, the mass matrix $\mathbf{M}$ and the speeds of clutch primary and secondary shafts by

$$\boldsymbol{\omega}_{\mathbf{C,k}} = -\mathbf{B}_\mathrm{C}^\mathrm{T}\mathbf{M}\mathbf{x}_\mathrm{k}. \tag{5}$$

We now want to compute the force, that is acting on the clutch $\boldsymbol{\tau}_{\mathrm{l,k}}$, and check, if it is still beyond a torque, where the clutches slip apart and if this is true, this torque is set as an input adequately. For clutch locking a necessary condition is $\boldsymbol{\omega}_{\mathbf{C,k}} \overset{!}{=} 0$ and the constraint for these clutches to remain locked is $\mathbf{K}_{\mathrm{l,k}}\boldsymbol{\omega}_{\mathrm{C,k+1}} - \mathbf{K}_{\mathrm{l,k}}\boldsymbol{\omega}_{\mathrm{C,k}} \overset{!}{=} 0$, which leads to

$$-\mathbf{K}_{\mathrm{l,k}}\mathbf{B}_\mathrm{C}^\mathrm{T}\mathbf{M}\mathbf{x}_{\mathrm{k+1}} \overset{!}{=} \mathbf{0}. \tag{6}$$

Considering equation (6) and (3), $\boldsymbol{\tau}_{\mathrm{l,k}}$ can be derived as follows:

$$\boldsymbol{\tau}_{\mathrm{l,k}} = -\Xi_\mathrm{k}\left[\boldsymbol{\Phi}\mathbf{x}_\mathrm{k} + \mathbf{H}_\mathrm{k}\mathbf{v}_\mathrm{k} + \mathbf{H}_\mathrm{c}\mathbf{K}_{\mathrm{s,k}}\boldsymbol{\tau}_{\mathrm{s,k}}\right] \quad \text{and} \tag{7}$$

$$\Xi_\mathrm{k} = \left(\mathbf{K}_{\mathrm{l,k}}\mathbf{B}_\mathrm{C}^\mathrm{T}\mathbf{M}\mathbf{H}_\mathrm{c}\mathbf{K}_{\mathrm{l,k}} + \mathbf{K}_{\mathrm{s,k}}\right)^{-1}\mathbf{K}_{\mathrm{l,k}}\mathbf{B}_\mathrm{C}^\mathrm{T}\mathbf{M}. \tag{8}$$

If equation (7) is inserted into (3), the switching discrete time system can be described by

$$\mathbf{x}_{\mathrm{k+1}} = \left(\mathbf{I} - \mathbf{H}_\mathrm{c}\mathbf{K}_{\mathrm{l,k}}\Xi_\mathrm{k}\right)\boldsymbol{\Phi}\mathbf{x}_\mathrm{k} + \left(\mathbf{I} - \mathbf{H}_\mathrm{c}\mathbf{K}_{\mathrm{l,k}}\Xi_\mathrm{k}\right)\mathbf{H}_\mathrm{v}\mathbf{v}_\mathrm{k} + \left(\mathbf{I} - \mathbf{H}_\mathrm{c}\mathbf{K}_{\mathrm{l,k}}\Xi_\mathrm{k}\right)\mathbf{H}_\mathrm{c}\mathbf{K}_{\mathrm{s,k}}\boldsymbol{\tau}_{\mathrm{s,k}}. \tag{9}$$

2.2.4 STEP SIZE ADJUSTMENT

When simulating with fixed and large step sizes, generally a zero crossing of a slip speed may be located between two steps. Since the tolerance of slip speed zero is subject to be low in value, zero crossings are detected rarely and couplings will barley go into locked state, which leads to chattering within the signals [3]. Chattering occurs, because every time a zero crossing is not detected, the sign of the slip torque changes and acts in the opposite direction. One solution varying the step size in the case of a zero crossing event is a major part of [3].

Therefore the discrete time system description in equation (10) (transition matrix $\boldsymbol{\Phi}(\mathrm{T_d})$, input vector $\mathbf{h}(\mathrm{T_d})$, sampled input value $\mathrm{u_k}$ and step size $\mathrm{T_d}$) is implemented in delta notation [5] (equation (11) and (12)). Through prediction it is possible to detect zero crossings for each time step. If a zero crossing is detected between two steps, the step size is changed by means of a factor, which puts the zero-crossing event approximately to the next discrete time step. By this method the clutch is prone to change its locking mode without chattering. In the subsequent time step, the step size has to be corrected in order to meet the general step size pattern by one full time step and the complementary “missing” time step of the previous corrected step.

$$\mathbf{x}_\mathrm{k} = \mathbf{\Phi}(\mathrm{T_d})\mathbf{x}_\mathrm{k} + \mathbf{h}(\mathrm{T_d})\mathrm{u_k} \tag{10}$$

$$\delta\mathbf{x}_\mathrm{k}(\mathrm{T_d}) = \frac{1}{\mathrm{T_d}}((\mathbf{\Phi}(\mathrm{T_d}) - \mathbf{I})\mathbf{x}_\mathrm{k} + \mathbf{h}(\mathrm{T_d})\mathrm{u_k}) \tag{11}$$

$$\mathbf{x}_\mathrm{k+1} = \mathbf{x}_\mathrm{k} + \mathrm{T_d}\delta\mathbf{x}_\mathrm{k}(\mathrm{T_d}) = \mathbf{x}_\mathrm{k} + ((\mathbf{\Phi}(\mathrm{T_d}) - \mathbf{I})\mathbf{x}_\mathrm{k} + \mathbf{h}(\mathrm{T_d})\mathrm{u_k}) \tag{12}$$

Figure 3 shows a graphical interpretation of the step size adjustment. The blue line shows the trajectory of the continuous time slip speed, which crosses zero at a certain point. The green dots represent the discrete time values for the slip speed with a step size of 2 seconds. To locate the approximate time variation for a slip speed of zero, the delta notation can be used by inserting the factor η, shown in equation (14), which is an approximation of the general equation (13), which exactly calculates the state at a time step $(\mathrm{k} + \eta)\mathrm{T_d}$. Equation (13) is inappropriate to calculate η, because the transition matrix $\mathbf{\Phi}(\eta\mathrm{T})$ is a nonlinear function of η.

$$\mathbf{x}_\mathrm{k+\eta} = \mathbf{x}_\mathrm{k} + \eta\mathrm{T_d}\delta\mathbf{x}_\mathrm{k}(\eta\mathrm{T_d}) = \mathbf{x}_\mathrm{k} + \eta((\mathbf{\Phi}(\eta\mathrm{T_d}) - \mathbf{I})\mathbf{x}_\mathrm{k} + \mathbf{h}(\mathrm{T_d}\eta)\mathrm{u_k}) \tag{13}$$

$$\mathbf{x}_\mathrm{k+\eta} \approx \mathbf{x}_\mathrm{k} + \eta\mathrm{T_d}\delta\mathbf{x}_\mathrm{k}(\mathrm{T_d}) = \mathbf{x}_\mathrm{k} + \eta((\mathbf{\Phi}(\mathrm{T_d}) - \mathbf{I})\mathbf{x}_\mathrm{k} + \mathbf{h}(\mathrm{T_d})\mathrm{u_k}) \tag{14}$$

The red line in Figure 3 shows the state transition according to equation (14), if η is used as continuous variable between the two states k and (k+1). Since the transition is approximated affine by means of this method, it is now easy to approximate the slip speed zero time, and consequently η. The consecutive step has to be extended by (2-η) to keep up with the real time constraint.

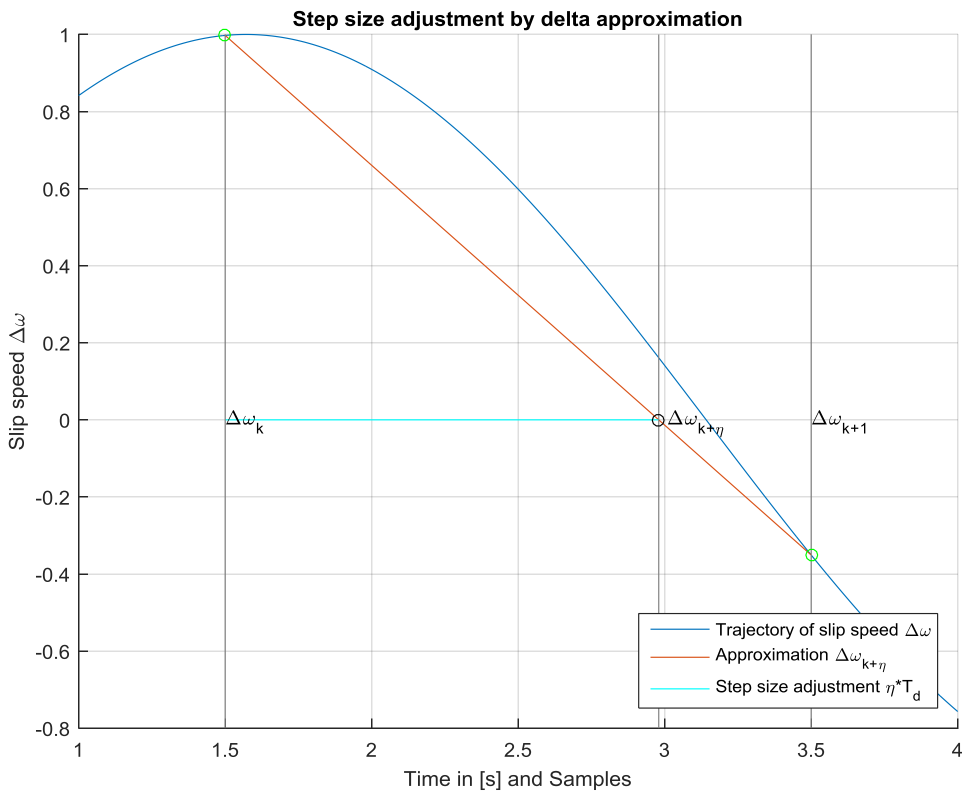

Figure 3: Step size adjustment by delta approximation

To compute the variation η of the step size for each clutch (equation (16)) crossing the zero line, following steps are necessary (Note that η_j might have unusual values and has to be sorted out by an algorithm, i.e.: negative, infinite):

Enforcing a zero slip (compare to equation (5)) at η for each clutch j leads to

$$\left(\omega_{C,k+\eta}\right)_j \approx -(\mathbf{B}_C^T\mathbf{M}\mathbf{x}_{k+\eta})_j \overset{!}{=} 0. \tag{15}$$

Substituting (14) in (15) leads to

$$\eta_j \approx -\frac{(\mathbf{B}_C^T\mathbf{M}\mathbf{x}_k)_j}{\left(\mathbf{B}_C^T\mathbf{M}\Delta\delta\mathbf{x}_k(\Delta)\right)_j}. \tag{16}$$

The approximation error and further discussion on this approximation can be found in [3] and in Appendix A.

If several zero crossings are detected within the next period (which is not an issue in practical use cases), the closest zero crossing to the previous step is chosen.

Additionally it has to be mentioned, that step size adjustment is only necessary, if a clutch goes from slipping into locking mode, because otherwise the slip speed is equal to zero.

2.3 RESULTS AND DISCUSSION

The goals of an essential simplification and a minimal variation of the simulation results to the reference model, which is part of a huge vehicle simulation, have been achieved to our uttermost satisfaction. Little differences come from the implementation as "functions blocks" of almost all equations and functions in the MATLAB®/Simulink® model, which leads to minimal numerical aberrations. Since the differences of both simulation outputs are quite small for recognizing differences in the plot, the reference signal and the magnified differences of the reference signals and the implemented simplified signals are shown for a step size of $T_d = 10\text{ms}$.

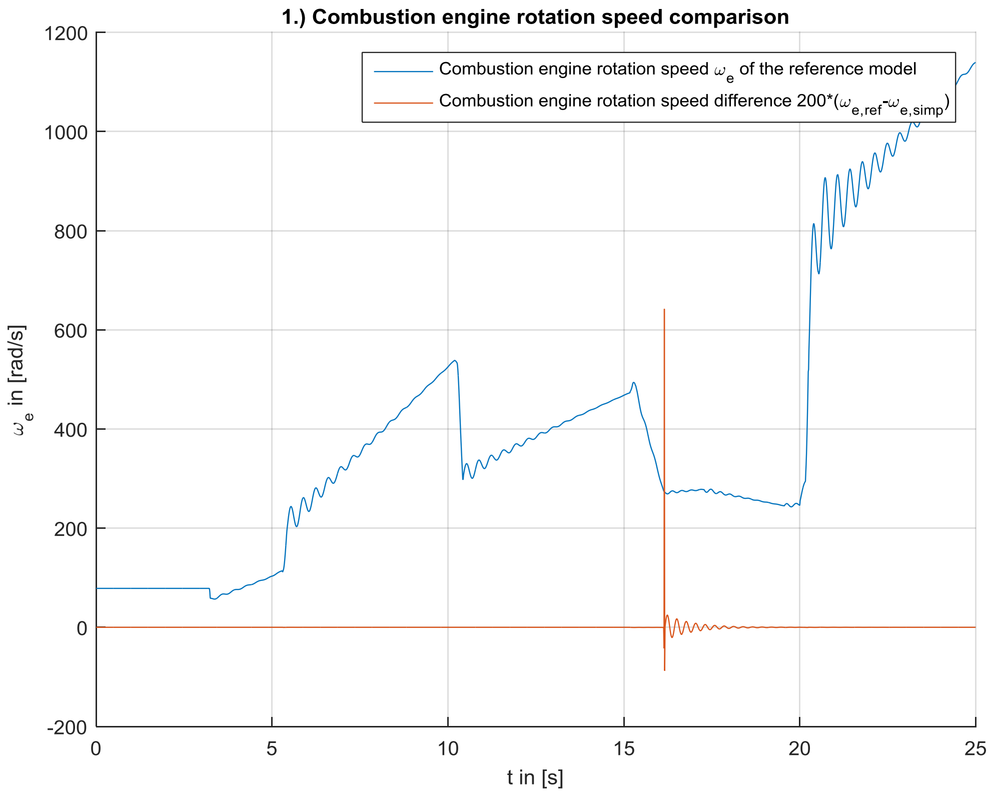

Figure 4: Final comparison of the combustion engine rotation speed

Figure 4 shows the results of the combustion engine rotational speed ω_e for a certain test case, provided by the reference model and the simulation error. The input variables for the model comprise various gear switching actions, performed by means of input functions for clutch pressure, engine torques and vehicle resistance, expressed by a wheel torque. Obviously an outstanding error occurs at a simulation time of around 17 seconds. Here the step size is adjusted slightly differently in both models, which leads to a relative high error, but which decays rapidly in the further simulation.

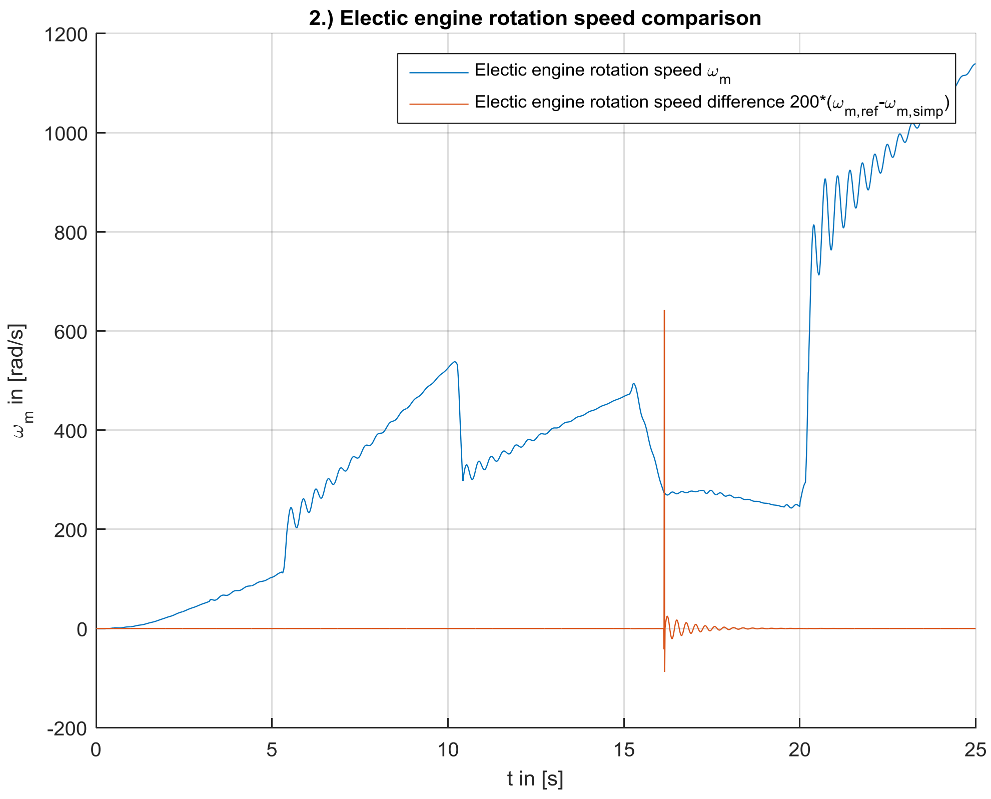

Figure 5: Final comparison of the electric engine rotation speed

A closer look on simulation results reveals little differences at a growing simulation time. Numerical differences sum up and lead to a small aberration, which is totally acceptable for further investigations.

The resulting output functions for the reference model and the simplified reduced model also coincide mostly for the electric engine rotation speed ω_m, shown in Figure 5 and for the wheel rotational speed ω_w, shown in Figure 6, but also the adjustment error dominates here.

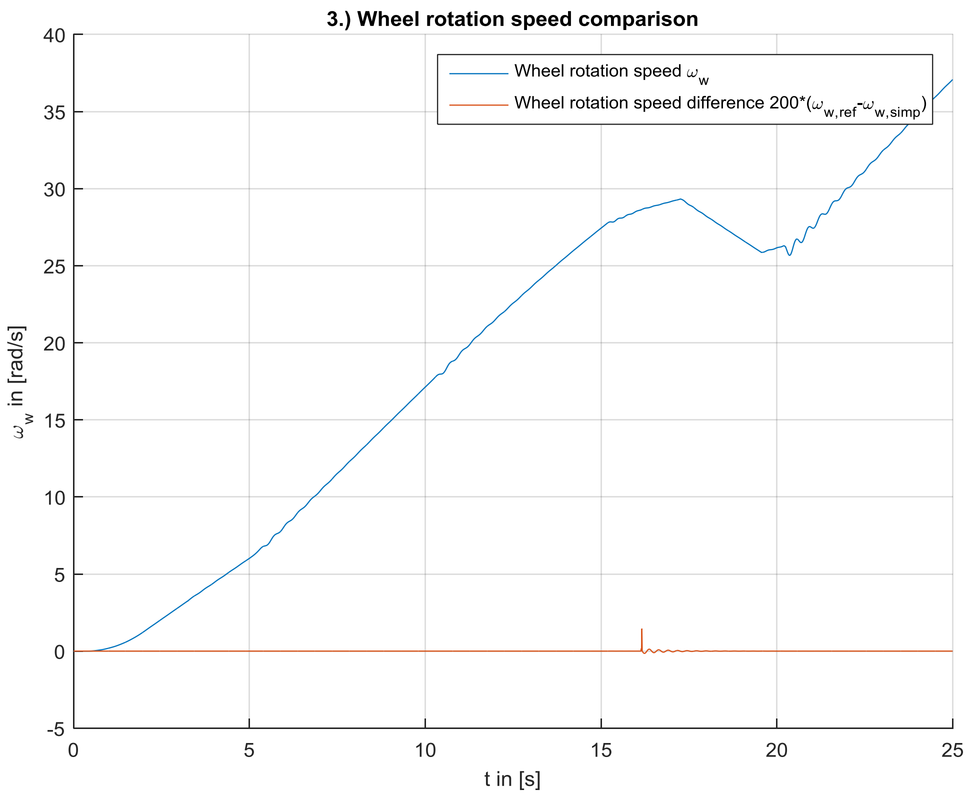

Figure 6: Final comparison of the wheel rotation speed

Figure 6 shows, that in the investigated simulation case a short deceleration occurs.

The step size adjustment variable η is very vulnerable to variances in the output results, due to its dependency on the exact time of a locking clutch in between two discrete time instances and consequently the plot for eta in Figure 7 is a proper tool for the issue of a high resolution comparison.

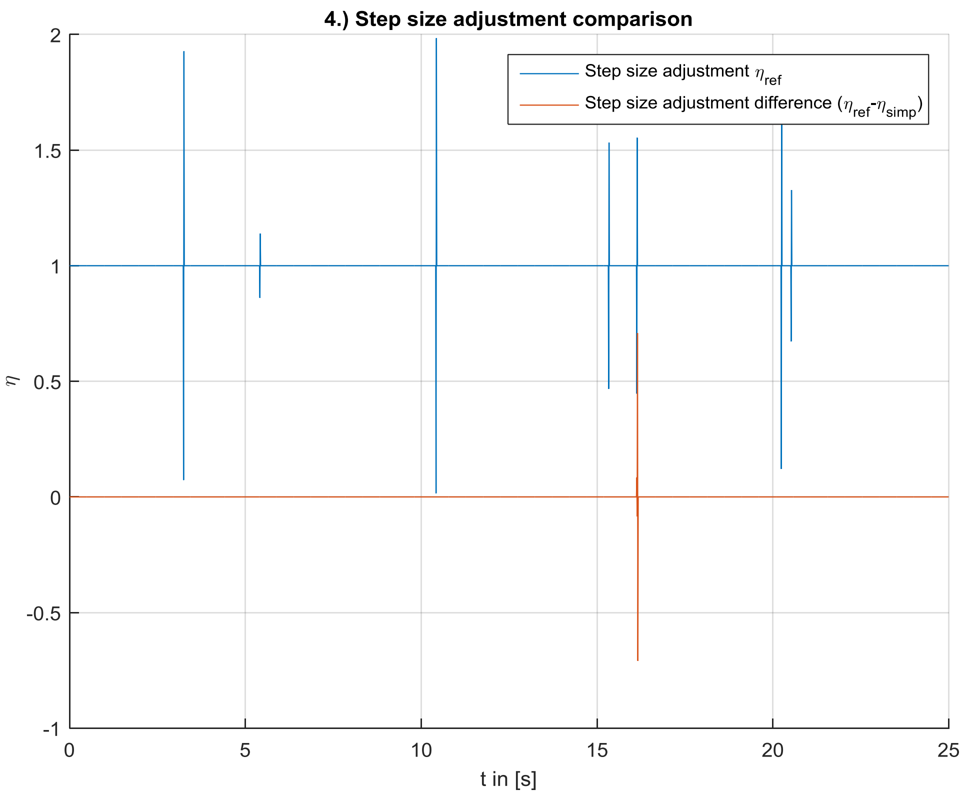

Figure 7: Final comparison of the step size adjustment factor η

Figure 8 shows each model clutch states for the period of the simulation. The outstanding simulation results of the step size adjustment variable η already assume a good convergence of the clutch states. Differing clutch states in the beginning emerge from undefined clutch states in the case of a slip speed of zero, an angular speed of zero and no torques.

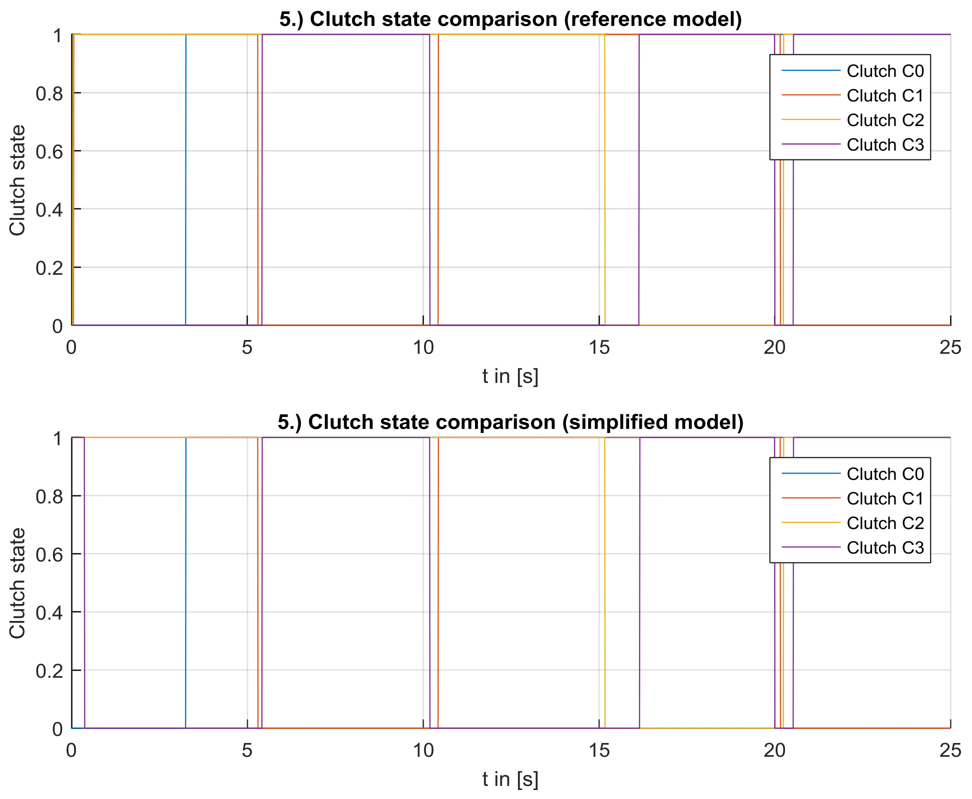

Figure 8: Final comparison of the clutch states in either model

Several simulation cases have been examined and resulted in similar findings. The designed model is accurate, easy to understand and also easy to adapt for other purposes.

3 NONLINEAR SPRING-DAMPER-MASS SYSTEMS

Consequences of nonlinear spring-damper-mass systems are of major concern [1]. Nonlinearities may include an abrupt change of the spring stiffness factor or a static friction component of the damper [6]. Nonlinear simulation methods must be capable of being integrated into existing simulation methods used by VIRTUAL VEHICLE – Research Center.

In this chapter, first of all several nonlinear configurations of spring-damper elements are examined for their behavior in a basic spring-damper-mass system in the continuous time domain. The mechanical problem is easily transformable for rotational problems, which mathematical description fits with translational problems.

Secondly the most relevant model configuration for VIRTUAL VEHICLE – Research Center is further investigated for methods of time discretization methods, with a special concern on partly linear systems.

Finally the most appreciated discretization methods are furthermore investigated for their applicability in existing drivetrain models, with a focus on the hybrid engine model of chapter 2.

3.1 NONLINEAR SPRING-DAMPER CONFIGURATIONS

All spring-damper configurations emerge from the basic Kelvin-Voigt model as shown in [4] and [7]. The Kelvin Voight model basically consists of a parallel spring and damper. In other models [7] these elements are used serially or combined to build larger models for special configurations. Figure 9 shows the Kelvin Voight model in a fundamental oscillator with a single degree of freedom.

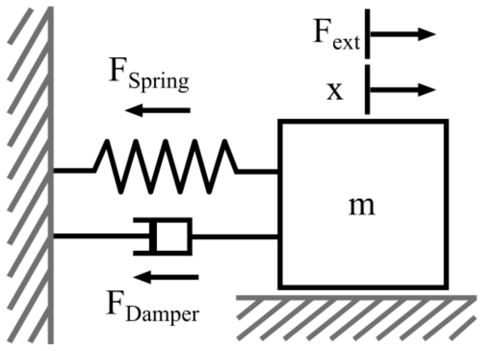

Figure 9: Spring-Damper-Mass System

Since the Kelvin-Voight model is defined for linear systems further investigations lead to a different behavior for some parts.

Nonlinearities may be located in both, spring or damper force function. The spring force is usually solely dependent on the deflection variable x. The damper force is commonly dependent on the speed variable $\dot{x}$, but might also have a cross dependency on the deflection x (mechanical design of a damper).

An outstanding circumstance in terms of damper nonlinearities is the known problem of switching between viscous damping and static friction, which was part of the previous chapter, regarding clutches. Consequently the aim is to adapt existing solutions [3] for the upcoming simulation purpose.

3.1.1 LINEAR CONFIGURATION

So far, the spring force F_{Spring} and damper force F_{Damper} (also referred to as F_{Spr}, F_{Damp}) are assumed as linear functions of either the deflection x of the mass m or the velocity $\dot{x}$.

$$F_{Spr,lin}\big(x(t)\big) = cx(t) \tag{17}$$

$$F_{Damp,lin}\big(\dot{x}(t)\big) = d\dot{x}(t) \tag{18}$$

In the linear system theory this configuration is well known as damped harmonic oscillator [5] and can be written as a linear second order system with the introduction of a state vector **x** and an external input u.

$$\mathbf{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \begin{bmatrix} x(t) \\ \dot{x}(t) \end{bmatrix} \tag{19}$$

$$u(t) = F_{ext}(t) \tag{20}$$

The dynamic behavior of the system can be written as a set of ordinary first order linear differential equations in matrix notation, with the system matrix **A** and the input matrix **b** as shown in (21).

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} 0 & 1 \\ -\frac{c}{m} & -\frac{d}{m} \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{b}u(t) \tag{21}$$

Several equations and parameters describe the fundamental dynamic aspects of the system [4]. Equations (22) to (24) show some values, which are of practical relevance. The damping ratio D is a measure for the damping and oscillation ability of a system [8]. A factor of zero is only possible if the damping factor d is equal to zero and this leads to a freely oscillating system.

A factor between zero and one leads to an oscillating system with an eigenfrequency ω_{eig} and a ratio D between zero and $\frac{1}{\sqrt{2}}$ will cause a resonance frequency ω_{res}.

A ratio D greater than one results in high damped system with no oscillation ability, if not excited externally.

$$D = \frac{d}{2\sqrt{cm}} \tag{22}$$

$$\omega_{eig} = \sqrt{\frac{c}{m} - \frac{d^2}{4m^2}} \quad \text{for } D < 1 \tag{23}$$

$$\omega_{res} = \sqrt{\frac{c}{m} - \frac{d^2}{2m^2}} \quad \text{for } 0 < D < \frac{1}{\sqrt{2}} \tag{24}$$

Once the system is expressed by means of those easy manageable parameters, further engineering work like control engineering tasks are well solvable [5].

Also a time discretization for discrete simulation tasks is straight forward and leads to linear difference equations.

A useful representation of system behavior, especially in the fields of nonlinear systems and systems with two state variables are phase plots. State plots are missing the time axis and therefore contain the other state variable, which eliminates the time dependency.

Figure 10 shows the phase plot of an autonomous linear system with parameters selected as in Table 1.

Table 1: Parameters for linear damped oscillator

$\boldsymbol{x}(t=0)$	c	d	m
$\begin{bmatrix} m \\ \frac{m}{s} \end{bmatrix}$	$\left[\frac{N}{m}\right]$	$\left[\frac{N}{ms}\right]$	$[kg]$
$\begin{bmatrix} 0.3 \\ 0 \end{bmatrix}$	10	0.5	1

It is remarkable that various system properties basically do not depend on the initial state. At first, the system is stable anyway (damping factor is always higher than zero). Moreover the ability of oscillating is determined by the system parameters and independent of the initial state. A doubled initial value would lead to the exact same appearance of the phase plot, excepted of the axis scaling. And for a constant time step $\mathrm{T_d}$ a simple vector transformation with a constant transition matrix $\mathbf{\Phi}(\mathrm{T_d})$ describes every state transition of the autonomous system (equation (25)) with the state vector $\bar{\mathbf{x}}$ and a vector $\mathbf{H}(\mathrm{T_d})$can be found [5], which integrates an input signal u that is sampled and hold to a constant value for the time of $\mathrm{T_d}$ and leads to equation (26) and the state vector $\mathbf{x}$.It is important to notice that these equations are no approximations. For a piecewise constant input signal u for the time of $\mathrm{T_d}$ the equations are exact solutions.

$$\bar{\mathbf{x}}(\mathrm{t_0} + \mathrm{T_d}) = \mathbf{\Phi}(\mathrm{T_d})\bar{\mathbf{x}}(\mathrm{t_0}) \tag{25}$$

$$\mathbf{x}(\mathrm{t_0} + \mathrm{T_d}) = \mathbf{\Phi}(\mathrm{T_d})\mathbf{x}(\mathrm{t_0}) + \mathbf{H}(\mathrm{T_d})\mathrm{u}(\mathrm{t_0}) \tag{26}$$

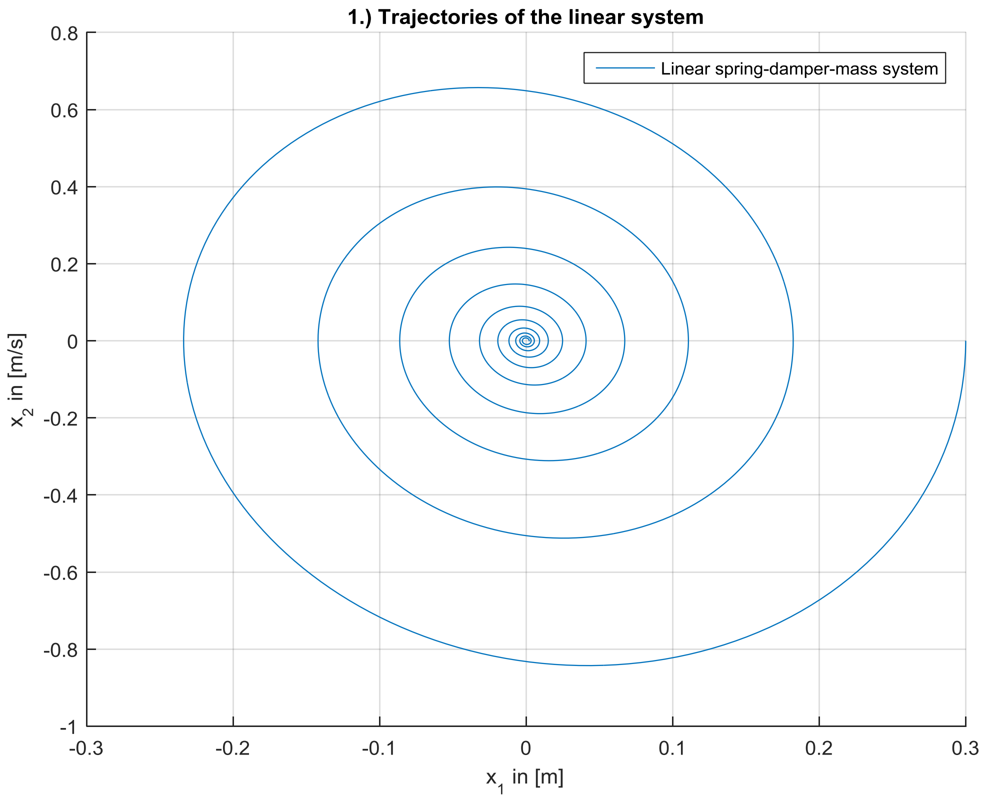

Figure 10: Trajectories of a linear damped oscillator

3.1.2 Linear damping force and nonlinear spring force

There are two intuitive approaches for implementing nonlinear spring force dependencies. The first approach will generally be more applicable for general nonlinear spring force functions. It uses point symmetric polynomials for the approximation of a given empiric function, as seen in (27).

$$F_{\mathrm{Spr,nlin1}}\big(x(t)\big) = \sum_{i=0}^{N} c_{2i+1} x(t)^{2i+1} \tag{27}$$

A second approach is appropriate for a geometrical spring composition, where the deflection of the spring may enter N different regions I_i with an erratic change of the spring stiffness at transition points x_i and shown in Figure 11 and equation (28). The index i displays the region index. However, inside a region the spring force is assumed affine. Since the spring force is assumed to be symmetrical with respect to the origin, stiffness regions are defined for absolute deflection values.

$$F_{\mathrm{Spr,nlin2}}\big(x(t)\big) = c_i x + \mathrm{sign}(x)\ F_{\mathrm{off,i}} \quad \text{for } x \in I_i, i = \{0,1,\dots,N-1\} \text{ and}$$

$$I_i = \begin{cases} \{x \in \mathbb{R} \mid x_{S,i} < |x| \le x_{S,i+1}\} & \text{if } i < N-1 \\ \{x \in \mathbb{R} \mid x_{S,i} < |x| \le \infty\} & \text{if } i = N-1 \end{cases} \tag{28}$$

$$F_{\mathrm{Damp,lin}}(\dot{x}(t)) = d\dot{x}(t) \tag{29}$$

Figure 11 shows an example for a piecewise affine spring force function with following parameter settings according to Table 2. The Parameter N represents the number of regions used in this example.

Table 2: Parameters for piecewise affine spring force

N	$x_{S,i}$	c_i	$F_{off,i}$
	$[m]$	$\left[\frac{N}{m}\right]$	$[N]$
4	$[0\ \ 0.05\ 0.1\ 0.15]$	$[0.05\ \ 1\ \ 2\ \ 10]$	$[0\ -0.047\ \ -0.147\ \ -1.347]$

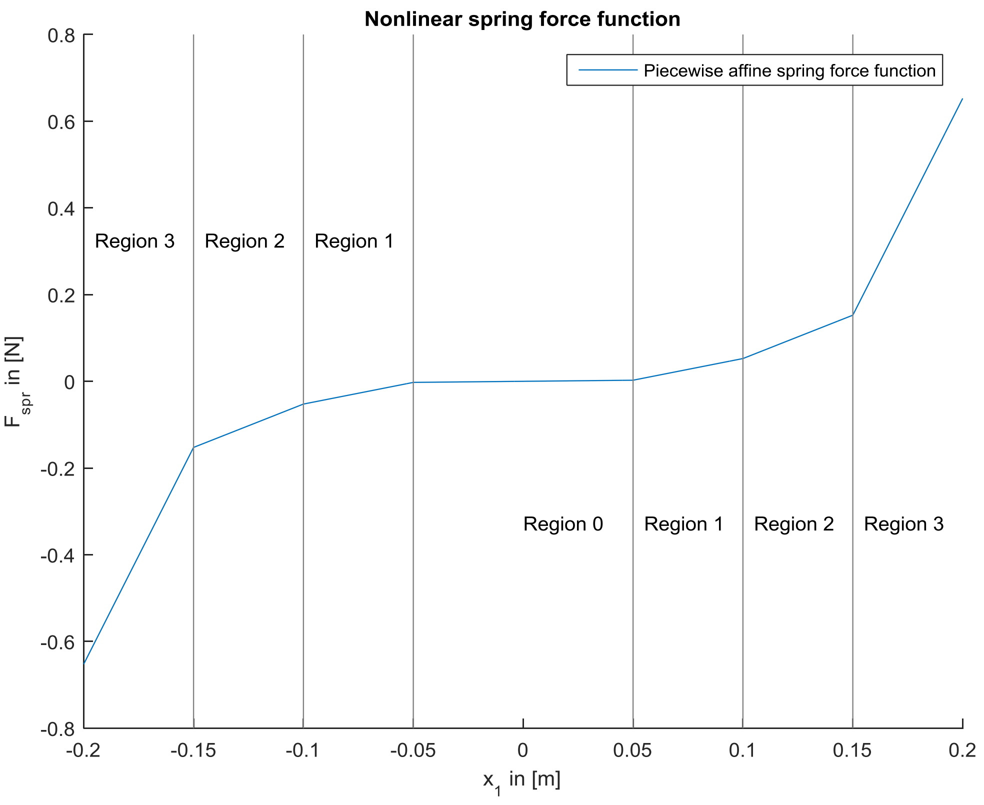

Figure 11: Piecewise affine spring force function

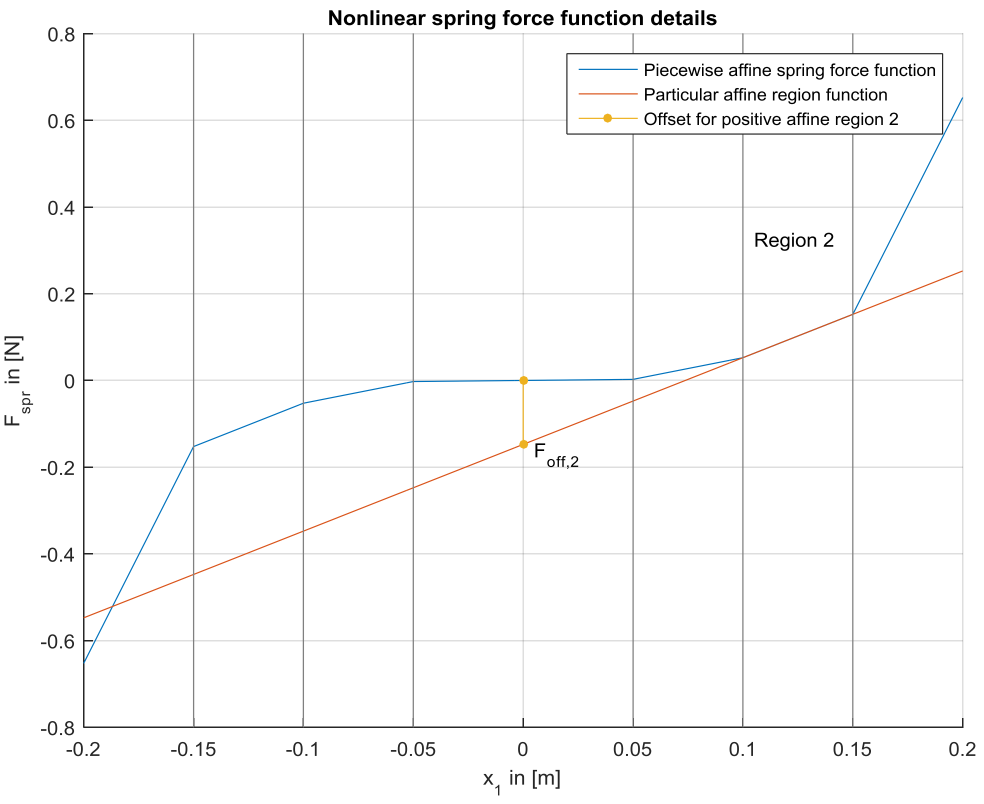

Figure 12: Details of positive affine spring force region 2

The nonlinear system can be written in vector notation as follows in (30).

$$\begin{aligned}\dot{\mathbf{x}}(t) &= \begin{bmatrix} 0 & 1 \\ 0 & -\frac{d}{m} \end{bmatrix} \mathbf{x}(t) - \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} F_{Spr,nlin2}\big(x_1(t)\big) + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u(t) \\ &= \mathbf{A}_{lin}\mathbf{x}(t) - \mathbf{a}_{nl}F_{Spr,nlin2}\big(x_1(t)\big) + \mathbf{b}u(t)\end{aligned} \tag{30}$$

Given a piecewise affine spring force function the equation expands to (31), assumed that x ∈ I_i.

$$\begin{aligned}\dot{\mathbf{x}}(t) &= \begin{bmatrix} 0 & 1 \\ 0 & -\frac{d}{m} \end{bmatrix} \mathbf{x}(t) - \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} \big(c_i x_1 + \mathrm{sign}(x_1)\, F_{off,i}\big) + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u(t) \\ &= \begin{bmatrix} 0 & 1 \\ -\frac{c_i}{m} & -\frac{d}{m} \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u(t) - \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} \mathrm{sign}(x_1)\, F_{off,i}\end{aligned} \tag{31}$$

By an easy adding of negative regions for negative deflections x, the term $\mathrm{sign}(x_1)$ is integrated into the region offset $F_{off,i}$. Now it is easy to perceive the system equation as piecewise affine with an added constant value for each region, which leads to equation (32) and will be of further interest when it comes to discretization methods.

Assuming x ∈ I_k, the system description (32) follows.

$$\begin{aligned}\dot{\mathbf{x}}(t) &= \begin{bmatrix} 0 & 1 \\ -\frac{c_k}{m} & -\frac{d}{m} \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u(t) - \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} \mathrm{sign}(x_1)\, F_{off,k} \\ &= \mathbf{A}_{lin,k}\mathbf{x}(t) + \mathbf{b}u(t) + \mathrm{sign}(x_1)\boldsymbol{\xi}_{const,k}\end{aligned} \tag{32}$$

$$\boldsymbol{\xi}_{const,k} = -\begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} F_{off,k} \tag{33}$$

As mentioned before, phase plots deliver a significant overview of the system behavior.

In figure (27) the linear system with values according to Table 1 is compared with the piecewise affine system according to Table 2. Inside one region the nonlinear system has a affine behavior and is fully described through equation (25) and (26), provided that the coordinate system is shifted by the region offset $F_{off,i}$.

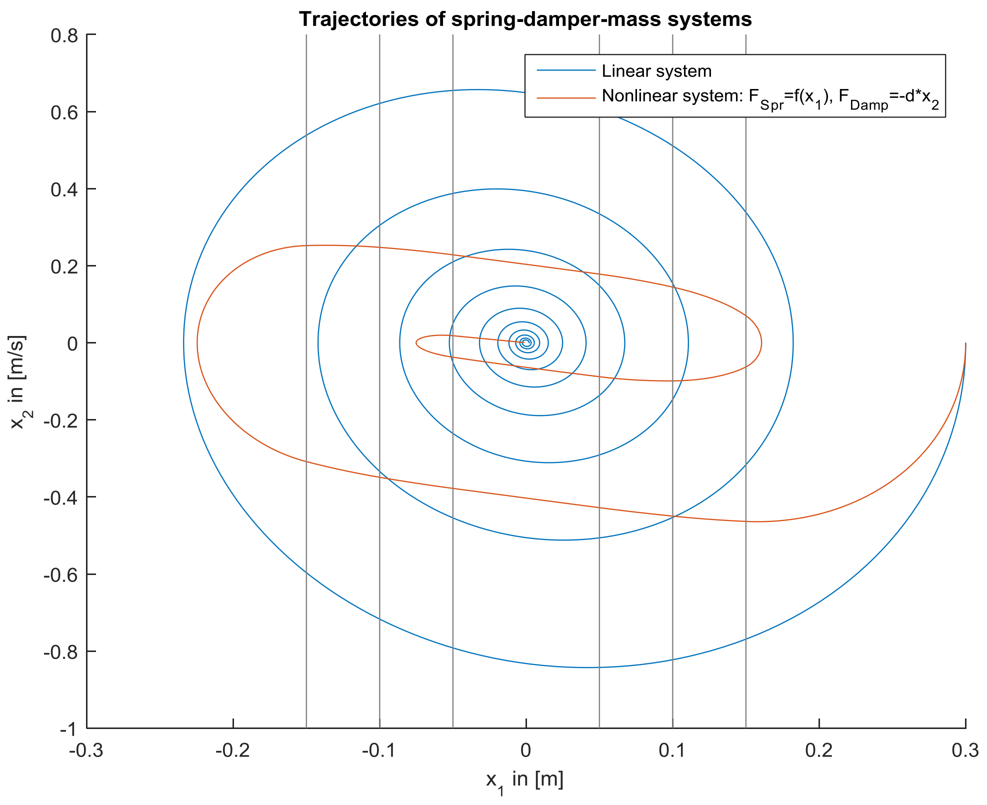

Figure 13: Comparison of linear and nonlinear trajectories

An essential part with this category of systems is the oscillation ability. As seen in equation (22), the damping ratio is a function of the spring stiffness c and the damping factor d. This offers the possibility of an oscillating behavior for high deflections and a highly damped system for low deflections, and can be seen in time domain of this example in Figure 14.

As an obvious contrast, the linear system will theoretically oscillate for all times.

Furthermore the oscillating frequency is changing with respect to the deflection amplitude, which is a reason for the lack of the possibility of using familiar methods for linear system analysis.

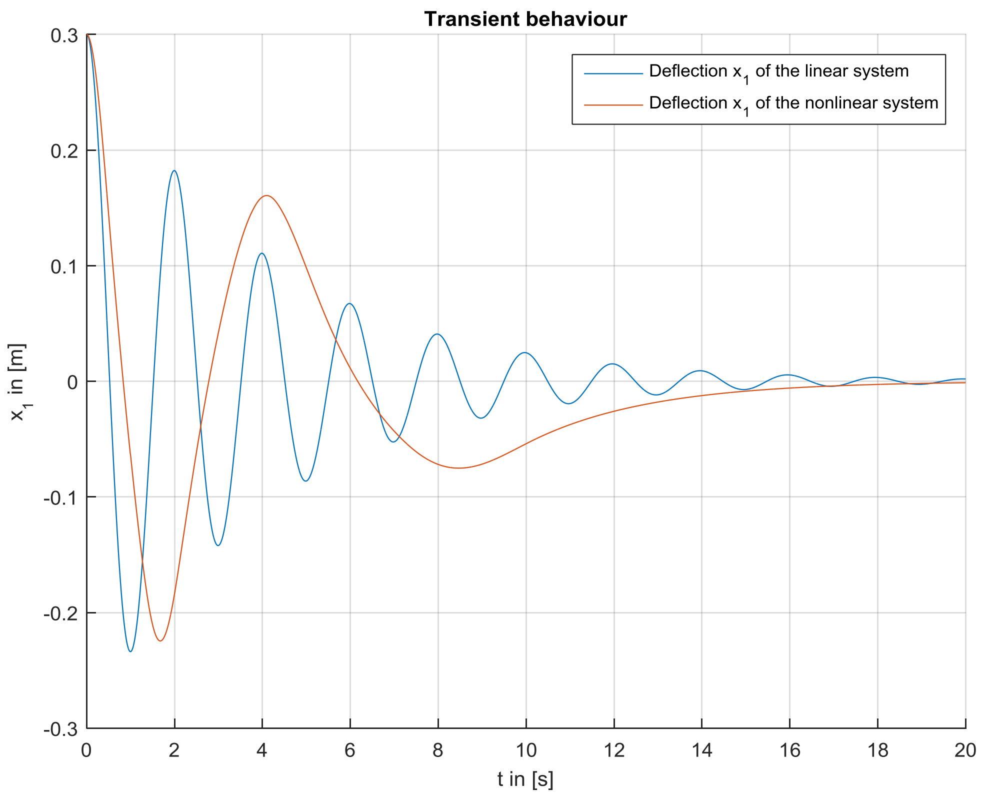

Figure 14: Transient function of the deflection of the linear/nonlinear system

3.1.3 Nonlinear spring force and nonlinear damper force dependent on a single variable

Another possibility to bring velocity dependent nonlinearities into the simulation is using an analog piecewise affine force curve with N_D regions for the damper force in particular, but dependent on just the system velocity, according to equation (34) and plotted in Figure 15.

$$F_{Damp,nlin}\big(\dot{x}(t)\big) = d_i \dot{x} + \operatorname{sign}(\dot{x})\, F_{damp\_off,i} \quad \text{for } \dot{x} \in J_i, \\ i = \{0,1,\dots N_D - 1\} \text{ and} \\ J_i = \begin{cases} \{\dot{x} \in \mathbb{R} |\, x_{2S,i} < \ |x| \le x_{2S,i+1}\} & \text{if } i < N_D - 1 \\ \{\dot{x} \in \mathbb{R} |\, x2_{S,i} < \ |x| \le \infty\} & \text{if } i = N_D - 1 \end{cases} \tag{34}$$

Parameters for simulation purposes are shown in Table 2 and Table 3.

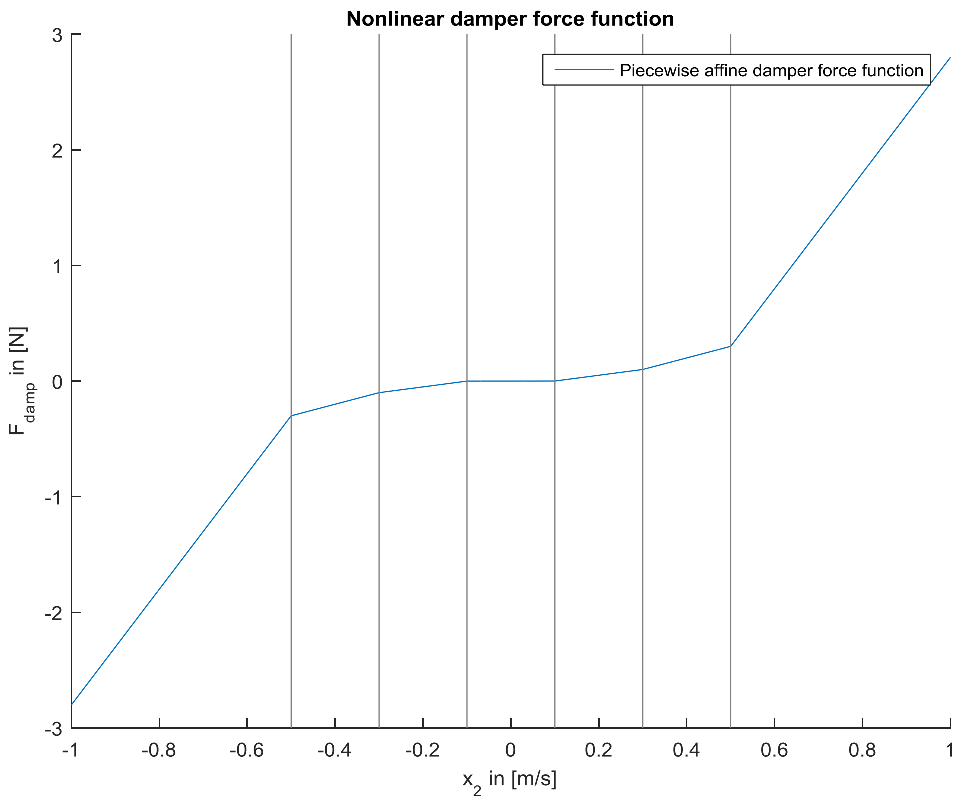

Figure 15: Nonlinear (piecewise affine) damper force function

Table 3: Parameters for the piecewise affine damper force function

N_d	$x_{2S,i}$	d_i	$F_{damp,off,i}$
	$\left[\frac{m}{s}\right]$	$\left[\frac{Ns}{m}\right]$	$[N]$
4	$[0\ \ 0.1\ 0.3\ 0.5]$	$[0.025\ \ 0.5\ \ 1\ \ 5]$	$[0\quad -0.0498\quad -0.1997\quad -2.1997]$

The system differential equations are still affine inside regions depended on both coordinates, which can be assumed as affine area boxes in the phase plot. Assuming that $x_1 \in I_i$ and $x_2 \in J_j$ the system equation for the behavior inside an affine area can be written as follows in (35).

$$\dot{\mathbf{x}}(\mathrm{t}) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \mathbf{x}(\mathrm{t}) - \begin{bmatrix} 0 \\ \frac{1}{\mathrm{m}} \end{bmatrix} \left(\left(\mathrm{c_i x_1} + \mathrm{sign}(\mathrm{x_1})\ \mathrm{F_{off,i}} \right) + \right. \\ \left. (\mathrm{d_j x_2} + \mathrm{sign}(\mathrm{x_2})\ \mathrm{F_{damp\_off,j}}) \right) + \begin{bmatrix} 0 \\ \frac{1}{\mathrm{m}} \end{bmatrix} \mathrm{u}(\mathrm{t}) \tag{35}$$

Repeating the previous steps for the velocity variable x_2 the system can be simplified as shown in equation (36) and (37). The nonlinear system now is described as a set of affine systems. At a transition of one region to another it is essential to shift the coordinate system since linear system theory does not support constants in the differential equation. Assuming that $x_2 \in J_j$ the following equations hold:

$$\begin{aligned} \dot{\mathbf{x}}(\mathrm{t}) &= \begin{bmatrix} 0 & 1 \\ -\frac{\mathrm{c_k}}{\mathrm{m}} & -\frac{\mathrm{d_l}}{\mathrm{m}} \end{bmatrix} \mathbf{x}(\mathrm{t}) + \begin{bmatrix} 0 \\ \frac{1}{\mathrm{m}} \end{bmatrix} \mathrm{u}(\mathrm{t}) - \begin{bmatrix} 0 \\ \frac{1}{\mathrm{m}} \end{bmatrix} (\ \mathrm{F_{off,k}} + \mathrm{F_{damp\_off,l}}) \\ &= \mathbf{A}_{\mathrm{lin,kl}} \mathbf{x}(\mathrm{t}) + \mathbf{b}\mathrm{u}(\mathrm{t}) + \boldsymbol{\xi}_{\mathrm{const,kl}} \end{aligned} \tag{36}$$

$$\boldsymbol{\xi}_{\mathrm{const,kl}} = -\begin{bmatrix} 0 \\ \frac{1}{\mathrm{m}} \end{bmatrix} (\ \mathrm{sign}(\mathrm{x_1})\mathrm{F_{off,k}} + \mathrm{sign}(\mathrm{x_2})\mathrm{F_{damp\_off,l}}). \tag{37}$$

The phase plot of the autonomous system with parameters according to Table 1 is shown in Figure 16. It is obvious that the system has a low damping in the area around zero velocity. Consequently it is prone to a low damped oscillation.

In phase plots the two nonlinear dependencies are visible in steepness of the deflection between two zero crossings of the velocity. If there is no nonlinear dependency of the damper force, the slope of the phase plot is rather congruent in the horizontal perspective as opposed to the nonlinear phase plot curve.

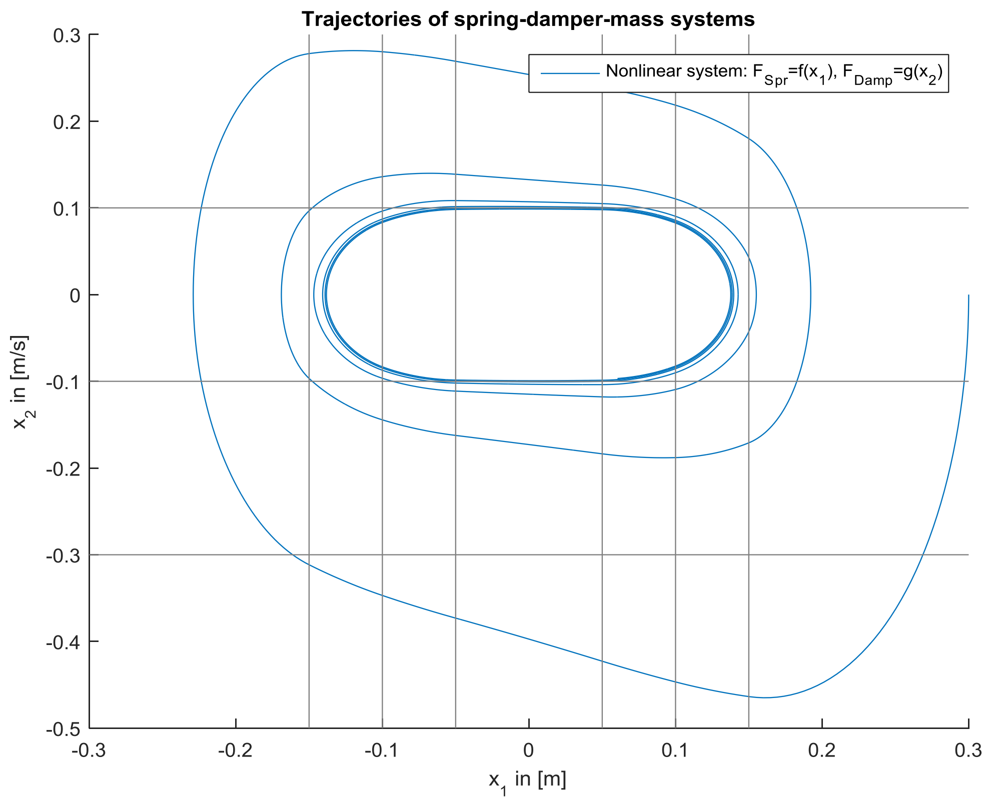

Figure 16: Phase plot of a nonlinear system in two variables

3.1.4 Nonlinear spring force and nonlinear damper force dependent on two state variables

In real mechanical systems it might be necessary to handle a deflection dependent damper force, due to the construction of spring damper elements. At least there might be a maximum deflection of the damper at which it hits construction limits. Presuming a symmetrical construction of the damper, the function has to be symmetric to the deflection variable $x_1$and point symmetric to the speed variable x_2. Again it is possible to use a piecewise affine function for both variables. For modeling coulomb friction a discontinuity (e.g. signum function) at a velocity of $x_2 = 0$ is necessary. A basic function with rather affine dependencies on each variable is used to demonstrate this.

$$F_{Damp,nlin2}\big(x_1(t), x_2(t)\big) = d_0 \operatorname{sign}(x_2) + d_\Delta |x_1| x_2 \tag{38}$$

For demonstration purposes parameters of Table 4 have been used.

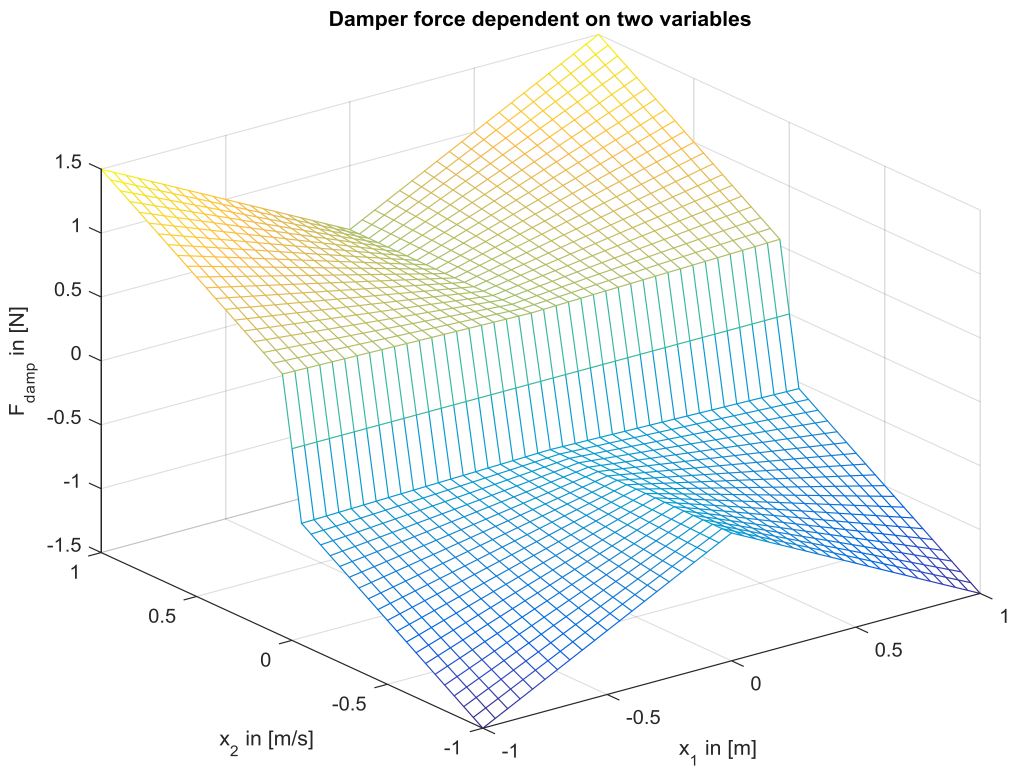

Figure 17: Nonlinear damper force dependent on both variables

Table 4: Parameters for nonlinear damper force in two variables

d_0	d_Δ
$\left[\frac{Ns}{m}\right]$	$\left[\frac{Ns}{m^2}\right]$
0.5	1

Equation (39) denotes the system equation of the nonlinear system and its phase plot is shown in Figure 18. The signum function, acting as coulomb friction, leads to a static equilibrium point, which is not at zero position.

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \mathbf{x}(t) - \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} \Big(\big(c_i x_1 + \mathrm{sign}(x_1)\, F_{\mathrm{off},i} \big) + \\ \big(d_0 \mathrm{sign}(x_2) + d_\Delta |x_1| x_2 \big) \Big) + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u(t) \tag{39}$$

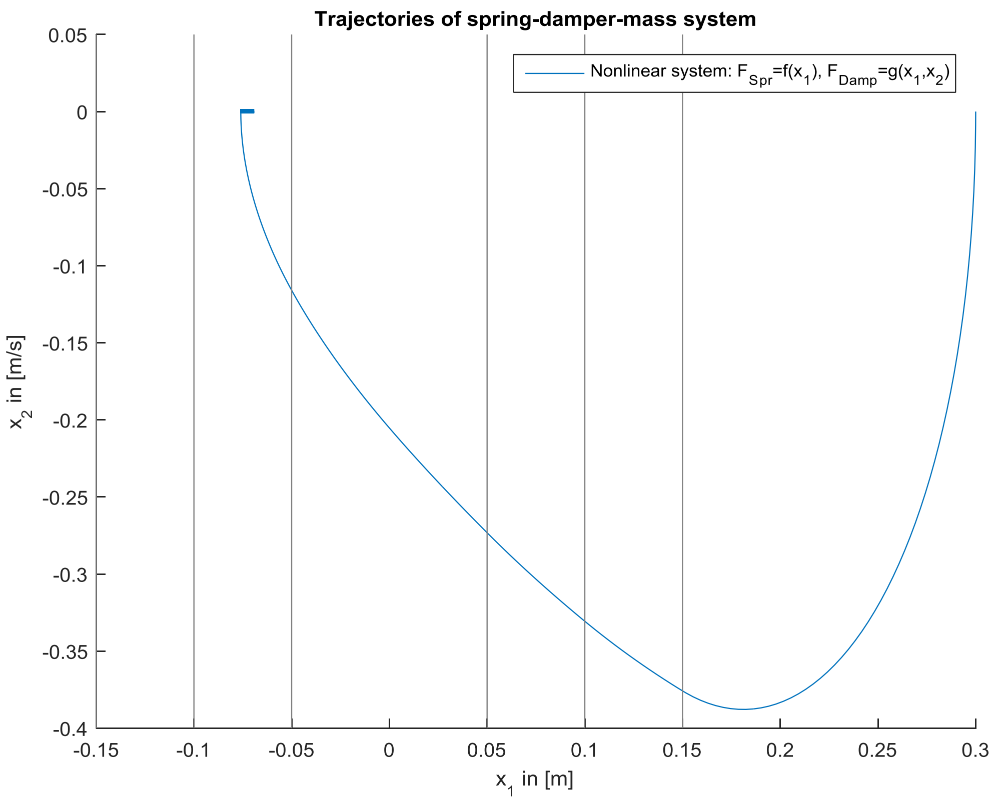

Figure 18: Phaseplot of nonlinear System with damperforce dependent on two variables

3.2 DISCRETIZATION METHODS FOR A CERTAIN SPRING-DAMPER CONFIGURATION

When it now comes to real time simulation, a spring damper mass system is used, which consists of a configuration seen in chapter 3.1.2 and an additional coulomb friction element for the damper which is added later in this chapter. The basic methods are derived just for the nonlinear spring force, since the damper force can be handled similar to existing methods used in the model [3].

A major goal for the capability of real time simulation is a short computation time of the numeric algorithm used to meet the dead line requirement and the ability for integration into the existing model [9]. Long step sizes usually lead to inaccuracy.

Several concepts of the nonlinear integration ability are being investigated and compared in terms of accuracy, algorithm performance and integration potential into the existing solution. Most concepts combine linear discretization methods with the additional integration of nonlinearities by means of coordinate transformation or switching models. This approach leads to a straight forward integration mechanism for existing linear models.

3.2.1 NUMERICAL SOLVERS FOR NONLINEAR DIFFERENTIAL EQUATIONS

For a general nonlinear model, consisting of a set of ordinary differential equations, various methods are well investigated and offered in software like MATLAB®/Simulink® . Variable step solvers are not suitable for an embedded real time software system. The step size can only be lowered until the maximum processor computation capacity is reached. Variable step solvers fit the step size to signal properties. If the signal is about to change rapidly for instance, the step size is lowered to achieve a better simulation result. Since a real time embedded system always has a certain computation time reserved for a simulation algorithm, there is no point in extending the step size, but it is most important to estimate the maximum computation time of the solver method used and meet the deadline requirement to ensure the real time requirements.

As a basis for numerical solution, the continuous nonlinear model (40) is required at first.

$$\dot{\mathbf{x}}(\mathrm{t}) = \mathbf{f}\big(\mathbf{x}(\mathrm{t})\big) + \mathbf{b}\mathrm{u}(\mathrm{t}) \tag{40}$$

3.2.1.1 FIRST ORDER FIXED-STEP METHOD (EULER ALGORITHM)

The most straight forward method to solve nonlinear equations is the first order fixed-step method or Euler algorithm, also offered by MATLAB®/Simulink® . It basically linearizes the nonlinear equation to calculate the following state vector as shown in equations (41) to (43).

$$\dot{\mathbf{x}}(\mathrm{t}) = \mathbf{f}\big(\mathbf{x}(\mathrm{t})\big) + \mathbf{b}\mathrm{u}(\mathrm{t}) \approx \frac{\mathbf{x}(\mathrm{kT_d} + \mathrm{T_d}) - \mathbf{x}(\mathrm{kT_d})}{\mathrm{T_d}} \tag{41}$$

$$\begin{aligned} \mathbf{x}_{\mathrm{k+1}} &= \mathbf{x}(\mathrm{kT_d} + \mathrm{T_d}) \\ \mathbf{x}_{\mathrm{k}} &= \mathbf{x}(\mathrm{kT_d}) \\ \mathrm{u}_{\mathrm{k}} &= \mathbf{x}(\mathrm{kT_d}) \end{aligned} \tag{42}$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k + T_d(\mathbf{f}(\mathbf{x}_k) + \mathbf{b}u_k) \tag{43}$$

Figure 19 shows the phase plot of an example discretization with a step size of $T_d = 0.01s$ and system values according to Table 5. It is obvious that for the piecewise affine system this method is an easy computable approximation.

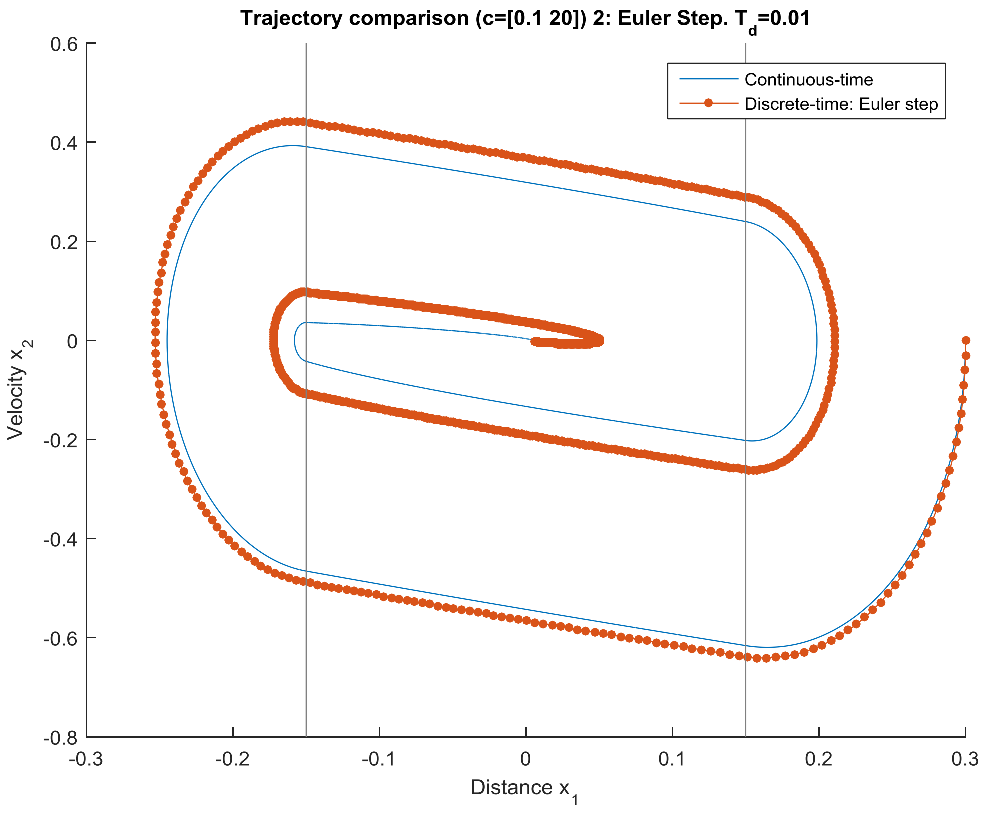

Figure 19: Discretization model: Euler step. Phase plot.

3.2.1.2 Matrix exponentials

The fundamental ideas to this approach are derived from [10] and for calculating the consecutive state vector $\mathbf{x}_{k+1}$, equation (47) has to be solved. The basis of this idea is a matrix exponential approximation of a nonlinear equation (44) in its general form for one time step, seen in equation (46), where N_M is a positive integer which defines the approximation accuracy and $\mathbf{I}$ is the identity matrix. Here the piecewise affine spring force function of (28) is used.

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{f}(\mathbf{x}(t)) + \mathbf{g}(\mathbf{x}(t))u(t) = \mathbf{f}(\mathbf{x}(t)) + \mathbf{b}u(t) \tag{44}$$

$$\frac{d\mathbf{x}(t)}{dt} = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{d}{m} \end{bmatrix} \mathbf{x}(t) - \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} \left(c_i x_1 + \mathrm{sign}(x_1)\, F_{off,i}\right) + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u(t), \text{for } x_1 \in I_i \tag{45}$$

$$e^{\mathbf{Z}} = \lim_{N_M \to \infty} \left(\mathbf{I} + \frac{\mathbf{Z}}{N_M}\right)^{N_M} \cong \left(\mathbf{I} + \frac{\mathbf{Z}}{N_M}\right)^{N_M} \tag{46}$$

We consider the time interval $t \in [t_k, t_{k+1})$ and it is supposed that $u(t) = u_k, t \in [t_k, t_{k+1})$ and $x(t = Tk) = x_k$. It is necessary to find the values for $h_k, \mathbf{J}_k$ and $\tilde{\mathbf{f}}_k$ in each time sample time step with equation (48) to (49).

$$\mathbf{x}_{k+1} = \mathbf{x}_k + [\mathbf{I} \quad 0] \begin{bmatrix} \mathbf{I} + h_k \mathbf{J}_k & h_k \tilde{\mathbf{f}}_k \\ \mathbf{0}^{\mathbf{T}} & 1 \end{bmatrix}^N \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix} \tag{47}$$

$$h_k = \frac{t_{k+1} - t_k}{N_M} = \frac{T_d}{N_M} \tag{48}$$

$$\mathbf{J}_k = \frac{\partial \mathbf{f}(\mathbf{x}_k)}{\partial \mathbf{x}} + \frac{\partial \mathbf{g}(\mathbf{x}_k)}{\partial \mathbf{x}} u_k \tag{49}$$

$$\tilde{\mathbf{f}}_k = \mathbf{f}(\mathbf{x}_k) + \mathbf{g}(\mathbf{x}_k) u_k \tag{50}$$

In a piecewise affine function, the parameters result from an easy calculation by means of equations (51) and (52):

$$\mathbf{J}_k = \begin{bmatrix} 0 & 1 \\ -\frac{c_i}{m} & -\frac{d}{m} \end{bmatrix}, \text{for } x_1 \in I_i. \tag{51}$$

$$\tilde{\mathbf{f}}_k = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{d}{m} \end{bmatrix} \mathbf{x}_k - \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} \left(c_i x_{1,k} + \mathrm{sign}\left(x_{1,k}\right) F_{off,i}\right) + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u_k, \text{for } x_{1,k} \in I_i \tag{52}$$

3.2.1.3 TAYLOR SERIES

The basic ideas to this approach have their origin in [10]. For the solution of (44) Taylor series [11] are used as follows in equation (53) to (57).

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \sum_{l=1}^{\infty} \frac{T_d^l}{l!} \frac{\partial^l \mathbf{x}}{\partial t^l}\bigg|_{t_k} = \mathbf{x}_k + \sum_{l=1}^{\infty} \mathbf{A}^l\left(\mathbf{x}(k), u(k)\right) \frac{T_d^l}{l!} \tag{53}$$

The vector $\mathbf{A}^l$ can be calculated recursively with equations (54) and (55).

$$\mathbf{A}^1\big(\mathbf{x}(k), u(k)\big) = \mathbf{f}(\mathbf{x}_k) + \mathbf{g}(\mathbf{x}_k)u_k \tag{54}$$

$$\mathbf{A}^{l+1}\big(\mathbf{x}(k), u(k)\big) = \frac{\partial \mathbf{A}^l\big(\mathbf{x}(k), u(k)\big)}{\partial \mathbf{x}} [\mathbf{f}(\mathbf{x}_k) + \mathbf{g}(\mathbf{x}_k)u_k] \tag{55}$$

In the case of our piecewise affine function the first two A-vectors are calculated as follows. Further approximation is not necessary for this purpose of comparing the methods. We use equation (28) for the piecewise affine spring force function $F_{spr}\big(x_{1,k}\big)$ and get the following equations for $x_{1,k} \in I_i$:

$$\mathbf{A}^1\big(\mathbf{x}(k), u(k)\big) = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{d}{m} \end{bmatrix} \mathbf{x}_k - \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} \big(c_i x_{1,k} + \operatorname{sign}\big(x_{1,k}\big) F_{off,i}\big) + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u_k \tag{56}$$

$$\begin{aligned} \mathbf{A}^2\big(\mathbf{x}(k), u(k)\big) &= \frac{\partial \mathbf{A}^1\big(\mathbf{x}(k), u(k)\big)}{\partial \mathbf{x}} [\mathbf{f}(\mathbf{x}_k) + \mathbf{g}(\mathbf{x}_k)u_k] \\ &= \begin{bmatrix} 0 & 1 \\ -\frac{c_i}{m} & -\frac{d}{m} \end{bmatrix} [\mathbf{f}(\mathbf{x}_k) + \mathbf{g}(\mathbf{x}_k)u_k] \\ &= \begin{bmatrix} \frac{F_{spr}\big(x_{1,k}\big)}{m} - \frac{d}{m} x_{2,k} + \frac{u_k}{m} \\ -\frac{c_i}{m} x_{2,k} - \frac{d}{m^2} \big(F_{spr}\big(x_{1,k}\big) - d x_{2,k} + u_k\big) \end{bmatrix} \end{aligned} \tag{57}$$

It is obvious that Taylor series require much more pre-work than the matrix exponential method, but once calculated, the computation effort of either algorithm is similar.

Figure 20 and Figure 21 show the phase plot and the transient comparison of either method with a the computation approximation factor N=2 for both and a sample period of $T_d = 0.02s$. System values have been chosen according to Table 2.

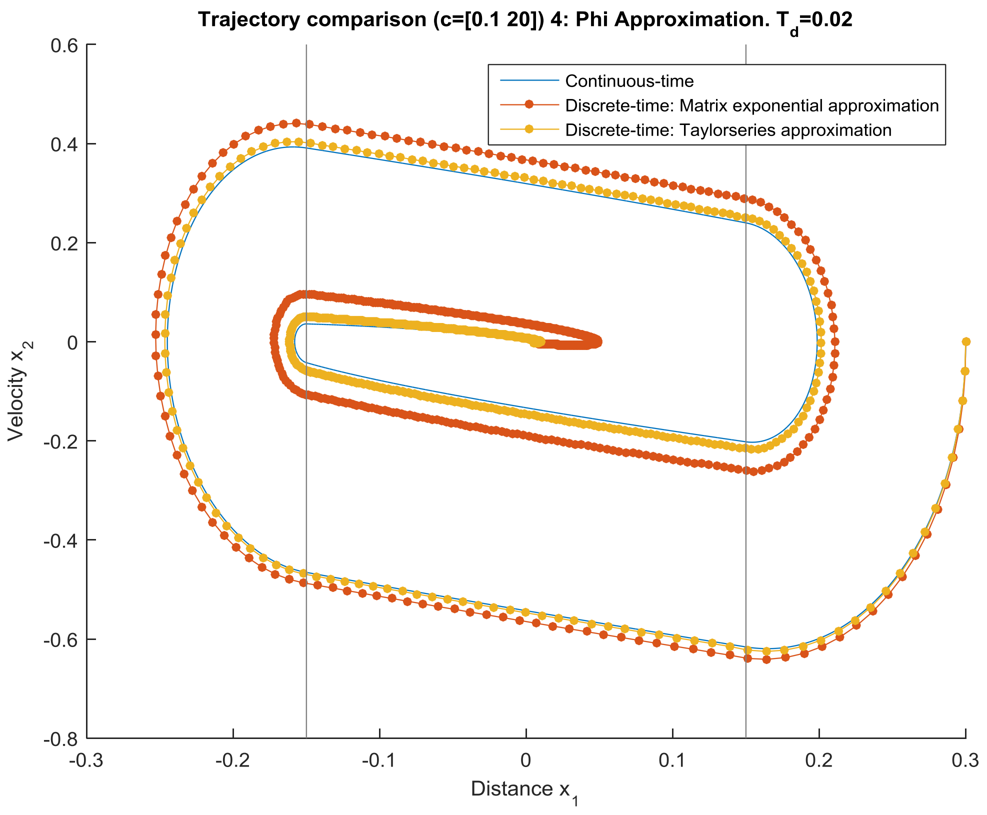

Figure 20: Discretization model: Matrix exponentials and Taylor series. Phase plot.

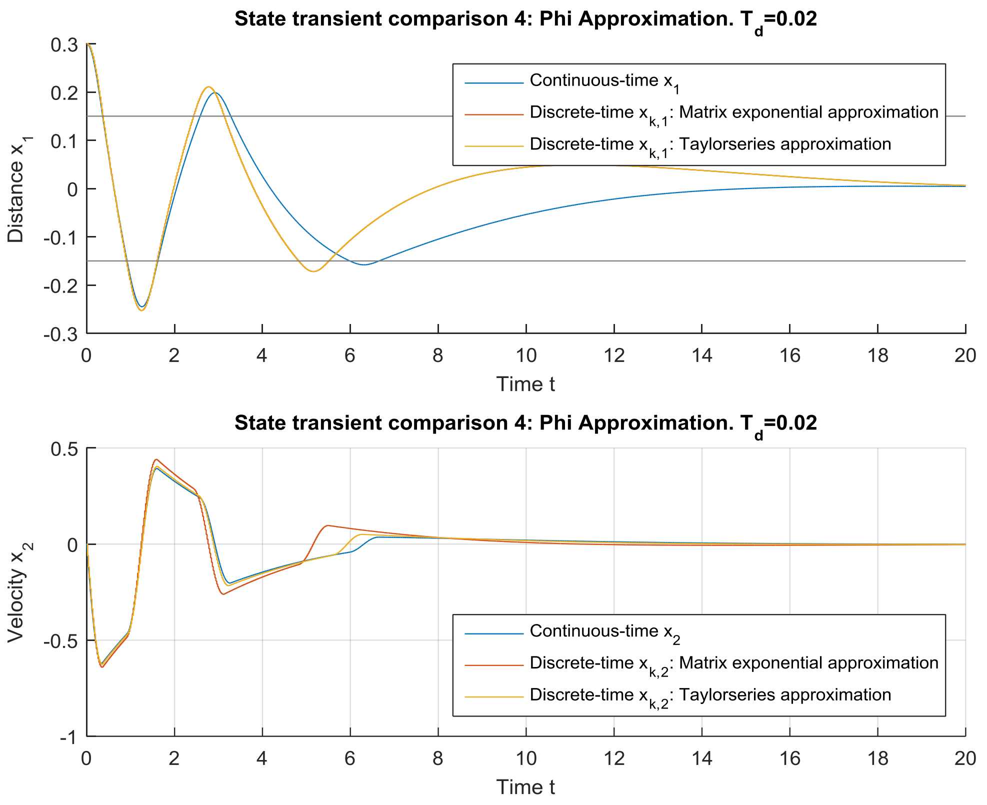

Figure 21: Discretization model: Matrix exponentials and Taylor series. Transient plot.

Apparently the Taylor series approximation works better in this case, but it is important to remember that computation time is not taken into account here. Since matrix exponential approximation might need a shorter computation time, higher parameter values for N are possible.

3.2.2 Approaches for integrating nonlinearities into linear system discretization

Due to already implemented linear models with a nonlinear feedback [3], it is obvious to extend linear models by further nonlinear feedback mechanisms as shown in this chapter.

3.2.2.1 Approximating the nonlinearity as constant for one sample period

This approach also holds for nonlinearities that are not piecewise constant. As we saw in equation (30) we can write the system as a linear part and a nonlinear part. We suppose that following simplification holds for one period of the sample time $\mathrm{T_d}$.

$$\mathrm{F_{Spr,nlin}}\big(\mathrm{x_1(t)}\big) \approx \mathrm{F_{Spr,nlin}}\big(\mathrm{x_1(kT_d)}\big) \quad \text{for } \mathrm{t} \in [\mathrm{kT_d}, (\mathrm{k}+1)\mathrm{T_d}) \tag{58}$$

Consequently the spring force can be seen as constant input for the linear part of the system.

$$\begin{aligned}\dot{\mathbf{x}}(\mathrm{t}) &= \begin{bmatrix} 0 & 1 \\ 0 & -\frac{\mathrm{d}}{\mathrm{m}} \end{bmatrix}\mathbf{x}(\mathrm{t}) - \begin{bmatrix} 0 \\ \frac{1}{\mathrm{m}} \end{bmatrix}\mathrm{F_{Spr,nlin}}\big(\mathrm{x_1(kT_d)}\big) + \begin{bmatrix} 0 \\ \frac{1}{\mathrm{m}} \end{bmatrix}\mathrm{u(t)} \\ &= \mathbf{A}_{\mathrm{lin}}\mathbf{x}(\mathrm{t}) + \boldsymbol{\zeta}_{\mathrm{k}} + \mathbf{b}\mathrm{u(t)} \quad \text{for } \mathrm{t} \in [\mathrm{kT_d}, (\mathrm{k}+1)\mathrm{T_d})\end{aligned} \tag{59}$$

The Laplace transformation [12] delivers an approach for solving the differential equation with the constant input variable $\boldsymbol{\zeta}_{\mathrm{k}}$ as shown in equation (60) to (65).

$$\mathbf{x}_0 = \mathbf{x}(\mathrm{t}=0) \tag{60}$$

$$\mathcal{L}\big(\dot{\mathbf{x}}(\mathrm{t})\big) = \mathrm{s}\mathbf{X}(\mathrm{s}) - \mathbf{I}\mathrm{x}_0 = \mathbf{A}_{\mathrm{lin}}\mathbf{X}(\mathrm{s}) + \frac{1}{\mathrm{s}}\boldsymbol{\zeta}_{\mathrm{k}} + \mathbf{b}\mathrm{U(s)} \tag{61}$$

$$(\mathrm{s}\mathbf{I} - \mathbf{A}_{\mathrm{lin}})\mathbf{X}(\mathrm{s}) = \mathbf{x}_0 + \frac{1}{\mathrm{s}}\boldsymbol{\zeta}_{\mathrm{k}} + \mathbf{b}\mathrm{U(s)} \tag{62}$$

$$\mathbf{X}(\mathrm{s}) = (\mathrm{s}\mathbf{I} - \mathbf{A}_{\mathrm{lin}})^{-1}\mathbf{x}_0 + (\mathrm{s}\mathbf{I} - \mathbf{A}_{\mathrm{lin}})^{-1}\frac{1}{\mathrm{s}}\boldsymbol{\zeta}_{\mathrm{k}} + (\mathrm{s}\mathbf{I} - \mathbf{A}_{\mathrm{lin}})^{-1}\mathbf{b}\mathrm{U(s)} \tag{63}$$

$$\mathbf{\Phi}_{\mathrm{lin}}(\mathrm{t}) = \mathcal{L}^{-1}((\mathrm{s}\mathbf{I} - \mathbf{A}_{\mathrm{lin}})^{-1}) \tag{64}$$

$$\mathcal{L}^{-1}(\mathbf{X}(\mathrm{s})) = \mathbf{x}(\mathrm{t}) = \mathbf{\Phi}_{\mathrm{lin}}(\mathrm{t})\mathbf{x}_0 + \int_{\tau=0}^{\mathrm{t}} \mathbf{\Phi}_{\mathrm{lin}}(\mathrm{t}-\tau)\mathbf{b}\mathrm{u}(\mathrm{t})\mathrm{d}\tau + \int_{\tau=0}^{\mathrm{t}} \mathbf{\Phi}_{\mathrm{lin}}(\tau)\boldsymbol{\zeta}_{\mathrm{k}}\mathrm{d}\tau \tag{65}$$

One constant calculation step can now be solved and the matrices $\mathbf{h}(\mathrm{T_d}), \mathbf{L}(\mathrm{T_d})$and $\mathbf{\Phi}_{\mathrm{lin}}(\mathrm{T_d})$ can be used to compute a fast simulation step, shown in equation (67).

$$\begin{gathered}
\mathrm{u}(\mathrm{t}) = \mathrm{u_k} \text{ for } \mathrm{t} \in [\mathrm{kT_d}, (\mathrm{k}+1)\mathrm{T_d}) \\
\mathbf{x}_{\mathrm{k}} = \mathbf{x}(\mathrm{t} = \mathrm{kT_d}) \\
\mathbf{x}_{\mathrm{k+1}} = \mathbf{x}(\mathrm{t} = (\mathrm{k}+1)\mathrm{T_d}) \\
\mathbf{h}(\mathrm{T_d}) = \int_{\tau=0}^{\mathrm{T_d}} \mathbf{\Phi}_{\mathrm{lin}}(\tau)\mathbf{b}\mathrm{d}\tau \\
\mathbf{L}(\mathrm{T_d}) = \int_{\tau=0}^{\mathrm{T_d}} \mathbf{\Phi}_{\mathrm{lin}}(\tau)\mathrm{d}\tau
\end{gathered} \tag{66}$$

$$\mathbf{x}_{\mathrm{k+1}} = \mathbf{\Phi}_{\mathrm{lin}}(\mathrm{T_d})\mathbf{x}_{\mathrm{k}} + \mathbf{h}(\mathrm{T_d})\mathrm{u_k} + \mathbf{L}(\mathrm{T_d})\boldsymbol{\zeta}_{\mathrm{k}} \tag{67}$$

In the consecutive step the nonlinear variable $\boldsymbol{\zeta}_{\mathrm{k+1}}$ has to be calculated from the nonlinear equation, before equation (67) can be applied again.

$$\boldsymbol{\zeta}_{\mathrm{k+1}} = -\begin{bmatrix} 0 \\ \frac{1}{\mathrm{m}} \end{bmatrix} \mathrm{F_{Spr,nlin}}(\mathrm{x_1}((\mathrm{k}+1)\mathrm{T_d})) \tag{68}$$

Figure 22 shows the MATLAB®/Simulink® model of the corresponding implementation, but with the additional input of a nonlinear damper force, which is set to zero in this case. This method is useful for general nonlinear functions. The piecewise affinity of piecewise affine force functions is not taken into an advantage.

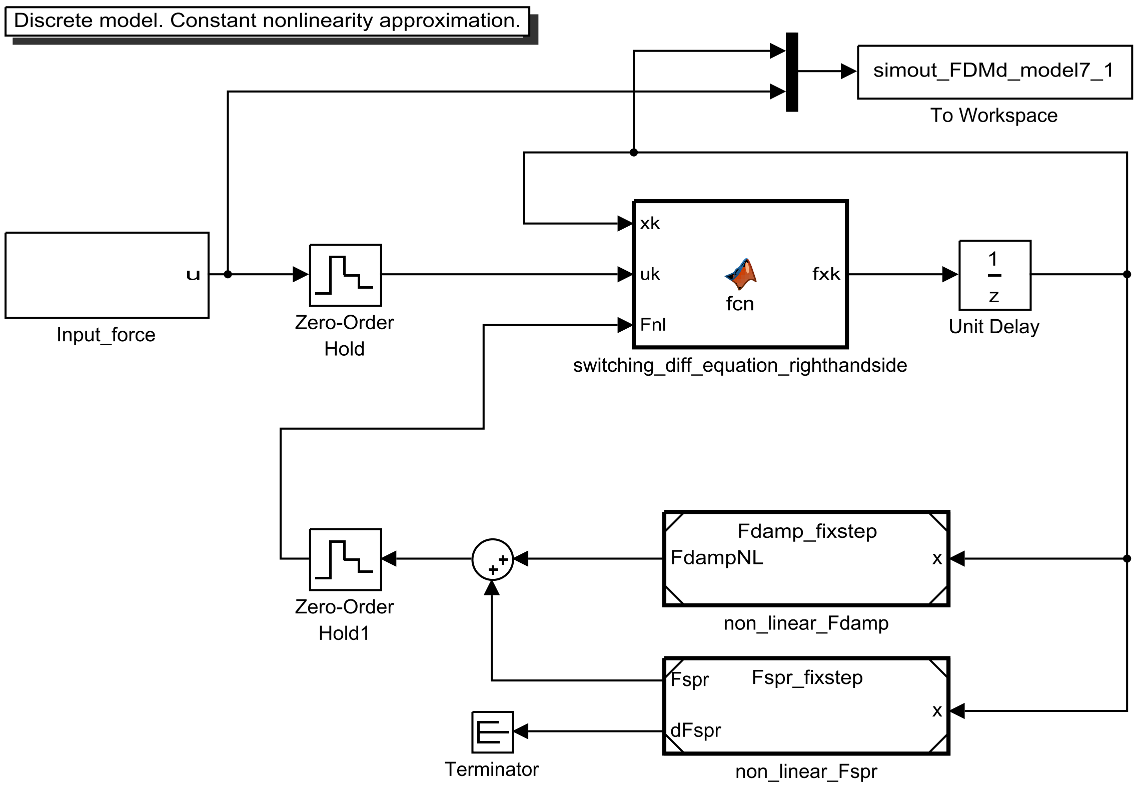

Figure 22: MATLAB®/Simulink® model of the constant nonlinearity approximation

As an improvement in accuracy a solution is presented, in which the nonlinear function is approximated with a linear function foremost and the deviation between the linear and nonlinear function, which is about of being smaller, is used in the feedback loop thus the discretization error subsides.

Therefore following variables have to be changed in equation (58).

$$\begin{aligned} F_{Spr,nlin}\big(x_1(t)\big) &= \frac{\bar{c}}{m}x_1(t) + \left(F_{Spr,nlin}\big(x_1(t)\big) - \frac{\bar{c}}{m}x_1(t)\right) \\ &= \frac{\bar{c}}{m}x_1(t) + \Delta F_{Spr,nlin}\big(x_1(t)\big) \end{aligned} \tag{69}$$

$$\mathbf{A}_{lin} = \begin{bmatrix} 0 & 1 \\ -\frac{\bar{c}}{m} & -\frac{d}{m} \end{bmatrix} \tag{70}$$

$$\Delta\boldsymbol{\zeta}_k = -\begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} \Delta F_{Spr,nlin}\big(x_1(kT_d)\big) \tag{71}$$

Equation (59) becomes the shape of equation (72) and the solution procedure is analog to the previous steps.

$$\dot{\mathbf{x}}(\mathrm{t}) = \mathbf{A}_{\mathrm{lin}}\mathbf{x}(\mathrm{t}) + \Delta\boldsymbol{\zeta}_{\mathrm{k}} + \mathbf{b}\mathrm{u}(\mathrm{t}) \quad \text{for } \mathrm{t} \in [\mathrm{kT_d}, (\mathrm{k}+1)\mathrm{T_d}) \tag{72}$$

In a test case the simulation was performed for a discretization time $\mathrm{T_d} = 0.2$ and system parameters according to Table 1and Table 5.

Table 5: Parameters for piecewise affine spring force in simulation case

N	$x_{S,i}$	c_i
	$[m]$	$\left[\frac{N}{m}\right]$
2	[0 0.15]	[0.1 20]

In a comparative study later in this chapter it will be shown, that all implemented algorithms converge to the continuous time function for $\mathrm{T_d} \rightarrow 0$.

Figure 23 and Figure 25 show the simulation results in the state space and the time transient domain. Obviously the improvement, which uses an internal linear approximation of the spring stiffness, does not deliver outraging better results on the one hand. But on the other hand an increasing step size $\mathrm{T_d}$ leads at a step time of $\mathrm{T_d} = 0.5\mathrm{s}$ to an unstable system only without the internal stiffness improvement (Figure 24).

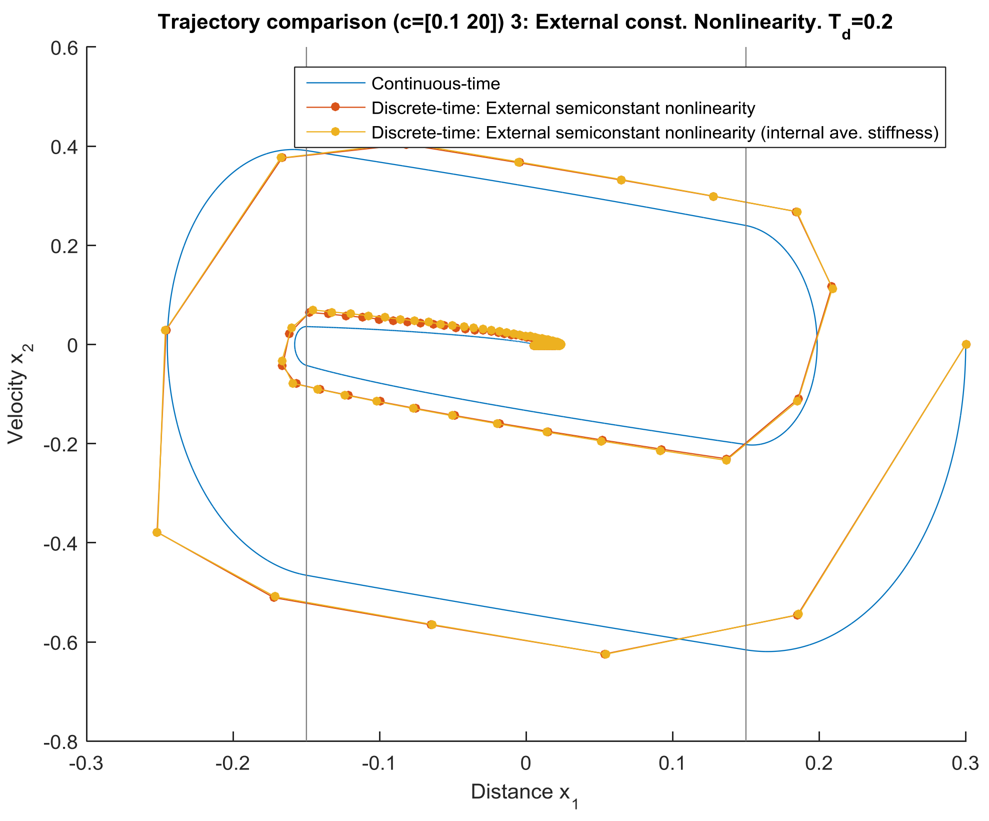

Figure 23: Discretization model: External constant nonlinearity. Phase plot.

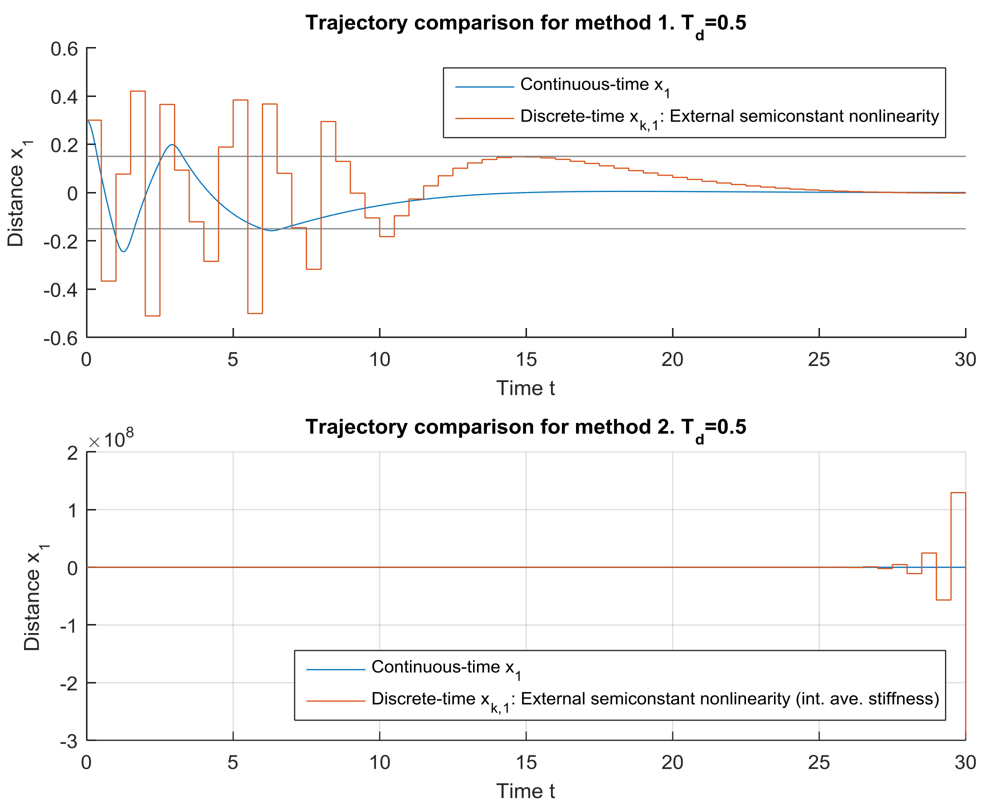

Figure 24: Discretization model: External constant nonlinearity at stability border

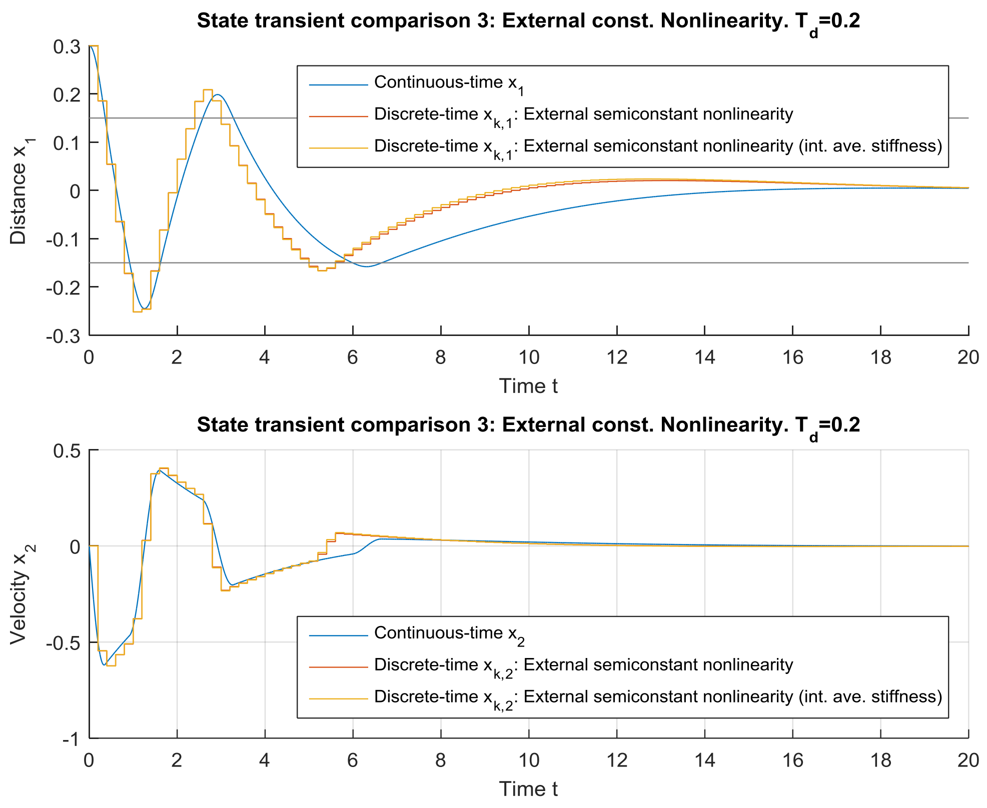

Figure 25: Discretization model: External constant nonlinearity. Transient comparison.

3.2.2.2 Online nonlinear coordinate transformation

A second approach of taking nonlinearities into account is to shift the coordinates corresponding to a nonlinear function dependent on those state variables. Here the spring force dependent on the deflection x_1 is of interest. Consequently the deflection is shifted in a feedback loop to reach the nonlinear result. Because the state variables are constant in the discrete time system for one period also the computed force is constant.

At first an arbitrary $\bar{c}$ is chosen for the system description seen in equation (70) and $\mathbf{\Phi}_{\text{lin}}(T_d)$ is computed by means of equation (64). It is important, that $\bar{c}$ approximates the dynamic system behavior in a region, where the dynamical behavior is of uttermost interest to the simulation. This can be a critical section, where the accuracy is more important or a region, where the system is likely to stay during a long time, since $\mathbf{\Phi}_{\text{lin}}(T_d)$ describes the transition from one $\mathbf{x}_k$ to a $\mathbf{x}_{k+1}$with the system dynamic induced by $\bar{c}$.

As seen in Figure 26, before each calculation, the coordinates are shifted according to equation (77), to achieve nonlinear behavior with the system description in equation (78).

$$F_{Spr,nlin,k} = F_{Spr,nlin}\big(x_1(t = kT_d)\big) \tag{73}$$

$$F_{Spr,lin,k} = \bar{c}x_{1,shifted}(t = kT_d) \tag{74}$$

$$F_{Spr,nlin,k} \overset{!}{=} F_{Spr,lin,k} = \bar{c}x_{1,shifted}(t = kT_d) \tag{75}$$

$$x_{1,shifted,k} = x_{1,shifted}(t = kT_d),\ \ F_{Spr,nlin,k} = F_{Spr,nlin}\big(x_1(t = kT_d)\big) \tag{76}$$

$$x_{1,shifted,k} = \frac{F_{Spr,nlin,k}}{\bar{c}} \tag{77}$$

$$\begin{bmatrix} x_{1,k+1} \\ x_{2,k+1} \end{bmatrix} = \boldsymbol{\Phi}_{lin}(T_d)\begin{bmatrix} x_{1,shifted,k} \\ x_{2,k} \end{bmatrix} + \mathbf{h}(T_d)u_k \tag{78}$$

Figure 26 shows the MATLAB®/Simulink® model of the implemented algorithm. The implementation is straight forward and easy to understand, which one advantage of this method is.

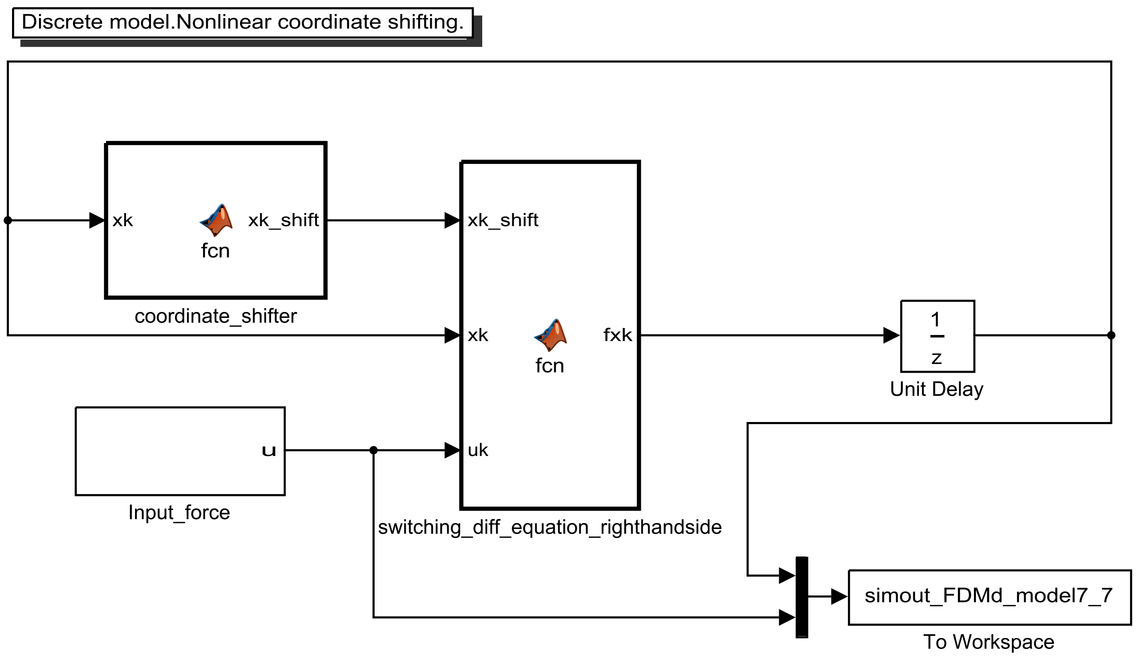

Figure 26: MATLAB®/Simulink® model of the nonlinear coordinate shifting method.

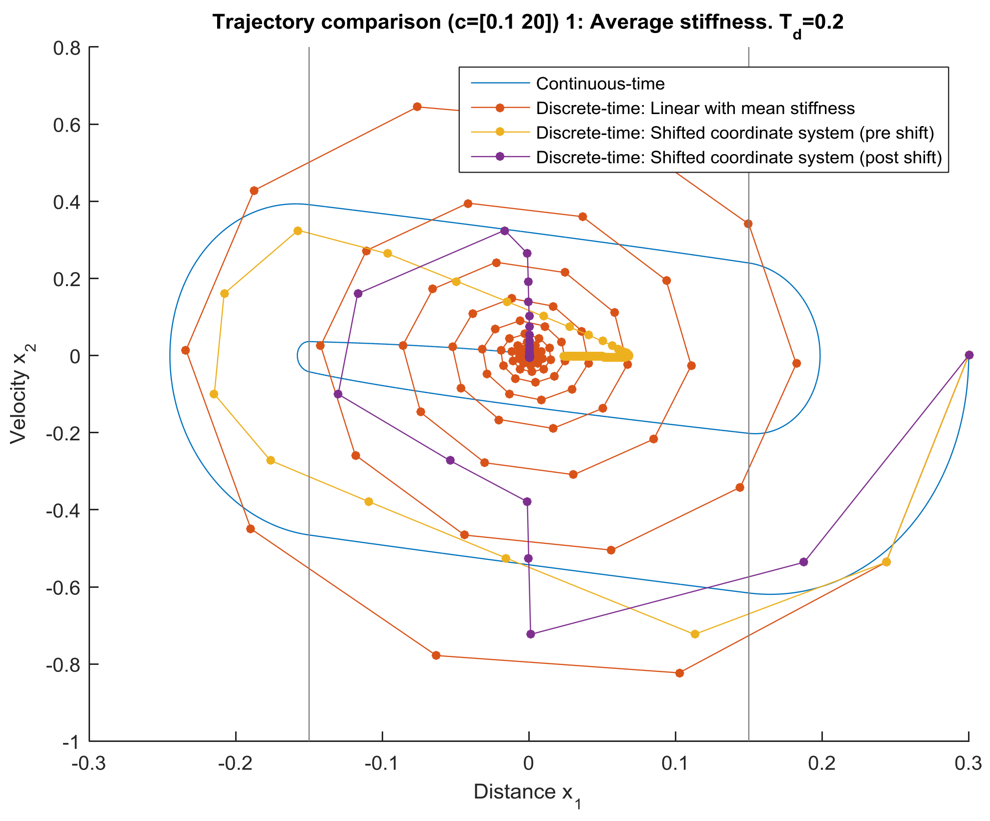

Figure 27: Discretization model: Nonlinear coordinate shifting. Phase plot.

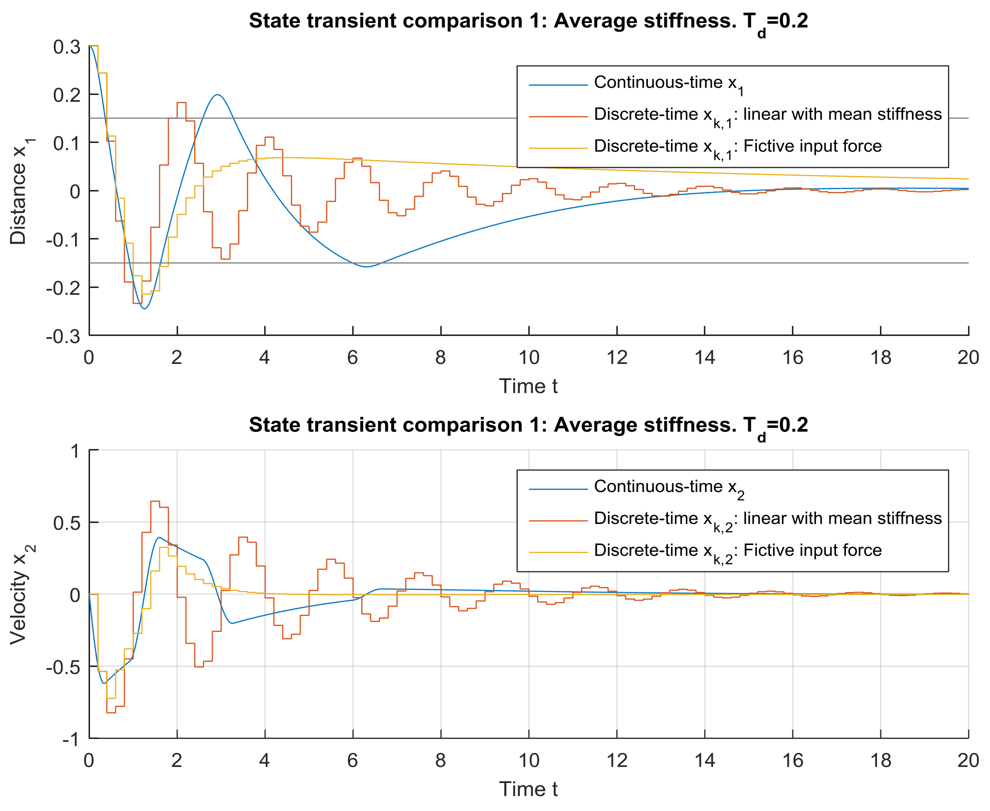

Figure 28: Discretization model: Nonlinear coordinate shifting. Transient comparison.

Figure 27 and Figure 28 show the phase plot and the transient plot of the implemented algorithm. Additionally a linear system with the same system matrix $\mathbf{\Phi}(\mathrm{T_d})$ is plotted to figure out the similar dynamic behavior of one iteration and between the two models. Figure 27 shows the state variable $\mathrm{x_1}$ before and after the shifting operation.

3.2.2.3 SWITCHING AFFINE MODELS

The approach of switching affine models offers the most suitable method of handling piecewise affine systems, which will be shown in a comparative study. Owing to the constant step size $\mathrm{T_d}$, a subset of N affine models can represent the whole dynamic behavior of a piecewise affine model with one nonlinear function, where N represents the number of affine regions in the nonlinear function. Generally this method holds for arbitrary systems with multiple nonlinear systems.

A nonlinear system can either be linearized or the nonlinear function can be given piecewise affine as seen in equation (28). Consider the system in equation (79) which represents a nonlinear system with a piecewise affine spring force dependency. Each region has an affine behavior between, which can be described by means of a constant matrix $\mathbf{\Phi}_\mathrm{j}(\mathrm{T_d})$ and $\mathbf{h}_\mathrm{j}(\mathrm{T_d})$ achieved through methods seen in equation (66). Since the spring force function is a continuous function of the deflection $\mathrm{x_1}$ an offset variable $\boldsymbol{\xi}_\mathrm{const,j}$ is introduced. This variable leads to a necessary coordinate transformation in the discrete time system, if the deflection enters another region and the system matrix is changed. The system comprises N different system descriptions, where N is the number of regions the spring force function enters.

$$\begin{aligned}\dot{\mathbf{x}}(\mathrm{t}) &= \begin{bmatrix} 0 & 1 \\ -\frac{\mathrm{c_j}}{\mathrm{m}} & -\frac{\mathrm{d}}{\mathrm{m}} \end{bmatrix}\mathbf{x}(\mathrm{t}) + \begin{bmatrix} 0 \\ \frac{1}{\mathrm{m}} \end{bmatrix}\mathrm{u}(\mathrm{t}) - \begin{bmatrix} 0 \\ \frac{1}{\mathrm{m}} \end{bmatrix}\mathrm{F_{off,j}} \\ &= \mathbf{A}_\mathrm{lin,j}\mathbf{x}(\mathrm{t}) + \mathbf{b}\mathrm{u}(\mathrm{t}) + \mathrm{sign}(\mathrm{x_1})\boldsymbol{\xi}_\mathrm{const,j} \qquad \text{for } \mathrm{x_1} \in \mathrm{I_j}\end{aligned} \tag{79}$$

$$\begin{gathered}\mathrm{I_j} = \begin{cases} \{\mathrm{x_1} \in \mathbb{R} |\, \mathrm{x_{S,j}} < |\mathrm{x_1}| \leq \mathrm{x_{S,j+1}}\} & \text{if } \mathrm{j} < \mathrm{N} - 1 \\ \{\mathrm{x_1} \in \mathbb{R} |\, \mathrm{x_{S,j}} < |\mathrm{x_1}| \leq \infty\} & \text{if } \mathrm{j} = \mathrm{N} - 1 \end{cases} \\ \mathrm{j} = \{0,1,\dots,\mathrm{N}-1\}\end{gathered} \tag{80}$$

The continuity of the spring force function can be achieved as seen in (81) and (82) and already written in the discrete time domain notation. The unwanted part in equation (79) of $\boldsymbol{\xi}_{\mathrm{const,j}}$ is handled by shifting the coordinates previously to the calculation of the next value $\mathbf{x}_{k+1}$ with the discrete time system matrix $\boldsymbol{\Phi}_{\mathrm{j}}$.

$$\mathbf{x}_{\mathrm{shifted,k,j}} = \mathbf{x}_{\mathrm{k}} + \frac{\mathrm{sign}(x_1)\boldsymbol{\xi}_{\mathrm{const,j}}}{c_{\mathrm{j}}} \tag{81}$$

$$\mathbf{x}_{\mathrm{shifted,k+1,j}} = \boldsymbol{\Phi}_{\mathrm{j}}\mathbf{x}_{\mathrm{sifted,k,j}} + \mathbf{h}_{\mathrm{j}}u_{\mathrm{k}} \tag{82}$$

After the operation, the deflection has to be re-shifted, since the shifting is just carried out for the purpose of the calculation.

$$\mathbf{x}_{\mathrm{k+1}} = \mathbf{x}_{\mathrm{shifted,k+1,j}} - \frac{\mathrm{sign}(x_1)\boldsymbol{\xi}_{\mathrm{const,j}}}{c_{\mathrm{j}}} \tag{83}$$

In an affine region this calculations lead to an exact result. The only error arises on the transition of two regions. The value for $\mathbf{x}_{k+1}$ might be located in region (j+1) while the value for $\mathbf{x}_{\mathrm{k}}$is located in region j. The longer the state variable is located in the relative period of the consecutive region and the higher the gradient difference of the regions is, the higher the error in the calculation of $\mathbf{x}_{k+1}$ will be. Therefore an algorithm was figured out, which recognizes a region transition prior to the event and carries out the transition calculation by means of another algorithm, which principles are described in the chapter 2.2.4. The essential idea is to set a sampling point as close to the region transition point as possible (computation time requirements) and set the region value j to its consecutive value, and afterwards extend the next sample point to its usual location, where it should be located, without an adjustment, thus requiring an extended sample time in the following step.

At first a method is presented, which can estimate the time it takes for a state variable to cross a certain value similar to the method in [3]. Generally equation (84) is valid for affine systems. It describes the calculation of a consecutive step with a variation factor η in the step size T_{d}. Approximation (86) is necessary for further calculations to achieve an analytical solution.

$$\mathbf{x}_{\mathrm{k+\eta}} = \mathbf{x}_{\mathrm{k}} + \eta T_{\mathrm{d}}\delta\mathbf{x}_{\mathrm{k}}(\eta T_{\mathrm{d}}) \tag{84}$$

$$\delta\mathbf{x}_{\mathrm{k}}(\eta T_{\mathrm{d}}) = \frac{1}{\eta T_{\mathrm{d}}}((\boldsymbol{\Phi}(\eta T_{\mathrm{d}}) - \mathbf{I})\mathbf{x}_{\mathrm{k}} + \mathbf{h}(\eta T_{\mathrm{d}})u_{\mathrm{k}}) \tag{85}$$

$$\delta\mathbf{x}_k(\eta T_d) \approx \delta\mathbf{x}_k(T_d) \approx \frac{1}{\eta T_d}((\boldsymbol{\Phi}(T_d) - \mathbf{I})\mathbf{x}_k + \mathbf{h}(T_d)u_k) \tag{86}$$

Let us assume the deflection $x_{S,i}$ is the value for x_1, to enter a consecutive region in the step size variation ηT_d. Then equation (87) holds, where $\mathbf{l} = \begin{bmatrix}1\\0\end{bmatrix}$ selects the deflection state.

$$\mathbf{l}^T\mathbf{x}_{k+\eta} \overset{!}{=} x_{S,i} \tag{87}$$

$$\mathbf{l}^T(\mathbf{x}_k + \eta T_d\delta\mathbf{x}_k(\eta T_d)) \approx \mathbf{l}^T(\mathbf{x}_k + \eta T_d\delta\mathbf{x}_k(T_d)) \overset{!}{=} x_{S,i} \tag{88}$$

The step size variation η and with it the absolute time $kT_d + \eta T_d$ can be calculated by solving the problem for equation (90).

$$\mathbf{l}^T(\mathbf{x}_k + \eta T_d\delta\mathbf{x}_k(\eta T_d)) \approx \mathbf{l}^T(\delta\mathbf{x}_k + \eta T_d\delta\mathbf{x}_k(T_d)) \overset{!}{=} x_{S,i} \tag{89}$$

$$\eta T_d = \frac{x_{S,i} - \mathbf{l}^T\delta\mathbf{x}_k}{\mathbf{l}^T\delta\mathbf{x}_k(T_d)} \tag{90}$$

Once a step size is calculated to set the next sample time to a region transition, the matrix exponential method is carried out as described in the chapter 3.2.1.2, for the two consecutive time steps ηT_d and $(2 - \eta)T_d$.

Figure 29 shows the phase plot of the system, one time using the step size adjustment and one time just using the standard switching system approach. This example is numerically challenging in its parameters since the step size and the stiffness value difference is extremely high, but it shows the good performance of the algorithm, if step size adjustment is used.

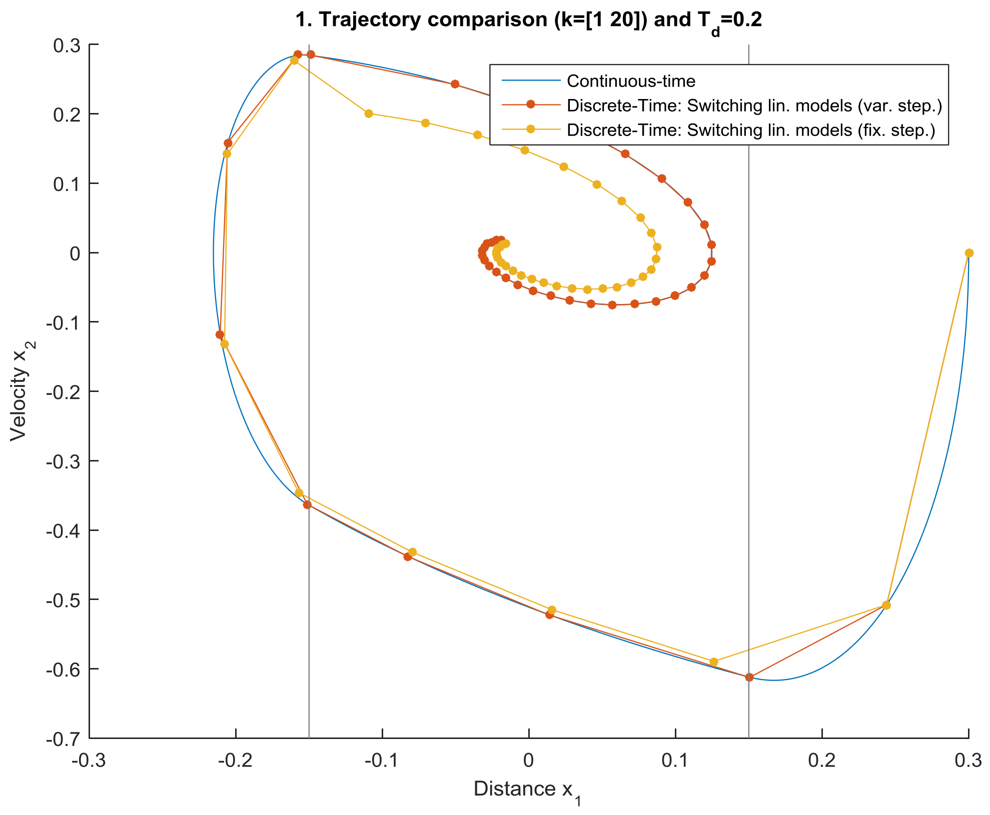

Figure 29: Switching affine models. Phase plot with step size adjustment comparison.

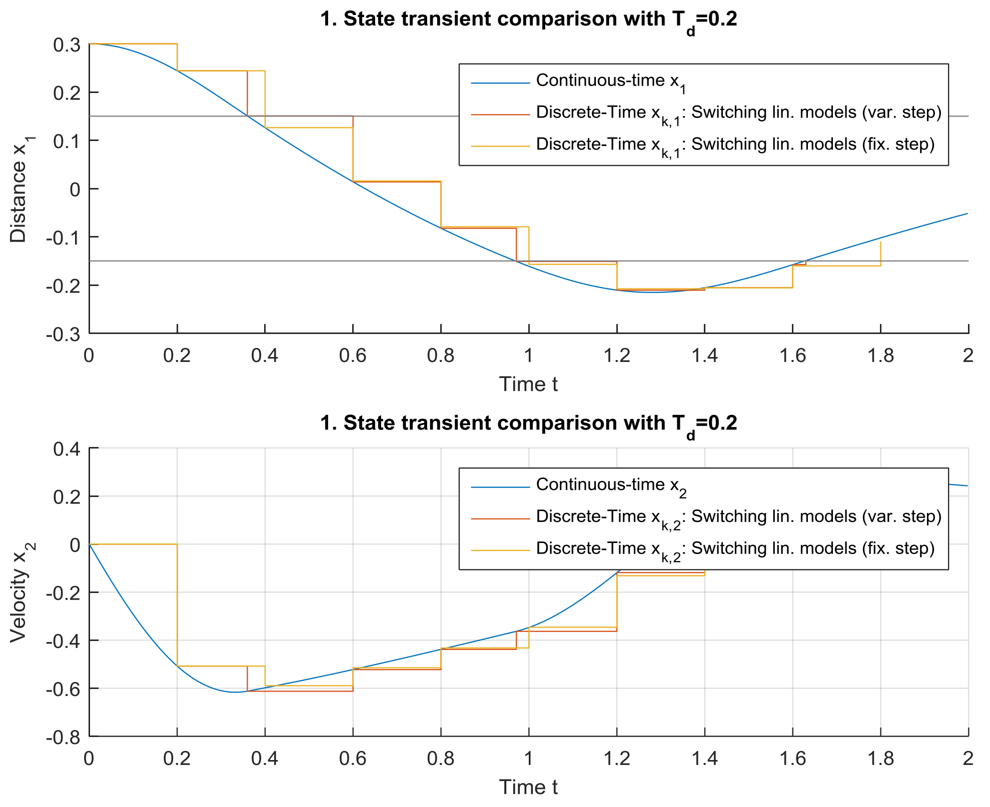

Figure 30: Nonlinear coordinate shifting. Transient and step size adjustment comparison.

3.2.3 Comparison of discretization methods

3.2.3.1 Accuracy with large step sizes

Of course one major quality criteria is the accuracy of the presented solutions. As the step size is increased, algorithms become less accurate. Figure 31 to Figure 36 show three different pairs of transient plots and phase plots for three different step sizes. The first pair shows a very short step size to ensure all algorithms converge towards the true solution. The second and third pair show bigger step sizes.

The algorithm "switching affine systems" is by far the most accurate algorithm, also because it is used with the step size adjustment.

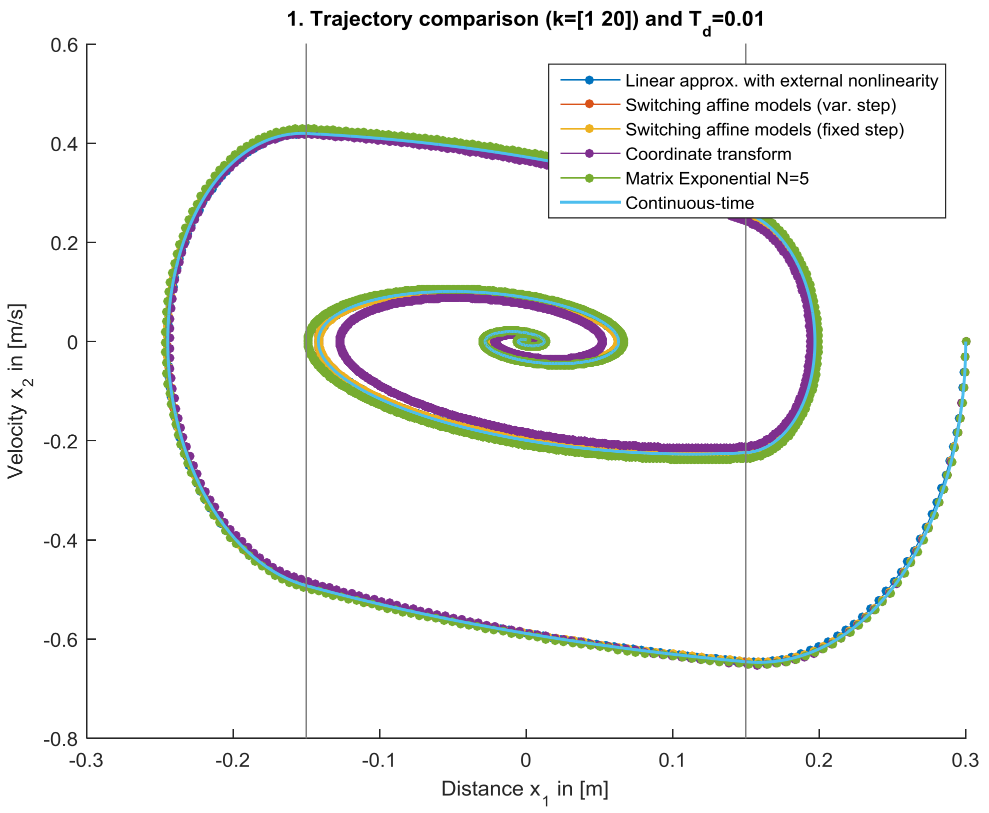

Figure 31: Discretization model comparison 1. Trajectory plot.

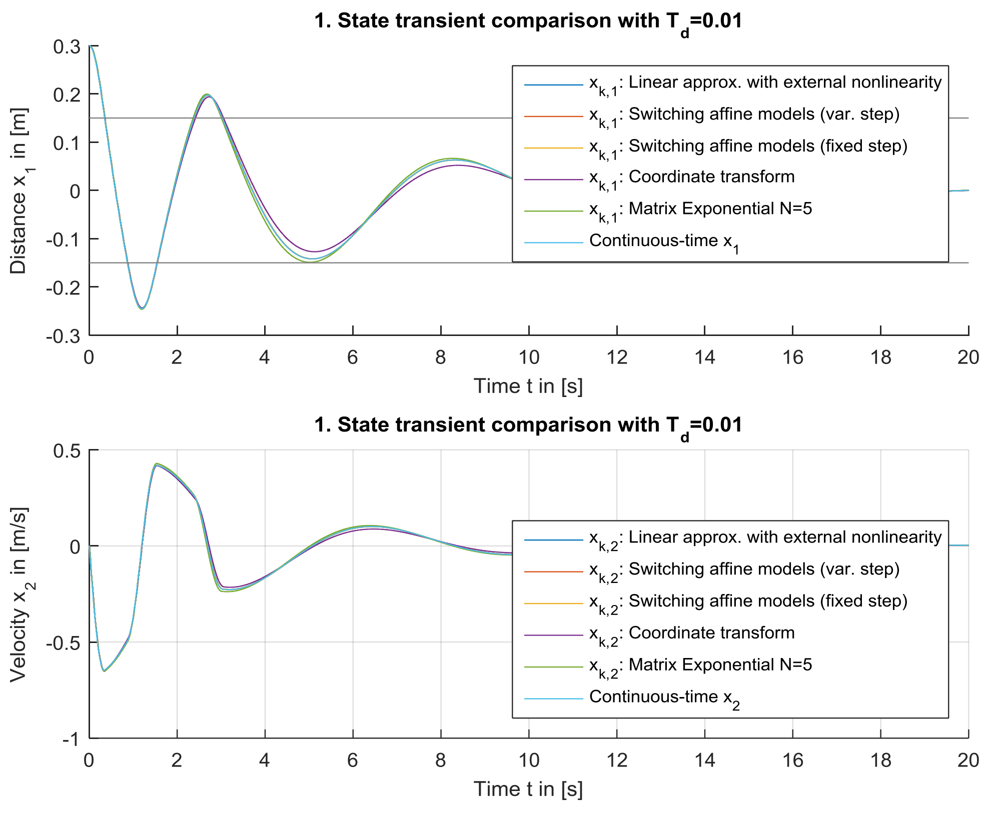

Figure 32: Discretization model comparison 1. Transient plot.

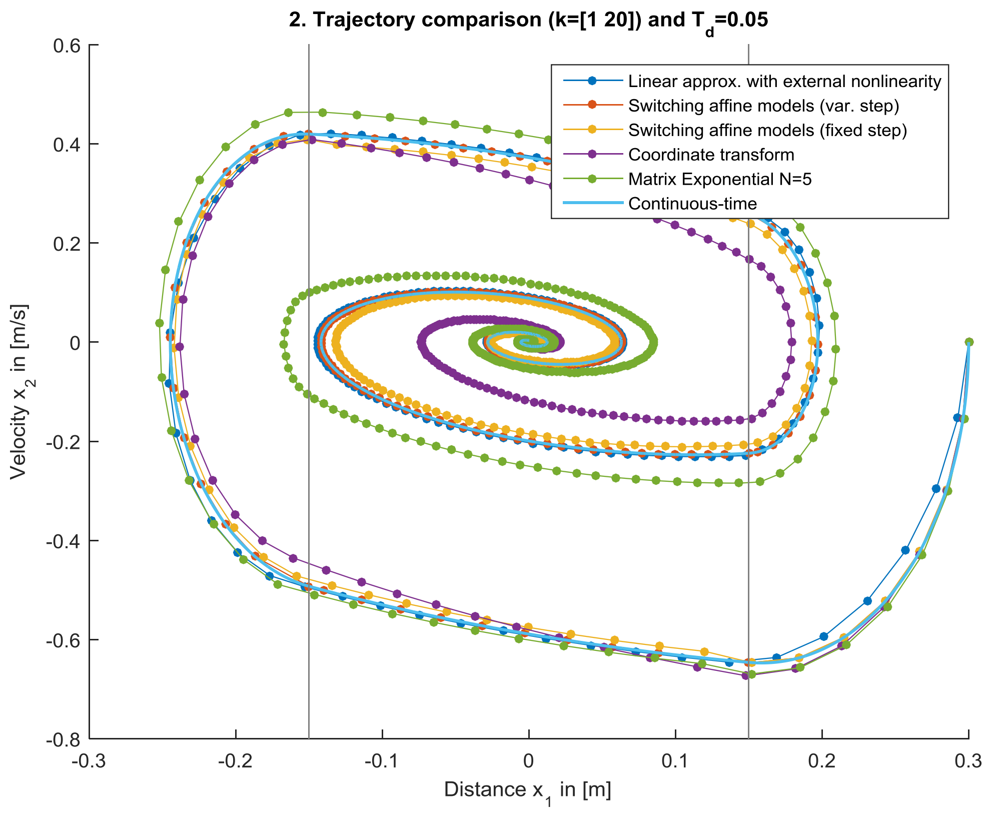

Figure 33: Discretization model comparison 2. Trajectory plot.

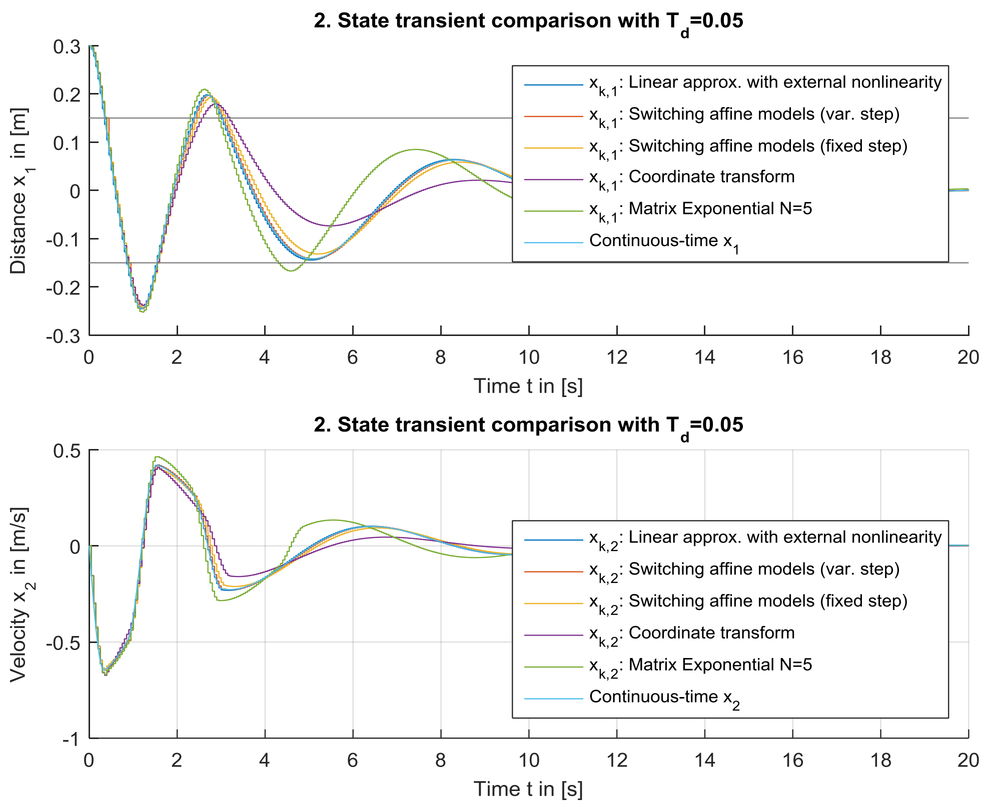

Figure 34: Discretization model comparison 2. Transient plot.

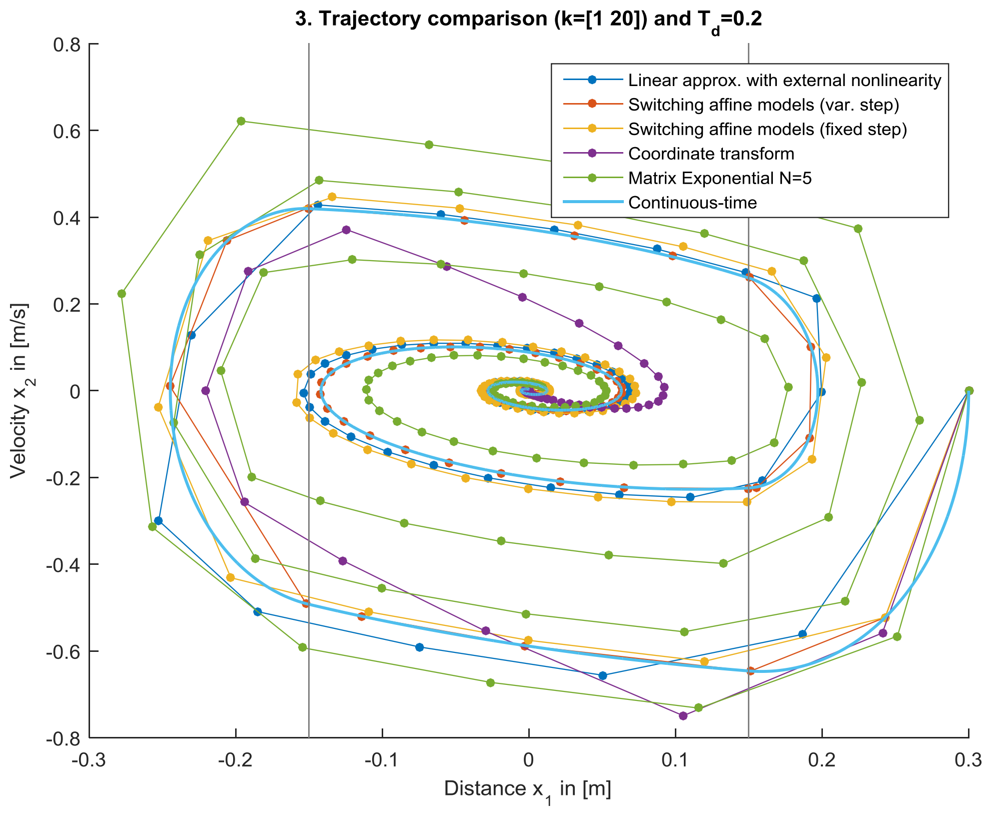

Figure 35: Discretization model comparison 3. Trajectory plot.

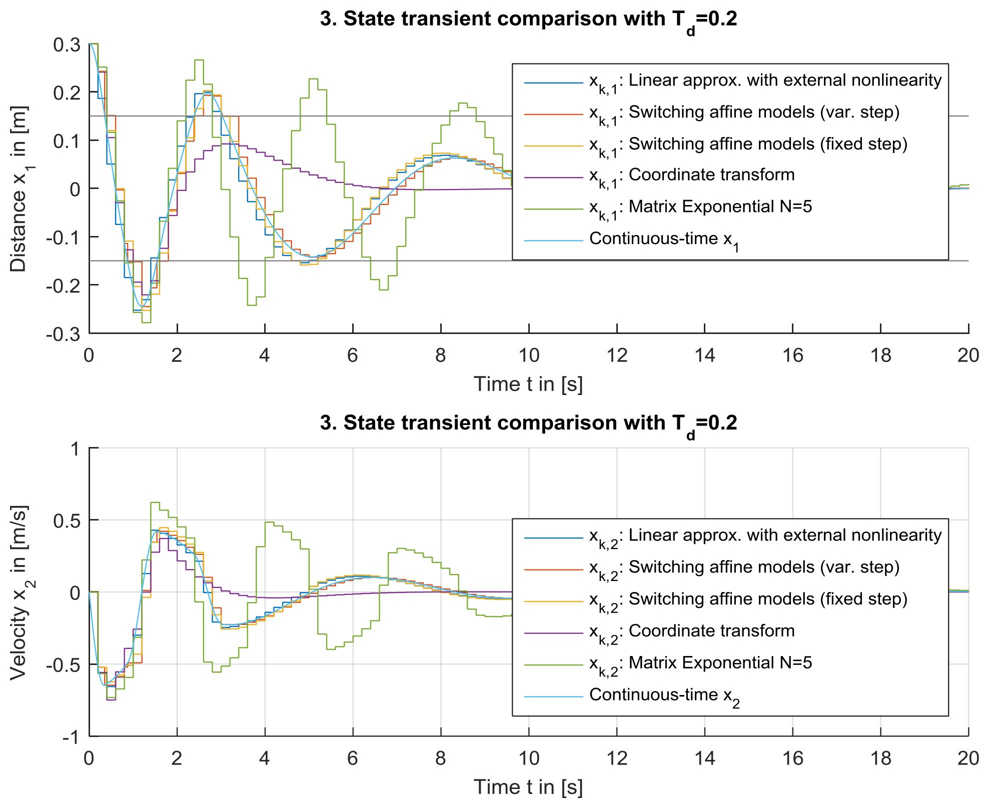

Figure 36: Discretization model comparison 3. Transient plot.

3.2.3.2 Computation time

Another essential criterion to choose an algorithm in association with real time systems and limited computation power, is the maximum computation time for a step. The computation limitation leads to engineering decisions as for a certain processor or the chosen sampling time.

MATLAB®/Simulink® offers possibilities to measure computation time for a simulation, but it is only possible to determine the computation time per step in average, particularly on a non-real-time operating system like Windows®. A coarse overview can be given by determining the MATLAB®/Simulink® computation time in a comparative study with the same parameter setting. Although during testing experiments it was figured out that a doubling in simulation time does not always result in a doubling in computation time, which was expected initially.

Table 7 shows the results of simulation time measurements. It its remarkable, that simulation of the coordinate transforming algorithm do not require a tremendous higher computation time. This circumstance results from the low computational afford in the algorithm implementation.

Step size adjustment requires pretty high computational time, but also delivers a high accuracy. Consequently it is important dependent on the engineering problem, how to choose the algorithm to meet requirements.

The highest computation time in this example is taken by the constant nonlinearity algorithm. This might also origin in the MATLAB®/Simulink® implementation, which uses blocks that are not used by other approaches like the coordinate transformation.

For a higher quality of calculation time analysis, further tools and software would have to be used, and the algorithms would need to be programmed in a comparable programming language, which must also be supported by the final CPU, where the algorithm will be computed on.

Table 6: Parameters for calculation time demonstration

Parameter	*Unit*	*Value*
$x_1(t=0)$	m	0.3
$x_2(t=0)$	m	0
c_i	$\frac{\mathrm{N}}{\mathrm{m}}$	[1 20]
d	$\frac{\mathrm{Ns}}{\mathrm{m}}$	0.5
T_d	s	0.1

Table 7: Computation time for different algorithms

Methode	*Simualtion time in [s] absolute*	*Simulation time in [ms] per step*
Linear discrete system	0.1681	0.562
Approximating constant nonlinearity	0.4474	1.494
Switching affine systems (var. steptime)	0.4050	1.346
Switching affine systems (fix. steptime)	0.3864	1.282
Coordinate transformation	0.1950	0.640
Matrix exponentials (N=10)	0.4204	1.401

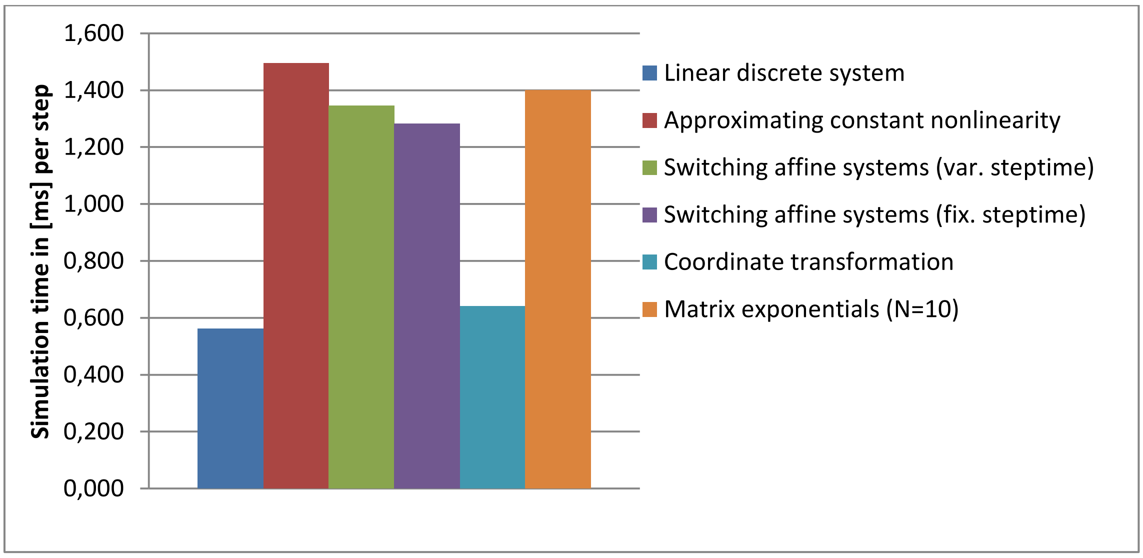

Figure 37: Diagram of the various step sizes of each algorithm

Table 8: Additional information for switching affine systems algorithm

Methode	*Number of step size adjustments*	*Average time of adjustment step in [ms]*
Switching affine systems (var. steptime)	20	9.60

3.2.3.3 Energy leaks and stability

Another vital part of analyzing algorithms for computing differential equations of mechanical systems is to examine basic physical laws in its discrete numerical solution. In particular the system energy has to follow its physical laws. In a simple spring damper mass system the energy must be constant if there is no internal damping that transforms energy into heat. That must be also true for discrete time systems in every discretization point, which will be analyzed in the following chapter.

Energy stability is not necessarily derived from the accuracy of algorithms. It might occur that an algorithm with a lower accuracy shows a symmetrical error distribution which cancel each other out and let the energy fluctuate around a mean value. On the other hand an algorithm might be very accurate but shows an asymmetric error distribution which leads to an unstable algorithm in the worst case, if the damping values are low for example.

Figure 38 shows the system energy of all relevant discretization methods. Obviously two approaches (Coordinate transform method and matrix exponential method with a computation factor N=100) drift rapidly away from the desired constant value, which originates from a damping of zero in this simulation. The energy loss of the coordinate transform method is due to its computation method. One has to keep in mind that this simulation uses no input forces and starts at an initial deflection, thus the autonomous system is subject of consideration.

Although the approach of transforming coordinates and the approach of an external non-linear stepwise constant force are similar in their fundamental idea, no energy drift takes place with the second approach, which can be seen in the red curve in Figure 38. Comparing those two methods, this is an essential advantage.

Taking a closer look at the more accurate calculation methods in Figure 39 a rather negligible drift in the system energy takes place, if the step size is adjusted and a more randomly distribution of the energy error is seen in the switching affine models approach without step size adjustment. The algorithm can be seen as stable if there is at least a small damping in the system parameters.

For the calculation of the discrete system energy first the continuous nonlinear system energy is shown in equation (91) to (93), where the velocity v and the deflection s are used.

$$\mathrm{E_{system}} = \mathrm{E_{kin}} + \mathrm{E_{pot}} \tag{91}$$

$$\mathrm{E_{kin}} = \frac{\mathrm{m x_2^2}}{2} \tag{92}$$

$$\mathrm{E_{pot}} = -\int_{\mathrm{x_{1,0}}}^{\mathrm{x_{1,1}}} \mathrm{F_{spr}(x_1)dx_1} \tag{93}$$

The discretization leads to the following equations (94) to (96) where the index k means the value of the continuous variable at the time $\mathrm{t = T_d k}$. Considering a partly affine spring force curve, the integral of equation (93) can be written as a summation in equation (96).

$$E_{system,k} = E_{kin,k} + E_{pot,k} \tag{94}$$

$$E_{kin,k} = \frac{m x_{2,k}^2}{2} \tag{95}$$

$$E_{pot,k} = -\sum_{i=0}^{k} \left[F_{spr}(x_{1,i})(x_{1,i} - x_{1,i-1}) - \frac{F_{spr}(x_{1,i}) - F_{spr}(x_{1,i-1})}{2(x_{1,i} - x_{1,i-1})} \right] \tag{96}$$

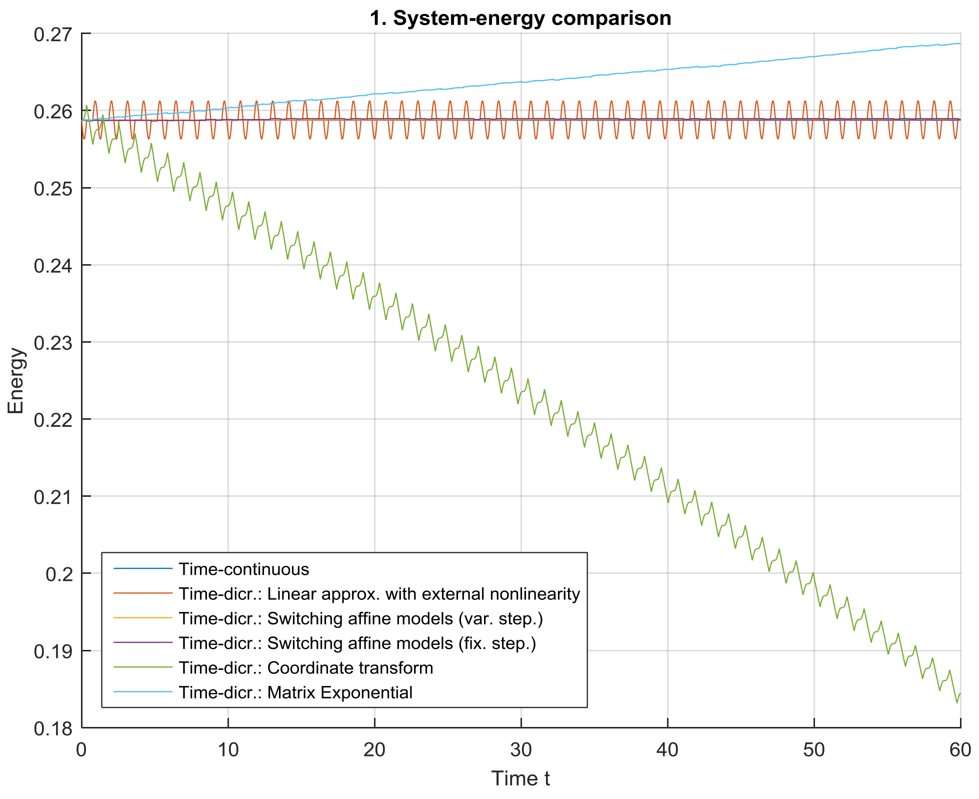

Figure 38: Discretization model energy comparison

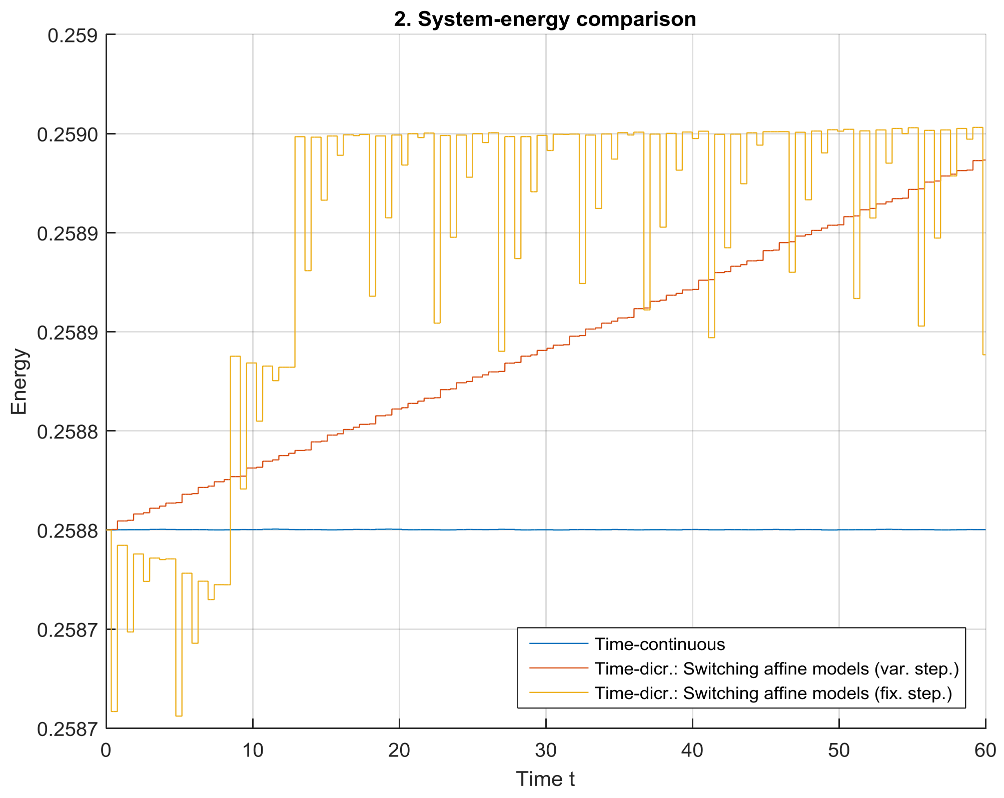

Figure 39: Discretization model energy comparison close-up

3.2.4 Combination of a switching system and static friction discretization with step size adjustment in two coordinates

As described in chapter “Comparison of discretization methods”, the implementation of the switching affine system algorithm with step size adjustment is generally of a high quality by calculating precise values and affording a low computation time.

Since one key element of drivetrain simulations is static friction implementation by implementing algorithms seen in [3], it is subject in this chapter implementing the static friction method into the spring damper mass system analog. Here, a step size adjustment is necessary in the speed variable x_2 to exactly set a sampling point at the speed of zero, where at the switching affine system algorithm with step size adjustment requires an adjustment dependent on the deflection variable x_1, to exactly set the sampling point to a region transition (see APPENDIX A for an investigation of the error made by the used step size adjustment approach). Basically these two adjustments work together fine, but it now must be possible to adapt the step size multiple times consecutively and afterwards regain the original step raster for real time performance. Therefore it is proposed in this work to use a buffer working as storage of step sizes consecutive lower than the value 1, and the buffer is added and cleared if no further step size adjustment is necessary. A step size adjustment takes place to keep up with the real time condition, where the consecutive step is extended by the summation of all missing time lengths $(1 - \eta_j)T_d$. Of course a threshold value limits this buffer usage.

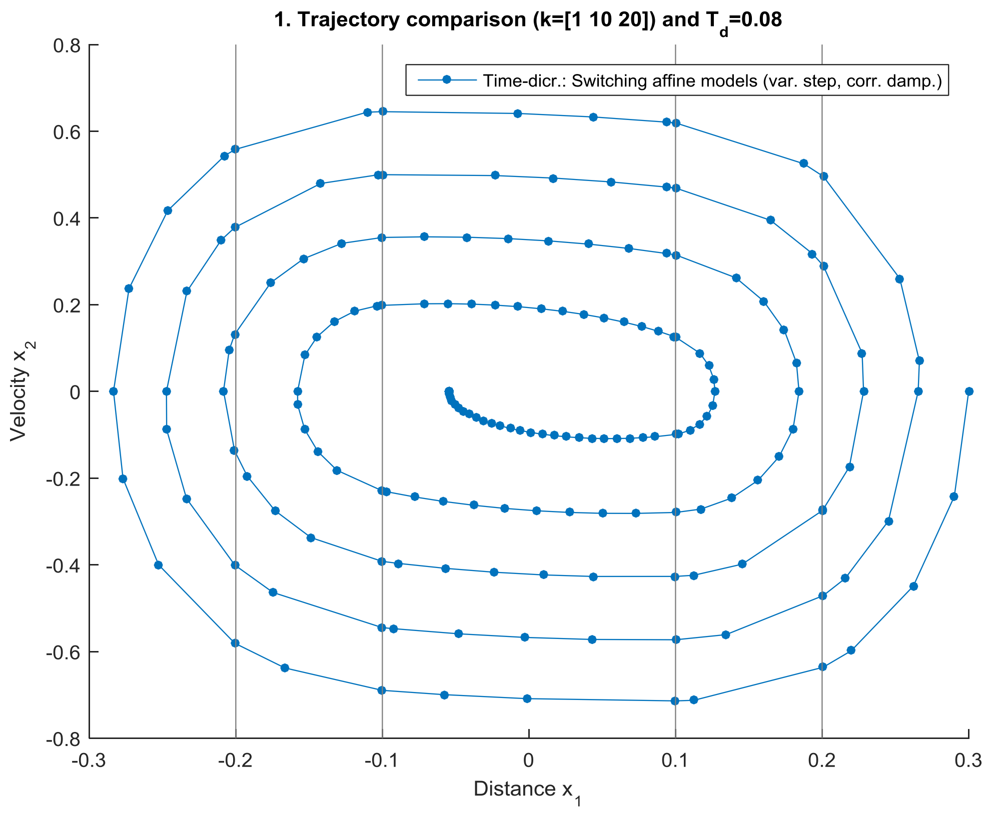

Figure 40: Discretization model: Switching affine models. Phase plot with step size adjustment for static friction and region transition

Figure 40 and Figure 41 show an illustrative example of the step size adjustment in two variables and the static friction implementation with values assigned according to Table 9. The freely oscillating mass stops as the static friction force exceeds the spring force. It is clearly visible, that region crossings in the distance variable and zero crossings in the speed variable always contain a sampling point.

Table 9: Parameters for step size adjustment in two state variables

N	$x_{S,i}$	c_i	F_{Damp0}	d
	$[m]$	$\left[\frac{Ns}{m}\right]$	$[N]$	$\left[\frac{Ns}{m}\right]$
3	[0 0.1 0.2]	[1 10 20]	0.05	0.05

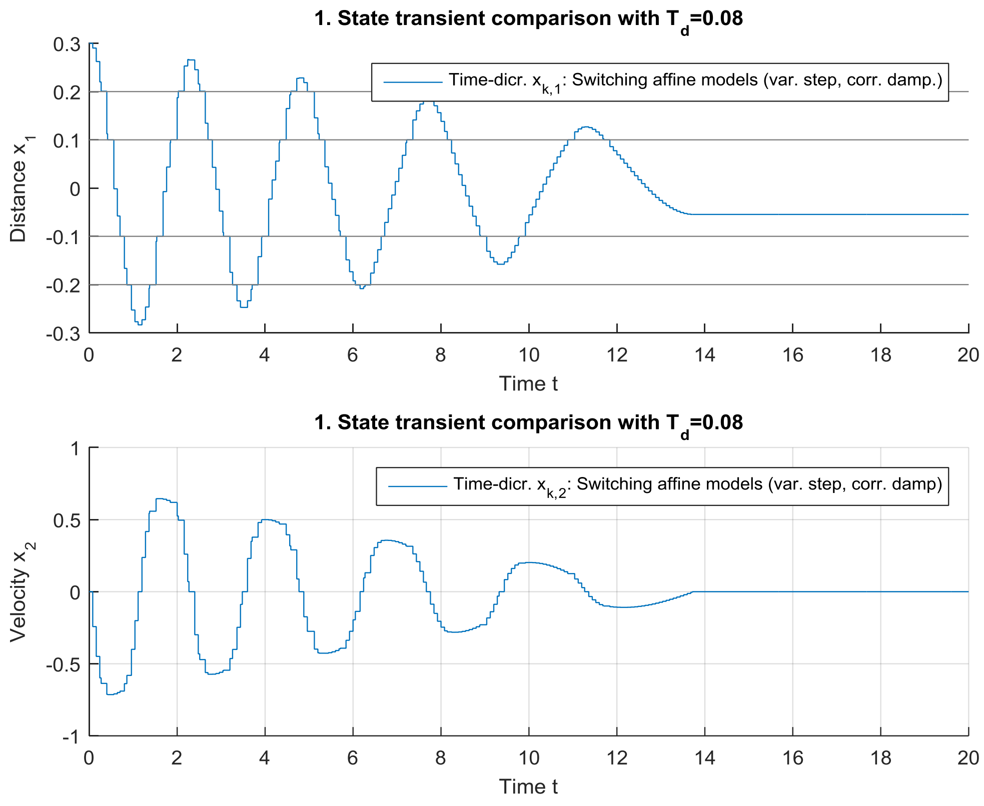

Figure 41: Discretization model: Nonlinear coordinate shifting. Transient plot with step size adjustment for static friction and region transition

3.2.5 Reaction to harmonic input signals

To verify the models reaction for input signals a harmonic input signal $u = \sin(\omega t)$is chosen and the switching affine systems approach will demonstrate the good performance of the discretization method.

Table 10: Parameters for input signal verification

Parameter	Unit	Value
$x_1(t=0)$	m	0.1
$x_2(t=0)$	m	0
c_i	$\frac{\text{N}}{\text{m}}$	[1 1000]
d	$\frac{\text{Ns}}{\text{m}}$	0
T_d	s	0.01
ω	$\frac{\text{rad}}{\text{s}}$	1

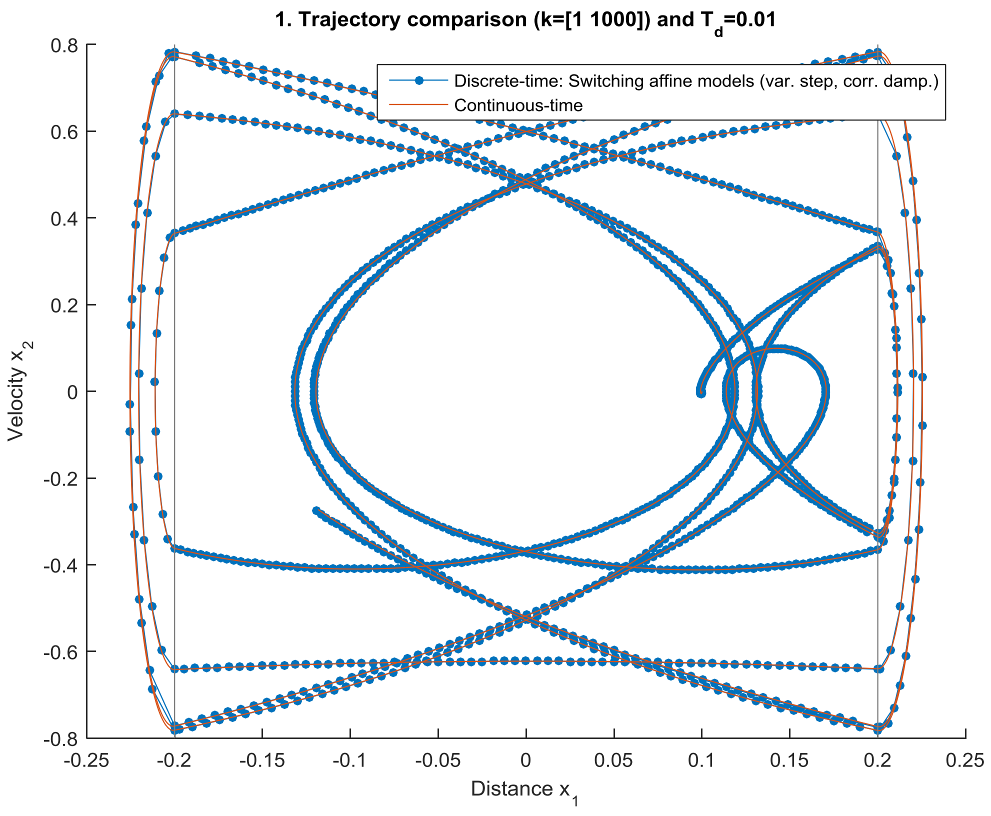

Figure 42: Phase plot of the system with a harmonic input force

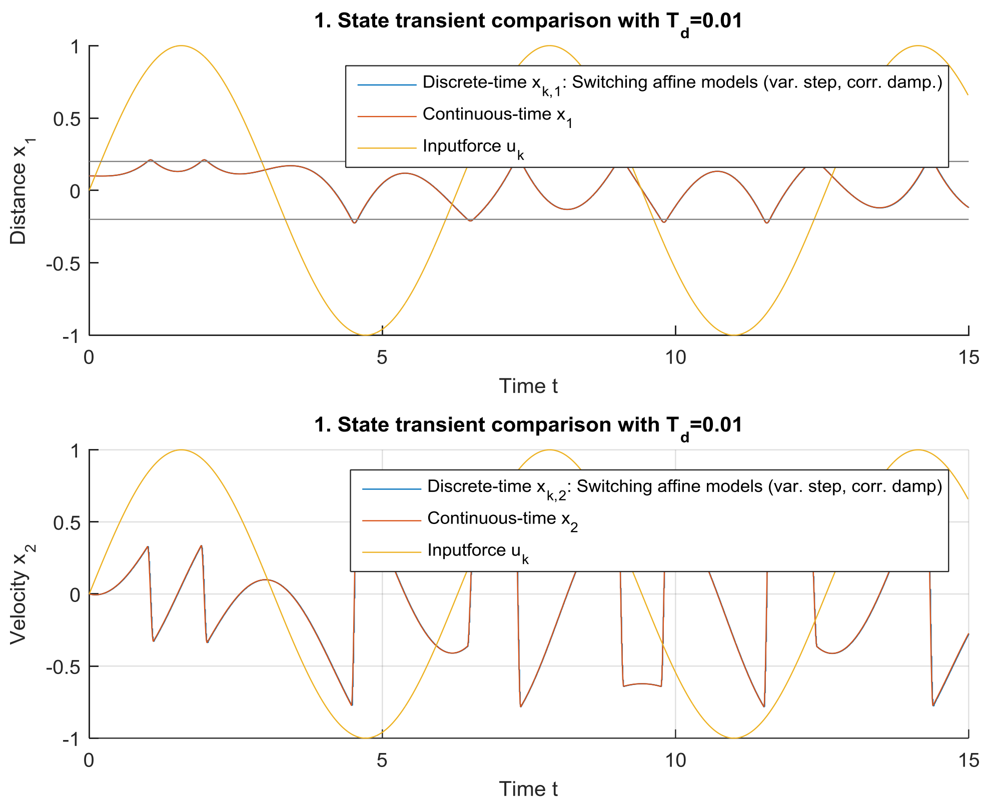

Figure 43: Transient plot of the system with a harmonic input force

Evaluating the performance of the algorithm it is obvious that the algorithm holds for input signals and particularly for this highly nonlinear system configuration.

3.3 INTEGRATION INTO EXISTING MODELS

3.3.1 GENERAL POINT OF VIEW

Closing the loop of this chapter to its initial point, now the potential of the distinctive methods for being integrated into drivetrain models with approaches derived of [3] are shown.

The characteristic of the assumed drivetrain model is its implemented approach for step-time adjustment to set calculation points on slip speed zero crossings and the usage of an external logic, that calculates the torque transferred by clutches. Both base on a affine system model and additionally the requirement for a real time calculation is given, thus requiring fixed time steps.

Since the damper force function is shaped the same way as the clutch force function it can be easily implemented as a clutch with appropriate parameter settings, shown in chapter 3.2.4 for the simple spring damper mass system.

Given the specified piecewise affine curve for the spring force shown in Figure 11 and the resulting changes in the mathematical system description the corresponding logic seen in [3] has to be adapted the same way.

Equation (7) calculates the torque of locked clutches, which is transferred, if the torque is seen as an external input, shown in the following equation (97). The definition of matrices and derivations can be found in chapter 2.2.3. Equation (98) shows the affine model with the nonlinear switching torque feedback and equation (99) is representing the calculation for the relative step size adjustment η_j, with the approximation $\delta\mathbf{x}_k(\eta T_d) \approx \delta\mathbf{x}_k(T_d)$.

$$\boldsymbol{\tau}_{l,k} = -\Xi_k[\boldsymbol{\Phi}\mathbf{x}_k + \mathbf{H}_k\mathbf{v}_k + \mathbf{H}_c\mathbf{K}_{s,k}\boldsymbol{\tau}_{s,k}] \tag{97}$$

$$\mathbf{x}_{k+1} = (\mathbf{I} - \mathbf{H}_c\mathbf{K}_{l,k}\Xi_k)\boldsymbol{\Phi}\mathbf{x}_k + (\mathbf{I} - \mathbf{H}_c\mathbf{K}_{l,k}\Xi_k)\mathbf{H}_v\mathbf{v}_k + (\mathbf{I} - \mathbf{H}_c\mathbf{K}_{l,k}\Xi_k)\mathbf{H}_c\mathbf{K}_{s,k}\boldsymbol{\tau}_{s,k} \tag{98}$$

$$\eta_j \approx -\frac{(\mathbf{B}_C^T\mathbf{M}\mathbf{x}_k)_j}{\left(\mathbf{B}_C^T\mathbf{M}\Delta\delta\mathbf{x}_k(T_d)\right)_j} \tag{99}$$

Considering now the discretization approach of the “switching affine models” described in chapter 3.2.2.3 used for the equations in chapter 2.2.3 these are proposed to be changed the following way (changes are marked in red color).

$$\boldsymbol{\tau}_{l,k} = -\Xi_k[\boldsymbol{\Phi}_j\mathbf{x}_k + \mathbf{H}_k\mathbf{v}_k + (\boldsymbol{\Phi}_j - \mathbf{I})\frac{\xi_{const,j}}{c_j} + \mathbf{H}_c\mathbf{K}_{s,k}\boldsymbol{\tau}_{s,k}] \tag{100}$$

$$\begin{aligned}\mathbf{x}_{k+1} &= (\mathbf{I} - \mathbf{H}_c\mathbf{K}_{l,k}\Xi_k)\boldsymbol{\Phi}\mathbf{x}_k + (\mathbf{I} - \mathbf{H}_c\mathbf{K}_{l,k}\Xi_k)\mathbf{H}_v\mathbf{v}_k \\ &+ (\mathbf{I} - \mathbf{H}_c\mathbf{K}_{l,k}\Xi_k)\mathbf{H}_c\mathbf{K}_{s,k}\boldsymbol{\tau}_{s,k} + (\mathbf{I} - \mathbf{H}_c\mathbf{K}_{l,k}\Xi_k)(\boldsymbol{\Phi}_j - \mathbf{I})\frac{\xi_{const,j}}{c_j}\end{aligned} \tag{101}$$

$$\delta\mathbf{x}_{k,j}(T_d) = \frac{1}{T_d}((\boldsymbol{\Phi}_j - \mathbf{I})\mathbf{x}_k + \mathbf{H}_v\mathbf{v}_k + \mathbf{H}_C\boldsymbol{\tau}_{C,k} + (\boldsymbol{\Phi}_j - \mathbf{I})\frac{\xi_{const,j}}{c_j}) \tag{102}$$

$$\eta_l \approx -\frac{(\mathbf{B}_C^T \mathbf{M} \mathbf{x}_k)_l}{\left(\mathbf{B}_C^T \mathbf{M} \Delta \delta \mathbf{x}_{k,j}(T_d)\right)_l} \tag{103}$$

Equations (100) to (103) can now be used combined with the general system equations of chapter 3.2.2.3. Additionally a second $\eta_{SprRegCross}$ hast to be calculated if the step size is also adjusted if the nonlinear spring is about to cross into another affine region, as shown in chapter 3.2.4.

A question may arise, if the implemented matrix exponential method that is used associated with the step size adjustment on a spring stiffness region crossing works properly. The answer is, yes, if an algorithm is implemented which always adjusts the step size due to the smallest calculated η. If the step size is adjusted due to a zero crossing event in slip speed, all proven methods take place. If the step size adjustment is due to a stiffness region crossing the next step can be calculated by either method, there won't be a slip speed zero crossing in between.

For implementing an approach out of chapter 3.2.1, the whole theory of [3] must be reworked again, because the calculation of $\mathbf{x}_{k+1}$ differs essentially and this is not part of this diploma thesis.

For implementing all other approaches of chapter 3.2.2, the analog procedure of this chapter can be applied.

When it comes to simulation time requirements, the method of the coordinate transformation offers the shortest execution time, particularly for the chosen simulation case seen in "Table 7: Computation time for different algorithms".

3.3.2 PARTICULAR INTEGRATION EXAMPLE

As shown in the previous chapters the "Switching Affine Systems" approach offers outstanding results and is consequently the method of choice for being integrated into the drivetrain topology for the drivetrain type "Future Hybrid Extended" (Figure 44). This integration procedure focuses on the basics of implementation, thus the focus lies on the nonlinear spring with two stiffness regions. Neither step size adjustment for the approximation improvement, nor the parallel clutch, that would act as a nonlinear damper are of interest here.

The nonlinear spring comprises two affine regions within the angle $\delta\varphi_z$. The first region $k_{2,reg1}$ shows a relative low stiffness value, whereas the second region $k_{2,reg2}$ shows a relative high stiffness value ($k_{2,reg2} = 25\ k_{2,reg1}$), to achieve simulation results where the nonlinearity is clearly visible.

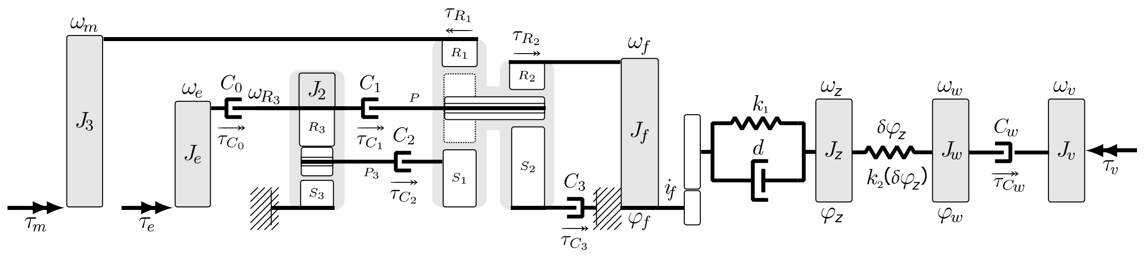

Figure 44: Drivetrain topology for "Future Hybrid Extended (nonlinear spring)"

The system state vector is extended as seen in (104), compared to the drive train topology seen in chapter 2. Added parameters are set to the values according to Table 11.

$$\begin{gathered} \mathbf{x} = [\omega_E \quad \omega_{R_3} \quad \omega_M \quad \omega_F \quad \omega_z \quad \omega_w \quad \omega_v \quad i_F^{-1}\varphi_F - \varphi_z \quad \delta\varphi_z] \\ \boldsymbol{\tau}_c = [\tau_{C0} \quad \tau_{C1} \quad \tau_{C2} \quad \tau_{C3} \quad \tau_{cw}] \end{gathered} \tag{104}$$

For the integration of the nonlinear spring without step size adjustment basically two things have to be considered in the MATLAB®/Simulink® model. First, two sets of system parameters have to be computed, based on the different spring stiffness values and integrated into the model, by means of a switching logic based on $\delta\varphi_z$. Secondly, an offset for the state vector has to be computed, to make the region transition continuous. Figure 45 and Figure 46 show the MATLAB®/Simulink® model of "Future Hybrid" example with the integrated switching affine system logic for integrating a nonlinear spring function.

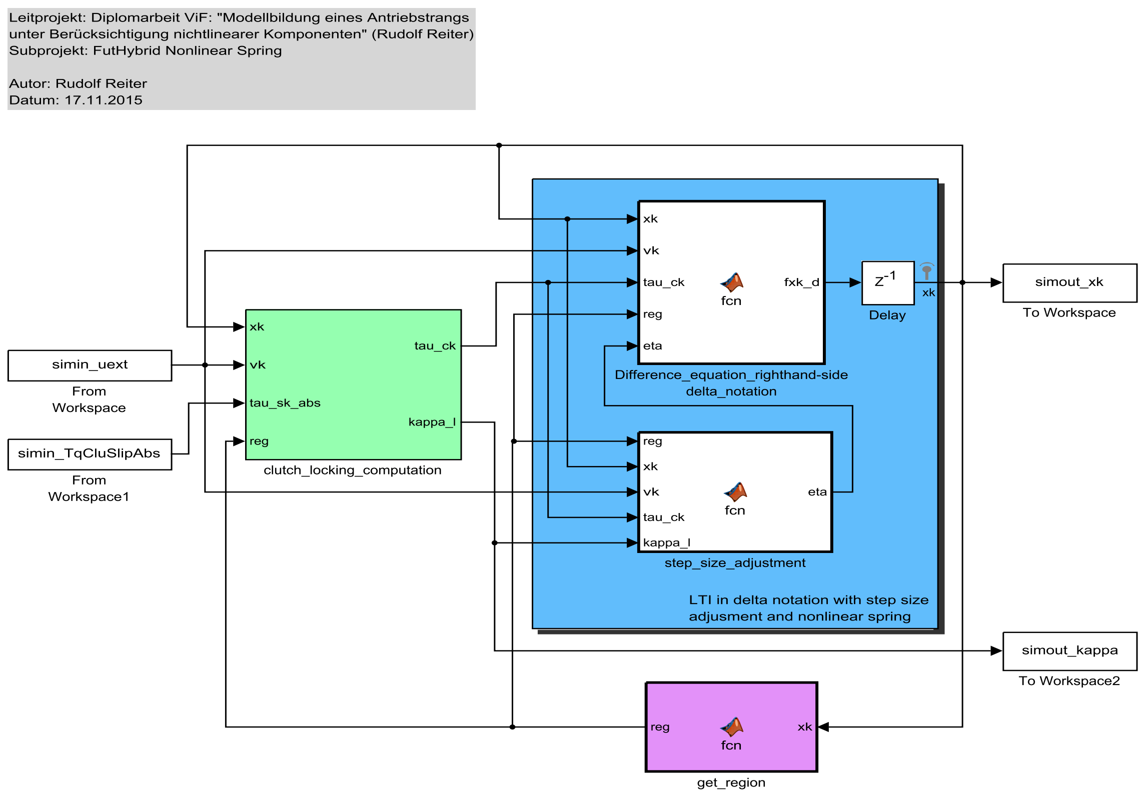
Leitprojekt: Diplomarbeit ViF: "Modellbildung eines Antriebstrangs unter Berücksichtigung nichtlinearer Komponenten" (Rudolf Reiter)
Subprojekt: FutHybrid Nonlinear Spring
Autor: Rudolf Reiter
Datum: 17.11.2015
simin_uext
From Workspace
simin_TqCluSlipAbs
From Workspace1
xk
vk
tau_sk_abs
reg
tau_ck
kappa_l
clutch_locking_computation
xk
vk
tau_ck
reg
eta
fcn
fxk_d
Difference_equation_righthand-side delta_notation
z^-1
xk
Delay
reg
xk
vk
tau_ck
kappa_l
fcn
eta
step_size_adjustment
LTI in delta notation with step size adjusment and nonlinear spring
simout_xk
To Workspace
simout_kappa
To Workspace2
reg
fcn
xk
get_region

Figure 45: MATLAB®/Simulink® model of hybrid drivetrain model with nonlinear spring and clutch locking computation. Top level.

Three test cases have been created to demonstrate the proper function of the algorithm. The first and second test cases display a nonlinear spring that is soft around zero deflection and gets stiff after the region switching angle $\delta\varphi_{z,sec}$. The third test case shows a spring that is stiff first, and gets soft after the switching angle.

Table 11: System parameters for extended drivetrain topology "nonlinear spring"

J_z	J_w	J_v	T_d
$[kg\ m^2]$	$[kg\ m^2]$	$[kg\ m^2]$	$[s]$
3.4093	1.5	135	0.1

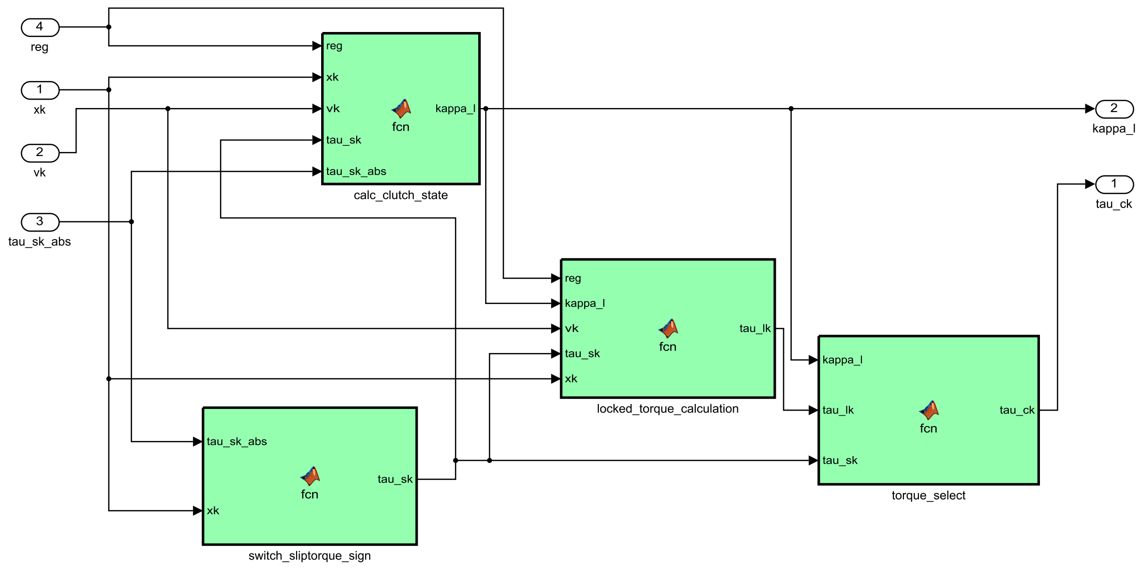

Figure 46: MATLAB®/Simulink® model of hybrid drivetrain model with nonlinear spring and clutch locking computation. Subsystem level.

Table 12: System parameters for test cases "nonlinear spring"

$testcase$	$\delta\varphi_{z,sec}$	$k_{2,reg1}$	$k_{2,reg2}$
	$[10^{-3}\text{rad}]$	$\left[\frac{N}{rad}\right]$	$\left[\frac{N}{rad}\right]$
1	6	800	20000
2	7	800	20000
3	0.25	20000	800

Input torques are applied according to Figure 47. Figure 48, Figure 49 and Figure 50 show the general system behavior, which is not drastically changed by altering the spring stiffness value for the chosen configuration.

The nonlinear effects can be seen in Figure 51, Figure 52 and Figure 53, which display the test cases of Table 12. The stiffer the second region is chosen, the more it resembles a collision, which is investigated in the consecutive chapter. Figure 52 shows an interaction of several inertias around the first entry into the stiff region.

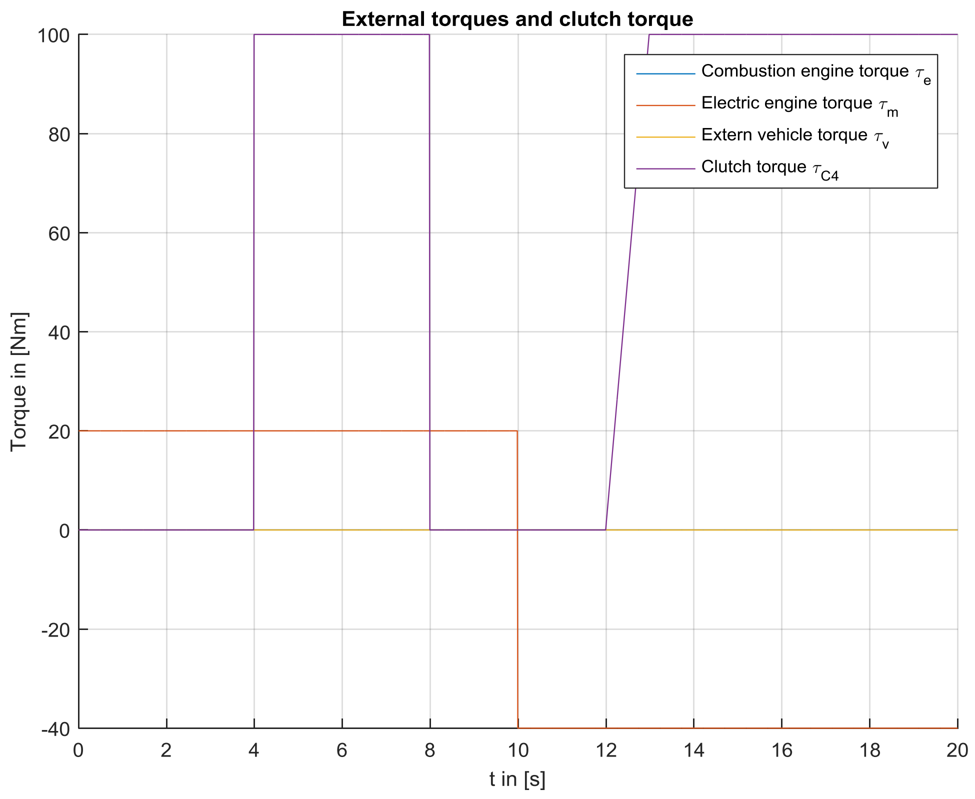

Figure 47: Simulation "Future Hybrid ext. nonlinear spring": input moments

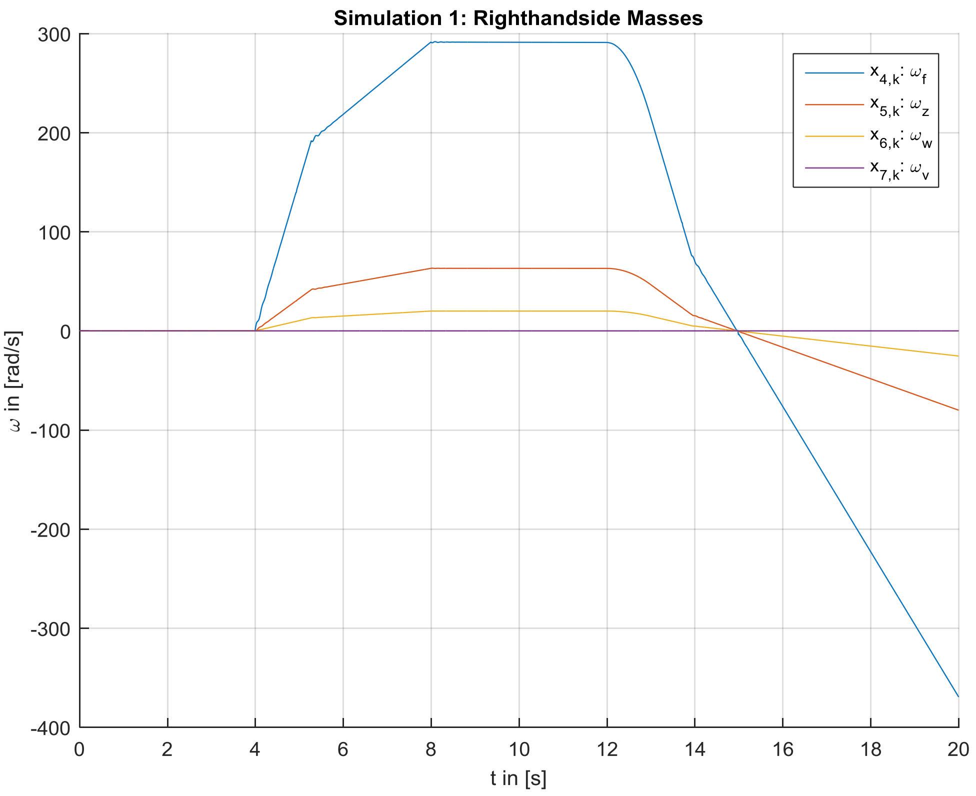

Figure 48: Simulation "Future Hybrid ext. nonlinear spring": revolution speeds 1

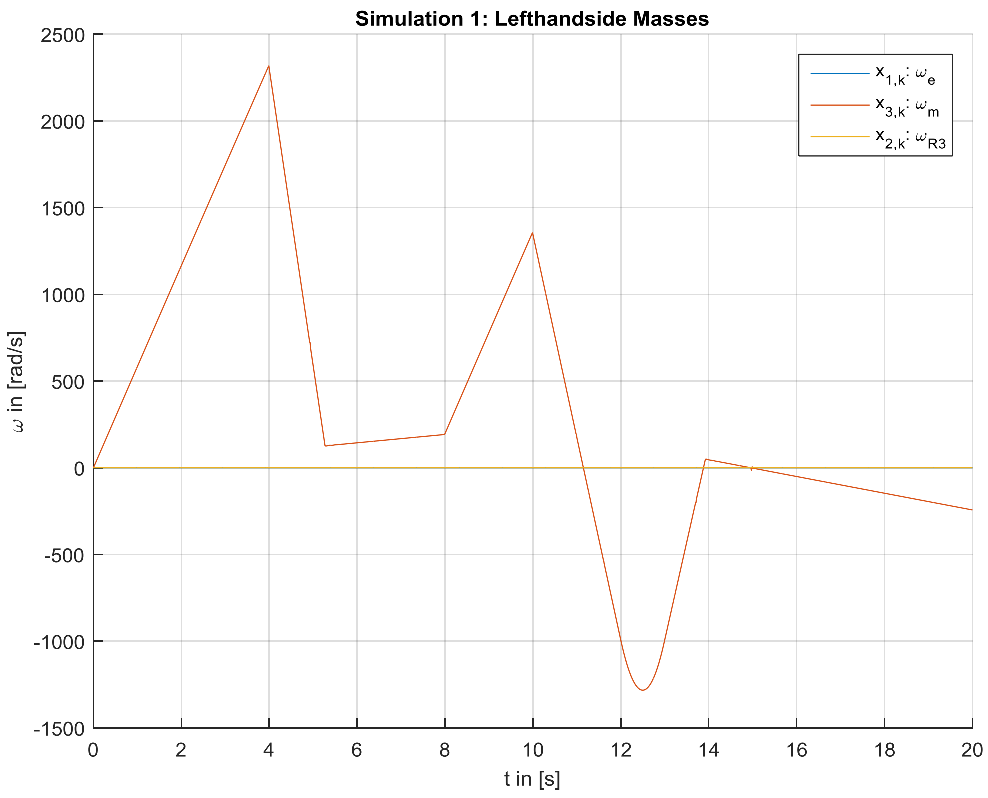

Figure 49: Simulation "Future Hybrid ext. nonlinear spring": revolution speeds 2

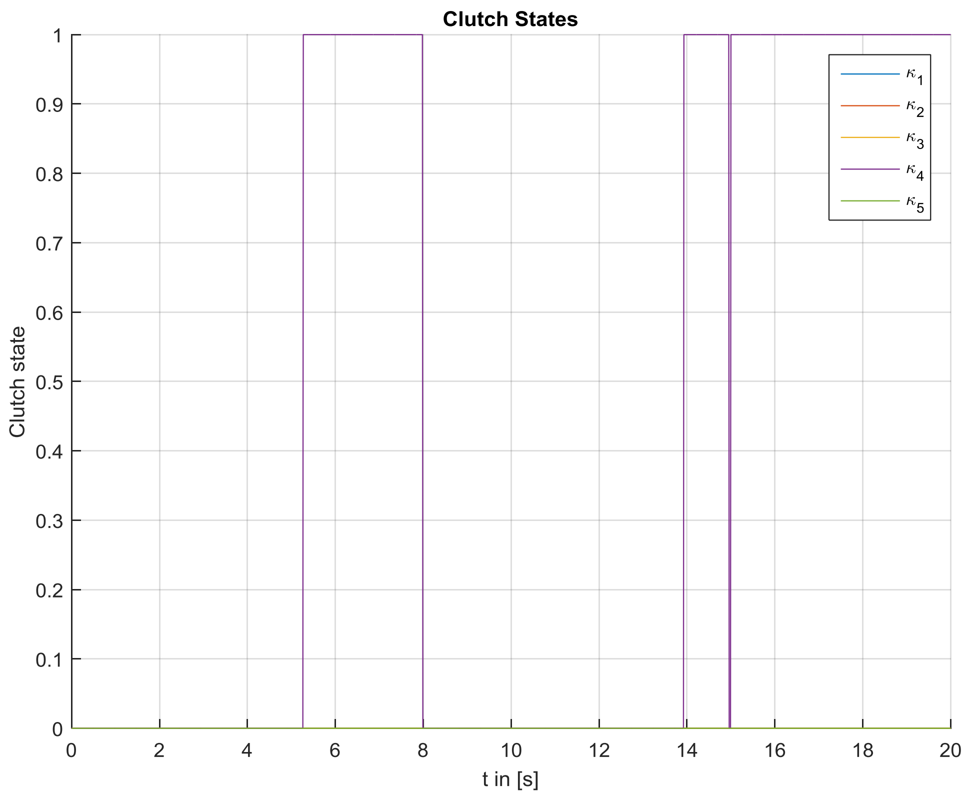

Figure 50: Simulation "Future Hybrid ext. nonlinear spring": clutch states

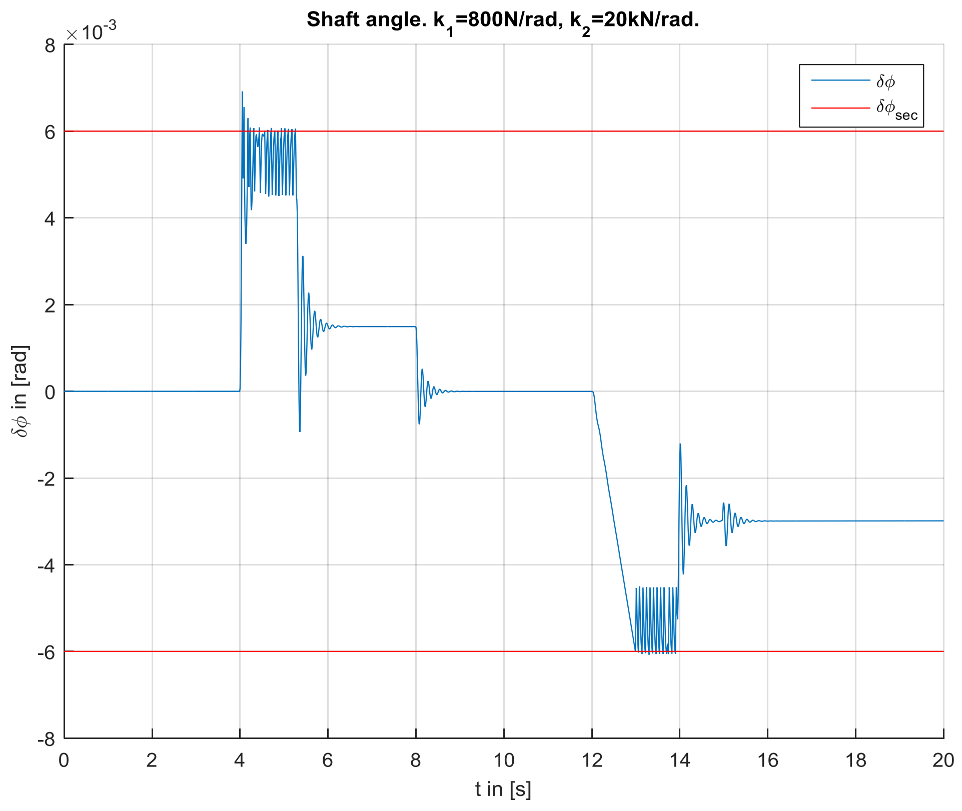

Figure 51: Simulation "Future Hybrid ext. nonlinear spring": shaft angle, test case 1

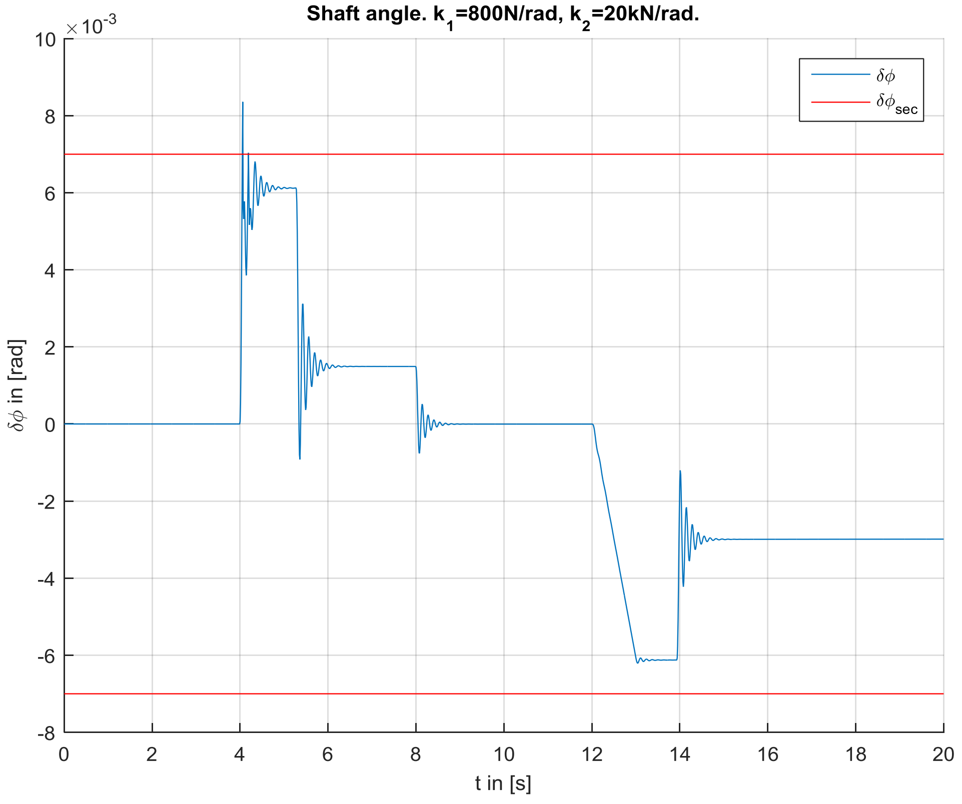

Figure 52: Simulation "Future Hybrid ext. nonlinear spring": shaft angle, test case 2

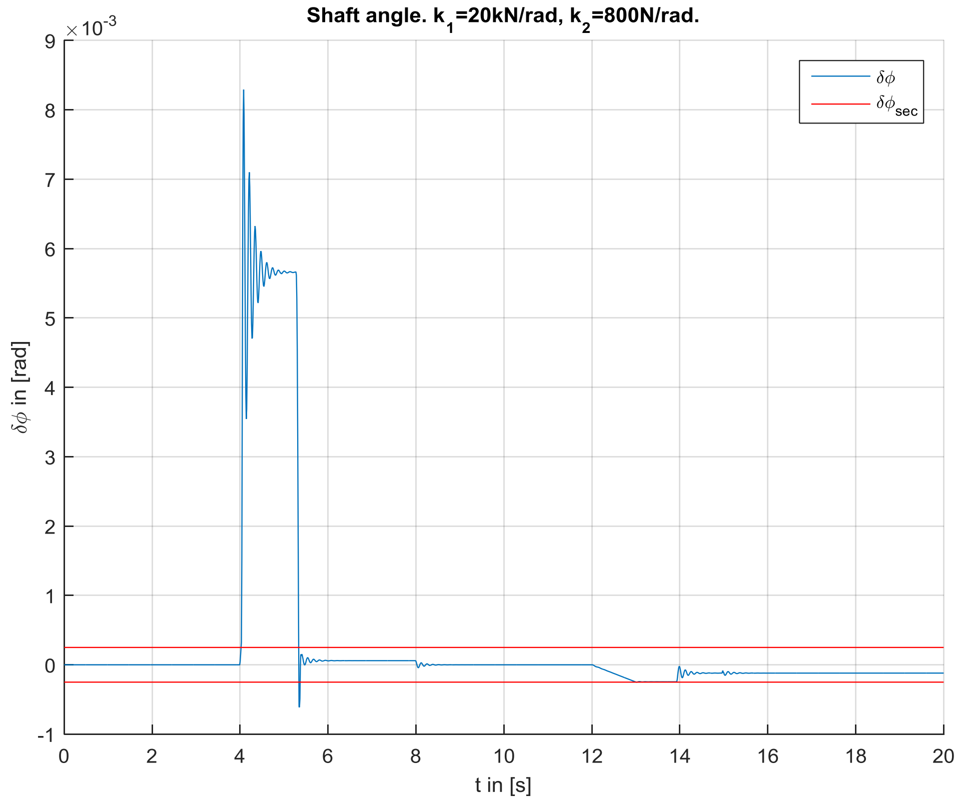

Figure 53: Simulation "Future Hybrid ext. nonlinear spring": shaft angle, test case 3

4 Gearwheel clearance

Mechanical systems in drivetrain topologies often contain gearwheels with a certain clearance [13]. Especially if one clearance is outstanding higher than others, it could be necessary to set up a nonlinear model. Depending on the dimensions of the clearance and the configuration of the gearwheel system, the clearance brings in nonlinear behavior.

The problem resembles the locking and sliding clutch problem of chapter 2 in its basics. As long as two gearwheels are rigidly connected to each other, the system acts as a system with a single degree of freedom and fixed gear ratio set up. As soon as the rotation speeds diverge of each other the two shafts move free of each other as two autonomous systems with two degrees of freedom.

If the two gearwheels are in an independent state and collide with each other, a collision takes place that is dependent on the properties of the material and which can be described by a coefficient of restitution. To simplify the problem, in this chapter two masses are used with a translational movement instead of a rotational one.

Basically a collision is an event which usually happens in an extremely short time period compared with the sample time and also can take place at any time between two sample steps. Therefore a step size adjustment is used similar to chapter 2 to set a sample point to the exact time of collision, afterwards the collision is computed using basic physical laws and material coefficients and finally the step size is extended to keep up with the real time requirement.

In the end of this chapter, an outlook is given to show the possibility of an extension of this solution to a system with an arbitrary number of masses, combined with the restriction of just one collision of two masses at one time and the problems of systems where several masses collide with each other in one sample period.

4.1 Fundamentals of collision computation

The following fundamental equations of mechanical engineering offer an approximation of calculating system trajectories after a collision [8]. Since a collision physically comprises a deformation and a rearrangement of material structures and consecutive energy loss an approximation focuses on the state before (speed values v_1, v_2) and after (speed values v_1', v_2') the collision and describes the energy loss by a coefficient of tion c_r, which can be obtained experimentally, seen in equation (105).

$$c_r = \frac{v_1' - v_2'}{v_2 - v_1} \tag{105}$$

The value range stretches from zero to one, where zero describes a fully plastic collision and one describes a fully elastic collision.

The energy loss ΔE is a function of the initial speeds and the coefficient of restitution, seen in equation (106).

$$\Delta E = \frac{m_1 m_2}{2(m_1 + m_2)}(v_1 - v_2)^2(1 - c_{rest}{}^2) \tag{106}$$

The speeds after the collision can be obtained by means of equation (107) and (108).

$$v_1' = \frac{m_1 v_1 + m_2 v_2 - m_2(v_1 - v_2)c_r}{m_1 + m_2} \tag{107}$$

$$v_2' = \frac{m_1 v_1 + m_2 v_2 - m_1(v_2 - v_1)c_r}{m_1 + m_2} \tag{108}$$

4.2 TEST SYSTEM CONFIGURATION

As a basis a system configuration with two spring-damper-mass systems is used, with two different idle positions $\mathbf{x}_{R,S1}$ for system 1 and $\mathbf{x}_{R,S2}$ for system 2 and with an initial state space vector $\mathbf{x}_0 = \begin{bmatrix} \mathbf{x}_{0,S1} \\ \mathbf{x}_{0,S2} \end{bmatrix}$ for both systems. The masses $m_1$and m_2 can be deflected by separate forces $F_{ext,1}, F_{ext,2}$ and are configured with two separate spring-damper systems with the stiffness coefficients c_1, c_2 and the damping coefficients d_1, d_2. Figure 54 shows the configuration in the independent moving state. In the simulation the masses are considered as punctual.

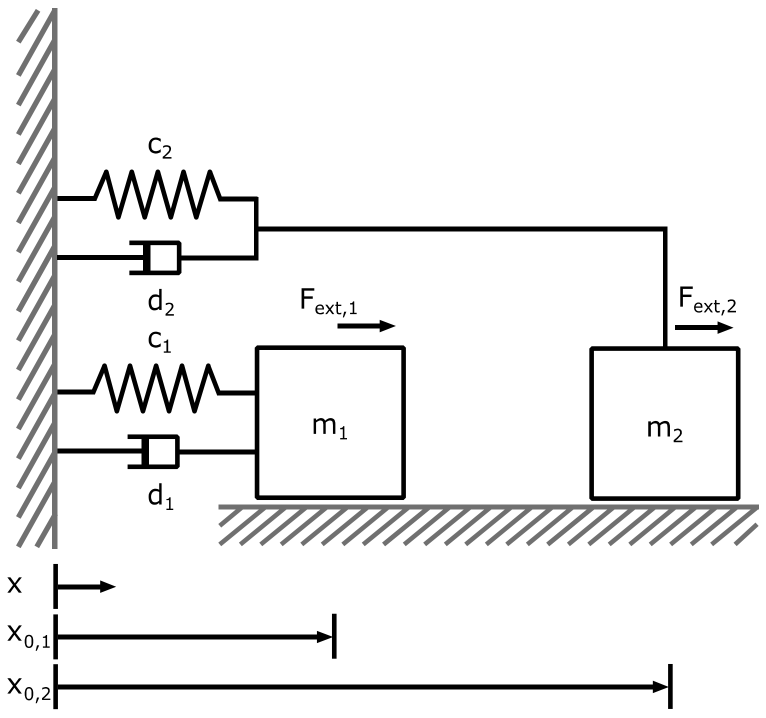

Figure 54: Two mass system configuration for collision simulation

The continuous time system can be described by the following comprising system description, seen in equation (109), which formally combines the two autonomous systems.

$$\begin{aligned}\dot{\mathbf{x}}_{\mathrm{sys1}}(\mathrm{t}) &= \begin{bmatrix} 0 & 1 \\ -\frac{c_1}{m_1} & -\frac{d_1}{m_1} \end{bmatrix} \mathbf{x}(\mathrm{t}) + \begin{bmatrix} 0 \\ \frac{1}{m_1} \end{bmatrix} \mathrm{F}_{\mathrm{ext,1}}(\mathrm{t}) + \begin{bmatrix} 0 \\ \frac{c_1}{m_1} x_{\mathrm{R,S1}} \end{bmatrix} \\ &= \mathbf{A}_{\mathrm{S1}} \mathbf{x}_{\mathrm{S1}}(\mathrm{t}) + \mathbf{b}_{\mathrm{S1}} \mathrm{F}_{\mathrm{ext,1}}(\mathrm{t}) + \mathbf{x}_{\mathrm{S1,off}}\end{aligned} \tag{109}$$

$$\begin{aligned}\dot{\mathbf{x}}_{\mathrm{sys2}}(\mathrm{t}) &= \begin{bmatrix} 0 & 1 \\ -\frac{c_2}{m_2} & -\frac{d_2}{m_2} \end{bmatrix} \mathbf{x}(\mathrm{t}) + \begin{bmatrix} 0 \\ \frac{1}{m_2} \end{bmatrix} \mathrm{F}_{\mathrm{ext,2}}(\mathrm{t}) + \begin{bmatrix} 0 \\ \frac{c_2}{m_2} x_{\mathrm{R,S2}} \end{bmatrix} \\ &= \mathbf{A}_{\mathrm{S2}} \mathbf{x}_{\mathrm{S2}}(\mathrm{t}) + \mathbf{b}_{\mathrm{S2}} \mathrm{F}_{\mathrm{ext,2}}(\mathrm{t}) + \mathbf{x}_{\mathrm{S2,off}}\end{aligned} \tag{110}$$

$$\begin{aligned}\dot{\mathbf{x}}(\mathrm{t}) &= \begin{bmatrix} \mathbf{A}_{\mathrm{S1}} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{\mathrm{S2}} \end{bmatrix} \mathbf{x}(\mathrm{t}) + \begin{bmatrix} \mathbf{b}_{\mathrm{S1}} & \mathbf{0} \\ \mathbf{0} & \mathbf{b}_{\mathrm{S2}} \end{bmatrix} \begin{bmatrix} \mathrm{F}_{\mathrm{ext,1}}(\mathrm{t}) \\ \mathrm{F}_{\mathrm{ext,2}}(\mathrm{t}) \end{bmatrix} + \begin{bmatrix} \mathbf{x}_{\mathrm{S1,off}} \\ \mathbf{x}_{\mathrm{S2,off}} \end{bmatrix} \\ &= \mathbf{A}\mathbf{x}(\mathrm{t}) + \mathbf{B}\mathbf{F}_{\mathrm{ext}}(\mathrm{t}) + \mathbf{x}_{\mathrm{off}}\end{aligned} \tag{111}$$

The transformation in the discrete time domain and the sampling of the input signals leads to equation (112), which is used subsequently in this chapter. The constant offset variable for the equilibrium position $\mathbf{x}_{\mathrm{off}}$ leads to the discrete time representation $\bar{\mathbf{x}}_{\mathrm{off}}$.

$$\begin{aligned}\mathbf{x}_{k+1} &= \begin{bmatrix}\mathbf{\Phi}_{S1} & \mathbf{0}\\ \mathbf{0} & \mathbf{\Phi}_{S2}\end{bmatrix}\mathbf{x}_k + \begin{bmatrix}\mathbf{h}_{S1} & \mathbf{0}\\ \mathbf{0} & \mathbf{h}_{S2}\end{bmatrix}\begin{bmatrix}F_{ext,1,k}\\ F_{ext,2,k}\end{bmatrix} + \begin{bmatrix}\bar{\mathbf{x}}_{S1,off}\\ \bar{\mathbf{x}}_{S2,off}\end{bmatrix}\\ &= \mathbf{\Phi}\mathbf{x}_k + \mathbf{H}\,\mathbf{F}_{ext,k} + \bar{\mathbf{x}}_{off}\end{aligned} \tag{112}$$

4.3 ALGORITHM IMPLEMENTATION

Basically the algorithm can be split into four parts. At first the step size is adjusted by an approximation similar to chapter 2.2, to achieve an appropriate approximation of the initial values (see APPENDIX A for an investigation of the error made by the used step size adjustment approach). Secondly the shock is calculated by basic mechanical laws. Afterwards the step size is extended to reach the real time grid and finally a decomposing force is added, which ensures the masses do not overlap.

The algorithm computation of both, the first part and the second part, are realized by means of a predicting step, which adds forces to achieve a previously calculated result.

4.3.1 FIRST PART: STEP SIZE ADJUSTMENT AND COLLISION COMPUTATION

The collision is implemented by a fictive force and velocity input which is added at the time step k. Therefore a prediction for the following step $k + \eta$ right after the collision has to be calculated in advance and the force and velocity input is computed backwards and added to the linear system in the time step k.

Since the collision event generally takes place in between two sample points an approximation of the state vector at the time of collision proves to be beneficial. Therefore we use the approximation seen in [3], which linearly computes the state vector in between two sample points of a discrete time system at the time $t = kT_d + \eta$, seen in equation (115). Figure 55 shows the discrete system state without step size adjustment at time step (k+1).

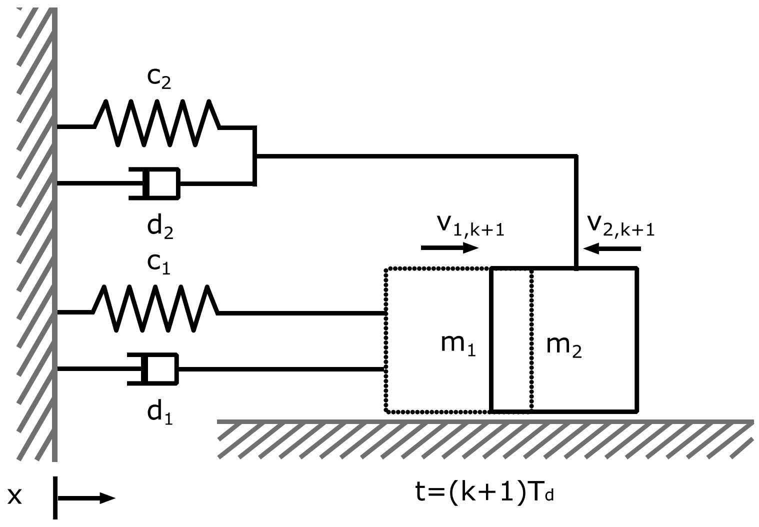

Figure 55: Mass system: Consecutive position without adjustment

$$\delta\mathbf{x}_k(T_d) = \frac{1}{T_d}((\mathbf{\Phi}(T_d) - \mathbf{I})\mathbf{x}_k + \mathbf{H}\,\mathbf{F}_{ext,k} + \bar{\mathbf{x}}_{off}) \tag{113}$$

$$\mathbf{x}_{k+\eta} = \mathbf{x}_k + \eta T_d \delta\mathbf{x}_k(\eta T_d) \tag{114}$$

$$\mathbf{x}_{k+\eta} \approx \mathbf{x}_k + \eta T_d \delta\mathbf{x}_k(T_d) \tag{115}$$

The step size adjustment is aiming to adjust the step to the exact point, where the two masses collide with each other. Therefore a difference $\Delta\mathbf{x}_{pred.k+1}$ of the position variables is predicted for the consecutive step, and if this position is negative the step size is adjusted by η according to equation (115) and (119). Consequently the step size variation will lead to $\Delta x_{k+\eta} = \mathbf{c}_\delta^T \mathbf{x}_{k+\eta} = 0$.

$$\mathbf{c}_\delta^T = [-1\ 0\ 1\ 0] \tag{116}$$

The velocity variables $v_{1,k+\eta}$ and $v_{2,k+\eta}$ of the state vector $\mathbf{x}_{k+\eta}$ serve as the input velocities for the collision calculation, which can be found in chapter 4.1

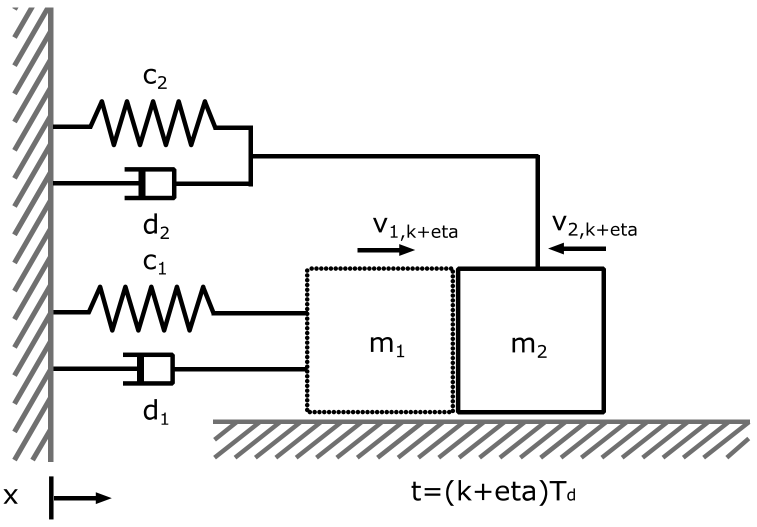

Figure 56: Mass system: Consecutive position with adjustment

After the collision calculation the velocities $v'_{1,k+\eta}$ and $v'_{2,k+\eta}$ result and the positions have not been changed by implication. This results in a desired vector $\mathbf{x}_{k+\eta}{}'$ (notation (120)) at the time step $k+\eta$ and previously computed input vector $\boldsymbol{\tau}_{c,k}$ at the time step k. Figure 56 shows the state of the mass system at the varied step $k+\eta$.

$$\Delta x_{pred.k+1} = \mathbf{c}_\delta^T \mathbf{x}_{k+1} \tag{117}$$

$$\Delta x_k = \mathbf{c}_\delta^T \mathbf{x}_k \tag{118}$$

$$\eta = \frac{\Delta x_k}{\Delta x_k - \Delta x_{pred,k+1}} \tag{119}$$

$$\mathbf{x}'_{k+\eta} = \begin{bmatrix} x_{1,k+\eta} \\ x'_{2,k+\eta} \\ x_{3,k+\eta} \\ x'_{4,k+\eta} \end{bmatrix} \text{ with } x'_{2,k+\eta} = v'_{1,k+\eta} \text{ and } x'_{4,k+\eta} = v'_{2,k+\eta} \tag{120}$$

To achieve all requirements $\mathbf{x}'_{k+\eta}$ for the time step k+η, the input vector $\boldsymbol{\tau}_{c,k}$ needs to represent a vector of the size four. For implementing the discrete time vector $\boldsymbol{\tau}_{c,k}$, first a continuous time input vector $\boldsymbol{\tau}_c(t)$ is introduced for equation (112) and leads to the continuous time equation (122), where the 4x4 input matrix $\mathbf{B}_C$ is set to

$$\mathbf{B}_C = diag\left(\begin{bmatrix} 1 & \frac{1}{m_1} & 1 & \frac{1}{m_2} \end{bmatrix}\right). \tag{121}$$

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{F}_{ext}(t) + \bar{\mathbf{x}}_{off} + \mathbf{B}_C\boldsymbol{\tau}_c(t) \tag{122}$$

Values of the input matrix $\mathbf{B}_\mathrm{C}$ are chosen to achieve full order and to interpret the second and fourth entry of the input vector $\boldsymbol{\tau}_\mathrm{c}(\mathrm{t})$ as forces acting on the system. Equation (112) is extended by the discrete time collision input vector $\boldsymbol{\tau}_\mathrm{c,k}$, the discrete time input matrix $\mathbf{H}_\mathrm{c}$, that is derived from $\mathbf{B}_\mathrm{C}$ and the step size adjustment η and consequently leads to equation (123) and can be used for the existing algorithms with a varying η.

$$\mathbf{x}_{\mathrm{k}+\eta} = \mathbf{x}_\mathrm{k} + \eta \mathrm{T}_\mathrm{d}(\frac{1}{\mathrm{T}_\mathrm{d}}\left((\boldsymbol{\Phi} - \mathbf{I})\mathbf{x}_\mathrm{k} + \mathbf{H}\mathbf{F}_{\mathrm{ext,k}} + \bar{\mathbf{x}}_{\mathrm{off}} + \mathbf{H}_\mathrm{c}\boldsymbol{\tau}_{\mathrm{c,k}}\right)) \tag{123}$$

With the condition $\mathbf{x}_{\mathrm{k}+\eta} \overset{!}{=} \mathbf{x}'_{\mathrm{k}+\eta}$ the fictive input vector $\boldsymbol{\tau}_\mathrm{c,k}$ can be computed as follows in equation (124).

$$\boldsymbol{\tau}_{\mathrm{c,k}} = \mathbf{H}_\mathrm{c}^{-1}(\frac{1}{\eta}\left(\mathbf{x}'_{\mathrm{k}+\eta} - \mathbf{x}_\mathrm{k}\right) - (\boldsymbol{\Phi} - \mathbf{I})\mathbf{x}_\mathrm{k} - \mathbf{H}\,\mathbf{F}_{\mathrm{ext,k}} - \bar{\mathbf{x}}_{\mathrm{off}}) \tag{124}$$

It is remarkable, that the first and the third entry of the collision input vector $\boldsymbol{\tau}_\mathrm{c}(\mathrm{t})$ and $\boldsymbol{\tau}_\mathrm{c,k}$ respectively represent input signals to the differential equations for the derivations of the deflection $\dot{\mathrm{x}}_1$ and $\dot{\mathrm{x}}_3$, which can be considered as differential equations for the speed. Input signals to these equations can be interpreted as speed shifts, but not as forces.

4.3.2 SECOND PART: STEP SIZE EXTENSION AND DECOMPOSING FORCE

To keep up with the desired fixed step size, the step size has to be extended with regards to the previous step. Therefore we use the step size extension $\eta_\mathrm{k} = \mathrm{T}_\mathrm{d}(2 - \eta_{\mathrm{k}-1})$ in the next step, where $\eta_{\mathrm{k}-1}$ is presumed to be the step size variation of the collision step before.

Preliminarily the following situation should raise awareness for the outstanding problem. Following the first part of the algorithm, both masses start at the same position with a $\Delta\mathrm{x}_\mathrm{k} = 0$ and two velocities, which possibly are equal to each other or the masses drift apart of each other (Figure 57). Following the collision law of chapter 4.1, it is initially impossible that the velocities lead to an overlapping in the first place.

In reality it is now possible, that the two masses collide with each other in very short time periods, which can be referred to as “rattling” especially in “stiff” systems or they might move on together in an fully plastic collision. In both cases deformation and restitution takes place which might transform energy into heat and keeps the masses apart or sticks them together.

Obviously this physical behavior can hardly be implemented in a discrete time system with a fixed sample period. If in a misleading manner the velocities after the collision are used for the extended consecutive step as initial values and the system equations are applied straight forward, overlapping is possible in the next step, which leads to a fatally wrong physical situation (Figure 58).

As a solution for this problem another fictive separation force $\boldsymbol{\tau}_{\mathrm{s,k}}$ is inserted before the extended step, which aims to keep the masses apart of overlapping (Figure 59). Notably, the vector $\boldsymbol{\tau}_{\mathrm{s,k}}$ is named as force vector, as opposed to the collision input vector $\boldsymbol{\tau}_{\mathrm{c,k}}$, because the separation force vector $\boldsymbol{\tau}_{\mathrm{s,k}}$ contains only input signals, that act on differential equations for the derivation of a speed (i.e.: $\dot{x}_2, \dot{x}_4$). In equation (125) and equation (128) that relation becomes obvious by the zero entries for every equation where an input single does not serve as a force.

If no energy is lost, the force on the active system must be equal to the reactive system. **If energy is lost** through rattling occurrence or deformation processes the force on the acting system (decelerating force) might be higher than the force on the reactive system, because the force on the active system takes energy away by reducing its speed and the force on the pushed system accelerates it, thus adding energy. The relation of the surpassing force on the active system to the force of the reactive system can be adjusted by a design parameter k_s. If no energy should be lost through this process the parameter is equal to one.

$$\boldsymbol{\tau}_{\mathrm{s,k}} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ -1 \end{bmatrix} \tau_{\mathrm{s,k}} = \mathbf{c}_{\mathrm{s}} \tau_{\mathrm{s,k}} \tag{125}$$

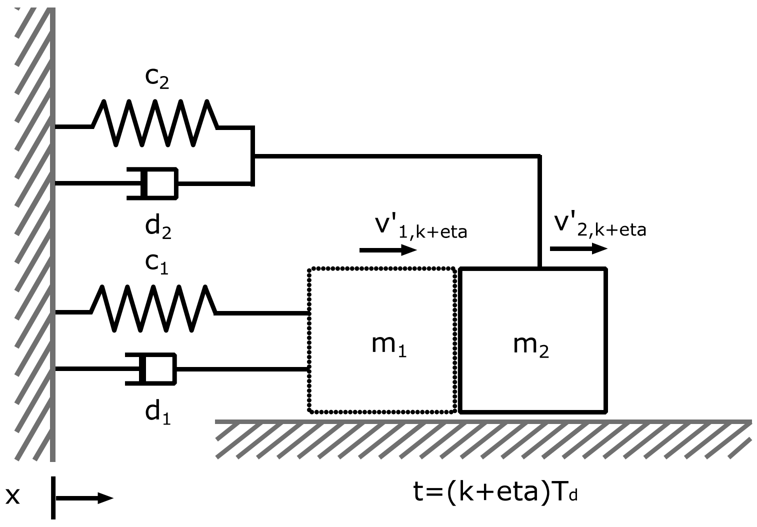

Figure 57: Mass system: State after collision calculation

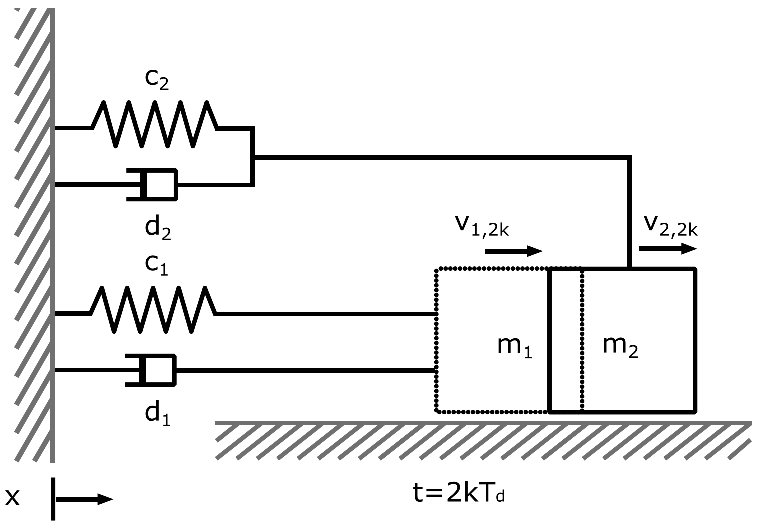

Figure 58: Mass system: Overlapping position after collision

In the algorithm a flag parameter κ displays, if the two masses stick together or rattle in the consecutive step. Another parameter $\mathrm{force\_dir}$ characterizes, in which direction the two masses are moving and consequently defines the associated pushing and breaking mass. If the two masses are moving from left to right in a one-dimensional horizontal plane, the left mass is specified as pushing system and sets the parameter $\mathrm{force\_dir} = 1$, and vice versa where $\mathrm{force\_dir} = -1$.

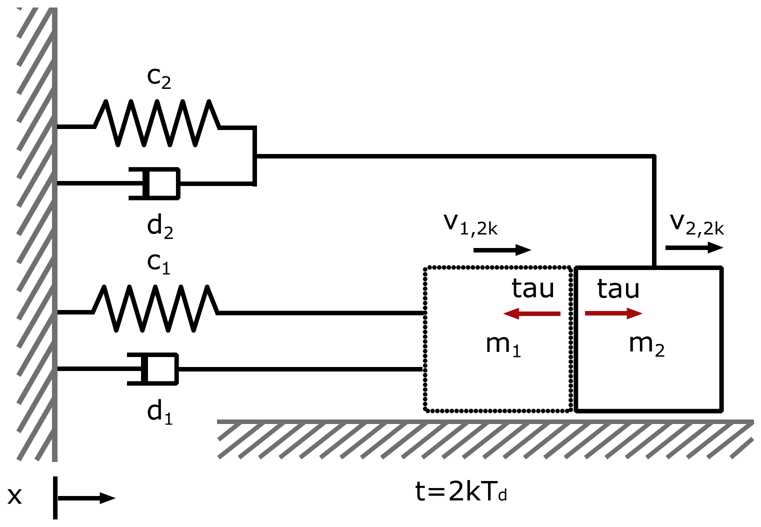

Figure 59: Mass system: Position with separation force

$$\mathrm{v\_dir}_k = \mathrm{sign}\left(\mathrm{sign}(v_{1,k}) + \mathrm{sign}(v_{2,k})\right)$$
$$\mathrm{E\_dir}_k = \mathrm{sign}\left(v_{1,k}^2 m_1 - v_{2,k}^2 m_2\right) \tag{126}$$

$$\mathrm{force\_dir}_k = \begin{cases} \mathrm{v\_dir}_k & \text{if } \mathrm{v\_dir}_k \neq 0 \\ \mathrm{E\_dir}_k & \text{else} \end{cases} \tag{127}$$

A situation with $\mathrm{force\_dir} = 0$ occurs, if the outer system forces the masses to redirect their speeds opposite to each other (note that the initial speeds are directed into the same direction or apart of each other after the shock). Further calculation has to determine the system with the higher energy, to lower the total system energy by analogy.

The basic condition for a separating force is a $\Delta x_k = 0$ and a predicted overlapping $\Delta x_{k+1} < 0$. If $\Delta x_k > 0$, a straight forward collision is computed as mentioned before. An approach for the separating force with respect to the assumption of not adding energy to the system and thus adding equal forces can be seen in equation (125), where the same absolute value of the force is added in the opposing direction and for the same time.

If energy should be lost by uneven separating forces, equation (128) takes place.

$$\boldsymbol{\tau}_{s,k} = \begin{cases} \begin{bmatrix} 0 \\ k_s \\ 0 \\ -1 \end{bmatrix} \tau_{s,k} & \text{if force\_dir}_k > 0 \\ \begin{bmatrix} 0 \\ 1 \\ 0 \\ -1 \end{bmatrix} \tau_{s,k} & \text{if force\_dir}_k = 0 \\ \begin{bmatrix} 0 \\ 1 \\ 0 \\ -k_s \end{bmatrix} \tau_{s,k} & \text{if force\_dir}_k < 0 \end{cases} \tag{128}$$

Since the separation force and the collision force can definitely not act at the same sample step equation (129) is used for the collision step, if $\Delta x_{k+1} < 0$ and $\Delta x_k > 0$ and equation (130) is used for following separating step, if $\Delta x_{k+1} < 0$ and $\Delta x_k = 0$. In the separating step the step size factor η might higher than 1.

$$\mathbf{x}_{k+\eta} = \mathbf{x}_k + \eta_k T_d (\frac{1}{T_d}\Big((\boldsymbol{\Phi} - \mathbf{I})\mathbf{x}_k + \mathbf{H}\mathbf{F}_{ext,k} + \bar{\mathbf{x}}_{off} + \mathbf{H}_c\, \boldsymbol{\tau}_{c,k}\Big)) \tag{129}$$

$$\mathbf{x}_{k+\eta} = \mathbf{x}_k + \eta_k T_d (\frac{1}{T_d}\Big((\boldsymbol{\Phi} - \mathbf{I})\mathbf{x}_k + \mathbf{H}\mathbf{F}_{ext,k} + \bar{\mathbf{x}}_{off} + \mathbf{H}_c\, \boldsymbol{\tau}_{s,k}\Big)) \tag{130}$$

If a system involves more than one clearance computation element, the separation and collision force may act simultaneously, but on different parts of the system, although influencing the total system behavior at the same time step. With just one clearance computation element implemented in the system the collision step always precedes the separation steps. The dependencies of guaranteed exclusion of both forces acting at the same time step are shown in (131).

$$\begin{gathered} \text{if } \eta_k \leq 1 \wedge \Delta x_k > 0 \Rightarrow \boldsymbol{\tau}_{s,k} = 0 \\ \text{if } \eta_k \geq 1 \wedge \Delta x_k = 0 \Rightarrow \boldsymbol{\tau}_{c,k} = 0 \end{gathered} \tag{131}$$

The condition for the consecutive step to not overlap can be noted as follows in equation (132).

$$\Delta x_{k+\eta} = \mathbf{c}_\delta^T \mathbf{x}_{k+\eta} = [-1 \quad 0 \quad 1 \quad 0]\mathbf{x}_{k+\eta} \overset{!}{=} 0 \tag{132}$$

It is important to mention that this demand claims no speed constraints for the step $\mathbf{x}_{k+\eta}$. This means a force is requested, which acts on the two masses to separate them and sets them to the identical position, but neglects either speed of the masses, which are naturally rattling also or equal to each other, if the masses “stick” to each other. By just adding “separating forces” to the corresponding difference equations it is impossible to desire both, equal speed and equal position of the masses in the consecutive step. One might come up with the idea of adding another input to the velocity difference equations, also with the precondition of not adding energy into the system. A rudiment approach is discussed later. By now it is sufficient if the speed values are neglected and oscillate around their mean value in the case of sticking or rattling masses, which turns out to work pretty well.

Solving equation (132) by means of equation (130), the separating force can be computed as follows.

$$\tau_{c,k} = \left(\mathbf{c}_\delta^T \mathbf{H}_c \mathbf{c}_s\right)^{-1} \mathbf{c}_\delta^T \left(-\frac{1}{\eta}\mathbf{x}_k - (\mathbf{\Phi} - \mathbf{I})\mathbf{x}_k - \mathbf{H}\mathbf{F}_{ext,k} - \bar{\mathbf{x}}_{off}\right) \tag{133}$$

4.4 Simulation results for the simplified model

4.4.1 Test cases

Several simulation examples aim to visualize the correct physical behavior and to show the proper functionality of the algorithm. Figure 60 shows the MATLAB®/Simulink® model of the basic setup with the essential step size adjustment, the collision momentum calculation and the addition of the separation momentum calculation.

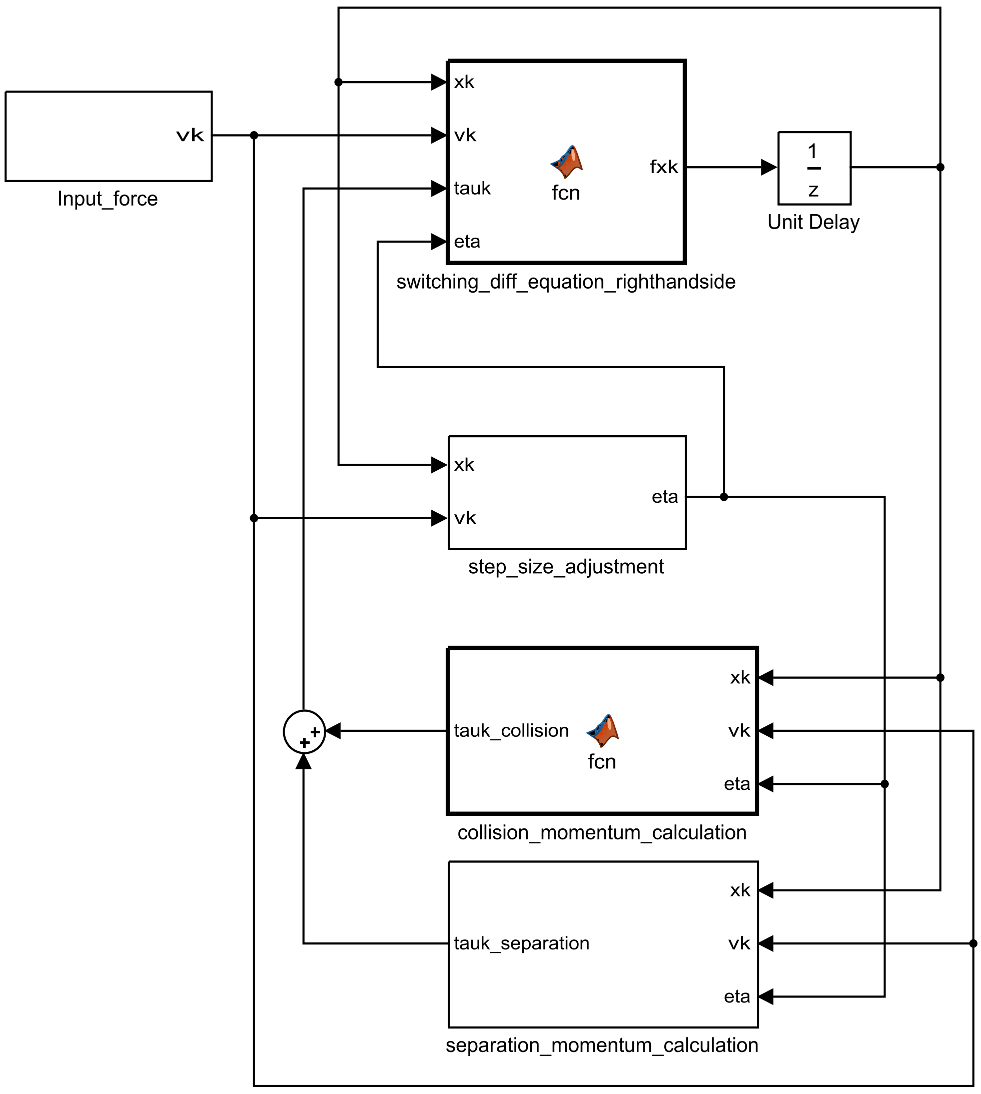

Figure 60: MATLAB®/Simulink® model of the two-mass-system

Table 13: System parameters 1 for test cases

$testcase$	c_{sys1}	c_{sys2}	d_{sys1}	d_{sys2}	m_1	m_2	k_s	c_r
	$\left[\frac{N}{m}\right]$	$\left[\frac{N}{m}\right]$	$\left[\frac{Ns}{m}\right]$	$\left[\frac{Ns}{m}\right]$	$[kg]$	$[kg]$		
1	20	20	0	0	10	10	1	1
2	20	20	0	0	10	10	1	0
3	20	40	0	0	10	10	1	0
4	20	40	0	0	10	10	10	0
5	10	40	0	0	40	10	1	0
6	20	20	1	0	20	100	10	0.9
7	5	5	1	0.1	10	0.1	1	0.1

Table 14: System parameters 2 for test cases

$testcase$	$\boldsymbol{x}_R$	$\boldsymbol{x}(t=0)$	T_d
	$[m]$	$[m]$	$[s]$
1	$[0 \quad 0 \quad 0 \quad 0]$	$[-0.5 \quad 0 \quad 0 \quad 0]$	0.1
2	$[0 \quad 0 \quad 0 \quad 0]$	$[-0.5 \quad 0 \quad 0 \quad 0]$	0.1
3	$[0 \quad 0 \quad 0 \quad 0]$	$[-0.5 \quad 0 \quad 0 \quad 0]$	0.1
4	$[0 \quad 0 \quad 0 \quad 0]$	$[-0.5 \quad 0 \quad 0 \quad 0]$	0.1
5	$[0 \quad 0 \quad 0 \quad 0]$	$[-0.5 \quad 1 \quad 0 \quad 0]$	0.2
6	$[0 \quad 0 \quad 0 \quad 0]$	$[-0.3 \quad 0.1 \quad 0 \quad 0]$	0.1
7	$[0 \quad 0 \quad 0 \quad 0]$	$[-0.3 \quad 0.1 \quad 0 \quad 0]$	0.1

As we can see in test case 1 (Figure 61), two identical spring-damper-mass systems are used, where one of them is deflected initially. The restitution coefficient is set to one, which leads to a fully elastic collision, where one mass transfers its whole kinetic energy to the other mass that consequently leads to a sinusoidal shape of the deflection of the moving system if the systems are identical. The same shape would occur, if there was only one system without collision. It is obvious that the step size is adjusted, if a collision takes place, which adds a little error to the computation, each time the system collides.

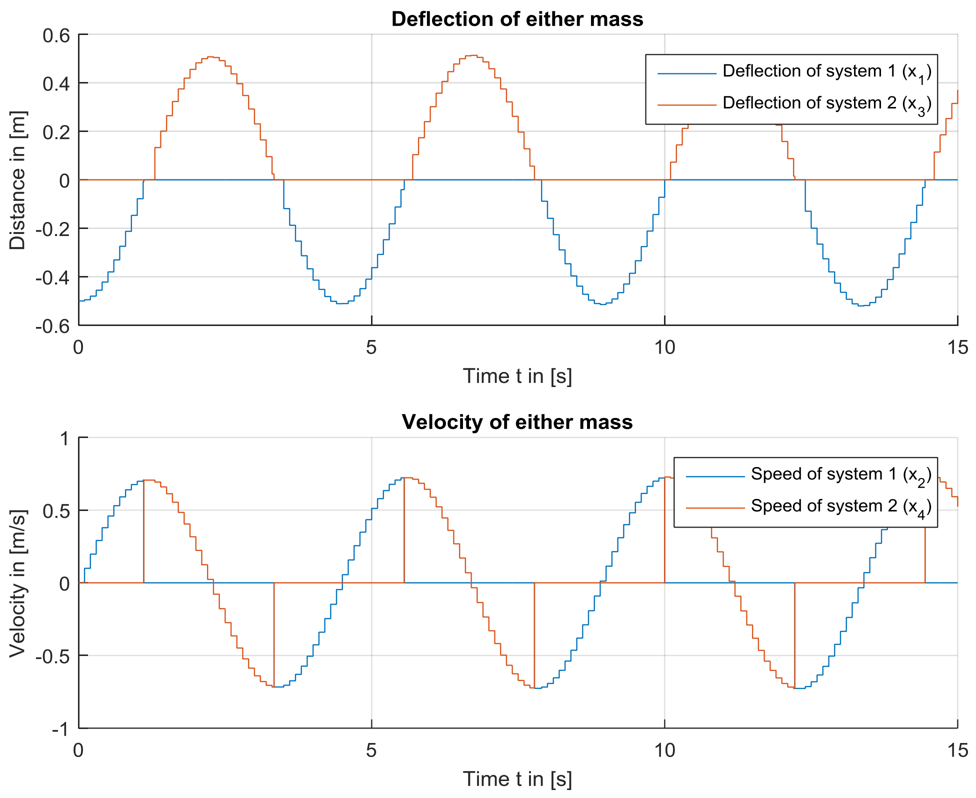

Figure 61: Collision computation. Test case 1.

Test case 2 (Figure 62) shows the same configuration as test case 1 does, but with a restitution factor of 0, which leads to a fully plastic collision. Additionally the system energy has been plotted, which shows the energy loss that is resulting from the collision. It is remarkable that the step size adjustment leads to an energy error. The error is caused by the approximation of the discrete time system matrix dependent on the step size. Since in this configuration the two mass systems are identical again, after the collision both masses move exactly along the same trajectories, consequently no separation force is needed and $\Delta x = 0$ past the collision event. Test case 3 (Figure 63) shows an unsymmetrical configuration with doubled spring stiffness for system 2. In reality this would lead to a continuing force acting on the stiffer system. As one can see, no energy is lost since the coefficient of separation is equal to 1. As opposed to test case 3, test case 4 (Figure 64) shows a high coefficient of separation equal to 10. Now energy is lost if the system sticks together, which can be referred to as deformation or rattling losses and are object to simulation evaluations of real systems.

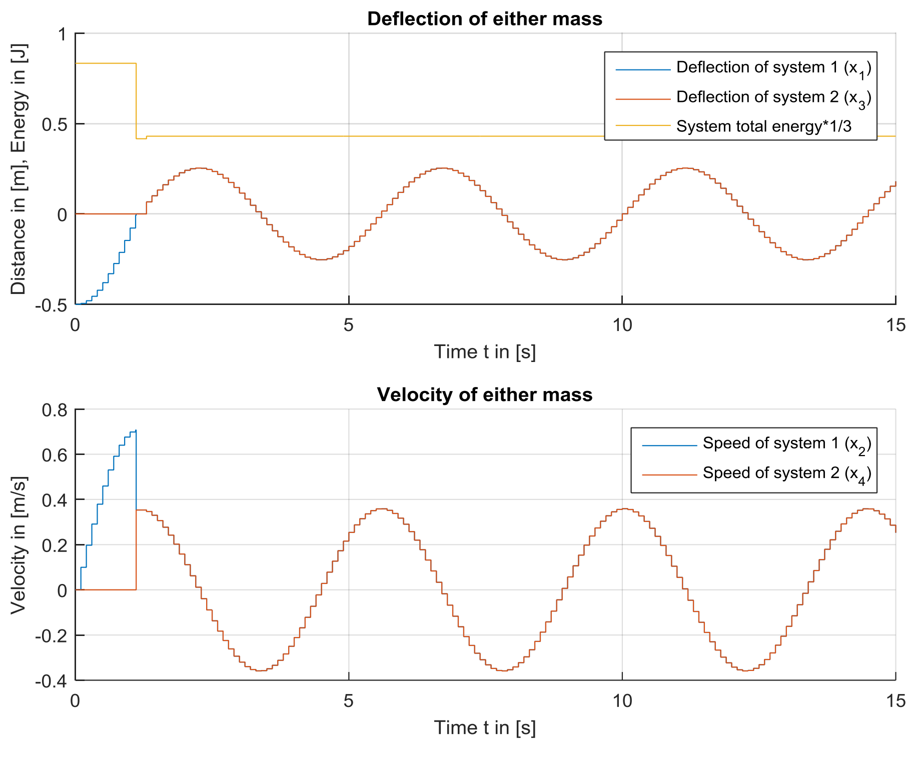

Figure 62: Collision computation. Test case 2.

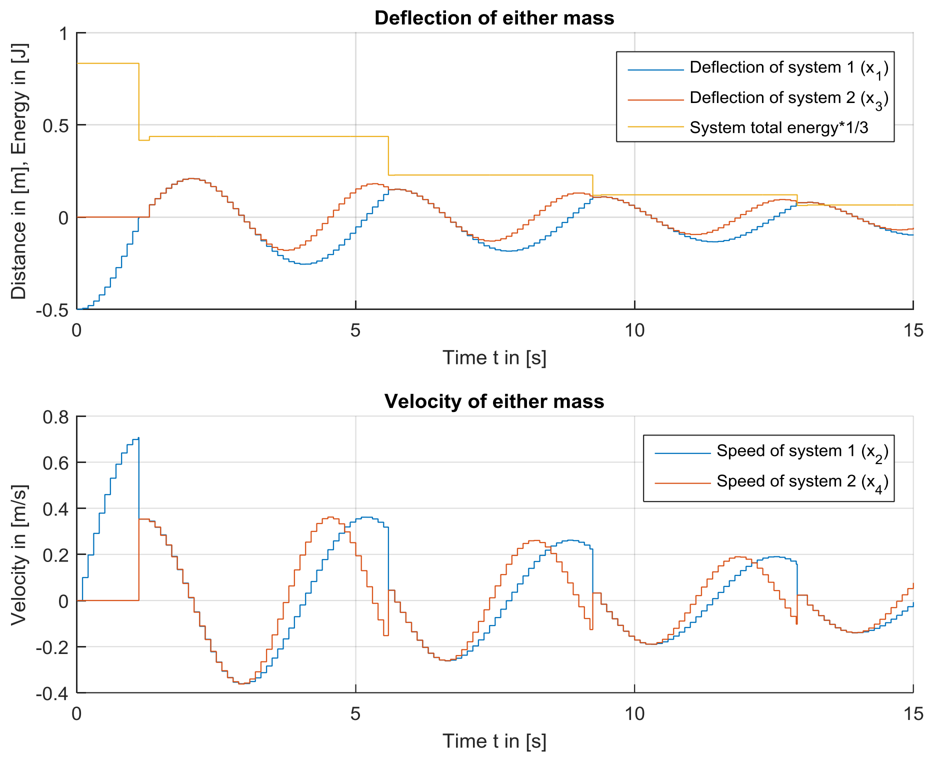

Figure 63: Collision computation. Test case 3.

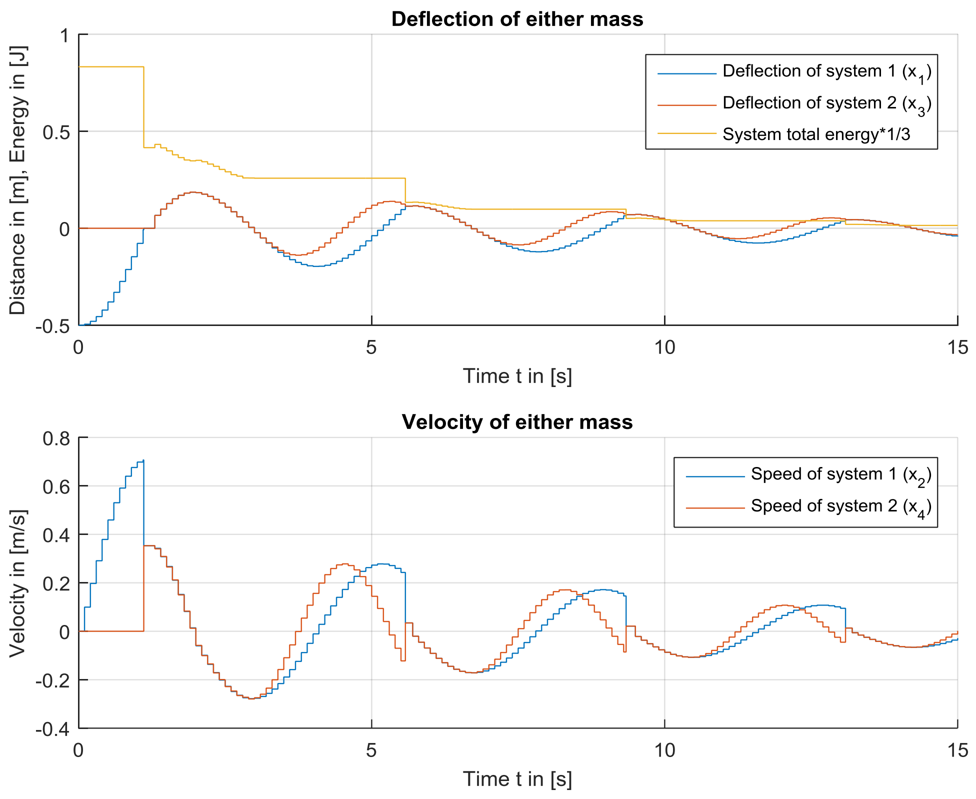

Figure 64: Collision computation. Test case 4.

Test case 5 (Figure 65) shows a configuration where the error in the system speed is visible, if the two systems stick together and the algorithm computes the separation force, to ensure $\Delta x = 0$, but does not emphasize the system speed equality at this step. Consequently the systems alternately comprise a speed difference in the sampling points, which is physically impossible, but the error oscillates around zero.

Test cases 6 and 7 (Figure 66 and Figure 67) show experiments with random parameters and visualize demonstrative examples of configurations. In test case 7, the parameters are chosen extremely regarding the action and reaction of the systems. System 1 comprises one hundred times more mass than system 2, thus it is hardly affected by system 1. The inequality of both systems leads to a heavily oscillating speed error if the systems are sticking to each other.

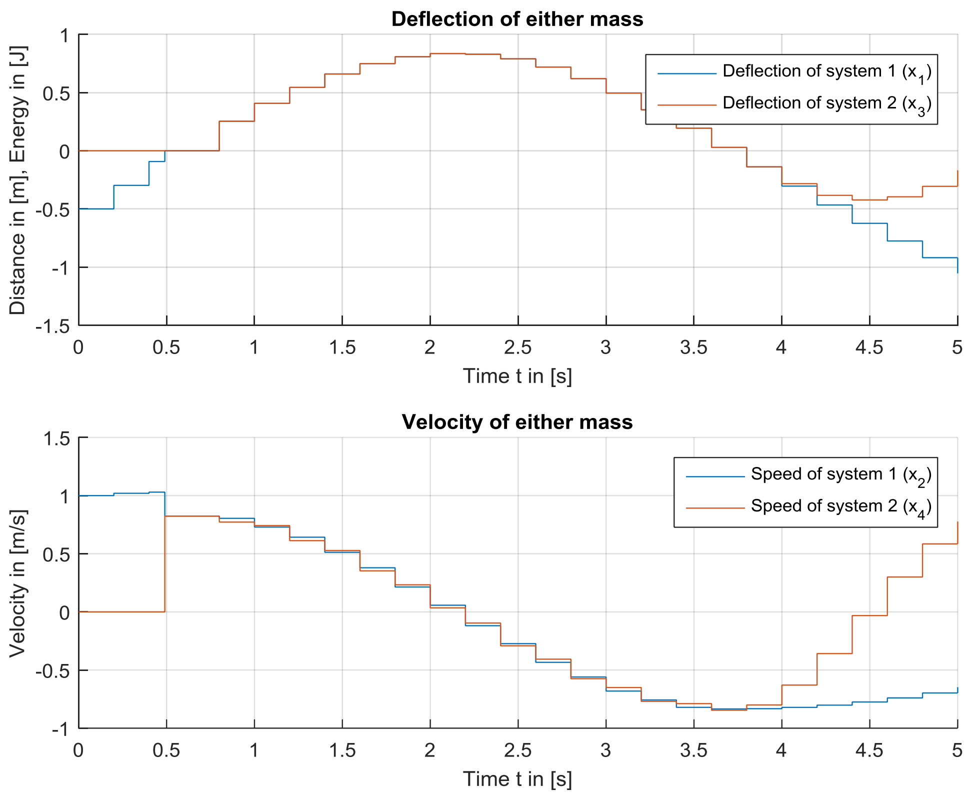

Figure 65: Collision computation. Test case 5.

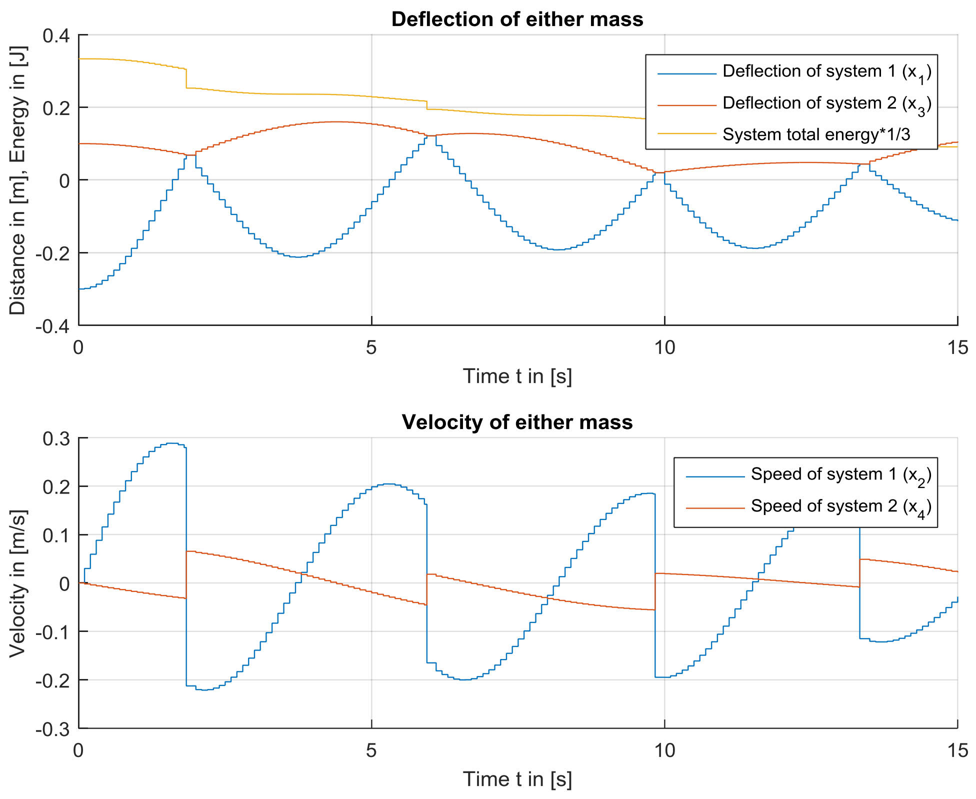

Figure 66: Collision computation. Test case 6.

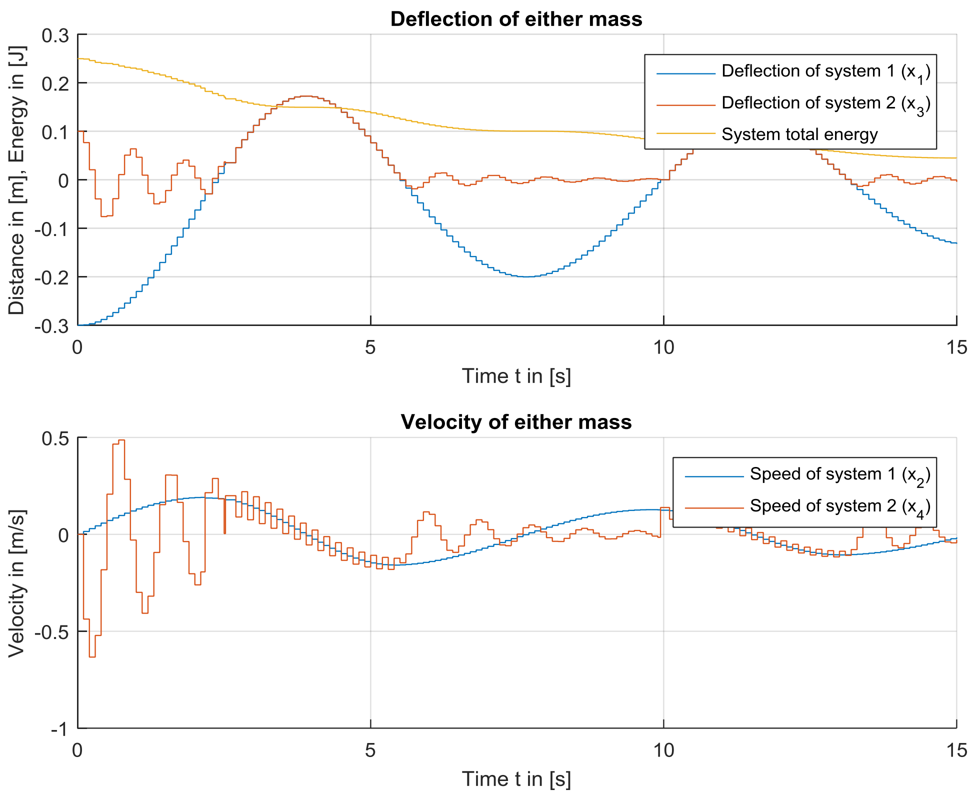

Figure 67: Collision computation. Test case 7.

4.4.2 COMPUTATION TIME

Since it is not possible to determine the exact computation time on an operating system that is not capable of a real time computation, a comparison of the simulation time with the implemented algorithms and a simulation just computing the two autonomous systems was performed. Test case 7 of chapter 4.4.1 was taken as an example.

For the simulation **without any implemented collision algorithms** a simulation time of **0.1143 seconds** was measured with MATLAB®/Simulink® whereas the **addition of the algorithms** leads to a simulation time of **0.1643 seconds**, which means a computation increase of **43.7%.**

4.5 Integration into the drivetrain model

Since collision computation works out well in a simple two-mass-system, the applicability on the "Future Hybrid Drivetrain" model of chapter 2 should be explored.

The "Future Hybrid Drivetrain" (Figure 68) was extended by several elements, to make the algorithm integration easier. The extension involves the wheel inertia $\mathrm{J_w}$, the clutch $\mathrm{C_w}$, which is subject of modelling the tire slip in the consecutive chapter, the intermediate inertia $\mathrm{J_z}$ and the system states for wheel rotational speed ω_w, the intermediate inertia rotational speed ω_z and the differential angle $\delta\varphi_\mathrm{z} = \varphi_\mathrm{z} - \varphi_\mathrm{w}$.

A maximum angle $\delta\varphi_\mathrm{z,max}$ for $\delta\varphi_\mathrm{z}$ is applied for positive angles to embed the collision algorithm.

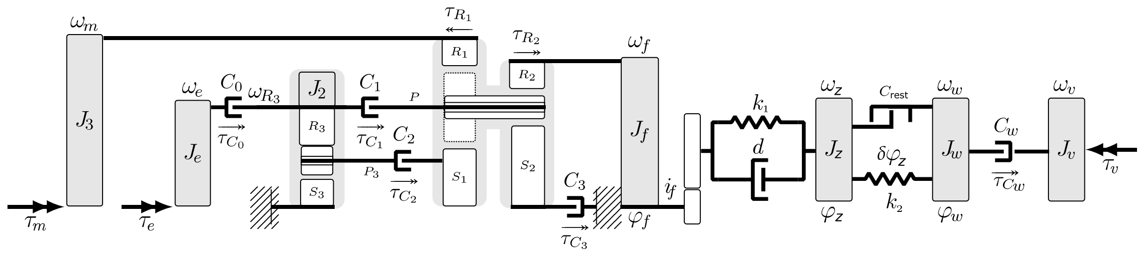

Figure 68: Drivetrain topology for "Future Hybrid Extended (clearance)"

The state vector and the clutch torque vector are consequently extended by a state for the wheel rotational speed ω_w and the torsion $\delta\varphi_\mathrm{z}$, as seen in equation (104).

The MATLAB®/Simulink® of the total model including the nonlinear collision computation and clutch locking computation is shown in Figure 69. The model can be divided into three essential parts, which are marked in different colors: The linear discrete time system with the step size adjustment for zero slip speed approximation (marked in blue color), the collision computation (marked in orange color and shown in Figure 70), and the clutch locking computation (marked in green color and shown in Figure 71).

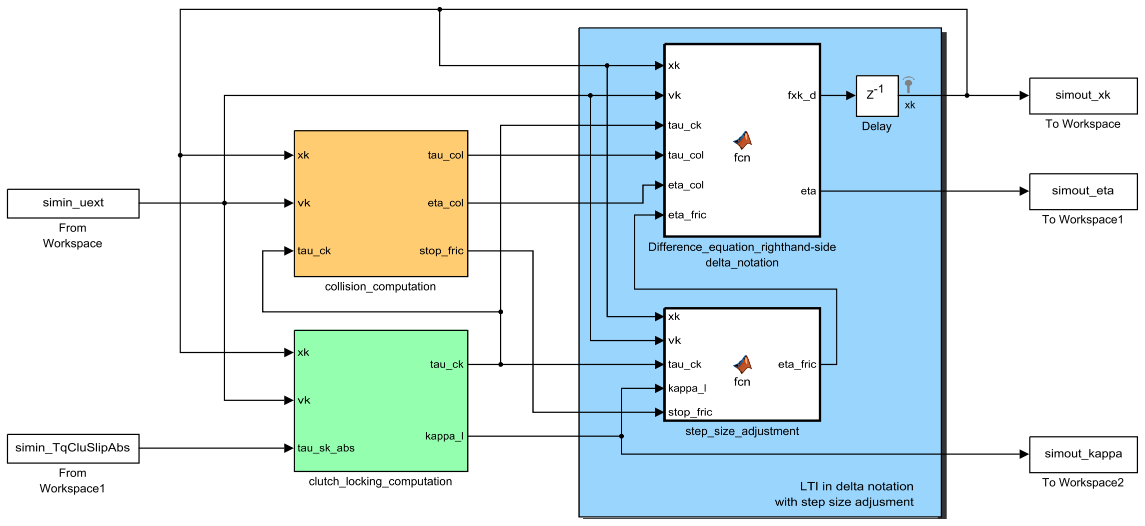

Figure 69: MATLAB®/Simulink® model of hybrid drivetrain model with clearance and clutch locking computation. Top level.

Basically the approach of the collision computation works according to the previous chapters, but several special problems are encountered.

First of all, both algorithms (clutch locking and collision computation) work with independent step size adjustments that might compete with each other. One might develop numerous strategies to handle competing and intervening algorithms. Here the two approaches are simply provided with priorities. Since a collision event is totally strict and physically unneglectable a step size adjustment originating from a collision event is given priority. In the algorithm approach, a collision event is detected before the event and the slip speed zero step size adjustment locked consequently by a flag parameter “stop_fric”.

Secondly, the parameter vectors for the algorithm computation change slightly, due to the state vector, which contains $\Delta\phi$ already as a state variable.

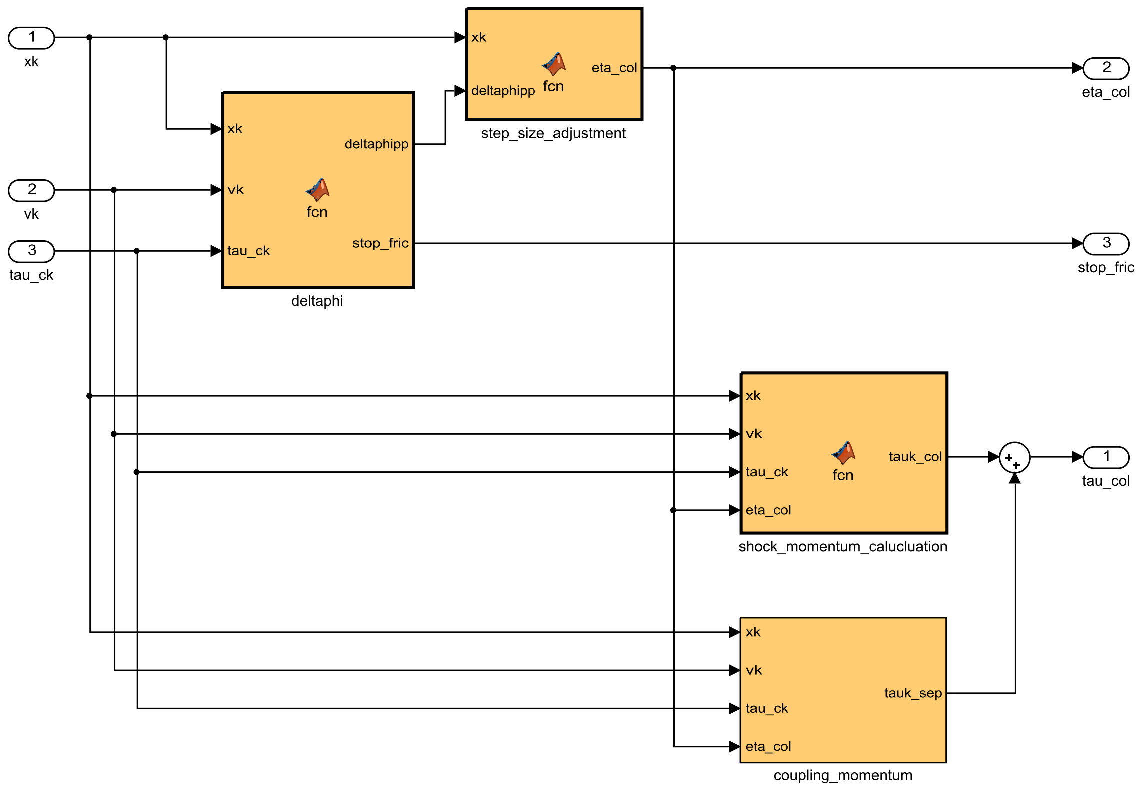

Figure 70: MATLAB®/Simulink® model of hybrid drivetrain model with clearance and clutch locking computation. Sub level: clearance computation.

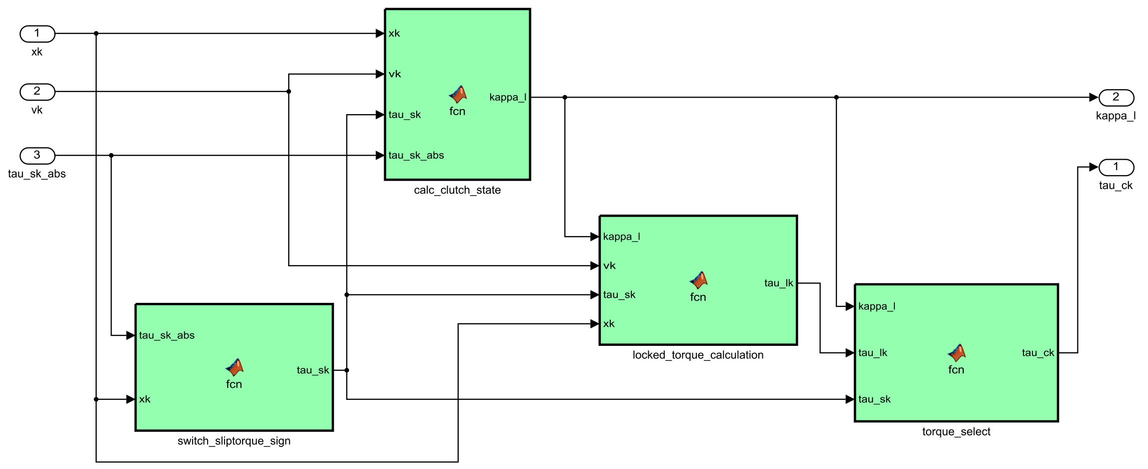

Figure 71: MATLAB®/Simulink® model of hybrid drivetrain model with clearance and clutch locking computation. Sub level: clutch locking computation.

.

Furthermore it is important to consider the gear box ratio at the correct position of the collision and the deflection angle of collision. For simplification purposes the gar box ratio in the case studies has been set to “1”.

The simulation was computed with following values (Table 15 and [3]) for the added elements in drivetrain topology. The inertia of the vehicle J_v was calculated for a wheel with a radius of 1 meter.

All initial values for the state vector were set to 0.

Table 15: System parameters for extended drivetrain topology "clearance"

J_z	J_w	J_v	k_2
$[kg\ m^2]$	$[kg\ m^2]$	$[kg\ m^2]$	$\left[\frac{N}{m}\right]$
3.4093	1.5	135	4000

Table 16: System parameters 2 for test cases

$testcase$	$\delta\varphi_{z,max}$	c_r	T_d
	$[m]$	$[m]$	$[s]$
1	10^{-4}	0	0.1
2	10^{-4}	1	0.1
3	$2 * 10^{-3}$	0	0.1

The input functions for the external moments of the electric and combustion engine as well as the external vehicle force and the clutch torques can be seen in the following Figure 72 and Figure 73.

The separation force input matrix is set to

$$\mathbf{B}_c = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{134}$$

The first two columns represent the vectors for the separating forces, which act on $\dot{\omega}_z$ (index 5) and $\dot{\omega}_w$ (index 6), and the last column is used to act on the speed $\delta\dot{\varphi}_z$ (index 9), for the collision condition. The matrix $\mathbf{H}_c$, used for the discrete time system results of transformation of the continuous time matrix $\mathbf{B}_c$ into the discrete time domain [5].

Basically one clutch of the drivetrain is locked and consequently a gear is selected while a constant torque of the electric engine is supplied. The clutch locking procedure leads to a first deflection of the bounded angle $\delta\varphi_z$. After the clutch is locked (Figure 74) a spontaneous high input torque of the electric engine is added to achieve a second high deflection.

Figure 75 and Figure 76 show the rotational speed of the inertias involved in the system. Masses left of the planetary gear set in the topology plan are referred to as "left hand side masses" just as the right hand side masses. The rotational speeds stay approximately the same for all three test cases since the clearance does not affect them much.

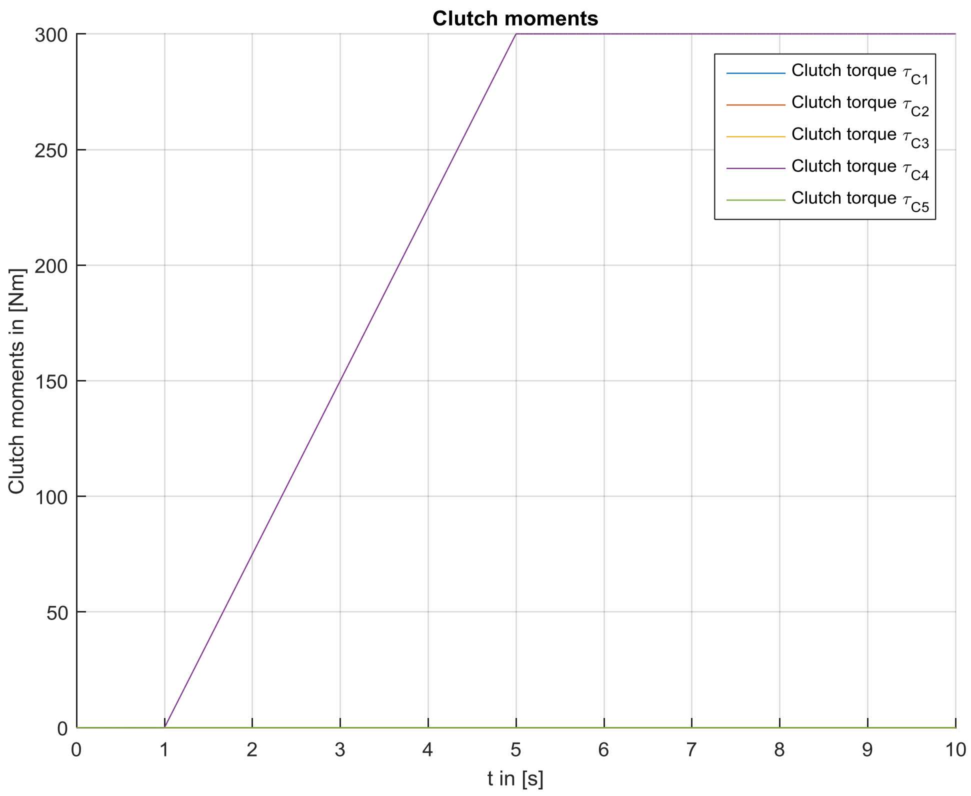

Figure 72: Clearance simulation "Future Hybrid ext.": Clutch input moments

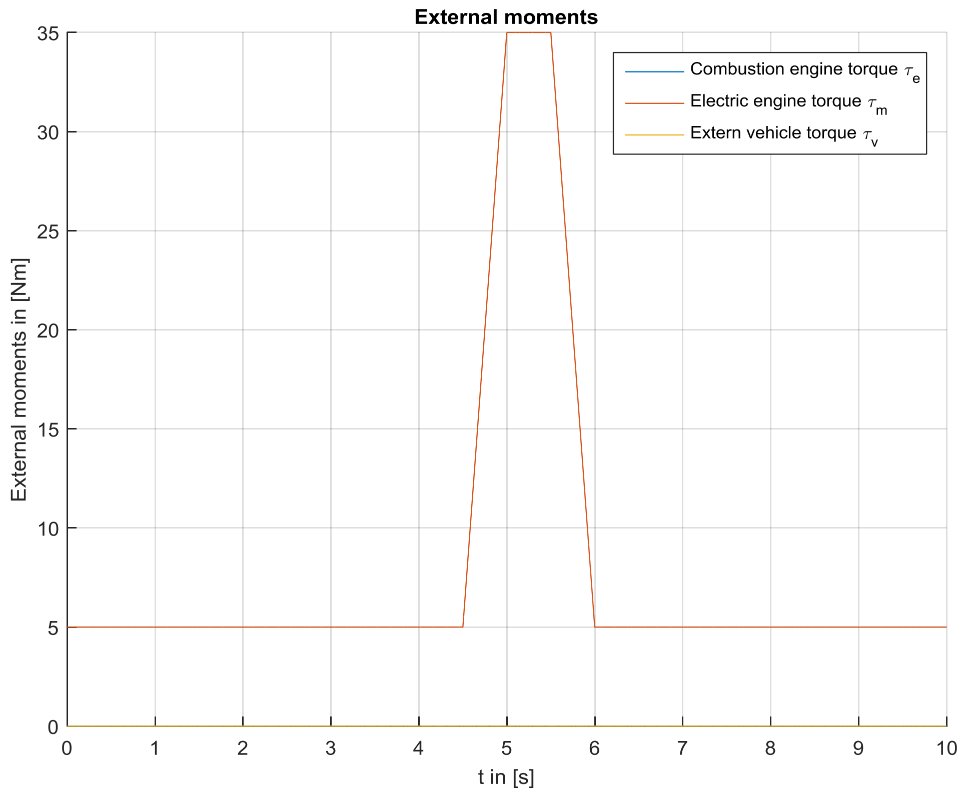

Figure 73: Clearance simulation “Future Hybrid ext.”: External input moments

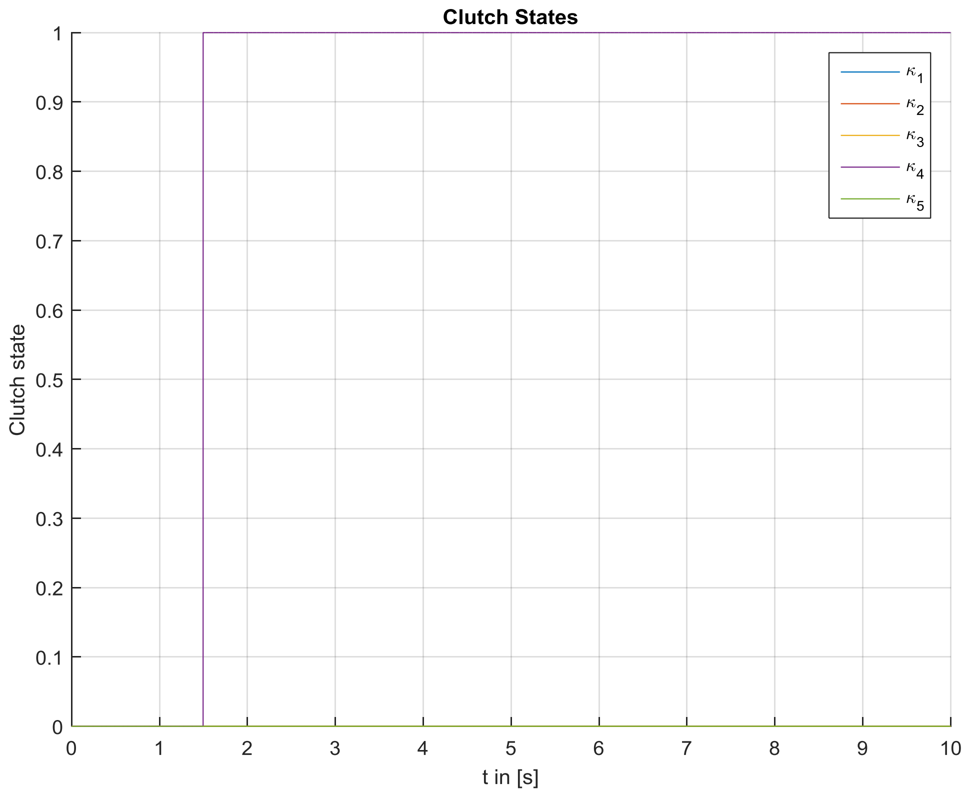

Figure 74: Clearance simulation “Future Hybrid ext.”: Clutch states

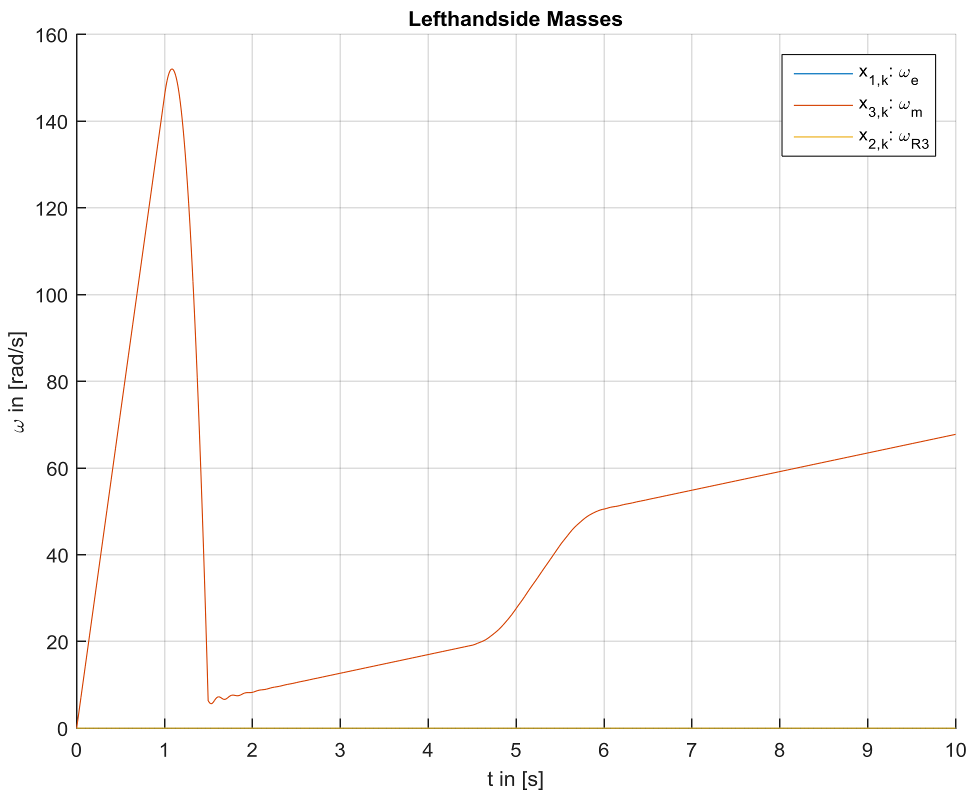

Figure 75: Clearance simulation "Future Hybrid ext.": Left hand side mass rot. speed

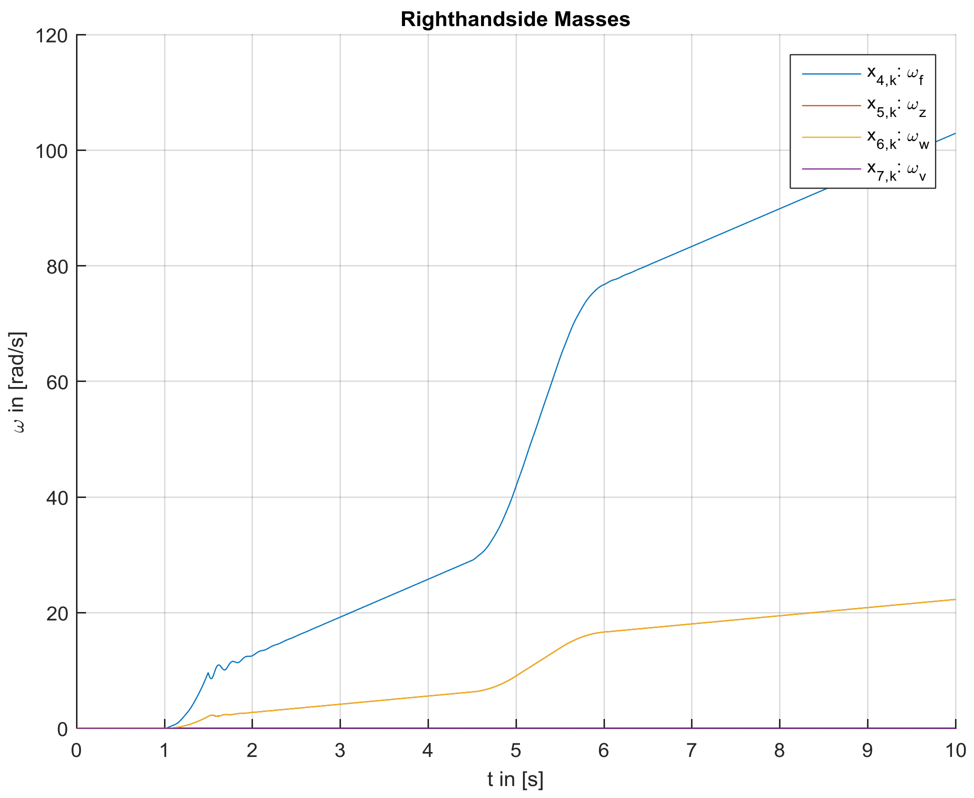

Figure 76: Clearance simulation "Future Hybrid ext.": Right hand side mass rot. speed.

Figure 77 and Figure 78 show the test cases 1 and 2 which differ in their restitution coefficient. Obviously the coefficient of 1 in test case 1 leads to a rattling phenomenon, which is portrayed by the simulation correctly. A restitution coefficient of 0 leads to a fully plastic collision and is physically associated with an energy loss and the abundance of rattling.

Due to the relative small maximum deflection angle in the first two test cases a further test case 3 should illustrate another behavior (Figure 79), with several collisions. It also displays part of the behavior of the spring without a maximum clearance.

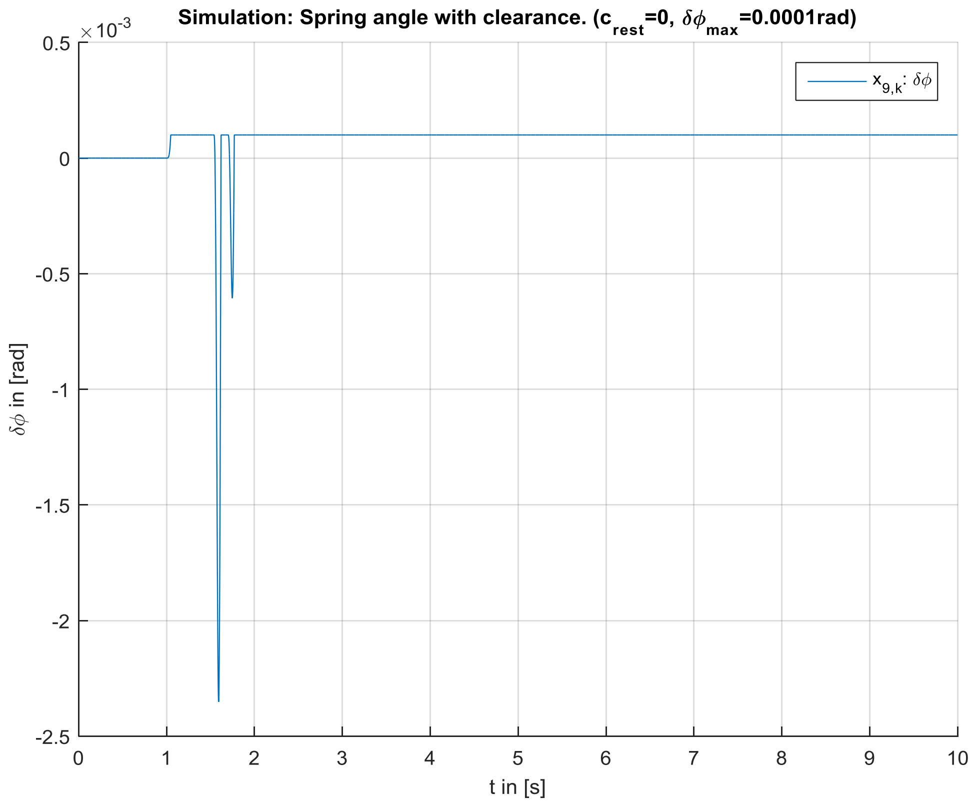

Figure 77: Clearance simulation "Future Hybrid ext.: Test case 1

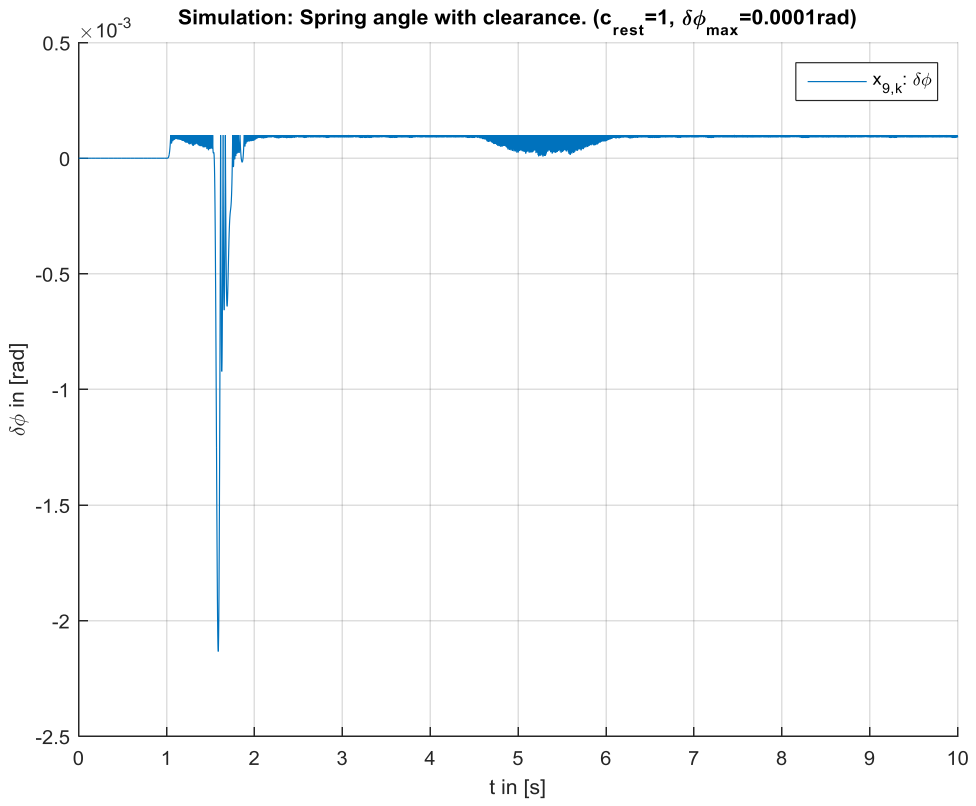

Figure 78: Clearance simulation "Future Hybrid ext.: Test case 2

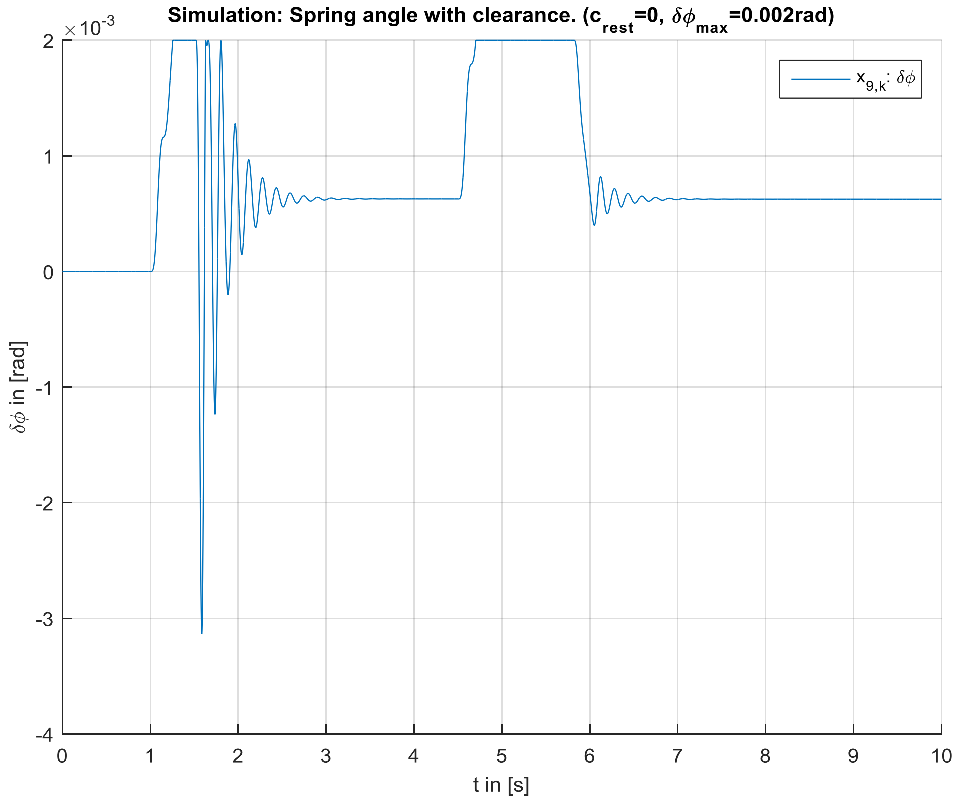

Figure 79: Clearance simulation "Future Hybrid ext.: Test case 3

4.6 OUTLOOK TO HIGHER ORDER SYSTEMS AND ACCURACY IMPROVEMENT

4.6.1 MULTIPLE DIMENSIONS

Though it is not necessary to implement collision simulation for more than one dimension it is easily possible to extend the used algorithm for two or three dimensions, by simply splitting the velocity and positions variables into all dimensions and applying the algorithm for each dimension.

4.6.2 MULTIPLE MASSES

If several masses are able to collide with each other within one sample period further investigation of the algorithm is needed. If we take a look at equation (130), where the separation force and the collision force are simply added, because they would not act in the same sample step, they must not be added if several masses collide, because the separation force might act on one mass pair, which sticks together, whereas the collision force is computed between a mass, which is about to collide in the same step.

No problem arises, if masses collide independently of each other.

4.6.3 ACCURACY IMPROVEMENT AFTER A COLLISION EVENT

Equations (125), (130) and (132) compute a separation force $\boldsymbol{\tau}_{\mathrm{s,k}}$, that guarantees the masses not to overlap in the consecutive step, thus $\Delta\mathbf{x}_{\mathrm{k+\eta}} \overset{!}{=} 0$. As it can be seen in the presented examples, particularly in Figure 65 and Figure 67, the velocity is not affected by this requirement for $\Delta\mathbf{x}_{\mathrm{k+\eta}}$. Since the input vector $\boldsymbol{\tau}_{\mathrm{s,k}}$, only acts at two difference equations, it is conceivable to add further inputs to the system on the remaining two difference equations, which lead to a velocity difference of zero, thus $\Delta \mathrm{v}_{\mathrm{k+\eta}} = \mathrm{v}_{\mathrm{sys2,k+\eta}} - \mathrm{v}_{\mathrm{sys2,k+\eta}} = \mathrm{x}_{3,\mathrm{k+\eta}} - \mathrm{x}_{1,\mathrm{k+\eta}} \overset{!}{=} 0$. Unfortunately the new inputs to the system cannot be seen as forces, they rather represent a velocity shift, consequently the requirement for this velocity shift not to add energy to the system is more sophisticated and leads to a series of problems which is not part of this thesis.

5 NONLINEAR TIRE MODEL

5.1 INTRODUCTION

In this chapter the integration of a nonlinear tire model by means of the existing clutch model is examined. Obviously a clutch acts similar than a vehicle tire, both transmit a friction force depending on the difference speed of the friction elements.

Numerous tire modeling approaches exist for vehicle modelling [14]. They usually diverge in several classifications, depending on the method used. More theoretical modeling methods by means of simple or complex physical models have a high degree of fit, allow a high number of special experiments and deliver a high insight in the tire behavior. More empirical methods that rather treat the tire as a black box, may also have a high degree of fit, but usually allow neither special experiments nor insight in the tire behavior. Because they are generally easier to apply they are widely used in the industry, especially the so called "Pacejka magic formula" [14], which was developed by TU-Delft and Volvo in the mid-eighties.

The Pacejka magic formula (135) is basically a try to construct a formula, which can be fit easily into measurement data originating from real tire experiments by a parameter variation [15]. The Parameters D (peak value), C (shape factor), B (stiffness factor) and E (curvature factor) shape the curve. Parameters S_V and S_H shift the curve off the origin point.

$$y(x) = D\sin(C\arctan(Bx - E(Bx - \arctan(Bx)))) \tag{135}$$

$$Y(X) = y(X + S_H) + S_V \tag{136}$$

The **output variable Y** can be used for several interesting components of a tire model, including the longitudinal force F_x, the side force F_y and the aligning torque M_z. The **input variable X** resembles the longitudinal slip κ or the slip angle α.

In this thesis only the longitudinal force F_x dependent on the longitudinal slip κ is of interest. The longitudinal slip κ is defined as follows in (137). The revolution speed Ω_0 represents the angular speed of the tire without any slip, thus $\Omega_0 = \frac{v_x}{r_e}$, with v_x as the vehicle speed and r_e as the effective tire radius. The wheel angular velocity Ω is bigger than Ω_0 during acceleration and smaller than Ω_0 during braking. A κ of 0 depicts a free rolling tire without external forces and a κ of -1 is associated with a full break.

$$\kappa = -\frac{\Omega_0 - \Omega}{\Omega_0} = -\frac{\Delta\Omega}{\Omega_0} \tag{137}$$

The definition of the slip leads to a limitation, since the case of $\Omega_0 = 0$ is not permitted since it would lead to $\kappa \to \infty$.

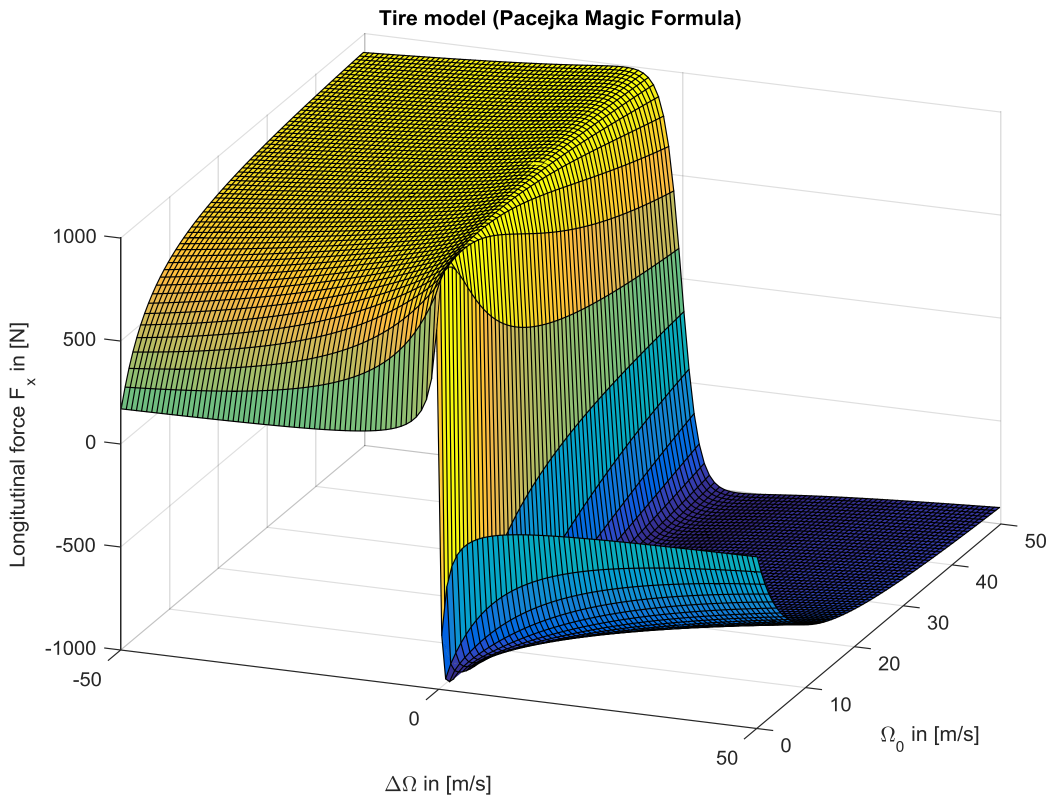

Figure 80: Tire model "Pacejka magic formula" for a chosen parameter set.

Figure 80 shows the Pacejka magic formula for the parameter set, based on empirical data for try tarmac (B=10, C=1.9, D=1, E=0.97) [16]. The longitudinal force is dependent on the slip speed $\Delta\Omega$ and the free rolling wheel speed Ω_0, which is proportional to the vehicle speed. Obviously the magic formula corresponds well with the signum function, which would be created by a substituted clutch in the model (Figure 81), but only for high vehicle speeds v_x and consequently Ω_0. For low speeds Ω_0 the curve changes drastically its shape, and the signum function leads to major errors in the simulation.

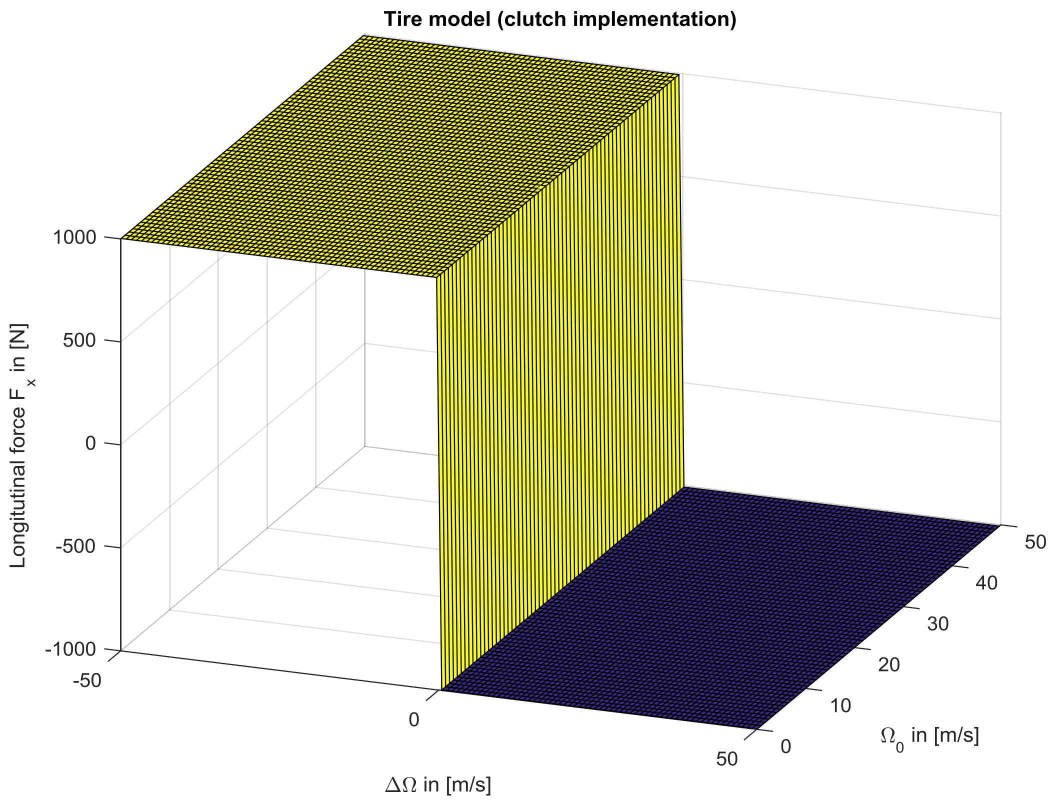

Figure 81: Tire model “Clutch” for a chosen parameter set.

5.2 SIMULATION COMPARISON

For a comparison of the effects in the extended drive train topology of a hybrid engine, first the MATLAB®/Simulink® **model with the integrated magic formula** is used for the simulation (Figure 83) and secondly the **friction is approximated by the clutch C_w itself** (Figure 82), with a constant pressure to achieve the best fit of the Pacejka magic formula, with the parameter set (B=10, C=1.9, D=800Nm, E=0.97).

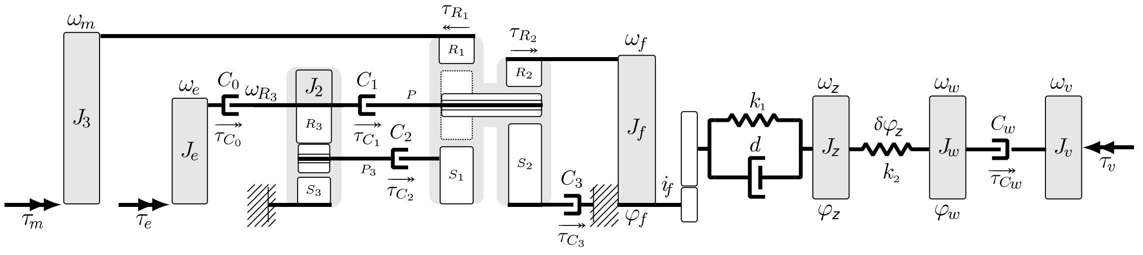

Figure 82: Drivetrain topology for "Future Hybrid Extended (wheel friction - clutch)"

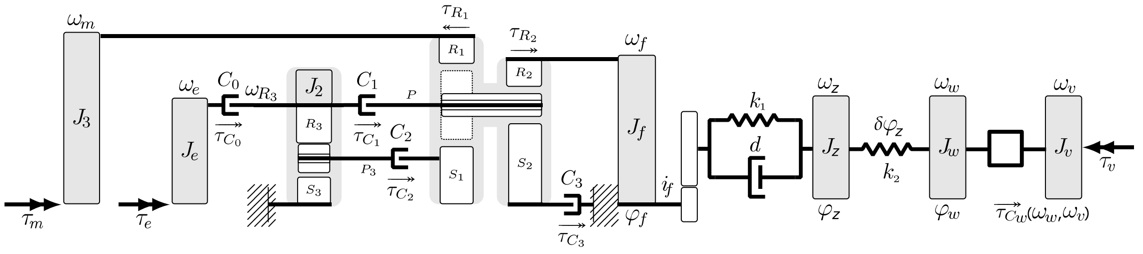

Figure 83: Drivetrain topology for "Future Hybrid Extended (wheel friction – magic formula)"

Parameters as well as the vehicle resistance torque τ_v are chosen according to Table 15 and [3]. The vehicle resistance torque τ_v is a function of the squared speed v_v^2, the rolling resistance, the breaking force and the road inclination. Latter two are set two zero for this comparative study.

The Pacejka magic formula for the tire-road interaction gets the following shape (equation (138)).

$$\tau_{Cw}(\omega_v, \omega_w) = D \sin\left(C \arctan\left(B\left(-\frac{\omega_v - \omega_w}{\omega_v}\right) - E\left(B\left(-\frac{\omega_v - \omega_w}{\omega_v}\right) - \arctan\left(B\left(-\frac{\omega_v - \omega_w}{\omega_v}\right)\right)\right)\right)\right) \tag{138}$$

The comparative simulation study reveals the following: As long as the slip speed is close to zero, particularly inside the two maximum ridges of the magic formula, which occur by varying the absolute slip speed $\Delta\Omega$ (Figure 80), the clutch modelling of the Pacejka magic formula delivers an excellent performance. As soon as the absolute slip speed exceeds the peak value in either direction the simulation results diverge drastically in the first consideration, especially for a low free rolling speed Ω_0, thus a low vehicle speed.

For the purpose of figuring out the performance four simulation cases have been applied (Figure 84 to Figure 92). The first case starts at an initial high vehicle velocity, but all other parts of the drivetrain start with a zero velocity, which leads to an initial high wheel slip. The drivetrain is accelerated quickly to the appropriate stationary deceleration due to the vehicle resistance. For a certain time on gear box clutch is closed and a torque of the electric engine is applied to accelerate the vehicle. The revolution speeds of both simulation variations equal each other very close.

The second case (Figure 87, Figure 88) shows acceleration with a low initial speed and a wheel slip just before the remarkable slip, where the transmitted wheel/road force decreases with a rising slip. The third test case shows a wheel slip which overruns drastically. It is remarkable that the applied torque was increased just by about 1% in this test case. Obviously the implemented clutch does not deliver converging results.

The last test case (Figure 91, Figure 92) shows a repeating acceleration with a high slip. Finally the revolution states of both systems show just a little error.

Taking a closer look to the simulation results the wheel speeds of both simulations diverge during the extraordinary high wheel slip (Figure 90), because less friction force can be transferred to the road. But since the input energy is equal in both simulations and most of the model parts are described linearly the system states of both models converge to each other pretty well at the end (Figure 88, Figure 90 and Figure 92). A considerable nonlinear factor is the drag friction resistance of the vehicle, which is definitely nonlinear. But the only case where this nonlinearity influences the final simulation results remarkably, would be a situation where the vehicle accelerates for a vast distance with an extremely high torque and consequently overrunning wheels, or performs a full break with slipping wheels. Since the first case can be usually excluded from ordinary vehicle simulations and the second case is only desired for certain cases, the simulation by means of clutch for the road-tire interaction works out rather well.

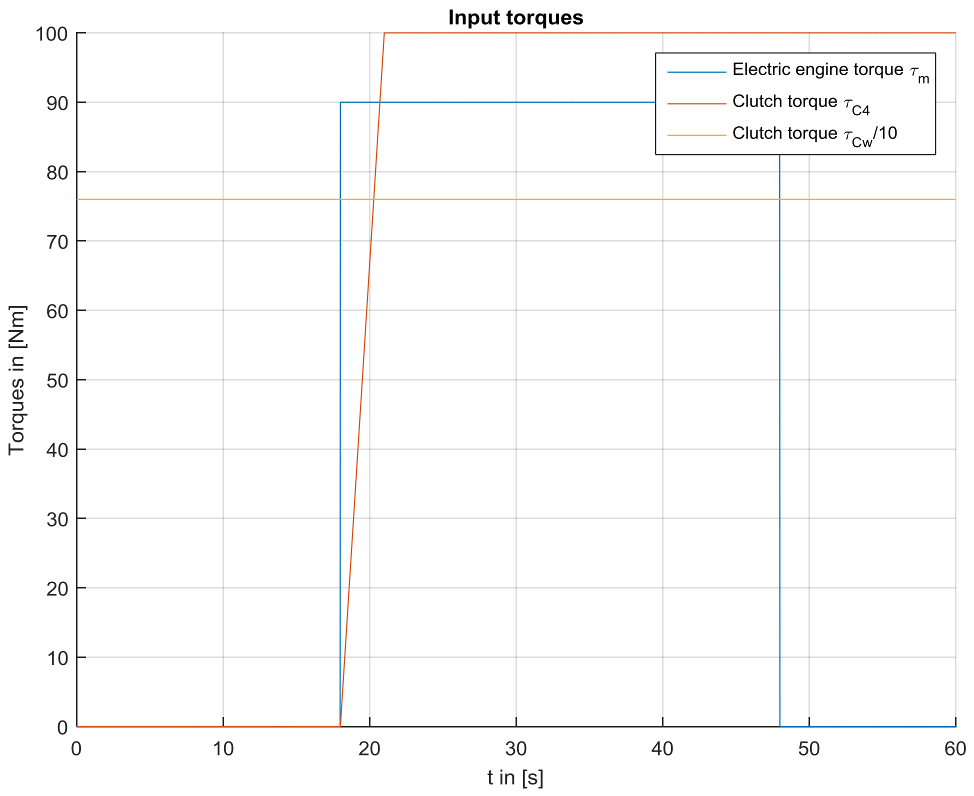

Figure 84: Tire model integration: Comparative simulation 1. Input torques.

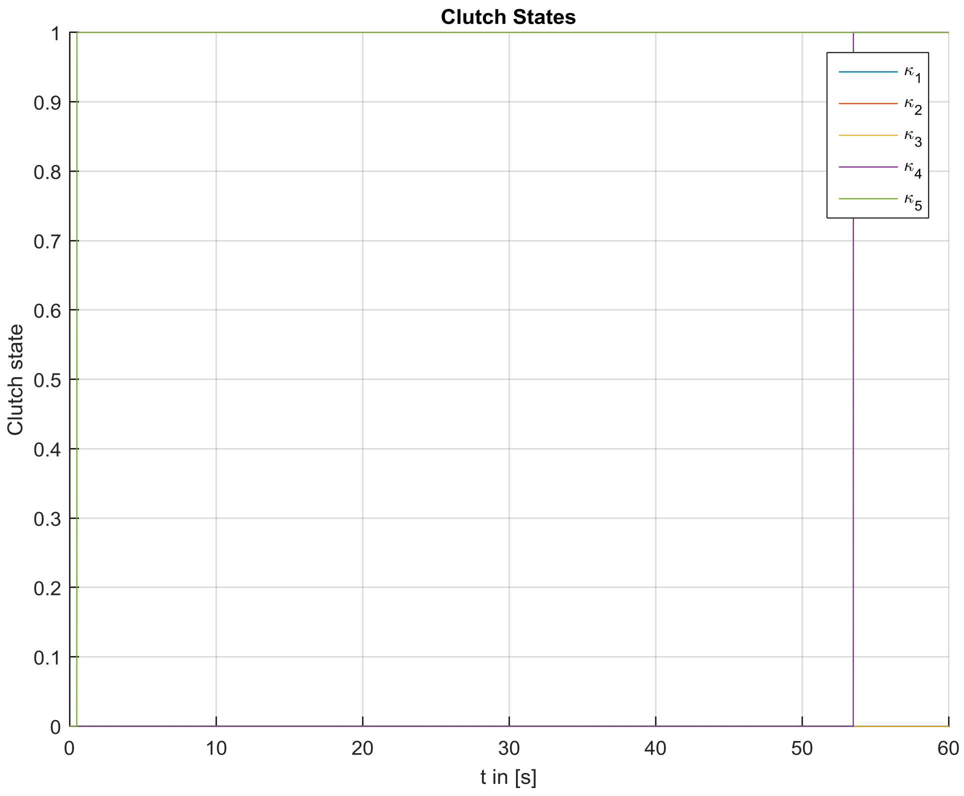

Figure 85: Tire model integration: Comparative simulation 1. Clutch states.

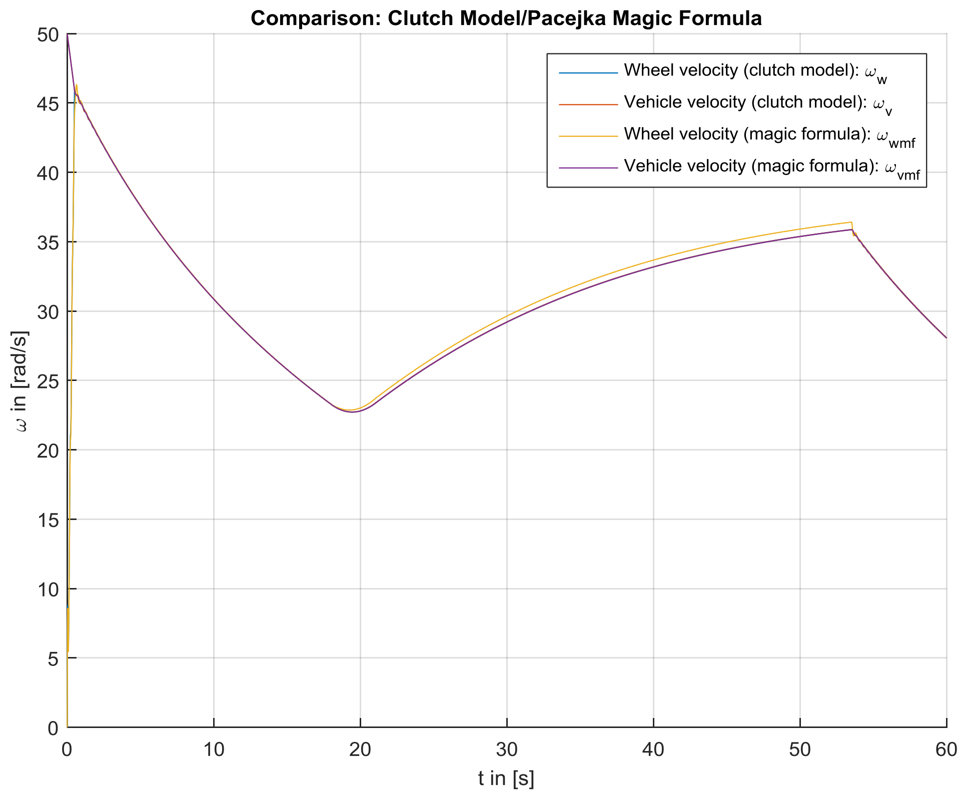

Figure 86: Tire model integration: Comparative simulation 1. Revolution speeds.

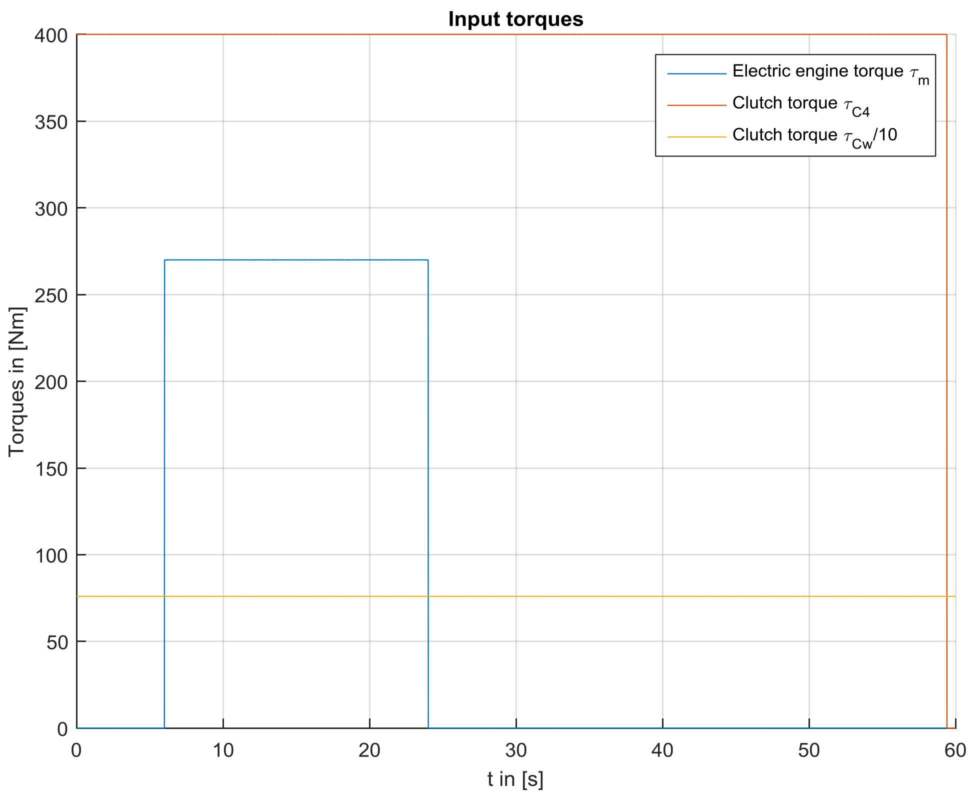

Figure 87: Tire model integration: Comparative simulation 2. Input torques.

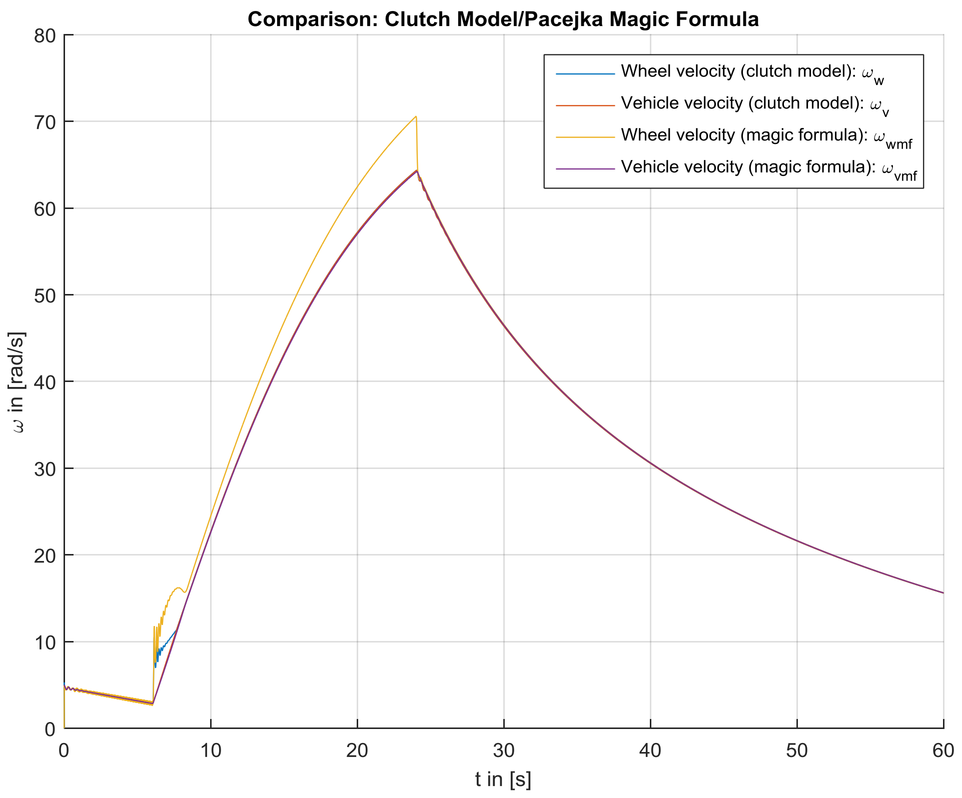

Figure 88: Tire model integration: Comparative simulation 2. Revolution speeds.

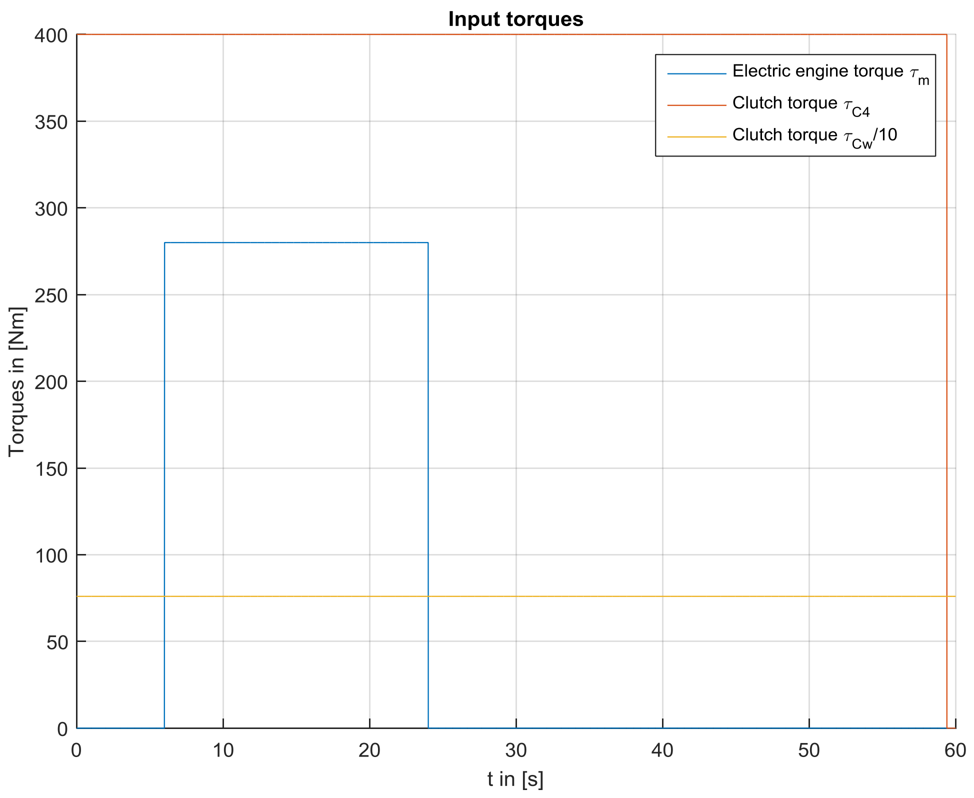

Figure 89: Tire model integration: Comparative simulation 3. Input torques.

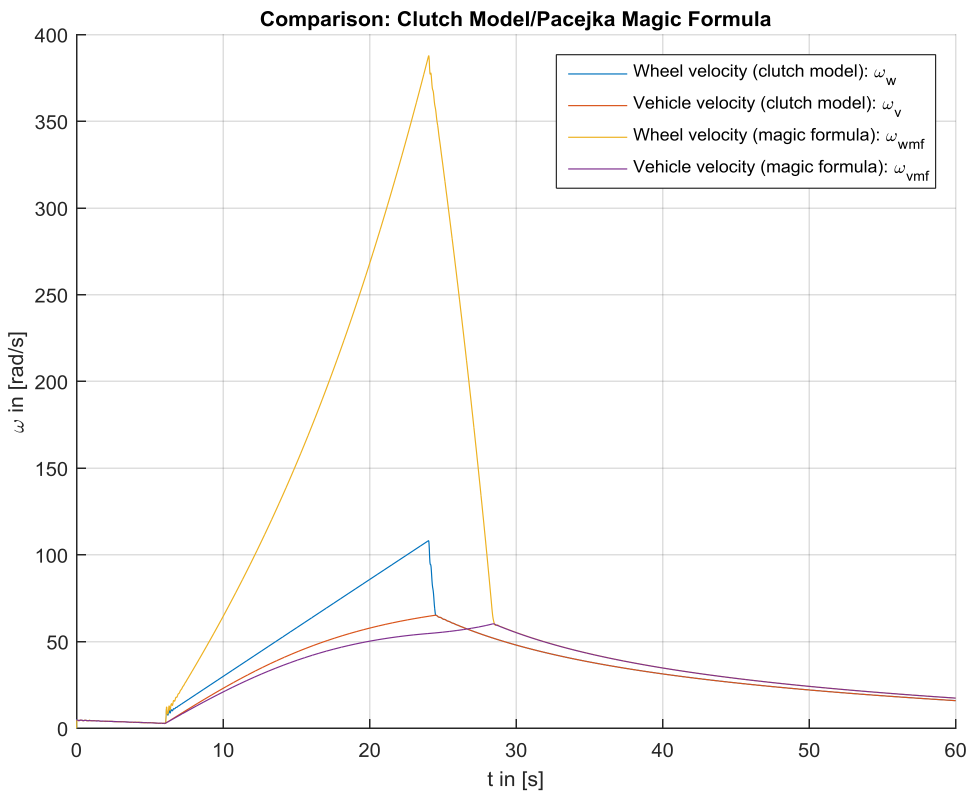

Figure 90: Tire model integration: Comparative simulation 3. Revolution speeds.

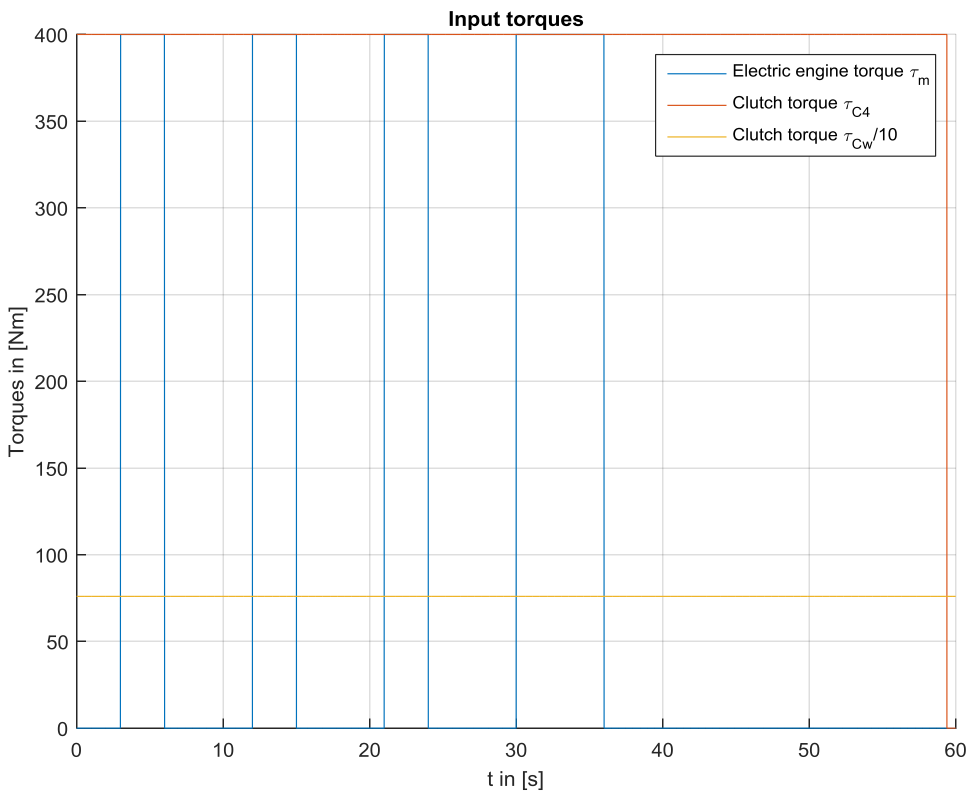

Figure 91: Tire model integration: Comparative simulation 4. Input torques.

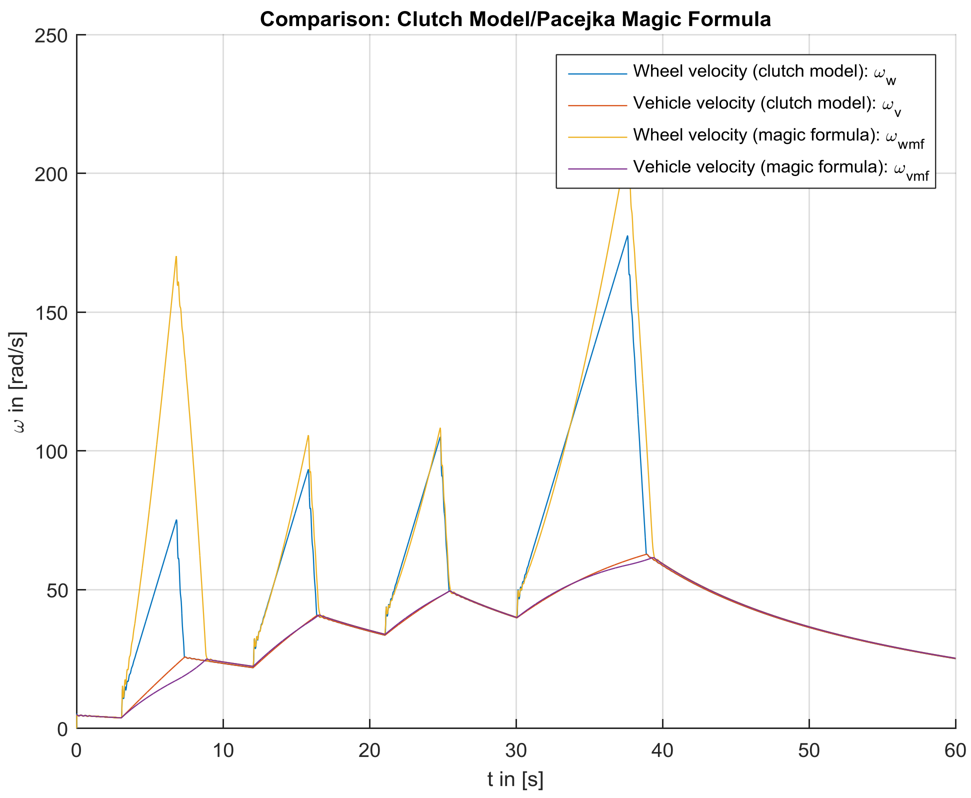

Figure 92: Tire model integration: Comparative simulation 4. Revolution Speeds.

6 RESULTS

Several methods have been developed to integrate nonlinear elements into state-of-the-art drivetrain simulations, with an emphasis on real time constraints, accuracy improvement and compatibility to existing simulation strategies [3].

The integration of typical nonlinear damper elements [4], which is part of chapter 3 and the integration of a nonlinear tire model by means of an existing clutch modeling approach [3], which is the content of chapter 5, is straight forward to implement and basically offers the possibility of extending drivetrain simulations by new elements.

Nonlinear spring elements can be integrated into discrete time drivetrain simulations by several methods, which differ in their accuracy, implementation affords and computation time and which is the main part of chapter 3. For piecewise affine spring force functions, methods have been found, which are highly accurate and rather low in their computation time and integration afford.

Collisions might occur in drivetrains, if gear wheels have a high clearance or if shaft elements collide according to the construction and extraordinary circumstances. Since collisions are highly nonlinear and affect the system behavior exceptionally high, the implemented algorithm offers a possibility of taking collision events into account.

Generally, this thesis focuses on the theoretical approaches of integrating nonlinear elements. When it comes down to the implementation on a specific embedded system, it is still affordable to consider the computation power of the certain hardware used and the capability of extending the simulation algorithm by nonlinear computation features.

7 APPENDIX

APPENDIX A INVESTIGATING THE TRANSITION MATRIX APPROXIMATION ERROR

This chapter deals with the error made by the approach of approximating a discrete time transition matrix $\delta\mathbf{\Phi}(\mathrm{t}_\eta) = (\mathbf{\Phi}(\mathrm{t}_\eta) - \mathbf{I})$ in delta notation linearly by a known transient matrix $\mathbf{\Phi}(\mathrm{T_d})$ for a basic step size $\mathrm{T_d}$, where $\mathrm{t}_\eta = \eta\mathrm{T_d}$ and $\eta \in \mathbb{R}|\ 0 < \eta \leq 2$.

Let us first take a look at the calculation of a discrete consecutive step in delta notation (equations (139) to (141)). The exact system state vector $\mathbf{x}_{\mathrm{k}+\eta}$ at the time t_η is computed in delta notation in equation (140) and the approximated state vector $\mathbf{x}'_{\mathrm{k}+\eta}$ is calculated by means of equation (141).

$$\delta\mathbf{x}_\mathrm{k}(\eta\mathrm{T_d}) = \frac{1}{\mathrm{T_d}}(\delta\mathbf{\Phi}(\eta\mathrm{T_d})\mathbf{x}_\mathrm{k} + \mathbf{h}(\eta\mathrm{T_d})\mathrm{u_k}) \tag{139}$$

$$\mathbf{x}_{\mathrm{k}+\eta} = \mathbf{x}_\mathrm{k} + \eta\mathrm{T_d}\delta\mathbf{x}_\mathrm{k}(\eta\mathrm{T_d}) = \mathbf{x}_\mathrm{k} + ((\mathbf{\Phi}(\eta\mathrm{T_d}) - \mathbf{I})\mathbf{x}_\mathrm{k} + \mathbf{h}(\eta\mathrm{T_d})\mathrm{u_k}) \tag{140}$$

$$\mathbf{x}_{\mathrm{k}+\eta} \approx \mathbf{x}'_{\mathrm{k}+\eta} = \mathbf{x}_\mathrm{k} + \eta\mathrm{T_d}\delta\mathbf{x}_\mathrm{k}(\mathrm{T_d}) = \mathbf{x}_\mathrm{k} + \eta((\mathbf{\Phi}(\mathrm{T_d}) - \mathbf{I})\mathbf{x}_\mathrm{k} + \mathbf{h}(\mathrm{T_d})\mathrm{u_k}) \tag{141}$$

For further investigations the error determination focuses on the **autonomous system**, with $\mathrm{u_k} \equiv 0\ \forall \mathrm{k} \in \mathbb{Z}$, since the influence of the approximation of $\mathbf{h}(\eta\mathrm{T_d})$ would be added to the approximation of $\delta\mathbf{\Phi}(\eta\mathrm{T_d})$. Consequently the equations are reduced to the following:

$$\mathbf{x}_{\mathrm{k}+\eta} = \mathbf{x}_\mathrm{k} + ((\mathbf{\Phi}(\eta\mathrm{T_d}) - \mathbf{I})\mathbf{x}_\mathrm{k}) \tag{142}$$

$$\mathbf{x}_{\mathrm{k}+\eta} \approx \mathbf{x}'_{\mathrm{k}+\eta} = \mathbf{x}_\mathrm{k} + \eta\big((\mathbf{\Phi}(\mathrm{T_d}) - \mathbf{I})\mathbf{x}_\mathrm{k}\big). \tag{143}$$

In Figure 93 the approximation of a continuous trajectory $\mathrm{x(t)}$ by the delta term variation can be shown at the step $\mathrm{x_k} = 1$. As it can be seen above, for the variation $\eta = 1$ the approximation becomes exact, because then $\delta\mathbf{\Phi}(\eta\mathrm{T_d}) = \eta\delta\mathbf{\Phi}(\mathrm{T_d})$. The approximation is used for η-values from zero to two.

A spring damper mass system for demonstration purposes has been chosen according to Table 17, where the basic step size T_d is chosen in different fractions of the system resonance period T_{res}.

Table 17: System parameters for delta term approximation

c_{sys}	d_{sys}	m	$\boldsymbol{x}(t=0)$	T_d
$\left[\frac{N}{m}\right]$	$\left[\frac{Ns}{m}\right]$	$[kg]$	$[m]$	$[s]$
10	5	1	$[0 \quad 0]$	$[0.1\ 0.2\ 0.5]T_{res} = [0.031\ 0.093\ \ 0.154]$

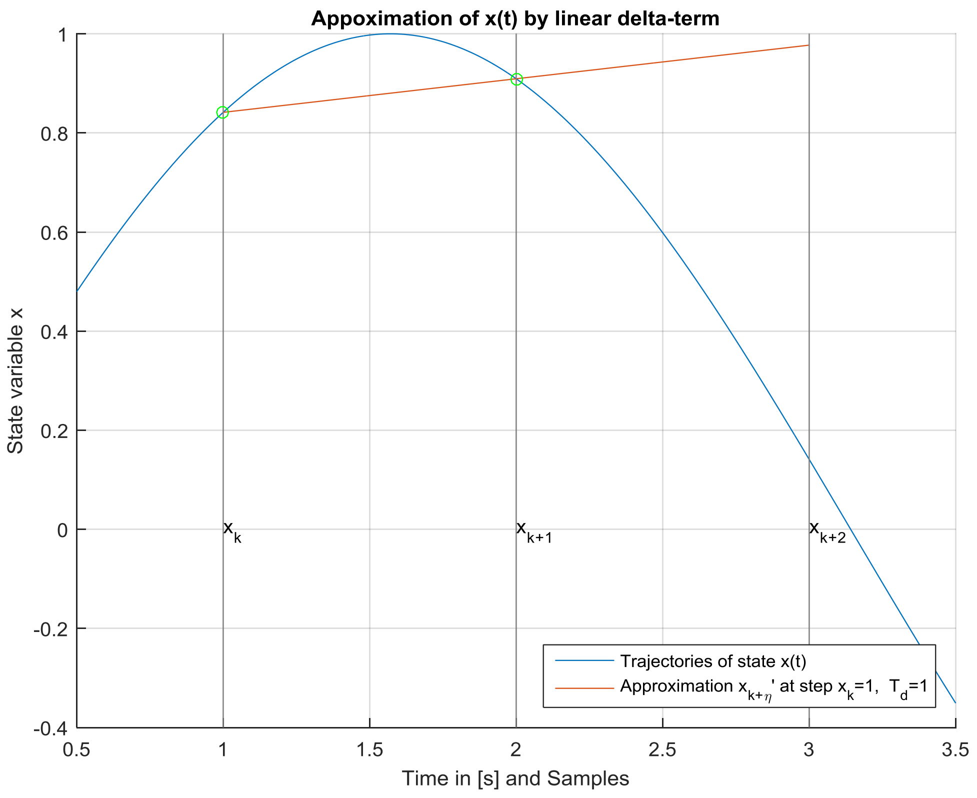

Figure 93: Approximation of x(t) by a linear delta term variation

The error made by the approximation can be described by the error difference matrix $\Delta\boldsymbol{\Phi}_{app}(\eta, T_d)$ in equation (144), which can be derived from the relative error $e_{2,rel}(\eta)$ in the L^2-norm (Euclidian norm),that can be seen in equation (145). It is sufficient to represent the error $e_1(\eta, T_d)$ by an upper bound $e_{1,max}(\eta, T_d)$, based on the Cauchy-Schwarz inequality [2],

$$\Delta\boldsymbol{\Phi}_{\mathrm{app}}(\eta) = \delta\boldsymbol{\Phi}(\eta \mathrm{T_d}) - \eta\delta\boldsymbol{\Phi}(\mathrm{T_d}) \tag{144}$$

$$\mathrm{e_1}(\eta, \mathrm{T_d}) = \frac{\left\|\mathbf{x}'_{\mathrm{k}+\eta} - \mathbf{x}_{\mathrm{k}+\eta}\right\|_2}{\|\mathbf{x}_\mathrm{k}\|_2} = \frac{\left\|\Delta\boldsymbol{\Phi}_{\mathrm{app}}(\eta, \mathrm{T_d})\mathbf{x}_\mathrm{k}\right\|_2}{\|\mathbf{x}_\mathrm{k}\|_2} \tag{145}$$

$$\mathrm{e_1}(\eta, \mathrm{T_d}) \le \mathrm{e_{1,max}}(\eta, \mathrm{T_d}) = \frac{\left\|\Delta\boldsymbol{\Phi}_{\mathrm{app}}(\eta, \mathrm{T_d})\right\|_2 \|\mathbf{x}_\mathrm{k}\|_2}{\|\mathbf{x}_\mathrm{k}\|_2} = \left\|\Delta\boldsymbol{\Phi}_{\mathrm{app}}(\eta, \mathrm{T_d})\right\|_2 \tag{146}$$

Figure 94 shows the Euclidian norm of the upper bound of the error made by the described approximation. Obviously the error becomes the highest, if the variation η comes close to the value 2 and generally becomes higher if step sizes are closer to the resonance frequency of the system, with $\mathrm{f_{res}} = \frac{1}{\mathrm{T_{res}}}$. The step size is limited by the Shannon sampling theorem.

For the purpose of estimating analytically the absolute error for the autonomous system, the transition matrix $\boldsymbol{\Phi}(\mathrm{T_d})$ of the continuous time system $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}$, and the step size $\mathrm{T_d}$ can be computed by means of a power series of the exponential function first, as seen in [17] and equation (147):

$$\boldsymbol{\Phi}(\mathrm{T_d}) = \sum_{\mathrm{v}=0}^{\infty} \frac{\mathbf{A}^\mathrm{v}\mathrm{T_d^v}}{\mathrm{v}!}. \tag{147}$$

Furthermore, equations (142) and (143) can be written as

$$\mathbf{x}_{\mathrm{k}+\eta} = \mathbf{x}_\mathrm{k} + \big((\boldsymbol{\Phi}(\eta\mathrm{T_d}) - \mathbf{I})\mathbf{x}_\mathrm{k}\big) = \mathbf{x}_\mathrm{k} + \left(\sum_{\mathrm{v}=0}^{\infty} \frac{\mathbf{A}^\mathrm{v}(\eta\mathrm{T_d})^\mathrm{v}}{\mathrm{v}!} - \mathbf{I}\right)\mathbf{x}_\mathrm{k} \tag{148}$$

$$\mathbf{x}'_{\mathrm{k}+\eta} = \mathbf{x}_\mathrm{k} + \eta\big((\boldsymbol{\Phi}(\mathrm{T_d}) - \mathbf{I})\mathbf{x}_\mathrm{k}\big) = \mathbf{x}_\mathrm{k} + \eta\left(\sum_{\mathrm{v}=0}^{\infty} \frac{\mathbf{A}^\mathrm{v}\mathrm{T_d^v}}{\mathrm{v}!} - \mathbf{I}\right)\mathbf{x}_\mathrm{k}. \tag{149}$$

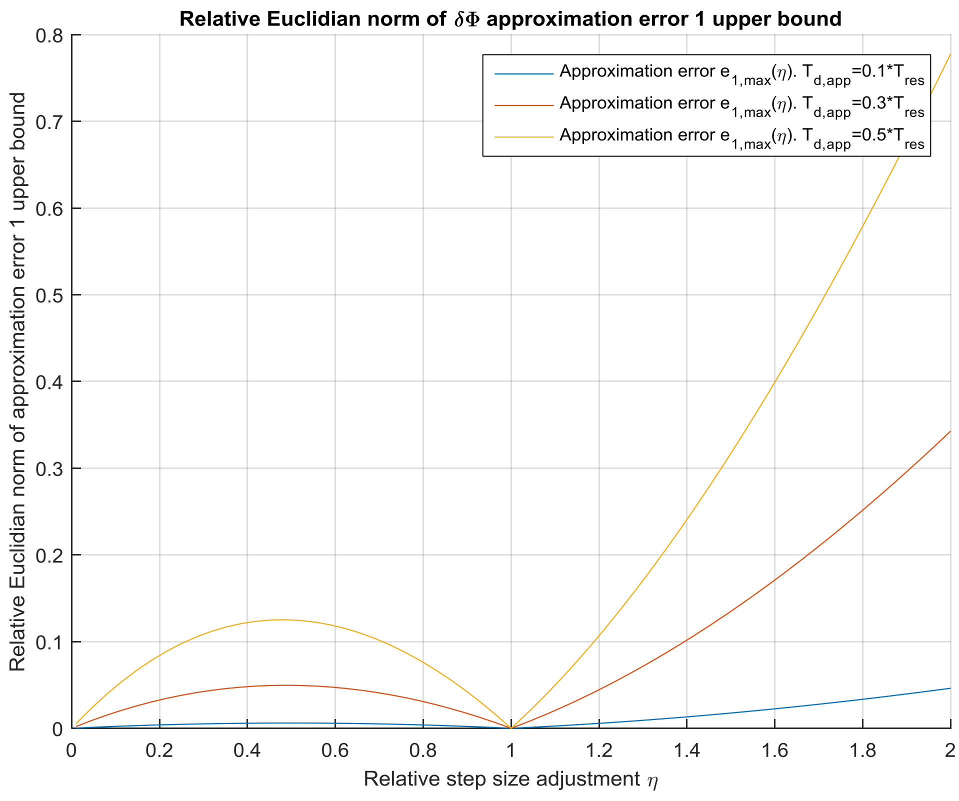

Figure 94: Norm of the upper bound error by using the linear delta term approximation

For further calculations (148) can be written as

$$\begin{aligned} \mathbf{x}_{k+\eta} &= \mathbf{x}_k + \left(\mathbf{I} + \mathbf{A}\eta T_d + \sum_{v=2}^{\infty} \frac{\mathbf{A}^v (\eta T_d)^v}{v!} - \mathbf{I}\right)\mathbf{x}_k \\ &= \mathbf{x}_k + \left(\mathbf{A}\eta T_d + \sum_{v=2}^{\infty} \frac{\mathbf{A}^v (\eta T_d)^v}{v!}\right)\mathbf{x}_k. \end{aligned} \tag{150}$$

Equation (149) can be written similarly as

$$\begin{aligned} \mathbf{x}'_{k+\eta} &= \mathbf{x}_k + \eta\left(\mathbf{I} + \mathbf{A}T_d + \sum_{v=2}^{\infty} \frac{\mathbf{A}^v T_d^{v}}{v!} - \mathbf{I}\right)\mathbf{x}_k \\ &= \mathbf{x}_k + \left(\mathbf{A}\eta T_d + \eta\sum_{v=2}^{\infty} \frac{\mathbf{A}^v T_d^{v}}{v!}\right)\mathbf{x}_k. \end{aligned} \tag{151}$$

The absolute error $\Delta\mathbf{x}_{k+\eta}$ can now be written as

$$\Delta\mathbf{x}_{k+\eta} = \mathbf{x}'_{k+\eta} - \mathbf{x}_{k+\eta} = \eta\sum_{v=2}^{\infty}\frac{\mathbf{A}^{v}T_{d}^{v}}{v!} - \sum_{v=2}^{\infty}\frac{\mathbf{A}^{v}(\eta T_{d})^{v}}{v!} \tag{152}$$

As a result, the term $\mathbf{A}\eta T_d$ is canceled, which leads to smaller error order in the end.

Another error is interesting in terms of the used step size adjustment. Generally an approximation is calculated for the step k with $0 < \eta_k < 1$ and $\eta_{k+1} = 2 - \eta_k$. That means the approximation is used twice, where it shortens the step size in the first place and then extends it. Here an upper bound error $e_{2,max}(\eta)$ can be shown for the autonomous system, that shows the Euclidian norm of the total error made by those two steps.

$$\begin{aligned}\mathbf{x}_{k+2} &= \mathbf{x}_{k+1} + (\mathbf{\Phi}(T_d) - \mathbf{I})\mathbf{x}_{k+1}\\ &= \mathbf{x}_k + (\mathbf{\Phi}(T_d) - \mathbf{I})\mathbf{x}_k + (\mathbf{\Phi}(T_d) - \mathbf{I})(\mathbf{x}_k + (\mathbf{\Phi}(T_d) - \mathbf{I})\mathbf{x}_k)\\ &= \mathbf{x}_k + 2(\mathbf{\Phi}(T_d) - \mathbf{I})\mathbf{x}_k + (\mathbf{\Phi}(T_d) - \mathbf{I})^2\mathbf{x}_k\end{aligned} \tag{153}$$

$$\begin{aligned}\mathbf{x}_{k+2} \approx \mathbf{x}'_{k+2} &= \mathbf{x}'_{k+\eta} + (2-\eta)(\mathbf{\Phi}(T_d) - \mathbf{I})\mathbf{x}'_{k+\eta}\\ &= \mathbf{x}_k + \eta(\mathbf{\Phi}(T_d) - \mathbf{I})\mathbf{x}_k\\ &\quad + (2-\eta)(\mathbf{\Phi}(T_d) - \mathbf{I})\,(\mathbf{x}_k + \eta(\mathbf{\Phi}(T_d) - \mathbf{I})\mathbf{x}_k)\\ &= \mathbf{x}_k + 2(\mathbf{\Phi}(T_d) - \mathbf{I})\mathbf{x}_k + 2\eta(\mathbf{\Phi}(T_d) - \mathbf{I})^2\mathbf{x}_k\\ &\quad - \eta^2(\mathbf{\Phi}(T_d) - \mathbf{I})^2\mathbf{x}_k\end{aligned} \tag{154}$$

$$\begin{aligned}e_2(\eta) &= \frac{\|\mathbf{x}'_{k+2} - \mathbf{x}_{k+2}\|_2}{\|\mathbf{x}_k\|_2} \le\\ e_{2,max}(\eta) &= |-1 + 2\eta - \eta^2|\,\|(\mathbf{\Phi}(T_d) - \mathbf{I})^2\|_2\end{aligned} \tag{155}$$

From equation (155) the dependency on η can be seen in Figure 95, if a constant step size T_d is chosen. It is remarkable that the matrix $\delta\mathbf{\Phi}(T_d)$ leads to a significant higher error and goes in with the power of two.

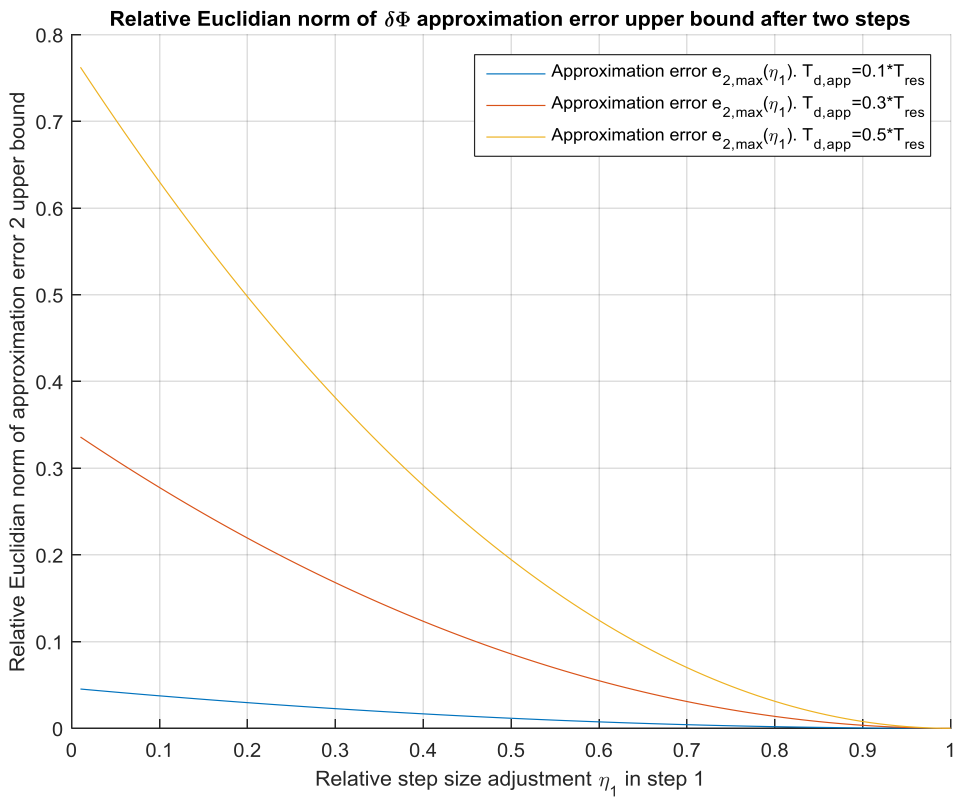

Figure 95: Error norm with using two consecutive step size adjustments

8 LIST OF FIGURES

9 LIST OF LITERATURE

[1] U. Kiencke and L. Nielsen, Automotive Control Systems: For Engine, Driveline, and Vehicle, 2nd ed., Berlin Heidelberg: Springer, 2005.

[2] K. Janschek, Systementwurf mechatronischer Systeme, Heidelberg: Springer, 2010.

[3] M. Bachinger, M. Stolz and M. Horn, "A Novel Drivetrain Modelling Approach for Real-Time-Simulation," *doi: 10.1016/j.mechatronics.2015.10.006,* 18 11 2015.

[4] R. Isermann, Mechatronische Systeme, Berlin Heidelberg: Springer-Verlag, 2008.

[5] M. Horn and N. Dourdoumas, Regelungstechnik, Graz: Pearson Verlag, 2003.

[6] H. Waller and R. Schmidt, Schwingungslehre für Ingenieure, Mannheim: BI-Wiss.-Verlag, 1989.

[7] C. Beerens, *Zur Modellierung nichtlinearer Dämpfungsphänomene in der Strukturmechanik*, Bochum: Ruhr-Universität Bochum, 1994.

[8] D. Mende and G. Simon, Physik: Gleichungen und Tabellen, VEB Fachbuchverlag Leibzig, 1981.

[9] G. F. Franklin, M. L. Workman and D. Powell, Digital Control of Dynamic Systems, 3rd ed., Addison Wesley Longman, Inc., 1997.

[10] Z. Zhang, D. Un An, K. Hyongsuk and C. T. Kil, *Comparative Study of Matrix exponential and Taylor series discretization methods for nonlinear ODEs,* China, South Korea: Elsevier B.V., 2008.

[11] E. Stiefel, Einführung in die numerische Mathematik, Stuttgart: B.G. Teubner, 1976.

[12] O. Föllinger, Laplace-, Fourier- und z-Transformation, Vde-Verlag, 2011.

[13] L. Guzzella and A. Sciarretta, Vehicle Propulsion Systems, Zurich: Springer Berlin Heidelberg, 2005.

[14] H. Pacejka, Tire and Vehicle Dynamics, Oxford, UK: Butterworth-Heinemann, 2012.

[15] D. Adamski, Simulation in der Fahrwerktechnik, Wiesbaden: Springer, 2014.

[16] "Matlab Documentation," The MathWorks, Inc., 2015. [Online]. Available: http://de.mathworks.com/help/physmod/sdl/ref/tireroadinteractionmagicformula.html. [Accessed 10 11 2015].

[17] O. Föllinger, Regelungstechnik, Heidelberg: Dr. Alfred Hüthig Verlag, 1985.

[18] H. Naunheimer, B. Bertsche and G. Lechner, *Fahrzeuggetriebe,* Heidelberg: Springer-Verlag Berlin Heidelberg, 2007.

[19] A. Ompusunggu, P. Sas and H. Van Brussel, "Modeling and simulation of the engagement dynamics of a wet friction clutch system subjected degradation: An application to condition monitoring and prognostics.," doi:10.1016/j.mechatronics.2013.07.007, 2013.

[20] G. Schmidt, Simulationstechnik, München: R. Oldenbourg Verlag München Wien, 1980.